

Methods in
Molecular Biology 1260

Springer Protocols



Hugh Cartwright *Editor*

Artificial Neural Networks

Second Edition



 Humana Press

METHODS IN MOLECULAR BIOLOGY

Series Editor
John M. Walker
School of Life Sciences
University of Hertfordshire
Hatfield, Hertfordshire, AL10 9AB, UK

For further volumes:
<http://www.springer.com/series/7651>

Artificial Neural Networks

Second Edition

Edited by

Hugh Cartwright

*Chemistry Department, Oxford University, Oxford, UK; Chemistry Department,
University of Victoria, BC, Canada*

 **Humana Press**

Editor

Hugh Cartwright
Chemistry Department
Oxford University
Oxford, UK

Chemistry Department
University of Victoria
BC, Canada

Additional material to this book can be downloaded from <http://extras.springer.com>

ISSN 1064-3745

ISSN 1940-6029 (electronic)

ISBN 978-1-4939-2238-3

ISBN 978-1-4939-2239-0 (eBook)

DOI 10.1007/978-1-4939-2239-0

Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2014956521

© Springer Science+Business Media New York 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Humana Press is a brand of Springer

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Artificial Neural Networks (ANNs) are among the most fundamental techniques within the field of Artificial Intelligence. Their operation loosely emulates the functioning of the human brain, but the value of an ANN extends well beyond its role as a biological model. An ANN can both memorize and reason: it provides a way in which a computer can learn from scratch about a previously unseen problem. Remarkably, the exact form of the problem is rarely critical; it might be financial (e.g., can we predict the direction of the stock market in the next few months?); it might be sociological (what factors make a face attractive?); it could be medical (can we tell from an X-ray whether a bone is broken?); or, as in this volume, the problem might be purely scientific.

This text brings together some productive and fascinating examples of how ANNs are applied in the biological sciences and related areas: from the analysis of intracellular sorting information to the prediction of the behavior of bacterial communities; from biometric authentication to studies of tuberculosis; from studies of gene signatures in breast cancer classification to the use of mass spectrometry in metabolite identification; from visual navigation to computer diagnosis of possible lesions; and more. The authors describe not only *what* they have done with ANNs but also *how* they have done it. Readers intrigued by the work described in this book will find numerous practical details, which should encourage further use of these rapidly developing tools.

Oxford, UK

Hugh Cartwright

Contents

<i>Preface</i>	<i>v</i>
<i>Contributors</i>	<i>ix</i>
1 Introduction to the Analysis of the Intracellular Sorting Information in Protein Sequences: From Molecular Biology to Artificial Neural Networks <i>R. Claudio Aguilar</i>	1
2 Protein Structural Information Derived from NMR Chemical Shift with the Neural Network Program <i>TALOS-N</i> <i>Yang Shen and Ad Bax</i>	17
3 Predicting Bacterial Community Assemblages Using an Artificial Neural Network Approach <i>Peter Larsen, Yang Dai, and Frank R. Collart</i>	33
4 A General ANN-Based Multitasking Model for the Discovery of Potent and Safer Antibacterial Agents <i>A. Speck-Planche and M.N.D.S. Cordeiro</i>	45
5 Use of Artificial Neural Networks in the QSAR Prediction of Physicochemical Properties and Toxicities for REACH Legislation <i>John C. Dearden and Philip H. Rowe</i>	65
6 Artificial Neural Network for Charge Prediction in Metabolite Identification by Mass Spectrometry <i>J.H. Miller, B.T. Schrom, and L.J. Kangas</i>	89
7 Prediction of Bioactive Peptides Using Artificial Neural Networks <i>David Andreu and Marc Torrent</i>	101
8 AutoWeka: Toward an Automated Data Mining Software for QSAR and QSPR Studies. <i>Chanin Nantasenamat, Apilak Worachartcheewan, Saksiri Jamsak, Likit Preeyanon, Watshara Shoombuatong, Saw Simeon, Prasit Mandi, Chartchalerm Isarankura-Na-Ayudhya, and Virapong Prachayasittikul</i>	119
9 Ligand Biological Activity Predictions Using Fingerprint-Based Artificial Neural Networks (FANN-QSAR) <i>Kyaw Z. Myint and Xiang-Qun Xie</i>	149
10 GENN: A GEneral Neural Network for Learning Tabulated Data with Examples from Protein Structure Prediction <i>Eshel Faraggi and Andrzej Kloczkowski</i>	165
11 Modulation of Grasping Force in Prosthetic Hands Using Neural Network-Based Predictive Control <i>Cristian F. Pasluosta and Alan W.L. Chiu</i>	179
12 Application of Artificial Neural Networks in Computer-Aided Diagnosis <i>Bei Liu</i>	195

13	Developing a Multimodal Biometric Authentication System Using Soft Computing Methods	205
	<i>Mario Malcangi</i>	
14	Using Neural Networks to Understand the Information That Guides Behavior: A Case Study in Visual Navigation	227
	<i>Andrew Philippides, Paul Graham, Bart Baddeley, and Philip Husbands</i>	
15	Jump Neural Network for Real-Time Prediction of Glucose Concentration	245
	<i>Chiara Zecchin, Andrea Facchinetti, Giovanni Sparacino, and Claudio Cobelli</i>	
16	Preparation of Ta-O-Based Tunnel Junctions to Obtain Artificial Synapses Based on Memristive Switching.	261
	<i>Stefan Niehörster and Andy Thomas</i>	
17	Architecture and Biological Applications of Artificial Neural Networks: A Tuberculosis Perspective	269
	<i>Jerry A. Darsey, William O. Griffin, Sravanthi Joginipelli, and Venkata Kiran Melapu</i>	
18	Neural Networks and Fuzzy Clustering Methods for Assessing the Efficacy of Microarray Based Intrinsic Gene Signatures in Breast Cancer Classification and the Character and Relations of Identified Subtypes	285
	<i>Sandhya Samarasinghe and Amphun Chaiboonchoe</i>	
19	QSAR/QSPR as an Application of Artificial Neural Networks	319
	<i>Narelle Montañez-Godínez, Aracely C. Martínez-Olguín, Omar Deeb, Ramón Garduño-Juárez, and Guillermo Ramírez-Galicia</i>	
	<i>Index</i>	335

Contributors

- R. CLAUDIO AGUILAR • *Department of Biological Sciences, Purdue University, West Lafayette, IN, USA*
- DAVID ANDREU • *Department of Experimental and Health Sciences, Universitat Pompeu Fabra, Barcelona, Spain*
- BART BADDELEY • *Centre for Computational Neuroscience and Robotics, University of Sussex, Brighton, UK*
- AD BAX • *Laboratory of Chemical Physics, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD, USA*
- AMPHUN CHAIBOONCHOE • *Integrated Systems Modelling Group, Lincoln University, Christchurch, New Zealand; Division of Science and Math, and Center for Genomics and Systems Biology, New York University Abu Dhabi, Abu Dhabi, UAE*
- ALAN W.L. CHIU • *Biomedical Engineering, Rose-Hulman Institute of Technology, Terre Haute, IN, USA*
- CLAUDIO COBELLI • *Department of Information Engineering, University of Padova, Padova, Italy*
- FRANK R. COLLART • *Biosciences Division, Argonne National Laboratory, Lemont, IL, USA*
- M.N.D.S. CORDEIRO • *Department of Chemistry and Biochemistry, Faculty of Sciences, University of Porto, Porto, Portugal*
- YANG DAI • *Department of Bioengineering, University of Illinois at Chicago, Chicago, IL, USA*
- JERRY A. DARSEY • *Department of Chemistry, University of Arkansas at Little Rock, Little Rock, AR, USA*
- JOHN C. DEARDEN • *School of Pharmacy & Biomolecular Sciences, Liverpool John Moores University, Liverpool, UK*
- OMAR DEEB • *Faculty of Pharmacy, Al-Quds University, Jerusalem, Palestine*
- ANDREA FACCHINETTI • *Department of Information Engineering, University of Padova, Padova, Italy*
- ESHEL FARAGGI • *Department of Biochemistry and Molecular Biology, Indiana University School of Medicine, Indianapolis, IN, USA; Battelle Center for Mathematical Medicine, Nationwide Children's Hospital, Columbus, OH, USA; Physics Division, Research and Information Systems, Carmel, IN, USA*
- RAMÓN GARDUÑO-JUÁREZ • *Instituto de Ciencias Fisicas, Universidad Nacional Autonoma de Mexico, Cuernavaca, Mexico*
- PAUL GRAHAM • *Centre for Computational Neuroscience and Robotics, University of Sussex, Brighton, UK*
- WILLIAM O. GRIFFIN • *Department of Applied Science, University of Arkansas at Little Rock, Little Rock, AR, USA*
- PHILIP HUSBANDS • *Centre for Computational Neuroscience and Robotics, University of Sussex, Brighton, UK*
- CHARTCHALERM ISARANKURA-NA-AYUDHYA • *Department of Clinical Microbiology and Applied Technology, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*

- SAKSIRI JAMSAK • *Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- SRAVANTHI JOGINPELLI • *Department of Bioinformatics, University of Arkansas at Little Rock, Little Rock, AR, USA*
- L.J. KANGAS • *Washington State University Tri-Cities, Richland, WA, USA*
- ANDRZEJ KLOCZKOWSKI • *Battelle Center for Mathematical Medicine, Nationwide Children's Hospital, Columbus, OH, USA; Department of Pediatrics, The Ohio State University, Columbus, OH, USA*
- PETER LARSEN • *Biosciences Division, Argonne National Laboratory, Lemont, IL, USA; Department of Bioengineering, University of Illinois at Chicago, Chicago, IL, USA*
- BEI LIU • *Radiation Oncology, City of Hope Medical Center, Duarte, CA, USA*
- MARIO MALCANGI • *Department of Computer Science, Università degli Studi di Milano, Milan, Italy*
- PRASIT MANDI • *Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- ARACELY C. MARTÍNEZ-OLGUÍN • *División de Estudios de Posgrado, Universidad del Papaloapan, Oaxaca, Mexico*
- VENKATA KIRAN MELAPU • *Department of Bioinformatics, University of Arkansas at Little Rock, Little Rock, AR, USA*
- J.H. MILLER • *Washington State University Tri-Cities, Richland, WA, USA*
- NARELLE MONTAÑEZ-GODÍNEZ • *División de Estudios de Posgrado, Universidad del Papaloapan, Oaxaca, Mexico*
- KYAW Z. MYINT • *NIDA Center of Excellence for Computational Chemogenomics Drug Abuse Research, Computational Chemical Genomics Screening Center, Department of Pharmaceutical Sciences, School of Pharmacy, University of Pittsburgh, Pittsburgh, PA, USA*
- CHANIN NANTASENAMAT • *Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand; Department of Clinical Microbiology and Applied Technology, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- STEFAN NIEHÖRSTER • *Thin films and Physics of Nanostructures, Physics Department, Bielefeld University, Bielefeld, Germany*
- CRISTIAN F. PASLUOSTA • *Electronics Core - Medical Device Solutions, Lerner Research Institute, Cleveland Clinic, Cleveland, OH, USA*
- ANDREW PHILIPPIDES • *Centre for Computational Neuroscience and Robotics, University of Sussex, Brighton, UK*
- VIRAPONG PRACHAYASITTIKUL • *Department of Clinical Microbiology and Applied Technology, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- LIKIT PREEYANON • *Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- GUILLERMO RAMÍREZ-GALICIA • *División de Estudios de Posgrado, Universidad del Papaloapan, Oaxaca, Mexico*
- PHILIP H. ROWE • *School of Pharmacy & Biomolecular Sciences, Liverpool John Moores University, Liverpool, UK*
- SANDHYA SAMARASINGHE • *Integrated Systems Modelling Group, Lincoln University, Christchurch, New Zealand*
- B.T. SCHROM • *Washington State University Tri-Cities, Richland, WA, USA*

- YANG SHEN • *Laboratory of Chemical Physics, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD, USA*
- WATSHARA SHOOMBATONG • *Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- SAW SIMEON • *Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- GIOVANNI SPARACINO • *Department of Information Engineering, University of Padova, Padova, Italy*
- A. SPECK-PLANCHE • *Department of Chemistry and Biochemistry, Faculty of Sciences, University of Porto, Porto, Portugal*
- ANDY THOMAS • *Thin films and Physics of Nanostructures, Physics Department, Bielefeld University, Bielefeld, Germany; Institute of Physics, Johannes Gutenberg University Mainz, Mainz, Germany*
- MARC TORRENT • *Medical Research Council Laboratory of Molecular Biology, Cambridge, UK; Vall d'Hebron Institute of Research, Universitat Autònoma de Barcelona, Vall d'Hebron University Hospital, Barcelona, Spain*
- APILAK WORACHARTCHEEWAN • *Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand*
- XIANG-QUN XIE • *NIDA Center of Excellence for Computational Chemogenomics Drug Abuse Research, Computational Chemical Genomics Screening Center, Department of Pharmaceutical Sciences, School of Pharmacy, University of Pittsburgh, Pittsburgh, PA, USA*
- CHIARA ZECCHIN • *Department of Information Engineering, University of Padova, Padova, Italy*

Chapter 1

Introduction to the Analysis of the Intracellular Sorting Information in Protein Sequences: From Molecular Biology to Artificial Neural Networks

R. Claudio Aguilar

Abstract

A precise spatial-temporal organization of cell components is required for basic cellular activities such as proliferation and for complex multicellular processes such as embryo development. Particularly important is the maintenance and control of the cellular distribution of proteins, as these components fulfill crucial structural and catalytic functions.

Membrane protein localization within the cell is determined and maintained by intracellular elements known as *adaptors* that interpret sorting information encoded in the amino acid sequence of cargoes. Understanding the sorting sequence code of cargo proteins would have a profound impact on many areas of the life sciences. For example, it would shed light onto the molecular mechanisms of several genetic diseases and would eventually allow us to control the fate of proteins.

This chapter constitutes a primer on protein-sorting information analysis and localization/trafficking prediction. We provide the rationale for and a discussion of a simple basic protocol for protein sequence dissection looking for sorting signals, from simple sequence inspection techniques to more sophisticated artificial neural networks analysis of sorting signal recognition data.

Key words Protein sorting, Sequence analysis, Artificial neural networks, Intracellular localization

1 Introduction

1.1 How Do the Emergent Properties of Cells Arise from a Mix of Proteins, Lipids, Sugars, and Nucleic Acids? (See Note 1)

This intriguing, complex question is at the core of modern biology, and it can be initially approached by simply stating that cells are not just a mix but an ordered array of their components. Cells require an internal spatial-temporal organization for the fulfillment of specific biochemical processes (e.g., by creating chemical potentials—Fig. 1a). Indeed, life can be conceived as resulting from the constant struggle to generate and maintain internal order against entropy trying to drive organisms to a lethal equilibrium.

This dynamic, yet highly ordered, steady state is particularly complex in eukaryotic cells (e.g., human cells) as they add a physical dimension to their spatial-temporal organization in the form of

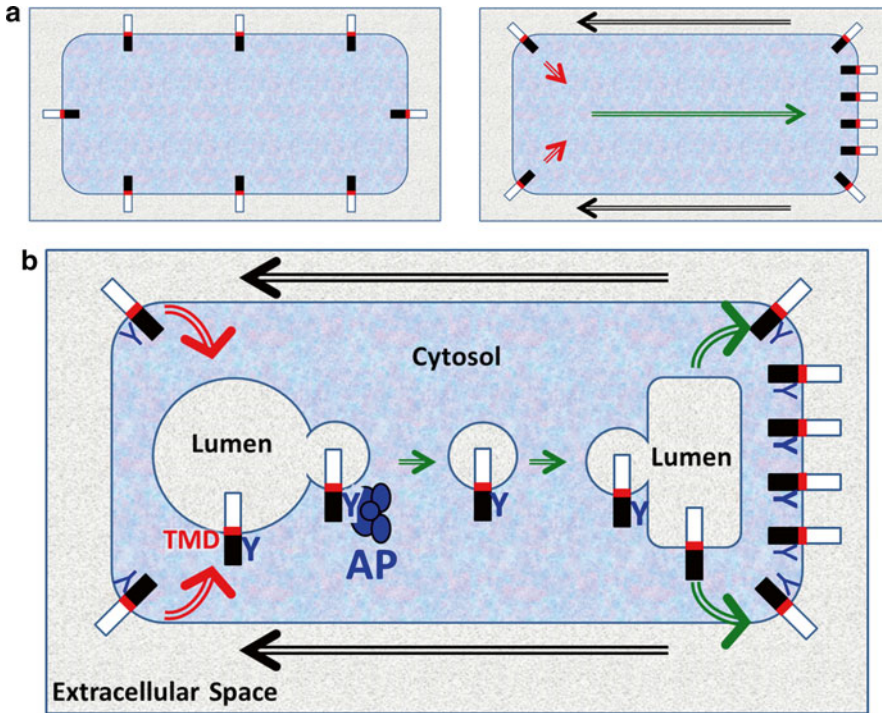


Fig. 1 Establishment and maintenance of cell organization. Cells are represented as compartments separating cytosol (*bluish*) from the extracellular space (*greyish*). A single membrane-spanning protein on the cell plasma membrane is represented as a rectangle with cytosolic and luminal regions (*black* and *white* sectors, respectively). A *red* segment within the protein represents the transmembrane domain (TMD). **(a) Left panel:** Isotropic (homogenous) distribution of the protein at equilibrium. This is a stage of low energy (no chemical potential) and without spatial differential properties. **Right panel:** The organized (polarized) distribution of key functional components (e.g., proteins) allows cell function. For example, asymmetrical distribution of ion membrane transporters can generate electrochemical potentials required for synaptic transmission. Diffusion forces (*black arrows*) oppose intracellular organization by promoting the dispersion of membrane proteins. Active (energy-consuming) transport (*red/green arrows*) of protein is constantly required to counteract diffusion in order to establish and maintain cell organization. **(b)** Transmembrane proteins display a tyrosine-based sorting signal (“Y”) recognized by adaptors (*blue* complex labeled “AP”). This cargo is actively removed from the plasma membrane (internalization, *red arrows*) and recycled back (*green arrows*) to polarized areas. Transport occurs through vesicle trafficking, i.e., vesicles loaded with cargo by the adaptor bud off from a donor compartment and fuse with a target compartment. Note that the protein topology is conserved: cytosolic regions always face the cytosol while luminal regions always face the lumen or the extracellular space

membrane-bound compartments or organelles (e.g., the nucleus). These physically separated entities constitute highly efficient functional stations that host specific biochemical processes by virtue of their chemical composition.

The evolutionary advantage that eukaryotic cells obtained from acquiring these specialized organelles was intimately linked to the development of a cargo transport/communication system among these processing stations. A highly efficient organelle is useless if it

cannot receive substrates or deliver products. In addition, these membrane-bound compartments cannot productively contribute to the whole if there is no flow of information or feedback control of their activity.

Given the membranous nature of the organelles, the trafficking system that links them must be based on exchange of membrane-bound carriers (or vesicles) that can pinch off from a donor compartment and be competent to fuse with and deliver their cargo to the membrane or lumen of a target compartment (Fig. 1b). This vesicle-trafficking system not only contributes to maintain the characteristic cellular steady state away from thermodynamic equilibrium, but it is also crucial for organelle inheritance and cell division among other essential cellular processes. For detailed descriptions and discussions on vesicle-trafficking mechanisms in health and disease, the reader is referred to excellent cell biology textbooks and reviews available in the literature (*see Note 2* and refs. 1, 2).

1.2 How Are Cargoes Selectively Loaded into the Right Vesicles and Delivered to the Right Compartments?

The answer to this question is (deceivably) simple, just like the mail. The “mailed” cargo displays a destination “address” (or “addresses”) and it is delivered via the interplay of cellular “postal workers” (that load specific mail items into the proper “bag”) with the “addressee.”

If we focus on proteins (i.e., protein sorting), and specifically on integral membrane proteins (*see Note 3*), their destination addresses and/or delivery routes are encoded as consensus amino acid sequences that may or may not be activated/deactivated by posttranslational modifications. Most of these sorting signals are located in the cytoplasmic domain of the protein cargo, readily available for cellular “postal workers” to interpret and bag the cargo in a vesicle (Fig. 1b). The loaded vesicles are connected to motors and placed on cytoskeleton tracks to be mobilized to their destinations (*see Note 2*).

Many different cellular “postal workers,” or *adaptors*, have been identified and characterized. These adaptors are also proteins, or protein complexes, that associate with the cytoplasmic side of membranes and vary in both location and type of sorting signals they interpret. Table 1 shows a list of some of the most important protein-sorting signals and their corresponding proposed adaptors.

Within this signal recognition machinery, the clathrin-associated adaptor proteins (APs) emerge as major players in the protein-trafficking system of higher eukaryotes [3, 4] (Fig. 1b). Five tetrameric AP complexes (AP1 through AP5) with different intracellular localizations have been identified and showed to mediate different protein-sorting events connecting several compartments [5, 6]. Whereas other AP subunits are engaged in interactions with various molecules, the μ (medium) subunit is in charge of recognizing

Table 1
Signal-adaptor specificity

Sorting signal consensus ^a	Adaptor(s) ^b	Proposed role in TMP vesicle traffic
YXXØ	APs	General vesicle traffic
NPXY	Dab2	Removal from PM
[D/E]XXLØ	GGAs	Endosome-Golgi complex transport
[D/E]XXXLØ	APs	General vesicle traffic
NPFXD	Sla1 ^c	Removal from PM
Acidic Cluster	PACS-1	Endosome-Golgi complex transport

^aAmino acids are indicated using the 1-letter code (e.g., Y=Tyr=Tyrosine; N=Asn=Asparagine—*see Note 2*). “X” represents a position occupied by any amino acid. Ø=amino acid with a bulky hydrophobic side chain (L, I, M, V, F). Amino acids within brackets indicate that one or the other can be found in that position within the consensus

^bAdaptor abbreviation: *AP* clathrin-associated adaptor protein; *Dab2* disabled 2; *GGA* Golgi-localized, gamma-ear-containing, ARF-binding protein; *Sla1* synthetic lethal with ABP1 protein 1; *PACS-1* phosphofurin acidic cluster sorting protein 1. Other abbreviations: *TMP* transmembrane protein; *PM* plasma membrane

^cAdaptor present in *Saccharomyces cerevisiae*

tyrosine-based sorting signals fitting a YXXØ consensus (where X=any amino acid, Y=tyrosine, and Ø=amino acids with a bulky hydrophobic side chain such as phenylalanine, leucine, isoleucine, methionine, and valine).

1.3 Can We Predict the Intracellular Destination of Transmembrane Proteins by Reading Their Amino Acid Sequences?

If we were able to *identify* and *interpret* the whole repertoire of sorting signals, we would be capable of *predicting* the fate of a protein cargo solely based on its amino acid sequence. This achievement would have a high impact on many different areas of the life sciences; for example, it would shed light on the nature of several hereditary diseases and would enhance our ability to control the intracellular fate of proteins within cells. However, we still do not fully understand how the sorting information is coded in protein sequences or the precise role in protein trafficking of all the different adaptors. In addition, multiple factors including topology/accessibility and context within the protein affect the recognition of sorting signals by adaptors (*see below*), adding extra layers of complexity to the process. It should also be noted that cargoes often display multiple sorting signals from one or more types and that the protein final destination results from the interplay of this composite of motifs.

The purpose of this chapter is to introduce the foundations of protein-sorting information analysis. Specifically, we will discuss how to identify candidate sorting signals within protein sequences and analyze their role in protein trafficking. We will also describe simple techniques to experimentally assess the relevance of putative signals for protein sorting in the context of model and native cargo.

2 Method Foundations

This simple procedure for protein sequence analysis is aimed to allow the investigator to identify (Subheading 2.1) and interpret (Subheading 2.2) signals, and it is ultimately intended to predict and/or test the role of sorting determinants on the traffic and localization (Subheading 2.3) of a protein of interest. While the foundations are provided here, a protocol for the implementation of the method can be found in Subheading 3.

2.1 Looking for Signals

In order to read sorting signals, first we need to find them. Finding sequences within cargoes that would fit into a given sorting consensus is not hard. Indeed, some of these motifs (e.g., YXXØ) are very commonly found within protein sequences (*see* Fig. 2), but not all of them are functional for cargo sorting; actually, only a few are. Therefore, the real task is to identify viable sorting signal candidates among other sequences that, even when fitting into the consensus, are likely to fulfill different functions within the protein. Although we only partially understand the information-coding system, it has been established that sequence motifs need to meet a few conditions related to its nature, topology/accessibility, and environment to be active in protein sorting. These requirements can be exploited to our advantage to highlight sequences with higher probability of being functional sorting signals:

- (a) Fit into a signal consensus: Although in some cases certain flexibility may exist, the better a sequence fits into a given consensus, the higher the probability that it works as a real sorting signal. It should also be kept in mind that posttranslational

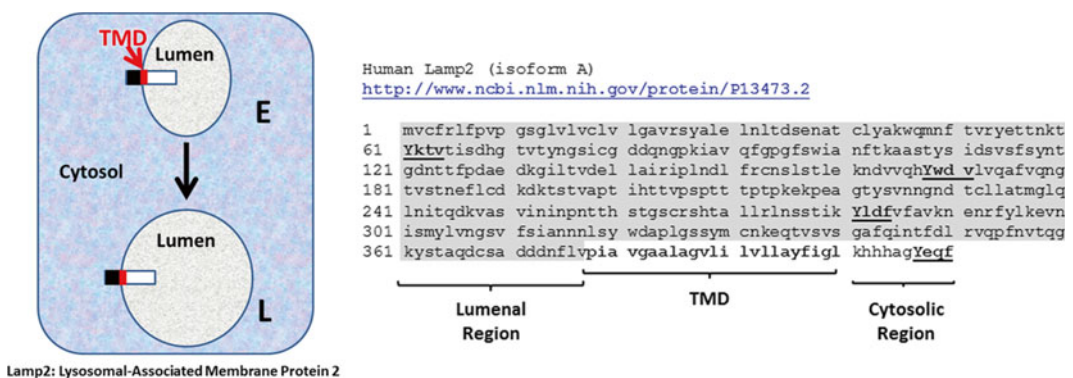


Fig. 2 Sequence analysis of the lysosomal protein Lamp2. *Left panel:* The lysosomal-associated membrane protein 2 (Lamp2) is transported from endosomes (E) to the lysosome (L). *Right panel:* The amino acid sequence of the isoform A of human Lamp2 was obtained from the NCBI protein database (URL provided). Although four YXXØ motifs (underlined) were identified in Lamp2, only one (Yeqf) was located in the cytosolic region of the protein. Further, this motif is located within the distance range (6–10 amino acids from the TMD) for AP recognition. *See text for details*

modifications can activate or deactivate signals. For example, tyrosine phosphorylation of an YXXØ motif will deactivate the Y-signal, while phosphorylation of a serine/threonine amino acid located at position -4 replacing D/E in a di-leucine motif (*see* Table 1) is likely to be required for function (*see* Note 4). Therefore, the possibility that the putative sorting signal overlaps with a known phosphorylation consensus and the effect of such a modification should also be carefully considered.

- (b) Topology and accessibility: Functional sorting signals need to be positioned within cargo proteins so that adaptors can find, recognize, and bind them. Since adaptors associate with the cytosolic side of the compartment membrane (Fig. 1b), even when perfectly fitting into a certain consensus (Table 1), a motif facing the luminal space (inside of a vesicle/organelle or outside of the cell—*see* Figs. 1b and 2) will *NOT* work as a classical sorting determinant. Therefore, information about the topology of a transmembrane protein is critical. This experimentally determined or predicted information can sometimes be found within the notes to the entry for the cargo under consideration in the National Center for Biotechnology Information (NCBI) databases (*see* Subheading 3.1). If not available in databases or the literature, it is up to the investigator to produce such information either experimentally or by prediction. Experimental approaches such as controlled digestion of extracellular regions followed by biochemical identification of fragments (e.g., by immunoblotting) can be performed [7]. Alternatively, transmembrane region/topology prediction algorithms can be used. Particularly useful are online resources based on the cargo sequence that can not only predict the position of transmembrane domains (TM, *see* Figs. 1 and 2, Note 3) but also estimate the probability that different protein segments, not embedded into the membrane, would be facing either the cytosol or the luminal space (*see* Subheading 3.2).

In addition to having the proper topology, most sorting signals need to be positioned within a certain range/distance from the compartment membrane to be accessible to adaptors and, therefore, to be functionally active. A signal too close to the membrane might be sterically hindered for adaptor recognition, while others might be too far for the adaptor's spatial range. For example, the critical tyrosine amino acid within YXXØ signals usually needs to be located at 6–10 amino acids from the membrane (TM boundary). This spacing yields the Y-signal within reach of the μ subunit from membrane-bound APs. It should also be kept in mind that posttranslational modifications, such as palmitoylation, can alter distance to the membrane by re-anchoring the cytoplasmic region of the protein

to the inner leaflet and, for example, bringing a distant signal back to adaptor reach [8].

- (c) Structural environment: Although not impossible, it is unlikely that signals listed in Table 1 would be located within functional domains (e.g., kinase domains—*see* **Note 5**). In particular, if the candidate motif is buried in the domain's hydrophobic core, it will be inaccessible to adaptors, and therefore, its potential role as sorting signal should be ruled out. Although it has been consistently indicated that TMs can participate in the sorting of proteins [9, 10], due to reasons of accessibility to classical adaptors, motifs fitting the consensus considered in this chapter (Table 1) cannot be located in the TM region of cargoes.

It is also known that some sorting signals need to be displayed on specific protein secondary structures; for example, dileucine signals (Table 1) are usually found embedded in α -helical structures and YXX \emptyset signals are normally found in a β -sheet conformation. Online resources can also be very useful for predicting secondary structures (*see* Subheading 3.4, **step 3**). Nevertheless, it is also possible that, while free, some signals are mostly unstructured, but they would acquire proper structure upon adaptor binding (i.e., by an “induced fit” mechanism).

Nevertheless, fulfillment of these three criteria might not be sufficient to guarantee that the candidate motif has a role as a sorting signal. Furthermore, as indicated above, some rule flexibility has been observed. Therefore, satisfaction of some criteria (i.e., (a) and (c)) should be considered as contributing to the probability that a sequence motif may act as a sorting signal. In fact, the role of the putative signal in the localization of the native or a model cargo must be experimentally demonstrated (*see* below).

2.2 Interpreting the Message

Following the identification of sorting motifs in cargoes of interest, most likely the investigator will aim to predict/determine (a) signal-adaptor specificity and (b) signal relevance to cargo trafficking/localization. Although this might be the ultimate goal of the analysis, it could also be the most challenging part of the process. This is mostly due to the previously alluded limited understanding of the sequence information code and of the cellular role of different adaptors. Furthermore, the inherent complexity of protein cargo sequence and structure (e.g., multiplicity of signals for sorting and posttranslational modifications—*see* Subheading 2.3) makes this task even harder. Therefore, in order to assess the specificity and relevance of a given sorting motif while minimizing confounding effects of other signals or modifiers, a reductionist approach is recommended; i.e., the signal under consideration should be isolated for analysis.

2.2.1 To Predict/
Determine Signal-Adaptor
Specificity: What Adaptor
will Recognize a Given
Signal?

Although not comprehensive (and many exceptions to these rules may exist), Table 1 indicates what sorting signal is recognized by each adaptor or family of adaptors. The signals identified in the cargo of interest should be contrasted against Table 1 or similar. In addition, when signals are recognized by families of adaptors (e.g., APs), determining the fine specificity for family members could be critical to predict cargo intracellular trafficking and distribution. For example, different AP adaptors localize to different compartments and are believed to mediate different protein-sorting events.

The fine YXXØ specificity of members of the AP family has been explored in detail [11, 12]. These studies indicate that, in addition to Y and Ø, the X-positions (including amino acids upstream of the crucial Y) also influence the process of signal recognition [11, 12]. Based on these findings, a table summarizing the amino acid preference of each AP complex at the X/Ø-positions of a XXXYXXØ generic signal was constructed (Table 2).

The Y-signal amino acid preference of the different APs documented in Table 2 points to several important trends of AP consensus recognition and their impact on cargo trafficking. For example, enrichment of acidic amino acids (D/E) preceding the Ø position and presence of the amino acid glycine (G) just before the critical Tyr would make a protein cargo a prime subject for AP3 recognition (Table 2). Interestingly, AP3 has been implicated in protein sorting to the lysosome and many lysosomal proteins display acidic sequences that fit into the GYXXØ signal consensus subset.

However, Table 2 does not predict the experimentally observed synergic and inhibitory effects between amino acids on

Table 2
Y-signal specificity of the mammalian AP family^a

Adaptor complex ^b	XXXYXXØ motif								Proposed role in TMP vesicle traffic
	-3	-2	-1	Y	+1	+2	Ø		
AP1	+	R	S	L	Y	R/Q	P	L	Golgi complex-endosome
	-	F/L	L	I			I	F/V	
AP2	+	G	F	P	Y	E/Q	P/R	L	Removal of TMP from PM
	-						F		
AP3	+	R	A	G/D/E	Y	E	P	I	Endosome to lysosome
	-	F/I				S	F/L/I		
AP4 ^c	+	C	Y	F	Y	D	P	F	Endosome to lysosome
	-	T	G	N/T				V	

^aTable condenses results published in ref. 11, 12

^bFor each adaptor amino acids, preferred (“+” row) and excluded (“-” row) at each position are indicated. Shaded cells indicate no significant preference or exclusion. Adaptor complex AP5 is excluded from this table as no analysis for signal preference is currently available

^cIn addition to YXXØ signals, AP4 has been shown to recognize YX[F/Y/L][F/L]E motifs

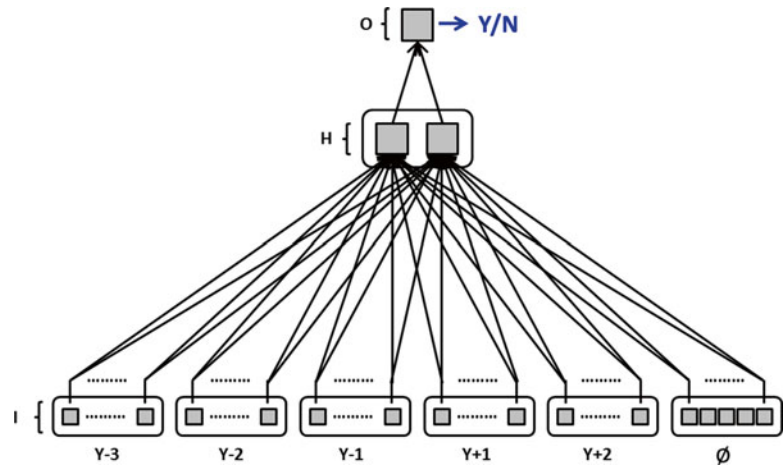


Fig. 3 YXXØ artificial neural network architecture. The neurons in the network are represented by squares and the connections between units by arrows. The input layer (I) is divided in 5 clusters (one for each X position within the XXXYXXØ motif) made up of 20 nodes each (representing the 20 possible amino acids—only 2 per cluster are shown); the Ø cluster contains only 5 nodes (for amino acids F, M, I, L and V), yielding 105 neurons in total. Hidden (H) and output (O) layers are shown. Final network output is denoted as binary Y or N value, denoting the presence or absence of interaction between the signal and adaptor under consideration, respectively (*see* ref. 13 for more details)

the recognition by APs. Indeed, although certain amino acids (occupying different positions within the motif) may independently favor signal recognition by an AP, when coexisting in the same Y-motif they lead to poor adaptor binding. Conversely, amino acids that independently did not contribute to enhance AP recognition, when together in the same signal, they may cooperate to yield substantial binding [11–13].

This high complexity of the signal recognition process makes it difficult to predict the adaptor specificity of a given Y-signal for the different APs. However, since an extensive collection of experimental examples of interactions between YXXØ motifs and APs were available [11, 12], we used an approach based on the artificial neural network (ANN) paradigm to assess the problem of Y-signal specificity [13].

This was possible thanks to the wealth of binding data obtained from a series of screens performed using the so-called “two-hybrid” technology [14] with a combinatorial library of XXXYXXØ motifs [11, 12]. Most of this data was used to train a few ANNs with the architecture depicted in Fig. 3 (training and validation of this type of ANNs can be found in ref. 13). The abovementioned ANNs can very efficiently predict yeast two-hybrid results but should be only considered as an approximation to what would occur in native context (e.g., in the cytoplasmic region of membrane-inserted proteins

within mammalian cells). Ideally, future investigators would be somehow able to perform screens analyzing the distribution of libraries of model cargoes displaying a combinatorial array of signals in target cells. These hypothetical results should be prime material to train future ANNs that would be able to accurately predict cargo localization.

Nevertheless, ANNs trained with two-hybrid data were capable of predicting recognition pattern and localization of certain Y-signal-containing proteins *in vivo* [13]. The effects of certain mutations affecting Y-signal specificity were also successfully predicted [13]. These encouraging results suggest that similar approaches could be used in the future to address fine recognition specificity of other signal consensus.

In addition, other factors should be kept in mind as they could also contribute to define realistic signal specificity. For example, lysosomal Y-signals, *i.e.*, candidates to interact with the AP3 complex, are located relatively near the compartment membrane (*e.g.*, ~6 amino acids from the TM). However, larger separation of the signal from the TM would not necessarily preclude its recognition by AP3.

It should also be considered whether the expression patterns of cargo and adaptor match: do the cells or species that express the cargo also express the adaptor? For example, AP4 and AP5 are not present in certain organisms such as *Drosophila melanogaster*; therefore, finding an AP4-specific Y-motif conserved in the fly homolog of the cargo under study should be treated with skepticism.

2.2.2 To Predict/
Determine Signal
Relevance to Cargo
Trafficking/Localization:
What Is the Impact
of Specific Adaptor-Signal
Interactions for Cargo
Trafficking/Localization?

In order to address this question, we need to understand the function of specific adaptors; *i.e.*, what is going to happen to the cargo when a signal is bound by its specific adaptor? The function of some adaptors is well established, but in other cases it is not. Nevertheless, knowing the adaptor specificity of a given signal, it is possible to make some predictions in terms of cargo localization or trafficking of normal and signal-mutated cargoes. For example, Fig. 3 depicts the trafficking of a transmembrane protein with a Y-signal to be recognized by AP3.

Following its targeting to the endoplasmic reticulum (ER), the nascent lysosomal-resident cargo is inserted into the ER membrane by sequential action of the signal recognition particle and the translocon systems (*see* Fig. 3 and **Note 2**). After being transported to the Golgi apparatus by vesicle trafficking, cargo would emerge at the trans-Golgi network (TGN) ready to be sorted. Cargo carrying an appropriate signal can be recognized by the AP3 complex and actively sent in route to late endosome/lysosomes (LE/L) or can be passively incorporated onto carriers destined to the plasma membrane (the “default pathway”). Once at the plasma membrane, AP2 will bind and promote cargo internalization (Fig. 4), leading

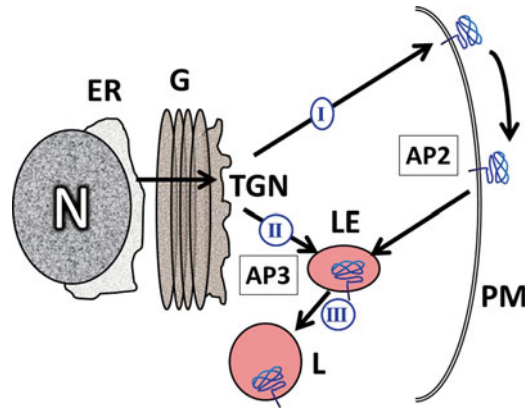


Fig. 4 Traffic of a generic AP3 cargo. Following translation of the messenger RNA in the endoplasmic reticulum (ER), cargo is transported to the Golgi apparatus (G) for further processing to finally emerge at the trans-Golgi network (TGN) from where it would be sorted. While AP3 actively recruits cargo for late endosomal (LE) to lysosome (L) targeting (route II–III), the “default” pathway (route I) would transport cargo to the plasma membrane (PM). At the PM, AP2 promotes for internalization, facilitating a new iteration of sorting (*see text for details*)

to another round of intracellular sorting. Therefore, mutation in this hypothetical AP3-specific signal is predicted to lead to a decrease in LE/L localization (by affecting route III—Fig. 3) and to plasma membrane accumulation (as more protein will be traveling via the default pathway, i.e., route I—Fig. 3). A similar result is to be expected in cases where AP3 function is defective [15].

While this may be perceived as compelling, predictions should be experimentally tested using a model cargo system. A very successful approach for testing the sorting role of a signal is the use of TAC chimeras. The interleukin-2 receptor α -subunit (also known as TAC) is a protein with a very short cytoplasmic region without sorting determinants, for which highly efficient and commercially available reagents for detection (i.e., antibodies: such as the monoclonal antibody 7G7) are available. Therefore, if TAC were to be modified to carry a signal of interest now devoid of the confounding effects of other sequences and motifs (i.e., subjected to a reductionist approach), it would act as a model or reporter cargo. The resulting TAC chimera will traffic and localize (*see Note 6*) only according to the sorting role of the genetically engineered signal.

These reporter cargoes can also be useful to experimentally test adaptor recognition by detecting the formation of a TAC chimera-adaptor complex using immunoprecipitation (IP) techniques (*see Note 6*). Besides TAC chimeras, other approaches such as the previously mentioned two-hybrid or *in vitro* binding techniques can also be instrumental to test the signal-adaptor interaction.

2.3 *Testing the Fate of Cargo*

Often the cytoplasmic regions of cargoes do not contain single but multiple sorting determinants, and those signals can overlap with each other and with sites for posttranslational modification. Therefore, the protein's final destination will result from the interplay of this composite of sequence-encoded information.

Although adaptors and signals discussed in this chapter will not be involved, TM and luminal domains can also play an important role in the fate of the proteins. Therefore, following studies with simplified cargo models (*see* above) and investigations with the actual cargo (i.e., the signal within its native context) should be conducted. The latter should involve the analysis of truncations and point mutations that in the native context would affect signals and regulatory components (e.g., palmitoylation and phosphorylation sites).

3 Protein Sequence Analysis Protocol

In this section we propose a simple protocol that brings into practice the above-discussed foundations of a method for the analysis of cargo sorting information. Here the protocol is applied to the lysosomal TMP Lamp2 (Fig. 2), and although specifically focused on Y-signals and recognition by APs, a similar procedure would apply to the study of any other sorting motif.

3.1 *Finding the Amino Acid Sequence of the Cargo of Interest*

There are several resources that can be explored, for example, the NCBI databases and some organism-specialized sites, such as the Saccharomyces Genome Database (SGD) or FlyBase:

NCBI: <http://www.ncbi.nlm.nih.gov/> (“Protein” database)

SGD: <http://www.yeastgenome.org/>

FlyBase: <http://flybase.org/>

In all cases, be aware of partial sequences, isoforms, and species variation (a corollary to the latter is that conservation of a putative sorting signal means functional relevance but not necessarily a role in protein sorting).

Example

Sequence information for Lamp2 can be found at the NCBI database (Fig. 2). <http://www.ncbi.nlm.nih.gov/protein/P13473.2>.

3.2 *Identification of the Cytosolic Region(s) Within the Cargo of Interest*

Information in terms of what cargo regions are facing the cytosol might be readily available from the database protein entries (*see* above), or it may be necessary to run a topology (TM and cytosolic region) prediction using the cargo sequence. If the latter is required, a few URLs for useful online sites are provided below:

<http://smart.embl-heidelberg.de/>

<http://phobius.sbc.su.se/>

<http://www.enzim.hu/hmmtop/html/submit.html>

Example

The cytosolic region of Lamp2 was predicted to encompass amino acids 401–410 in agreement with information provided by the corresponding NCBI entry (Fig. 2).

**3.3 Search for Motifs
Fitting Sorting Signal
Consensuses**

Contrast the sequence of the cargo cytosolic region(s) against motifs listed in Table 1.

Example

The cytosolic region of Lamp2 was found to contain a C-terminal YXXØ motif (YEQF).

**3.4 Prioritize
Isolated Motifs
According to:**

1. Consensus fitting
2. Accessibility to adaptors
3. Structural environment

http://npsa-pbil.ibcp.fr/cgi-bin/npsa_automat.pl?page=/NPSA/npsa_seccons.html

**3.5 Predict Adaptor
Recognition**

What adaptor would bind the putative sorting signal?

- Prediction of signal-adaptor specificity based on Table 1 (also use Table 2 and ANNs if the AP family is involved).
- Experimental testing of adaptor recognition. Determine the isolated candidate signal ability to bind adaptors: two-hybrid technology, in vitro binding assays, localization analysis, and IP of TAC chimeras.

Example

The Lamp2 YEQF motif fulfills the criteria to be a signal recognized by AP3.

**3.6 Back-to-Context
Analysis**

Mutation (truncations followed by point mutations) of candidate signal within native context followed by adaptor-cargo IP and localization analysis.

4 Conclusion

This chapter discusses the basis and application of a simple method for the identification/testing of candidate sorting signals in TM proteins. The understanding and control of protein sorting and organization is not only important to further our basic knowledge of cell function but also to support the development of medicinal and biotechnological applications.

The proposed protocol takes advantage of the knowledge obtained by researchers using the genetics, molecular biology, and biochemistry toolkits. Now is the turn of information and computational approaches to move our knowledge and capabilities forward.

These approaches will not only create tools to facilitate the analysis of protein information, but will also allow the generation of new hypotheses to be tested on the classical experimental realm.

4.1 Where to Go from Here?

The method described in this chapter has been successfully and routinely used to analyze protein cargo of different origins (yeast, mammalian, etc.). However, in its current form, it relies on manual inspection of sequences and on the use of scattered resources. One obvious limitation of this “manual” approach is its inability to support high-throughput analysis. Although complete collections of protein sequences from entire organisms exist, we lack the tools to systematically analyze the databases for cargos carrying specific sorting signals. This limitation, for example, precludes us from searching for cargoes potentially affected by genetic diseases due to deficiencies in specific adaptors (e.g., Hermansky-Pudlak syndrome forms due to AP3 abnormal function—*see* [15]). If we had this capability, we could predict (and perhaps prevent) cellular abnormalities in patients.

Therefore, there is a clear need for applications that incorporate the myriad of factors discussed in previous sections. Furthermore, the application of information theory [16] should be instrumental for the development of methods to crack the protein-sorting information code.

4.2 Is Benchwork Still Needed?

Absolutely. Our basic cell and molecular knowledge still needs to be pushed forward and classical benchwork will play a central role in this effort. Indeed, there still are some basic research aspects that need to be resolved/clarified, for example, controversies about the specificity of certain adaptors and several reports of functional signals that deviate from the discussed consensus. The existence of previously unrecognized signals in the cytoplasm region of cargoes is another possibility that should be experimentally investigated.

In addition to discussing the basis and application of a method based on resources and analytical tools, from molecular biology to ANNs, this chapter emphasizes the necessity to amalgamate experimental and theoretical methods. The coalescence of *in silico*, *in vitro*, and *in vivo* approaches is what it will take to fully understand adaptor-signal interaction and to therefore move forward knowledge and to fulfill our medical and biotechnological needs.

5 Notes

1. Emergent properties: Properties of the whole that do not result from the sum of the parts. Examples: based on the known properties of proteins, lipids, sugars, and nucleic acids, it is unlikely that we would predict that their combination would lead to a living cell. At the multicellular level, the

properties of individual neurons and accessory cells do not predict that their grouping would lead to consciousness.

2. For basic concepts and terminology in cell biology and biochemistry, from the 1- and 3-letter amino acid codes to the mechanisms of vesicle trafficking, a few textbooks are recommended:

Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, Peter Walter (2007). *Molecular Biology of the Cell*, 5th edition, Garland Science. ISBN: 0815341059; ISBN-13: 9780815341055.

Thomas D. Pollard & William C. Earnshaw (2008). *Cell Biology*, 2nd edition; Saunders/Elsevier. ISBN10:1416022554; ISBN13: 9781416022558.

3. Integral membrane proteins: These proteins are partially embedded into and permanently attached to the membranes of cellular compartments. Transmembrane (TM) proteins (TMP) are a very large subset of integral membrane proteins and are characterized by spanning the membrane. TMP have a TM domain or region made up of approximately two dozen hydrophobic amino acids that separate luminal and cytosolic regions (*see* Fig. 2).
4. The effect on protein structure/function of amino acids such as aspartic and glutamic acids (i.e., with a negatively charged carboxyl group) can be emulated by amino acids serine and threonine following the introduction of a negatively charged phosphate group by phosphorylation. In fact, in order to test the functional relevance of their phosphorylation, amino acids serine and threonine are often intentionally mutated to aspartic or glutamic acid.
5. However, several domains or 3D structures (e.g., ubiquitin) have been found to be capable of conveying sorting information on their 3D array of exposed amino acids (conformational or 3D motifs).
6. Introduction to Biochemistry and cell biology techniques (e.g., immunoprecipitation, immunofluorescence) can be found in *Short Protocols in Cell Biology*. Juan S. Bonifacino (Editor), Mary Dasso (Editor), Joe B. Harford (Editor), Jennifer Lippincott-Schwartz (Editor), Kenneth M. Yamada (Editor). ISBN: 978-0-471-48339-7. February 2004.

Acknowledgments

We are indebted to Arpita Sen (UC Berkeley) and members of the Aguilar lab for critical reading of the manuscript. The Aguilar lab is supported by the Center for the Science of Information, an

NSF Science and Technology Center, under grant agreement CCF-0939370, by the National Institutes of Health and by the National Science Foundation under Grants No. 5 R21 CA151961-02 and 1021377, respectively.

References

1. Esposito G, Clara FA, Verstreken P (2012) Synaptic vesicle trafficking and Parkinson's disease. *Dev Neurobiol* 72(1):134–144. doi:[10.1002/dneu.20916](https://doi.org/10.1002/dneu.20916)
2. Aridor M, Hannan LA (2002) Traffic jams II: an update of diseases of intracellular transport. *Traffic* 3(11):781–790. doi:[10.1034/j.1600-0854.2002.31103.x](https://doi.org/10.1034/j.1600-0854.2002.31103.x)
3. Boehm M, Bonifacino JS (2002) Genetic analyses of adaptin function from yeast to mammals. *Gene* 286(2):175–186. doi:[10.1016/s0378-1119\(02\)00422-5](https://doi.org/10.1016/s0378-1119(02)00422-5)
4. Bonifacino JS (2014) Adaptor proteins involved in polarized sorting. *J Cell Biol* 204(1):7–17. doi:[10.1083/jcb.201310021](https://doi.org/10.1083/jcb.201310021)
5. Hirst J, Irving C, Borner GHH (2013) Adaptor protein complexes AP-4 and AP-5: new players in endosomal trafficking and progressive spastic paraplegia. *Traffic* 14(2):153–164. doi:[10.1111/tra.12028](https://doi.org/10.1111/tra.12028)
6. Ohno H (2006) Physiological roles of clathrin adaptor AP complexes: lessons from mutant animals. *J Biochem* 139(6):943–948. doi:[10.1093/jb.mvj120](https://doi.org/10.1093/jb.mvj120)
7. Green N, Fang H, Kalies KU et al. (2001) Determining the topology of an integral membrane protein. *Current protocols in cell biology/editorial board, Juan S Bonifacino [et al.]. Chapter 5: Unit 5.2.* doi:[10.1002/0471143030.cb0502s00](https://doi.org/10.1002/0471143030.cb0502s00)
8. Vargarajauregui S, Puertollano R (2006) Two di-leucine motifs regulate trafficking of mucopolipin-1 to lysosomes. *Traffic* 7(3):337–353. doi:[10.1111/j.1600-0854.2006.00387.x](https://doi.org/10.1111/j.1600-0854.2006.00387.x)
9. Neumann U, Brandizzi F, Hawes C (2003) Protein transport in plant cells: in and out of the Golgi. *Ann Bot* 92(2):167–180. doi:[10.1093/aob/mcg134](https://doi.org/10.1093/aob/mcg134)
10. Barman S et al (2001) Transport of viral proteins to the apical membranes and interaction of matrix protein with glycoproteins in the assembly of influenza viruses. *Virus Res* 77(1):61–69. doi:[10.1016/s0168-1702\(01\)00266-0](https://doi.org/10.1016/s0168-1702(01)00266-0)
11. Ohno H, Aguilar RC, Yeh D et al (1998) The medium subunits of adaptor complexes recognize distinct but overlapping sets of tyrosine-based sorting signals. *J Biol Chem* 273(40):25915–25921. doi:[10.1074/jbc.273.40.25915](https://doi.org/10.1074/jbc.273.40.25915)
12. Aguilar RC, Boehm M, Gorshkova I et al (2001) Signal-binding specificity of the mu 4 subunit of the adaptor protein complex AP-4. *J Biol Chem* 276(16):13145–13152. doi:[10.1074/jbc.M010591200](https://doi.org/10.1074/jbc.M010591200)
13. Mukherjee D, Hanna CB, Aguilar RC (2012) Artificial neural network for the prediction of tyrosine-based sorting signal recognition by adaptor complexes. *J Biomed Biotechnol*. doi:[10.1155/2012/498031](https://doi.org/10.1155/2012/498031)
14. Parrish JR, Gulyas KD, Finley RL Jr (2006) Yeast two-hybrid contributions to interactome mapping. *Curr Opin Biotechnol* 17(4):387–393. doi:[10.1016/j.copbio.2006.06.006](https://doi.org/10.1016/j.copbio.2006.06.006)
15. Dell'Angelica EC, Shotelersuk V, Aguilar RC et al (1999) Altered trafficking of lysosomal proteins in Hermansky-Pudlak syndrome due to mutations in the beta 3A subunit of the AP-3 adaptor. *Mol Cell* 3(1):11–21. doi:[10.1016/s1097-2765\(00\)80170-7](https://doi.org/10.1016/s1097-2765(00)80170-7)
16. Adami C (2012) The use of information theory in evolutionary biology. *Year Evol Biol* 1256:49–65. doi:[10.1111/j.1749-6632.2011.06422.x](https://doi.org/10.1111/j.1749-6632.2011.06422.x)

Chapter 2

Protein Structural Information Derived from NMR Chemical Shift with the Neural Network Program *TALOS-N*

Yang Shen and Ad Bax

Abstract

Chemical shifts are obtained at the first stage of any protein structural study by NMR spectroscopy. Chemical shifts are known to be impacted by a wide range of structural factors, and the artificial neural network based TALOS-N program has been trained to extract backbone and side-chain torsion angles from ^1H , ^{15}N , and ^{13}C shifts. The program is quite robust and typically yields backbone torsion angles for more than 90 % of the residues and side-chain χ_1 rotamer information for about half of these, in addition to reliably predicting secondary structure. The use of TALOS-N is illustrated for the protein DinI, and torsion angles obtained by TALOS-N analysis from the measured chemical shifts of its backbone and $^{13}\text{C}\beta$ nuclei are compared to those seen in a prior, experimentally determined structure. The program is also particularly useful for generating torsion angle restraints, which then can be used during standard NMR protein structure calculations.

Key words NMR, Chemical shifts, Protein structure, Side-chain conformation, Artificial neural network, Secondary structure, Backbone torsion angle

1 Introduction

1.1 Relations Between Chemical Shifts and Protein Structure

The first step of any protein structural study by NMR spectroscopy typically involves assignment of the multitude of NMR resonances to individual nuclei. Originally, for proteins extracted from natural sources, this only involved assignment of the hydrogen NMR spectra [1, 2]. However, due to extensive resonance overlap in ^1H NMR spectra, this technology was restricted to relatively small proteins. With advances in molecular biology, the vast majority of today's structural studies focus on cloned proteins, typically over-expressed in *Escherichia coli* [3–5]. By using suitable isotopically enriched growth media, it then is readily feasible to obtain essentially full incorporation of the NMR-observable stable isotopes ^{13}C and ^{15}N . These nuclei not only are key to dispersing the crowded NMR spectra in three or four orthogonal frequency dimensions, dramatically reducing the resonance overlap problem; the ^{13}C and

^{15}N chemical shifts themselves have proven to be important reporters on local backbone conformation [6–8]. NMR chemical shifts in proteins are exquisitely sensitive to local conformation. However, they depend on many different factors, including backbone and side-chain torsion angles, neighboring residues, ring currents caused by nearby aromatic groups, hydrogen bonding, electric fields, local strain and geometric distortions, as well as solvent exposure [9–15]. This not only has made it difficult to separately quantify the relation between each of these parameters and the chemical shift; it also makes it impossible to uniquely attribute such a structural parameter to any individual chemical shift.

For protein NMR spectroscopy, triple resonance correlation experiments, which link the resonances of directly bonded ^1H , ^{13}C , and ^{15}N nuclei, are commonly used to assign the chemical shifts of ^1H , ^{13}C , and ^{15}N nuclei in proteins [16–18]. The chemical shift assignment procedure usually consists of two steps: (1) sequence-specific assignment of the backbone atoms and (2) side-chain assignments. Nearly complete chemical shift assignments for backbone and side-chain atoms are commonly required to assign nuclear Overhauser enhancement (NOE) spectra, which classically are used to derive interproton distances that serve as the primary experimental restraints for calculating the protein structure. The backbone ($^1\text{H}\alpha$, $^{13}\text{C}'$, $^{13}\text{C}\alpha$, ^{15}N , and $^1\text{H}^{\text{N}}$) and $^{13}\text{C}\beta$ chemical shifts, which are generally obtained in the earliest stage of any protein NMR study, are particularly useful reporters on local conformation. Their link to secondary structure, as well as to hydrogen bonding and χ_1 side-chain torsion angles, has been long recognized and has been the focus of both empirical studies as well as quantum-chemical calculations [11–15, 19, 20].

1.2 Protein Backbone and Side-Chain Conformation from NMR Chemical Shifts

The rapid increase in the number of proteins, for which both high-resolution structural coordinates have been deposited in the Protein Data Bank (PDB) [21] and NMR chemical shift assignments are available in the BioMagResBank (BMRB) [22], has stimulated the development of quantitative empirical methods to study the relation between protein structure and chemical shifts [23]. Among the wide array of empirical methods, TALOS [20] and its two successors TALOS+ [24] and TALOS-N [25] have become particularly widely used for making accurate ϕ/ψ backbone torsion angle predictions on the basis of the backbone ($^{13}\text{C}\alpha$, $^{13}\text{C}'$, ^{15}N , $^1\text{H}\alpha$, and $^1\text{H}^{\text{N}}$) and $^{13}\text{C}\beta$ chemical shift assignments. These ϕ/ψ predictions can be used to validate NOE-derived NMR structures that did not use chemical shift-derived input parameters or, conversely, to generate additional restraints as input to the protein structure calculation and refinement protocols.

The original TALOS program (Torsion Angle Likelihood Obtained from Shift) searches a protein database, consisting originally of only 20 proteins but later expanded to ca 200 proteins,

with both high-resolution X-ray coordinates and NMR chemical shift assignments. TALOS identifies the ten tripeptide fragments that represent the best match in terms of chemical shifts and residue types to those of a tripeptide segment whose assignments are known and whose structure is under study (the “target protein”). The assumption underlying TALOS is that fragments with similar chemical shifts and residue type typically have similar backbone conformations. Thus, if these ten best-matched fragments have consistent, narrowly clustered values for the ϕ/ψ angles of their center residue, their averages and standard deviations are used as a prediction for the ϕ/ψ angles of the center residue of the target protein tripeptide. If the ϕ/ψ angles of the center residue of these ten best-matched tripeptides fall in different regions of the Ramachandran map, the matches are declared ambiguous, and no prediction is made for the central residue. With this quality control criterion, TALOS predicted ϕ/ψ torsion angles for, on average, ca 72 % of the residues in any given target protein. For TALOS validation proteins, where the true ϕ/ψ angles are known, only about 1.8 % of the predictions were inconsistent with crystallographically determined ϕ/ψ torsion angles. Excluding these 1.8 % erroneous predictions, a root mean square (RMS) difference of ca 13° is observed between predicted and crystallographically observed ϕ/ψ torsion angles.

Although rather robust, the original TALOS program was unable to make definitive predictions for about 28 % of the residues in any given protein. Most of these 28 % are located outside regular secondary structure, exactly those regions where backbone torsion angle information is most needed. TALOS+ was developed to address this shortcoming and to extend the coverage of the program [24]. For a given residue in the target protein, TALOS+ first uses an artificial neural network (ANN) module to predict its three-state distribution in the Ramachandran map, i.e., α , β , and positive- ϕ . This three-state distribution is subsequently used to guide the database search procedure for the ten best matches. With the incorporation of the ANN, TALOS+ is able to increase its coverage to ca 88 %, without sacrificing accuracy. Thus, compared to the original TALOS program, the fraction of residues whose backbone angles cannot be predicted is reduced from ~28 % to ~12 %. Importantly, most of the additional ϕ/ψ torsion angle predictions are made for residues in loop or turn regions, where this information is needed most.

The recently introduced TALOS-N program relies far more extensively on neural network analysis of the input chemical shift data than TALOS+, thereby further increasing coverage, accuracy, and reliability. In addition, TALOS-N is the first program to generate quite accurate predictions for the side-chain χ_1 torsion angles (Fig. 1).

For the ϕ/ψ torsion angle prediction of a given residue i in the target protein, a well-trained two-level feed-forward multilayer

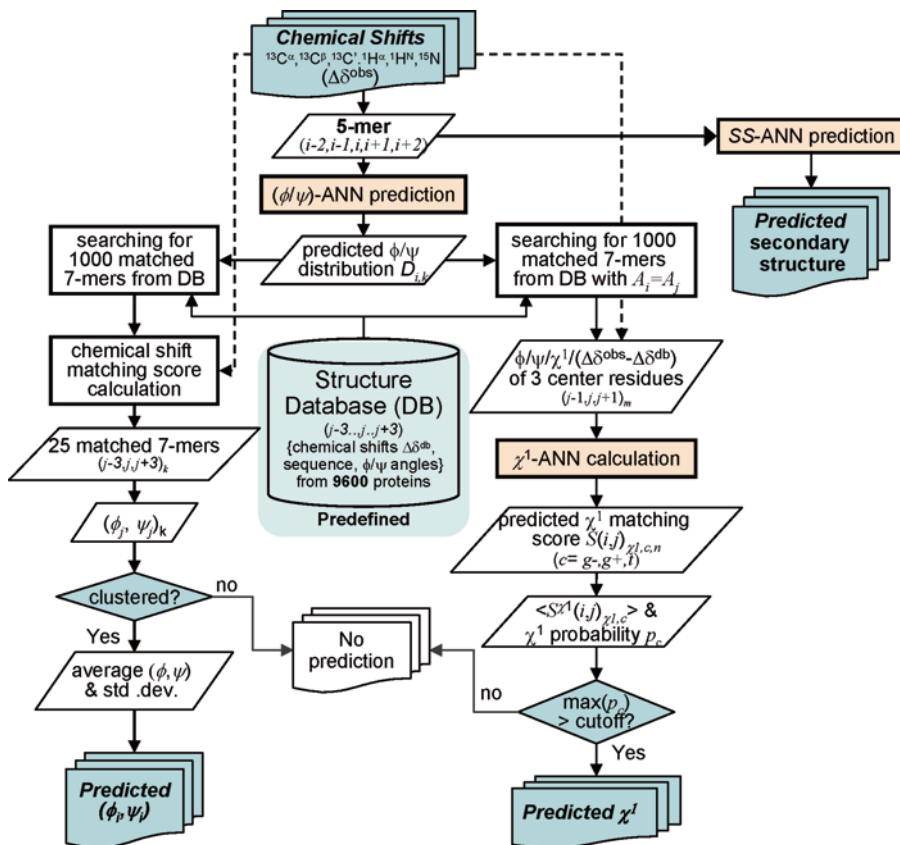


Fig. 1 Flowchart of the TALOS-N program (reproduced from [25] with permission from Springer)

ANN, referred to as a (ϕ, ψ) -ANN, is first used by TALOS-N to predict the 324-state ϕ/ψ distribution of residue i on the basis of the NMR chemical shifts and residue type of itself and its adjacent residues ($i-2$ to $i+2$). Here, the 324-state ϕ/ψ distribution corresponds to the likelihood that residue i adopts torsion angles that fall in any of the 324 voxels, of $20^\circ \times 20^\circ$ each, that make up the Ramachandran map. The ANN-predicted ϕ/ψ distribution is then used solely to search a large crystallographic database (containing 9,523 proteins, with chemical shifts added by a computational method [26]), for a pool of 1,000 heptapeptide fragments with ϕ/ψ angles that best match the 324-state ϕ/ψ distribution. These top 1,000 fragments then are further evaluated for the agreement between their computed chemical shifts and experimental values of the corresponding heptapeptide segment ($i-3$ to $i+3$) in the target protein. The 25 best-matched database heptapeptides are retained, and the ϕ/ψ angles of their center residues are inspected by using an advanced clustering analysis, and subsequently used to make a prediction for the ϕ/ψ angles of the query residue. Validation on an independent set of proteins indicates that backbone

torsion angles can be predicted for a larger, ≥ 90 % fraction of the residues, with an error rate of ca 3.5 % when using an acceptance criterion that is nearly twofold tighter than that used previously by TALOS and TALOS+. The RMS difference between predicted and crystallographically observed ϕ/ψ torsion angles is ca 12° , also slightly better than what was obtained with the earlier versions of the program.

To predict the χ_1 rotameric state (g^- , g^+ or t) for a given residue i (of residue type a) in the target protein, TALOS-N uses another set of ANNs, referred to as $(\chi_1)_a$ -ANNs. The $(\chi_1)_a$ -ANN has been trained to correlate the center residue likelihood of adopting each of the three χ_1 rotameric states to the differences between its observed chemical shifts and those expected on the basis of its backbone conformation. A separate database search procedure is subsequently used to estimate the three-state probability of residue i to adopt the three χ_1 rotameric states. With an optimized error control criterion, TALOS-N predicts χ_1 rotameric states for ca 50 % of the residues, with an “error rate” of ca 10 % when comparing the predicted χ_1 rotameric state to that of any given reference structure. However, we note that the true error is likely to be much lower, as for proteins that have multiple available independently solved X-ray structures, the χ_1 rotameric states of any “erroneous” χ_1 prediction is typically in agreement with that of another X-ray structure [25].

Similar to TALOS+, TALOS-N is also implemented with an ANN-based module for predicting secondary structure (SS) from the NMR chemical shifts. For this purpose, TALOS-N uses two separate ANNs, referred to as SS-ANN and SS_{seq}-ANN, which are trained to correlate the three-state secondary structure classification (helix, sheet, and coil) of a residue to both the chemical shifts and amino acid sequence or to amino acid sequence alone, respectively. The output of these two ANNs is used in a hybrid manner to predict secondary structure for any residue in a protein, regardless of the completeness of chemical shift assignments. The overall correctness of the SS prediction is ca 88 % when NMR chemical shifts are available, dropping to ca 81 % when no chemical shifts are available. In the absence of chemical shifts, TALOS-N matches the accuracy of the best sequence-only secondary structure prediction programs [27, 28].

2 Materials

In this chapter, we use the protein DinI [29] to illustrate the use of TALOS-N for predicting its backbone ϕ/ψ and side-chain χ_1 torsion angles, as well as its secondary structure classification. To follow the examples, both the TALOS-N software package and an input file with correctly formatted chemical shift assignments are needed.

2.1 Software Requirements

The TALOS-N software package, including the required binaries for three of the most common operating systems, Linux, Mac OS X, and Windows, as well as the requisite protein database and scripts, can be downloaded from <http://spin.niddk.nih.gov/bax/software/TALOS-N/> and installed straightforwardly (*see Note 1*). Alternatively, a server version of TALOS-N can be used directly, without installing the TALOS-N software (<http://spin.niddk.nih.gov/bax/nmrserver/talosn/>).

2.2 Data Requirements

An input table containing both the full amino acid sequence and the NMR chemical shift assignments is required, to be prepared with a specific data format (general purpose NMRPipe table format). As an example, an excerpt of such a file is shown below for the protein DinI:

```
DATA FIRST_RESID 2
DATA SEQUENCE RIEVTIAKT SPLPAGAI DA LAGELSRRIQ
YAFPDNEGHV SVRYAAANNL
DATA SEQUENCE SVIGATKEDK QRISEILQET WESADDWFVS E
VARS   RESID RESNAME ATOMNAME SHIFT
FORMAT %4d %1s %4s %8.3f
  2 R    C   174.123
  2 R   CA   55.537
  2 R   CB   32.786
  2 R    H    8.772
  2 R   HA    4.994
  2 R    N  123.394
  3 I    C  173.941
  3 I   CA   60.986
```

The protein's amino acid sequence should be provided in one or more lines starting with the tag "DATA SEQUENCE". Only the one-character residue name is allowed (*see Note 2*) and space characters in the sequence are ignored. An optional line beginning with a tag of "DATA FIRST_RESID" is needed to specify the first residue number of the amino acid sequence listed in the "DATA SEQUENCE" line if the first residue listed is not residue number 1. For the chemical shift table, columns for residue number, one-character residue type (*see Note 2*), atom name (*see Note 3*), and chemical shift value must be included, and their definitions ("RESID," "RESNAME," "ATOMNAME," and "SHIFT," respectively) must be predeclared in a line beginning with a "VARS" tag; a line beginning with a "FORMAT" tag is also required (immediately after the "VARS" line) to define the data type of each corresponding column of the table.

Note that all chemical shifts used as input for TALOS-N are required to be properly referenced (*see Note 4*) to ensure the accuracy and reliability of the predictions. If the protein sample used to collect the NMR chemical shift data is perdeuterated, ^2H isotope

corrections [30] need to be applied for $^{13}\text{C}\alpha$ and $^{13}\text{C}\beta$ chemical shifts (*see Note 5*).

Other standard chemical shift formats, such as the NMR-Star format used by the BMRB database, can also be used as input after a format conversion. A conversion script is provided in the TALOS-N package for this purpose (*see Note 6*). The server version of TALOS-N includes automated chemical shift format identification and can use the NMR-Star format chemical shift file directly as input, without requiring prior format conversion.

3 Methods

3.1 TALOS-N Prediction

The TALOS-N prediction can be performed for DinI with an input chemical shift file of name “inCS.tab” by typing the command:

```
talosn -in inCS.tab
```

The program first converts the chemical shifts (δ) of each query residue to its corresponding secondary chemical shifts ($\Delta\delta$) by subtracting a residue-type-dependent random coil value, as well as corrections to account for the residue types of its two immediate neighbors. The converted secondary chemical shifts are stored in a file named “predAdjCS.tab” (in the “SHIFT” column), together with the original chemical shifts (“CS_OBS”) and the corresponding corrections (“CS_ADJ”, which is the random coil value including nearest neighbor ($i\pm 1$) residue-type correction) used to calculate the secondary chemical shifts. To make a ϕ/ψ angle prediction, the converted secondary chemical shifts together with the amino acid-type information are used as inputs for the (ϕ/ψ)-ANN to calculate the 324-state ϕ/ψ distribution for each predictable residue (*see Note 7*), with the output stored in a file named “predANN.tab”. A database search step is then performed to search a 9,523-protein database for the 25 best-matched heptapeptides in terms of the 324-state ϕ/ψ angle distribution, the secondary chemical shifts, and the amino acid type. A single file, “predAll.tab”, is generated in this step to store the information of those best database matches for each of the residues in the target protein. A final summarization and quality control step is performed to identify outliers in the 25 best-matching heptapeptides by evaluating the clustering of the ϕ/ψ angles of their center residues in the Ramachandran map or by using the observed ϕ/ψ of a reference structure if such a structure is available (this requires an additional option “-ref ref.pdb” in the command line, where “ref.pdb” is the name for the reference structure). A summary file “pred.tab” is then created, displaying the average ϕ and ψ values (in the PHI and PSI columns) and their respective standard deviations (DPHI and DPSI), as well as an aggregate, weighted χ^2 score (DIST, *see Eq. 12 of reference [25]*), reflecting how well the target

protein chemical shifts match those of the database fragments. An excerpt of this file for DinI is shown below:

```

VARS    RESID RESNAME PHI PSI DPHI DPSI DIST
S2 COUNT CS_COUNT CLASS
FORMAT %4d %s %8.3f %8.3f %8.3f %8.3f %8.3f
%5.3f %2d %2d %s
      2 R -107.206  129.115      9.502      7.843
0.293 0.873 25 16 Strong
      3 I -117.237  126.352      6.691      6.523
0.180 0.883 25 18 Strong

```

For each predictable residue, or residue with sufficient input chemical shifts (*see Note 7*), a final classification is made (listed as the last “CLASS” column in the “pred.tab” file) for its ϕ/ψ angle prediction by a summarization step, detailed below. Prior to making this final classification, the program calculates the predicted backbone rigidity as reflected in the “random coil index” order parameter, RCI-S² [31], which scales between 0 (total disorder) and 1 (fully rigid). Its values are included under the “S2” column in the “pred.tab” file. Residues below the threshold RCI-S² ≤ 0.6 are assigned as dynamic (receiving a “Dyn” classification) in “pred.tab”. For other residues, a classification of strongly unambiguous is assigned (with a “Strong” tag) if the center residues of all 25 best-matching heptapeptides locate in a consistent ϕ/ψ region in the Ramachandran map. A generously unambiguous classification is assigned (with a “Generous” tag) if the center residues of only the top ten best matches cluster in a consistent ϕ/ψ region. All other cases are considered ambiguous (classified with a “Warn” tag), even though inspection of their Ramachandran map population may contain very useful information. For example, often the ambiguous residues will cluster in two distinct regions of the Ramachandran map, and the investigator can explore both options during structure calculations.

For the predictable residues, the ϕ/ψ angles are calculated by averaging the ϕ/ψ angles of the center residues of all 25 best matches (for residues classified as “Strong”) or from the top ten best matches (for a “Generous” prediction) and shown in the “PHI”/“PSI” columns. The estimated uncertainties in the predicted ϕ/ψ angles are calculated from their standard deviations from these averages and listed in the “DPHI” and “DPSI” columns. Only when a known reference structure is provided as input to the program will the predicted ϕ/ψ values be compared to the observed ϕ/ψ angles in this reference structure for all unambiguously predicted (“Strong” and “Generous”) residues. A prediction is labeled as “Bad” if the predicted and the observed ϕ/ψ angles are not consistent (*see Note 8*).

For DinI, 71 residues (out of a total of 81) are obtained with unambiguous ϕ/ψ angles prediction, 2 have an ambiguous ϕ/ψ angle prediction, and 6 are predicted as dynamic (Fig. 2).

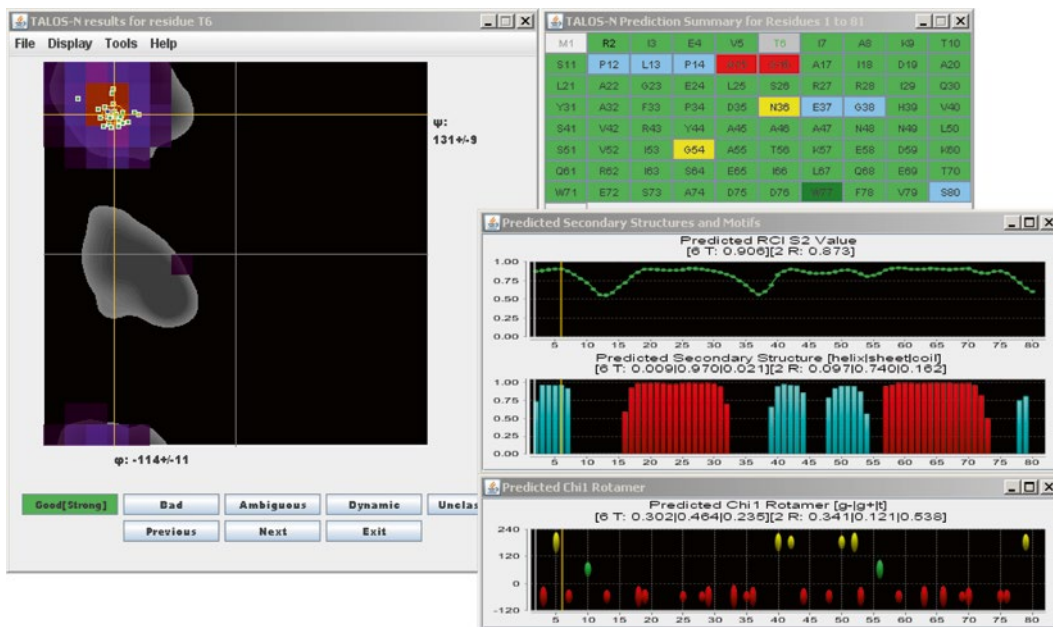


Fig. 2 Graphic TALOS-N inspection interface for protein DinI. (For details, *see* Subheading 3.2 or the TALOS-N webpage <http://spin.niddk.nih.gov/bax/software/TALOS-N/>)

Among those 71 unambiguous predictions, 70 are classified as “Strong” and two (Ala15 and Gly16) are designated “Bad” after inspecting their consistency relative to the ϕ/ψ angles observed in the reference NMR structure. It is worth noting that, in the reference structure, these latter two residues (with ϕ/ψ angles of $-57^\circ/49^\circ$ and $-176^\circ/-18^\circ$, respectively) are located in very lowly populated regions of the Ramachandran map, i.e., they are statistically unlikely to occur. Without further experimental data, it is not possible to decide whether the “Bad” classification refers to the reference structure or to the quality of the prediction.

After TALOS-N prediction of ϕ/ψ angles has been completed, another database search and ANN-based procedure is performed to predict the χ_1 rotameric states. A χ_1 rotamer prediction summary file “predChi1.tab” is created with an excerpt of this file shown below for DinI:

```
VARS RESID RESNAME CS_COUNT CHI1_OBS Q_Gm Q_Gp Q_T CLASS
      FORMAT %4d %s %2d %8.3f %5.3f %5.3f %5.3f %s
      2 R 16 -69.938 0.341 0.121 0.538 na
      3 I 18 -62.494 0.873 0.063 0.063 g-
      4 E 18 -61.087 0.312 0.093 0.595 na
      5 V 18 178.554 0.073 0.055 0.872 t
      6 T 18 66.182 0.302 0.464 0.235 na
      7 I 18 -75.725 0.713 0.143 0.143 g-
```

For a query residue of residue type a (excluding Gly, Pro, and Ala) with sufficient input chemical shifts (*see Note 7*), TALOS-N first searches the database for the 1,000 best-matched heptapeptides in terms of the backbone torsion angles and residue types. It then uses a trained $(\chi_1)_a$ -ANN to calculate a χ_1 matching score for each database match, which measures the likelihood of the center residue of the database heptapeptide to adopt the same χ_1 rotameric state as the query residue. The program then derives a three-state probability score, P_c , for the query residue to adopt each of the three χ_1 rotameric states ($c = g, g^+,$ and t , stored in the columns “Q_Gm,” “Q_Gp,” and “Q_T,” respectively, in “pred-Chil.tab”). TALOS-N then classifies the prediction for the query residue to adopt χ_1 -rotamer state c ($g, g^+,$ or t , as listed in the last column of “CLASS” in the “predChil.tab” file) only when the predicted probability for state c is significantly higher than that for the other two states, by default $P_c > 0.6$. Otherwise, an ambiguous classification is assigned (with a “na” tag). Details of other contents in “predChil.tab” are as follows: the column of “CS_COUNT” is for the count of the experimental chemical shifts of the target residue itself and its two neighbors; when a reference structure is provided, a “CHI1_OBS” column is provided to display the χ_1 angle observed in the reference structure. For DinI, TALOS-N makes χ_1 rotameric state predictions for 30 out of a total of 61 (non-Gly/-Pro/-Ala) residues, among which three (Asp35, Asn48, and Asp75) differ in their predicted χ_1 rotameric state from the reference NMR structure (PDB entry 1GHH).

Next to predicting ϕ , ψ , and χ_1 torsion angles, TALOS-N also predicts the protein’s secondary structure. For residues with chemical shift assignments, a two-level neural network, SS-ANN, is trained to make a three-state secondary structure prediction (H, E, or L, representing for α -helix, β -sheet, and loop, respectively) on the basis of both the chemical shifts and the amino acid sequence information. In addition, another two-level ANN, referred to as SS_{seq}-ANN, is trained by using solely the amino acid sequence information. It can be used to make predictions for residues that lack chemical shift information. However, this SS_{seq}-ANN is used more generally by TALOS-N in a hybrid manner with the SS-ANN to make secondary structure prediction for proteins when chemical shift assignments are incomplete. TALOS-N generates an output file “predSS.tab” to store the predicted secondary structure. An excerpt of this file is shown below for DinI:

```

VARS RESID RESNAME CS_CNT CS_CNT_R2 Q_HQ_EQ_L CONFIDENCE SS_CLASS
FORMAT %4d %1s %2d %2d %8.3f %8.3f %8.3f %4.2f %s
      1 M 10 4    0.333    0.333    0.333 0.00 L
      2 R 16 6    0.097    0.740    0.162 0.58 E
      3 I 18 6    0.027    0.970    0.003 0.94 E
      4 E 18 6    0.006    0.968    0.026 0.94 E

```

5	V	18	6	0.004	0.963	0.033	0.93	E
6	T	18	6	0.009	0.970	0.021	0.95	E

Details of its contents are as follows: the column of “CS_CNT_R2” lists the number of experimental chemical shifts of the target residue; “CS_CNT” contains the count of experimental chemical shifts of the target residue plus its two immediate neighbors; the columns “Q_H,” “Q_E,” and “Q_L” list the SS-ANN (or SS_{seq}-ANN) predicted probability for the target residue to be of secondary structure type “H,” “E,” and “L,” respectively; the values shown in the “CONFIDENCE” column represent the confidence of the three-state secondary structure prediction for a given target residue, calculated from the difference of maximal and median values of “Q_H,” “Q_E,” and “Q_L”; and the text listed in the “SS_CLASS” column shows the final secondary structure classification assigned by the program, i.e., one of the three states with the maximal predicted probability.

For DinI, when comparing to the output of the DSSP program [32] for the reference structure (PDB entry 1GHH), the overall correctness ratio of the TALOS-N predicted secondary structure is 70/81. In this respect, it is important to note that, even for proteins of known structure, secondary structure assignment can be ambiguous, as reflected in only ca 90 % agreement among the output of some of the most popular programs [23].

As mentioned above, TALOS-N predictions can either be made locally by downloading the requisite programs or be performed via the TALOS-N server (<http://spin.niddk.nih.gov/bax/nmrserver/talosn/>), which requires a chemical shift file as input and an e-mail address to send back the prediction results, including all abovementioned output files, such as “pred.tab”, “predChil.tab”, “predSS.tab”, “predS2.tab”, “predAll.tab”, “predAdjCS.tab”, and “predANN.tab”.

3.2 Manual Inspection and Adjustment

The TALOS-N predictions can be inspected and further adjusted by using a Java graphic program, jrama. Two examples of command line calls of this program are:

```
jrama -in pred.tab
jrama -in pred.tab -ref DinI.pdb
```

Figure 2 shows the jrama graphic interface, loaded with the TALOS-N predicted results for DinI. The left panel of the graphic interface shows a map of the ϕ/ψ angles of the center residues of the 25 best-matched heptapeptides in the database (green squares) and the query residue Thr-6 (blue, depicting the angles observed in the NMR-derived PDB entry 1GHH), superimposed on a Ramachandran map, depicting in gray the “most favorable” ϕ/ψ angles for Thr, i.e., those most commonly observed in high-resolution crystal structures of a very large array of proteins. The 324 (ϕ/ψ)-ANN-predicted scores for Thr-6 are shown as colored

voxels but only for those that are populated at least one standard deviation above the average predicted voxel density. The top right panel displays the amino acid sequence of DinI, with residues colored according to their ϕ/ψ prediction classification. Missing predictions (e.g., residue M1) are shown in light gray, consistent predictions in light or dark green (for “Strong” and “Generous” predictions, respectively), ambiguous predictions in yellow, and dynamic residues in blue. Three other panels correspond to the RCI-S² value, the predicted secondary structure (red, α -helix; aqua, β -sheet), with the height of the bars reflecting the probability assigned by the SS-ANN secondary structure prediction. The bottom right panel depicts the χ_1 rotamer predictions (red oval, \mathcal{G} ; green, \mathcal{G}^+ ; yellow, \mathcal{I}), with the height of the ovals corresponding to the probability assigned by the χ_1 rotameric state prediction.

The TALOS-N prediction (including the summary of TALOS-N predicted ϕ/ψ angles) is normally performed with the default parameters and settings. However, the left panel also can be used to manually adjust the prediction classification of a given query residue according to a user’s preference. The prediction files then will be overwritten to reflect any changes made interactively.

3.3 Generation of Angular Restraints

The TALOS-N output can be converted into ϕ and ψ torsion angle restraints that then can be used directly as input for a conventional protein NMR structure calculation procedure [33, 34]. Two convenient scripts, “talos2dyana.com” and “talos2xplor.com”, are included in the TALOS-N software package for this purpose. These scripts read predicted ϕ and ψ angles from the TALOS-N prediction summary file “pred.tab” and generate for each residue with an unambiguous TALOS-N prediction (classified as “Strong” or “Generous”) a ϕ and a ψ torsion angle restraint (*see Note 9*). These torsion angle restraints can be stored in either CYANA format, as shown below for residues 2 and 3 of DinI:

2	ARG	PHI	-127.2	-87.2
2	ARG	PSI	109.1	149.1
3	ILE	PHI	-137.2	-97.2
3	ILE	PSI	106.4	146.4

or in XPLOR format:

```
assign (resid 1 and name C ) (resid 2 and name N )
      (resid 2 and name CA ) (resid 2 and name C ) 1.0 -107.2 20.0 2
assign (resid 2 and name N ) (resid 2 and name CA )
      (resid 2 and name C ) (resid 3 and name N ) 1.0 129.1 20.0 2
assign (resid 2 and name C ) (resid 3 and name N )
      (resid 3 and name CA ) (resid 3 and name C ) 1.0 -117.2 20.0 2
assign (resid 3 and name N ) (resid 3 and name CA )
      (resid 3 and name C ) (resid 4 and name N ) 1.0 126.4 20.0 2
```

These input restraints then can be used for protein structure calculations as a complement to the conventional NOE distance restraints. Note that such chemical shift-derived torsion angle restraints alone are typically insufficient to reach a converged protein structure as each torsion angle contains a substantial uncertainty ($\pm 20^\circ$, in the above example), and these uncertainties rapidly accumulate when building the protein chain. Moreover, as mentioned above, predictions are generally only about 90 % complete and may contain errors.

4 Notes

1. An installation shell script “install.com” is provided with the TALOS-N software package, which can be used for installing and configuring the TALOS-N program on a Linux or a Mac OS X system. After the installation, two starting shell scripts “talosn” and “jrama” are generated with properly configured installation paths for the system-specific binary and all required databases. For a Windows system, TALOS-N can be installed by simply uncompressing the package. However, when running the TALOS-N program, the TALOS-N installation path (“\$talosnDir”) must be specified on the fly, for example, with the command (*see* Subheading 3.1) of “\$talosnDir/bin/TALOS.exe -in inCS -talosnDir \$talosnDir”.
2. In both the sequence header and the chemical shift data table, the lowercase “c” must be used for oxidized Cys ($\delta^{13}\text{C}\beta \sim 42.5$ ppm) and uppercase “C” for reduced Cys ($\delta^{13}\text{C}\beta \sim 28$ ppm), “h” for protonated His, and “H” for deprotonated His.
3. Atom names should be given exactly as: “HA” for $\text{H}\alpha$ atoms of all non-Gly residues; “HA2” for the first $\text{H}\alpha$ atom of a Gly residue and “HA3” for the second; “C” for C' (CO) atoms; “CA” for $\text{C}\alpha$ atoms; “CB” for $\text{C}\beta$ atoms; “N” for amide nitrogen atoms; and “HN” for amide hydrogens. Data for all other atom types will be ignored.
4. All ^{13}C chemical shifts (including $\delta^{13}\text{C}\alpha$, $\delta^{13}\text{C}\beta$, and $\delta^{13}\text{C}'$) should be referenced relative to the methyl groups of 4,4-dimethyl-4-silapentane-1-sulfonic acid, or DSS [35]. The ^{15}N chemical shifts used as input for TALOS-N should be referenced relative to liquid ammonia at 25 °C [35]. A pre-check module in TALOS-N will be used to identify possible referencing problems with the $\delta^{13}\text{C}\alpha$, $\delta^{13}\text{C}\beta$, $\delta^{13}\text{C}'$, and $\delta^1\text{H}\alpha$ chemical shift inputs [36] when running a typical TALOS-N command with an additional “-check” option, for example, by using the command line input argument “talosn -in inCS.tab -check”. This module first converts the chemical shifts (δ) of each residue to secondary chemical shifts ($\Delta\delta$; *see*

Subheading 3.1) and subsequently evaluates these by correlating $\Delta\delta^{13}\text{C}\alpha$, $\Delta\delta^{13}\text{C}\beta$, $\Delta\delta^{13}\text{C}'$, and $\Delta\delta^1\text{H}\alpha$ to the reference-free entity, $\Delta\delta^{13}\text{C}\alpha - \Delta\delta^{13}\text{C}\beta$ [36]. The estimated chemical shift referencing offsets, as well as their corresponding fitting error, will be printed for $\delta^{13}\text{C}\alpha$, $\delta^{13}\text{C}\beta$, $\delta^{13}\text{C}'$, and $\delta^1\text{H}\alpha$. An offset correction generally is only needed when the estimated referencing offset exceeds the average fitting error by more than about five standard deviations. This pre-check module will also identify residues with unusual chemical shifts, for which secondary chemical shifts fall outside the expected range. Such chemical shift outliers, especially those with highly unusual chemical shifts, for which secondary chemical shifts deviate from the expected range by more than two times of the normal range of secondary chemical shifts, may correspond to experimental errors and need to be inspected carefully prior to using them for making torsion angle predictions.

5. ^2H isotope chemical shift corrections for $^{13}\text{C}\alpha$ and $^{13}\text{C}\beta$ [30] can be applied before starting the TALOS-N prediction, i.e., when generating the secondary chemical shifts. To do this, an additional option “-iso” must be added when running a TALOS-N prediction, for example, by using a command line argument of the form “talosn -in inCS.tab -iso”.
6. A conversion Unix shell script, `bmr2talos.com`, is included with the TALOS-N package and can be used to convert a NMR-Star format chemical shift table, used by the BMRB database, to TALOS format. An example command line for using this script is “`bmr2talos.com bmr.str > inCS.tab`”.
7. To ensure the prediction accuracy and reliability for a given query residue, the chemical shift sufficiency is first inspected by the program for the residue itself and its two immediate neighbors. If at least two of the three residues have at least three chemical shifts, the center residue is considered to be predictable.
8. The consistency between the predicted ϕ/ψ values ($\phi_{\text{pred}}/\psi_{\text{pred}}$) and the observed ϕ/ψ angles ($\phi_{\text{obs}}/\psi_{\text{obs}}$) is defined by

$$\sqrt{(\phi_{\text{pred}} - \phi_{\text{obs}})^2 + (\psi_{\text{pred}} - \psi_{\text{obs}})^2} < 60^\circ.$$
9. For a residue with a “Strong” classification of its prediction, the ϕ and ψ angle restraints are set to $\langle\phi\rangle \pm 2\sigma$ and $\langle\psi\rangle \pm 2\sigma$, where $\langle\phi\rangle$ and $\langle\psi\rangle$ are the averaged TALOS-N predictions and 2σ is the larger of 20° or two standard deviations of the TALOS-N prediction. For a residue classified with a “Generous” prediction, the ϕ and ψ angle restraints are less tight, $\langle\phi\rangle \pm 3\sigma$ and $\langle\psi\rangle \pm 3\sigma$, with an allowed range of the larger of 30° or three standard deviations of the TALOS-N prediction.

Acknowledgments

This work was funded by the Intramural Research Program of the NIDDK, NIH.

References

1. Wüthrich K (1986) NMR of proteins and nucleic acids. Wiley, New York
2. Englander SW, Wand AJ (1987) Main-chain-directed strategy for the assignment of ^1H NMR spectra of proteins. *Biochemistry* 26: 5953–5958
3. Oh BH, Westler WM, Darba P et al (1988) Protein ^{13}C spin systems by a single two-dimensional nuclear magnetic resonance experiment. *Science* 240:908–911
4. Ikura M, Kay LE, Bax A (1990) A novel approach for sequential assignment of ^1H , ^{13}C , and ^{15}N spectra of larger proteins: heteronuclear triple-resonance three-dimensional NMR spectroscopy. Application to calmodulin. *Biochemistry* 29:4659–4667
5. Wagner G (1993) Prospects for NMR of large proteins. *J Biomol NMR* 3:375–385
6. Saito H (1986) Conformation-dependent $\text{C}13$ chemical shifts—a new means of conformational characterization as obtained by high resolution solid state $\text{C}13$ NMR. *Magn Reson Chem* 24:835–852
7. Wishart DS, Sykes BD, Richards FM (1991) Relationship between nuclear magnetic resonance chemical shift and protein secondary structure. *J Mol Biol* 222:311–333
8. Spera S, Bax A (1991) Empirical correlation between protein backbone conformation and Ca and Cb ^{13}C nuclear magnetic resonance chemical shifts. *J Am Chem Soc* 113: 5490–5492
9. Haigh CW, Mallion RB (1979) Ring current theories in nuclear magnetic resonance. *Prog Nucl Magn Reson Spectrosc* 13:303–344
10. Avbelj F, Kocjan D, Baldwin RL (2004) Protein chemical shifts arising from alpha-helices and beta-sheets depend on solvent exposure. *Proc Natl Acad Sci U S A* 101: 17394–17397
11. de Dios AC, Pearson JG, Oldfield E (1993) Secondary and tertiary structural effects on protein NMR chemical shifts—an ab initio approach. *Science* 260:1491–1496
12. Case DA (1998) The use of chemical shifts and their anisotropies in biomolecular structure determination. *Curr Opin Struct Biol* 8: 624–630
13. Vila JA, Aramini JM, Rossi P et al (2008) Quantum chemical C-13(alpha) chemical shift calculations for protein NMR structure determination, refinement, and validation. *Proc Natl Acad Sci U S A* 105:14389–14394
14. Kohlhoff KJ, Robustelli P, Cavalli A et al (2009) Fast and accurate predictions of protein NMR chemical shifts from interatomic distances. *J Am Chem Soc* 131:13894–13895
15. Asakura T, Taoka K, Demura M et al (1995) The relationship between amide proton chemical shifts and secondary structure in proteins. *J Biomol NMR* 6:227–236
16. Bax A, Grzesiek S (1993) Methodological advances in protein NMR. *Acc Chem Res* 26:131–138
17. Sattler M, Schleucher J, Griesinger C (1999) Heteronuclear multidimensional NMR experiments for the structure determination of proteins in solution employing pulsed field gradients. *Prog Nucl Magn Reson Spectrosc* 34:93–158
18. Salzmann M, Wider G, Pervushin K et al (1999) TROSY-type triple-resonance experiments for sequential NMR assignments of large proteins. *J Am Chem Soc* 121:844–848
19. Wagner G, Pardi A, Wuthrich K (1983) Hydrogen-bond length and H-1-NMR chemical-shifts in proteins. *J Am Chem Soc* 105:5948–5949
20. Cornilescu G, Delaglio F, Bax A (1999) Protein backbone angle restraints from searching a database for chemical shift and sequence homology. *J Biomol NMR* 13:289–302
21. Berman HM, Kleywegt GJ, Nakamura H et al (2012) The Protein Data Bank at 40: reflecting on the past to prepare for the future. *Structure* 20:391–396
22. Markley JL, Ulrich EL, Berman HM et al (2008) BioMagResBank (BMRB) as a partner in the Worldwide Protein Data Bank (wwPDB): new policies affecting biomolecular NMR depositions. *J Biomol NMR* 40:153–155
23. Wishart DS (2011) Interpreting protein chemical shift data. *Prog Nucl Magn Reson Spectrosc* 58:62–87
24. Shen Y, Delaglio F, Cornilescu G et al (2009) TALOS+: a hybrid method for predicting

- protein backbone torsion angles from NMR chemical shifts. *J Biomol NMR* 44:213–223
25. Shen Y, Bax A (2013) Protein backbone and sidechain torsion angles predicted from NMR chemical shifts using artificial neural networks. *J Biomol NMR* 56:227–241
 26. Shen Y, Bax A (2010) SPARTA plus: a modest improvement in empirical NMR chemical shift prediction by means of an artificial neural network. *J Biomol NMR* 48:13–22
 27. Jones DT (1999) Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol* 292:195–202
 28. Rost B, Sander C (1993) Prediction of protein secondary structure at better than 70 percent accuracy. *J Mol Biol* 232:584–599
 29. Ramirez BE, Voloshin ON, Camerini-Otero RD et al (2000) Solution structure of DinI provides insight into its mode of RecA inactivation. *Protein Sci* 9:2161–2169
 30. Maltsev AS, Ying JF, Bax A (2012) Deuterium isotope shifts for backbone ^1H , ^{15}N and ^{13}C nuclei in intrinsically disordered protein alpha-synuclein. *J Biomol NMR* 54:181–191
 31. Berjanskii MV, Wishart DS (2005) A simple method to predict protein flexibility using secondary chemical shifts. *J Am Chem Soc* 127:14970–14971
 32. Kabsch W, Sander C (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22:2577–2637
 33. Schwieters CD, Kuszewski JJ, Tjandra N et al (2003) The Xplor-NIH NMR molecular structure determination package. *J Magn Reson* 160:65–73
 34. Herrmann T, Guntert P, Wuthrich K (2002) Protein NMR structure determination with automated NOE assignment using the new software CANDID and the torsion angle dynamics algorithm DYANA. *J Mol Biol* 319:209–227
 35. Markley JL, Bax A, Arata Y et al (1998) Recommendations for the presentation of NMR structures of proteins and nucleic acids (Reprinted from *Pure and Applied Chemistry*, vol 70, pp. 117–142, 1998). *J Mol Biol* 280:933–952
 36. Wang LY, Eghbalian HR, Bahrami A et al (2005) Linear analysis of carbon-13 chemical shift differences and its application to the detection and correction of errors in referencing and spin system identifications. *J Biomol NMR* 32:13–22

Predicting Bacterial Community Assemblages Using an Artificial Neural Network Approach

Peter Larsen, Yang Dai, and Frank R. Collart

Abstract

Microbial communities are found in nearly all environments and play a critical role in defining ecosystem service. Understanding the relationship between these microbial communities and their environment is essential for prediction of community structure, robustness, and response to ecosystem changes. Microbial Assemblage Prediction (MAP) describes microbial community structure as an artificial neural network (ANN) that models the microbial community as functions of environmental parameters and community intra-microbial interactions. MAP models can be used to predict community assemblages over a wide range of possible environmental parameters, extrapolate the results of point observations across spatial scales, and make predictions about how microbial communities may fluctuate as the result of changes in their environment.

Key words Modeling, Microbial assemblage prediction, Microbial ecology, Systems biology, Metagenomics

1 Introduction

From boiling lakes [1, 2] to habitats under miles of Antarctic ice [3], from the upper atmosphere [4, 5] to deep in the planet's crust [6–8], bacteria are found not only to thrive but to be critical drivers of Earth's biological and geochemical systems. Microbial metabolism contributes to all known biogeochemical cycles and has both direct and indirect impacts on Earth's climate. These organisms have been implicated in mass extinction events [9] and are predicted to play important roles in future global warming events [10].

The submitted manuscript has been created by UChicago Argonne, LLC, operator of Argonne National Laboratory ("Argonne"). Argonne, a US Department of Energy Office of Science laboratory, is operated under contract no. DE-AC02-06CH11357. The US Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in the said article to reproduce and prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the government.

The identity of these key players in Earth's ecosystems has until recently remained elusive. The diversity of these communities is vast, with the majority of microbial species uncharacterized in the lab and their very ubiquity challenges systematic sampling. Metagenomic sequencing, the sequencing of bacterial genomic DNA extracted directly from environmental samples, has opened these previously intractable ecosystems for analysis [11]. However, there is a wide divergence between the scale at which metagenomic sampling can be conducted and the scale of possible mechanisms by which microbial population may affect ecosystems [12]. This divergence supports the need for computational tools that leverage relatively sparse microbial population observations and extrapolate those biological observations to prediction of population structure changes over time, space, or environmental conditions [13].

Here, we present a computational method, Microbial Assemblage Prediction (MAP), for leveraging microbial population data using artificial neural networks (ANNs) to model microbial populations, first described by Larsen et al. [14]. MAP uses an ANN approach to predict a bacterial population structure, to a given taxonomic resolution, as a function of environmental parameters. This approach considers not only the impact of environmental factors on bacterial taxonomic abundance but also the interrelationships between the abundance of one taxonomic group on another. Required data for this approach includes a set of metagenomic-derived microbial population structure profiles collected across a range of environmental conditions.

There are two principal steps in the process for construction of the MAP model (Fig. 1). The first is to generate an environmental interaction network (EIN). The EIN is a directed acyclic graph that is derived from the statistical dependencies between environmental parameters and relative abundance of microbial taxa. The EIN is generated such that the root nodes of the network are always environmental parameters and never microbial taxa. In the second step, the EIN is used as the underlying topology of the MAP ANN. Each node in the ANN has a value that is defined as a function of the values of its parent nodes with the root environmental parameter nodes taking values from biological observations. In this way, values for all nodes in the ANN are ultimately mathematical functions of environmental parameter data. This enables the generation of MAP models of microbial populations that can be used to predict microbial population structure across multiple observed or extrapolated environmental conditions (Fig. 2). Model predictions can be used to direct future sampling efforts and design hypothesis-generated experiments that interrogate the ecological relationships in microbial communities.

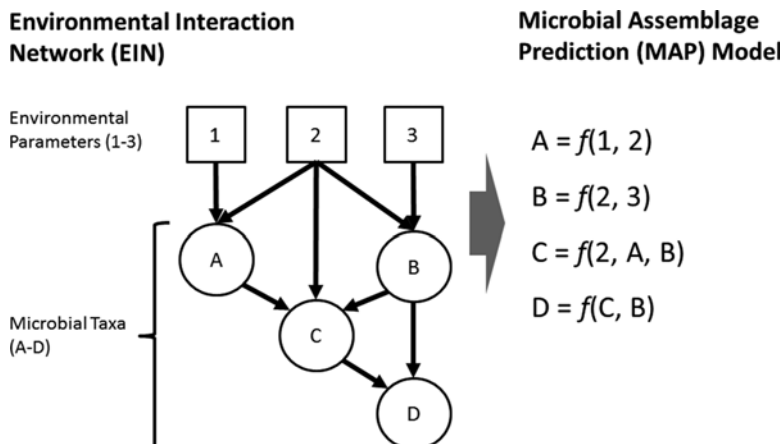


Fig. 1 Two steps to generate MAP model. Generation of MAP model requires as input a set of related microbial population structures and corresponding environmental parameter data. The first step is to generate the environmental interaction network (EIN) (*left panel*). The EIN is the set of identified statistical relationships between environmental parameters and microbial taxa abundances and between microbial taxa abundances, presented as a directed acyclic graph. In the small, theoretical example above, there are three environmental parameters and four microbial taxa represented in the network. In an EIN, root nodes are always environmental parameters. The second step of generating a MAP model (*right panel*) is fitting the data to equations derived from the EIN. Each value of a node in the EIN is defined to be a function of the values of its parent nodes. In the complete MAP model, the relative abundance of every taxa node in the network is ultimately a function of the value of environmental parameter nodes

2 Materials

2.1 Collected Biological Observations

Two kinds of data are required in MAP models: microbial population structure and environmental parameter data (*see Note 1*).

While the list of possible environments and techniques for metagenomic sampling are outside of the scope of this protocol, methods for metagenomic sample collection and sequencing are reviewed by Thomas et al. [11]. Raw metagenomic data must be analyzed to identify the relative abundance of specific taxonomic groups present in the microbial populations. There are many available computational tools for this, but “QIIME” for 16 s RNA sequence data (<http://qiime.org/>) [15] and “MG-Rast” for shotgun metagenomic data (<http://metagenomics.anl.gov/>) [16] are examples of excellent and freely available software solutions. The population data needs to be binned to some level of taxonomy (e.g., order, class, or genus). The selection of specific taxonomic level to which data is binned depends greatly on the nature of the microbial population being modeled but generally should be

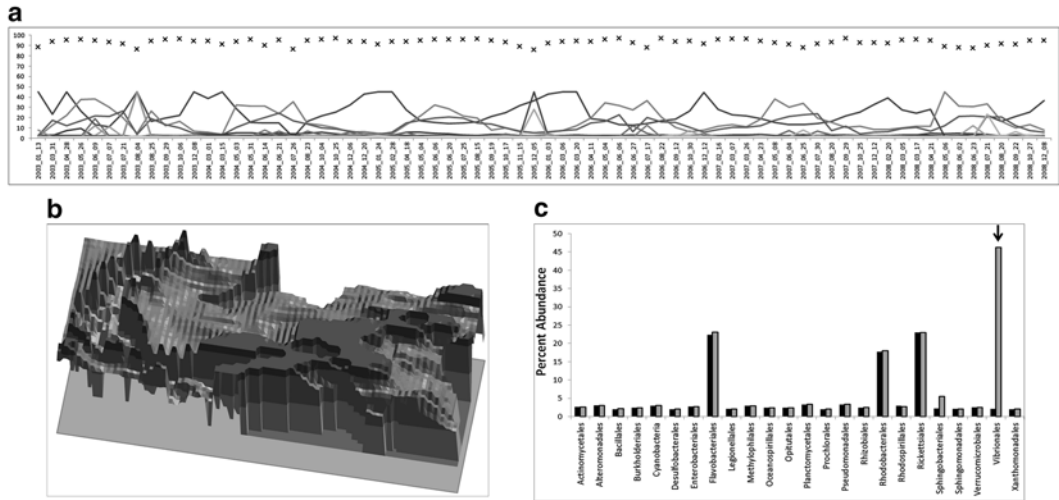


Fig. 2 Examples of MAP model applications. MAP models can be used to investigate microbial populations at a variety of temporal and spatial scales. The included examples drawn from MAP model and data originally presented in Larsen et al. [14]. (a) In a MAP model *extrapolated across time*, marine microbial populations are predicted for a single location over a total of 6 years. For every day in this model, environmental parameters at a single location in the Western English Channel were inferred from satellite imagery and used as input to a MAP model. In this figure, *gray lines* indicate changing relative abundance of individual taxa. Black “X”s indicate the Bray-Curtis similarity score between the predicted and observed population structure at every time point. (b) In this MAP model *extrapolated over space*, MAP model generated from data collected at a single point is used to extrapolate the population structure over a wide geographical area using satellite image modeling data as environmental parameters. In this figure, height of surface is proportional to relative abundance of the taxa *Rickettsiales* in the surface waters of the Western English Channel on March 17, 2008. The geographical region pictured is between 48.5°N, 7.6°W and 50.8°N, 1.5°W. (c) In this figure, MAP model is used to *extrapolate over multiple environmental conditions*. Using a MAP model generated using satellite image data collected between 2007 and 2008, an initial microbial population structure (*black bars*) was generated for August 20th, 2008. In the figure, y-axis is percent abundance of total population. X-axis is the set of 24 bacterial taxa in this MAP model. Starting with these observed environmental parameters, environmental parameters were changed in silico until a “bloom” of *Vibrionales* was observed in microbial population without greatly altering the abundance of all other microbial taxa (*gray bars*, *Vibrionales* abundance highlighted with arrow). Strikingly, the conditions required to generate an August *Vibrionales* bloom in MAP model (i.e., lower soluble phosphorus, increase in chlorophyll A concentration, and moderate levels of dissolved organic carbon) are nearly identical to environmental parameters associated with a *Vibrionales* bloom in August 2003

selected by using the highest taxonomic level for which the majority of observations in all populations, the observed abundance of a taxa is nonzero.

Each microbial population structure observation must be accompanied by a set of environmental parameters. The nature of this data may vary widely depending on the nature of the environment being sampled and the scale of the desired MAP model but may include chemical and physical attributes collected from in situ observations, data collected from automated sensors deployed in the environment, or remotely sensed data such as by satellite image.

2.2 Required Computational Tools

Some computational tools are required to generate MAP models, and specific tools are recommended here for use in construction of MAP models (*see Note 2*).

For generating EIN, the Bayesian Network Inference tool “Banjo” [17] is recommended. Banjo is a software application and framework for structure learning of static and dynamic Bayesian networks using a score-based structure inference. Banjo is freely available under a noncommercial use license (<http://www.cs.duke.edu/~amink/software/banjo/>).

For generating the equations in the ANN from the topology of the EIN, Eureqa is recommended. “Eureqa Formulize” is a software package available from Nutonian Inc. (<http://www.nutonian.com/>), which is freely available for academic research (<http://www.nutonian.com/products/eureqa/>). As described by Eureqa documentation, this analysis tool searches the space of mathematical equations using symbolic regression to identify a mathematical model that best fits the data provided. Starting with initial expressions that randomly couple mathematical building blocks, both the form and parameters of possible equations are generated by recombining previous equations and varying their sub-expressions using a statistics-based evolutionary algorithm. The resulting models are ranked using a ratio of equation accuracy and complexity.

3 Methods

There are two steps for construction of the MAP model: generation of EIN and fitting equations to ANN. Before beginning, the available data should be divided into training and validation subsets. The training subset is used to construct the MAP model and then verify the accuracy of the MAP model with the validation subset.

3.1 Normalize Data from Biological Observations

Environmental parameter data should be normalized to a uniform dynamic range, for example, to arbitrary units ranging from 0 to 100 (Fig. 3) (*see Note 3*).

Typical microbial population data is made up of a relatively small number of species comprising the majority of the population and a long tail of numerous low-abundance species comprising the remainder of the population [18]. To best normalize the data distribution for MAP model, first report taxonomic abundances as a percentage of total population. This normalized data is then \log_2 transformed (Fig. 4).

3.2 Generate EIN from Normalized Data

The following steps are specific to Banjo software (*see Note 4*). Before generating BN network in Banjo, a file of unallowed edges must be made (“edgesNotAllowed” parameter in Banjo settings file). This parameter must be set such that environmental parameter nodes

		Environmental Parameters		
		1	2	3
Observations	1	0.032	320	3.5
	2	0.126	169	4.6
	3	0.504	114	1.9
	4	0.027	235	2.8

↓

		Normalized Environmental Parameters		
		1	2	3
Observations	1	20.6	80.0	55.6
	2	32.5	36.0	80.0
	3	80.0	20.0	20.0
	4	20.0	55.2	40.0

Fig. 3 Example normalization of environmental parameters. This very small, hypothetical sample dataset contains three environmental parameters and four observations. Raw data for collected environmental parameters (*top panel*) can be in different units and cover very different dynamic ranges. Before using in MAP models, data must be normalized to uniform ranges in arbitrary units. In this example (*bottom panel*), data is normalized such that the lowest observed value for an environmental parameter is equal to 20 and the highest is equal to 80

have no possible parent nodes. Additional parameters in Banjo, such as maximum number of parents, searcher, proposer, and run duration, depend entirely on the nature of data being considered and must be determined empirically. Investigators should refer to the Banjo user manual for specific guidelines in selecting search parameters appropriate for a particular dataset.

The output of Banjo is the network that best fits the microbial population data. The final network is used to generate the equations in the subsequent step.

3.3 Generate ANN from EIN

The same, normalized dataset used to generate the EIN is used to generate equations for the MAP model ANN. Specifics for running Eureqa on particular platforms are available from the Nutonian Eureqa website (<http://www.nutonian.com/products/eureqa/>). From the output of Banjo EIN, network topology in the format of a list of parent nodes for each child node is converted into Eureqa. The operations “constant,” “add,” “subtract,” “multiply,” “divide,” and “exponential” are recommended. As Eureqa outputs several

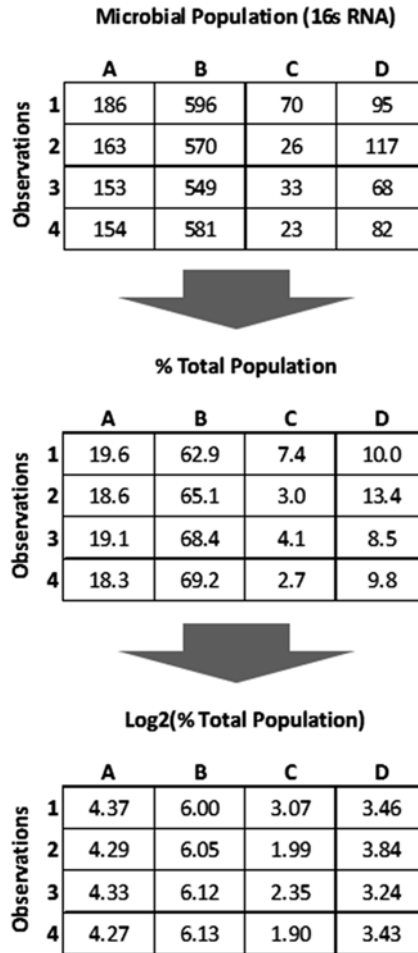


Fig. 4 Example normalization of microbial population data. In this small, hypothetical microbial population dataset, there are four microbial taxa (a–d) and four observations. As in the *top panel*, the initial data will be in the format of number of 16s RNA sequences associated with each taxa. This data is transformed (*middle panel*) to representation as percent of total population. Percent total population data is log transformed before use in MAP analysis (*bottom panel*)

possible functions to fit observed data, the equation selected for use in MAP was selected to be the most parsimonious one as defined by the following, as many as possible, optimization criteria:

- Smallest equation size.
- Highest correlation with observed data.
- If there is a choice between an equation with an exponential term and one without, choose the equation without an exponential term.
- If there is an obvious peak or drop in observed relative bacterial abundance, choose the equation that best models that peak

or drop, even if it means selecting an equation with a greater size or lower overall correlation.

Eureqa's equation search is "done" when the stability metric is close to 100 %. The time it takes to achieve stable solutions will depend on the nature of the data and on the speed of the computer running Eureqa. It is important to note that even if a node is predicted to have more than one parent in the EIN, not all of those parents will necessarily be incorporated into the Eureqa-generated equations that best fit the observed data.

3.4 Implement MAP Model

The results of the previous step are a set of linked mathematical equations in which the results of some equations will be the terms in other equations. Taken together, given a single set of normalized environmental parameter data, values for all taxonomic relative abundances can be calculated with a unique solution. Implementation of this model can be easily accomplished in a number of mathematical programming platforms (e.g., R-Project (<http://www.r-project.org/>) or MatLab (<http://www.mathworks.com/products/matlab/>)) or common spreadsheet applications.

3.5 Validate MAP model

Once the MAP model has been generated, the model must be validated. Similarity between predicted population structures in the validation subset can be calculated by a variety of means, such as Pearson's correlation coefficient, product moment correlation coefficient, or Bray-Curtis similarity score [19]. The specific similarity metric must be chosen depending on the distribution of the data used to generate the model. Due to the typical distribution of microbial population data, in which a few taxa are consistently of high relative abundance and many taxa are consistently of low relative abundance [18], it is possible to get deceptively high correlations between predicted and observed population structures so long as the model correctly identifies the small number of high-abundance taxa. To correct for this, there are two null models that can be easily tested. The first null model is to set all taxa's predicted relative abundance equal to the average taxa abundance across all samples. The other null model is to set all taxa abundances equal to the minimum observed taxa's abundance. If either of these null models has a better correlation with biological observations than the MAP model, it is likely that the MAP model is not identifying useful relationships in the data (*see Note 5*).

To assign a level of statistical significance to the MAP model results, a bootstrap-style approach is recommended (*see Note 6*). In a bootstrap-like method, the model data is randomly permuted a large number of times. The similarity of observed data to permuted data is calculated each interaction. A significance of the model can be reported as:

$$\frac{\# \text{ iterations in which random permutation similarity} > \text{ model similarity}}{\text{Total \# of iterations}}$$

This permutation approach can be accomplished in a number of mathematical programming tools or in common spreadsheet applications.

Identifying a MAP model that both accurately predicts observed microbial population structures and is likely to provide useful biological insight is a process that may take several iterations of model building and testing.

4 Notes

1. As with any biological experiment or computational model, there needs to be a sufficient number of training data observations to draw meaningful biological conclusion from MAP model. The exact number of observations required to achieve results with statistical significance or biological relevance will depend entirely on the nature of the data collected and the complexity of the biological system being studied and must be empirically determined.
2. Other computational tools that perform the same or similar functions could also be incorporated, depending on availability and preference of individual investigators.
3. If using a minimum of 0 and a maximum of 100 for environmental parameter data in MAP model, it may be useful to normalize observed data to values within that range, for example, a minimum of 20 and maximum of 80. This allows the possibility of modeling environmental parameters slightly outside the range of actual observations without running into the potential computational errors that might be caused by using input values equal to or less than zero.
4. If an alternate program for BN inference is used, then follow the documentation for that software.
5. If the resulting MAP model does not yield satisfactory results, it is necessary to reconsider the parameters used to build the model. The following are common steps to consider when troubleshooting a MAP model:

Generation of EIN

- Data discretization: Banjo requires discretized data for learning networks. It is possible that discretization could mask important variability in some microbial taxonomic abundance values. Consider a higher discretization index or using quantile discretization in Banjo.
- Rare taxa present in data: Some taxa may be undetected in many samples. Rarely observed taxa tend to be over-fitted in the network. Consider removing these rare taxa from the analysis or selecting a lower taxonomic resolution.

- Very sparse networks: Increasing the number of allowed parents in Banjo requires more computer memory but may generate networks with higher connectivity.
- Lack of interaction between environmental parameters and taxa abundance nodes: It is possible that Banjo will not identify many direct interactions between environmental parameters and microbial taxa abundances. It may be useful to log transform environmental parameter data or select an alternate discretization policy in Banjo. If no statistically significant dependencies between environmental parameters and microbial taxa abundances can be found in Banjo, it may be necessary to resort to an alternate method, such as strong correlation coefficients between environmental parameters and microbial taxa abundance to identify potential edges in EIN.

Generation of MAP ANN

- Over-fitting of equations by “Eureqa”: It may be that the training set correlates well with biological observations, but the validation set correlated poorly. From the set of Eureqa-proposed equations, reselect an equation with lower complexity, even if the equation has lower correlation with training data.
 - Unsolvable equations in MAP model: Some combinations of data may cause Eureqa-generated equations to return errors, such as attempting to divide by zero. Reselect any equations for which a division-by-zero error may occur or consider removing rare taxa from the model for which relative abundance is sometimes or often equal to zero.
 - Biologically meaningless results: It is possible for MAP model to predict biologically meaningless results such as negative taxonomic abundances or taxa that comprise more than 100 % of the total population. To prevent this, avoid selecting Eureqa equations with very large constants or exponential terms. Also, it is often helpful to set upper and lower limits on equation results, setting values less than zero equal to zero (or an empirically determined very small number to avoid possible division-by-zero errors) and establishing an upper limit on possible abundances (e.g., setting an upper limit equal to a value 20 % larger than the maximum relative abundance of the most abundant taxa actually observed in the biological dataset).
6. There are many ways to permute the data, and the most appropriate way to randomize the data will depend on the nature of the data and the experimental design. One common approach to randomization is to randomly reassign predicted abundances within the same taxa and across observations.

References

1. Hugenholtz P, Goebel BM, Pace NR (1998) Impact of culture-independent studies on the emerging phylogenetic view of bacterial diversity. *J Bacteriol* 180:4765–4774
2. Barns SM, Takala SL, Kuske CR (1999) Wide distribution and diversity of members of the bacterial kingdom Acidobacterium in the environment. *Appl Environ Microbiol* 65:1731–1737
3. Shtarkman YM et al (2013) Subglacial Lake Vostok (Antarctica) accretion ice contains a diverse set of sequences from aquatic, marine and sediment-inhabiting bacteria and eukarya. *PLoS One* 8(7):e67221
4. Fierer N et al (2008) Short-term temporal variability in airborne bacterial and fungal populations. (Translated from eng). *Appl Environ Microbiol* 74(1):200–207
5. Bowers RM et al (2009) Characterization of airborne microbial communities at a high-elevation site and their potential to act as atmospheric ice nuclei. (Translated from eng). *Appl Environ Microbiol* 75(15):5121–5130
6. Takai K et al (2001) Archaeal diversity in waters from deep South African gold mines. *Appl Environ Microbiol* 67(12):5750–5760
7. Edwards RA et al (2006) Using pyrosequencing to shed light on deep mine microbial ecology. *BMC Genomics* 7:57
8. Teske A, Sorensen KB (2008) Uncultured archaea in deep marine subsurface sediments: have we caught them all? *ISME J* 2(1):3–18
9. Baune C, Bottcher ME (2010) Experimental investigation of sulphur isotope partitioning during outgassing of hydrogen sulphide from diluted aqueous solutions and seawater. *Isotopes Environ Health Stud* 46(4):444–453
10. Bardgett RD, Freeman C, Ostle NJ (2008) Microbial contributions to climate change through carbon cycle feedbacks. *ISME J* 2(8):805–814
11. Thomas T, Gilbert J, Meyer F (2012) Metagenomics - a guide from sampling to data analysis. *Microb Inform Exp* 2(1):3
12. Larsen PE, Gibbons SM, Gilbert JA (2012) Modeling microbial community structure and functional diversity across time and space. *FEMS Microbiol Lett* 332(2):91–98
13. Larsen P, Hamada Y, Gilbert J (2012) Modeling microbial communities: current, developing, and future technologies for predicting microbial community interaction. *J Biotechnol* 160(1–2):17–24
14. Larsen PE, Field D, Gilbert JA (2012) Predicting bacterial community assemblages using an artificial neural network approach. *Nat Methods* 9(6):621–625
15. Caporaso JG et al (2010) QIIME allows analysis of high-throughput community sequencing data. *Nat Methods* 7(5):335–336
16. Meyer F et al (2008) The metagenomics RAST server—a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinform* 9:386
17. Smith VA et al (2006) Computational inference of neural information flow networks. *PLoS Comput Biol* 2(11):e161
18. Sogin ML et al (2006) Microbial diversity in the deep sea and the underexplored “rare biosphere”. *Proc Natl Acad Sci U S A* 103(32):12115–12120
19. Mumby PJ, Clarke KR, Harborne AR (1996) Weighting species abundance estimates for marine resource assessment. *Aquat Conserv* 6(3):115–120

A General ANN-Based Multitasking Model for the Discovery of Potent and Safer Antibacterial Agents

A. Speck-Planche and M.N.D.S. Cordeiro

Abstract

Bacteria have been one of the world's most dangerous and deadliest pathogens for mankind, nowadays giving rise to significant public health concerns. Given the prevalence of these microbial pathogens and their increasing resistance to existing antibiotics, there is a pressing need for new antibacterial drugs. However, development of a successful drug is a complex, costly, and time-consuming process. Quantitative Structure-Activity Relationships (QSAR)-based approaches are valuable tools for shortening the time of lead compound identification but also for focusing and limiting time-costly synthetic activities and in vitro/ vivo evaluations. QSAR-based approaches, supported by powerful statistical techniques such as artificial neural networks (ANNs), have evolved to the point of integrating dissimilar types of chemical and biological data. This chapter reports an overview of the current research and potential applications of QSAR modeling tools toward the rational design of more efficient antibacterial agents. Particular emphasis is given to the setup of multitasking models along with ANNs aimed at jointly predicting different antibacterial activities and safety profiles of drugs/chemicals under diverse experimental conditions.

Key words Antibacterial activity, Drug resistance, QSAR, Topological indices, Ontology, Moving average approach, Artificial neural networks, mtk-QSBER models

1 Introduction

For centuries, mankind has been particularly affected by microbial diseases, those caused by bacteria being among the most dangerous and lethal. Even though antibiotics have saved millions of lives and alleviated the suffering of people for many years [1], drug-resistant, disease-causing bacteria have emerged and spread so much in recent decades [2–5] that they have created a global public health problem. This situation is so serious that nowadays bacteria cause high mortality rates not only in communities but also in healthcare environments and facilities [6]. Given the prevalence of the major gram-positive pathogens, especially those belonging to the genera *Staphylococcus*, *Streptococcus*, and *Enterococcus*, transferability of resistance genes is of special concern. This is exemplified

by the ability of enterococci to interact with methicillin-resistant *S. aureus* (MRSA), transferring to the latter vancomycin-resistant genes [7]. Thus, MRSA can become resistant to vancomycin, which is the unique antibacterial chemotherapeutic alternative. On the other hand, gram-negative pathogens such as *Escherichia coli* and *Pseudomonas aeruginosa* exhibit intrinsic resistance to almost any antibiotic [8]. The most deadly bacterial disease has been tuberculosis (TB), which is prone also to multidrug resistance. Indeed, more than 9 million new TB cases and about 1.7 million deaths were reported in 2009 [9], while a recent report confirmed that in 2010, 5.7 million official cases of TB were notified, with 8.8 million incidents worldwide (equivalent to 128 cases *per* 100,000 population) [10].

Diseases and infections caused by bacteria are very complex and depend on multiple pathogenic and epidemiological factors [6, 8]. For this reason, the search for new antibacterial drugs is very urgent. This battle against bacterial diseases/infections will depend on the efficiency of chemotherapies and have a profound influence on human health.

High-throughput screening technologies [11] along with combinatorial chemistry [12] were expected to solve the drug discovery problem by a massive parallelization of the process. In practice, however, while the number of identified hits can substantially be increased using these methods, no corresponding growth in the number of drugs entering the market has been observed [13]. This fact has progressively led to a reconsideration and rationalization of the drug discovery process, in which chemoinformatics methods, led by Quantitative Structure-Activity Relationships (QSAR) tools [14], have gained a role of tremendous importance [15–17]. To be clearly effective, these methods should aim at targeting both pharmacological profiles and desirable ADMET (absorption, distribution, metabolism, elimination, toxicity) properties, as the latter are the major causes of non-approval of drugs [18–20]. Particularly, in recent years, promising multi-target (mt-QSAR) models have been reported, revolutionizing concepts regarding QSAR paradigm prediction [21–26]. These mt-QSAR models have been applied to simultaneously predict several features of biological activity by considering different biological targets such as biomolecules, cell lines, tissues, and organisms. The aforementioned mt-QSAR models have evolved to the point that nowadays it is possible to predict multiple biological profiles by considering different measures of the effects, many biological targets, and diverse levels of curation of the experimental information [27–32]. These new models, known as multitasking Quantitative Structure-Biological Effect Relationships (mtk-QSBER), have often employed classification techniques such as linear discriminant analysis (LDA) [33]. But sometimes, due to the complexity of the data and a lack of accuracy in the experiments, linear classification tools do not lead to models

with enough statistical quality and predictivity. In such cases, artificial neural networks (ANNs) can instead be applied to deal with nonlinear behavior and usually do enhance the performance of the predictions [22, 29]. This chapter is devoted to an overview of the setup and use of mtk-QSBER models based on ANNs. Overall, specific insights from our perspective and personal experience will be provided.

2 General Procedure for the Setup of QSAR Models

Generally speaking, QSAR approaches seek to uncover possible relationships between chemical structures (*molecular descriptors*) and one or more endpoints of interest (e.g., biological activity, toxicity, or pharmacokinetic profiles) [14]. Several steps should be followed for establishing the QSAR models (Fig. 1), and these, if well devised, may help yielding more accurate results. Firstly, the data must be retrieved typically from a public source and subsequently curated in, for example, a spreadsheet application like Excel. Secondly, molecular descriptors are calculated from the molecular structures; finally, by applying a statistical modeling approach (either linear or nonlinear), the best QSAR model can be obtained.

2.1 Databases and Handling of the Retrieved Data

Databases are typically organized as public sources and contain large collections of data describing the most relevant aspects of phenomena and/or experiments [34, 35]. A well-known database is ChEMBL [34]—an online free and dynamic source available at

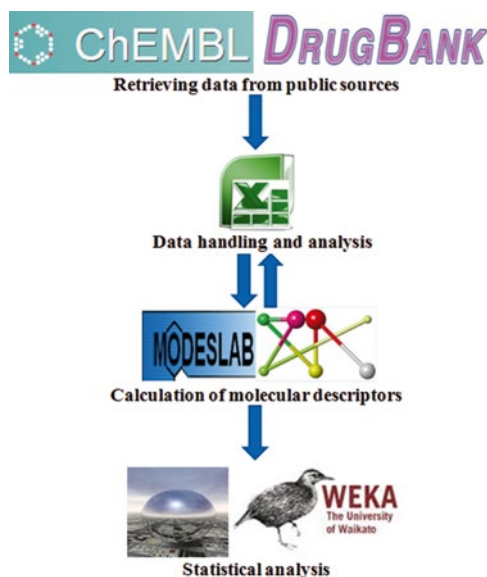


Fig. 1 Necessary steps required for building a QSAR model

<https://www.ebi.ac.uk/chembl/>, which contains more than 12 million assay outcomes. These biological assays derive from applying more than 1.3 million drugs/chemicals against at least one out of more than 9,300 biological targets (proteins, cell lines, microorganisms, mammals), and it is possible to obtain specific information about the diverse strains or breeds in the case of non-molecular biological entities. There is also information related to the type of the assay, i.e., if a test was carried out to measure binding (B), functional (F), or ADMET (A) properties. All data can be easily downloaded into Excel-like files. Most importantly, the ChEMBL database provides good transparency concerning the accuracy and/or reliability of the experimental information. For that, two columns of ChEMBL are of particular interest, namely, (1) a “CONFIDENCE SCORE” that contains values ranging from 4 to 9 for biomacromolecular targets, the highest number being given to a very accurately determined experiment, and (2) a “CURATED BY” that shows how much the experimental information has been verified, depending on the availability of data in the literature and certain details associated to the assay protocols, with three categories: autocuration (the lowest reliability), intermediate, and expert (highest reliability). In addition, in this database, each compound has one ChEMBL identifier and SMILES (Simplified Molecular Input Line Entry Specification, i.e., the 2D chemical structure) code. Therefore, ChEMBL is an interesting and potentially valuable resource for tackling drug discovery. In fact, it has been stated that mining ChEMBL is an excellent alternative to generate models for virtual screening of large libraries of drugs and compounds [36], allowing a greater coverage of the chemical space. With these considerations in mind, ChEMBL should be the database of first choice.

A further very important database is DrugBank, which is available at <http://www.drugbank.ca/>. This is a unique free source chemo-bioinformatics database that combines specific drug information (chemical structure, pharmacological, and therapeutic data) with details from the respective target(s) (sequences, structures, and pathways) [35]. DrugBank comprises 6,825 drug entries including 1,541 small-molecule drugs approved by the FDA (Food and Drug Administration) and other molecular entities such as 86 nutraceuticals, 5,082 experimental chemicals, and 150 FDA-approved biotech (protein/peptide) drugs. DrugBank is a suitable source of raw data for deep studies of drug-target and drug-drug interactions. QSAR modeling using this database can facilitate the discovery of new targets for antibacterial drugs. At the same time, studies focused on drug-drug interactions can be performed in advanced stages of the drug discovery process to analyze, e.g., possible chemicals with which an antibacterial drug should not be administrated. In addition, ChEMBL and DrugBank offer the possibility of downloading substantial volumes of data in the form of tabular-based files which can be easily opened, modified, and filtered/curated using Excel.

2.2 Molecular Descriptors and Their Applicability in QSAR Approaches

Molecular descriptors are essential hot spots of any QSAR study. As pointed up nicely some years ago by Todeschini and Consonni [37]: “The molecular descriptor is the final result of a logic and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardised experiment.” The use of certain types of molecular descriptors depends on the kind of QSAR approach that is going to be employed. Molecular descriptors used in classical QSAR approaches may be experimentally determined (e.g., physicochemical properties like the Hammett σ constant, refractivity, etc.) via Hansch analysis or through the use of simple indicator descriptors (R-group substitutions of the parent core), Free-Wilson analysis [38]. 3D-QSAR approaches have emerged as a natural extension to the classical QSAR approaches pioneered by Hansch and Free-Wilson and are based on correlating the activity with non-covalent molecular interaction fields [38]. These approaches require mandatorily 3D structures, e.g., based on protein crystallography or molecule superimposition, and assume that all molecules interact with the bio-target in the same manner, i.e., with an identical binding mode. In 3D-QSAR approaches, pair potentials (e.g., Lennard-Jones) are calculated for the whole molecule to model the effects of shape and size (steric fields), the presence of hydrophobic/hydrophilic regions, and the behavior of the electronic density in different parts of the molecule (electrostatic fields). *CoMEA* (*Comparative Molecular Field Analysis*) and *CoMSIA* (*Comparative Molecular Similarity Index Analysis*) [39] are the most applied 3D-QSAR approaches, in particular, to support docking calculations. Over time, 3D-QSAR approaches have evolved to improved variants such as 4D-QSAR, additionally including ensemble of ligand conformations and protonation states in 3D-QSAR [40]; 5D-QSAR, adding induced-fit effects of the adaptation of the receptor-binding pocket to the ligand in 4D-QSAR [41]; and 6D-QSAR approaches, further incorporating different solvation models in 5D-QSAR [42].

As an alternative to 3D-QSAR approaches, the structures of the molecules can be optimized without any superimposition of alignment. Toward that end, calculations based on, e.g., density functional theory (DFT) and molecular dynamics (MD) are carried out [43], and both local (charge, polarizability HOMO-/LUMO-based parameters) and global descriptors (different energies, volumes, and areas) can be determined and used to obtain correlations with biological activities. Further, highly accurate quantum-mechanics calculations allow the study of more specific interactions such as the strength of hydrogen interactions, stability of coordination bonds, etc.

Several works using 3D-QSAR approaches or QM calculations have been reported in the field of antimicrobial research [44–48].

But from our point of view, QSAR research should be focused on developing models best suited for the analysis of a large number of diverse compounds and virtual screening of databases. For that purpose, 3D-QSARs and related approaches have several important drawbacks, namely:

- Although improved approaches such as 4D-, 5D-, and 6D-QSARs have emerged, uncertainties regarding the selection of the active conformations and the binding mode of ligands remain.
- 3D-QSARs are usually used to study the inhibitory activity of compounds related to *in vitro* data. For the successful application of these approaches to *in vivo* data, there must be a thorough understanding of the binding mechanisms of all underlying ligands.
- For 3D-QSAR approaches, optimization of the 3D structures of ligands can be particularly time consuming. Moreover, the lowest energy conformation of any ligand is usually considered to be the bioactive conformation and responsible for exerting the binding effects.
- In the case of QM-based models, the computational cost and required time to perform calculations can be very high, depending on the complexity of the molecules to be modeled, the QM method to be applied (ranging from molecular mechanics to semiempirical and DFT methods, including higher-level QM methods), and the availability of computational resources.
- If QM calculations are performed without alignment, there will be more uncertainty than in 3D-QSAR approaches for the selection of the active ligand conformation.

These handicaps can somehow be solved if 2D topological descriptors are applied [37]. More than 100 families of topological descriptors have been described in the literature [49]. They are among the most useful sets of molecular descriptors, as corroborated by the large number of QSAR applications reported to date [50–60]. However, these descriptors have been criticized for a lack of clarity regarding their physicochemical meaning and structural interpretation [38]. Definitely, these descriptors are unable to provide a complete understanding of the 3D structural aspects of molecules, which can be considered as a possible disadvantage if the analysis of the binding mode of a ligand with its bio-target is needed. But as proven by Estrada, although topological descriptors are derived from 2D representations of the molecular structures, connectivity indices, for example, encode information related to the molecular accessibility for the surrounding medium. For instance, first- and second-order connectivity indices represent molecular accessibility areas and volumes, respectively, while

higher-order connectivity indices represent hyper-volumes [61]. The same author has also demonstrated that 2D topological descriptors can account for “pure” 3D structural parameters, such as the dihedral angle between phenyl rings in alkylbiphenyls [60].

At this point, it is easy to conclude that topological descriptors do not depend on the conformation and thus do not need the superimposition rules and alignments used in 3D-QSAR approaches. Further, topological descriptors can be used in QSAR studies involving both in vitro and in vivo assay data. When these descriptors are used to generate QSAR models based on statistical classification techniques like LDA, the databases can be formed by compounds belonging to dissimilar chemical families, because the mechanisms of action are not so important. These descriptors will afford the structural patterns to successfully separate the dataset of compounds into groups having the observed biological activity or not [21–32, 54, 59]. Topological descriptors have exhibited a great applicability for virtual screening of antibacterial agents [62–66]. Finally, extremely interesting kinds of topological descriptors are the graph-based spectral moments, designed according to the *TOPS-MODE* (*TOP*ological *S*ubstructural *MO*lecular *DE*sign) approach [67–69]. The greatest advantage of the *TOPS-MODE* approach, over other traditional QSAR methods, stems from its substructural nature. This means that one can transform the resulting QSAR model into a bond additive scheme and thus describe the endpoint activity as a sum of bond contributions related to different structural fragments of the molecules. Moreover, one can detect the fragments on a given molecule that contribute positively or negatively (by summing up bond contributions) to the underlying activity [51].

Many programs have been built for the calculation of molecular descriptors. One of the most widely applied is the MODESLAB software [70] developed by Estrada and Gutierrez for Windows. This software allows the calculation of physicochemical properties (e.g., polar surface area, van der Waals radii, etc.) and classical topological descriptors like atom and bond connectivity indices, as well as other indices [49]. The program also calculates topographical descriptors resulting from mixing topological indices with QM calculation features. But the most powerful set of descriptors implemented in MODESLAB is the spectral moments. The MODESLAB software is freely available through the webpage <http://www.estradalab.org/links/index.html>.

Another computer program which has attracted the attention of QSAR practitioners is PaDEL-Descriptor, devised by Wei [71]. The current version of this Java-based software calculates 905 descriptors (770 1D and 2D descriptors, as well as 135 3D descriptors) and ten types of fingerprints. These are calculated principally using the well-known Chemistry Development Kit. PaDEL-Descriptor is a free open-source software, which can work on all

major platforms (Windows, Linux, MacOS), supporting more than 90 dissimilar file formats to encode chemical information of the molecules. This software can be downloaded from <http://padel.nus.edu.sg/software/padeldescriptor/>.

But perhaps the best known program for the calculation of molecular descriptors is DRAGON [72]. In the latest version of this program, 4,885 molecular descriptors can be calculated for small, medium, and even larger molecules. Furthermore, after computing the molecular descriptors for a given dataset, correlations between variables can be analyzed by applying the same software in order to exclude redundancies. At the same time, almost any version of DRAGON allows one to carry out principal component analysis (PCA) for feature selection, in order to identify the variables which best explain the variability of the data. The DRAGON software is available at http://www.taletc.mi.it/products/dragon_description.htm, covering different academic and commercial licenses.

2.3 Modeling Techniques

The last stage pertains to the creation of a statistically significant QSAR model, where the molecular descriptors serve as the independent variables and the desired endpoint response(s) as the dependent variable(s). For that purpose, there are many different modeling techniques available in a variety of software packages. This section will just focus on two of the most commonly applied statistical techniques to generate QSAR models. The first group embraces regression approaches like partial least squares (PLS) [33], fundamentally used in 3D-QSAR studies, and the traditional multiple linear regression (MLR) [38]. The second group comprises classification approaches such as LDA and ANNs.

A very important aspect should be highlighted here, whenever one is retrieving a large volume of data to develop the QSAR model. Usually databases are compilations of biological and chemical data reported in the literature, which have been determined within different laboratories by diverse workers and using several types of assay protocols, consequently making it difficult to estimate the associated experimental errors. As regression techniques try to predict the real response values of the compounds, it is then evident that if one is working with a large and heterogeneous dataset of chemicals, it would be very difficult to find a good predictive QSAR model using such techniques because it is almost impossible to control the uncertainty of the data. The aim of classification techniques is to generate lines, planes, hyperplanes, and/or surfaces able to separate compounds into different groups (e.g., active/inactive). So, one has only to preestablish the cutoff value(s) of the endpoint response(s) under study to then categorize the compounds accordingly. Our personal experience, and looking at diverse published studies in the past, led us to conclude that when classification techniques are employed even for the prediction of diverse

endpoints, there is a greater possibility of obtaining high-quality QSAR models [21–32, 54, 59]. This reflects the fact that when classification techniques are used, there is no need to predict the exact value(s) of the endpoint(s) under study, so problems related to the uncertainty of the data are significantly reduced.

However, as previously commented, sometimes simple modeling techniques like LDA are unable to cope with the complexity of the data, and thus ANNs are required for gathering a deeper knowledge about a particular target phenomenon [22, 29, 31]. Any ANN is an effort to emulate biological intelligent systems (*human brain*) by simulating their structure and/or functional aspects [73]. ANNs entail simple processing units (*neurons*) that are linked by weighted connections (*synapses*), with the output (*axon*) signal transmitted through the neuron's outgoing connection. To the best of our knowledge, three types of ANNs have been widely used (Fig. 2) in drug discovery [22, 29, 31], namely, linear neural networks (LNN), multilayer perceptron (MLP), and radial basis function (RBF) networks. LNN consists of a first entry layer, comprising the input nodes directly associated to the molecular descriptors, and a final output layer—the predicted response. MLP and RBF include additionally a second layer, known as the hidden

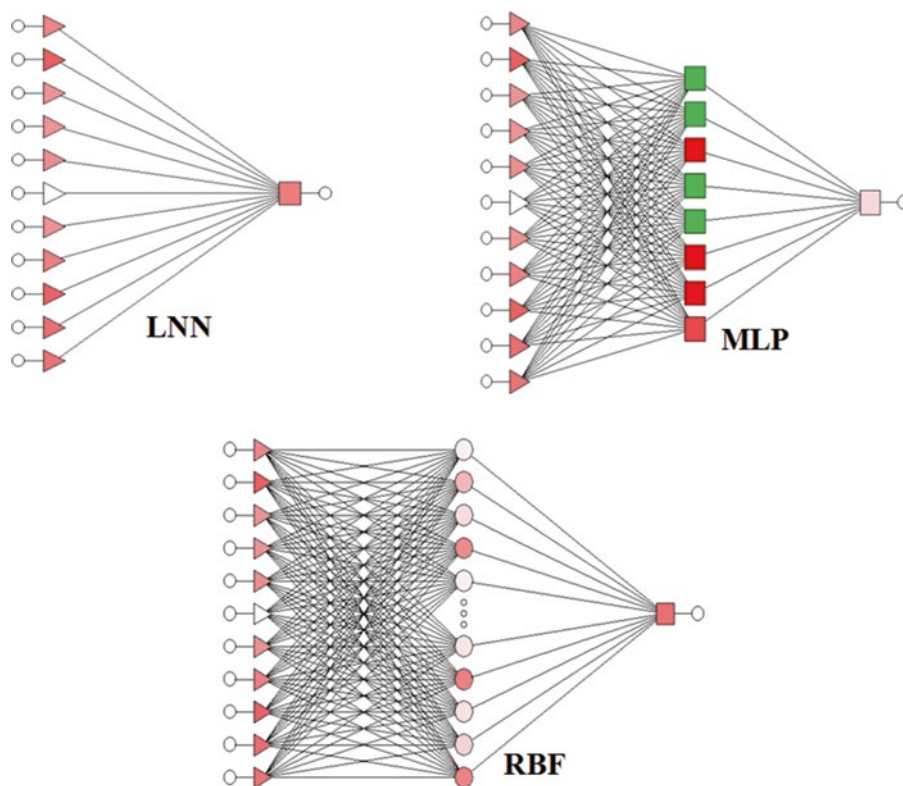


Fig. 2 Different architectures of artificial neural networks

layer, consisting of an array of neurons that receive, transform, and transmit the incoming signals from the first layer. The functions applied on the hidden layer are very important—the so-called activation functions [74], because they model the signals coming from the input layer, as well as the signal from the hidden layer to the output layer. LNN models are very similar to those which can be obtained through LDA, but the algorithms to optimize them are different. LNNs are conceived to establish direct relationships between the inputs (molecular descriptors) and the output, i.e., the endpoint activity usually expressed as a binary variable. MPLs use linear combinations of weights from the input layer to the hidden layer, and the signal from the hidden layer to the output is modeled by a nonlinear activation function (usually a sigmoid function). In RBF networks, nonlinear activation functions (usually Gaussian functions) are applied in the first step, i.e., from the input layer to the hidden layer. Following on, a linear combination of such functions is applied to generate the output layer. Regarding the nomenclature, as shown in Fig. 2, for instance, the second ANN has the form MLP 11:11-8-1:1, meaning that this is an MLP (multilayer perceptron) containing 11 variables (descriptors) which were used as entries to generate 11 input nodes in the first layer, 8 nodes in the second layer, and one node in the third (output) layer from which one response variable (endpoint activity) is to be predicted. Similarly, the first and the last ANNs shown have profiles LNN 11:11-1:1 and RBF 11:11-305-1:1, respectively. Despite the increasing use of ANNs in different areas [22, 29, 31], in the last 15 years, few papers have described the use of ANN-based models to predict antibacterial profiles of compounds [48, 75–77]. In these works, some models were derived with the aim of predicting a reduced series of analogue compounds [48, 75, 77] or heterogeneous but using relatively medium-sized datasets of chemicals [76]. This further illustrates the need to use ANNs with the aim of improving the discovery of desirable antibacterial drugs.

Of the software available for carrying out these modeling techniques, we will mention only two programs because of their applicability, user-friendly nature, and availability. One of the programs widely used in different fields of modern science is STATISTICA [73]. This is a software package developed by StatSoft, which affords data analysis and statistics, as well as data visualization and data mining procedures. The latest version of STATISTICA can handle large amounts of data and is able to tackle various file formats. This software can also generate codes for posterior programming after the QSAR model is built. There are different classification techniques implemented in STATISTICA such as LDA, ANNs, support vector machines (SVM), classification trees (CT), random forest (RF), and many others [73]. Several types of commercial and academic licenses are available for different versions of STATISTICA at <http://www.statsoft.com/>.

In the past, STATISTICA has often been applied to construct QSAR models aimed at predicting antibacterial activity of heterogeneous series of compounds [62–64, 66].

Another freely available program to perform statistical analysis is Weka [78]. This Java program can be viewed as a collection of machine learning algorithms for data mining purposes. The algorithms can either be applied directly to a dataset or accessed from the user's own Java code. Weka contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization and a powerful tool for feature selection [78]. It is also well suited for developing new machine learning schemes. Weka can be downloaded from <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>.

3 An mtk-QSBER Model Combined with ANN for Virtual Screening of Antibacterial Agents

Despite the wide applicability of alignment-free QSAR models based on topological descriptors, an unfavorable aspect of such models is that the antibacterial activity is predicted nonspecifically, meaning that any compound can be predicted to have antibacterial activity but without information regarding the bacteria against which it may be active [76]. What is more, antibacterial activity data should be integrated with other relevant properties such as those related to ADMET profiles to effectively discover reliable antibacterial agents. Of particular interest is that the developed models can potentially guide the screening and discovery of effective antibacterial agents. The next subsections will describe the series of steps involved in building up mtk-QSBER models based on ANNs, which in principle overcome these limitations. Details of the diverse steps for the development of mtk-QSBER-ANN models are depicted in the flowchart of Fig. 3. As a typical example application, we will describe recent work that reported an mtk-QSBER-ANN model whose role was to simultaneously predict the anti-enterococci activity and toxicity in laboratory animals of a set of compounds [27].

3.1 Curation of the ChEMBL Database

In the work referred to above [27], 13,073 endpoints belonging to more than 9,000 chemicals/drugs were retrieved from ChEMBL in an Excel-compatible file format [34]. Chemicals for which data were incomplete were eliminated, as were duplicates (i.e., the same chemicals assayed under the same experimental conditions), leading to a final dataset comprising 8,560 chemicals/drugs and to 10,918 statistical cases, considering the various experimental conditions (c_j). The experimental conditions c_j cover an ontology of the form $c_j \Rightarrow (m_c, b_c, a_c, l_c)$, where m_c describes the different measures of biological effects (anti-enterococci activity or toxicity),

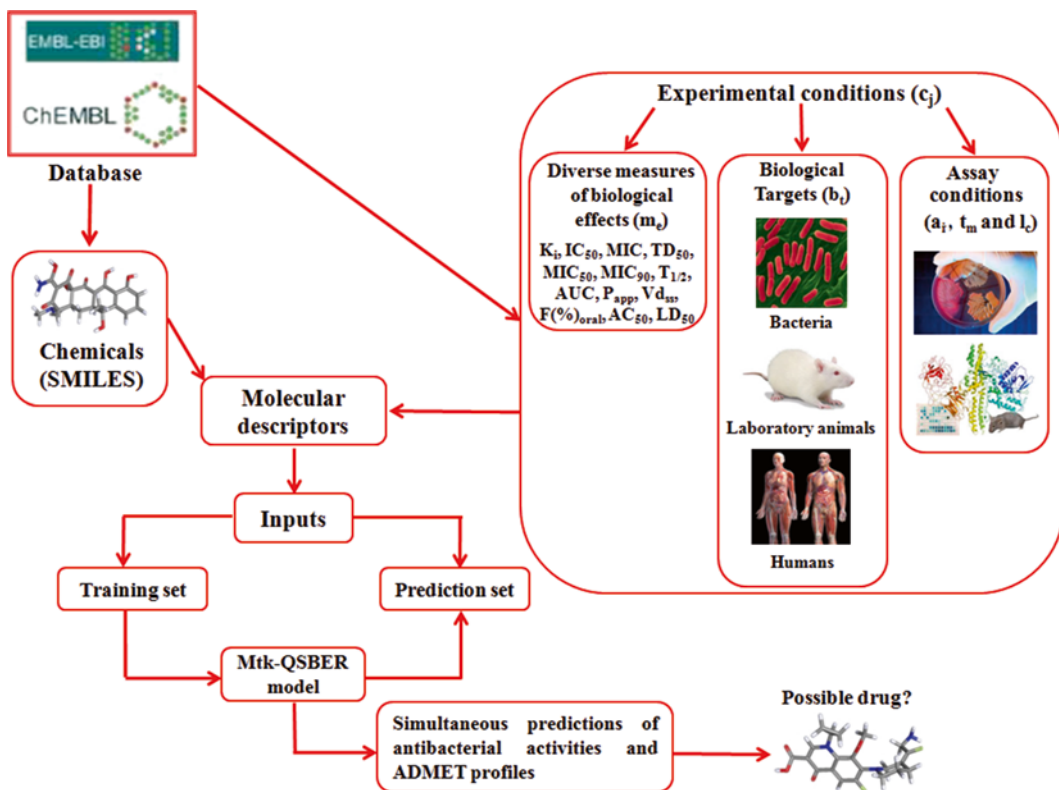


Fig. 3 General steps involved in the development of an mtk-QSBER model

and b_t denotes the dissimilar biological targets such as enterococci (including their respective strains), human immune system cells (lymphocytes), and laboratory animals like mice (*Mus musculus*) and rats (*Rattus norvegicus*). Additionally, a_i refers to the specific assay information, meaning that it provides details whether the assay focused on the assessment of functional (F) or ADMET profiles (A), and l_c describes the degree of curation or verification of the experimental information of a particular assay. Therefore, the elements m_c , b_t , a_i , and l_c embody factors that may be modified to obtain a specific experimental condition.

One should point out also here that the 10,918 cases were the results of using 8,560 chemicals/drugs to assess one out of 18 measures of biological effects, against at least one out of 131 biological targets, by considering at least one out of two labels of assay information, with at least one out of three categories of curation of the experimental information. An important detail to take into account is that the measures of antibacterial activity (e.g., the minimum inhibitory concentration, MIC) were often expressed in different units, such as nM or $\mu\text{g}/\text{mL}$, and something similar applies also to the toxicity measures (e.g., the median lethal dose, LD_{50}). It is clearly essential that all data for a defined measure be in the same

units; for instance, antibacterial activity should be converted to nM or a multiple such as μM . All these conversions can be easily made and are necessary if one is to correctly compare all of the compounds' biological effects (anti-enterococci potency and toxicological profiles). Further, each of the 10,918 cases was assigned to one out of two possible groups related with the biological effect $[\text{BE}_i(c_j)]$ of a defined compound i in a specific experimental condition c_j . That is, a chemical/drug was assigned to the group of positives $[\text{BE}_i(c_j) = +1]$ when its value of biological effect fulfilled certain requirements within a preestablished cutoff [27]; otherwise, the compound was considered as negative $[\text{BE}_i(c_j) = -1]$.

3.2 Moving Average Approach

A useful predictive mtk-QSBER model clearly depends on the molecular structure of the compound and on the experimental conditions/ontology c_j under which the compound was assayed. But if the original descriptors are calculated as previously described (*see* Subheading 2.2), one will not be able to differentiate the biological effects for a given molecule when different elements of the ontology c_j are modified. For this reason, a new set of molecular descriptors was computed by applying the moving average approach [73]:

$$D_i(c_j) = D_i - D_i(c_j)_{\text{avg}} \quad (1)$$

In Eq. 1, D_i is the molecular descriptor of the compound i . The descriptor $D_i(c_j)_{\text{avg}}$ characterizes a defined set of n_j compounds tested under the same experimental conditions c_j . This descriptor is calculated as the average of the D_i values for the compounds in subset n_j that were considered as positive (desirable) cases $[\text{BE}_i(c_j) = +1]$ for the same element of c_j . It should be clarified here that, though we have resorted to an arithmetic mean for computing the $D_i(c_j)_{\text{avg}}$ descriptors, which in fact is one of the best known measures of the central tendency of a list of data, other measures like geometric, harmonic means or standard scores could have been used instead or in addition.

To sum up, $\Delta D_i(c_j)$ are clearly very important descriptors because they encode information based on the chemical structure and the characteristics of c_j . That is, each $\Delta D_i(c_j)$ represents, in structural terms, how much a compound deviates from the group of molecules which were classified as positive.

3.3 Setup of the mtk-QSBER Model

Firstly, the dataset under study (10,918 cases) was randomly divided into two series: training and prediction sets. The training set contained 8,298 cases, 4,217 assigned as positive and 4,081 as negative. The prediction or external set included 2,620 cases, 1,353 positive and 1,267 negative cases.

Then, the original molecular descriptors D_i were calculated—i.e., in this case, MODESLAB spectral moments [70] and those of the type $\Delta D_i(c_j)$ determined by applying Eq. 1. The total final

number of $\Delta D_i(c_j)$ descriptors can be found by multiplying the number of original descriptors by the number of elements of the ontology c_j . In this work, we have thus started with 120 $\Delta D_i(c_j)$ descriptors. As usually happens, this is a large number of molecular descriptors, and so the next task encompasses selecting a proper subset of descriptors to then build the mtk-QSBER model. In this work, we have resorted to program Weka [79], which contains a series of powerful algorithms for variable selection. Notice that the final set of descriptors should yield the best, or at least a very good, discrimination between positive and negative groups.

Two algorithms are usually followed for selecting the variable subset using Weka. In the first, known as the filter algorithm (attribute evaluator), an independent assessment based on the general characteristics of the data is performed. The second algorithm is used in combination with the first one, being focused on the subset evaluation using a defined machine learning algorithm. The latter is called the wrapper (search) algorithm, because a machine learning algorithm is wrapped into the selection procedure. More details about the filter and wrapper algorithms implemented in Weka can be found in Witten et al. [80]. We suggest using at least a minimum of two combinations of filter and wrapper algorithms, “running” the data with such combinations at least five times. In so doing, one is able to easily select the variables common to all such runs, which embrace the most important attributes. In the case under study [27], combinations of the attribute evaluator CFSSubsetEval and the wrapper algorithms called BestFirst and GeneticSearch were employed. Another possibility might be to use the forward stepwise (FS) technique as the variable selection strategy, generally along with LDA. After that, the chosen descriptors can be applied as inputs to derive the best mtk-QSBER model based on ANNs. Our previous analyses and results have shown us that when LDA is used with FS, there is a high probability of then obtaining ANNs of the type RBF. In our opinion, several techniques of variable selection should be combined to provide a solid background on which molecular descriptors do have more influence in the final model.

To find the best mtk-QSBER model based on ANNs, LNN, MLP, and RBF profiles were considered. A fourth ANN called a probabilistic neural network (PNN) was also analyzed. The software STATISTICA was used to carry out this procedure [81]. This program has a package called “Intelligent Problem Solver,” which contains a huge number of tools, and one of them allows the evaluation of the most important variables, by performing a task known as sensitivity analysis. That is, with the latter, only the variables with sensitivity values higher than one are chosen to enter in the final model. “Intelligent Problem Solver” should be run at least five times to check enough different architectures and quality of the resulting ANNs. Another important aspect is to ensure that at

least one descriptor belonging to each element of the ontology c_j is in the final mtk-QSBER model. Moreover, possible correlation between descriptors should be analyzed because, if some of the selected descriptors are redundant, they should be discarded, since the stability of the model as well as its ability to make future predictions might be affected.

In this work, the best model for the simultaneous prediction of anti-enterococci activity and toxicity in laboratory animals was found to be an ANN with an RBF 5:5-767-1:1 profile [27]. As can be judged from the results in Table 1, the RBF ANN had a far better accuracy and predictive power than the LNN—overall accuracy of around 59 %, two MLPs—overall classification ranging in the interval 72–77 %, and PNN—accuracy of around 61 %.

The quality and predictive power of the final mtk-QSBER model were also assessed by checking overall and group-specific performance measures on the training and prediction sets, respectively (Table 2). These included the *sensitivity*, percentage

Table 1
Comparative analysis of the different ANNs exploited in this study

ANN ^a	Training set		Prediction set	
	Sensitivity (%)	Specificity (%)	Sensitivity (%)	Specificity (%)
LNN 5:5-1:1	58.88	58.61	58.17	61.01
MLP 5:5-8-1:1	72.11	72.38	72.28	73.72
MLP 5:5-7-10-1:1	77.35	77.82	77.01	78.06
RBF 5:5-767-1:1	92.98	93.58	90.76	92.11
PNN 5:5-8298-2-2:1	95.04	27.96	94.38	28.02

^aThe first MLP is a three-layer perceptron (TLP), while the second MLP is a four-layer perceptron (FLP)

Table 2
Quality and classification performance of the mtk-QSBER model

Classification ^a	Training set	Prediction set
Sensitivity/TP (%)	92.98	90.76
Specificity/TN (%)	93.58	92.11
Accuracy (%)	93.28	91.41
MCC	0.866	0.828
AUROC	0.981	0.965

^aTP, TN, MCC, and AUROC stand for the true positive, true negative, the Matthews correlation coefficient, and for the area under the ROC curves, respectively

of actual positives that are correctly identified as such or true positive rate; *specificity*, true negative rate; *accuracy*, percentage of correct classification for all cases; the Matthews correlation coefficient (MCC) [82]; and the area under the receiver-operating characteristic (ROC) curve [83]. The latter allows one to confirm that the model is not a random classifier, that is, a model that only correctly predicts half of the cases (with an area under the ROC curve of 0.5).

As can be seen in Table 2, the areas under the ROC curves for the mtk-QSBER model in training and predictions sets were 0.981 and 0.965, respectively, indicating that the model is not a random classifier. Also, the attained MCC values further corroborate the very good quality and performance of the proposed mtk-QSBER model, since for both the training and prediction sets they are near to one. MCC will return values between -1 and $+1$, with $+1$ representing a perfect prediction, 0 a random prediction, and -1 , a total disagreement between observed and predicted biological effects.

3.4 Virtual Screening of Antibacterial Agents

Many scientists argue that QSAR models are not entirely validated as long as no novel compounds are synthesized and biologically tested. Yet a QSAR model is feasibly validated if the external cases (not used to build the model) are correctly predicted. That is the major reason for splitting the datasets into training and prediction series. The critical point here, however, is the chemical space that a derived model can cover. The greater the number of different families of compounds used to build the model, the greater will be the chemical space in which the model can be used prospectively. Besides, a well-validated QSAR model should also show promising results when applied on the virtual screening of chemicals/drugs. The best way to reveal the promising applications of the present model was to predict the multiple biological effects of the antibacterial agent known as BC-3781 [84]. This investigational drug was reported to exhibit high inhibitory activity against different strains belonging to the genus *Enterococcus*. By applying our model, BC-3781 was predicted as a possible antibacterial agent against different enterococci using multiple experimental conditions, in good agreement with the experimental evidence. No toxicological data could be obtained from the literature for this investigational drug, but the mtk-QSBER model led us to infer that BC-3781 should not be potentially harmful for laboratory animals and, in principle, toxicologically safe for humans as well. Notice here that although there is still no experimental data about the toxicity of BC-3781, our toxicity predictions explain why this compound is already undergoing phase II clinical trials and that no significant toxic/adverse effects have been reported so far for humans.

4 Conclusions and Future Perspectives

Nowadays, modern societies are aware of the devastating power of bacterial diseases and infections. The process of creating innovative antibacterial chemotherapies can be accelerated by using chemoinformatics approaches such as QSAR tools supported by powerful statistical techniques like ANNs. In this chapter, we have presented a general overview of the evolution of QSAR models in antibacterial drug discovery. Particular attention was paid to test the ability of a recent ANN-based mtk-QSBER model in the discovery and virtual screening of antibacterial agents. It was shown as well how these types of models are able to participate in dissimilar stages of drug discovery. In our opinion, future perspectives regarding the use of mtk-QSBER models combined with ANNs may be focused on extending them by forward-integrating data of other biological assays against relevant bacteria such as staphylococci, gram-negative pathogens, bacteria causing diseases like pneumonia, or those involved in the appearance and development of nosocomial infections. As a final point, one should remark here that ANNs can be viewed as graphs of interconnected nodes. For this reason, the use of complex network theory may well be useful to analyze more deeply the topology of these machine learning techniques. Perhaps, new horizons could be opened and applied to the field of antibacterial research.

Acknowledgments

A. Speck-Planche acknowledges the Portuguese Fundação para a Ciência e a Tecnologia (FCT) and the European Social Fund for financial support (grant SFRH/BD/77690/2011). This work has been further supported by FCT through grant N^o Pest-C/EQB/LA0006/2011.

References

1. Grayson ML, Crowe SM et al (eds) (2010) Kucers' the use of antibiotics. A clinical review of antibacterial, antifungal, antiparasitic, and antiviral drugs, 6th edn. CRC Press, Taylor & Francis Group, LLC, Boca Raton, FL
2. Shatalin K, Shatalina E et al (2011) H2S: a universal defense against antibiotics in bacteria. *Science* 334:986–990
3. Cordero OX, Wildschutte H et al (2012) Ecological populations of bacteria act as socially cohesive units of antibiotic production and resistance. *Science* 337:1228–1231
4. Rossolini GM, Mantengoli E (2008) Antimicrobial resistance in Europe and its potential impact on empirical therapy. *Clin Microbiol Infect* 14(Suppl 6):2–8
5. Gonzales R, Corbett KK et al (2008) Drug resistant infections in poor countries: a shrinking window of opportunity. *BMJ* 336: 948–949
6. Lautenbach E, Abrutyn E (2009) Healthcare-acquired bacterial infections. In: Brachman PS, Abrutyn E (eds) *Bacterial infections of humans: epidemiology and control*, 4th edn. Springer Science + Business Media, LLC, New York, NY, pp 543–575
7. Rigottier-Gois L, Alberti A et al (2011) Large-scale screening of a targeted *Enterococcus*

- faecalis mutant library identifies envelope fitness factors. *PLoS One* 6:e29023
8. Tenover FC, McGowan JE Jr (2009) The epidemiology of bacterial resistance to antimicrobial agents. In: Brachman PS, Abrutyn E (eds) *Bacterial infections of humans: epidemiology and control*, 4th edn. Springer Science + Business Media, LLC, New York, NY, pp 91–104
 9. Feuerriegel S, Oberhauser B et al (2012) Sequence analysis for detection of first-line drug resistance in *Mycobacterium tuberculosis* strains from a high-incidence setting. *BMC Microbiol* 12:90
 10. Lienhardt C, Glaziou P et al (2012) Global tuberculosis control: lessons learnt and future prospects. *Nat Rev Microbiol* 10:407–416
 11. Hann MM, Oprea TI (2004) Pursuing the leadlikeness concept in pharmaceutical research. *Curr Opin Chem Biol* 8:255–263
 12. Lazo JS, Wipf P (2000) Combinatorial chemistry and contemporary pharmacology. *J Pharmacol Exp Ther* 293:705–709
 13. Bleicher KH, Bohm HJ et al (2003) Hit and lead generation: beyond high-throughput screening. *Nat Rev Drug Discov* 2:369–378
 14. Hansch C, Leo A (1995) *Exploring QSAR: fundamentals and applications in chemistry and biology*. American Chemical Society, Washington, DC
 15. Jorgensen WL (2004) The many roles of computation in drug discovery. *Science* 303:1813–1818
 16. Oprea T (2005) *Chemoinformatics in drug discovery*. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim
 17. Brown N, Lewis RA (2006) Exploiting QSAR methods in lead optimization. *Curr Opin Drug Discov Devel* 9:419–424
 18. Borchardt RT, Kerns EH et al (eds) (2006) *Optimizing the “drug-like” properties of leads in drug discovery*. Springer Science + Business Media, LLC, New York, NY
 19. Croes S, Koop AH et al (2012) Efficacy, nephrotoxicity and ototoxicity of aminoglycosides, mathematically modelled for modelling-supported therapeutic drug monitoring. *Eur J Pharm Sci* 45:90–100
 20. Hau J, Schapiro SJ (2011) *Handbook of laboratory animal science: essential principles and practices*. CRC Press, Taylor & Francis Group, LLC, Boca Raton, FL
 21. Vina D, Uriarte E et al (2009) Alignment-free prediction of a drug-target complex network based on parameters of drug connectivity and protein sequence of receptors. *Mol Pharm* 6:825–835
 22. Prado-Prado FJ, Garcia-Mera X et al (2010) Multi-target spectral moment QSAR versus ANN for antiparasitic drugs against different parasite species. *Bioorg Med Chem* 18:2225–2231
 23. Garcia I, Fall Y et al (2011) First computational chemistry multi-target model for anti-Alzheimer, anti-parasitic, anti-fungi, and anti-bacterial activity of GSK-3 inhibitors in vitro, in vivo, and in different cellular lines. *Mol Divers* 15:561–567
 24. Speck-Planche A, Kleandrova VV et al (2012) Fragment-based approach for the in silico discovery of multi-target insecticides. *Chemometr Intell Lab Syst* 111:39–45
 25. Speck-Planche A, Kleandrova VV et al (2012) In silico discovery and virtual screening of multi-target inhibitors for proteins in *Mycobacterium tuberculosis*. *Comb Chem High Throughput Screen* 15:666–673
 26. Speck-Planche A, Kleandrova VV et al (2012) Chemoinformatics in anti-cancer chemotherapy: multi-target QSAR model for the in silico discovery of anti-breast cancer agents. *Eur J Pharm Sci* 47:273–279
 27. Speck Planche A, Cordeiro MNDS (2013) In Chemoinformatics in drug design. Artificial neural networks for simultaneous prediction of anti-enterococci activities and toxicological profiles. *Proceedings of the 5th International joint conference on computational intelligence, NCTA-International conference on neural computation theory and applications, Vilamoura, Algarve, Portugal, 20–22 Sept*, pp 458–465
 28. Luan F, Cordeiro MNDS et al (2013) TOPS-MODE model of multiplexing neuroprotective effects of drugs and experimental-theoretic study of new 1,3-rasagiline derivatives potentially useful in neurodegenerative diseases. *Bioorg Med Chem* 21:1870–1879
 29. Tenorio-Borroto E, Penuelas Rivas CG et al (2012) ANN multiplexing model of drugs effect on macrophages; theoretical and flow cytometry study on the cytotoxicity of the antimicrobial drug G1 in spleen. *Bioorg Med Chem* 20:6181–6194
 30. Speck-Planche A, Kleandrova VV et al (2013) New insights toward the discovery of antibacterial agents: multi-tasking QSBER model for the simultaneous prediction of anti-tuberculosis activity and toxicological profiles of drugs. *Eur J Pharm Sci* 48:812–818
 31. Speck-Planche A, Kleandrova VV et al (2013) Chemoinformatics for rational discovery of safe antibacterial drugs: simultaneous predictions of biological activity against streptococci

- and toxicological profiles in laboratory animals. *Bioorg Med Chem* 21:2727–2732
32. Speck-Planche A, Cordeiro MNDS (2013) Simultaneous modeling of antimycobacterial activities and ADMET profiles: a chemoinformatic approach to medicinal chemistry. *Curr Top Med Chem* 13:1656–1665
 33. van de Waterbeemd H (1995) *Chemometrics methods in molecular design*. VCH Publishers, Weinheim
 34. Gaulton A, Bellis LJ et al (2012) ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res* 40:D1100–D1107
 35. Knox C, Law V et al (2011) DrugBank 3.0: a comprehensive resource for ‘omics’ research on drugs. *Nucleic Acids Res* 39:D1035–D1041
 36. Mok NY, Brenk R (2011) Mining the ChEMBL database: an efficient chemoinformatics workflow for assembling an ion channel-focused screening library. *J Chem Inf Model* 51: 2449–2454
 37. Todeschini R, Consonni V (2000) *Handbook of molecular descriptors*. WILEY-VCH Verlag GmbH, Weinheim
 38. Kubinyi H (1993) *QSAR: Hansch analysis and related approaches*. VCH Publishers, Weinheim
 39. Kubinyi H, Folkers G et al (eds) (2002) *3D QSAR in drug design: recent advances*. Kluwer Academic Publishers, New York
 40. Klein CD, Hopfinger AJ (1998) Pharmacological activity and membrane interactions of antiarrhythmics: 4D-QSAR/QSPR analysis. *Pharm Res* 15:303–311
 41. Vedani A, Dobler M (2002) 5D-QSAR: the key for simulating induced fit? *J Med Chem* 45:2139–2149
 42. Vedani A, Dobler M et al (2005) Combining protein modeling and 6D-QSAR. Simulating the binding of structurally diverse ligands to the estrogen receptor. *J Med Chem* 48:3700–3703
 43. Carloni P, Alber F (eds) (2003) *Quantum medicinal chemistry*. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim
 44. Li P, Yin J et al (2013) Synthesis, antibacterial activities, and 3D-QSAR of sulfone derivatives containing 1,3,4-oxadiazole moiety. *Chem Biol Drug Des* 82:546–556
 45. Lu X, Lv M et al (2012) Pharmacophore and molecular docking guided 3D-QSAR study of bacterial enoyl-ACP reductase (FabI) inhibitors. *Int J Mol Sci* 13:6620–6638
 46. Uddin R, Lodhi MU et al (2012) Combined pharmacophore and 3D-QSAR study on a series of *Staphylococcus aureus* Sortase A inhibitors. *Chem Biol Drug Des* 80:300–314
 47. Bhonsle JB, Venugopal D et al (2007) Application of 3D-QSAR for identification of descriptors defining bioactivity of antimicrobial peptides. *J Med Chem* 50:6545–6553
 48. Bucinski A et al (2004) Artificial neural networks for prediction of antibacterial activity in series of imidazole derivatives. *Comb Chem High Throughput Screen* 7:327–336
 49. Todeschini R, Consonni V (2009) *Molecular descriptors for chemoinformatics*. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim
 50. Estrada E, Matamala AR (2007) Generalized topological indices. Modeling gas-phase rate coefficients of atmospheric relevance. *J Chem Inf Model* 47:794–804
 51. Estrada E, Uriarte E et al (2000) A novel approach for the virtual screening and rational design of anticancer compounds. *J Med Chem* 43:1975–1985
 52. Roy K, Ghosh G (2004) QSTR with extended topochemical atom indices. 2. Fish toxicity of substituted benzenes. *J Chem Inf Comput Sci* 44:559–567
 53. Roy K, Ghosh G (2008) QSTR with extended topochemical atom indices. 10. Modeling of toxicity of organic chemicals to humans using different chemometric tools. *Chem Biol Drug Des* 72:383–394
 54. Castillo-Garit JA, Vega MC et al (2011) Ligand-based discovery of novel trypanosomicidal drug-like compounds: in silico identification and experimental support. *Eur J Med Chem* 46:3324–3330
 55. Casañola-Martin GM, Marrero-Ponce Y et al (2010) Bond-based 2D quadratic fingerprints in QSAR studies: virtual and in vitro tyrosinase inhibitory activity elucidation. *Chem Biol Drug Des* 76:538–545
 56. Barigye SJ, Marrero-Ponce Y et al (2013) Event-based criteria in GT-STAF information indices: theory, exploratory diversity analysis and QSPR applications. *SAR QSAR Environ Res* 24:3–34
 57. Barigye SJ, Marrero-Ponce Y et al (2013) Relations frequency hypermatrices in mutual, conditional and joint entropy-based information indices. *J Comput Chem* 34:259–274
 58. Vazquez-Prieto S, Gonzalez-Diaz H et al (2013) A QSPR-like model for multilocus genotype networks of *Fasciola hepatica* in Northwest Spain. *J Theor Biol* 343C:16–24
 59. Alonso N, Caamano O et al (2013) Model for high-throughput screening of multi-target drugs in chemical neurosciences; synthesis, assay and theoretic study of rasagiline carbamates. *ACS Chem Neurosci* 4:1393–1403

60. Estrada E, Molina E et al (2001) Can 3D structural parameters be predicted from 2D (topological) molecular descriptors? *J Chem Inf Comput Sci* 41:1015–1021
61. Estrada E (2002) Physicochemical interpretation of molecular connectivity indices. *J Phys Chem A* 106:9085–9091
62. Molina E, Diaz HG et al (2004) Designing antibacterial compounds through a topological substructural approach. *J Chem Inf Comput Sci* 44:515–521
63. Gonzalez-Diaz H, Torres-Gomez LA et al (2005) Markovian chemicals “in silico” design (MARCH-INSIDE), a promising approach for computer-aided molecular design III: 2.5D indices for the discovery of antibacterials. *J Mol Model* 11:116–123
64. Marrero-Ponce Y, Marrero RM et al (2006) Non-stochastic and stochastic linear indices of the molecular pseudograph’s atom-adjacency matrix: a novel approach for computational in silico screening and “rational” selection of new lead antibacterial agents. *J Mol Model* 12:255–271
65. Marrero-Ponce Y, Medina-Marrero R et al (2005) Atom, atom-type, and total nonstochastic and stochastic quadratic fingerprints: a promising approach for modeling of antibacterial activity. *Bioorg Med Chem* 13:2881–2899
66. Speck-Planche A, Scotti MT et al (2009) Design of novel antituberculosis compounds using graph-theoretical and substructural approaches. *Mol Divers* 13:445–458
67. Estrada E (1996) Spectral moments of the edge adjacency matrix in molecular graphs. 1. Definition and applications for the prediction of physical properties of alkanes. *J Chem Inf Comput Sci* 36:844–849
68. Estrada E (1997) Spectral moments of the edge adjacency matrix in molecular graphs. 2. Molecules containing heteroatoms and QSAR applications. *J Chem Inf Comput Sci* 37:320–328
69. Estrada E (1998) Spectral moments of the edge adjacency matrix in molecular graphs. 3. Molecules containing cycles. *J Chem Inf Comput Sci* 38:23–27
70. Estrada E, Gutiérrez Y (2002–2004) MODESLAB. v1.5, Santiago de Compostela
71. Yap CW (2011) PaDEL-descriptor: an open source software to calculate molecular descriptors and fingerprints. *J Comput Chem* 32:1466–1474
72. Todeschini R, Lasagni M et al (1994) New molecular descriptors for 2D and 3D structures. *Theory. J Chemometr* 8:263–272
73. Hill T, Lewicki P (2006) STATISTICS methods and applications. A comprehensive reference for science, industry and data mining. StatSoft, Tulsa
74. Suzuki K (ed) (2011) Artificial neural networks: methodological advances and biomedical applications. InTech, Rijeka
75. Sabet R, Fassihi A et al (2012) Computer-aided design of novel antibacterial 3-hydroxypyridine-4-ones: application of QSAR methods based on the MOLMAP approach. *J Comput Aided Mol Des* 26:349–361
76. Garcia-Domenech R, de Julian-Ortiz JV (1998) Antimicrobial activity characterization in a heterogeneous group of compounds. *J Chem Inf Comput Sci* 38:445–449
77. Lata S, Sharma BK et al (2007) Analysis and prediction of antibacterial peptides. *BMC Bioinformatics* 8:263
78. Hall M, Frank E et al (2009) The WEKA data mining software: an update. *SIGKDD Explor* 11:10–18
79. Hall M, Frank E et al (1999–2013) WEKA. Waikato Environment for Knowledge Analysis. v3.6.9, Hamilton
80. Witten IH, Frank E et al (2011) Data mining: practical machine learning tools and techniques. Morgan Kaufmann Publishers, Elsevier, Amsterdam
81. StatSoft (2001) STATISTICA 6.0. Data analysis software system
82. González-Díaz H, Pérez-Bello A et al (2007) Chemometrics for QSAR with low sequence homology: Mycobacterial promoter sequences recognition with 2D-RNA entropies. *Chemometr Intell Lab Syst* 85:20–26
83. Hanczar B, Hua J et al (2010) Small-sample precision of ROC-related estimates. *Bioinformatics* 26:822–830
84. Sader HS, Biedenbach DJ et al (2012) Antimicrobial activity of the investigational pleuromutilin compound BC-3781 tested against Gram-positive organisms commonly associated with acute bacterial skin and skin structure infections. *Antimicrob Agents Chemother* 56:1619–1623

Use of Artificial Neural Networks in the QSAR Prediction of Physicochemical Properties and Toxicities for REACH Legislation

John C. Dearden and Philip H. Rowe

Abstract

With the introduction of the REACH legislation in the European Union, there is a requirement for property and toxicity data on chemicals produced in or imported into the EU at levels of 1 tonne/year or more. This has meant an increase in the *in silico* prediction of such data. One of the chief predictive approaches is QSAR (quantitative structure–activity relationships), which is widely used in many fields.

A QSAR approach that is increasingly being used is that of artificial neural networks (ANNs), and this chapter discusses its application to the range of physicochemical properties and toxicities required by REACH. ANNs generally outperform the main QSAR approach of multiple linear regression (MLR), although other approaches such as support vector machines sometimes outperform ANNs. Most ANN QSARs reported to date comply with only two of the five OECD Guidelines for the Validation of (Q)SARs.

Key words QSAR, Artificial neural networks, Physicochemical properties, Toxicity, REACH

1 Introduction

The European Union's REACH legislation (concerned with the *Registration, Evaluation, Authorisation and restriction of CHEMicals*) [1] became operational in 2008. From that date, chemicals produced in or imported into the EU at more than 1 tonne/year have to be registered based on the characterization of their toxicities and physicochemical properties. However, complete evaluation of these properties by experimental tests can be time-consuming and expensive. Hence the REACH legislation permits the use of (Q)SARs ((quantitative) structure–activity relationships) for toxicity and property prediction where appropriate.

A QSAR is a mathematical equation that correlates a toxicity or property for a series of compounds with one or more structural properties of the compounds; that equation can then be used to predict the toxicity or property of another (usually related) compound.

It may seem strange that a complex endpoint such as a toxicity can be modeled in such a way, but it should be remembered that all molecular properties—chemical, physical, and biological—are a function of molecular structure. An example of a simple QSAR is Eq. 1, which models the toxicity of barbiturates to the mouse [2]:

$$\begin{aligned} \text{Log}(1 / \text{LD}_{50}) &= 1.02 \log P - 0.27 (\log P)^2 + 1.86 & (1) \\ n &= 13 \quad R^2 = 0.852 \quad s = 0.113 \end{aligned}$$

In this QSAR, LD_{50} is the dose required to kill 50 % of the mice, and P is the octanol–water partition coefficient, an important property that reflects the ability of compounds to penetrate lipid membranes in an organism; n is the number of compounds used to develop the QSAR; R^2 is the coefficient of determination, the fraction of the toxicity that is described by the equation; and s is the standard error of the prediction. It can be seen that the partition coefficient, a simple physicochemical property, models 85.2 % of the acute toxicity of barbiturates to the mouse.

Another example, shown in Eq. 2, concerns the toxicity of nitrobenzenes to the aquatic ciliate *Tetrahymena pyriformis* [3]:

$$\begin{aligned} \text{Log}(1 / \text{IGC}_{50}) &= 0.467 \log P - 1.60 E_{\text{LUMO}} - 2.55 & (2) \\ n &= 42 \quad R^2 = 0.881 \quad s = 0.246 \end{aligned}$$

IGC_{50} is the dose that reduces growth rate of the ciliate by 50 %. E_{LUMO} is the energy of the lowest unoccupied molecular orbital, a measure of the electron-attracting ability (electrophilicity) of a compound. Equation 2 demonstrates the use of both a hydrophobic descriptor property, P , and an electronic property, E_{LUMO} , in the same equation.

Equations 1 and 2 are examples of multiple linear regression (MLR), which is the most widely used QSAR modeling technique. There are, however, numerous other techniques that are used in QSAR modeling, including artificial neural networks (ANNs) [4], nonlinear regression, partial least squares, genetic algorithms, and support vector machines (SVM) [5, 6]. Each has its own strengths and weaknesses.

The main strength of ANNs is that they are nonlinear (strictly, nonrectilinear) methods and so can deal with structure–activity relationships in which there are nonrectilinear correlations between activity and one or more descriptor properties. In Eq. 1 above, there is a nonrectilinear correlation between barbiturate toxicity and partition coefficient. In this case, the correlation is parabolic, which can be accounted for by the incorporation of a $(\log P)^2$ term. However, in many cases, the form of nonrectilinearity is not known, so an MLR approach will not yield good correlations. This is particularly important when non-congeneric data sets are being modeled.

It should be noted that most ANN-based QSAR studies have used a back-propagation algorithm [7].

The main weaknesses of ANNs are that they do not yield a QSAR equation directly, they are difficult to interpret, and they do not work well with small data sets [7].

It is important to know the REACH requirements and restrictions concerning the use of QSARs, as stated by its Annex XI:

Results obtained from valid qualitative or quantitative structure-activity relationship models ((Q)SARs) may indicate the presence or absence of a certain dangerous property. Results of (Q)SARs may be used instead of testing when the following conditions are met:

- results are derived from a (Q)SAR model whose scientific validity has been established,
- the substance falls within the applicability domain of the (Q)SAR model,
- results are adequate for the purpose of classification and labelling and/or risk assessment, and,
- adequate and reliable documentation of the applied method is provided.

In addition, guidance on the use of (Q)SARs for the implementation of REACH has been provided by the European Chemicals Agency (ECHA) [8], which states that:

In principle, (Q)SARs can be applied in a number of ways, namely to:

- (a) provide information for use in priority setting procedures;
- (b) guide the experimental design of an experimental test or testing strategy;
- (c) improve the evaluation of existing test data;
- (d) provide mechanistic information (which could be used, for example, to support the grouping of chemicals into categories);
- (e) fill a data gap needed for hazard and risk assessment;
- (f) fill a data gap needed for classification and labelling;
- (g) fill a data gap needed for PBT or vPvB assessment.

The first four applications (a–d) are more general regulatory applications of QSARs, whereas the last three applications (e–g) are more REACH-specific.

In some situations, (Q)SARs could be used to replace test data, whereas in other situations, the models would be used to provide supplementary information to experimental data. In practice, it is foreseen that (Q)SAR information will most often be used to supplement experimental test data within chemical categories and endpoint-specific Integrated Testing Strategies (ITS). However, it is expected that (Q)SARs will be used increasingly for the direct replacement of test data, as relevant and reliable models become increasingly available, and as experience in their use becomes more widespread.

The acronym QSAR (quantitative structure–activity relationship) in REACH documentation is taken to include QSPR (quantitative structure–property relationship).

To be used in a regulatory context, QSAR models should have, according to the five principles proposed by the Organisation for Economic Co-operation and Development (OECD) [9, 10]:

- (a) A defined endpoint
- (b) An unambiguous algorithm
- (c) A defined domain of applicability
- (d) Appropriate measures of goodness of fit, robustness, and predictivity
- (e) A mechanistic interpretation, if possible

In particular, a series of internal and external validation tests are used to demonstrate the reliability of a QSAR model. Goodness of fit is evaluated for the compounds used to derive the model (i.e., the compounds of the training set). Robustness is estimated by cross validation and/or randomization techniques. Predictive power is generally evaluated using an external validation set of compounds that were not used to develop the model, and it is now a standard requirement that external validation of QSAR models is performed.

A useful review of QSAR modeling for regulatory purposes is given by Fjodorova et al. [11].

2 The Value of ANN-Derived QSARs

In general, the neural network approach to QSAR development has several important advantages over the standard multiple linear regression (MLR) approach [12]. Firstly, as pointed out above, the ability of ANNs to utilize nonrectilinear functions can result in better models being obtained. Secondly, some ANNs are able to deal with interactions between descriptors. Thirdly, they are able to handle very noisy data, and fourthly, as a corollary to the above, they are valuable for the derivation of QSARs for non-congeneric series.

There is a plethora of ANN methods available for QSAR modeling [13–15], such as back-propagation NNs, counterpropagation NNs, probabilistic NNs, radial basis function NNs, and computational NNs [16]; Baskin et al. [13] list 35 types of ANNs that have been used in QSAR studies. Care must be taken to select the most appropriate method and to guard against overtraining of the network, which lowers the predictivity of the derived model.

It should be noted that in comparison with other methods for the derivation of QSAR models, ANN methods are not always the best performers. A study of the prediction of biomagnification factors for organochlorine compounds [17] found that a feed-forward ANN approach gave better results than did an MLR approach. However, for the prediction of acute toxicity of diverse organic chemicals to the fathead minnow [18], a support vector machine (SVM) approach gave better results than did a back-propagation ANN.

3 Do ANN-Derived QSARs Meet Validity Requirements for Regulatory Purposes?

Clearly, ANN-derived QSARs must have a defined endpoint in order to satisfy the first OECD principle, and there is no greater difficulty with that than there is with any other QSAR approach. It must be stressed, however, that rigor is required in assembling and checking the database used to develop any QSAR [19].

The second OECD principle requires an unambiguous algorithm, which cannot be obtained with ANN techniques, since they are generally regarded as “black-box” methods. The descriptors used to build an ANN model are known, since they form most of the input neuron layer, but their signs and coefficients are not known. Hence the algorithm cannot be said to be unambiguous. However, Baskin et al. [20] and Jurs and coworkers [21, 22] have proposed ways in which the relative importance of the descriptors in an ANN-derived QSAR can be obtained. This means [13] that “the interpretability of neural network models is not reduced in comparison with traditional statistical linear approaches, at least for interpretation methods based on ranking the relative importance of descriptors.” However, these proposals appear to have been little used to date.

The third OECD principle requires a defined applicability domain, which to a first approximation represents the chemical and biological space covered by the chemicals in the training set, irrespective of the QSAR approach used [23].

The fourth OECD principle calls for appropriate measures of goodness of fit, robustness, and predictivity, and these are as readily available for ANN models as for other types of QSAR models. It should be mentioned, however, that great care must be taken (a) to avoid overtraining in the development of an ANN QSAR and (b) to ensure its external validation.

The fifth OECD principle calls for a mechanistic interpretation of the QSAR model, *if possible*. Thus there is a recognition that such interpretations cannot always be achieved. Johnson [24] has pointed out that “QSAR has devolved into a perfectly practiced art of logical fallacy; *cum hoc ergo propter hoc* (with this, therefore because of this).” He also commented that a statement to the effect that, say, a particular descriptor represents molecular shape is not a mechanistic interpretation. It may also be noted that there is currently a commendable increase in the number of QSARs published that are derived from compounds known to act by a given mechanism. In such cases there is no (or less) need to interpret the QSAR mechanistically. An example is the identification of different mechanisms of action of chemicals that are skin sensitizers, prior to QSAR analysis [25].

Following publication of the OECD principles [8], Vračko et al. [26] carried out a validation study of counterpropagation ANN modeling of fish toxicity. While acknowledging the potential

drawbacks of the method, they concluded that “a (Q)SAR model can be derived and validated using a counter propagation NN approach whilst still satisfying most if not all of the OECD principles for validation of (Q)SAR models.”

However, this does not necessarily mean that ANN-derived QSARs are automatically acceptable for regulatory purposes, including those relating to the REACH legislation. A number of peer-reviewed QSARs are stored in the (Q)SAR Model Reporting Format (QMRF) database at the European Commission Joint Research Centre (JRC) at Ispra, Italy [27]. The website [28] states: “The database is intended to help to identify valid (Q)SARs, e.g. for the purposes of REACH...The information is structured according to the OECD principles for the validation of (Q)SAR models.” Nevertheless, it is important to note that the quality and relevance of QSARs in the QMRF database are not checked by the JRC before inclusion.

There are, at the time of writing, 70 QSARs in the QMRF database, of which 13 are ANN-derived QSARs, covering a range of toxicity endpoints such as acute toxicity to *Daphnia magna*, fathead minnow, birds, and rat (oral), mouse carcinogenicity, dermal irritation, skin sensitization, and chromosomal aberration, as well as QSARs for biodegradation and octanol–water partition coefficient.

It should be noted that QSARs for the prediction of physico-chemical properties are usually referred to as QSPRs (quantitative structure–property relationships).

4 Some Case Studies of ANN-Derived QSARs

As pointed out above, QSARs to be used in connection with REACH need to comply with most of the OECD principles for validation of (Q)SAR models. Hence in this chapter an indication is given of that compliance; for example, (OECD 1, 2, 4I,E) indicates that a model complies with OECD principles 1, 2, and 4, with I and E indicating that there is both internal and external validation of the model.

4.1 Physicochemical Properties

The REACH legislation [29] requires information on up to 19 physicochemical properties, depending on annual supply levels. More detailed information can be found in Regulation (EC) No. 1907/2006 [30].

From Annex VII (required for substances at a supply level of ≥ 1 tonne/year):

State of the substance, melting/freezing point, boiling point, relative density, vapor pressure, surface tension, water solubility, *n*-octanol–water partition coefficient, flash point, flammability*, explosive properties, self-ignition temperature, oxidizing properties, and granulometry (particle size distribution).

*Flammability includes “pyrophoric properties,” “flammability on contact with water,” and “flammability upon ignition.”

From Annex VIII (additionally required for substances at a supply level of ≥ 10 tonnes/year):

Adsorption/desorption (generally taken to mean relating to soil).

From Annex IX additionally (required for substances at a supply level of ≥ 100 tonnes/year):

Dissociation constant, viscosity, and stability in organic solvents.

In addition, the ECHA endpoint-specific guidance [29] includes an appendix on the determination of the air–water partition coefficient, otherwise known as Henry’s law constant. This is an important property with applications in, for example, inhalation toxicology and environmental distribution of chemicals.

Of the above properties, two (state of the substance and granulometry) are not amenable to QSPR analysis, and for two others (oxidizing properties and stability in organic solvents), QSPRs have not yet been derived.

Dearden et al. [31] have discussed the *in silico* prediction of all the above properties that are amenable to QSPR analysis, and Taskinen and Yliruusi [32] and Devillers [14] have reviewed the ANN QSPR modeling of a number of those properties.

4.1.1 Melting/ Freezing Point

Godavarthy et al. [33] used a back-propagation NN to model the melting points of over 1,200 organic chemicals, with $R^2 = 0.99$ and RMSE (root mean square error) = 12.6° (OECD compliance 1,4IE) for the training set and 10.8° for the test set. A typical melting point prediction error is around 30° – 40° , so this result is surprisingly good. However, the statistics are perhaps too good, suggesting that over-fitting may have taken place. More realistic results were obtained [34] for a data set of 4,173 organic chemicals using a feed-forward back-propagation NN and 2- and 3-dimensional (2D and 3D) descriptors; the use of 2D descriptors gave the better results, with a training-set mean absolute error (MAE) of 37.6° and a 277-compound test-set MAE of 32.6° (OECD compliance 1,3,4IE,5).

It should be noted that melting point is probably the least well predicted physicochemical property, partly because of the difficulty of modeling crystal packing.

4.1.2 Boiling Point

Boiling point is somewhat easier to predict. Hall and Story [35] used a back-propagation NN with atom-type electrotopological descriptors to model the boiling points of 298 diverse organic chemicals, with an MAE of 3.9° (OECD compliance 1,4IE). Using a feed-forward NN with quantum-mechanical descriptors, Chalk et al. [36] modeled the boiling points of a large number of organic chemicals, with a 6,000-compound training-set standard deviation of 16.5° and a 629-compound test-set standard deviation of 19.0°

(OECD compliance 1,4IE). They commented that the low error found by Hall and Story [35] could have been caused by overtraining of their model.

4.1.3 *Relative Density*

Relative density is quite simple to calculate, for liquids at least [31]. Gakh et al. [37] used a feed-forward back-propagation NN to predict the densities of 134 liquid hydrocarbons at 25 °C, using descriptors derived from graph theory. They found an MAE of 0.6 % for a 25-chemical external test set (OECD compliance 1,4E). Group contributions and a feed-forward back-propagation NN with two hidden layers were used to predict the densities of 82 ionic liquids [38]. An external test set of 24 ionic liquids yielded an MAE of 0.26 % (OECD compliance 1,4E).

4.1.4 *Vapor Pressure*

Vapor pressure, being temperature dependent, must be reported for a specific temperature, such as 25 °C (298 K). McClelland and Jurs [39] used a computational NN to model the vapor pressures at 25 °C of 420 diverse organic chemicals. They obtained RMSE values of 0.19 and 0.33 log unit for the training and validation sets, respectively (OECD compliance 1,4IE). Chalk et al. [40] used a feed-forward NN to model the variation of vapor pressure with temperature, using quantum-mechanical descriptors with a data set of 7,681 measurements for 2,349 chemicals. Their standard deviations were 0.322 and 0.326 log unit for training and validation sets, respectively (OECD compliance 1,4IE).

4.1.5 *Surface Tension*

The surface tensions at various temperatures of 82 organic liquids were modeled with a multilayer perceptron NN, with five physical properties as descriptors [41], and with an MAE of 1.41 % for the training set of 734 data points and 1.95 % for the test set of 314 data points (OECD compliance 1,4E). Gharagheizi et al. [42] used a much larger number (752) of organic liquids and a range of temperatures. Using a feed-forward NN with group contribution descriptors, they obtained standard deviations of 1.7 % for training and internal and external validation sets (OECD compliance 1,4IE).

4.1.6 *Water Solubility*

One of the most important physicochemical properties is aqueous solubility, and there have been many QSPR studies carried out in this area [43], including those using ANN approaches. Yan and Gasteiger [44] used a back-propagation NN with 3D descriptors to model a set of 1,293 organic chemicals. They obtained, for the 797-compound training set and the 496-compound test set, MAEs 0.41 and 0.49 log unit (OECD compliance 1,4IE). These compare well with the typical experimental error in aqueous solubility of about 0.58 log unit [45]. They are also considerably better than an MLR model for the same chemicals (MAEs 0.70 and 0.68 log unit for training and test sets, respectively). However, when an external set of 1,587 Merck chemicals was tested, the NN MAE

was 0.77 log unit. The authors commented that this was probably due to greater chemical diversity of the Merck chemicals. Similar results were obtained by Bruneau [46] using a Bayesian NN with physicochemical descriptors for a training set of 1,560 organic chemicals (standard error=0.53 log unit) and an external test set of 934 organic chemicals (standard error=0.81 log unit) (OECD compliance 1,4IE).

4.1.7 *n*-Octanol–Water Partition Coefficient

Without doubt the most important physicochemical property for modeling bioactivity is the *n*-octanol–water partition coefficient (P , K_{ow}). In QSAR studies it is always used in logarithmic form as $\log P$ or $\log K_{ow}$. Some 70 % of all QSARs include this term or one related to it, such as the chromatographic retention term R_m . Its importance lies in the fact that it is a surrogate for lipid–water partitioning and hence models xenobiotic transport in an organism, although it can contribute also to receptor binding. There are a number of reviews of the topic [47, 48] and many published studies. Chen [49] compared the performance of MLR, radial basis function neural networks, and support vector machines for the prediction of $\log P$ of about 3,500 organic chemicals. The SVM results were best (training set $s=0.54$, test set $s=0.56$), with MLR slightly worse (training set $s=0.62$, test set $s=0.56$) and RBFNN somewhat worse (training set $s=0.56$, test set $s=0.72$) (OECD compliance 1,4IE). Tetko et al. [50] used electrotopological state (E-state) indices [51] with an ensemble of 50 different neural networks to model $\log P$ values of a very large number of organic chemicals; for a 12,908-compound training set they obtained RMSE=0.38 log unit, and for a 1,174-compound test set RMSE=0.36 log unit (OECD compliance 1,4IE). These results were better than those from MLR and from several commercially available software programs. Considering the very large number of chemicals used, this is a very good result.

4.1.8 Flash Point

The flash point of a chemical is the temperature at which the vapor concentration is equal to the lower explosive limit. Katritzky et al. [52] used both MLR and back-propagation NN with CODESSA descriptors to model the flash points of up to 758 organic chemicals. For the training sets, MLR yielded MAE=13.9°, while NN yielded better results: MAE=12.6° (OECD compliance 1,4IE). No test-set results were reported. Gharagheizi et al. [53], using a feed-forward NN and group contributions as descriptors with a large data set of organic chemicals, found MAE=7.9° for the 1,241-compound training set and MAE=9.9° for the 137-compound test set (OECD compliance 1,4IE).

4.1.9 Flammability Limits

Upper and lower flammability limits represent the range of concentrations in air within which a chemical will ignite provided that an ignition source is present. Studies of both limits have been made by

Gharagheizi. He used a feed-forward NN with group contributions to model the upper flammability limits of 867 organic chemicals [54]; for the 694-compound training set MAE = 7.26 % and for the 173-compound test set MAE = 6.23 % (OECD compliance 1,4IE), whereas using MLR on the same data set [55], he obtained MAE = 9.2 % for the full data set (MAEs not given for training and test sets) (OECD compliance 1,2,4IE).

In similar fashion Gharagheizi investigated the lower flammability limits of a data set of 1,057 organic chemicals. Using MLR [56] he obtained MAE = 7.68 % for the full data set (MAEs not given for training and test sets) (OECD compliance 1,2,4IE), while with a feed-forward NN [57], Gharagheizi obtained MAE = 4.35 % for the 846-compound training set and MAE = 5.70 % for the 211-compound test set (OECD compliance 1,4IE). Clearly, for the prediction of both upper and lower flammability limits, the neural network approach yielded considerably better results.

4.1.10 Explosive Properties

Explosive properties have to date been investigated very little by QSAR, with the exception of impact sensitivity. Cho et al. [58] used back-propagation NN with structural, topological, and physico-chemical descriptors to model the impact sensitivity of 263 nitro compounds, with a standard error of prediction of 0.211 log unit (standard errors not given for training and test sets) (OECD compliance 1,4IE). Wang et al. [59] compared the performance of MLR, partial least squares (PLS), and back-propagation NN in modeling the impact sensitivity of 156 diverse nitro compounds. The RMSEs found for the 127-compound training set and the 49-compound test set, respectively, were MLR 0.210, 0.251; PLS 0.214, 0.250; and BPNN 0.192, 0.247 (OECD compliance 1,4IE). Clearly the back-propagation NN approach gave the best results.

4.1.11 Self-Ignition Temperature (Autoignition Temperature)

MLR, PLS, and radial basis function and back-propagation NNs with structural and physicochemical descriptors were used by Tetteh et al. [60] to model the self-ignition temperature of 233 organic chemicals. Overall MAEs were MLR 90.2°, PLS 38.4°, RBFNN 30.1°, and BPNN 29.9° (MAEs not given for training and test sets separately) (OECD compliance 1,4IE). Pan et al. [61] used MLR and back-propagation NNs with E-state descriptors and group contributions, respectively, to model the self-ignition temperatures of 118 hydrocarbons. MAEs for the 42-compound test set were MLR 32.4°, E-state BPNN 21.6°, and group contribution BPNN 24.8° (OECD compliance 1,4IE).

4.1.12 Adsorption/Desorption

For REACH purposes, adsorption/desorption relates mainly to sorption on soil and also on sediment and sludge. Liu and Yu [62] modeled the sorption of 42 anilines and phenols on soil, using both MLR and back-propagation NN. With log *P*, quantum-chemical, and topological descriptors, they found standard errors

of prediction of 0.374 log unit for MLR and 0.282 log unit for BPNN (OECD compliance 1,4I). No external validation was performed. Soil sorption of 124 pesticides was investigated by Goudarzi et al. [63] using SPA-MLR and SPA-ANN with physicochemical and structural descriptors (SPA is successive projection algorithm); the type of NN used was not specified. The RMSEs obtained (log units) were training set 0.420 (MLR) and 0.282 (NN) and test set 0.371 (MLR) and 0.289 (NN) (OECD compliance 1,4IE).

4.1.13 Dissociation Constant

Dissociation constant is a key property of many chemicals, and hence much effort has gone into developing methods of predicting pK_a values. Luan et al. [64] investigated both MLR and radial basis function NN approaches to pK_a prediction of 74 drugs, using CODESSA descriptors [65]. The neural network approach gave the better results, with MLR yielding RMSE = 0.482 (training set) and 0.987 (test set) and RBFNN yielding RMSE = 0.458 (training set) and 0.613 (test set) (OECD compliance 1,4IE). Habibi-Yangjeh et al. [66], using MLR and feed-forward back-propagation NN with physicochemical descriptors on 242 benzoic acids and phenols, also found much better performance with the neural network approach. RMSE values (log units) were MLR 0.86 (205-compound training set) and 0.94 (37-compound test set) and BPNN 0.26 (training set) and 0.27 (test set) (OECD compliance 1,4IE).

4.1.14 Viscosity

Using their ADAPT descriptors, Kauffman and Jurs [67] developed both MLR and computational NN models to predict the viscosity of almost 200 organic solvents. For 170 training-set chemicals, RMSE (MLR) was found to be 0.257 mPa s, while RMSE (CNN) was 0.147 mPa s. RMSEs for the 21-compound test set were (MLR) 0.278 and (CNN) 0.242 mPa s (OECD compliance 1,4IE). Artemenko et al. [68] found a back-propagation NN, with molecular fragment descriptors, to yield better predictions than did MLR for a diverse set of liquid organic chemicals. For 293 training-set chemicals, RMSEs (log units) were MLR 0.111 and BPNN 0.212. RMSEs for the 37-compound test set were MLR 0.212 and BPNN 0.141 (OECD compliance 1,4IE).

4.1.15 Air–Water Partition Coefficient (Henry's Law Constant)

Although information on Henry's law constant (HLC) is not a REACH requirement, its documentation [29] discusses the measurement and prediction of HLC, so it is pertinent to consider ANN approaches to its prediction. Modarresi et al. [69] compared the performance of MLR and radial basis function NN for prediction of HLC values of 940 diverse organic chemicals using a wide range of physicochemical and structural descriptors. The neural network approach gave slightly better RMSE values (log units) (training set = 0.564, test set 0.520) than did MLR

(training set = 0.570, test set 0.520) (OECD compliance 1,4IE,5). Gharagheizi et al. [70] used a group contribution method with a feed-forward NN to predict HLC values of a large group of organic chemicals. RMSE values (log units) were 1,649-compound training set = 0.08 and 97-compound test set = 0.12 (OECD compliance 1,3,4E).

4.2 Toxicological Properties

The REACH legislation [71] requires information on a total of 19 toxicological properties, depending on annual supply levels. Some of the properties relate to human health, and some to ecotoxicity. More detailed information can be found in Regulation (EC) No. 1907/2006 [30].

From Annex VII (required for substances at a supply level of ≥ 1 tonne/year):

Skin irritation or corrosion, eye irritation, skin sensitization, mutagenicity, and acute toxicity (oral if inhalation data are not available).

Aquatic toxicity (*Daphnia* preferred), growth inhibition of aquatic plants, and degradation (ready biodegradability).

From Annex VIII (additionally required for substances at a supply level of ≥ 10 tonnes/year):

Acute toxicity (dermal, if inhalation is unlikely), short-term repeated-dose toxicity, reproductive toxicity, and toxicokinetics if available.

Short-term fish toxicity, activated sludge respiration inhibition, abiotic degradation (hydrolysis), and adsorption/desorption (generally taken to mean relating to soil).

Note: adsorption/desorption is also a requirement under *physicochemical properties* and is covered in Subheading 4.1.12 above.

From Annex IX (additionally required for substances at a supply level of ≥ 100 tonnes/year):

Sub-chronic rodent toxicity, prenatal developmental toxicity, and two-generation reproductive toxicity.

Long-term aquatic toxicity, bioaccumulation in aquatic species, and effects on terrestrial organisms (invertebrates, soil microorganisms, plants).

From Annex X (additionally required for substances at a supply level of $\geq 1,000$ tonnes/year):

Carcinogenicity, long-term toxicity to invertebrates, sediment microorganisms, plants, and birds.

Note: Annex X also states that additional information may be required on some toxicity endpoints listed under Annexes VII–IX.

4.2.1 Skin Irritation or Corrosion

Salina et al. [72] have reviewed the QSAR prediction of skin irritation and corrosion. Golla et al. [73] used a feed-forward back-propagation NN with physicochemical and structural descriptors to model the rabbit skin irritation of 186 organic chemicals and obtained RMSE = 1.1 unit, which is reasonable for a data range of 0–8 units.

An external test set of 22 chemicals was also tested, but no RMSE value was given (OECD compliance 1,4IE). Skin corrosivity was studied by Barratt [74] using a back-propagation NN with a small number of physicochemical properties for classification as corrosive or noncorrosive; for 50 organic acids, 40 organic bases, and 33 phenols, he found 97.02, 93.86, and 95.85 % correct classification (OECD compliance 1,4). No external test set was used.

4.2.2 Eye Irritation

Salina et al. [72] have reviewed the QSAR prediction of eye irritation. Barratt [75] investigated the eye irritation of 57 aliphatic alcohols and other neutral organic chemicals using a back-propagation NN with a small number of physicochemical properties for classification as irritant or nonirritant and obtained 97.37 % correct classification (OECD compliance 1,4). No external test set was used.

Patlewicz et al. [76] investigated a set of 19 cationic surfactants with back-propagation NN using a small number of physicochemical properties, which yielded $R^2=0.702$ (OECD compliance 1,4). No error values were given.

4.2.3 Skin Sensitization

Patlewicz et al. [77] have reviewed the QSAR prediction of skin sensitization. Devillers [78] used a feed-forward back-propagation NN with structural and physicochemical descriptors in a classification model of the skin sensitization of 259 organic chemicals and obtained an overall correct classification of 89.19 %, compared with 82.63 % using linear discriminant analysis [79] (OECD compliance 1,4IE). No external test-set results were given.

Golla et al. [80] also used a feed-forward back-propagation NN with structural and physicochemical descriptors in a classification model of skin sensitization, using three different endpoints (mouse local lymph node assay (LLNA), guinea-pig maximization test (GPMT), and a test from the German Federal Institute for Health Protection of Consumers and Veterinary Medicine (BgVV)). The training-set results were LLNA (358 chemicals) 90 % correct classification, GPMT (307 chemicals) 95 %, and BgVV (251 chemicals) 90 % (OECD compliance 1,4IE). No test-set results were given.

4.2.4 Mutagenicity

Benigni [81] has reviewed the QSAR prediction of mutagenicity. Xu et al. [82] investigated a very large data set of 7,617 organic chemicals, of which 4,252 were mutagens and 3,365 were non-mutagens. Using molecular fingerprints as descriptors, they employed five different approaches, namely, SVM, decision tree, feed-forward NN, k-nearest neighbor, and naïve Bayes, to classify the chemicals. The prediction accuracies for the training set were, respectively, 84.1 %, 80.4 %, 80.2 %, 82.1 %, and 65.8 % (OECD compliance 1,4). The NN method, while good, was not quite as accurate as the SVM, decision tree, and k-nearest neighbor methods. No test-set results were given.

For a set of aromatic amines, Leong et al. [83] also employed different approaches, namely, MLR, PLS, hierarchical support vector regression, SVM, radial basis function NN, and genetic function algorithm. The RMSEs (log units) found for the 97-compound training set were, respectively, 1.46, 1.66, 0.51, 0.71, 0.85, and 0.90; for the 25-compound test set the RMSEs were 1.13, 0.91, 0.65, 0.85, 0.78, and 0.83 (OECD compliance 1,4IE). Again, the NN approach did not yield the best results.

4.2.5 Acute Toxicity (Mammalian)

Tsakovska et al. [84] have reviewed the QSAR prediction of mammalian toxicity. Devillers [85] employed PLS regression and back-propagation NN to model acute oral toxicity (LD_{50}) to the rat of 51 organophosphorus pesticides, with the use of physicochemical descriptors. The NN approach yielded RMSE=0.29 log unit for the training set and 0.26 log unit for the test set (OECD compliance 1,4IE); these figures were greatly superior to those from PLS regression.

The oral toxicity of 54 benzodiazepine drugs to the mouse was modeled by MLR, back-propagation ANN, SVM, and PLS methods [86]. RMSE values (log units) were, respectively, training set 0.213, 0.142, 0.183, and 0.174 and test set 0.194, 0.124, 0.192, and 0.165, indicating the superior predictivity of the ANN method (OECD compliance 1,4IE).

No mammalian inhalation toxicity QSAR studies appear to have been done using a neural network approach.

4.2.6 Aquatic Toxicity

Kaiser and Niculescu [87] used a probabilistic NN to model a large data set of toxicity of organic chemicals to *Daphnia magna*, using a combination of physicochemical and structural descriptors. For the 700-compound training set, $s=0.512$ log unit, and for the 76-compound test set $s=0.668$ log unit (OECD compliance 1,4IE).

Another aquatic invertebrate that has been widely used in QSAR toxicity studies is the ciliate *Tetrahymena pyriformis*. Over 2,000 chemicals have been tested in the same laboratory against this organism, which means that experimental error should be very low. Kahn et al. [88] compared the performance of MLR and back-propagation NN on a large number of organic chemicals, using CODESSA descriptors. The BPNN results were better (914-compound training set $s=0.442$ log unit, 457-compound test set $s=0.484$ log unit) than were those from MLR (training set $s=0.551$ log unit, test set $s=0.561$ log unit) (OECD compliance 1,4IE). Kahn et al. also reported 18 previous ANN QSAR studies on toxicity to *T. pyriformis*.

Izadiyan et al. [89] modeled the effects of 40 ionic chemicals on the limnic green alga *Scenedesmus vacuolatus*, using both MLR and multilayer perceptron NN, with structural descriptors. Standard errors of prediction in the validation set were MLR 0.525 log unit and MLPNN 0.440 log unit (OECD compliance 1,4IE).

4.2.7 Degradation (Ready Biodegradability)

Pesticide field half-lives were modeled [90] in a classification approach with a back-propagation NN, using structural fragments as descriptors. For 110 training-set compounds and 13 test-set compounds, correct classifications were 95.5 % and 84.6 %, respectively (OECD compliance 1,4IE). These results were much better than those obtained with discriminant factor analysis.

Aerobic biodegradation of 21 aromatic chemicals was modeled [91] by MLR, principal component regression, and back-propagation NN, using quantum-mechanical descriptors. The BPANN model yielded RMSE = 0.136 log unit (training set) and 0.024 log unit (validation set), markedly better than the results from the MLR and PCR models (OECD compliance 1,4IE,5).

4.2.8 Acute Dermal Toxicity (Mammalian)

There do not appear to be any neural network QSAR studies of mammalian toxicity through dermal absorption.

4.2.9 Short-Term Repeat-Dose Toxicity

A QSAR (No. Q17-10-31-264) in the (Q)SAR Model Reporting Format (QMRF) [28] uses a back-propagation NN to model repeat-dose reproductive/developmental toxicity in the rat. The reported statistics are RMSE (training set) = 0.434 log unit and RMSE (test set) = 2.521 log unit (OECD compliance 1,3,4IE,5). The fact that the test-set compounds are so poorly predicted suggests either overtraining or that the test-set compounds were not within the applicability domain of the training set.

4.2.10 Reproductive Toxicity

Most QSAR studies of reproductive toxicity have been made on endocrine disruption. Liu et al. [92] used least-square SVM, counterpropagation NN, and k-nearest neighbor (k-NN) approaches to classify the endocrine-disruption capability of 232 training-set chemicals and 87 external test-set chemicals, using molecular structure descriptors. The three methods gave very similar results, with correct predictions as follows: LS-SVM 89.66 %, CP-NN 87.50 %, and k-NN 89.22 % for the training set and LS-SVM 83.91 %, CP-NN 82.76 %, and k-NN 83.91 % for the external test set (OECD compliance 1,3,4IE).

Roncaglioni et al. [93] investigated the performance of five different nonlinear QSAR models (decision forest (DF), adaptive fuzzy partition (AFP), decision tree (CART), multilayer perceptron feed-forward NN (MLPNN), and support vector machine (SVM)) to classify the endocrine-disruption capability of 232 training-set chemicals and 87 external test-set chemicals, using topological and structural descriptors. The correct classifications were DF 97.04 %, AFP 86.36 %, CART 85.38 %, MLPNN 84.39 %, and SVM 89.92 % for the training set and DF 85.33 %, AFP 85.33 %, CART 85.33 %, MLPNN 84.00 %, and SVM 86.67 % for the external test set (OECD compliance 1,3,4IE).

4.2.11 *Toxicokinetics*

There are numerous toxicokinetic parameters, and many QSAR studies have been made of most of them. Wajima et al. [94] investigated the prediction of human clearance from data for 68 drugs on humans, dogs, and rats, with physicochemical descriptors and dog and rat data, using several approaches. RMSEs (log units) were MLR 0.350, PLS 0.350, and feed-forward NN 0.395 (OECD compliance 1,4). No external test set was used.

Blood levels of the antibiotic tobramycin in about 300 patients were modeled with a back-propagation NN, using patient data (e.g., age, weight, sex, illness) as descriptors [95]. The MAE was 33.9 %, which was considered acceptable (OECD compliance 1,4). No external test set was used. The authors concluded that “neural networks were capable of capturing the relationships between plasma drug levels and patient-related prognostic factors from routinely collected sparse within-patient pharmacokinetic data.”

4.2.12 *Short-Term Fish Toxicity*

Singh et al. [96] compared the performance of a number of artificial intelligence approaches (multilayer perceptron NN, radial basis function NN, generalized regression NN, gene expression programming (GEP), SVM, and decision tree (DT)) to model the acute fish toxicity of 573 chemicals, of which 458 formed the training set and 115 the test set. The MAEs (log units) obtained for the training set were MLPNN 0.54, RBFNN 0.57, GRNN 0.34, GEP 0.71, and DT 0.43, and those for the test set were MLPNN 0.46, RBFNN 0.41, GRNN 0.37, SVM 0.49, GEP 0.48, and DT 0.54 (OECD compliance 1,4IE,5). The generalized regression NN in particular performed very well.

Gong et al. [97] used radial basis function NN and a heuristic method to model the acute fish toxicity of 92 substituted benzenes (training set 74, test set 18 chemicals) using CODESSA descriptors. The RMSEs (log units) for the training and test sets, respectively, were RBFNN 0.220 and 0.205 and heuristic method 0.273 and 0.223 (OECD compliance 1,4IE,5).

4.2.13 *Activated Sludge Respiration Inhibition*

Okey and Martis [98] employed a feed-forward NN to model the inhibition of activated sludge respiration by 139 (training set) and 25 (test set) diverse organic chemicals, using quantum-mechanical and topological descriptors. They obtained MAE = 0.41 log unit for the test set, which is a reasonable result (OECD compliance 1,4E,5).

A QSAR (No. Q17-10-1-226) in the (Q)SAR Model Reporting Format (QMRF) [28] employs a multilayer perceptron back-propagation NN to model the inhibition of activated sludge respiration, using physicochemical and structural descriptors. The reported statistics are RMSE (68-chemical training set) = 0.029 log unit and RMSE (15-chemical test set) = 0.03 log unit (OECD compliance 1,2,3,4IE,5).

4.2.14 Abiotic Degradation (Hydrolysis)

The acid hydrolysis of a large number of carboxylic acid esters in water and in water–organic solvent mixtures was modeled [99] using a feed-forward NN and quantum-chemical descriptors, descriptors representing the various organic solvents, and temperature and solvent concentration. For the training set (1,883 records), RMSE = 0.271 log unit, and for the test set (209 records), RMSE = 0.342 log unit (OECD compliance 1,4IE).

4.2.15 Adsorption/ Desorption

See Subheading 4.1.12 above.

4.2.16 Sub-chronic Rodent Toxicity

Sub-chronic toxicity is defined as the ability of a toxic substance to cause toxic effects for more than 1 year but less than the lifetime of the exposed organism. It thus excludes carcinogenicity. A recent paper [100] makes it clear that sub-chronic toxicology is still a young science, with only a few published prediction studies, and apparently none using ANN approaches.

4.2.17 Prenatal Developmental Toxicity

There do not appear to be any neural network QSAR studies of either of these endpoints.

4.2.18 Two-Generation Reproductive Toxicity

4.2.19 Long-Term Aquatic Toxicity

Meng and Lin [101] used feed-forward back-propagation NNs to model both acute and chronic (up to 56-day) toxicity of alcohol ethoxylates (nonionic surfactants) to fathead minnow (*Pimephales promelas*), *Daphnia magna*, and green alga (*Pseudokirchneriella subcapitata*). Unfortunately they presented their results largely in graphical form, making it virtually impossible to obtain accurate numerical predictions, even though they used both training and external test sets. It appears from the graphs that chronic toxicity ran more or less in parallel to acute toxicity. For example, for *D. magna* the mean ratio of acute median effect concentration to chronic no-observed-effect concentration was 2.8.

4.2.20 Bioaccumulation in Aquatic Species

Bioaccumulation incorporates both bioconcentration factor, BCF (which relates to the uptake of chemicals by fish or other aquatic species from water), and biomagnification, which is the increase in concentration from one link in a food chain to another. For REACH purposes, bioaccumulation is taken to mean BCF.

Fatemi et al. [102] used a genetic algorithm-feed-forward back-propagation NN on a training set of fish BCF values for 44 diverse organic chemicals and a test set of 9 similar chemicals, using topological and physicochemical descriptors. They obtained $s = 0.259$ log unit for the training set and $s = 0.398$ log unit for the test set (OECD compliance 1,4IE).

Zhao et al. [103] used MLR, radial basis function NN, and SVM to model a large data set of fish BCF values, using topological and physicochemical descriptors. For the 378-chemical training set, standard errors of prediction (log units) were MLR 0.70, RBFNN 0.58, and SVM 0.63, and for the 95-chemical test set, the errors were MLR 0.74, RBFNN 0.63, and SVM 0.62. A hybrid RBFNN, from two models with different descriptors, yielded $s=0.56$ for the training set and 0.59 for the test set (OECD compliance 1,4IE).

4.2.21 *Effects on Terrestrial Organisms (Invertebrates, Microorganisms, Plants)*

Honeybees are susceptible to pesticides, and Devillers et al. [104] developed a feed-forward back-propagation NN model of the acute toxicity of 100 pesticides to *Apis mellifera*. Using the auto-correlation vectors of physicochemical properties as descriptors, they obtained RMSE=0.430 log unit for the 89-chemical training set and 0.386 log unit for the 11-chemical test set (OECD compliance 1,4IE). These good results contrasted sharply with poor results from a PLS model (data not given).

A study of fungicidal activities of thiazoline derivatives against rice blast (*Magnaporthe grisea*) used both MLR and feed-forward back-propagation NN, with physicochemical descriptors [105]. Three sets of thiazoline derivatives, with different substitution patterns, were used, and similar good results were obtained for all three sets. The largest set used 82 training compounds (standard error (log units)=0.139 (MLR) and 0.097 (NN)) and 18 test compounds (standard error (log units)=0.162 (MLR) and 0.122 (NN)) (OECD compliance 1,4IE).

4.2.22 *Carcinogenicity*

It is perhaps surprising that carcinogenicity determination is required only for production rates over 1,000 tonnes/year.

Counterpropagation NN was used by Fjodorova et al. [106] to model both qualitative and quantitative measures of rat carcinogenicity, using topological and molecular properties. For a 644-compound training set and 27 descriptors, they obtained RMSE=1.52 log unit, while for a 161-chemical test set, they obtained RMSE=1.80 (OECD compliance 1,4E). Strangely, this model was developed using both carcinogens and noncarcinogens. However, when they redeveloped the model using only the 421 carcinogens, the model was no better. The authors comment that their results for the test set “are not high enough to fulfill criteria for the predictive power of QSAR models concerning prediction of carcinogenic potency.” They therefore developed a qualitative (classification) model using the same training and test sets as before. For the training set, they obtained 92.2 % correct classification but only 68.3 % for the test set (OECD compliance 1,4E).

Singh et al. [107] also employed NN approaches to model both qualitative and quantitative measures of carcinogenicity of rodents, using two NN methods with physicochemical, structural,

and topological descriptors. For the qualitative (classification) work on rat data, they used probabilistic NN and, with a training set of 373 active and 294 inactive chemicals, found 93.85 % correct classification; for the test set of 93 active and 74 inactive chemicals, they found 85.03 % correct classification (OECD compliance 1,2,4IE). For their quantitative modeling, they employed generalized regression NN on 457 carcinogens tested in the rat and, with an unspecified number of training and test-set chemicals, found MAE (training)=0.44 log unit and MAE (test)=0.72 log unit. They then used the model to predict the carcinogenicity of 292 chemicals in the mouse (MAE=0.68 log unit) and of 38 chemicals in the hamster (MAE=0.57 log unit) (OECD compliance 1,4IE).

4.2.23 *Long-Term Effects on Terrestrial Organisms (Invertebrates, Microorganisms, Plants, Birds)*

There do not appear to be any neural network QSAR studies of long-term effects on terrestrial organisms.

4.3 Software for Property and Toxicity Prediction

Many computer packages are available for the prediction of physicochemical and toxicological properties. Some of these are freely distributed; others are commercial. Dearden et al. [31] have reviewed the software available for physicochemical property prediction, and Dearden [108] and Lapenna et al. [109] have reviewed the software available for toxicity prediction. Unfortunately there is a general lack of transparency concerning the nature of the algorithms used in these packages, and so it is impossible to say with confidence, in many cases, whether or not the package uses neural networks for prediction and whether there is OECD compliance.

Among those that predict physicochemical properties, some are clearly described as using methods other than neural networks. For example, Molecular Modelling Pro [110] is described as modeling melting and boiling points by the method of Joback and Reid [111], which is a multiple linear regression method. Others use neural network methods. For example, ADMET Predictor [112] uses neural networks in the prediction of p*K*_a, aqueous solubility, and octanol–water partition coefficient. ChemSilico [113] also uses neural network methods in the prediction of aqueous solubility and octanol–water partition coefficient.

There is a similar situation in regard to toxicity prediction. TOPKAT [114] is described by Lapenna et al. [109] as using regression analysis to predict rat oral LD₅₀. In contrast, TerraQSAR [115] is described [116] as using neural net technology to predict mouse and rat oral and intravenous LD₅₀, fathead minnow and *Daphnia magna* LC₅₀, and skin irritation. ADMET Predictor [112] uses neural networks to predict mutagenicity and endocrine disruption, and ChemSilico [113] uses neural networks to predict mutagenicity.

5 Conclusions

Various types of artificial neural networks are now widely used in the prediction of physicochemical properties and a wide range of toxicity endpoints relevant to REACH requirements. Generally ANNs perform well in both classification and quantitative endpoint models. The ANN QSAR models reported herein are acceptable with regard to some of the OECD Guidelines for the Validation of (Q)SARs, usually Guidelines 1 and 4. However, there is no reason why ANN QSAR models cannot comply with Guideline 3 as well and perhaps with Guideline 5 also. It is unfortunate that, to date, very few commercially available software packages for property and toxicity prediction give any indication on their websites as to whether or not ANNs are used in their predictive models.

Acknowledgments

We are grateful to Dr. T.I. Netzeva and Dr. A.P. Worth for valuable comments on the draft manuscript.

References

1. European Parliament: Regulation (EC) N° 1907/2006 of the European Parliament and of the Council of 18 December 2006 concerning the Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH). Available at <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=oj:l:2006:396:0001:0849:en:pdf>. Accessed 5 Jan 2014
2. Hansch C, Clayton JM (1973) Lipophilic character and activity of drugs II: the parabolic case. *J Pharm Sci* 62:1–23
3. Cronin MTD, Gregory BW, Schultz TW (1998) Quantitative structure-activity analyses of nitrobenzene toxicity to *Tetrahymena pyriformis*. *Chem Res Toxicol* 11:902–908
4. Devillers J (ed) (1996) Neural networks in QSAR and drug design. Academic, London
5. Gasteiger J, Engel T (2003) Chemoinformatics—a textbook. Wiley, Weinheim
6. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques. Morgan Kaufmann, San Francisco
7. Devillers J (1996) Strengths and weaknesses of the backpropagation neural network QSPR studies. In: Devillers J (ed) Neural networks in QSAR and drug design. Academic, London, pp 1–46
8. ECHA Guidance on information requirements and safety assessment. Chapter R.6. QSARs and grouping of chemicals, 2008. Available at http://echa.europa.eu/documents/10162/13632/information_requirements_r6_en.pdf. Accessed 5 Jan 2014
9. OECD Principles for the Validation of (Q)SARs. Available at <http://www.oecd.org/dataoecd/33/37/37849783.pdf>. Accessed 5 Jan 2014
10. OECD Environment Directorate, Joint Meeting of the Chemicals Committee and the Working Party on Chemicals, Pesticides and Biotechnology. Available at [http://www.oecd.org/olis/2004doc.nsf/LinkTo/NT00009192/\\$FILE/JT00176183.PDF](http://www.oecd.org/olis/2004doc.nsf/LinkTo/NT00009192/$FILE/JT00176183.PDF). Accessed 5 Jan 2014
11. Fjodorova N, Novich M, Vrachko M et al (2008) Directions in QSAR modeling for regulatory use in OECD member countries, EU and in Russia. *J Environ Sci Health C Environ Carcinog Ecotoxicol Rev* 26: 201–236
12. Devillers J (2005) A new strategy for using supervised artificial neural networks in QSAR. *SAR QSAR Environ Res* 16:433–442
13. Baskin II, Palyulin VA, Zefirov NS (2008) Neural networks in building QSAR models. In: Livingstone DS (ed) Artificial neural

- networks: methods and protocols. Humana Press, New York, pp 137–158
14. Devillers J (2009) Artificial neural network modeling of the environmental fate and ecotoxicity of chemicals. In: Devillers J (ed) Ecotoxicology modeling. Springer, Dordrecht, pp 1–28
 15. Gleeson MP, Modi S, Bender A et al (2012) The challenges involved in modelling toxicity data *in silico*: a review. *Curr Pharm Des* 18: 1266–1291
 16. Moore DRJ, Breton RL, MacDonald DB (2003) A comparison of model performance for six quantitative structure-activity relationship packages that predict acute toxicity to fish. *Environ Toxicol Chem* 22:1799–1809
 17. Fatemi MH, Abraham MH, Haghdadadi M (2009) Prediction of biomagnification factors for some organochlorine compounds using linear free energy relationship parameters and artificial neural networks. *SAR QSAR Environ Res* 20:453–465
 18. Tan N-X, Li P, Rao H-B et al (2010) Prediction of the acute toxicity of chemical compounds to the fathead minnow by machine learning approaches. *Chemometr Intell Lab Syst* 100:66–73
 19. Dearden JC, Cronin MTD, Kaiser KLE (2009) How not to develop a quantitative structure-activity or structure-property relationship (QSAR/QSPR). *SAR QSAR Environ Res* 20:241–266
 20. Baskin II, Ait AO, Halberstam NM et al (2002) An approach to the interpretation of backpropagation neural network models in QSAR studies. *SAR QSAR Environ Res* 13: 35–41
 21. Guha R, Jurs PC (2005) Interpreting computational neural network QSAR models: a measure of descriptor importance. *J Chem Inf Model* 45:800–806
 22. Guha R, Stanton DT, Jurs PC (2005) Interpreting computational neural network QSAR models: a detailed interpretation of the weights and biases. *J Chem Inf Model* 45: 1109–1121
 23. Chaudhry Q, Piclin N, Cotterill J et al (2010) Global QSAR models of skin sensitizers for regulatory purposes. *Chem Cent J* 4(Suppl 1): S1–S6
 24. Johnson SR (2008) The trouble with QSAR (or how I learned to stop worrying and embrace fallacy). *J Chem Inf Model* 48:25–26
 25. Enoch SJ, Madden JC, Cronin MTD (2008) Identification of mechanisms of toxic action for skin sensitisation using a SMARTS pattern based approach. *SAR QSAR Environ Res* 19:555–578
 26. Vračko M, Bandelj V, Barbier P et al (2006) Validation of counter propagation neural network models for predictive toxicology according to the OECD principles: a case study. *SAR QSAR Environ Res* 17:265–284
 27. Worth AP (2010) The role of QSAR methodology in the regulatory assessment of chemicals. In: Puzyn T, Leszczynski J, Cronin MTD (eds) Recent advances in QSAR studies: methods and applications. Springer, Dordrecht, pp 367–382
 28. (Q)SAR Model Reporting Format. Available at <http://qsar.db.jrc.it/qmrf>. Accessed 16 Jan 2014
 29. ECHA Guidance on information requirements and chemical safety assessment. Chapter R.7a. Endpoint specific guidance, in Guidance for the Implementation of REACH, ECHA, Helsinki, 2012. Available at <http://echa.europa.eu/support/guidance-on-reach-and-clp-implementation/consultation-procedure>. Accessed 25 Jan 2014
 30. Regulation (EC) No. 1907/2006 of the European Parliament and of the Council of 18 December 2006. Available at <http://eurlex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:2006>. Accessed 25 Jan 2014
 31. Dearden JC, Rotureau P, Fayet G (2013) QSPR prediction of physico-chemical properties for REACH. *SAR QSAR Environ Res* 24:279–318
 32. Taskinen J, Yliruusi J (2003) Prediction of physicochemical properties based on neural network modelling. *Adv Drug Deliv Rev* 55:1163–1183
 33. Godavarthy SS, Robinson RL, Gasem KAM (2006) An improved structure-property model for predicting melting-point temperatures. *Ind Eng Chem Res* 45:5117–5126
 34. Karthikeyan M, Glen RC, Bender A (2005) General melting point prediction based on a diverse compound data set and artificial neural networks. *J Chem Inf Model* 45:581–590
 35. Hall LH, Story CT (1996) Boiling point and critical temperature of a heterogeneous data set. QSAR with atom type electrotopological state indices using artificial neural networks. *J Chem Inf Comput Sci* 36:1004–1014
 36. Chalk AJ, Beck B, Clark T (2001) A quantum mechanical/neural network model for boiling points with error estimation. *J Chem Inf Comput Sci* 41:457–462
 37. Gakh AA, Gakh EG, Sumpter BG et al (1994) Neural network-graph theory approach to the prediction of the physical properties of organic compounds. *J Chem Inf Comput Sci* 34:832–839

38. Valderrama JO, Reátegui A, Rojas RE (2009) Density of ionic liquids using group contribution and artificial neural networks. *Ind Eng Chem Res* 48:3254–3259
39. McClelland HE, Jurs PC (2000) Quantitative structure-property relationships for the prediction of vapor pressures of organic compounds from molecular structure. *J Chem Inf Comput Sci* 40:967–975
40. Chalk AJ, Beck B, Clark T (2001) A temperature-dependent quantum mechanical/neural net model for vapor pressure. *J Chem Inf Comput Sci* 41:1053–1059
41. Roosta A, Setoodeh P, Jahanmiri A (2012) Artificial neural network modeling of surface tension for pure organic compounds. *Ind Eng Chem Res* 51:561–566
42. Gharagheizi F, Eslamianesh A, Mohammadi AH et al (2011) Use of artificial neural network-group contribution method to determine surface tension of pure compounds. *J Chem Eng Data* 56:2587–2601
43. Dearden JC (2006) *In silico* prediction of aqueous solubility. *Expert Opin Drug Discov* 1:31–52
44. Yan A, Gasteiger J (2003) Prediction of aqueous solubility of organic compounds based on a 3D structure representation. *J Chem Inf Comput Sci* 43:429–434
45. Katritzky AR, Wang Y, Sild S et al (1998) QSPR studies on vapor pressure, aqueous solubility, and the prediction of water-air partition coefficients. *J Chem Inf Comput Sci* 38:720–725
46. Bruneau P (2001) Search for predictive generic model of aqueous solubility using Bayesian neural nets. *J Chem Inf Comput Sci* 41:1605–1616
47. Livingstone DJ (2003) Theoretical property predictions. *Curr Top Med Chem* 3:1171–1192
48. Klopman G, Zhu H (2005) Recent methodologies for the estimation of *n*-octanol-water partition coefficients and their use in the prediction of membrane transport properties of drugs. *Mini Rev Med Chem* 5:127–133
49. Chen H-F (2009) *In silico* log P prediction for a large data set with support vector machines, radial basis neural networks and multiple linear regression. *Chem Biol Drug Des* 74:142–147
50. Tetko IV, Tanchuk VY, Villa AEP (2001) Prediction of *n*-octanol-water partition coefficients from PHYSPROP database using artificial neural networks and E-state indices. *J Chem Inf Comput Sci* 41:1407–1421
51. Kier LB, Hall LH (1999) Molecular structure description: the electrotopological state. Academic, New York
52. Katritzky AR, Stoyanova-Slavova IB, Dobchev DA et al (2007) QSPR modelling of flash points: an update. *J Mol Graph Model* 26:529–536
53. Gharagheizi F, Alamdari RF, Angaji MT (2008) A new neural network-group contribution method for estimation of flash point temperature of pure components. *Energy Fuel* 22:1628–1635
54. Gharagheizi F (2010) Chemical structure-based model for estimation of the upper flammability limit of pure compounds. *Energy Fuel* 24:3867–3871
55. Gharagheizi F (2009) Prediction of upper flammability limit percent of pure compounds from their molecular structures. *J Hazard Mater* 167:507–510
56. Gharagheizi F (2008) Quantitative structure-property relationship for prediction of the lower flammability limit of pure compounds. *Energy Fuel* 22:3037–3039
57. Gharagheizi F (2009) A new group contribution-based model for estimation of lower flammability limit of pure compounds. *J Hazard Mater* 170:595–604
58. Cho SG, No KT, Goh EM et al (2005) Optimization of neural networks architecture for impact sensitivity of energetic molecules. *Bull Korean Chem Soc* 26:399–408
59. Wang R, Jiang J, Pan Y et al (2009) Prediction of impact sensitivity of nitro energetic compounds by neural network based on electrotopological-state indices. *J Hazard Mater* 166:155–186
60. Tetteh J, Metcalfe E, Howells SL (1996) Optimisation of radial basis and backpropagation neural networks for modelling auto-ignition temperature by quantitative structure-property relationships. *Chemometr Intell Lab Syst* 32:177–191
61. Pan Y, Jiang J, Wang R et al (2008) Prediction of auto-ignition temperatures of hydrocarbons by neural network based on atom-type electrotopological-state indices. *J Hazard Mater* 157:510–517
62. Liu GS, Yu JG (2005) QSAR analysis of soil sorption coefficients for polar organic chemicals: substituted anilines and phenols. *Water Res* 39:2048–2055
63. Goudarzi N, Goodarzi M, Araujo MCU et al (2009) QSPR modeling of soil sorption coefficients (K_{oc}) of pesticides using SPA-ANN and SPA-MLR. *J Agric Food Chem* 57:7153–7158

64. Luan F, Ma W, Zhang H et al (2005) Prediction of pKa for neutral and basic drugs based on radial basis function neural networks and the heuristic method. *Pharm Res* 22: 1454–1460
65. CODESSA software. Available at www.semi-chem.com. Accessed 26 Jan 2014
66. Habibi-Yangjeh A, Danandeh-Jenagharad M, Nooshyar M (2005) Prediction acidity constant of various benzoic acids and phenols in water using linear and nonlinear QSPR models. *Bull Korean Chem Soc* 26:2007–2016
67. Kauffman GW, Jurs PC (2001) Prediction of surface tension, viscosity and thermal conductivity for common organic solvents using quantitative structure-property relationships. *J Chem Inf Comput Sci* 41:408–418
68. Artemenko NV, Baskin II, Palyulin VA et al (2001) Prediction of physical properties of organic compounds using artificial neural networks within the substructure approach. *Doklady Chem* 381:317–320
69. Modarresi H, Modarress H, Dearden JC (2007) QSPR model of Henry's law constant for a diverse set of organic chemicals based on genetic algorithm-radial basis function network approach. *Chemosphere* 66:2067–2076
70. Gharagheizi F, Abbasi R, Tirandazi B (2010) Prediction of Henry's law constant of organic compounds in water from a new group-contribution-based method. *Ind Eng Chem Res* 49:10149–10152
71. ECHA Guidance on information requirements and chemical safety assessment. Chapter R.7b. Endpoint specific guidance, in Guidance for the Implementation of REACH, ECHA, Helsinki, 2012. Available at <http://echa.europa.eu/support/guidance-on-reach-and-clp-implementation/consultation-procedure>. Accessed 17 Jan 2014
72. Salina AG, Patlewicz G, Worth AP (2008) A review of QSAR models for skin and eye irritation and corrosion. *QSAR Comb Sci* 27: 49–59
73. Golla S, Madihally S, Robinson RL et al (2009) Quantitative structure-property relationships modeling of skin irritation. *Toxicol In Vitro* 23:176–184
74. Barratt MD (1996) Quantitative structure-activity relationships (QSARs) for skin corrosivity of organic acids, bases and phenols: principal components and neural networks analysis of extended datasets. *Toxicol In Vitro* 10:85–94
75. Barratt MD (1997) QSARs for the eye irritation potential of neutral organic chemicals. *Toxicol In Vitro* 11:1–8
76. Patlewicz GY, Rodford RA, Ellis G et al (2000) A QSAR model for the eye irritation of cationic surfactants. *Toxicol In Vitro* 14:79–84
77. Patlewicz G, Roberts DW, Uriarte E (2008) A minireview of available skin sensitization (Q)SARs/expert systems. *Chem Res Toxicol* 21:521–541
78. Devillers J (2000) A neural network SAR model for allergic contact dermatitis. *Toxicol Methods* 10:181–193
79. Cronin MTD, Basketter DA (1994) Multivariate QSAR analysis of a skin sensitization database. *SAR QSAR Environ Res* 2: 159–179
80. Golla S, Madihally S, Robinson RL et al (2009) Quantitative structure-property relationship modeling of skin sensitization: a quantitative prediction. *Toxicol In Vitro* 23: 454–465
81. Benigni R (ed) (2003) Quantitative structure-activity relationship (QSAR) models of mutagens and carcinogens. CRC Press, Boca Raton
82. Xu C, Cheng F, Chen L et al (2012) In silico prediction of chemical Ames mutagenicity. *J Chem Inf Model* 52:2840–2847
83. Leong MK, Lin S-W, Chen H-B et al (2010) Predicting mutagenicity of aromatic amines by various machine learning approaches. *Toxicol Sci* 116:498–513
84. Tsakovska I, Lessigiarska I, Netzeva T et al (2008) A mini review of mammalian toxicity (Q)SAR models. *QSAR Comb Sci* 27:41–48
85. Devillers J (2004) Prediction of mammalian toxicity of organophosphorus pesticides from QSTR modelling. *SAR QSAR Environ Res* 15:501–510
86. Funar-Timofei S, Ionescu D, Suzuki T (2010) A tentative quantitative structure-toxicity relationship study of benzodiazepine drugs. *Toxicol In Vitro* 24:184–200
87. Kaiser KLE, Niculescu SP (2001) Modeling acute toxicity of chemicals to *Daphnia magna*: a probabilistic neural network approach. *Environ Toxicol Chem* 20:420–431
88. Kahn I, Sild S, Maran U (2007) Modeling the toxicity of chemicals to *Tetrahymena pyriformis* using heuristic multilinear regression and heuristic back-propagation neural networks. *J Chem Inf Model* 47:2271–2279
89. Izadiyan P, Fatemi MH, Izadiyan M (2013) Elicitation of the most important structural properties of ionic liquids affecting ecotoxicity in limnic green algae: a QSAR approach. *Ecotoxicol Environ Saf* 87:42–48
90. Domine D, Devillers J, Chastrette M et al (1993) Estimating pesticide field half-lives

- from a backpropagation neural network. SAR QSAR Environ Res 1:211–219
91. Jing G-H, Li X-L, Zhou Z-M (2011) Quantitative structure-biodegradability relationship study about the aerobic biodegradation of some aromatic compounds. Chin J Struct Chem 30:368–375
 92. Liu H, Papa E, Walker JD et al (2007) *In silico* screening of estrogen-like chemicals based on different nonlinear classification models. J Mol Graph Model 26:135–144
 93. Roncaglioni A, Piclin N, Pintore M et al (2008) Binary classification models for endocrine disrupter effects mediated through the estrogen receptor. SAR QSAR Environ Res 19:697–733
 94. Wajima T, Fukumura K, Yano Y et al (2002) Prediction of human clearance from animal data and molecular structural parameters using multivariate regression analysis. J Pharm Sci 91:2489–2499
 95. Chow H-H, Tolle KM, Roe DJ et al (1997) Application of neural networks to population pharmacokinetic data analysis. J Pharm Sci 86:840–845
 96. Singh KP, Gupta S, Rai P (2013) Predicting acute aqueous toxicity of structurally diverse chemicals in fish using artificial intelligence approaches. Ecotoxicol Environ Saf 95: 221–232
 97. Gong Z, Xia B, Zhang R et al (2008) Quantitative structure-activity relationship study on fish toxicity of substituted benzenes. QSAR Comb Sci 27:967–976
 98. Okey RW, Martis MC (1999) Molecular level studies of the origin of toxicity: determination of key variables and selection of descriptors. Chemosphere 38:1419–1427
 99. Halberstam NM, Baskin II, Palyulin VA et al (2002) Quantitative structure-conditions-property relationship studies. Neural network modelling of the acid hydrolysis of esters. Mendeleev Commun 12(5):185–186
 100. Dewhurst I, Renwick AG (2013) Evaluation of the threshold of toxicological concern (TTC)—challenges and approaches. Regul Toxicol Pharmacol 65:168–177
 101. Meng Y, Lin B-L (2008) A feed-forward artificial neural network for prediction of the aquatic ecotoxicity of alcohol ethoxylate. Ecotoxicol Environ Saf 71:172–186
 102. Fatemi MH, Jalali-Heravi M, Konuze E (2003) Prediction of bioconcentration factor using genetic algorithm and artificial neural network. Anal Chim Acta 486:101–108
 103. Zhao C, Boriani E, Chana A et al (2008) A new hybrid system of QSAR models for prediction bioconcentration factors (BCF). Chemosphere 73:1701–1707
 104. Devillers J, Pham-Delègue MH, Decourtye A et al (2002) Structure-toxicity modelling of pesticides to honey bees. SAR QSAR Environ Res 13:641–648
 105. Song JS, Moon T, Nam KD et al (2008) Quantitative structure-activity relationship (QSAR) studies for fungicidal activities of thiazoline derivatives against rice blast. Bioorg Med Chem Lett 18:2133–2142
 106. Fjodorova N, Vračko M, Tušar M et al (2010) Quantitative and qualitative models for carcinogenicity prediction for non-congeneric chemicals using CP NN method for regulatory uses. Mol Divers 14:581–594
 107. Singh KP, Gupta S, Rai P (2013) Predicting carcinogenicity of diverse chemicals using probabilistic neural network modelling approaches. Toxicol Appl Pharmacol 272: 465–475
 108. Dearden JC (2010) Expert systems for toxicity prediction. In: Cronin MTD, Madden JC (eds) *In silico* toxicology: principles and applications. Royal Society of Chemistry, London, pp 478–507
 109. Lapenna S, Fuart-Gatnick M, Worth A (2010) Review of QSAR models and software tools for predicting acute and chronic systemic toxicity. Available at http://ihcp.jrc.ec.europa.eu/our_labs/predictive_toxicology/doc/EUR_24639_EN.pdf/view. Accessed 25 Jan 2014
 110. Molecular Modeling Pro. Available at <http://www.chemsw.com/Software-and-Solutions/Laboratory-Software/Drawing-and-Modeling-Tools/Molecular-Modeling-Pro.aspx>. Accessed 25 Jan 2014
 111. Joback KG, Reid RC (1987) Estimation of pure-component properties from group-contributions. Chem Eng Commun 57: 233–243
 112. ADMET Predictor. <http://www.simulations-plus.com>. Accessed 28 Jan 2014
 113. ChemSilico. Available at www.chemsilico.com. Accessed 28 Jan 2014
 114. TOPKAT. Available at www.accelrys.com. Accessed 28 Jan 2014
 115. TerraBase. Available at www.terrabase-inc.com. Accessed 28 Jan 2014
 116. Kaiser KLE (2003) Neural networks for effect prediction in environmental health issues using large datasets. QSAR Comb Sci 22:185–190

Artificial Neural Network for Charge Prediction in Metabolite Identification by Mass Spectrometry

J.H. Miller, B.T. Schrom, and L.J. Kangas

Abstract

Collision-induced dissociation (CID) is widely used in mass spectrometry to identify biologically important molecules by gaining information about their internal structure. Interpretation of experimental CID spectra always involves some form of *in silico* spectra of potential candidate molecules. Knowledge of how charge is distributed among fragments is an important part of CID simulations that generate *in silico* spectra from the chemical structure of the precursor ions entering the collision chamber. In this chapter we describe a method to obtain this knowledge by machine learning.

Key words Mass spectrometry, Collision-induced dissociation, Charge transfer, Metabolite identification, Lipid metabolites

1 Introduction

Mass spectrometry (MS) is an analytical method that measures the mass-to-charge ratio (m/z) of charged particles. In some cases this quantity alone is sufficient to identify the molecules in the sample being analyzed. For biological samples that contain large molecules, fragmentation patterns in addition to precursor-ion mass are usually required for identification. Collision-induced dissociation (CID) is a commonly used method of fragmenting ions to obtain a fingerprint of fragment masses that is unique to the molecules under investigation. Precursor ions are accelerated toward, and collide with, neutral atoms in a gas such as helium. These collisions transform some of the kinetic energy into internal energy, which causes bonds to break by a mechanism similar to thermal dissociation. The fragment ions are then analyzed by the mass spectrometer in the usual way. Patterns of fragmentation obtained by this type of experiment are commonly referred to as MS/MS spectra.

Software tools like SEQUEST [1] and Mascot [2] are widely used in the application of MS/MS spectra for protein identification. Since the peptide bond between amino acids in a protein is

the most likely bond to break in CID, in silico MS/MS spectra can be derived from protein sequence without detailed knowledge of protein structure. For metabolites, the connection between molecular structure and CID is much more complex than for polypeptides; consequently, software tools for metabolite identification from MS/MS spectra have been more difficult to develop [3].

MetISIS [4] is a software package being developed to aid in the identification of metabolites. Given a MS/MS spectrum acquired from an unknown sample, MetISIS attempts to match the spectrum to an entry in a database of in silico spectra developed from the chemical structure of metabolites. Most of the entries in the database can be ignored because their mass is significantly different from the mass of the precursor ions fragmented to produce the MS/MS spectrum of the unknown. If the masses of the unknown metabolite and the database candidates are within a specified tolerance, then the similarity of CID spectra is quantified in various ways, including Pearson correlation coefficient (PCC). Requiring similarity of precursor-ion mass and PCC near unity produces a shortlist of candidate molecules with a high probability of being the unknown metabolite.

The success of this identification technique depends on the quality of the database of MS/MS spectra. It should contain a large number of metabolites for which high-resolution CID spectra are available. Populating such a database with experimental MS/MS spectra is both time-consuming and costly. To circumvent this difficulty, in silico databases, such as LipidBlast [5], have been developed. MetISIS [4] generates in silico MS/MS spectra of metabolites based on their molecular structure, such as those found in the LIPID Metabolites and Pathways Strategy (LIPID MAPS) database [6] and illustrated in Fig. 1 for phosphatidylcholine 18:0/18:0 (*see Note 1*).

The simulation of CID that MetISIS [4] performs to generate an MS/MS spectrum for a metabolite of known chemical structure has two components: (1) prediction of which bonds are most likely to break in a collision and (2) prediction of which fragments will continue to carry charge and be detected by the mass spectrometer.

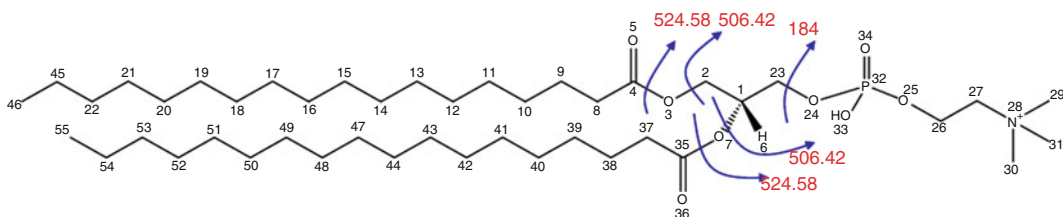


Fig. 1 Molecular structure of phosphatidylcholine 18:0/18:0. Arrows mark bond cleavages in collision-induced dissociation (CID) that generate the fragments most frequently observed in a linear ion-trap mass spectrometer

MetISIS uses machine learning by artificial neural networks (ANN) to make both types of prediction. The charge-prediction problem is the main focus of this chapter.

Collision-induced dissociation (CID) is a slow fragmentation process, which means that most collisions change the internal energy of ions without producing fragmentation. Consequently, ions have the opportunity to relax to a state where the location of charge minimizes their free energy. When the precursor ion contains an atom like nitrogen with a low ionization potential, charge is likely to remain localized during the dissociation process and the question of which fragment is charged reduces to which fragment has the nitrogen atom. Dissociation events of this type are commonly referred to as “neutral loss.”

Three mechanisms of neutral loss are marked by the arrows in Fig. 1. Due to the symmetry of phosphatidylcholine 18:0/18:0, neutral loss of a 266 Da tail can occur by cleavage of C-O bonds between C4 and O3 and between C36 and O7. Also due to symmetry, neutral loss of a 284 Da tail can occur by breaking of bonds between C2 and O3 and between C1 and O7. These neutral losses give rise to ions of mass 524 and 506 Da, respectively. The third mechanism of neutral loss shown in Fig. 1 is loss of both tails by cleavage of the bond between C23 and O24, which produces an ion of 184 Da.

Production of the 184 Da ion by loss of both tails is by far the most frequent event in the fragmentation of phosphatidylcholine 18:0/18:0. It is 93 times more frequent than loss of the 284 Da tail and 116 times more frequent than loss of the 266 Da tail. Cleavage of the bond between C23 and O24 can cause charge transfer, resulting in a zwitterion (an overall neutral fragment) of mass 183 Da and an ion of mass 608 Da, but the neutral loss of both tails by the same bond cleavage is almost 3,000 times more frequent. Mechanisms of neutral loss are more complex in lipids that contain more than one nitrogen atom, and in some cases, the relative stability of positively charged atoms is not clear from the molecular structure. In future research, computational chemistry methods will be used to investigate the relative stability of sites of positive charge.

A schematic of the CID simulation and associated machine-learning algorithm developed by Lars Kangas as part of his doctoral thesis research is shown in Fig. 2.

A Monte Carlo simulation of the CID process is performed on a large number of identical precursor ions of known molecular structure. The physical model of dissociation induced by ion-atom collisions must be consistent with the characteristics of the mass spectrometer used to acquire the MS/MS spectra for machine learning and metabolic analysis. Initially, we chose to simulate CID in a linear ion-trap mass spectrometer because the acceleration of precursor ions in that type of spectrometer is a resonance process (*see* **Note 2**). Fragments

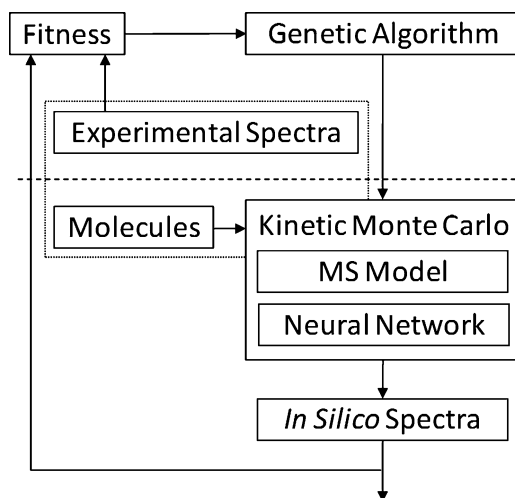


Fig. 2 Flow chart of the algorithm developed to learn the relative frequency of bond cleavages in collision-induced dissociation (CID) from experimental data. When training is complete, the three components above the *dotted line* are ignored and in silico CID spectra can be generated from the molecular structure of lipid metabolites. The charge-prediction component is a separate machine-learning task using the methods described in this chapter. (Reproduced from [4] with permission from Oxford University Press)

of precursor-ion dissociation are off resonance and unlikely to undergo secondary fragmentation, which allowed us to focus on the chemistry of the precursor ion (i.e., how strong its bonds are and where charge is likely to be located).

Since neutral fragments are not detected, an algorithm to generate in silico MS/MS spectra must have a component to predict which fragment after a collision-induced dissociation continues to carry the charge of the precursor ion. This charge-prediction component is separate from the neural network used to learn the relative frequency of bond cleavages. The method section of this chapter describes a machine-learning approach to charge prediction that uses the same input vectors as the ANN used to predict frequencies of bond cleavage.

In the machine-learning phase, weights of the neural network component in Fig. 2 are optimized through a genetic algorithm [7]. The exemplar database for this phase of machine learning was derived from experimental CID spectra acquired on a linear ion-trap spectrometer with pure samples of the type of metabolites under investigation. Lipid metabolites were chosen to demonstrate a proof of concept because their CID spectra were known to be relatively simple. High-quality CID spectra were obtained for about 100 molecular species distributed more or less uniformly over the major classes of lipid metabolites (*see Note 3*).

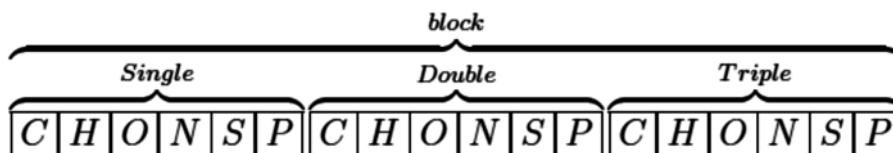


Fig. 3 A block in an input vector displaying 18 options for how an atom is bonded to its successor in the tree derived from of a breadth-first search of a fragment from its root atom

Both the position (m/z) and intensity of peaks in experimental MS/MS spectra were used to find patterns of bond cleavage, giving rise to detectable charged fragments of the precursor ion. Spectra of singly charged fragments provide most of the data for pattern discovery. After weights in the neural network component converged, the process of generating in silico CID spectra for a large database of metabolites of known chemical structure can begin. The LIPID MAPS database [6] contains more than 22,000 such structures. This application of the algorithm illustrated by Fig. 2 does not require the components above the dotted line.

A method to input a molecular structure into the Monte Carlo simulation of CID is fundamental to the machine-learning process and subsequent generation of in silico spectra by MetISIS. Faulon et al. [8] showed that molecules could be represented by undirected graphs of atoms and bonds stored in adjacency matrices. Tree structures and vectors for machine learning can be derived from adjacency matrices [9] by selecting one atom to be the root and constructing the predecessor graph of a breadth-first search [10] to a specified depth. MetISIS [4] uses vectors, made up of blocks like that shown in Fig. 3, to encode the fragments that are produced when a bond in the chemical structure of the precursor ion is cleaved during CID.

Most lipid metabolites are made up of six atom types: carbon (C), hydrogen (H), oxygen (O), nitrogen (N), sulfur (S), and phosphorus (P). An encoding scheme that records how a given atom is connected to another atom can be constructed with a block that is a vector with 18 components. The first six components are CHONSP for single-bond connections, the next six components are for double-bond connections, and the final six components are for triple-bond connections. The root atom of a fragment is the atom involved in the bond that was cleaved to produce the fragment. A bonding pattern that connects the root atom to any other atom in the fragment results in a series of CHONSP components between the selected atom and the root. Since all of the six atom types and three bond types are possible, the upper bound on the number of paths for a distance (in bonds), D , away from the fragmentation site is $(6)(6 \times 3)^{D-1}$. We find that $D=8$ is adequate to capture the chemical neighborhood of bonds in lipid metabolites (see Note 4).

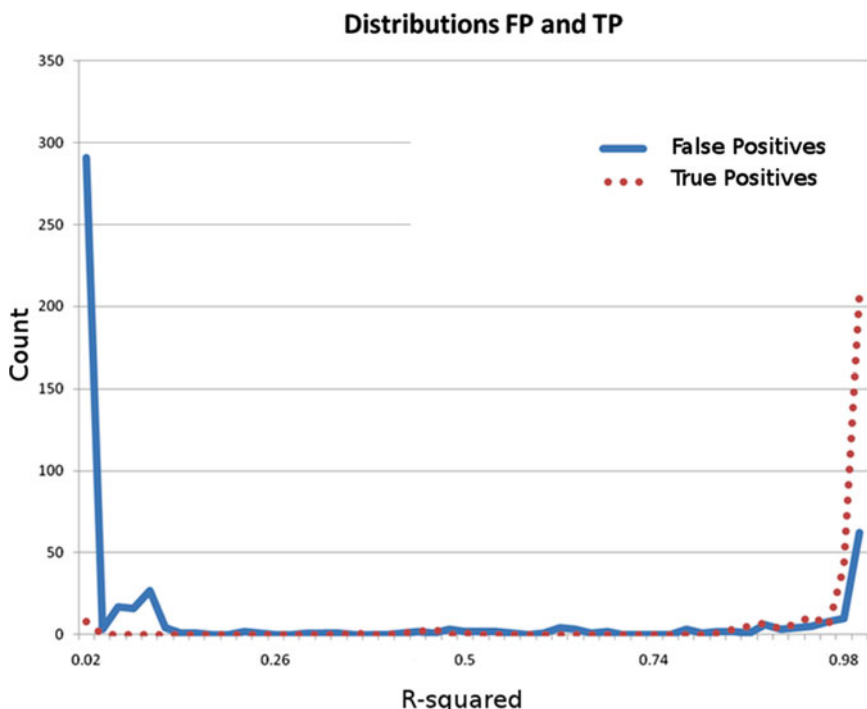


Fig. 4 The distribution of true- and false-positive identifications as a function of Pearson correlation coefficient (R^2) in a test of the MetISIS algorithm that did not contain the charge-prediction component (see Fig. 2). While the algorithm performed reasonably well, the approximately 50 false positives with R^2 near unity suggested that charge prediction was essential for accurate simulation of MS/MS spectra

In cases where the major peaks in the MS/MS spectrum of a metabolite are well separated and the spectrum also contains a peak for the precursor ion, charge prediction in simulations of CID in a linear ion-trap spectrometer is of minor importance. Accurate predictions of the frequency of bond cleavage can be used to generate potential fragment masses for the major peaks in the MS/MS spectrum. Given a close alignment of a predicted fragment mass with a peak in the experimental spectrum, the mass of the precursor ion can be used to determine the mass of the corresponding neutral fragment, thereby allowing its peak to be eliminated from the *in silico* spectrum. Figure 4 shows the results of an early test of MetISIS performance without charge prediction.

Without charge prediction, the number of false-positive identifications is significant even when the PCC is close to unity. This undesirable result could be traced directly to the cases where simulation of CID predicted masses of neutral fragments that were too close to peaks in experimental spectra to distinguish them from charged fragments. These results clearly showed a need to predict which fragments are charged after CID events.

2 Methods

2.1 Construction of Training and Testing Sets

1. The construction of training and testing sets begins by acquiring experimental CID spectra from pure samples of specified metabolites. The training/testing process for charge prediction in CID of lipid metabolites was based on MS/MS experiments with 94 lipids of known chemical structure (*see Note 3*).
2. The molecular structure of each lipid metabolite with an experimental CID spectrum was provided by the chemical supplier.
3. Input vectors (*see Note 4*) for all bond cleavages that might be associated with peaks in experimental spectra were generated from the molecular structures. Bonds to hydrogen atoms for which cleavage would reduce the precursor-ion mass by only 1 Da were ignored. Bonds between carbon atoms in tails of lipids were also ignored because it seemed likely that any experimental peak could be explained by this type of fragmentation.
4. To generate the output label, the left and right mass fields in the <meta data> block of the encoded vector (*see Note 4*) were used to determine the mass for each fragment and the total mass of the lipid. The experimental CID spectrum of the lipid was then examined to see if it contained a peak at either of these two masses. In the absence of such a peak, the potential input vectors for a charged and a neutral fragment were removed from consideration. If there was a peak in the experimental spectrum at only one of potential fragment masses, then the exemplar with that mass in left-hand fragment was labeled 0.8 and the exemplar with that mass as right-hand fragment was labeled 0.2. If peaks were found at both masses, then the mass with the higher intensity was considered the winner and the output of the pair of exemplars was labeled accordingly. A tolerance of ± 2 Da in matching peak masses with fragment masses was used. For the 44 lipids selected to generate a training set, this process gave rise to 658 pairs of exemplars. For the 50 lipids selected to generate a testing set, the same process gave rise to 2,183 pairs of exemplars.
5. Both training and testing sets contained labeled exemplars from all 15 classes of lipids (*see Note 3*) for which experimental MS/MS spectra were available.

2.2 ANN Structure

As shown in Fig. 5, the ANN architecture had an input layer with the 1,270 nodes required for input vectors that characterize the chemical environment around bond-cleavage sites (*see Note 4*) in the lipids with experimental spectra (*see Note 3*).

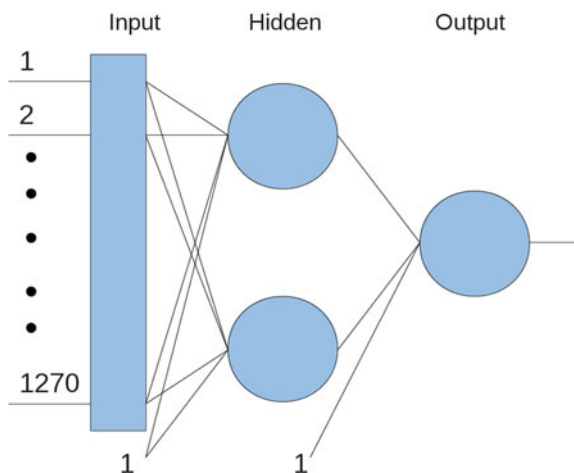


Fig. 5 Structure of the neural network used to develop charge prediction for a set of 94 lipid metabolites distributed over 15 classes (*see Note 3*). The input vector size (1,270) reflects the amount of compression that was possible with this set of molecules (*see Note 4*). Bias nodes in the input and hidden layers are indicated by their constant value of unity

Between this input layer and the single node of the output layer is a hidden layer with two nodes. The number of nodes in the hidden layer was chosen equal to the number of classes the input data needs to distinguish, which are “left fragment charged” and “left fragment neutral.” The bias nodes of the input and hidden layers are indicated by their constant value of 1. Although charge prediction could be treated as classification, we chose to treat it as multivariable nonlinear regression. Sigmoidal transformations are used to generate an output between 0 and 1. Back-propagation of the difference between this output and the label of exemplars was used to optimize the weights of the neural network with a learning rate of 0.25 and momentum coefficient of 0.95.

2.3 Training Results

The root-mean-square deviation (RMSD) between output values and exemplar labels was selected as a measure of training convergence. The lower curve in Fig. 6 shows the convergence achieved by 500 epochs of training. These results reveal that convergence is rapid, as would be expected for a neural network with only two nodes in a single hidden layer. The “elbow” in the RMSD for the training set occurs after about 25 epochs, and the RMSD decreases very slowly after about 200 epochs.

The upper curve in Fig. 6 shows the RMSD of output values relative to the label on exemplars of the test set, which was calculated after every epoch of training. The RMSD relative to the test set follows the RMSD of the training set for only about 10 epochs, after which it deviates sharply, remains approximately constant, but increases slightly as the RMSD of the training set

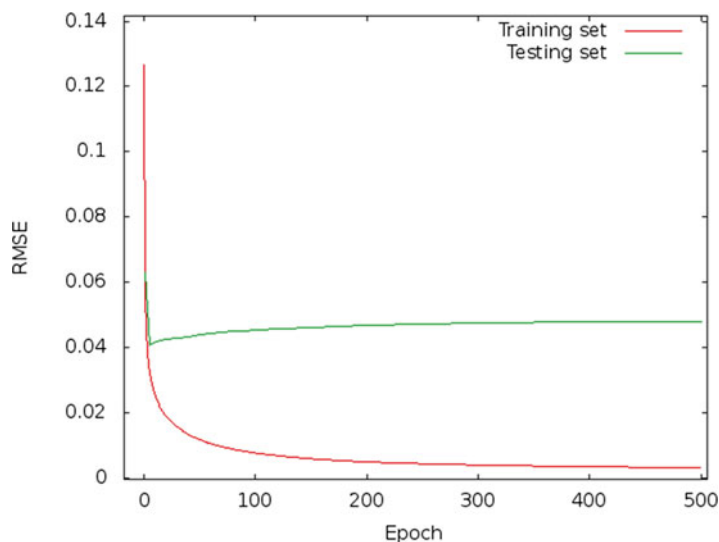


Fig. 6 Root-mean-square deviation (RMSD) of network output from exemplar labels in training and testing sets as a function of cycles of refinements on the training set

Table 1

Accuracy of charged-fragment prediction in a test set derived from 50 lipid metabolites distributed over 15 classes (*see Note 3*) after 500 cycles of ANN refinement on a similar training set

Correct overall percentage	99.3 %
Incorrect overall percentage	0.7 %
True positives	2,170 (99.4 %)
True negatives	2,166 (99.2 %)
False positives	13 (0.6 %)
False negatives	17 (0.7 %)

continues to decrease. We interpret this behavior as an indication that there is a small amount of “overfitting” with 500 epochs of training. Details of the performance of the model on the test set, after 500 epochs of training, are summarized in Table 1.

2.4 Future Research

Results, shown in Table 1, of the machine-learning experiment with data on 94 lipid metabolites are encouraging; nevertheless, work is in progress to improve the ANN model for prediction of charged fragments. This work includes the following: (1) additional experimental CID spectra of non-lipid metabolites, (2) inclusion of secondary fragmentation so that data from instruments other than ion-trap spectrometers can be modeled, (3) larger training

sets with inclusion of cross validation, and (4) more combinations of training and test sets with the use of random numbers to ensure uniform representation of all types of metabolites.

3 Notes

1. The 18:0/18:0 notation refers to the number of carbon atoms and number of double bonds that are in each hydrophobic tail attached to the lipid's head group. The unlabeled atoms (i.e., numbered only) are understood to be carbon. Hydrogen atoms needed to saturate chemical bonding are not shown. More information about the LIPID MAPS database and its nomenclature can be found at http://www.lipidmaps.org/data/classification/lipid_cns.html.
2. Details regarding linear ion-trap mass spectrometers and CID energy calculations can be found in the supplemental material of Kangas et al. [4].
3. Training and testing exemplar sets were based on experimental data and molecular structures for the following lipids: *phosphatidylcholine* (14:0/14:0, 14:0/16:0, 16:0/16:0, 16:1/16:1, 17:0/17:0, 18:0/18:0, 18:1/18:1, 18:2/18:2, 18:3/18:3, 20:1/20:1, 20:4/20:4, 23:0/23:0), *phosphatidylcholine ether* (13:0/13:0, 18(P)/18:1, 18:1/18:1, 18(P)/20:4), *lysophosphatidylcholine* (14:0, 15:0, 16:0, 17:0, 17:1, 18:0, 18:1), *phosphatidylethanolamine* (12:0/12:0, 15:0/15:0, 16:1/16:1, 16:0/18:1, 17:1/17:1, 18:0/18:0, 18:0/18:1), *lysophosphatidylethanolamine* (14:0, 16:0, 18:0, 18:1), *phosphatidylserine* (12:0/12:0, 14:0/14:0, 16:0/18:2, 17:0/17:0, 18:0/18:0, 18:0/18:1, 18:0/18:2), *lysophosphatidylserine* (16:0, 18:0, 18:1), *sphingomyelin* (d18:1/12:0, d18:1/16:0, d18:1/17:0, d18:1/18:0, d18:1/24:1), *ceramide* (d18:1/12:0, d18:1/17:0, d18:1/18:0, d18:1/20:0, d18:1/22:0, d18:1/24:0), *ceramide 1-phosphate* (d18:1/8:0, d18:1/12:0, d18:1/16:0, d18:1/18:1, d18:1/24:0), *hexaceramide* (d18:1/8:0, d18:1/12:0, d18:1/16:0, d18:1/24:0, d18:1/24:1), *dihexaceramide* (d18:1/8:0, d18:1/12:0), *cholesterol*, *cholesterolester* (18:2, 18:3, 20:4, 20:5), *diglycerol* (17:0/17:0, 20:0/20:0, 20:2/20:2, 20:4/20:4, 20:5/20:5), and *triglycerol* (14:0/16:1/14:0, 15:0/18:1/15:0, 16:0/18:0/16:0, 16:0/18:1/22:6, 16:1/17:1/17:2, 16:1/18:1/18:2, 16:1/18:1/20:4, 17:0/17:1/17:0, 18:1/18:1/18:2, 20:0/20:1/20:0, 20:2/18:3/20:2, 20:4/18:2/20:4, 20:5/22:6/20:5).
4. The complete encoded vector has the form <Meta data> <Block 1><Block 2>...<Block 8>. The <Meta Data> block, which includes information about the lipid metabolite under

investigation, is not part of the input attributes but is referred to in the generation of the output label. MetISIS uses an undirected graph derived from the chemical structure to represent a metabolite. When a bond is broken in the CID simulation, the two atoms that were involved in the bond become root nodes of a disconnected graph consisting of two subgraphs. One subgraph is denoted the left fragment and the other one is denoted the right fragment. The CHONSP count for each atom within eight bonds of the root atoms in the fragments is easily obtained by a breadth-first search [10].

Each exemplar used in machine learning is associated with a particular bond cleavage in a specified metabolite. CHONSP counts for both fragments generated in the cleavage are retained in the same input vector. Since each peak in the experimental CID spectrum contributes two exemplars to the charge-prediction exemplar set, a charged and a neutral fragment, a second input vector is generated for each bond cleavage by interchanging the left and right components. The output label of each exemplar identifies the charge (1 or 0) of the left component. The equal number of charged and neutral exemplars in the training set prevents the encoding scheme from introducing bias in charge prediction.

In addition to this basic encoding, MetISIS performs two more processing steps on the encoded vectors. The first is to compact the vectors. This is done by examining a large set of molecules, encoding them as described above and then removing CHONSP components that are always zero. For example, if there is no triple-bonded P in any of the molecules at a given distance N from the root atoms, then the triple-bonded P will be removed from the corresponding $\langle \text{BlockN}_P \rangle$ components. This mode of compression adapts input vectors to a particular set of metabolites. For lipid metabolites in **Note 3**, the compressed input vectors have 1,270 components. The second step is normalization of the data. After compression, the remaining CHONSP components are normalized to values between 0 and 1.

Acknowledgments

Results reported in this chapter were part of masters- and doctoral-degree programs in computer science at Washington State University Tri-Cities for B. T. Schrom and L. J. Kangas, respectively. J. H. Miller received partial support from the Low Dose Radiation Research Program of the US Department of Energy.

References

1. Eng JK, McCormack AL, Yates JR (1994) An approach to correlate tandem mass-spectral data of peptides with amino-acid-sequences in a protein database. *J Am Soc Mass Spectrom* 5:976–989
2. Perkins DN, Pappin DJ, Creasy DM et al (1999) Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis* 20: 3551–3567
3. Scheubert K, Hufsky F, Bocker S (2013) Computational mass spectrometry for small molecules. *J Cheminform.* doi:[10.1186/1758-2946-5-12](https://doi.org/10.1186/1758-2946-5-12)
4. Kangas LJ, Metz TO, Isaac G et al (2012) *In silico* identification software (ISIS): a machine learning approach to tandem mass spectral identification of lipids. *Bioinformatics* 28: 1705–1713
5. Kind T, Liu KH, Lee do Y et al (2013) LipidBlast *in silico* tandem mass spectrometry database for lipid identification. *Nat Methods* 10:755–758
6. LIPID Metabolites and Pathways Strategy (LIPID MAPS) program. www.lipidmaps.org
7. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, MA
8. Faulon JL, Visco D, Roe D (2005) Enumerating molecules. In: Lipkowitz KB et al (eds) *Reviews in computational chemistry*, vol 21. Wiley, Hoboken, NJ, pp 209–286
9. Schietgat L, Ramon J, Bruynooghe M et al (2008) An efficiently computable graph-based metric for the classification of small molecules. In: *Proceedings of the 11th international conference on discovery science (LNAI 5525)*, pp 197–209
10. Cormen TH, Leiserson CE, Rivest RL et al (2009) *Elementary graph algorithms*. In: *Introduction to algorithms*, 3rd edn. MIT Press, Cambridge, pp 594–602

Prediction of Bioactive Peptides Using Artificial Neural Networks

David Andreu and Marc Torrent

Abstract

Peptides are molecules of varying complexity, with different functions in the organism and with remarkable therapeutic interest. Predicting peptide activity by computational means can help us to understand their mechanism of action and deliver powerful drug-screening methodologies. In this chapter, we describe how to apply artificial neural networks to predict antimicrobial peptide activity.

Key words Artificial neural network, Peptide, Antimicrobial, Screening, Drug

1 Introduction

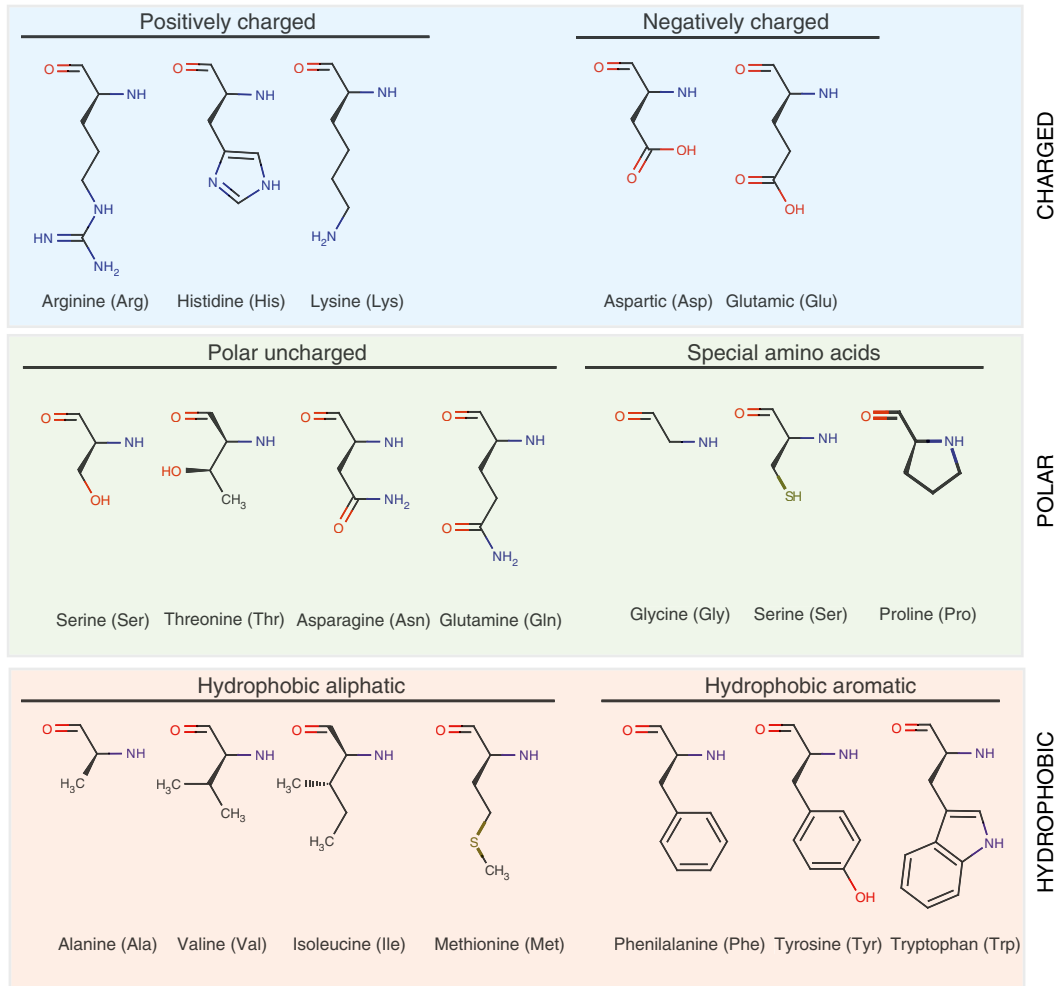
1.1 Peptides and Their Use in Medicinal Chemistry

Peptides are natural or synthetic polymers composed of amino acids linked by an amide bond. The different amino acids have different physicochemical characteristics. In nature, most proteins and peptides are built from 20 natural amino acids, which are depicted in Fig. 1a.

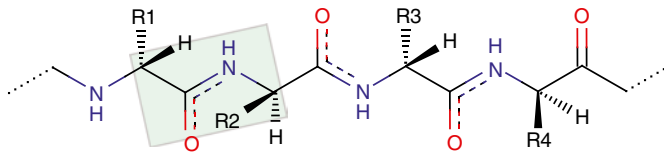
The peptide chain has a reduced conformational flexibility due to a significant double-bond character between the carbonyl carbon and the nitrogen (Fig. 1b). The peptide chain is mainly planar and conformational changes are restricted, with flexibility mainly limited to rotations around the α -carbon atoms [1]. In addition, physicochemical constraints, such as side-chain volume and charge, contribute to peptide structure. Therefore, peptides can be either unfolded or structured in α -helix, β -strands with loops and coils (Fig. 1c), or a combination of them. Globally, the amino acid composition, together with the three-dimensional structure, is what determines peptide action.

Electronic Supplementary Material: The online version of this chapter (doi: [10.1007/978-1-4939-2239-0_7](https://doi.org/10.1007/978-1-4939-2239-0_7)) contains supplementary material, which is available to authorized users

a



b



c

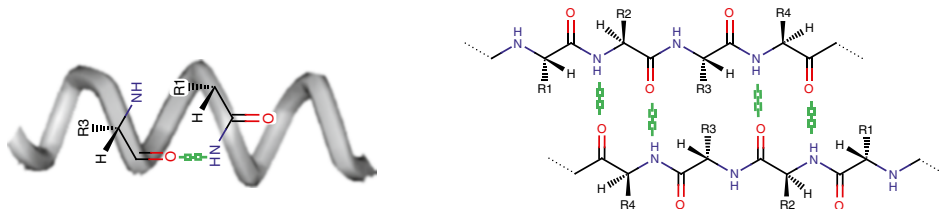


Fig. 1 Peptide composition and structure. Peptides are built from amino acids with different chemical properties (a) linked by amide bonds. The peptide chain is mainly planar, with flexibility mostly limited to rotations around the α -carbon atoms (b). Despite these constraints, peptides can adopt different secondary structures (c) including α -helix and β -strands with loops and coils

Table 1
Selected examples of natural peptides

Peptide name	Peptide sequence	Function
Glutathione	γ -Glu-Cys-Gly	Antioxidant
Angiotensin II	Asp-Arg-Val-Tyr-Ile-His-Pro-Phe	Vasoconstrictor
Bradykinin	Arg-Pro-Pro-Gly-Phe-Ser-Pro-Phe-Arg	Vasodilator
Calcitonin	Cys-Gly-Asn-Leu-Ser-Thr-Cys-Met-Leu-Gly-Thr-Tyr-Thr-Gln-Asp-Phe-Asn-Lys-Phe-His-Thr-Phe-Pro-Gln-Thr-Ala-Ile-Gly-Val-Gly-Ala-Pro	Calcium metabolism
Somatostatin	Ala-Gly-Cys-Lys-Asn-Phe-Phe-Trp-Lys-Thr-Phe-Thr-Ser-Cys	Growth and cell division
Melittin	Gly-Ile-Gly-Ala-Val-Leu-Lys-Val-Leu-Thr-Thr-Gly-Leu-Pro-Ala-Leu-Ile-Ser-Trp-Ile-Lys-Arg-Lys-Arg-Gln-GlnNH ₂	Antimicrobial peptide
Enkephalin	Tyr-Gly-Gly-Phe-Met/Leu	Nociception

There are different sources of natural peptides of varying complexity with different functions in the organism [2]. For example, glutathione is an antioxidant peptide that protects cells from free radicals [3], and angiotensin is a hormone that acts as a vasoconstrictor, increasing blood pressure [4]. Some well-known natural peptides are listed in Table 1.

The number of possible combinations of amino acids into peptides is huge and depends upon the peptide length. For example, for a peptide with 15 amino acids, we could build 20^{15} different molecules, meaning more than 10^{19} different combinations.

Advanced synthetic strategies now allow us to build hundreds or even thousands of peptides easily [5–8]. Thanks to these technologies, we are not restricted to natural peptide templates and can design new peptides with novel functions that do not occur in nature but have relevant applications in pharmacology and medicine. Given the number of possible different combinations of amino acids, the biomedical applications of peptides are nearly infinite.

Unfortunately, we are far from exploring the effects of millions and millions of amino acid combinations. Even though modern screening methods are very versatile and widespread, testing large libraries of compounds can be time intensive, especially when cellular or animal models are required [9, 10]. In these situations, mathematical models are essential to narrow down the experimental work required. In addition, these models help us to understand peptide activity and can be used to dissect the molecular properties of active compounds and discover the receptors and pathways in which they are involved [11]. For all these reasons, mathematical models are essential in peptide design and have real relevance in the pharmaceutical industry for the development of new antibiotics or anti-inflammatory drugs.

1.2 Use of Prediction Models in the Search for New Drugs

As detailed in Table 1, peptides can have diverse functions, which are correlated with amino acid composition and structure. It is of particular interest to develop computational tools that are able to predict active peptides, based on their physicochemical and structural characteristics. In this chapter we will focus on antimicrobial peptide classification, but the methods and techniques described here are widely applicable. Therefore, the same strategies can be applied in the prediction of other classes of biologically active peptides.

We aim to describe in this chapter how one can use artificial neural networks (ANNs) to predict peptide activity. We will describe how to build and curate datasets, train an ANN, and validate the results in the software package R.

2 Materials

- 1, 2. Software package R (*see Note 1*) and software package RStudio (*see Note 2*).
3. Libraries (Interpol, corrplot, caret, RSNNS, RColorBrewer, SDMTools, devtools, reshape) (*see Note 3*).
4. ANN script (Supplementary Materials).
5. Peptide dataset AMPlist (Supplementary Materials).
6. Descriptor list (Supplementary Materials).

3 Methods

In this chapter we will describe the basics of peptide classification using ANNs. In our example, we will develop a tool to classify peptides as antimicrobial or non-antimicrobial.

The protocol is structured in the following steps:

1. Dataset building
2. Descriptor selection
3. Network training
4. Model validation
5. Network visualization

All these steps are covered in the following points with tips and hints that can be used to apply the present model (or an extension of it) to analyze similar problems.

3.1 Build Datasets

Though it might appear obvious, it is worth stating that the results of computational analysis are strongly dependent on the quality of the databases provided. For a supervised learning strategy such as the

one described in this chapter, we need to provide a list of peptides (input) and a measure of the biological property of interest for each peptide (output). In the classification example we are presenting here, the outputs are in a yes/no format, denoting that the peptide has or does not have antimicrobial properties respectively.

When using data from one's own laboratory, it is possible to fix a set of criteria that will be applied to all samples, which are then processed using the same protocols. However, when obtaining data from the literature, caution is required to ensure consistency. Previously published data from other laboratories may have been gathered using many different protocols. For example, different media for bacteria cultures can affect the assessment of antimicrobial properties. Whenever possible, it is wise to use only values obtained from the same method.

When this is not possible, it may be necessary to include data from other sources. For antimicrobial peptides and bioactive peptides in general, there are public databases that gather information about peptides. In general, the classification criterion in these databases is based on bibliography searches, without taking into account the concerns described above. It is important to keep in mind that this can affect the performance of the calculations and that as a result less accurate prediction systems may be obtained. We describe in Subheading 3.1.1 how to build a positive dataset from public databases.

It is also important to avoid overrepresentation of particular peptide families. When a single family of peptides represents a large part of the dataset, this may bias the results. Always check for peptide similarity and use a filter as described in Subheading 3.1.2 to remove large groups of similar peptides.

Though positive databases can be difficult to obtain, more serious problems can arise from the need for a negative database, because it is hard to find a detailed list of negative results. For example, numerous antimicrobial peptides can be found in the literature but only a few examples of non-antimicrobial peptides are described. This is a major problem, as it may bias the training process.

Frequently, large negative databases may not be built because not enough data is available. Therefore, an alternative method is required to obtain a suitable negative dataset. This is the case of antimicrobial peptides. Though public databases contain around 2,000–5,000 antimicrobial peptides (positive database), it is impossible to build an experimentally validated negative dataset of similar size for non-antimicrobial peptides. In Subheading 2.1.2 we describe a method to build such a dataset. Though this approach may be useful in antimicrobial peptides, it may not be applicable to other bioactive peptides, and particular strategies may need to be found, depending on the type of study.

Table 2
Databases of antimicrobial peptides

Database	Number of entries	Data source	References
APD(APD2)	2,338	Pubmed	[12]
BACTIBASE (BACTIBASE 2)	177	Uniprot Pubmed	[13]
CAMP	4,020	NCBI	[14]
Defensins knowledgebase	350	SciFinder Scholar Uniprot EMBL/GenBank	[15]
PenBase	110	Pubmed EMBL/GenBank	[16]
PhytAMP	271	Uniprot Pubmed	[17]
RAPD	179	Pubmed	[18]

3.1.1 Build the Positive Dataset

The positive dataset is built from a list of peptides which have been shown experimentally to have the property of interest. The elements of the dataset can be retrieved from experiments done in the laboratory or from public databases. The dataset consists of a text file built using either a text engine or a spreadsheet containing the list of peptides. It should be saved as a non-formatted text file (.txt extension) and the peptides should be delimited by a carriage return.

For antimicrobial peptides, useful databases are listed in Table 2. In our working example, we have selected the cationic peptides from the APD2 database (*see* AMPlist.txt included in the Supplementary Material).

3.1.2 Build the Negative Dataset

This dataset contains a list of peptides validated to not have the property of interest. As was the case with the positive dataset, it can be built based on experiments performed in the laboratory, or by relying on public databases, but also by extrapolation as described in the introduction to this section. The dataset consists of a text file as described in Subheading 2.1. Like the positive dataset, it should be saved as a non-formatted text file (.txt extension) and the peptides should be delimited by a carriage return.

To build our example negative dataset, we have used the Uniprot server (<http://uniprot.org>) to search for non-antimicrobial, nontoxic, and non-antibiotic peptides, with restricted length (from 5 to 45 amino acids). Only Uniprot-refined entries were selected. A list of >3,000 peptides was finally included in the negative dataset. As these peptides may have high sequence similarity, we used CD-Hit (http://weizhong-lab.ucsd.edu/cdhit_suite/, [19])

to reduce sequence redundancy (90 %). Our final negative dataset contained 2,234 peptides. We have also manually curated the database to delete protamines and histones that are erroneously not categorized by Uniprot as antimicrobials (*see* AMPlist.txt included in the Supplementary Material).

3.1.3 Merge Datasets and Add the Class Vector

Merging the datasets is an optional but convenient step. To merge the datasets, we have merely copied the text contained in the positive and negative datasets (in that order) into a new text file, called AMPlist (*see* AMPlist.txt included in the Supplementary Material).

To train the classification system, peptides in the list must be labeled as active or inactive. To do that, active peptides were labeled 1 and inactive peptides labeled 0. While a list of 0 s and 1 s can be built using a spreadsheet and then loaded into the training software, we have for the sake of convenience included instead a code line to build the class vector (*see* below).

3.1.4 Script Explanation

The script provided uses the R function `read.table()` to load the peptide list file. In the example, the peptide list called AMPlist.txt has a list of 2,048 antimicrobial peptides and 2,234 non-antimicrobial peptides, built as described above. To load a new list instead of the example, just change the name in the script (line 13, bold text):

```
peptide_list<- read.table("AMPlist.txt", quote="\")
```

The class vector, containing an appropriate number of 0 s and 1 s, is created in line 19 of the script:

```
y<- c(rep(1,2048),rep(0,2234))
```

The first parenthesis contains the number of active peptides and the second parenthesis the number of inactive peptides. Numbers can be changed but must be consistent with the list supplied as peptide_list.

3.2 Select Descriptors

Molecular descriptors are numerical values that characterize properties of the molecule of interest [20]. For example, a simple molecular descriptor could be the molecular weight or the charge of the peptide. There are many types of molecular descriptors, but they can be classified as either (1) structure-based, which depend on the structural properties and include topological or geometrical descriptors, or (2) property-based, which are experimentally or theoretically determined values and include properties such as hydrophobicity or accessible surface area.

Structure-based descriptors can refer to different levels of organization. For example, in peptides we can refer to (1) the amino acid sequence, (2) the secondary structure of particular segments in the peptide, and (3) the complete tertiary structure of the peptide.

Most commonly, a three-dimensional structure is not available, so no experimentally validated values can be obtained for structure-based descriptors. At this point, structure modeling can be used, but this is time consuming and generally not reliable. Therefore, in this chapter we will only consider sequential descriptors and simple predictions of secondary structure elements (e.g., the propensity of the peptide to adopt an α -helix structure).

3.2.1 Descriptor Calculation

One of the most extensive lists of descriptors for amino acids is encoded in the AAindex database [21, 22]. The AAindex database contains a selected list of 533 descriptors, based on amino acid physicochemical properties, substitution matrices, and statistical protein contact potentials.

The method of calculating properties based on the AAindex values is also flexible. The simplest strategy is to compute the average for all amino acids present in the peptide sequence. This approach is computationally inexpensive but may miss important details like the order in which the amino acids appear in the structure. For example, the peptide ASLP will have the same value as the peptide PALS, though the peptides may have completely different biological properties.

Another approach is to use autocorrelation, which takes into account the distribution of these properties along the sequence of peptides [23, 24]. There are several strategies to compute autocorrelation; one example is the Moreau-Broto autocorrelation:

$$AC_{MB} = \sum_{i=1}^{N-d} P_i P_{i+d}$$

where the property of interest is named P and the autocorrelation value is computed as the summation of products along the sequence for amino acids separated by a length d . For the example described before, assuming a distance d of 1 amino acid and the arbitrary values $A=1$, $S=2$, $L=3$, and $P=4$, we would have an $AC_{MB} = 1 \times 2 + 2 \times 3 + 3 \times 4 = 20$ for peptide ASLP and $AC = 4 \times 1 + 1 \times 3 + 3 \times 2 = 13$ for peptide PALS.

In the script provided in this chapter, we will use the average strategy, but autocorrelation can be also implemented in the script. To learn more about other methods to calculate autocorrelation, *see* refs. 23, 24.

Though it is possible to calculate all descriptors in the script provided, several limitations must be taken into account. First, the number of descriptors should be small compared with the number of examples provided for the training process to be accurate. For example, it is not suitable to use 400 descriptors if the peptide dataset has only 200 peptides. Second, using a disproportionate number of descriptors could easily lead to model over-fitting. Third, many descriptors are highly correlated with others, generating

redundancy in the data. For example, the AAindex list of descriptors contains several different measures of amino acid hydrophobicity. Fourth, the more complex the model, the more difficult it is to analyze the results. When using a reduced number of parameters, it is easy to understand the contribution of each of them to the model and to understand its biological relevance. When large numbers of parameters are used, their contribution tends to balance and details are easily lost (*see Note 4*).

3.2.2 Script Explanation

In the script provided, only a small subset of parameters [13] is used for prediction. To load a different subset of parameters, substitute the numbers given in lines 22 and 27 (highlighted in bold) by the corresponding numbers in the descriptor list (*see Supplementary Material*):

```
vector<- as.numeric(c("9", "37", "38", "39", "40", "69", "71",
  "96", "146", "151", "319", "321", "381", "401"))
rownames(ff)<- c("AA9", "AA37", "AA38", "AA39", "AA40",
  "AA69", "AA71", "AA96", "AA146", "AA151", "AA319",
  "AA321", "AA381", "AA401")
```

In lines 30–39, we implement a code for deleting highly correlated parameters. Therefore, those parameters that display a correlation index of 0.90 with another parameter are disregarded for the network training (*see Note 5*).

The threshold can be changed in the script, in line 33:

```
highlyCor<- findCorrelation(cor2, 0.90)
```

3.3 Develop the Artificial Neural Network Model

Artificial neural networks are computational models inspired in the brain neuronal system [25, 26]. The ANNs consist of inputs (representing the signal received by the neuron) that are multiplied by weights (representing the strength of the signal) and then mathematically transformed to produce the outputs (the response to the input signal). This is represented schematically in Fig. 2a.

In summary, ANNs can be represented as weighted directed graphs with neurons represented by nodes and connections represented by directed edges. Following this representation, ANNs can be classified as (1) feed-forward networks or as (2) recurrent (or feedback) networks. The difference between them is that the first does not contain any loops while the later does contain loops (Fig. 2b). In this chapter we will focus only on feed-forward networks and, concretely, on multilayer perceptron networks, a particular ANN in which neurons are organized in layers and are connected with unidirectional edges (Fig. 2c).

In this chapter, we aim to build ANNs that learn from given examples (known active and inactive peptides) to be able to predict new inputs (unknown peptides). This scheme is known as supervised learning because we provide a set of examples to help the network to

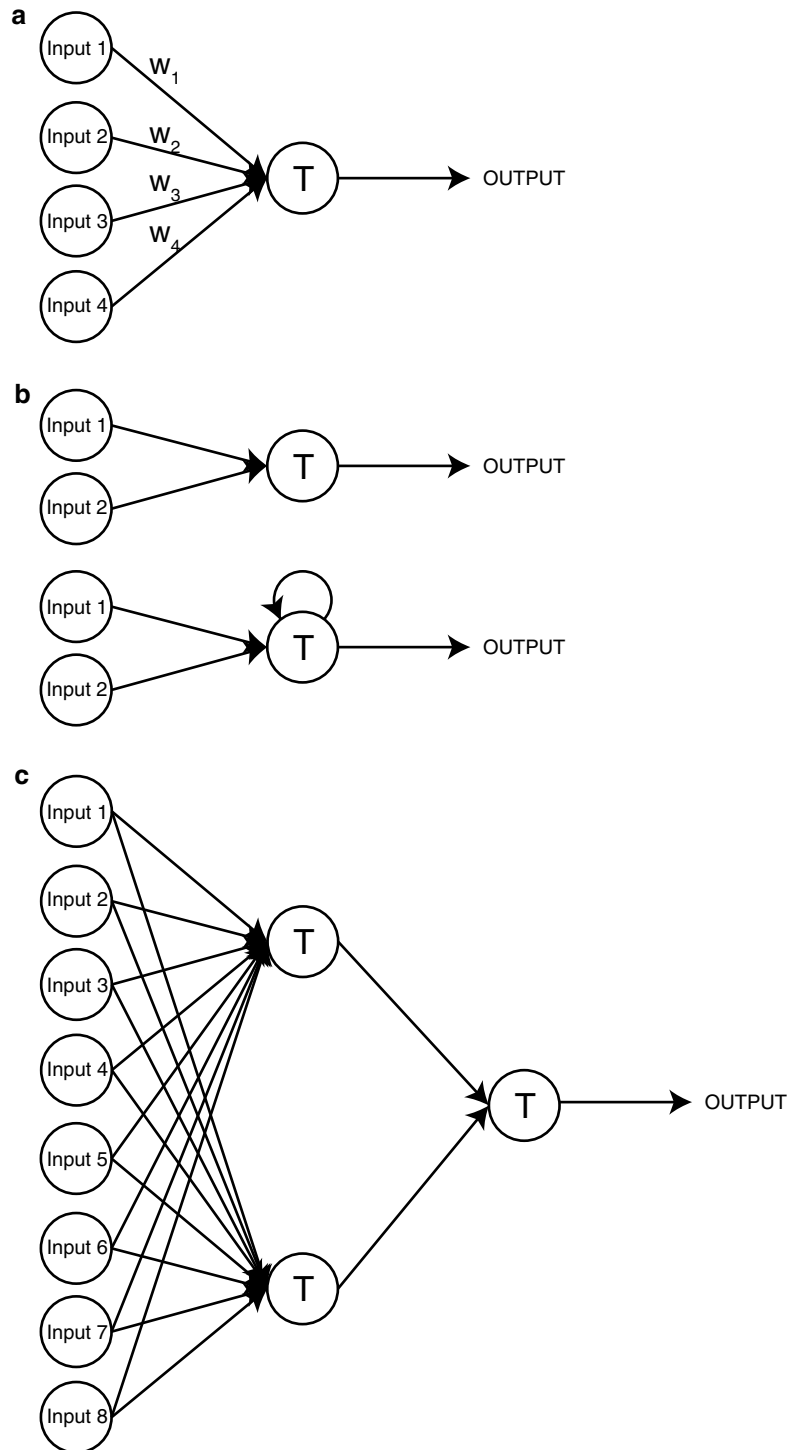


Fig. 2 Typical representation of artificial neural networks. Artificial neural networks consist of weighted inputs that are mathematically transformed to produce the outputs (a). These networks can be represented as weighted directed graphs and can be classified as feed-forward networks and recurrent (or feed-back) networks (b). Concretely, in multilayer perceptron networks, neurons are organized in layers connected by unidirectional edges (c)

learn from them, in a process called network training. The network learns how to reproduce the outputs by modifying the weights that multiply the inputs.

To successfully train a network, we have to provide an environment in which the network will operate (called the learning paradigm) and the rules that the network will use to update the weights (called the learning algorithm).

In our case, the learning paradigm is defined by peptide sequences and measurable properties (physicochemical parameters, biological activity, etc.). The learning rule will be an error-correction rule. In this rule, we randomly initialize the weights and feed the network with a training vector (a list of peptides with their properties and the classification index, 1 for active and 0 for inactive). The network will then update the weights in order to minimize the difference between the calculated and the given output (cost function). To do that, we use the back-propagation method that incorporates gradient descent to minimize the squared-error cost function.

Several packages are available to simulate neural network models in R, including neuralnet [27], RSNNs [28], and CARET [29] among others. Among the advantages of the CARET package are that it can run most network models using the same syntax and can automatically apply different pretreatment methods (e.g., principal component analysis) and validation protocols (e.g., k -fold cross validation). Therefore, we will be using the CARET package in our example.

3.3.1 Selection of the Training and Test Datasets

In the script provided, we have divided the entire dataset (AMPLIST) into a training set that contains 75 % of the observations and a testing set that contains 25 % of the total observations. This division is variable and can be changed in line 46 (bold number) (*see Note 6*):

```
smp_size<- floor(0.75 * nrow(M))
```

3.3.2 Training and Validation

The training step consists of feeding the model with input data and allowing it to optimize the weights in order to find the values that best fit the data. Different training strategies have been described in Subheading 2.3, and therefore, in this section, we will focus on explaining how validation is performed and why it is important.

Validation is an essential step in assessing the strength of the method. If the model were trained with the entire set of input data, we could expect that it would find a set of weights that fit the data with high accuracy. However, when this model is tested with a new set of examples, previously unseen by the model, we may find a poor prediction accuracy. This behavior is called over-fitting, which is a computational artifact that does not generate a true fitting rule but an overcompensated model that only fits the example data.

A validation step is thus required to evaluate when over-fitting is happening. The simplest form of validation consists of a k -fold cross validation. In this method we divide the dataset into k different equal-sized sets. In k different training rounds, one set is used as the testing set while the remaining sets are used for training the model. The average error and standard deviation of each round provides an estimation of how robust is the method.

A particular k -fold cross-validation strategy is called leave-one-out cross-validation strategy. This is achieved when k is equal to the number of observations minus one; then, only one prediction is made in each round (*see* **Note 7**).

As detailed before, in our example we use a multilayer perceptron (mlp) to fit our data using the CARET package in R. This package allows multiple options (*see* the CARET documentation for further details [29]).

Interestingly, we can specify the validation method through the `trControl` instruction. In our script, the validation strategy is fivefold cross validation, as detailed in line 56:

```
trC=trainControl(method="cv", number=5)
```

Multilayer perceptron networks can have different neuron layers (called hidden layers) with a variable number of neurons. We can specify this using the `tuneGrid` instruction provided in the `caret` package. In our script we have used only one hidden layer with a variable number of neurons (line 57, from 0 to 5):

```
my.grid<- expand.grid(.size=c(0,1,2,3,4,5))
```

Training is specified in the CARET package by using the `train` function. For the `train` function to be correctly defined, we must specify the training data, the classification data, the method used for training, the `preProcess` strategy, the validation strategy, and the variations in the `tuneGrid` parameter. The training instruction is detailed in line 58 of the script provided:

```
network <- train(train[,1:9], train[,10], method="mlp",
  preProcess=NULL, trControl=trC, tuneGrid=my.grid)
```

We have reproduced here typical output for a multilayer perceptron trained as described above:

3,211 samples

9 predictors

No pre-processing

Resampling: cross validation (fivefold)

Summary of sample sizes: 2,569, 2,569, 2,569, 2,569, 2,569

Resampling results across tuning parameters:

Size	RMSE	Rsquared	RMSE SD	Rsquared SD
0	0.41	0.366	0.0235	0.0477
1	0.406	0.376	0.0102	0.0409
...				
4	0.358	0.515	0.0185	0.0386
5	0.346	0.539	0.00706	0.0187

RMSE was used to select the optimal model using the smallest value. The final value used for the model was size = 5.

The results output summarize the method, meaning that we have used 9 predictors over a database of 3,211 examples, with no pre-processing and with a fivefold cross-validation strategy (each step comprising 2,569 examples used to train the method).

The method has tested the different number of neurons in the hidden layer, from 0 to 5. To determine which model was best, the method identifies the model with the lowest root mean square error (RMSE), which is the model with 5 neurons. The RMSE is the square root of the variance of the residuals, indicating the absolute fit of the model to the data. For classification purposes, our model outputs have values between 0 and 1 while the true values are binarized (either 0 or 1). RMSE measures the error distance between predicted and real data. As we need a threshold to map predicted to real data, RMSE does not account for prediction accuracy but is a good indicator of how well the model fits the data. As a rule of thumb, the lower the better; it should not be higher than 0.5.

3.3.3 Testing

We use the isolated section of the dataset (in our case the 25 %) to test the prediction power of the method. In the testing step, the method is applied to the testing dataset and the predictions are compared with the real values. In the best-case scenario, the predicted values will be identical to the real values. We define true positive values (TP) as those positive observations that are predicted as positive (in our case, antimicrobial peptides correctly predicted as antimicrobial) and true negatives (TN) as negative observations that are correctly predicted as negative (non-active peptides predicted as non-antimicrobial). We can also have false positives (FP, non-active peptides predicted as antimicrobials) and false negatives (FN, active peptides predicted as non-antimicrobials).

Therefore, we can build a matrix, as depicted in Table 3, known as the confusion matrix. Good prediction methods will have a large number of observations as true positives or true negatives and low numbers of false negatives and false positives.

Table 3
Confusion matrix

		Prediction	
		0	1
Real	0	TN	FP
	1	FN	TP

Good indicators of successful methods include a good sensitivity (the proportion of positives that are true) and specificity (the ability of the method to identify negative results that are truly negative). The mathematical expression of these indicators is

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Accuracy can be defined as the ability to predict true outcomes (positives and negatives) and can be calculated as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

In binary classifications (e.g., our case to classify peptides in anti-microbial and non-antimicrobial), we can use the Matthews correlation coefficient as a measure of the prediction quality. This coefficient is, briefly, a measure of the correlation between the observations and predictions. It can be calculated using the following formula:

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \times (\text{TP} + \text{FN}) \times (\text{TN} + \text{FP}) \times (\text{TN} + \text{FN})}}$$

Finally, when possible, an experimental validation of the method can also be performed as the ultimate test to validate the strategy presented. This is accomplished by generating a new list of compounds, not previously seen by the algorithm and analyzed by the method. Then a subset of predictions containing both positive and negative examples is experimental validated.

In the script provided, the confusion matrix, specificity, selectivity, and accuracy are calculated in lines 63 and 64 of the script (*see Note 8*):

```
confusion.matrix(test3, pred, threshold=0.5)
```

```
accuracy(test3, pred, threshold=0.5)
```

The results obtained in our network example give around 85 % sensitivity, 85 % specificity, and 85 % accuracy. This is a good result, as the network detected 85 % of all positive examples in the testing dataset and 85 % of the negative results were true negatives.

Despite this, there is still room for improvement; this might be accomplished by a variety of different means. Possible strategies to obtain a better classification are: (1) include more parameters, particularly those covering other physicochemical characteristics, (2) include more neurons in the hidden layer, and (3) include another hidden layer.

It is difficult to suggest a priori which is the best strategy to follow but it is important to remember that all the strategies will add one or more degrees of complexity, therefore complicating the interpretation of the results. A compromise must always be considered between how important it is to get good predictions and how important it is to understand how the model works.

3.4 Network Visualization

To help understand the structure of the neural network and how it operates, it is useful to represent it as shown in Fig. 2. Also, it is important to specify the weights, that is, the contribution of each node to the next layer of the network. The higher the contribution of the node, the more important it is to the prediction.

In the script provided (line 69), we have used a function called `plot.nnet` to visualize our network (please see <http://beckmw.wordpress.com/> for a complete description of this function):

```
Repnet<- mlp(train[,1:9], train,10, size=4)
plot.nnet(repnet)
```

First, we recompute the neural network with a number of hidden neurons (`size`) specified by the output results as detailed in Subheading 3.3.2 and then use the `plot.nnet` function to plot the results (Fig. 3).

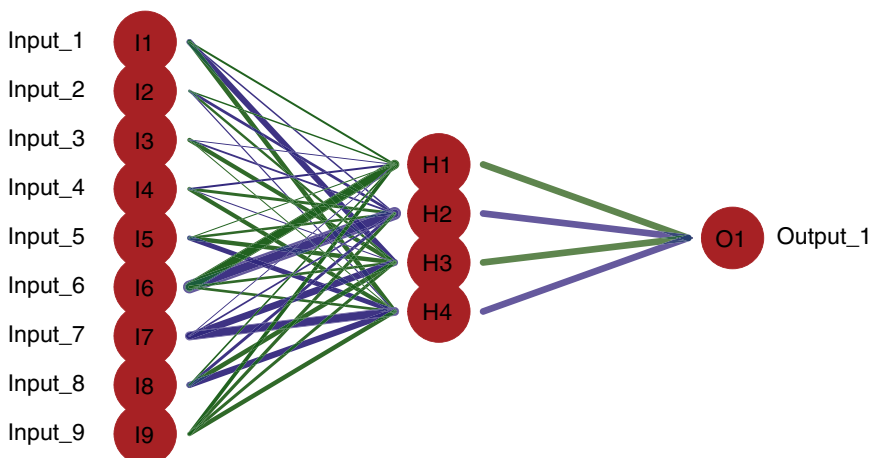


Fig. 3 Example of a neural network obtained using the example model proposed in this chapter

Weights are depicted in the edges that connect the nodes. The absolute value of the weight is symbolized by the width of the line (the wider the line, the higher the contribution). The line color symbolizes whether the contribution is positive (green) or negative (blue). In this example, descriptors 6 and 7 are contributing significantly to the model while descriptor 2 is contributing less.

The `plot.nnet` function does not depict the bias values (constant values added to each hidden and output layer) for the `mlp` neural network; however, this has no impact in the interpretation of our results. The `plot.nnet` function only depicts the bias values for the `nnet` function.

4 Notes

1. The R package can be downloaded from <http://www.r-project.org/>.
2. The RStudio package can be downloaded from <http://www.rstudio.com/>.
3. To install the required libraries in R, use the `install.packages("")` function.
4. Our suggestion is always to select a number of parameters that are thought to be relevant and then, if the results are not satisfactory, try other parameters or extend the list by a small amount.
5. A threshold between 0.9 and 0.95 can be used normally. If the threshold is much higher than 0.95, only a limited degree of filtering will be achieved; on the other hand, if it is less than 0.9, there is a risk that information that may be important for the network can be lost.
6. Care should be taken not to use too small a testing set, since that will lead to nonrepresentative results. At the other extreme, a very large testing set will leave too few observations for use during training and the model will fit poorly. In general, this parameter can be adjusted between 0.6 and 0.9 depending on the total number of total observations though values between 0.75 and 0.85 are recommended.
7. When the database is large, it is time consuming to perform a leave-one-out cross-validation approach because many training cycles will be required for large datasets. In that case, a more suitable strategy is to validate the method using a k -fold cross validation, usually with k values ranging from 4 to 10.
8. Because the activation function in the neural network is the logistic function, the output values are in the range $[0,1]$. The threshold value will binarize the results. Therefore, if the threshold is 0.5, values below 0.5 will be assigned the output value 0 and above 0.5 the output value 1.

Acknowledgements

This work was supported by the European Union [grant number HEALTH-F3-2008-223414 to D.A. and FP7-PEOPLE-2012-IEF-330352 to M.T.], the Ministerio de Economía y Competitividad [grant number SAF2011-24899 to D.A.], and the Generalitat de Catalunya [grant number SGR2009-494 to D.A.]. M.T. would also like to acknowledge support from the Ramón y Cajal Program (Spain).

References

1. Nelson DL, Cox MM (2005) *Lehninger principles of biochemistry*. 4th edn. p 1100
2. Liu F, Baggerman G, Schoofs L et al (2008) The construction of a bioactive peptide database in Metazoa. *J Proteome Res* 7:4119–4131
3. Zhang H, Forman HJ (2012) Glutathione synthesis and its role in redox signaling. *Semin Cell Dev Biol* 23:722–728
4. Patel BM, Mehta AA (2012) Aldosterone and angiotensin: role in diabetes and cardiovascular diseases. *Eur J Pharmacol* 697:1–12
5. Amblard M, Fehrentz J-A, Martinez J et al (2005) Fundamentals of modern peptide synthesis. *Methods Mol Biol* 298:3–24
6. Coin I, Beyermann M, Bienert M (2007) Solid-phase peptide synthesis: from standard procedures to the synthesis of difficult sequences. *Nat Protoc* 2:3247–3256
7. Hilpert K, Winkler DFH, Hancock REW (2007) Peptide arrays on cellulose support: SPOT synthesis, a time and cost efficient method for synthesis of large numbers of peptides in a parallel and addressable fashion. *Nat Protoc* 2:1333–1349
8. Winkler DFH, Campbell WD (2008) The spot technique: synthesis and screening of peptide macroarrays on cellulose membranes. *Methods Mol Biol* 494:47–70
9. Reddy AS, Pati SP, Kumar PP et al (2007) Virtual screening in drug discovery—a computational perspective. *Curr Protein Pept Sci* 8:329–351
10. Boix E, Nogues VM, Torrent M (2012) Discovering new in silico tools for antimicrobial peptide prediction. *Curr Drug Targets* 13:1148–1157
11. Torrent M, Andreu D, Nogués VM et al (2011) Connecting peptide physicochemical and antimicrobial properties by a rational prediction model. *PLoS One* 6:e16968
12. Wang G, Li X, Wang Z (2009) APD2: the updated antimicrobial peptide database and its application in peptide design. *Nucleic Acids Res* 37:D933–D937
13. Hammami R, Zouhir A, Le Lay C et al (2010) BACTIBASE second release: a database and tool platform for bacteriocin characterization. *BMC Microbiol* 10:22
14. Waghu FH, Gopi L, Barai RS et al (2014) CAMP: collection of sequences and structures of antimicrobial peptides. *Nucleic Acids Res* 42:D1154–D1158
15. Seebah S, Suresh A, Zhuo S et al (2007) Defensins knowledgebase: a manually curated database and information source focused on the defensins family of antimicrobial peptides. *Nucleic Acids Res* 35:D265–D268
16. Gueguen Y, Garnier J, Robert L et al (2006) PenBase, the shrimp antimicrobial peptide penaeidin database: sequence-based classification and recommended nomenclature. *Dev Comp Immunol* 30:283–288
17. Hammami R, Ben HJ, Vergoten G et al (2009) PhytAMP: a database dedicated to antimicrobial plant peptides. *Nucleic Acids Res* 37:D963–D968
18. Li Y, Chen Z (2008) RAPD: a database of recombinantly-produced antimicrobial peptides. *FEMS Microbiol Lett* 289:126–129
19. Huang Y, Niu B, Gao Y et al (2010) CD-HIT Suite: a web server for clustering and comparing biological sequences. *Bioinformatics* 26:680–682
20. Jenssen H (2011) Descriptors for antimicrobial peptides. *Expert Opin Drug Discov* 6:171–184
21. Kawashima S, Pokarowski P, Pokarowska M et al (2008) AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res* 36:D202–D205

22. Kawashima S, Ogata H, Kanehisa M (1999) AAindex: amino acid index database. *Nucleic Acids Res* 27:368–369
23. Li ZR, Lin HH, Han LY et al (2006) PROFEAT: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. *Nucleic Acids Res* 34:W32–W37
24. Rao HB, Zhu F, Yang GB et al (2011) Update of PROFEAT: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. *Nucleic Acids Res* 39:W385–W390
25. Krogh A (2008) What are artificial neural networks? *Nat Biotechnol* 26:195–197
26. Heaton J. (2012) Introduction to the Math of Neural Networks. Heaton Research Inc. 119 pages.
27. Günther F, Fritsch S (2010) neuralnet: training of neural networks. *R J* 2:30–38
28. Bergmeir C, Benitez JM (2012) Neural Networks in R using the Stuttgart Neural Network Simulator: RSNNS. *J Stat Software* 46:1–26
29. Kuhn M (2008) Building predictive models in R using the caret package. *J Stat Software* 28: 1–26

AutoWeka: Toward an Automated Data Mining Software for QSAR and QSPR Studies

Chanin Nantasenamat, Apilak Worachartcheewan, Saksiri Jamsak, Likit Preeyanon, Watshara Shoombuatong, Saw Simeon, Prasit Mandi, Chartchalerm Isarankura-Na-Ayudhya, and Virapong Prachayasittikul

Abstract

In biology and chemistry, a key goal is to discover novel compounds affording potent biological activity or chemical properties. This could be achieved through a chemical intuition-driven trial-and-error process or via data-driven predictive modeling. The latter is based on the concept of quantitative structure-activity/property relationship (QSAR/QSPR) when applied in modeling the biological activity and chemical properties, respectively, of compounds. Data mining is a powerful technology underlying QSAR/QSPR as it harnesses knowledge from large volumes of high-dimensional data via multivariate analysis. Although extremely useful, the technicalities of data mining may overwhelm potential users, especially those in the life sciences. Herein, we aim to lower the barriers to access and utilization of data mining software for QSAR/QSPR studies. AutoWeka is an automated data mining software tool that is powered by the widely used machine learning package Weka. The software provides a user-friendly graphical interface along with an automated parameter search capability. It employs two robust and popular machine learning methods: artificial neural networks and support vector machines. This chapter describes the practical usage of AutoWeka and relevant tools in the development of predictive QSAR/QSPR models. Availability: The software is freely available at <http://www.mt.mahidol.ac.th/autoweka>.

Key words Quantitative structure-activity relationship, Quantitative structure-property relationship, QSAR, QSPR, Data mining

1 Introduction

Interactions of drugs with their target proteins are governed by a multitude of molecular forces including electrostatic, hydrophobic, polar, and steric. They can be rationalized through the use of the quantitative structure-activity/property relationship (QSAR/QSPR) paradigm, which has been one of the foremost tools in the arsenal of recent drug discovery efforts and has been almost 100 years in the making. The prelude to the formulation of QSAR/QSPR was the preliminary and independent efforts of Brodin [1]

and Cros [2] in the mid-1850s; they established that there exists an association between chemical constitution and physiological properties. This was at a time prior to the reporting of the aromatic ring structure of benzene in 1865 by Kekulé [3]. In 1868, Brown and Fraser observed changes to physiological action upon methylation of the basic nitrogen atom in alkaloids. In 1869, Richardson [4] showed that aliphatic alcohols of different molecular weight also displayed varied narcotic effects. Similarly, a few decades later in 1893, Richet [5] showed that the water solubility of polar chemicals (such as alcohols, ethers, and ketones) was inversely related to their toxicities, whereby a decrease in the solubility of a polar chemical results in an increase in its toxicity. Toward the 1900s, Meyer and Overton [6, 7] reported that the lipophilicity of a group of organic compounds exhibited a linear relationship with their narcotic potencies. In 1917, Moore [8] investigated the effects of chemical “fumigants” on insects and showed that toxicity of these compounds increased with their boiling point. It can be seen that the majority of these efforts pertained to the establishment of qualitative relationships between chemicals and their respective activity/property.

Quantitative relationships between chemical structures and activity/property started to take shape when Hammett [9] introduced a simple equation (later to be known as the Hammett equation) in 1937 that considers the substituent effect caused by electron-withdrawing or electron-donating groups as summarized by the sigma constant, while the rho constant describes the structural class or chemotype under study. Such an equation allows determination of electronic effects caused by the placement of substituent groups at different positions (i.e., ortho-, meta-, and para-) of the benzene ring. The subsequent work of Taft [10] led to the introduction of the steric parameter. It is these two contributions from Hammett and Taft that paved the way for further contributions from Hansch et al. In the mid-1950s to 1960s, Hansch and Muir employed the Hammett substituent constant and partition coefficients in their formulation of equations to correlate the chemical substituents and growth of *Avena* plant using plant growth regulators comprising of phenoxyacetic acids and chloromycetins [11–13]. Finally in 1964, Hansch and Fujita [14] investigated the biological effects of a wide range of chemicals using a combination of physicochemical properties as summarized by the following equation:

$$\log \frac{1}{C} = a \cdot \sigma + b \cdot \pi - c \cdot \pi^2 + d$$

where C is the molar concentration of hormone, σ is the Hammett parameter, and π is a measure of hydrophobicity. In parallel, Free and Wilson [15] analyzed such structure-activity relationships using a binary approach in which the presence or absence of a substituent

was described as 0 and 1, respectively. These landmark reports marked the beginning of classical QSAR/QSPR. Later in 1969, Hansch stressed the importance of computers in analyzing structure-activity relationships [16], which was an observation that is true to this very day, in which the ever-increasing amount of biological and chemical data makes computers indispensable in the analysis of these relationships.

The essence of QSAR/QSPR lies in the predictive ability of these models to relate a set of explanatory variables (\mathbf{X}) with their response variables (\mathbf{Y}). Such explanatory variables of compounds are represented by molecular descriptors, which are correlated with functional moieties present in the chemical structure. Descriptors can be calculated directly from the chemical structure, although in some circumstances some form of geometry optimization is needed prior to obtaining the molecular features. Depending on the software used, the number of descriptors can span thousands of collinear and redundant descriptors, a situation that would typically call for the use of feature selection. Structure-activity/property relationships can then be discerned from traditional multiple linear regression as well as from a wide array of available machine learning algorithms. The reliability of constructed QSAR/QSPR models could be assessed from their internal and external validation, statistical measures of predictive performance, Y-randomization experiments, etc. Assessment of the applicability domain of the constructed QSAR/QSPR models may also provide useful knowledge of the range or scope of compounds that are covered by the model, since the model is only as good as the compounds used to train it. The aforementioned procedures are common steps in a typical QSAR/QSPR workflow as summarized in Fig. 1; further details are discussed in previous review articles [17, 18].

As noted previously, the early days of QSAR/QSPR were predominantly concerned with its utilization for investigating toxicity and the narcotic effects of compounds. Over time, this use had expanded to encompass a wide range of biological activities and chemical properties, as summarized in Table 1. In a simplistic system, compounds are isolated in the sense that they act or behave a certain way (e.g., as revealed by their boiling point or melting point) owing directly to their unique chemical structure. To complicate the matter, it may be the case that minor changes to the chemical structure may exert a drastic influence on its observed property. This issue is known as the structure-activity cliff and had been documented previously [19, 20]. The underlying reason for this is that a majority of biological activities arises from the interaction of compounds with target proteins. A minor change in the chemical structure, such as the replacement of a methyl group by an ethyl group, may then cause steric clashes with an amino acid side chain inside the protein's binding pocket and thus radically alter the activity of the compound.

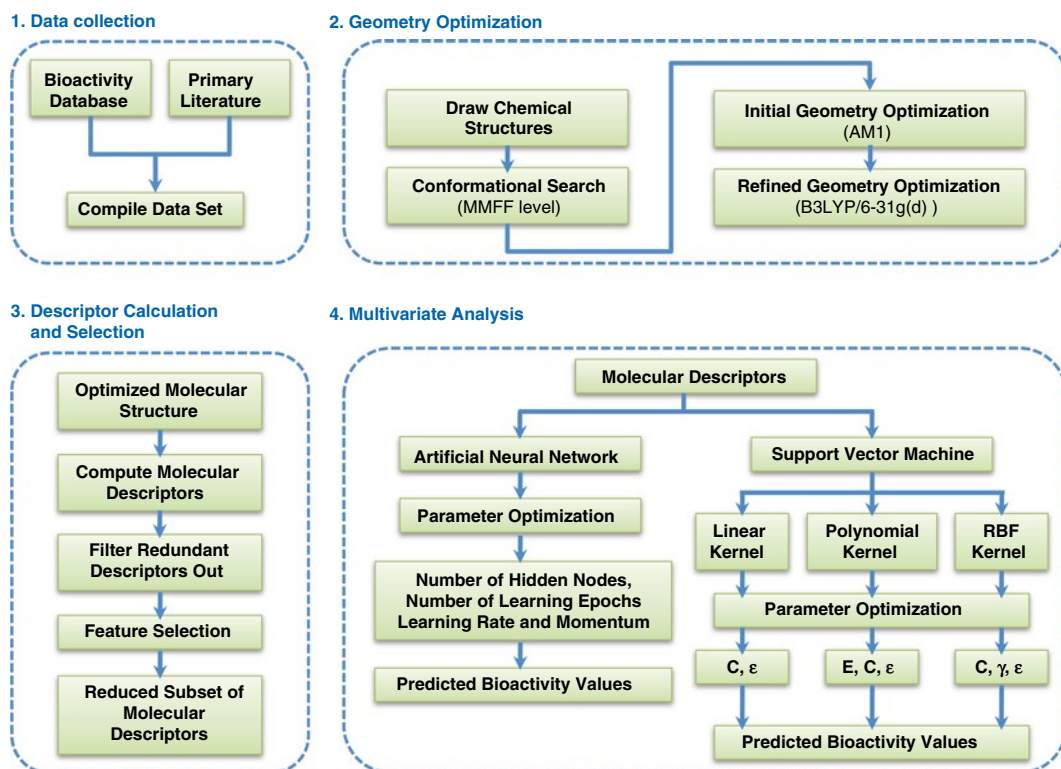


Fig. 1 Schematic representation of the QSAR/QSPR workflow

Although QSAR/QSPR may afford a wide range of utility in the life sciences, there are several potential pitfalls that practitioners need to be aware of. As this topic is beyond the scope of this chapter, readers are directed to previously reported accounts of such flaws [21–25] and solutions [26–29]. In particular, Dearden et al. [30] discuss 21 common types of error encountered in the QSAR/QSPR literature; they also provide recommendations on how to tackle such problems.

Efforts to automate some aspects of the QSAR workflow first appeared in 2001 when Jewell et al. [31] performed automatic generation of alignments for 3D-QSAR studies. A subsequent work from Tervo et al. [32] compared the performance of manual and automatic alignments. The first automated PLS-based feature selection was reported in 2004 by Olah et al. [33] in their exhaustive analysis of biological data from WOMBAT. A year later, Bhonsle et al. [34] reported an automated quasi-4D-QSAR study of CXCR4 inhibitors based on PLS and scripting language. In the same year, Cartmell et al. [35] described an automated QSPR workflow to investigate ADME data sets based on a software architecture called competitive workflow as implemented in the Discovery Bus. Zhang et al. [36] introduced the Automated Lazy

Table 1
Types of biological activities modeled by QSAR and selected examples

Type	Examples
Absorption	Blood-brain barrier penetration Human intestinal absorption P-glycoprotein Skin permeability
Distribution	Aqueous solubility (logS) Octanol-water partition coefficient (logP) Octanol-water distribution coefficient (logD)
Metabolism	Cytochrome P450 induction/inhibition
Excretion	Hepatic microsomal intrinsic clearance
Toxicity	AMES mutagenicity Carcinogenicity Hepatotoxicity Skin sensitivity Target/organ cytotoxicity
Receptor agonist/antagonist	A3 adenosine receptor antagonist Angiotensin II receptor antagonist CCR5 receptor antagonist Glucagon receptor antagonist Peroxisome proliferator-activated receptor gamma agonists
Enzyme activator/inhibitor/modulator	Aromatase inhibitor Dipeptidyl peptidase IV inhibitor Gamma-aminobutyric acid modulator Gamma-secretase modulator Glucokinase activator

Learning QSAR (ALL-QSAR) modeling approach in 2006 as applied to three sets of anticonvulsant agents. Subsequently in 2007, Bhonsle et al. [37] reported a semiautomated QSAR workflow using Tcl-based Cerius2 scripts in their investigation of a set of insect repellants. Furthermore, Obrezanova et al. [38] described an automated QSAR modeling approach of ADME data sets using the Gaussian process. Moreover, Rodgers et al. [39] introduced an approach to automatically update the QSAR model with measured data of new compounds in their human plasma protein binding model. In 2008, in a continuation of their earlier work, Obrezanova et al. compared the results from automated QSAR models with those of manual efforts in their study of blood-brain barrier penetration and aqueous solubility. In the same year, Ma et al. [40] reported an inductive data mining approach in the automatic generation of decision trees for modeling of the ecotoxicity of chemicals as well as in the analysis of a historical data from wastewater

treatment plant. In 2011, Wood et al. [41] presented an automated QSAR procedure employed in AstraZeneca's AutoQSAR system as tested against three properties comprising of $\log D$, solubility, and human plasma protein binding. Furthermore, Sushko et al. introduced the Online Chemical Modeling Environment (OCHEM) as a web-based tool for automating the process of QSAR modeling. In 2012, Pérez-Castillo et al. [42] described the genetic-algorithm-(meta)-ensembles approach for binary classification of five data sets from the literature. In 2013, Cox et al. [43] reported the QSAR Workbench, which is based on the Pipeline Pilot workflow tool and evaluated against two public domain data sets. Furthermore, Martins and Ferreira [44] introduced software called QSAR modeling for generating and validating QSAR models.

As we can see, development of QSAR models may not be a straightforward task, especially for the non-bioinformatician, and this is concomitant with the fact that efforts to automate the QSAR/QSPR process are an active area of research. These and many other factors sparked our interests and motivated us to develop AutoWeka starting from late 2009, which we have been coding ever since. The project began as an effort to automate our own data mining workflow as applied to QSAR/QSPR modeling and finally became publicly and freely available in 2012. The data mining capabilities, particularly through the use of artificial neural networks and support vector machines, of AutoWeka are powered by the popular and widely used Weka machine learning package [45]. It should be noted at the outset that the AutoWeka software is used for the multivariate analysis phase of QSAR/QSPR modeling, for which it also performs an extensive parameter optimization, the results of which could be scrutinized and selected for further rounds of computation (*see Note 1*). Nevertheless, specific details of chemical structure drawing and molecular descriptor generation are also mentioned in this chapter. It is also worthy of mention that this chapter will not cover the software development of AutoWeka, which will be discussed elsewhere.

In the following sections, we will provide an example of the practical use of AutoWeka in constructing QSAR/QSPR models (*see Note 2*). This is demonstrated in a step-by-step manner using a set of curcumin analogs as a case study.

2 Materials

2.1 Data Source

There are several ways in which one can acquire the data that is to be used for QSAR/QSPR modeling. It can be measured data from one's own experiments, compiled from the primary literature, retrieved from curated databases, or even all of the above. In most cases, accessibility to the primary literature is heavily dependent on institutional or personal subscription, while there are a growing number of open access journals that are freely available for readers.

Typical journals describing the implementation of QSAR/QSPR models include but are not limited to the following:

- *Bioinformatics*
- *Bioorganic & Medicinal Chemistry*
- *Bioorganic & Medicinal Chemistry Letters*
- *BMC Bioinformatics*
- *Briefings in Bioinformatics*
- *Chemical Biology & Drug Design*
- *Chemometrics and Intelligent Laboratory Systems*
- *Chemosphere*
- *Computational and Theoretical Chemistry* (formerly *Journal of Molecular Structure: THEOCHEM*)
- *Computational Biology and Chemistry*
- *European Journal of Medicinal Chemistry*
- *International Journal of Quantum Chemistry*
- *Journal of Chemical Information and Modeling*
- *Journal of Cheminformatics*
- *Journal of Chemistry*
- *Journal of Chemometrics*
- *Journal of Computational Biology*
- *Journal of Computational Chemistry*
- *Journal of Computer-Aided Molecular Design*
- *Journal of Enzyme Inhibition and Medicinal Chemistry*
- *Journal of Medicinal Chemistry*
- *Journal of Molecular Graphics and Modelling*
- *Journal of Molecular Modeling*
- *Journal of Theoretical and Computational Chemistry*
- *Letters in Drug Design & Discovery*
- *Medicinal Chemistry Research*
- *Molecular Informatics*
- *Molecular Simulation*
- *PLoS Computational Biology*
- *SAR and QSAR in Environmental Research*

The following databases contain curated bioactivity data that have either been deposited by the laboratory generating the data or compiled from the literature:

- BindingDB—a database containing ~1,000,000 pieces of interaction data for ~6,500 protein targets and ~427,000 small molecules. It is accessible at <http://www.bindingdb.org>.

- ChEMBL—a database containing ~12,000,000 pieces of interaction data for ~9,000 protein targets and ~1,500,000 small molecules. It is accessible at <https://www.ebi.ac.uk/chemblpdb>.
- DrugBank—a database containing ~6,800 drug entries spanning FDA-approved small molecules, FDA-approved peptide/protein drugs, nutraceuticals, and experimental drugs. It is accessible at <http://www.drugbank.ca>.
- PubChem—is comprised of three linked databases spanning substance, compound, and bioassay. It is accessible at <http://pubchem.ncbi.nlm.nih.gov>.
- WOMBAT—a database containing ~79,000 in vivo measurement data, ~330,000 compounds, and ~1,900 protein targets. It is commercially available at <http://www.sunsetmolecular.com>.

2.2 Drawing and Refining Chemical Structures

Chemical structures can be drawn into the computer using any of the following software tools:

- Accelrys Draw (available at <http://www.accelrys.com>)
- MarvinSketch (available at <http://www.chemaxon.com>)
- OpenEye VIDA (available at <http://www.eyesopen.com>)

Chemical structure file format conversion tools of use include:

- Babel (available at <http://www.eyesopen.com>)
- Open Babel (available at <http://www.openbabel.org>)

Molecular structures can be refined using online tools:

- CORINA Online Demo (available at http://www.molecular-networks.com/online_demos/corina_demo)
- COSMOS (accessible at <http://cosmos.igb.uci.edu>)

In certain cases where the molecular structure is relevant or of importance for use in prediction, it can also be subjected to geometry optimization via quantum chemical calculations:

- Gaussian (commercially available at <http://www.gaussian.com>)
- GAMESS (available at <http://www.msg.ameslab.gov/gamess>)
- HyperChem (commercially available at <http://www.hyper.com>)
- MOLCAS (commercially available at <http://www.molcas.org>)
- MOPAC (available at <http://openmopac.net>)

2.3 Calculating Molecular Descriptors

The next step is to generate numerical descriptions of compounds to be investigated; a wide range of software is available to perform this task. For example, common compounds may be available from

the MOLE-db website (Accessible at http://michem.disat.unimib.it/mole_db). Moreover, quantum chemical descriptors can be obtained from the aforementioned software such as Gaussian, GAMESS, etc. Several other categories of molecular descriptors could be obtained from the following software:

- CDK Descriptor GUI (available at <http://www.rguha.net/code/java/cdkdesc.html>)
- ChemAxon JChem Calculator Plugins (available at <http://www.chemaxon.com/jchem>)
- Dragon (commercially available from <http://www.talete.mi.it>)
- E-Dragon (accessible at <http://www.vcclab.org/lab/edragon>)
- MODEL (accessible at <http://jing.cz3.nus.edu.sg/cgi-bin/model/model.cgi>)
- PaDEL (available at <http://padel.nus.edu.sg>)

2.4 Data Compilation

Prior to constructing the QSAR/QSPR model, the block of **X** molecular descriptors and **Y** bioactivity values are combined and prepared in ARFF file format for use as input to AutoWeka.

- Spreadsheet program
 - Microsoft Excel (commercially available at <http://office.microsoft.com/excel>)
 - OpenOffice (freely available at <http://www.openoffice.org>)
- Text editor
 - Notepad++ (freely available at <http://notepad-plus-plus.org>)
 - EditPad Lite (freely available at <http://www.editpadlite.com>)
- CSV to ARFF converter
 - csv2arff (accessible from <http://slavnik.fe.uni-lj.si/markot/csv2arff/csv2arff.php>)

2.5 Multivariate Analysis

It should be noted here that multivariate analysis or the actual process of constructing the QSAR model is performed using AutoWeka (Freely available at <http://www.mt.mahidol.ac.th/autoweka>). It is in this phase that parameter optimization takes place in an automated fashion. Completed results will contain information on the best set of parameters for the selected machine learning algorithm.

2.6 Plotting Graphs

After completing AutoWeka calculations, results from parameter optimization could be visualized using the Python scripts available in the Download section at <http://www.mt.mahidol.ac.th/autoweka>. Alternatively, plots of the results could also be created using statistical graphical software such as SigmaPlot (Commercially available from <http://www.sigmaplot.com>).

3 Methods

3.1 Data Source

At this phase, the practitioner decides on the data to be used in their QSAR/QSPR project and therefore retrieves the necessary information (i.e., chemical name, SMILES, and bioactivity values) from resources mentioned in Subheading 2.1; the data are then compiled as outlined in Subheading 3.4. In the case study described in this chapter, we will be using a data set of 22 curcumin analogs with DPPH free radical-scavenging activity as previously reported by Venkateswarlu et al. [46] and modeled by Worachartcheewan et al. [47]. The bioactivity values were binned to the binary labels “low” and “high” activity, denoting compounds affording IC_{50} values of greater than and less than 10 μ M, respectively. It should be noted that in the previously reported QSAR modeling study described by Worachartcheewan et al. [47], three compounds (nos. 5, 6, and 10) were identified as outliers and subjected to removal from the data set, thereby reducing it to 19 compounds.

3.2 Drawing and Refining Chemical Structures

Chemical structures of curcumin analogs (Fig. 2) are drawn into the computer using software described in Subheading 2.2. Subsequently, chemical structures are either subjected to a rough energy minimization (using a molecular mechanics force field) or a more extensive quantum chemical calculation (using a tool such as HF, B3LYP, etc.) in order to obtain a more refined structure. These structures can be subjected to an initial geometry refinement using the semiempirical Austin Model 1 (AM1) followed by Becke’s three-parameter hybrid method using the Lee-Yang-Parr correlation functional (B3LYP) along with 6-31g(d) basis set as performed previously by Worachartcheewan et al. [47].

3.3 Calculating Molecular Descriptors

Molecular descriptors can be derived from unrefined chemical structures, which may be applicable for 1D and 2D descriptors. However, in order to derive 3D descriptors, the chemical structures must first be subjected to geometry optimization using quantum chemical calculations as noted above. Depending on the software used to generate the molecular descriptors, this can lead to the production of hundreds or thousands of descriptors. It is common to observe a high degree of collinearity or redundancy among these descriptors, and this issue is best handled by performing feature selection to select a smaller subset of informative descriptors for further multivariate analysis. As this issue is beyond the scope of this chapter, readers are directed to previous review articles.

3.4 Data Compilation

One of the first phases of a QSAR/QSPR study is compiling the data set of interest, which can be performed by entering essential data into an appropriate spreadsheet program. For example, a typical

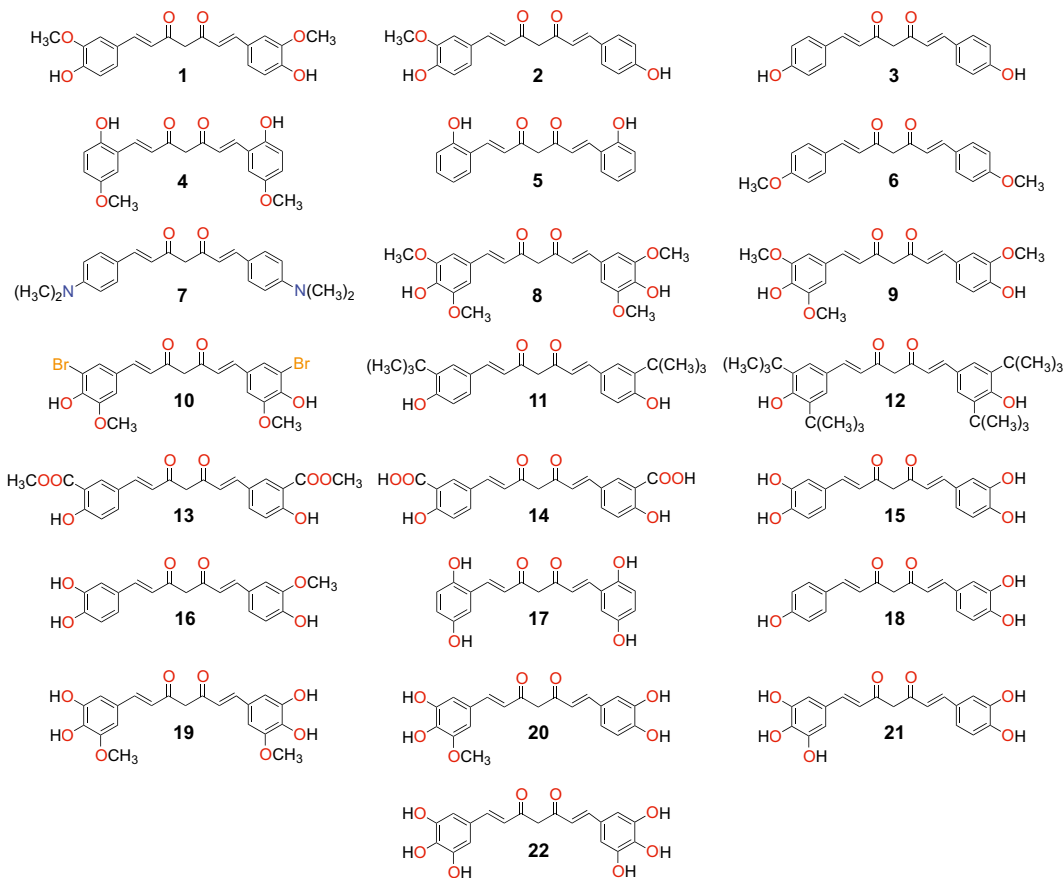


Fig. 2 Chemical structures of 22 curcumin analogs

molecule or compound will occupy a row in the spreadsheet where the columns can contain the following information: compound name, compound ID, XC_{50} of bioactivity (where X refers to the type of bioactivity such as inhibition concentration, effective concentration, etc.), SMILES notation, molecular descriptors (molecular weight, logP, number of hydrogen bond donor atoms, number of hydrogen bond acceptor atoms, etc.), and the reference or data source from which the compounds were obtained (Fig. 3). Therefore, as there are 19 curcumin analogs (after removal of three outliers) for this case study, this would correspond to 19 rows plus the header row (containing the descriptor labels of each column) resulting in a total of 20 rows.

The next step is to transform our data set from the spreadsheet format to an ARFF file format that is compatible with Weka, since this is the machine learning package that powers the software developed in AutoWeka. This spreadsheet to ARFF transformation can be carried out using a text editor or alternatively using the online csv2arff tool. Examples of ARFF input files are shown in

	A	B	C	D	E	F	G	H	I
1	Compound ID	SMILES	dipole	gap	hardness	softness	OH	pIC50	Class
2	1	COC1=CC(C=C=	4.820	-0.134	0.067	7.463	2	-1.322	Low
3	2	COC1=C(O)C=C=	3.530	-0.141	0.071	7.092	2	-1.531	Low
4	3	OC1=CC=C(C=C	3.793	-0.143	0.072	6.993	2	-1.519	Low
5	4	COC1=CC(C=C=	5.910	-0.124	0.062	8.065	2	-1.380	Low
6	7	CN(C)C1=CC=C	5.819	-0.127	0.064	7.874	0	-1.716	Low
7	8	COC1=CC(C=C=	7.447	-0.133	0.067	7.519	2	-1.415	Low
8	9	COC1=CC(C=C=	6.721	-0.129	0.065	7.752	2	-1.407	Low
9	11	CC(C)C1=CC=C	4.584	-0.142	0.071	7.042	2	-1.681	Low
10	12	CC(C)C1=CC=C	5.270	-0.139	0.070	7.194	2	-1.633	Low
11	13	COC(=O)C1=CC	5.122	-0.142	0.071	7.042	2	-2.000	Low
12	14	OC(=O)C1=CC	3.643	-0.146	0.073	6.849	2	-2.000	Low
13	15	OC1=CC=C(C=C	3.277	-0.134	0.067	7.463	4	-0.778	High
14	16	COC1=CC(C=C=	3.949	-0.134	0.067	7.463	3	-0.845	High
15	17	OC1=CC=C(C=C	6.099	-0.125	0.063	8.000	4	-0.903	High
16	18	OC1=CC=C(C=C	2.758	-0.138	0.069	7.246	3	-0.881	High
17	19	COC1=CC(C=C=	1.995	-0.129	0.065	7.752	4	-0.732	High
18	20	COC1=CC(C=C=	3.507	-0.127	0.064	7.874	4	-0.799	High
19	21	OC1=CC=C(C=C	3.901	-0.129	0.065	7.752	5	-0.716	High
20	22	OC1=CC(C=C=C	4.116	-0.127	0.064	7.874	6	-0.663	High

Fig. 3 Screenshot of compiled data using Microsoft Excel

Tables 2 and 3 with quantitative and qualitative Y labels. Line 1 represents the name of the data set, lines 3–8 represent the descriptor names, and lines 11–29 represent the block of descriptors and their Y values or labels as correspondingly specified by the following terms with the @ symbol preceding it: RELATION, ATTRIBUTE, and DATA. The NUMERIC terms that follow the descriptor names in lines 3–8 are syntax that are recognized by the Weka program as descriptors having quantitative values, whereas the braces {} encapsulating the Low and High terms are also syntax that are recognized by the Weka program as descriptors having qualitative values. It should be noted that the Y descriptor is typically located as the last variable, whereas X descriptors precede it. Generally, QSAR modeling of data sets having quantitative or qualitative Y variables is subjected to either regression or classification analysis, respectively.

3.5 Machine Learning Algorithms

Before we move on to the multivariate analysis phase and actually construct the QSAR model, it is pertinent to first provide a glimpse of the algorithmic details of machine learning algorithms that are commonly used in QSAR/QSPR on biological [47–56] and chemical systems [57–61] and which are implemented in AutoWeka.

3.5.1 Artificial Neural Network

The artificial neural network (ANN) is a well-known multivariate method that is commonly used to develop QSAR/QSPR models. ANN takes its inspiration from the biological brain with its organization of neurons [62]. The single-layer perceptron (SLP), which

Table 2
Example of ARFF input file of curcumin analogs with
quantitative Y variable

@RELATION RadicalScavengingRegression
@ATTRIBUTE dipole NUMERIC
@ATTRIBUTE gap NUMERIC
@ATTRIBUTE hardness NUMERIC
@ATTRIBUTE softness NUMERIC
@ATTRIBUTE OH NUMERIC
@ATTRIBUTE pIC50 NUMERIC
@DATA
4.82,-0.134,0.067,7.463,2,-1.322
3.53,-0.141,0.071,7.092,2,-1.531
3.793,-0.143,0.072,6.993,2,-1.519
5.91,-0.124,0.062,8.065,2,-1.38
5.819,-0.127,0.064,7.874,0,-1.716
7.447,-0.133,0.067,7.519,2,-1.415
6.721,-0.129,0.065,7.752,2,-1.407
4.584,-0.142,0.071,7.042,2,-1.681
5.27,-0.139,0.07,7.194,2,-1.633
5.122,-0.142,0.071,7.042,2,-2
3.643,-0.146,0.073,6.849,2,-2
3.277,-0.134,0.067,7.463,4,-0.778
3.949,-0.134,0.067,7.463,3,-0.845
6.099,-0.125,0.063,8,4,-0.903
2.758,-0.138,0.069,7.246,3,-0.881
1.995,-0.129,0.065,7.752,4,-0.732
3.507,-0.127,0.064,7.874,4,-0.799
3.901,-0.129,0.065,7.752,5,-0.716
4.116,-0.127,0.064,7.874,6,-0.663

Table 3
Example of ARFF input file of curcumin
analog with qualitative Y variable

@RELATION RadicalScavengingClassification
@ATTRIBUTE dipole NUMERIC
@ATTRIBUTE gap NUMERIC
@ATTRIBUTE hardness NUMERIC
@ATTRIBUTE softness NUMERIC
@ATTRIBUTE OH NUMERIC
@ATTRIBUTE class {Low, High}
@DATA
4.82,-0.134,0.067,7.463,2,Low
3.53,-0.141,0.071,7.092,2,Low
3.793,-0.143,0.072,6.993,2,Low
5.91,-0.124,0.062,8.065,2,Low
5.819,-0.127,0.064,7.874,0,Low
7.447,-0.133,0.067,7.519,2,Low
6.721,-0.129,0.065,7.752,2,Low
4.584,-0.142,0.071,7.042,2,Low
5.27,-0.139,0.07,7.194,2,Low
5.122,-0.142,0.071,7.042,2,Low
3.643,-0.146,0.073,6.849,2,Low
3.277,-0.134,0.067,7.463,4,High
3.949,-0.134,0.067,7.463,3,High
6.099,-0.125,0.063,8,4,High
2.758,-0.138,0.069,7.246,3,High
1.995,-0.129,0.065,7.752,4,High
3.507,-0.127,0.064,7.874,4,High
3.901,-0.129,0.065,7.752,5,High
4.116,-0.127,0.064,7.874,6,High

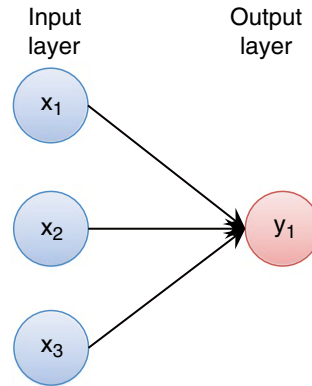


Fig. 4 Schematic architecture of the single-layer perceptron

is the classical model of ANN, is the simplest type of ANN; it is comprised of several input nodes and a single output node as shown in Fig. 4.

For a given training sample $D = (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$, we can estimate the variable y_i by combining the weighted sum of its N inputs as follows:

$$y_i = \theta \left(\sum_{i=1}^N w_i x_i \right) \quad (1)$$

where y_i belongs to class 1 if the weighted sum is greater than the selected threshold; otherwise, y_i belongs to class 0. $\theta(\bullet)$ is the activation function that maps the weighted sum of inputs to the output. The most popularly currently used activation functions are the logistic sigmoid ($1 / (1 + e^{-x_i})$) and hyperbolic tangent ($\tanh(x_i)$). In practical application, SLP is unable to solve a nonlinearly separable data problem, owing to the fact that during the learning process, this approach cannot properly predict all data points on D . Thus, the multilayer perceptron (MLP) was proposed for handling nonlinearly separable data by adding one or more hidden layers (see Fig. 5) [63, 64].

The MLP approach is a type of supervised learning method in which the back-propagation algorithm is applied to estimate the optimized parameter w_i by changing the weighted connection, which is dependent on the magnitude of the error (the difference between the actual and predicted value). The back-propagation algorithm is comprised of two major phases: (1) forward phase and (2) backward phase, as briefly described below.

Step 1. Initialize a connection weight w_i with a small random value.

Step 2. Randomly select a training sample $D_i \subset D$, $|D_i| = p$ where $p < N$.

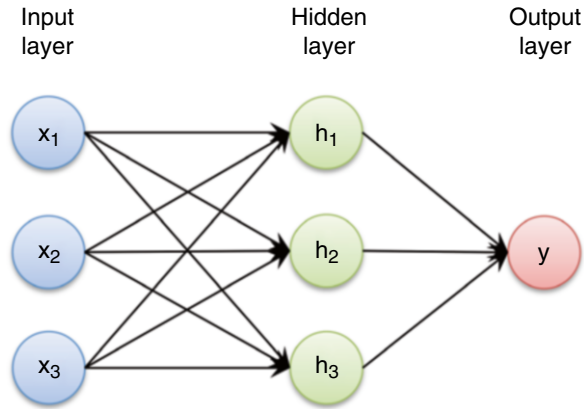


Fig. 5 Schematic architecture of the multilayer perceptron

Step 3. Calculate the partial derivative of weight w_i .

Step 4. Update weight w_i according to the following equations:

$$w_{ij}(n+1) = w_{ij}(n) + \eta \frac{\partial E(n)}{\partial w_{ij}} \quad (2)$$

$$w_{ij}(n+1) = w_{ij}(n) + \eta \frac{\partial E}{\partial w_{ij}} \quad (3)$$

where $w_{ij}(n)$ is the value of w_{ij} prior to updating by n times while $w_{ij}(n+1)$ is the value of w_{ij} after such updates and η is the learning rate that determines both the convergence rate and stability of the training process while E is the cost or error function.

Step 5. Repeat **steps 2–4** for every training sample D_i , and repeat for these sets until the cost of output errors is minimized.

3.5.2 Support Vector Machine

The support vector machine (SVM) is a popular learning method that had been adopted for solving a plethora of problems via classification and regression analysis. A notable property of SVM is its estimation of model parameters by means of the convex optimization approach that guarantees that a local solution is also the global optimum [65]. Vapnik first introduced this technique based on the principles of structure risk minimization of the statistical learning theory [66, 67]. In practice, the SVM method constructs a maximum-margin hyperplane to separate two classes. To solve nonlinearly separable data, the SVM method acts in conjunction with a mapping function $\Phi(x): x \in R^M \rightarrow R^P$ that is used to transform the original data set of M -dimension onto a higher dimensional space or feature space of P -dimension where $M \ll P$. Subsequently, a simple linear classifier $f(x_i)$ can then be used for classifying P -dimensional samples [68, 69] (shown in Fig. 6).

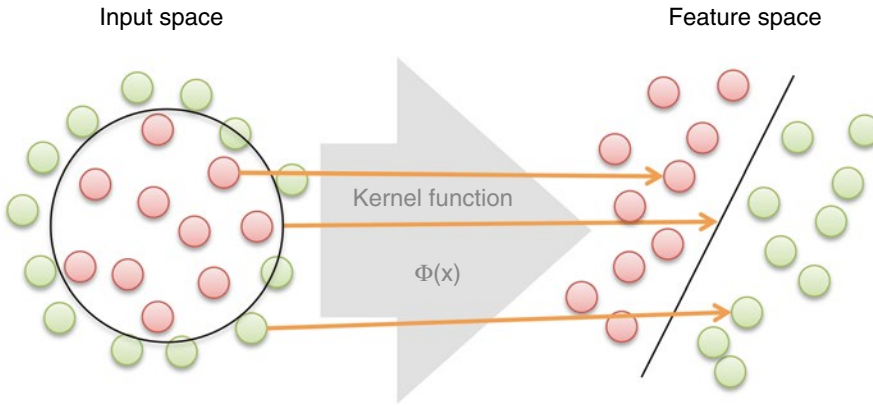


Fig. 6 Schematic representation of relationship between input and feature spaces using a mapping function

The kernel function $K(x_i, x_j)$ is used to represent the mapping function by taking the inner product between two samples x_i and x_j in D , which is defined as:

$$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) = \sum_{i,j=1}^N \Phi(x_i)^T \Phi(x_j) \quad (4)$$

In practice, given D , SVM is used to construct a linear function $f(x_i)$ representing the correlation between the structure and biological activities/chemical properties of data x_i :

$$f(x_i) = \sum_{i=1}^N w_i \Phi(x_i) + b \quad (5)$$

where $w_i \in R^M$ are the coefficients, $b \in R$ is the bias, and N is the number of samples. The most popularly used kernel comprises the following:

- Linear kernel:

$$\Phi(x_i)^T \Phi(x_j) \quad (6)$$

- Polynomial kernel:

$$\left(1 + \Phi(x_i)^T \Phi(x_j)\right)^d \quad (7)$$

where $d=2, 3$, and 4 (it should be noted that $d=1$ for a linear kernel).

- Radial basis function (RBF) kernel:

$$\exp\left(-\gamma(x_i - x_j)\right) \quad (8)$$

where γ is greater than 0.

Table 4
Summary of computational methods used in QSAR/QSPR modeling

Method	Advantage	Disadvantage	Built-in feature selector	Single/ensemble
ANN	Performs well on complex data	Low interpretability	No	Single
SVM	Solves nonlinearly separable data	Low interpretability	No	Single
PCA	Summarizes a data set without losing too much variation	Does not consider relationship between X and Y	Yes	Single
PLS	Simple and interpretable model	Requires cross-term	Yes	Single
DT	Simple and interpretable model	Requires a number of training data sets	Yes	Single
RF	High interpretability and low risk of over-fitting	Complexity method	Yes	Ensemble

ANN, SVM, PCA, PLS, DT, and RF are acronyms for artificial neural network, support vector machine, principal component analysis, partial least squares regression, decision tree, and random forest, respectively

In regression problems, when y is a numerical value, estimation of the parameter w_i can be achieved by utilizing the ε -insensitive loss function ($L_\varepsilon(y, f(x, w))$) [70, 71] described as follows:

$$L_\varepsilon(y, f(x, w)) = \begin{cases} |y - f(x, w)| - \varepsilon, & y - f(x, w) \geq \varepsilon \\ 0, & y - f(x, w) < \varepsilon \end{cases} \quad (9)$$

where y is the actual value, $f(x, w)$ is the predicted function for estimating the output value, and ε is the insensitivity parameter. Although SVM is widely popular, SVM and ANN methods both suffer from the fact that they are a black-box approach and therefore lack interpretability; they do not readily indicate which feature(s) is of greatest importance in the structure of the prediction model. Table 4 highlights and compares the advantages and disadvantages of SVM and ANN in the context of other commonly used learning algorithms in QSAR/QSPR modeling.

3.6 Multivariate Analysis

We will now proceed with the step-by-step case study of constructing the QSAR model using AutoWeka. Although we try to provide as much detail as possible on the practical aspects of using AutoWeka, some adjustments may need to be made as the reader adapts this protocol to their own projects. As previously mentioned in Subheading 3.1, the case study used in this chapter is based on the set of 22 curcumin analogs reported by Venkateswarlu et al. [46].

Let us start by going to the website of AutoWeka that is accessible at <http://www.mt.mahidol.ac.th/autoweka>. Readers can click on either the Download link on top or the orange Download button down below (Fig. 7).

A Data Mining Software that is user friendly for novices and yet powerful for experts.

Description

AutoWeka is an automated data mining software that facilitates the rapid development of predictive data mining models. The software provides an intuitive user interface that can be used right out of the box.

Best of all, it is free!

[Download AutoWeka](#)

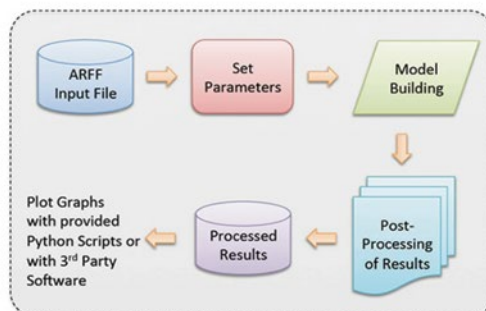


Fig. 7 Screenshot of AutoWeka's website

On the following page, readers have access to the AutoWeka User Manual, the Python scripts for creating graphs and plots, as well as the AutoWeka software. After agreeing to the terms of the license agreement, click on the “*I Agree, Please Proceed to Download AutoWeka*” button (Fig. 8) to proceed. The next page will then bring out a registration form that users can fill out, and after its submission, the Download link will appear. The zip file of the software is approximately 19 MB in size. After successfully downloading the file, unzip it to a desired location, and the following contents as shown in Fig. 9 (left panel) will appear. After double-clicking on the *AutoWeka.exe* file (left panel) the program window will appear (right panel) as shown in Fig. 9. The menu bar contains three possible options, *Run*, *Tools*, and *About*, which correspondingly allow users to run the machine learning algorithms for constructing the QSAR/QSPR model, adjust the memory value to use (Fig. 10), and provide access to the *About* window.

Now that the software is up and running, let us get started with setting up an ANN calculation (Fig. 11) by first clicking on *Run* → *Artificial Neural Network*. This brings up a new window that allows us to set up the various ANN parameters or choose the easiest option, which is to use the default parameters. Here we click on the *Browse* button and select the *ARFF* input file of interest (in our case, the *Curcumin_regression.arff*) file. Next, click on the *Default* button and finally the *Start* button to proceed with building the ANN model.

AutoWeka
An Automated Data Mining Software Based on Weka

Home About Features **Download** Citing Us

Download

Download AutoWeka User Manual

[AutoWeka User Manual](#) (Size: 1.4 MB)

Download Python Scripts for Creating Plots and Graphs

[Python Scripts for Creating Plots and Graphs](#) (Size: 4 KB)

Download AutoWeka Software

To download the AutoWeka software please read the license agreement below then click on the orange button to proceed with the download.

License Agreement

AutoWeka is a free software subject to the terms of the license agreement as detailed below.

Copyright © 2012 Chanin Nantasenamat. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to use the Software as obtained. The above copyright notice and this permission notice shall be included in all copies of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

By downloading the software, users agree to abide by the terms of the software license agreement

I Agree, Please Proceed to Download AutoWeka

Fig. 8 Screenshot of the Download page on AutoWeka's website

Subsequently, a new window appears that summarizes the parameter settings that will be used for the calculation. Upon clicking on the *OK* button, a pop-up window asks for confirmation of the name of an automatically generated folder (Fig. 12). Here, we can click on the *OK* button again to use the default name. Next, the progress window (Fig. 13) appears; the time required for completion will depend on the complexity of the input data. Upon completion, a pop-up notification box appears, and we can then click on the *OK* button to finalize the calculation.

All calculation files are located in the *Results* folder (Fig. 14) that also goes by the same folder name in the root folder of AutoWeka, meaning that if we unpacked AutoWeka directly to the C drive, the relative path of the root folder would be *C:\AutoWeka* and thereby the Results folder could be found at *C:\AutoWeka\Results*. Inside the Results folder, double-click on the *Curcumin_analogs*

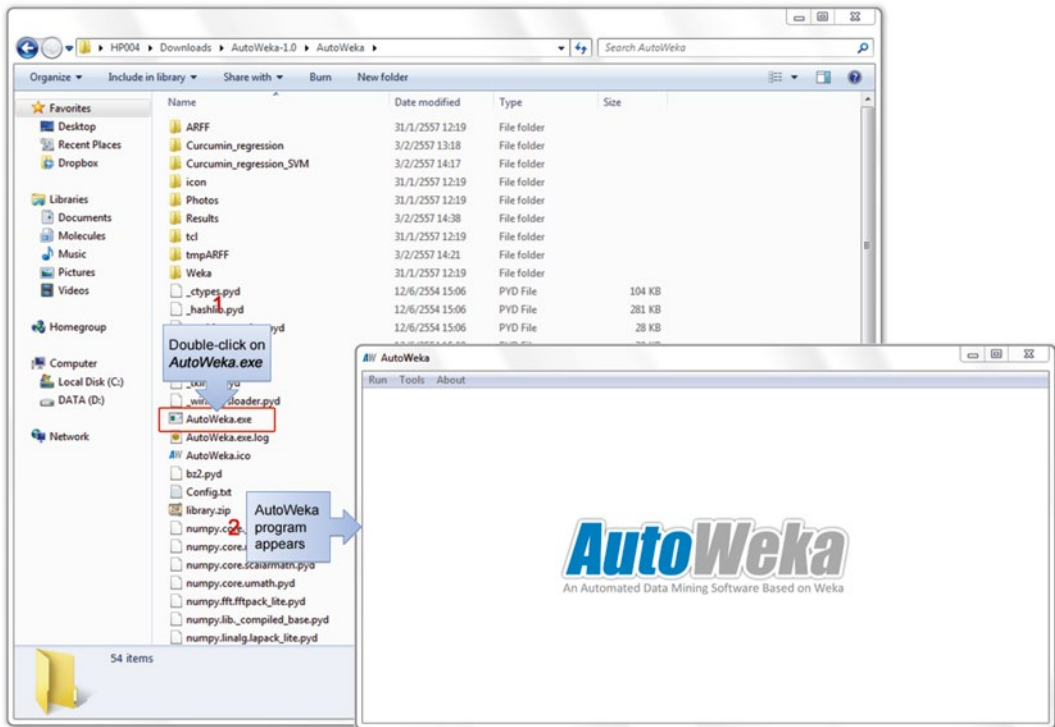


Fig. 9 Screenshot of the contents of the zip file of AutoWeka software (*left panel*) and upon opening the AutoWeka program (*right panel*)

folder to show a sub-folder that is automatically named according to the learning method used and the selected parameters. For example, if ANN was selected as the learning method and hidden nodes of 1–25 was chosen then the folder name would start with *ANN_Hidden_1_to_25*. The same convention also applies to the other parameters that will be appended at the end of the abovementioned folder name.

Inside the ANN results folder (Fig. 15), there are three sub-folders corresponding to the three ANN parameters, accordingly named as *HiddenNode*, *LearningAndMomentum*, and *TrainingTime*. Double-clicking on one of the sub-folders reveals a collection of sequentially numbered files where one parameter setting will generate one calculation output file. Thus, for an investigation of 25 hidden nodes (also bearing in mind that for each parameter investigated, ten separate runs are performed owing to the inherently random nature of the weight initialization of the back-propagation algorithm of ANN), a total of $25 \times 10 = 250$ output files will be generated. Subsequently, an average value for each investigated parameter will be derived from these ten calculations and saved into a new file called *AvgHidden.txt*. As data values within the *AvgHidden.txt* results file are in a tab-delimited format,

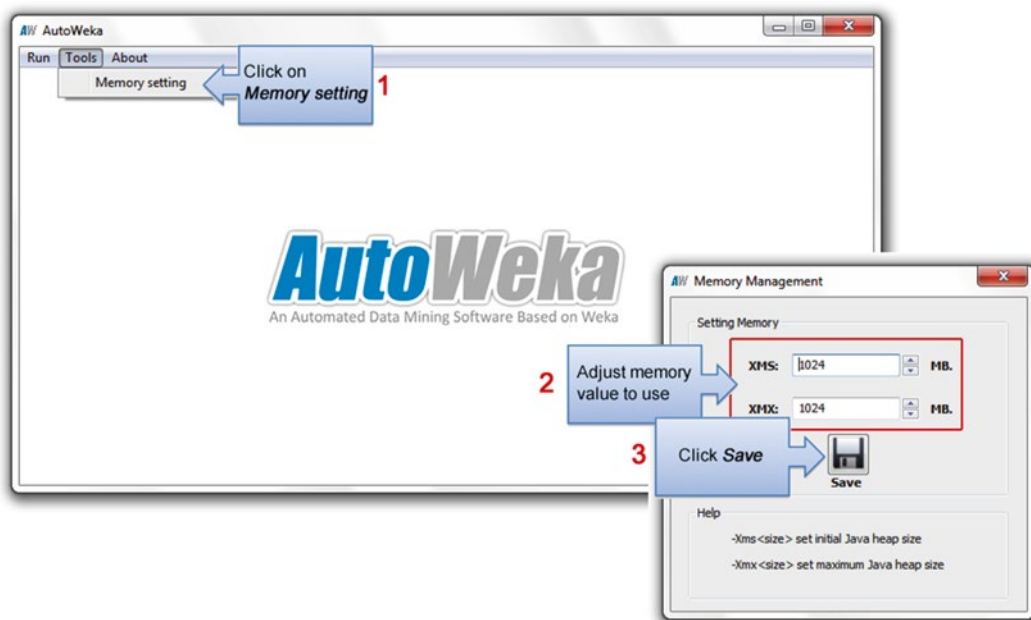


Fig. 10 Screenshot of adjusting the memory settings to be used by AutoWeka

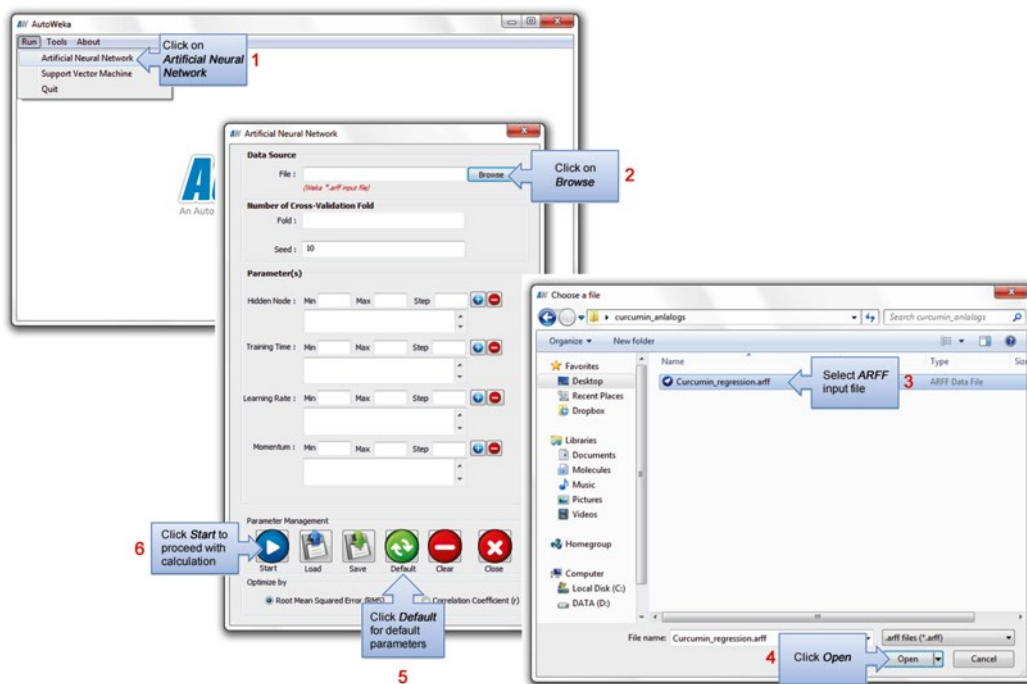


Fig. 11 Screenshot of setting up an ANN calculation

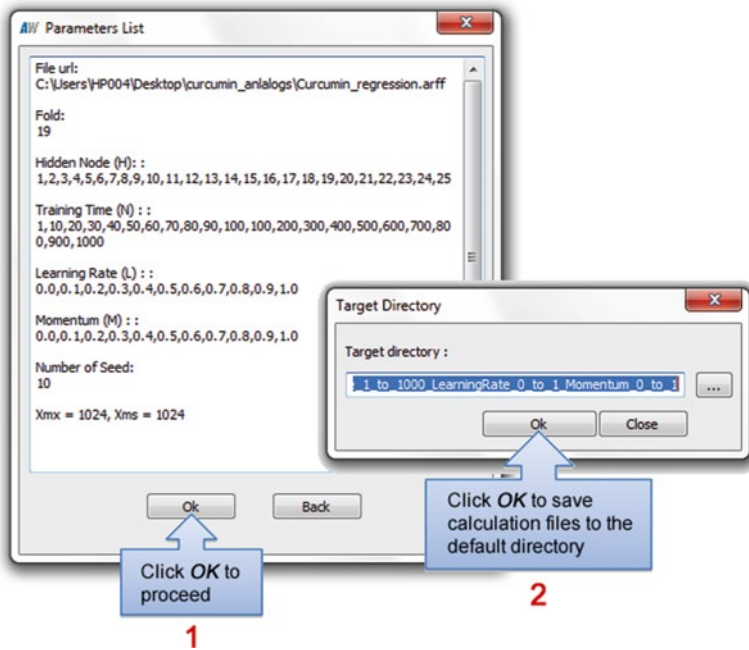


Fig. 12 Screenshot of pop-up windows that appear prior to ANN calculations

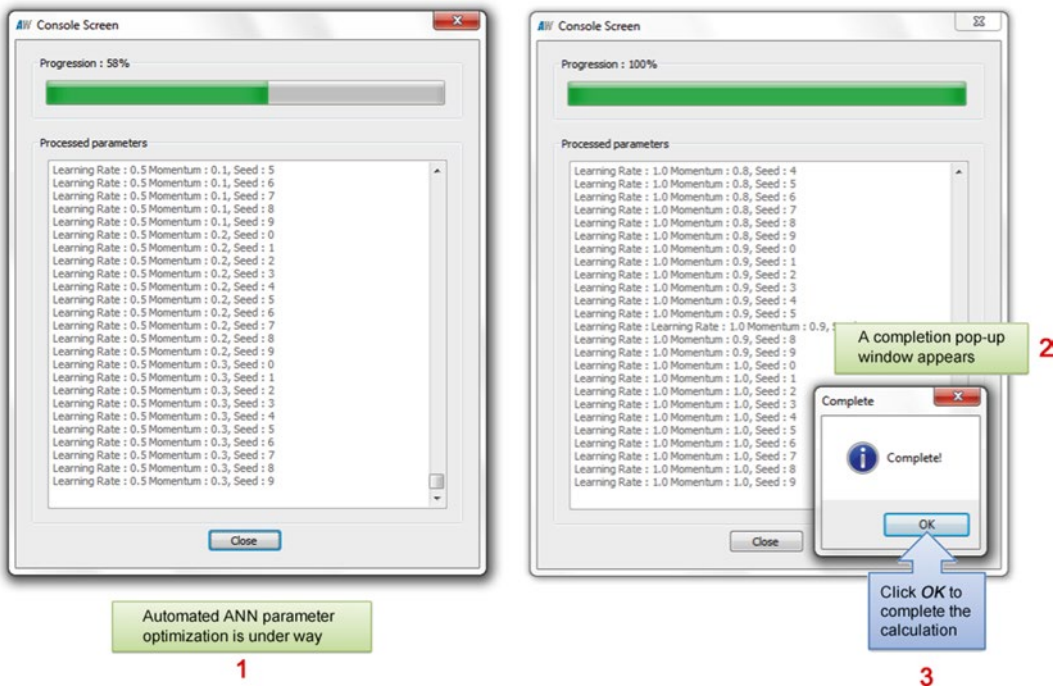


Fig. 13 Screenshot of an ongoing (left pane) and completed (right pane) ANN calculation

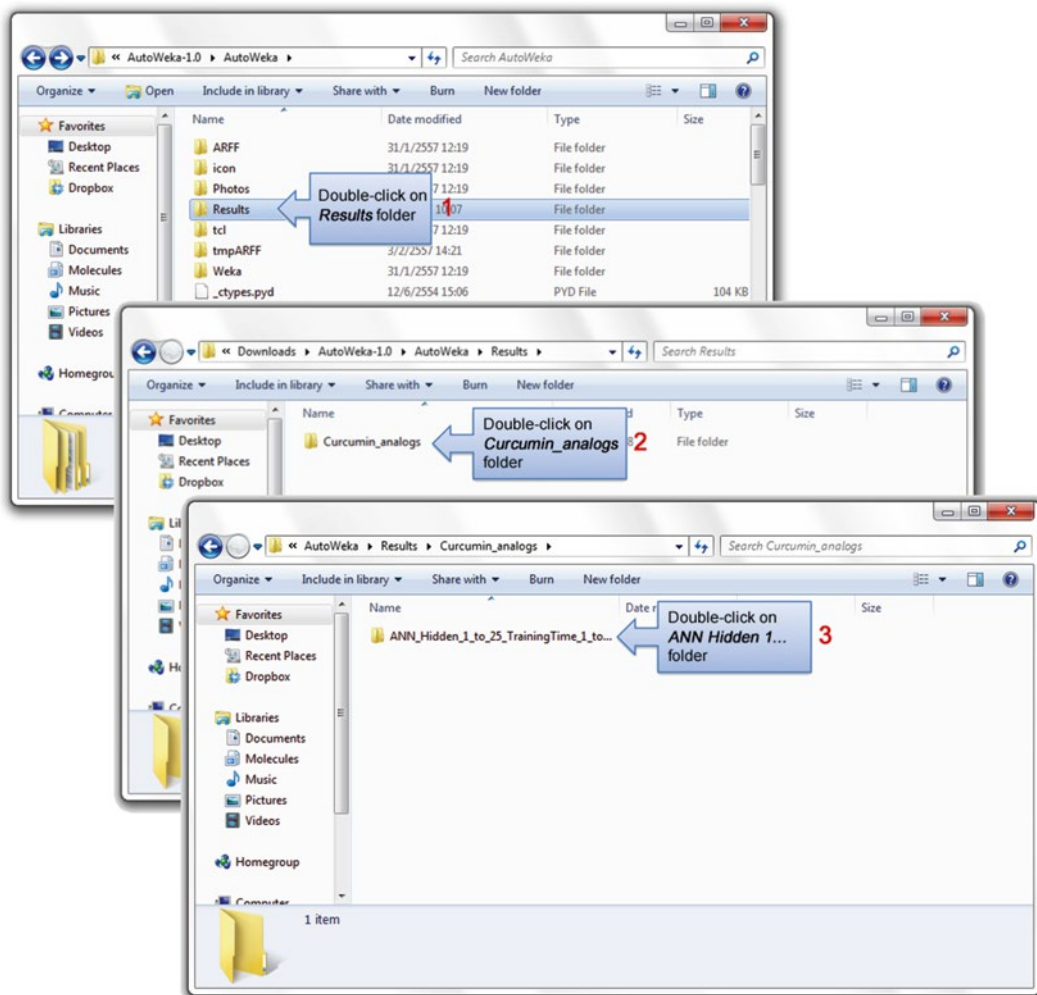
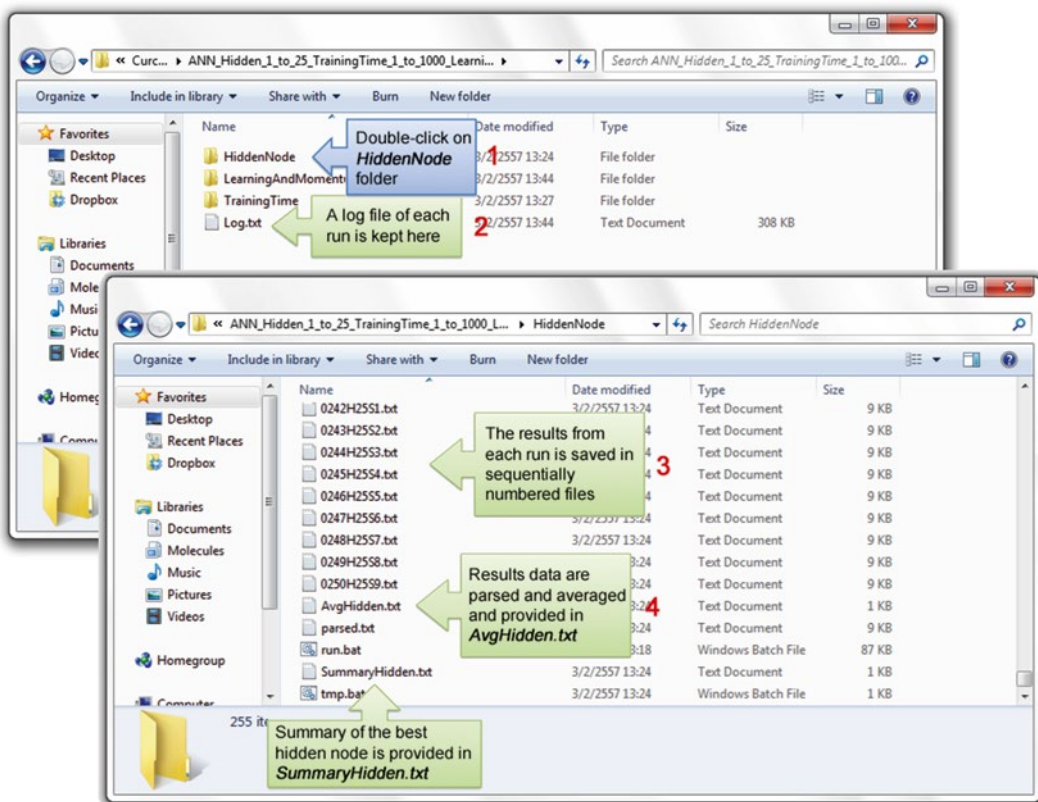


Fig. 14 Screenshot of the method for accessing the calculation results folder

it can be readily visualized by importing the file (or copying and pasting the data) into Microsoft Excel (Fig. 16). For convenience, the best parameter settings along with a summary of the statistical performance are provided in the *SummaryHidden.txt* file.

An alternative way of assessing the raw numerical data of the calculation results is to make a visual representation of it by creating graphical plots. This can be carried out by using the prewritten Python scripts or the graphical plot software mentioned in Subheading 2.6. Graphical plots created by the former are shown in Fig. 17.



5

Fig. 15 Screenshot of calculation results as divided by the three folders of the optimized parameters: HiddenNode, LearningRateAndMomentum, as well as TrainingTime. Shown are contents from the AvgHidden.txt file found in the HiddenNode folder. It should be noted that the same structure and organization of calculation results apply to the other two folders

	A	B	C	D	E
	Hidden Node	Training_correlation	Training_RMS	Testing_correlation	Testing_RMS
1	1	0.95245	0.16827	0.83159	0.26999
2	2	0.98218	0.10577	0.90972	0.1836
4	3	0.98582	0.09363	0.90431	0.19019
5	4	0.98494	0.09657	0.89656	0.19444
6	5	0.9865	0.08973	0.89388	0.1966
7	6	0.98583	0.09218	0.89449	0.1951
8	7	0.98392	0.10065	0.90638	0.18508
9	8	0.98447	0.0954	0.89864	0.19127
10	9	0.98442	0.0983	0.90487	0.18853

Fig. 16 Screenshot of AvgHidden.txt file as depicted in Microsoft Excel

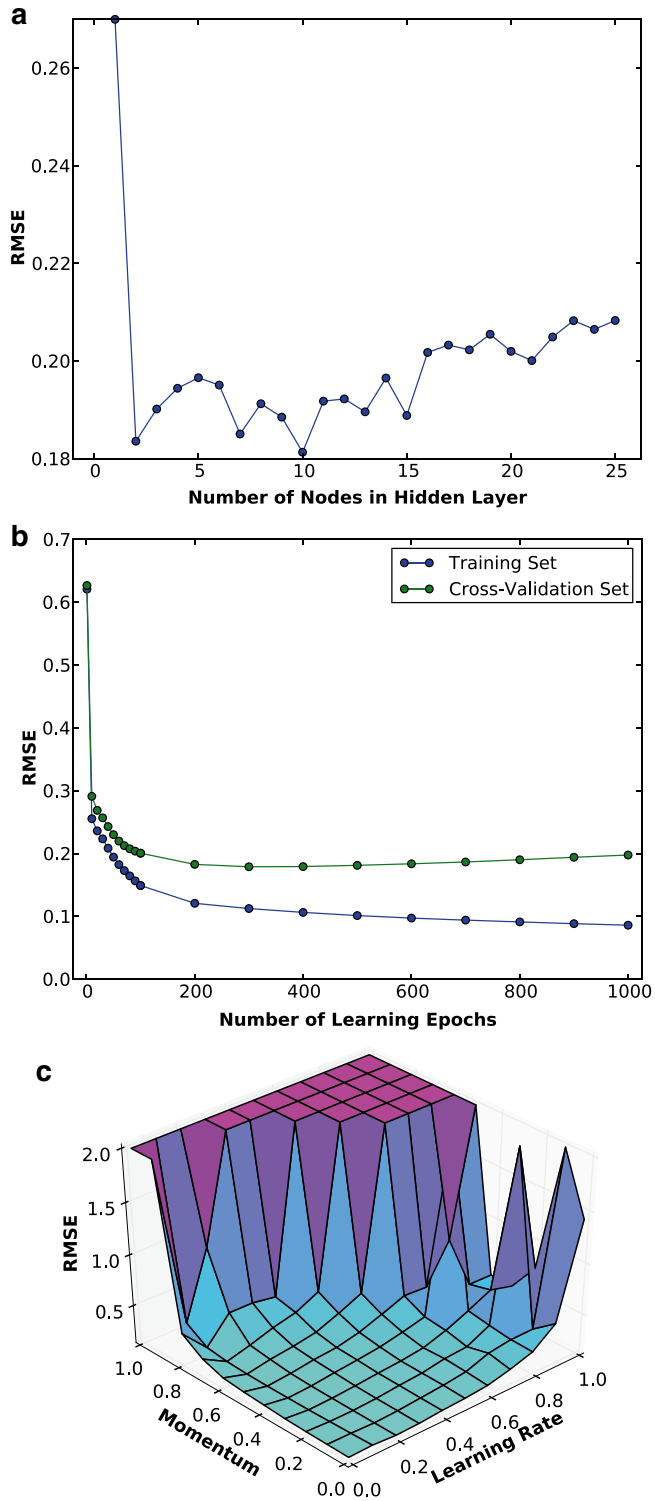


Fig. 17 Graphical plots from the parameter optimization process for (a) the number of hidden nodes, (b) the number of learning epochs, and (c) the learning rate and momentum

4 Notes

1. AutoWeka can be used in conjunction with Weka if users would like to benchmark their models with other learning algorithms.
2. In a typical QSAR modeling project, the rate-limiting step, that is, the lengthiest step, is generally that of data compilation and curation. It is here that critical errors (e.g., correctness of data entry) must be identified to ensure data integrity. The next most time-consuming step is the parameter optimization phase, in which several iterations of parameter fine-tuning are carried out to produce the best performance. AutoWeka handles the latter point automatically as it seeks the best set of parameters.

Acknowledgments

This work was supported by Mahidol University via the Goal-Oriented Research Grant to C.N.; postdoctoral fellowship to W.S.; research assistantships to P.M., S.J., and L.P.; and partial financial support to S.S.

References

1. Brodin A (1858) On the analogy of arsenic and phosphoric acid with respect to chemical and toxicology. Medico-Surgical Academy, St. Petersburg, Russia
2. Cros A (1863) Action de l'alcool amylique sur l'organisme. University of Strasbourg, Strasbourg
3. Kekulé A (1865) Sur la constitution des substances aromatiques. Bull Soc Chim Fr 3:98
4. Richardson B (1869) Physiological research on alcohols. Med Times Gaz 2:703–706
5. Richet C (1893) On the relationship between the toxicity and the physical properties of substances. Compt Rendus Seances Soc Biol 9:775–776
6. Overton E (1897) Osmotic properties of cells in the bearing on toxicology and pharmacology. Z Phys Chem 22:189–209
7. Meyer H (1899) On the theory of alcohol narcosis. Arch Exp Pathol Pharmacol 42:109–118
8. Moore W (1917) Volatility of organic compounds as an index of the toxicity of their vapors to insects. J Agric Res 10(7):365
9. Hammett LP (1937) The effect of structure upon the reactions of organic compounds. Benzene derivatives. J Am Chem Soc 59(1): 96–103
10. Taft RW (1952) Polar and steric substituent constants for aliphatic and o-benzoate groups from rates of esterification and hydrolysis of esters. J Am Chem Soc 74(12):3120–3128
11. Hansch C, Maloney PP, Fujita T et al (1962) Correlation of biological activity of phenoxyacetic acids with Hammett substituent constants and partition coefficients. Nature 194:178–180
12. Hansch C, Muir RM, Fujita T et al (1963) The correlation of biological activity of plant growth regulators and chloromycetin derivatives with Hammett constants and partition coefficients. J Am Chem Soc 85(18):2817–2824
13. Hansch C, Muir RM (1950) The ortho effect in plant growth-regulators. Plant Physiol 25(3):389
14. Hansch C, Fujita T (1964) ρ - σ - π analysis. A method for the correlation of biological activity and chemical structure. J Am Chem Soc 86(8): 1616–1626
15. Free SM Jr, Wilson JW (1964) A mathematical contribution to structure-activity studies. J Med Chem 7:395–399
16. Hansch C (1969) Quantitative approach to biochemical structure-activity relationships. Acc Chem Res 2(8):232–239

17. Nantasenamat C, Isarankura-Na-Ayudhya C, Naenna T et al (2009) A practical overview of quantitative structure-activity relationship. *Excli J* 8:74–88
18. Nantasenamat C, Isarankura-Na-Ayudhya C, Prachayasittikul V (2010) Advances in computational methods to predict the biological activity of compounds. *Expert Opin Drug Discov* 5(7):633–654
19. Medina-Franco JL, Martinez-Mayorga K, Bender A et al (2009) Characterization of activity landscapes using 2D and 3D similarity methods: consensus activity cliffs. *J Chem Inf Model* 49(2):477–491
20. Bajorath J (2012) Modeling of activity landscapes for drug discovery. *Expert Opin Drug Discov* 7(6):463–473
21. Doweyko AM (2008) QSAR: dead or alive? *J Comput Aided Mol Des* 22(2):81–89
22. Doweyko AM (2008) Is QSAR relevant to drug discovery? *IDrugs* 11(12):894–899
23. Tropsha A, Golbraikh A (2007) Predictive QSAR modeling workflow, model applicability domains, and virtual screening. *Curr Pharm Des* 13(34):3494–3504
24. Golbraikh A, Tropsha A (2002) Beware of q²! *J Mol Graph Model* 20(4):269–276
25. Huang J, Fan X (2011) Why QSAR fails: an empirical evaluation using conventional computational approach. *Mol Pharm* 8(2):600–608
26. Tropsha A, Gramatica P, Gombar VK (2003) The importance of being earnest: validation is the absolute essential for successful application and interpretation of QSPR models. *QSAR Comb Sci* 22(1):69–77
27. Tropsha A (2010) Best practices for QSAR model development, validation, and exploitation. *Mol Inf* 29(6–7):476–488
28. Fourches D, Muratov E, Tropsha A (2010) Trust, but verify: on the importance of chemical structure curation in cheminformatics and QSAR modeling research. *J Chem Inf Model* 50(7):1189–1204
29. Scior T, Bender A, Tresadern G et al (2012) Recognizing pitfalls in virtual screening: a critical review. *J Chem Inf Model* 52(4):867–881
30. Dearden JC, Cronin MT, Kaiser KL (2009) How not to develop a quantitative structure-activity or structure-property relationship (QSAR/QSPR). *SAR QSAR Environ Res* 20(3–4):241–266
31. Jewell NE, Turner DB, Willett P et al (2001) Automatic generation of alignments for 3D QSAR analyses. *J Mol Graph Model* 20(2):111–121
32. Tervo AJ, Nyronen TH, Ronkko T et al (2004) Comparing the quality and predictiveness between 3D QSAR models obtained from manual and automated alignment. *J Chem Inf Comput Sci* 44(3):807–816
33. Olah M, Bologa C, Oprea TI (2004) An automated PLS search for biologically relevant QSAR descriptors. *J Comput Aided Mol Des* 18(7–9):437–449
34. Bhonsle JB, Wang Z-X, Tamamura H et al (2005) A simple, automated quasi-4D-QSAR, quasi-multi way PLS approach to develop highly predictive QSAR models for highly flexible CXCR4 inhibitor cyclic pentapeptide ligands using scripted common molecular modeling tools. *QSAR Comb Sci* 24(5):620–630
35. Cartmell J, Enoch S, Krstajic D et al (2005) Automated QSPR through competitive workflow. *J Comput Aided Mol Des* 19(11):821–833
36. Zhang S, Golbraikh A, Oloff S et al (2006) A novel automated lazy learning QSAR (ALL-QSAR) approach: method development, applications, and virtual screening of chemical databases using validated ALL-QSAR models. *J Chem Inf Model* 46(5):1984–1995
37. Bhonsle JB, Bhattacharjee AK, Gupta RK (2007) Novel semi-automated methodology for developing highly predictive QSAR models: application for development of QSAR models for insect repellent amides. *J Mol Model* 13(1):179–208
38. Obrezanova O, Csanyi G, Gola JM et al (2007) Gaussian processes: a method for automatic QSAR modeling of ADME properties. *J Chem Inf Model* 47(5):1847–1857
39. Rodgers SL, Davis AM, Tomkinson NP et al (2007) QSAR modeling using automatically updating correction libraries: application to a human plasma protein binding model. *J Chem Inf Model* 47(6):2401–2407
40. Ma CY, Buontempo FV, Wang XZ (2008) Inductive data mining: automatic generation of decision trees from data for QSAR modelling and process historical data analysis. *Comput Aid Chem Eng* 25:581–586
41. Wood DJ, Buttar D, Cumming JG et al (2011) Automated QSAR with a hierarchy of global and local models. *Mol Inf* 30(11–12):960–972
42. Perez-Castillo Y, Lazar C, Taminau J et al (2012) GA(M)E-QSAR: a novel, fully automatic genetic-algorithm-(meta)-ensembles approach for binary classification in ligand-based drug design. *J Chem Inf Model* 52(9):2366–2386
43. Cox R, Green DV, Luscombe CN et al (2013) QSARworkbench: automating QSAR modeling to drive compound design. *J Comput Aided Mol Des* 27(4):321–336

44. Martins JPA, Ferreira MMC (2013) QSAR modeling: a new open source computational package to generate and validate QSAR models. *Quim Nova* 26:554–560
45. Hall M, Frank E, Holmes G et al (2009) The WEKA data mining software: an update. *SIGKDD Explorations* 11 (1)
46. Venkateswarlu S, Ramachandra MS, Subbaraju GV (2005) Synthesis and biological evaluation of polyhydroxycurcuminoids. *Bioorg Med Chem* 13(23):6374–6380
47. Worachartcheewan A, Nantasenamat C, Isarankura-Na-Ayudhya C et al (2011) Predicting the free radical scavenging activity of curcumin derivatives. *Chemometr Intell Lab Syst* 109(2):207–216
48. Mandi P, Nantasenamat C, Srungboonmee K et al (2012) QSAR study of anti-prion activity of 2-aminothiazoles. *Excli J* 11:453–467
49. Nantasenamat C, Isarankura-Na-Ayudhya C, Naenna T et al (2008) Prediction of bond dissociation enthalpy of antioxidant phenols by support vector machine. *J Mol Graph Model* 27(2):188–196
50. Nantasenamat C, Li H, Mandi P et al (2013) Exploring the chemical space of aromatase inhibitors. *Mol Div*. doi:[10.1007/s11030-11013-19462-x](https://doi.org/10.1007/s11030-11013-19462-x)
51. Nantasenamat C, Piacham T, Tantimongcolwat T et al (2008) QSAR model of the quorum-quenching *N*-acyl-homoserine lactone lactonase activity. *J Biol Syst* 16(2):279–293
52. Pingaew R, Tongraung P, Worachartcheewan A et al (2012) Cytotoxicity and QSAR study of (thio) ureas derived from phenylalkylamines and pyridylalkylamines. *Med Chem Res* 22:4016–4029
53. Prachayasittikul S, Wongsawatkul O, Worachartcheewan A et al (2010) Elucidating the structure-activity relationships of the vasorelaxation and antioxidation properties of thionicotinic acid derivatives. *Molecules* 15(1):198–214
54. Thippakorn C, Suksrichavalit T, Nantasenamat C et al (2009) Modeling the LPS neutralization activity of anti-endotoxins. *Molecules* 14(5):1869–1888
55. Worachartcheewan A, Nantasenamat C, Isarankura-Na-Ayudhya C et al (2013) Predicting antimicrobial activities of benzimidazole derivatives. *Med Chem Res* 22:5418–5430
56. Worachartcheewan A, Nantasenamat C, Naenna T et al (2009) Modeling the activity of furin inhibitors using artificial neural network. *Eur J Med Chem* 44(4):1664–1673
57. Nantasenamat C, Li H, Isarankura-Na-Ayudhya C et al (2012) Exploring the physico-chemical properties of templates from molecular imprinting literature using interactive text mining approach. *Chemometr Intell Lab Syst* 116:128–136
58. Nantasenamat C, Isarankura-Na-Ayudhya C, Tansila N et al (2007) Prediction of GFP spectral properties using artificial neural network. *J Comput Chem* 28(7):1275–1289
59. Nantasenamat C, Naenna T, Isarankura N-AC et al (2005) Quantitative prediction of imprinting factor of molecularly imprinted polymers by artificial neural network. *J Comput Aid Mol Des* 19(7):509–524
60. Nantasenamat C, Isarankura-Na-Ayudhya C, Naenna T et al (2007) Quantitative structure-imprinting factor relationship of molecularly imprinted polymers. *Biosens Bioelectron* 22(12):3309–3317
61. Nantasenamat C, Srungboonmee K, Jamsak S et al (2013) Quantitative structure-property relationship study of spectral properties of green fluorescent protein with support vector machine. *Chemometr Intell Lab Syst* 120:42–52
62. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys* 5(4):115–133
63. Lawrence J (1993) Introduction to neural networks: design, theory, and applications, 6th edn. California Scientific Software, California
64. Smith M (1993) Neural networks for statistical modeling. Van Nostrand Reinhold, New York
65. Bishop CM, Nasrabadi NM (2006) Pattern recognition and machine learning, vol 1. Springer, New York
66. Vapnik V (2000) The nature of statistical learning theory. Springer, New York
67. Vapnik V (1998) Statistical learning theory. Wiley, New York
68. Cortes C, Vapnik V (1995) Support-vector networks. *Mach Learn* 20(3):273–297
69. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
70. Platt JC (1999) Fast training of support vector machines using sequential minimal optimization. In: Schoelkopf B, Burges C, Smola A (eds) *Advances in kernel methods: support vector learning*. MIT Press, Cambridge, USA, pp 185–208
71. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. *Stat Comput* 14(3):199–222

Ligand Biological Activity Predictions Using Fingerprint-Based Artificial Neural Networks (FANN-QSAR)

Kyaw Z. Myint and Xiang-Qun Xie

Abstract

This chapter focuses on the fingerprint-based artificial neural networks QSAR (FANN-QSAR) approach to predict biological activities of structurally diverse compounds. Three types of fingerprints, namely ECFP6, FP2, and MACCS, were used as inputs to train the FANN-QSAR models. The results were benchmarked against known 2D and 3D QSAR methods, and the derived models were used to predict cannabinoid (CB) ligand binding activities as a case study. In addition, the FANN-QSAR model was used as a virtual screening tool to search a large NCI compound database for lead cannabinoid compounds. We discovered several compounds with good CB2 binding affinities ranging from 6.70 nM to 3.75 μ M. The studies proved that the FANN-QSAR method is a useful approach to predict bioactivities or properties of ligands and to find novel lead compounds for drug discovery research.

Key words Quantitative structure-activity relationship (QSAR), Fingerprint, Artificial neural networks (ANN), Biological activity, Cannabinoid

1 Introduction

Quantitative structure-activity relationship (QSAR) studies play essential roles in pharmaceutical research to identify and generate high-quality leads in the early stages of drug discovery [1–3]. QSAR studies help reduce the costly failures of drug candidates by identifying promising lead compounds and reducing the number of costly experiments. Such studies correlate chemical or structural features of compounds with their bioactivities. Descriptors are used to encode molecular features, and a mathematical relationship among a set of descriptors and biological endpoints of existing compounds is derived to construct a QSAR model which is then used to predict activities of new structures. Hence it is important to have a robust model which can learn important molecular features from a set of training compounds and then recognize such features in new structures to effectively predict their biological endpoints or activities.

The artificial neural networks (ANN) method is a machine learning algorithm with adaptive learning behavior in which the algorithm learns from previous examples and adapts to changes in input parameters. In comparison with receptor docking search [4], ligand similarity search [5, 6], machine-learning-based search [7, 8], and QSAR approaches [9–11], the ANN method possesses good generalization and pattern recognition ability for unseen data. Such features make it effective and robust for nonlinear regression problems with multiple inputs. In fact, several studies have used the ANN algorithm to predict molecular properties or biological endpoints of chemical analogs in several case studies such as anti-diabetes, anticancer, anti-HIV, and allergenicity prediction [12–16].

In this chapter, we focus on the fingerprint-based ANN-QSAR (FANN-QSAR) research work [17] in predicting biological activities of structurally diverse cannabinoid (CB) ligands using ANN. To the best of our knowledge, there have been no previous studies which have used molecular fingerprints as descriptors to predict biological activities (such as pIC_{50} or pK_i), although a few studies have been reported to predict ligand classes [18, 19]. Three types of molecular fingerprints were used as network inputs to train ANN-QSAR models, and the results were compared to well-known 2D and 3D QSAR methods [1] using five data sets. As a case study, we used the FANN-QSAR method to predict binding affinities of cannabinoid ligands using a large and structurally diverse CB ligand data set [20]. In addition, we applied the method as a virtual screening tool to find new cannabinoid ligands from a large NCI database containing over 200,000 compounds, and we found three compounds with good cannabinoid receptor binding affinities which were experimentally validated. The results demonstrated that combination of molecular fingerprints and ANN can lead to a reliable and robust high-throughput virtual screening method which can be a useful tool in chemogenomics and computer-aided drug discovery research.

2 Methods

2.1 Data Sets

As shown in Table 1, a total of six data sets were used in this study. Five of them were compiled by Sutherland et al. [21] and were downloaded from their supplemental data. The sixth data set, cannabinoid receptor 2 (CB2) data set, was curated by the Xie lab [5, 20]. For this data set, if there were more than one reported CB2 activity for a ligand, an average activity was used.

Once the structure-activity relationship (SAR) data were collected, each data set was then divided into training, validation, and testing sets. For the ACE, AchE, BZR, COX2, and DHFR data sets, the same training and testing data sets provided by

Table 1
Summary of six data sets used in the FANN-QSAR model development

Target/receptor name	Number of inhibitors	pIC ₅₀ range	Reference
Angiotensin-converting enzyme (ACE)	114	2.1–9.9	[45]
Acetylcholinesterase (AChE)	111	4.3–9.5	[46, 47]
Benzodiazepine receptor (BZR)	147	5.5–8.9	[48]
Cyclooxygenase-2 (COX2)	282	4.1–9.0	[49–58]
Dihydrofolate reductase inhibitors (DHFR)	361	3.3–9.8	[59–63]
Cannabinoid receptor subtype-2 (CB2)	1,699	3.9–10.8	[20]

Table 2
Number of training, validation, and testing set compounds in each data set

	ACE	AChE	BZR	COX2	DHFR	CB2
Training set	69	67	89	170	214	1,361
Validation set	7	7	9	18	23	169
Test set	38	37	49	94	124	169
Total	114	111	147	282	361	1,699

Sutherland et al. were used in order to allow the direct comparison of FANN-QSAR models to 3D and 2D QSAR models reported [21]. For each data set, 10 % of randomly selected compounds from the training set were used as a validation set. For the cannabinoid data set, the training and test sets were randomly divided. The training set contained 80 % of the compounds while the test set contained 10 %; the other 10 % were used as a validation set. The training set was used to train the model while the validation set was used to prevent overfitting of the model. The test set was used as an external set to evaluate the generalization ability of the trained FANN-QSAR models. The numbers of compounds found in each training, validation, and test sets for each data set are summarized in Table 2. For statistical modeling, the process was repeated five times for each data set, resulting in five different pairs of randomly divided training and test sets.

2.2 Implementation of Fingerprint-Based Artificial Neural Network QSAR (FANN-QSAR)

A feed-forward back-propagation neural network method was implemented using MATLAB® R2007b Neural Network Toolbox [22, 23]. As shown in Fig. 1, there are three layers in the network: an input layer, a hidden layer, and an output layer. Three different types of molecular fingerprints, namely FP2 [24], MACCS [25],

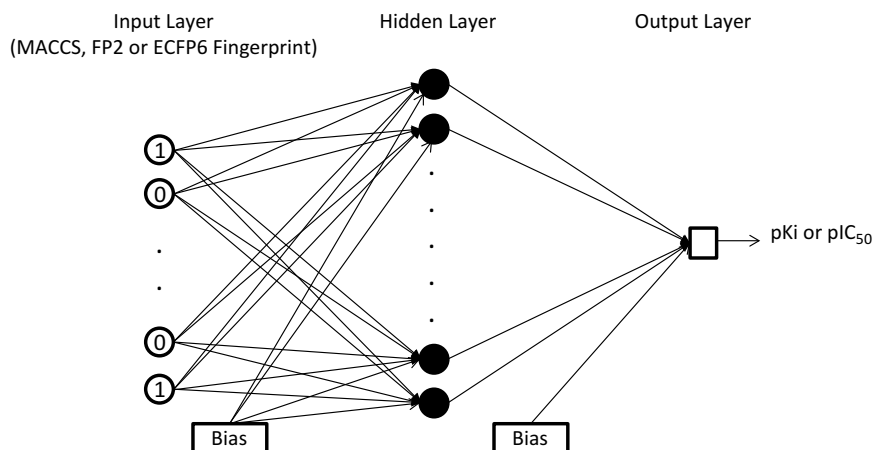


Fig. 1 Graphical representation of the fingerprint-based ANN-QSAR (FANN-QSAR) model. (Reprinted with permission from ref. 17. Copyright (2012) American Chemical Society)

and Extended-Connectivity Fingerprint (ECFP6) [26], were used in this study. The generation of fingerprints is described in Subheading 3. The number of input layer neurons was equal to the size of the fingerprint. For example, FP2 and ECFP6 fingerprints have 1,024 bits and therefore, the number of input neurons is equal to 1,024. Similarly, there are 256 input neurons for the MACCS fingerprint. The number of hidden layer neurons was varied between 100 and 1,000. The networks were trained using gradient descent with momentum training function (*traingdm*) to update weights and biases, the tangent sigmoid transfer function (*tansig*) for the hidden layer and the linear transfer function (*purelin*) for the output layer. 10 % randomly selected compounds from the training data were used as a validation set to decide when to stop training. The model training was stopped after 4,000 epochs (iterations) or if the mean square error (MSE) of prediction on the training set had reached the minimum value of 0.1. In addition, early stopping was enabled when the prediction error on the validation set increased for 300 epochs and the weights and biases at the minimum of the validation error were returned. The optimal number of hidden neurons was selected via cross-validation experiments in which the model was trained using different numbers of hidden neurons, and an average of training set and validation set mean squared errors (MSE) was calculated. The number of hidden neurons which gave the lowest average MSE was used as the optimal number for subsequent model testing on the test set. The mean squared error (MSE) is defined as

$$\text{MSE} = \frac{1}{T} \sum_{i=1}^T (t(i) - p(i))^2$$

where T is the total number of training samples, $t(i)$ is the target value of the i th sample, and $p(i)$ is the predicted value of the i th sample.

2.3 Comparison of the FANN-QSAR Model with Other Methods

The performance of the FANN-QSAR was compared to those of other reported 3D- and 2D-QSAR methods, including CoMFA [27], CoMSIA [28], Hologram QSAR (HQSAR) [29, 30], QSAR by eigenvalue analysis (EVA) [31], back-propagation feed-forward neural network implemented in Cerius2 using 2.5D descriptors (NN 2.5D), and ensemble neural network [32] (NN-ens) using 2.5D descriptors which were implemented and tested by Sutherland et al. [21]. Three different fingerprints, namely FP2, ECFP6, and MACCS, were used as inputs for FANN-QSAR models, and each model was trained separately for each fingerprint type. During each training process, a cross-validation experiment was performed to decide the optimal number of hidden neurons which was used subsequently on the test set prediction. To compare objectively, the FANN-QSAR models were trained and tested on the same training and test data sets provided by Sutherland et al. [21]. Three FANN-QSAR models were named as follows based on which molecular fingerprint was used as an input: ECFP6-ANN-QSAR, FP2-ANN-QSAR, and MACCS-ANN-QSAR. Results of CoMFA, CoMSIA basic, HQSAR, EVA, NN (2.5D), and NN-ens (2.5D) methods were taken from the work of Sutherland et al. [21].

Final correlation coefficient (r^2 test) values of each data set are listed in Table 3. Comparisons of r^2 (test) values across all data sets show that ECFP6 fingerprint-based ANN-QSAR model (ECFP6-ANN-QSAR) performed better than FP2 and MACCS fingerprint-based models for all data sets. For ACE, AchE, and COX2 data sets, the CoMFA model performed better than ECFP6-ANN-QSAR model but by a small margin. The ECFP6-ANN-QSAR model performed better for the DHFR and BZR data sets. Performance of the CoMSIA model was similar to that of the ECFP6-ANN-QSAR model. It is important to note that CoMFA

Table 3
Comparison of results from different QSAR methods

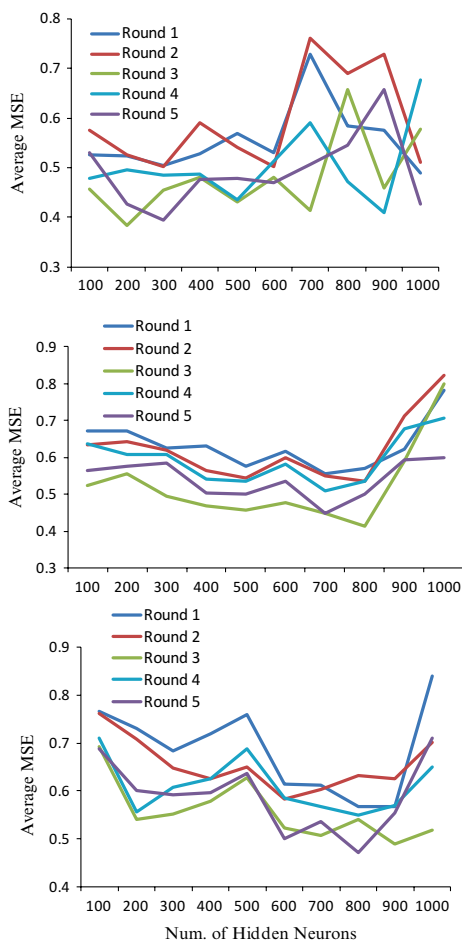
	ECFP6-ANN-QSAR	FP2-ANN-QSAR	MACCS-ANN-QSAR	CoMFA	CoMSIA basic	HQSAR	EVA	NN (2.5D)	NN-ens (2.5D)
ACE	0.41	0.2	0.08	0.49	0.52	0.3	0.36	0.39	0.51
AchE	0.43	0.13	0.04	0.47	0.44	0.37	0.28	-0.04	0.21
BZR	0.31	0.08	0.06	0	0.08	0.17	0.16	0.39	0.34
COX2	0.28	0.22	0.23	0.29	0.03	0.27	0.17	0.31	0.32
DHFR	0.63	0.43	0.48	0.59	0.52	0.63	0.57	0.42	0.54

and CoMSIA are field-based 3D QSAR methods which require similar scaffolds and high-quality molecular alignments to make effective predictions [11]. On the other hand, ECFP6-ANN-QSAR is a fingerprint-based method which works on structurally diverse data sets and requires no alignment during the model training process, which makes it more robust and high throughput in virtual screening. However, different fingerprints can produce different results and, in our work, ECFP6 produced an overall better result across different data sets compared to FP2 and MACCS fingerprints. In addition to 3D QSAR methods, the FANN-QSAR models were compared to another 2D QSAR method known as hologram QSAR (HQSAR), which is based on molecular holograms containing counts of molecular fragments similar to fingerprints. It can be observed that ECFP6-ANN-QSAR performed consistently better than HQSAR in all data sets except for DHFR data set resulting in the same r^2 test value (0.63). The FANN-QSAR models were also compared to other neural network approaches that used 2.5D descriptors as reported by Sutherland et al. The ECFP6-ANN-QSAR model performed better than the NN (2.5D) method in three out of five data sets and an ensemble of ten neural networks (NN-ens) approach using 2.5D descriptors performed slightly better than ECFP6-ANN-QSAR model in three out of five data sets. It is important to note that all QSAR models failed for COX2 and BZR data sets (r^2 test < 0.34) and had moderate performances (r^2 test < 0.64) for the other three data sets. Overall, the ECFP6-ANN-QSAR model performed consistently across all data sets and its performance was comparable to other 3D, 2D, and neural networks QSAR methods previously reported.

2.4 Cannabinoid Receptor Binding Activity Prediction for Known Cannabinoid Ligands

We extended the application of FANN-QSAR by predicting the binding activities of cannabinoid ligands. A total of 1,699 structurally diverse cannabinoid ligands with reported CB2 binding affinities were used. The CB2 binding activity data were downloaded from the CBID data set compiled by the Xie lab (<http://www.cbligand.org/cbid/index.php>). The ligands were randomly divided into training and test sets. FANN-QSAR models using different fingerprints were trained on training sets and the optimal numbers of hidden neurons were selected via cross-validation. Figure 2 contains a summary of cross-validation results for all three FANN-QSAR models. It can be observed that different training and test sets as well as different types of fingerprints resulted in different optimal numbers of hidden neurons, which suggested that cross-validation experiments are necessary to train neural networks for the best results.

After such training and parameter tuning, the predictive accuracy of the final model on the test set was evaluated. The process was repeated five times and a summary of r^2 values from each round of experiments can be seen in Table 4. Within each round, the



ECFP6-ANN-QSAR	Optimal No. of Hidden Neurons	Training Set MSE	Validation Set MSE	Average MSE
Round 1	1000	0.182	0.795	0.488
Round 2	600	0.257	0.749	0.503
Round 3	200	0.176	0.593	0.384
Round 4	900	0.223	0.594	0.409
Round 5	300	0.151	0.638	0.394

FP2-ANN-QSAR	Optimal No. of Hidden Neurons	Training Set MSE	Validation Set MSE	Average MSE
Round 1	700	0.300	0.809	0.554
Round 2	800	0.360	0.713	0.537
Round 3	800	0.360	0.466	0.413
Round 4	700	0.305	0.716	0.511
Round 5	700	0.290	0.606	0.448

MACCS-ANN-QSAR	Optimal No. of Hidden Neurons	Training Set MSE	Validation Set MSE	Average MSE
Round 1	800	0.354	0.781	0.567
Round 2	600	0.384	0.783	0.583
Round 3	900	0.361	0.617	0.489
Round 4	800	0.352	0.749	0.550
Round 5	800	0.339	0.605	0.472

Fig. 2 Cross-validation results of each FANN-QSAR method on CB2 ligand data set. (Reprinted with permission from ref. 17. Copyright (2012) American Chemical Society)

same training and test compounds were used across all three FANN-QSAR models. For example, the same training and test compounds in Round 1 of the ECFP6-ANN-QSAR model were used in the Round 1 of the FP2-ANN-QSAR and MACCS-ANN-QSAR models. As shown in the table, the ECFP6-ANN-QSAR model consistently outperformed the FP2- and MACCS-ANN-QSAR models in all five rounds of experiments. The ECFP6-ANN-QSAR model achieved an average r^2 test value of 0.56 ($r=0.75$) across all repeat experiments compared with 0.48 ($r=0.69$) and 0.45 ($r=0.67$) for the FP2- and MACCS-ANN-QSAR models, respectively. Results showed that the ECFP6 fingerprint was better than FP2 and MACCS fingerprints for the cannabinoid data set as well as the other five data sets. In fact, it has been also reported that circular fingerprints such as ECFP6 fingerprints are found to be more useful in virtual screening and ADMET properties'

Table 4
A summary of the performance of each FANN-QSAR model on CB2 ligand data set

Round	r^2 Training	r^2 Test
ECFP6-ANN-QSAR		
1	0.86	0.55
2	0.81	0.63
3	0.87	0.53
4	0.84	0.56
5	0.89	0.54
FP2-ANN-QSAR		
1	0.78	0.55
2	0.74	0.60
3	0.74	0.38
4	0.77	0.46
5	0.79	0.40
MACCS-ANN-QSAR		
1	0.74	0.48
2	0.72	0.53
3	0.74	0.37
4	0.74	0.47
5	0.75	0.41

prediction studies [33, 34]. Our results suggested that an ECFP6 fingerprint-based ANN-QSAR model can be used in virtual screening of chemical ligands in a high-throughput manner since it only requires 2D fingerprints as inputs instead of 3D molecular alignments and bioactive conformations, as required by other 3D QSAR methods.

2.5 Cannabinoid Receptor Binding Activity Prediction on Newly Reported Cannabinoid Ligands

To more rigorously test the predictive ability of the FANN-QSAR method on new cannabinoid compounds which are not in the Xie group's cannabinoid ligand training data set, the most recently reported cannabinoid ligands and associated CB2 binding affinity data were downloaded from ChEMBL database [35]. These compounds were not found in the training (CBID) data set and were collected to be used as a new test set in order to evaluate the FANN-QSAR performance. The new test data set consisted of 295 compounds with reported CB2 K_i values which were then

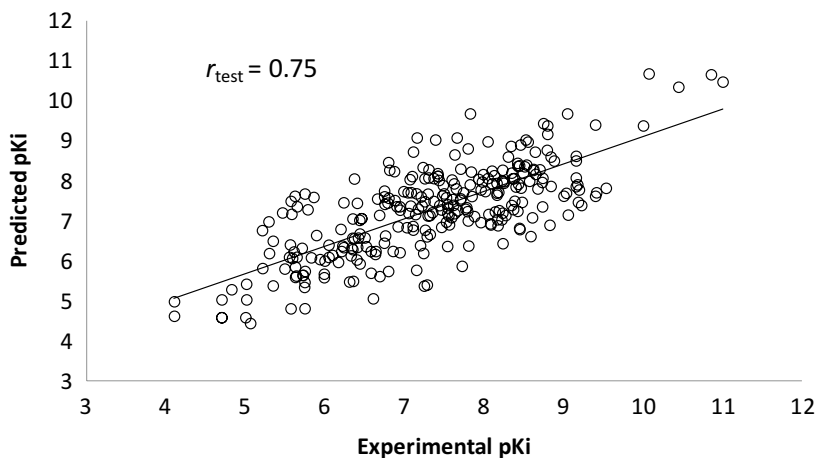


Fig. 3 Scatter plot between experimental pK_i and predicted pK_i values of 278 test cannabinoid ligands after the removal of 17 outliers. (Reprinted with permission from ref. 17. Copyright (2012) American Chemical Society)

converted to pK_i values. 41.55 % of new CB2 ligands were less than 80 % similar (2D Tanimoto similarity) and 25.34 % were less than 70 % similar to the training compounds. This similarity analysis indicated that the newly reported CB2 compounds contained a good mixture of similar and dissimilar compounds to the training database. The ECFP6-ANN-QSAR model was trained using the 1,699 CB2 ligand (CBID) data set. Twenty independent rounds of training and testing were performed. For each round, a randomly selected 90 % of the database was used for training and the remaining 10 % was used for validation. As a result, 20 independent trained models were derived. After 20 rounds of predictions, an average predicted value for each test compound was calculated. The average residual value was 0.046 and the standard deviation was 1.03. Seventeen outlier compounds with residuals more than two standard deviations away from the average residual were removed. Figure 3 shows a scatter plot of experimental and predicted pK_i values of 278 test compounds after such outlier removal. The linear regression of these 278 data points provided an r of 0.75, slope of 0.686, and y intercept of 2.249. This result indicated that there was a good correlation between experimental and predicted values, given the fact that many of these test compounds have novel structures and were not included in the model training and validation process. The result suggested that the FANN-QSAR possessed good generalization ability for newly reported cannabinoid ligands.

2.6 Virtual Screening of the NCI Compound Database for Lead Cannabinoid Ligands

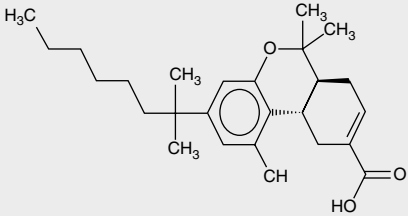
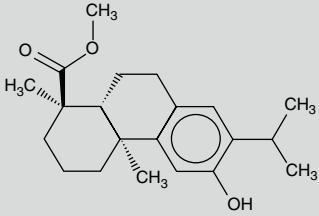
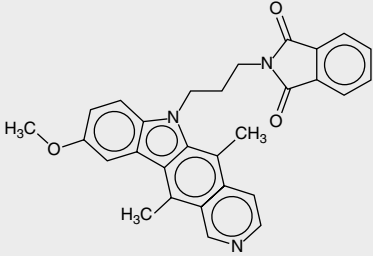
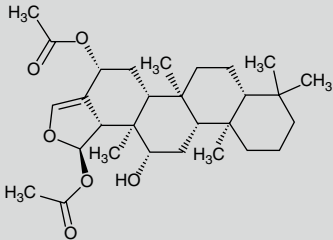
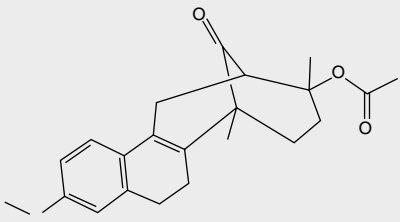
In order to illustrate how the FANN-QSAR model could be used in drug discovery research, we applied it as a virtual screening tool to search for CB2 lead ligands from the NCI compound database [36]. For consistency, the same 20 trained models in the

previous section were used. The NCI database, containing 329,089 compounds, was filtered to remove duplicate compounds, isotopes, metals, and mixtures using the Tripos Selector program [30, 37]. This filtering reduced it to 211,782 compounds that were used as a test set for each round of prediction. For each compound the ECFP6 fingerprint was generated and used as the network input to predict the CB2 receptor binding activity. After 20 rounds of predictions, an average predicted value for each compound was calculated. The top ranked 50 compounds were selected, but only 10 compounds were physically available from the NCI via material transfer agreement (MTA). These ten compounds were experimentally tested for CB2 activities using a [^3H]CP-55940 competition binding assay experiment. The experimental protocol for this validation assay is described in Subheading 3.

Among the ten tested NCI compounds, four (NSC49888, NSC174122, NSC369049, and NSC76301) had CB2 K_i between 6.70 nM ($\text{p}K_i=8.17$) and 3.80 μM ($\text{p}K_i=5.42$). One compound, which has a similar chemical scaffold to the well-known cannabinoid ligand, delta-9-tetrahydrocannabinol, was found to be a high-affinity compound with an average CB2 K_i value of 6.70 nM ($\text{p}K_i=8.17$). These four compounds and other similar compounds (70 % 2D Tanimoto similarity threshold was used) [38] were not found in the training database. Among the top 50 ligands, there was one NCI compound (NSC768843) which was more than 90 % similar (Tanimoto coefficient ≥ 0.9) to a known classical cannabinoid ligand (CAS ID: 112830-95-2 or HU210), an analog of delta-9-tetrahydrocannabinol, reported in the literature [39]. These findings proved that the FANN-QSAR method can find not only novel compounds with good CB2 binding affinities but also compounds similar to known ligands from a testing database containing thousands of compounds with diverse scaffolds. Hit ligands with novel scaffolds can be used as lead compounds for further medicinal chemistry optimization and SAR studies, while hits similar to known ligands provide additional information for scaffold hopping and R-group variations which may be useful for medicinal chemists. Table 5 contains the structures of NCI hit compounds and their experimental $\text{p}K_i$ as well as predicted values. Apart for one compound (NSC746843) that was not available from NCI, the other four compounds were experimentally tested in our lab and competition binding curves are shown in Fig. 4.

It should be noted that the predicted $\text{p}K_i$ correlated well with experimental $\text{p}K_i$ for two of the five hit ligands but not for the other three ligands. This finding could be attributed to the experimental variability of the reported CB2 binding activities of training compounds among different research labs, or to a possible limitation of 2D fingerprint descriptors which considers individual

Table 5
Identified NCI hit compounds with CB2 binding activities

Structure	NSC ID	MW	ClogP	Experimental pK_i	Predicted pK_i
	746843	400.55	6.61	8.81 ^a	8.66
	49888	330.46	5.59	8.17 ^b	8.28
	174122	463.52	4.76	5.59 ^c	8.41
	369049	488.66	4.00	5.51 ^c	8.48
	76301	354.44	3.99	5.42 ^c	8.21

^aAn average literature reported K_i value of a known cannabinoid compound (HU210) which is more than 90 % similar to 746843

^bAn average K_i value of two independent experiments performed in duplicate

^cAn experimental K_i value of one experiment performed in duplicate

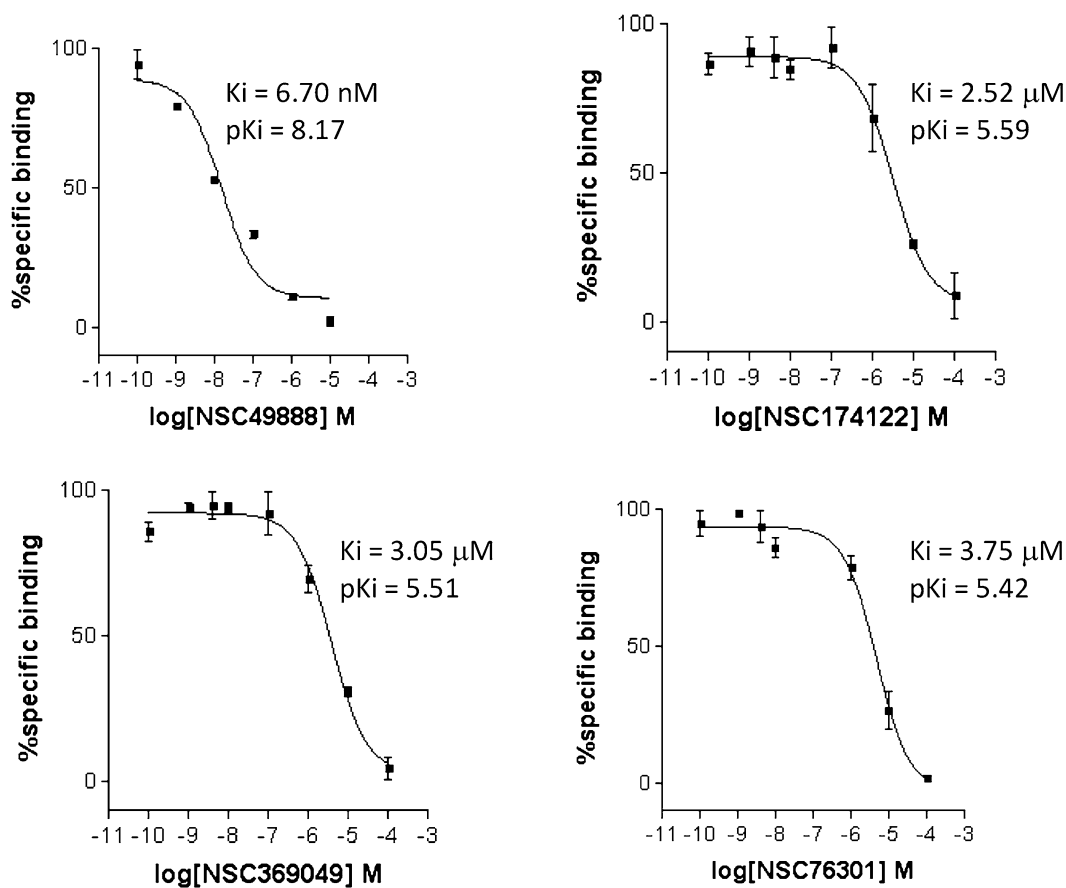


Fig. 4 CB2 receptor binding affinity K_i values of four NCI hit compounds measured by [3 H]CP-55940 radioligand competition binding assay using human CB2 receptors harvested from transfected CHO-CB2 cells. (Reprinted with permission from ref. 17. Copyright (2012) American Chemical Society)

fragment contributions but sometimes may not be as effective as other 3D descriptors when considering the overall structure of a ligand. Fingerprints such as ECFP6 have, however, been found to be useful in this study as well as in other several cheminformatics studies [5, 33, 34], and they are known to be robust and time efficient for high-throughput virtual screening applications where hundreds of thousands of chemicals are involved, as in this study. To conclude, results from the virtual screening exercise that was validated experimentally demonstrated that the derived FANN-QSAR model is capable of successfully identifying lead CB2 compounds with good binding affinities as well as compounds similar to known cannabinoid ligands, and providing additional insights for R-group and scaffold hopping of known ligands.

3 Notes

We used three different types of molecular fingerprints, namely FP2 [24], MACCS [25], and Extended-Connectivity Fingerprint (ECFP6) [26]. FP2 is a path-based fingerprint which indexes molecular fragments and MACCS is a key-based fingerprint which uses 166 predefined keys, whereas ECFP6 is a circular topological fingerprint which is derived using a variant of the *Morgan* algorithm [40]. FP2 and MACCS fingerprints were generated using the “*babel*” command from the OpenBabel program [24] while ECFP6 fingerprints were generated using the “*generatemd*” command from the ChemAxon program (<http://www.chemaxon.com>). Ligand chemical structures stored in SDF format were used as inputs to generate fingerprints. For each ligand, polar hydrogens were added using the OpenBabel program [24] before fingerprint generation. All fingerprints were fixed-length binary representations with 1,024 bits for both ECFP6 and FP2, and 256 bits for MACCS fingerprint. Fingerprints were generated for each ligand in the data sets and used as inputs to train the FANN-QSAR models.

In order to evaluate CB₂ binding activity of virtually screened ligands, competition binding assays were performed by displacing radioactive [³H]CP-55940 radioligand. The experimental protocol has been established based on previously reported procedures [41–44] and is described briefly below.

A Perkin Elmer 96-well TopCounter is used in our laboratory to measure the CB receptor binding affinity (K_i) of the *in silico*-screened ligands by displacing [³H]CP-55940. In competition binding experiments, ligands were diluted in dilution buffer (50 mM Tris, 5 mM MgCl₂, 2.5 mM EGTA) containing 0.1 % (w/v) fatty acid-free bovine serum albumin (BSA), 10 % dimethyl sulfoxide, and 0.4 % methyl cellulose. Various concentrations of ligands/samples are added in the same volume to 2.5 nM [³H]CP-55940. Incubation buffer (50 mM Tris, 2.5 mM EGTA, 5 mM MgCl₂, 0.1 % (w/v) fatty acid-free BSA) and cell membrane preparations from CHO cells expressing CB₂ receptors (5 µg per well) are added to a final volume of 200 µL. For the saturation binding experiments, varying concentrations of [³H]CP-55940 (0.05–4 nM) with or without 5 µM of an unlabeled known ligand (CP-55940) were incubated with the receptor membrane preparations to determine K_d and nonspecific binding. After the binding suspensions are incubated at 30 °C for 1 h, the reaction is terminated by rapid filtration through microfiltration plates (Unifilter GF/B filterplate, Perkin Elmer) followed by five washes with ice-cold TME buffer containing 0.1 % BSA on a Packard Filtermate Harvester (Perkin Elmer). The plates are then dried overnight and 30 µl MicroScint 0 scintillation liquid is added

to each well of the dried filter plates. Then the bound radioactivity is counted using a Perkin Elmer 96-well TopCounter. The K_i is calculated by using nonlinear regression analysis (Prism 5; GraphPad Software Inc., La Jolla, CA), with the K_d values for [^3H]CP-55940 determined from saturation binding experiments. This assay is used for determining binding affinity parameters (K_i) of ligand-receptor interactions between the CB2 receptor and ligands.

References

1. Myint KZ, Xie X-Q (2010) Recent advances in fragment-based QSAR and multi-dimensional QSAR methods. *Int J Mol Sci* 11(10): 3846–3866
2. Perkins R, Fang H, Tong W et al (2003) Quantitative structure-activity relationship methods: perspectives on drug discovery and toxicology. *Environ Toxicol Chem* 22(8): 1666–1679
3. Salum L, Andricopulo A (2009) Fragment-based QSAR: perspectives in drug design. *Mol Divers* 13(3):277
4. Chen JZ, Wang J, Xie XQ (2007) GPCR structure-based virtual screening approach for CB2 antagonist search. *J Chem Inf Model* 47(4):1626–1637. doi:10.1021/ci7000814
5. Wang L, Ma C, Wipf P et al (2013) TargetHunter: an in silico target identification tool for predicting therapeutic potential of small organic molecules based on chemogenomic database. *AAPS J* 15:395–406, PMID: 23292636
6. Tandon M, Wang L, Xu Q et al (2012) A targeted library screen reveals a new inhibitor scaffold for protein kinase D. *PLoS One* 7(9): e44653. doi:10.1371/journal.pone.0044653, PMID: 23028574
7. Ma C, Wang L, Yang P et al (2013) LiCABEDS II. Modeling of ligand selectivity for G-protein coupled cannabinoid receptors. *J Chem Inf Model* 53(1):11–26. doi:10.1021/ci3003914
8. Wang L, Ma C, Wipf P et al (2012) Linear and nonlinear support vector machine for the classification of human 5-HT1A ligand functionality. *Mol Inf* 31(1):85–95. doi:10.1002/minf.201100126
9. Myint K, Xie X-Q (2011) Fragment-based QSAR algorithm development for compound bioactivity prediction. *SAR QSAR Environ Res* 22(3):385–410
10. Chen JZ, Myint KZ, Xie X-Q (2011) New QSAR prediction models derived from GPCR CB2-antagonistic triaryl bis-sulfone analogues by a combined molecular morphological and pharmacophoric approach. *SAR QSAR Environ Res* 22(5–6):525–544. doi:10.1080/1062936x.2011.569948
11. Chen J-Z, Han X-W, Liu Q et al (2006) 3D-QSAR studies of arylpyrazole antagonists of cannabinoid receptor subtypes CB1 and CB2. A combined NMR and CoMFA approach. *J Med Chem* 49(2):625–636
12. Vilar S, Santana L, Uriarte E (2006) Probabilistic neural network model for the in silico evaluation of anti-HIV activity and mechanism of action. *J Med Chem* 49(3): 1118–1124. doi:10.1021/jm050932j
13. González-Díaz H, Bonet I, Terán C et al (2007) ANN-QSAR model for selection of anticancer leads from structurally heterogeneous series of compounds. *Eur J Med Chem* 42(5):580–585. doi:10.1016/j.ejmech.2006.11.016
14. Patra JC, Chua BH (2011) Artificial neural network-based drug design for diabetes mellitus using flavonoids. *J Comput Chem* 32(4): 555–567. doi:10.1002/jcc.21641
15. Dimitrov I, Naneva L, Bangov I et al (2014) Allergenicity prediction by artificial neural networks. *J Chemometrics*. doi:10.1002/cem.2597
16. Vanyúr R, Héberger K, Kövesdi I et al (2002) Prediction of tumoricidal activity and accumulation of photosensitizers in photodynamic therapy using multiple linear regression and artificial neural networks. *Photochem Photobiol* 75(5):471–478. doi:10.1562/0031-8655(2002)0750471potaaa2.0.co2
17. Myint K-Z, Wang L, Tong Q et al (2012) Molecular fingerprint-based artificial neural networks QSAR for ligand biological activity predictions. *Mol Pharm* 9(10):2912–2923. doi:10.1021/mp300237z
18. Molnár L, Keserű GM (2002) A neural network based virtual screening of cytochrome P450 3A4 inhibitors. *Bioorg Med Chem Lett* 12(3):419–421. doi:10.1016/s0960-894x(01)00771-5
19. Muresan S, Sadowski J (2005) “In-House Likeness”: comparison of large compound

- collections using artificial neural networks. *J Chem Inf Model* 45(4):888–893. doi:[10.1021/ci049702o](https://doi.org/10.1021/ci049702o)
20. Wang L, Xie XQ (2012) Cannabinoid Ligand Database. Accessed Nov 2011
21. Sutherland JJ, O'Brien LA, Weaver DF (2004) A comparison of methods for modeling quantitative structure–activity relationships. *J Med Chem* 47(22):5541–5554. doi:[10.1021/jm0497141](https://doi.org/10.1021/jm0497141)
22. Greenidge PA, Carlsson B, Bladh L-G et al (1998) Pharmacophores incorporating numerous excluded volumes defined by x-ray crystallographic structure in three-dimensional database searching: application to the thyroid hormone receptor. *J Med Chem* 41(14):2503–2512
23. Mathworks (2007) MATLAB. 7.5.0.342 (R2007b) edn, Natick, MA
24. O'Boyle N, Banck M, James C et al (2011) Open Babel: an open chemical toolbox. *J Cheminform* 3:33
25. Durant JL, Leland BA, Henry DR et al (2002) Reoptimization of MDL keys for use in drug discovery. *J Chem Inf Comput Sci* 42(6):1273–1280. doi:[10.1021/ci010132r](https://doi.org/10.1021/ci010132r)
26. Rogers D, Hahn M (2010) Extended-connectivity fingerprints. *J Chem Inf Model* 50(5):742–754. doi:[10.1021/ci100050t](https://doi.org/10.1021/ci100050t)
27. Cramer R, Patterson D, Bunce J (1988) Comparative molecular field analysis (CoMFA). 1. Effect of shape on binding of steroids to carrier proteins. *J Am Chem Soc* 110:5959–5967
28. Klebe G (1998) Comparative molecular similarity indices analysis: CoMSIA. 3D QSAR in Drug Design. Three-Dimensional Quantitative Structure Activity Relationships 3:87–104
29. Lowis D (1997) HQSAR: a new, highly predictive QSAR technique. Tripos Technical Notes 1(5):17
30. Jain AN (2004) Ligand-based structural hypotheses for virtual screening. *J Med Chem* 47(4):947–961
31. Ferguson AM, Heritage T, Jonathon P et al (1997) EVA: a new theoretically based molecular descriptor for use in QSAR/QSPR analysis. *J Comput Aided Mol Des* 11(2):143–152. doi:[10.1023/a:1008026308790](https://doi.org/10.1023/a:1008026308790)
32. Agrafiotis DK, Cedeño W, Lobanov VS (2002) On the use of neural network ensembles in QSAR and QSPR. *J Chem Inf Comput Sci* 42(4):903–911. doi:[10.1021/ci0203702](https://doi.org/10.1021/ci0203702)
33. Bender A, Jenkins JL, Scheiber J et al (2009) How similar are similarity searching methods? A principal component analysis of molecular descriptor space. *J Chem Inf Model* 49(1):108–119. doi:[10.1021/ci800249s](https://doi.org/10.1021/ci800249s)
34. Glem R, Bender A, Arnby C et al (2006) Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to ADME. *IDrugs* 9(3):199–204
35. Bellis LJ, Akhtar R, Al-Lazikani B et al (2011) Collation and data-mining of literature bioactivity data for drug discovery. *Biochem Soc Trans* 39(5):1365–1370. doi:[10.1042/BST0391365](https://doi.org/10.1042/BST0391365), [BST0391365](https://doi.org/10.1042/BST0391365) [pii]
36. Collins J, Crowell J (2011) Drug Synthesis and Chemistry Branch, Developmental Therapeutics Program (DTP), Division of Cancer Treatment and Diagnosis, National Cancer Institute. <http://dtp.nci.nih.gov/>
37. Tripos (2012) SYBYL-X 1.2. 1699 South Hanley Rd., St. Louis, Missouri, 63144, USA
38. Xie XQ, Chen JZ (2008) Data mining a small molecule drug screening representative subset from NIH PubChem. *J Chem Inf Model* 48(3):465–475. doi:[10.1021/ci700193u](https://doi.org/10.1021/ci700193u)
39. Huffman JW, Yu S, Showalter V et al (1996) Synthesis and pharmacology of a very potent cannabinoid lacking a phenolic hydroxyl with high affinity for the CB2 receptor. *J Med Chem* 39(20):3875–3877. doi:[10.1021/jm960394y](https://doi.org/10.1021/jm960394y)
40. Morgan HL (1965) The generation of a unique machine description for chemical structures—a technique developed at Chemical Abstracts Service. *J Chem Doc* 5(2):107–113. doi:[10.1021/c160017a018](https://doi.org/10.1021/c160017a018)
41. Gertsch J, Leonti M, Raduner S et al (2008) Beta-caryophyllene is a dietary cannabinoid. *Proc Natl Acad Sci* 105(26):9099–9104. doi:[10.1073/pnas.0803601105](https://doi.org/10.1073/pnas.0803601105)
42. Raduner S, Majewska A, Chen J-Z et al (2006) Alkylamides from Echinacea are a new class of cannabinomimetics: cannabinoid type 2 receptor-dependent and -independent immunomodulatory effects. *J Biol Chem* 281(20):14192–14206
43. Zhang Y, Xie Z, Wang L et al (2011) Mutagenesis and computer modeling studies of a GPCR conserved residue W5.43(194) in ligand recognition and signal transduction for CB2 receptor. *Int Immunopharmacol* 11(9):1303–1310. doi:[10.1016/j.intimp.2011.04.013](https://doi.org/10.1016/j.intimp.2011.04.013)
44. Yang P, Wang L, Feng R et al (2013) Novel triaryl sulfonamide derivatives as selective cannabinoid receptor 2 inverse agonists and osteoclast inhibitors: discovery, optimization, and biological evaluation. *J Med Chem* 56(5):2045–2058. doi:[10.1021/jm3017464](https://doi.org/10.1021/jm3017464)
45. DePriest SA, Mayer D, Naylor CB et al (1993) 3D-QSAR of angiotensin-converting enzyme and thermolysin inhibitors: a comparison of CoMFA models based on deduced and experimentally determined active site geometries.

- J Am Chem Soc 115(13):5372–5384. doi:[10.1021/ja00066a004](https://doi.org/10.1021/ja00066a004)
46. Sugimoto H, Tsuchiya Y, Sugumi H et al (1992) Synthesis and structure-activity relationships of acetylcholinesterase inhibitors: 1-benzyl-4-(2-phthalimidoethyl)piperidine, and related derivatives. J Med Chem 35(24):4542–4548. doi:[10.1021/jm00102a005](https://doi.org/10.1021/jm00102a005)
 47. Sugimoto H, Tsuchiya Y, Sugumi H et al (1990) Novel piperidine derivatives. Synthesis and anti-acetylcholinesterase activity of 1-benzyl-4-[2-(N-benzoylamino)ethyl]piperidine derivatives. J Med Chem 33(7):1880–1887. doi:[10.1021/jm00169a008](https://doi.org/10.1021/jm00169a008)
 48. Haefely W, Kyburz E, Gerecke M et al (1985) Recent advances in the molecular pharmacology of benzodiazepine receptors and in the structure-activity relationships of their agonists and antagonists. Adv Drug Res 14:165–322
 49. Chavatte P, Yous S, Marot C et al (2001) Three-dimensional quantitative structure-activity relationships of cyclo-oxygenase-2 (COX-2) inhibitors: a comparative molecular field analysis. J Med Chem 44(20):3223–3230. doi:[10.1021/jm0101343](https://doi.org/10.1021/jm0101343)
 50. Talley JJ, Brown DL, Carter JS et al (2000) 4-[5-Methyl-3-phenylisoxazol-4-yl]-benzenesulfonamide, Valdecoxib: a potent and selective inhibitor of COX-2. J Med Chem 43(5):775–777. doi:[10.1021/jm990577v](https://doi.org/10.1021/jm990577v)
 51. Huang H-C, Li JJ, Garland DJ et al (1996) Diarylspiro[2.4]heptenes as orally active, highly selective cyclooxygenase-2 inhibitors: synthesis and structure-activity relationships. J Med Chem 39(1):253–266. doi:[10.1021/jm950664x](https://doi.org/10.1021/jm950664x)
 52. Penning TD, Talley JJ, Bertenshaw SR et al (1997) Synthesis and biological evaluation of the 1,5-diarylpyrazole class of cyclooxygenase-2 inhibitors: identification of 4-[5-(4-Methylphenyl)-3-(trifluoromethyl)-1H-pyrazol-1-yl]benzenesulfonamide (SC-58635, Celecoxib). J Med Chem 40(9):1347–1365. doi:[10.1021/jm960803q](https://doi.org/10.1021/jm960803q)
 53. Li JJ, Norton MB, Reinhard EJ et al (1996) Novel terphenyls as selective cyclooxygenase-2 inhibitors and orally active anti-inflammatory agents. J Med Chem 39(9):1846–1856. doi:[10.1021/jm950878c](https://doi.org/10.1021/jm950878c)
 54. Li JJ, Anderson GD, Burton EG et al (1995) 1,2-Diarylcyclopentenes as selective cyclooxygenase-2 inhibitors and orally active anti-inflammatory agents. J Med Chem 38(22):4570–4578. doi:[10.1021/jm00022a023](https://doi.org/10.1021/jm00022a023)
 55. Reitz DB, Li JJ, Norton MB et al (1994) Selective cyclooxygenase inhibitors: novel 1,2-diarylcyclopentenes are potent and orally active COX-2 inhibitors. J Med Chem 37(23):3878–3881. doi:[10.1021/jm00049a005](https://doi.org/10.1021/jm00049a005)
 56. Khanna IK, Yu Y, Huff RM et al (2000) Selective cyclooxygenase-2 inhibitors: heteroaryl modified 1,2-diarylimidazoles are potent, orally active anti-inflammatory agents. J Med Chem 43(16):3168–3185. doi:[10.1021/jm0000719](https://doi.org/10.1021/jm0000719)
 57. Khanna IK, Weier RM, Yu Y et al (1997) 1,2-diarylimidazoles as potent, cyclooxygenase-2 selective, and orally active anti-inflammatory agents. J Med Chem 40(11):1634–1647. doi:[10.1021/jm9700225](https://doi.org/10.1021/jm9700225)
 58. Khanna IK, Weier RM, Yu Y et al (1997) 1,2-Diarylpyrroles as potent and selective inhibitors of cyclooxygenase-2. J Med Chem 40(11):1619–1633. doi:[10.1021/jm970036a](https://doi.org/10.1021/jm970036a)
 59. Gangjee A, Vidwans AP, Vasudevan A et al (1998) Structure-based design and synthesis of lipophilic 2,4-diamino-6-substituted quinazolines and their evaluation as inhibitors of dihydrofolate reductases and potential antitumor agents. J Med Chem 41(18):3426–3434. doi:[10.1021/jm980081y](https://doi.org/10.1021/jm980081y)
 60. Rosowsky A, Mota CE, Wright JE et al (1994) 2,4-Diamino-5-chloroquinazoline analogs of trimetrexate and piritrexim: synthesis and antifolate activity. J Med Chem 37(26):4522–4528. doi:[10.1021/jm00052a011](https://doi.org/10.1021/jm00052a011)
 61. Rosowsky A, Cody V, Galitsky N et al (1999) Structure-based design of selective inhibitors of dihydrofolate reductase: synthesis and antiparasitic activity of 2,4-diaminopteridine analogues with a bridged diarylamine side chain. J Med Chem 42(23):4853–4860. doi:[10.1021/jm990331q](https://doi.org/10.1021/jm990331q)
 62. Graffner-Nordberg M, Kolmodin K, Åqvist J et al (2001) Design, synthesis, computational prediction, and biological evaluation of ester soft drugs as inhibitors of dihydrofolate reductase from *Pneumocystis carinii*. J Med Chem 44(15):2391–2402. doi:[10.1021/jm010856u](https://doi.org/10.1021/jm010856u)
 63. Gangjee A, Elzein E, Queener SF et al (1998) Synthesis and biological activities of tricyclic conformationally restricted tetrahydropyridoannulated furo[2,3-d]pyrimidines as inhibitors of dihydrofolate reductases. J Med Chem 41(9):1409–1416. doi:[10.1021/jm9705420](https://doi.org/10.1021/jm9705420)

Chapter 10

GENN: A General Neural Network for Learning Tabulated Data with Examples from Protein Structure Prediction

Eshel Faraggi and Andrzej Kloczkowski

Abstract

We present a General Neural Network (GENN) for learning trends from existing data and making predictions of unknown information. The main novelty of GENN is in its generality, simplicity of use, and its specific handling of windowed input/output. Its main strength is its efficient handling of the input data, enabling learning from large datasets. GENN is built on a two-layered neural network and has the option to use separate inputs–output pairs or window-based data using data structures to efficiently represent input–output pairs. The program was tested on predicting the accessible surface area of globular proteins, scoring proteins according to similarity to native, predicting protein disorder, and has performed remarkably well. In this paper we describe the program and its use. Specifically, we give as an example the construction of a similarity to native protein scoring function that was constructed using GENN. The source code and Linux executables for GENN are available from Research and Information Systems at <http://mamiris.com> and from the Battelle Center for Mathematical Medicine at <http://mathmed.org>. Bugs and problems with the GENN program should be reported to EF.

Key words Neural network, Protein scoring, Windowed input, Automatic learning, GENN, Protein structure prediction

1 Introduction

Proteins perform their function through structure-based spatial and temporal interactions and give rise to the biosphere as we know it. More knowledge about them will contribute to our fundamental understanding of life and in practical terms will revolutionize medicine, bioscience, and other fields. However, in the context of present day computational and analytical tools they are too large and too complex. This is best exemplified in the ever growing gap between the number of known protein structures (relatively few) and the number of known protein sequences (relatively many). As of the end of 2013, there are under 100,000 structures deposited in the Protein Data Bank but over 33 million

protein sequences from about 30,000 different organisms in the NCBI RefSeq database. In many other areas of human endeavor constantly growing amount of information is being collected. Frequently, this information can be useful in other, related or unrelated, human endeavors [1–15]. Consumer purchasing history is an example where the past purchases of a consumer group can be used to assign a probability distribution of future purchases, and could, for example, give advertisers critical information for marketing strategy. In protein structure prediction too, experimentally solved protein structures can be used to infer the structure of unsolved proteins.

These vast amounts of data require specialized tools to extract useful information from them. Here we present one such tool that can be used to analyze large amounts of information and learn from it based on examples. The learning is achieved through the steepest descent training of a two layer neural network. It is able to efficiently handle instance-based and window-based training as will be described below. Its novelty is in its ability to handle any quantitative information, limited only by hardware, and that it can efficiently store window-based data, enabling modification of the modeling approaches without the need to modify the database. For large window-based training these are unique features as far as we could find. We term this tool GENN for GENEral Neural Network.

GENN was programmed in FORTRAN 90. In its design and implementation its efficiency and usability were of major concern. It is constructed out of several subroutines that process the data, initialize the model, and train it. It is also capable of producing predictions from existing single models or producing ensemble predictions with expected deviations. Its execution is terminal based. It was built on Ubuntu Linux, under the BASH (Bourne Again SHell) environment. For the rest we will use that environment to describe usage.

Although GENN can handle any data, it was designed and built for questions related to the relationship between the residue sequence of a protein and its native structure. Therefore we shall use terminology from that field to describe various features of the program. In general terms the problem is as follows. We are given the residue sequence of a protein, which can be derived from the genetic code. This residue sequence is in essence the chemical formula of the protein molecule. In turn we are to provide the best prediction for the physical structure of this molecule. However, since this is a difficult problem, certain approximations are usually made, either by coarse graining the full atom model, or by looking at derived structural properties such as the accessible surface area [16–26].

The general architecture of GENN is given in Fig. 1. The user supplied information includes the feature files which include information that will be used to establish a prediction, and the

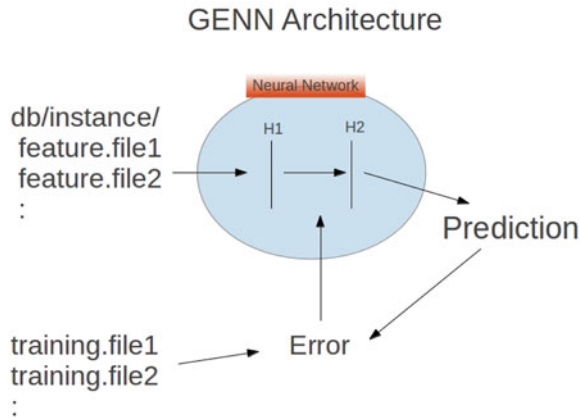


Fig. 1 Schematic diagram describing the architecture of GENN. The feature files include information used to establish a prediction. The training files contain information to be predicted. The error between the neural network predicted value and the training values is used to train its weights. H1 and H2 refer to the first and second hidden layers, respectively

training files which contain the information that is to be predicted. The error between the neural network predicted value and the training values is used to train the weights using the steepest descent back propagation method [21]. These weights are between the inputs through hidden layers one and two (H1 and H2, respectively, in Fig. 1) and to the prediction output.

2 Types of Training and Prediction in GENN

2.1 Instance-Based Prediction

We use the term *instance-based prediction* to describe cases in which each example to be learned from is independent of every other example in the database. This is in contrast to *windows-based prediction*, to be discussed next, where examples covered in overlapping windows will share common features. All programs related to instance-based prediction have a base name of “genn2inst.”

An example of instance-based prediction is prediction of global protein features. When one wants to predict quantities for entire protein sequences, for example the radius of gyration, one is dealing with instance-based prediction. For each protein, various input features are gathered but in general different proteins will have unique input features that are not shared by other proteins. Another example we have implemented using GENN, that will be described below, is that of predicting the fitness of model protein structures to the native state. Such a scheme was implemented successfully using GENN for the CASP10 experiment [27].

2.2 Window-Based Prediction

Window-based prediction is used to describe cases where different instances, with different outputs, share common inputs. Maybe the easiest case to describe is that of predictions for properties of residues along a protein sequence. In this case, we use sequentially neighboring residues (within a window of a certain width) to obtain the input features, then as we go along the chain neighboring residues will share common input features. In this case it is wasteful to record the inputs for each instance (residue) separately. It is more efficient to only store the input/output features of the protein and extract the appropriate inputs and outputs appropriate for each residue. All programs related to window-based prediction have a base name of “genn2wind.” More discussion of window-based prediction is given in earlier works [21, 25, 28–31].

Boundary conditions are important for window-based prediction. For example, for residues near a protein terminus the sliding window may fall out of range. These considerations are internally controlled in GENN. Windows that encompass regions outside the boundary of the instance are truncated to include only those parts that fall within range.

Because of the relative complexity associated with partitioning a database of windowed instances we designed the genn2wind version to automatically perform an n -fold cross validation, where n is determined by the command line options. By default n is equal to ten and the last partition is taken as the test set. In this mode 5% out of the remaining training data is used for over-fit protection. If one wishes to use only an over-fit protection set without a test set, for example while developing a server, this can be achieved using the “-sv” option. This built-in n -fold cross validation mode is limited to the window-based programs. For the instance-based programs only an over-fit set is chosen by default.

3 Database and Initialization File

In general the database is composed of separate directories each representing a given instance, a protein in our example. Each directory contains the input and output files associated with each instance. In our example the output file contains a list of the normalized ASA values to be predicted for each residue. The exact details of this normalization will be given later in the text when RASA will be defined. The input files then should have quantities that have predictive power with respect to the desired output. Specific details of the input and output files will be given when discussing ASAquick.

The initialization file is used to describe the locations of the required information for a given run. An example is provided in Fig. 2. The first line in the file gives the path of the head directory

```
genn.in ✖
"/home/user/Super/db/sid40/"
blosnorm physpar
asa2minmax
pdb.1TGS1
pdb.1U09A
pdb.3VRDB
pdb.3RQ1A
pdb.3DODA
pdb.1SFUA
pdb.3SHYA
pdb.1KZQA
pdb.2F9ZC
pdb.3GI7A
pdb.2DTJA
pdb.3DWCA
pdb.3KQ5A
```

Fig. 2 Example of the input file for GENN. The first line gives the head directory where all the instance directories are. This should be enclosed in double quotes to preserve the slashes. The next line describes the inputs used separated by spaces. The third line lists the outputs to be predicted represented by file names separated by spaces. The remaining lines list the instances on which the training will be performed

where all the instance directories are located. This should be enclosed in double quotes to preserve the slashes in the path name. The next line describes the inputs used, which are assumed the same for all instances. Each input is represented by a file name that will contain its values for a particular instance in its instance directory. Multiple inputs should be separated by spaces. In a similar way the third line lists the outputs to be predicted, represented by files in the instance directory that contains these values. Multiple outputs should be separated by spaces. The design of GENN is intended to facilitate changing both inputs and outputs for testing and research. The remaining lines list the instances on which the training will be performed. For the case of instance-based prediction this lists individual instances. For window-based prediction this lists a collection of instances grouped together in single files, for example the residues in a given protein. To allow for user control GENN does not modify the order of the list of instances. Randomization of the order of training instances, which can be useful, is left to the user.

One should note that for window-based prediction, since each input file can contain many instances (residues) an additional requirement on the database is imposed. To ascertain a match between inputs and outputs for a given instance two identifiers are used (originally the crystallographer index and the residue type for proteins). During the reading of the database both identifiers are checked and the program halts on mismatches between different files.

4 Training and Prediction

The specifics of using GENN for training and predicting from data are given below.

4.1 *Training a Model*

The program is run from the command line. The options associated with training are given in Table 1. By default the program looks for a file called “genn.in” to read the training dataset structure from (initialization file). If this file does not exist in the directory it is necessary to include in the command line a “-I” option followed by the name of the initialization file.

Given additional data a model can be retrained by including it as the initial condition for the new round of training. This is achieved using the “-wf” option followed by the name of the file containing the weights and parameters of the trained model. In this case the weights and other model parameters are used as initial values and no randomization is carried out. Then new training is performed on the new instances.

4.2 *Predictions from Trained Models*

Once a model has been trained all parameters are stored in a file. This file then can be used to give a prediction for new instances. In addition a collection of models can be used to produce an average prediction with an accompanying standard deviation. This standard deviation can prove beneficial in cases where quality of prediction is desired, but this requires testing on specific cases. The basic options associated with prediction from a trained model are given in Table 1. Note that some options are common between training and prediction tasks, however, model parameters such as the activation parameter or others that are related to the model architecture are stored in, and read from, the model file and should be reissued only in the uncommon event that one desires to override their values.

5 Special Options

Several unique features are provided with GENN. These options grew out of several applications related to protein structure prediction [21, 25, 29–31]. In this section we outline their possible uses. Refer to Table 1 for the required syntax. These options are available for the window-based programs.

5.1 *Filter Network*

Some cases for window-based prediction can benefit from a filter, where predictions over a window are used as inputs for a follow-up prediction. Examples include protein secondary structure where a filter prediction layer is often used [21, 25, 29]. While a filter layer can always be run using a second-stage neural network, for the

Table 1
GENN options

Category ^a	Flag ^b	Description	Default ^c
	-l	db list file	genn.in
	-owf	Weights output file	
	-m	Maximum number of epochs	1000
	-np	Number of database files to use	whole db
	-r	Random seed	time dependent
	-cv	Test fold index	10
	-f	Fraction of database for test fold	0.1
	-wi	Input window size	21
	-wo	Output window size	1
	-h1	Number of hidden layer one nodes	21
	-h2	Number of hidden layer two nodes	21
Train	-mi	Maximum number of bad iterations	100
	-a	Activation parameter	0.2
	-u	Learning rate	0.001
	-p	Momentum	0.4
	-hf	Number of filter hidden layer nodes	11
	-nf	Run network filter	
	-gf	Guiding factor	2.d0
	-ng	Don't use distance guiding	
	-gi	Number of global inputs	0
	-go	Number of global outputs	0
	-df	Use degeneracy files	
	-pr1	Predict single file (give id)	
Predict	-prl	File of ids to predict	
	-dw	Not same weight types, reread database	
	-aw	Average over weights in file	
	-d	Database head directory	
	-wf	Weights input file	
Both	-so	Average over outputs (nvo)	
	-po	Probability output	
	-h	Print help	

Options available for GENN

^a We distinguish three types of categories for options. “Train” for training specific options. “Predict” for prediction specific options. And, “universal” for options that used in both

^b The flag or command line option

^c Default values, some flags either do not have a default but require a value or are logical and do not require a value

window-based programs a filter layer can be included by using an option in the command line. Refer to Table 1 for the appropriate syntax. By default the filter network runs with 11 nodes, this value can be changed as needed. Note that a specific name is used to save the filter model. When getting predictions from trained filter models the filter option should also be used.

5.2 Guided Learning

In some instances of window-based prediction the relevance of a given location to a particular prediction site can be dependent on the distance between the location and the prediction site. In these cases it may be helpful to guide the network in that direction. One way of achieving this is by introducing an extra set of weights that have such distance dependence [21]. This is the default behavior for the window-based program. An option is available to switch off this behavior.

5.3 Global Inputs and Outputs

For certain window-based prediction global variables may be important. We shall consider the case of protein secondary structure to illustrate the point. In a window-based approach the secondary structure of a given residue is considered in the context of a window around it. However, certain parameters such as the amino acid composition of the protein may contain information about general tendencies to form a specific protein class (all-alpha, all-beta, alpha/beta, or alpha + beta). This information would typically not be contained in a limited window view and hence can help the prediction. Indeed global input features can significantly contribute to the prediction accuracy. To use global inputs/outputs, one should store their values in files called `genn.gin/genn.gut`, respectively, in the directory corresponding to that instance. An option, “-gi” for global input and/or “-go” for global output, is required in the command line to include these global files. These options should be followed by the number of values to use from these global files enabling control of how much global information to use. Note that options for global files are only required for training. The input structure is recorded in the weight files and is retrieved from them when called to predict.

5.4 Degeneracy Training

In some cases of window-based prediction one may like to sample certain cases more often than others. For example, for disordered proteins one may like to sample more of the disordered residues in a given protein instance [31]. In instance-based prediction one can include instances and their occurrence at will by repeating them in the file “`genn.in`.” For window-based prediction to change the sampling frequency within an instance, one includes a file with an integer count of the number of times (including zero) to train on a given training output. Hence, the degeneracy file order should match the rest of the input/output and should include the corresponding indexes. Degeneracy files should be stored in instance directories under files named “`genn.dgn`,” and an option should be given for their use.

5.5 Types of Output

GENN is able to predict multiple outputs. This in turn enables specific functionality besides general multi-output prediction. The first option is to treat the multiple outputs as components of a multi-state probability vector and train, optimize, and report accordingly. While training is not affected by this option, optimization is carried on the ability of the trained network to correctly distinguish the most dominant component of the probability vector. Reporting also follows the same protocol. This behavior was found useful when an ancestor of GENN was used to predict protein secondary structure [25].

Another possibility is to predict several instances of the same output and generate an average prediction from the same trained network. This has proven beneficial in several instances of continuous value prediction [29]. It provides an extra layer of averaging, with more control over variation between prediction models and hence the ability to average over specific random errors in certain circumstances. This also generates an estimate for the prediction stability in the reported standard deviation over predictions and in this form may be beneficial in assigning prediction stability and accuracy.

6 Automated Learning

GENN was started as the first, and simplest, component of an automated learning machine. Automated learning here means being able to “sit” on a big database (e.g., the Internet) and answer in meaningful ways to questions posed by either humans or machines. A sort of Watson [[wikipedia.org/wiki/Watson_\(computer\)](http://wikipedia.org/wiki/Watson_(computer))] but without human design. Rather, continuously, progressively, and automatically developing an image of the database pertinent to the questions it was exposed to in its existence. Here we specifically mean a non-memorizing learner, it is expected that on certain questions, different learners will respond differently depending on their histories. In this respect GENN was set up to create local networks on the fly.

The other, more complex component of such a learning machine would require to extract information in a meaningful way to describe both input and output features of a general question/answer pair. Note that some questions have multiple acceptable answers. Also, it would require building a self-sustaining learning mechanism. Both these tasks are monumental and would require breaking down the various problems further.

7 Examples

In this section we describe work that grew out of this version of GENN. Older versions of GENN helped produce an NMR fluctuation predictor [30] and a protein disorder predictor termed

SPINE-D [31], as well as some other applications. GENN was also involved in various testing in several labs. SPINE-D participated in the CASP9 experiment and was ranked among the top five methods [32]. This version of GENN was also used to create ASAquick, a fast accessible surface area predictor that uses only single sequence information [33], and a knowledge-based protein scoring function termed Seder [27] which we will present here as an example of use of GENN. Seder participated in the CASP10 [32] experiment as the “Kloczkowski_Lab” group. It was ranked second in predicting the structure of the “hard” targets category [34], third for “all” targets. It was the only prediction approach that was ranked in the top three for both “hard” and “all” cases.

This is a good place to point out that for any practical problem, the machine learner is only secondary to the problem of representation and interpretation of the input and the output. Care and thought of how to present the input and extract the output can produce great improvements in learning and prediction quality. To a second degree, given the same input/output information, different quality learners can produce different outcomes. For example, it has been our general experience that smaller sets of data are better learnt by methods such as support vector machines while larger sets are better learnt by neural networks, the difference being several relative percent accuracy. On the other hand, for example, changes of the representation of the input/output for predicting dihedral angles of proteins has resulted in almost 100% reduction in the prediction error [18, 21, 28, 29].

Seder [27] is an example of an implementation of GENN. It was developed specifically for structured proteins, to rank structural models according to their similarity to a native structure. In the directory “example/Seder” of the source code distribution we give examples for the feature files used to make Seder. We use the same names to introduce them here, more information is available in the Seder paper [27]. The first line of the input file “genn.in” gives the directory location of the database of instances. In the second line of “genn.in” the inputs are listed. typ233w.norm contains information about the distance of individual atoms to the solvent. There are a total of 936 real numbers making up this feature. res2res refers to the partial energy sums between pairs of residues. There are a total of 441 real numbers making up this feature. 4bod, dfire2, and rwplus refer to the four-body [35–37], DFIRE2 [38, 39], and RWPlus [40] potentials, respectively. The desired output from the neural network, given on the third line, is a transformation of the TM-score [41, 42] used to measure similarity between native and model structures (decoys). For training we used server models from the Critical Assessment of protein Structure Prediction (CASP) [32] rounds 5 through 9 (94,717 structures). The fourth line of “genn.in” points to the single example of an instance given here. This points to subdirectories of

the “db” directory. Actual training will involve a list of such instances. We average over several training realizations with different initial conditions. For each, in addition to randomizing the initial weights, we also randomize the list of training proteins (PDB + CASP) and select thirty thousand of them. From this subset, 30% were used to compose the over-fit-protection set while the remaining 70% was used for online training. Over-fit testing was done after each training epoch.

The command line for training one realization of a Seder neural network is given by “nohup time./genn2inst.e -l seder.in -f 0.3 -mi 200 -h1 51 -h2 31 -r \$RANDOM &.” We have included three features of Linux/BASH that are not directly related to this work but are nonetheless useful. The first is the “nohup” command which allows a training of the weights even after logout or loss of connection. Note that you must have the ampersand symbol at the end of the line for this functionality. The second is the “time” command which is useful in estimating run times and optimization. The third is “\$RANDOM” which is a BASH reserved word for generation of pseudorandom numbers, it comes in handy in automation of neural network creation since seeding is done independently of your application and depends on the system state (hence arbitrary). The training command line given above will train a set of weights and output them with other necessary information for prediction into a file. By design the weights file name is constructed uniquely from the process ID and other parameters associated with the run. Here we shall assume its name to be “wei.out.” A single prediction from this file for a single protein with ID, PID, would look something like “./genn2inst.e -wf wei.out -prl PID.” Where it is assumed that the directory mentioned in the beginning of wei.out contains a subdirectory called PID with the necessary files. Note that GENN is designed to calculate prediction error, hence it will look for target files even if asked to predict. If such a target file does not exist, is not in your possession, or a complete blind test of the software is sought, trivial (zero filled) target files should be generated. Of course then the reported prediction errors are meaningless. For a collection of proteins with PIDs listed in the file “list.PID” and taking the average over a collection of weight files listed in file “list.wei” prediction and prediction variation are obtained via the command “./genn2inst.e -aw list.wei -prl list.PID.” Both these prediction methods produce a text output to screen which can be piped to a text file. If a list of proteins is given the first column in the output is the PID listed in “list.wei.” The program “awk” can then be easily used to generate individual prediction files if those are necessary. Note that we have given an example of instance-based prediction, and the syntax for window-based prediction is identical. Example files for window-based prediction are in the subdirectory “db/pdb.window.”

8 Summary

We have presented GENN, a general neural network designed to train on ad-hoc data. GENN was designed with efficiency and modularity in mind as part of a more complex algorithm of an automated learner. It can take any numerical input/output problem and prepare a corresponding, non-memorizing, model structure to represent this data. Data can be organized in files containing individual instances or a collection of ordered instances where each line is an individual input/output target. GENN is available from Research and Information Systems at <http://mamiris.com>, and from the Battelle Center for Mathematical Medicine at <http://mathmed.org>.

Acknowledgements

We gratefully acknowledge the financial support provided by the National Institutes of Health (NIH) through Grants R01GM072014 and R01GM073095 and the National Science Foundation through Grant NSF MCB 1071785. Both authors would like to thank the organizers of CASP10 conference in Gaeta, Italy, for inviting them to the conference and providing free registration to EF. EF would also like to thank Yaoqi Zhou and Keith Dunker for hosting him at IUPUI and general discussions.

References

1. Kassin SM (1979) Consensus information, prediction, and causal attribution: a review of the literature and issues. *J Pers Soc Psychol* 37:1966
2. Crick NR, Dodge KA (1994) A review and reformulation of social information-processing mechanisms in children's social adjustment. *Psychol Bull* 115:74
3. Fielding AH, Bell JF (1997) A review of methods for the assessment of prediction errors in conservation presence/absence models. *Environ Conserv* 24:38–49
4. Makhoul J (1975) Linear prediction: a tutorial review. *Proc IEEE* 63:561–580
5. Fontenot RJ, Wilson EJ (1997) Relational exchange: a review of selected models for a prediction matrix of relationship activities. *J Bus Res* 39:5–12
6. Rost B et al (2001) Review: protein secondary structure prediction continues to rise. *J Struct Biol* 134:204–218
7. Maier HR, Dandy GC (2000) Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environ Model Softw* 15: 101–124
8. Chang JC, Wooten EC, Tsimelzon A, Hilsenbeck SG, Gutierrez MC, Elledge R, Mohsin S, Osborne CK, Chamness GC, Allred DC et al (2003) Gene expression profiling for the prediction of therapeutic response to docetaxel in patients with breast cancer. *Lancet* 362:362–369
9. Schofield W et al (1985) Predicting basal metabolic rate, new standards and review of previous work. *Hum Nutr Clin Nutr* 39:5
10. Blundell T, Sibanda B, Sternberg M, Thornton J (1987) Knowledge-based prediction of protein structures. *Nature* 326:26
11. Chou PY, Fasman GD (1978) Empirical predictions of protein conformation. *Annu Rev Biochem* 47:251–276
12. Floudas C, Fung H, McAllister S, Mönnigmann M, Rajgaria R (2006) Advances in protein structure prediction and de novo protein design: a review. *Chem Eng Sci* 61:966–988

13. Moult J (2005) A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. *Curr Opin Struct Biol* 15:285–289
14. Vazquez A, Flammini A, Maritan A, Vespignani A (2003) Global protein function prediction from protein-protein interaction networks. *Nat Biotechnol* 21:697–700
15. Borgwardt KM, Ong CS, Schönauer S, Vishwanathan S, Smola AJ, Kriegel H-P (2005) Protein function prediction via graph kernels. *Bioinformatics* 21:i47–i56
16. Chothia C (1974) Hydrophobic bonding and accessible surface area in proteins. *Nature* 248:338–339
17. Moret M, Zebende G (2007) Amino acid hydrophobicity and accessible surface area. *Phys Rev E* 75:011920
18. Dor O, Zhou Y (2007) Real-spine: an integrated system of neural networks for real-value prediction of protein structural properties. *Proteins: Struct Funct Bioinf* 68:76–81
19. Durham E, Dorr B, Woetzel N, Staritzbichler R, Meiler J (2009) Solvent accessible surface area approximations for rapid and accurate protein structure prediction. *J Mol Model* 15: 1093–1108
20. Zhang H, Zhang T, Chen K, Shen S, Ruan J, Kurgan L (2009) On the relation between residue flexibility and local solvent accessibility in proteins. *Proteins: Struct Funct Bioinf* 76: 617–636
21. Faraggi E, Xue B, Zhou Y (2009) Improving the prediction accuracy of residue solvent accessibility and real-value backbone torsion angles of proteins by guided-learning through a two-layer neural network. *Proteins: Struct Funct Bioinf* 74:847–856
22. Zhang T, Zhang H, Chen K, Ruan J, Shen S, Kurgan L (2010) Analysis and prediction of RNA-binding residues using sequence, evolutionary conservation, and predicted secondary structure and solvent accessibility. *Curr Protein Pept Sci* 11:609–628
23. Gao J, Zhang T, Zhang H, Shen S, Ruan J, Kurgan L (2010) Accurate prediction of protein folding rates from sequence and sequence-derived residue flexibility and solvent accessibility. *Proteins: Struct Funct Bioinf* 78:2114–2130
24. Nunez S, Venhorst J, Kruse CG (2010) Assessment of a novel scoring method based on solvent accessible surface area descriptors. *J Chem Inf Model* 50:480–486
25. Faraggi E, Zhang T, Yang Y, Kurgan L, Zhou Y (2012) Spine x: improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *J Comput Chem* 33:259–267
26. Wang C, Xi L, Li S, Liu H, Yao X (2012) A sequence-based computational model for the prediction of the solvent accessible surface area for α -helix and β -barrel transmembrane residues. *J Comput Chem* 33:11–17
27. Faraggi E, Kloczkowski A (2013) A global machine learning based scoring function for protein structure prediction. *Proteins: Struct Funct Bioinf*. doi:10.1002/prot.24454
28. Xue B, Dor O, Faraggi E, Zhou Y (2008) Real value prediction of backbone torsion angles. *Proteins: Struct Funct Bioinf* 72:427–433
29. Faraggi E, Yang Y, Zhang S, Zhou Y (2009) Predicting continuous local structure and the effect of its substitution for secondary structure in fragment-free protein structure prediction. *Structure* 17:1515–1527
30. Zhang T, Faraggi E, Zhou Y (2010) Fluctuations of backbone torsion angles obtained from nmr-determined structures and their prediction. *Proteins: Struct Funct Bioinf* 78: 3353–3362
31. Zhang T, Faraggi E, Xue B, Dunker AK, Uversky VN, Zhou Y (2012) Spine-d: accurate prediction of short and long disordered regions by a single neural-network based method. *J Biomol Struct Dyn* 29:799–813
32. Moult J, Fidelis K, Kryshchuk A, Tramontano A (2011) Critical assessment of methods of protein structure prediction (casp) round ix. *Proteins: Struct Funct Bioinf* 79:1–5
33. Faraggi E, Yaoqi Z, Kloczkowski A (2014) Accurate single-sequence prediction of solvent accessible surface area using local and global features. *Proteins: Struct, Funct, and Bioinf*. DOI: 10.1002/prot.24682
34. CASP10 (2012) Official group performance ranking. http://www.predictioncenter.org/casp10/groups_analysis.cgi. Accessed 10 June 2012
35. Feng Y, Kloczkowski A, Jernigan R (2007) Four-body contact potentials derived from two protein datasets to discriminate native structures from decoys. *Proteins: Struct Funct Bioinf* 68:57–66
36. Feng Y, Kloczkowski A, Jernigan RL (2010) Potentials' r'us web-server for protein energy estimations with coarse-grained knowledge-based potentials. *BMC Bioinf* 11:92
37. Gniewek P, Leelananda SP, Kolinski A, Jernigan RL, Kloczkowski A (2011) Multibody coarse-grained potentials for native structure recognition and quality assessment of protein models. *Proteins: Struct Funct Bioinf* 79:1923–1929
38. Zhou H, Zhou Y (2002) Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure

- selection and stability prediction. *Protein Sci* 11:2714–2726
39. Yang Y, Zhou Y (2008) Ab initio folding of terminal segments with secondary structures reveals the fine difference between two closely related all-atom statistical energy functions. *Protein Sci* 17:1212–1219
40. Zhang J, Zhang Y (2010) A novel side-chain orientation dependent potential derived from random-walk reference state for protein fold selection and structure prediction. *PLoS One* 5:e15386
41. Zhang Y, Skolnick J (2004) Scoring function for automated assessment of protein structure template quality. *Proteins: Struct Funct Bioinf* 57:702–710
42. Xu J, Zhang Y (2010) How significant is a protein structure similarity with tm-score = 0.5? *Bioinformatics* 26:889–895

Modulation of Grasping Force in Prosthetic Hands Using Neural Network-Based Predictive Control

Cristian F. Pasluosta and Alan W.L. Chiu

Abstract

This chapter describes the implementation of a neural network-based predictive control system for driving a prosthetic hand. Nonlinearities associated with the electromechanical aspects of prosthetic devices present great challenges for precise control of this type of device. Model-based controllers may overcome this issue. Moreover, given the complexity of these kinds of electromechanical systems, neural network-based modeling arises as a good fit for modeling the fingers' dynamics. The results of simulations mimicking potential situations encountered during activities of daily living demonstrate the feasibility of this technique.

Key words Nonlinear control, Prosthetic hands, Neural networks, Optimization

1 Introduction

A prosthetic hand must be able to exercise precise control of many degrees of freedom (DoF) to maintain stable grasping for various scenarios in the activities of daily living (ADL). A prosthetic hand must be also designed to avoid psychological and muscle fatigue (human-machine interaction design) and must meet the electro-mechanical specifications such as anthropomorphic dimensions, weight, and power consumption (mechatronic design).

The device control approach can be classified in four directions (Fig. 1). One approach is to control the dynamics of the hand using direct information decoded from bio-signals, with vision as the only feedback modality (Fig. 1a) [1–9]. In an attempt to reduce errors in control, a second approach has been proposed in which artificial sensory information (i.e., tactile and proprioceptive) is sent back to the user (Fig. 1b) [10–15]. A third approach has been designed to reduce psychological effort and hence increase usability and controllability of the device. In this case, the control structure is organized in a hierarchical fashion by dividing it into low and high levels of control (Fig. 1c) [16–29]. In the high level of

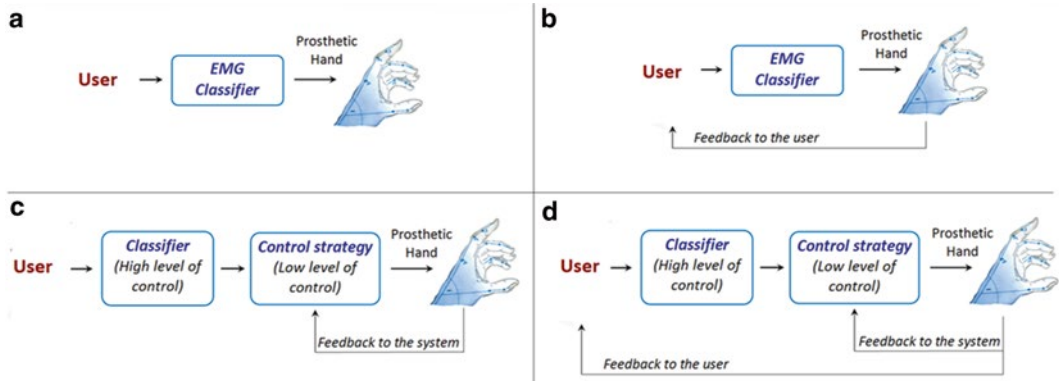


Fig. 1 Control approaches, modified from ref. 34, with permission

control, bio-signals are decoded to interpret the intention of the user (for example, open/close the hand, finger pre-shape), and in the low level of control a stand-alone feedback controller is used to modulate the finger dynamics automatically. This approach feeds artificial sensory information back to the controller but not to the user. Finally, in an effort to combine the advantages of the second and third approaches into a fourth control structure, researchers have adopted the abovementioned hierarchical control strategy with sensory feedback to the user (Fig. 1d) [30–33]. *This chapter focuses on the third approach.* More specifically, we discuss the implementation of a neural networks-based control system for the low level of control.

Several prototypes with a hierarchical control structure have been developed. Cipriani et al. developed a control strategy for the CyberHand [23] that makes use of the information from an extensive array of sensors placed at the prosthetic hand. This control structure is divided into two stages. The first one consists of a pre-shaping of the hand, whereas a position control system arranges the fingers in a specific configuration (such as precision pinch, tripod, etc.). In the second stage, a force control system adjusts the grasping force around the target object. Another hierarchically organized control strategy was presented by Light et al. and was applied to control the Southampton-REMEDI hand [25]. In a multi-stage fashion, the fingers first pre-shape and close around the object while applying minimal force. When indicated by the user, the system modulates the force applied by the fingers to produce stable grasping and to avoid slippage. The object is then released when the hand opens after a squeeze signal coming from the user. Engeberg et al. have used the Motion Control hand to test several control approaches, in which the user controls the grip force from EMG signals directly, but at the same time taking advantage of a force control system [19–21]. Although this is not a hierarchical

control structure per se, the modulation of the grip force using artificial sensory information can be considered within this line of research. In their approach, force-derivative feedback was introduced into the force control loop to improve the sensibility and to reduce the force overshoot at the beginning of the grasping [20]. A hybrid force-velocity-position control was also implemented to address the overshoot issue [21]. Further, using the derivative of the shear force, an adaptive slip prevention system was implemented to improve the force control [19]. Another slip prevention algorithm that also minimizes object deformation was presented by Engeberg and Meek [27]. Andrecioli and Engeberg utilized an adaptive controller based on the detection of object stiffness [28]. Wettels et al. implemented a control system that adjusts the grip force according to a slippage detection system that uses a fluid-based tactile sensor to compute the normal and tangential components of the grip force [17]. Peerdeman et al. utilized the University of Bologna hand IV to test a low level of control based on the intrinsically passive controller technique [29].

Consider now the low-cost mechatronic design of a prosthetic hand shown in Fig. 2 [35]. This is a five-fingered hand with an opposable thumb, with each finger containing two phalanges and their tips slightly bent. The thumb, index and middle fingers move independently (active fingers), while the ring and little fingers (passive fingers) are mechanically coupled to the middle finger. The fingers are flexed by DC gear motors by means of a tendon-cable

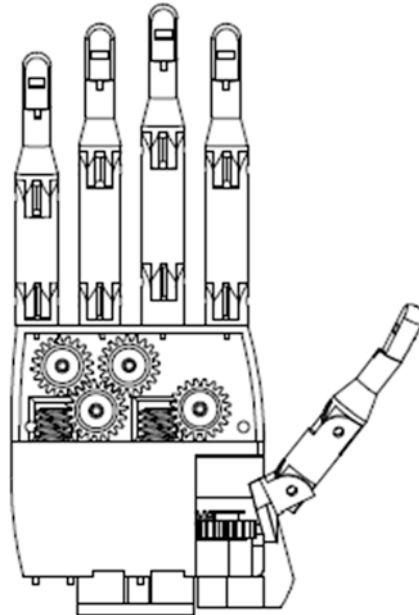


Fig. 2 Mechatronic design of a low-cost prosthetic hand, modified from ref. 34, with permission

system connected through a worm gear system attached to a pulley. An extensor tendon cable attached to a spring extends the fingers when the flexor cable is released. Force sensor resistors (FSR) placed at the tips and resistive flex sensors (RFS) placed at the dorsal part of the fingers are used to obtain force and flexion information, respectively.

Focusing on the low level of control of the hand (from a control system perspective), a prosthetic hand can be represented as a plant, in which inputs control the actuators (i.e., current motor) and the outputs are the kinematics and kinetics of the hand (i.e., contact force, joint angle, and torque). Thus, prostheses are highly nonlinear devices mainly due to the existence of motor dead bands, friction and large gear ratios [20], and nonlinear responses typically observed in low-cost sensors. In order to control the force/position of the fingers the controller must be able to deal with these nonlinearities. In this case, a linear controller (i.e., a PID controller) will have difficulties stabilizing the system for all the possible scenarios.

One way of dealing with nonlinear systems is by modeling their dynamics. A nonlinear model of the relationship between inputs (i.e., motor current) and the outputs (i.e., grip force) can be used to predict future behaviors and adjust the input accordingly. Under these conditions, nonlinear model predictive control (NMPC) presents an attractive solution. This technique employs a nonlinear model to determine future controllable inputs over a defined time horizon [36]. Then, if we can predict future outputs, we can use them to compute the optimal input to maintain the desired force over this horizon [36].

Creating an acceptable model of the system can be very challenging, especially if little is known about its dynamics. A potential solution arises from a technique commonly referred to as “black box modeling”. In black box modeling, several inputs and outputs are presented to an algorithm that maps their relationship in an iterative fashion [36]. In this context, artificial neural networks are commonly used as a black box modeling technique given their ability to map any nonlinear function to a specific degree of accuracy [37]. Neural networks-based NMPC has been demonstrated to be a powerful tool in a wide variety of different scenarios [38–44].

Figure 3 shows a block diagram of an implementation of a neural network-based NMPC for controlling the finger dynamics of a prosthetic hand. Here u , y , \hat{y} , θ , and r represent the motor current/voltage, the current fingertip force, the predicted fingertip force, the finger flex angle, and the force reference, respectively. The force reference can be determined for example by a slippage detection system so that when slippage is detected, the force reference is changed to counteract this phenomenon. In the following sections, we describe the main two blocks of the NMPC: the system model and the optimization. These procedures are based mainly on the general methodology presented in [36, 45].

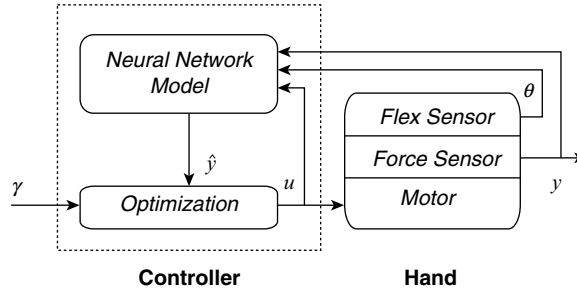


Fig. 3 Neural network-based control scheme (from ref. 35, with permission)

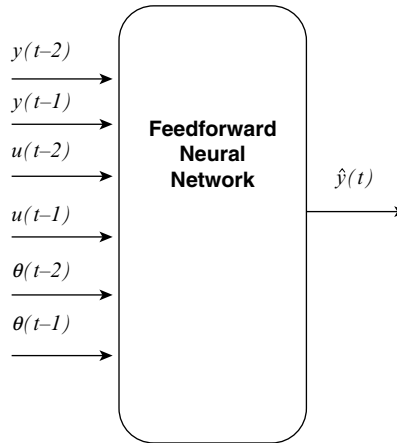


Fig. 4 NNARX structure (from ref. 34, with permission)

2 Neural Network-Based Modeling

Neural networks can be used as predictor machines by using an autoregressive external input structure (NNARX) (Fig. 4; θ is the angle measured by the RFS, which is a non-controllable variable). A feedforward architecture is utilized with its inputs delayed in time. The neural network is trained with time-delayed inputs and current outputs. It maps the relationship between past inputs and current outputs, hence is capable of predicting future outputs from current inputs. Different training approaches, such as the Levenberg–Marquardt method [4], can be applied.

Using a hyperbolic tangent and linear function for the hidden and output layer, respectively, the predictions of the NNARX model of Fig. 4 are obtained as follows:

$$\hat{y}(t+i) = \mathbf{lw}[\tanh(\mathbf{iwp} + b_1)] + b_2 \tag{1}$$

$$\mathbf{p} = [y(t-2), y(t-1), u(t-2), u(t-1), \theta(t-2), \theta(t-1)] \tag{2}$$

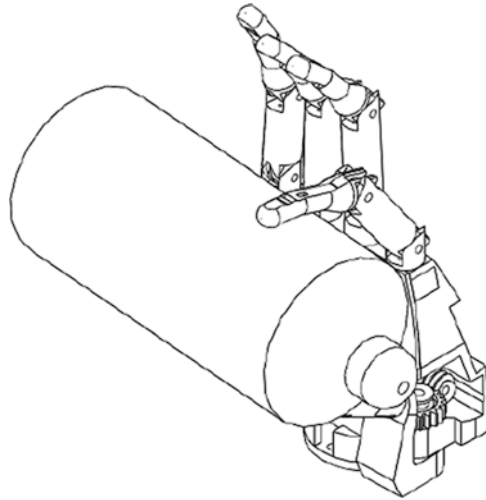


Fig. 5 Experiment set up (from ref. 34, with permission)

where \mathbf{iw} is the input-to-hidden weight matrix, \mathbf{lw} is the hidden-to-output weight matrix, \mathbf{p} is the two-sample-delayed input vector, and b_1 and b_2 are the bias terms

Designing the network structure includes finding the optimum number of inputs (number of lags) and hidden layers. In general, the rule of thumb is to produce a small testing error. However, it is possible that with a high sample frequency (compared to the dynamics of the system), a low test error may not necessarily lead to a good model [36]. Therefore, in order for the dynamics of the system to be modeled correctly, it is also required that no correlation exists between the inputs and the training error and no auto-correlation of the training error itself [36] (Fig. 5).

The following experiments were conducted to obtain training and testing data in order to find the best model. We apply an input signal to the motor to make the finger close around objects with different shapes and compliances. A chirp signal is applied as the input signal to the motor (following [36]):

$$u(t) = u_o + A \sin(\omega_t t) \quad (3)$$

$$\omega_t = \omega_{\text{start}} + \frac{t}{N}(\omega_{\text{final}} - \omega_{\text{start}}) \quad (4)$$

The input signal is swept with different values of u_o and A as well as with different values of ω_{start} and ω_{final} to excite the full range of variables involved in the model. The force and the angle are measured for each active finger. The signal amplitudes of the training and testing data sets are finally scaled to zero mean and unity variance. Using these data, several models are then created off-line and the ones with the best performance (based on the abovementioned measurements) are chosen for the online implementation.

3 Optimization

In the optimization block an objective function (Eq. 5) is minimized with respect to the future control inputs. This objective function takes into account the error between the predictive outputs and the reference values, as well as the changes in the inputs at each iteration.

$$J(t, \mathbf{U}(t)) = \sum_{i=N_1}^{N_2} [r(t+i) - y(t+i)]^2 + \rho \sum_{i=1}^{N_u} [\Delta u(t+i-1)]^2 \quad (5)$$

$$\mathbf{U}(t) = [u(t) \cdots u(t+N_u-1)]^T \quad (6)$$

In Eqs. 5 and 6, the parameters N_2 , N_1 and N_u are the prediction horizon, minimum prediction horizon, and control horizon, respectively. The constant ρ restricts the changes in the control input [36]. The input $u(t)$ is supplied to the motor at each sample point and it is bounded for convergence purposes of the optimization algorithm.

The algorithm presented in refs. 36, 45 can be used to minimize the objective function (in other words, to find the optimum $u(t)$). This algorithm utilizes a gradient descent technique where the future set of control inputs is determined by the update rule indicated in Eq. 7:

$$\mathbf{U}(t+1) = \mathbf{U}(t) - \eta(t) \frac{\partial J}{\partial \mathbf{U}(t)} \quad (7)$$

$$\eta(t) = \eta_0 e^{[\alpha(r(t)-y(t))]} \quad (8)$$

The constant α is determined empirically. Expressing Eq. 5 in matrix notation:

$$J(t, \mathbf{U}(t)) = \mathbf{E}(t)^T \mathbf{E}(t) + \rho \Delta \mathbf{U}(t)^T \Delta \mathbf{U}(t) \quad (9)$$

where:

$$\mathbf{E}(t) = [e(t+1), \dots, e(t+N_2)]^T \quad (10)$$

$$e(t+i) = r(t+i) - \hat{y}(t+i); \quad \text{for } i = 1, \dots, N_2 \quad (11)$$

$$\Delta \mathbf{U}(t) = [\Delta u(t), \dots, \Delta u(t+N_u-1)]^T \quad (12)$$

We have that

$$\frac{\partial J}{\partial \mathbf{U}(t)} = 2\mathbf{E}(t) \frac{\partial \hat{\mathbf{Y}}(t)}{\partial \mathbf{U}(t)} + 2\rho \Delta \mathbf{U}(t) \frac{\partial \Delta \mathbf{U}(t)}{\partial \mathbf{U}(t)} \quad (13)$$

$$\hat{\mathbf{Y}}(t) = [\hat{y}(t+1), \dots, \hat{y}(t+N_2)]^T \quad (14)$$

$$\frac{\partial \Delta \mathbf{U}(t)}{\partial \mathbf{U}(t)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -1 & 1 & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} N_u \times N_u \text{ matrix} \quad (15)$$

$$\frac{\partial \hat{\mathbf{Y}}(t)}{\partial \mathbf{U}(t)} = \begin{bmatrix} \frac{\partial \hat{y}(t+1)}{\partial u(t)} & 0 & \cdots & 0 \\ \frac{\partial \hat{y}(t+2)}{\partial u(t)} & \frac{\partial \hat{y}(t+2)}{\partial u(t+1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}(t+N_2)}{\partial u(t)} & \frac{\partial \hat{y}(t+N_2)}{\partial u(t+1)} & \cdots & \frac{\partial \hat{y}(t+N_2)}{\partial u(t+N_u-1)} \end{bmatrix} \quad (16)$$

A recursive algorithm is presented by Noriega and Wang [45] to calculate Eq. 16:

$$\frac{\partial \hat{y}(t+n)}{\partial u(t+m-1)} = \frac{\partial \hat{y}(t+n-1)}{\partial u(t+m-1)} \left[1 + \frac{\hat{\partial} f(\mathbf{p})}{\partial \hat{y}(t+n-1)} \right], \quad (17)$$

$$\frac{\partial \hat{y}(t+n-1)}{\partial u(t+m-1)} = \mathbf{Iw} \left[\sec h^2(\mathbf{iwp} + b_1) \right] \mathbf{iw} \frac{\mathbf{dp}}{\mathbf{du}} \quad (18)$$

$$\frac{\hat{\partial} f(\mathbf{p})}{\partial \hat{y}(t+n-1)} = \mathbf{Iw} \left[\sec h^2(\mathbf{iwp} + b_1) \right] \mathbf{iw} \frac{\mathbf{dp}}{\mathbf{d\hat{y}}} \quad (19)$$

Here, $n=1, \dots, N_2$, $m=1, \dots, N_2$ and,

$$\frac{\mathbf{dp}}{\mathbf{du}} = [0, 0, 0, \dots, 1, 0, \dots, 0]^T \quad (20)$$

$$\frac{\mathbf{dp}}{\mathbf{d\hat{y}}} = [1, 0, \dots, 0, 0, \dots, 0]^T \quad (21)$$

The motor power consumption and undesirable oscillations are reduced by turning off the motor when the measured force falls within a tolerance range (which we set at $\pm 5\%$ of the reference value).

4 Testing the Neural Network-Based NMPC on a Single Prosthetic Finger

Once an optimum model of the finger dynamics has been obtained, a simple way to evaluate the performance of the neural network-based NMPC system is by testing its step response. In this procedure, the finger is closed around different objects (Table 1), while a

Table 1
Objects used to evaluate the step response of the neural network-based NMPC system

Object	Weight (g)	Diameter (cm)
Styrofoam cup	2	7
Small plastic bottle	15	6.5
Big plastic bottle	40	8.5
Aluminum cylinder	110	7.3

Adapted from ref. 34, with permission

force reference signal is applied to the control system. For the results presented in this chapter, the nonlinear model of the dynamics of the fingers was created using the Levenberg–Marquardt method of the NNSYSID toolbox for Matlab [46].

Figure 6 shows a representative recording of the behavior of the system as the reference signal is incremented in steps of different heights, with an approximately time interval of 20 s ($N=5$). The performance metrics for each reference step are the average closing time, the average overshoot, the average rise time (the time in which the system reaches 85 % of the reference value), and the average steady state error (SSE, absolute mean difference between the finger force and the reference value during steady state). The transient state, which precedes the steady state, is defined as the time in which the peak value of the force oscillations is greater than 15 % of the reference value.

Once the finger closed around the object the rise time for each step was around 0.5 s (Fig. 7) for most of (for the cylinder, at 0.89 N was 1.22 s). Although the response time of the finger is significantly longer with the cylinder, the overall response time of the control system is within the acceptable range (less than 1 s).

Maximum overshoot was found at contact and it was reduced during the subsequent force steps (Fig. 8). Rigid objects (such as the aluminum cylinder) gave rise to a smaller overshoot than softer objects (such as the plastic bottle). In general, the overshoot decreased as the applied force reference increased.

The experiments showed a small SSE in all the step sizes (Fig. 9), with an incremental trend as the force reference increased. This was expected as the motor was turned off when the measured force was within 5 % of the reference force.

Overall, the force control system is capable of adjusting the grasping force adequately. The response of the system is fast enough to meet a rise time of approximately 500 ms, with an overshoot small enough to avoid object damage. Regarding long term force control, the SSE is small enough (~ 0.02 – 0.13 N) to maintain the same force during an extended period of time.

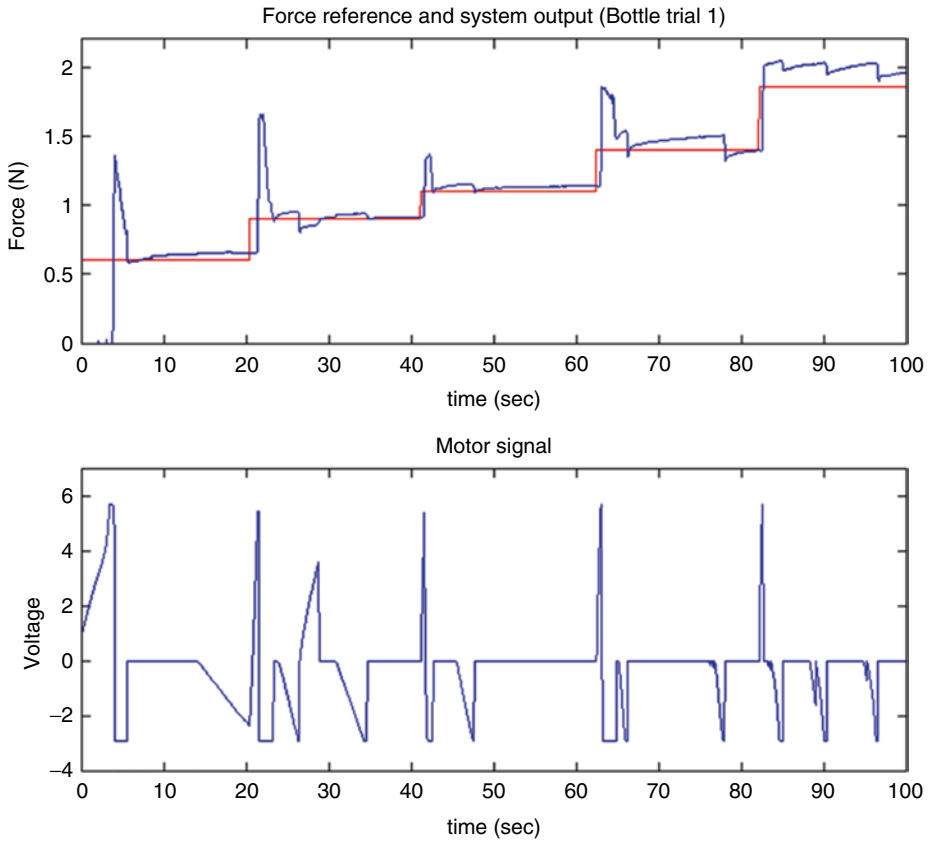


Fig. 6 Representative recording of the system step response (from ref. 34, with permission)

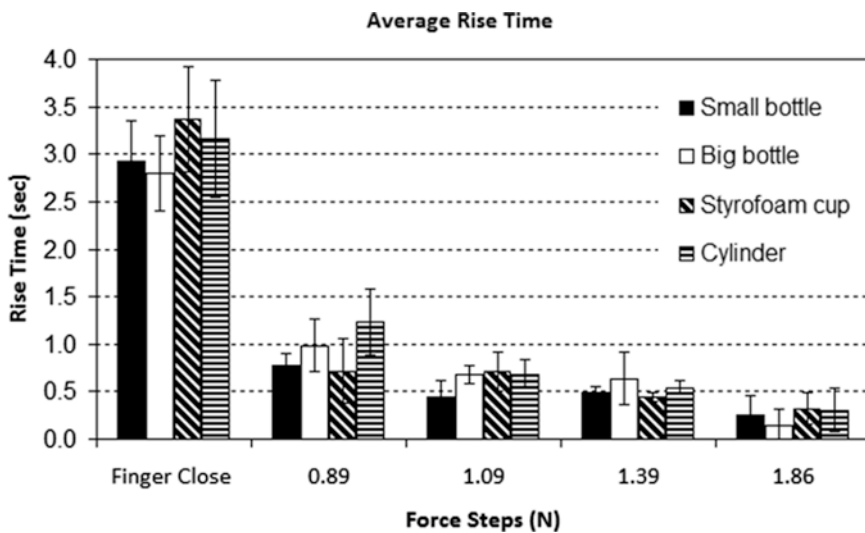


Fig. 7 Average closing and rise time for each object. Modified from ref. 34, with permission

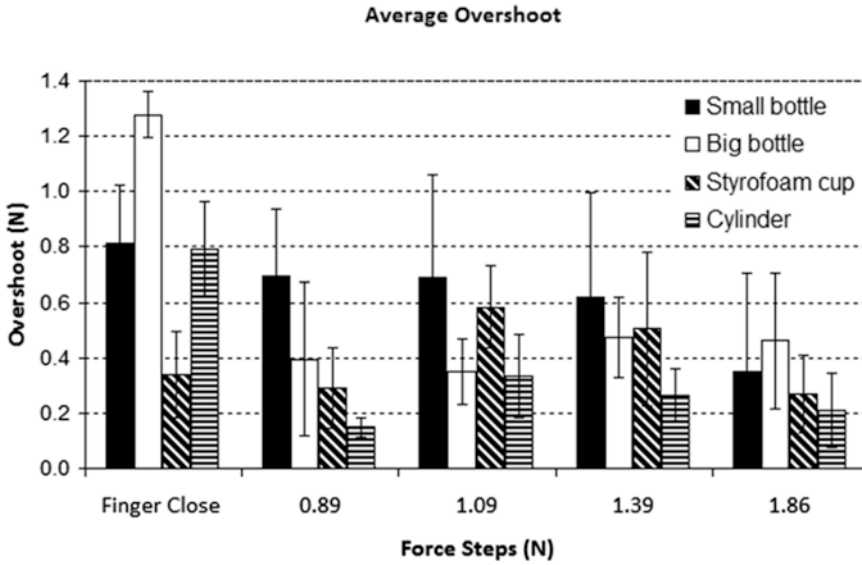


Fig. 8 Average overshoot for each step and all objects. Modified from ref. 34, with permission

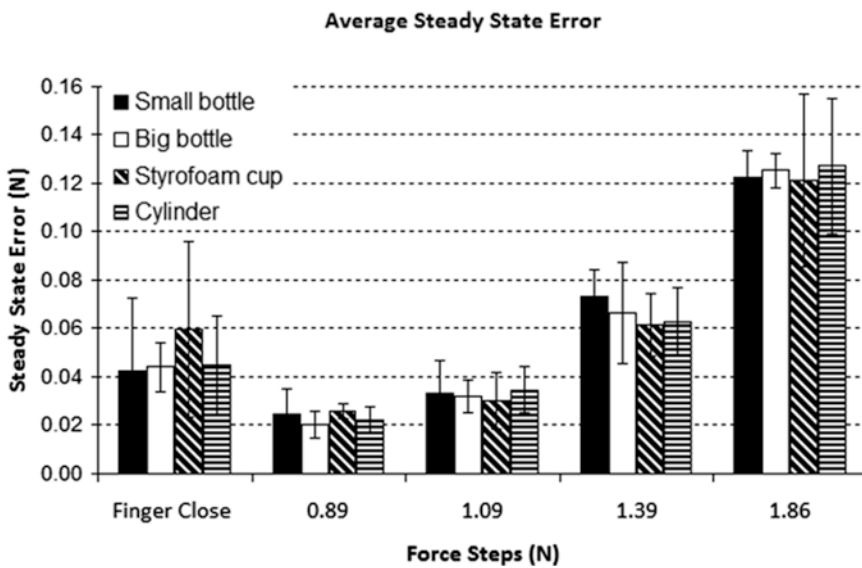


Fig. 9 Average steady state error for each object. Modified from ref. 34, with permission

5 Implementation on a Whole Hand Prototype

So far we have discussed how to control the force applied by an artificial finger using a neural network-based NMPC. We describe now how to incorporate this approach into a hierarchically structured control strategy of an artificial hand.

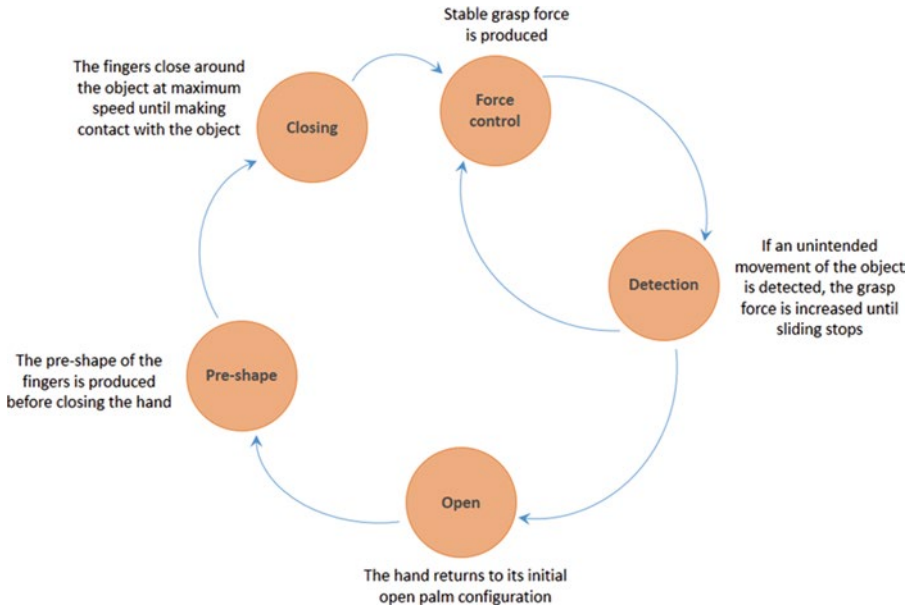


Fig. 10 Block diagram of the control strategy (modified from ref. 34, with permission)

Figure 10 shows implementation of the force control system in the control of the hand prototype described early in this chapter. This control strategy consists of five stages: Pre-shape, Closing, Force Control, Detection, and an Open. The same scheme was applied to drive the thumb, index and middle fingers independently.

The control strategy starts at the Pre-shape stage, where a hand configuration (i.e., cylindrical, tip, lateral, etc.) is selected by the decoding of the user's bio-signals. This stage will determine also which fingers will be involved in the grasping (i.e., in the tip configuration only the thumb and index fingers are used). Immediately after this stage, the system switches to the Closing stage, where the fingers close around the object at maximum speed until making contact with the object. Once contact with the object is established, the Force Control stage comes into play to maintain an initial minimum force over the object. In the Force Control stage the neural network-based NMPC technique described earlier is implemented to maintain this minimum force over the object. The value of this minimal force is empirically determined to prevent slippage and object deformation. This force will be maintained while no unintended movement of the object is detected by the Detection stage. If some undesired movement occurs, the reference force of the NMPC is increased in discrete steps until the movement ceases. The Detection stage thus alternates with the Force Control stage. The Detection stage is defined by the derivative of the normal force measured by the FSR placed at the tips of the fingers (please refer to [35] for more detail).

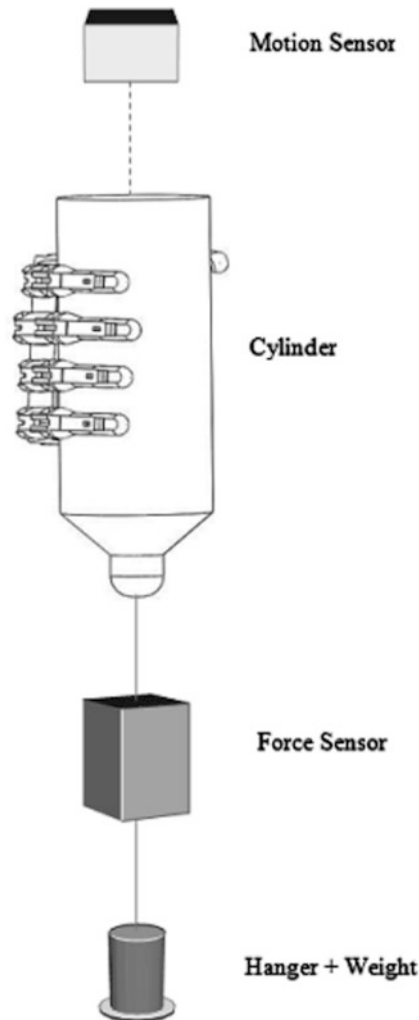


Fig. 11 Whole hand experiments (modified from ref. 35, with permission)

The control strategy is tested in three different conditions that simulate potential scenarios of daily life [35]. These conditions are adopted from established testing methods [47–49]. A representative example for the first condition is shown in Fig. 11. The first scenario involves applying both sudden and gradual changes to the mass of the object while it is grasped with a transverse volar grip. In the second scenario a sudden perturbation is made to the grasped object while maintaining a pinch grip. Finally, the third scenario tests the ability to adjust grasping forces while rotational forces are applied to the grasped object. In all the three experiments, the displacement of the object (i.e., the distance traveled by the object after applying the disturbance) is recorded by a motion sensor [35].

For the transverse volar grip experiments, the maximum average displacement of 7.6 mm and the maximum average displacement for the pulp pinch configuration of 3.05 mm are achieved. For the torque experiment, the maximum average angle displacement of 10.7° at a load of 500 g (11 N cm) is obtained. A more detailed discussion of these results can be found in ref. 35. Taken together, these results suggest that the control strategy adjusted the grasping force in response to the applied disturbance.

The performance assessment suggests that this low-cost prosthetic hand can effectively mimic and perform daily life tasks. The neural-network based control system accounted for nonlinear behaviors in a reliable fashion, leading to reproducible finger force and fast response. Embedded in a hierarchically structured control strategy, this property enabled object grasping with minimal slippage and deformation effects. We anticipate the use of neural-network approaches could optimize the function of other prosthetic devices where nonlinear behaviors compromise their performance.

6 Notes

The degree of object deformation and sliding rely heavily on the relationship between the initial reference force and the force increment step. As a result, the choice of these two parameters is critical in fine-tuning the performance of this control strategy. While this methodology proved successful in the grasping of rigid objects, it is anticipated that mimicking the grasping of softer objects would require an on-the-fly adjustment of the initial reference force applied so as to avoid a very high grasping force of the hand which could lead to subsequent object deformation.

Acknowledgement

This work was supported by the NIH NCRR INBRE grant P20RR016456, and the Louisiana Board of Regents RCS LEQSF(2007-10)-RD-A-20.

References

1. Smith RJ, Tenore F, Huberdeau D et al (2008) Continuous decoding of finger position from surface EMG signals for the control of powered prostheses. 30th Annual international conference of the IEEE Engineering in Medicine and Biology Society, 20–25 Aug 2008, pp 197–200
2. Matrone G, Cipriani C, Secco EL et al (2009) Bio-inspired controller for a dexterous prosthetic hand based on principal components analysis. Annual international conference of the IEEE Engineering in Medicine and Biology Society, 3–6 Sep 2009, pp 5022–5025
3. Jingdong Z, Li J, Shicai S et al (2006) A five-fingered underactuated prosthetic hand system. Proceedings of the IEEE international conference on mechatronics and automation, 25–28 Jun 2006, pp 1453–1458

4. Jingdong Z, Zongwu X, Li J et al (2005) Levenberg-Marquardt based neural network control for a five-fingered prosthetic hand. Proceedings of the IEEE international conference on robotics and automation, 18–22 Apr 2005, pp 4482–4487
5. Zajdlík J (2006) The preliminary design and motion control of a five-fingered prosthetic hand. Proceedings of the international conference on intelligent engineering systems, 0–0 0, pp 202–206
6. Dapeng Y, Jingdong Z, Yikun G et al (2009) EMG pattern recognition and grasping force estimation: improvement to the myocontrol of multi-DOF prosthetic hands. IEEE/RSJ international conference on intelligent robots and systems, 10–15 Oct 2009, pp 516–521
7. Tsujiuchi N, Takayuki K, Yoneda M (2004) Manipulation of a robot by EMG signals using linear multiple regression model. Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, vol 1992, 28 Sep–2 Oct 2004, pp 1991–1996
8. Jingdong Z, Zongwu X, Li J et al (2006) EMG control for a five-fingered underactuated prosthetic hand based on wavelet transform and sample entropy. International conference on intelligent robots and systems, 9–15 Oct 2006, pp 3215–3220
9. Weir RF, Ajiboye AB (2003) A multifunction prosthesis controller based on fuzzy-logic techniques. Proceedings of the 25th annual international conference of the IEEE Engineering in Medicine and Biology Society, vol 1672, 17–21 Sep 2003, pp 1678–1681
10. Pylatiuk C, Mounier S, Kargov A et al (2004) Progress in the development of a multifunctional hand prosthesis. 26th Annual international conference of the IEEE Engineering in Medicine and Biology Society, 1–5 Sep 2004, pp 4260–4263
11. Cipriani C, Antfolk C, Balkenius C et al (2009) A novel concept for a prosthetic hand with a bidirectional interface: a feasibility study. IEEE Trans Biomed Eng 56(11):2739–2743. doi:10.1109/TBME.2009.2031242
12. Panarese A, Edin BB, Vecchi F et al (2009) Humans can integrate force feedback to toes in their sensorimotor control of a robotic hand. IEEE Trans Neural Syst Rehabil Eng 17(6): 560–567. doi:10.1109/TNSRE.2009.2021689
13. Rodriguez-Cheu LE, Casals A (2006) Sensing and control of a prosthetic hand with myoelectric feedback. The first IEEE/RAS-EMBS international conference on biomedical robotics and biomechatronics, 20–22 Feb 2006, pp 607–612
14. Dhillon GS, Horch KW (2005) Direct neural sensory feedback and control of a prosthetic arm. IEEE Trans Neural Syst Rehabil Eng 13(4): 468–472. doi:10.1109/TNSRE.2005.856072
15. Lundborg G, Rosén B (2001) Sensory substitution in prosthetics. Hand Clin 17(3): 481–488
16. Mangieri E, Ahmadi A, Maharatna K et al (2008) A novel analogue circuit for controlling prosthetic hands. IEEE biomedical circuits and systems conference, 20–22 Nov 2008, pp 81–84
17. Wettels N, Parnandi AR, Moon JH et al (2009) Grip control using biomimetic tactile sensing systems. IEEE/ASME Trans Mechatron 14(6): 718–723
18. Tura A, Lamberti C, Davalli A et al (1998) Experimental development of a sensory control system for an upper limb myoelectric prosthesis with cosmetic covering. J Rehabil Res Dev 35(1):14–26
19. Engeberg ED, Meek SG (2008) Adaptive object slip prevention for prosthetic hands through proportional-derivative shear force feedback. IEEE/RSJ International conference on intelligent robots and systems, 22–26 Sep 2008, pp 1940–1945
20. Engeberg ED, Meek S (2008) Improved grasp force sensitivity for prosthetic hands through force-derivative feedback. IEEE Trans Biomed Eng 55(2):817–821
21. Engeberg ED, Meek SG, Minor MA (2008) Hybrid force-velocity sliding mode control of a prosthetic hand. IEEE Trans Biomed Eng 55(5):1572–1581
22. Zhao DW, Jiang L, Huang H et al (2006) Development of a multi-DOF anthropomorphic prosthetic hand. International conference on robotics and biomimetics, 17–20 Dec 2006, pp 878–883
23. Cipriani C, Zaccone F, Stellin G et al (2006) Closed-loop controller for a bio-inspired multi-fingered underactuated prosthesis. Proceedings of the IEEE international conference on robotics and automation, 15–19 May 2006, pp 2111–2116
24. Carrozza MC, Cappiello G, Micera S et al (2006) Design of a cybernetic hand for perception and action. Biol Cybern 95(6):629–644
25. Light CM, Chappell PH, Hudgins B et al (2002) Intelligent multifunction myoelectric control of hand prostheses. J Med Eng Technol 26(4):139–146
26. Connolly C (2008) Prosthetic hands from Touch Bionics. Ind Rob 35(4):290–293
27. Engeberg ED, Meek SG (2013) Adaptive sliding mode control for prosthetic hands to simultaneously prevent slip and minimize deformation of grasped objects. IEEE/ASME

- Trans Mechatron 18(1):376–385. doi:[10.1109/TMECH.2011.2179061](https://doi.org/10.1109/TMECH.2011.2179061)
28. Andrecioli R, Engeberg ED (2013) Adaptive sliding manifold slope via grasped object stiffness detection with a prosthetic hand. *Mechatronics* 23(8):1171–1179
 29. Peerdeman B, Fabrizi U, Palli G et al (2012) Development of prosthesis grasp control systems on a robotic testbed. 4th IEEE RAS & EMBS international conference on biomedical robotics and biomechatronics, 24–27 Jun 2012, pp 1110–1115
 30. Luo Z-z, Wang F, Wang R-c (2006) Study of multi-freedom myoelectric prostheses with tactile sense. 27th Annual international conference of the Engineering in Medicine and Biology Society, 17–18 Jan 2006, pp 3004–3007
 31. Cipriani C, Zaccone F, Micera S et al (2008) On the shared control of an EMG-controlled prosthetic hand: analysis of user-prosthesis interaction. *IEEE Transactions on Robotics* 24(1):170–184. doi:[10.1109/TRO.2007.910708](https://doi.org/10.1109/TRO.2007.910708)
 32. Cipriani C, Controzzi M, Vecchi F et al (2008) Embedded hardware architecture based on microcontrollers for the action and perception of a transradial prosthesis. 2nd IEEE RAS & EMBS International conference on biomedical robotics and biomechatronics, 19–22 Oct 2008, pp 848–853
 33. Rodriguez-Cheu LE, Gonzalez D, Rodriguez M () Result of a perceptual feedback of the grasping forces to prosthetic hand users. 2nd IEEE RAS & EMBS international conference on biomedical robotics and biomechatronics, 19–22 Oct 2008, pp 901–906
 34. Pasluosta CF (2010) Nonlinear control strategy for a cost effective myoelectric prosthetic hand. Dissertation, Louisiana Tech University, Ruston, Louisiana, USA
 35. Pasluosta CF, Chiu AWL (2012) Evaluation of a neural network-based control strategy for a cost-effective externally-powered prosthesis. *Assist Technol* 24(3):196–208
 36. Norgaard M, Ravn O, Poulsen NK, Hansen LK (2000) Neural networks for modelling and control of dynamic systems: a practitioner's handbook. Advanced textbooks in control and signal processing. Springer, Great Britain
 37. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Netw* 2(5):359–366
 38. Dahunsi OA, Pedro JO, Nyandoro OT (2009) Neural network-based model predictive control of a servo-hydraulic vehicle suspension system. *AFRICON '09*, 23–25 Sep 2009, pp 1–6
 39. Haichen Y, Zhijun Z (2006) Predictive control based on neural networks of the chemical process. Chinese Control Conference, 7–11 Aug 2006, pp 1143–1147
 40. Bandyopadhyay B (2005) Neural network based predictive controller (NNPC): an investigation into its application in Textiles. International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce, 28–30 Nov 2005, pp 963–967
 41. Jianbin H, Shaohua T, Vandewalle J (1993) One step ahead predictive control of nonlinear systems by neural networks. Proceedings of international joint conference on neural networks, vol 2763, 25–29 Oct 1993, pp 2761–2764
 42. Soloway D, Haley P (2001) Aircraft reconfiguration using neural generalized predictive control. Proceedings of the 2001 American Control Conference, 2001, vol 2924, pp 2924–2929
 43. Qiang S, Fang L, Findlay RD (2006) Generalized predictive control for a pneumatic system based on an optimized ARMAX model with an artificial neural network. International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce, Nov 28–Dec 1 2006, pp 223–223
 44. Haley P, Soloway D, Gold B (1999) Real-time adaptive control using neural generalized predictive control. Proceedings of the American Control Conference, vol 4276, 1999, pp 4278–4282
 45. Noriega JR, Wang H (1998) A direct adaptive neural-network control for unknown nonlinear systems and its application. *IEEE Trans Neural Netw* 9(1):27–34
 46. Norgaard M, Ravn O, Poulsen NK (2001) NNSYSID and NNCTRL tools for system identification and control with neural networks. *Computing and Control Engineering Journal* 12(1):29–36
 47. Gunji D, Mizoguchi Y, Teshigawara S et al (2008) Grasping force control of multi-fingered robot hand based on slip detection using tactile sensor. IEEE international conference on robotics and automation, 19–23 May 2008, pp 2605–2610
 48. Schuurmans J, Van Der Linde RQ, Plettenburg DH et al (2007) Grasp force optimization in the design of an underactuated robotic hand. IEEE 10th international conference on rehabilitation robotics, 13–15 Jun 2007, pp 776–782
 49. Kamikawa Y, Maeno T (2008) Underactuated five-finger prosthetic hand inspired by grasping force distribution of humans. IEEE/RSJ international conference on intelligent robots and systems, 22–26 Sep 2008, pp 717–722

Application of Artificial Neural Networks in Computer-Aided Diagnosis

Bei Liu

Abstract

Computer-aided diagnosis is a diagnostic procedure in which a radiologist uses the outputs of computer analysis of medical images as a second opinion in the interpretation of medical images, either to help with lesion detection or to help determine if the lesion is benign or malignant. Artificial neural networks (ANNs) are usually employed to formulate the statistical models for computer analysis. Receiver operating characteristic curves are used to evaluate the performance of the ANN alone, as well as the diagnostic performance of radiologists who take into account the ANN output as a second opinion. In this chapter, we use mammograms to illustrate how an ANN model is trained, tested, and evaluated, and how a radiologist should use the ANN output as a second opinion in CAD.

Key words Artificial neural network (ANN), Medical image, Mammogram, CAD, Receiver operating characteristic (ROC)

1 Introduction

The goal of computer-aided diagnosis (CAD) is to improve the sensitivity, specificity, and efficiency of a radiologist's diagnosis by using computer analysis of medical images as a second opinion. Using CAD to detect lesions and using CAD to classify lesions are known as CADE and CADx, respectively.

Recently, Eadie et al. [1] reviewed 48 CAD studies (16 CADE and 32 CADx) from 1992 to 2010. They concluded that, overall, there is no clear benefit to using CADE, but CADx significantly improves diagnosis in mammography and breast ultrasound, while CADx shows an adverse impact on diagnosis based on lung CT and dermatologic imaging. While the improvement of breast cancer diagnosis using CAD shows the great potential of the method, the lack of evidence of benefit in some other applications indicates that CAD is still in its infancy stage and more research and development are needed in this field.

Artificial neural networks (ANNs) have been widely used in CAD for tasks such as pattern recognition and lesion classification [2–6]. An ANN is a computational model loosely related to how the human brain is presumed to operate; it can be viewed as a multivariate mathematical function that consists of interconnected computational nodes (or neurons). The connection between a neuron in one layer and the neurons in the preceding layer is represented by a weighted linear combination of the output of nodes in the preceding layer, modified by a nonlinear activation function (e.g., a sigmoid function). The weights are determined through iterative ANN training, which minimizes the sum of the squared error between the ANN output and the known target values for training cases.

2 Materials

2.1 ANN Architecture Three-layer feedforward and error-backpropagation neural networks can approximate any multidimensional continuous function, according to the universal approximation theorem [7], and this ANN architecture is commonly used in CAD applications. The number of input nodes equals the dimensionality of the input feature space and, in the application described in this chapter, the output layer has only a single node that outputs the likelihood of malignancy of a known lesion. In a CAD application, the number of hidden nodes is limited for better generalizability because the number of training cases is finite [8, 9]; this number is determined empirically based on the dimensionality of the input feature space and the size of the training dataset. Once the number of hidden nodes has been chosen, the mathematical function format that the ANN represents is also delimited, and it is no longer true that the ANN can approximate any arbitrary multidimensional continuous function.

For the ANN used by Rana et al. in the CAD study based on mammograms, eight input features were available, and thus eight input nodes were used, together with six hidden-layer nodes [10]. The architecture of such an ANN is illustrated in Fig. 1.

2.2 Data Collection An ANN is actually a mathematical model with many weight parameters. For the ANN architecture in Fig. 1, there are 54 weight parameters that need to be optimized in the ANN training; the number of training cases should be much larger than the number of parameters to ensure the generalizability of the ANN model.

Therefore, medical images for a large number of patients need to be collected, in order to build an ANN model for CAD application. The patients are partitioned into three groups: training cases, validation cases, and testing cases, to ensure generalizability of the ANN. For a patient with multiple images, such as a breast MRI and

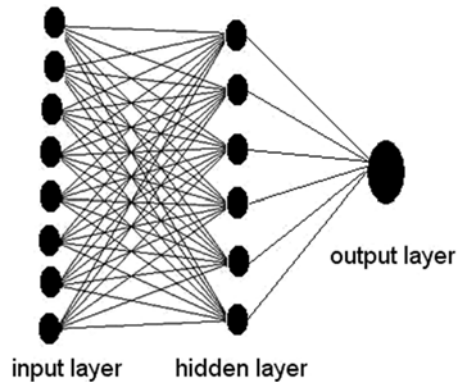


Fig. 1 8-6-1 ANN architecture. There are 54 internode connections, and thus 54 weight parameters

a mammogram, or a CC-view mammogram and an MLO-view mammogram, the data are considered as one case; therefore, all the images of this patient should be in just one of the training group, the validation group, and the testing group. The data cannot be partitioned such that, for example, the CC-view mammogram of a patient is in the training group while the MLO-view mammogram of the same patient is in the validation or the testing group.

The training cases are used to optimize the weight parameters through minimization of the cost function, which is usually the sum of the squared error between the ANN outputs and the training target. The validation cases are used to find the best performing ANNs among the set of ANNs trained with different learning rates/momentum, and among ANNs trained to a different number of epochs, etc. The testing cases are used to evaluate the ANN performance independently.

2.3 Performance Evaluation

The receiver operating characteristic (ROC) is used to evaluate the overall performance of the trained ANN model and the CAD system. An ROC curve completely describes the trade-off between sensitivity and specificity at different operating points [11, 12], where sensitivity or true positive fraction (TPF) is defined as the probability that an actually positive case is diagnosed as positive and specificity is defined as the probability that an actually negative case is diagnosed as negative. More often, the false positive fraction (FPF) is used, which is defined as the probability that an actually negative case is diagnosed as positive. One can easily see that, by definition, $FPF = 1 - \text{specificity}$. The plot of TPF vs. FPF is the ROC curve (Fig. 2). The area under an ROC curve (AUC) is a very useful summary index to describe the CAD system's overall performance: AUC can be viewed as the average sensitivity or average specificity of the CAD system.

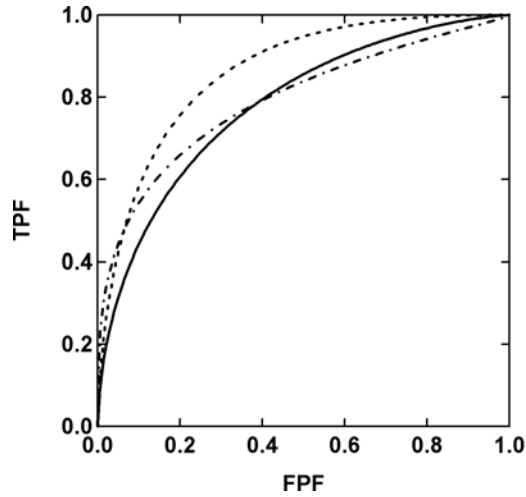


Fig. 2 Three ROC curves. AUC is the average sensitivity or average specificity

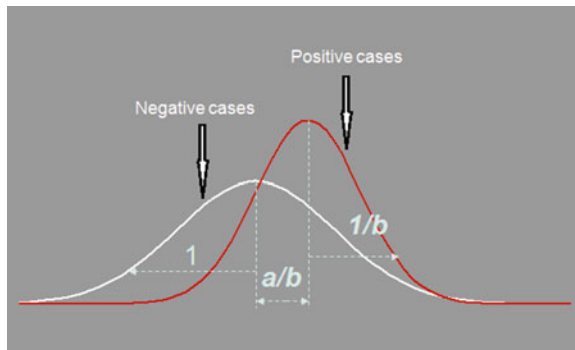


Fig. 3 A binormal distribution of negative cases and positive cases

A binormal model has been successfully used to fit data in a wide variety of practical situations [13, 14]; this assumes that the latent decision variable can be transformed to a pair of normal distributions monotonically: a standard normal distribution for actually negative cases, and a normal distribution with mean of a/b and standard deviation of $1/b$ for actually positive cases (Fig. 3). Under a binormal model, the AUC value is usually called A_z value.

Free ANN software can be downloaded from many websites, which can be found using a web search; professional ANN packages are available from Matlab or SAS; these require payment of a license fee. ROC analysis software can be downloaded from URL <http://metz-roc.uchicago.edu/MetzROC/software> free by setting up a user account.

3 Methods

3.1 *Medical Image Processing*

1. After collecting enough medical images (digital or film based) for patient cases to build an ANN model for CAD applications, all the film-based images need to be digitized and stored in a CAD server along with the digital images.
2. Image pre-processing is often necessary to remove noise and artifacts in the images. Histogram equalization is applied on each image to improve contrast.
3. Regions of interest (ROIs) are defined so that the ROIs, which are simply rectangular regions, enclose the lesions. It is possible that there will be multiple ROIs on one medical image.
4. Lesions are segmented automatically using computer software. Depending on the applications, many segmentation methods can be used: from simple thresholding to region growing, watershed, or an active contour model (snake). For microcalcification lesions of the breast, the shape and size features strongly depend on the segmentation algorithm used [15, 16]; therefore, various segmentation methods should be tried and tested in order to choose a segmentation method that most effectively fits the CAD application.

3.2 *Feature Extraction and Feature Selection*

5. Feature extraction: Based on the lesion segmentation, lesion features such as shape and size can be calculated. Feature extraction is a form of dimension reduction of the input data, which are the raw medical images in a CAD application. Feature extraction is essential in order to reduce computation time, and, more importantly, to avoid overfitting in ANN training. One convenient way to accomplish feature extraction is to quantify the radiologists' perception in image reading. For example, based on radiologists' experience, Jiang et al. [4] used eight features to design a CAD system for the classification of microcalcifications of the breast: cluster circularity, cluster area, number of microcalcifications, average effective volume of microcalcifications, average area of microcalcifications, second highest microcalcification-shape-irregularity measure in a cluster, relative standard deviations in effective thickness, and volume.
6. Nonimage features such as patient age can also be used to build the ANN model [17, 18].
7. Feature selection: Even after feature extraction, which is a form of dimension reduction of the input image data, the dimensionality of the feature space needs to be reduced to avoid overfitting. More importantly, multicollinearity, in which two or more features are highly correlated, must be removed, in order to train the ANN more effectively and to obtain a more

stable ANN model [19]. Another goal of feature selection is to remove irrelevant features, which have no predicting power.

Stepwise feature selection can use either bottom-up or top-down methods. Bottom-up methods start with an empty feature space, and then add one feature at a time to build an ANN or a linear discriminant analysis (LDA) model. If, when a feature is added, model performance is better than a preset criterion, then this feature is retained; otherwise, it is discarded; this procedure is repeated until all the features have been tried. On the other hand, the top-down method starts with all the features; one feature is removed and an ANN or an LDA model is constructed. If the model performance is not too much worse based on a preset criterion, then this feature is discarded; otherwise it is retained. For large training datasets, LDA should be used for feature selection [20] since it is fast when handling large datasets; when the training dataset is small, ANN is used for feature selection.

3.3 ANN Training

8. The image dataset is partitioned into three sub-datasets for ANN modeling: a training dataset, a validation dataset, and a testing dataset. The goal of ANN training, which is an iterative process, is to optimize the weight of each nodal connection to minimize the cost function. The cost function is the sum of the squared error between ANN outputs and training target values, which is defined from the golden truth based on biopsy. In the task of classifying a breast lesion as benign or malignant, the target values for benign cases and malignant cases are set to be 0 and 1, respectively (*see* **Notes 1** and **2**).
9. Backpropagation is used for ANN training. In this method the weight correction is $\Delta w_k(i) = -\eta \partial E / \partial w_k + \alpha \Delta w_k(i-1)$ in the i -th iteration, where w is the weight factor, E is the cost function, and $\partial E / \partial w_k$ is the gradient descent force that drives the cost function to the minimum. The parameter η is the learning rate; this controls how rapidly the ANN training converges, and the parameter α is the momentum, which is used to reduce the weight fluctuation. The optimal values of η and α are problem dependent and are determined by trial and error for a specific problem [21].
10. Since backpropagation is a gradient descent method, there is a danger that the ANN training might get trapped in a local minimum. It is generally impossible to be certain that the global minimum has been located, so various initial weights should be tried for ANN training in order to obtain a good ANN model. The area under the ROC curve (AUC) is calculated from the validation dataset for each epoch, or every ten epochs, and the ANNs with the maximum AUC values are tested using the testing dataset. Since the testing dataset is not involved in the ANN training or ANN selection, it is

completely independent, and the ROC curve calculated from the testing dataset is an independent evaluation of the ANN performance (*see* **Notes 3** and **4**).

3.4 ROC Analysis and Observer Study

11. ROC analysis is used to evaluate the ANN performance and LABROC4 is used for binormal ROC curve fitting [12, 22]. LABROC4 and some other ROC analysis program such as CORROC, which calculates the p value for two ROC curves, are developed by the University of Chicago and can be downloaded free from <http://metz-roc.uchicago.edu/MetzROC/software>. The 95 % confidence interval for A_z , which is actually the AUC after binormal fitting, is also calculated.
12. An observer study is conducted to evaluate the performance of the CAD. A number of radiologists should be recruited for an observer study since the CAD performance is radiologist dependent. Before the observer study, each radiologist is trained so that they are familiar with the CAD system, including how to use the software, and to become familiar and comfortable with the setting and the flow of the study. The radiologists are given some details of how the ANNs are trained, such as the image features and nonimage features that are used for ANN training, the size of the training dataset, the validation dataset and the testing dataset, as well as the A_z value of the ANN model applied on the testing case and its standard error. The observers also read a number of cases with and without computer aid, and compare those data with the known outcomes, so that they can build up adequate confidence in use of the CAD system. After this brief training, observers are asked to view a set of medical images, which can be the ANN testing images or other independent images, but cannot be the training or validation images, with or without computer-estimated likelihood of malignancy.

For example, in the CAD observer study conducted by Rana et al. [10], four radiologists were asked to view a set of mammograms with microcalcification lesions with and without computer aid. Each radiologist drew a square ROI to enclose the microcalcification lesion and the computer algorithm automatically calculated the number of calcifications and segmented them. For 93 out of 332 cases, observers were also asked to place the number of calcifications into one of the four categories to help the computer algorithm set a threshold for calcification identification [23]: (a) 3–5, (b) 6–10, (c) 11–30, or (d) 31 or more calcifications. The A_z value of the ANN model used was 0.79 with a standard error of 0.05. With and without computer-estimated likelihood of malignancy, each observer read both the CC-view and MLO-view mammogram for each patient and assigned a BI-RADS category for each patient: benign findings, probably benign, suspicious abnormality, and highly suggestive

of malignancy. The other categories are not used because all the mammograms have calcification lesions. ROC analysis was then performed and compared with the radiologists' diagnosis with and without computer aid (*see* **Notes 5** and **6**).

4 Notes

- 1 When to stop ANN training: Early stopping is often used to avoid overfitting as a form of regularization. However, the problem with this approach is that the question of just when training should stop is not a trivial one; consequently, early stopping has often been employed in an ad hoc fashion. Our experience is to calculate and monitor the ANN performance vs. epoch using an independent validation dataset and use this to help choose the best epoch to stop ANN training, even though this method leaves a smaller dataset for ANN training.
- 2 It is crucial to collect a big dataset in order to develop a CAD system for clinical use. However, in the research phase, it may be too expensive, or even impossible, for a research group to collect a dataset of suitable size; thus small datasets were often used to test new CAD ideas, such as a new segmentation method, a new image feature, or a new imaging modality. In this situation, a round-robin method (also known as a leave-one-out method) is often used [4]. In this method, all but one of the cases are used for ANN training and the one case left out is used for testing; thus this test case is independent of the training cases. This training and testing procedure is repeated until each case has been left out once; therefore each case has an independent test score and ROC analysis can be applied to these scores to obtain the overall ANN performance.
- 3 Ideally, the ANN output is an estimation of the posterior probability, provided that the training dataset is sufficiently large and the ANN architecture is sufficiently complex [24, 25]. Therefore the ideal ANN output has zero variance. However, in reality the ANN output has finite variance because the training data size is finite. In our application, the goal of ANN training is to achieve a high AUC under the ROC curve and low variability in the ANN outputs. However, if we trained several ANNs with similar overall performance as measured by the AUC, a significant variability in ANN outputs is actually beneficial: it indicates that there is significant room to improve the overall ANN performance, and it can be simply achieved by taking the average, the maximum, or the minimum. So one trick to achieve a higher performance classifier is to train several ANN with different methods and then combine the results. For example, one can train an ANN for each of the

segmentation methods (region-growing, watershed, or active contour model) and then combine the ANN outputs in order to achieve higher overall ANN performance.

- 4 When the ANN training truth provides richer information than merely benign/positive, this extra information can sometimes be used in ANN training if the dataset is small. For example, the golden truth of ANN training for breast cancer CAD is based on the biopsy result: training targets of 0 and 1 are assigned to benign and malignant cases, respectively. However, the biopsy provides more information than benign/malignant. For benign cases, two histology subtypes exist: benign and benign with higher risk; for malignant cases, there are four histology subtypes: noncomedo ductal carcinoma in situ (DCIS), comedo DCIS, DCIS with microinvasion, and invasive carcinoma. This sequence of histology subtypes is one possible representation of an ordered spectrum from benign tissue at one extreme to aggressive cancer at the other [26]. Ordered ANN training target values can be assigned according to the histology subtypes in ANN training for small training datasets [25].
- 5 In the mammogram study [10], eight input features were used and an 8-6-1 ANN architecture was employed. However, if the amount of training data is large, more hidden nodes can be used to increase the ANN complexity, and thus enhance its predictive ability.
- 6 In the breast cancer CAD based on mammograms, each patient will have two ANN scores that correspond to the probabilities of malignancy calculated from CC- and MLO-view mammograms [27–29]. A natural way to use this data is to calculate the average score for the patients. As expected, averaging improves the AUC under the ROC curve compared to single-view images [30]; however, averaging is not always the optimal way of combining the two scores: depending on the standard deviation σ of the distribution of positive cases after fitting to a binormal model and the correlation of the two ANN scores, taking the maximum score or taking the minimum score can outperform the averaging approach in some situations [31, 32].

References

1. Eadie LH, Taylor P, Gibson AP (2012) A systematic review of computer-aided diagnosis in diagnostic cancer imaging. *Eur J Radiol* 81: e70–e76
2. Baker JA, Kornguth PJ, Lo JY et al (1996) Artificial neural network: improving the quality of breast biopsy recommendations. *Radiology* 198:131–135
3. Chan H, Sahiner B, Petrick N et al (1997) Computerized classification of malignant and benign microcalcifications on mammograms: texture analysis using an artificial neural network. *Phys Med Biol* 42:549–567
4. Jiang Y, Nishikawa RM, Schmidt RA et al (1999) Improving breast cancer diagnosis with computer-aided diagnosis. *Acad Radiol* 6: 22–33
5. Huo Z, Giger ML, Vyborny CJ et al (2002) Breast cancer: effectiveness of computer-aided diagnosis—observer study with independent

- database of mammograms. *Radiology* 224: 560–568
6. Suzuki K, Li F, Sone S et al (2005) Computer-aided diagnosis scheme for distinction between benign and malignant nodules in thoracic low-dose CT by use of massive training artificial neural network. *IEEE Trans Med Imaging* 24:1138–1150
 7. Richard HH, Lippman RP (1991) Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Comput* 3:461–483
 8. Swingler K (1996) *Applying neural networks: a practical guide*. Academic, London
 9. Berry MJA, Linoff G (1997) *Data mining techniques*. Wiley, New York
 10. Rana RS, Jiang Y, Schmidt RA et al (2007) Independent evaluation of computer classification of malignant and benign calcifications in full-field digital mammograms. *Acad Radiol* 14:363–370
 11. Swets JA (1978) ROC analysis applied to the evaluation of medical imaging techniques. *Invest Radiol* 14:109–121
 12. Metz CE (1986) ROC methodology in radiologic imaging. *Invest Radiol* 21:720–733
 13. Swets JA (1986) Forms of empirical ROCs in discrimination and diagnostic tasks: implication for theory and measurement of performance. *Psychol Bull* 99:181–198
 14. Hanley JA (1988) The robustness of the binormal assumption used in fitting ROC curves. *Med Decis Making* 8:197–203
 15. Veldkamp WJH, Karssemeijer N (1996) Influence of segmentation on classification of microcalcifications in digital mammography. *IEEE engineering in medicine and biology society, bridging disciplines for biomedicine, proceedings of the 18th annual international conference of the IEEE* 3, pp 1171–1172
 16. Paquerault S, Yarusso LM, Papaioannou J et al (2004) Radial gradient-based segmentation of mammographic microcalcifications: observer evaluation and effect on CAD performance. *Med Phys* 31:2648–2657
 17. Shi J, Sahiner B, Chan HP et al (2008) Characterization of mammographic masses based on level set segmentation with new image features and patient information. *Med Phys* 35(1):280–290
 18. Singh S, Maxwell J, Baker JA et al (2011) Computer-aided classification of breast masses: performance and interobserver variability of expert radiologists versus residents. *Radiology* 258:73–80
 19. Matignon R (2005) *Neural network modeling using SAS enterprise miner*. AuthorHouse, Bloomington
 20. Way TW, Sahiner B, Hadjiiski LM et al (2010) Effect of finite sample size on feature selection and classification: a simulation study. *Med Phys* 37(2):907–920
 21. Rojas R, Feldman J (1996) *Neural networks: a systematic introduction*. Springer, New York
 22. Metz CE, Herman BA, Shen JH (1998) Maximum likelihood estimation of receiver operating characteristic (ROC) curves from continuously distributed data. *Stat Med* 17: 1033–1053
 23. Salfity MF, Nishikawa RM, Jiang Y et al (2003) The use of priori information in the detection of mammographic microcalcifications to improve their classification. *Med Phys* 30:823–831
 24. Rojas R (1996) A short proof of the posterior probability classifier neural networks. *Neural Comput* 8:41–43
 25. Liu B, Jiang Y (2013) A multitarget training method for artificial neural network with application to computer-aided diagnosis. *Med Phys* 40:011908. doi:[10.1118/1.4772021](https://doi.org/10.1118/1.4772021)
 26. Rosen PP (2001) *Rosen's breast pathology*. Lippincott Williams & Wilkins, Philadelphia
 27. Jiang Y, Nishikawa RM, Wolverton DE et al (1996) Malignant and benign clustered microcalcifications: automated feature analysis and classification. *Radiology* 198:671–678
 28. Chan H, Sahiner B, Lam KL et al (1998) Computerized analysis of mammographic microcalcifications in morphological and texture feature spaces. *Med Phys* 25:2007–2019
 29. Huo Z, Giger ML, Vyborny CJ (2001) Computerized analysis of multiple-mammographic views: potential usefulness of special view mammograms in computer-aided diagnosis. *IEEE Trans Med Imaging* 20: 1285–1292
 30. Metz CE, Shen J (1992) Gains in accuracy from replicated reading of diagnostic images: prediction and assessment in terms of ROC analysis. *Med Decis Making* 12:60–75
 31. Liu B, Metz CE, Jiang Y (2004) An ROC comparison of four methods of combining information from multiple images of the same patient. *Med Phys* 31:2552–2563
 32. Liu B, Metz CE, Jiang Y (2005) Effect of correlation on combining diagnostic information from two images of the same patient. *Med Phys* 32:3329–3338

Developing a Multimodal Biometric Authentication System Using Soft Computing Methods

Mario Malcangi

Abstract

Robust personal authentication is becoming ever more important in computer-based applications. Among a variety of methods, biometric offers several advantages, mainly in embedded system applications. Hard and soft multi-biometric, combined with hard and soft computing methods, can be applied to improve the personal authentication process and to generalize the applicability.

This chapter describes the embedded implementation of a multi-biometric (voiceprint and fingerprint) multimodal identification system based on hard computing methods (DSP) for feature extraction and matching, an artificial neural network (ANN) for soft feature pattern matching, and a fuzzy logic engine (FLE) for data fusion and decision.

Key words Artificial neural network, Fuzzy decision logic, Soft-biometric data, Multi-biometric

1 Introduction

More and more of the devices we deal with every day depend on embedded systems. This trend can reasonably be expected to accelerate going forward. Because of this, as well as due to other factors, the security of systems and of data is likely to continue to present challenges that developers will attempt to meet by devising new tools. There can be little doubt that many such methods will rely on the biometric approach [1–4].

Biometrics is uniquely suited to authentication tasks in embedded systems because it offers simultaneous solutions to the twin problems of the meager demand for resources that we wish to place on such systems and of the high security requirements that their potential applications can be expected to mandate [5, 6]. It is not hard to imagine wanting to limit demand on resources if we start from the very basic concept of a system that has no keyboard, is designed to have the smallest possible footprint, and can be considered subsidiary to a larger system, machine, or process. If we consider the embedded system's role as gatekeeper to a much

larger and more complex reality, say the electronic ignition key to a sophisticated, automated installation, it is not hard, either, to envision the degree of security requirements for which a minuscule device is ultimately responsible. As such devices proliferate, the demand for lightweight yet robust authentication methods can only grow.

Because biometrics depends on physiological traits or on distinctive behavior [7, 8] unique to the individual whose identity is to be authenticated, or—indeed—a combination of the two, it can rightly stake a claim to being superior, at least potentially, to current and established authentication methods like personal identification numbers (PINs), passwords, or smart cards. The individual's biometric data is always available, is nonrandomly unique, cannot be transferred to another party, cannot be forgotten, is not subject to theft, and cannot be guessed. These advantages mean that biometrics provides very high security compared to traditional identification methods that rely on the possession of a token, such as a card, key, or chip, or of personal knowledge, as in the case of a password [9].

Despite the superiority of biometric authentication due to such advantages and despite its rapid spread in microelectronics, mass-market adoption has lagged. The primary reason widespread application of biometrics has not taken off is that it does not offer surefire authentication the way a password does. A password can always perform. On the other hand, biometric features, in their original form, are analog information. As a result, they are subject to variation when captured by a biometric scanning device. This applies to fingerprint readers, microphones for voice-pattern recognition, cameras for facial feature matching, and any other digital system that has to match measured, analog, human traits. Of course such features can be digitized, but the data will nonetheless have been processed through fuzzy logic. Fuzziness can only be minimized so much, given that the original, analog features are, themselves, inherently fuzzy.

False acceptance rate (FAR) and false rejection rate (FRR) [10] can, of course, be minimized with classical pattern-matching techniques [11–13] but a 100 % correct acceptance rate cannot be achieved this way. However, the intrinsic fuzziness of biometric features makes it logical to look to soft-computing approaches [14–16] in the design of processes that match biometric feature patterns in the hope that nearly 100 % correct authentication might be attained that way. For example, even in prohibitive conditions, human beings always manage to recognize a familiar face. The human mind processes biometric features fuzzily and neurally.

Soft-computing data are variations and uncertainties. Biometric features vary constantly, present challenges to analytical description, and inherently fall short of belonging to their owner 100 % of the time.

It follows that soft-computing methods ought to work well for measuring and matching biometrics. Moreover, what humans do to reach nearly 100 % authentication rates amounts to acquiring data from multiple sources. They combine speech traits and facial features when matching a biometric identity to an individual [17]. In other words, they carry out neural and fuzzy processing on aggregated biometric feature sets.

While biometric authentication has seen surprisingly slow adoption generally, this has proved even more the case with the application of traditional biometric solutions to embedded systems [18]. Because embedded systems have limited resources—smaller memory and slower processors—they are ill suited to hungry authentication applications. Consequently, current installations of embedded systems typically rely on PIN codes and the like.

However, an authentication method based on soft biometric [19] data has the potential to be more miserly with computing resources than hard biometrics, using less data to map the identity of the person to be authenticated. Furthermore, multiple-criteria biometric authentication processes [8, 20, 21] can be devised so as to match the input capacities of embedded systems. By combining soft-computing methods and multi-biometrics, we can optimize such authentication for implementation on embedded systems.

One approach that holds promise in this regard involves setting up authentication based on a combination of voiceprint and fingerprint that uses hard-computing digital signal processing to extract features and match them but then turns to an artificial neural network (ANN) [22] for soft feature pattern matching and to a fuzzy logic inferential engine (FLE) for data fusion and decision making. Such a setup can be designed to provide highly robust, personal, biometric authentication. Experiments have been carried out where this can be accomplished with a single-chip, floating-point, digital signal processor.

2 Materials

To design a multimodal soft computing-based hard/soft biometric embedded system several methodologies and technologies occur:

- Biometric sensors.
- Sensor data acquisition.
- Feature extraction algorithms.
- Hard computing pattern matching.
- Soft computing pattern matching (artificial neural network).
- Soft computing fusion and decision (fuzzy logic).
- System modeling and simulation environments.

2.1 Biometric Sensors

The biometric sensors are specifically designed to capture physiologic or behavioral signals generated by human beings. The sensor is the first device of the signal chain. It captures and converts a physical property into analog signals (commonly electrical) or digital signals (when it embeds the mixed-signal electronics).

2.1.1 Voiceprint Sensors

The voiceprint sensor is a voice-grade microphone system able to capture the utterance, preserving the intelligibility of the speech.

- Single microphone: captures the acoustic wave of the utterance as a direct sound source (position sensitive; sensitive to surrounding noise).
- Dual microphone: captures the acoustic sound wave of the utterance and the surrounding audio noise (partially position sensitive; low sensitivity to surrounding noise).
- Microphone array: captures the acoustic sound wave of the utterance by a set of microphones spatially distributed (position insensitive; highly insensitive to surrounding noise).

A single microphone has been used in this design (*see Note 1*).

2.1.2 Fingerprint Sensors

A fingerprint sensor is an electronic device capable to capture the image of the fingerprint pattern and make it available as a bit map. Several technologies have been applied to develop fingerprint sensors:

- Optical (special digital camera): captures visible light emitted from a phosphor layer which illuminates the surface of the finger—advantages: noncontact, not electrostatic discharge sensitive—disadvantages: capabilities affected by the quality of skin, easily fooled.
- Ultrasonic (based on medical ultrasonic imaging principles): captures the images of a fingerprint by penetrating the epidermal layer of skin with high-frequency sound—advantages: noncontact, not electrostatic discharge sensitive, capabilities not affected by the quality of skin, very difficult to fool—disadvantages: the price is significantly higher than for a capacitive fingerprint scanner.
- Capacitance (based on electrical capacitance): consists of an array of capacitors in which one of the two plates is the dermal layer and the dielectric is the epidermal
 - Passive: Capacitance is measured across each sensor capacitor to form a pixel of the fingerprint image.
 - Active: A voltage is applied to the skin first, and then the charge of each capacitor of the array is measured to form a pixel of the fingerprint image.

Advantages: compatible with microelectronic integration—disadvantages: electrostatic discharge sensitive.

A passive capacitive sensor has been used in this design (*see Note 2*).

2.2 Sensor Data Acquisition Chain

Sensor data acquisition concerns the conditioning and digital representation of the voiceprint and fingerprint information captured by the respective biometric sensors. Conditioning is necessary if the sensor is affected by nonlinearities and if it is not dynamically compatible with analog-to-digital conversion (ADC) subsystem.

2.2.1 Microphone Sensor Data Acquisition

The microphone sensor data acquisition process is sensitive to several nonlinearities and mixed signal constraints. The microphone chain path needs conditioning and digital-to-analog adaptation, so optimal data can be available at the processing stage:

- Transducer linearization.
- Automatic gain control.
- Filtering.
- Sampling.
- Quantization.

The microphone nonlinearities need to be compensated to avoid distorted measurements at the level of feature extraction. This can be done in the analog domain by electronic circuitry tuned on the specific microphone, or in the digital domain at the preprocessing stage of the signal acquisition. In this design the second option has been applied because it is more flexible and adaptive. The microphone transfer function is estimated at calibration time, and then the nonlinearity compensated for by applying at run time the inverse function to the captured signal.

The amplification is required because the small signals from the microphone need to be adapted to the analog-to-digital converter (ADC) input dynamic range to have higher signal-to-quantization-noise ratio (SQNR).

Low-pass filtering (antialiasing) and high-pass filtering (offset removal) are applied prior to sampling the microphone signal to ensure lowest frequency distortion of the pulse code modulated (PCM) stream to be quantized and optimal SQNR.

2.2.2 Fingerprint Sensor Data Acquisition

The solid-state capacitive fingerprint sensor integrates all the required conditioning and digitalization resources so that optimal digital fingerprint image data is available at its output. Fingerprint image is captured at 513 dpi (dot per inch) resolution as a bitmap image consisting of 64,512 pixels (224 horizontal and 288 vertical). Each pixel has 8-bit-depth quantization level (gray-level images). The fingerprint digital image array acquisition process is completed by synchronous serial transfer to the application processor (DSP).

2.3 Feature Extraction Algorithms

To extract the features, a set of digital signal processing algorithms is applied to the captured utterance.

2.3.1 Voiceprint Hard Features

The following formulae have been applied to measure the voiceprint hard features [23]:

- Root mean square (RMS):

$$\text{RMS}_j = \sqrt{\frac{1}{N} \sum_{m=0}^{N-1} s_j^2(m)} \quad (1)$$

- Zero-crossing rate (ZCR):

$$\text{ZCR}_j = \sum_{m=0}^{N-1} 0.5 \left| \text{sign}(s_j(m)) - \text{sign}(s_j(m-1)) \right| \quad (2)$$

- Autocorrelation (AC):

$$\text{AC}_j = \sum_{i=1}^N \sum_{j=1}^{N+1-i} s_j(j) s_j(i+j-1) \quad (3)$$

- Cepstral linear prediction coefficients (CLPC):

$$\text{CLPC}_j = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m} \right) c_k a_{m-k} \quad (4)$$

with

$c_0 = r(0)$: first autocorrelation coefficient

a_m : prediction coefficients

$s_j(n) = w_N(n) s$: j -th windowed part of the uttered stream s

$w_N(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right)$: N -length weighting Hamming window

ing window

The above are short-time measurements executed by multiplying the N samples-wide weighting Hamming window $w(n)$ by the uttered speech stream $s(n)$.

RMS is the root mean square of the windowed speech segment. Such measurement helps to identify phonetic unit end-points.

ZCR is the time-domain measurement of the dominant frequency information. It is used to determine whether or not the current processed uttered speech segment is voiced or unvoiced and also to find the frequency (band) with the major energy concentration.

AC is the measurement of the speech-pitch frequency. It preserves information about pitch-frequency amplitude while ignoring phase. Phase is unimportant for the purpose of speech identification.

CLPC is the LPC-Cepstral feature vector that models the vocal tract.

2.3.2 Voiceprint Soft Features

The following soft features are extracted from speech:

- Speed.
- Stress.

Speed is measured as the total duration of the speech utterance.

Stress is measured as the ratio between the peak amplitude of the stressed vowel and the average amplitude of the whole utterance.

Both these voiceprint features are related to the way the person is used to speaking a requested word.

2.3.3 Fingerprint Hard Features

Several preprocessing steps are executed on the captured grayscale fingerprint image from the stage of bitmap to its minutiae-based representation:

- Orientation field and region of interest.
 - Normalization of the captured fingerprint image.
 - Image segmentation in blocks.
 - x and y gradient computation for each pixel in each block.
 - Local orientation for each pixel.
 - Orientation field correction.
- Positioning (delta and core localization).
- Ridging.
- Ridge thinning.

Fingerprint hard features are the minutiae. The criteria used for minutiae extraction from the thinned ridge image are the following:

- If a ridge pixel has two or more 8-nearest, then it is a bifurcation.
- If a ridge pixel has only one 8-nearest, then it is a termination.

As the crest-valley image is two levels encoded (1-0), the mapping algorithm is

$$\begin{aligned} &\text{if } \sum_{i=0}^7 p_i > 2 \\ &\quad \text{the pixel is B} \\ &\text{else} \\ &\quad \text{the pixel is T} \end{aligned} \tag{5}$$

where p_i are the 8-nearest pixels of the pixel to be classified as bifurcation B or termination T.

The most critical step in this procedure is to avoid computing false minutiae caused by noise in the scanned fingerprint image.

To overcome this problem, a backtracking control is executed on each feature pattern before it is validated, to check that each of the three branches of the bifurcation is significantly long:

- Starting from the bifurcation all the three paths are checked, stopping if more than k pixels or a termination pixel is detected.
- If the stop is due to the termination detection, then the bifurcation and the terminations are invalidated.

The scanned fingerprint image is transformed into a set minutiae encoded by its x, y coordinates and the direction of the ridge corresponding at that position.

2.3.4 Fingerprint Soft Features

Two fingerprint soft features are extracted from fingerprint captured image:

- Total area.
- Mean intensity.

Both these two fingerprint features are related to the way the person approaches contact with the fingerprint sensor.

The total area is measured as the ratio between the total pixels available on the fingerprint scanning device and the total pixels of the captured fingerprint image that have a value higher than an estimated peak noise level. The peak noise level is estimated at calibration time and it is applied at run time (enrolling and identification).

The mean intensity is measured as average intensity of all the pixels with a value higher than a threshold level. The threshold level is 10 % higher than the peak noise level.

2.4 Hard Computing Pattern Matching

The captured fingerprint and voiceprint hard features have to be matched with enrolled features (templates).

2.4.1 Voiceprint Hard Features

Two methods are applied to score the person’s identity. One is based on measuring Mahalanobis distance, and the other on measuring the distance of dynamic time warping— k -nearest neighbor (DTW-KNN).

The Mahalanobis distance measurement is

$$D_i(x) = (x - \bar{x})^T W^{-1} (x - \bar{x}) \tag{6}$$

where W is the covariance array computed using the average and the standard deviation features of the utterance. The input pattern x is processed with reference to the utterance-averaged feature vector \bar{x} that represents the person to be identified. The distance $D_i(x)$ is a score for the authorized user.

The DTW-KNN (dynamic time warping— k -nearest neighbor) method combines the dynamic-time-warping measurement with the k -nearest neighbor decision algorithm. The DTW clusters similar

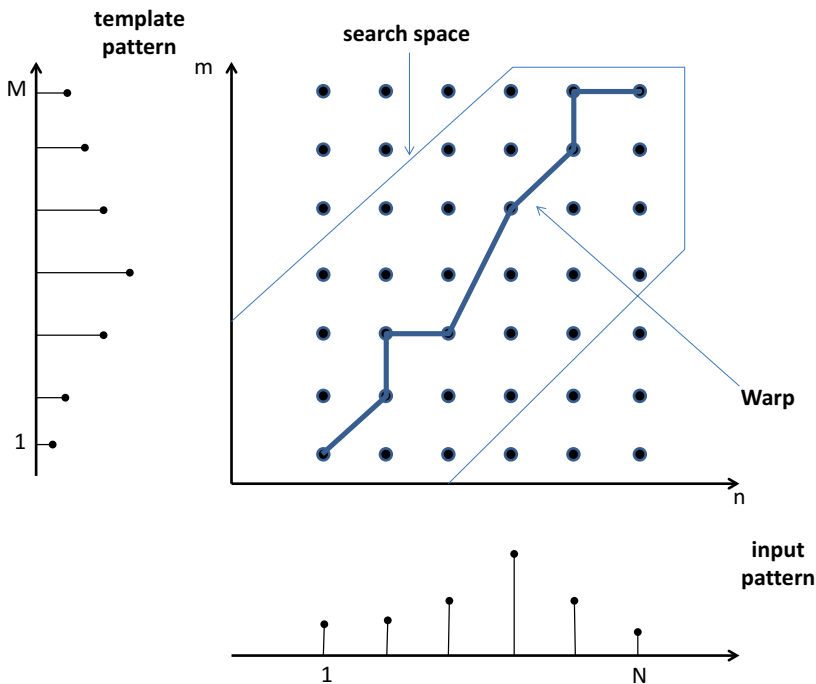


Fig. 1 DTW boundaries for voiceprint matching

elements that refer to a feature into classes (Fig. 1). The cost function is computed using Euclidean distance, with a granularity of one frame. The KNN algorithm is then applied to select k minimal distance matching and to choose the most recurring person in k minimal distance matches. This results in lower false-positive and false-negative rates during identification, compared to the original DTW algorithm.

2.4.2 Fingerprint Hard Features

Fingerprint hard feature pattern matching is based on minutiae. Pattern matching consists of a procedure that first tries to align the template pattern and the input pattern, and then computes an overlapping score (Fig. 2). The score is a measurement of the authenticity of the person who enrolled the input.

2.5 Soft Computing Pattern Matching

The captured fingerprint and voiceprint soft features have to be matched with enrolled features (templates). This is done using the artificial neural network (ANN) soft computing paradigm.

2.5.1 Artificial Neural Network (ANN)

The ANN is a signal processing and pattern classification paradigm inspired by the structure and functions of biological neural networks. Information signals that flow through the ANN modify the connections, enabling the learning process.

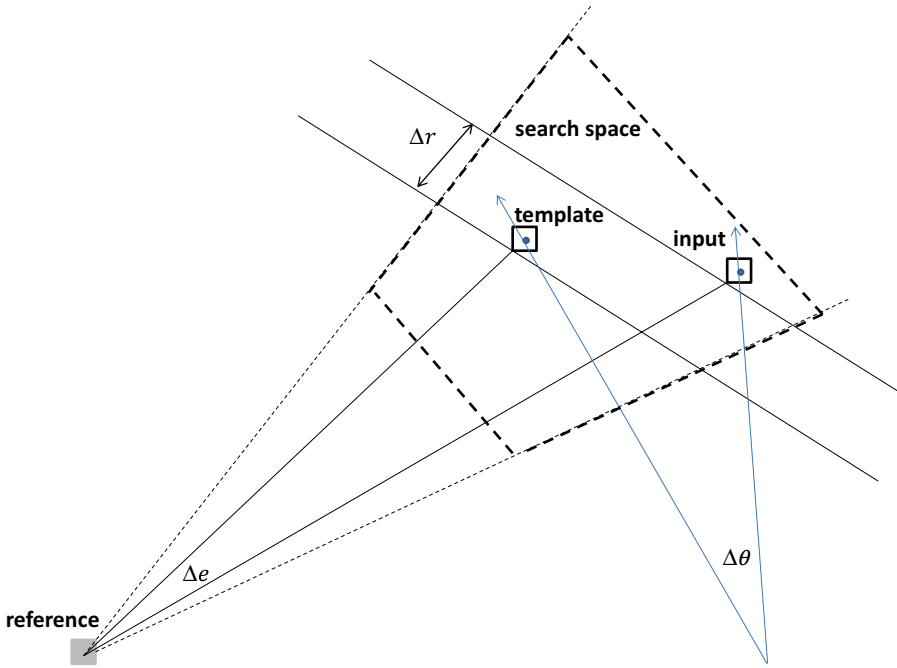


Fig. 2 Boundaries for fingerprint matching

The ANN applied to classify the soft biometric features is a feed-forward back-propagation neural network (FFBP-ANN) paradigm because of the behavioral nature of this kind of biometric data (Fig. 3a). It is a three-layered parallel processing network that processes the input data patterns at the lower layer, matches the data at the middle layer, and organizes the results at the upper layer. Its input nodes are fully connected to all the nodes in the hidden layer, and the hidden layer is fully connected to the output nodes. In this network the feed-forward action consists in that the i node at $l+1$ th layer receives signals from the j node in the l th layer conditioned via weight w_{ij}^l . The activation of the node i at the $l+1$ th layer is given by

$$f\left(\sum_{j=1}^M w_{ij}^l x_j^l(k) - \vartheta_i\right) \tag{7}$$

This is the activity of the i th node at the layer $l+1$ for the k th input x_j processed by the ANN, assuming that M nodes are in the l th layer.

Input and output layers have a linear activation function that controls the connection. A nonlinear (sigmoid) activation function

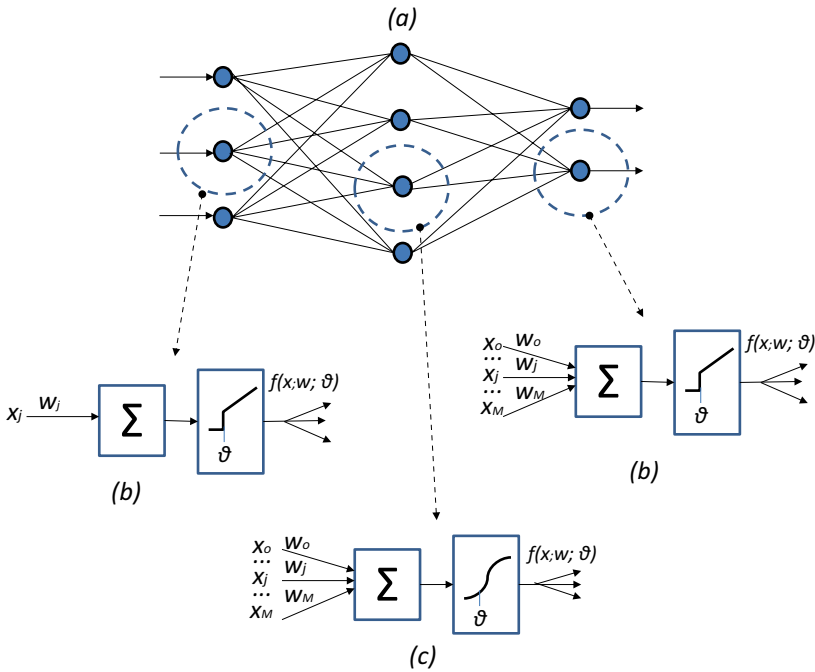


Fig. 3 (a) FFBP-ANN for voiceprint and fingerprint soft feature classification, (b) linear activation function for input and output layers, and (c) sigmoid activation function for hidden layer

(Fig. 3b) connects hidden-layer nodes to output-layer nodes according to the following formulae:

$$s_i = \frac{1}{1 + e^{-\beta I_i}} \tag{8}$$

$$I_i = \sum_j w_{ij} x_j - \vartheta_i$$

where ϑ is the activation threshold of the i th node and β is a constant that controls the slope of the semi-linear region of the sigmoid. When β is very small, the sigmoid approximates the linear activation function (Fig. 3c), so the same function can be applied to input, hidden, and output layers by choosing appropriate values for β constant.

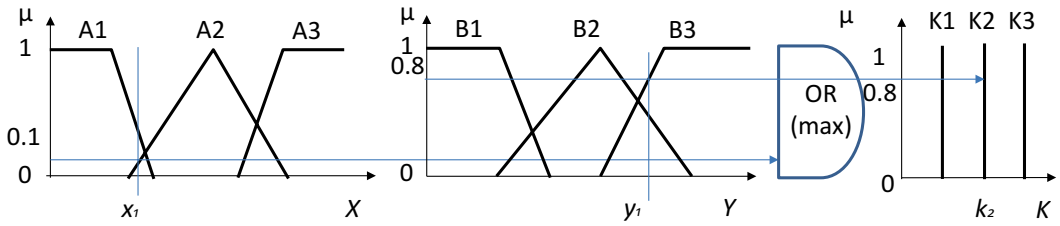
2.6 Soft Computing Fusion and Decision

The fusion and decision logic is based on the fuzzy logic inferential paradigm. A fuzzy logic engine processes the crisp inputs (scores and features), applies to them a set of inferential rules, and makes the fuzzy decision. This is done by the fuzzy logic engine.

2.6.1 Fuzzy Logic Engine

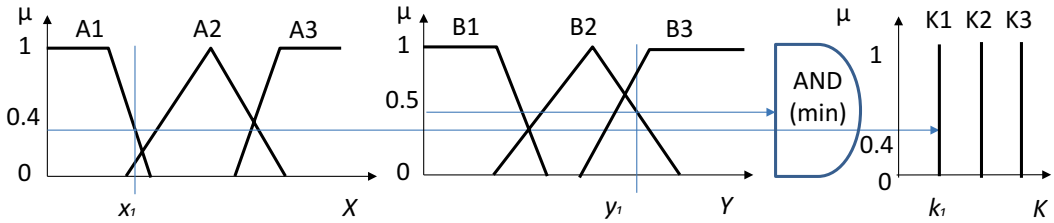
The fuzzy logic engine (FLE) used to implement the data and decision fusion and to infer about the final decision is a zero-order Sugeno-type (Fig 4a). It fuzzifies the inputs (soft features and scores)

a



Rule 1: IF x IS A_2 OR y IS B_3 THEN k is k_2

$K_2 = 30$



Rule 2: IF x IS A_1 AND y IS B_2 THEN k is k_1

$K_1 = 10$

b

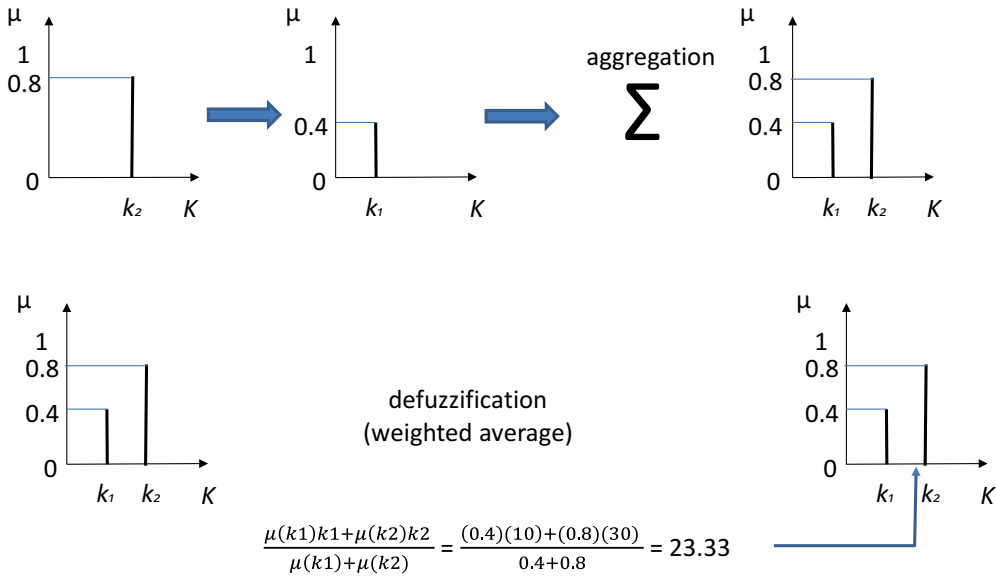


Fig. 4 (a) Zero-order Sugeno-type fuzzy logic inference engine. **(b)** Aggregation and defuzzification

using trapezoidal and triangular membership functions, and applies a set of inferential rules of the form:

IF x IS A AND y IS B THEN k is K

IF x IS A OR y IS B THEN k is K

IF x IS A THEN k IS K

The antecedent (input) part of the rule combines the fuzzyfied inputs by AND (minimum) fuzzy operator or by OR (maximum) fuzzy operator, or directly. The output of each rule is constant (zero order), and then the consequents are represented by singleton membership functions.

To defuzzify the output, so a crisp value is available, the weighted average (WA) method is applied (Fig 4b) (*see Note 3*):

$$WA = \frac{\sum \mu(x)x}{\sum \mu(x)} \quad (9)$$

where $\mu(x)$ is the degree to which the inputs x belong to the appropriate fuzzy sets.

2.7 System Modeling and Simulation Environments

Matlab environment and the DSP toolbox are used to code the signal processing algorithms, to run them in simulation mode, and then to export the source code as ANSI-C. The Data Acquisition (DAQ) toolbox is used to collect data from the fingerprint sensor and from the microphone.

2.7.1 ANN

To model and simulate the FFBP-ANN the Matlab environment plus the DSP toolbox is used to code the FFBP-ANN, to run it in simulation mode, and then to export the source code as ANSI-C. The Data Acquisition (DAQ) toolbox is used to collect data from the fingerprint sensor and from the microphone.

2.7.2 FLE

The FLE is a Sugeno-type fuzzy logic inferential paradigm. To model and simulate it the Matlab environment plus the Fuzzy Logic toolbox is used to code the FLE, to run it in simulation mode, and then to export the source code as ANSI-C. The Data Acquisition (DAQ) toolbox is used to collect data from the fingerprint sensor and from the microphone.

2.8 Digital Signal Processor

To implement the embedded system, a system-on-chip (SoC) processor is used. This is an application-specific processor (ASP) optimized for signal processing, with on-chip memory and peripherals.

A 32-bit floating-point DSP, architecturally optimized to process data of different bit format (8-bit, 16-bit, 32-bit), has been chosen. Due to its computing architecture peculiarity, it is able to process efficiently the 8-bit data of the image pixels and the 16-bit data of the voice samples. The 32-bit floating-point data format ensures the required computing precision both for the image and the speech processing.

3 Methods

The biometric authentication system (Fig. 5) consists of three processing layers, the feature-extraction layer, the matching layer, and the fuzzy logic-based decision layer. The feature-extraction layer uses signal processing-based algorithms for hard and soft feature extraction. The matching layer uses the hard computing (DSP) to identify the hard features and the soft computing (ANN) to identify the soft features. The decision layer uses the soft computing (FLE) to fuse the identification scores with the soft features.

3.1 Feature Extraction Layer

The feature extraction layer implements the biometric information capture (voiceprint and fingerprint) and the signal processing algorithm (hard computing) methods for feature extraction.

3.1.1 Voiceprint and Fingerprint Capture

The voiceprint is captured by a voice-type microphone, which is conditioned (linearized, amplified, and filtered), in the analog domain and then 16-kHz sampled, and 16-bit quantized. The utterance is optimally end-pointed [21] and segmented. The Hamming window is applied to extract 10-ms frames from the speech-data stream, using a 50 % overlap between adjacent frames to avoid data loss at feature-extraction time.

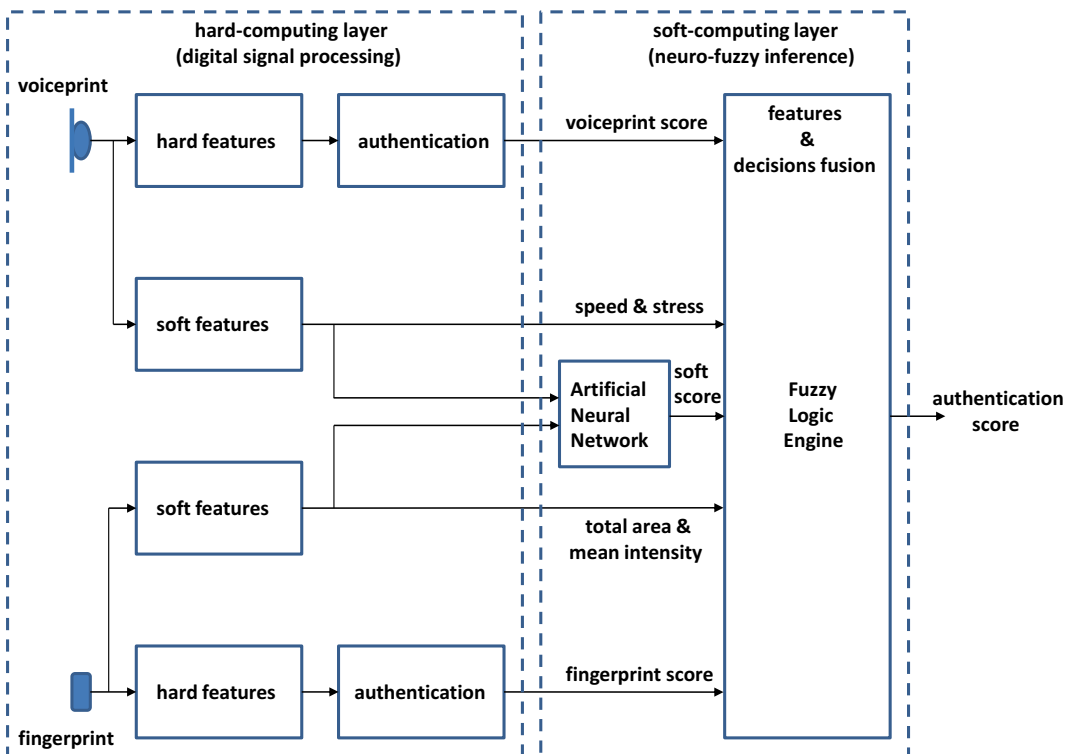


Fig. 5 System architecture

The fingerprint image is captured by a 512 dot-per-inch (dpi) solid-state fingerprint sensor. The image is available at the sensor output as 64,512 pixels 8-bit quantized array (224 rows and 288 columns) image.

3.1.2 Feature Extraction

Features (hard and soft) are extracted from the captured voiceprint and fingerprint applying the hard computing algorithm. Data are assembled in data structures useful to be processed at the pattern matching layer.

Voiceprint hard features are structured as time-sequenced vectors of data:

RMS(N)

ZCR(N)

AC(N)

CLPC(N)

where N is the vector length and each vector element is the feature measurement at the window application time.

Fingerprint hard features are structured sequences of minutiae as:

MINUTAE(M)

where M is the vector length and each vector element is the minutiae data.

From voiceprint the two soft features are measured as:

SPEED

STRESS

The two features are scalar data (one measurement executed on the whole captured voiceprint data stream).

From the fingerprint two soft features are measured as:

TOTAL_AREA

MEAN_INTENSITY

The two features are scalar data (one measurement executed on the whole captured fingerprint data array).

3.2 Matching Layer

The matching layer implements the hard and soft computing scoring systems. Each of these applies the appropriate algorithms for scoring the input data referred to a set of template data belonging to the persons to be identified. This layer runs in two different modes, enrolling and identifying.

The enrolling mode is active when a new person is to be added to the set of persons that are to be accepted. The identifying mode is active when a person (allowed or not allowed) is accessing the system furnishing its voiceprint and fingerprint. Both enrolling and identification modes run the same pattern matching algorithms, the first to store the reference templates, and the second to execute the scoring.

3.2.1 Enrollment Mode Voiceprint enrollment mode consists in storing as multiple templates the features measured each time the allowed person utters at the microphone a specific (requested) vocal sequence. Fingerprint enrollment mode consists in storing as multiple templates the features measured each time the allowed person puts a finger (requested) on the sensor.

3.2.2 Identifying Mode The voiceprint and fingerprint identifying mode runs the pattern matching algorithms to score the input hard features related to the stored templates. For each input (voiceprint and/or fingerprint) a score is available at the matching layer output.

3.2.3 Soft Biometric Training and Scoring The soft biometric scoring is executed running the FFBP-ANN. To do this the FFBP-ANN inputs the soft biometric features and outputs the score according to how it learned about the soft feature belonging to the authorized person. A training phase is requested to embed the knowledge in the FFBP-ANN nodes (neurons). This is run each time the soft features at the FFBP-ANN inputs belong to an authorized person, explicitly during the enrollment (training mode), and implicitly each time the person is identified as authorized (evolving mode). Learning is executed running the error back-propagation algorithm.

Back Propagating Networks (BPNs) are trained according to a generalized least mean-square (LMS) algorithm:

$$w_j(k+1) = w_j(k) + \eta(x^t(k) - x(k))f_j(k) \quad (10)$$

The weights $w_j(k)$ are modified by the k th input activity pattern $f_j(k)$, so that a new updated weight $w_j(k+1)$ is available at $k+1$ th input time. The modification is proportional to the difference between the $x^t(k)$ target response and the current response $x(k)$. The constant η controls the learning rate and is in the range $0 < \eta < 1$.

During the learning activity the difference between the target activity and the current activity at the output layer is a learning error indicator. To measure it, the total error E is measured as

$$E = \sum_{k=1}^K \frac{1}{2} \sum_{i=1}^N (x_i^t(k) - x_i^L(k))^2 \quad (11)$$

This is the total error measured at the N nodes of the output layer L after K patterns of the training set have been applied. If E is minimum, then weights at the end of the training are the best for the L -layer BPN.

Applying the best trained weights set to the three-layer FFBP ($L=3$) enables this ANN to perform the scoring of the biometric features. The ANN executes also the data fusion among the soft biometric data from voiceprint and fingerprint.

3.3 Decision Layer

The decision layer implements the data fusion and the decision fusion information from the matching layer. This is executed by the FLE that evaluates four kinds of data input:

- Voiceprint score.
- Fingerprint score.
- Soft-biometric measurements.
- Soft-biometric score.

Prior to a run, the FLE needs to be tuned for processing the data inputs and producing the decision. Membership functions and rules set have to be designed using the modeling and simulation toolbox.

Input data are fuzzified using optimally tuned membership functions (Figs. 6, 7, and 8a). Singleton membership functions are applied for rule consequents (Fig. 8b).

The rules are tuned combining the fuzzified inputs as follows (only the most significant are reported):

1. *IF voiceprint_score IS high AND fingerprint_score IS high AND soft_score IS high THEN authentication IS very high.*
2. *IF voiceprint_score IS medium AND fingerprint_score IS medium AND soft_score IS high THEN authentication IS high.*
3. *IF voiceprint_score IS medium AND speech_speed IS high AND soft_score IS medium THEN authentication IS high.*

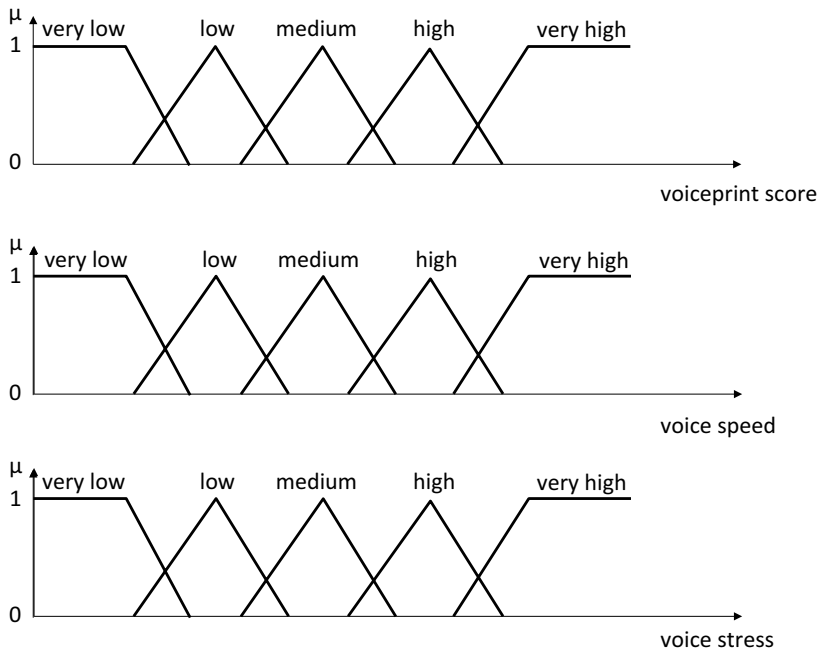


Fig. 6 Membership functions to fuzzify inputs (voiceprint)

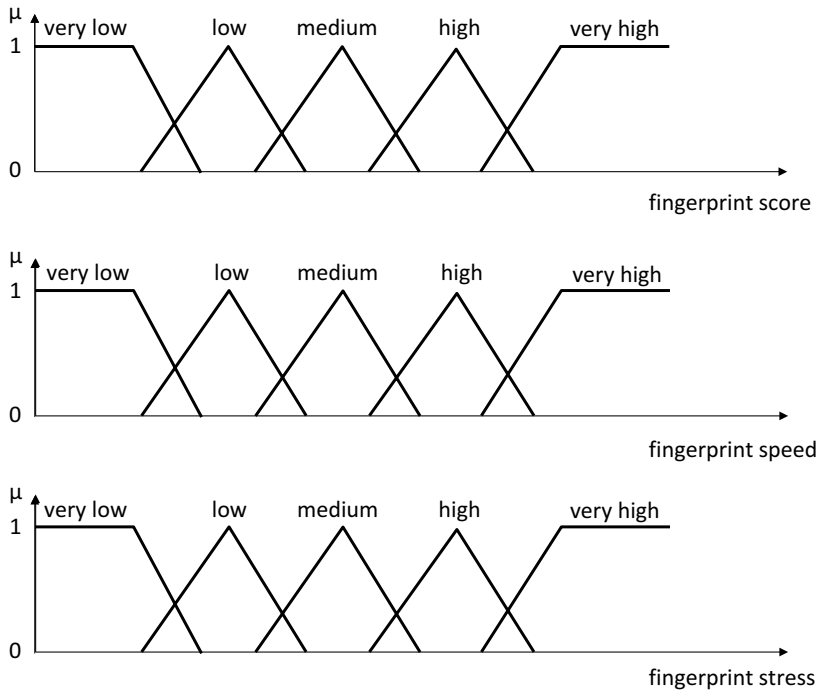


Fig. 7 Membership functions to fuzzify inputs (fingerprint)

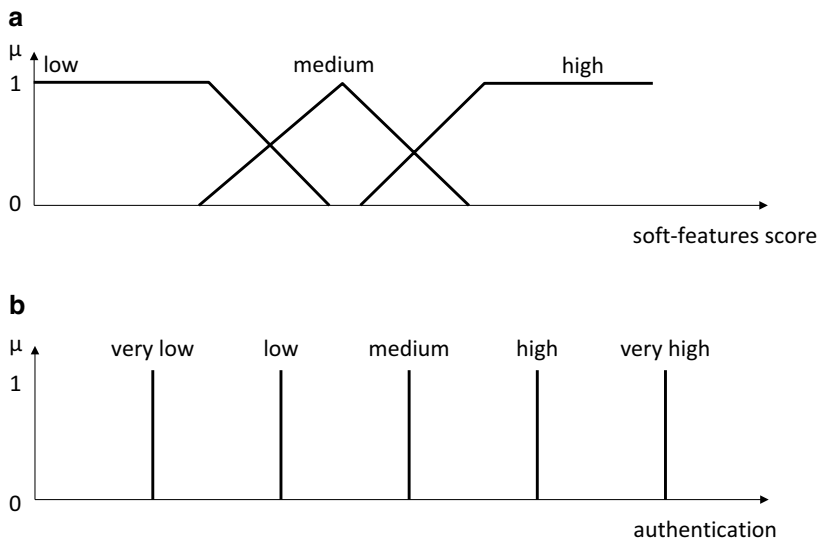


Fig. 8 (a) Membership functions to fuzzify inputs (soft-features score) and (b) to defuzzify outputs (authentication)

4. *IF voiceprint_score IS low AND speech_speed IS high AND speech_stress IS high THEN authentication IS average.*
5. *IF fingerprint_score IS medium AND fingerprint_total_area IS high THEN authentication IS high.*
6. *IF finger_print_score IS low AND fingerprint_total_area IS high AND fingerprint_mean_intensity IS high THEN authentication IS average.*
7. *IF voiceprint_score IS medium and fingerprint_score IS medium AND soft_score IS low THEN authentication IS low.*
8. *IF voiceprint_score IS high and fingerprint_score IS low AND soft_score IS low AND fingerprint_total_area_match IS low AND fingerprint_mean_intensity_match IS low AND speech_speed_match IS low AND speech_stress_match IS low THEN authentication IS very low.*
9. *IF voiceprint_score IS low and fingerprint_score IS high AND soft_score IS low AND fingerprint_total_area_match IS low AND fingerprint_mean_intensity_match IS low AND speech_speed_match IS low AND speech_stress_match IS low THEN authentication IS very low.*

Rules were derived from feature distribution. Each rule was manually tuned using a fuzzy logic rule editor, the simulator, and the knowledge of an expert:

- Rule 1 is a reinforcement of voiceprint and fingerprint matchers.
- Rule 2 combines a voiceprint matcher and a fingerprint matcher when both scores are too close to the decision threshold.
- Rules 3, 4, 5, and 6 act as recovery rules when the voiceprint or fingerprint matchers generate a false rejection.
- Rules 7, 8, and 9 protect against false acceptance.

For fine-tuning, many other rules can be generated to take additional soft-biometric measurements into account. Using more rules leads monotonically toward greater reliability in the authentication process.

Trapezoidal and triangular membership functions are used to process inputs. The inference rule set is then applied. The result of all the rules is evaluated using the WA method, so the crisp output value can be computed. Singleton membership functions were used to defuzzify the final decision.

3.4 Performance Evaluation

Performance evaluation aims to measure the reliability of the implemented biometric identification method. Voiceprint and fingerprint authentication were first implemented and tested separately, and then jointly, and finally combined through the fuzzy logic inference engine and the artificial neural network applied to soft-biometric features. An “all-against-all” test strategy was applied to obtain match and mismatch scores.

Evaluation of joint voiceprint and fingerprint authentication consists of taking as good the better of the two matches (OR). Single-user authentication was performed, so this test had minimal system requirements. The following results were produced:

- Voiceprint alone: 90.5 % correctly accepted.
- Fingerprint alone: 85.7 % correctly accepted.
- Voiceprint OR fingerprint: 92.3 % correctly accepted.
- Fuzzy logic decision fusion of voiceprint, fingerprint, and artificial neural network evaluated soft features: 95.8 % correctly accepted.

The OR test confirmed that multi-biometrics can improve performance compared to single-biometric authentication. System performance can be significantly improved, while keeping complexity to a minimum, using fuzzy logic as decision fusion and reinforcing it with an artificial neural network applied to soft-biometric features.

4 Notes

1. A dual microphone or an array of microphones needs to be used when the biometric application is targeted to outdoor applications and the person to be identified is close to other persons. Beam forming can be implemented for noise reduction purpose.
2. The capacitive fingerprint sensors are implemented as a two-dimensional or a mono-dimensional (strip) scanner. The performance of the biometric system is not sensitive to this form factor. The first is less computationally intensive and more intuitive, but it is not optimal for system dimension reduction. The second is computationally intensive because it implies that the two-dimensional image has to be built by software and it is less intuitive, but it is optimal for system dimension reduction.
3. Weighted Average (WA) defuzzification method is a derivation of the Centroid (Center of Gravity) method that fits well the singleton membership function.

References

1. Jain AK, Pankanti S, Prabhakar et al (2004) Biometrics: a grand challenge. Proc 17th international conference on pattern recognition (ICPR), vol 2. pp 935–942
2. Anil K, Jain AK (2012) Biometric recognition: an overview. In: Mordini E, Tzovaras D (eds) Second generation biometrics: the ethical, legal and social context. The International Library of Ethics, Law and Technology, vol 11. Springer, pp 49–79
3. Baird SL (2002) Biometrics: security technology. The technology teacher 61(5):1, pp 8–22

4. Sujithra M, Padmavathi G (2012) Next generation biometric security system: an approach for mobile device security, Proc second international conference on computational science, engineering and information technology. ACM, New York, NY, USA, pp 377–381
5. Corcoran P, Cucos A (2005) Techniques for securing multimedia content in consumer electronic appliances using biometric signatures. *IEEE Trans Consumer Elect* 51:545–551
6. Jain AK, Ross A, Pankanti S (2006) Biometrics: a tool for information security. *IEEE Trans Inf Forensics Security* 1(2):125–143
7. Jain AK, Nandakumar K, Lu X et al. (2004) Integrating faces, fingerprint, and soft biometric traits for user recognition. Proc biometric authentication workshop, LNCS 3087, Prague, pp 259–269
8. Hong A, Jain S, Pankanti S (1999) Can multi-biometrics improve performance? Proc AutoID'99, Summit, NJ, pp 59–64
9. Anil J, Lin H, Sharath P (2000) Biometric identification. *Commun ACM* 43(2):90–98
10. Cappelli R, Maio D, Maltoni D et.al (2006) Performance evaluation of fingerprint verification systems. *IEEE Trans Pattern Anal Mach Intell* 28:3–18
11. Bishop C (1995) *Neural networks for pattern recognition*. Oxford University Press, Oxford
12. Ciota Z (2001) Improvement of speech processing using fuzzy logic approach. Proc of IFSA World congress and 20th NAFIPS Int Conf, 2
13. Bosteels RTK, Kerre EE (2007) Fuzzy audio similarity measures based on spectrum histogram and fluctuation patterns. Proc Int Conf Multimedia and Ubiquitous Engineering 2007, Seoul, Korea, 27–28 April
14. Malcangi M (2002) Soft-computing approach to fit a speech recognition system on a single-chip. Proc 2002 international workshop system-on-chip for real-time applications, Banff, Canada, 6–7 July
15. Malcangi M (2004) Improving speech endpoint detection using fuzzy logic-based methodologies. Proc thirteenth Turkish symposium on artificial intelligence and neural networks, Izmir, Turkey, pp 10–11
16. Wahab A, Ng GS, Dickiyanto R (2005) Speaker authentication system using soft computing approaches. *Neurocomputing* 68:13–17
17. Kar B, Kartik B, Dutta PK (2006) Speech and face biometric for person authentication. Proc IEEE international conference on industrial technology, India, pp 391–396
18. Jang-Hee Y, Jong-Gook K et al (2007) Design of embedded multimodal biometric systems. Proc of the int conf on signal image technologies and internet based systems, SITIS 2007:1058–1062
19. Jain AK, Dass SC, Nandakumar K (2004) Soft biometric traits for personal recognition systems. Proc international conf on biometric authentication, LNCS, vol 3072. Hong Kong, pp 731–738
20. Pak-Sum Hui H, Meng HM, Mak M (2007) Adaptive weight estimation in multi-biometric verification using fuzzy logic decision fusion. Proc IEEE international conference on acoustic, speech, and signal processing ICASSP'07, vol 1. pp 501–504
21. Runkler TA (1997) Selection of appropriate defuzzification methods using application specific properties. *IEEE Trans Fuzzy Sys* 5: 72–79
22. Abiyev RH, Altunkaya K (2007) “Neural network based biometric personal identification”, *Frontiers in the Convergence of Bioscience and Information Technologies*. pp 682–687
23. O'Shaughnessy D (1987) *Speech communication—human and machine*. Addison-Wesley, Reading, MA

Chapter 14

Using Neural Networks to Understand the Information That Guides Behavior: A Case Study in Visual Navigation

Andrew Philippides, Paul Graham, Bart Baddeley,
and Philip Husbands

Abstract

To behave in a robust and adaptive way, animals must extract task-relevant sensory information efficiently. One way to understand how they achieve this is to explore regularities within the information animals perceive during natural behavior. In this chapter, we describe how we have used artificial neural networks (ANNs) to explore efficiencies in vision and memory that might underpin visually guided route navigation in complex worlds. Specifically, we use three types of neural network to learn the regularities within a series of views encountered during a single route traversal (the training route), in such a way that the networks output the familiarity of novel views presented to them. The problem of navigation is then reframed in terms of a search for familiar views, that is, views similar to those associated with the route. This approach has two major benefits. First, the ANN provides a compact holistic representation of the data and is thus an efficient way to encode a large set of views. Second, as we do not store the training views, we are not limited in the number of training views we use and the agent does not need to decide which views to learn.

Key words Visual navigation, Computational neuroscience, Machine learning, Boosting, Restricted Boltzmann machine, Infomax

1 Introduction

All animals are faced with the challenge of extracting useful information from the environment and using that information to shape their behavior in an adaptive way. A major goal for neuroscience is to understand how neural circuits are organized to achieve that process. This goal depends on understanding the information provided by the environment as well as the information required for a particular behavior. For many animals, the primary sensory channel is vision and scientists have long sought to understand how vision drives behavior and underpins perception by asking questions such

as “What does the frog’s eye tell the frog’s brain?”¹ or “How do humans recognize their grandmother?”²

If we take the example of a very simple visually guided behavior, we can begin to understand the style of information processing that we see in biological systems. Many animals perform phototaxis, where they try and move in the direction of the brightest light; for instance to gain warmth or rise to the surface of the sea. For some microorganisms, such as zooplankton, we know in detail the visual circuits that produce this behavior. They have two photoreceptors which detect light from one side of the body only. These photoreceptors are directly connected to motile hair cells [1] which beat rhythmically to propel the organism through the world. Light on one side of the organism stimulates one photoreceptor, which inhibits the movement of the hair cells on the same side of the organism. The undamped beating from hairs on the “dark” side of the body will ensure a turn towards the light. Evolution tends not to produce overelaborate solutions; we see that the zooplankton has a minimal yet perfectly efficient sensory circuit for the task in hand.

For more complex visually guided behaviors, however, it is nontrivial to understand what an efficient sensory system would look like and we are faced with the task of being objective about animals with visual systems that are very different from ours, who have different perspectives on the world [2, 3] and different behavioral requirements. As it is objective in terms of the way it treats sensory data, a good first step in this endeavor is to try to explore regularities in the sensory information perceived by animals during their natural behavior.

One available method, which can be used to objectively explore sensory data, is to use artificial neural networks (ANNs), or machine learning more broadly, to explore the information available in a raw sensory array. Such uses of ANNs provide a mapping from the sensory environment to behavior. Essentially ANNs can be used to interrogate complex data and find the regularities and affordances that biological agents might well be tuned to (rather than being used as explicit models of animal brains). Here, we present case studies describing how we have used ANNs to understand how animals might use visual information for navigation.

¹ Here Lettvin and colleagues found that parts of the frog’s visual system are tuned to detect small moving objects. Thus the eye is providing information to the frog about where to direct the tongue in the hope of catching a fly. McCulloch and Pitts, authors of this paper, are commonly held up as pioneers of neural network research. J. Y. Lettvin, H. R. Maturana, W. S. McCulloch, and W. H. Pitts, “What the Frog’s Eye Tells the Frog’s Brain,” *Proc. IRE* 47 (1959) 1940–1951.

² Lettvin again came up with the concept of a sparse code in the brain where small numbers of individual neurons may be highly selective to concepts such as one’s grandmother. See Quian-Quiroga, R., Fried, I., and Koch, C. (2013) *Brain Cells for Grandmother*. *Scientific American*, Feb.

Navigation is an essential task for most animals [4] and indeed some artificial autonomous systems. One of the champions of this behavior is the ant whose foragers spend much of their working life travelling huge distances searching for food and then accurately and efficiently bringing that food back to their nest along idiosyncratic habitual routes [5]. Ants possess a number of mechanisms for orientation, including for some species social cues provided by pheromones [6]. However, for many ants the principal source of information for navigation comes from the visual scenes that they experience when travelling along their familiar routes [7].

Behavioral experiments have shown us the efficient way that ants use vision for navigation which depends on a fundamental property of the visual world. In a complex world, two photographs taken with the same camera can only be identical when the camera location and orientation are matched. However, Zeil et al. [8] showed that the difference between images taken from different nearby locations is minimized when the orientations match. This is also true for natural visual systems. Thus, if a view of the world from a forward-facing camera is memorized when travelling along a route, that memory can be used later to recover the original direction of travel from nearby locations, by finding the best match with the currently perceived view [8, 9]. In behavioral terms for the ant this means scanning the world and finding the direction that provides the best match with their memory [10]. Using this simple strategy the required knowledge for an experienced ant forager is equivalent to the views required to set the appropriate directions at all locations along their foraging routes.

What kind of challenge does this present to an ant? The views experienced by ants are of low resolution (Fig. 1) with an almost panoramic scene being encoded with the equivalent of 1,000 pixels. What's more their visual systems, unlike those of flying insects, are not particularly fast, so the ant may have an effective "frame rate" of roughly ten views per second. However, we know that ants easily learn routes 10s of metres in length and, given their walking speed, we can estimate that along a single foraging route ants will experience views of the world totalling millions of pixels. Theoretically, if ants had perfect memories they could store the raw pixel values for every view experienced along a route and use those perfect "photographic" memories for navigation. However, this seems unlikely given a total brain size of 500,000 neurons. Surely evolution has been able to come up with a more efficient system both in terms of efficiencies in the way a visual system encodes a view and also in the way visual information is stored. Here we describe how we have used ANNs to explore efficiencies in vision and memory that might underpin visually guided navigation in complex worlds.

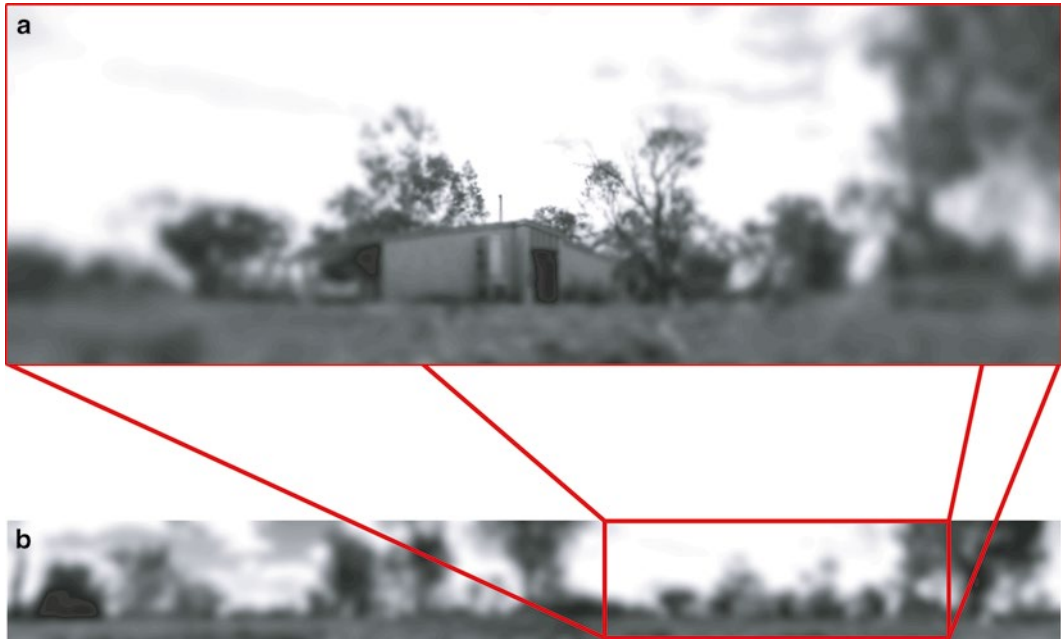


Fig. 1 (a) The world as seen by a human eye. The human field of view is quite large, almost 120° wide. However, we only have high-resolution vision at the center of our visual field, which we make up for by moving our eyes. (b) An ant's eye view of the world. Ants have a horizontal field of view spanning almost 360° , but with homogenous low resolution. Notice how you cannot recognize the house in the ant's eye image. This picture shows a vertical angle of 45° , but ants do see at increased elevations, because they extract celestial compass information

2 Our Approach

Biological background: Ants can learn the information to visually guide a route after traversing it once [11]. Subsequently, they habitually travel within a corridor defined by the views experienced along the route. These stored views define the directions in which to travel [12, 13] for all locations along the route. Perhaps surprisingly, ants do not seem to learn discrete waypoints representing key locations along a route [9]. Further, ants accomplish this task with a low-resolution visual system (Fig. 1) which is not well suited to object recognition but provides ants with a coarse visual encoding representing the broad layout of terrestrial objects contrasted against the sky [14, 15].

Problem from a machine learning perspective: Given a single route traversal (to gather training images), we need to learn a compact representation which encodes the set of training images such that they can subsequently be used to navigate the route. Specifically, the network which has been used to encode the views should be able to be used to set a direction of travel given the current visual input from locations along or close to the original route.

Insight: We can use the views experienced in the first route traversal as training data for an ANN designed to perform familiarity judgements. After training the ANN will be able to output a familiarity score for any novel view presented to it.

The fact that ants are moving, and therefore facing, in the correct direction most of the time during the first route traversal means that these training views implicitly define the movement directions required to stay on the route. Thus, it is sufficient to learn the views as they are experienced. After the initial route learning, the problem of navigation is reframed as a search for views similar to those associated with the route. If a novel view is familiar, it is likely that a similar view has been experienced before and so the ant is likely to be close to the route and heading in an appropriate direction. So, by intermittently scanning the environment and moving in the direction that gives the most familiar view an ant can reliably retrace its route.

To learn a route we thus train an ANN to learn the regularities within a series of views in such a way that it can make a familiarity judgement about any new view presented to it. This approach has two major benefits. First, an ANN is an efficient way to encode a series of views. We do not attempt to store every view experienced along the route, but instead use them to train the ANN. Second, and as a corollary, because we are not storing all the views used for training, we are not limited in the amount of training views we can use and so the agent does not need to decide when or which views to learn.

There are many different possibilities for training ANNs for this task. The only prerequisite for our approach is that the ANN should provide a measure of the familiarity of a presented view. In our case we also wanted biological plausibility, in the sense that the vision, memory size, and learning behavior should be plausibly achieved by an ant. These requirements have driven our research trajectory.

3 Approach 1: ANN as a Classifier

Our first attempt at modelling route navigation with an ANN [16] used training data to establish a classifier which could determine whether a given view is an on-route view (i.e., facing the correct direction) or an off-route view (i.e., facing in a non-route direction). When novel views were presented to the classifier the associated confidence in the classification could then be used as a proxy for the familiarity of the presented view.

In this approach, a robot with a panoramic camera is first moved along a predefined route to gather the training images (every 4 cm) along the route. In order to train a classifier it is necessary to generate positive and negative training examples of the input to be classified. In our case this means collecting views that

are part of the route and views that are not part of the route. The positive examples are simply the forward-facing views experienced along the route. The negative views consisted of views from the route taken facing to the left and right of the direction of movement at an angle of 45° relative to the route heading (Fig. 2a).

Classifying views is a difficult task because of the high dimensionality of the input if one adopts a pixel-by-pixel representation. In order to make learning more tractable one can project this high-dimensional space into a lower dimensional space by passing views through a set of visual filters. Following Viola and Jones [17] we chose to construct a classifier using randomly generated filters composed of Haar-like features. Haar-like features act like edge detectors or crude approximations to Gabor filters and are maximally activated at high-contrast boundaries in the image (Fig. 2b). The set of outputs from these feature detectors form the basis of our image representation and are used to train a classifier via the Adaboost learning algorithm, a commonly used variant of boosting [18].

Boosting is a supervised learning technique for constructing a strong classifier from a set of weak classifiers given a training set of labelled positive and negative examples. A weak classifier is one that performs only slightly better than chance. Conversely, a strong classifier is one that performs arbitrarily well. A strong classifier is constructed from a linear weighted combination of the outputs of weak classifiers. In our context, each visual filter is a candidate weak classifier, $h_j(x)$, and consists of a Haar feature f_j , a threshold θ_j , and a parity p_j that determines whether the output should be greater or less than the threshold

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{else} \end{cases} \quad (1)$$

This process is illustrated in Fig. 2c–d. We initially generate a large pool (5,000) of such filters of random types and sizes and at random positions in the visual field. We then use Adaboost to select and combine a prespecified number, T , of these weak classifiers into a strong classifier. By specifying T , we thus control the complexity of the classifier. As an aside we highlight an interesting outcome of the boosting process. Because Adaboost starts with a large pool of feature detectors, and then performs feature selection to pick out and use only those features that are most useful for the current classification problem, we can interrogate sets of filters to see if they relate to biologically salient features or portions of the world.

The Adaboost algorithm works as follows. At each iteration, the training data x_i (m on- and l off-route views) are weighted according to a distribution that indicates the current importance of each example in the dataset, with initial weights $w_{0,1}$ set as $1/(2m)$ or $1/(2l)$ for on- and off-route views, respectively. The weak classifiers are evaluated according to the weighted error

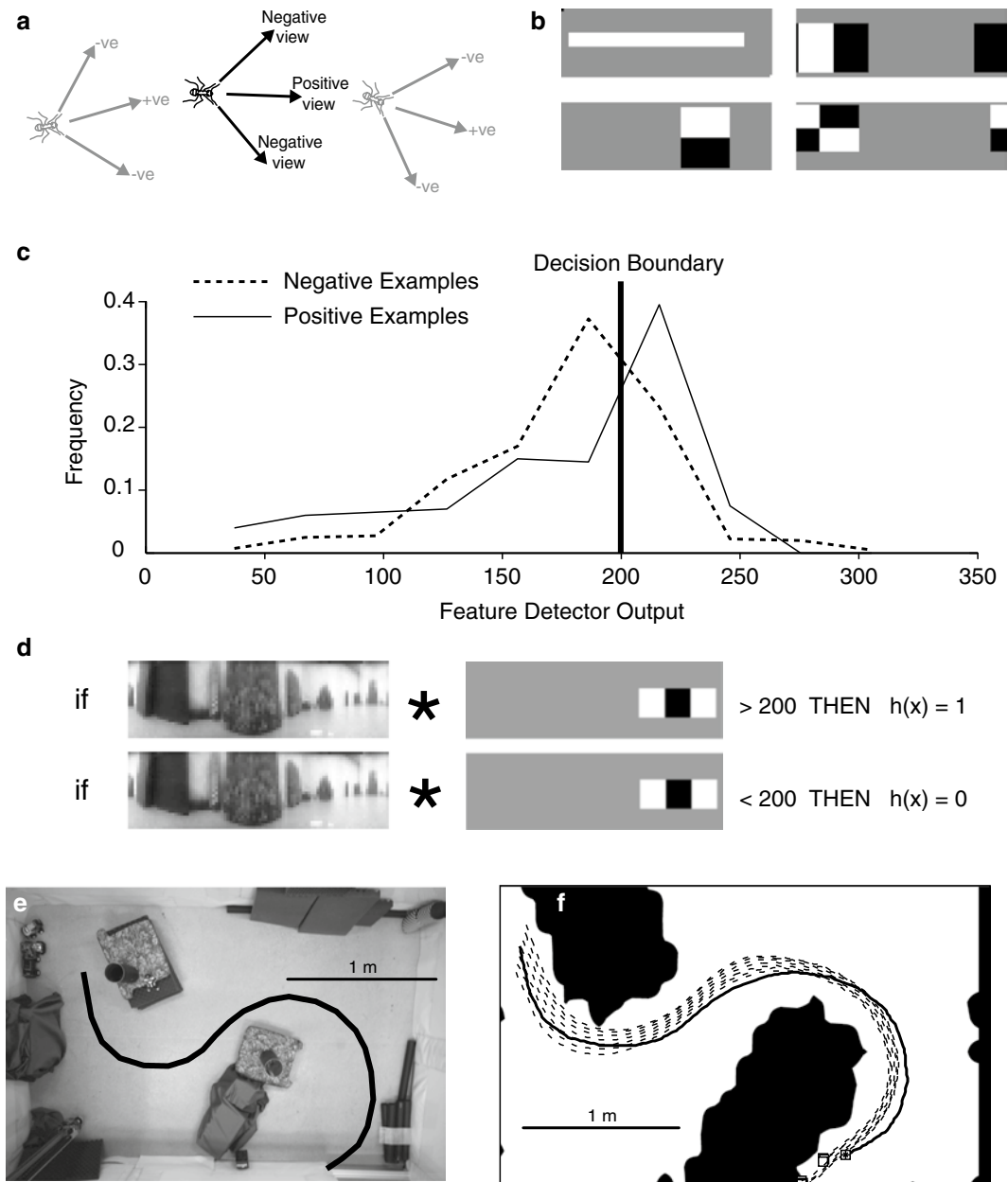


Fig. 2 Using an ANN as a classifier of on- and off-route views. **(a)** Training images are collected in three directions every 4 cm along the training route. A forward-facing image is collected as an example of an on-route view and two off-route views are also collected at $\pm 45^\circ$ relative to the route heading. **(b)** Examples of the four different classes of Haar-like feature detectors, from *left to right*, a single block, edge-detector, Gabor-like edge detector, and diagonal edge detector. Each feature detector can vary in size, shape, and position. As images are panoramic, feature detectors wrap-around if they extend beyond the left or right edge of the image. *White* is a weight of 1, *grey* 0, and *black* -1. **(c-d)** Constructing a weak classifier using a single Haar-like feature. Each training image is convolved with the feature detector to determine an optimal threshold which determines the output for the unseen data. **(c)**: Distribution of feature detector outputs for the two classes for a Haar feature f . A threshold, θ , is determined that optimally separates the distributions (*vertical bar*). **(d)** The final weak classifier is defined by a feature detector, a threshold, and a parity. For the example shown, the output of the weak classifier, $h(x)$, for the image, x , is 1 if the feature detector output exceeds a threshold of 200, or 0 otherwise. **(e-f)** Learning a circuit of the gantry workspace. The *dark line* represents the training route. **(e)** The gantry workspace as viewed from above. **(f)** Performance of the algorithm from ten different start positions (*dashed lines*) using a boosted classifier with 50 features

$$e_{t,j} = \sum_{i=1}^{m+l} w_{t,i} |h_j(x_i) - y_i| \quad (2)$$

where y_i is 1 for on-route and 0 for off-route views. If the minimum error across the classifiers at iteration t , e_t , is 0, the algorithm stops. Otherwise, the classifier associated with the lowest error is denoted as h_i , added to the strong classifier and removed from the pool of features. The weights associated with the training data are then updated according to

$$w_{t,i+1} = w_{t,i} \left(\frac{e_t}{1 + e_t} \right)^{1-c_i} \quad (3)$$

where $c_i=0$ if x_i is classified correctly and 1 if not, with weights normalized after each iteration, so they sum to 1. Essentially, this process increases the weights of incorrectly classified views relative to correctly classified views, thereby encouraging the next weak classifier to focus on examples that were incorrectly classified at the last iteration. Weak classifiers are added until the overall classification performance exceeds a threshold or the maximum number of weak classifiers, T , is reached. The final classification is then

$$H(x) = \text{sign} \left(\sum_{i=1}^T a_i h_{i(x)} \right) \quad (4)$$

with an associated confidence value of

$$C(x) = \left\| \sum_{i=1}^T a_i h_i(x) \right\| \quad (5)$$

which is simply the degree to which the sum of the combined weak classifiers differs from zero, prior to the sign being taken. This confidence value is key to our approach as it is used to determine the most likely direction of travel in subsequent navigation.

To navigate a route the classifier is fed multiple views in different directions from a given position. By weighting each of the viewing directions that produce positive classifications by their associated confidence values, a movement direction is determined which is most likely to keep the agent on the learned route. To demonstrate the efficacy of this approach we implemented it on a large Cartesian gantry robot equipped with a panoramic camera moving through a $3 \text{ m} \times 2 \text{ m}$ environment (Fig. 2c). During a training run a series of views are collected for on- and off-route directions (Fig. 2c). These views are used to train our classifier following the procedures outlined above. During route recapitulation the camera is positioned at the start of the route facing in the correct direction. From this position views are sampled in a range of directions from -60° to $+60^\circ$ in steps of 5° relative to the current heading. Views

are passed into the classifier and all of the viewing directions that produce a positive classification contribute to a weighted average which determines the direction of travel and a 5 cm step is made in this direction. The process is then iterated.

As can be seen (Fig. 2f), the approach works well and robust route navigation is achievable. As is typical of ants, the results show that the agent navigates along a corridor defined by the training path but influenced by noise and the environment. It was also noted that the routes “smoothed out” some of the kinks that were present in the original training run. While results were dependent on the number of classifiers used (more classifiers resulted in better performance) near-perfect results were achieved with 50 classifiers, while very robust results could still be achieved with only 20 classifiers. We have thus shown that it is possible to learn a nontrivial route through an environment using a simple view classification strategy based on positive and negative views collected during a single episode of learning. By using an ANN as a classifier, we reframed the problem of route navigation in terms of a search for familiar views. The benefits of the ANN are that it both provides a compact, holistic representation of the information required for route navigation but also a measure of the expected uncertainty of the classification.

We have thus demonstrated the principle that ANNs can be used to learn a holistic representation of the views that determine a route. Secondly, we have shown that a coarse visual system CAN encode a complex world because it is used to drive a specific behavior through familiarity recall rather than object or place recognition.

4 Approach 2: Training an ANN Without Negative Examples

One issue with the solution presented above is the biological-plausibility of the training as it requires both on- and off-route views. Additionally, an iterative Adaboost algorithm does not seem viable for implementation in the brain.

For our second approach we used the same general method of employing a training route to gather data but in a simulated world which mimics a desert ant habitat and therefore has the inherent richness of natural visual scenes (Fig. 3a). Additionally, we did not want to use negative examples in training. If we have no negative example to tell us we are wrong, an alternative approach is to learn a compact approximation of the distribution of views encountered during training which can subsequently be used to infer whether or not a novel view is part of the original training set.

An efficient way to do this is to learn regularities in the views experienced via sets of feature detectors which reduce the dimensionality of the sensory data. One very successful machine learning approach to this kind of problem is the use of “autoencoder”

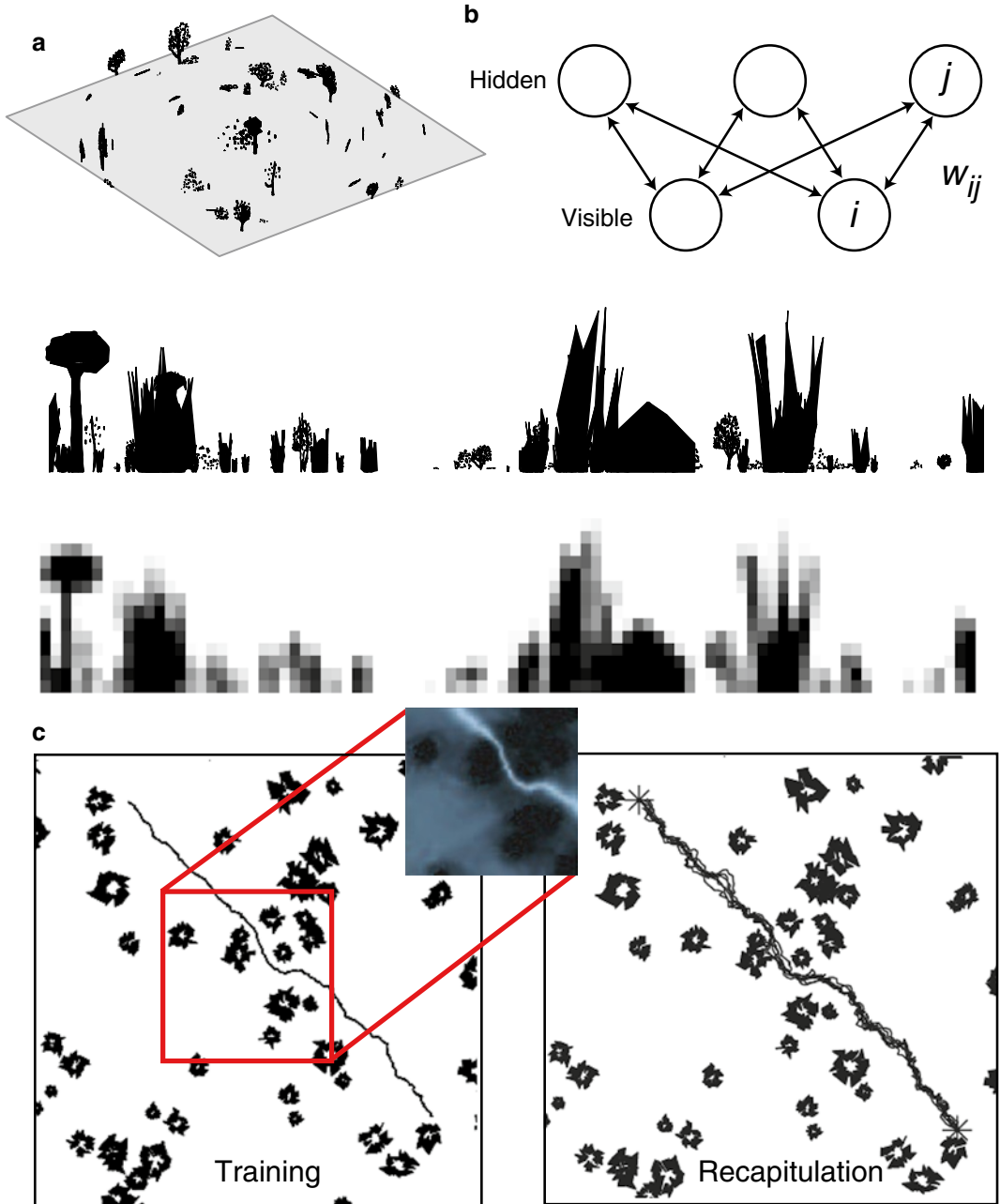


Fig. 3 (a) The simulation environment. *Top left:* A typical simulated environment. *Middle:* High-resolution view of the world from an ant's perspective. View is panoramic, so forward is middle of the row and the view wraps round from left to right. *Bottom:* Low-resolution representation of the view shown in *middle panel*. (b) Architecture of a Restricted Boltzmann Machine. The visible nodes are associated with the input data and the hidden nodes with the hidden variables of the model to be learnt. (c) *Left:* The training route taken through a complex desert ant habitat. *Upper right:* A zoomed-in view of the familiarity landscape produced by the trained network. *Right:* Ten recapitulated routes through the environment

networks [19] which are able to both learn a compact representation of the data distribution and, after training, auto-generate the data. These processes can be thought of as encoding and decoding the original data, effectively learning a generative model. In the application of this approach to our ant navigation problem, the way in which visual features are selected is not generated in a supervised fashion as in Approach 1, but is driven by statistical regularities within the set of views experienced as the ant follows a route through the world.

A powerful implementation of “autoencoder” networks is as a Restricted Boltzmann Machine (RBM) [20, 21]. Hence we attempt to learn a compact representation of the distribution of views experienced along the route by training an RBM on the set of training views. Once the RBM has been trained we can use it to infer the probability that a novel view was part of the training set. Behavior is now driven by scanning the world and searching for the viewing direction with the highest such probability, i.e., the one most likely to be part of the training route.

The basic RBM architecture is shown in Fig. 3b. Nodes in the network are stochastic binary neuron-like elements. The visible nodes are associated with the visible variables (in our case the pixel values from on-route views) and the hidden nodes are associated with the hidden variables (or “causes”) inherent in the data (in our case the higher level features). The layers are joined by symmetrically weighted connections in the manner shown in the figure, but there are no visible-visible or hidden-hidden connections. This latter constraint is what makes this class of network “restricted” in relation to the standard fully connected Boltzmann Machine [22, 23] and greatly simplifies learning and inference [21]. In general, RBMs can be stacked to create several hidden layers where the outputs of one layer act as the data for training the next layer (producing so-called deep networks) [19]. In our ant problem application a single hidden layer sufficed—we used an RBM with 1,360 visible units and 500 hidden units.

The stochastic nature of the RBM network nodes means that it is natural to import methods and language from statistical physics in describing their operation. Hence we talk about a joint configuration (v, h) of the visible and hidden units having an “energy” given by the function $E(v, h)$ shown in Eq. 6

$$E(v, h) = - \sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i, j} v_i h_j \quad (6)$$

where v_i and h_j are the binary states of the visible unit i and hidden unit j respectively, a_i , and b_j are their biases and m_{ij} is the weight between them. Through this energy function it is possible to express the probability of a given configuration by

$$P(v, h) = \frac{e^{-E(v, h)}}{Z} \tag{7}$$

where Z is a normalization constant. Thus, the probability assigned to a single data point, v , is found by summing over all possible hidden vectors as shown in Eq. 8

$$P(v) = \sum_h P(v, h) = \frac{\sum_h e^{-E(v, h)}}{Z} \tag{8}$$

Following a maximum likelihood principle, training an RBM consists of increasing the probability that the network assigns to the observed data by adjustment of the weights and biases. This is achieved by decreasing the energy of the observed data and increasing the energy of unobserved configurations using some kind of gradient descent algorithm [20, 21]. There are many possibilities, but in the current work we employ an efficient approach called Fast Persistent Contrastive Divergence; full details of the training procedure can be found in Hinton [21, 24] and its application to the ant problem in Baddeley et al. [25].

Having trained an RBM, the probability of a visible datum (in our case the visual scene) is found by summing over all possible configurations of the hidden units as described by Eq. 8. Unfortunately this computation is intractable for large h . However, we can calculate the probability of a visible datum up to a normalization constant by using an alternative, tractable, formulation for the probability of a datum expressed in terms of the free energy, $F(v)$, as described by Eq. 9:

$$F(v) = -\log \sum_h e^{-E(v, h)} \tag{9}$$

which allows us to write $P(v) = e^{-F(v)} / Z$, where Z is a normalization constant. For an RBM the free energy has an analytic form given by Eq. 10:

$$F(v) = \sum_i a_i v_i - \sum_j \log(1 + e^{x_j}) \tag{10}$$

where $x_j = b_j + \sum_i w_{ij} v_i$. Since we wish to compare the relative probabilities of different views, the normalization constants cancel out. This allows us to directly compare the probabilities that each of the views in a scan is part of the training route by simply calculating their free energies.

The routes we produced can be seen in Fig. 3c. As before, there is a broad route corridor defined by the training data and the structure of the environment. As with the classifier approach, this shows that an ANN-based holistic memory can achieve robust route homing.

However, in this instance, no “negative views” were needed and so the approach is more attractive in terms of biological plausibility. In addition, a nice feature of this algorithm is that once trained, the network can be queried to see the types of regularities that have been encoded which again allows us to understand the visual filters that are appropriate for the visual navigation.

Unfortunately, the training process for RBMs has a so-called “burn-in” period before unbiased samples can be produced. This results in a computation and time heavy algorithm. So while this works satisfactorily without negative training views, it is not ideal for our biological desideratum of one-trial learning.

5 Approach 3: Learning and Discarding Views

We have shown that it is possible to learn a compact holistic route memory that gives a confidence that any novel view is part of that route and that this can be done without off-route (i.e., negative examples). However, so far we have used an ANN’s confidence in its classification as a proxy for familiarity. In so doing, during ANN training we have needed to have all the views available which seems somewhat counterintuitive biologically; the agent would need an almost perfect memory before then being able to acquire a compact representation by training the network. We now turn to an alternative network and training regime. Instead of storing all of the views experienced on a training route, the views are used to train a two-layered ANN to perform familiarity discrimination using an Infomax learning rule [26]. That is, once trained, the network takes a view as input and outputs a familiarity measure based on the views that the network was trained with. The key difference to the previous ANN approaches is that each training view is presented to the network once only and then discarded following a single cycle of weight updating. Thus the memory load does not scale with the length of route but remains constant.

The ANN architecture (Fig. 4a) consists of an input layer and a novelty layer. The number of input units is equal to the dimensionality of the input which in our case is the number of pixels in a view of the world, $N=90 \times 17=1,530$. The number of novelty units, M , is arbitrary and we typically used the same number of novelty units as inputs, i.e., $M=N=1,530$. We found that using as few as 200 novelty units can still work well, but we did not explore this aspect of the problem in any detail. The network is fully connected by feedforward connections w_{ij} with the activation of the i ’th novelty unit being

$$a_i = \sum_{j=1}^M w_{ij} x_j \quad (11)$$

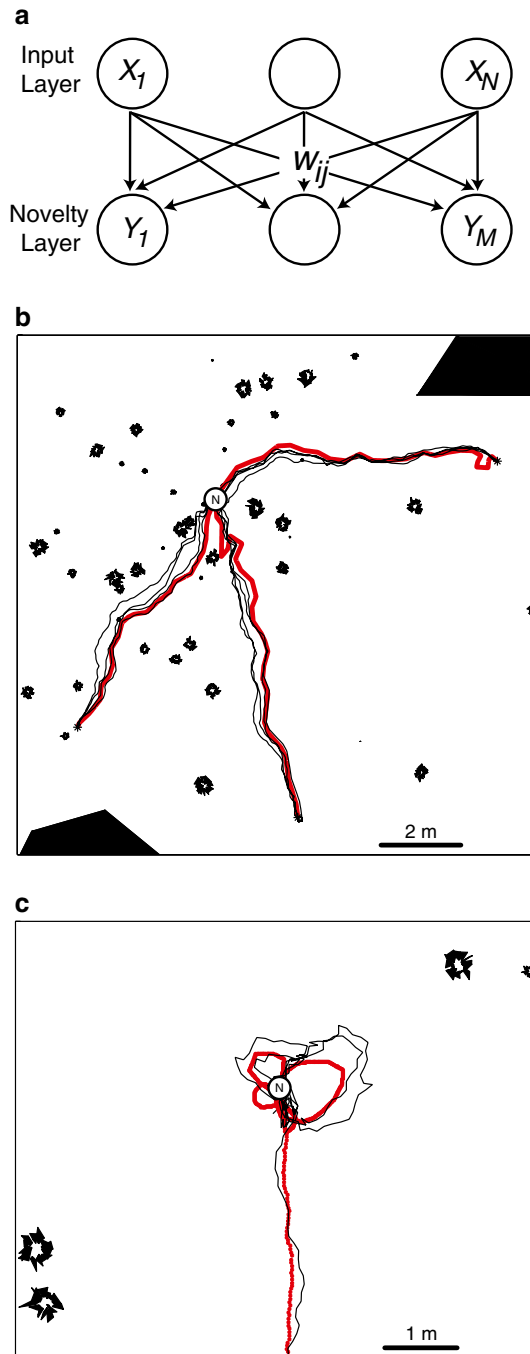


Fig. 4 (a) The two-layered network used with the Infomax learning rule. *Circles* represent units and *arrows* denote connections between the input units on the *top* and the novelty units on the *bottom*. There is no output from the network as such since the response of the network is a function of novelty unit activations. (b) Navigating using a trained ANN to assess scene familiarity. Three separate routes (*thick grey lines*) learned in an environment containing both small and large objects. For each of the three routes, that consisted of between 700 and 980 views taken every 1 cm, we show three recapitulations (*thin black lines*). During route recapitulations the headings at each step were subject to normally distributed noise

where x_j is the value of the j 'th input pixel. Weights are initialized randomly from a uniform distribution in the range $[-0.5, 0.5]$ and then normalized so that the mean of the weights feeding into each novelty unit is 0 and the standard deviation 1. The Infomax approach [27] decomposes each view into a fixed number of components (determined by the number of hidden units in the network) which remains constant, independent of the number of views experienced. It does this by adjusting the weights so as to maximize the information that the novelty units provide about the input, by following the gradient of the mutual information. Since two novelty units that are correlated carry the same information, adjusting weights to maximize information will tend to decorrelate the activities of the novelty units and the algorithm can thus be used to extract independent components from the training data [28]. The core update equation is

$$\Delta w_{ij} = \frac{\eta}{N} \left[w_{ij} - (y_i + b_i) \sum_{k=1}^N b_k w_{kj} \right] \quad (12)$$

where $y_i = \tanh(b_i)$ is the output of each novelty unit, and $\eta = 0.01$ is the learning rate. This performs gradient ascent using the natural gradient [29] of the mutual information over the weights [28] which avoids the computationally expensive calculation of the inverse of the entire weight matrix.

Once trained, subsequent route navigation is achieved by the agent sampling views at different headings from a given position. Each view is fed into the ANN and the agent takes a 5 cm step in the direction associated with the most familiar view where the familiarity of a view, X , is given by the network output:

$$d(X) = \sum_{i=1}^M |b_i| \quad (13)$$

The network response could be viewed as an output layer but as it is a function of the activations of the novelty units, we follow [26] and do not represent it with another layer. The Infomax familiarity measure is abstract and reflects whether an input is well described in terms of the learned components that the hidden units represent. While with a limited amount of data the algorithm is unlikely

← with a standard deviation of (15°). (c) Including learning walks prevents return paths from overshooting the goal. Without a learning walk the simulated ant overshoots and carries on in the direction it was heading as it approached the nest location. By including the views experienced during a learning walk the simulated ant, instead of overshooting, gets repeatedly drawn back to the location of the nest. *Thick grey line*: training path; *thin black lines*: recapitulations

to converge to a particularly good set of independent components, it is enough that the components that are extracted provide a more suitable decomposition of the training data than of an arbitrary input. By decomposing the input in this way it is possible to compress redundant data resulting in more efficient memory storage.

This process achieves robust route navigation, again displaying a corridor within which route navigation is easily achieved (Fig. 4b). The algorithm is able to learn multiple routes to a single goal and can be made more robust by including a number of training runs. Further, by including a set of views that face the goal from a number of different directions, inspired by the learning walk behavior of ants [30], the same algorithm and ANN memory are able to exhibit place search as well as route navigation (Fig. 4c).

6 Discussion and Future Directions

Our aim was to use artificial neural networks (ANNs) as a tool to investigate the connection between the sensory environment and behavior for the problem of visual navigation. It was hoped that through this machine learning approach we might explore the complex sensory array and find regularities and affordances that biological agents might be tuned to. Over a series of implementations we have shown how the visual scenes experienced during a single route traversal can be used to train ANNs to learn a compact representation of the visual information needed to recapitulate that route. Such recapitulation is driven by sampling the world in a range of directions and assessing the familiarity of the associated views as judged by the ANN; one then has to simply move in the most familiar direction. As we have shown, this familiarity-based navigation works robustly in a number of environments.

A more general outcome that emerged from the use of ANNs in this series of experiments was that they allowed us to objectively assess the opportunities for efficient use of vision in a complex task (visual navigation). Our first hope was that objective neural network modelling would allow us to investigate what visual regularities are extracted by ANNs in complex natural worlds. In our first two approaches we were able to see which visual features emerge as useful descriptors of the world for navigation. We suggest this as a new method by which biologists might try to understand the visual preprocessing that is useful for a particular task.

A second major outcome has been a deeper understanding of how memory systems might be efficiently organized for visual navigation. Our algorithms are existence proofs that route memories can be held within a single ANN, which is thus a “holistic” representation of the route. This means that memory load does not scale with the length of the training route and, for instance, multiple or complex paths to a single goal can be stored together. From a

biological perspective it is pleasing that recent research suggests that a particular brain region in insects (the mushroom body) may be well suited to computations similar to our algorithms (Barbara Webb, personal communication, June 10, 2014). Another property of these “holistic” networks is that at no point does the agent have a spatial memory in terms of knowing “where it is” but rather it has a spatial memory for “what it should do.” This has often been noted of insect navigators [15, 31].

Overall, we hope to have outlined in this chapter how neural networks, and machine learning techniques more generally, are not restricted to being used as specific models of brain areas, but can be used to provide a deeper understanding of animal behavior.

References

- Jékely G, Colombelli J, Hausen H et al (2008) Mechanism of phototaxis in marine zooplankton. *Nature* 456(7220):395–399
- von Uexküll J (1931) *Der Organismus und die Umwelt*. In Driesch H, Woltereck H. (Eds.), *Das Lebensproblem im Lichte der modernen Forschung*, Quelle und Meyer, Leipzig, pp 189–224
- Nagel T (1974) What is it like to be a bat? *Philos Rev* 83:435–450
- Shettleworth SJ (2010) Clever animals and kill-joy explanations in comparative psychology. *Trends Cogn Sci* 14(11):477–481. doi:10.1016/j.tics.2010.07.002
- Wehner R (2009) The architecture of the desert ant’s navigational toolkit (Hymenoptera: Formicidae). *Myrmecological News* 12:85–96
- Hölldobler B (1990) *The ants*. Harvard University Press, Cambridge, MA
- Collett TS, Graham P, Harris RA et al (2006) Navigational memories in ants and bees: Memory retrieval when selecting and following routes. *Adv Study Behav* 36:123–172. doi:10.1016/S0065-3454(06)36003-2
- Zeil J, Hofmann MI, Chahl JS (2003) The catchment areas of panoramic snapshots in outdoor scenes. *J Opt Soc Am A Opt Image Sci Vis* 20:450–469
- Philippides A, Baddeley B, Cheng K et al (2011) How might ants use panoramic views for route navigation? *J Exp Biol* 214(3):445–451. doi: 10.1242/Jeb.046755
- Wystrach A, Philippides A, Aurejac A et al (2014) Visual scanning behaviours and their role in the navigation of the Australian desert ant *Melophorus bagoti*. *J Comp Physiol A*:1–12
- Mangan M, Webb B (2012) Spontaneous formation of multiple routes in individual desert ants (*Cataglyphis velox*). *Behav Ecol* 23(5): 944–954. doi: 10.1093/beheco/ars051
- Wystrach A, Beugnon G, Cheng K (2012) Ants might use different view-matching strategies on and off the route. *J Exp Biol* 215(1):44–55. doi:10.1242/Jeb.059584
- Collett M (2010) How desert ants use a visual landmark for guidance along a habitual route. *Proc Natl Acad Sci U S A* 107(25):11638–11643. doi:10.1073/pnas.1001401107
- Graham P, Cheng K (2009) Ants use the panoramic skyline as a visual cue during navigation. *Curr Biol* 19(20):R935–R937
- Wystrach A, Graham P (2012) What can we learn from studies of insect navigation? *Anim Behav* 84(1):13–20. doi:10.1016/j.anbehav.2012.04.017
- Baddeley B, Graham P, Philippides A et al (2011) Holistic visual encoding of ant-like routes: navigation without waypoints. *Adapt Behav* 19(1):3–15. doi:10.1177/1059712310395410
- Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: *Computer vision and pattern recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, 2001. IEEE*, vol 511. pp I-511–I-518
- Freund Y, Schapire R, Abe N (1999) A short introduction to boosting. *J Jpn Soc Artif Intell* 14(771–780):1612
- Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. *Science* 313(5786): 504–507
- Smolensky P (1986) Information processing in dynamical systems: foundations of harmony theory. In Rumelhart D, McClelland J, the PDP Research Group (Eds.). *Parallel distributed*

- processing: Explorations in the microstructure of cognition. Vol. 1: Foundations. MIT Press, Cambridge, MA, pp 194–281
21. Hinton GE (2002) Training products of experts by minimizing contrastive divergence. *Neural Comput* 14(8):1771–1800
 22. Ackley DH, Hinton GE, Sejnowski TJ (1985) A learning algorithm for Boltzmann machines. *Cognit Sci* 9(1):147–169
 23. Hinton GE, Sejnowski TJ (1986) Learning and relearning in Boltzmann machines. MIT Press, Cambridge, MA, 1 (282–317):4.2
 24. Hinton G (2010) A practical guide to training restricted Boltzmann machines. *Momentum* 9(1):926
 25. Baddeley B, Graham P, Philippides A et al (2011) Models of visually guided routes in ants: embodiment simplifies route acquisition. *Intelligent robotics and applications*. Springer, New York, pp 75–84
 26. Lulham A, Bogacz R, Vogt S et al (2011) An infomax algorithm can perform both familiarity discrimination and feature extraction in a single network. *Neural Comput* 23(4):909–926
 27. Bell AJ, Sejnowski TJ (1995) An information-maximization approach to blind separation and blind deconvolution. *Neural Comput* 7(6):1129–1159
 28. Lee T-W, Girolami M, Sejnowski TJ (1999) Independent component analysis using an extended infomax algorithm for mixed sub-gaussian and supergaussian sources. *Neural Comput* 11(2):417–441
 29. Amari S-I, Cichocki A, Yang HH (1996) A new learning algorithm for blind signal separation. *Adv Neural Inform Process Syst* 8:757–763
 30. Muller M, Wehner R (2010) Path integration provides a scaffold for landmark learning in desert ants. *Curr Biol* 20(15):1368–1371. doi:[10.1016/j.cub.2010.06.035](https://doi.org/10.1016/j.cub.2010.06.035)
 31. Wehner R, Wehner S (1990) Insect navigation—use of maps or ariadnes thread. *Ethol Ecol Evol* 2(1):27–48

Chapter 15

Jump Neural Network for Real-Time Prediction of Glucose Concentration

Chiara Zecchin, Andrea Facchinetti, Giovanni Sparacino, and Claudio Cobelli

Abstract

Prediction of the future value of a variable is of central importance in a wide variety of fields, including economy and finance, meteorology, informatics, and, last but not least important, medicine. For example, in the therapy of Type 1 Diabetes (T1D), in which, for patient safety, glucose concentration in the blood should be maintained in a defined normoglycemic range, the ability to forecast glucose concentration in the short-term (with a prediction horizon of around 30 min) might be sufficient to reduce the incidence of hypoglycemic and hyperglycemic events. Neural Network (NN) approaches are suitable for prediction purposes because of their ability to model nonlinear dynamics and handle in their inputs signals coming from different domains. In this chapter we illustrate the design of a jump NN glucose prediction algorithm that exploits past glucose concentration data, measured in real-time by a minimally invasive continuous glucose monitoring (CGM) sensor, and information on ingested carbohydrates, supplied by the patient himself or herself. The methodology is assessed by tuning the NN on data of ten T1D individuals and then testing it on a dataset of ten different subjects. Results with a prediction horizon of 30 min show that prediction of glucose concentration in T1D via NN is feasible and sufficiently accurate. The average time anticipation obtained is compatible with the generation of preventive hypoglycemic and hyperglycemic alerts and the improvement of artificial pancreas performance.

Key words Type 1 diabetes, Forecast, Continuous glucose monitoring, Nonlinear modeling, Time series modeling

1 Introduction

Prediction of the future value of a variable is of central importance and widely applicable within a variety of disciplines. For example, as reported in ref. 1, economic forecasting is used in financial management, for setting fiscal policies and budgeting, and for business planning. A variety of applications exist in both long- and short-term weather forecasting, for example in the generation of civil danger warnings, decision support in agriculture, planning of electricity demand, control and safety of air traffic. In computer science,

forecasting the behavior of Internet networks is used to optimize resources allocation and detect security attacks.

In medicine, prediction methods are largely used for tuning therapies and for forecasting risks of development and progression of diseases, *see* for example ref. [2] for a review of advances in genetics for predicting the occurrence of some diseases. In subjects affected by chronic pathologies, highly risky events could be forecasted by exploiting short-time prediction methods, with 20–30 min of anticipation, *see* [3] for an application in epilepsy for impending seizure detection. In Type 1 Diabetes (T1D) management, which is the field of application considered in the present chapter, real-time short-term prediction of future glucose concentration in the blood from its past history measured by minimally invasive subcutaneous continuous glucose monitoring (CGM) sensors [4, 5] could allow diabetic patients to administer treatment and adjust their therapy on the basis of future, instead of current, glucose concentration, permitting them to mitigate, and in some cases avoid, critical events, *see* [6, 7] for methodological review aspects, [8, 9] for clinical applications and [10, 11] for industrial challenges. In addition, CGM and short-term prediction are key inputs to the so-called artificial pancreas, a minimally invasive pump device which subcutaneously administers insulin according to a temporal profile determined in real-time by a sophisticated closed-loop control algorithm, *see* for example refs. [12, 13].

In this chapter we present, in detail, the recent glucose prediction algorithm of ref. 14, which is based on a jump NN model that uses, as inputs, not only the information on glucose concentration history measured by the CGM sensor but also the quantity of ingested CHO provided by the patient in concomitance with the meal. Parameters of the NN are first tuned on data of ten T1D subjects, while the effectiveness of the prediction method is tested on ten different T1D subjects monitored for 2–3 consecutive days by a widely used commercial CGM sensor (Dexcom SEVEN® PLUS, Dexcom Inc., San Diego, CA).

2 The Diabetes Disease and Its Therapy

Diabetes mellitus is characterized by dysfunctions in insulin secretion and action: in T1D the pancreas is unable to produce insulin, while in Type 2 diabetes derangements in insulin secretion and action occur. As a consequence, the glucose concentration in the blood, which in a healthy individual remains within the normal euglycemic range of 70–180 mg/dl, often exceeds these limits, with short- and long-term complications. Hypoglycemia (glycemia below 70 mg/dl) can progress from measurable cognition impairment to aberrant behavior, seizure and coma [15]. Hyperglycemia (glycemia above 180 mg/dl), if left untreated, can become severe

and lead to serious complications requiring emergency care, such as diabetic coma. Moreover, prolonged hyperglycemia predisposes, in the long term, to invalidating pathologies, as neuropathy, nephropathy, retinopathy, and diabetic foot ulcers [16].

Conventional T1D therapy aims at maintaining euglycemia by adjusting diet, physical activity, and insulin injections. The therapy is normally tuned on the basis of 3–4 daily fingerstick self-monitoring blood glucose (SMBG) measurements, obtained by the patient through portable lancing devices [17]. The recent development of portable minimally invasive CGM sensors, able to measure glucose concentration every 5–10 min for up to 7–10 days, allows the tracking of glucose dynamics much more effectively than via SMBG. It is today largely accepted in clinical research that CGM sensors permit the improvement of diabetes management [18–20], both by suggesting the refinement of the patient’s individual therapy on the basis of the retrospective (Holter-like) assessment of glycemc recordings and in real-time, by alerting the patient when hypoglycemic and hyperglycemic thresholds are exceeded. However, taking therapeutic decisions on the basis of the current glycemia does not allow the patient to avoid imminent critical events.

To help readers of this book to better understand this issue, in Fig. 1 the time-course of glucose concentration (black dots linearly interpolated) measured during the day of a T1D subject by a CGM sensor, together with information on timing and quantity of ingested carbohydrates (CHO, blue stems) and timing and quantity of injected insulin bolus (green stems) are shown. Hypoglycemic and hyperglycemic thresholds are also reported (thin horizontal lines). At the end of the first night glucose concentration was in the euglycemic range, but fell below 70 mg/dl around time 07:30. The subject had breakfast around 08:00 but did not inject any insulin bolus in concomitance with his meal. During the morning

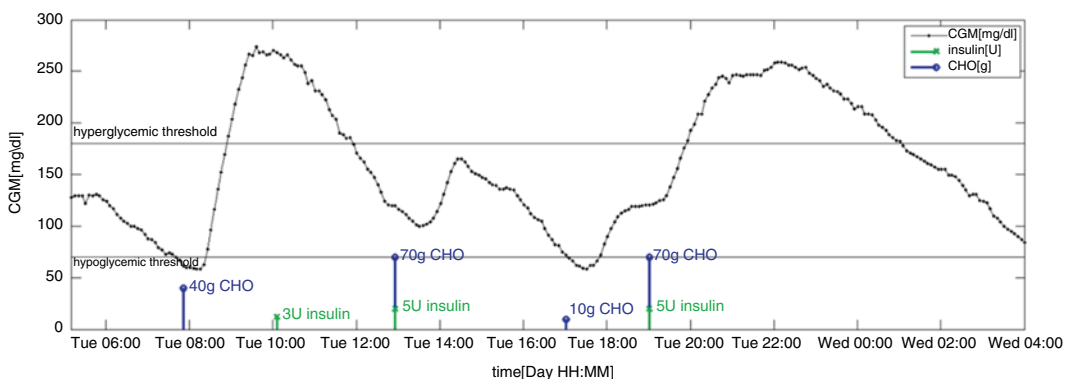


Fig. 1 Representative CGM signal (*black dots* linearly interpolated) measured by the Dexcom SEVEN PLUS device and information on insulin doses (*green stems*) and meals (*blue stems*) of a T1D subject

glucose concentration reached hyperglycemic values and the subject injected a correction bolus of insulin around time 10:00 and reentered the euglycemic range around time 12:00. At time 13:00 and 19:00 the subject ate and injected insulin to counterbalance the effects of CHO. Notably, around time 17:00 the CGM signal fell in the hypoglycemic range and the subject promptly ingested 10 g of sugar to increase the glucose concentration and reenter the safe range. After dinner, around time 20:00, the subject experienced another hyperglycemia and reentered the safe range only after time 01:00. Some of (or even all of) these critical events could have been avoided, in theory, if the alerts were generated about 20 min before the crossing of the 70 mg/dl threshold, e.g., on the basis of a predicted profile of future glucose concentration. In view of the availability of CGM sensors from the beginning of the twenty-first century this possibility stimulated research and development of glucose prediction methods [4, 7]. However, forecasting glucose concentration in a certain prediction horizon (PH) is a challenging topic. Indeed, glycemia is influenced by many (often not measurable/quantifiable) factors (e.g., CHO ingestion, physical activity, administration of drugs including insulin, stress, and emotions) and interindividual and intraindividual variability is high. Furthermore, information relative to several key variables that influence glucose dynamics is not directly/easily usable. For instance, glucose concentration is measured by the CGM sensor and is returned as a time series, thus it is directly available and easy to embed in formal mathematical models. On the other hand, information on timing and CHO content of meals and on timing and dose of injected insulin is impulsive and should be adequately preprocessed, before being used, to generate signals more informative of meal and insulin effects on glucose concentration. Finally, information on factors such as stress or emotions is hard to quantify and highly subjective. For these reasons, the majority of published glucose prediction methods rely solely on the use of the CGM signal as input.

3 Glucose Prediction: A Brief State of the Art

Popular algorithms, which exploit CGM information only, include autoregressive (AR) and AR with moving average (ARMA) models. Some examples are [21–24]. Some attempts to exploit information on CHO and insulin therapy by ARX and ARMAX models have been recently proposed, *see* for example ref. [25–28]. However, it is natural to expect that glucose prediction performed exclusively on the basis of CGM data, possibly incorporating additional information in simple linear models, cannot be as effective as if additional input signals were used in nonlinear models.

Neural networks (NN), thanks to their ability to learn the behavior of empirical nonlinear models and to utilize heterogeneous signals among their inputs, are valuable models for forecasting glycemia using collateral information, in addition to the CGM signal. In the literature, only a few papers report investigations of the use of NN algorithms for glucose prediction. In ref. 29 the authors proposed a feed-forward NN whose inputs were previous CGM samples and the current time instant. In ref. 30 Pappada and colleagues developed a feed-forward NN incorporating, in addition to CGM data, other inputs such as SMBG readings, information on insulin and CHO, information on hypoglycemic and hyperglycemic symptoms, lifestyle, activity, and emotions. In ref. 31 Daskalaki and colleagues compared an ARX and a NN model exploiting, respectively, CGM and insulin and CGM, insulin and CHO information. In ref. 32 our research group proposed a model based on the parallel of a first order polynomial algorithm and a feed-forward NN, using information on CGM and announcement of future ingested CHO. In ref. 14 we demonstrated that a simpler architecture, without any announcement of future ingestion of CHO, gives results comparable to those of ref. 32.

4 A Jump Neural Network Methodology for Glucose Prediction

A jump NN is a feed-forward NN with inputs connected not only to the first hidden layer but also to the output layer. According to ref. 33, the jump NN structure is particularly suitable, and better than a simple feed-forward NN, for fitting and predicting time series characterized by the presence of both linear and nonlinear dynamics, as in the case of glucose signals, where we assume that each input is able to influence future blood glucose concentration with both linear and nonlinear effects. Indeed, hidden neurons, with their nonlinear activation functions, model the nonlinear relationship between inputs and targets, while the output neurons, with their linear activation functions, learn the linear relationship between inputs and targets.

To explain the adopted methodology, Fig. 2 shows the architecture of the proposed jump NN. Note that, in our implementation, data have a sampling period T_s of 5 min and the PH is 30 min, corresponding to $N=6$ steps ahead prediction.

4.1 Inputs Selection and Preprocessing

Inputs were selected through a multiple consecutive steps procedure, using data drawn from the training set.

1. First, a pool of possible candidate input signals was chosen, based on a priori physiological knowledge of the glucose insulin system and of diabetes. Going into details, the history of the glucose concentration signal, measured by the CGM sensor,

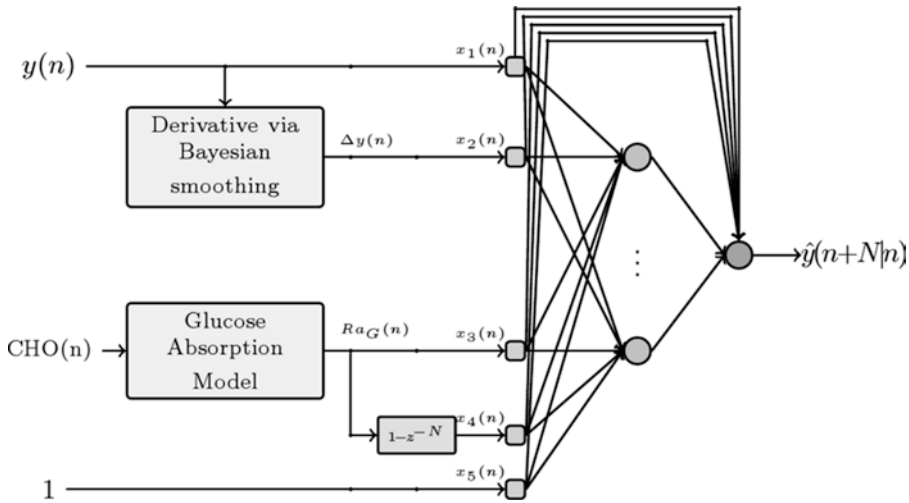


Fig. 2 Block scheme of the proposed jump NN model for glucose prediction. z^{-k} indicates the k -steps backward shift operator

was chosen since glucose concentration is a slow-varying signal, thus its future values are correlated, in the short term, with the past time-course of the signal. Moreover, information on quantity and timing of ingested CHO, which is assumed to be provided by the patient himself/herself, was used. Ingestion of sugar is known to increase glucose concentration; however, while the ingestion of a meal can recall an impulsive process, CHO effects on glycemia are known to be smoother in both appearance and disappearance. For this reason, the physiological model of oral glucose absorption proposed in ref. 34, completed with the population parameters obtained in ref. 35, was used to obtain the glucose rate of appearance (Ra_G) in the blood, a signal that mimics the velocity of glucose absorption in the blood stream after the ingestion of a CHO load.

2. As second step, various signals, related to CGM history and to Ra_G were generated. The glucose concentration time derivative was computed from the CGM signal, using a Bayesian smoothing approach [36] to deal with ill-conditioning which tends to amplify measurement noise. Ra_G first order difference was computed, with time steps of 5, 15, and 30 min, and Ra_G cumulative sum in the preceding 30, 60, 90, and 120 min was also computed.
3. Afterwards, the cross-correlation between target glucose concentration and the signals generated at **step 2** was computed, for various time shifts. This step allows one to obtain two major results: the signals with the highest correlation with the target are selected, which are likely to be more useful to predict the target itself; time delays that should be applied to these signals are also selected.

4. Finally, k -fold cross-validation was used to pick out the best subset of inputs, among those selected with the cross-correlation analysis. In this step, all the possible input combinations are considered and, for each one, a NN is optimized. In particular, in k -fold cross-validation the training set is divided into k subsets, each network is trained on $k-1$ subsets and tested on the remaining subset and the procedure is repeated k times, leaving out each time a different subset for testing the models. The performance of each NN is the average obtained in the k experiments.

The above four steps provided the data necessary to select four input signals. Two of them are related with the CGM signal and are the current value of glucose concentration, measured by the CGM sensor, indicated as $y(n)$ in Fig. 2 and its current time derivative, which will be indicated as $\Delta y(n)$ in the rest of the chapter. The other two inputs are relative to information on ingested CHO and are the current value of Ra_G and its average trend in the last 30 min, computed as

$$\frac{1}{N}(1-z^{-N})Ra_G(n) = \frac{1}{N}(Ra_G(n) - Ra_G(n-N)) \quad (1)$$

with z^{-k} indicating the k time steps backward shift operator, n current time instant, and N equal to six steps.

Several preprocessing techniques are commonly applied before the data are used for training the NN to accelerate convergence and to ease the problem to be learned [37]. An essential operation is a rough scaling of the data between the lower and upper bounds of the neuron transfer function. We normalized inputs and output so that they had zero mean and standard deviation equal to 1. This step guarantees that, at the beginning of the training procedure, all the signals have, potentially, the same importance and they all belong to the approximately linear range of the tangent sigmoidal activation function. Data normalization also limits the extent to which larger numbers may dominate smaller ones and avoids premature saturation of hidden nodes, which would degrade the learning process.

4.2 Mathematical Representation of the Jump NN Model

Let us define: N_{in} number of input signals and N_{hn} number of hidden neurons. The jump NN predicts a signal that may be expressed, in a vector matrix form, as

$$\hat{y}(n+N | n) = \Omega \times X(n) + \Psi \times \Phi(\Gamma \times X(n)) \quad (2)$$

where $X(n)$ indicates the column vector of size $[N_{in}+1]$ of input signals at time step n , plus an input equal to 1 associated with the weight accounting for the bias:

$$X(n) = \left[1, y(n), \Delta y(n), Ra_G(n), \frac{1}{N} (1 - z^{-N}) Ra_G(n) \right]^T \quad (3)$$

Ω is the $[N_{in} + 1]$ row vector of weights connecting every input directly with the output neuron, thus $\Omega_i = \omega_i$ indicates the weight connecting the i th input to the output neuron. Ψ is the row vector of size N_{hn} of weights connecting every hidden neuron to the output neuron, thus $\Psi_k = \psi_k$ is the weight connecting the k th hidden neuron to the output neuron. Γ is the $[N_{hn} \times N_{in}]$ matrix of weights connecting every input to every hidden neuron, thus $\Gamma_{ji} = \gamma_{ji}$ indicates the weight connecting the i th input to the j th hidden neuron. Equation 2 can be expressed explicitly as

$$\hat{y}(n + N | n) = \sum_{i=0}^{N_{in}} \omega_i x_i(n) + \sum_{j=1}^{N_{hn}} \psi_j \varphi \left(\sum_{i=0}^{N_{in}} \lambda_{ji} x_i(n) \right) \quad (4)$$

4.3 Structure Optimization

The number of hidden neurons and hidden layers was chosen with k -fold cross-validation on the training set. Several candidate NNs with one hidden layer and with an increasing number on hidden neurons were evaluated. NNs with two hidden layers were also assessed, but they did not significantly outperform NN architectures with one hidden layer only. Note that the selected model is usually a compromise between performance and structure simplicity. Indeed the NN with the lowest possible number of neurons that significantly outperforms simpler structures is chosen. In our case, the best and most parsimonious architecture was a NN with one hidden layer with five neurons. Thus, since the number of input signals is equal to 4 and there is 1 output, there are 31 free parameters to tune during training.

4.4 NN Training Procedure

The NN weights were optimized on the training and validation set (*see* Subheading 5.1 for details) and then the architecture was tested on an independent test set. The NN was trained with the backpropagation Levenberg–Marquardt training algorithm, applied in batch mode, i.e., weights and biases are only updated after all the inputs and targets are presented to the NN, thus on the basis of the average error obtained on the entire training set. The minimized objective function J was the mean square error:

$$J = \frac{1}{N_{TS}} \sum_{i=1}^{N_{TS}} (\hat{y}_i - y_i)^T (\hat{y}_i - y_i) = \frac{1}{N_{TS}} \sum_{i=1}^{N_{TS}} \frac{1}{M_i} \sum_{k=1}^{M_i} (\hat{y}_i(k + N | k) - y_i(k))^2 \quad (5)$$

with N_{TS} the number of training time series and M_i the length of the i th training time series. The training procedure was arrested using early stopping [38]. Ordinarily, a NN learns in stages, increasing the performance in the training set as the training session progresses, towards a local minimum of the error surface.

However, the NN might end up overfitting the training data and generalizing poorly. The onset of overfitting can be identified using cross-validation: the training data are split into an effective training set, used for computing the error and its gradient and updating the network weights, and a validation set, used for monitoring the error during training. The training session is stopped periodically and the error on the validation set is computed. The validation error normally decreases during the initial phase of training; however, when the network begins to overfit the data, the error on the validation set begins to rise. When the error increases for a predefined number of consecutive iterations the training is stopped and the weights at the minimum of the validation error are returned. In our implementation, we stopped the training after 100 consecutive iterations worsened NN performance on the validation set.

Remark: To avoid overfitting, a training procedure alternative to cross-validation is Bayesian regularization. This technique updates the weight and bias values according to Levenberg–Marquardt optimization, minimizing a combination of squared errors and squared weight values so that, at the end of training, the resulting network has good generalization without early stopping being required. In addition, any redundant weights in the network should assume values close or equal to zero at the end of the training and should, potentially, be eliminated from the NN without compromising its performance. In our implementation, this training procedure gave results comparable to those obtained with cross-validation; however, it was considerably more time-consuming, so we adopted the classical backpropagation with the cross-validation algorithm. A positive feature of Bayesian regularization is that it does not require any validation set, thus all the data, other than those belonging to the test set, can be used for training. For this reason the method might be preferable to cross-validation when the available database is of small size.

5 Database and Assessment Criteria

5.1 Database

The NN was optimized and tested on data of 20 T1D patients, monitored for 2 or 3 consecutive days in real-life conditions. Data were collected during the DIAdvisor™ project [39]. The glucose concentration was measured by the Dexcom SEVEN PLUS CGM sensor, which has a sampling time T_s of 5 min. Information on ingested CHO was extracted by picture of meals taken by the patients with a standard mobile phone with camera.

The database was divided into a training and validation set, constituted by ten time series and a test set, formed by the other ten time series. The training and validation set was further randomly divided into the training set constituted by 70 % of the data,

used for minimizing the prediction error and the validation set constituted by the remaining 30 % of data, used for stopping the training procedure.

5.2 Assessment Metrics

The quality of prediction obtained with the proposed jump NN algorithm is quantitatively assessed by computing three metrics commonly used in the glucose prediction literature. The considered indexes, which measure different merits of the predicted glucose profile, are:

1. The Root Mean Square Error (RMSE), (mg/dl) between the predicted time-series and the original glucose time-series measured by the CGM sensor, calculated as

$$RMSE = \sqrt{\frac{1}{M} \sum_{i=1}^M \left(\hat{y}(i + N | i) - y(i) \right)^2} \quad (6)$$

with M being the length of the time series.

2. The average Time Gain (TG), (min)

$$TG = PH - \text{delay} \quad (7)$$

with the delay quantified as the temporal shift that minimizes the distance between $\hat{y}(n + N | n)$ and $y(n)$

$$\text{delay} = \underset{k \in [0, N]}{\operatorname{argmin}} \left\{ \frac{1}{M - N} \sum_{i=1}^{M-N} \left(\hat{y}(i + k | i + k - N) - y(i) \right)^2 \right\} T_s \quad (8)$$

with T_s being the sampling period of the signal.

3. The Energy of Second Order Differences (ESOD) (i.e., the sum of the squared second order differences) of the predicted time series, normalized by the ESOD of the target time series [23].

$$ESOD_{\text{norm}} = \frac{ESOD(\hat{y})}{ESOD(y)} \quad (9)$$

where

$$ESOD(\hat{y}) = \sum_{i=3}^M \left(\hat{y}(i | i - N) - 2\hat{y}(i - 1 | i - 1 - N) + \hat{y}(i - 2 | i - 2 - N) \right)^2 \quad (10)$$

and

$$ESOD(y) = \sum_{i=1}^M \left(y(i) - 2y(i - 1) + y(i - 2) \right)^2 \quad (11)$$

The RMSE is a widely used metric in the CGM literature [22–25, 40]; however, it has some limitations: it does not

penalize spurious oscillations around the target and it is unable to penalize differently underestimation and overestimation of the target. TG is one of the most important indices from a practical perspective, since it quantifies the average anticipation with which events could be, in theory, detected and thus have a clinical value. The closer the TG is to the PH, the better the prediction, since the patient could decide therapeutic actions ahead in time and, likely, avoid critical events. Notably, the definition of the delay given in Eq. 8 is consistent with those of ref. 22, 41. $ESOD_{norm}$ reflects how (possibly spurious) oscillations are amplified in the predicted time series. Thus it roughly quantifies the risk of generating false hypo/hyper alerts. The closer to 1, the better the predicted time series.

6 Results

Figure 3 shows prediction in one typical subject (blue line), together with the target CGM signal (black) and meal information provided by the patient (blue stems). The thin horizontal lines represent hypoglycemic and hyperglycemic thresholds.

The predicted signal is plotted at the time instant to which it refers, i.e., the value plotted at a certain time is obtained N time steps earlier, using only data available until N time steps earlier. As we can note, the jump NN predicts the target accurately ($RMSE = 17.6$ mg/dl), with a satisfactory average anticipation ($TG = 15$ min) and with an acceptable amplification of measurement noise ($ESOD_{norm} = 9.9$). Results on the other test subjects are in line with those of the representative subject of Fig. 3 and are summarized in Table 1. $RMSE$, TG , and $ESOD_{norm}$ values are satisfactory, suggesting that the jump NN is an appropriate architecture for glucose concentration prediction and is able to learn the

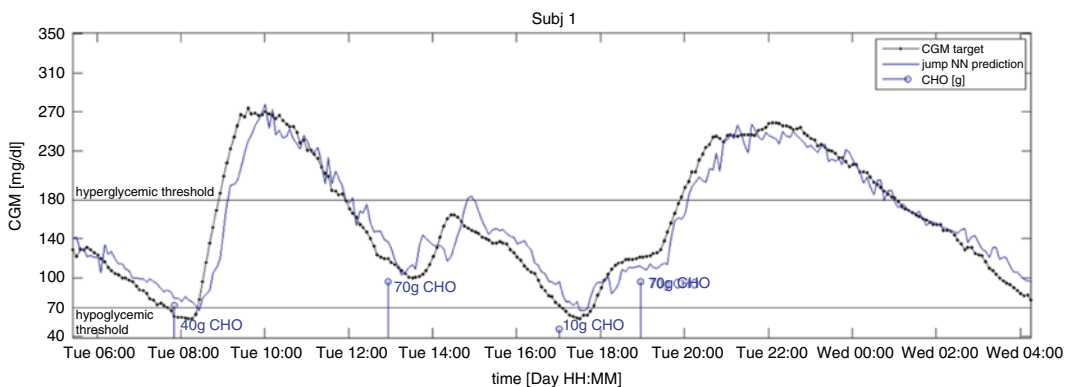


Fig. 3 CGM profile (black) and prediction obtained with the proposed jump NN (blue line). Stems indicate CHO ingestion, horizontal thin lines represent the hypoglycemic and the hyperglycemic threshold

Table 1
Results obtained on the ten test subjects (with PH = 30 min), average values and standard deviation

	RMSE (mg/dl)	TG (min)	ESOD _{norm} (-)
Subj 1	17.6	15	9.9
Subj 2	20.3	15	10.0
Subj 3	13.1	20	8.7
Subj 4	12.0	25	7.4
Subj 5	15.6	20	8.5
Subj 6	15.9	20	12.6
Subj 7	13.7	20	9.4
Subj 8	17.8	15	8.1
Subj 9	18.2	20	12.0
Subj 10	21.3	15	9.8
Mean ± sd	16.6 ± 3.1	18.5 ± 3.4	9.6 ± 1.6

The lower the RMSE, the closer to 30 min the TG, the closer to one ESOD_{norm}, the better the quality of the predicted profiles

relationship between the history of CGM and of CHO ingestion and future glucose concentration accurately. Moreover, the average TG is approximately 20 min, which is sufficient for taking effective therapeutic decisions ahead in time.

Another interesting index is the prediction time anticipation in correspondence of hypoglycemic and hyperglycemic thresholds. In Fig. 3, for example, the first hypoglycemic event is not predicted, while the second is anticipated by 5 min. Regarding hyperglycemia, the first event is anticipated by 15 min and the second by 20 min. In fact, in the dataset, the number of hypoglycemic events is limited (12 in the 10 time series used as the training set and 18 in the 10 time series used as the test set). As a consequence, the NN performance in forecasting these events cannot be solidly assessed. A preliminary visual analysis, restricted to the hypoglycemic events that the model is effectively able to predict, shows an anticipation around 20 min in crossing the 70 mg/dl threshold. However, some challenging events were not predicted. On the contrary, several hyperglycemic events are present in the dataset (43 in the 10 time series used as the training set and 58 in the 10 time series used as the test set). This allows the NN algorithm to accurately learn them. Indeed 94 % of the hyperglycemic events are correctly forecasted with an average anticipation of 23 min in crossing the 180 mg/dl threshold. This result is highly encouraging and suggests that the NN predictor, when trained on enough representative samples, can be used to anticipate critical events.

7 Conclusions

Short-time prediction of glucose concentration can improve T1D therapy and management; however, it is challenging since glucose time course is affected by many exogenous disturbances (e.g., ingestion of CHO, injection of insulin, physical activity) and inter-individual and intraindividual variability. The great majority of the tens of prediction methods proposed in the literature in the last decade are based on time series models and rarely exploit information other than CGM. NN-based algorithms are able to learn nonlinear input–output relationships, and can easily use inputs belonging to different domains, as information on meals, insulin therapy and physical activity; thus they appear suitable candidate models for predicting future glycemia for both open and closed loop control applications.

This chapter describes a new method based on a jump NN, with inputs directly connected both to the first hidden layer and to the output neuron. Such a NN architecture is appropriate when inputs and output to be learned present both linear and nonlinear relationships. Information on timing and amount of CHO in the ingested meal was suitably preprocessed, before entering the NN, by exploiting a physiological model. The NN was optimized on data of ten T1D and tested on data of ten different T1D, monitored with a minimally invasive CGM sensor. Results demonstrate that this approach is able to accurately predict glucose concentration, with an average temporal gain compatible with the generation of preventive hypoglycemic and hyperglycemic alerts.

Future work could include the investigation of other possible inputs for the NN, such as insulin information and physical activity related signals, whose correlation with changes in glucose dynamics has recently been quantitatively investigated [42, 43]. Moreover, the NN training procedure could be customized to minimize an objective function that takes into account, apart from the adherence to the target, also the time anticipation of prediction, possibly penalizing differently errors in hypoglycemia, euglycemia, and hyperglycemia.

To conclude, it is worthwhile noting that even if the presented NN is specifically designed for glucose prediction, the methodology could easily be adapted to the prediction of any other time series whose dynamics are influenced both linearly and nonlinearly by exogenous measurable signals.

References

1. De Gooijer J, Hyndman R (2006) 25 years of time series forecasting. *Int J Forecasting* 22(3):443–473
2. Kruppa J, Ziegler A, Knig I (2012) Risk estimation and risk prediction using machine learning methods. *Hum Genet* 131(7):1–16

3. Iasemidis L (2011) Seizure prediction and its applications. *Neurosurg Clin N Am* 22(4): 489–513
4. Bequette BW (2010) Continuous glucose monitoring: real-time algorithms for calibration, filtering, and alarms. *J Diabetes Sci Technol* 4(2):404
5. Sparacino G, Zanon M, Facchinetti A et al (2012) Italian contributions to the development of continuous glucose monitoring sensors for diabetes management. *Sensors* 12(10):13753–13780
6. Sparacino G, Facchinetti A, Maran A et al (2008) Continuous glucose monitoring time series and hypo/hyperglycemia prevention: requirements, methods, open problems. *Cur Diabetes Rev* 4(3):181–192
7. Sparacino G, Facchinetti A, Cobelli C (2010) “Smart” continuous glucose monitoring sensors: on-line signal processing issues. *Sensors* 10(7):6751–6772
8. Buckingham B, Cobry E, Clinton P et al (2009) Preventing hypoglycemia using predictive alarm algorithms and insulin pump suspension. *Diabetes Technol Ther* 11(2):93–97
9. Buckingham B, Chase HP, Dassau E et al (2010) Prevention of nocturnal hypoglycemia using predictive alarm algorithms and insulin pump suspension. *Diabetes Care* 33(5):1013–1017
10. Bode B, Gross K, Rikalo N et al (2004) Alarms based on real-time sensor glucose values alert patients to hypo- and hyperglycemia: the guardian continuous monitoring system. *Diabetes Technol Ther* 6(2):105–113
11. Garcia A, Rack-Gomer AL, Bhavaraju NC et al (2012) Dexcom G4AP: an advanced continuous glucose monitor for the artificial pancreas. *J Diabetes Sci Technol* 7(6):1436–1445
12. Cobelli C, Renard E, Kovatchev B (2011) Artificial pancreas: past, present, future. *Diabetes* 60(11):2672–2682
13. Thabit H, Hovorka R (2012) Closed-loop insulin delivery in type 1 diabetes. *Endocrinol Metab Clin North Am* 41(1):105–117
14. Zecchin C, Facchinetti A, Sparacino G et al (2014) Jump neural network for online short-time prediction of blood glucose from continuous monitoring sensors and meal information. *Comput Methods Programs Biomed* 113(1):144–152
15. Cryer PE (2007) Hypoglycemia, functional brain failure, and brain death. *J Clin Invest* 117(4):868–870
16. Williams G, John CP (2004) *Handbook of diabetes*. Blackwell, Oxford
17. The American Diabetes Association (2013) Standards of medical care in diabetes: 2013. *Diabetes Care* 36(S1):S11–S66
18. Tamborlane WV, Beck RW, Bode BW (2008) Continuous glucose monitoring and intensive treatment of type 1 diabetes. *N Engl J Med* 359(10):1464–1476
19. Battelino T, Phillip M, Bratina N et al (2011) Effect of continuous glucose monitoring on hypoglycemia in type 1 diabetes. *Diabetes Care* 34(4):795–800
20. Deiss D, Bolinder J, Riveline JP et al (2006) Improved glycemic control in poorly controlled patients with type 1 diabetes using real-time continuous glucose monitoring. *Diabetes Care* 29(12):2730–2732
21. Reifman J, Rajaraman S, Gribok A et al (2007) Predictive monitoring for improved management of glucose levels. *J Diabetes Sci Technol* 1(4):478–486
22. Gani A, Gribok AV, Rajaraman J et al (2009) Predicting subcutaneous glucose concentration in humans: data-driven glucose modeling. *IEEE Trans Biomed Eng* 56(2):246–254
23. Sparacino G, Zanderigo F, Corazza S et al (2007) Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series. *IEEE Trans Biomed Eng* 54(5):931–937
24. Eren-Oruklu M, Cinar A, Quinn L et al (2009) Estimation of the future glucose concentrations with subject specific recursive linear models. *Diabetes Technol Ther* 11(4):243–253
25. Finan DA, Doyle FJ, Palerm CC et al (2009) Experimental evaluation of a recursive model identification technique for type 1 diabetes. *J Diabetes Sci Technol* 5(3):1192–1202
26. Castillo-Estrada G, del Re L, Renard E (2010) Nonlinear gain in online prediction of blood glucose profile in type 1 diabetic patients. 49th IEEE Conference on Decision and Control (CDC), p 1668–1673
27. Eren-Oruklu M, Cinar A, Rollins DK et al (2012) Adaptive system identification for estimating future glucose concentrations and hypoglycemia alarms. *Automatica* 48(8):1892–1897
28. Turksoy K, Bayrak ES, Quinn L et al (2013) Hypoglycemia early alarm systems based on multivariable models. *Ind Eng Chem Res* 52:12329–12336
29. Pérez-Gandía C, Facchinetti A, Sparacino G et al (2010) Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring. *Diabetes Technol Ther* 12(1):81–88

30. Pappada SM, Cameron BD, Rosman PM et al (2011) Neural network-based real-time prediction of glucose in patients with insulin-dependent diabetes. *Diabetes Technol Ther* 13(2):135–141
31. Daskalaki E, Prountzou A, Diem P et al (2012) Real-time adaptive models for the personalized prediction of glycemic profile in type 1 diabetes patients. *Diabetes Technol Ther* 14(2): 168–174
32. Zecchin C, Facchinetti A, Sparacino G et al (2012) Neural network incorporating meal information improves accuracy of short-time prediction of glucose concentration. *IEEE Trans Biomed Eng* 59(6):1550–1560
33. McNelis PD (2005) *Neural networks in finance: gaining predictive edge in the market*. Elsevier Academic Press, London
34. Dalla Man C, Camilleri M, Cobelli C (2006) A system model of oral glucose absorption: validation on gold standard data. *IEEE Trans Biomed Eng* 53(12):2472–2478
35. Dalla Man C, Rizza RA, Cobelli C (2007) Meal simulation model of the glucose insulin system. *IEEE Trans Biomed Eng* 54(10): 1740–1749
36. Facchinetti A, Sparacino G, Cobelli C (2011) Online denoising method to handle intraindividual variability of signal-to-noise ratio in continuous glucose monitoring. *IEEE Trans Biomed Eng* 58(9):2664–2671
37. Basheer IA, Hajmeer M (2000) Artificial neural networks: fundamentals, computing, design, and application. *J Microbiol Methods* 43(1):3
38. S. Amari, N. Murata, K.-R. Muller, M. Finke, H. Yang (1996) *Statistical Theory of Overtraining - Is Cross-Validation Asymptotically Effective?*, *Advances in Neural Information Processing Systems* 8, Proceedings of the 1995 Conference, edited by David S. Touretzky, Michael C. Mozer and Michael E. Hasselmo pp 176–182
39. DIAdvisor. Personal glucose predictive diabetes advisor. <http://www.diadvisor.eu/>. Accessed 22 Jan 2014
40. Gani A, Gribok AV, Lu Y et al (2010) Universal glucose models for predicting subcutaneous glucose concentration in humans. *IEEE Trans Inf Technol Biomed* 14(1):157–165
41. Facchinetti A, Sparacino G, Cobelli C (2010) Modeling the error of continuous glucose monitoring sensor data: critical aspects discussed through simulation studies. *J Diabetes Sci Technol* 4(1):4–14
42. Manohar C, Levine JA, Nandy DK et al (2012) The effect of walking on postprandial glycemic excursion in patients with type 1 diabetes and healthy people. *Diabetes Care* 35(12):2493–2499
43. Zecchin C, Facchinetti A, Sparacino G et al (2013) Physical activity measured by physical activity monitoring system correlates with glucose trends reconstructed from continuous glucose monitoring. *Diabetes Technol Ther* 15(10):836–844

Preparation of Ta-O-Based Tunnel Junctions to Obtain Artificial Synapses Based on Memristive Switching

Stefan Niehörster and Andy Thomas

Abstract

Magnetron sputtering and optical lithography are standard techniques to prepare magnetic tunnel junctions with lateral dimensions in the micrometer range. Here we present the materials and techniques to deposit the layer stacks, define the structures, and etch the devices. In the end, we obtain tunnel junction devices exhibiting memristive switching for potential use as artificial synapses.

Key words Memristors, Artificial synapses, Magnetron sputtering, Optical lithography, Tunnel junctions

1 Introduction

Memristors and memristive systems have attracted a lot of interest in recent years. The most interesting implementation of memristive systems is neuromorphic devices. These aim to use biological mechanisms operating within the brain as a prototype to construct new computer architectures [1, 2].

A memristor is a portmanteau of memory and resistor that Chua proposed in 1971 [3]. A straightforward way to envisage a memristor is as an adjustable resistor, e.g., a potentiometer. If a certain amount of flux ($V \times s$) flows in one direction, the resistance increases. If the current flows in the other direction through the device, the resistance decreases. This means that the resistance of a memristor depends on its past states, which can be used to mimic the synaptic functionality in a brain.

In particular, it is possible to demonstrate the equivalents of long-term potentiation (LTP), long-term depression (LTD) as well as spike timing dependent plasticity (STDP) [4–6]. Consequently, the memristors can be used as electronic synapses in circuits, while the integrate-and-fire functionality can easily be achieved via conventional electronics.

In many cases, memristive systems consist of metal and insulator thin films ($\sim 1\text{--}10$ nm) [7]. Tantalum oxide based systems are known to be stable over 10^{10} switching cycles [8–10], but TiO_2 is often used as well [11, 12]. We realized the memristor as a metal/insulator/metal trilayer, in which a thin tantalum oxide layer acts as a tunneling barrier. In the production process, the insulator grows on top of the lower metal electrode, while the upper metal electrode grows on top of the insulator. This asymmetry of the manufacturing process introduces oxygen vacancies in the crystal lattice at the bottom of the insulator. If we apply an electric voltage across the tunnel barrier, the electric field acts on the oxygen vacancies and they change their position slightly, i.e., the oxygen vacancies will be shifted within the tunnel barrier. The resistance changes considerably, because the resistance of a tunnel barrier depends strongly on the properties (e.g., width and height) of the tunneling barrier. The oxygen vacancies remain in the tunnel contact on their new position, which explains the memory effect. If the applied voltage is reversed and the current flows in the opposite direction, the oxygen vacancies also move back and, consequently, change the resistance back to its initial value.

In the following sections, we describe the necessary techniques and procedures to prepare tunnel junction pillars. First, we list the required materials, i.e., the sputter targets, the substrates, and the chemicals for the lithography and etching processes. Then, we go into the details of the substrate preparation, the thin film deposition, the optical lithography, including ion beam etching, and finally the production of the contact pads with insulation layer.

2 Materials

1. The substrates consist of $525\ \mu\text{m}$ thick silicon wafers in $\langle 100 \rangle$ orientation with a 4" diameter. The substrates are thermally oxidized to provide a 50 nm silicon dioxide film. The front surface is polished, while the backside remains etched. The wafer was doped with Boron (holes) and had a resistivity of $10\text{--}20\ \Omega\ \text{cm}$. Wafers with the listed properties can be purchased from, e.g., Semiconductor Wafer, Inc. We cut the wafer into pieces of $10 \times 10\ \text{mm}^2$ for further processing.
2. Films are deposited on the wafer pieces by magnetron sputtering. We utilized a "CLAB600" from "Leybold Optics, Alzenau, Germany" with an additional oxidation chamber. The metal films are prepared via dc- and the insulator films are deposited via rf-sputtering.
3. We used the following sputter targets:
 - (a) Tantalum 3" (4N).
 - (b) Palladium 4" (3N5).
 - (c) Gold 2" (4N).

4. The lithography was performed in a clean room equipped with a spin-coater. We used the positive photoresist and the developer manufactured by “ALLRESIST GmbH, Strausberg, Germany,” namely, “AR-P 5350” (resist) and “AR300-35” (developer). The exposure was carried out by a Hg-short-arc lamp with an emission maximum of 436 nm or by the lithography laser system “DWL66” from “Heidelberg Instruments Mikrotechnik GmbH, Heidelberg, Germany”, using a laser emission maximum of 405 nm.
5. The etching of the junction pillars was completed in an argon ion-etching chamber with a rotatable sample holder. The system was combined with a secondary-ion-mass-spectrometer (SIMS) to detect the depth of the etching process.

3 Methods

3.1 Substrate Preparation

Substrates with a size of $10 \times 10 \text{ mm}^2$ are trimmed out of a 4" silicon wafer. The wafer is scratched at the edge with a diamond scraper and cut into pieces over an edge, e.g., five sheets of paper. The substrate is affixed by a clamp to the sample holder. The substrate surface has to be contacted with silver paint to the sample holder (*see Note 1*). Before loading the sample into the load lock of the sputter machine, we cleaned the sample surface with a jet of nitrogen gas.

3.2 Thin Film Deposition

1. The thin films are deposited by magnetron sputtering. The base pressure inside the sputter chamber is less than 2.0×10^{-7} mbar, with a set point of 2.0×10^{-7} mbar. The sputter pressure is approximately 1.3×10^{-3} mbar and is regulated via the Argon flow (20 sccm) and the throttle position (21 %) in front of the turbo-pump. In our case, the tantalum film is sputtered from a 3" target with a power of 65 W and the palladium film is sputtered from a 4" target with a power of 115 W. The machine specific deposition rates are calibrated by X-ray reflectivity measurements (0.1–0.6 nm/s).
2. We subsequently deposit a stack of 5 nm tantalum, 10 nm palladium, and 2 nm tantalum on the silicon dioxide side of the wafer.
3. The sample is next moved in-situ into the oxidation chamber. The base pressure inside this chamber is approximately 2.5×10^{-7} mbar, with a set point of 7.0×10^{-6} mbar. The pressure during the oxidation process is approximately 2.0×10^{-3} mbar and depends on the oxygen flow (13 sccm) and the throttle position (80 %) in front of the turbo-pump. The microwave power to ionize the oxygen is set to 275 W and the bias voltage to accelerate the oxygen ions towards the sample is -80 V . The sample is oxidized for 150 s to obtain the tantalum oxide tunnel barrier.

- The sample (in-situ) is then moved back into the sputter chamber and layers of 10 nm Ta, 5 nm Pd, 5 nm Ta, and 30 nm Au are deposited. Gold needs a higher sputter pressure of approximately 4.5×10^{-3} mbar, which is realized by a smaller opening of the throttle (8 %). The sputter power for gold is 29 W, because of the smaller target size of 2".

3.3 Optical Lithography

This is carried out as follows:

- Place the sample in the middle of the spin-coater and tape it with double-sided sticky tape. Fill a one-way pipette with photoresist and clean the sample with a jet of nitrogen gas. Drop the photoresist on the sample until the surface is fully covered. Rotate the sample at 4,000 rpm (or 6,000 rpm for the laser lithography) and subsequently bake it at 80 °C for approximately 30 min.
- Place a mask with an adequate layout on the sample and irradiate both with the UV-light (Fig. 1). For positive resist cover the junction pillars. Alternatively, use the DWL-66 laser lithography to create the desired layout. Our layout consists of squares with sizes of $7.5 \times 7.5 \mu\text{m}^2$, $12.5 \times 12.5 \mu\text{m}^2$, and $22.5 \times 22.5 \mu\text{m}^2$.
- Mix the photo developer with pure water at a ratio of 2 to 1. Stir the sample in the mixture for about 8 s until the vapor dissolves (*see Note 2*), then stop the developer process and clean it with pure water. Afterwards, dry the sample with nitrogen gas.

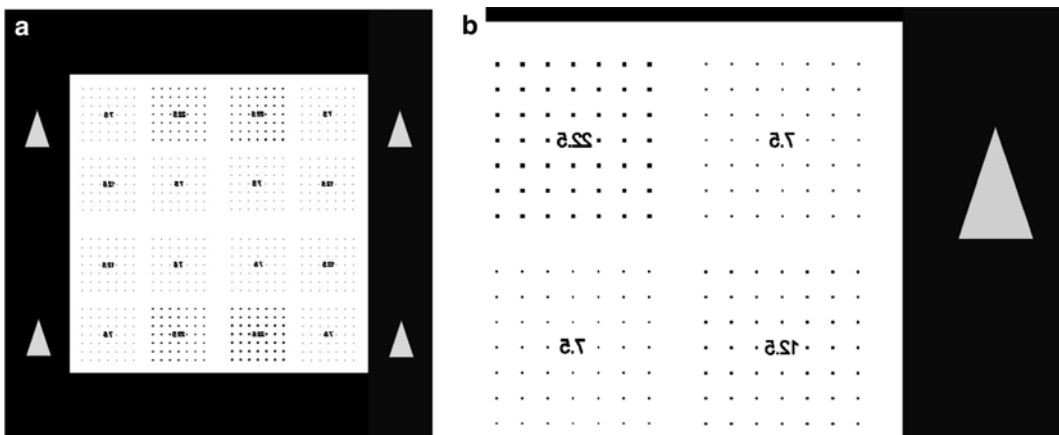


Fig. 1 (a) The complete mask layout. (b) The upper right edge of the layout design. The black parts cover the sample (positive resist). The triangles in the frame are markers for easier positioning, in particular, if a second lithography step is necessary. The mask has to be placed upside down on the sample, so the characters are reversed to get a correct layout on the sample

3.4 Ion Beam Etching

1. Clean dust specks from the sample with nitrogen gas and lock it in the etching chamber.
2. The base pressure inside the etching chamber should be approximately $3.0\text{--}4.0 \times 10^{-8}$ mbar and the etching (sputter) pressure approximately 3.5×10^{-5} mbar. The pressure is regulated by the Argon flow (approximately 2.0 sccm). The cathode current should be in the range 300–400 μA . The sample holder should rotate to achieve an equal etching rate over the entire sample. We have to etch through the tantalum oxide barrier into the palladium film (Fig. 2) (*see Note 3*).
3. After the etching procedure, it is possible to remove the remaining squares of photoresist through the use of an ultrasonic bath in acetone for approximately 5 min (*see Note 4*) and in ethanol for approximately 1 min. Only remove the photoresist, if no contact pads are needed and skip Subheading 3.5 as well as 3.6.

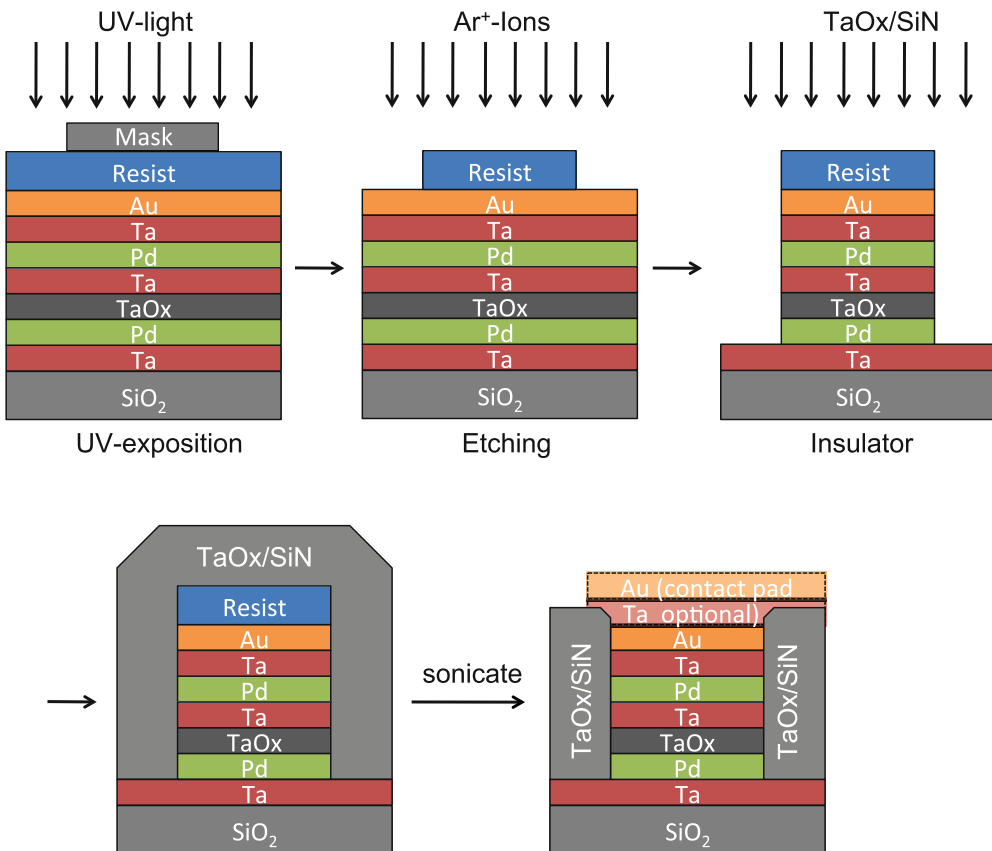


Fig. 2 The main steps from UV-exposition to the finished sample

3.5 Insulator

1. Before removing the photoresist, we mark one sample edge with a felt pen. Fill the spaces between the junction squares with an insulator such as silicon nitride or tantalum oxide via rf-sputtering. The insulator should have at least the same thickness as the sum of the etched films (here ≥ 60 nm).
2. Now, remove the photoresist as well as the felt pen mark with acetone in an ultrasonic bath. In our case, the sample has to stay 5–10 min in the ultrasonic bath in a beaker filled with acetone because of the processed photoresist (*see Note 4*).
3. The felt pen mark covered the surface and allows an easy removal of the insulator on top of it, therefore, providing access to the lower electrode.
4. This procedure makes it easier to contact the elements without a short circuit and protects the sample from scratches and other damage or adverse environmental conditions.

3.6 Contact Pads

It is possible to deposit contact pads on the elements, if the sizes of the junction pillars become very small. While it is easy to contact a junction pillar with lateral dimensions larger than $100\ \mu\text{m}$, it is very challenging to do this if the sizes approach $10\ \mu\text{m}$. In our case, we added contact pads with a size of $50 \times 100\ \mu\text{m}^2$ (*see Notes 5 and 6*).

1. At first, it may be necessary to spin coat the sample with photoresist as described in Subheading 3.3, **step 1**.
2. The contact pads are prepared using a lift-off process. Therefore we need a mask with a layout matching the previously used junction mask, i.e., the whole sample is covered and the $50 \times 100\ \mu\text{m}^2$ holes are on top of the junction pillars. Place the mask on the sample and irradiate it with the UV-light.
3. Sputter a 5 nm film of tantalum and a 60 nm film of gold, as is described in Subheading 3.2, **steps 1 and 4**.
4. Place the sample in a beaker filled with acetone for the lift-off process and place it for 5–10 min in the ultrasonic bath (similar to Subheading 3.5)

4 Notes

1. The sample is connected with silver paint to the sample holder. This is very important for the oxidation process, because it prevents charging of the sample.
2. According to the age of the resist and the developer, the size of the sample and of the junction pillars, the developing time may vary about some seconds.
3. The exact etching depth is not important. However, we have to meet two requirements. First, we must etch through the

barrier to prevent a short-circuit of the junctions via the top metal layers. Secondly, we must stop before the lower electrode thickness falls below approximately 10 nm, otherwise, the resistance of the lower electrode becomes higher than the resistance of the tunneling barrier which may lead to artifacts in the measurements.

4. The time in the ultrasonic bath is determined by the size of the junction pillars. Smaller junctions require longer removal, e.g., 30 min for sizes below lateral sizes of 1 μm . The removal of the resist cap should be checked with an optical microscope or a scanning electron microscope for junction sizes below 0.5 μm .
5. If contact pads are used, it is essential to increase the distance of the junction pillars in the design of the first mask. Otherwise, the larger contact pads will overlap and short-circuit neighboring junctions.
6. The contact pads should be placed off-center, with one edge close to the junction pillar. This prevents the junctions from damage, if we want to contact them by for example wire-bonding.

References

1. Thomas A (2013) Memristor-based neural networks. *J Phys D Appl Phys* 46(9):093001. doi:[10.1088/0022-3727/46/9/093001](https://doi.org/10.1088/0022-3727/46/9/093001)
2. Turel O, Lee J, Ma X et al (2004) Neuromorphic architectures for nanoelectronic circuits. *Int J Circ Theor Appl* 32(5):277–302. doi:[10.1002/cta.282](https://doi.org/10.1002/cta.282)
3. Chua L (1971) Memristor: the missing circuit element. *IEEE Trans Circ Theor* 18:507–519
4. Krzysteczko P, Münchenberger J, Schäfers M et al (2012) The memristive magnetic tunnel junction as a nanoscopic synapse-neuron system. *Adv Mater* 24:762–766
5. Afifi A, Ayatollahi A, Raissi F (2009) STDP implementation using memristive nanodevice in CMOS-Nano neuromorphic networks. *IEICE Electron Exp* 6(3):148–153. doi:[10.1587/ele.6.148](https://doi.org/10.1587/ele.6.148)
6. Indiveri G, Chicca E, Douglas R (2006) A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans Neural Netw* 17(1):211–221. doi:[10.1109/TNN.2005.860850](https://doi.org/10.1109/TNN.2005.860850)
7. Krzysteczko P, Reiss G, Thomas A (2009) Memristive switching of MgO based magnetic tunnel junctions. *Appl Phys Lett* 95(11):112508. doi:[10.1063/1.3224193](https://doi.org/10.1063/1.3224193)
8. Yang JJ, Zhang MX, Strachan JP et al (2010) High switching endurance in TaOx memristive devices. *Appl Phys Lett* 97(23):232102. doi:[10.1063/1.3524521](https://doi.org/10.1063/1.3524521)
9. Strachan JP, Torrezan AC, Medeiros-Ribeiro G et al (2011) Measuring the switching dynamics and energy efficiency of tantalum oxide memristors. *Nanotechnology* 22(50):505402. doi:[10.1088/0957-4484/22/50/505402](https://doi.org/10.1088/0957-4484/22/50/505402)
10. Torrezan AC, Strachan JP, Medeiros-Ribeiro G et al (2011) Sub-nanosecond switching of a tantalum oxide memristor. *Nanotechnology* 22(48):485203. doi:[10.1088/0957-4484/22/48/485203](https://doi.org/10.1088/0957-4484/22/48/485203)
11. Yakopcic C, Sarangan A, Gao J et al. (2011) TiO2 memristor devices. *IEEE National Aerospace & Electronics Conference*, July 2011, p 101–104
12. Pickett MD, Strukov DB, Borghetti JL et al (2009) Switching dynamics in titanium dioxide memristive devices. *J Appl Phys* 106(7):074508. doi:[10.1063/1.3236506](https://doi.org/10.1063/1.3236506)

Chapter 17

Architecture and Biological Applications of Artificial Neural Networks: A Tuberculosis Perspective

Jerry A. Darsey, William O. Griffin, Sravanthi Joginipelli,
and Venkata Kiran Melapu

Abstract

Advancement of science and technology has prompted researchers to develop new intelligent systems that can solve a variety of problems such as pattern recognition, prediction, and optimization. The ability of the human brain to learn in a fashion that tolerates noise and error has attracted many researchers and provided the starting point for the development of artificial neural networks: the intelligent systems. Intelligent systems can acclimatize to the environment or data and can maximize the chances of success or improve the efficiency of a search. Due to massive parallelism with large numbers of interconnected processers and their ability to learn from the data, neural networks can solve a variety of challenging computational problems. Neural networks have the ability to derive meaning from complicated and imprecise data; they are used in detecting patterns, and trends that are too complex for humans, or other computer systems. Solutions to the toughest problems will not be found through one narrow specialization; therefore we need to combine interdisciplinary approaches to discover the solutions to a variety of problems. Many researchers in different disciplines such as medicine, bioinformatics, molecular biology, and pharmacology have successfully applied artificial neural networks. This chapter helps the reader in understanding the basics of artificial neural networks, their applications, and methodology; it also outlines the network learning process and architecture. We present a brief outline of the application of neural networks to medical diagnosis, drug discovery, gene identification, and protein structure prediction. We conclude with a summary of the results from our study on tuberculosis data using neural networks, in diagnosing active tuberculosis, and predicting chronic vs. infiltrative forms of tuberculosis.

Key words Artificial neural networks, Intelligent systems, Bioinformatics, Drug discovery, Pattern recognition, Tuberculosis, Protein structure prediction, Gene identification

1 Introduction

Artificial neural networks (ANNs) are biologically inspired computer programs designed to simulate the way in which the human brain processes information [1]. ANNs are inspired by the way in which a biological nervous system, such as the brain, processes the information. The information paradigm in ANNs gathers knowledge from learning experience (through inspection of training

data) and makes intelligent predictions on the specified data set. An ANN is formed through the combination of a few to as many as hundreds of single units, artificial processing elements such as nodes which are connected with coefficients, or weights of processing elements and is organized into layers. A simple ANN has input, hidden, and output layers, while more complex networks can have more than one hidden layer with one input and one output layer. ANNs have the ability to derive meaning from complicated and imprecise data and can extract patterns and detect trends that are too complex for the human brain or other computer techniques to recognize. Neural networks are different from conventional computer techniques, as they encompass adaptive learning and real-time operation, whereas conventional computer techniques use algorithmic and programming approaches.

1.1 Human Brain

The average human brain has about 100 billion neurons or nerve cells, and a slightly greater number of neuroglial cells, which support and protect neurons. Neurons are considered to be the information transmission units of the brain, and each neuron is connected to up to 10,000 other neurons [2]. Neurons are the basic units of the nervous system, and have a cell body or soma, together with dendrites (inputs) and axon (outputs), as shown in Fig. 1. Each neuron is connected to other neurons through a synapse; typically a synapse is filled with a neurotransmitter chemical and connects the axon of one neuron and the dendrite of a second neuron. Signals are transmitted from one neuron to another through synaptic space; these signals can be either excitatory or inhibitory. Neurons are organized and extensively connected to form a complex network in the human brain, and act as messengers in sending and receiving impulses (signals). The complex network makes the human brain intelligent, with learning, memory, recognition, and prediction capabilities.

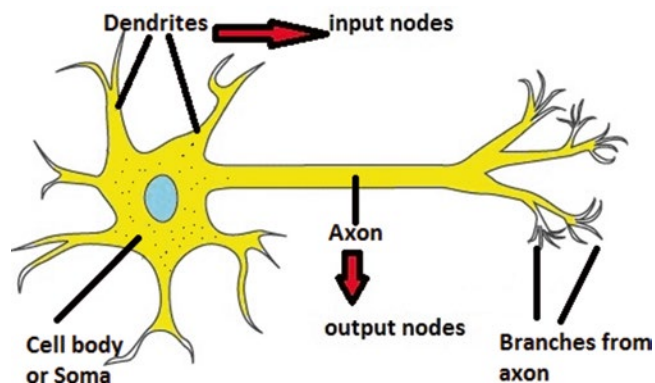


Fig. 1 Basic structure of neuron with cell body or Soma, dendrites as input nodes, and axon as output nodes. Branches from axon are connected to dendrites of other neuron in complex network of human brain

1.2 Architecture of Artificial Neural Networks

Although the basic unit in the structure of an ANN is computationally similar to the structure of a neuron in the human brain, it is much simpler biologically than a neuron in the brain. ANNs incorporate two fundamental elements of biological neural networks; neurons are represented as nodes and synapses are represented as weighted interconnections [3]. Artificial neurons are organized into layers: the input layer, hidden layer(s), and an output layer. Neural networks with one hidden layer are sufficient to make decisions and predictions on simple non-linear approximations, but networks with two or more hidden layers are used in more complex applications. A Neural network with one hidden layer is known as a simple perceptron, and with more than one hidden layer is called a multilayer perceptron [4]. As shown in Fig. 2, the input layer is made up of nodes, which contain an activated function. This layer communicates with one or more hidden layers, where the actual processing is done, through weighted connections. The final hidden layer then links to the output layer, which generates the network outcome or predictions.

$x_1, x_2,$ and $x_3 \dots$ are inputs and $w_1, w_2,$ and $w_3 \dots$ are weighted connections expressing the importance of inputs to the outputs (Fig. 1). The network’s output can in principle be any value, but in many applications is limited to the binary set {0, 1} This output is

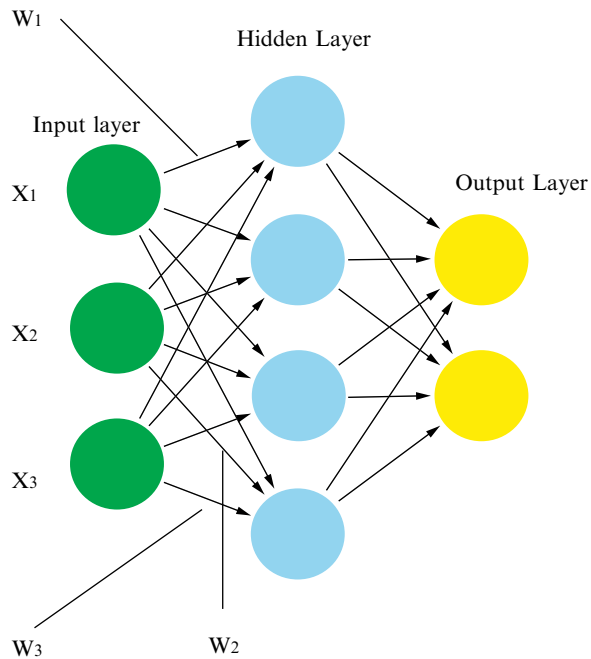


Fig. 2 Architecture of artificial neural network or simple perceptron with input, hidden, and output layers. $x_1, x_2,$ and x_3 of input layer corresponding to activation function and $w_1, w_2,$ and w_3 interconnections corresponding to weights

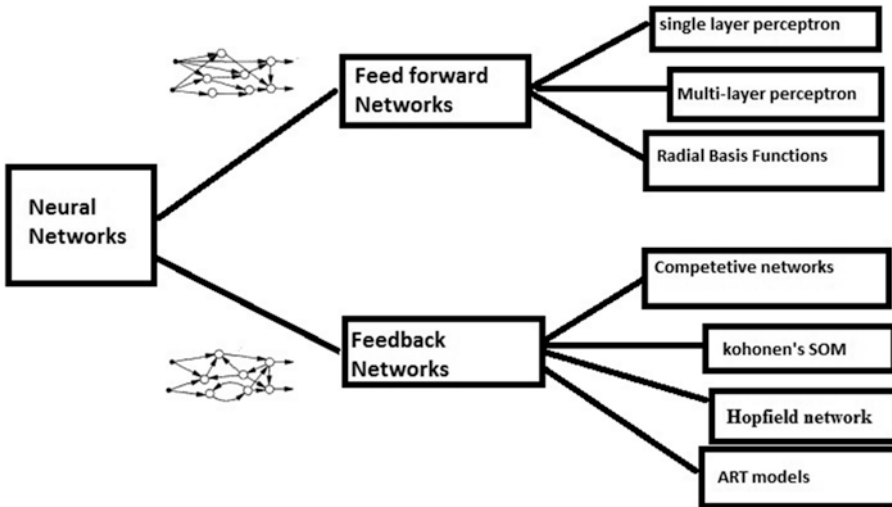


Fig. 3 Architecture of neural networks based on the connection patterns

determined by whether the weighted sum $\sum_j w_j x_j$ is less than or greater than a threshold value [5]:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

The working mechanism of ANNs

Artificial neural networks are considered as learning algorithms with nodes as activation functions and weights as edges or connections between the nodes. Based on the connection pattern or architecture of the network, ANNs can be either feed-forward networks or recurrent (feedback) networks (Fig. 3).

1.2.1 Feed-Forward Network

Feed-forward networks can be either single- or multilayer perceptrons with unidirectional connections, and are generally static and memory-less systems [5, 6]. In feed-forward networks the flow of information is in one direction only, with no loops, thus leading to only one set of output values. As the response to input is independent of the previous state, they are considered as memory-less systems [6].

1.2.2 Feedback Network or Recurrent Network

Feedback networks are dynamic, and the output is dependent on the previous layer.

When a new input pattern is presented, the neuron outputs are computed. Because of the presence of feedback paths, the inputs to each neuron are then modified, which leads the network to enter a new state [6].

1.3 Learning Algorithm of Neural Networks

The learning process in ANNs can be viewed as the process of updating network architecture and adjusting weighted patterns to efficiently achieve a specific task. A network learns from available training patterns, and the performance is improved over time by iteratively adjusting weight patterns in the network. The major advantage of neural networks when compared to traditional expert systems is that they learn from a given set of representative examples of training data rather than by following a set of rules previously defined by human experts, as in traditional computer systems [6].

The learning process or learning architecture of ANNs can be designed from known, available information about the training data set. A learning paradigm can be defined by having a model from a known environment, and learning rules can be defined by figuring out the update process for connection weights [6]. Identifying a procedure to adjust weights by use of a learning rule and following a learning paradigm will define the learning process or learning algorithm of the network.

The major learning paradigms of ANNs are

1. Supervised.
2. Unsupervised.
3. Hybrid.

In a supervised learning paradigm, a correct answer is provided for each input pattern and the weights are adjusted based on the degree to which network output agrees with that answer. In an unsupervised paradigm, as the answers are not provided, the system itself recognizes a correlation and organizes the patterns into categories accordingly. The hybrid paradigm combines both supervised and unsupervised learning methods. In hybrid learning methods, some weights are provided with correct answers, while some are auto corrected.

The four basic learning rules for a learning paradigm of ANNs are

1. Error-correction rules.
2. Boltzmann learning rule.
3. Hebbian rule.
4. Competitive learning rules.

1.3.1 Error- Correction Rule

All these learning rules can be applied to supervised, unsupervised, and hybrid paradigms. In the error-correction rule learning process, the error generated in each step is used to adjust the weights to efficiently achieve a task [6]. In the learning process the actual output y is different from the desired output d , so an expression related to the error ($d-y$) is used to adjust the weights to gradually reduce the error.

1.4 Boltzmann Learning Rule

Boltzmann learning is used in symmetric recurrent networks, i.e., those in which $w_{ij} = w_{ji}$, and consists of binary function units such as +1 for on and -1 for off. Outputs in this rule are produced according to Boltzmann statistical mechanics. Boltzmann learning adjusts weights until the visible units satisfy a desired probabilistic distribution [6].

1.4.1 Hebbian Rule

Hebbian rule is based on neurobiological experiments and is the oldest learning rule [6]. An important property of this rule is that learning is done locally; that is, change in weight depends only on the activities of two layers or neurons connected by it. Orientation selectivity occurs from a Hebbian training network.

1.4.2 Competitive Learning Rule

The basis of competitive learning is the “winner-take-all” process observed in biological neural networks. All input units are connected together and all units of output are also connected via inhibitory weights; however, the latter are fed back with excitatory weights. As a result of this learning process the pattern in the winner unit (weight) becomes closer to the input pattern.

2 Biological Applications of ANNs

ANNs are extremely powerful systems, with massive parallelism. They can learn and generalize from training data; the amount of programming required to efficiently achieve the task is usually modest. They are noise and fault tolerant, so they can cope with situations in which more conventional systems have difficulty. The inherent ability of ANNs to learn and recognize highly nonlinear and complex relationships in data makes them ideally suited to a wide range of applications. ANNs are useful in inferring function or predicting function from a set of observations where the complexity of data suggests that the use of conventional algorithms is unlikely to be successful. ANNs are used in regression analysis, function prediction, classification of patterns, character recognition, and image compression. Neural network applications are used in many disciplines such as medicine, bioinformatics, molecular biology, and pharmacology. ANNs are currently used in medicine for medical diagnosis, biochemical analysis, image analysis, and drug discovery [7].

Medical diagnosis: Due to the ability of neural networks to learn and recognize patterns from data, ANNs are currently used in medical diagnosis of cancer, to predict the outcome of chemotherapy in various stages of cancer research [7].

Biochemical analysis: ANNs have been used to analyze blood, urine samples, and body fluids. They can track the level of glucose in diabetes, and track the level of proteins in body fluids to detect

abnormalities in disease conditions. They can identify drug resistance in pathological conditions such as tuberculosis [7].

Image analysis: Medical image data-generated ultrasound, MRI, and X-rays can be analyzed with ANNs to detect disease conditions. Electrocardiographic (ECG) data can be analyzed to identify possible cardiovascular disease [7].

Drug discovery: Neural networks are used to identify potential drug candidates by virtual screening of large number of molecules of potential value in the treatment of AIDS, cancer, and tuberculosis [7].

ANNs are used in molecular biology and bioinformatics, for recognition of transcription and translational signals, protein structure prediction, gene identification, and sequence classification [8].

Recognition of transcription and translational signals: Recognition of transcription and translation promoter regions and termination of transcription and translational signals are useful in understanding defects in protein synthesis and defects in protein folding in inherited diseases such as Parkinson's disease and multiple sclerosis. Kalate, Tambe, and Kulkarni used ANNs to predict mycobacterial promoter sequences, and were able to achieve 97 % success in prediction using a multilayered neural network, trained with error back-propagation [8]. Scheila de Avila and Günther have used neural networks to predict prokaryotic promoter sequences and were able to achieve good results with high accuracy [9]. Zhang et al. successfully predicted promoter regions in *Escherichia coli* using a feed-forward neural network [10].

Protein structure prediction: Protein structure prediction involves identification of secondary structure and tertiary structure of a protein, and identification of three classes of secondary structures: α -helix, β -sheets, and random coils. Researchers have successfully applied neural networks to the prediction of protein structure. Holley and Karplus have used neural networks to predict protein structure, and achieved a maximum overall predictive accuracy of 63 % for helix, sheet, and coil. When filtering was used to include only the strongest 31 % of predictions, the predictive accuracy rose to 79 % [11].

Gene identification: Gene identification is simple in prokaryotes because the coding region in prokaryotes is a single strand of continuous reading frame, whereas in eukaryotes it consists of introns and exons. Therefore, an important task in identification of the gene-coding region in eukaryotes involves differentiation of introns, exons, and splice sites. The ANN approach has been applied to the recognition of coding region and gene identification; suitable gene identification tools and software have been developed by researchers. GRAIL and NetGene are gene identification software tools developed using ANNs. GRAIL is a multi-agent neural network system for gene identification developed by Ying Xu et al. [12].

NetGene server is a service producing neural network predictions of splice sites in human, *C. elegans*, and *A. thaliana* DNA [13, 14]. NetGene server is available online at URL <http://genome.cbs.dtu.dk/services/NetGene2/>

Sequence classification: A three-layered, feed-forward, back-propagation neural network was used by Wu et al. for full-scale protein sequence classification; using sequence encoding with singular value decomposition, they achieved close to 90 % sensitivity [15]. Blekas et al. have used ANNs for motif-based sequence classification; their experimental results on real datasets indicated that the ANN method was highly efficient and superior to other well-known protein classification methods [16].

We have successfully applied a neural network approach in our research on tuberculosis (TB), and Parkinson's disease. ANNs were used for identification of TB+ and TB-, and for the prediction of chronic and infiltrative TB from a given dataset. Neural networks combined with molecular modeling studies on Geldanamycin and similar compounds have predicted potential drug candidates for treating Parkinson's disease. Results generated from ANN applications on TB are shown in the results section of this chapter.

3 Methodology

It is believed that knowledge in the brain is gained by constant adaptation of synapses to different input signals, yielding better output signals; the results are constantly fed back as new inputs. In a similar fashion, ANNs try to mimic the adaptation of synapses by iteratively adjusting the weights associated with each node according to the differences between desired output and actual or obtained output.

As mentioned in Subheading 1, the ANN can have three or more layers and the results generated in each layer are back propagated to minimize the error. Figure 4 shows a sample of how the formulae coalesce into a neuron during training on a set of incoming data, passing through a hidden layer of nodes, and predicting an output value [17]. In all but the input layer the nodes compute the sum of the product of the input values passed on by the previous layer and the weight of the connection. After processing by a squashing or activation function, the value is sent to nodes in the next layer. The simplest configuration has one hidden layer with multiple nodes, but so-called deep networks have multiple hidden layers, to find complex interrelationships of input values [18]. An ANN with one hidden layer can be used to approximate any function, as shown in the Cybenko theorem [19].

Selection of a training set is an important step in training ANNs. The dataset is divided into three portions: a control group,

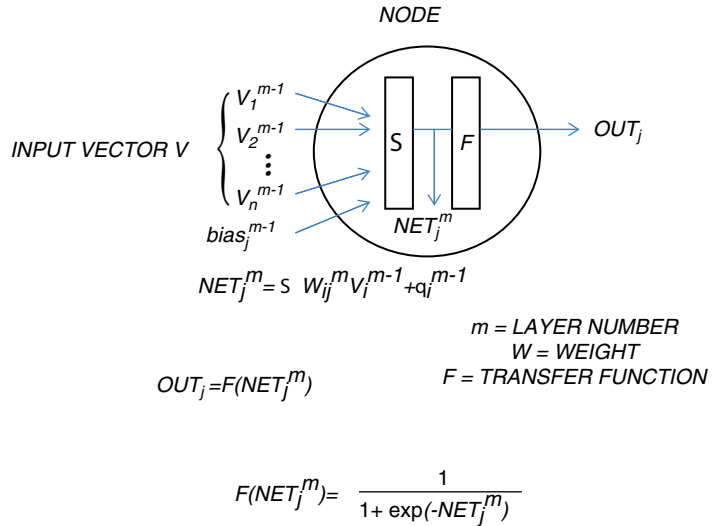


Fig. 4 The diagram of the node and transfer function of an artificial neural network

a training set, and a test set. Each of these portions contains approximately one-third of the data, with the training set often the smallest and the test set the largest. Control data is used as a supplier of feed information for correcting the ANN architecture for the learning approach. The training in an artificial neural network proceeds by passing values from the training set through the network many times, and inspecting the network outcome. The connection weights are adjusted as training proceeds as training seeks to minimize the error in the output. There are also parameters that govern an overall rate for learning and a momentum for the training which makes adjustments to how the weights are altered. A bias (or threshold) value regulates the prediction around a value, which is shown in Fig. 4.

The data which the neural network generalizes is typically pre-normalized to values between zero and one. It is also critical for the data to avoid the extremes of a sigmoidal curve; typical ANN practice uses values near zero and one, i.e., 0.05 and 0.95, to replace outputs of less than 0.05 and greater than 0.95, respectively. This constraint limits the data which can be predicted or analyzed. One consequence is that data values may need to be altered, for example by taking the logarithm, if values within the dataset lie outside these limits. A Boolean data type (true or false) can also be predicted using these numeric values as well by rounding the network output to integer values.

The quality of an ANN prediction can be measured in its adaptability, its ability to recognize patterns, and low generalization error. This greatly reduces the computational time for large datasets.

All prediction networks, which include artificial neural networks, are susceptible to factors which can limit their success, including overtraining, there being too few examples in the data set to train effectively, unreliable or noisy data, or missing factors needed for the prediction. Overextended or convoluted predictions are avoided by limiting the number of hidden nodes and the number of cycles for training the neural networks. Furthermore, the order in which data are presented to the network is randomized each training cycle to minimize selection bias in the test set and training set.

The learning algorithm computes good predictions by adjusting weights in a manner that keeps the network predictions simple and straightforward. Cross-validation is needed to check for overtraining and generalization errors [20]. The cross-validation is done with the leave n out technique, where n test cases are removed from the dataset in order to see if the neural network has learned from the data set. The number n can vary from 1 to as much as about 10 % of the total data available for training. Passing the value(s) withheld to the neural network, its prediction also improves over several learning cycles.

A partial least squares linear regression model applied to the data may help to validate the correlation found in the ANN model [21, 22]. Other tests available to researchers to confirm ANN are the runs test [22], which examines the number of series of consecutive values in the data for nonlinearity; Mallows augmented partial residuals plot [23, 24], a universal diagnostic tool which finds nonlinearities; Durbin-Watson test [25], which looks at the null hypothesis that there is no correlation; principal component analysis [26], a transformation procedure of the data into linear correlated parameters and scored components; and partial least squares analysis [26], a linear regression method which projects predicted and observable variables into a new space.

The program we used for prediction, developed at NASA's Johnson Space center, is called NETS [27]. The data used in this study was acquired from patients at the Kyrgyzstan National Center for Tuberculosis Research [28]. The presence or absence of genetic factors and types of TB is displayed in the data set as well as a few physiological factors: blood type, Rh factor, sex, age range, and northern or southern dwelling Kyrgyz. There were 149 patients in the control group and 146 in the test group. Although some of the occurrences of types of TB were rare within the test group, all patients were assessed in the ANN. Normally outliers in the prediction would be removed if warranted and the dataset examined again without the outliers. Since the distribution is bimodal, we were double blinded from examining results or patients, and the data set is from a small portion of the TB population from one country; the data were not corrected for outliers.

4 Results

As an example of biomedical applications of ANNs, we present results of a study that we performed on prediction of active TB from genetic factors in TB patient data. The network was predictable for many of the types of TB where sufficient training examples were available in the data set. At a plateau in the data, the network achieved stable training over several cycles. We used the results early on in these areas to extract weight values. In an analysis of the resulting weights, weight values lower than the mean were removed and the factors with significant influence on the outcome are shown in the inset of Fig. 5. Promotion or inhibition of the TB is given as a weighted value of the factors in the positive or negative direction.

The network predicted TB correctly in 72 % of the patients after ~145,000 cycles as seen in Fig. 5. The results revealed certain genes that inhibited or promoted TB likelihood. The top half of the weight values over the mean are shown in the inset as genes which promoted or inhibited the prediction.

A second network which predicted both the presence and absence of TB plateau for both cases after ~85,000 cycles is shown in Fig. 6. The resulting weight values were extracted and the top half of the weight values over the mean are shown in the inset of the graph. The weight values mirror each other exactly as one would expect when predicting both causative and inhibitory factors.

The third network was trained to predict chronic and infiltrative TB cases in the data set as seen in Fig. 7. The chronic type of

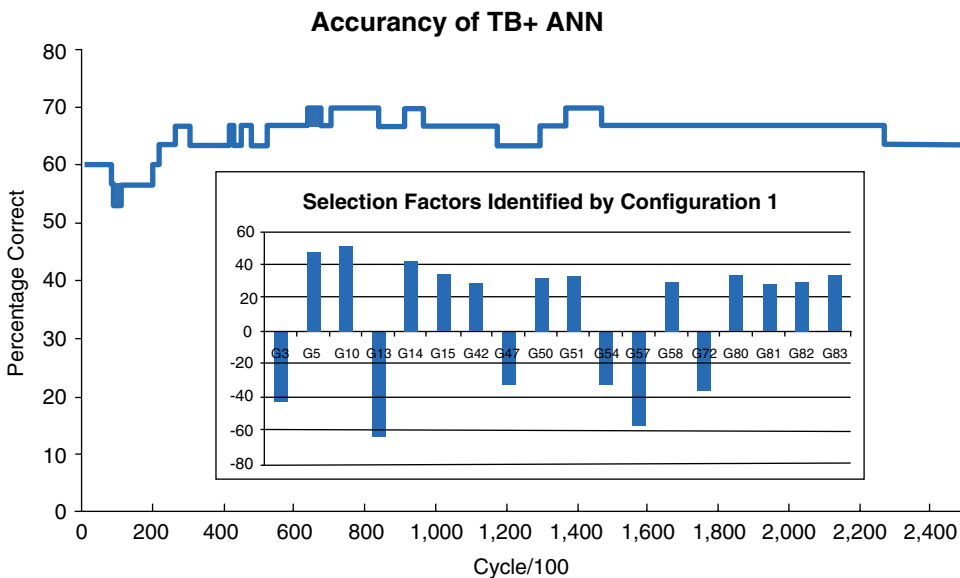


Fig. 5 The neural network prediction of TB cases in the dataset. Inset figure shows selection factors for the prediction which promoted or inhibited prediction by their relative weight strength

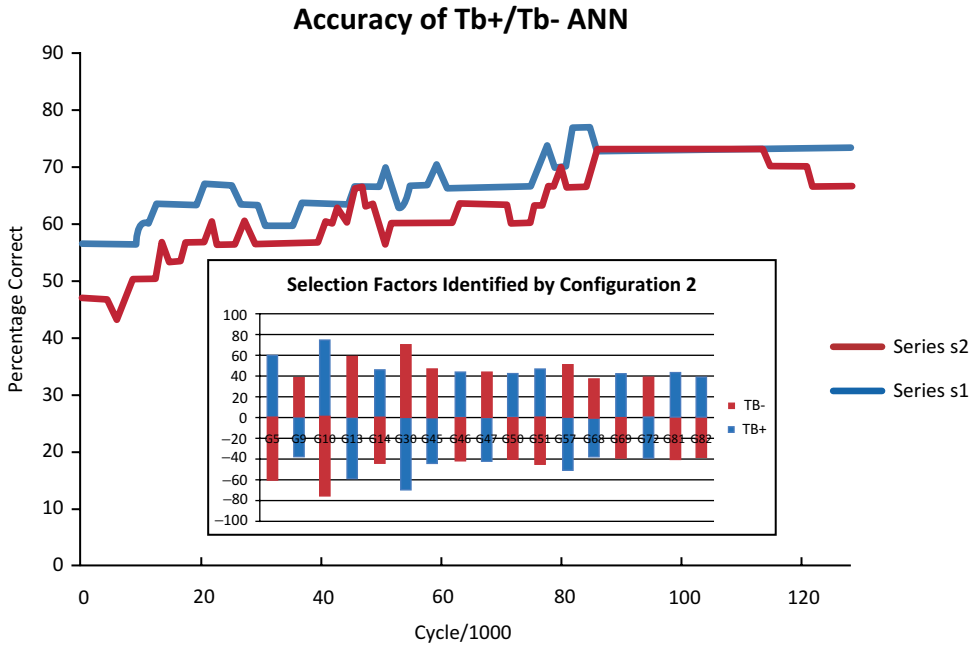


Fig. 6 Prediction of TB-infected and control group (non-TB) patients with ANN. The inset shows the selection factors and the weight strength of those factors

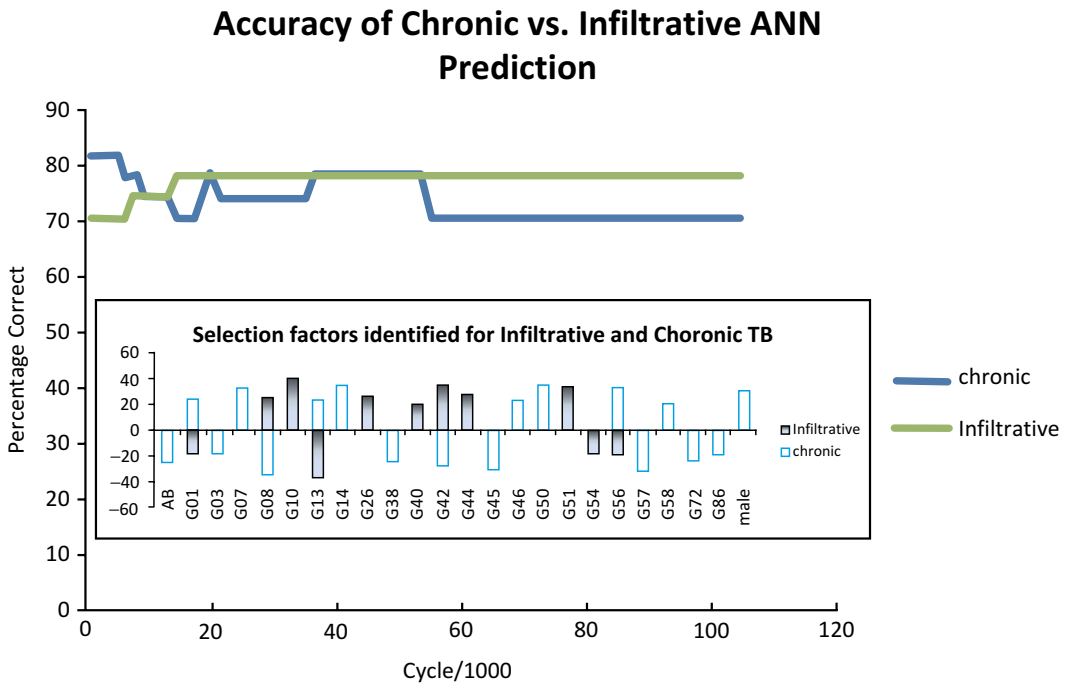


Fig. 7 The prediction of chronic and infiltrative TB from the data set. Inset shows factors which promoted or inhibited prediction of each type of TB

TB contained data for patients labeled: chronic, FKT, and empyema. The infiltrative type of TB contained the categories: infiltrative, positive test for mycobacterium TB, and drug-resistant TB. After ~18,000 cycles, the plateau region gave 77 % predictability. The inset figure shows the weight values which promote or inhibit both kinds of TB.

5 Discussion and Conclusions

The most important aspect of a TB infection control program is to identify patients with active TB; they can be isolated, as active TB is contagious, and can be efficiently treated. Early diagnosis is the key to effectively treating and preventing infection. The ANN method can be applied to diagnose active TB. Drug resistance is the most alarming and growing threat to the effective treatment of TB. Identification of drug resistance takes from several days to a few weeks, delaying the ability to treat a patient in the early stages. Predictive models such as neural networks can be used to identify drug resistance in mycobacterium; this could reduce the time required to diagnose drug resistance by traditional culture methods.

Radiological and clinical information from suspected patients can be used to diagnose TB, and that will be superior to a physician's opinion. A multilayered network approach could be used to diagnose TB, and to detect drug resistance. Computational methods and predictive models such as neural networks or hidden Markov models could reduce the time and cost associated with diagnosis and detection of lethal and infectious diseases [29]. The outcome of treatment by chemotherapy and radiotherapy for cancer and cardiovascular diseases can be predicted with neural networks.

The factors which promote and inhibit TB susceptibility are tools for researchers interested in tracing the genetic pathways of TB infection. Doctors with this information could more accurately identify patients who may be infected but as the waxing and waning of the symptoms progress, they are unaware of their infection. The genetic factors which inhibit TB could be used to study resistant populations and help find ways to find effective vaccines for TB. Identification is a key step in research, because as the mycobacterium tuberculosis develops more drug-resistant forms, the search for vaccines becomes more critical. The triggers for TB are unknown, and more research into those triggers would help identify more TB risk factors. Because genetic and physical factors are unalterable by the patients and environment, they can be attributed to the infection directly and experimented in a more controlled manner than in a study of external factors.

The first two predictions are comparable in their scope by predicting in the control versus infected patients. Both give the same percent predictability at nearly 70 % of the withheld group.

The data in the second prediction (TB+/TB-) has an inset of selection factors which mirror the results obtained by both tests. The results show that the associations the ANN link to the predicted parameter are not random and corroborate the promotion and inhibition factors consistently.

The third ANN prediction found parameters which affect the chronic and infiltrative types of TB. In addition to genetic links, it found links in the AB blood type, and male sex for chronic TB. There are certain genetic parameters which oppositely reflect both cases, either inhibiting or promoting the cases of chronic and infiltrative TB (genes 1, 8, 13, 42, and 56 in inset of Fig. 7). These oppositely reflecting genes may hold keys which identify the type of TB a patient would be likely to contract.

Improvements can be made in how the data is used. The TB and non-TB patients group each had almost 150 patients per group. This sample size would be improved by removing outliers to the prediction, and with 150 cases in each group, this is a large enough study to prune outliers. The number of input factors could also be trimmed since there are some factors which have more influence than others.

References

1. Agatonovic-Kustrin S, Beresford R (2000) Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J Pharm Biomed Anal* 22(5):717-727, ISSN 0731-7085. <http://www.sciencedirect.com/science/article/pii/S0731708599002721>
2. MacDonald M (2008) *Your brain: the missing manual*. Pogue, Sebastopol, CA
3. Gonzalez AJ, Dankel DD (1993) *The engineering of knowledge-based systems*. Prentice-Hall, Englewood Cliffs, NJ. ISBN 0-13-334293
4. Zahedi Z (1993) *Intelligent systems for business: expert systems with neural networks*. Wadsworth Inc., Belmont, CA. ISBN 0-534-18888-5
5. Nielsen MA (2014) *Neural networks and deep learning*, Chapter 3, *Improving the way neural networks learn*. Determination Press. <http://neuralnetworksanddeeplearning.com/chap1.html>
6. Jain AK, Mao JC, Mohiuddin KM (1996) *Artificial neural networks: a tutorial*. *Computer* 29(3):31
7. Dybowski R (2000) *Neural computation in medicine: perspective and prospects*. *Artificial neural networks in medicine and biology*. Proceedings of the ANNMAB-1 conference, Sweden, ISBN 978-1-85233-289-1
8. Kalate RN, Tambe SS, Kulkarni BD (2003) *Artificial neural networks for prediction of mycobacterial promoter sequences*. *Comput Biol Chem* 27(6):555-564, ISSN 1476-9271
9. de Avila S, Silva GJLG (2011) *Rules extraction from neural networks applied to the prediction and recognition of prokaryotic promoters*. *Genet Mol Biol* 34(2):353-360
10. Zhang F, Kuo MD, Brunkhors A (2006) *E. coli promoter prediction using feed-forward neural networks*. Proceedings of the 28th IEEE, EMBS Annual International Conference, New York City, USA 2025-2027
11. Holley HHL (1989) *Protein secondary structure prediction with a neural network*. *Proc Natl Acad Sci U S A* 86(1):152-156
12. Xu Y, Mural RJ, Einstein et al (1996) *GRAIL: a multi-agent neural network system for gene identification*. *Proc IEEE* 81(10):1544-1552
13. Hebsgaard PGK, Tolstrup N, Engelbrecht J et al (1996) *Splice site prediction in Arabidopsis thaliana DNA by combining local and global sequence information*. *Nucleic Acids Res* 24(17):3439-3452
14. Brunak S, Engelbrecht J, Knudsen (1991) *Prediction of human mRNA donor and acceptor sites from the DNA sequence*. *J Mol Biol* 220:49-65
15. Wu C, Shivakumar S, McLarty J (1995) *Neural networks for full-scale protein*

- sequence classification: sequence encoding with singular value decomposition. *Machine Learning* 21(1–2):177–193
16. Blekas K, Likas A (2005) Motif-based protein sequence classification using neural networks. *J Comput Biol* 12:64–82
 17. Griffin WO, Razorilova S, Kitaev M et al (2010) An artificial neural network evaluation of tuberculosis using genetic and physiological patient data. *AIP Conference Proceedings*, Little Rock, 1229, p 49–53
 18. Bengio Y (2009) Learning deep architectures for AI. In: *Foundations and trends in machine learning*, Now Publishers Inc. 2(1):1–127
 19. Cybenko GV (1989) Approximation by superpositions of a sigmoidal functions. *Math Control Signal Syst* 2:303–314
 20. Gasteiger J (1993) *Neural networks for chemists: an introduction*. VCH Publishers, New York, NY
 21. Lavine BK (2000) *Chemometrics*. *Anal Chem* 72(12):91R–97R
 22. Goicoechea HC, Collado MS, Satuf ML et al (2002) Complementary use of partial least-squares and artificial neural networks for the non-linear spectrophotometric analysis of pharmaceutical samples. *Anal Bioanal Chem* 374:460–465
 23. Centner V, de Noord OE, Massart DL (1998) Detection of nonlinearity in multivariate calibration. *Anal Chim Acta* 376(2):153–163
 24. Mallows CL (1986) Augmented partial residuals. *Technometrics* 28(4):313–319
 25. Drapper NR, Smith H (1981) *Applied regression analysis*, 2nd edn. New York, NY, Wiley
 26. Brereton RG (2003) *Chemometrics: data analysis for the laboratory and chemical plant*. John Wiley & Sons Ltd., West Sussex
 27. Shelton RO (2000) A neural network development tool: NETS, NASA's Johnson Space center. Open Channel Foundation, Publishing software from academic and research institutions. <http://www.openchannelfoundation.org/projects/NETS/>
 28. Tarasenko OJ, Kitaev M, Alisherov A et al (2000) Polymorphisms of HLA-A and -B genes in the Kyrgyz population. *Anthrop Sci* 4(108):293–304
 29. Rafei A, Pasha E, Jamshidi OR (2012) Tuberculosis surveillance using hidden Markov model. *Iran J Public Health* 41(10):87–96

Neural Networks and Fuzzy Clustering Methods for Assessing the Efficacy of Microarray Based Intrinsic Gene Signatures in Breast Cancer Classification and the Character and Relations of Identified Subtypes

Sandhya Samarasinghe and Amphun Chaiboonchoe

Abstract

In the classification of breast cancer subtypes using microarray data, hierarchical clustering is commonly used. Although this form of clustering shows basic cluster patterns, more needs to be done to investigate the accuracy of clusters as well as to extract meaningful cluster characteristics and their relations to increase our confidence in their use in a clinical setting. In this study, an in-depth investigation of the efficacy of three reported gene subsets in distinguishing breast cancer subtypes was performed using four advanced computational intelligence methods—Self-Organizing Maps (SOM), Emergent Self-Organizing Maps (ESOM), Fuzzy Clustering by Local Approximation of Memberships (FLAME), and Fuzzy C-means (FCM)—each differing in the way they view data in terms of distance measures and fuzzy or crisp clustering. The gene subsets consisted of 71, 93, and 71 genes reported in the literature from three comprehensive experimental studies for distinguishing Luminal (A and B), Basal, Normal breast-like, and HER2 subtypes. Given the costly procedures involved in clinical studies, the proposed 93-gene set can be used for preliminary classification of breast cancer. Then, as a decision aid, SOM can be used to map the gene signature of a new patient to locate them with respect to all subtypes to get a comprehensive view of the classification. These can be followed by a deeper investigation in the light of the observations made in this study regarding overlapping subtypes. Results from the study could be used as the base for further refining the gene signatures from later experiments and from new experiments designed to separate overlapping clusters as well as to maximally separate all clusters.

Key words Breast cancer, Neural networks and fuzzy clustering, DNA microarray, Gene expression signatures, Subtype classification, Gene clustering

1 Introduction

Breast cancer, a cancer in the breast tissues, is the second most common cancer (after lung cancer) and the most common cancer in women. Breast cancer is comprised of two types depending on its original tissue: ductal carcinoma and lobular carcinoma. Breast cancer molecular subtypes can be deciphered by using their distinct

patterns of gene expression. There are at least five intrinsic subtypes: luminal A (estrogen receptor [ER] and/or progesterone receptor [PR] positive, HER2-), luminal B (ER and/or PR+, HER2+), HER2 (ER and/or PR-, HER2+), basal-like (ER-, PR-, HER2-) and normal breast-like. Recent understanding on the biology of breast cancer can be found in a review by Carey [1].

Microarray technology provides an extensive source of gene expression data that promise to pave the way for better prediction and diagnosis of cancer and identification of key targets for drug development. It allows simultaneous measurement of expression of a large number of genes that is typically represented by an expression matrix in which the rows represent experiments or patients and columns represent genes. Microarray data have been widely used in breast cancer and recent reviews and details of microarray based research on breast cancer can be found in [2–5]. Currently, microarray research falls into two major themes:

- Classification of breast cancer subtypes [6–8].
- Development of prognostic gene signatures and clinical gene markers to predict disease recurrence, response to therapy, survival outcome and drug sensitivity [9–17].

In classification of breast cancer subtypes using microarray data, clustering is commonly used. This involves clustering gene expression patterns (profiles, signatures) representing patients. As unknown subtypes are typically sought, unsupervised clustering methods are used where the expression profiles are divided into highly similar groups without predefined group labels, based on a similarity/dissimilarity or distance measure. The main clustering method used so far in these studies is hierarchical clustering. Although this reveals basic patterns of clusters, more needs to be done to investigate the accuracy of clusters as well as to extract meaningful cluster characteristics and relations from them to increase confidence in the results for use in a clinical setting. To this end, it is important to understand the features of the clustering tools and how to select the appropriate tools in order to reveal biological knowledge embedded in gene expression profiles in relation to molecular subtypes.

Breast cancer is a complex and heterogeneous disease. Many studies have been conducted but the challenge still is to better understand the biology of breast cancer. Microarrays have been used to better understand molecular subtypes of breast cancer and this is an ongoing research area: microarray data on breast cancer is now available for public access and recent research has also been reviewed by several groups [18–21]. There are many studies on breast cancer gene signatures, including those that attempt to distinguish between subtypes or find a prognostic gene signature. Some details on breast cancer gene signatures from previous studies are provided in Table 1, adapted from Weigelt, Peterse, and van't Veer [22].

Table 1
Microarray studies on prognostic gene-expression profiles in breast cancer (adapted from Weigelt, Peterse, and van't Veer [22])

Microarray type	Informative genes	Outcomes	References
cDNA	496 “intrinsic” genes	Identified 4 subtypes :Luminal-like/ER+ gene, ERBB2+ overexpressed, Basal-like, Normal breast-like	[6]
cDNA	456 “intrinsic” genes	“Luminal A” tumors have a better outcome than “luminal B” tumors. Worst outcome is for “basal-like” and “ERBB2+” tumors	[7]
cDNA	534 intrinsic genes	Tested results from [7] using independent datasets	[8]
Oligonucleotide	70 genes “70-gene Mamma Print”	“Identified a Good signature” related to low metastasis risk; a “poor signature” associated with a high metastasis risk. Sensitivity: 91 %, specificity: 73 %	[12]
Oligonucleotide	70 genes	“Good signature” versus “poor signature.” Sensitivity: 93 %, specificity: 53 %	[36]
Oligonucleotide	“metagenes”	Accuracy: 90 %	[37]
Oligonucleotide	76 genes	“Good 76-gene signature” versus “poor 76 gene-signature.” Sensitivity: 93 %, specificity: 48 %	[38]
Oligonucleotide	442 genes	Expression of “serum-activated” signature versus no expression. Sensitivity: 91 %, specificity: 29 %	[39, 40]

(continued)

Table 1
(continued)

Microarray type	Informative genes	Outcomes	References
The “combined test set” data from Sorlie et al. [7, 8] (cDNA microarrays), van’t Veer et al. [12] (custom Agilent oligo microarrays) and Sotiriou et al. [28] (cDNA microarrays)	306 genes	Validated the breast cancer “intrinsic subtypes” by combining existing data sets	[27]
The “combined test set” data from Sorlie et al. [7, 8] (cDNA microarrays), Hu et al. [27] and Perreard et al. [41] [GEO:GSE2607] (Agilent Human A1, Agilent Human A2, and custom oligonucleotide)	50 genes subtype predictor “PAM50”	Incorporated “intrinsic” subtypes into risk model for breast cancer prognosis and prediction	[42]
Oligonucleotide Affymetrix U133a microarrays: Wang and colleagues [GEO:GSE2034], Miller and colleagues [GEO:GSE3494], and Pawitan and associates [GEO:GSE1456]	306 genes	Identified relations between breast subtypes, pathway deregulation, and drug sensitivity.	[43]

We focus on breast cancer subtype classification in this chapter. As shown in Table 1, Perou et al. [6] is the first group to focus on distinguishing breast cancer subtypes, having carried out a series of microarray experiments in 2000, 2001, and 2003, with subsequent analysis based on hierarchical clustering. Their objective was to identify genes characteristic of subtypes. From a long list of differentially expressed genes characterizing the subtypes, they were able to extract a small number of “intrinsic gene subsets” for each subtype. More recent studies tend to use the combined samples from these previous studies and have resulted in a further reduced number of informative genes [11]. A review of the current situation of breast cancer classification, in terms of classification efficacy, shows that the search is still on to identify the most crucial “intrinsic genes” that can be used at the clinical level to effectively classify breast cancer subtypes which, in turn, can be incorporated into cancer prognosis and prediction. As stated earlier, most of the proposed intrinsic genes and subsets have been analyzed and identified using hierarchical clustering. Therefore, this study attempts to study the consistency, character, and relations of the subtypes based on the intrinsic gene subsets reported in literature, when viewed through the lens of different and more advanced clustering methods. The two main objectives in this study are as follows:

Objective 1: Study the effectiveness of the reported intrinsic gene subsets in representing the breast cancer subtypes.

Objective 2: Test the efficacy of the gene subsets in classifying subtypes and investigate the relationship/similarity between subtypes

In order to achieve these objectives, we study gene clustering and patient clustering. We used the original “intrinsic” genes from Sorlie et al. [7, 8] for gene clustering. We used four selected clustering tools: Self-Organizing Maps (SOM) [23], Emergent Self-Organizing Maps (ESOM) [24], Fuzzy clustering by Local Approximations of MEMberships (FLAME) [25], and Fuzzy C-Means (FCM) [26]. In patient clustering, we compare the “intrinsic gene” sets from Sorlie et al. [8] and Hu et al. [27] using three clustering methods: SOM, SOM clustering aided by the Ward Method, and FCM.

2 Materials

2.1 Dataset

There are three main datasets used in this study:

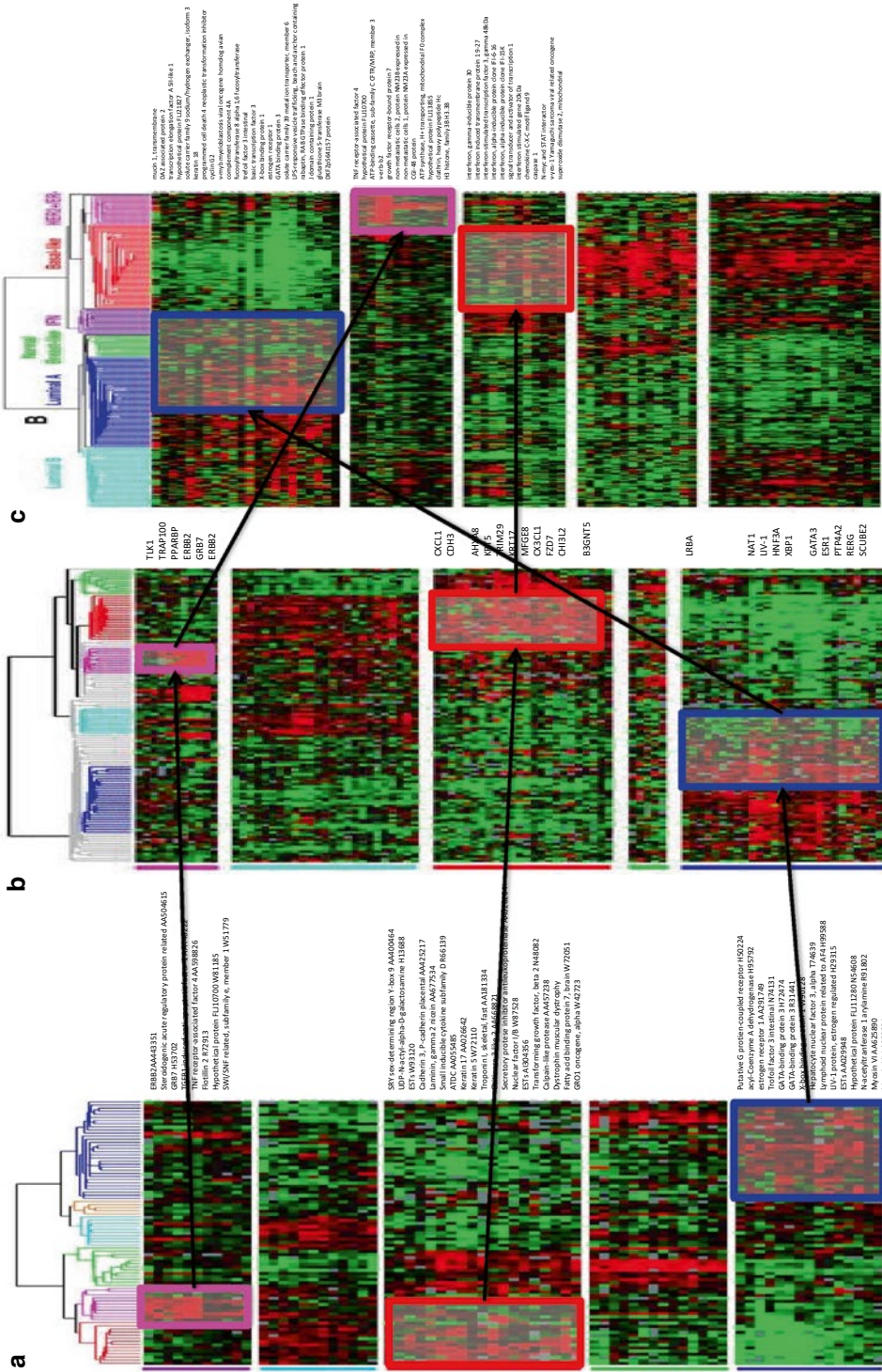
The first dataset was from Sorlie et al. [7]. Specifically, a 71-gene subset selected from 456 intrinsic genes as shown in Fig. 1a: Group C—ERBB2+/HER2 (11 genes), Group D—Novel unknown (12 genes), Group E—Basal-like (19 genes), Group F—Normal breast-like (14 genes), and Group G—Luminal (15 genes). These were derived from the raw data from a total of 85 cDNA microarray experiments on tumor and normal breast tissues (71 ductal, five lobular, two ductal carcinoma in situ, three fibroadenoma (benign tumor), and four normal breasts).

The dataset was retrieved from http://genome-www.stanford.edu/breast_cancer/mopo_clinical/ or <http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE3193>.

The second dataset was from Sorlie et al. [8]. We used a 93-gene subset derived from 534 intrinsic genes (see Fig. 1b): group C (ERBB2+/HER2 (11 genes)), group D (Luminal Subtype B (28 genes)), group E (Basal-like) (24 genes), group F (Normal breast-like) (5 genes), and group G (Luminal Subtype A) (25 genes). These were derived from a total of 122 samples from Norway/Stanford cohort which included the above mentioned 85 samples from the previous (2001) study used to classify cancer subclasses by average linkage hierarchical clustering.

The dataset was retrieved from http://genome-www.stanford.edu/breast_cancer/robustness/ (also available at <http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE4335>).

In the above studies the raw data (cDNA) were processed and genes were selected with a criterion of at least fourfold expression from the median in at least three of the samples tested. Then differentially expressed genes were used to find the intrinsic genes by calculating the “within-pair variance” score and the “between-subject variance.” The ratio between the two scores is called D and intrinsic genes are those with a small value of D . This resulted in



456 intrinsic genes from the first dataset and 534 intrinsic genes from the second dataset. The authors then analyzed these genes by hierarchical clustering, resulting in five distinct subtypes. Finally, a small number of novel genes representing each subtype were identified in these studies, as shown in the highlighted boxes in Fig. 1a and b for 456 and 534 genes, respectively.

The last dataset was retrieved from Hu et al. [27]. This had a 71-gene subset derived from 306 intrinsic genes (see Fig. 1c): Group C—Luminal (21 genes), Group D—HER2 (11 genes), Group E—Interferon-regulated cluster containing STAT1 (12 genes), Group F—Basal-like (14 genes), and Group G—Proliferation cluster (13 genes). A total of 311 tumors and 4 normal breast cancer samples was created by combining three existing datasets of Sorlie et al. [7, 8], van't Veer et al. [12], and Sotiriou et al. [28]. The gene subsets containing 71 of the 306 intrinsic genes are highlighted in boxes in Fig. 1c. The dataset was retrieved from the UNC Microarray Database (<http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE1992> or <https://genome.unc.edu/pubsup/breastTumor/>).

In our study reported in this chapter, we explored these small subsets of intrinsic genes separately, specifically, 71 and 93-gene subsets from Sorlie et al. [7, 8] and 71-gene subset from Hu et al. [27]. Some additional details of these subsets and the studies are given in Table 2.

Table 2
Summary of data, intrinsic genes, and identified subtypes in studies from 2000 to 2006 [6–8, 27]

	2000 (Perou et al.)	2001 (Sorlie et al.)	2003 (Sorlie et al.)	2006 (Hu et al.)
No. of samples	42 individuals (36 ductal + 2 lobular + 1 ductal carcinoma in situ + 1 fibroadenoma + 3 normal breast)	85 tissue samples representing 84 individuals (71 ductal + 5 lobular + 2 ductal carcinoma in situ + 3 fibroadenoma + 4 normal breast)	122 individuals (including 84 from 2001)	315 samples combined from Sorlie et al. [7, 8], van't Veer et al. [12] and Sotiriou et al. [28]
Intrinsic genes	496	456	534	306
Subtypes	1. Luminal-like / ER+ gene 2. HER2 3. Basal-like 4. Normal breast-like	1. Luminal 2. HER2 3. Basal-like 4. Normal breast-like 5. A novel unknown cluster	1. Luminal A 2. Luminal B 3. HER2 4. Basal-like 5. Normal breast-like	1. Luminal A 2. Luminal B 3. HER2 4. Basal-like 5. Normal breast-like and 2 new clusters: Proliferation, Interferon-reg.

3 Neural Networks and Fuzzy Clustering Methods

Four methods were used in this study: Self-Organizing Maps (SOM), Emergent Self-Organizing Maps (ESOM), Fuzzy Linear Approximation of Membership (FLAME), and Fuzzy c means (FCM). A summary of these methods is given in the subsections that follow:

3.1 Self-Organizing Feature Map (SOM)

An artificial neural network (ANN) is a collection of interconnected neurons that incrementally learn from their environment to capture essential linear or nonlinear trends in complex data, so that it provides reliable predictions for new situations containing even noisy and partial information [29]. There are various ANNs that can be constructed and used for various purposes. This research study uses Self-Organizing Feature Map (SOM). Self-organizing maps are unsupervised neural networks that preserve the exact topology of data space on a two-dimensional grid of neurons. The components of SOMs are neurons that are normally arranged in a two-dimensional hexagonal or rectangular grid, as shown in Fig. 2. The input layer represents the input variables x_1, x_2, \dots, x_n for the case of n inputs and each neuron (circles in Fig. 2) is associated with an n -dimensional weight vector $w_i = [w_1, w_2, \dots, w_n]$. The competition among the neurons is the mechanism responsible for self-organization in the brain, a fact which is utilized in developing SOM networks. Through a training process involving such competition to iteratively modify weights, inputs (e.g., gene expression vectors) are mapped from high dimensions to this two-dimensional map space, where cluster structure as well as proximity of clusters to each other can be visualized.

Training starts with assigning to neurons random weight vectors with the same dimensions as the input vectors; Euclidean distance (or any other distance measure) is then used to calculate the distance between an input vector and each of the weight vectors, as shown in Eq. 1. The neuron with the weight vector closest to the input vector (i.e., with the smallest distance) is declared the winner.

$$\text{Euclidean Distance} = d_j = \|x - w_j\| = \sqrt{\sum_{i=1}^n (x_i - w_{ij})^2} \quad (1)$$

The objective of learning in SOM networks is to accurately project the input data onto the map neurons. Over repeated exposure to inputs during learning, neurons learn to respond to inputs in such a way that neurons that are closer on the SOM respond to inputs that are closer in the actual data space. This feature, known as topology preservation, is achieved by moving not only the winner neuron but also its neighbor neurons towards the input in order to minimize the distance between these neurons and the input. In

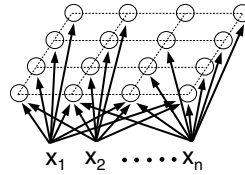


Fig. 2 Self-organizing feature map

this process, the winner moves by a larger amount than its neighbors according to a Neighbor Strength (NS) function.

For each neuron $j \in N_c$, where N_c is the neighborhood surrounding the winner neuron c , learning involves adjusting the new weight as follows:

$$w_j^{\text{new}} = w_j^{\text{old}} + \varepsilon(t) \text{NS}(d, t) [x_i - w_j^{\text{old}}] \quad (2)$$

where w_j^{new} is new weight, w_j^{old} is the previous weight, ε is learning rate, x_i is the i th input vector, and NS is a neighborhood function defining the strength of weight adjustment of neighbors with respect to the winner at iteration t and can take the form of a linear or nonlinear function of distance d from the winner. The learning rate $\varepsilon(t)$ is user-defined and can take the form of a linear, exponential, etc. function. Starting with a larger value, $\varepsilon(t)$ decreases with iterations. $\text{NS}(d, t)$ also can be linear, exponential, Gaussian, etc. and shrinks with iterations so that an initially large neighborhood size gets reduced to just the winner, or the winner and its immediate neighbors at the end of training. Training stops when there is no acceptable difference in weight adjustment in successive iterations. In this study, we used the Neural Networks Toolbox (Matlab) for training the SOM.

SOM training results in a map where inputs that are closer in the data space are projected onto neurons that are closer on the SOM. Therefore, any large gaps between data in the data space are shown as large distances between corresponding neurons on the map, thereby revealing on the map potential clusters and cluster boundaries in the actual data. With an appropriate clustering tool, such as Ward Clustering, neurons that form clusters on the map can be grouped to form classes for classification purposes. An unknown input vector then will belong to the class that it is closest to, i.e., the class with the shortest distance between its mean vector and the input vector. This provides a crisp classification for the input vector—one input vector belongs to one class.

In a trained SOM, each neuron represents the center of gravity of a number of input vectors. SOM results can be visualized in various ways. In one form, a U-matrix (Unified distance matrix) the average distance between a neuron and its neighbors is shown with colors on the map. This can show distinct subtypes or gene clusters

depending on whether patient or gene clustering is done. Another visualization scheme shows the spread of values of each input variable (e.g., expression of a gene across all patients or expression of all genes in a patient depending on whether genes or patients are clustered) in a separate map pane. In order to quantify clusters, trained SOM neurons (weight vectors) are subjected to a clustering algorithm. In this study, the Ward method, an efficient clustering method suitable for small datasets, was used to find clusters of genes and cancer subtypes on the SOM using an in-house developed algorithm implementing Ward clustering.

Ward clustering [30] is an agglomerative clustering method invented in 1963. For incrementally forming clusters, the method calculates the distance between data and cluster centers for all potential clusters by analyzing the variance amongst them. Specifically, it aims at minimizing the sum of square distance between all data points in a cluster and the centroid of the cluster formed as a result of joining two clusters. The two clusters that produce the least square distance are ultimately joined in each step of the process. The distance is calculated as:

$$d_{\text{ward}} = \frac{(n_r \times n_s)}{(n_r + n_s)} \|x_r - x_s\|^2 \quad (3)$$

where r and s symbolize two clusters and n_r and n_s are the number of data points in those clusters. $\|x_r - x_s\|$ gives the magnitude of the distance between two cluster centers calculated using Euclidean norm. The Ward distance increases with the increase in the number of data points. The two clusters with minimum d_{ward} are joined at each step of this process. The ward index gives the likelihood of each cluster. The separation of clusters depends upon the value of the index. The higher the index, the more distinctive are the clusters.

$$\text{Ward Index} = \frac{1}{\text{NC}} \times \frac{d_t - d_{t-1}}{d_{t-1} - d_{t-2}} = \frac{1}{\text{NC}} \times \frac{\Delta d_t}{\Delta d_{t-1}} \quad (4)$$

where Δd_t is the distance between the centers of two clusters to be merged and Δd_{t-1} is the distance between the centers of the clusters merged previously and prior to that step. NC is the number of clusters remaining. The method is very effective when the dataset is not too large. Therefore, it is quite suitable for use on a trained SOM to cluster map neurons. In this case, it uses the neuron weights generated by the trained SOM as input vectors.

3.2 Emergent Self-Organizing Maps (ESOM)

ESOM is an extension of SOM which allows a map to grow in size from the initial map. ESOM is arranged in a toroidal grid using a larger number of neurons than a typical SOM [31]. An ESOM-map can be visualized in three forms: U-Matrix (distance-based), P-Matrix (density-based)

and U*-Matrix (distance and density based). The U-Matrix, which indicates the average distance of each neuron to its nearest neighbors is the same as the U-Matrix in SOM, so this study uses the P-Matrix in order to analyze density of gene expression data. The P-Matrix contains entries denoting density of neurons in the neighborhood of a neuron. A neuron with a large P is located in a dense data region while a small p indicates sparse data regions in the data space. For neuron n , the density $P(n)$ in the data space X can be defined as:

$$P(n) = p(w(n), X), \quad (5)$$

where $p(x, X)$ denotes an empirical density estimation at point x (i.e., $w(n)$ which is the weight vector of neuron n) in the data space X and ESOM uses Pareto Density Estimation (PDE) [31], an optimal information kernel density estimation.

Data can be analyzed by using publicly available ESOM software (Databionics ESOM Tool) developed by Ultsch and Morchen [31]. This software is available to download from <http://databionic-esom.sourceforge.net/>.

3.3 Fuzzy Clustering by Local Approximation of MEmbership (FLAME)

The Fuzzy clustering by Local Approximation of MEmbership (FLAME) algorithm starts by calculating similarities of gene expression patterns using Pearson correlation, and then creating a connected graph of all K-Nearest Neighbors (KNN) of each expression vector. For each expression pattern, density, which indicates the number of neighbors within a specified distance, is calculated to classify it into one of three groups: (1) *cluster supporting object* (CSO) which has higher density than their neighbors, (2) *outlier*, which has lower density than its neighbors and below a predefined threshold, and (3) the *rest*. In the next step, the Local Approximation of fuzzy membership (initialised to random values) of the three groups is determined and each object in group 3 (*rest*) is updated by a linear combination of the fuzzy memberships of its nearest neighbors. The degree of membership of vector x in cluster i is $p_i(x)$ and is given by:

$$x : p(x) = (p_1(x), p_2(x), \dots, p_M(x)), \quad (6)$$

where $0 \leq p_i(x) \leq 1$; $\sum_{i=1}^M p_i(x) = 1$ and M is the total number of clusters: $M = |X_{\text{CSO}}| + 1$, where X_{CSO} is the set of cluster supporting objects with Local Maximum Density, and 1 represents the outlier cluster.

The $p(x)$ is represented by a membership vector of each object x and is updated by the weighted summation of membership vectors of all its nearest neighbors, as in Eq. 7.

$$p^{t+1}(x) \approx \sum_{y \in \text{KNN}(x)} w_{xy} p^t(y) \quad (7)$$

where $p^t(x)$ is a fuzzy membership vector of object x at iteration t and w_{xy} are the coefficients reflecting relative proximities of nearest neighbours such that the sum of the coefficients over the $\text{KNN}(x)$ is equal to 1 and KNN is the k nearest neighbors of x .

Basically, each iteration process attempts to reduce the Local (Neighborhood) Approximation Error E [24] which is the difference between the approximation of membership vectors in the current and previous iterations, defined by:

$$E(\{p\}) = \sum_{\forall x} \left\| p(x) - \sum_{y \in \text{KNN}(x)} w_{xy} p(y) \right\|^2 \quad (8)$$

Finally, based on fuzzy memberships the clusters can be represented in two ways: one gene to one cluster or one gene to multiple clusters. FLAME results are given as line plots showing three patterns: (1) each cluster which includes *CSO and the rest*, (2) *all outliers*, and (3) *all CSOs*. The most important is the type (1) which shows expression patterns for each cluster in a separate line plot. FLAME is integrated with Gene Expression Data Analysis Studio (GEDAS) (<http://sourceforge.net/projects/gedas>).

3.4 Fuzzy C-Means (FCM)

FCM is the most widely used fuzzy clustering algorithm; it was developed by Dunn in 1973 [32] and improved by Bezdek in 1981 [33]. It is based on the minimization of the following objective function:

$$J = \sum_{j=1}^c J_j = \sum_{i=1}^c \sum_{k, x_k \in c_i} u_{ki}^m d(x_k - c_i), 1 \leq m < \infty \quad (9)$$

where c_i is the centroid of cluster i , $d(x_k - c_i)$ is the distance between i^{th} centroid (c_i) and k^{th} input vector x_k , c is the total number of clusters, m is a fuzziness parameter which can be any real number greater than 1, and u_{ki} is the degree of membership of x_k in cluster i .

FCM is a clustering method that allows data to belong to more than one cluster with a degree membership u_{ij} . The FCM algorithm starts with a predefined number of clusters where each datum has an equal degree of belonging to every cluster. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with updates of membership u_{ij} and cluster centers c_j . The iteration stops when a termination criterion is met. FCM is integrated with Gene Expression Data Analysis Studio (GEDAS).

Each of the selected methods allows the user to adjust parameters. We set parameters for each method as follows: SOM based on correlation distance and batch learning was conducted on Matlab (<http://www.mathworks.com/>) Neural Networks toolbox followed by an in-house developed algorithm implementing Ward clustering to define clusters. ESOM based on correlation distance

used online (example-by-example) learning while growing map size to 50×82 in 20 batch runs; linear functions were used for both shrinking the neighborhood size and learning rate. FLAME used Pearson or Squared Pearson Correlation with 10K-Nearest Neighbors with no threshold. FCM used Pearson or Squared Pearson Correlation distance with fuzziness parameter (m) of 1.2. This fuzziness parameter was calculated by using an empirical equation from Dembele and Kastner [34]. They proposed that the value of m should be lower than or equal to 2 in order to get gene members with strong relationships to clusters. Ozkan and Turksen [35] also mention that level of fuzziness is essential and it can affect FCM membership values.

4 Results and Discussion

4.1 Gene Clustering Based on the 71 and 93 Genes by SOM, ESOM, FCM, and FLAME

In this section, the results from clustering the small subsets of 71 and 93 genes (extracted from the two intrinsic gene sets of 456 and 534, respectively) using SOM, ESOM, FLAME, and FCM are presented, highlighting how these two small gene subsets support the identified cancer subtypes by assessing their clustering patterns when subjected to these clustering algorithms. As highlighted earlier, the 71-gene subset from the 456 genes represented four subtypes: HER2 (refers to ERBB2+/HER2), Basal-like, Normal (i.e., Normal breast-like) and Luminal. The 93-gene subset from the 534 genes represented five subtypes: HER2, Basal-like, Normal, Luminal A, and Luminal B.

SOM was developed and the map was clustered using the Ward method to find optimum clusters. Results of clusters and corresponding Ward index are shown in Fig. 3. Maximum Ward likelihood index for both datasets as evidenced by the largest Ward likelihood index indicates that the optimum number of clusters in each case is three. Table 3 shows details of each SOM-based gene cluster in terms of the number of genes from the small gene subsets of 71 or 93 represented in each cluster against the number of genes in the original subtypes.

The fact that both gene sets resulted in three clusters indicates the strong likelihood that not all five subtypes are distinct. The members in each SOM cluster for the first dataset (Table 3a) represent two or more subtypes. Cluster 1 is divided between Luminal and Normal breast-like (hereafter referred to as either Normal-like or Normal), which implies that Luminal and Normal-like have some similarity in gene expression patterns. In contrast, the 93-gene subset extracted from the 546 “intrinsic” genes shows clear separation of Luminal A into cluster 1 (Table 3b). However, Table 3b indicates a weak similarity between Basal-like and Normal-like in its cluster 2 which is predominantly Basal-like. Similarly, cluster 2 in Table 3a is predominantly Basal (13/19) with some Normal-like (3/14) cases and one of HER2 (1/11) subtype

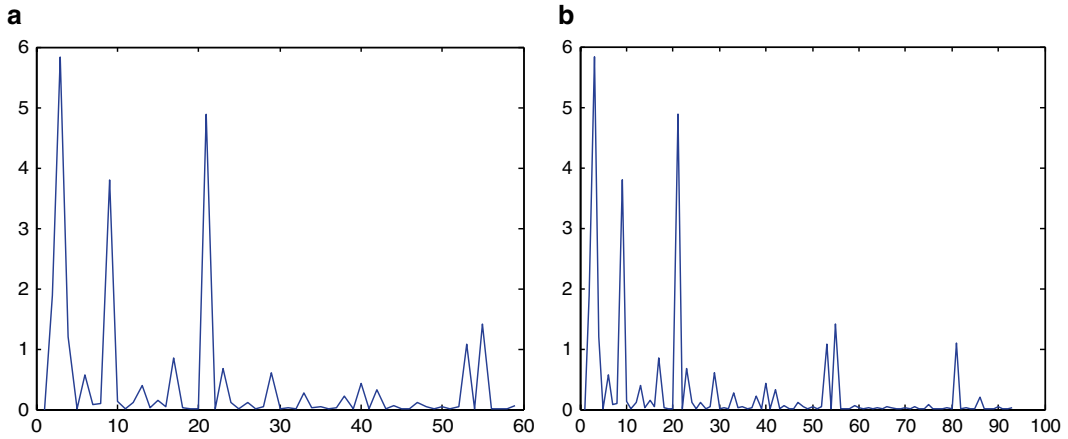


Fig. 3 Ward likelihood index for various number of clusters (a) for 71 genes from 456 “intrinsic” gene set (b) for 93 genes from 546 “intrinsic” gene set ((x-axis—number of clusters and y-axis—Ward likelihood index). The larger the likelihood index, more likely that number of clusters

Table 3

Proportion of genes belonging to the original subtypes found in each SOM cluster (numerator is the number of genes belonging to a particular subtype found in an SOM cluster and the denominator is the total number of genes belonging to that subtype)

Number of genes in original subtype/number of genes in SOM cluster	1	2	3
(a) Clustering based on the 71-gene subset extracted from the 456		“intrinsic” genes	
HER2 (11 genes)		1/11	10/11
Basal-like (19 genes)		13/19	6/19
Normal (14 genes)	11/14	3/14	
Luminal (15 genes)	14/15		1/15
(b) Clustering based on the 93-gene subset extracted from the 534		“intrinsic” genes	
HER2 (11 genes)			11/11
Basal-like (24 genes)		18/24	6/24
Normal (5 genes)		4/5	1/5
Luminal A (28 genes)	28/28		
Luminal B (25 genes)			25/25

supporting not only the possible similarity between Basal-like and Normal-like but also showing the potential to overlap with the HER2 subtype. Cluster 3 in Table 3a is dominated by HER2 (10/11) with a moderate presence of Basal-like (6/19) and weak presence of Luminal (1/5). Cluster 3 in Table 3b also has all HER2 genes (11/11) but it also has all the Luminal B gene activity patterns indicating a strong similarity between HER2 and Luminal

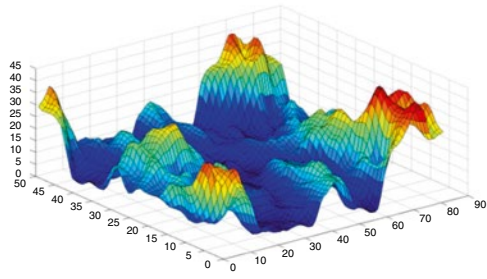
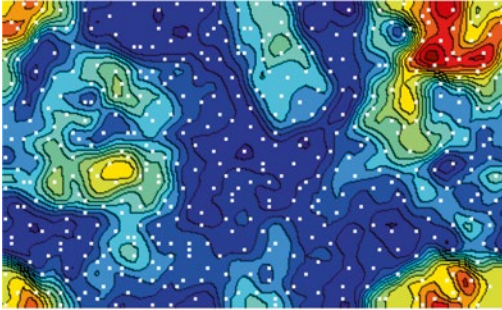
B. The rest in this cluster are from Basal-like and Normal-like whose similarity was already revealed in cluster 2 of Table 3a, b. Overall, Luminal A appears distinct from Luminal B and others, Basal and Normal-like show overlap, and HER 2 shares similarities with Basal-like and Luminal B.

ESOM results, given in Fig. 4, show information on the gene density for the two intrinsic gene sets. The red color represents dense areas while the lighter color represents sparse areas. The dots indicate neurons on the map. In this figure, the toroidal shape of the map has been converted to a rectangular shape for visualization purposes but one can visualize the toroid by bending the corners appropriately. This map also shows three dense areas (red to yellow areas in the corners and within) for both gene sets. A further clustering step was attempted but clusters seemed unstable in multiple trials and therefore, the ESOM results are shown here mainly to present a visualization of the characteristics of data density, not for clustering purposes. However, we use the results to highlight where the genes for three subtypes tend to cluster by investigating the best match ESOM vector to each gene expression vector (Fig. 5) for the case of 93-gene subset from the 534 intrinsic genes. For example, Basal-like cluster has all its genes in the same location while HER2 has all but SMARCE1 in the same location. Normal-like cluster is close to the Basal-like and has all its genes in the same location except for KIAA0857 and MEN1.

The last two clustering methods (FLAME and FCM) provide cluster information which can be used to compare with the original clusters and with SOM clusters. In doing so, the focus is on investigating the efficacy of the gene subsets in distinguishing the subtypes as found by the original authors by analyzing if each subset is clustered separately. We present in Table 4 the results of clustering tumor subtypes using the two sets of 71 and 93 genes by the three methods (SOM, FLAME, and FCM) and compare with the clusters from the original study. Recall that SOM produced three optimum clusters as shown in Table 3. FLAME and FCM used squared Pearson correlation. Squared Pearson captures the similarity between profiles in terms of trends irrespective of whether upregulated or downregulated. These can help identify co-regulated genes which may be either upregulated or downregulated in a related fashion.

FLAME finds the optimum number of clusters by itself and produced four clusters for the 71 genes, similar to the number of original subtypes, as shown in Table 4. FCM requires the number of clusters to be predefined so the same number of clusters as the original subtypes was given to FCM in order to compare their results. In Table 4, cluster results are presented according to the number label given by the computational method to each specific cluster followed by a number within brackets that indicates the number of genes in the original subset for each subtype found in the computed clusters. The question we ask is: Do the clusters

ESOM: 71 genes from 456 intrinsic genes



ESOM: 93 genes from 534 intrinsic genes

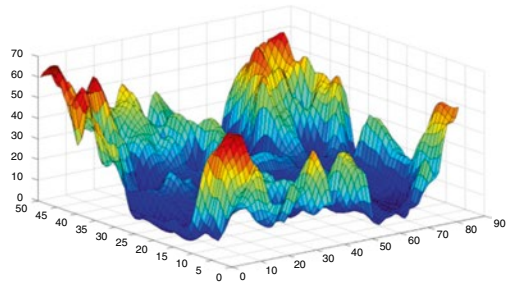
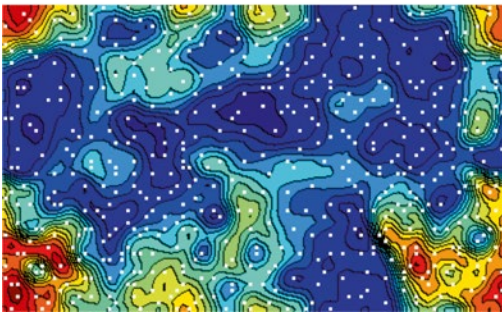


Fig. 4 ESOM: P-Matrix (density-based) and 3D view of P-Matrix for the two datasets. *White dots* represent best match (closest) vector to each data point. *Red* indicates higher density and *blue* indicates smaller density of input vectors

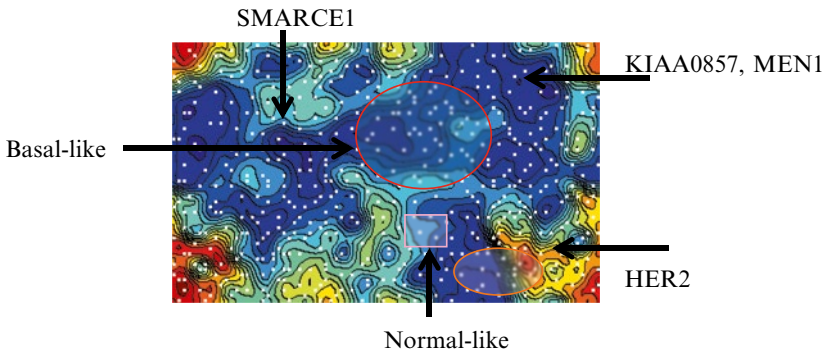


Fig. 5 Mapping of three cancer subtypes (found by original authors from the 93-gene subset) onto ESOM

support the gene sets identified as representing the cancer subtypes? Results indicate that for the 71-gene set, FCM put all four subsets of genes (subtypes) into four distinct clusters. FLAME put all but Basal-like into distinct clusters. Even in the case of Basal-like, most (68.4 %) are in one cluster (Cluster 5) and the rest shared by Luminal (26.34 %) and Normal-like (5.26 %), further supporting

Table 4

Clustering results from the three methods: SOM, FLAME, and FCM using 71 and 93 genes (number in brackets refers to number of genes from the original subtypes from Sorlie et al. [7, 8])

Subtype	Subtype genes (original study)	SOM	FLAME	FCM
71 genes extracted from 456 intrinsic genes Sorlie et al. [7]				
HER2 (11)	ERBB2, ESTs T57034, KIAA0130, ERBB2, steroidogenic, GRB7, TGFB1, TNF, flotillin 2, FLJ10700, SWI/SNF	Cluster 2 (1)+ cluster 3 (10)	Cluster 3 (11)	Cluster 3 (11)
Basal-like (19)	SRY, UDP-N, ESTs W93120, CDH3, laminin, small inducible cytokine subfamily, ATDC, KRT17, KRT 5, troponin I, CHI3L2, SLPI, nuclear factor I/B, ESTs AI304356, transforming growth factor beta 2, calpain-like protease, dystrophin, fatty acid binding protein 7, GRO1	Cluster 2 (13)+ cluster 3 (6)	Cluster 0 (5)+ cluster 2 (1)+ cluster 5 (13)	Cluster 0 (19)
Normal (14)	CD36, CD36, glutathione peroxidase 3, glycerol-3-phosphate dehydrogenase 1, lipoprotein lipase, ESTs T62068, four and a half LIM domains 1, retinol-binding protein 4, amine oxidase, integrin, alpha 7, alcohol dehydrogenase 2 (class I), MY047 protein, aquaporin 7, 30 kDa protein	Cluster 1 (11)+ cluster 2 (3)	Cluster 2 (14)	Cluster 5 (14)
Luminal (15)	Putative G, acyl-Coenzyme A, ERS1, TFF3, GATA3, GATA3, XBPI, HNF3A, lymphoid nuclear protein, LIV1, ESTs AA029948, FLJ 11280, N-acetyltransferase, Myosin VI, Myosin VI	Cluster 1 (14)+ cluster 3 (1)	Cluster 0 (15)	Cluster 2 (15)
93 genes extracted from 534 intrinsic genes Sorlie et al. [8]				
HER2+ (11)	FLJ10700, FLOT2, SMARCE1, TLK1, TRAP100, GRB7, PPARBP, ERBB2, ERBB2, Homo sapiens mRNA, TBPL1	Cluster 3 (11)	Cluster 2 (11)	Cluster 0 (11)
Basal-like (24)	CXCL1, CDH3, ANXA8, KRT5, TRIM29, KRT17, MFGE8, CX3CL1, FZD7, CHI3L2, B3GNT5, EXT2, FLJ10697, FLJ31360, FLJ 14761, SLPI, EST, TONDU, GABRP, ZDHHC5, FLJ 11796, SLC5A6, GABRP, FLJ 11796	Cluster 2 (18)+ cluster 3 (6)	Cluster 1 (24)	Cluster 1 (8)+cluster 2 (15)+cluster 3 (1)

(continued)

Table 4
(continued)

Subtype	Subtype genes (original study)	SOM	FLAME	FCM
Normal (5)	KIAA0857, MEN1, PIK3R1, AKR1C1, FACL2	Cluster 2 (4)+ cluster 3 (1)	Cluster 1 (3)+ outlier (2)	Cluster 2 (5)
Luminal subtype A (28)	LRBA, Homo sapiens mRNA, LIV-1, HNF3A, XBP1, GATA3, ESRI, PTP4A2, RERG, ACADSB, GATA3, VAV3, KIAA1243, LOC51313, DKFZp, KIA0876, FLJ 10980, TLE3, MGC 22588, CEGP1, FBP1, MGC 27171, HSD 17B4, Homo sapiens Mrna, Homo sapiens mRNA, NAT1, Homo sapiens, FLT1	Cluster 1 (28)	Cluster 3 (28)	Cluster 2 (7)+ cluster 3 (1)+ cluster 4 (20)
Luminal subtype B (25)	ABCD3, ADSL, BTG3, ATP5G1, D123, PRNPIP, EBNA1BP2, NSEP1, GGH, LC27, PRDX4, HSPC163, GARS, FLJ12442, TMSB10, KDELR2, KIAA1691, NRBF-2, CCNE1, CALU, LANP, POLR2F, SQLE, S100A10	Cluster 3 (25)	Cluster 4 (25)	Cluster 3 (24)+ cluster 4 (1)

the evidence for some similarity between these three subtypes. As discussed before, SOM (i.e., SOM/Ward) produced only three clusters and its performance overall is weaker than that for FLAME and FCM. The SOM puts 93 % Luminal and 78 % Normal-like in one cluster, Normal, Basal-like and HER2 in one cluster, and HER2, Basal-like, and Luminal in another cluster, thereby splitting the original subtype genes as shown clearly in Table 3a. Closer inspection reveals that SOM cluster 3 is predominantly HER2 with 91 % (10/11) HER2 genes and SOM cluster 2 is predominantly Basal-like with 68 % (13/19) Basal-like genes in it (Table 3a). Thus, although SOM clusters correspond with some subtypes, they also share genes from other subtypes; especially, in the case of Normal, Luminal, and Basal-like. The largest split was for Basal and FLAME also has difficulty demarcating the Basal-like subtype.

To complement results for the 93-gene subset in Table 4 and to clarify the results better, the same results are produced in summary form in Table 5 to shed more light on the efficacy of this 93-gene set extracted from the 534 intrinsic genes (note that these 534 genes were found from an extension of the study that produced the 456 genes) in identifying the five subtypes. Specifically, Table 5 indicates the percentage of the genes defining the original subtypes found in each cluster from SOM, FLAME, and FCM. Results in Table 5b show that FLAME produced four optimum clusters and

all but Normal-like and Basal-like genes were put into distinct clusters. Specifically, some (60 %) Normal-like genes were clustered with Basal-like and the rest of the Normal-like genes were considered as outliers (see Table 5b). Note though that the Normal-like subset now has only five genes and this may have affected the results. SOM (Table 5a) dedicated one cluster (cluster 1) to Luminal A; and another (cluster 2) to Basal-like and Normal-like genes but put roughly 25 % of genes in these subtypes in cluster 3 where all Luminal B and HER2 genes have also been placed. Thus cluster 3 represents four subtypes with the majority (58 %) of the cluster being genes from Luminal B followed by HER2 (26 %), Basal-like (14 %) and only 2 % of Normal-like genes. Overall, FLAME shows more definite support for original subtypes than SOM with Lum A, Lum B, and HER2 in separate clusters and Basal and Normal-like sharing one cluster. This shows that the FLAME's choice of four clusters as opposed to the original authors' five subtypes is based on

Table 5
Percentage of genes from each subtype found in each cluster of (a) SOM/Ward, (b) FLAME, and (c) FCM based on the 93-gene subset

	Cluster 0 (%)	Cluster 1 (%)	Cluster 2 (%)	Cluster 3 (%)	Cluster 4 (%)
(a) SOM/Ward					
LumA (28)		100			
LumB (25)				100	
HER2 (11)				100	
Basal (24)			75	25	
Normal (5)			80	20	
(b) FLAME					
LumA (28)				100	
LumB (25)					100
HER2 (11)		100			
Basal (24)			100		
Normal (5) (outlier 40 %)			60		
(c) FCM					
LumA (28)			25	3.5	71.5
LumB (25)				96	4
HER2 (11)	100				
Basal (24)		33.34	62.5	4.16	
Normal (5)			100		

the similarity between Basal-like and Normal-like subtypes that was persistently shown across all results in this study.

FCM was used with five clusters to match the original number of subtypes. It puts two gene subsets into distinct clusters (all HER2 genes in cluster 0 and almost all Luminal B genes in cluster 3 and over 70 % of Luminal A genes in cluster 4 (Table 5c)). However, it puts all Normal-like genes with the majority of Basal-like genes in cluster 2 which also contain 25 % of Luminal A genes (Table 5c). This again indicates some overlap between Basal-like, Normal-like, and Luminal A subtypes. Interestingly, some 33 % of Basal-like genes have gone into a separate cluster (cluster 1). The FCM summary results in Table 5c show complete support for HER2 subtype and strong support for LumB subtype with most of the latter's genes in cluster 3. All other subtypes are spread across one or more clusters (e.g., LumA and Basal-like). Overall results from both analyses (71 and 93 genes) show FLAME with strong support for the identified subtypes followed by FCM. Support from SOM is weak in both cases.

Similarity or distance measure used can affect the gene membership in each cluster. Therefore, selecting clustering tools with a priori knowledge of the expected characteristics of genes will enhance the confidence in and accuracy of results. To assess the influence of the distance measure used, we conducted another introspective study with only the 93-gene subset extracted from the 534 genes found by Sorlie et al. [8]. Specifically, FLAME and FCM are used here with Pearson or squared Pearson correlation as the distance measure and the results are shown in Table 6.

FLAME does not need a predefined number of clusters and identified only four clusters from both distance measures. FCM needs to predefine the cluster number, so two options are studied where five is given according to known subtypes from the original authors' work and four is given in order to compare with FLAME clusters. Entries in this table indicate the cluster number (1–4 or 0–3) followed by two numbers within brackets: the number of genes in the original authors' subtype found in this cluster (numerator) and the total number of genes in the cluster (denominator). Table 6 shows that the distance measure and the defined number of clusters (for FCM) have an effect on how genes are clustered. Pearson gives better cluster results for both FLAME and FCM (four clusters) than squared Pearson correlation.

Four clusters from FLAME and FCM based on Pearson correlation are the most similar to the original group; an exception is that both FLAME and FCM define Basal-like gene expression patterns similar to those for Normal-like subtype (cluster 1 for FLAME and cluster 2 for FCM), an observation previously made from our SOM, FLAME, and FCM results. In fact, 4- (as well as 5-) cluster FCM based on Pearson correlation indicated the similarity between these two subtypes (cluster 2 contains all five Normal-like and most or a large number of Basal-like genes). These agree with the

Table 6
Comparison of clustering results from FLAME and FCM using only the 93-gene subset from the 534 intrinsic genes with Pearson and Squared Pearson correlation coefficient as the distance measure

Subtype	FCM					
	FLAME	Pearson 4 clusters	Squared Pearson 4 clusters	Pearson 5 clusters	Pearson 4 clusters	Squared Pearson 5 clusters
HER2 (11)	Cluster 2 (11/11)	Cluster 2 (11/11)	Cluster 2 (11/11)	Cluster 0 (11/11)	Cluster 0 (11/11)	Cluster 0 (11/11)
Basal-like (24)	Cluster 1 (24/27)	Cluster 0 (13/46)+ cluster 3 (11/12)	Cluster 1 (8/8)+ cluster 2 (15/27)+ cluster 3 (1/26)	Cluster 2 (23/28)+ cluster 3 (1/26)	Cluster 2 (23/27)+ cluster 4 (1/21)	Cluster 2 (23/27)+ cluster 4 (1/21)
Normal (5)	Cluster 1 (3/27)+ outlier (2/2)	Cluster 0 (3/46)+ cluster 3 (1/12)+ outlier (1/1)	Cluster 2 (5/27)	Cluster 2 (5/28)	Cluster 2 (23/27)+ Cluster 4 (1/21)	Cluster 2 (23/27)+ Cluster 4 (1/21)
Luminal subtype A (28)	Cluster 3 (28/28)	Cluster 0 (28/46)	Cluster 2 (7/27)+ cluster 3 (1/26)+ cluster 4 (20/21)	Cluster 1 (28/28)	Cluster 1 (8/9)+ cluster 4 (20/21)	Cluster 1 (8/9)+ cluster 4 (20/21)
Luminal subtype B (25)	Cluster 4 (25/25)	Cluster 0(2/46)+cluster 1 (23/23)	Cluster 3 (24/26)+ cluster 4 (1/21)	Cluster 3 (25/26)	Cluster 3 (25/25)	Cluster 3 (25/25)

The numbers within brackets are genes from the original author's [8] subtype found in each cluster (numerator) and the total number of genes in that cluster (denominator)

results from hierarchical clustering where Normal and Basal-like are close to each other with similar gene expression patterns (Fig. 1). FLAME identified that two genes, KIAA0857 and PIK3R1, were distinct from the others in the Normal-like cluster and classified them as outliers. In fact, ESOM also put KIAA0857 separately from the rest of the genes in the Normal-like cluster (Fig. 5). Both FLAME and FCM with four clusters clearly separates Luminal A and Luminal B into distinct clusters; but if pressed for five clusters, FCM puts some Luminal A genes with Basal-like and one with Luminal B (see clusters 2 and 3 in Luminal A) and splits Basal-like genes even further than before. Whether it is four or five clusters, HER2 genes make a distinct cluster, indicating the efficacy of these genes in characterizing this subtype.

Comparing FCM 5 clusters based on Pearson correlation and Squared Pearson correlation, HER2 cluster is still intact, Luminal B separates into a distinct cluster (marginal improvement), and Basal-like cluster has become clearly more distinct separating very much into one cluster (a large improvement). Luminal A cluster has become very slightly more distinct, and Normal-like has slightly deteriorated. With FLAME, squared Pearson correlation has deteriorated the clustering results. Thus, overall, four clusters based on Pearson correlation seem to affirm the efficacy of the gene subsets in distinguishing breast cancer subtypes with FLAME producing more accurate clustering outcomes.

4.2 Classification of Subtypes: Clustering Patients Using SOM and FCM

Another useful investigation that can shed light on the selected gene subsets is clustering patients. Therefore, we conducted an SOM and FCM cluster analysis of patients based on the 93-gene subset extracted from the 534 intrinsic genes [8] and the 71 subset of genes extracted from the 306 intrinsic genes [27] (see Table 2, last column). The latter set has been proposed from a combined dataset from three sources including the one that produced the above 534 intrinsic genes. The results from the previous section show that each clustering method produces one or more clusters that contain genes from one or more subtypes. Fuzzy clustering is a soft clustering approach that allows a measure of the strength of membership of an item in each cluster and we utilize this aspect of FCM more in this patient clustering.

4.2.1 SOM Clustering of Patients

Gene expression vectors, each containing expression results for all 93 genes (from the 534 intrinsic genes) for each of the 79 patients were clustered using SOM and results are shown in Fig. 6. The first map Fig. 6a (left) indicates the number of patients (hits) represented by each neuron, and Fig. 6a (right) shows the average distance between neurons where lighter color indicates smaller distances and darker color indicates larger distances between neurons. Such a distance map allows visualization of potential clusters. Dark regions indicating larger gaps appear at least in three places and these separate the samples into at least three groups.

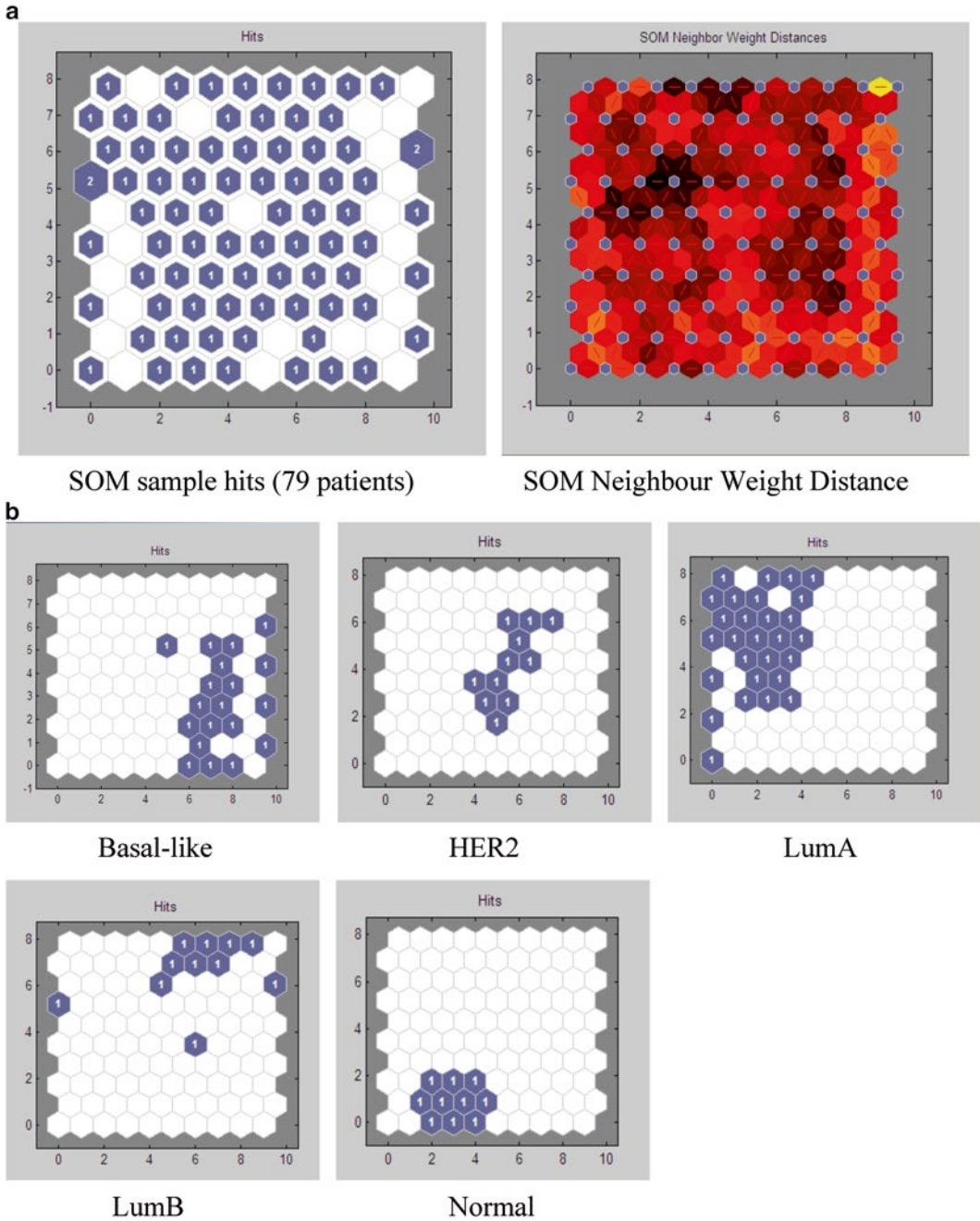


Fig. 6 (a) Projection of all patients (subtypes) on to the trained SOM based on the 93-gene subset (*left*), and U-matrix (*right*). (b) SOM sample hits (patients) for each “intrinsic” subtype according to the 93-gene subset from the original study (79 patients in total)

In the last five panels (Fig. 6b), the actual expression pattern for patients from each original subtype is projected onto the trained map. A powerful feature of SOM is topology preservation where data (expression patterns; in this case patients) that are biologically

closer are organized closer in the map allowing one to see not only clusters but also biological similarity between different clusters (subtypes). The SOM panels for the five subtypes show that most patients of each subtype are organized in the same region; for example, Normal-like patients are located at the bottom left-hand corner of the Map. Moreover, Normal-like is located opposite to LumB; Basal-like is located opposite to Luminal A; and HER2 is closer to Normal-like and Basal-like in that these three subtypes together occupy a triangular region in the bottom right part of the map. LumA and LumB occupy the top left triangle on the map. This corresponds to classification of breast cancer molecular subtypes based on the estrogen receptor (ER) status: ER-positive consists of two main subtypes (Luminal A and B) and ER-negative consists of three subtypes (Her-2, Basal-like, and Normal-like).

Next, we investigated whether the 71-gene subset extracted from the 306 intrinsic gene set provides better accuracy in classification of subtypes; results are shown in Fig. 7. There are 248 patients in this dataset. SOM produced similar results to those with the previous dataset indicating similar relative positioning of subtypes in the SOM panels. For example, Normal-like, Basal-like, and HER2 subtypes occupy the top left region of the map and LumA and LumB occupy the bottom left region. This reversal of space occupied by subsets is not an issue as relative position is what gives a map its meaning. In this case, different intrinsic genes do not vary the final SOM result.

In an attempt to find final clusters on SOM, Ward clustering of SOM neurons provided the Ward index vs. number of clusters relation for the 93 and 71 gene signatures as shown in Fig. 8. The optimum number of clusters for these two signatures is four and two clusters, respectively. This indicates that the 93 gene signature has greater discriminating ability than the 71 gene signature that sees more similarity between the subtypes leading to only two optimum clusters. A summary of clustering results for the two gene sets are given in Table 7 which indicates that 93 gene signature distinguished the 28 LumA patients but put some Lum B patients with them in the same cluster. It also separated most (90 %) of Normal-like and Basal-like into separate clusters. It sees all HER2 patients together in a cluster but it is shared with 60 % of LumB patients and a small number of Basal-like and Normal-like patients. The 71-gene subset in contrast cannot distinguish between LumA and B; identifies all 64 Basal-like patients in one cluster but it shares over 50 % of Normal-like and HER2 patients. The rest of the Normal-like and HER2 patients are clustered with LumA and B. Clearly, the 93 gene signature shows greater discriminating ability in this case.

4.2.2 FCM Clustering of Patients

In this section, we cluster the patients based on the 93 and 71 gene signatures using FCM in order to ascertain the efficacy of this clustering method on the selected datasets. Results are shown in

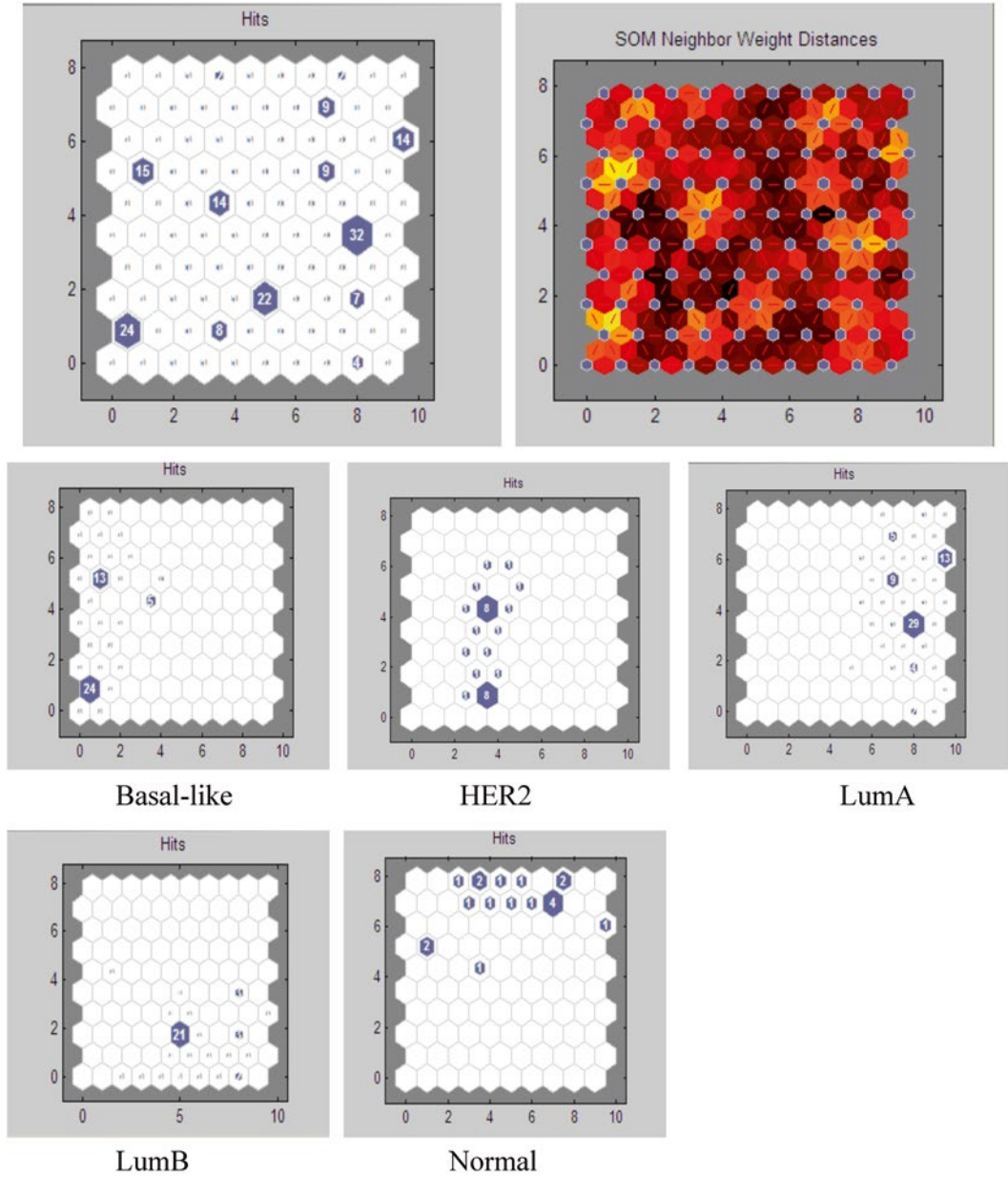


Fig. 7 SOM sample hits (patients) for each “intrinsic” subtype according to the small subset of 71 genes from the 306 “intrinsic” genes (248 patients in total)

Table 8 where a and b sections are for the 93 gene signature and c and d parts are for the 71 gene signature. Table 8a, c show original samples (patient IDs), FCM clusters, and samples in each of these clusters. In Table 8b, d, FCM clusters are compared with the original classification. For the 93-gene set, one FCM cluster has 100 % members belonging to Normal-like group while other clusters contain one or more subtypes. Cluster 1 has only LumA patients

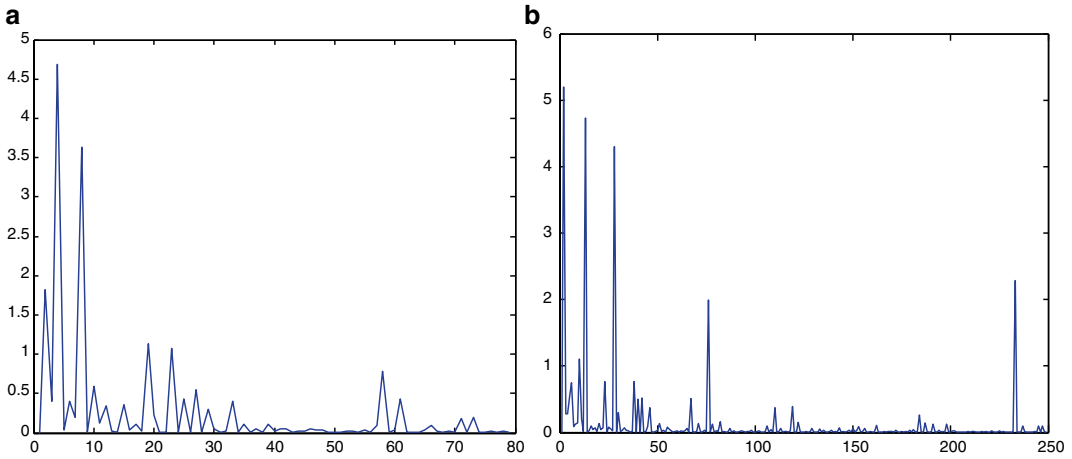


Fig. 8 Ward likelihood index for various number of clusters of map neurons **(a)** for the 93 subset extracted from the 546 “intrinsic” genes **(b)** for the 71-gene subset extracted from the 306 “intrinsic” genes (*x*-axis indicate the number of clusters and *y*-axis-Ward likelihood index). The larger the likelihood index, more likely that number of clusters

Table 7

Members (patients) in each SOM cluster according to subtype based on: (a) 93-gene subset and (b) 71-gene subset

Patient cluster	1	2	3	4
(a) 93-gene subset from the 546 “intrinsic” genes				
HER2 (11 patients)				11
Basal-like (19 patients)			17/19	2/19
Normal (10 patients)		9/10		1/10
Luminal A (28 patients)	28/28			
Luminal B (11 patients)	4/11		1/11	6/11
(b) 71-gene subset from the 306 “intrinsic” genes				
HER2 (29 patients)	15/29	14/29		
Basal-like (64 patients)	64/64			
Normal (19 patients)	11/19	8/19		
Luminal A (90 patients)		90/90		
Luminal B (46 patients)		46/46		

but contain only 57 % of them. Similarly, with 71 gene signature, each cluster consists of two or more subtypes as shown in Table 8d. It appears that the 93 gene signature based on the original 534 gene list enables more accurate clustering than the 71-gene subset from the 306 gene signature. To be able to see clearly how patients are grouped into “intrinsic” subtypes, we projected the FCM sample cluster results from the 93 gene signature onto our SOM in Fig. 6 and results are shown in Fig. 9. It shows that Normal-like is distinct (cluster 5), Basal-like is almost distinct (cluster 4) with an

Table 8

FCM (with 1.2 degree of fuzziness) classification results for two selected datasets: (a) and (b) for 93 genes from the 534 “intrinsic” genes; (c and d) for 71 genes from the 306 “intrinsic” genes

(a) Sample allocation from FCM (based on 93-gene subset from the 534 “intrinsic” genes)			
Subtypes	Sorlie et al. [8]	FCM	
LumA (28)	Sample : 1–28	Cluster 1	Samples: 1:5,7:8,10:12,14,23:26,28
LumB (11)	Sample : 29–39	Cluster 2	Samples: 6,9,13,15:22,27,29,60
HER2 (11)	Sample : 40–50	Cluster 3	Samples : 30:36,37,39:50
Basal (19)	Sample : 51–69	Cluster 4	Samples: 38,51:59,61:69
Normal (10)	Sample : 70–79	Cluster 5	Samples: 70–79

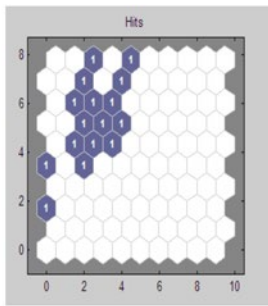
(b) Percentage of patients found in each cluster according to subtype (based on 93 genes from 534 “intrinsic” genes)					
	Cluster 1 (%)	Cluster 2 (%)	Cluster 3 (%)	Cluster 4 (%)	Cluster 5 (%)
LumA (28)	57.15	42.85			
LumB (11)		9.09	81.82	9.09	
HER2 (11)			100		
Basal (19)		5.27		94.73	
Normal (10)					100

(c) Sample allocation from FCM (based on 71 genes from 306 “intrinsic” genes)			
Subtypes	Hu et al. [27]	FCM	
LumA (90)	Sample : 1–46	Cluster 1	Samples: 1:2,7,10:11,13:17,24,26:27,31,42,45,122,138,144:145,147,149,160,162,213:219,221–241,243:249
LumB (46)	Sample : 46–136	Cluster 2	Samples: 3:6,8:9,12,18:23,25,28:30,32:41,43:44,46:47,49:50,53,57:58,60:62,70,72:76,78,85,102:105,109,112:113,126,134,136,242
Normal (29)	Sample : 137–155	Cluster 3	Samples: 48,51:52,54:56,59,63:69,71,77,79:84,86:101,106:108,110:111,114:121,123:125,127:133,135,13,139:140,142:143, 148,150:155
Basal (64)	Sample : 156–220	Cluster 4	Samples: 141,146,156:159,161,163:212,220
HER2 (19)	Sample : 221–249		

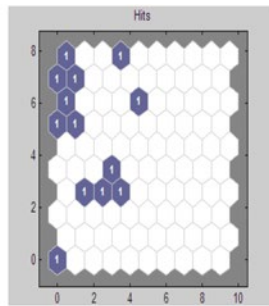
(continued)

Table 8
(continued)

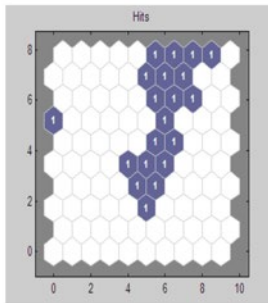
(d) Percentage of patients found in each cluster according to subtype (based on 71 genes from 306 “intrinsic” genes)				
	Cluster 1 (%)	Cluster 2 (%)	Cluster 3 (%)	Cluster 4 (%)
LumA (90)	35	65		
LumB (46)	1	33	66	
Normal (29)	26		64	10
Basal (64)	14			86
HER2 (19)	97	3		



Cluster 1 (**LumA** 16)



Cluster 2 (**LumA** (12)+ LumB (1)+ Basal (1))



Cluster 3 (**LumB** (9)+**HER2** (11)) Cluster 4 (LumB (1)+ **Basal** (18)) Cluster 5 (**Normal** 10)

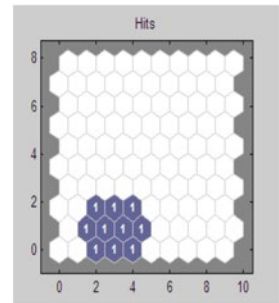
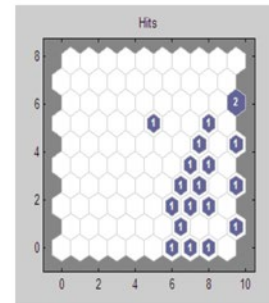


Fig. 9 SOM sample hits for each FCM cluster according to the subset of 93 genes from 546 intrinsic genes

odd Lum B patient showing in a distance, all HER2 patients in cluster 3 with the majority in the middle region of the cluster. The top region of this cluster contains all LumB patients. LumA is divided into Cluster 1 and Cluster 2 but overlaying cluster 1 and 2 maps indicate that LumA occupies the top left corner of the map. What we see here is that Normal-like, Basal-like, and HER2 in that order occupy the lower diagonal region of the map. Lum A and Lum B are on the opposite side of the map.

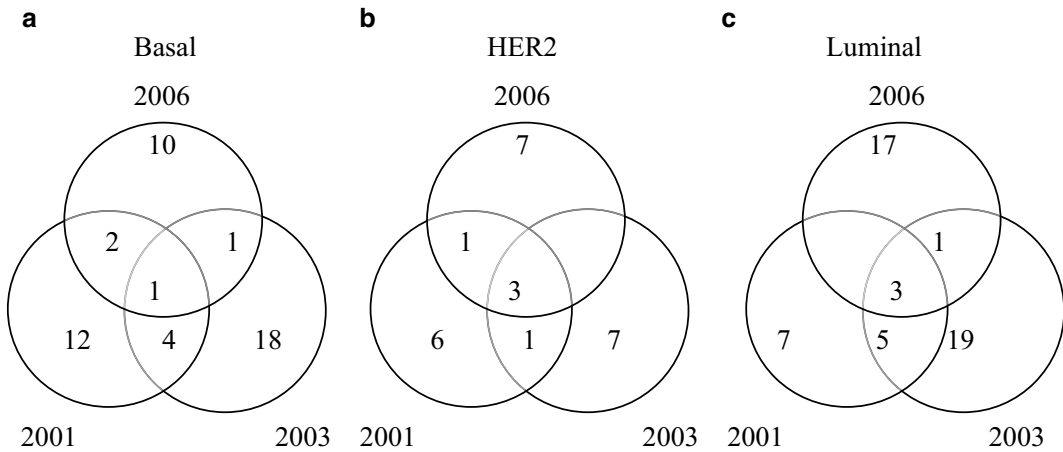


Fig. 10 Venn diagram of original gene clusters: **(a)** Basal-like, **(b)** HER2, and **(c)** Luminal using the small gene subsets from Sorlie et al. [7, 8] and Hu et al. [27]

We also tested the number of common genes among the subsets from the three studies, Sorlie et al. [7, 8] (2001 and 2003) and Hu et al. [27] (2006) for three subtypes. As shown in Fig. 10, there were very few common genes—1, 3, and 3 genes respectively for Basal-like, HER2, and Luminal. The lack of agreement between the reported genes continues to pose challenges in identifying the minimum intrinsic gene subset for distinguishing breast cancer subtypes.

5 Summary and Conclusions

In this study, an in-depth investigation of the efficacy of three novel gene sets reported in the literature in distinguishing breast cancer subtypes was performed using three emergent advanced computational methods. Selecting computational methods to analyze experimental data is still a challenging task as there is a myriad of methods available. Using different methods and distance measures can produce apparently different results. A good understanding of the cluster structure of a dataset and what relations are used whether trends or magnitudes is essential in order to select the right tools and distance measurement. The three selected tools (SOM, FLAME, and FCM) were assessed based on the novel gene signatures proposed by Sorlie et al. [7, 8] and Hu et al. [27] to investigate the efficacy of 71 and 93 and 71 novel gene subsets, respectively, proposed by the authors. These small subsets were extracted by the respective authors from 456, 534, and 306 intrinsic genes found differentially expressed in their studies.

First, the two gene sets consisting of 71 and 93 genes characterizing 4 and 5 cancer subtypes, respectively, were analyzed using

the four methods: SOM, ESOM, FLAME, and FCM. SOM and ESOM are visualization methods for high dimensional data projected onto a two-dimensional map and they gave an overview of how genes are organized in terms of distance and density. More consistent clusters were obtained from FCM and FLAME. FLAME and FCM with fuzzy characteristics showed that some genes have expression patterns that characterize two or more subtypes.

We found some similarities and differences in clustering the selected breast cancer genes with different clustering tools. For the 93-gene subset from the 534 intrinsic genes, SOM gives three clusters: cluster 1 (LumA), cluster 2 (Basal and Normal-like), and cluster 3 (LumB, HER2, Basal, and Normal-like); FLAME produced four clusters: cluster 1 (HER2), cluster 2 (Basal and Normal-like), cluster 3 (LumA), cluster 4 (LumB); and FCM defined five clusters: cluster 1 (HER2), cluster 2 (Basal-like), cluster 3 (LumA, Basal, and Normal-like), cluster 4 (LumA, LumB, and Basal), cluster 5 (LumA and LumB). Thus FLAME clusters genes in almost the same way as the original results but has difficulty in separating gene expression patterns from Basal-like and Normal-like subtypes. FCM has more than one subtype in each cluster except for the one representing HER2 subtype. SOM also has more than one subtype in each cluster except for the one representing LumA. For the 71-gene subset from 456 intrinsic genes, FCM had perfect clustering with one subtype only in each cluster. Next best was FLAME with all but one cluster representing one subtype each. SOM had spread each subtype in two clusters. In summary, the 93-gene subset from 534 “intrinsic” genes gave better subtype classification when compared with the 71-gene subset from 456 “intrinsic” genes based on SOM and FLAME but vice versa for FCM which gave perfect subtype classification for the 71 gene subset. However, this 71-gene set represented only four subtypes whereas the 93-gene set represented five subtypes. Therefore, the 93-gene set can provide a further classification of Luminal into Luminal A and Luminal B.

Before patient clustering based on the Sorlie et al. [8] and Hu et al. [27] gene sets, we checked and found that there were only a few overlapping genes from the three different proposed subsets (Sorlie et al. [7, 8] and Hu et al. [27]) in representing the subtypes. When using the 93-gene subset from the 546 “intrinsic” genes, we retrieved better patient classification results from both SOM and FCM compared with the 71-gene subset from the 306 “intrinsic” genes [26]. In this regard, the main contribution is using SOM to show the relationship/similarity between subtypes through visualization. In this subtype classification, we first used SOM and investigated the spatial organization of original subtypes based on the 93 and 71-gene subsets. We found that both gene sets provide consistent results, revealing that each subtype occupies a distinct location on the map. For example, SOM sample hits (Figs. 6 and 7) show the opposite positioning of Basal-like and

Luminal A as well Normal-like and Luminal B; HER2 always appeared in a distinct central position but closer to Normal-like and Basal-like than either LumA or B. Furthermore, LumA and LumB together occupy the opposite diagonal area from that occupied by the combined HER2, Basal-like, and Normal-like subtypes. These and all clustering results indicated the similarity between the Normal-like and Basal-like subtypes.

We further clustered the SOM using the Ward method and found that the four optimum clusters produced by the 93-gene set are more similar to the original subtypes than the two clusters produced by the 71-gene set from the Hu et al. [27] study. Additionally, as a validation, we used FCM to cluster original gene expression profiles with 93 and 71 genes and projected the resulting patient clusters on to the SOM which showed that the 93-gene subset had more discriminating ability than the 71-gene set. Thus the proposed 93-gene set demonstrated its strength in differentiating subtypes to a reasonably high degree of accuracy. There is evidence that this subset can represent some subtypes such as HER2 well and do reasonably well for LumA and B, indicating that it has captured some essential genes. There still is overlap between some subtypes such as Normal-like and Basal-like.

Given the costly procedures involved in clinical studies, the proposed 93-gene set can be used for preliminary classification of breast cancer. The patient based SOM model can be used to map the gene signature of new patients to locate them on the SOM with respect to all subtypes to get a comprehensive view of the classification. This can be followed by a deeper investigation in light of the observations made in this study regarding overlapping subtypes. We used gene signatures generated from three comprehensive studies conducted in 2001, 2003 and 2006 with one feeding into the other. Our approach and results could be the base for further refining the gene signatures from later experiments and those potentially designed to separate as much as possible overlapping clusters as well as to maximally separate all clusters.

Acknowledgements

This research was funded by a Research Grant awarded by Lincoln University, New Zealand.

References

1. Carey LA (2010) Through a glass darkly: advances in understanding breast cancer biology, 2000–2010. *Clin Breast Cancer* 10(3):188–195
2. Stadler ZK, Come SE (2009) Review of gene-expression profiling and its clinical use in breast cancer. *Crit Rev Oncol Hematol* 69(1):1–11
3. Marchionni L et al (2008) Systematic review: gene expression profiling assays in early-stage breast cancer. *Ann Intern Med* 148(5): 358–369
4. Desmedt C, Ruiz-Garcia E, André F (2008) Gene expression predictors in breast cancer:

- current status, limitations and perspectives. *Eur J Cancer* 44:2714–2720
5. Uma RS, Rajkumar T (2007) DNA microarray and breast cancer – a review. *Int J Human Genet* 7(1):49–56
 6. Perou C et al (2000) Molecular portraits of human breast tumours. *Nature* 406(6797):747–752
 7. Sorlie T et al (2001) Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proc Natl Acad Sci* 98(19):10869–10874
 8. Sorlie T et al (2003) Repeated observation of breast tumor subtypes in independent gene expression data sets. *Proc Natl Acad Sci* 100(14):8418–8423
 9. Weigelt B et al (2005) Molecular portraits and 70-gene prognosis signature are preserved throughout the metastatic process of breast cancer. *Cancer Res* 65(20):9155–9158
 10. Henshall S (2005) Review of classification of human breast cancer using gene expression profiling as a component of the survival predictor algorithm. *Breast Cancer Online* 7(12)
 11. Loi S (2008) Review of comparison of gene expression profiles predicting progression in breast cancer patients treated with tamoxifen. *Breast Cancer Online* 11(12):null
 12. van't Veer LJ et al (2002) Gene expression profiling predicts clinical outcome of breast cancer. *Nature* 415(6871):530–536
 13. van de Vijver MJ et al (2002) A gene-expression signature as a predictor of survival in breast cancer. *N Engl J Med* 347(25):1999–2009
 14. West M et al (2001) Predicting the clinical status of human breast cancer by using gene expression profiles. *Proc Natl Acad Sci* 98(20):11462
 15. Loi S et al (2005) Breast cancer gene expression profiling: clinical trial and practice implications. *Pharmacogenomics* 6(1):49–58
 16. Hess KR et al (2006) Pharmacogenomic predictor of sensitivity to preoperative chemotherapy with paclitaxel and fluorouracil, doxorubicin, and cyclophosphamide in breast cancer. *J Clin Oncol* 24(26):4236–4244
 17. Naderi A et al (2006) A gene-expression signature to predict survival in breast cancer across independent data sets. *Oncogene* 26(10):1507–1516
 18. Wirapati P et al (2008) Meta-analysis of gene expression profiles in breast cancer: toward a unified understanding of breast cancer subtyping and prognosis signatures. *Breast Cancer Res* 10(4):R65
 19. Weigelt B et al (2012) Challenges translating breast cancer gene signatures into the clinic. *Nat Rev Clin Oncol* 9(1):58–64
 20. Reis-Filho JS, Pusztai L (2011) Gene expression profiling in breast cancer: classification, prognostication, and prediction. *Lancet* 378(9805):1812–1823
 21. Arpino G et al (2013) Gene expression profiling in breast cancer: a clinical perspective. *Breast* 22(2):109–120
 22. Weigelt B, Peterse JL, van't Veer LJ (2005) Breast cancer metastasis: markers and models. *Nat Rev Cancer* 5(8):591–602
 23. Kohonen T (1997) Exploration of very large databases by self-organizing maps. *Neural Networks, International Conference on I:PL1–PL6*
 24. Ultsch A, Morchen F (2005) ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM. Technical report 17. Data Bionics Research Group, University of Marburg, Marburg
 25. Fu L, Medico E (2007) FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioinformatics* 8(1):3
 26. Dembele D, Kastner P (2003) Fuzzy C-means method for clustering microarray data. *Bioinformatics* 19(8):973–980
 27. Hu Z et al (2006) The molecular portraits of breast tumors are conserved across microarray platforms. *BMC Genomics* 7(1):96
 28. Sotiriou C et al (2003) Breast cancer classification and prognosis based on gene expression profiles from a population-based study. *Proc Natl Acad Sci U S A* 100(18):10393–10398
 29. Samarasinghe S (2006) *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. CRC Press, Boca Raton, FL
 30. Ward JH (1963) Hierarchical grouping to optimize an objective function. *J Am Stat Assoc* 58(301):236–244
 31. Ultsch A, Mörchen F (2005) ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM. Technical report 46. CS Department, University Marburg, Marburg
 32. Dunn JC (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics* 3(3):32–57
 33. Bezdek JC (1981) *Pattern recognition with fuzzy objective function algorithms*. Kluwer, Baltimore, MD, p 256
 34. Dembélé D, Kastner P (2003) Fuzzy C-means method for clustering microarray data. *Bioinformatics* 19(8):973–980

35. Ozkan I, Turksen IB (2007) Upper and lower values for the level of fuzziness in FCM. *Inform Sci* 177(23):5143–5152
36. Van De Vijver MJ et al (2002) A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine* 347(25):1999–2009
37. Huang E et al (2003) Gene expression predictors of breast cancer outcomes. *Lancet* 361(9369):1590–1596
38. Wang Y et al (2005) Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet* 365(9460):671–679
39. Chang HY et al (2005) Robustness, scalability, and integration of a wound-response gene expression signature in predicting breast cancer survival. *Proc Natl Acad Sci U S A* 102(10):3738–3743
40. Chang HY et al (2004) Gene expression signature of fibroblast serum response predicts human cancer progression: similarities between tumors and wounds. *PLoS Biol* 2(2):E7
41. Perreard L et al (2006) Classification and risk stratification of invasive breast carcinomas using a real-time quantitative RT-PCR assay. *Breast Cancer Res* 8(2):R23
42. Parker JS et al (2009) Supervised risk predictor of breast cancer based on intrinsic subtypes. *J Clin Oncol* 27(8):1160–1167
43. Bild AH et al (2009) An integration of complementary strategies for gene-expression analysis to reveal novel therapeutic opportunities for breast cancer. *Breast Cancer Res* 11(4):55

Chapter 19

QSAR/QSPR as an Application of Artificial Neural Networks

Narelle Montañez-Godínez, Aracely C. Martínez-Olguín, Omar Deeb, Ramón Garduño-Juárez, and Guillermo Ramírez-Galicia

Abstract

Quantitative Structure–Activity Relationships (QSARs) and Quantitative Structure–Property Relationships (QSPRs) are mathematical models used to describe and predict a particular activity/property of compounds. On the other hand, the Artificial Neural Network (ANN) is a tool that emulates the human brain to solve very complex problems. The exponential need for new compounds in the drug industry requires alternatives for experimental methods to decrease development time and costs. This is where chemical computational methods have a great relevance, especially QSAR/QSPR-ANN. This chapter shows the importance of QSAR/QSPR-ANN and provides examples of its use.

Key words Artificial neural networks, Quantitative structure–activity relationship, Quantitative structure–property relationship, Principal component analysis

1 Introduction

The first Neural Network (NN) model goes back to 1943 when it was proposed by Warren McCulloch (1898–1969), a neurophysiologist, and Walter Pitts (1923–1969), a mathematician. The first attempt to use NN in computers took place in 1956 by a group of IBM researchers and neuroscientists from McGill University in Canada. Not only neurosciences have contributed to the development of Artificial Neural Networks (ANN). For instance, the network unit known as a Perceptron was proposed in 1957 by Frank Rosenblatt (1928–1971), a psychologist, and is still in use due to its simplicity.

In addition, the development of computer science has followed the way of the mathematician John von Neumann. In 1947, he designed a structure based on sequential processing of data and instructions. This structure rigorously follows a sequence defined by data stored in memory. Von Neumann's architecture is based on the logic of procedures, normally used for partial solutions in some problems to link the specific problem with the result. Although the

future of traditional computers seems to be very encouraging, promising, and complementary to human skills, biological neurons have attracted the attention of many scientists interested in the functioning of the brain and its structures, because its operations offer a possible alternative to computational processors [1].

Nowadays, ANN has reach maturity due to the efforts of many research groups around the world; they are applied in the solution of numerous complex real-world problems. ANN and artificial neural systems in general are multidisciplinary fields, making large contributions to different areas such as physics, mathematics, biology, engineering, and others [2]. Their best performance is in very complex problems that are too difficult for conventional technology—in other words, problems for which no algorithmic solution has been found. Some of the areas of application are the stock market, sales prediction, medical diagnosis, and object recognition, among others.

The best example of biological neural networks is the brain, an organ made of billions of specialized cells called neurons, interconnected by synapses. The synapses is the zone where two neurons are connected. Two further important parts of the cell are the dendrites, which function as the input channel to the cell, and the axons, which function as the output channel from it. A very simple representation of a cell is shown in Fig. 1.

The neuron body evaluates the incoming impulses and combines them through a network function that provides the strength of the neuron. As can be seen an artificial neuron behaves very much like the biological neuron but at a simplified level. Just as the biological neuron cannot exists by itself, artificial neurons as independent units are not useful for treating information; in order for them to work they must be grouped into larger structures, the ANN.

Artificial neurons are models that simulate the behavior of biological neurons. Each artificial neuron is represented as a

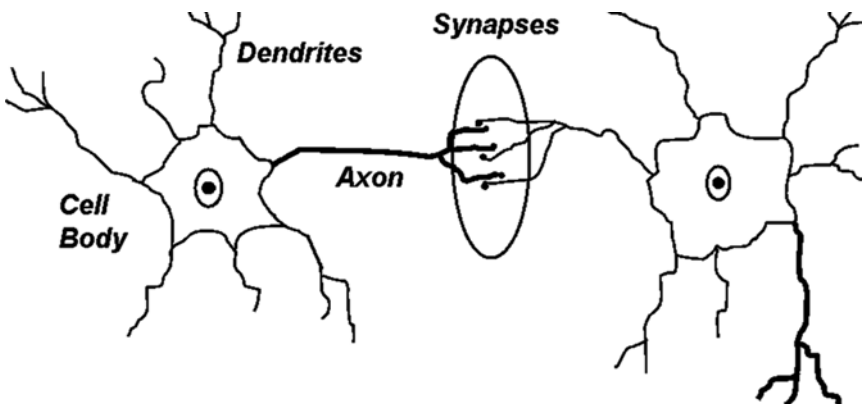


Fig. 1 Components of a biological neuron

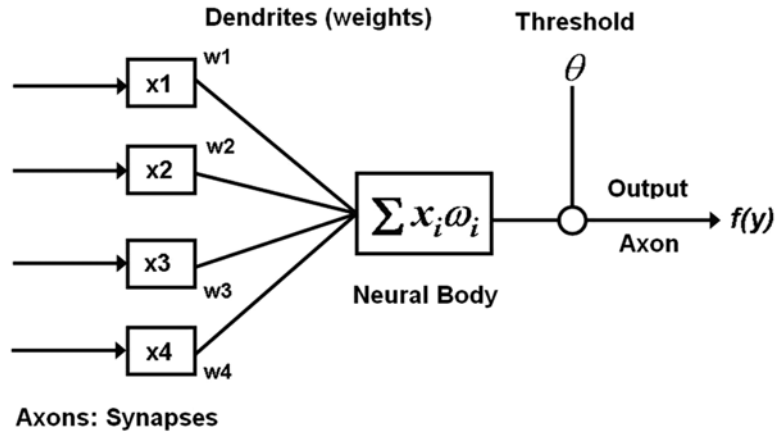


Fig. 2 Model of an artificial neuron according to McCulloch and Pitts (1943)

processing unit as illustrated in Fig. 2. This illustration shows a processing unit made of a series of inputs X_i , that are equivalent to the dendrites that receive the input signals regulated by synaptic weights represented by w_i and a parameter θ that is interpreted as the minimal threshold that the neuron has to overcome to be activated. In the neuron body, the output $f(y)$ is processed as a function of the synaptic weights (w_1, w_2, \dots, w_n) of each one of the input data (x_1, x_2, \dots, x_n) and the threshold θ ; mathematically: $f(y) = \sum_{i=1}^n w_i \cdot x_i + \theta$. Finally, the output value goes through an activation function that transforms the domain value, possibly infinite, into a determined value within a specific interval. A biological neuron can be active (excited) or inactive (non-excited); that is, it has an activation state. In a similar fashion, an artificial neuron also has different activation states; some artificial neurons can adopt only two states, in the same way as biological neurons; however others can take any values within a given interval. In artificial neurons the activation values are determined by the activation functions, the most common of which are sigmoidal or logistic and the hyperbolic tangent. Both functions can receive input values within the range $[-\infty, +\infty]$; the difference between them is that the first constrains the output values between $[0, 1]$ and the second between $[-1, +1]$. However, there are many activation functions depending on the application.

The ANN has featured in the scientific field of artificial intelligence because the interaction of information among the processors depends on the behavior of the whole system. That is, it deals with understanding from a computerized view of intelligent behavior. ANN emulates the human brain using a simulation system that is a computer program; the structures are modulated with high-capacity parallel computing, known as emulation, or by building systems similar to biological neural networks, which can learn from

stimuli provided by their environment, whilst artificial neurons must be programmed by computer based on the fastest microprocessor or Von Neumann's microprocessor that is capable of reliably executing complex instructions. ANN can emulate the set of cognitive and intellectual skills performed by the human brain. These ANN are interconnected with a hierarchical organization that allows performing tasks such as sorting and optimization by the information of sequential instructions that have been stored in memory, manipulating data [3, 4].

ANN is a nonparametric, nonlinear modeling technique that has attracted increasing attention in recent years [5]. Nonlinear multivariate maps apply a nonlinear transformation of the input variable space to project inputs onto the designated attribute values in output space. The power of modeling with layered, feed-forward artificial neural networks lies in the flexibility of the model defined by the weights of connections between units within the network. Together, linear and nonlinear mapping functions may be modeled by appropriate configuring of the network. Multilayer feed-forward neural networks trained with a back-propagation learning algorithm have become increasingly accepted approaches.

An important feature of ANN is its dynamic adaptability or ability to change its behavior in different contexts and with distinct problems. This is because it relies upon techniques inspired in learning, generalization or self-organization using elementary processing units. This general behavior determines its ability to test hypotheses, detect statistical patterns and regularities, as well as to adjust an implicit model which is implemented in the architecture of the network, which does not depend on the sum of the potentials of neurons [4].

The most common way to implement the models and algorithms adjusted by ANN has been through simulations on conventional computers. Moreover, this can be performed through architectures that are oriented to the execution of parallel processes completed on a set of processors connected with some regularity operating concurrently (neuro-computers), in this type of model, we can find the classic models Delta+ and Mark [4].

Formally, an ANN can be explained with the concept of the graph, an object composed of a set of nodes (vertices) and connections (links) that can be led when all connections are assigned a path and not led when connections are directional. Heavy graphs are those in which all nodes are connected to each other, while dispersed graphs have few associations. The graphs can be composed of different types of connections and nodes.

One way of representing graphs is with circles for the nodes, and lines or arrows for the connections (*see* Fig. 3). ANN normally keeps with certain properties and characteristics as the architecture of ANN and the modes of operation of the network [3].

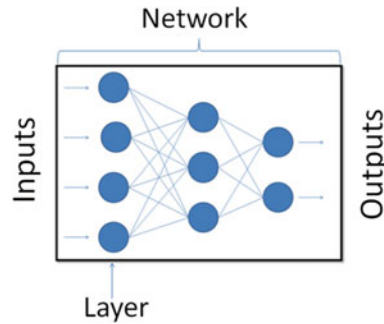


Fig. 3 A representation of an ANN

ANN is generally formed by a set of elementary processors that are called *artificial neurons*; these constitute simple devices of calculation based on information from other neurons or input from the outside to provide a unique response (output). According to their location in the network three types of layers can be distinguished: Input Layer—neurons that directly receive the information coming from an external source; Output Layer—neurons that present the processed data; Hidden-Layer—neurons that are neither in the input layer nor the output layer. These neurons are essentially hidden from view, and their number and organization can typically be treated as a black box to people who are interfacing with the system.

There are a number of different parameters that must be decided upon when designing a neural network. Among these parameters are: the number of layers, the number of neurons per layer, the number of training iterations, etc. Some of the more important parameters in terms of training and network capacity are the number of hidden neurons, the learning rate, and the momentum parameter.

The input neurons are those that receive information from the outside or the environment, through sensors or parts of the system. Meanwhile, the output neurons are those that send a signal directly out of the system when the information obtained has been analyzed or treated is marked by the incoming pulses; hidden neurons receive stimuli and emit output information but only within the system, so they have no contact with the outside world, they carry out the basic processing [3, 4].

The basic Perceptron concept is associated with a sensor, which might provide information on temperature, moisture, liquid levels, etc.; alternatively, input data may be fed in from a numerical database. The Perceptron is essentially a device that, given the presence of input data, can generate a single, recognizable output signal. However, an improvement can be achieved if we add more input channels to this simple device; it will then be able to differentiate

the different input data and to deliver an outcome through analysis of that data.

The successful use of neural networks to solve tasks in different areas is due to the ease with which an ANN acquires information or knowledge of a problem, as well as its ability to store information already acquired for later use, and, of course, its effectiveness in solving problems from sub-solutions. However, difficulties arise when we try to understand how a solution has been reached, because an ANN provides solutions to problems or outputs, without providing at the same time an explanation as to how that output was arrived at [6].

2 Methodology

Extraction is an important technique in ANN, even though this has a cost in resources and effort.

In artificial intelligence, the term of *explanation* refers to an explicit structure which can be used internally within, for example, an ANN, to elucidate and understand information and then explain the results obtained to the user. The information can be externalized with object hierarchy, semantic networks, frames, and so on. This explanation also includes the steps of the reasoning process or intermediate steps of the process as the trace of activity rules or test structure. This can assist the user in checking the internal logic of the system, a particularly helpful step if the explanations are very coarse or limited.

In many systems, there are deficiencies in rule extraction because it is very difficult to generate the explanatory structure of the process, even though the helpful capacity and quality of explanation are important in use of a trained ANN. The quality of explanation refers to how direct the task of extraction from ANN is. For example, Rule of trails may be used to capture an explanation, but it has been shown that this approach is often too rigid or inflexible, since these rules may contain references about internal calculations and repetitions [6].

2.1 Software

In most examples of the use of ANN in QSAR, the methodology uses quantum level calculations to optimize the molecular geometries data set, using semi-empirical methods or Density Functional Theory methods to generate descriptors and the subsequent selection of them to generate a new source of descriptors. The principal software tools used are Gaussian [7], MOPAC [8], and Hyperchem [9]. Numerical analysis can be realized with MATLAB [10] for linear regressions, an ANN for non-(multi)linear regression and SPSS [11] for multilinear regressions. Furthermore, Dragon [12] software is used in the calculation of descriptors.

3 QSAR/QSPR

For both financial and social reasons, the drug industry would like to develop new drugs in much shorter times than is possible using a purely experimental (lab-based) approach. A large number of novel compounds are synthesized each year. The number of molecules that can be obtained by organic synthesis is so high that the probability of choosing randomly a molecule with the desired biological activity is practically nil. In this regard, medicinal chemistry techniques that allow discovery and optimization of new templates are used. However, a large fraction of these compounds are not tested for physicochemical properties or biological activities, which still remain unknown owing to unavailability of the compounds or possible risks of toxicity. A method able to predict, within a realistic error boundary, the biological activities of untested compounds is necessary to evaluate these molecular features in a rapid and inexpensive approach.

Among these methods, computer aided design has found its way to the rational design of new compounds; the most important practice is known as Quantitative Structure–Activity Relationship (QSAR). In recent years, numerous quantitative structure–activity/property relationship QSAR/QSPR models have been introduced for calculating the biological activities from a variety of numerical descriptors of chemical structures. These relationships develop correlations between the descriptors of individual compounds and their biological activity/chemical properties. This approach has proved especially productive when coupled with solid optimizing tools that help to narrow down the field of potential molecules to just the most promising candidates, namely, Genetic Algorithms, Ant Colony Optimization, Support Vector Machines, and Artificial Neural Networks.

An inevitable difficulty, when dealing with molecular descriptors, is that of collinearity, which frequently exists between independent variables. In the worst case, this creates a rigorous problem in certain types of mathematical treatment such as matrix inversion [13], though collinearity of molecular descriptors is not normally so complete that matrix inversion will fail. A better predictive model can be obtained by orthogonalization of the variables by means of principal component analysis (PCA) and the subsequent method is called principal component regression (PCR) [14].

In order to decrease the dimensionality of the independent variable space, a restricted number of principal components (PCs) are used. For this reason, the choice of significant and informative PCs is an important task in PCA–based calibration methods [15, 16].

Diverse methods have been proposed to select the important PCs for calibration purposes. The simplest and most frequent one is a top–down variable selection where the factors are ranked in the

order of decreasing eigenvalue. The factor with the highest eigenvalue is considered as the most important one; further factors are then progressively introduced into the calibration model until no significant improvement of the calibration model is obtained. In another method, correlation ranking, the factors are ranked by their correlation coefficient with the property to be correlated (i.e., biological activity) and selected by the procedure of eigenvalue ranking.

QSAR is a mathematical model that attempts to relate the structure-derived features of a compound to its biological or physicochemical activity and it is used to understand drug action as well as to design new compounds. Elaboration of QSAR models starts with the collection of a representative set of molecules with the desired biological activity, and a control group that does not possess the activity under study. This is followed by the selection of molecular descriptors obtained by a mathematical logic that transforms the chemical and/or structural information about a molecule into a set of numbers. In favorable cases, the molecular descriptors can then be related to the desired biological activity. In order to validate the descriptors, the full data set is divided into a training set, and a learning test set. In the learning process, there are many ways of building a model, among which we can mention Multiple Linear Regression (MLR), logistic regression, or machine learning methods. The optimal model is obtained from the modeling parameters and features. Finally, this model is validated to ensure that it will be useful (*see* Fig. 4) [17].

A QSAR model correlates, within congeneric molecules, many biological activities such as inhibition constants, affinities of ligands

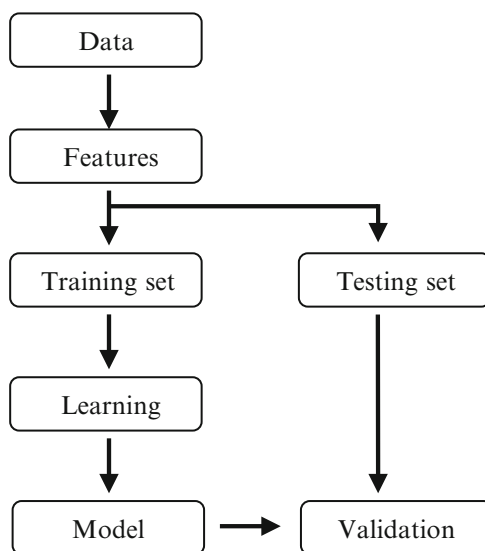


Fig. 4 General workflow of developing a QSAR model [17]

to their binding sites and rate constants, with either structural features or with atomic group or molecular properties such as solubility, lipophilicity, electronic effects, ionization, and stereochemistry. The structural features were proposed by Free and Wilson [18] and the use of molecular properties was proposed by Hansch and Fujita [19], both published independently in 1964. QSAR has been important in the integration of ADMET (absorption, distribution, metabolism, excretion, and toxicity) profiling and prediction, which is mostly dependent on a type of molecular descriptors as described by the Lipinski Rule of 5. This procedure is applied to improve our understanding of structure–activity links, to identify hazardous compounds at low cost, and to reduce the number of animals used for experimental toxicity testing. In the field of QSAR and in the prediction of animal toxicity, various artificial intelligence techniques have been proposed and developed such as ANN and statistical understanding of knowledge networks [20–22].

There are different tools that have been used to predict the activity of molecules that can present some biological activity; MLR, nonlinear regression using ANN and molecular simulations from molecular recognition (Docking) are also used in QSAR [23].

The calculation of chemical descriptors is an important tool for QSAR applications, because it makes possible prediction of toxic properties, harmful characteristics, and pharmacological biological properties that a group of molecules could present. Many chemical descriptors can be measured experimentally but this is cost and time expensive, and therefore development of QSAR tools is useful to replace or augment measurement.

Most QSAR/QSPR practice uses quantum-mechanical descriptors. QSAR analysis uses a variety of descriptors including:

- Topological,
- Quantum chemical,
- Galvez topological charge index,
- 2D-autocorrelation,
- Radial Distribution Functions (RDF),
- 3D-MoRSE (Molecule Representation of Structures based on Electron diffraction),
- Weighted Holistic Invariant Molecular (WHIM),
- GEometry Topology and Atom-Weight Assembly (GETAWAY), and so on [17, 24].

3.1 QSAR-ANN Application

In the literature, we can find work that reports applications in which only QSAR has been used, e.g., a QSAR model that describes the antispasmodic activity of molecules isolated from Mexican Medicinal Flora and some synthetic based on stilbenoid bioisosteres [25]. Equally, there are reports of applications that use only

ANN, such as the unsupervised mapping of samples [26], the prediction of the maximum power of a low concentration photovoltaic module [27], or drug analysis [28].

The importance of ANN-QSAR methods is their flexibility to cover a range of applications. In addition they have the advantage of providing results with higher speed and lower costs than experiments. Furthermore, the ability of ANN to discover more complex relationships has allowed this method to find a wide application in QSAR studies [29, 30]. A principal component-artificial neural network (PC-ANN) method, which combines the PCA with ANN, is another PCA-based calibration method for nonlinear modeling between the PCs and biological activities [13]. Many of these QSAR studies have been performed in our group.

As an initial example, it is possible to predict toxicity using QSAR algorithms. In this example, 278 substituted benzenes were selected and the binding affinities to human serum albumin (HSA) were modeled for 94 these compounds. HSA is the most abundant protein in plasma and has a high capacity to bind to several drugs. We have compared SR-PC-ANN (stepwise regression-principal components-artificial neural networks) and CR-PC-ANN (correlation ranking-principal components-artificial neural networks) procedures, both combined with two PCA approaches, the individual PCA approach, PCA(I), and the combined PCA approach, PCA(C). More accurate results were obtained with CR-PC-ANN and the PCA(I) approaches [31].

As a second example, it is possible to describe and predict the antinociceptive activity of morphinans derivatives (*see* Fig. 5) using QSAR analysis. MRL and Leave-One-Out Cross-Validation (LOO-CV) were applied to the best QSAR models. Furthermore, in this instance 31 morphinan derivatives reported by the US Drug

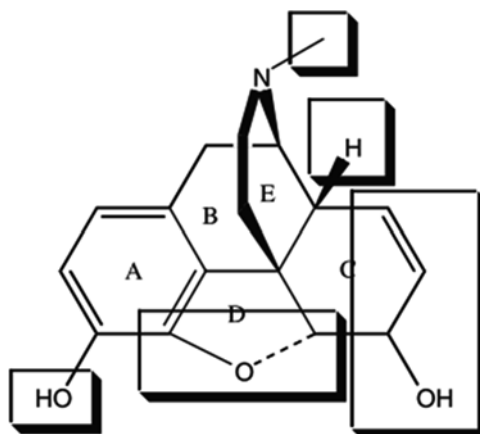


Fig. 5 Different changes of morphine structure localized on the 31 morphinan studies [29]

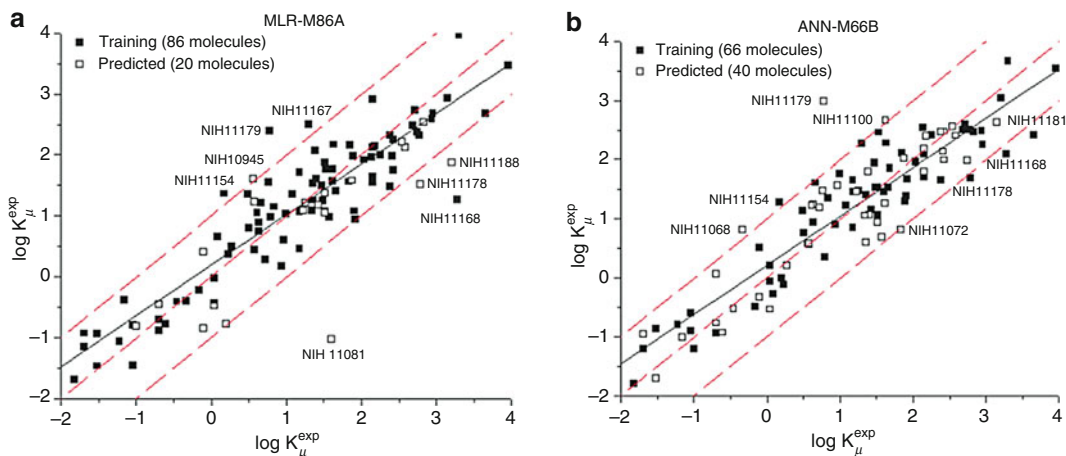


Fig. 6 Examples of data set calculated by Multiple Linear Regression (*solid line*), ideal line (*dashed line*), and residual line limits (*dotted line*, cutoff = 11.0). These were the best ANN calculated results [32]

Evaluation Committee were analyzed. The models calculated achieved good descriptive and predictive power. This was improved by using ANN; however it was unable to predict an external validation set of morphinan derivatives [29].

Third, it is feasible to study biological activity of 106 morphinan derivatives (reported by the US Drug Evaluation Committee) to describe μ -receptor-binding affinity. Twenty one descriptors were selected to perform the MLR. ANN was then used to improve the results (*see* Fig. 6). The context of this work is that opiates often used for the control of pain can also have toxic side effects. There are different types of opioid receptors, including μ -receptor that shows the highest affinity for morphine, so the affinity in this site is an interesting topic of study [29, 32].

Fourth, a QSAR analysis has been conducted on analgesic activity (log IC) for 95 heterogeneous analgesic compounds using the PC-ANN modeling method, in which we applied the eigenvalue ranking factor selection procedure. This study demonstrates that the PC-ANN can be used to generate improved general models for heterogeneous data sets without splitting them into categories. The PC-ANN gives better regression models with good prediction ability using a relatively low number of principal components. A 0.834 correlation coefficient was obtained using PC-ANN with six extracted principal components [33].

Fifth, a QSAR analysis has been performed on three types of Carbonic Anhydrase (CA) I, II and Bovine IV isozyme inhibitory activities for 53 sulfonamides using multiple linear regression (MLR), principal component artificial neural network (PC-ANN), and correlation ranking–principal component regression (CR-PCR) analyses. It was observed that the interaction between the ligand and receptor varies from one type of CA isozyme to another.

The results obtained provide very good regression models with good prediction ability. Correlation ranking-principal component regression analysis provides models with better prediction capability for the three types of the CA isozyme inhibitory activity, while PC-ANN analysis provides models with better prediction capability for the hCAII isozyme activity. Generally, the models obtained for modeling the hCAII isozyme inhibitory activity are superior over those obtained for modeling the hCAI and bCAIV isozyme inhibitory activities. The QSAR model obtained for CAI indicated that the molecules with higher hydration energies were found to bind strongly to the receptor. In addition, it was found that higher molecular connectivity tends to block the drug binding to the receptor.

By contrast, for the binding of sulfonamides to CAII isozyme, it was found that the presence of such rings in the sulfonamides blocks their binding to the receptor. For the CAIV isozyme, the obtained QSAR model illustrates the extremely significant role of softness, so that the more polarizable is the ligand, the stronger it binds to the receptor. It was found that higher molecular connectivity tends to block the drug binding to the receptor for the three types of CA isozyme inhibitory activities. The results obtained show that linear and nonlinear regression analyses are useful tools to distinguish between the inhibitory activities of sulfonamides toward different CA isozyme types I, II, and IV [34].

Sixth, a QSAR study was performed to understand the inhibitory activity of a set of 192 vascular endothelial growth factor receptor-2 (VEGFR-2) compounds. QSAR models were developed using MLR and PLS as linear methods, while PC-ANN modeling method with application of eigenvalue ranking factor selection procedure was used as a nonlinear method. The results obtained offer good regression models having good prediction ability. The results obtained by MLR and PLS are close, and better than those obtained by principal component-artificial neural network. The best model obtained had a correlation coefficient of 0.87. The strength and the predictive performance of the proposed models were verified using both internal (cross-validation and Y-scrambling) and external statistical validations [35].

Seventh, Counter propagation neural network (CPNN) is an attractive classification tool (active and non-active) in QSAR studies. A major obstacle in classification by CPNN is finding the best subset of variables. In this study, the performance of some different feature selection algorithms including F score-based ranking, eigenvalue ranking of PCs obtained from the data set, Non-Error-Rate (NER) ranking of both descriptors and PCs, and 3-way handling of data, Parallel Factor Analysis (PARAFAC), were evaluated in order to find the best classification model. The methods were applied for modeling protein-tyrosine kinase inhibition of 105 flavonoid derivatives using substituent electronic descriptors

(SED) as a novel source of electronic descriptors. The results showed that the best performance was achieved by F-score ranking, while the NER ranking of principal components (PCs) showed very fluctuating results; the worst performance belonged to PARAFAC-CPNN. Furthermore, comparison of results of these nonlinear algorithms with a linear discriminate analysis method revealed better predictions by the former [36].

3.2 QSPR-ANN Application

Quantitative structure–property relationship studies have also been performed using PC-ANN in our group.

First, a QSAR analysis has been conducted on bioconcentration factor (BCF) for 227 different non-ionic organic compounds. The terms bioaccumulation and bioconcentration refer to the uptake and buildup of chemicals that can occur in living organisms. Experimental measurement of bioconcentration is time-consuming and expensive, and is not feasible for a large number of chemicals of potential regulatory concern. A highly effective tool depending on a quantitative structure–property relationship (QSPR) can be used to describe the tendency of chemical concentration organisms represented by the important ecotoxicological parameter, the logarithm of Bio Concentration Factor (\log BCF) with molecular descriptors for a large set of non-ionic organic compounds. QSPR models were developed using multiple linear regression, partial least squares, and neural networks analyses. Linear and nonlinear QSPR models to predict \log BCF of the compounds were developed for relevant descriptors. The results obtained offer good regression models with good prediction ability. The descriptors used in these models depend on the volume, connectivity, molar refractivity, surface tension, and the presence of atoms accepting H-bonds [37].

Second, a quantitative-structural property relationship analysis has been performed on the logarithm of gas/particle partitioning coefficient ($\log K_p$) for 70 different semi-volatile organic compounds by using the PC-ANN modeling method, with application of an eigenvalue ranking factor selection procedure. The PC-ANN gives good regression models with good prediction ability using a relatively low number of PCs. The optimal models obtained by PC-ANN and partial least square (PLS) analyses are in close proximity from the statistical point of view. The results obtained offer excellent regression models that hold good prediction ability compared to other studies on the same data set of compounds. A coefficient of determination around 0.97 was obtained using PC-ANN and PLS analysis [38].

Third, a quantitative structure–property relationship analysis has been performed on the logarithm of solubility ($\log 1/S$) in water for 219 different pesticide compounds by using the PLS method as well as a PC-ANN method, with application of the eigenvalue ranking factor selection procedure. The PLS and PC-ANN give

good regression models with good prediction ability using a relatively low number of PCs. The optimal models obtained by PC-ANN are better than those obtained by PLS analyses from the statistical point of view. The results obtained offer excellent regression models that hold good prediction ability. The optimal PLS model has coefficients of determination of 0.7998 and 0.7944 for the training and test sets, respectively, while the PC-ANN model has coefficients of determination of 0.8435 and 0.8193 for training and test sets, respectively. The descriptors used in these models are consistent with the experimental factors that are presumed to affect the solubility of pesticide compounds in water [39].

4 Conclusions and Perspectives

Currently, QSAR analysis has not reached a level at which the predictive power is as good as the descriptive power, but when ANN is included in the models, the predictive power is improved, compared to a model that uses only MLR. When using better computers and models, predictions will be closer to reality.

Acknowledgement

AC Martínez-Olguín wishes to thank CONACyT for a graduate scholarship. The English was kindly reviewed by Miss Désirée Argott.

References

1. de la Fuente Aparicio Ma J, Calonge Cano T (1999) Aplicaciones de las redes de neuronas en supervisión, diagnosis y control de procesos. Universidad Simón Bolívar, Caracas-Venezuela, p 11
2. Brégains JC (2007) Análisis síntesis y control de diagramas de radiación generados por distribuciones continuas y discretas utilizando algoritmos estadísticos y redes neuronales artificiales aplicaciones. Universidad de Santiago de Compostela, Galicia-Spain, p 104
3. Flores López R, Fernández Fernández JM (2008) Las redes neuronales artificiales Fundamentos teóricos y aplicaciones prácticas. Netbiblo, S.L., Spain, pp 12–14, 17, 21
4. Pino Diez R, Gómez Gómez A, de Abajo Martínez N (2001) Introducción a la inteligencia artificial: Sistemas expertos, redes neuronales artificiales y computación evolutiva. Universidad de Oviedo, Asturias-Spain, pp 27, 28
5. Douali L, Villemin D, Cherqaoui D (2003) Neural networks: accurate nonlinear QSAR model for HEPT derivatives. *J Chem Inf Comput Sci* 43:1200–1207
6. Andrews R, Diederich J, Tickle AB (1995) Survey and critique of techniques for extracting rules from trained artificial neuronal networks. *Knowl Base Syst* 8:373–389
7. Frisch MJ, Trucks GW, Schlegel HB et al (2009) Gaussian 09. Gaussian, Inc., Wallingford CT
8. Stewart JJP (2009) MOPAC 2009. Fujitsu Limited, Tokyo, Japan
9. HyperChem(TM) 2003 Professional 7.51, Hypercube, Inc., 1115 NW 4th Street, Gainesville, Florida 32601, USA
10. MATLAB Version 7.2.0.232 (R2006a). The MathWorks, Inc. (2006) Natick, MA
11. SPSS Version 13 for Windows (2004) Chicago, IL
12. Todeschini R, Consonni V, Pavan M (2002) Dragon Software version 2.1-2002 Pisani 13, Milano. Dragon Software and references therein

13. Vendrame R, Braga RS, Takahata Y et al (1999) Structure–activity relationship studies of carcinogenic activity of polycyclic aromatic hydrocarbons using calculated molecular descriptors with principal component analysis and neural network methods. *J Chem Inf Comput Sci* 39:1094–1104
14. Jolliffe IT (1986) *Principal component analysis*. Springer, New York, NY
15. Xie YL, Kalivas JH (1997) Evaluation of principal component selection methods to form a global prediction model by principal component regression. *Anal Chim Acta* 348:19–27
16. Depczynski U, Frost VJ, Molt K (2000) Genetic algorithm applied to the selection of factors in principal component regression. *Anal Chim Acta* 420:217–227
17. Dehmer M, Varmuza K, Bonchev D (2012) *Statistical modelling of molecular descriptors in QSAR/QSPR*. Wiley, Germany, p 2
18. Free SM, Wilson JA (1964) A mathematical contribution to structure-activity studies. *J Med Chem* 7:395–399
19. Hansch C, Fujita T (1964) ρ - σ - π Analysis. A method for the correlation of biological activity and chemical structure. *J Am Chem Soc* 86:1616–1626
20. Hansch C, Leo A, Hoekman D (1995) *Exploring QSAR: fundamentals and applications in chemistry and biology*. American Chemical Society, Washington, DC
21. Hansch C, Leo A, Hoekman D (1995) *Exploring QSAR: volume 2: hydrophobic, electronic, and steric constants*. American Chemical Society, Washington, DC
22. Helma C, Kramer S (2003) A survey of the predictive toxicology challenge 2000–2001. *Bioinformatics* 19:1179–1182
23. Waterbeemd H, Gifford E (2003) ADMET in silico modeling: towards prediction paradise? *Nat Rev Drug Discov* 2:192–204
24. Karelson M (2000) *Molecular descriptors in QSAR/QSPR*. Wiley-Interscience, New York, USA
25. Ramírez-Galicia G, Garduño-Juárez R, Hemmateenejad B et al (2007) QSAR study on the relaxant agents from some Mexican medicinal plants and synthetic related organic compounds. *Chem Biol Drug Des* 70:143–153
26. Marini F, Bucci R, Magri AL et al (2008) Artificial neural networks in chemometrics: history, examples and perspectives. *Microchem J* 8:178–185
27. Fernández EF, Almonacid F, Sarmah N et al (2014) A model based on artificial neuronal network for the prediction of the maximum power of a low concentration photovoltaic module for building integration. *Sol Energy* 100:148–158
28. Agatonovic-Kustrin S, Beresford R (2000) Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J Pharm Biomed Anal* 22:717–727
29. Ramírez-Galicia G, Garduño-Juárez R, Hemmateenejad B et al (2007) QSAR Study on the antinociceptive activity of some morphinans. *Chem Biol Drug Des* 70:53–64
30. Schbeider G, Wrede P (1998) Artificial neural networks for computer-based molecular dosing. *Prog Biophysics Mol Biol* 70:175–222
31. Deeb O (2010) Correlation ranking and stepwise regression procedures in principal components artificial neural networks modeling with application to predict toxic activity and human serum albumin binding affinity. *Chemometr Intell Lab Syst* 104:181–194
32. Ramírez-Galicia G, Garduño-Juárez R, Deeb O et al (2008) MLR–ANN and RTO Approach to μ -opioid receptor-binding affinity. Pooling data from different sources. *Chem Biol Drug Des* 71:260–270
33. Deeb O, Drabh M (2010) Exploring QSARs of some analgesic compounds by PC-ANN. *Chem Biol Drug Des* 76:255–262
34. Deeb O, Goodarzi M, Khadikar PV (2012) Quantum chemical QSAR models to distinguish between inhibitory activities of sulfonamides against human carbonic anhydrase I and II and bovine IV isozymes. *Chem Biol Drug Des* 79:514–522
35. Deeb O, Jawabreh S, Goodarzi M (2013) Exploring QSAR's of vascular endothelial growth factor receptor-2 (VEGFR-2) tyrosine kinase inhibitors by MLR, PLS and PC-ANN. *Curr Pharm Des* 19:2237–2244
36. Hemmateenejad B, Mehdipou A, Deeb O et al (2011) Toward an optimal approach for variable selection in counter-propagation neural networks: modeling protein–tyrosine kinase inhibitory of flavanoids using substituent electronic descriptors. *Mol Inform* 30:939–949
37. Deeb O, Khadikar PV, Goodarzi M (2010) QSPR modeling of bioconcentration factors of nonionic organic compounds. *Environ Health Insights* 4:33–47
38. Deeb O, Khadikar PV, Goodarzi M (2011) Study of gas/particle partitioning coefficients of semivolatle organic compounds via QSPR methods. *J Iran Chem Soc* 8:176–192
39. Deeb O, Goodarzi M (2010) Predicting the solubility of pesticides compounds in water via QSPR methods. *Mol Phys* 108:181–192

INDEX

A

Accelrys126
 Activation function.....54, 116, 133, 196, 214,
 215, 249, 251, 271, 272, 276, 319
 Actuator.....182
 Adaboost232, 235
 Adaptive learning150, 270
 Adaptor2–4, 6–14
 ADMET46, 48, 55, 56, 83, 155, 325
 Amino acid.....3–6, 8–10, 12, 13, 15, 21–23, 26, 28, 89,
 101–104, 106–108, 121, 172
 AMPList.....104, 106, 107, 111
 Analgesic activity.....327
 Ant-Colony Optimization.....323
 Antibacterial activity.....55–57
 Antibacterial agent45–61
 Antibiotic45, 46, 80, 103, 106
 Antibody.....11
 Antimicrobial49, 103–107, 113, 114
 Architecture.....9, 53, 58, 95, 122, 133, 134, 164,
 167, 170, 183, 196, 197, 202, 203, 217, 218, 236,
 237, 239, 249, 252, 255, 257, 261, 269–282,
 317, 320
 Artificial pancreas.....246
 Artificial synapse261–267
 Authentication.....205–224
 Autocorrelation.....82, 108, 184, 210, 325
 Autoencoder network237
 AutoWeka119–145

B

Babel.....126, 161
 Backbone.....18–21, 24, 26
 Backpropagation.....200, 252, 253
 Bacteria.....33, 45, 46, 55, 61, 105
 Bacterial community.....33–42
 Bayesian network.....37
 Bayesian Network Inference tool (BANJO).....37, 38, 41, 42
 Bayesian regularization.....253
 Bayesian smoothing.....250
 Benzene.....80, 120
 Binding activity154–160
 Binning.....35, 128

Bioactive peptide101–116
 Bioactivity.....72, 125, 127–129
 Biochemical process.....1, 2
 Biochemistry13, 15
 Bioconcentration81, 329
 Bioinformatics48, 124, 125, 274, 275
 Biological activity46, 47, 51, 111, 149–162,
 323–325, 327
 Biological mechanism.....261
 Biology1, 15, 125, 286
 Biometric authentication.....205–224
 Biometric feature set.....207
 Biometric identity.....207
 Biopsy.....200, 203
 Bitmap image209
 Black-box69, 136
 Blood type278, 282
 Boiling-point.....70–72, 83, 120, 121
 Boltzmann learning rule273, 274
 Boosting232
 Bootstrap40
 Bottom-up method.....200
 Bray-Curtis similarity.....36, 40
 Breast cancer.....195, 203, 285–315

C

Camera206, 208, 229, 231, 234, 253
 Cannabinoid (CB).....150, 151, 154–161
 Carbohydrate.....247
 Carcinogenicity70, 76, 81–83, 123
 CARET.....111, 112
 Cargo.....2–8, 10–14
 Cargo localisation.....10
 Catalytic function.....1
 CD-Hit106
 Cell.....2, 3, 6, 13–15, 46, 48, 161, 270, 318
 Cepstral210
 CGM sensor.....246–249, 253, 254, 257
 Character recognition.....274
 Charge prediction.....89–99
 Charge transfer.....91
 Chemical shift17–31
 Chemogenomics.....150
 Chemotherapy.....274, 281

Chimera.....	11, 13
Classification	21, 24, 26–28, 30, 46, 51–55, 59, 60, 67, 77, 79, 82–84, 96, 104, 105, 107, 111–115, 130, 134, 196, 199, 213, 215, 232, 234, 235, 239, 274–276, 285–315, 328
Classifier	
strong.....	232, 234
weak.....	232–234
Cluster.....	4, 9, 24, 199, 212, 285–315
Clustering.....	20, 23, 55, 285–315
Coil.....	21, 23, 24, 101, 102, 275
Collision-induced dissociation (CID).....	89–95, 97–99
Combinatorial chemistry.....	46
Combinatorial library.....	9
CoMFA.....	49, 153
Compartment.....	2–4, 6, 8, 10, 15
Competitive learning rule.....	273, 274
Computational neuroscience.....	227
Computer-aided diagnosis (CAD).....	195–203
COMSIA.....	49, 153, 154
Conformation.....	7, 18, 19, 21, 49–51, 156
Confusion matrix.....	113, 114
Controller.....	180–182
CORINA.....	126
Correlation coefficient.....	40, 42, 59, 60, 90, 94, 114, 153, 210, 304, 324, 327, 328
Corrplot.....	104
COSMOS.....	126
Cost function.....	111, 197, 200, 213
Counter-propagation neural network (CPNN).....	328, 329
Crisp clustering.....	285
Cross-validation	68, 98, 111–113, 116, 152–155, 168, 251–253, 278, 326, 328
Curation	46, 55–57, 145
Curcumin	124, 128, 129, 131, 132, 136–138
Cybenko theorem.....	276
CyberHand.....	180
D	
Database	
AAindex.....	108
ACE.....	150, 151
AChE.....	150, 151
APD2.....	106
BindingDB.....	125
BZR.....	150, 151, 153, 154
ChEMBL.....	47, 48, 55–57, 126, 156
COX2.....	150, 151, 153
DHFR.....	150, 151, 153, 154
DrugBank.....	48, 126
flybase.....	12
LipidBlast.....	90
NCBI protein.....	5
NCI.....	150, 157–160
Protein Data Bank.....	18, 165
PubChem.....	126
SGD genome.....	12
WOMBAT.....	126
Data mining.....	54, 55, 119–145
Degradation.....	76, 79, 81
Devtools.....	104
Diabetes.....	150, 246–248, 274
Dimensionality.....	196, 199, 232, 235, 239, 323
Discretization.....	41, 42
Disease	3, 4, 14, 45, 46, 61, 246–248, 275, 276, 281, 286
DNA.....	34, 276
DNA-microarray.....	287
Domain	3, 6, 7, 12, 15, 67–69, 79, 121, 124, 209, 210, 218, 257, 300, 319
DRAGON.....	52, 127, 322
Drug	
antibacterial.....	46, 48, 54, 61
development time.....	103
discovery.....	46, 48, 53, 61, 119, 149, 150, 157, 274, 275
resistance.....	275, 281
Drug-resistant bacteria.....	45, 281
Durbin-Watson test.....	278
E	
Early stopping.....	152, 202, 252, 253
Ecosystem.....	34
Eigenvalue.....	153, 324, 327–329
Eigenvalue ranking.....	324, 327–329
Electronic synapse.....	261
Embedded system.....	205, 207, 217
Embryo.....	1
Emergent properties.....	14
Emergent Self-organising Map (ESOM).....	288, 290, 294–305, 313
Environmental interaction network (EIN).....	34, 35, 37–42
Error function.....	134
Etching.....	262, 263, 265, 266
Euclidean distance.....	213, 291
Eukaryotic cell.....	1, 2
Eureqa.....	37, 38, 40, 42
European Union (EU).....	65, 117
Evolutionary algorithm.....	37
Excel.....	47, 48
Extended-Connectivity Fingerprint (ECFP6).....	152, 158, 160, 161
F	
False acceptance rate (FAR).....	206
False rejection rate (FRR).....	206
FANN.....	149–162
FANN-QSAR.....	149–162
FCM. <i>See</i> Fuzzy C-means (FCM)	
Feature extraction.....	199–200, 207, 209–212, 218–219

- Feedback.....3, 109, 110, 179–181, 272
Feedforward..... 183, 196, 239
Feedforward network..... 183
Filter..... 58, 105, 162, 170–172, 232, 239
Fingerprint 51, 77, 89, 149–162, 206–209,
211–215, 217–224
Fingers..... 180–182, 187, 190
5D-QSAR.....49
FLAME. *See* Fuzzy Clustering by Local Approximation of
Memberships (FLAME)
Flash-point..... 70, 73
Food and Drug Administration (FDA).....48
4D-QSAR..... 49, 122
Fragmentation pattern.....89
Free-Wilson.....49
Fuzzy clustering.....285–315
Fuzzy Clustering by Local Approximation of Memberships
(FLAME).....289, 290, 295–306, 312–314
Fuzzy C-means (FCM)..... 289, 290, 295–314
Fuzzy logic.....206, 207, 215–217, 223, 224
Fuzzy membership294, 295
- G**
Gabor 232, 233
GAMESS..... 126, 127
Gaussian 54, 123, 126, 127, 292, 322
Gene
 clustering..... 288, 293, 296–305
 expression signature.....286
 identification.....275–276
 marker286
 signature285–315
 subset.....288–290, 296–299, 302–307, 309–314
General Neural Network (GENN)165–176
Genetic algorithm 66, 92, 124, 323
Genetic disease 14
GENN. *See* General Neural Network (GENN)
Global warming..... 33
Globular protein.....165
Glucose.....245–257, 274
Glycemia246–250, 257
Gradient descent111, 152, 185, 200, 238
Grasping force.....179–182, 187, 191, 192
- H**
Haar232, 233
Hammett..... 49, 120
Hamming210, 218
Hansch analysis49
Hebbian rule.....273, 274
Helix..... 21, 26, 28, 101, 102, 108, 275
Hereditary disease4
Hidden layer..... 54, 72, 96, 112, 113, 115, 133,
151, 152, 167, 171, 184, 196, 214, 215, 237, 249,
252, 257, 270, 271, 276, 321
Hidden-Markov model281
Hidden neuron 115, 152–155, 249, 251, 252, 321
Hierarchical clustering 286, 288–290, 305
High-throughput screening.....46
Histology.....203
Horizon 61, 182, 185, 248
Human serum albumin (HSA).....326
Hydrogen bonding18
HyperChem 126, 322
Hyperglycemia..... 246–248, 256, 257
Hyperglycemic event256
Hypoglycemia.....246, 257
Hypoglycemic event256
- I**
Image analysis.....274, 275
Infomax239–241
Information code 7, 14
Inhibitory effect.....8
Input layer9, 54, 95, 96, 151, 152, 239, 271,
276, 291, 321
Insulin246–249, 257
Intelligent systems53, 270
Interpol.....104
Intracellular sorting1–16
Irritation 70, 76–77, 83
Isotope..... 17, 22, 30, 158
Isozyme327, 328
- J**
Jump neural network245–257
- K**
Kernel.....135
k-fold cross-validation 111, 112, 116, 251, 252
k-nearest-neighbor (KNN).....77, 79, 212, 213, 294–296
Kyrgyz278
- L**
LDA. *See* Linear discriminant analysis (LDA)
Learning paradigm 111, 273
Learning rate 96, 134, 144, 171, 197, 200, 220,
241, 292, 296, 321
Lennard-Jones.....49
Lesion
 benign.....200
 malignant.....200
Levenberg-Marquardt 183, 187, 252, 253
Linear activation function 214, 215, 249
Linear discriminant analysis (LDA)..... 46, 51–54, 58,
77, 200
Lipid.....1–3, 14, 66, 91, 95, 98
Lipid metabolites..... 90, 92, 93, 95–99
Localization.....3, 5, 7, 10–11, 13, 211

Local strain.....18
 Loop..... 19, 26, 101, 102, 109, 181, 257, 272
 Luminal space.....6

M

MACCS..... 151–155, 161
 Machine learning..... 55, 58, 61, 91–93, 97, 99,
 121, 124, 127, 129–137, 150, 228, 230, 235, 242,
 243, 324
 Magnetron sputtering.....262, 263
 Mahalanobis-distance.....212
 Mammogram..... 196, 197, 201–203
 MAP. *See* Microbial Assemblage Prediction (MAP)
 MarvinSketch.....126
 Mascot.....89
 Mass extinction33
 Mass spectrometry.....89–99
 MATLAB 40, 151, 187, 198, 217, 292, 296, 322
 Matthews correlation coefficient (MCC)..... 59, 60, 114
 Mean squared error (MSE) 152, 155
 Medical image195, 196, 199, 201, 275
 Melting-point..... 71, 121
 Membrane2–6, 10, 11, 15, 66, 161
 Memory.....42, 137, 140, 217, 229, 231, 238, 239,
 242, 243, 261, 262, 270, 317, 320
 Memristive-switching261–267
 Memristor261, 262
 Metabolite89–99
 Metagenomics35
 Metagenomic sampling 34, 35
 Metagenomic sequencing34
 Methicillin-resistant *S. aureus* (MRSA).....46
 MetiSIS.....90, 91, 93, 94, 99
 Microarray.....285–315
 Microbial Assemblage Prediction (MAP)34–42
 Microbial community34
 Microbial ecology34
 Microbial pathogen 45, 46, 61
 Microbial population34–41
 Microcalcification.....199, 201
 MLP. *See* Multilayer perceptron (MLP)
 MLR. *See* Multiple linear regression (MLR)
 Modelling..... 83, 125, 231, 242
 Model validation104
 MODESLAB 51, 57
 MOLCAS.....126
 Molecular biology..... 1–17, 274, 275
 Molecular fingerprint77, 150, 151, 153, 161
 Momentum 96, 144, 152, 171, 197, 200, 277, 321
 Monte-Carlo 91, 93
 MOPAC.....126, 322
 Morphinan derivatives.....326, 327
 Moving average57
 mtk-QSBER 46, 47, 55–61

mt-QSAR46
 Multilayer perceptron (MLP).....53, 54, 58, 59, 112,
 115, 116, 133
 Multiple linear regression (MLR)52, 66, 68, 72–75,
 78–80, 82, 83, 121, 324, 325, 327–330
 Multivariate analysis..... 124, 127, 128, 130, 136–144
 Mutagenicity 76–78, 83, 123

N

Navigation227–243
 Network root node34, 35
 Network topology.....38
 Neuron9, 15, 53, 54, 69, 109, 110, 112, 113,
 115, 130, 152–155, 196, 220, 228, 229, 249, 251,
 252, 257, 270–272, 274, 276, 290–294, 299, 306,
 309, 318–321
 Neutral-loss91
 Node..... 9, 34, 35, 37, 38, 40, 42, 53, 54, 61, 77, 95,
 96, 99, 109, 115, 133, 139, 143, 144, 171, 172,
 196, 203, 214, 215, 220, 236, 237, 251, 270–272,
 276–278, 320
 Noise199, 208, 211, 212, 224, 235, 240, 250, 255, 274
 Nonlinear control182
 Nonlinearity 182, 209, 278
 Nonlinear modelling..... 320, 326
 Normalisation..... 38, 39, 99, 168, 211, 238, 251
 Nuclear magnetic resonance (NMR)..... 17–31, 173

O

Objective function185, 252, 257, 295, 296
 Octanol.....66, 70, 73, 83, 123
 OECD. *See* Organisation for Economic Co-operation and
 Development (OECD)
 Ontology 55, 57–59
 Open Babel.....126
 OpenEye126
 Opioid receptor327
 Optical lithography262, 264
 Organelle2, 3, 6
 Organisation for Economic Co-operation
 and Development (OECD).....68–84
 Oscillation 186, 187, 255
 Output layer53, 54, 96, 116, 151, 152, 183, 196,
 214, 215, 220, 241, 249, 270, 271, 321
 Overfitting.....97, 151, 199, 202, 253
 Overshoot..... 181, 187, 189, 241

P

Parity232, 233
 Parkinson's disease275, 276
 Partial least squares (PLS)52, 66, 74, 78, 80, 82,
 122, 136, 278, 328–330
 Pathogen..... 45, 46, 61
 Patient clustering..... 288, 289, 306, 314

- Pattern recognition 150, 196
PCA. *See* Principal component analysis (PCA)
PCC. *See* Pearson's correlation coefficient (PCC)
Pearson correlation 295, 296, 299, 303, 305
Pearson's correlation coefficient (PCC) 40, 90, 94, 304
Peptide 48, 89, 101–105, 107, 108, 111, 126
Personal authentication 205
Pesticide 75, 78, 79, 82, 329, 330
Pharmacology 103, 274
Pheromone 229
Phosphorylation 6, 12, 15
Photographic memory 229
Photoreceptor 228
Phototaxis 228
PLS. *See* Partial least squares (PLS)
P-Matrix 294, 298
Potentiation 261
Prediction 6, 10–13, 18–31, 34, 46, 47, 52, 57,
59, 60, 65–84, 89–99, 101–116, 126, 136,
149–162, 165–176, 183, 185, 210, 245–257, 270,
271, 274–282, 286–288, 291, 318, 325–330
Predictive control 199–192
Principal component analysis (PCA) 52, 111, 136,
278, 323, 326
Property prediction 65, 83
Prosthetic hand 179–192
Protein
 disorder 173
 scoring 174
 secondary structure 7, 26, 170, 172, 173
 sequence 1–15, 90, 165–168, 276
 sorting 3–5, 8, 12–14
 structure 15, 17–31, 90
 structure prediction 165–176, 275
 trafficking 3, 4
 transmembrane 4, 6, 10, 15
- Q**
mtk-QSBER 46, 47, 55–61
Quantitative structure-activity relationship
 (QSAR) 46–55, 60, 61, 65–84, 119,
 121–125, 127, 128, 130, 136, 137, 145, 149–154,
 156, 317–330
Quantitative structure-property relationship (QSPR) 67,
70–72, 119–145, 317–330
Quantization 209
Quantum mechanics (QM) 49–51
- R**
Radial basis function (RBF) 53, 54, 58, 59, 74, 135
Radial basis function network (RBFN) 68, 73–75,
78, 80, 82
Ramachandran map 19, 20, 23–25, 27
RBF. *See* Radial basis function (RBF)
RColorBrewer 104
REACH 65–84
Recapitulation 234, 240–242
Receiver operating characteristic (ROC) 59, 60, 197,
198, 200–203
Receptor binding 49, 73, 150, 154–156, 158,
160, 161, 327
Recurrent network 272, 274
Regression 37, 52, 55, 66, 78–80, 83, 96, 121,
130, 134, 136, 137, 150, 157, 162, 274, 278,
322–325, 327–330
Reshape 104
Residue 18–31, 166, 168, 169, 172, 174
Residue sequence 166
Resistor 182, 261
Resonance overlap 17
Response 52–54, 109, 121, 182, 186–188, 192,
220, 240, 241, 272, 286, 321
Restricted Boltzmann machine 236, 237
Ring current 18
Robot 231, 234
ROC. *See* Receiver operating characteristic (ROC)
Root mean squared deviation (RMSD) 96, 97
Rotamer 25, 26, 28
Route navigation 231, 235, 241, 242
R-Project 40, 116
RSNNS 104, 111
RStudio 104, 116
Runs test 278
- S**
Sample frequency 184
Sampling 34, 35, 172, 209, 241, 242, 249, 253, 254
Screening 46, 48, 50, 51, 55–61, 103, 150,
154–160, 275
SDMTTools 104
Security 205, 206, 246
Segmentation 199, 202, 203, 211
Self-organising map (SOM) 288–294, 296–310,
312–314
Sensitivity analysis 58
Sensitization 70, 76, 77
Sensor
 biometric 207–209
 glucose monitoring 246–248, 251, 253, 254, 257
Sensory array 228, 242
Sensory information 179–181, 228
Sequence analysis 5, 12–13
SEQUEST 89
Sheet 21, 263, 275
Sigmoid 54, 133, 152, 196, 214, 215
Sigmoid activation function 214, 215
Silicon 262, 263, 266
Similarity analysis 157

- Simulation90, 91, 93, 94, 99, 125, 207, 217,
221, 236, 319, 320, 325
- Single layer perceptron (SLP) 130, 133
- 6D-QSAR..... 49, 50
- SMILES..... 48, 128, 129
- Soft computing.....205–224
- Sorting signal.....2–8, 12–14
- SPSS.....322
- STATISTICA 54, 55, 58
- Strand.....275
- Structural factor.....17
- Structure activity cliff121
- Structure activity relationship (SAR)66, 67, 70, 79,
80, 120, 121, 125, 150, 158
- Subtype classification288, 314
- Support vector machine (SVM)54, 66, 68, 73,
77–80, 82, 124, 134–136, 174, 323
- Symbolic regression37
- Synergic effect8
- Systems.....10, 22, 33, 53, 105, 125, 130, 176, 182,
205, 207, 219, 228, 229, 242, 261, 262, 272–274,
318, 319, 322
- Systems biology.....33
- T**
- TALOS 18, 19, 21, 30
- Tantalum oxide..... 262, 263, 265, 266
- Taxonomic group..... 34, 35
- Testing set 95, 97, 111, 112, 116, 150, 151
- 3D-QSAR..... 49–52, 122, 150, 154, 156
- Threshold24, 109, 113, 114, 116, 133, 158, 199,
201, 212, 215, 223, 232–234, 247, 248, 255, 256,
272, 277, 295, 296, 319
- Time-delayed inputs.....183
- Time series 248, 249, 252–257
- Tolerance 90, 95, 186
- Top-down method.....200
- TOPKAT83
- Topological descriptors..... 50, 51, 55, 71, 74, 80, 83
- Topological indices.....51
- TOPS-MODE51
- Torsion angle 18–21, 26, 28–30
- Toxicity.....46, 47, 55, 56, 59, 60, 65–70, 76, 78–84,
120, 121, 123, 323, 325, 326
- Trafficking.....2–4, 7, 8, 10, 11, 15
- Training 9, 37, 57, 68, 92, 95–97, 104,
111–113, 133, 149, 167–168, 170, 172, 183, 196,
200–201, 220, 230, 235–239, 249, 252–253, 269,
291, 321
- Training image 230, 231, 233
- Training set.....42, 57, 59, 68, 69, 71–83, 95–97, 99,
111, 151, 152, 154, 155, 220, 232, 235, 237, 249,
251–253, 256, 276–278, 324
- Training subset37
- Transcription275
- Translation.....275
- Traversal 230, 231, 242
- True negative59, 60, 97, 113, 115
- True positive59, 60, 97, 113, 197
- Tuberculosis (TB).....46, 275, 276, 278–282
- Tunnel-junction.....261–267
- U**
- Ultrasound.....195, 275
- U-matrix.....293, 294, 307
- Uniprot.....106, 107
- V**
- Validation 9, 19, 20, 42, 68–70, 72, 75, 78, 79, 84, 98,
104, 111–114, 116, 121, 150–152, 155, 157, 158,
168, 196, 197, 200–202, 252–254, 314, 327, 328
- Validation subset.....37, 40
- Vancomycin46
- Variance.....113, 184, 202, 290, 293
- Vesicle.....2–4, 6, 8, 10, 15
- Virtual screening 48, 50, 51, 55–61, 150,
154–160, 275
- Viscosity71, 75
- Vision memory231
- Visual filter232, 239
- Visual navigation227–243
- Voice pattern recognition.....206
- Voiceprint.....207–213, 215, 218–221, 223, 224
- W**
- Ward clustering 293, 296, 306
- Ward distance.....293
- Water-solubility..... 70, 72–73, 120
- Watson173
- Weight.....80, 107, 116, 120, 129, 133, 134, 139,
171, 172, 175, 179, 184, 187, 196, 197, 200, 214,
220, 233, 237, 239, 241, 251–253, 273, 274, 276,
279–281, 291–294
- Weka55, 58, 124, 129, 130, 145
- Window.....22, 29, 51, 52, 137, 138, 141, 166–172,
175, 210, 218, 219
- Y**
- Yeast9, 14