Artificial Neural Networks

Artificial Neural Networks

Methods and Applications

David J. Livingstone
Editor

ChemQuest, Sandown, UK

Humana Press

Editor
David J. Lvingstone
ChemQuest
Sandown, Isle of Wight
United Kingdom, PO36 8LZ, UK

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

ISBN: 978-1-58829-718-1 e-ISBN: 978-1-60327-101-1
DOI: 10:1007/978-1-60327-101-1

Library of Congress Control Number: 2008929435

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

Artificial neural networks (ANN) is the name given to a branch of artificial intelligence (AI) research that aims to simulate intelligent behavior by mimicking the way that biological neural networks work. Most AI methods seek to reproduce human intelligence by imitating "what we do," ANN seek to reproduce it by imitating "the way that we do it." The origins of ANN predate computers by some decades, but it was not until computers became generally available that real progress could be made in the development of these methods. There was a slight "glitch" of a decade or so following the publication of a book that heavily criticized the possibility of ANN developing into anything useful; since then, progress has been dramatic and these tools have moved on from being oddities used by specialists to general-purpose algorithms for data analysis and pattern recognition tasks.

As with all new techniques, the adoption of ANN by scientists in other fields had a slow start, which soon accelerated. Their use in chemistry, for example, has been documented (see J. Zupan and J. Gasteiger, *Neural Networks in Chemistry*, Wiley-VCH, Weinheim, Germany) by the number of papers: 3 (1988), 5 (1989), 20 (1990), 105 (1991), 290 (1992), . . ., 927 (1997). *Artificial Neural Networks: Methods and Applications* reports the history of the application of ANN to chemical and biological problems, gives a guide to network architectures, training, and the extraction of rules from trained networks, and covers many examples of the application of ANN to chemistry and biology.

David J. Livingstone

Contents

Contributors

Igor I. Baskin, Dr, PhD
Department of Chemistry, Moscow State University, Moscow, Russia

Antony Browne, Dr.
Department of Computing, School of Engineering
and Physical Sciences, University of Surrey, Guildford, Surrey, UK

Frank Burden, BSc(Hons), PhD, MRACI
Scimetrics, Carlton North, Victoria, Australia

Hugh M. Cartwright, BSc, PhD
Department of Chemistry, University of Oxford, Physical and Theoretical Chemistry Laboratory, Oxford, UK

Raymond Crichton
Centre for Molecular Design, University of Portsmouth, Portsmouth, Hampshire, UK

James Devillers, Dr, PhD
CTIS, Rillieux La Pape, France

Yi Han, PhD
Discovery Chemistry, Hoffmann-La Roche Inc., Nutley, NJ, USA

Brian D. Hudson, BSc, PhD
Centre for Molecular Design, University of Portsmouth, Portsmouth, Hampshire, UK

Mehdi Jalali-Heravi, PhD
Department of Chemistry, Sharif University of Technology, Tehran, Iran

Brendan Larder, PhD
HIV Resistance Response Database Initiative, London, UK

David J. Livingstone, CChem FRSC
ChemQuest, Sandown, UK and Centre for Molecular Design,
University of Portsmouth, Portsmouth, Hampshire, UK

Marjana Novic, PhD
National Institute of Chemistry, Ljubljana, Slovenia

Vladimir A. Palyulin, Dr, PhD
Department of Chemistry, Moscow State University, Moscow, Russia

Marco Punta, PhD
Department of Biochemistry and Molecular Biophysics, Columbia University, and Columbia University Center for Computational Biology and Bioinformatics, New York, NY, USA

Andy Revell, PhD
HIV Resistance Response Database Initiative (RDI), London, UK

Burkhard Rost, PhD
Department of Biochemistry and Molecular Biophysics, Columbia University Center for Computational Biology and Bioinformatics, and North East Structural Genomics Consortium (NESG), Columbia University, New York, NY, USA

Sung-Sau So, PhD
Discovery Chemistry, Hoffmann-La Roche Inc., Nutley, NJ, USA

Igor V. Tetko, PhD
GSF – Institute for Bioinformatics, Neuherberg, Germany; and Institute of Bioorganic and Petrochemistry, Kiev, Ukraine

Dechao Wang, PhD
HIV Resistance Response Database Initiative (RDI), London, UK

David Whitley, Dr, PhD
Centre for Molecular Design, University of Portsmouth, Portsmouth, Hampshire, UK

David Winkler, PhD
CSIRO Molecular and Health Technologies, Clayton, Victoria, Australia

Zheng Rong Yang, PhD
School of Biosciences, University of Exeter, Exeter, UK

Nikolai S. Zefirov, Prof. Dr.
Department of Chemistry, Moscow State University, Moscow, Russia

Jinming Zou, PhD
Locus Pharmaceuticals, Inc., Blue Bell, PA, USA

Chapter 1
Artificial Neural Networks in Biology and Chemistry

The Evolution of a New Analytical Tool

Hugh M. Cartwright

Abstract Once regarded as an eccentric and unpromising algorithm for the analysis of scientific data, the neural network has been developed in the last decade into a powerful computational tool. Its use now spans all areas of science, from the physical sciences and engineering to the life sciences and allied subjects. Applications range from the assessment of epidemiological data or the deconvolution of spectra to highly practical applications, such as the electronic nose. This introductory chapter considers briefly the growth in the use of neural networks and provides some general background in preparation for the more detailed chapters that follow.

Keywords Artificial intelligence, neural network, analytical chemistry, chromatography, life sciences, mass spectrum, sensor, electronic nose, QSAR, olive oil

1. Early Artificial Intelligence Tools

Artificial intelligence (AI), a rapidly evolving tool for the treatment of scientific data, shows great promise in chemistry and biology. The earliest scientific applications of AI focused on the use of expert systems (knowledge-based systems). These computer programs, which encapsulate the knowledge of experts in a well-defined area, make quite modest demands on computing power. As the theoretical basis of expert systems is straightforward and the programs are easy to implement, it was natural that these should be among the first programs to be of value when AI was introduced into the physical and life sciences.

An expert system approach is appropriate only for certain types of problem, but within analytical chemistry in particular, these methods were widely deployed. The initial dominance of expert systems is reflected in an early text on the use of AI in chemistry [1]. Within Pierce and Hohne's book, which focused on expert systems, there is little to suggest that quite different and far more powerful methods of analysis were waiting in the wings. Nevertheless, techniques were emerging that would lead to a dramatic rise in the use of AI in science.

doi: 10/1007/978-1-60327-101-1 Springerprotocols.com

While expert systems would run adequately on the slow computers of the 1960s and 1970s, the computers then available were not sufficiently powerful to implement more demanding AI methods, such as genetic algorithms or artificial neural networks. Furthermore, the theoretical foundation of most AI algorithms at that stage was poorly developed. However, faster computers gradually found their way into the laboratory, and as understanding of the methods grew, the potential of AI within the scientific laboratory started to be recognized.

Goldberg's seminal book on genetic algorithms [2], which was itself greatly influenced by Holland's earlier text [3], suggested that the revolutionary tools that computer scientists were developing in due course would reach out into other branches of science. Two years before Goldberg's book, Bounds' brief but notable review in *Nature* gave a clue to the rosy future for these methods [4], and shortly thereafter Zupan and Gasteiger [5, 6] were arguing that chemistry was a fertile field for the use of neural networks. An early survey of the potential of AI methods within chemistry [7] hinted at the range of applications about to emerge.

We have come a long way. Within 20 years, AI has been transformed from a method widely perceived by physical and life scientists as an exotic, eccentric, even daft method of solving scientific problems, into one whose power is widely recognized and that is being employed across all scientific fields.

There are many reasons for this transformation in attitude among scientists. The most simple, and yet most fundamental, is that AI methods work. When scientists first encounter neural networks, expert systems or genetic algorithms, a common reaction is to wonder whether algorithms that seem at their core to be so straightforward can really succeed where more sophisticated algorithms may fail. No computational technique provides a magic bullet, capable of slaying any scientific problem, and the literature contains examples of the unconvincing application of AI to problems that are actually more amenable to other techniques. Nevertheless, numerous areas exist in science for which AI methods are now the method of choice; this text illustrates just how wide is that spread of areas.

Each AI method has its own particular range of applicability, and the algorithm most likely to be effective is strongly correlated with the nature of the problem. The structure of genetic algorithms renders them particularly well-suited to ordering or sorting tasks, so they have been used in problems such as flowshop scheduling [8] and the prediction of protein structure. Knowledge-based systems encapsulate the knowledge of experts in a way that other techniques still find hard to match; such algorithms have a long and successful track record in analytical chemistry (see, for example, [9] and references therein). Self-organizing maps, which flatten multidimensional data onto a smaller number of dimensions, apply in a more restricted field, but can act as powerful tools in categorization problems. Neural networks, the focus of this text, have a happy ability to discover the patterns and rules that are buried so deeply within large data sets that, to a casual observer, the data may seem almost random in nature.

2. Neural Networks

The key characteristic of a neural network is its ability to learn. If a convenient mathematical model that describes a data set is already known, a neural network is unlikely to be needed, but when the rules that underlie the data are known only partially, or not at all, a neural network may discover interesting relationships as it rambles through the database. Neural networks are able to learn complex behavior and are highly adaptable; even a brief study shows them to be capable of undertaking a remarkable variety of tasks. They have been used to identify fingerprints and to recognize objects on production lines, to determine whether a bank customer would be a good risk for a loan and whether, on the basis of an X-ray, a hospital patient might have fractured a limb. In more outré areas, they have been used to predict the box-office success of motion pictures [10] and even (unsuccessfully, so far) to predict winning lottery numbers.

However, despite their wide applicability, neural networks also have drawbacks. Most notably, they are "black box" solvers. Although "rule discovery networks," which can identify and report the rules that underlie a data set, show great potential, these are still some way from being a routine laboratory tool. Therefore, if we need to know the analytical form of relationships that link samples in a database, neural networks may not be the most suitable tool. In other areas though, such as sensors (so-called electronic noses), reliability of prediction is the key; a knowledge of how a prediction was made is unimportant, and the opaque nature of the algorithm's decision making is of no consequence, as long as the algorithm can demonstrate that its conclusions are trustworthy.

What characteristics of neural networks give them such potential in science? To begin with, both their theoretical basis and their practical implementation are well understood, and numerous texts and papers provide a suitable introduction to the principles (for example, see [6, 11–13]). Also, numerous reports in the literature compare the performance of neural networks with other methods [14–18]. There is thus plenty of supporting material available to the newcomer in the field.

Their broad range of applicability has already been noted. The scientific tasks to which they are now applied are so varied that this brief introduction can only hint at this richness. Later chapters provide a deeper discussion of some of the fascinating and productive scientific areas in which neural networks are now used.

3. Neural Networks in Analytical Chemistry

The most characteristic property of neural networks is their ability to discern patterns. Science is replete with patterns and images, from electron micrographs to NMR spectra; and assessment of these is a particular strength of neural networks. A typical example of spectral analysis (Fig. 1.1), which is significant as an early real-world application of the method, is the assessment of olive oils [19].

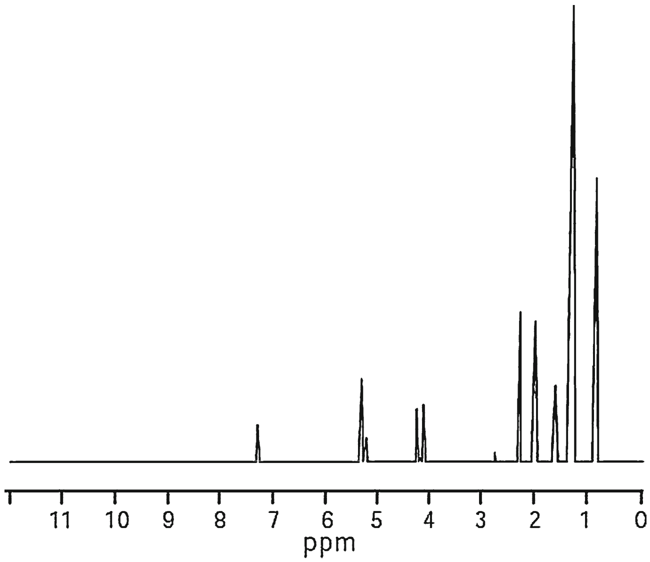


Fig. 1.1 The NMR spectrum of a typical olive oil. The spectrum appears quite simple, but at higher resolution, it can be seen to be composed of a very large number of peaks. The high degree of overlap that occurs in such a spectrum renders analysis challenging

The composition of olive oil is uniquely related to the region from which it is derived. Top-quality oils are much more valuable than oils of ordinary quality, and it has been known for a dishonest producer to dilute a good oil with poorer oils, hoping that the adulterated product could still be sold at a premium price. A need exists to identify inferior oils passed off as premium product and to offer a tool to allow the producers of the best oils to demonstrate the quality of their products.

A complex natural product, olive oil yields GC, MS, IR, and NMR spectra that are rich with information. The spectra of different oils display numerous subtle differences and, because of their complexity, are difficult to interpret in full. The spectra encode the unique differences in the characteristics of the oils and can be used to identify oils from different regions, but the task of reaching a reliable assessment of the quality of the oil from spectra is challenging. This problem has been revisited in the literature many times, among the more recent contributions being that of Rezzi and coworkers [20].

Both standard feedforward neural networks and Kohonen self-organizing maps can be used to study olive oils. In a conventional feedforward network, the more widely used of the methods discussed in this text, by inspecting representative samples of oil, the network develops rules that allow it to distinguish between oils of different quality, even when the spectra are very similar. Taking a different approach, the Kohonen network reduces multidimensional data (in which the characteristics of different samples depend on many variables, such as the different wavelengths for which intensity data are available in infrared spectra) to a smaller number of dimensions. Its role is thus that of simplification. The spectral data are compressed onto a (generally) two-dimensional field that clusters oils of similar quality in neighboring regions of the map.

The assessment of olive oils is a problem in analytical chemistry, an area that has proven to be very productive for neural network applications. Experiments in analytical chemistry often generate large volumes of data. For example, infrared and UV/visible spectra consist of many intensity-wavelength data pairs, and the mass spectrum of a biological or medical sample may contain thousands, or even tens of thousands, of peaks (Fig. 1.2). There is generally considerable redundancy in these types of spectra, but despite this (or, sometimes, because of it) a complete analysis of such a spectrum is challenging. Networks used as pattern recognition algorithms can help in the global analysis of spectra or can help to locate those areas or peaks in the spectrum that are of particular diagnostic value.

A second area of analytical chemistry that has developed rapidly in the past decade, and in which neural networks have been widely employed, is that of sensor technology. Although chemists might claim this area for their own, it is of increasing importance in the life and environmental sciences. Sensors provide a route into the determination of a wide variety of materials (Fig. 1.3) within chemistry, the food industry, engineering, and the environmental and biological sciences [21]. Sensor instrumentation often takes the form of "electronic noses," in which several different sensors feed their output to a single neural network, which in turn provides rapid quantitative data.

Recent work by Ozmen and coworkers [22], in which an array of phthalocyanine-coated QCM sensors was used to determine the composition of a mixture of gases, is a straightforward example of this approach. A more challenging problem was tackled by Dutta and coworkers [23], who used electronic noses to assess *Staph. aureus* infections in hospital environments. Sensors of this sort are far more convenient in use than a conventional GC or GC/MS analysis and can compete with the reliability (though not always the sensitivity) of these well-established methods.

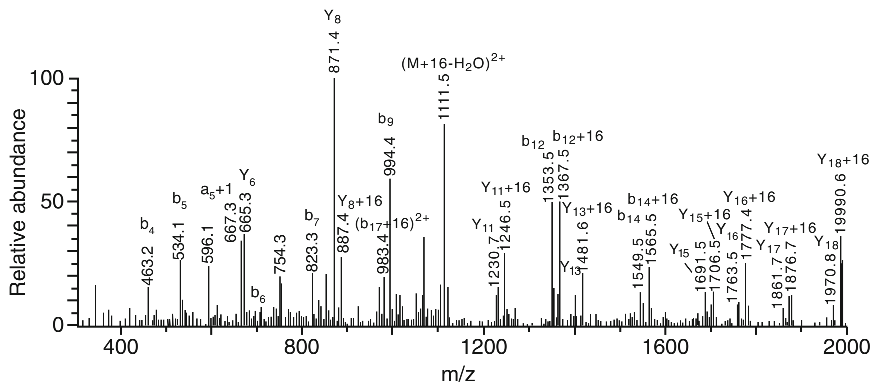


Fig. 1.2 Tandem mass spectrum of double-charged bovine RNase-S tryptic peptide 85-104

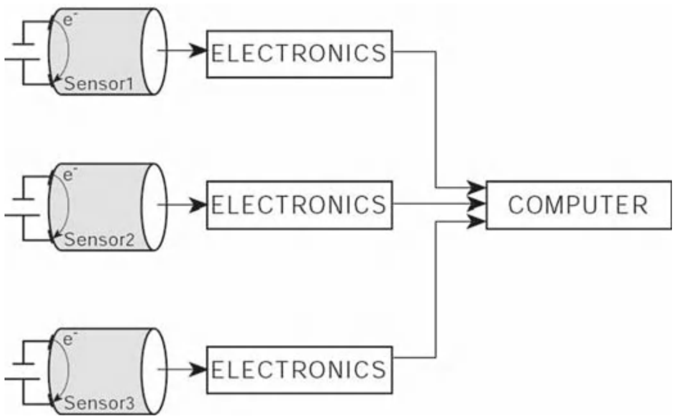


Fig. 1.3 A diagrammatic representation of a sensor array. Sensor elements are covered with various types of membranes which are semipermeable to the analyte gases. Different sensors respond in distinct ways to the presence of each analyte, and a trained neural network, fed the output of the sensors, can be used to provide a rapid analysis of the sample

In studies of sensor arrays [24], techniques such as principal components analysis (PCA) and principal components regression (PCR) have been widely used, but there must be doubts whether such linear models are entirely suitable, since the data that sensors generate are to some degree nonlinear. Instead, the nonlinear advantages of neural networks [25, 26] have been used with success.

More traditional applications of analytical chemistry, such as determining the concentration of metals ions in a solution, are also amenable to a neural network approach, as illustrated by the reports of Lubal and coworkers [27] in the analytical determination of molybdenum (VI) and tungsten (VI).

4. Neural Networks in Property Prediction

Neural networks have an inherent ability to model nonlinear data. Since so many scientific relationships are nonlinear, this is a crucial advantage. In the past, linear QSARs (quantitative structure-activity relationships) and QSPRs (quantitative structure-property relationships) have been used to quantitatively link molecular properties, such as dipole moment or shape, to biological and other types of activity, but there is little theoretical justification for the view that linear models are the most effective. Consequently, this has proven to be a productive field for the nonlinear abilities of networks. Devillers proposed a new QSAR model for use in environmental applications [28], in which residuals from a linear regression equation on the water/octanol partition coefficients are modeled using a neural network that takes molecular descriptors as input; 569 organic materials were used in the study.

Cartwright [29] used self-organizing maps to investigate the relationship between structure and biodegradability in PCBs (polychlorinated biphenyls), which are important environmental pollutants. This followed earlier work by Zitko [30] with a similar aim. Agreement to within 25% between experimental and predicted biodegradability was obtained in most cases, and it was concluded that prediction of the biodegradability of disubstituted PCBs was likely to be more challenging than that of other types of PCB.

The choice of suitable descriptors to characterize the molecules in QSAR and QSPR studies is crucial [31], and a wide variety of descriptor sets and methods for selecting them have been proposed [32]. A pruning or optimization algorithm may be useful to determine an optimal set of descriptors; the CODES procedure described by Stud [33] is one of many that has found some success. (The choice of descriptors in fact is an example of a far more fundamental problem in the application of neural networks and other AI methods: the question of what data should be presented to the algorithm and how that data should be encoded. In evolutionary algorithms, this is a question often encountered when the data are real valued, where there remains disagreement about whether binary coding or gray coding is more appropriate [34]).

Many groups have investigated the prediction of toxicity or biological activity using neural networks, a QSAR problem that spans the life sciences and chemistry. QSAR studies of MAO inhibitors have been carried out by Cherqaoui and coworkers [35], the toxicity of organophosphorus insecticides was studied by Devillers [28], the biological activity of a large set of dihydrofolate reductase inhibitors was investigated by Burden [36], and Halberstam and coworkers [37] assembled an extensive review of the use of QSPR for organic compounds. The number of studies in this area runs well into three figures and suitably illustrates the value of networks in understanding and predicting toxicity and biological activity.

Biological data sets are often very large, and data compression may be required as the first step in analysis. Effective compression may be carried out using autoassociative artificial neural networks (AAANNs), in which data are fed through a hidden layer bottleneck. In the past, data reduction of MS spectra has been accomplished mainly through the use of principal component analysis [38, 39] but Goodacre enthusiastically championed the use of neural networks for this purpose [40] and Jacobsson [41] investigated the compression of the IR spectra of polysaccharides using AAANNs.

A key difficulty in the use of all forms of neural networks is that the size of the network has a substantial influence on the value of the completed network as a tool, but this parameter is not known in advance. Some authors generate a series of networks of differing geometry to determine the optimum parameters, but this trial-and-error approach is cumbersome and time consuming. An alternative is to evolve both network weights and the network geometry as the training takes place. This optimization is an important consideration for both conventional and Kohonen networks, and research continues on ways in which the geometry and the weights can best be optimized simultaneously [42]. A promising recent method in this area is the use of genetic neural networks, which have been widely applied to allow the network to optimize its own geometry and learning parameters while limiting overlearning. Such a system also can be used to locate an optimum descriptor set for QSAR studies.

5. Biomedical, Spectroscopic, and Other Applications

In self-organizing maps, an injudicious choice of map size can seriously reduce the effectiveness of the algorithm. A number of methods exist in which the algorithm locates its own optimum size. In the "growing grid" method, columns or rows are added as the calculation proceeds [43], while in "incremental grid growing" new nodes are generated at the boundaries of the map [44]; this latter procedure can lead to disjoint networks in which the network fragments into several separate grids. In the "growing neural gas algorithm" [45], the authors use Hebbian techniques to insert new nodes where the accumulated error is greatest.

In an early application of growing cell structure networks, Walker, Cross, and Harrison [46] used visualization of biomedical data that linked patient age and ten further parameters with breast cancer data. They showed a convincing division between positive and negative posterior probabilities for breast cancer cases. The growing cell structure (GCS) method starts with a primitive triangular network; during training, nodes may be added but also removed [47]. Sammon's mapping has been applied to the assessment of similar data sets; Wu and Yen [48] compared this technique to GCS and showed a good separation of data derived from the chemical abstracts of papers relating to polymer cements. Fonseca's group [49] investigated the use of self-organizing maps in the identification of crude oils, following work by Cartwright [50], who used neural networks to identify degraded oil spills. Fonseca's group used GC-MS data to assess a data set containing 188 samples; the trained network was able to identify roughly two thirds of samples contained within the test set.

Spectroscopy has proved a fertile field for applications. Zhang and coauthors [51] investigated a problem common to many spectroscopic techniques, that of overlapping spectral features. A rich variety of methods exists to tackle such problems, including neural networks, but any type of procedure that relies on learning must be applied with care, since the quantity of data specifying each spectrum gives rise to the possibility of overfitting in trained methods [52]. Zhang's group investigated three types of networks in the quantification of overlapped peaks in micellar electrokinetic capillary chromatography. They assessed spectra first simplified by a PCA preanalysis and applied the method to mixtures of vitamin B1 and diazolium, whose UV absorption spectra are very similar. This work extended similar earlier investigations into the use of neural networks in chromatography [53–55].

The effectiveness of gradient elution in preparative liquid phase chromatography can be improved by adjusting the solvent composition as elution proceeds, and because of the importance of this technique in synthetic chemistry and in the analysis of complex mixtures, there is a considerable literature on the subject (see, for example

[56] and references therein). Although models exist to predict chromatographic concentration profiles, under overloaded conditions, adsorption isotherms are non-linear and numerical solution of the models is required [57]. Both linear and step-function changes in composition are used in practice, but the precise experimental procedure is often determined empirically. Parameters in elution models can be optimized by feeding them into a network for which the objective function is calculated by the Craig model [58]. Shan and Seidel-Morgenstern [59] used this approach to optimize the gradient profiles in preparative liquid chromatography for the separation of ternary mixtures. They found, not unexpectedly, that the optimum conditions for component separation are particularly sensitive to the way that components with similar retention times respond to changes in experimental parameters.

A number of groups investigated the applicability of neural networks to the study of fuels and prediction of their properties. Most fuels are mixtures of many components in varying proportions, with a correspondingly wide range of properties. Santana and coworkers [60] applied neural networks to the prediction of the cetane number of different components in diesel fuels. Early attempts to predict the octane number of fuels using networks [61] have been built upon by various authors, including Basu [62], who used neural networks to analyze data from NMR, LC, and GCMS experiments. Pasadakis, Sourligas, and Foteinopoulos [63] considered the prediction of distillation curves, pour, and cloud points for cold diesels, with a view to the use of such data as a tool for quality control in the production of diesel, using infrared spectra as the input.

Several reviews provide an overview of the application of neural networks to the life sciences and medicine. Some, such as those by Ochoa, Chana, and Stud [64] and Agatonovic-Kustrin and Beresford [65], provide a general review of the field, but more specialized reviews have also been published. Many applications of neural networks focus on quantitative structure–activity relationships applied to medical or biological systems, and the area has been reviewed by several authors [66, 67]. Molnar and coworkers [68] claim high accuracy in their prediction of cytotoxicity, using a large set of 30,000 druglike compounds as input to a feedforward neural network (though they report little data and it will be easier to judge the success of their approach when a larger set of results is published).

The solubility of drugs is an important factor determining their effectiveness, and Balakin, Savchuk, and Tetko [69] recently reviewed the various models available for this purpose, focusing on the assessment of solubility in water and DMSO. Hernandez-Carabello and Marco-Parra [70] are among those who used a back-propagation neural network to categorize medical data. Although 96% of the samples in their prediction set were classified correctly, the small size of the sample (27 cancer patients and 32 healthy volunteers) limited the usefulness of the study.

Biological systems are invariably complex, so sophisticated experimental as well as algorithmic techniques are required. Pyrolysis-MS samples undergo thermal decomposition in a vacuum and the fragments are then delivered to a mass spectrometer. This produces complex fragmentation patterns that serve as distinctive fingerprints for specific biological molecules [71]. It has been claimed that identification of these fingerprints by neural networks is readily accomplished and that this provides a powerful means of identification. Pyrolysis-MS has also been used for identifying bacterial strains of industrial importance [72].

Neural networks are increasingly used in the analysis of food and drink and in online monitoring of its quality. A recent example is a study by Fernandez-Pachon et al. [73], who combined multiple regression analysis with neural networks to determine the relation between phenolic composition and antioxidant activity, an area previously studied by other groups [74, 75]. O'Farrell and coworkers [21] analyzed visible spectra from the surface of food as it is being cooked to assess online quality control of food in large-scale industrial ovens, measuring the color of the surface, comparing the success of an AI approach to k-nearest neighbor clustering.

The picture that emerges from this brief overview of the use of neural networks in chemistry and biology is rich but fragmented. Even the few examples mentioned, however, make it clear that the numerous groups working in the area are tackling a very broad spectrum of problems; the research effort is increasing rapidly. There remains a need for a wider appreciation of both the limitations and the capabilities of neural networks, so that they can be applied effectively in the most promising areas; this is just what the chapters that follow provide.

References

1. Pierce, TH, Hohne, BA (eds) (1986) Artificial intelligence applications in chemistry. ACS Symposium Series 306, American Chemical Society, Washington, DC.
2. Goldberg, DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, MA.
3. Holland, JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor.
4. Bounds, DG (1987). New optimization methods from physics and biology. Nature, 329: 215–219.
5. Zupan J, Gasteiger J (1991) Neural networks-a new method for solving chemical problems or just a passing phase? Anal Chim Acta 248:1–30.
6. Zupan J, Gasteiger J (1993) Neural networks for chemists—an introduction. VCH, Weinheim.
7. Cartwright HM (1993) Applications of artificial intelligence in chemistry. Oxford University Press, Oxford.
8. Cartwright HM, Long RA (1993) Simultaneous optimization of flowshop sequencing and topology using genetic algorithms. Ind Eng Chem Res 32:2706–2713.
9. Lahiri S, Stillman MJ (2000) Knowledge transfer: human experts to expert systems. In: Cartwright HM (ed.) Intelligent data analysis in science. Oxford University Press, Oxford, pp. 19–43.
10. Sharda R, Delen D (2006) Predicting box-office success of motion pictures with neural networks. Expert Systems with Applications 30:243–254.
11. Wasserman PD (1989) Neural computing: theory and practice. Van Nostrand Reinhold, New York.
12. Sumpter BG, Getino C, Noid DW (1994) Theory and applications of neural computing in chemical science. Ann Rev Phys Chem 45:439–481.
13. Peterson K (2000) Artificial neural networks and their use in chemistry. In: Lipkowitz KB, Boyd DB (eds) Reviews in computational chemistry. Wiley-VCH, New York, pp. 53–140.

14. Eldred DV, Weikel CL, Jurs P, Kaiser KLE (1999) Prediction of fathead minnow acute toxicity of organic compounds from molecular structure. Chem Res Toxico 12:670–678.
15. Pantakar SJ, Jurs PC (2000) Prediction of IC50 values for ACAT inhibitors from molecular structure. J Chem Inf Comput Sci 40:706–723.
16. Lucic B, Trinajstic N J (1999) Multivariate regression outperforms several robust architectures of neural networks in QSAR modelling. J Chem Inf Comput Sci 39:121–132.
17. So S-S, Karplus M J (1999) A comparative study of ligand-receptor complex binding affinity prediction methods based on glycogen phosphorylase inhibitors. J Comput-Aided Mol Des 13:243–258.
18. Beck B, Glen R, Clark TJ (1996) The inhibition of α-chymotrypsin predicted using theoretically derived molecular properties. J Mol Graph 14:130–135.
19. Goodacre R, Kell DB, Bianchi G (1992) Neural networks and olive oil. Nature 359:594.
20. Rezzi S, Axelson DE, Heberger K, Reniero F, Mariani C, Guillou C (2005) Classification of olive oils using high throughput flow ^{1}H NMR fingerprinting with principal component analysis, linear discriminant analysis and probabilistic neural networks. Anal Chim Acta 552:13–24.
21. O'Farrell M, Lewis E, Flanagan C, Lyons W, Jackman N (2005) Comparison of k-NN and neural network methods in the classification of spectral data from an optical fibre-based sensor system used for quality control in the food industry. Sensors and Actuators B 111–112:254–362.
22. Ozmen A, Tekce F, Ebeolgu MA, Tasaltin C, Ozturk ZZ (2006) Finding the composition of gas mixtures by a phthalocyanine-coated QCM sensor array and an artificial neural network. Sensor and Accuators B 115:450–454.
23. Dutta R, Morgan D, Baker N, Gardner JW, Hines EL (2005) Identification of *Staphylococcus aureus* infections in hospital environment: electronic nose based approach. Sensors and Actuators B 109, 355–362.
24. Hodgins D, Simmonds D (1995) The electronic NOSE and its application to the manufacture of foods. J. Automated Chemistry 17:179–185.
25. Auge J, Hauptmann P, Hartmann J, Rosler S, Lucklum R (1995) Versatile microcontrolled gas sensor array system using the quartz microbalance principle and pattern recognition methods. Sensors and Actuators B 26:181–186.
26. Xing W-L, He X-W (1997) Crown ether-coated piezoelectric crystal sensor array for detection of organic vapour mixtures using several chemometric methods. Analyst 122:587–592.
27. Lubal P, Koprivova H, Sedo O, Havel J, Lis S, But S (2006) Simultaneous determination of molybdenum (VI) and tungsten (VI) and its application in elemental analysis of polyoxometalates. Talanta 69:800–806.
28. Devillers J (2000) Prediction of toxicity of organophosphorus insecticides against the midge, *Chironomus riparius*, via a QSAR neural network model integrating environmental variable. Toxicol Methods 10:69–79.
29. Cartwright HM (2002) Investigation of structure-biodegradability relationships in polychlorinated biphenyls using self-organising maps. Neural Comput & Applic 11:30–36.
30. Zitko V (1991) Prediction of the biodegradability of organic chemicals by an artificial neural network. Chemosphere 23:305–312.
31. Grover M, Singh B, Bakshi M, Singh S (2000) Quantitative structure-property relationships in pharmaceutical research-part 1. PSTT 3:28–35.
32. Viswanadhan VN, Ghose AK, Revankar GR, Robins R (1989) Atomic physicochemical parameters for three dimensional structure directed quantitative structure-activity relationships. 4. Additional parameters for hydrophobic and dispersive interactions and their application for an automated superposition of certain naturally occurring nucleoside antibiotics. J Chem Inf Comput Sci 29:163–172.
33. Stud M (1993) Neural networks in medicinal chemistry: encoding graphic structures and their prediction capability. In: VII congreso de la sociedad espanola de quimica terapeutica, September 28–30, Salamanca, Spain.

34. Caruana R, Schaffer JD (1988) Representation and hidden bias: gray vs. binary coding for genetic algorithms. In: Proc. Vth int. conf. on machine learning, Morgan Kaufmann, San Mateo, CA, pp. 132–161.
35. Cherqaoui D, Esseffar M, Zakarya D, Mesbah A, Villemin D (1998) Structure-selectivity relationships of MAO inhibitors using neural networks. ACH-Models Chem 135:79–92.
36. Burden FR (1996) Using artificial neural networks to predict biological activity from simple molecular structural considerations. Quant Struc-Act Relat 15:7–11.
37. Halberstam NM, Baskin II, Palyulin VA, Zefirov NS (2003) Neural networks as a method for elucidating structure-property relationships for organic compounds. Uspekhi Khimii 72:706–727.
38. Jolliffe IT (1986) Principal components analysis. Springer-Verlag, New York.
39. Everitt BS (1993) Cluster analysis. Edward Arnold, London.
40. Goodacre R (2000) Applications of artificial neural networks to the analysis of multivariate data. In: Cartwright HM (ed.) Intelligent data analysis in science, Oxford University Press, Oxford, pp. 123–152.
41. Jacobsson SP (1994) Feature extraction of polysaccharides by low-dimensional internal representation neural networks and infrared spectroscopy. Anal Chim Acta 291:19–27.
42. Palmes PP, Usui S (2005) Robustness, evolvability and optimality of evolutionary neural networks. BioSystems 82:168–188.
43. Fritzke B (1995) Growing grid-a self-organizing network with constant neighborhood range and adaption strength. Neural Processing Letters 2:9–13.
44. Blackmore J, Miikkulainen R (1995) Visualizing high-dimensional structure with the incremental grid growing network. In: Proc. XIIth internat. conf. on machine learning. Morgan Kaufmann, San Francisco, pp. 55–63.
45. Martinetz M, Schulten KJ (1991) A neural gas network learns topologies. In: Kohonen KMT, Simula O, Kangas J. (eds) Artificial neural networks. North Holland, Amsterdam, pp. 397–402.
46. Walker AJ, Cross SS, Harrison RF (1999) Visualisation of biomedical datasets by use of growing cell structure networks: a novel diagnostic classification technique. The Lancet 354:1518–1521.
47. Wong JWH, Cartwright HM (2005) Deterministic projection by growing cell structure networks for visualization of high-dimensionality datasets. J Biomed Inform 38:322–330.
48. Wu Z, Yen G (2003) A SOM projection technique with the growing structure for visualising high-dimensional data. Int J Neural Systems 13:353–365.
49. Fonseca AM, Biscaya JL, Aires-de-Sousa J, Lobo AM (2006) Geographical classification of crude oils by Kohonen self-organizing maps. Anal Chim Acta 556:374–382.
50. Cartwright HM (2008) Neural network analysis of the degradation of oil spills. (In preparation).
51. Zhang X, Li H, Hou A, Havel J (2006) Artificial neural networks based on principal component analysis input selection for quantification in overlapped capillary electrophoresis peaks. Chemometr Intell Lab Syst 82:165–175.
52. Tetko IV, Luik AI, Poda GI (1993) Applications of neural networks in structure-activity relationships of a small number of molecules. J Med Chem 36:811–814.
53. Havel J, Pena-Mendez EM, Rojas-Hernandez A, Doucet J-P, Panaye A (1998) Neural networks for optimization of high-performance capillary zone electrophoresis methods: a new method using a combination of experimental design and artificial neural networks. J Chromatogr A 793:317–329.
54. Yannis LL (2000) Artificial neural networks in liquid chromatography: efficient and improved quantitative structure-retention relationship models. J Chromatogr A 904:119–129.
55. Polaskova P, Bocaz G, Li H, Havel J (2002) Evaluation of calibration data in capillary electrophoresis using artificial neural networks to increase precision of analysis. J Chromatogr A 979:59–67.
56. Guiochon G, Shirazi SG, Katti AM (1994) Fundamentals of preparative and nonlinear chromatography. Academic Press, Boston.

57. Guiochon GJ (2002) Preparative liquid chromatography. J Chromatogr A 965:129–161.
58. Craig LC (1944) Identification of small amounts of organic compounds by distribution studies. II. Separation by counter-current distribution. J Biol Chem 155:519–534.
59. Shan Y, Seidel-Morgenstern A (2005) Optimization of gradient elution conditions in multi-component preparative liquid chromatography. J Chromatography A 1093:47–58.
60. Santana RC, Do PT, Santikunaporn M, Alvarez WE, Taylor JD, Sughrue EL, Resasco DE (2006) Evaluation of different reaction strategies for the improvement of cetane number in diesel fuels. Fuel 85:643–656.
61. Billingsley D (1995) Octane prediction of gasoline blends using neural nets. Proc NPRA Comput Conf.
62. Basu B, Kapur GS, Sarpal AS, Meusinger R (2003) A neural network approach to the prediction of cetane number of diesel fuels using nuclear magnetic resonance (NMR) spectroscopy. Energy and Fuels 6:1570–1575.
63. Pasadakis N, Sourligas S, Foteinopoulos C (2006) Prediction of the distillation profile and cold properties of diesel fuels using mid-IR spectroscopy and neural networks. Fuel 85:1131–1137.
64. Ochoa C, Chana A, Stud M (2001) Applications of Neural Networks in the Medicinal Chemistry Field. Curr Med Chem-Central Nervous System Agents 1:247–256.
65. Agatonovic-Kustrin S, Beresford R (2000) Basic concepts of artificial neural network (ANN) modelling and its application in pharmaceutical research. J Pharm Biomed Anal 22:717–727.
66. Kovesdi I, Dominguez-Rodriguez MF, Orfi L, Naray-Szabo G, Papp JG, Matyus P (1999) Application of neural networks in structure-activity relationships. Med Res Rev 19:249–269.
67. Manallack DT, Livingstone DJ (1999) Neural networks in drug discovery: have they lived up to their promise? Eur J Med Chem 34:195–208.
68. Molnar L, Kereru GM, Papp A, Lorincz Z, Ambrus G, Darvas F (2005) A neural network based classification scheme for cytotoxicity predictions: validation on 30,000 compounds. Bioorg & Med Chem Letts 16:1037–1039.
69. Balakin KV, Savchuk NP, Tetko IV (2006) *In silico* approaches to prediction of aqueous and DMSO solubility of drug-like compounds: trends, problems and solutions. Curr Med Chem 13:223–241.
70. Hernandez-Caraballo EA, Marco-Parra LM (2003) Direct analysis of blood serum by total reflection X-ray fluorescence spectrometry and application of an artificial neural network approach for cancer diagnosis. Spectrochimica Acta Part B 58:2205–2213.
71. Goodacre R, Neal MJ, Kell DB, Greenham LW, Noble WC, Harvey RG (1994) Rapid identification using pyrolysis mass spectrometry and artificial neural networks of *Propionibacterium acnes* isolate from dogs. J Appl Bacteriology 76:124–134.
72. Nilsson T, Bassani, MR, Larsen TO, Montanarella L (1996) Classification of species of the genus *Penicillium* by Curie Point pyrolysis mass spectrometry followed by multivariate analysis and artificial neural networks. J Mass Spectrom 31:1422–1428.
73. Fernandez-Pachon MS, Villano D, Troncoso AM, Garcia-Parilla MC (2005) Determination of the phenolic composition of sherry and table white wines by liquid chromatography and their relation with antioxidant activity. Anal Chim Acta 563:101–108.
74. Cao G, Sofic E, Prior RL (1997) Antioxidant and prooxidant behaviour of flavonoids: structure-activity relationships. Free Radic Biol Med 22:749–760.
75. Heijnene CGM, Haenen GRMM, Van Acker FAA, Van Der Vijgh W, Bast A (2001) Flavonoids as peroxynitrite scavengers: the role of the hydroxyl groups. Toxicol in Vitro 15:3–6.

Chapter 2
Overview of Artificial Neural Networks

Jinming Zou, Yi Han, and Sung-Sau So

Abstract The artificial neural network (ANN), or simply neural network, is a machine learning method evolved from the idea of simulating the human brain. The data explosion in modern drug discovery research requires sophisticated analysis methods to uncover the hidden causal relationships between single or multiple responses and a large set of properties. The ANN is one of many versatile tools to meet the demand in drug discovery modeling. Compared to a traditional regression approach, the ANN is capable of modeling complex nonlinear relationships. The ANN also has excellent fault tolerance and is fast and highly scalable with parallel processing. This chapter introduces the background of ANN development and outlines the basic concepts crucially important for understanding more sophisticated ANN. Several commonly used learning methods and network setups are discussed briefly at the end of the chapter.

Keywords Transfer function, Hopfield network, Kohonen network, perceptron, unsupervised learning, supervised learning

1. Introduction

Modern computers can perform complicated numerical and symbolic computations at amazing speed. But they are nowhere near the performance of human brains to perform perceptual tasks such as language and image recognition. Computers require precise input information and follow the instructions sequentially, while the human brains perform tasks via distributed and parallel fashion. Designing artificial intelligence based on biological neural network gives birth to an artificial neural network (ANN). In this chapter, we first discuss the biological neural networks and follow with an overview of ANN. Finally, we review several examples of neural networks.

2. Biological Neural Network

The artificial neural network is inspired by the architecture of biological neurons such as the human brain. The human brain is composed of a very large number of interconnected neurons. Each neuron is a cell that performs a simple task, such as a response to an input signal. However, when a network of neurons are connected together, they can perform complex tasks, such as speech and image recognition, with amazing speed and accuracy. Usually, it takes a few hundred milliseconds for a human to complete a task such as recognizing a face, while the brain's individual neuron has a computing speed of a few milliseconds. This shows that the brain takes just a hundred computing steps to perform such task [1], compared to millions of steps needed for a computer to perform a similar task. Such short processing time implies that the information transmitted between neurons must be very small. Unlike the traditional computer, the whole information is not passed from neuron to neuron but encoded in the complex interconnection of the neuron network. This is why neural network is also called connectionism.

A neuron is consisted of dendrites, a cell body, and an axon as shown in Fig. 2.1. The cell body, also called the *soma*, contains the nucleus, just like other cells. The dendrites are branches that connected to the cell body and extended in space to receive signals from other neurons. The axon is the transmitter of the neuron. It sends signals to neighboring neurons. The connection between the end of one neuron's axon and the neighboring neutron's dendrites is called the *synapse*, which is the communication unit between two neurons. Electrochemical signals are transmitted across the synapse. When the total signal a neuron received is greater than the synapse threshold, it causes the neuron to fire, that is, send an electrochemical

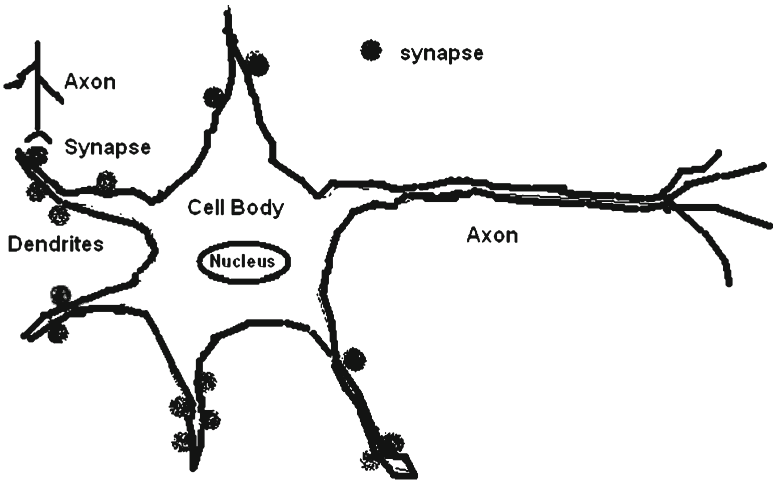


Fig. 2.1 A sketch of biological neuron

signal to neighboring neurons. It is postulated that the altering of the strength of the synaptic connection is the basis of memory [2].

3. Overview of an ANN

The artificial neural network is modeled on the biological neural network. Like the biological neural network, the ANN is an interconnection of nodes, analogous to neurons. Each neural network has three critical components: node character, network topology, and learning rules. Node character determines how signals are processed by the node, such as the number of inputs and outputs associated with the node, the weight associated with each input and output, and the activation function. Network topology determines the ways nodes are organized and connected. Learning rules determine how the weights are initialized and adjusted. The following sections explain each of the components in detail.

3.1. Node Character

The basic model for a node in the ANN is shown in Fig. 2.2. Each node receives multiple inputs from others via connections that have associated weights, analogous to the strength of the synapse. When the weighted sum of inputs exceed the threshold value of the node, it activates and passes the signal through a transfer function and sends it to neighboring nodes. This process can be expressed as a mathematical model [3]:

$$y = f\left(\sum_{i=0}^{n} w_i x_i - T\right)$$

where y is the output of the node, f is the transfer function, w_i is the weight of input x_i, and T is the threshold value. The transfer function has many forms. A nonlinear transfer function is more useful than linear ones [4], since only a few problems are linearly separable. The simplest one is the step function (see Fig. 2.3):

$$y = \begin{cases} 0 \ldots if \sum_{i=0}^{n} w_i x_i > T \\ 1 \ldots if \sum_{i=0}^{n} w_i x_i < T \end{cases}$$

The sigmoid function also is often used as the activation function, since the function and its derivative are continuous (see Fig. 2.3):

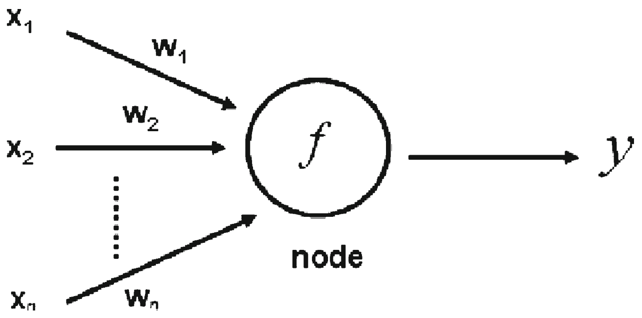


Fig. 2.2 A basic model of a single node: x_i = input, w_i = weight, *f* = transfer function, *y* = output

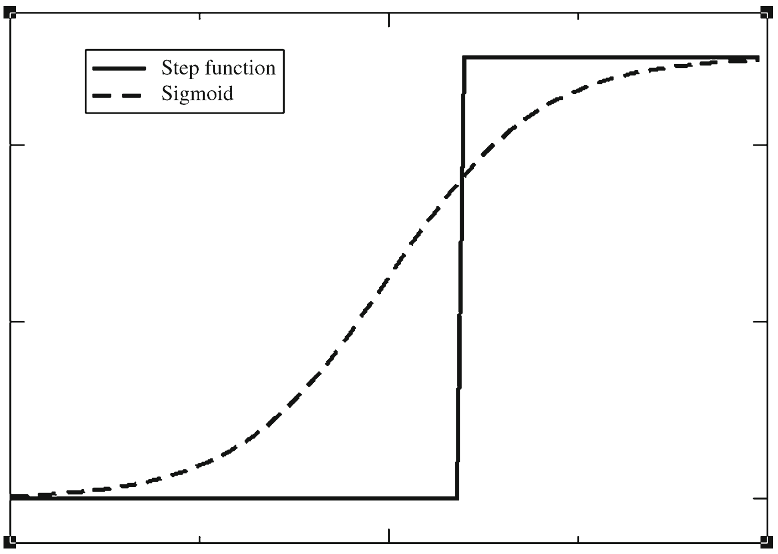


Fig. 2.3 Transfer function

$$y = \frac{1}{1 + \exp(-\beta x)}$$

3.2. Network Topology

Figure 2.4(a) shows the general architecture of an ANN. The nodes are organized into linear arrays, called *layers*. Usually, there are input layers, output layers, and hidden layers. There can be none to several hidden layers. Designing the network topology involves the determining the number of nodes at each layer, the number of layers in the network, and the path of the connections among the nodes. Usually, those factors are initially set by intuition and optimized through multiple cycles of experiments. Also some rational methods can be used to design a neural network. For example, the genetic neural network (GNN) uses a generic algorithm to select the input features for the neural network solving QSAR problems [5]. Fig. 2.4

There are two types of connections between nodes. One is a one-way connection with no loop back. The other is a loop-back connection in which the output of the nodes can be the input to previous or same level nodes. Based on the aforementioned type of connections, neural networks can be classified into two types: feedforward network and feedback network, as shown in Fig.2.4. Because the signal travels one way only, the feedforward network is static; that is, one input is associated with one particular output. The feedback network is dynamic. For one input, the state of the feedback network changes for many cycles, until it reaches an equilibrium point, so one input produces a series of outputs. Perceptron is a widely used feedforward network. Some well-known feedback networks include the Hopfield net and the Kohonen self-organizing maps. We briefly describe these networks later in the chapter. They are covered in detail in other chapters.

3.3. Learning

The ANN uses a learning process to train the network. During the training, weights are adjusted to desired values. The learning can be classified into two major categories: supervised learning and unsupervised learning. In supervised learning, a training set, that is, examples of inputs and corresponding target outputs, is provided. The weights are adjusted to minimize the error between the network output and the correct output. Special consideration is needed to construct the training set. The ideal training set must be representative of the underlying model. An unrepresentative training set cannot produce a very reliable and general model. For networks using supervised learning, the network must be trained first. When the network produces the desired outputs for a series of inputs, the weights are fixed and the network can be put in operation. In contrast, unsupervised learning does not use target output values from a training set. The network tries to discover the underlying pattern or trend in the input data alone. Different types of networks require different learning processes.

Many different learning schemes have been invented for ANNs to achieve different learning goals. The most frequently used learning approaches are error correction methods and nearest neighbor methods.

Error correction methods normally have a back propagation mechanism. Let y_k,n be the output of the kth output node at step n and y_k^* be the target output for the kth node. An error function can be defined as the difference betwen the node output and target output:

$$e_k = y_{k,n} - y_k^*$$

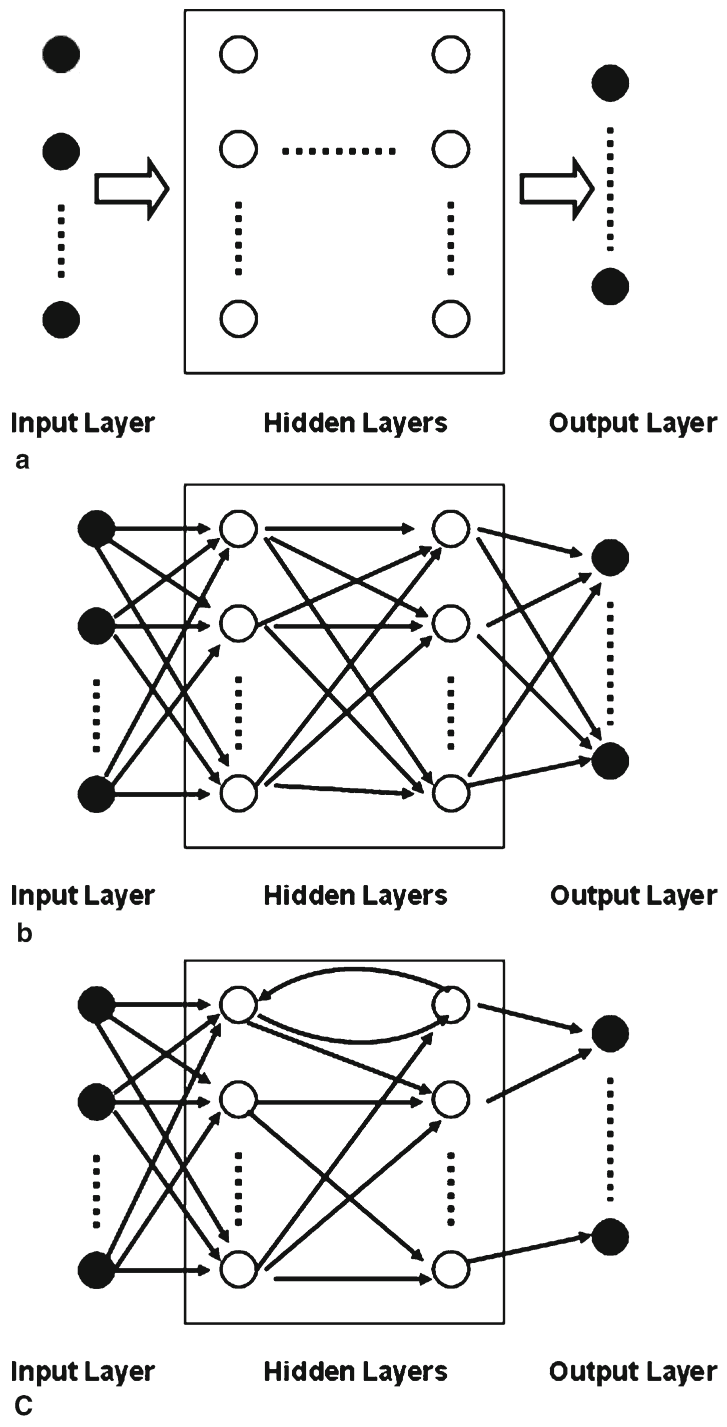


Fig. 2.4 **(a)** A general topology of ANN. **(b)** Feedforward ANN (perceptron). **(c)** Feedback ANN

Let λ be a positive constant that modulates the rate of weight adjustment. The new weight for input x_j is calculated as $w_{kj,n+1} = w_{kj,n} - \lambda e_k \cdot X_j$. The weight vector is updated every step until the system converges. The rate of learning, λ, has major impact on the system converging rate.

Another approach of ANN learning is based on the spatial similarity. This type of learning approach is more common in classification problems. The entire training set data are explicitly stored inside the network memory. For any test sample x_t, its nearest neighbor must have a distance that satisfies $\min_{i=1}^{N} [D(X_i - X_t)]$, where D is a distance measurement function and N is the training sample size. The test sample will assume the properties identical or similar to its closest neighbor based on a predefined function. The nearest neighbor learning approach is conceptually simple and appealing. However, as the dimension of the problem increases, the error rate can become unacceptable. This phenomenon is commonly called the *curse of dimension*.

4. Several Examples of ANNs

In this section, we briefly review several popular ANNs: perceptron, Hopfield net, and Kohonen maps. The readers can find more detailed descriptions of these ANNs in the corresponding chapters of this book.

4.1. Perceptron

The perceptron is a feedforward network and the earliest type of the neural network, developed by Rosenblatt [6]. A single hidden-layer perceptron has its limitations: It can solve only linearly separable problems. The classic example is the XOR problem, which cannot be simulated with the single-layer perceptron [4]. The multiple-layer perceptron (MLP), as shown in Fig. 2.4(b), is the most used neural network. It can be used to approximate any continuous functions. A back-propagation algorithm [7] is usually used in the training of MLP. In the algorithm, the input first is propagated through the network and the output calculated. Then the error between the calculated output and the correct output, called the *cost function*, is propagated backward from the output to the input to adjust the weights. Mathematically the algorithm minimizes the cost function with a gradient descent method, so that it can be applied only to networks with differentiable transfer functions and it is often difficult to train.

4.2. Hopfield Net

The Hopfield net [8] is a feedback neural network with only a single layer. A three-node Hopfield net is shown in Fig. 2.5. In the Hopfield net, every node is connected with all other nodes (fully connected), but not with itself. Given an input, the state

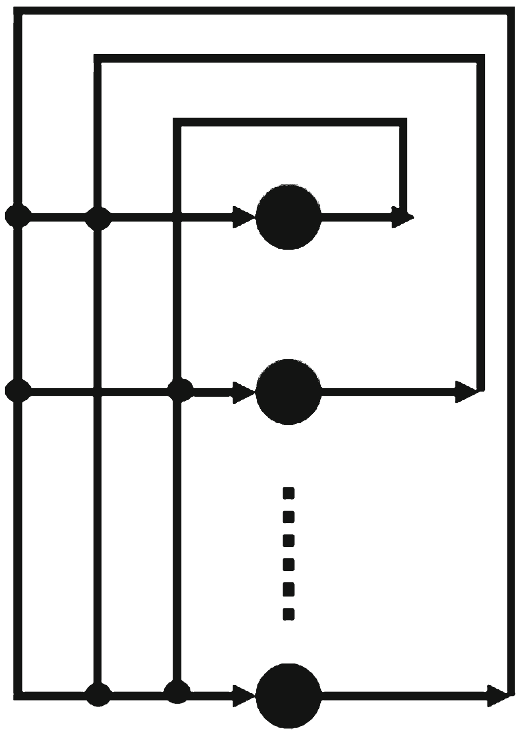


Fig. 2.5 Hopfield network

of the Hopfield net is updated until it converges to an equilibrium state, called the *attractor state*. Since any input eventually leads the network to one of the attractor states, the Hopfield net can be used as an associative memory to retrieve partially corrupted patterns. The classic traveling salesman problem can be solved with the Hopfield net.

4.3. Kohonen Maps

The Kohonen map [9] is another feedback network. It has a single hidden layer. The nodes in the hidden layer are organized typically as two-dimensional rectangular grids, as shown in Fig. 2.6. Every node in the hidden layer is connected to all the input nodes. The Kohonen map uses unsupervised learning with the winner-takes-all learning rule. Only the node with the highest response and its neighboring nodes get their weights updated, so that they are more likely to response to similar input patterns. Kohonen maps can be used for projection of high-dimensional data into two-dimensional space, clustering of data, and the like.

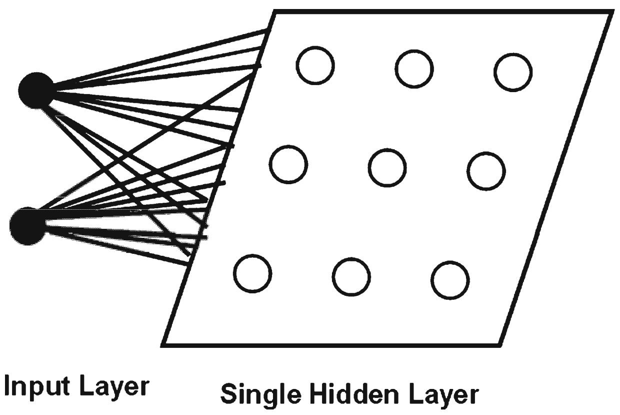


Fig. 2.6 Kohonen maps

5. Notes

1. An artificial neural network is an algorithm that simulates the fast learning performance of large number of biological neuron cells. The node is the most basic function using, in the ANN, simulating the behavior of a neuron cell. The synaptic weights, bias, and activation function together to determine the behavior of the node.
2. The network topology is the linkage among nodes. These connections define the way nodes interact with one another. the ANN topology can be divided into two major categories based on the presence or absence of a feedback connection.
3. Learning can be performed in an ANN system either with or without a training set. This is called *supervised learning* and *unsupervised learning*, respectively. For supervised training, two very popular learning schemes are the error-minimization scheme and the nearest neighbor scheme.

References

1. Feldman J, Fanty MA, Goddard NH (1998) Computing with structured neural networks. Computer 21:91–103.
2. Hebb DO (1993) The organization of behavior. John Wiley & Sons, New York.
3. McCulloch WS, Pitts W (1943) A logical calculus of ideas immanent in nervous activity, Bull Mathematical Biophysics 5:115–133.
4. Minsky ML, Papert SA (1969) Perceptrons. MIT Press, Cambridge, MA.
5. Chiu T-L, So S-S (2003) QSAR Comb Sci 22:519–526.
6. Rosenblatt, F (1962) Principles of neurodynamics: perceptrons and the theory of brain mechanisms. Spartan Books, New York.
7. Rumelhart DE, McClelland JL (1986) Parallel distributed processing: exploration in the microstructure of cognition. MIT Press, Cambridge, MA.
8. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Nat Acad Sci 79:2554–2558.
9. Kohonen T (1989) Self Organization and associative memory, 3 edn. Springer-Verlag, New York.

Chapter 3
Bayesian Regularization of Neural Networks

Frank Burden and Dave Winkler

Abstract Bayesian regularized artificial neural networks (BRANNs) are more robust than standard back-propagation nets and can reduce or eliminate the need for lengthy cross-validation. Bayesian regularization is a mathematical process that converts a nonlinear regression into a "well-posed" statistical problem in the manner of a ridge regression. The advantage of BRANNs is that the models are robust and the validation process, which scales as $O(N^2)$ in normal regression methods, such as back propagation, is unnecessary. These networks provide solutions to a number of problems that arise in QSAR modeling, such as choice of model, robustness of model, choice of validation set, size of validation effort, and optimization of network architecture. They are difficult to overtrain, since evidence procedures provide an objective Bayesian criterion for stopping training. They are also difficult to overfit, because the BRANN calculates and trains on a number of effective network parameters or weights, effectively turning off those that are not relevant. This effective number is usually considerably smaller than the number of weights in a standard fully connected back-propagation neural net. Automatic relevance determination (ARD) of the input variables can be used with BRANNs, and this allows the network to "estimate" the importance of each input. The ARD method ensures that irrelevant or highly correlated indices used in the modeling are neglected as well as showing which are the most important variables for modeling the activity data.

This chapter outlines the equations that define the BRANN method plus a flowchart for producing a BRANN-QSAR model. Some results of the use of BRANNs on a number of data sets are illustrated and compared with other linear and nonlinear models.

Keywords QSAR, artificial neural network, Bayesian regularizsation, early stopping algorithm, automatic relevance determination (ARD), overtraining

1. Introduction

The power of artificial neural networks (ANNs) as pattern classifiers, feature selectors, or paradigms for modeling complex data has been used in many fields such as character recognition, image compression, stock market prediction, medicine, electronic noses, security, and loan applications, as well as for modeling bioactivity. Many of these applications use very large networks with hundreds or thousands of neurodes. However, in the case of QSAR studies [1, 2], often the quality of the molecular descriptors has received the most attention, leaving the actual modeling method in a subordinate position. Even though most QSAR problems are unlikely, from both an a priori and an observational viewpoint, to be linear, many models in the literature utilize linear regression methods for lack of a simple nonlinear alternative.

Artificial neural networks have been used in a large number of QSAR studies, because they have a number of advantages over other regression techniques. Principal among these are that neural networks are universal approximators [3], capable of modeling any continuous, nonlinear function, given suitable training data. They are not without their problems, as they can overfit data, be overtrained and lose their ability to predict well, validation of the models can be problematic, and optimization of the network architecture is sometimes time consuming. They also are often viewed with suspicion because they are seen as slightly mysterious, unreliable, and difficult to interpret (black boxes). Some argue [4] that a nonlinear (quadratic) PLS is a safer and more reliable way of assessing nonlinear effects and cross-terms. However, by modifying the standard back-propagation neural network that has most widely been used for QSAR by inclusion of a *regularization* step incorporating Bayesian statistics, the benefits of neural networks can be retained and almost all the disadvantages removed. A Bayesian regularized artificial neural network [3, 5] (BRANN), coupled with molecular descriptors that relate easily to chemical features, is an excellent approximation to this ideal neural network.

The advantage of BRANN is that the models are robust and the validation process [6], which scales as $O(N^2)$ in normal regression methods, is unnecessary. These networks automatically solve a number of important problems that arise in QSAR modeling, such as choice of model, robustness of model, choice of validation set, size of validation effort, and optimization of network architecture.

Bayesian regularized neural networks have additional advantages:

- They are difficult to overtrain, as an evidence procedure provides an objective criterion for stopping training and removes the need for a separate validation set to detect the onset of overtraining.
- They are difficult to overfit, because they calculate and train on the effective number of parameters (essentially the number of nontrivial weights in the trained neural network). This is considerably smaller than the number of weights in a standard fully connected back-propagation neural net. These more parsimonious networks are much less likely to be overfitted. Bayesian neural nets essentially

incorporate Occam's razor, automatically and optimally penalizing excessively complex models. As the architecture is made more complex (e.g., by increasing the number of hidden-layer neurodes), the number of effective parameters converges to a constant. This provides an optimum balance between *bias* (where the model is too simple to explain the underlying SAR) and *variance* (where the model is excessively complex and fits the noise).

- Bayesian neural nets are inherently insensitive to the architecture of the network, as long as a minimal architecture has been provided.
- It has been shown mathematically that they do not strictly need a test set, as they produce the best possible model most consistent with the data. This has the advantage that a single, optimal model is provided, all available data may be used in the model (an advantage where data are scarce and expensive to acquire), and the validation effort (which may be demanding for large data sets) is removed.

The qualifiers *Bayesian* and *regularized* need to be carefully defined, and to do this, it is necessary to define simple regression methods in Bayesian terms. Because the terminology and symbols vary from one method to another and this leads to confusion, the symbols used here are consistent even if a little unfamiliar in some circumstances. As an example, the term *coefficients* is used in linear regression whereas the term *weights* is used in the literature of neural networks. Choices, therefore, have to be made; and in this case, the term *weights* has been chosen, since the chapter is about neural networks. For readers with an interest in Bayesian methods, a comprehensive overview applied to pattern recognition and regression is given by Bishop [7] and Nabney [8].

2. Regression and Regularization

Let the independent data be denoted as a matrix **X** with N_D rows and N_V columns and the dependent data as **y** with N_D rows and one column. This can easily be generalized to accommodate more columns, although this is not usually the case in QSAR studies. Linear regression is often written as

$$\hat{\mathbf{y}} = \mathbf{a} + \mathbf{X}\mathbf{b} \tag{1}$$

where $\hat{y}$ is an estimate of **y** derived from the model. The sum-squared-error in the data is given by

$$E_D = \sum_{i=1}^{N_D} (y_i - \hat{y}_i)^2 \tag{2}$$

and the estimate of the weights (coefficients), $\hat{\mathbf{b}}$, is found by minimizing the sum-squared-error E_D. This can be accomplished by the transform

$$\hat{\mathbf{b}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \text{ (T denotes a matrix transpose)} \tag{3}$$

where a column of ones has been entered into **X** to account for **a**.

A more generalized form of linear regression that includes powers of **x** can be written as

$$\hat{y} = \mathrm{Hw} \tag{4}$$

where **H** is known as the design matrix and may include other powers, or functions of x. The elements of **H** are $h_{ij}(x)$ with $i = 1$ to N_D and $j = 1$ to N_P, with N_P being the number of parameters (columns of **H**) used in forming **H**. Following convention, **w** has replaced **b** and is a vector of weights.

The least-squares solution for **w** in

$$\hat{y} = f(\boldsymbol{X}) = \sum_{i=1}^{N_P} w_i h_i(\boldsymbol{X}) \tag{5}$$

is obtained by minimizing

$$E_D = \sum_{i=1}^{N_D} [\mathbf{y}_i - f(\mathbf{X}_i)]^2 \tag{6}$$

Unfortunately, the minimization of Eq. 6 can fall into one of many local minima, not in the global minimum.

To control the problem of the inflation of the weights that can signal an overly complex model, the simple device of adding a diagonal matrix, Λ, to **H** *regularizes* the solution. The term Λ is sometimes known as a weight penalty, and its diagonal elements may be identical in the simplest, most common case. Such a procedure is called *regularization* and is used in many modeling methods such as ridge regression.

When Λ has constant diagonal elements, λ, E_D in Eq. 6, is replaced by a cost function, S(**w**), which is subsequently minimized with respect to the weights:

$$\boldsymbol{S(w)} = \sum_{i=1}^{N_D} [\mathbf{y}_i - \mathbf{f}(\mathbf{X}_i)]^2 + \lambda \sum_{j=1}^{N_P} w_j{}^2, \ \textit{Where}\ 0 \le \lambda \le 1. \tag{7}$$

The optimal value of λ can be determined as follows. Writing $\mathbf{A} = \mathbf{H}^T\mathbf{H} + \Lambda$, the minimized solution is given by

$$\hat{\mathbf{w}} = \mathbf{A}^{-1}\mathbf{H}^T\,\mathbf{y}, \text{ equivalent to } \hat{\mathbf{b}} = (\mathbf{X}^T\,\mathbf{X})^{-1}\,\mathbf{X}^T\,\mathbf{y} \text{ in Eq. 3} \tag{8}$$

The optimal value of λ is found by iterating Eqs. 8 and 9:

$$\hat{\lambda}\frac{Y^T \mathrm{P}^2 Y\left(\mathrm{A}^{-1} - \hat{\lambda}\,\mathrm{A}^{-2}\right)}{\hat{\mathrm{W}}\mathrm{A}^{-1}\hat{\mathrm{W}}\,\mathrm{trace}\ (\mathrm{P})} \tag{9}$$

Here, **P** is the projection matrix of **y** onto the line or plane of the best fit by $\mathrm{y} = \mathrm{P}\hat{\mathrm{y}}$ and can be computed from

$$\mathbf{P} = I_p - \mathbf{HA}^{-1}\,\mathbf{H}^T = \mathbf{H}(\mathbf{H}^T\,\mathbf{H}+\Lambda)^{-1}H^T \tag{10}$$

The use of λ in the preceding procedure is said to *regularize* the regression, and λ is known as the *regularization* parameter.

The effective number of parameters γ is given by

$$\gamma = N_P - \text{trace}\ (\mathbf{P}) \tag{11}$$

and shows how many terms are sufficient to form the model.

3. Bayesian Regularized Neural Networks

3.1. Bayesian Inference

Bayes' theorem says that conditional probability can be used to make predictions in reverse. Bayes' theorem, sometimes called the *inverse probability law*, is an example of statistical inference and is very powerful. People often make bad intuitive guesses about probabilities, when they could do much better if they understood Bayes' theorem.

Bayes' theorem can be derived fairly simply. If two events are independent of each other, the probability of *A* and *B* occurring is the product of the individual probabilities, P(*A* and *B*) = P(*A*)P(*B*). For example, in tossing a coin, the probability that a head will show is 0.5, but if two coins are tossed the probability that both will be heads is 0.5 × 0.5 = 0.25. However, the conditional probability of the second coin being a head if the first coin has already been tossed and shows a head is again 0.5.

The conditional probability (probability of *B* given *A*) is written as P(*B*|*A*). It is defined as

$$\mathrm{P(B|A) = \ P(A\ and\ B)/\ P(A)} \tag{12}$$

This can be rearranged to P(*A* and *B*) = P(*A*)P(*B*|*A*).

Similarly,

$$\mathrm{P\ (A|B)= P(A\ and\ B\ )/\ P(B)} \tag{13}$$

Combining Eqs. 12 and 13 gives Bayes' theorem:

$$\mathrm{P(A|B)= P(B|A)\ P(A)/P(B)} \tag{14}$$

We can use Bayes' theorem to find the conditional probability of event *A* given the conditional probability of event *B* and the independent probabilities of events *A* and *B*.

Not everyone is comfortable with Bayes' theorem. Some find it difficult to accept that, instead of using probability to predict the future, probability is used to make inferences about the past.

Event	Description	Probability
A	Someone has kidney cancer	0.000002
B	Someone has microscopic MH	0.1
$B \mid A$	Conditional probability of having MH given kidney cancer	1.0

Here, is another example of Bayes' theorem. Suppose that you are diagnosed with microscopic hematuria (MH). This symptom occurs in 10% of all people and 100% of people with kidney cancer. You would like to know the probability that you have kidney cancer, which occurs in 0.0002% of all people.

Using Bayes' theorem, we derive the probability that you have cancer as $P(A|B) = P(B|A)P(A)/P(B) = 1.0 \times 0.000002/0.1 = 0.00002$ or 0.002%.

That is, you still have a very low probability of kidney cancer. The reason is that the symptom of MH is relatively common in the healthy population. If the incidence of MH in the general population were much lower, say one hundredth of 1%, then MH would be a much more powerful indicator of kidney cancer. The probability would then be $P(A|B) = P(B|A)P(A)/P(B) = 1.0 \times 0.000002/0.0001 = 0.02$ or 2%.

Bayes' theorem can be used to optimally control regularization and make ANNs more robust, parsimonious, and interpretable.

3.2. Back Propagation

Most ANNs used in QSAR work have one hidden layer (sufficient for problems of any complexity, though not necessarily the most efficient) and one dependent variable. The cost function to be minimized, as before, is

$$\boldsymbol{S}(\boldsymbol{W}) = \sum_{i=1}^{N_D} [\mathbf{y}_i - f(\mathbf{X}_i)]^2 \tag{15}$$

Minimization of Eq. 15 is achieved by an efficient algorithm known as *back propagation of errors*. The efficiency comes from the use of a sigmoid transfer function (or sometimes hyperbolic tan, which has similar properties):

$$\text{sigmoid}(x) = f(x) = 1/[1+\exp(-x)] \textit{ that has a derivative } f'(x) = f(x)[1+f(x)] \tag{16}$$

which is cheap to compute.

Since the minimization of Eq. 15 is an iterative procedure, it is necessary to provide a criterion to stop the minimization. This is usually accomplished by using a validation set of samples not used in forming the model. The iterations are stopped when the squared error in the prediction of the validation set is a minimum. Further iterations not only increase the squared error but lead to a less predictive model. This phenomenon, known as *overtraining*, is illustrated in Figure 3.1.

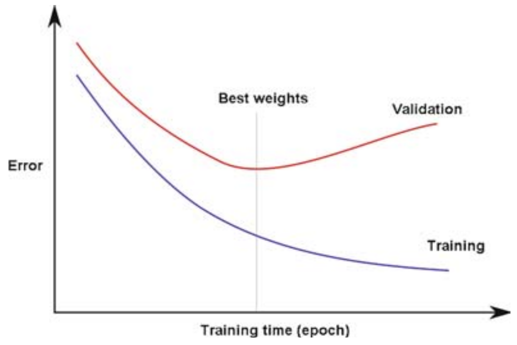


Fig. 3.1 Plot of error vs. training cycles. The training set error continues to decrease as training continues, whereas the validation set error reaches a minimum then increases as training continues

As validation sets are usually chosen arbitrarily, it is usual to repeat the whole procedure with alternative validation sets chosen from the same data set.

4. Bayesian Regularization of Neural Networks

Bayesian regularized ANNs (BRANNs) attempt to overcome these problems by incorporating Bayes' theorem into the regularization scheme.

Equation 7 can be slightly rewritten in terms of hyperparameters α and β instead of λ as

$$S(\boldsymbol{w}) = \boldsymbol{\beta}\sum_{i=1}^{N_D}[\mathbf{y}_i - f(\mathbf{X}_i)]^2 + \alpha\sum_{j=1}^{N_W} w_j^{\ 2} \tag{17}$$

where N_W is the number of weights. Given initial values of the hyperparameters α and β, the cost function, $S(\mathbf{w})$, is minimized with respect to the weights **w.** A reestimate of α and β is made by maximizing the evidence.

Assuming the weight and data probability distributions are Gaussian, the prior probability over the weights, **w,** may be written as

$$P(\mathrm{w} \mid \alpha, H) = \frac{1}{Z_W(\alpha)}\exp(-\alpha E_W) \tag{18}$$

with

$$E_W = \sum_{i=1}^{N_W} w_j^{\ 2} \text{ being the "error" of the weights.} \tag{19}$$

The probability of the errors may similarly be written as

$$P(D\,|\,\mathbf{w},\beta,H)=\frac{1}{Z_D(\beta)}\exp(-\beta E_D) \tag{20}$$

with

$$E_D=\sum_{i=1}^{N_D}[y_i-f(\mathrm{X}_i)]^2 \text{ being the error of the data.} \tag{21}$$

For a given model *H*, using Eq. 14, the Bayesian inference for the weights, w, can be written as

$$P(\mathbf{w}\,|\,D\,|\,\alpha,\beta,H)=\frac{P(D\,|\,\mathbf{w},\beta,H)P(\mathbf{w}\,|\,\alpha,H)}{P(D\,|\,\alpha,\beta,H)}=\frac{1}{Z_S}\exp[-S(\mathbf{w})] \tag{22}$$

S(**w**) can be written as a Taylor expansion about the most probable (MP) value of the weights, $\mathbf{w}_{MP,}$

$$S(\mathbf{w})\approx S(\mathbf{w}_{\mathrm{MP}})+\frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^T\mathbf{G}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}}) \tag{23}$$

If **G** is the Hessian matrix of the *total* error function, $S(\mathbf{w}_{MP})$,

$$\mathbf{G}=\nabla\nabla\ \mathrm{S}(\mathbf{w}_{\mathrm{MP}})=\beta\nabla\nabla\ E_D\ (\mathbf{w}_{\mathrm{MP}})+\alpha\ \mathrm{I}\}=\boldsymbol{\beta}\mathbf{D}+\alpha\ \mathrm{I} \tag{24}$$

and **D** is the Hessian of the data alone, the distribution of w can be approximated as a Gaussian,

$$P(\mathbf{w}\,|\,D,\alpha,\beta,H)\cong\frac{1}{Z_S^*}\exp[-S(\mathbf{w}_{\mathrm{MP}})]-\frac{1}{2}(\Delta\mathbf{w}^{\mathbf{T}}\mathbf{G}\Delta\mathbf{w}) \tag{25}$$

where $\Delta\mathbf{w}=\mathbf{w}-\mathbf{w}_{MP}$ and Z_S^* is a normalizing function.

Using Eq. 22 and dropping the model identifier *H*, the inference for the hyperparameters α and β is

$$P(\alpha,\beta\,|\,D)=\frac{P(D\,|\,\alpha,\beta)P(\alpha,\beta)}{P(D)} \tag{26}$$

Nabney [8] and Appendix 1 show that only the evidence, *P*(*D*|α, β), needs to be maximized, and the priors *P*(*D*) and *P*(α, β) can be ignored. Hence, the log of the evidence for α and β can be written as (Appendix 2)

$$\log P(D\,|\,\alpha,\beta)=\alpha E_W^{\mathrm{MP}}-\frac{1}{2}\mathrm{In}(|\,\mathbf{G}\,|)-\frac{N_W}{2}\log\alpha-\frac{N_D}{2}\log\beta-\frac{k}{2}\log 2\pi \tag{27}$$

which is maximized with respect to **α** and **β** (N_W is the number of weights and N_D is the number of data points).

There are two optimizations to be carried out: minimize Eq. 17 with respect to the weights and maximize Eq. 27 with respect to α and β until self-consistency is achieved. Within the double loop of Eqs. 17 and 27, the new values of α and β are reevaluated (see Appendix 2) using

$$\alpha = \gamma / 2E_W,$$
$$\beta = (N_D - \gamma) / 2E_D$$
$$\gamma = \sum_{i=1}^{N_W} \frac{\lambda_i}{\lambda_i + \alpha} = N_P - \alpha\,\mathrm{trace}\,(\mathbf{G}^{-1}) \tag{28}$$

The term γ is the effective number of parameters necessary for the model as in Eq. 11. The time-consuming step is the minimization of Eq. 17. This cannot be achieved using back propagation as for a simple ANN, and use of a procedure such as a conjugate-gradient method is required, and possibly the production and inversion of **G**.

Error bars can be ascribed to predictions made from BRANN models by use of the inverse of the data Hessian, **D** (see Appendix 2), which has already been evaluated in the evidence maximization loop. The derivative (**g**) of the predictions (*y)* with respect to the weights is calculated by a finite difference method that evaluates the network after adding δ*w* to each weight in turn. The variance of each prediction is then given by

$$\sigma^2 = \frac{1}{\beta} + \mathbf{g}^{\mathrm{t}}\mathbf{D}^{-1}\mathbf{g} \tag{29}$$

Programming of the BRANN algorithms is relatively straightforward and some Matlab© routines can be found in Nabney's book [8].

5. Practical Aspects of BRANNs

Since a validation set is unnecessary, the iterative procedure to self-consistency need be run only once to produce the "most generalizable" model. However, since the procedure makes use of a conjugate gradient descent or similar minimizer, it is possible to arrive at a local minimum rather than the global one. Experience shows that the precaution of running the whole procedure five times is sufficient to avoid any anomalous behavior. This can be contrasted with the possible hundreds or thousands of repeat calculations with unregularized ANNs.

As mentioned already, QSAR work rarely demands more than ten hidden neurodes, because most response surfaces are not highly nonlinear. Overfitting problems can arise if too many neurodes are used, since each additional neurode introduces $N_V + 2$ new weights (N_V being the number of independent variables or molecular descriptors). For instance, for a regression using 20 molecular descriptors and 5 hidden neurodes, there will be (20 + 1) × 5 + (5 + 1) = 111 weights, including input and hidden-layer bias neurodes. The addition of a sixth neurode adds a further 22 weights. Since the number of weights generally should not exceed half the number of samples (molecules), care needs to be taken not to use too many neurodes.

For ANNs, it is not always clear when the optimum number of neurodes has been employed because the addition of new neurodes may cause the training error to decrease, but overfitting may occur decreasing the predictivity of the net. In training an ANN, a large number of iterations is needed to reach the minimum in the validation set error, and many repeats need to be made with alternative validation sets. If the net architecture is changed in attempting to find the optimum number of hidden neurodes, this training and validation effort is magnified.

BRANNs avoid overfitting because the regularization pushes unnecessary weights towards zero, effectively eliminating them. The effective number of parameters in the model γ in Eq. 28 approaches a constant value once a sufficient number of hidden neurodes is reached, thereby preventing overfitting (see the example later). In the case of BRANNs, since there is no need for validation set during training, the necessary number of neurodes is easily found to the extent that if a few too many are used, the number of effective parameters, γ, hardly changes. This is effectively a pruning of the network in that many of the weights are set to near zero.

Overall, the BRANN method is far more robust, parsimonious, and efficient than a simple ANN and the network weights are more useful, in that they are not made up from a committee of networks, as is commonly used when multiple validation sets are used. Convergence is reached more reliably if α and β are reevaluated after a few iterations of the conjugate gradient minimizer. When a BRANN is used to predict the biological activity or toxicity of a new molecule, the read-back procedure is as simple and fast as that with a trained ANN, since the network structure is identical. Furthermore, the error bars associated with the prediction are built in to the Bayesian underpinning as described previously.

A few minor difficulties with implementing BRANNs come from both theory and practice. For some, the notion of Bayesian posterior prediction is inherently unpalatable. The method of estimation of the priors needs justification, and there are buried assumptions about the distribution of the weights and data (in the present case, Gaussian). This assumption can be justified theoretically, and it simplifies the algebra.

In practical terms, there are two difficulties. The first arises from the need to provide initial values of α and β. This can be achieved by setting α to a small value between 0 and 10 and β to a larger value of 100 or greater. In most cases, this will converge during the iterations to appropriate values that minimize Eq. 19. An alternative, and more logical, method of choosing the initial values of α and β is to compute a multiple linear regression (MLR) on the data and set and using Eq. 28 set

$$\beta = (N_D - N_V)/2E_D \tag{29}$$

$$\alpha = \gamma/2E_W \text{ with } \gamma = N_V \text{ and } \mathrm{E}_W = 1\text{; that is, } \alpha = \mathrm{N}_V/2 \tag{30}$$

6. Interpreting the Weights

Before discussing the interpretation of neural net QSAR models, we wish to clearly distinguish between the two purposes of QSAR modeling. QSAR can be used to generate primarily *interpretative* models, in which the model is dissected in terms of the contributions of particular molecular attributes to the activity. These data sets are often smaller, the descriptors have been chosen to be readily interpretable in terms of structure, and the compounds in the set are often purpose built to cover property space using experimental design principles. QSAR can also be used to build primarily *predictive* models, constructed using large, diverse data sets and computationally efficient (if somewhat opaque) descriptors. Such types of model are often used to screen large databases of molecules to find new leads, not to interpret the nuances of the model.

Because the two purposes for QSAR modeling are not always clearly understood, one of the perceived problems with adopting ANNs as a universal modeling method is that they are hard to interpret. Although ANN models may be statistically superior and more predictive than those from other methods it is difficult to "tease the model apart" and understand how molecular variables affect activity because of the nonlinearity of the model. We contend that an ANN model that used descriptors that can be readily related to molecular features, or physicochemical properties of molecules can still be interpreted by employing at least two approaches:

- *Sensitivity analysis*. The relevance of each of the ANN or BRANN inputs (indices) can be computed via a sensitivity analysis [9]. This involves running the network in feed-forward mode, varying each input by a small amount and determining how much the output (activity) changes. However, the sensitivity is dependent on what part of the complex response surface is chosen as the base point. There is ambiguity in choosing the base point, and three rational options are to choose those of the most active compound, the least active compound, or some average value. This choice is subjective (and somewhat unpalatable), even if straightforward.
- *Automatic relevance determination*. This makes use of the Bayesian underpinning to modify the number of hyperparameters used to regularize the neural network. It employs one hyperparameter α_i (i = 1 to N_V) for each independent variable (descriptor). The significance of each input variable is then computed as

$$\text{Relevance}_i = \log(1/\alpha_i) \tag{31}$$

This works well for problems with a small number of descriptors (~<10) but tends to break down for larger values, since the evidence maximization landscape becomes too complex for stable solutions to be found reliably [10].

Since α is proportional to $1/E_W$, (Eq. 28), and $E_W = \sum_{j=1}^{N_P} \mathbf{w}_j^2$, then α_i is inversely proportional to $\sum_{j=1}^{P} \mathbf{w}_{ij}^2$. This can be used as an approximate method to evaluate the relevance of each input variable. Allowance needs to be made if the input data is not of the same scale or has not been normalized.

7. Examples to Illustrate the Advantages of BRANN

The examples used here are mainly from the work of the authors. We used data sets that are difficult to model because easily modeled data sets fail to show good discrimination between different types of modeling methods (i.e., all methods can produce similarly good models). However, they do illustrate the power of ANNs and the robustness of BRANNs. The indices used in the examples are either physicochemical properties or easily computed molecular indices.

7.1. Advantages of a Nonlinear QSAR Method

Farnesyl protein transferase (FT) is a heterodimeric zinc-catalyzed enzyme involved in the modulation of runaway cell replication in cancerous tumors.

A set of 1,412 compounds with very diverse structures was compiled from available literature studies. These were split into two groups by *k*-means clustering in descriptor space: a training set (80%, or 1,129 molecules) and a test set (20%, or 283 molecules) [11].

The descriptors selected were the atomistic (*A*) [12], Burden (*B*) [13], and charge fingerprint (*C*) indices [14]. The *A* indices are simple counts of each type of atom in the molecule (which has been shown to correlate with lipophilicity and molecular refractivity), the *B* indices are widely known descriptors derived from the eigenvalues of modified adjacency matrices obtained from the molecular graph. The charge fingerprint indices (*C*) are derived by binning the Gasteiger-Marsili charges [15] for each atom type into specified ranges. The modeling techniques correlated the molecular descriptors against the available in vitro $-\log IC_{50}$ (pI_{50}) data for the human isoform of FT. The BRANN used three hidden-layer neurodes. The simple descriptors used in this study contain sufficient relevant information that even linear regression methods (MLR) produce good models. However, as Table 3.1 shows, the BRANN method is superior to the MLR method illustrating that there is a degree of nonlinearity in the response surface.

Table 3.1 MLR and BRANN QSAR models of the farnesyltransferase data set

	Training set		Test set	
Method	Normalized standard error of estimation, SEE	R^2	Normalized standard error of prediction, SEP	Q^2
MLR	0.14	0.74	0.15	0.70
BRANN	0.10	0.86	0.13	0.76

Source: M.J. Polley, D.A. Winkler, and F.R. Burden. (2004) Broad-based quantitative structure-activity relationship modeling of potency and selectivity of farnesyltransferase inhibitors using a bayesian regularized neural network. *J Med Chem* 47:6230–6238.

R^2 = Correlation between experimental and predicted data for the training set.

Q^2 = Correlation between experimental and predicted data for the test set.

7.2. Relative Independence of Model on Neural Net Architecture

This example uses of an extension of a toxicity data set reported and analyzed by us [16]. In this model, we use the CIMI indices [17] (another type of Burden eigenvalue index) together with charge fingerprint descriptors [14].

Table 3.2 and Fig. 3.2 shows the result of a BRANN calculation with an increasing number of hidden-layer neurodes. It can be seen that increasing the number of neurodes from three to nine makes essentially no difference to the statistics of the training or test set. The number of effective parameters tends to an asymptote as compared to the number of weights, which increase linearly with number of neurodes. Essentially, any number of hidden-layer neurodes above three produces almost identical models. As the models use the effective number of parameters rather than the numbers of weights, this ensures that the BRANN is resistant to overfitting.

7.3. Robustness and Reproducibility of BRANN Models

Blood-brain barrier (BBB) [18] permeability is an important ADME property. Since the surface area of the human BBB is very large, the BBB is considered the main region controlling the uptake of drugs into the brain and is the target for delivering drugs to the brain. The model was recalculated five times and the results are summarized in Table 3.3. The statistics of the five runs are essentially identical. In more complex response surface cases, the method can find two alternative solutions that are easily discriminated by means of the evidence. However, in most cases, even the solution with lower evidence is still a model with good predictivity.

Table 3.2 Architecture independence of BRANN models using a data set of toxicity to the fathead minnow

Number of hidden layer neurodes	SEE	R^2	SEP	Q^2	Number of weights, N_W	*Effective number of parameters,* γ
MLR	0.12	0.46	0.11	0.53	22	22
1	0.12	0.44	0.11	0.53	25	20
3	0.10	0.63	0.09	0.64	73	47
5	0.10	0.62	0.09	0.64	121	45
7	0.10	0.66	0.09	0.65	169	54
9	0.10	0.67	0.09	0.67	217	61

Source: D.V. Eldred, C.L., Weikel, P.C. Jurs, and K.L. Kaiser, L. (1999) Prediction of fathead minnow acute toxicity of organic compounds from molecular structure. *Chem Res Toxicol* 1999, 12:670–678.

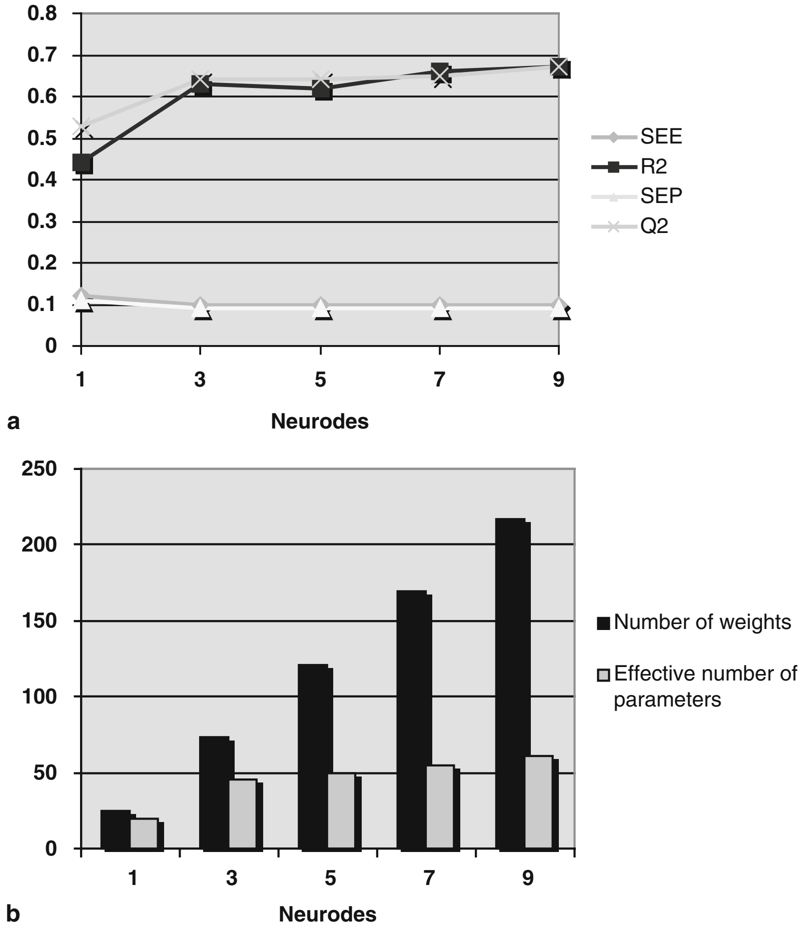


Fig. 3.2 **a** Training and test set statistics as a function of the number of hidden-layer neurodes (model complexity). **b** The number of ANN weights and number of BRANN effective parameters as a function of the number of hidden-layer neurodes

Table 3.3 Robustness and reproducibility of BRANNS models for the blood brain barrier data set

	Training set		Test set		
Run number	SEE %	r^2	SEP %	q^2	Effective number of parameters
1	14.2	0.62	15.5	0.63	21
2	14.3	0.62	15.6	0.63	23
3	14.4	0.62	15.6	0.63	24
4	14.4	0.62	15.6	0.63	25
5	14.5	0.62	15.7	0.63	25

Source: Reference [*18*].

7.4. Assessing the Relevance of the Input Variables

The blood-brain barrier data set is also useful to illustrate how automatic relevance determination (ARD) can be used to determine the importance of the descriptors used to form the model. This model used descriptors such as log *P*, number of rotatable bonds, polar surface area, numbers of hydrogen bond donors and acceptors; and ARD was used to determine which of these were the most important contributors to the model. The relative importance of the molecular descriptors is summarized illustrated in Fig. 3.3. The ARD relevances were consistent with known mechanisms for BBB penetration and with conclusions from other literature models of BBB partitioning.

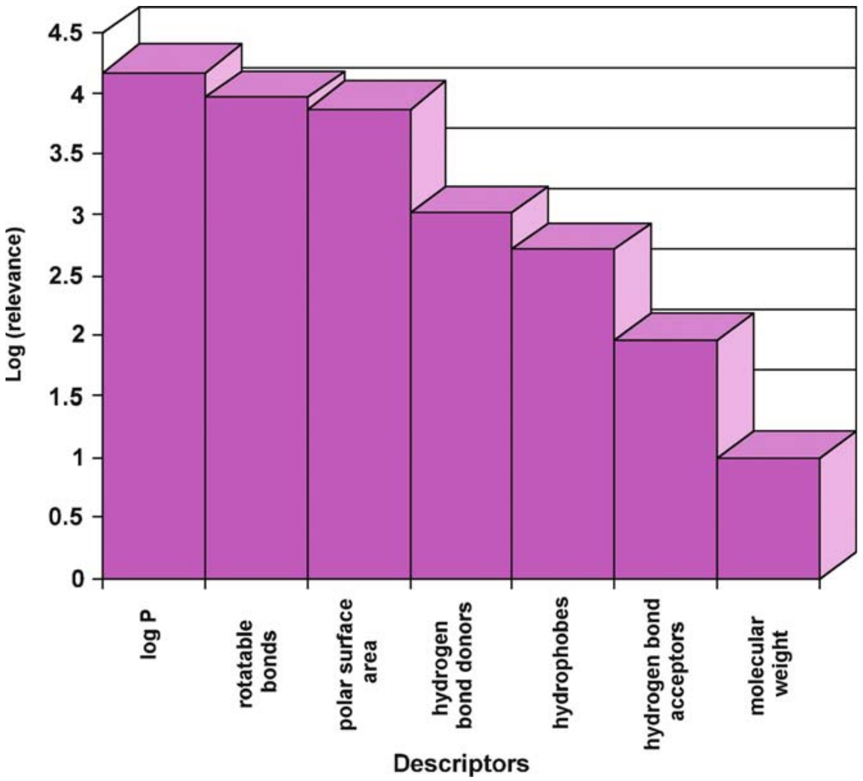


Fig. 3.3 Automatic relevance determination bar plot showing significance of property-based descriptors in the blood-brain barrier model [*18*]

8. BRANN Software

Calculations were performed using software written in-house using the open source Python [19, 20, 21 language. The BRANN algorithms were taken, in part, from Nabney [8], and most of the basic mathematical routines were included from the Python Numpy library.

9. Conclusion

Bayesian regularized neural networks offer a simple and usable form of artificial neural network. They are robust, in that they are difficult to overtrain or overfit; and they find a good, generalizable model with very few repeat calculations. There are a number of examples of their application to QSAR modeling in the literature [1, 2, 10, 22, 23, 24, 25] that illustrate these advantages and the need for using a robust nonlinear method.

10. Notes

1. Bayesian regularized neural networks are basically back-propagation networks with an additional ridge parameter added to the objective function (Eq. 17) plus a Bayesian-based method for terminating the training. BRANNs are trained until the log of the evidence is a maximum (Eq. 27)
2. Since the initial weights are taken from a uniform random distribution, it is usual to run the BRANN a few times to ensure that the starting weights are not pathological. The result with the largest log of the evidence is taken to be the most general model.
3. The BRANN is not sensitive to overtraining, and it is usually found that the number of effective parameters (Eq. 28) barely increases with the addition of an extra node above the minimum necessary number as determined experimentally.
4. As with back-propagation ANNs, the time taken for each BRANN run depends on the computer speed, the number of hidden nodes, the number of variables, and the number of data samples. As an example, a data set of 14,000 samples and 100 independent variables took less than five minutes on 1800-Ghz Pentium computer using two hidden nodes.
5. If used with data sets containing a very large number of independent variables, such as those obtained from microarrays, the size of the problem can be profitably reduced using principal component analysis prior to the BRANN.
6. The time-consuming steps in the training of a BRANN are in the gradient descent or similar routine, when minimizing the objective function. This is followed by an optimization of the hyperparameters α and β; and it has been

found that it is best to halt the conjugate gradient routine after very few iterations, optimize α and β, and return to the conjugate gradient routine. The dual cycle is terminated when the change in the log of the evidence is below a set threshold (<0.1% in our program).

7. Two appendices showing more details of the algebra are provided.
8. Three diverse examples of the use of BRANNs are used to illustrate the method.

Appendix 1

Equation 28 shows that the inference for the hyperparameters, α and β, is given by

$$P(\alpha,\beta \mid D) = \frac{P(D \mid \alpha,\beta)P(\alpha,\beta)}{P(D)} \tag{A1.1}$$

The denominator is a normalizer, in that it is a shorthand for the integral of the numerator over the conditional parameters, α and β.

$$P(D) = \iint P(D \mid \alpha,\beta)P(\alpha,\beta)d\alpha\, d\beta \tag{A1.2}$$

However, since the simplified view has been taken that we are dealing with the most probable evidence, we need only the peaks of this density, and we can ignore this term.

The hyperprior $P(\alpha, \beta)$ still needs to be chosen. Many choices can be made here, and a simple one is to consider it a uniform distribution. This means that it is an *improper prior*, since its integral is infinite.

Therefore, using these two assumptions, only $P(D|\alpha, \beta)$ needs to be maximized to achieve the maximization of $P(\alpha, \beta|D)$.

Appendix 2

Equation 27 presented the log of the evidence for α and β as

$$\log P(D \mid \alpha,\beta) = \alpha E_W^{\mathrm{MP}} - \frac{1}{2}\mathrm{In}(|\mathrm{C}|) - \frac{N_W}{2}\log\alpha - \frac{N_D}{2}\log\beta - \frac{k}{2}\log 2\pi \tag{A2.1}$$

which needs to be maximized with respect to α and β.

Tackling the maximization with respect to α first we need the derivative of Eq. A2.1 with respect to α. The difficult term is ln (|**C**|).

Let $\lambda_1, \lambda_2, \ldots, \lambda_N$ be the eigenvalues of the data Hessian, **D** (see Eq. 26), then the **C** will have eigenvalues $\lambda_i + \alpha$ and

$$\frac{d}{d\alpha}\mathrm{In}\,|\mathrm{C}| = \frac{d}{d\alpha}\left(\prod_{i=1}^{N_W}\lambda_i + \alpha\right) = \frac{d}{d\alpha}\left[\sum_{i=1}^{N_W}\mathrm{In}(\lambda_i + \alpha)\right] \tag{A2.2}$$
$$= \left(\sum_{i=1}^{N_W} 1/(\lambda_i + \alpha)\right) = \mathbf{trace}(\mathbf{G}^{-1})$$

A further discussion of this approximation can be found in Nabney [8] and Mackay [26], and the derivative of Eq. A2.1 with respect to α is

(continued)

Appendix 2 (continued)

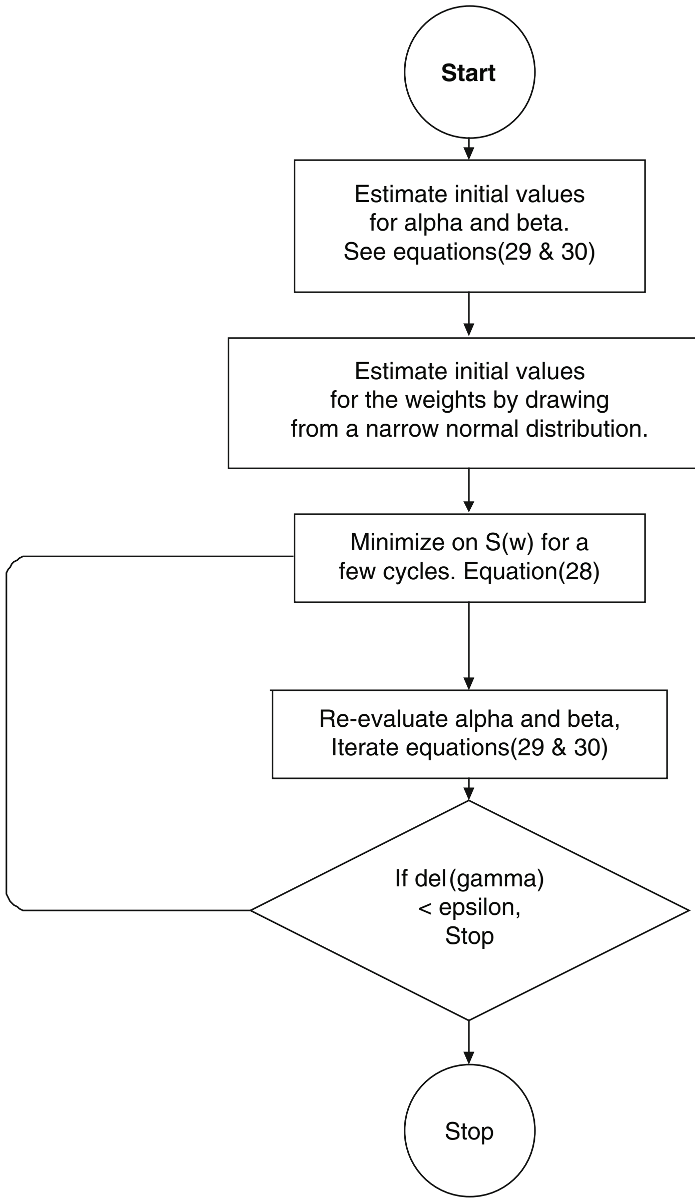

Start
Estimate initial values for alpha and beta. See equations(29 & 30)
Estimate initial values for the weights by drawing from a narrow normal distribution.
Minimize on S(w) for a few cycles. Equation(28)
Re-evaluate alpha and beta, Iterate equations(29 & 30)
If del (gamma) < epsilon, Stop
Stop

Fig. 3.4 Flowchart for a BRANN calculation

$$-E_W^{\mathrm{MP}} - \frac{1}{2}\sum_{i=1}^{N_W} \frac{1}{\lambda_i + \alpha} + \frac{N_W}{2\alpha} \tag{A2.3}$$

If this derivative is set equal to zero, we get

(continued)

Appendix 2 (continued)

$$2\alpha E_W^{\mathrm{MP}} = N_W - \sum_{i=1}^{N_W} \frac{\alpha}{\lambda_i + \alpha} \tag{A2.4}$$

The right-hand side is equal to a value γ, defined as

$$\gamma = \sum_{i=1}^{N_W} \frac{\lambda_i}{\lambda_i + \alpha} \tag{A2.5}$$

which can be viewed as a measure of the number of *well-determined* parameters, Bishop [7, Section 10.4].

Equations 27 and A2.2 also need to be optimized with respect to β. Let μ_I denote the eigenvalues of$\nabla\nabla E_D$; and since **D** = $\beta\nabla\nabla E_D$, it follows that

$$\frac{d\lambda_i}{d\beta} = \mu_i = \frac{\lambda_i}{\beta} \tag{A2.6}$$

hence,

$$\frac{d}{d\beta}\ln|A| = \frac{d}{d\beta}\sum_{i=1}^{N_W}\ln(\lambda_i + \alpha) = \frac{1}{\beta}\sum_{i=1}^{N_W}\frac{\lambda_i}{\lambda_i + \alpha} \tag{A2.7}$$

From this, following the previous logic,

$$2\beta E_D^{\mathrm{MP}} = N_D - \sum_{i=1}^{N_W}\frac{\lambda_i}{\lambda_i + \alpha} = N_D - \gamma \tag{A2.8}$$

Using these equations, the values of α, β, and γ are computed, following the minimization of *S*(**w**), by using Eqs. A2.4, A2.5, and A2.8, as

$$\alpha = \gamma / 2E_W, \beta = (N_D - \gamma)/2E_D, \text{and} \gamma = \sum_{i=1}^{N_W}\frac{\lambda_i}{\lambda_i + \alpha} \qquad \text{A(2.9)}$$

in an iterative loop that converges in γ.

The time-consuming step in this cycle is the production of the eigenvalues, λ_i. However, for most QSAR problems, this loop is much faster than the minimization of *S*(**w**), which uses a conjugate-gradient or some such minimizer.

Figure 3.4 shows the flow chart of a typical BRANN.

References

1. Burden FR, Winkler DA (1999) Robust QSAR models using Bayesian regularized neural networks. J Med Chem 42:3183–3187.
2. Winkler DA, Burden FR (2000) Robust QSAR models from novel descriptors and Bayesian regularized neural networks. Mol Simul 24:243–258.
3. MacKay DJC (1992) A practical Bayesian framework for backpropagation networks. Neural Computation 4:448–472.
4. Lucic B, Amic D, Trinajstic N (2000) Nonlinear multivariate regression outperforms several concisely designed neural networks on three QSPR data sets. J Chem Inf Comput Sci 40:403–413.
5. Neal RN (1996) Bayesian learning for neural networks. Springer-Verlag New York, Inc., Secaucus, NJ.

6. Hawkins DM, Basak SC, Mills D (2003) Assessing model fit by cross-validation. J Chem. Inf Comput Sci 43:579–58.
7. Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford.
8. Nabney IT (2002) Netlab: algorithms for pattern recognition. Springer-Verlag, London.
9. Baskin II, Ait AO, Halberstamc NM, PalyulinVA, Zefirov NS (2002) An approach to the interpretation of backpropagation neural network models in QSAR studies. SAR QSAR Environ Res 13:35–41.
10. Burden FR, Ford MG, Whitley DC, Winkler DA (2000) Use of automatic relevance determination in QSAR studies using Bayesian neural networks. J Chem Inf Comput Sci 40:1423–1430.
11. Polley MJ, Burden FR, Winkler, DA. (2005) Predictive human intestinal absorption QSAR models using Bayesian regularized neural networks. Aust J Chem 58:859–863.
12. Burden F R (1996) Using artificial neural networks to predict biological activity from simple molecular structure considerations. Quant Struct-Act Relat 15:7–11.
13. Burden FR (1989) Molecular identification number for substructure searches. J Chem Inf Comput Sci 29:225–227.
14. Winkler DA, Burden FR (2004) Bayesian neural nets for modeling in drug discovery. Biosilico 2:104–111.
15. Gasteiger J, Marsili, M (1980) Iterative partial equalization of orbital electronegativity—a rapid access to atomic charges. Tetrahedron . 36:3219–3288.
16. Burden FR, Winkler DA (2000) A QSAR model for the acute toxicity of substituted benzenes to tetrahymena pyriformis using Bayesian Regularized neural networks. Chem Res Toxicol 13:436--440.
17. Burden FR (1997) A chemically intuitive molecular index based on the eigenvalues of a modified adjacency matrix. Quant Struct-Act Relat 16:309–314.
18. Winkler DA, Burden FR (2004) Modelling blood brain barrier partitioning using Bayesian neural nets, J Mol Graph Model 22:499–508.
19. van Rossum G (1995) Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May1995.
20. van Rossum G, Drake FL Jr (eds) (2003) Python/C API reference manual. PythonLabs, release 2.2.330 May.
21. van Rossum G, Drake FL Jr (eds) (2003) Python library reference. PythonLabs, release 2.2.330 May.
22. Winkler DA, Burden FR (2000) Robust QSAR models from novel descriptors and Bayesian regularized neural networks. Mol Simul 24:243–258.
23. Winkler DA, Burden FR (2002) Application of neural networks to large dataset QSAR, virtual screening and library design. in: Bellavance-English,L (ed) Combinatorial chemistry methods and protocols., Humana Press, Totowa, NJ.
24. Bruneau P (2001) Search for predictive generic model of aqueous solubility using Bayesian neural nets. J Chem Inf Comput Sci 41:1605–1616.
25. Klocker J, Wailzer B, Buchbauer G, Wolschann P (2002)Bayesian neural networks for aroma classification. J Chem Inf Comput Sci 42:1443–1449.
26. MacKay DJC (1992) Bayesian interpolation. Neur Comput 4:415–447.

Chapter 4
Kohonen and Counterpropagation Neural Networks Applied for Mapping and Interpretation of IR Spectra

Marjana Novič

Abstract The principles of learning strategy of Kohonen and counterpropagation neural networks are introduced. The advantages of unsupervised learning are discussed. The self-organizing maps produced in both methods are suitable for a wide range of applications. Here, we present an example of Kohonen and counterpropagation neural networks used for mapping, interpretation, and simulation of infrared (IR) spectra. The artificial neural network models were trained for prediction of structural fragments of an unknown compound from its infrared spectrum. The training set contained over 3,200 IR spectra of diverse compounds of known chemical structure. The structure-spectra relationship was encompassed by the counterpropagation neural network, which assigned structural fragments to individual compounds within certain probability limits, assessed from the predictions of test compounds. The counterpropagation neural network model for prediction of fragments of chemical structure is reversible, which means that, for a given structural domain, limited to the training data set in the study, it can be used to simulate the IR spectrum of a chemical defined with a set of structural fragments.

Keywords Counterpropagation neural network, infrared spectra, Kohonen neural network, mapping, predictive ability, reliability of predictions, spectra interpretation, spectra simulation, structural fragments, supervised learning, unsupervised learning.

1. Introduction

The unsupervised and supervised learning strategies of the Kohonen [1–2] and counterpropagation [3–5] neural networks and the application of them in infrared (IR) spectroscopy for spectral interpretation and simulation are the topics addressed in this chapter. We present an overview over the advantages and complementary roles of both methods. The common basis for both methods is a nonlinear mapping of a set of input variables in to the corresponding network output.

The self-organization of the inputs in the Kohonen neural network and in the input layer of the counterpropagation neural network is the reason why they are also called *self-organizing maps* (SOMs). It is of a special advantage that no requirement of expressing the relation between the input and output variables in a mathematical form is needed. The methods are flexible and thus applicable in many areas, even if noise is present in the input data or the input data matrix is jeopardized by errors, outliers, or missing data [6–7].

The principle of every predictive modeling is to relate independent input variables to a set of dependent output variables. Here, we present the model for infrared spectra interpretation, in which we search for the chemical structure of a compound of a known IR spectrum. The input variables are the intensities of the infrared spectrum of an investigated compound, while the output variables represent a set of binary vectors with components assigned to a unique structural fragment. For the input, equidistant spectral intensities are considered over the spectral region from 3,500 to 550 cm^{-1}. To reduce the dimension of the input vector, the Fourier or Hadamard transformation may be applied and corresponding coefficients may be taken as the input variables instead of raw spectral intensities. The output vectors represent chemical structures of compounds, encoded as sets of structural fragments present. For example, the first component of the output vector defines presence or absence (the value of the first component being one or zero, respectively) of an OH group. The consecutive components of the output vector introduce the difference between the alcohol and phenol OH groups.

One advantage of the counterpropagation neural network model is the reversibility of the input and output layers in the prediction phase. Once the model is trained with the pairs of spectra-structures as input-output data obtained, one may use it in a reverse manner. Any properly encoded chemical structure can be introduced to the network through the output layer; the corresponding spectral representation vector is thus pinpointed in the input layer, as the trained network retains the relationship between the inputs and outputs obtained from the training phase. Of course, one can expect that the quality of predicted (simulated) infrared spectra depends on the coverage of the structural domain by the training data.

2. Methods

2.1. Kohonen Neural Networks

Kohonen neural networks were initially developed with the aim to mimic human brain functioning, mainly the storage of information and memory. In human brains, similar information is stored in certain regions (neighboring neurons) of the cortex. This is related to the mapping of inputs in the Kohonen map. The unsupervised nature of learning strategy of the Kohonen neural networks is rationalized by the way young children learn to recognize objects. They do not have to know the words of objects, they just look at the images (pictures) and automatically relay, for example, houses in the same group of objects, no matter how many windows or chimneys they have.

2.1.1. Unsupervised Learning and self-Organizing Strategy in Kohonen Neural Networks

For the unsupervised learning strategy, one needs only the description of objects, that is, the independent variables for the input vectors. The properties are not given, so the map obtained shows only the relationship between the independent variables of the objects, regardless of the properties that may also be known but not presented in the objects representation vectors. Let us define the input vectors $\boldsymbol{X}_s\ (x_{s1}, x_{s2}, \ldots, x_{si}, \ldots, x_{sm})$ for a particular object s described by m independent variables. The input vectors are relayed to the Kohonen network. This means that they are compared with the neurons organized in a special way in the single layer Kohonen network. To enable the comparison between the input vectors and neurons, we have to define the neurons as m-dimensional vectors, $\boldsymbol{W}_j(w_{j1}, w_{j2}, \ldots, w_{ji}, \ldots, w_{jm})$. They are composed of weights being adapted during the training process.

2.1.2. Basic Architecture of Kohonen Neural Networks

Basically, we differ between one- and two-dimensional organization of neurons in the single layer of the Kohonen neural network. In one-dimensional organization, the neurons are gathered in a line, each neuron having only two neighbors. The two-dimensional organization of neurons, in which the neurons are arranged in a plane with well-defined topology (rectangular or hexagonal neighborhood), is much more interesting and applicable in many different areas. To define the two-dimensional layer of neurons, we split the index j of a consecutive neuron into j_x and j_y: $\boldsymbol{W}_{i_x j_y}$ $(W_{j_x j_y,}1, W_{j_x j_y,}2, \ldots, W_{j_x j_y,i}, \ldots, W_{j_x j_y,m})$.The number of neurons in the network (N_{net}) is equal to the product of numbers of neurons in both dimensions $N_{\text{net}} = N_x \times N_y$. For illustration of a Kohonen neural network architecture see Figure 4.1.

2.1.3. Training the Kohonen Neural Network

Before training starts all weights in the network are randomized in the interval [0, 1]. Afterward, the learning strategy named *winner takes all* is applied. At each object input, the winning neuron chosen according to a predefined criterion is stimulated. This competition among the neurons is referred as *competitive learning*. The selection of the winning neuron (also recognized as the central neuron) is based on the Euclidean distance between the neurons, that is, vectors of weights, $\boldsymbol{W}_j\ (w_{j1}, w_{j2}, \ldots,$

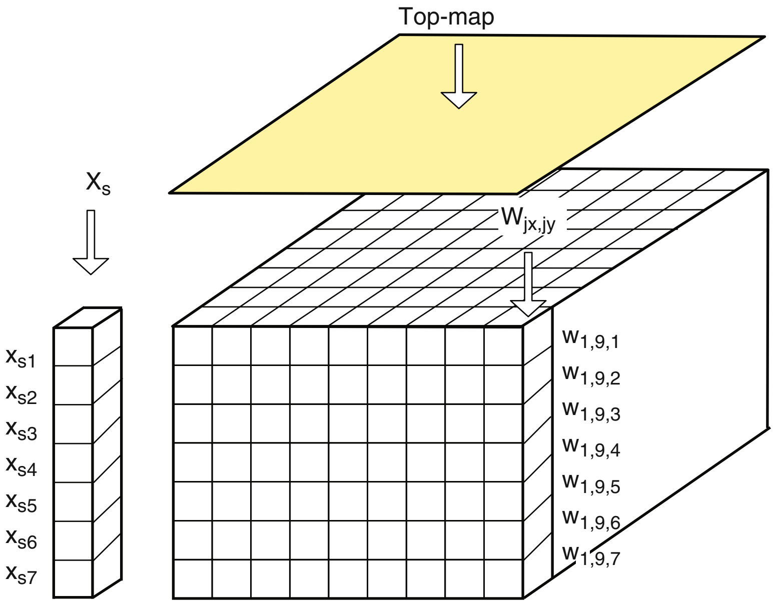


Fig. 4.1 Kohonen neural network architecture: $\boldsymbol{X}_s$ illustrates the sth input vector; $\boldsymbol{W}_{j_x j_y}$ stands for the neuron at the position $j_x j_y$. Top-map provides the storage of rectangular mesh $(N_x \times N_y)$ with information about each located object.

$w_{ji}, \ldots, w_{jm}$), and the objects, $\boldsymbol{X}_s$ $(x_{s1}, x_{s2}, \ldots, x_{si}, \ldots, x_{sm})$. Minimal distance over all N_{net} neurons defines the winner or central neuron $\boldsymbol{W}_c$:

$$d_{j,s}^{\text{Eucl}} = \sqrt{\sum_{i=1}^{m} (W_{j,i} - X_{s,i})^2}\,, \ \min\left\{d_{j,s}^{\text{Eucl}},\ j = 1 \ldots N_{\text{net}}\right\} \Rightarrow W_c \tag{1}$$

Having chosen the central neuron, it is stimulated with the correction of weights, so that it becomes even more similar to the object $\boldsymbol{X}_S$. The amount of correction is defined by a time-dependent learning rate parameter $\eta(t)$. The time is related to the number of iteration in the training, which is an iterative procedure. In each of the iterations, a new object enters the network. One training epoch is defined as the number of iterations in which all objects were entered once. The iterations are repeated with the same set of objects N_{epoch}-times until the minimal error in one epoch, defined as a sum of differences between the objects and winning neurons, is obtained. Besides the correction of the winning (central) neuron, the weights of the neighboring neurons are corrected. The following equation defines the corrections:

$$w_{ji}^{\text{new}} = w_{ji}^{\text{old}} + \eta(t) \times b(d_c - d_j) \times (x_{si} - w_{ji}^{\text{old}}) \tag{2}$$

Parameter η determines the rate of learning: It is maximal at the beginning ($t = 1$, $\eta = a_{\max}$) and minimal at the end of the learning procedure ($t = t_{\max}$, $\eta = a_{\min}$). The time dependency of η (dependency on the consecutive iteration number N_{it}) is given by Eq. 3:

$$\eta(t) = (a_{\max} - a_{\min,}) \times p + a_{\min,}\ p = (N_{\text{it-tot}} - N_{it})/(N_{\text{it-tot}} - 1) \tag{3}$$

The function $b(\cdot)$ in Eq. 2 describes how the correction of the weights decreases with increasing topological distance between the central neuron and the neuron being corrected. Usually, we apply the triangular correction function, which decreases linearly from the central neuron to the furthest neighbor. Index *j* specifies the individual neuron and runs from 1 to *n*. The topological distance of the *j*th neuron from the central one is defined according to the topology used for the distribution of neurons in the plane: In the rectangle net, the central neuron has 8 first neighbors (d_c–d_j = 1), 16 second neighbors (d_c d_j = 2), and so forth. The minimal distance is zero ($j = c$, d_c–d_j = 0), which corresponds to the maximal correction function (b = 1). The maximal distance (d_c–$d_{\max}$) to which the correction is applied shrinks during the learning procedure. The correction function at maximal distance is minimal (b = 0). At the beginning, the d_c–$d_{\max}$ covers the entire network, while at the end, at $t = t_{\max}$, it is limited to only the central neuron.

The unsupervised training is usually carried out for a predefined number of training epochs or iteration cycles, $N_{\text{it-tot}}$, although other stop criteria, such as the minimal threshold for the total difference between all the input vectors and the corresponding winning neurons can be used.

The topology of the Kohonen neural network can be influenced by the "toroid" boundary conditions, which means that the network is continuously connected through the edges. The right or top edge is, in a computational sense, connected to the left and low edge, respectively, and vice versa. Unfortunately, the toroid conditions decrease the available mapping area for a factor of 4 compared to the same dimensional nontoroid network. In cases where a large area for mapping is needed, the possibility of a two times larger maximal topological distance between two neurons can be a predominant factor for the use of nontoroid boundary conditions.

2.1.4. Properties of Trained Kohonen Neural Networks

A trained Kohonen neural network consists of μ-dimensional neurons organized in a matrix $N_x \times N_y$, with the weights adapted to accommodate the objects from the training set. In other words, the corrected weights are stored as cells in a block defined by the length, height, and depth equal to the network parameters ($N_x \times N_y \times m$). Presenting the entire set of objects to the trained network, we obtain the locations of the winning neurons in the $N_x \times N_y$ map, excited by individual objects. If we mark the excited neurons in the map by labels corresponding to individual objects, we obtain a so-called top map. The labels can be chosen according to our knowledge of properties of objects (chemical classes, boiling constants, toxicity, to list just a few possibilities from different applicability domains). In the top map, one can find clusters of objects, empty spaces (neurons that were not excited by any of the training objects), or conflicts (neurons excited by two or more objects from different classes or having different properties). Clusters and empty spaces can be inspected without prior knowledge of property (dependent variables) of studied objects, while the conflicts can be determined only by knowing the properties as well.

Kohonen neural network presented as a block of weights $N_x \times N_y \times m$ can be inspected through the levels of weights, that is, the third dimension of the block. The number of levels is equal to the number of independent variables; each level is associated with one variable. The weight maps of dimension $N_x \times N_y$ can be very informative if compared to the top maps with labels of objects distributed over the network at the end of the training.

2.2. *Counterpropagation Neural Networks*

The counterpropagation neural network is based on a two-step learning procedure, which is unsupervised in the first step. The first step corresponds to the mapping of objects in the input layer (also called the *Kohonen layer*). This part is identical to the Kohonen learning procedure just described. The second step of the learning is supervised, which means that, for the learning procedure, the response or target value is required for each input. The network is thus trained with a set of input-target pairs $\{\boldsymbol{X}_s,\boldsymbol{T}_s\}$, where $\boldsymbol{T}_s$ is the vector representing dependent variables.

2.2.1. Supervised Strategy of Learning

The training of the counterpropagation neural network means adjusting the weights of the neurons in such a way that, for each input sample $\boldsymbol{X}_s$ from the training set, the network responds with the output $\mathbf{Out}_s$ identical to the target $\boldsymbol{T}_s$. The training is an iterative procedure similar to the procedure described for the Kohonen neural network, only dependent variables or target vectors are considered as well. It involves feeding all input-output pairs $\{\boldsymbol{X}_s,\boldsymbol{T}_s\}$ to the network, finding the central neuron in the input layer for each $\boldsymbol{X}_s$, and correcting the weights of the neurons, not only in the input but also in the output layer, according to the differences between the targets and current outputs $(\boldsymbol{T}_s - \mathbf{Out}_s)$. As already stressed, the targets are needed only in the last part of each iterative learning step. The unsupervised element in the counterpropagation neural network learning procedure is the mapping of the objects vectors into the Kohonen layer, which is based solely on the independent variables, that is, the $\boldsymbol{X}$-part of the $\{\boldsymbol{X}_s, \boldsymbol{T}_s\}$ pairs of the objects from the training set. For this step, no knowledge about the target vector (property) is needed. Once the position (central neuron c) of the input vector is defined, the weights of the input and output layers of the counterpropagation neural network are corrected accordingly. The corrections in the output layer are defined next (Eq. 4), while the corrections of the weights in the input layer were given previously, see Eq. 2.

$$Out_{ji}^{\mathrm{new}} = Out_{ji}^{\mathrm{old}} + \eta(t) \times b(d_c - d_j) \times (T_{si} - Out_{ji}^{old}) \tag{4}$$

2.2.2. Properties of Trained Counterpropagation Neural Networks

The input layer of the trained counterpropagation neural network is identical to the Kohonen neural network; it accommodates all objects from the training set. At the end of the training, all objects from the training set are recognized. Consequently, we can inspect top maps with the labels of objects and identify clusters, empty spaces, and conflicts, as discussed in section 4.2.1.4. The trained counterpropagation neural network also reveals the distribution of weights in *m* levels of the input layer. In addition, the output layer is formed, which adopts the structure of weights by receiving the information about the location of the central neuron and considering the actual target value of a particular object in each iteration step, for each object input. The output layer enables the predictive ability of the counterpropagation neural networks. The prediction of a property ($\boldsymbol{T}_{\mathbf{new}}$) for any unknown input object ($\boldsymbol{X}_{\mathbf{new}}$) is performed by inserting the $\boldsymbol{X}_{\mathbf{new}}$ into the input layer of the trained counterpropagation neural network, locating it (finding the most similar or central neuron), then disclosing the weights in the output layer at the position of the central neuron. Basically, the counterpropagation acts as a pointer device: As many predictions are available as there are units in the output layer (equal to the number of neurons). However, it has to be stressed that not only the exact values equal to the targets of the training objects can be obtained. The intermediate (interpolated) values can be also predicted, resulting from the training process that encompasses all neurons, also those that remain empty after the input of training objects to the trained network. The empty neurons may better accommodate a new, unknown object, not equal or similar to any of the objects from the training set. The degree of similarity is a question of definition and by no means a simple one. Here, we would like to focus on only the Euclidean distance between the object representation vector $\boldsymbol{X}_s$ and the excited neuron $\boldsymbol{W}_c$ (see Eq. 1). If this distance for an unknown object becomes very large, it also means that the empty neurons cannot accommodate the structure of the new object, which may be treated as an outlier. The resulting prediction becomes unreliable. This would happen, for example, if we prepared the infrared spectra interpretation model with only the aliphatic alcohols in the training set; and then we would like to predict a carboxylic acid or a compound containing benzene rings. Fortunately, we can incorporate a reliability factor in the counterpropagation model, as will be explained in the example presented next.

3. Data Used for the Application Study

3.1. *Infrared Spectral Data*

The collection of 3,284 IR spectra of organic compounds of known chemical structure was used for building and validating the model. Each spectrum was stored as a set of 650 intensity points, recorded in the region from 4,000 to 200 cm^{-1}. Only 512 intensities in the range from 3,500 to 550 cm^{-1} with the corresponding resolution were selected for further manipulation. On the sets of 512 intensity points, the fast Hadamard transformation was applied [8–9]. After the transformation of 512 intensities, the resulting sets of 512 Hadamard coefficients were truncated to 128; hence, a one to four reduction of the original spectrum representation was achieved. The reduction reflects mainly in a lower resolution (high-frequency terms). The reduced spectra given with representation vectors $\boldsymbol{X}_s = (x_{s1}, x_{s2}, \ldots, x_{si}, \ldots, x_{s128})$ were used for model input.

3.2. *Representation of the Chemical Structure of the Organic Compounds Studied*

The structure representation vectors were 34-dimensional binary vectors, $\boldsymbol{T}_s$ denoting the presence ($t_{sj} = 1$) or absence ($t_{sj} = 0$) of the *j*th structural fragment. The structural fragments were chosen on the basis of their characteristic IR absorption frequencies. All 3,284 compounds in the data set were scanned for the occurrence of 1 or more of the 34 fragments (Table 4.1).

4. Infrared Spectra Interpretation Model

A heuristic model on the basis of artificial neural networks (ANNs) for prediction of structural fragments of an unknown compound from its infrared spectrum is described as a case study (from [10]). The information needed for training the ANN model was drawn from a collection containing IR spectral and structural data of 3,284 compounds. The model was based on two ANNs: the self-organising Kohonen neural network for mapping of IR spectra into a two-dimensional plane and the counterpropagation neural network for the determination of the structural features. The preliminary learning in the Kohonen neural network with all spectra from the collection yields information for possible grouping. The preliminary grouping was used for separating the spectra into the training and test sets, containing 755 and 2,529 spectrum-structure pairs, respectively. The counterpropagation neural network was trained with the spectrum-structure pairs collected in the training set. The spectra from the test set were used to assess the ability of the obtained ANN model to predict structural fragments of an unknown compound from its IR spectrum.

Table 4.1 Structural Fragments of the compounds from the data set defining the structure-representation vector $\boldsymbol{T}_s = (t_{s1}, \ldots, t_{sj}, \ldots, t_{s34})$ of the sth compound

j	Label	Functional group (fragment)	No. of compounds with t_{sj}=1
1	H	OH	630
2	M	Alcohol	462
3	P	Primary alcohol	288
4	S	Secondary alcohol	171
5	T	Tertiary alcohol	30
6	G	1,2 glycol	29
7	F	Phenol	68
8	R	Aryl-CH_2-OH	31
9	N	NH	412
10	1	Primary amine	250
11	2	Secondary amine	146
12	0	CN	842
13	3	Tertiary amine	170
14	C	C=O	1006
15	A	COOH	87
16	W	CO-O-	415
17	E	Ester	409
18	Y	Aldehyde	120
19	K	Ketone	284
20	D	Amide	65
21	J	Benzene	1055
22	4	Naphthalene	28
23	5	Furan	25
24	6	Tiophene	29
25	7	Pyridine	81
26	Z	NO_2	83
27	V	Aryl-NO_2	60
28	8	C-O-	1456
29	O	Ether	471
30	X	C-X	841
31	U	C-F	218
32	L	C-Cl	438
33	B	C-Br	214
34	I	C-I	42

Source: Taken from [*10*].

4.1. Mapping of IR Spectra Using the Kohonen Neural Network

The Kohonen neural network of 900 neurons (i.e., 30 × 30 layout) was used for the mapping procedure. The number of weights in the neurons was equal to the dimension of the input vector, 128 in the case of Hadamard-transformed IR spectra. Initially, the weights of the neurons were randomized in the interval [0, 1]; the parameters used in the network training were 20 epochs (20 × 3,284 = 65,680 iterations), 0.5 correction factor, and a nontoroidal condition.

The number of weights in each neuron defined the number of the *levels* (maps) in the Kohonen network. In addition to 128 weight maps, the zeroth or the *top-level map of labels* was constructed for a quick inspection of formed clusters. Each label from the top-level map indicated the position of a spectrum tagged with this label. The labels are associated with structural fragments (see Table 4.1). Obviously, the compounds containing more than 1 of 34 structural fragments cannot be properly assigned one label. The label of the first fragment found in the applied substructure search procedure was given to the input spectrum of a compound with a particular chemical structure. Therefore, the labels have to be regarded as only partial information. Nevertheless, the complete information about the presence of multiple fragments is available in the associated output files for each chemical structure. The map of labels was generated at the end of the training. For every input spectrum, at the position of the excited neuron in the 30 × 30 neuron matrix, the corresponding position in the 30 × 30 label map received the label of the input spectrum. Because we had about four times as many spectra as neurons in the network, individual neurons were excited by different spectra; all 3,284 labels should be displayed in the 30 × 30 map. For example, six neurons in the 30 × 30 network were excited by at least 18 different spectra; therefore, each of these six neurons carried at least 18 labels.

4.2. Selection of Training and Test Sets

By analyzing the *top-level map of labels*, we can track the formation of clusters. The inspection of chemical structures of spectra in the same cluster confirms common structural fragments. This indicates that the predictions of structural features can be linked to the spectral features that have produced the grouping of spectra reflected in the top maps. A good overlap between the contours encircling clusters of individual fragments in the top map and the contours in the Kohonen layer indicating different IR spectral absorption regions was found. This overlap confirms that Kohonen and counterpropagation neural networks are able to extract correlations between chemical structure and IR spectra automatically. From the Kohonen neural network, 755 spectra-structure pairs, one from each excited neuron, were selected for supervised training of the counterpropagation neural network described in the paragraphs that follow. The selected training set fulfills the demand for the even distribution of the training data in the spectral representation space.

4.3. Construction of Predictive Models on the Basis of Counterpropagation Neural Networks

The architecture of the counterpropagation neural network was 30 × 30 × (128 + 34). The other parameters used for the counterpropagation learning were 100 epochs (75,500 iterations), a triangular correction function of 0.5 maximum, and a nontoroidal condition. After accomplishing the training with 755 objects (spectra and corresponding 34 structural descriptors), all the neurons (30 × 30 = 900) in the Kohonen layer had been adapted to all 755 spectra with an overall discrepancy of 5% per weight. The discrepancy of 5% is a total root mean square (RMS) error calculated for 96,640 weights (755 × 128). At the same time, the 30,600 (34 × 900) weights in the output layer had been changing iteratively with the 34-dimensional target vector. The resulting 34-dimensional output neurons are therefore the best fit to the 755 binary target vectors distributed over the network.

The output layer of 30,600 [(30 × 30) × 34] weights arranged in 34 levels was interpreted as 34 response surfaces. The *j*th response surface (map of output weights) contains 30 × 30 probabilities, and the location of the *s*th structure defines the probability of presence of the *j*th structural fragment connected to the corresponding map. Each weight of the 34-dimensional *j*th output neuron carries a probability value between $0.0 \le \text{out}_j \le 1.0$. The probability values in the map are influenced by the neighboring values during the training procedure. Therefore, the probabilities out_j vary due not only to the different absolute number of fragments present in the training structures but also to the number of clusters of spectra in the map. The ratio of hits of objects with and without the *j*th fragment defines one single out_j and influences the closest neighborhood.

The resulting model based on counterpropagation neural network is able to predict, within certain reliability limits, the probability of presence or absence of any of the 34 structural fragments. The predictions of 34 structural fragments of any test object are found in the output layer (constructed of 34 response surfaces) immediately below the excited neuron in the Kohonen layer. The probabilities in each response surface are spread over the entire map of dimension 30 × 30, thus offering 900 values for out_j in the range from 0.0 to 1.0. For the reasons described in the preceding paragraph (the nature of supervised training of the counterpropagation neural network), the test is not likely to obtain "clear" predictions, that is, the probability of out_j being exactly 1.0 or 0.0 for presence or absence of structural fragment, respectively. Therefore, a threshold value should be defined, above which the prediction for the presence of the *j*th fragment is positive. The threshold u_j^{thr} is determined as the probability value at which the number of hits of the objects having the fragment *j* exceeds the number of hits of those not having it. The threshold values were determined for all 34 structural fragments. For an affirmative (positive) decision about the presence of the *j*th structural fragment, the predictive probability value, response out*j* at the position of the excited neuron, must be larger or equal to the threshold u_j^{thr} determined for the *j*th response surface. The affirmative prediction can be either a true positive or a false positive. A true positive is a correct prediction, while a false-positive decision is a prediction in which the system classifies an unknown compound as if it had the fragment *j* even though it does not have it.

4.4. Assessing the Prediction Ability and Reliability of Predictions of Structural Features

The prediction ability of the constructed counterpropagation ANN was obtained by testing 2,529 spectra-structure pairs that were not used in the training phase. In Table 4.2 the predictions results about the presence for each of the 34 structural fragments and the reliability (see explanation later) of each affirmative predictions are given.

The prediction ability ${}^{+}p_j$ or ${}^{-}p_j$ is a fraction of the true-positive (or false-positive) decisions for the *j*th fragment. Equation 5 describes the true-positive prediction ability:

$$
{}^{+}p_j = \frac{{}^{+}n_j}{{}^{+}N_{fr_j}} \tag{5}
$$

where ${}^{+}n_j$ is the number of correct affirmative predictions (true positive) and ${}^{+}N_{fr_j}$ is the number of compounds in the test set with fragment *j*. Evidently, an

exactly equivalent equation holds for the false-positive decisions ${}^{-}p_j$, where ${}^{-}n_j$ is the number of incorrect affirmative predictions (false positive) and ${}^{-}N_{frj}$ is the number of compounds in the test set without the *j*th fragment. The prediction ability, ${}^{+}p_j$ and ${}^{-}p_j$ for the *j*th fragment by a counterpropagation neural network model are given in the sixth and ninth columns of Table 4.2, respectively.
However, the prediction ability alone does not show information how reliable such predictions are. If, for example, the prediction ability ${}^{+}p_j = 1$, meaning that all compounds from the test set having the fragment *j* are predicted correctly, is obtained by the rule *any compound has the* j*th fragment*, the result contains no useful information. This rule would classify wrongly all objects not having the fragment *j* as if they had it, whicht would make the 100% correct predictions completely useless, that is, not reliable regarding the actual presence of fragment *j* in test compounds. For a reliable result, the fraction of false-positive decisions ${}^{-}p_j$ in the test set has to be considered simultaneously. From the two predictions, the true-positive ${}^{+}p_j$ and the false-positive ${}^{-}p_j$, the reliability R_j for predictions of the *j*th fragment is calculated according to Eq. 6:

$$
R_j = \frac{2({}^{+}p_j - {}^{-}p_j)}{1 + \left|{}^{+}p_j - {}^{-}p_j\right|}
$$

The reliability value R_j ranges from 1 to –1. In the best case, all compounds having fragment *j* are correctly predicted as such and none of the "not haves" is predicted as to have the fragment *j*, then the reliability is maximal ($R_j = 1$). It is interesting to note that in the case when false-positive prediction is zero (${}^{-}p_j = 0$), the reliability R_j is always higher or equal to the true-positive prediction value. This means that, if the system predicts the presence of a fragment, such prediction is at

Table 4.2 Prediction abilities and reliability of predictions for 34 structural fragments with the counterpropagation neural network model obtained on 2,529 test objects

j	Fragment	u_j^{thr}	$^{+}Nfr_j$	$^{+}n_j$	$^{+}p_j$	$^{-}Nf_{rj}$	$^{-}n_j$	$^{-}p_j$	R_j
1	OH	0.3	519	473	0.91	2010	71	0.04	0.93
2	Alcohol	0.3	377	349	0.93	2152	59	0.03	0.95
3	Prim. alcohol	0.5	239	144	0.60	2290	61	0.03	0.73
4	Sec. alcohol	0.3	132	100	0.76	2397	118	0.05	0.83
5	Tert. alcohol	0.2	24	15	0.63	2505	2	0.00	0.77
6	1,2 glycol	0.3	23	10	0.43	2506	27	0.01	0.59
7	Phenol	0.3	53	41	0.77	2476	30	0.01	0.87
8	Aryl-CH_2-OH	0.3	26	10	0.38	2503	27	0.01	0.54
9	NH	0.3	316	268	0.85	2213	80	0.04	0.90
10	Prim. amine	0.3	202	168	0.83	2327	28	0.01	0.90
11	Sec. amine	0.3	105	75	0.71	2424	86	0.04	0.80
12	CN	0.4	633	463	0.73	1896	184	0.10	0.77
13	Tert. amine	0.3	123	74	0.60	2406	53	0.02	0.73
14	C=O	0.5	775	743	0.96	1754	40	0.02	0.97
15	COOH	0.3	78	73	0.94	2451	10	0.00	0.97
16	CO-O-	0.3	319	284	0.89	2210	50	0.02	0.91
17	Ester	0.3	316	281	0.89	2213	50	0.02	0.91
18	Aldehyde	0.3	92	42	0.46	2437	39	0.02	0.59
19	Ketone	0.4	211	159	0.75	2318	87	0.04	0.83
20	Amide	0.3	49	25	0.51	2480	16	0.01	0.67
21	Benzene	0.5	786	616	0.78	1743	153	0.09	0.82
22	Naphthalene	0.2	18	10	0.56	2511	16	0.01	0.71
23	Furan	0.3	16	5	0.31	2513	5	0.00	0.47
24	Tiophene	0.2	21	4	0.19	2508	25	0.01	0.31
25	Pyridine	0.3	55	27	0.49	2474	42	0.02	0.64
26	NO_2	0.3	68	54	0.79	2461	24	0.01	0.88
27	Aryl-NO_2	0.4	50	33	0.66	2479	18	0.01	0.79
28	C-O-	0.5	1117	979	0.88	1412	143	0.10	0.88
29	Ether	0.4	340	210	0.62	2189	132	0.06	0.72
30	C-X	0.4	623	410	0.66	1906	245	0.13	0.69
31	C-F	0.3	161	97	0.60	2368	97	0.04	0.72
32	C-Cl	0.3	323	167	0.52	2206	244	0.11	0.58
33	C-Br	0.3	162	49	0.30	2367	131	0.06	0.39
34	C-I	0.4	35	0	0.00	2494	2	0.06	–0.13
$(N_{frj} \times +p_j)/\, N_{frj}$					0.77	$\Sigma(N_{frj} \times R_j)/\Sigma N_{frj}$			0.82

Source: Taken from [*10*].Note: Sequential number (column 1), description of a fragment (column 2), corresponding threshold value u_j^{thr} (column 3), number of objects in the test set containing the *j*th fragment (column 4), number of correct answers (column 5), fraction of true positive answers (column 6), number of objects not having the particular fragment (column 7) and fragments predicted wrongly as having it (column 8), fraction of false-positive predictions (column 9), reliability of each prediction according to Eq. 6 (column 10)

least as reliable as the true-positive prediction value. One extreme situation is the case where all compounds having a fragment *j* are predicted as not having it ($^{+}p_j = 0$) and all "not haves" are predicted as having it ($^{-}p_j = 1$); the reliability R_j is equal to –1 signaling that such decision is extremely reliable: Because all of them are wrong, one must only reverse the sign of the decision to obtain the correct one.

In the remaining two extreme cases, the system gives the same answer for all objects; that is, either all objects have the fragment *j* ($^+p_j = 1$, $^-p_j = 1$) or none of the objects has it ($^+p_j = 0$, $^-p_j = 0$), the reliability is zero. In general, if true-positive and false-positive predictions are equal ($^+p_j = {}^-p_j$), the reliability of prediction ability for the *j*th fragment is zero.

It can be seen from Table 4.2 that several structural fragments were predicted correctly in more than 90% of cases; the most successful was the prediction for the –C=O, 96% of correct predictions. It should be emphasized that the reliability of these predictions is very high, too. Good performance, that is, the ability to predict the presence of a specific fragment in more then 80% cases, was found for the fragments No. 1, 2, 9, 10, 15, 16, 17, and 28 (see Table 4.1 for the description of fragments). However, poor prediction results of less than 35% were obtained for fragments No. 23, 24, 33, and 34 (furan, tiophene, C-Br, and C-I, respectively). The most characteristic fragment from this group is No. 33 (C-Br). Even though 52 compounds with this fragment are in the training set (compared to 18 compounds having the –COOH group yielding $^+p_i = 0.94$ and $R_i = 0.97$), the achieved results are quite poor ($^+p_i = 0.30$ and $R_i = 0.39$). It can be concluded that the vibrations caused by this and similar fragments are not significant in the infrared domain or their influences on the spectrum are lost among other, more dominant spectral features. For better predictions of such fragments, more investigation would be necessary in the direction of changing the spectral representation, subtracting the elementary characteristic spectral lines, and emphasizing the variations of the characteristic spectral lines. Since the infrared spectrum reflects the whole chemical structure, the correlation of narrow spectral regions to only one structural fragment is not always straightforward.

4.5. Reversibility of the Counterpropagation Neural Network Model for IR Spectra Simulation

We already mentioned in the Introduction the potential of counterpropagation neural network models for the reverse prediction. In this example, the infrared spectra were simulated from their structure. The simulation of infrared spectra by the proposed scheme is really simple. The roles of the input and output data are simply reversed. Instead of entering the 128-dimensional spectral representation to the Kohonen layer, the 34-dimensional binary structural description based on the presence or absence of considered structural fragments is entered into the output layer of the existing (trained) counterpropagation neural network. The central (excited) neuron *c* is obtained in the output, not in the Kohonen layer, by applying the criterion given in Eq. 1 on the structural representations in the output neurons. Once the central neuron *c* is determined, the simulated spectrum is found in the Kohonen layer, in the neuron exactly above the neuron *c* of the output layer. It has to be stressed that the counterpropagation neural network models capable of simulating any IR spectrum would become rather complex, because they would have to be trained with a large amount of available IR spectra and therefore the dimension of the map would increase considerably.

5. Notes

1. In the study outlined here, a classification model is constructed by training the counterpropagation neural network. Those familiar with the more frequently used error-back-propagation neural networks often judge and interpret counterpropagation neural network models from the wrong perspective. The two main differences between the error-back-propagation and counterpropagation neural networks are the learning strategy and the connection between the layers, explained in notes 2 and 3.
2. Contrary to error-back-propagation neural networks, the learning strategy of the counterpropagation neural network is not supervised in all subsequent stages of the training process. Two steps are iteratively repeated for all object of the training data: (i) finding the position of the object in the two-dimensional map (the most similar neuron in the input or Kohonen layer), which is an unsupervised routine based solely on the object representation or independent variables, and (ii) correction of the weights, which also encompasses the output neurons and consequently the property or target values are needed for this purpose. In this stage, the supervised part is introduced into the training process.
3. There is no hidden layer in the counterpropagation neural network. The output (Grossberg) layer is positioned directly below the input (Kohonen) layer, with a one-to-one correspondence of neurons. This means that each neuron from the input layer at a position (N_x, N_y) has an ascribed property stored in the output layer at the same position (N_x, N_y). In fact, the output layer, when properly trained with training data, serves as a lookup table for all neurons from the input layer. It has to be stressed here that, in the process of training, all the neurons are affected, not only the central neurons fired by the object. The neighboring neurons around the central one may remain “unoccupied” at the end of the training; consequently, the output layer contains also values different from the properties of the training objects (interpolated values between those from two occupied neurons). However, there is no chance to obtain predictions out of the range of properties of the training data, which means that extrapolations are not feasible with counterpropagation neural networks. This can be regarded as an advantage, because it prevents unreliable extrapolated predictions, not viable in the experience-based neural network models.
4. The dimensionality of counterpropagation models presented in this study may seem to be a drawback if compared with error-back-propagation of multilinear regression models. However, we stress that the large number of variables (points of IR spectra) is not problematic from the conceptual point of view, because the counterpropagation neural network model functions in the first stage as a mapping device, followed by the so-called supervised training. Thus, only two

parameters, the coordinates of the position of each multidimensional object in the Kohonen map, are correlated to the target (property or class) to create the response surface in the output layer of the counterpropagation neural network. Nevertheless, a large number of variables might have a negative influence on the formation of clusters in the Kohonen layer. Nonrelevant variables may be compared to a noise in the structure representation vectors. To reduce the noise and build more robust models with a minimal bias to the particular data set, a variable selection procedure may be employed and reduced models constructed. Instead, in this study, we employed a Hadamard transformation to reduce the dimension of objects and consequently the noise in IR spectra.

References

1. Kohonen T (1988) Self-organization and associative memory. Springer-Verlag, Berlin.
2. Kohonen T (2001) Self organizing maps (3rd edn). Springer, Heidelberg.
3. Dayhof J (1990) Neural network architectures, an introduction. Van Nostrand Reinhold, New York.
4. Carpenter G, Grossberg S (1988) The art of adaptive pattern recognition by a self-organizing neural network. IEEE Computer 21:77–88.
5. Hecht-Nielsen R (1987) Counterpropagation networks. Appl. Optics 26:4979–4984.
6. Zupan J, Gasteiger J (1999) Neural networks in chemistry and drug design (2nd edn). Wiley-VCH, Weinheim.
7. Zupan J, Novič M, Ruisanchez I (1997) Kohonen and counterpropagation artificial neural networks in analytical chemistry: tutorial. Chemometr Intell Lab Syst 38:1–23.
8. Razinger M, Novič M (1990) Reduction of the information space for data collections. In: Zupan J (ed) PCs for chemist. Elsevier, Amsterdam, pp. 89–103.
9. Graff DK (1995) Fourier and Hadamard: transforms in spectroscopy, J Chem Ed 72:304–309.
10. Novič M, Zupan J (1995). Investigation of infrared spectra-structure correlation using Kohonen and counterpropagation neural-network, J Chem Info Comp Sci 35:454–466.

Chapter 5
Artificial Neural Network Modeling in Environmental Toxicology

James Devillers

Abstract Artificial neural networks are increasingly used in environmental toxicology to find complex relationships between the ecotoxicity of xenobiotics and their structure and/or physicochemical properties. The raison d'être of these nonlinear tools is their ability to derive powerful QSARs for molecules presenting different mechanisms of action. In this chapter, the main QSAR models derived for aquatic and terrestrial species are reviewed. Their characteristics and modeling performances are deeply analyzed.

Keywords Supervised artificial neural network, noncongeneric QSAR, bacteria, protozoa, crustacea, insects, fish

1. Introduction

Worldwide, the estimate of humanmade chemicals in use is in excess of 120,000 [1]. With such a large number of xenobiotics, which potentially can be released into the aquatic and terrestrial ecosystems, it is impossible to test all of them exhaustively for their short- and long-term effects on the living species. This is due to time and funding constraints but also because laboratory toxicity tests are more and more fought by Animal Rightists.

Fortunately, to overcome the different problems in relation to laboratory testing, attempts have been made for proposing modeling approaches allowing the simulation of the environmental fate and toxicity of chemicals.

Since the mid-20th century, various modeling approaches, generically termed *QSARs* (quantitative structure-activity relationships), have been developed for predicting the activity of chemicals from their molecular structure and/or physicochemical properties. Initially used in the pharmaceutical industry in the development process of new drugs for optimizing their targeted activity and also for reducing their side effects, the SAR (structure-activity relationship)/QSAR methodology started to be employed only in the 1980s for predicting the toxicity of chemicals of environmental concern [2].

In the past, the common rule was to design QSAR models from small data sets of chemicals of rather similar structures and acting according to a same mechanism of action (MOA). Because it was implicitly postulated that the relationship between the biological activity of a chemical and its structure or physicochemical properties, encoded by molecular descriptors, was obligatorily linear, only classical regression analyses and PLS analysis [3] were used. In fact, like most of the real-world problems, which are dominated by nonlinearity, the relations between the structure of the molecules and their biological activity are most often nonlinear. Consequently, the use of artificial neural networks (ANNs) was particularly suited for deriving such relationships [4]. From these tools it became possible to design powerful SAR and QSAR models from large sets of structurally diverse chemicals encompassing different MOA [5].

This chapter will look at some of the issues, results and recent attempts in the field of ANN modeling applied to environmental toxicology. The SAR and QSAR models are presented according to the different types of ecosystems and species.

2. Aquatic Ecosystem

2.1. QSAR Models on Bacteria

The Microtox™ test measures the reduction in light output of the natural luminescence (EC_{50}) of the marine luminescent bacterium *Vibrio fischeri* (formerly known as *Photobacterium phosphoreum*) following a short exposure to a toxicant of interest [6]. Originally developed for quick toxicity testing of effluents, the assay was rapidly used to test xenobiotics, yielding the production of a huge number of toxicity results [7] particularly useful for deriving noncongeneric QSAR models.

Therefore, a three-layer feedforward ANN trained by the back-propagation algorithm (BPA) [8, 9] was used by Devillers et al. [10–12] to find nonlinear relationships between autocorrelation descriptors [13] and Microtox toxicity data [7]. After preliminary investigations allowed finding an optimal modeling strategy, a general QSAR model was derived from 30-min EC50 values expressed as log 1/*C* in mmol/l [12]. Toxicity values recorded after 5 and 15 min of exposure were also used for structurally interesting chemicals for which 30-min EC50s were not available. A learning set of 1,068 chemicals and a testing set of 240 chemicals were selected by means of the N2M method [14]. Molecules were described by means of four autocorrelation vectors [13] encoding lipophilicity (H), molar refractivity (MR), H-bonding acceptor (HBA) ability, and H-bonding donor (HBD) ability. While good simulation results were found with different BNN configurations, in all cases, large outliers were detected. Because there was no reason to remove these outliers, it was decided to increase the complexity of the network by introducing the time of exposure as additional input variable, yielding a new training set and testing set of 2,795 and 385 results, respectively (i.e., 1,068 and 240 chemicals tested at different times of exposure, respectively). On the basis of numerous trials, a significant reduction in the number of large outliers was noted. The best result was obtained with a 36/26/1 ANN (n = 2,795, $\eta = 0.5$, $\alpha = 0.9$, 4,977 cycles, input neurons = H_0 to H_{14}, MR_0 to MR_{14}, HBA_0 to HBA_3, $HBD_{0,}$ and time of exposure). Plots of the observed versus calculated toxicity values for the training and external testing sets are shown in Figs. 5.1 and 5.2, respectively. Interestingly, the performances of the selected ANN model were equivalent or superior to those of regression models designed for chemicals acting with a specific MOA [12].

2.2. *QSAR Models on Protozoa*

Protozoa play a key role in many ecological communities, where they occupy a range of trophic levels. Indeed, they can be herbivores, decomposers, or predators of microorganisms. The protozoa are also an important food source for various macroinvertebrates.

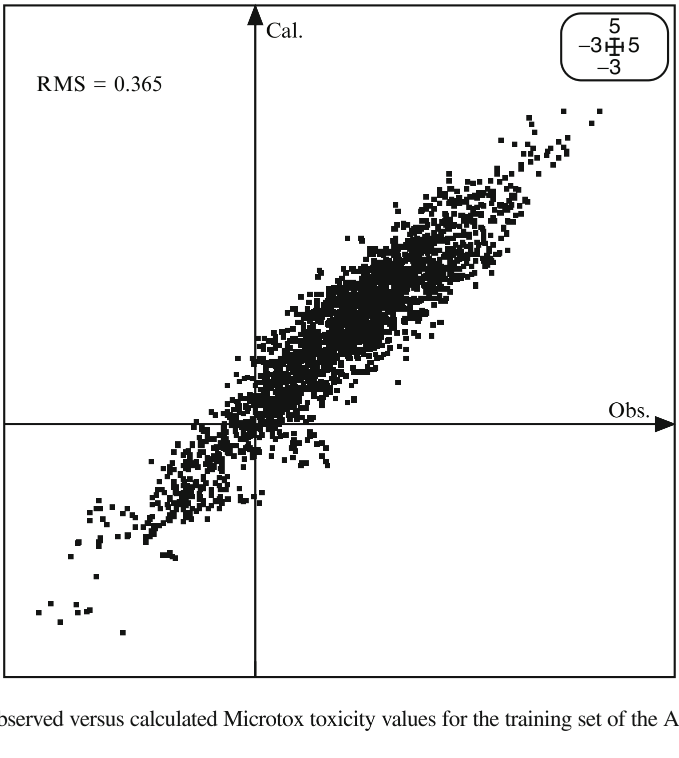


Fig. 5.1 Observed versus calculated Microtox toxicity values for the training set of the ANN model

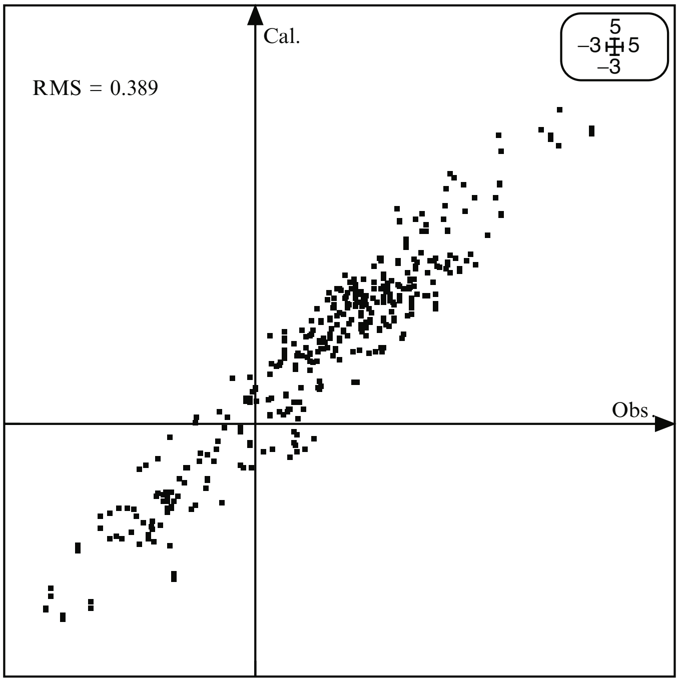


Fig. 5.2 Observed versus calculated Microtox toxicity values for the external testing set of the ANN model

The freshwater ciliate *Tetrahymena pyriformis* has been tested on a large number of industrial organic chemicals. The TETRATOX database (www.vet.utk.edu/TETRATOX) includes comprehensive toxicity results for 2,400 aliphatic and aromatic chemicals tested on this protozoan. This database represents an invaluable source of toxicity data for deriving QSAR models. For example, a congeneric QSAR model (Eq. 1) was proposed by Baker, Wesley, and Schultz [15] for predicting the toxicity of phenols eliciting a polar narcosis mechanism of toxic action.

$$\begin{gathered}\log \mathrm{BR} = 0.609 \log \mathrm{Kow} - 0.235\,\mathrm{pKa} + 1.132 \\ n = 50,\ r^2 = 0.916,\ s = 0.197,\ F = 255.17\end{gathered} \tag{1}$$

Using the same toxicity data but a larger set of 135 molecular descriptors, the performances of multiple linear regression analysis and a three-layer feedforward neural network trained by the BPA or quasi-Newton algorithm (QNA) [16] were compared by Xu and coworkers [17]. An equation at nine parameters encoding the topology, shape, and charged partial surface area of phenols was first selected. The descriptors were then used as inputs in various neural network architectures. The best results were obtained with a 9/3/1 ANN trained with QNA during 840 cycles. Even if the linear and nonlinear models obtained by Xu and coworkers [17] provide better fit of the experimental toxicity data than Eq. 1, they are of lesser quality, including many more parameters and being more difficult to interpret. It is noteworthy that the same modeling strategy was applied on a larger set of molecules [18].

Another comparison exercise was performed by Burden and Winkler [19] using a PLS analysis and a Bayesian regularized artificial neural network (BRANN) [20] to develop a QSAR model from a set of 278 substituted benzenes described by means of topological and structural indices. Principal components analysis (PCA) was employed for data reduction and scaling. The best PLS model included ten components and showed a standard error of estimation (SEE), r^2, standard error of prediction (SEP), and Q^2 of 0.104, 0.688, 0.112, and 0.631, respectively. The most interesting BRANN model included 18 principal components and presented a SEE, r^2, SEP, and Q^2 of 0.064, 0.943, 0.106, and 0.808, respectively. BRANNs were also used by Winkler and Burden [21] for successfully modeling a larger set of 505 aromatic compounds tested on *T. pyriformis*.

PLS and ANN QSARs on *T. pyriformis* were also developed and compared by Devillers [22] from a training and a testing set of 200 and 50 phenols, respectively. The chemicals, encompassing five MOA (i.e., polar narcotic, respiratory uncoupler, proelectrophile, soft electrophile, proredox cycler), were previously analyzed by Cronin and coworkers [23] from regression and PLS analysis. The seven following molecular descriptors were selected by Cronin et al. [23]: the distribution coefficient (log D) at pH = 7.35, the energy of the lowest unoccupied molecular orbital (LUMO), the molecular weight (MW), the negatively charged molecular surface area (P_{NEG}), the electrotopological state index for the hydroxy group (SsOH), the sum of absolute charges on nitrogen and oxygen atoms (ABSQon), and the largest positive charge on a hydrogen atom (MaxHp). From these descriptors, a first PLS model using the NIPALS (nonlinear estimation by iterative partial least squares) algorithm at two significant components was derived (L1). The same set of descriptors was used for computing a three-layer ANN model with a minimum number of connections and cycles, to avoid overfitting and overtraining. A simple 4/1/1 ANN model (N1) with no bias and only log D, LUMO, MW, and P_{NEG} as inputs yielded interesting results in only 145 cycles (100 with BPA and 45 with the conjugate gradient descent algorithm, or CGDA). All the synaptic functions for the three layers were linear; and the activation functions were linear, hyperbolic, and logistic from the input to the output layer.

A significant improvement was obtained by the introduction of indicator variables encoding hydroquinone derivatives, para-amino groups, and 2,6-substituted phenols. A new PLS model, including three significant components (L2), was obtained from the seven original and three Boolean descriptors. The introduction of these descriptors as inputs in the neural network often yielded the exclusion of SsOH. Consequently, the best results were obtained with a 9/3/1 ANN without bias (N2). The number of cycles was 218 (i.e., 100 BPA and 118 CGDA) and the synaptic and activation functions were the same as those for the N1 model. Analysis of the distribution of residuals obtained with L1, L2, N1, and N2 models (Table 5.1) clearly shows the superiority of the nonlinear methods over the linear

Table 5.1 Distribution of residuals (absolute values), differences between the experimental and calculated *Tetrahymena* toxicity data computed from the PLS (L1, L2) and ANN (N1, N2) models

Range	L1	N1	L2	N2	Cronin et al. [23]
<0.25	104(22)[a]	117(22)	117(23)	137(28)	(17)
0.25 to 0.50[b]	82(16)	67(14)	80(16)	79(14)	(23)
0.50 to 0.75[b]	37(5)	40(8)	40(8)	27(7)	(6)
0.75 to 1.00[b]	15(3)	16(5)	7(2)	3	(2)
1.00 to 1.25[b]	7(3)	3	3(1)	2(1)	(1)
1.25 to 1.50[b]	1	1	1	2	(1)
1.50 to 1.75[b]	2(1)	4	2	0	
1.75 to 2.00[b]	0	1(1)	0	0	
≥2.00	2	1	0	0	

[a]Including number of residuals from the external testing set.
[b]Excluded value.

ones to find complex structure-toxicity relationships. It is noteworthy that the results obtained with the nonlinear methods were also better than those obtained by Cronin et al. [23].

A set of 153 phenols tested on *T. pyriformis* was used by Yao and coworkers [24] for comparing the performances of multiple regression analysis (MRA), radial basis function artificial neural network (RBFANN) [25], and support vector machine (SVM) [26]. The studied molecules were described by their log Kow, pKa, frontier orbital energies (E_{homo}, E_{lumo}), and hydrogen bond donor/acceptor counts (N_{hdon}). The leave-one-out (LOO) cross-validation r (and RMS) obtained with MRA, RBFANN, and SVM were 0.911 (0.352), 0.945 (0.260), and 0.947 (0.257), respectively. In this study, the RBFANN, and SVM models presented similar performances, but both outperformed the regression model.

Six nonlinear methods were used by Ren [27] to capture the intrinsic nonlinearity in a data set of 203 aromatic chemicals containing a nitro or a cyano group and for which log (1/IGC$_{50}$) values on *T. pyriformis* were available. These methods included generalized additive model (GAM) [28], least absolute shrinkage and selection operator version 2 (LASSO2) [29], locally weighted regression scatter plot smoothing (LOESS) [30], multivariate adaptive regression splines (MARS) [31], projection pursuit regression (PPR) [32], and a three-layer ANN. The original data set was randomly split into a training and an external testing set of 162 and 41 chemicals, respectively. All the chemicals were described by their log Kow and $A_{\max}$ (the maximum acceptor superdelocalizability). Because 4-nitroaniline was found to be an outlier with the six nonlinear methods, it was removed from the training set for designing the final models. Analysis of their statistics, summarized in Table 5.2, shows that they broadly present the same predictive performances.

Recently, the same training and testing sets, as well as same molecular descriptors, were used by Panaye and coworkers [33] to estimate whether general regression artificial neural network (GRANN), RBFANN, and SVM provided better simulation results. The best results were obtained with GRANN yielding an r^2

Table 5.2 Summary of the results of model fitting, prediction (pred), and tenfold cross-validation

Nonlinear method	r^2	r^2_{pred}	s	s_{pred}	F	F_{pred}	r^2_{cv}	PMSE[a]	sPMSE[b]
GAM	0.75	0.61	0.32	0.46	464.66	60.07	0.70	0.16	0.05
LASSO2	0.73	0.69	0.33	0.42	430.96	87.96	0.70	0.16	0.06
LOESS	0.75	0.73	0.32	0.46	483.81	105.91	0.70	0.17	0.05
MARS	0.73	0.74	0.33	0.42	426.92	112.71	0.69	0.17	0.06
PPR	0.80	0.71	0.29	0.43	647.95	93.49	0.71	0.17	0.06
ANN[c]	0.73	0.73	0.33	0.45	438.49	107.21	0.70	0.16	0.05

[a]Average predicted mean square error calculated over the ten CV subsets.
[b]Standard error of PMSE.
[c]2/3/1 ANN.

(RMS) of 0.79 (0.35) and 0.72 (0.42) for the training and testing set, respectively. The authors concluded that their methods favorably competed with those used by Ren [27].

A set of 225 phenols tested on *T. pyriformis* was used by Novic and Vracko [34] for deriving counterpropagation artificial neural network (CPANN) models [35]. Chemicals were described by means of 157 descriptors and the QSAR models were tested on an external testing set of 40 chemicals. The r^2 (and RMS) values for the training and testing set were 0.983 (0.110) and 0.571 (0.462), respectively.

Niculescu, Kaiser, and Schultz. [36] designed a probabilistic artificial neural network (PANN) [37] model for estimating the protozoan toxicity of a much larger set of chemicals. Their original data set included toxicity values for 825 chemicals, encompassing various MOA. A subset containing 750 randomly selected compounds was used to produce five distinct training and testing sets of 600 and 150 chemicals, respectively. The remaining 75 chemicals were employed as an external testing set. All the chemicals were described by means of 32 functional group descriptors, MW, and the number of individual occurrences of C, H, Br, Cl, F, I, N, O, and S atoms. In a first step, five distinct PANN models were derived. Their analysis suggested the possibility of improving the quality of the PNN-based predictions by using appropriate linear corrections. Therefore, four equations were used for deriving four definitive models. The predictive performance of these four models was critically analyzed in Devillers [5].

The same ANN modeling strategy was applied by Kaiser, Niculescu, and Schultz. [38] on a larger set of 1,084 organic compounds tested on *T. pyriformis*.

2.3. QSAR Models on Arthropoda

2.3.1. QSAR Models on Crustacea

The waterflea, *Daphnia magna*, plays a key ecological role in surface waters, being an efficient herbivore and aiding in the transfer of energy from autotrophs to the top of the food web. Consequently, this small freshwater organism, easy to rear, is used worldwide for estimating the aquatic toxicity of chemicals.

Kaiser and Niculescu [39] designed PANN models for estimating the acute toxicity of chemicals to *D. magna*. The main interest of their study is the large data set of 776 chemicals used to derive and test the performances of their models. Indeed, all the published QSAR models performed against this important freshwater crustacean have been elaborated from considerably more reduced training sets. Their publication is also interesting for the comparison exercise performed with ECOSAR [40], which is a compilation of functional group QSARs. Kaiser and coworkers [41] divulged that most of the linear regression equations proposed in ECOSAR were based on fewer than five compounds, in fact, many of them were derived from a single chemical. It is obvious that these particular QSAR models are not statistically valid and their use must be avoided even if they are recommended by the U.S. EPA. It is also interesting to stress that these authors [41] indicated that the classification scheme of chemicals to select a QSAR model in ECOSAR was unclear to the user, inconsistent, or faulty. Some representative results of this comparison exercise are given in Table 5.3. The ANN models designed by Kaiser and Niculescu [39] have been critically analyzed elsewhere [5].

The toxicity of chemicals to the aquatic organisms depends on their life stage, size, and the experimental conditions in which the tests are performed (e.g., temperature, pH, hardness, time of exposure). The ability of PLS analysis and ANN to integrate these variables within a classical QSAR modeling process was investigated by Devillers [42] for the acute toxicity prediction of pesticides to the freshwater amphipod *Gammarus fasciatus*. A database of 130 LC50 (lethal concentration 50%) values for 51 pesticides presenting various structures and mechanisms of toxicity was randomly split into a training set of 112 toxicity results and a testing set of 18 LC50 values. Experimental variables included life stage (mature or immature),

Table 5.3 Observed (Obs.) and calculated (PANN and ECOSAR) *Daphnia magna* toxicity data (log 1/***C***, ***C*** in mmol/l) for various chemicals [*39*]

Chemical	Obs.	PANN[a]	ECOSAR
Aldicarb	3.40	1.45	–0.48
Bromacil	0.34	0.72	–0.28
Endrin	3.64	3.76	3.05
Fonophos	4.96	3.98	1.94
Mancozeb	2.62	2.76	–1.16
Acenaphthene	0.58	2.35	2.06
Anthracene	2.37	2.28	2.24
2,4-Dinitrotoluene	0.72	1.45	1.50
3,5-Dinitrotoluene	0.61	1.45	1.50
2,3,4-Trichloroaniline	2.43	2.35	2.44
1,4-Dichloro-2-nitrobenzene	1.24	1.60	1.10
4-Hydroxyacetanilide	1.22	1.17	0.56
1,4-Dibromobenzene	2.36	2.30	1.71
3-Aminophenol	2.00	1.76	1.69
Diethylamine	0.12	–0.21	1.00
N-Nitrosodiphenylamine	1.41	1.91	2.37

[a]Average value obtained from four PANN models [*39*].

temperature (°C), pH, water hardness (mg/l as $CaCO_3$), and time of exposure in hours. Pesticides were described by means of autocorrelation descriptors [13].

The five experimental variables and eight autocorrelation descriptors (H_0–H_5, HBA_0, H_2, HBD_0) yielded the design of a PLS model at four significant components ($r^2 = 0.581$). A plot of the observed versus calculated toxicity values obtained with this PLS model for both the training and external testing sets is shown in Fig. 5.3. Different assays were performed with the 13 specified descriptors as inputs in a three-layer feedforward ANN. The best results were obtained with a 13/3/1 configuration.

Feature selection, by forward and backward procedures and also with a genetic algorithm [43], was used for reducing the number of autocorrelation descriptors. Among the eight autocorrelation descriptors, HBA_0, HBD_0, H_2, and H_5 were widely proposed to elimination. Their removal yielded a 9/3/1 ANN. The best model was obtained after 245 cycles (100 with BPA and 145 with CGDA). The learning rate and momentum for the BPA were equal to 0.01 and 0.3, respectively. All the synaptic functions for the three layers were linear; and the activation functions were linear, hyperbolic, and logistic from the input to the output layer. The learning and testing errors were equal to 0.0695 and 0.0676, respectively. A plot of the observed versus calculated toxicity values obtained with this ANN model for both the training and external testing sets is displayed in Fig. 5.4. Comparison of Figs. 5.3 and 5.4 clearly shows that, conversely to PLS analysis, a three-layer feedforward ANN is particularly suited to integrate any kind of variables in a QSAR modeling process.

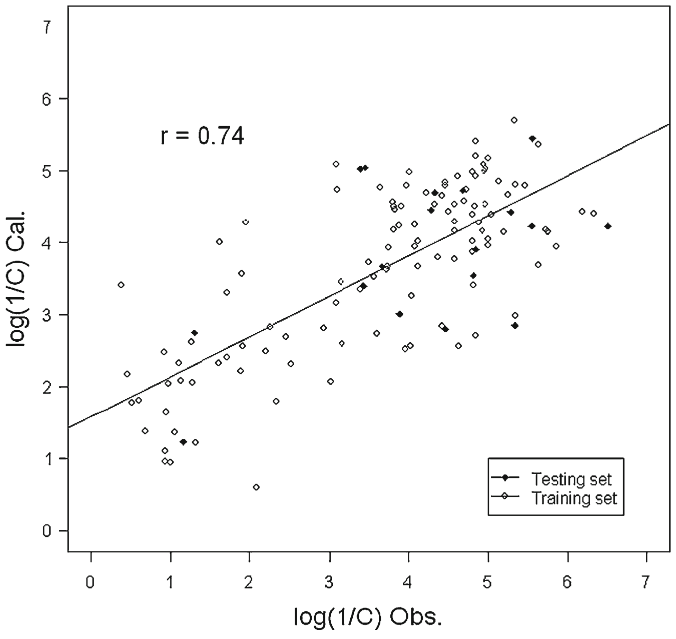


Fig. 5.3 Observed versus calculated Gammarid toxicity values (PLS model)

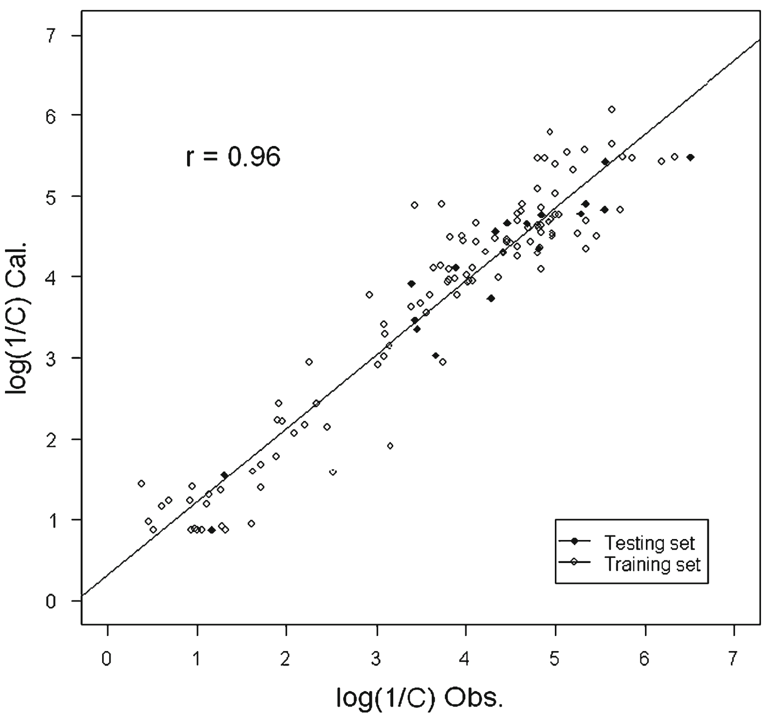


Fig. 5.4 Observed versus calculated Gammarid toxicity values (ANN model)

2.3.2. QSAR Models on Insects

Landrum and coworkers [44] proposed different QSAR models for predicting the toxicity of organophosphorus insecticides against midge larvae (*Chironomus riparius*, Diptera, Chironomidae), which are sediment-dwelling organisms serving as prey for many aquatic species and, hence, represent an extremely important part of food chains in freshwater ecosystems. These authors derived numerous regression equations for organophosphorus insecticides tested with or without sediment and according different temperature and pH conditions.

From the same toxicity data set, a three-layer feedforward ANN trained by the BPA was used for deriving a powerful QSAR model that allowed estimatinge the toxicity of organophosphorus compounds measured against *C. riparius* under varying temperature (11, 18, and 25°C) and pH (6, 7, and 8) conditions and with or without sediment [45]. It is noteworthy that the chemicals were described by means of an autocorrelation vector encoding lipophilicity [13]. The obtained results were of better quality that those generated by Landrum et al. [44].

2.4. QSAR Models on Fish

Many research groups have developed ANN QSAR models for predicting the acute toxicity of chemicals to the fathead minnow (*Pimephales promelas*) from the invaluable U.S. EPA database (Duluth, Minnesota), which includes 96-h LC50 values for a large collection of chemicals, structurally heterogeneous and showing different MOA (www.epa.gov/nheerl/dsstox).

Kaiser, Niculescu, and Schüürmann [46] have designed a three-layer feedforward ANN QSAR model (n = 419) for estimating the acute toxicity of chemicals to *P. promelas* from this observed against *V. fischeri* and from 50 different molecular descriptors (log MW, log Kow, 31 functional groups, 17 formula descriptors). This very original model, combining a biological descriptor and various molecular descriptors, was designed to predict the toxicity of both organic and inorganic chemicals. Unfortunately, the prediction performances of the selected 51/7/1 BNN was not truly estimated, since only a leave-20%-out procedure was used instead of an external testing set. However, some restrictions can be made. For example, we can assume that their model requiring Microtox toxicity data will fail for chemicals presenting a specific behavior against *V. fischeri* (e.g., some pesticides). In addition, some of the selected descriptors present a very weak frequency (e.g., Zn = 2/419); and it is obvious that their use to calculate toxicity will often yield biased results. Kaiser, Niculescu, and McKinnon [47] also estimated the interest of a PANN on the same data set.

Eldred and coworkers [48] tried to improve this model. Among the 419 molecules selected by Kaiser et al. [46], 375 met the requirements of the ADAPT software. From this 375-member data set, 88 chemicals were randomly selected to constitute a cross-validation set and an external testing set. The remaining 287 compounds were used as a training set. All the chemicals were described by means of 242 molecular descriptors. After the removal of the featureless and redundant descriptors and those pairwise correlated with an $r > 0.90$, a reduced pool of 123 molecular descriptors was obtained. This descriptor space was thoroughly explored by multiple searches using simulated annealing with multiple linear regression analysis. A type I regression model including eight descriptors was selected. These eight descriptors were then submitted to a BNN yielding the design of a type II model (8/6/1 architecture). Last, using the characteristics of this type II model, a genetic algorithm was employed for optimizing the selection of the descriptors among the reduced pool, yielding a type III model (8/6/1) with RMS values of 0.71, 0.77, and 0.74 for the training set, cross-validation set, and external testing set, respectively. Despite this rational modeling approach, it is noteworthy that the simulation performances of the final model are not good and simple regression models designed for molecules presenting specific modes of action generally provide better results. The critical analysis of these models has been made elsewhere [5, 49].

Gaining experience from their preliminary modeling investigations [46, 47, 50], Kaiser and Niculescu [51] proposed another PANN QSAR model designed from a larger database of 865 compounds tested for their acute toxicity on the fathead minnow. Chemicals were described by their log MW, presence/absence of 33 types of functional groups (e.g., $-NH_2$) and their number of C, H, As, Br, Cl, F, Fe, Hg, I, K, Mn, N, Na, O, P, S, Se, Si, Sn, and Zn atoms. The 20% cross-validated r^2_{cv} equaled 0.766. On the basis of an external testing set of 130 chemicals, this model compared favorably (Pearson correlation coefficient $r_p = 0.552$) with five others ($r_p = 0.393$ to 0.528) [49], including ECOSAR [40] and the ANN model derived by Eldred and coworkers [48].

It is noteworthy that recently, Niculescu et al. [52] proposed a rather similar PANN QSAR model derived from a training set and an external testing of 800 and 86 LC50 values, respectively.

Different ANN paradigms and modeling strategies were experienced on fathead minnow data sets of various sizes. A QSAR model of rather limited interest was designed by Espinosa, Arenas, and Giralt [53] from a reduced set of 69 benzene derivatives by using a fuzzy-ARTMAP ANN [54]. Mazzatorta et al. [55] used LC50 data from the U.S. EPA database for illustrating the interest of a hybrid system combining a multilayer perceptron and a fuzzy ANN for toxicity modeling. They used a data set of 562 organic chemicals randomly partitioned into a training set and a testing set of 392 and 170 chemicals, respectively. The main goal of their study was to explain how this type of ANN had to be tuned in practice for deriving structure-toxicity models. Mazzatorta et al. [56] and Novic and Vracko [34] employed a CPANN for modeling 568 LC50 values on *P. promelas*. The database was split into a training set and a testing set of 282 and 286 compounds, respectively. Chemicals were described by means of a huge number of descriptors encoding their topology and physicochemical properties. The r^2 values of the training and testing sets equaled 0.953 and 0.527, respectively. Better statistics were obtained after the removal of outliers in both sets. A CPANN and a rather similar data set of LC50 values were used by Vracko et al. [57] for illustrating the five OECD principles for validation of (Q)SAR models used for regulatory purposes. In this study, some interesting graphical displays are proposed to better interpret the modeling results. Last, recently, a new type of QSTR model has been proposed by Devillers [58] for the common situation in which the biological activity of molecules depends mainly on their log Kow. Briefly, in a first step, a classical regression equation with log Kow is derived. The residuals obtained with this simple linear equation are then modeled from a three-layer feedforward ANN including different molecular descriptors as input neurons. Finally, results produced by both the linear and non-linear models are considered for calculating the toxicity values, which are then compared to the initial toxicity data. A heterogeneous set of 569 chemicals with 96-h LC50s measured to the fathead minnow, and randomly divided into a training set of 484 chemicals and a testing set of 85 chemicals, was used as illustrative example. The autocorrelation method was employed in the second phase of the modeling process for describing the topology and physicochemical properties of the molecules. The linear model (Eq. 2) showed a RMS of 0.906 and 0.781 for the training and external testing set, respectively.

$$\log(1/\mathrm{LC50}) = 0.667 \log \mathrm{Kow} - 0.656$$
$$n = 484,\ r^2 = 0.557,\ s = 0.908,\ F = 605 \tag{2}$$

By using a 9/5/1 ANN (243 cycles) on the residuals calculated from Eq. 2, a linear/nonlinear hybrid model was designed, yielding a RMS of 0.818 and 0.664 for the training and testing set, respectively. A 22/11/1 ANN (250 cycles) was allowed to improve the performance of the final model. Indeed, in that case, RMS values of 0.749 and 0.639 were obtained for the training and testing set, respectively. The advantages and limitations of this modeling strategy, based on the concept of obvious and hidden models [59–62], have been deeply analyzed by Devillers [58].

The fathead minnow (*P. promelas*) is not the unique fish species for which large collections of valuable acute toxicity data are available. Consequently, noncongeneric ANN QSAR models have been also derived on other freshwater fish species. The rainbow trout (*Oncorhynchus mykiss*) is a stenotherm freshwater fish highly sensitive to the variations of temperature. Devillers and Flatin [63], proposed a three-layer feedforward ANN QSAR model trained by a classical BPA for predicting the acute toxicity of pesticides to *O. mykiss*. The inputs of the ANN included autocorrelation descriptors for characterizing the structures of the pesticides as well as the following experimental variables: water temperature, pH, hardness, time of exposure, and weight of the fish. The model is able to simulate the influence of temperature on the toxicity of pesticides, the effect of soft or hard water on the toxicity, and so on. The same modeling strategy was successfully applied to the prediction of the acute toxicity of pesticides against the bluegill (*Lepomis macrochirus*) [64].

3. Terrestrial Ecosystem

The number of linear and nonlinear QSAR models derived on terrestrial invertebrates and vertebrates is very scarce. This is mainly due to a lack of data except for a reduced number of species such as honeybees.

One hundred pesticide toxicity values on honeybees (LC50 or LD50 in μmol/bee) obtained under well-defined and -controlled laboratory conditions were used by Devillers et al. [65, 66] for deriving a QSAR model based on a three-layer feedforward ANN trained by a BPA. The different families of pesticides were described by means of autocorrelation descriptors [13]. Different training and testing sets of, respectively, 89 and 11 chemicals were experienced. The best results were obtained with a 12/3/1 ANN ($\eta = 0.5$, $\alpha = 0.9$, 2,492 cycles). The RMSR values for the training and testing

Table 5.4 Distribution of residuals (absolute values) obtained with the bee ANN model

Range	Training set	Testing set
<0.3	48	5
0.3 to 0.6[a]	27	6
0.6 to 0.9[a]	10	0
0.9 to 1.2[a]	4	0
≥1.2	0	0

[a]Excluded value.

sets were 0.430 and 0.386, respectively. Only four pesticides were not correctly predicted by the model (Table 5.4). Attempts were also made to derive a model from a PLS regression analysis. Unfortunately, this did not yield satisfying results. Indeed, in all cases, very large outliers were obtained for numerous pesticides.

4. Conclusion

Analysis of the ecotoxicology modeling literature shows that most of the SAR and QSAR studies based on artificial neural networks yield results that are at least equivalent and frequently superior to those obtained with classical linear methods, such as discriminant analysis, regression analysis, and PLS analysis. This is particularly true for data sets including noncongeneric chemical structures. This is not surprising because ANNs are suited to draw conclusions when presented with complex, noisy, and partial information. Indeed, it is noteworthy that the relationships between the structure of the molecules and their ecotoxicity are not straightforward. In addition, even if standardized protocols are used in ecotoxicology, most of the obtained data suffer from a inherent degree of variability due to the experimental conditions, the biological material, and so on. Because it is well known that some degree of fuzziness in the data increases the modeling performances of an ANN, this explains that these nonlinear tools outperform the classical linear methods commonly used in SAR and QSAR modeling. However, to be efficient, they require large data sets. Consequently, there is a rather limited number of endpoints and species for which ANN SAR/QSAR models exist. The situation is particularly obvious with the terrestrial species.

In environmental toxicology modeling, all the linear QSAR models are derived from toxicity results obtained according to very specific experimental conditions. However, the correct hazard assessment of xenobiotics requires simulating the ecotoxicological behavior of chemicals by accounting for various abiotic factors, such as temperature. These parameters cannot be introduced as variables in the linear models, while the ANNs do not suffer from this limitation and their introduction in the modeling process allows us to derive very flexible and more realistic QSAR models.

5. Notes

1. Artificial neural networks in toxicology modeling are neither a panacea nor a Pandora's box. Their value, or lack thereof, depends on the problem at hand and the availability of biological data. Chemicals acting according to a same mechanism of action can be easily modeled by a simple or multiple regression analysis or PLS analysis if, obviously, they have been correctly encoded from molecular descriptors. Conversely, if the data set includes chemicals acting with different

MOA, on the basis of the same set of descriptors, a correctly tuned supervised ANN always outperforms a regression or PLS analysis. However, it is worth noting that, to be efficient, a supervised ANN needs a large training set.

2. In a three-layer perceptron, the number of output neurons depends on the modeled activity. Therefore, only one output neuron corresponding to the LC50s or EC50s is used in aquatic and terrestrial toxicology modeling. A feature selection of the molecular descriptors allows the optimization of the number of input neurons. The number of neurons in the hidden layer is not linked to the modeled activity or the molecular descriptors and has to be determined. However, according to the Ockham's Razor principle (also called the *principle of economy*), it must be as reduced as possible, because too many free parameters in the ANN allows the network to fit the training data arbitrarily closely (overfitting) but does not lead to an optimal generalization.
3. A crucial step in the training of a three-layer perceptron is the decision to stop the iterative learning at the right time. Prolonged training (overtraining) to reduce an error beyond a realistic value deteriorates the generalization performances of the ANN model. Different techniques exist for avoiding overtraining, such as the use of a cross-validation set to monitor the learning phase. In all cases, the number of epochs must be as reduced as possible. In practice, the judicious combination of different algorithms (e.g., a back-propagation algorithm and a conjugate gradient descent algorithm) during the learning phase often allows one to reduce the number of epochs.
4. In environmental toxicology, the QSAR models are commonly derived from a training (learning) set, then their generalization performances are assessed from an external testing set. During the testing phase, unknown patterns are presented to the ANN to check whether the model is able to predict their actual outputs. It is well admitted that the intrapolation performances of a three-layer perceptron are better than those of extrapolation. The use of an in-sample testing set (ISTS) and an out-of-sample testing set (OSTS), respectively, allows one to estimate the intrapolation and extrapolation performances of the ANN model. Indeed, the ISTS contains patterns similar to those belonging to the training set of the ANN model and therefore allows one to assess its interpolation performances. Conversely, the OSTS includes patterns presenting various degrees of difference with those of the training set and can be used for estimating the extrapolation performances of the ANN model [9, 11]. In practice, chemicals with experimental toxicity data of lesser quality also have to be included in the OSTS.

In all cases, it is best to start with a large data set, which is then divided into training and testing sets. The selection of these sets is problem dependent. It is generally performed by means of linear and nonlinear multivariate methods to secure the representativeness of the biological activities and chemical structures in both sets.

It is worth noting that sometimes, a leave-one-out cross-validation procedure or a bootstrap is used to try to estimate the performances of the QSAR model. However, more and more in environmental toxicology modeling, these statistical techniques are considered like a pis aller.

5. Ensemble network models have proven their interest in numerous domains to increase the accuracy of the simulation results. Surprisingly, in environmental toxicology modeling, their usefulness seems to be more questionable. For example, unsuccessful results were obtained in the design of toxicity ANN models on *Vibrio fischeri* [12] and *Gammarus fasciatus* [42].
6. ANNs are now widely used in ecology modeling due to their capacity to integrate various interactional abiotic and biotic factors in the modeling processes. It is expected that this will be also the case in environmental toxicology modeling for deriving more realistic QSAR models. Promising results still have been obtained with the design of powerful ANN QSAR models from the combined use of molecular descriptors and laboratory or environmental variables [42, *45*, *63*, *64*].

References

1. Russon CL, Breton RL, Walker JD, Bradbury SP (2003) An overview of the use of quantitative structure-activity relationships for ranking and prioritizing large chemical inventories for environmental risk assessments. Environ Toxicol Chem 22:1810–1821.
2. Kaiser KLE (2003) Neural networks for effect prediction in environmental and health issues using large datasets. QSAR Comb Sci 22:185–190.
3. Geladi P, Tosato ML (1990) Multivariate latent variable projection methods: SIMCA and PLS. In: Karcher W, Devillers J (eds) Practical applications of quantitative structure-activity relationships (QSAR) in Environmental Chemistry and Toxicology. Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 171–179.
4. Devillers J (1996) Neural networks in QSAR and drug design. Academic Press, London, p. 284.
5. Devillers J (2001) QSAR modeling of large heterogeneous sets of molecules. SAR QSAR Environ Res 12:515–528.
6. Kaiser KLE (1998) Correlations of *Vibrio fischeri* bacteria test data with bioassay data for other organisms. Environ Health Pespect 106:583–591.
7. Kaiser KLE, Devillers, J. (1994) Ecotoxicity of Chemicals to *Photobacterium phosphoreum*. Gordon and Breach Science Publishers, Reading, UK, p. 879.
8. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536.
9. Devillers J (1996) Strengths and weaknesses of the backpropagation neural network in QSAR and QSPR studies. In: Devillers J (ed) Neural networks in QSAR and drug design. Academic Press, London, pp. 1–46.
10. Devillers J, Bintein S, Karcher W (1995) QSAR for predicting luminescent bacteria toxicity. In: Sanz F, Giraldo J, Manaut F, (eds) QSAR and molecular modelling: concepts, computational tools and biological applications. J.R. Prous, Barcelona, pp. 190–192.
11. Devillers J, Bintein S, Domine D, Karcher W (1995) A general QSAR model for predicting the toxicity of organic chemicals to luminescent bacteria (Microtox® test). SAR QSAR Environ Res 4:29–38.
12. Devillers J, Domine D (1999) A noncongeneric model for predicting toxicity of organic molecules to *Vibrio fischeri*. SAR QSAR Environ Res 10:61–70.
13. Devillers J (1999) Autocorrelation descriptors for modeling (eco)toxicological endpoints. In: Devillers J, Balaban AT (eds) Topological indices and related descriptors in QSAR and QSPR. Gordon and Breach, The Netherlands, pp. 595–612.
14. Domine D, Devillers J, Wienke D, Buydens L (1996) Test series selection from nonlinear neural mapping. Quant Struct Act Relat 15:395–402.

15. Baker L, Wesley, SK, Schultz TW (1988) Quantitative structure-activity relationships for alkylated and/or halogenated phenols eliciting the polar narcosis mechanism of toxic action. In: Turner JE, England MW, Schultz TW, Kwaak NJ (eds) QSAR 88: proceedings of the third international workshop on quantitative structure-activity relationships in environmental toxicology CONF-880520, pp. 165–168.
16. Kelley CT (2003) Solving nonlinear equations with Newton's method (fundamentals of algorithms). Society for Industrial and Applied Mathematics, Philadelphia, p. 116.
17. Xu L, Ball JW, Dixon SL, Jurs PC (1994) Quantitative structure-activity relationships for toxicity of phenols using regression analysis and computational neural networks. Environ Toxicol Chem 13:841–851.
18. Serra JR, Jurs PC, Kaiser KLE (2001) Linear regression and computational neural network prediction of *Tetrahymena* acute toxicity of aromatic compounds from molecular structure. Chem Res Toxicol 14:1535–1545.
19. Burden FR, Winkler DA (2000) A Quantitative structure-activity relationships model for the acute toxicity of substituted benzenes to *Tetrahymena pyriformis* using Bayesian-regularized neural networks. Chem Res Toxicol 13:436–440.
20. Neal RM (1996) Bayesian learning for neural networks. Lecture Notes in Statistics, 118, Springer, Berlin, p. 204.
21. Winkler D, Burden F (2003) Toxicity modelling using Bayesian neural nets and automatic relevance determination. In: Ford M, Livingstone D, Dearden J, van de Waterbeemd H (eds) EuroQSAR 2002: designing drugs and crop protectants: processes, problems and solutions. Blackwell Publishing, Malden, UK, pp. 251–254.
22. Devillers J (2004) Linear versus nonlinear QSAR modeling of the toxicity of phenol derivatives to *Tetrahymena pyriformis*. SAR QSAR Environ Res 15:237–249.
23. Cronin MTD, Aptula, AO, Duffy JC, Netzeva TI, Rowe PH, Valkova IV, Schultz TW (2002) Comparative assessment of methods to develop QSARs for the prediction of the toxicity of phenols to *Tetrahymena pyriformis*. Chemosphere 49:1201–1221.
24. Yao XJ, Panaye A, Doucet JP, Zhang RS, Chen HF, Liu MC, Hu ZD, Fan BT (2004) Comparative study of QSAR/QSPR correlations using support vector machines, radial basis function neural networks, and multiple linear regression. J Chem Inf Comput Sci. 44:1257–1266.
25. Bengio Y (1996) Neural networks for speech and sequence recognition. International Thomson Computer Press, London, Chapter 6, p. 167.
26. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge, UK, p. 189.
27. Ren S (2003) Modeling the toxicity of aromatic compounds to *Tetrahymena pyriformis*: The response surface methodology with nonlinear methods. J Chem Inf Comput Sci 43:1679–1687.
28. Hastie TJ, Tibshirani RJ (1990) Generalized additive models. Chapman and Hall, New York..
29. Osborne MR, Presnell B, Turlach BA (2000) On the LASSO and its dual. J Comput Graph Stat 9:319–337.
30. Cleveland WS (1979) Robust locally weighted regression and smoothing scatterplots. J Am Stat Assoc 46:175–185.
31. Friedman JH (1991) Multivariate additive regression splines. Annals Stat 19:1–141.
32. Friedman JH, Stuetzle W (1981) Projection pursuit regression. J Am Stat Assoc 76:817–823.
33. Panaye A, Fan BT, Doucet JP, Yao XJ, Zhang RS, Liu MC, Hu ZD (2006) Quantitative structure-toxicity relationships (QSTRs): a comparative study of various non-linear methods. General regression neural network, radial basis function neural network and support vector machine in predicting toxicity of nitro- and cyano- aromatics to *Tetrahymena pyriformis*. SAR QSAR Environ Res 17:75–91.
34. Novic M, Vracko, M. (2003) Artificial neural networks in molecular-structures-property studies. In: Leardi R (ed.) Nature-inspired methods in chemometrics: genetic algorithms and artificial neural networks. Elsevier, Amsterdam, pp. 231–256.

35. Zupan J, Gasteiger J (1993) Neural networks for chemists. VCH, Weinheim, p. 305.
36. Niculescu SP, Kaiser KLE, Schultz TW (2000) Modeling the toxicity of chemicals to *Tetrahymena pyriformis* using molecular fragment descriptors and probabilistic neural networks. Arch Environ Contam Toxicol 39:289–298.
37. Wasserman PD (1993) Advanced methods in neural computing. Van Nostrand Reinhold, New York, p. 255.
38. Kaiser KLE, Niculescu SP, Schultz TW (2002) Probabilistic neural network modeling for the toxicity of chemicals to *Tetrahymena pyriformis* with molecular fragment descriptors. SAR QSAR Environ Res 13:57–67.
39. Kaiser KLE, Niculescu, S P (2001) Modeling acute toxicity of chemicals to *Daphnia magna*: a probabilistic neural network approach. Environ Toxicol Chem 20:420–431.
40. ECOSAR, version 0.99f, January 2000.
41. Kaiser KLE, Dearden JC, Klein W, Schultz,TW (1999) A note of caution to users of ECOSAR. Water Qual Res J Canada 34:179–182.
42. Devillers J (2003) A QSAR model for predicting the acute toxicity of pesticides to gammarids. In: Leardi R (ed) Nature-inspired methods in chemometrics: genetic algorithms and artificial neural networks. Elsevier, Amsterdam, pp. 323–339.
43. Devillers J (1996) Genetic algorithms in molecular modeling. Academic Press, London, p. 327.
44. Landrum PF, Fisher SW, Hwang H, Hickey J (1999) Hazard evaluation of ten organophosphorus insecticides against the midge, *Chironomus riparius* via QSAR. SAR QSAR Environ Res 10:423–450.
45. Devillers J (2000) Prediction of toxicity of organophosphorus insecticides against the midge, *Chironomus riparius*, via a QSAR neural network model integrating environmental variables. Toxicol Methods 10:69–79.
46. Kaiser KLE, Niculescu SP, Schüürmann G (1997) Feed forward backpropagation neural networks and their use in predicting the acute toxicity of chemicals to the fathead minnow. Water Qual Res J Canada 32:637–657.
47. Kaiser KLE, Niculescu SP, McKinnon MB (1997) On simple linear regression, multiple linear regression, and elementary probabilistic neural network with Gaussian kernel's performance in modeling toxicity values to fathead minnow based on Microtox data, octanol/water partition coefficient, and various structural descriptors for a 419-compound dataset. In: Chen F, Schüürmann G (eds) Proceedings of the 7th international workshop on QSAR in environmental sciences. SETAC Press, Pensacola, FL, pp. 285–297.
48. Eldred DV, Weikel CL, Jurs PC, Kaiser KLE (1999) Prediction of fathead minnow acute toxicity of organic compounds from molecular structure. Chem Res Toxicol 12:670–678.
49. 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.
50. Niculescu SP, Kaiser KLE, Schüürmann G (1998) Influence of data preprocessing and kernel selection on probabilistic neural network modeling of the acute toxicity of chemicals to the fathead minnow and *Vibrio fischeri* bacteria. Water Qual Res J. Canada 33:153–165.
51. Kaiser KLE, Niculescu SP (1999) Using probabilistic neural networks to model the toxicity of chemicals to the fathead minnow (*Pimephales promelas*): a study based on 865 compounds. Chemosphere 38:3237–3245.
52. Niculescu SP, Atkinson A, Hammond G, Lewis, M (2004) Using fragment chemistry data mining and probabilistic neural networks in screening chemicals for acute toxicity to the fathead minnow. SAR QSAR Environ. Res 15:293–309.
53. Espinosa G, Arenas A, Giralt F (2002) An integrated SOM-fuzzy ARTMAP neural system for the evaluation of toxicity. J Chem Inf Comput Sci 42:343–359.
54. Wienke D, Domine D, Buydens L, Devillers J (1996) Adaptive resonance theory based neural networks explored for pattern recognition analysis of QSAR data. In: Devillers J (ed) Neural networks in QSAR and drug design. Academic Press, London, pp. 119–156.
55. Mazzatorta P, Benfenati E, Neagu CD, Gini G (2003) Tuning neural and fuzzy-neural networks for toxicity modeling. J Chem Inf Comput Sci 43:513–518.

56. Mazzatorta P, Vracko M, Jezierska A, Benfenati E (2003) Modeling toxicity by using supervised Kohonen neural networks. J Chem Inf Comput Sci 43:485–492.
57. Vracko M, Bandelj V, Barbieri P, Benfenati E, Chaudhry Q, Cronin M, Devillers J, Gallegos A, Gini G, Gramatica P, Helma C, Mazzatorta P, Neagu D, Netzeva T, Pavan M, Patlevicz G, Randic M, Tsakovska I, Worth A (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.
58. Devillers J (2005) A new strategy for using supervised artificial neural networks in QSAR. SAR QSAR Environ Res 16:433–442.
59. Devillers J, Chessel D (1995) Can the enucleated rabbit eye test be a suitable alternative for the *in vivo* eye test? A chemometrical response. Toxicol Model 1:21–34.
60. Klopman G (1998) The MultiCASE program II. Baseline activity identification algorithm (BAIA). J Chem Inf Comput Sci 38:78–81.
61. Klopman G, Saiakhov R, Rosenkranz HS, Hermens JLM (1999) Multiple computer-automated structure evaluation program study of aquatic toxicity 1: guppy. Environ Toxicol Chem 18:2497–2505.
62. Klopman G, Saiakhov R, Rosenkranz HS (2000) Multiple computer-automated structure evaluation study of aquatic toxicity 2: fathead minnow. Environ Toxicol Chem 19:441–447.
63. Devillers J, Flatin J (2000) A general QSAR model for predicting the acute toxicity of pesticides to *Oncorhynchus mykiss.* SAR QSAR Environ Res 11:25–43.
64. Devillers J (2001) A general QSAR model for predicting the acute toxicity of pesticides to *Lepomis macrochirus*. SAR QSAR Environ Res 11:397–417.
65. Devillers J, Pham-Delègue MH, Decourtye A, Budzinski H, Cluzeau S, Maurin G (2002) Structure-toxicity modeling of pesticides to honey bees. SAR QSAR Environ Res 13:641–648.
66. Devillers J, Pham-Delègue MH, Decourtye A, Budzinski H, Cluzeau S, Maurin G (2003) Modeling the acute toxicity of pesticides to *Apis mellifera*. Bull Insect 56:103–109.

Chapter 6
Neural Networks in Analytical Chemistry

Mehdi Jalali-Heravi

Analytical chemistry is an art. It requires a high level of skill that can be developed by experience and training.

Abstract This chapter covers a part of the spectrum of neural-network uses in analytical chemistry. Different architectures of neural networks are described briefly. The chapter focuses on the development of three-layer artificial neural network for modeling the anti-HIV activity of the HETP derivatives and activity parameters (pIC_{50}) of heparanase inhibitors. The use of a genetic algorithm-kernel partial least squares algorithm combined with an artificial neural network (GA-KPLS-ANN) is described for predicting the activities of a series of aromatic sulfonamides. The retention behavior of terpenes and volatile organic compounds and predicting the response surface of different detection systems are presented as typical applications of ANNs in chromatographic area. The use of ANNs is explored in electrophoresis with emphasizes on its application on peptide mapping. Simulation of the electropherogram of glucagons and horse cytochrome C is described as peptide models. This chapter also focuses on discussing the role of ANNs in the simulation of mass and ^{13}C-NMR spectra for noncyclic alkenes and alkanes and lignin and xanthones, respectively.

Keywords Artificial neural network, supervised network, unsupervised network, self-organizing maps, QSAR, chromatographic parameters, electrophoretic mobility, peptide mapping, simulation of spectra, sulfonamides, heparanase inhibitors, terpenes, response factor, lignins, xanthones

1. General Objectives and Concepts

Artificial intelligence (AI) is the application of mathematical and logic techniques by human-made systems to carry out "reasoning" abilities similar to those of human beings. There are two main categories of AI developments. The first are expert systems, which are simply computer programs that simulate human experience and draw conclusions from a set of rules. A characteristic difference between an expert system and a conventional computer program is the strict separation of the knowledge from the program code. The second category includes systems that model the way the brain works, such as artificial neural networks (ANNs).

The original work on neural networks (perceptrons, as they were called at that time) was written nearly 60 years ago by McCulloch and Pitts [1, 2] and Hebb [3]. The introduction of nonlinearity and feedback coupling of outputs with inputs by Hopfield [4] gave new flexibility to the old architecture of the perceptron. Since the appearance of Hopfield's article, research on neural networks is back in vogue. In chemistry and related fields of research, like biochemistry, pharmacy, and analytical chemistry, interest in neural-network computing has grown rapidly in the past decade. This chapter covers a part of spectrum of neural-network uses in analytical chemistry,such as QSAR/QSPR studies, prediction of chromatographic parameters and biological activities, eletrophoretic mobilities and peptide mapping, and simulation of spectra. The common denominator for most of these attempts is modeling. In fact, a model that represents the transformation from a set of inputs to a set of outputs should be developed. Such a model is obtained by training the network. In training, the network is repeatedly presented with input/output pairs that have to be related by the transformation being modeled. Although training sometimes can be quite lengthy, training time is compensated by the extremely rapid processing in the trained network. Generally, a model is sought from an available set of data, such as a chromatographic/spectroscopic database. Such sets of data clearly contain a number of very interesting relationships, feature correlations, and other information that cannot be deduced in a straightforward manner from first principles, by theoretical calculations, or even with numerical methods. This leads to some conclusions on how to make ANNs more transferable or how the benefits of these techniques can be further introduced into the analytical community.

Before beginning a detailed description of the application of neural networks in analytical chemistry, some architectures of neural networks are described briefly. Particular emphasis is placed on the methods used in most research in analytical chemistry area.

2. Architecture of Neural Networks

Neural networks (NNs) arose from attempts to model the functioning of the human brain. They are a compact group of connected, ordered in layer elements able to process information. Artificial neural network (ANN)-based approaches have several advantages, which include a capacity to self-learn and model complex data without need for a detailed understanding of the underlying phenomena [5– 11]. An artificial neural network is a biological inspired computational model formed from hundreds of artificial neurons (processing elements), connected with coefficients (weights), which constitute the neural structure. Each processing element

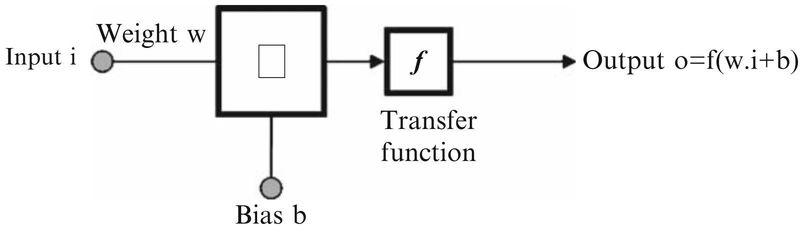


Fig. 6.1 A simple neuron model

(PE) has weighted inputs, a transfer function, and one output and is essentially an equation that balances inputs and outputs. Fig. 6.1 shows the model of a simple neuron. The input to the neuron i multiplied by its weight w and summed with the bias value b goes to the transfer function f. The output of the neuron is calculated as $O = f(w \times i + b)$.

Due to the increasing applications of artificial neural networks, many types of neural networks have been designed. When neural networks are used for data analysis, it is important to distinguish between artificial neural network models and neural network algorithms.

The neural network models are the network's arrangement, whereas ANN algorithms are computations that eventually produce the network outputs. The structure of a network depends on its application. When a network is structured according to its application, it is ready to be trained. The two approaches to training are supervised and unsupervised. The aim in supervised learning is to predict one or more target values from one or more input variables. Therefore, in this case, both inputs and outputs should be known. The supervised ANNs are useful for the prediction or classification purposes. ANNs with unsupervised algorithms are known as Kohonen or self-organizing maps. These algorithms are excellent at finding relationships among complex sets of data. A schematic view of both supervised and unsupervised learning is shown in Fig. 6.2.

2.1. ANN Models Based on Supervised Learning Algorithms

The relevant principle of supervised learning in ANN is that it takes numerical inputs (the training data) and transfers them into the desired output. The network approach consists of three stages:

1. Planning the experiment. Selecting the data and choosing their representatives (variables) that are to be used for input to the neural network. The data set is divided into training, test (monitoring), and prediction sets.
2. Designing the network architecture and implementation of personally programmed algorithms.

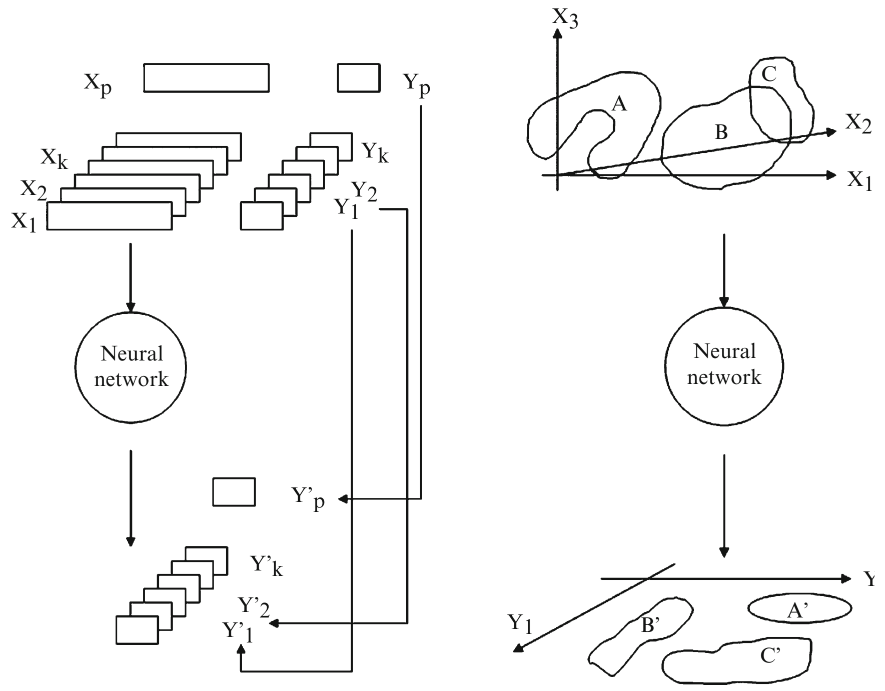


Fig. 6.2 Supervised (left) and unsupervised (right) learning

3. Evaluation of the predictive ability of the generated network and validation of the results obtained.

The architecture of an NN is determined by the way in which the outputs of the neurons are connected to the other neurons. The neurons of a network are divided into several groups, called *layers*. Single- and multiple-layer architectures are possible. Therefore, an ANN consists of some single units, artificial neurons, connected with coefficients called p*rocessing elements* or weights, which constitute the neural structure. The way that the neurons are connected to each other has a significant impact on the operation of the network. The two types of connections are feedforward and feedback architectures. A feedforward structure does not have a connection back from the output to the input neurons and, therefore, does not keep a record of its previous output values (Fig. 6.3). In contrast, feedback architecture has connections from output to input neurons (Fig. 6.4). Such a neurons keeps a memory of previous state so that the next state depends not only on input signals but also on the previous states of the network.

There are many types of neural architecture, but the most popular one for analytical chemistry is a supervised network with a back-propagation learning rule.

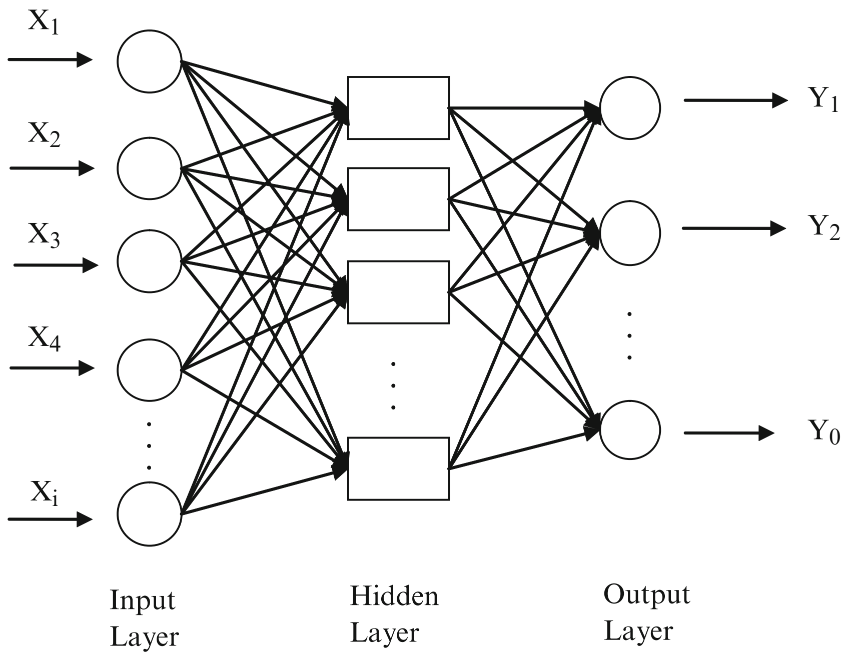


Fig. 6.3 Feedforward network

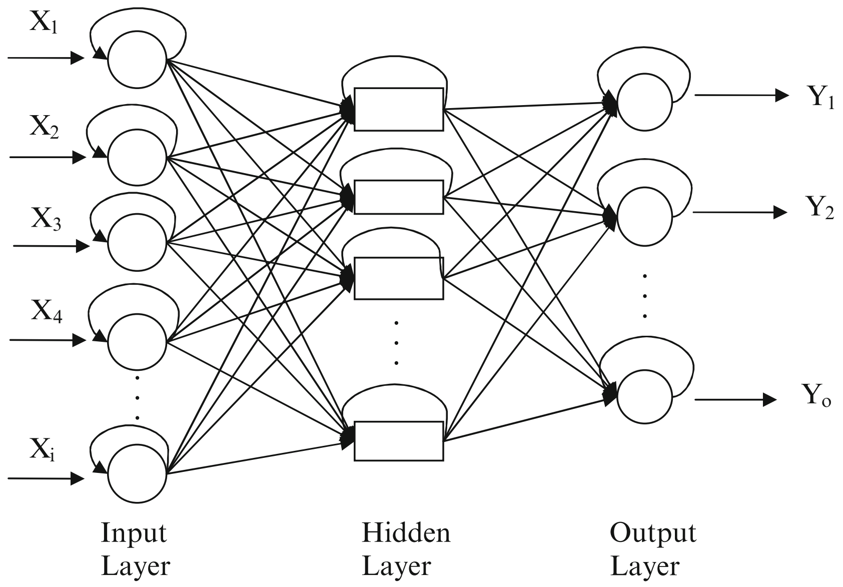


Fig. 6.4 Feedback network

2.2. Back-Propagation Artificial Neural Network

Back-propagation artificial neural networks (BPANNs) are multilayer feedforward networks based on a back-propagation algorithm. With given input vectors and targets, such networks can approximate a function or classify input vectors in an appropriate way, as defined by the user. A typical feedforward neural network with back-propagation has three layers: the input, the hidden, and the output layers. The input layer neurons receive data from a data file. The output neurons provide the ANN's response to the input data. Hidden neurons communicate only with other neurons. They are part of the large internal pattern that determines a solution to the problem. The information passed from one processing element to another is continued within a set of weights. Some of the interconnections are strengthened and some are weakened, so that a neural network produces a more correct answer. The activation of a neuron is defined as the sum of the weighted input signals to that neuron:

$$\mathrm{Net}_j = \sum_i W_{ij} X_i + \mathrm{bias}_j \tag{1}$$

where W_{ij} is the weight connection to neuron *j* in the actual layer from neuron *i* in the previous layer and bias_j is the bias of neuron *j*. The Net_j of the weighted inputs is transformed with a transfer function, which is used to get to the output level. Several functions can be used for this purpose, but the sigmoid function is most often applied. This function is as follows:

$$y_j = \frac{1}{1 + \mathrm{e}^{-\mathrm{Net}_j}} \tag{2}$$

where y_j is output of the neuron *j*. To train the network using the back-propagation algorithm, the differences between the ANN output and its desired value are calculated after each iteration. The changes in the values of the weights can be obtained using the equation:

$$\Delta w_{ij}(n) = \eta \delta_i O_j + \alpha \Delta w_{ij}(n-1) \tag{3}$$

where Δw_{ij} is the change in the weight factor for each network node, δ_i is the actual error of node *i*, and O_j is the output of node *j*. The coefficients η and α are the learning rate and the momentum factor, respectively. These coefficients control the velocity and the efficiency of the learning process. These parameters would be optimized before training the network. The goal of training a network is to change the weights between the layers in a direction that minimizes the error, *E*:

$$E = \frac{1}{2} \sum_p \sum_k (y_{pk} - t_{pk})^2 \tag{4}$$

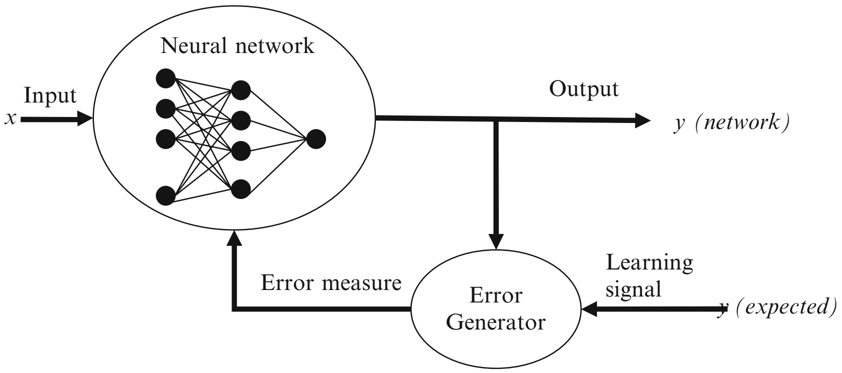


Fig. 6.5 Supervised network with a back-propagation learning rule

The error E of a network is defined as the squared differences between the target value t and the output y of the output neurons summed over p training patterns and k output nodes. In back-propagation learning, the error in prediction is fed backward through the network to adjust the weights and minimize the error, thus preventing the same error happening again. This process is continued with multiple training set until the error is minimized across many sets. Figure 6.5 shows a typical supervised network with a back-propagation learning rule.

The number of the neurons in the hidden layer influences the number of connections. During the training phase, inputs are adjusted (transformed) by the connection weights. Therefore, the number of connections has a significant effect on the network performance and should be optimized. Too few hidden neurons hinder the learning process and too many depress the process's abilities through overtraining. When the ANN is trained to a satisfactory level, the weighted links between the units are saved. These weights are then used as an analytical tool to predict results for a new set of input data. This is a prediction phase when the network works only by forward propagation of data and there is no backward propagation of error. The output of a forward propagation is the predicted model for the validation data.

2.3. ANN Models Based on Unsupervised Learning Algorithms

This type of network is called a *Kohonen network*, and it is good at finding relationships among complex sets of data. This type of architecture most closely resembles the connections and learning procedure of biological neurons [12– 16]. In unsupervised training, the network is left to itself to stabilize its inputs for all objects in an appropriate region of space created by the neural network (see Fig. 6.2). The learning is done by stimulation of the most active neuron and its neighborhood. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must decide what features it will use to group the input data. Consequently, this type of networks referred to as *self-organizing maps*. The objective is to construct a network that maps both input and output variables at once.

3. General Comments on Analytical Applications

The use of artificial neural networks is a relatively new but expanding area in the field of analytical chemistry. The artificial neural network represents a promising modeling technique, especially for data sets having the kind of nonlinear relationships frequently encountered in analytical chemistry. Neural networks require less formal statistical training and are able to detect complex nonlinear relationships between dependent and independent variables and all possible interactions without complicated equations.

The various applications of ANNs can be summarized into classification or pattern recognition, prediction, and modeling. Neural networks are able to recognize patterns even from noisy and complex data in which there is a considerable degree of variation and estimate nonlinear relationships. Therefore, ANNs are useful in all fields of research where the recognition of peak-shaped signals in analytical data is important. In terms of modeling, ANNs require no knowledge of the data source but, since they often contain many weights that are estimated, they require large training sets.

The potential applications of ANN methodology in analytical chemistry are broad. Among the different applications, this chapter focuses on the development of quantitative structure activity-property relationships (QSAR-QSPR) for modeling and interpretation of chromatographic data and biological activities. Also the use of these techniques is explored in electrophoresis as a powerful tool in separation techniques with emphasis on its application in peptide mapping. The ANNs have also been applied to establish reliable correlations between different types of spectra, such as infrared, mass, ^{1}H-NMR, ^{13}C-NMR, and the chemical structure of the corresponding compounds. Here, the application of ANNs on a simulation of ^{13}C-NMR and mass spectra is highlighted.

4. Applications of ANNs in Analytical Chemistry

A comprehensive look at all applications for ANNs in different areas of analytical chemistry is impractical. However, as examples, we focus on the most recent applications of neural networks in our laboratory in QSAR studies of biological activities, modeling of chromatographic parameters, peptide mapping, and simulations of spectra.

4.1. Quantitative Structure-Activity Studies of Biological Activities

QSAR methods correlate structural or property descriptors of compounds with their chemical or biological activities. The general assumption in QSAR-QSPR modeling is that the molecular structure is responsible for the observed behavior of a compound. The most challenging part of developing ANN QSAR models is choosing suitable variables as their inputs. These parameters, which include descriptors to account for hydrophobicity, topology, geometric or steric effects, and electronic properties, can be determined empirically or by computational methods. The most popular software, which is able to calculate more than 1,450 descriptors for a compound, is Dragon software [17].

Jalali-Heravi and Parastar developed a three-layer network with a sigmoidal transfer function and a supervised back-propagation learning algorithm for modeling and predicting the anti-HIV activity of the HETP derivatives [18]. The data set consists of 107 inhibitors of the HIV-1 reverse transcriptase, derivatives of 1-[2-hydroxyethoxy)-methyl]-6-(phenylthio)thymine (HETP) [19]. The chemical structure of these compounds is shown in Fig. 6.6, where R_1, R_2, and R_3 consist of a variety of substituents [18]. This set was divided into a training set for the generation of the ANN model and a prediction set for the evaluation of the model and taking care of the overtraining.

A total of 80 compounds was chosen as training set by which a 6-6-1 neural network was developed with the optimum momentum and learning rate of 0.9 and 0.2, respectively [18]. The remaining 27 inhibitors were used as prediction set. The value of log 1/*C* was used as the dependent variable, in which *C* represents the molar concentration of drug required to achieve 50% protection of MT-4 cells against the cytopathic effect of HIV-1 (HTLV-III$_B$ strain) [20]. One of the most critical points in the neural network approach is the effect of overtraining. To prevent the overtraining, the calculated mean square errors (MSEs) for the training and prediction sets were plotted against the number of iteration. The training of the network should be stopped when the MSE starts to increase for the prediction set. The plot of the calculated versus the experimental anti-HIV activity for the HETP derivatives shows a correlation coefficient of $R = 0.959$.

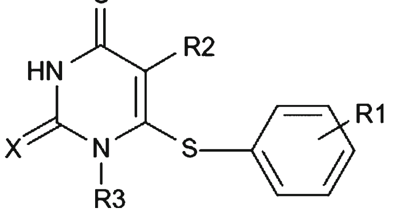


Fig. 6.6 The structure of HIV-1 reverse transcriptase inhibitors

Since Hansch et al.'s seminal work on quantitative structure-activity relationships analysis [21], many QSAR methods, including two- and three-dimensional QSAR, approaches have been developed. These models have been used to study action mechanisms of chemical-biological interactions in modern drug discovery [22]. However, a common problem for these models is choosing an optimal set of structural descriptors. A large number of structural descriptors can be generated by existing software [17, 23], but choosing adequate descriptors for QSAR-QSPR studies is difficult and challenging. To overcome this problem, a powerful variable selection technique is needed. Recently, Jalali-Heravi and Kyani introduced the genetic algorithm-kernel partial least square (GA-KPLS) system, as a novel nonlinear feature selection method [24]. This method is combined with a feedforward back-propagation artificial neural network to develop a nonlinear QSAR model (GA-KPLS-ANN) for predicting the activities of a series of substituted aromatic sulfonamides as carbionic anhydrase II (CA II) inhibitors.

Sulfonamides represent an important class of biologically active compounds. The antibacterial sulfonamides continue to play an important role in chemotherapy, alone or in combination with other drugs. The sulfonamides that inhibit the zinc enzyme carbionic anhydrase (CA, EC 4. 2. 1. 1) possess many applications as diuretic, antiglaucoma, or antiepileptic drugs. [25]. The aromatic-hetrocyclic sulfonamides act as carbionic anhydrase inhibitors; and other types of derivatives show hypoglycemic activity, anticancer properties, or may act as inhibitors of the aspartic HIV protease used for the treatment of AIDS and HIV infection. [26]. The data set studied in this work consisted of 114 molecules, for which the biological activities were reported as log IC_{50} values [26, 27]. QSAR studies of these compounds have been reported, but they were restricted to linear regression models using two- and three-dimensional descriptors. [26, 27]. The inhibition effects are expressed as log IC_{50} in terms of nanomolar affinity for the investigated carbionic anhydrase isozymes. The structure of sulfonamides studied in this work is given in Fig. 6.7.

In developing the network, the output layer represents log IC_{50} of the CA II inhibitors. The descriptors were selected using GA-KPLS and considered as inputs for ANN. In this investigation, the sigmoid function was used as transfer function. The initial weights of the network were randomly selected from a uniform distribution that ranged between –0.3 and +0.3. The initial values of biases were set to be 1. These values were optimized based on the back-propagation learning algorithm during the network training. Before training, the inputs were normalized between –2 and 2 and output between 0.2 and 0.8. The network parameters, such as the number of nodes in hidden layer, learning rate, and momentum, were optimized based on obtaining the minimum standard error of training (SE_T) and standard error of prediction (SE_P) [28]. The architecture and specification for the hybrid method of GA-KPLS-ANN are given in Table 6.1.

The method of leave-one-out cross-validation (LOOCV) was used for the evaluation of the ANN model. Also a leave-multiple-out cross-validation (LMOCV) method was used for comparing the consistency of the generated model and comparing its results with GA-PLS-ANN, for which the variable selection method is linear. Figure 6.8 shows the plot of the predicted GA-KPLS-ANN values of log IC_{50}

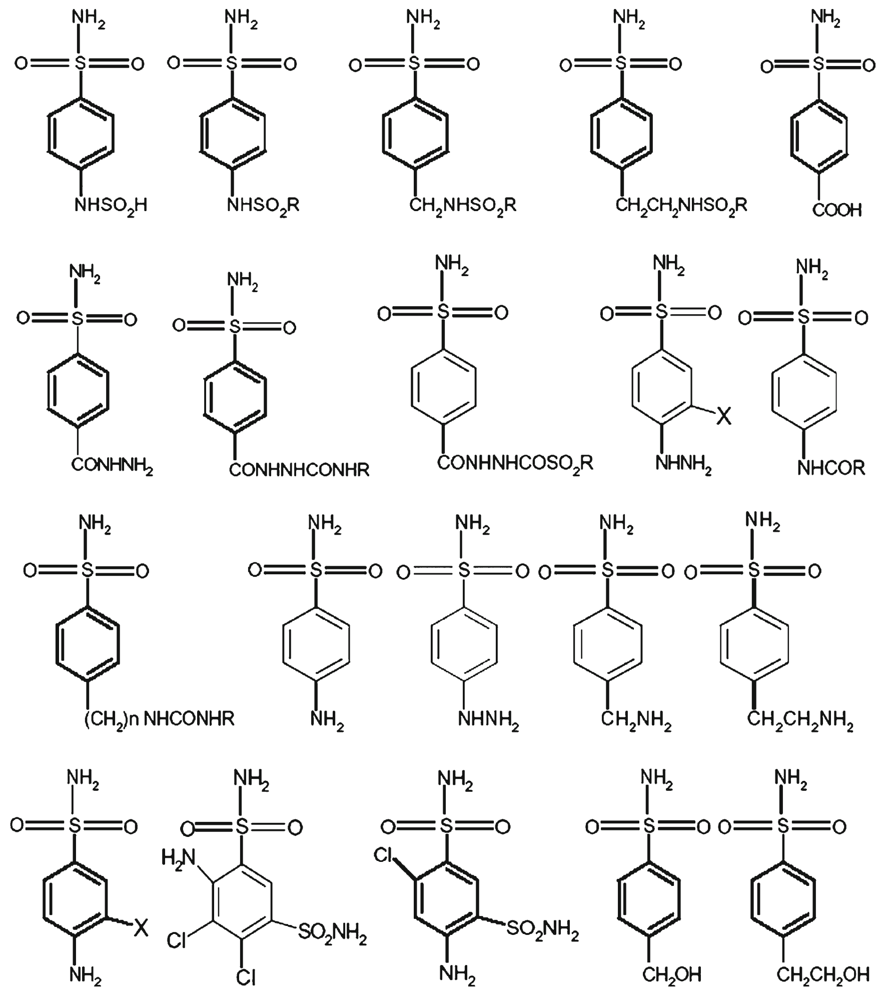


Fig. 6.7 The structures of sulfonamides as CA II inhibitors

Table 6.1 Architecture and specifications for hybrid method of GA-KPLS-ANN

	GA-KPLS-ANN
No. of nodes in the input layer	8
No. of nodes in the hidden layer	4
No. of nodes in the output layer	1
Learning rate	0.1
Momentum	0.1
Transfer function	Sigmoid
No. of iterations	1100

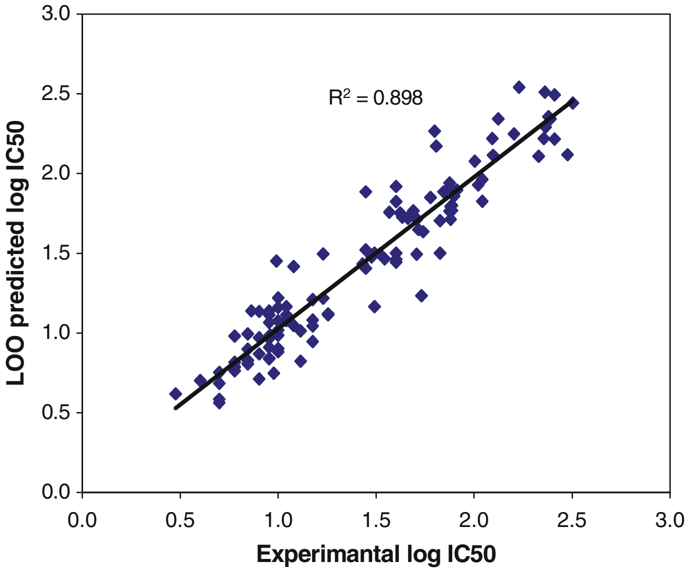


Fig. 6.8 Plot of the ANN calculated values of log IC_{50} against experimental values

based on LOOCV versus the experimental ones for the inhibitors in the data set. This plot by revealing a correlation (R^2) of 0.898 shows the strength of the model in predicting the inhibitor activities.

The descriptors appearing in the recommended model reveal the role of acceptor-donor pair, hydrogen bonding, hydrosolubility, and lipophilicity properties of inhibitor active sites and the size of inhibitors on inhibitor-isozyme interaction.

In another attempt, a QSAR model was developed to predict the activity parameter ($p\mathrm{IC}_{50}$) of heparanase inhibitors. Heparan sulfate proteoglycans (HSPGs) play a key role in the self-assembly, insolubility, and barrier properties of basement membranes and extracellular matrices [29]. The basic HSPG structure consists of a protein core to which several linear heparan sulfate (HS) chains are covalently O-linked. Hence, cleavage HS affects the integrity and functional state of tissues and thereby fundamental normal and pathological phenomena involving cell migration and response to changes in the extracellular microenvironment [30]. Heparanase enzyme cleaves heparan sulfate, so heparanase may facilitate both tumor cell invasion and neovascularization in critical steps in cancer progression [29]. In addition, expression of heparanase has long been correlated with the metastatic potential of tumor cell [29, 31]. In fact, treatment with heparanase inhibitors markedly reduces tumor growth, metastasis, and autoimmune disorders in animal models. Although these data indicate the importance of heparanase as a drug target, progress in this area has been limited by the lack of small molecules acting as inhibitors. However, it is shown that the heparanase inhibition mode of action is complicated because of its large structure [32]. Therefore, the challenging problem is to find small, druglike molecules with acceptable dystrophia myotonica-protein kinase (DMPK) properties for evaluation in animal models and to use it as therapeutic leads. Recently, different derivatives of three series of molecules have been reported as heparanase inhibitors [32, 33]. These are the derivatives of 2,3-dihydro-1,3-dioxo-1H-isoindole-5-carboxylic acid, furanyl-1,3-thiazol-2-yl and benzoxazol-5-yl acetic acid. The structure of these compounds is given in Fig. 6.9.

A quantitative structure-activity relationship study was carried out on these compounds [34]. In developing ANN models, different weight update functions have been chosen and their influence on the reliability of the model examined. The weight update functions were shown to play an important role in the reliability of the models using ANN. Therefore, selecting a suitable weight update function is essential for developing a good model.

The activity parameter IC_{50} is a measure of antiviral potency and refers to the molar concentration of each compound required to reduce the concentration of heparanase viral by 50% with respect to the level measured in an infected culture.

The data set was separated into three groups: training, test, and prediction sets. All molecules were randomly placed in these sets. The training and test sets, consisting of 45 and 25 molecules, respectively, were used to generate the model. However, the test set was used to take care of the overtraining. The prediction set, consisting of 22 molecules, was used to evaluate the generated model. A three-layer network with a sigmoid transfer function was designed for each ANN. Before training, the networks the input and output values were normalized between –1 and 1. The initial weights were selected randomly between –0.3 and 0.3. The network was then trained, using the training set by the back-propagation strategy for optimization of the weights and bias values. It is common practice to optimize the parameters of a number of nodes of the hidden layer, learning rate, and momentum in developing a reliable network. Also, one has to report the number of

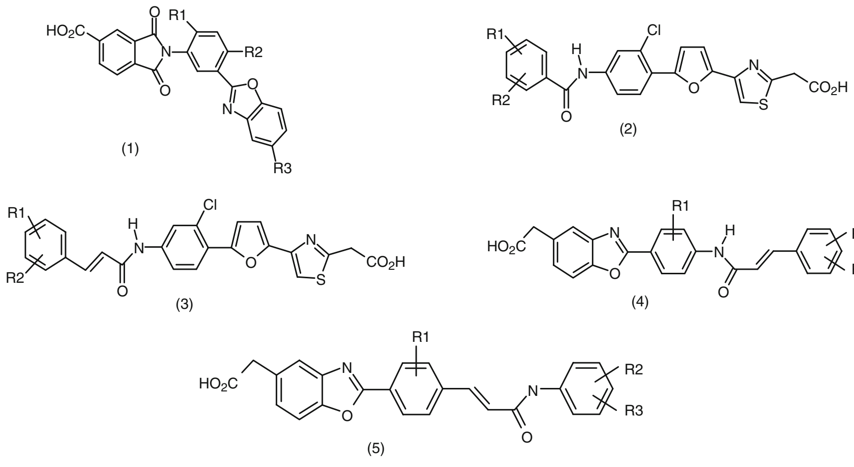


Fig. 6.9 The structure of molecules acting as heparanase inhibitors

the iteration when the overtraining begins. The procedure for optimizing these parameters is given elsewhere [35]. However, as it can be seen from Eq. 6, there is a term called weight update function, F_n,

$$\Delta W_{ii,n} = F_n + a\Delta W_{ij,n-1} \tag{6}$$

which indicates the way that weights are changed during the learning process. Jalali-Heravi, Asadollahi-Baboli, and Shahbazikhah focused on investigating the role of the weight update function, a specification of the networks that has not been reported by many researchers [34]. The three weight update functions of the basic back-propagation (BBP) algorithm, conjugate gradient (CG) algorithm, and Levenberg-Marquardt (L-M) algorithm, in conjunction with the optimized parameters, were used, for which the statistical parameters are given in Table 6.2. Inspection of this table reveals the importance of the role of algorithms by which the weight update functions are considered. While the learning rate and momentum are almost constant, the statistics of R^2, Q^2, and SE changed considerably. As the trend of variations for different algorithms are consistent for the modified data sets of leave one out (LOO), leave 8 out (L8O), and leave 12 out (L12O), one may choose the best algorithms by comparing the values of R^2, Q^2, and SE for these algorithms. Table 6.2 shows the superiority of L-M algorithm over the BBP and CG algorithms. Therefore, a 4-3-1 BPANN was developed using the L-M algorithm (L-M ANN) to predict the $p\mathrm{IC}_{50}$ of heparanase inhibitors. Figure 6.10 demonstrates the plot of the L-M ANN predicted versus the experimental values of the $p\mathrm{IC}_{50}$ for the data set.

As a final step, the proposed linear (MLR) and nonlinear (ANN) models were used to interpret the mechanism of the inhibition. This means that one should investigate, among the four descriptors appearing in the MLR model and used as ANN input, which variables are the most important predictors. In the case of the MLR, the mean effect of each descriptor can be considered as measure of its role in predicting the $p\mathrm{IC}_{50}$ of the inhibitors. For the neural network, the sensitivity analysis was incorporated to determine the importance or effect of the input variables on

Table 6.2 Statistical and optimized parameters for three weight update functions

		Optimized	LOO		L8O[b]		L12O[b]	
Update weight function	Optimized learning rate	momentum constant	Q^2	SE	R^2	SE	R^2	SE
Basic back propagation	0.4	0.9	0.768	0.40	0.692	0.37	0.696	0.36
Conjugate gradient	0.2	0.8	0.865	0.34	0.725	0.28	0.703	0.26
Levenberg-Marquardt[a]	0.2	0.9	0.936	0.21	0.875	0.18	0.860	0.17

[a]Initial values: $\mu = 0.01$ and $\lambda = 10$
[b]Calculation of R^2_{LMO} was based on 200 random selections of groups of 8 and 12 samples.

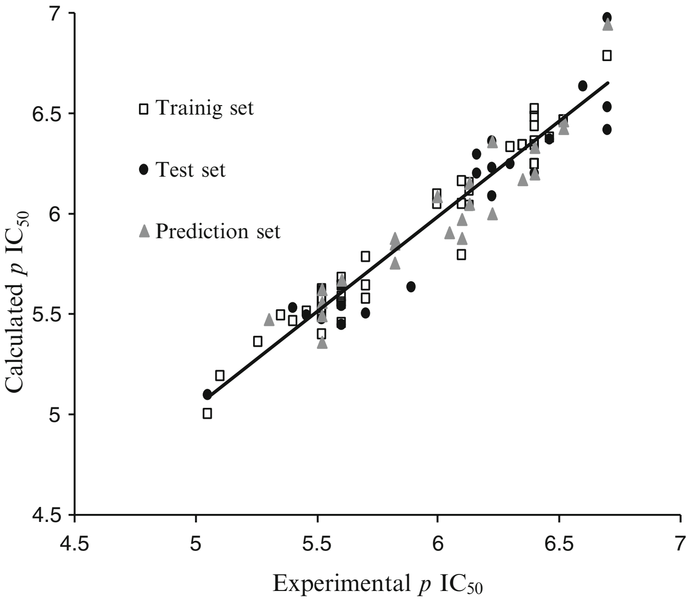


Fig. 6.10 Experimental value of $p\text{IC}_{50}$ versus calculated values

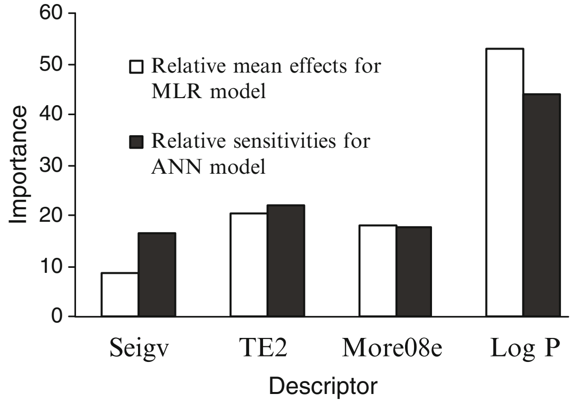


Fig. 6.11 Relative mean effects and sensitivities for each variable for the MLR and ANN models

the network output. Figure 6.11 shows the relative mean effect and sensitivity of each variable for the MLR and ANN models, respectively. Both models show that log P is the most important parameter affecting the inhibitory behavior of the molecules.

QSAR methodologies have been applied successfully to establish a mathematical relationship between the activity of heparanase inhibitors and physicochemical, topological, and electronic indices. The superiority of nonlinear (L-M BPANN) over the linear (MLR) model revealed that the inhibitory behavior has nonlinear characteristics.

4.2. Chromatographic Parameters

Within analytical chemistry, chromatography is a particularly active area where research is being undertaken to implement artificial neural networks. The most important aim of mathematical and statistical methods in chemistry is to provide the maximum information about the selected molecular property by analyzing chemical data. In chromatography, retention is a phenomenon that depends on the solute-solute, solute–stationary phase, and solute–mobile phase interactions. If the mobile and stationary phases are the same for the solutes, then only the differences in the structures of the solute molecules need to be investigated. The separation and identification of different compounds in a complex mixture rely heavily on gas or liquid chromatographic techniques. In some cases, GC may be the sole means of identification based on direct comparison of retention times with standards or precise knowledge of the Kovats retention indices (*I* value). A Kovats retention index reports the retention behavior of a compound relative to a standard set of hydrocarbons, utilizing a logarithmic scale. The retention index of compound A, I_A, is defined as

$$I_A = 100N + 100 \bullet \frac{\log t'_{R(A)} - \log t'_{R(N)}}{\log t'_{R(N+1)} - \log t'_{R(N)}} \tag{7}$$

where $t'_{R(A)}$ is the adjusted retention time of compound A and $t'_{R(N+1)}$ and $t'_{R(N)}$ are the adjusted retention times of n-alkanes of carbon number n + 1 and n, which are, respectively, larger and smaller than the adjusted retention time of the unknown [36]. Retention is a phenomenon primarily dependent on the interactions between the solute and the stationary phase molecules. The forces associated with the interactions can be related to the geometric and topological structures and electronic environments of the molecule. Therefore, structural parameters, such as topological, geometric, electronic, and physicochemical descriptors, can be generated and used as inputs for developing an ANN model to predict the Kovats retention index or the retention time of the molecules.

A series of six comprehensive descriptors that represent different features of the gas-liquid partition coefficient, K_L, for commonly used stationary phases was developed by Jalali-Heravi and Parastar [37]. These descriptors consist of the dipole moment (DIPOL), the highest occupied molecular orbital (HOMO), the partial charge of the most negative atom (PCHNEG), the partial charge of the most positive hydrogen (PCHPOSH), molecular mass (Mr), and van der Waals volume. The DIPOL represents the ability of a molecule to take part in dipole-dipole and dipole–induced dipole interactions. The parameter of HOMO is a measure of the ability of a molecule to interact with the π- and n-electron pairs of the other molecules. PCHPOSH and PCHNEG can be considered measures of the acidity and basicity of a molecule, respectively. It is obvious that, as the molecular mass and the volume of a molecule increase, the cavitation energy and dispersion interaction increase. A data set consisted of 54 molecules [38] from a variety of organic compounds was divided into training (39 molecules) and prediction (15 compounds) sets. A separate multiple linear regression (MLR) model was developed by using these descriptors for each stationary phase of poly(ethylene glycol adipate((EGDA), N,N,N',N'-tetrakis (2-hydroxypropyl)ethylendiamine (THPED)), poly(ethylene glycol) (Ucon 50 HB 660) (U50HB), di(2-ethylhexyl)phosphoric acid (DEHPA), and tetra-n-butylammonium N, N-(bis-2-hydroxylethyl)-2-aminoethanesulfonate (QBES). The results obtained using these model show correlation coefficients ranged from 0.944 to 0.966 and are in good agreement with the experiment.

A back-propagation strategy was used for the training of an ANN using the aforementioned descriptors as its inputs. There have been two purposes behind developing such a model: First, as mentioned before, a separate MLR model was developed for each stationary phase; that is, five models were developed. Success in developing the ANN model would reduce these models to only one model. Second, comparison of the MLR and ANN results is useful for investigating the linearity or nonlinearity of the retention mechanism. Therefore, a 6-6-5 BPANN model was developed with the optimum momentum and learning rate of 0.4 and 0.2, respectively. To prevent the overtraining, the mean square errors (MSEs) for the training and the prediction sets were plotted against the number of iterations. The overtraining started after 35,500 iterations training of the network. To evaluate the neural network, the MSEs of its results for the training and the prediction sets are compared with the MSEs of the regression models for different stationary phases in Table 6.3.

Comparison of the MSEs shows the superiority of the BPANN model over that of the MLRs. This indicates that some of the descriptors appearing in the MLR models interact with each other, and on the whole, the retention behaviors of the molecules on different columns show some nonlinear characteristics.

In an other contribution, a QSPR study based on MLR and ANN techniques was carried out to investigate the retention behavior of some terpenes on the polar

Table 6.3 Comparison of the MSEs for the results obtained using ANN and regression models

	MSE			
	Training		Prediction	
Column[a]	MLR	ANN	MLR	ANN
EGAD	0.020	0.005	0.021	0.010
THPED	0.024	0.005	0.016	0.005
U50HB	0.022	0.006	0.018	0.005
DEHPA	0.021	0.004	0.019	0.010
QBES	0.027	0.007	0.012	0.009

[a]Definition of the columns is given in the text.

stationary phase (Carbowax 20 M) [39]. Terpenes are natural products that exist in many families of plants. A collection of 53 noncyclic and monocyclic terpenes was chosen as the data set, which was randomly divided into training (41 terpenes) and prediction (12 molecules) sets. The data set and corresponding observed and ANN predicted values of the Kovats retention indices are shown in Table 6.4.

A total of six descriptors appeared in the MLR model using the stepwise procedure. These parameters consist of electron density on the most positive atom (EDPA), second-order kappa index ($^2\kappa$), molecular shadow surface area on *x-y* plane (S_{xy}), standardized molecular shadow area on *x-z* plane (SS_{xz}), molecular density (MD), and path on connectivity index (^1X_p). These parameters reveal the role of important factors, such as size, degree of branching, and steric interactions, on the retention behavior of terpenes. A 6-5-1 ANN model was generated by using the six descriptors appearing in the MLR model as inputs. The back-propagation strategy was used for the optimization of the values of the weights and biases. The optimized parameters of momentum, weights learning rate, and biases learning rate are 0.5, 0.7, and 0.5, respectively. The mean of relative errors between the ANN calculated and the experimental values of the Kovats retention indices for the prediction set was 1.88%. This is in agreement with the relative error obtained by experiment. The plot of the ANN calculated versus the experimental *I* values of the prediction set shows a correlation coefficient of 0.984, which confirms the ability of the ANN model in predicting *I* for noncyclic and monocyclic terpenes.

In 2002, Jalali-Heravi and Garkani-Nejad wrote a paper in which an artificial neural network and, for the first time, a self-training artificial neural network (STANN) were employed to predict the relative retention times of 13 classes of organic compounds [40]. The data set in this study consisted of 122 compounds, including alcohols, ketones, aldehydes, esters, alkenes, alkynes, alkanes, halides, thiols, nitro, ethers, cyanides, and sulfides [41].

A STANN is a procedure for updating the node's weights and training of the networks in parallel fashion. An important aspect of the STANN is a network that trains another network. The architecture of a STANN is shown in Fig. 6.12.

The structure of network 2 in this figure is the same as a BPANN. However, during the training, the normalized inputs are increased by some infinitesimal amount, delta (Δ). In this regard, because the transfer function being utilized, a sigmoid, has a linear region around the value 0.5, it is desirous when adding the delta value to the normalized input to adjust the input toward the linear region. Therefore, the positive delta value should be added to normalized inputs less than 0.5 and negative delta values to normalized inputs greater than 0.5. In the hidden layers, a similar step is used. Network 1 uses weights from updates produced by the training network 2. Consequently, the training of artificial neural network 1 is not carried out with algorithmic code but rather by a network training a network. In this research, an MLR model is included to choose a suitable set of numerical descriptors among the vast number of parameters available and to use them as inputs for neural network generation. The five descriptors appearing in the selected MLR model are molecular density, Wiener number, boiling point, polarizability, and square of polarizability. A 5-6-1 ANN and a 5-4-1 STANN are generated using these parameters as their inputs [40]. For comparison purposes, the statistics for the STANN,

Table 6.4 Observed and ANN predicted values of the retention indexes of terpenes

Number	Name	I_{obs}	I_{ANN}	I_{rel}%
Training set				
1	Citronellyl formate	1,638	1,627	–0.97
2	1,2 Dihydrolinalool	1,537	1,521	–1.04
3	Geranyl formate	1,717	1,702	–0.87
4	Terpinene-4-yl acetate	1,640	1,627	–0.79
5	Myrcene-8-ol	1,919	1,896	–1.20
6	Neoisocarvonmenthyl acetate	1,672	1,664	–0.48
7	Neryl butyrate	1,868	1,825	–2.30
8	Isopulegyl acetate	1,608	1,643	2.17
9	Linalyl butyrate	1,698	1,676	–1.30
10	Carveyl acetate	1,795	1,810	0.84
11	Citronellyl isobutyrate	1,739	1,739	0.00
12	Neryl formate	1,700	1,718	1.06
13	Geranyl isobutyrate	1,821	1,821	0.00
14	6,10-Dihydromyrcenol	1,473	1,501	1.90
15	Isomenthyl acetate	1,599	1,655	3.50
16	Linalyl acetate	1,569	1,606	2.36
17	Dihydrocarvyl acetate	1,700	1,722	1.29
18	Myrcenol	1,631	1,678	2.88
19	Neryl acetate	1,735	1,767	1.85
20	Carvomenthyl acetate	1,641	1,670	1.76
21	Dihydronerol	1,725	1,780	3.18
22	Linalool	1,555	1,565	0.64
23	Neoisomenthyl acetate	1,623	1,651	1.72
24	*cis*-Ocimenol	1,660	1,708	2.90
25	Tetrahydrolinalyl acetate	1,422	1,457	2.46
26	Citronellol	1,765	1,767	0.11
27	Citronellyl acetate	1,671	1,672	0.05
28	*cis*-β-Terpinyl acetate	1,622	1,682	3.69
29	Tetrahydrogeranyl acetate	1,582	1,618	2.27
30	Perillyl acetate	1,791	1,789	–0.12
31	6,7-Dihydrolinalool	1,449	1,513	4.41
32	Lavandulol	1,707	1,687	–1.17
33	Neryl propionate	1,794	1,856	3.45
34	Nerol	1,808	1,838	1.66
35	Citronellyl butyrate	1,811	1,856	2.48
36	Geranyl butyrate	1,904	1,939	1.84
37	Tetrahydrogeraniol	1,675	1,723	2.86
38	Tetrahydromercenol	1,449	1,458	0.63
39	Terpinyl acetate	1,722	1,668	–3.13
40	Linalyl propionate	1,624	1,629	0.30
41	Tetrahydrolinalool	1,431	1,459	1.96
Prediction set				
42	*trans*-Carveyl acetate	1,759	1,749	–0.57
43	α-Citronellol	1,760	1,789	1.65
44	Citronellyl propionate	1,738	1,774	2.07
45	Geraniol	1,842	1,788	–2.93
46	Geranyl propionate	1,834	1,840	0.33
47	Lavandunyl acetate	1,609	1,553	–3.48
48	Linalyl isobutyrate	1,622	1,588	–2.10
49	Menthyl acetate	1,600	1,652	3.25
50	Neomenthyl acetate	1,569	1,599	1.91
51	Tetrahydrolavandulol	1,600	1,562	–2.38
52	Nerylisobutyrate	1,790	1,791	0.05
53	*trans*-Ocimenol	1,685	1,717	1.90

ANN, and MLR models are shown in Table 6.5. The correlation coefficient (*R*) and standard error of predictions (SEPs) of these models indicate that the obtained results using STANN and ANN are much better than those obtained using the MLR model. This is believed to be due to the nonlinear capabilities of the STANN and ANN. Inspection of the results of STANN and ANN (Table 6.5) shows there are few differences between these methods. However, optimization of STANN is much faster and the number of adjustable parameters of this technique is much less than those of the conventional ANN.

A hybrid method consisting of principal components analysis (PCA), multiple linear regression (MLR), and artificial neural network was developed to predict the retention time of 149 C_3–C_{12} volatile organic compounds for a DB-1 stationary phase [42]. The problem of the inability of the ANNs to select the appropriate descriptors as their inputs was overcome by using the PCA and linear regression techniques. PCA can be used as a powerful tool for clustering the descriptors when

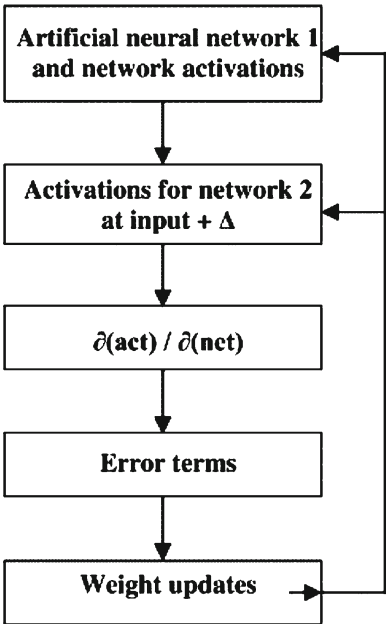

Artificial neural network 1 and network activations
Activations for network 2 at input + Δ
∂(act) / ∂(net)
Error terms
Weight updates

Fig 6.12 The architecture of a self-training artificial neural network

Table 6.5 Statistical parameters obtained using the STANN, ANN, and MLR models

Model	SET (%)	SEP (%)	R_{training}	$R_{\text{prediction}}$
STANN	2.135	2.364	0.996	0.992
ANN	2.036	2.279	0.995	0.992
MLR	32.027	33.326	0.960	0.951

a large number of them with different features are available. The descriptors of the total information index of atomic composition (IAC), Wiener number (W), solvation connectivity index (X1sol), and number of substituted aromatics $C_{(sp_2)}$ (nCaR) appeared in the MLR model and were used as inputs for the ANN generation. The appearance of these parameters shows the importance of the dispersion interactions in the mechanism of retention. Also, the solvation connectivity index shows a large contribution to the retention time, revealing the importance of the entropy of solvation in retention behavior. Our success in developing the ANN model indicates that choosing the appropriate inputs is the main key in developing an ANN model. The techniques of PCA and MLR can be considered powerful feature-selection tools for this purpose.

The development of sensitive and selective detectors has played a major role in the establishment of chromatography as an unrivaled analytical tool. The retention time can be used for identification of compounds, but it is well accepted that more than one compound can have similar retention times. However, different detector responses can be used for peak identification of compounds with the same retention time. The response factor (RF) is the fundamental measure of the response of the detector to a given compound and can be considered as a correlation factor. Since numerous compounds are unavailable as standards, the development of theoretical methods for estimating response factor seems to be useful.

Jalali-Heravi and coworkers used artificial neural networks for predicting the response factors of flame ionization detection (FID) [35], thermal conductivity detection (TCD) [43], photoionization detection (PID) [44], and electron capture detection (ECD) [45] systems for different series of organic molecules. In all these contributions, as a first step, MLR models were employed to find information subsets of descriptors that can predict the relative response factors (RRFs) of these compounds. The descriptors appearing in the MLR models were considered inputs for the ANN or STANN models. In all cases, comparison of the results indicates the superiority of neural networks over that of the MLR and demonstrates a nonlinear behavior for RRFs of all type of GC detection systems. Table 6.6 shows different factors affecting the RRFs of TCD, FID, Ar-PID, Kr-PID, and ECD detection systems. The results of these research studies reveal a nonlinear characteristic for the mechanism of RRFs. Also, one may conclude that the mechanism of RRFs is very complicated and depends on different features of the organic molecules.

4.3. Electrophoresis and Peptide Mapping

Peptide mapping is a widely used technique for the characterization of the protein structure that involves digestion of a protein through enzymatic or chemical means and the subsequent separation and detection of the resultant peptide mixture. The peptide maps serve as "fingerprints" that can be applied to rapid protein identification and the detection of posttranslational modifications. Capillary liquid chromatography (combined with MS/MS) is currently one of the most commonly used techniques

Table 6.6 The most important factors affecting the RRFs of different GC detection systems [43, 44]

Detection method	Descriptor
FID	Boiling point
	Maximum bond order of C-*X*
	Path one connectivity index
	Polarizability
	Square of polarizability
	Relative number of C atoms
	Maximum valency of H atoms
Ar-PID	Molecular density
	Dipole moment
	Cluster three connectivity index
	Heat of formation
	Highest occupied molecular orbital
	Distance between center of mass and center of charge
Kr-PID	Molecular density
	Path four connectivity index
	Highest occupied molecular orbital
	Principal moment of inertia about the *x* axis
	Relative weight of effective C atoms
TCD	Molecular mass
	No. of vibrational modes
	Molecular surface area
	Balaban index

for peptide mapping [46, 47]. While this method provides excellent resolution, it is often slow and generally consumes relatively large quantities of peptides. Capillary zone electrophoresis (CZE) has received considerable attention as peptide mapping method because of its high speed and high resolution for peptide analysis and also its small sample size requirement. However, electrophoretic peptide profiles obtained are sometimes very complex owing to the complicated nature of the samples. Generally, capillary electrophoresis separation of peptides is more successful than that of proteins because the smaller molecules tend to interact less compared with the capillary wall.

The key parameter for separation of peptides, especially in low ionic strength buffers, is their electrophoretic mobilities. This parameter can be converted to migration time and a CZE electropherogram can be simulated using a Gaussian function. Therefore, calculation and prediction of this parameter is very useful in peptide mapping studies.

In CZE, electrophoretic mobilities of charged compounds is expressed as

$$\mu_{\mathrm{ef}} = \frac{q}{6\pi\eta r} \tag{8}$$

where μ_{ef} is the effective electrophoretic mobility at a given ionic strength, r is the effective ion radius, q is the ion charge, and η is the solution viscosity.

Experimentally, the effective mobility of peptides can be determined by Eq. 9:

$$\mu_{\mathrm{ef}} = \frac{L_t L_d}{V}\left(\frac{1}{t_r} - \frac{1}{t_{\mathrm{eo}}}\right) \tag{9}$$

where L_t is the total length of the capillary, L_d is the length from the capillary inlet to the detection point, V is the applied voltage, t_r is the peptide retention time, and t_{eo} is the retention time of the EOF marker, such as mesityl oxide.

Voluminous numbers of attempts are reported for calculating the electrophoretic mobilities of peptides [48– 52], but a robust model has not been developed yet to predict accurately this parameter for all categories of peptides, especially highly charged and hydrophobic peptides. Several empirical models have been developed for the calculation or prediction of electrphoretic mobilities [48– 51]. Most of these models are based on Stoke's law for ion mobility in an electric field that is valid mainly for rigid spherical molecules in low ionic strength buffers. Among different models, the Offord model ($Q/M^{2/3}$) shows a reasonable correlation. Although the Offord model is superior to others, this model does not account for several factors that affect peptide mobility. It means that μ_{ef} cannot be successfully predicted for all categories of peptides relying only on the two parameters of charge and size. Recently, two papers have been published [53, 54] that aim to (1) accurately predict the electrophoretic mobilities of peptides, (2) achieve a better understanding of the physicochemical basis of the motion of a peptide in an electric field, and (3) investigate the linear and nonlinear relationships between the electrophoretic mobilities of peptides and different structural parameters of amino acids. The general strategy used in these contributions consisted of the following steps:

1. Measure the electrophoretic mobilities of the training set of peptides with different compositions and a sequence ranging in size between 2 and 42 amino acids.
2. Use a subjective method to select the proper descriptors as inputs for the generation of the quantitative structure-migration relationships (QSMR) models.
3. Generate an ANN model to investigate the nonlinear behavior of mobility and the development of a model to accurately predict peptides migration behavior.

In the first attempt, a diverse data set consisting of 125 peptides, ranging in size from 2 to 14 amino acids and a charge from 0.743 to 5.843, was chosen; and their electrophoretic mobilities were measured at pH 2.5 using CZE [53]. The stepwise MLR procedure was used for selecting suitable descriptors and generating a linear model. The best MLR model was consisted of three descriptors of Offord's charge over size term ($Q/M^{2/3}$) [48], steric substituent constant (E_s) [6], and molar refractivity [7, *8*], as in Eq. 10:

$$\mu = p\frac{Q}{M^{2/3}} + e\sum E_s + m\sum MR \tag{10}$$

The molar refractivity is a constitutive-additive property calculated by the Lorenz-Lorentz formula [7]. MR is strongly related to the volume of the molecules (molecular bulkiness). The steric substituent constant (ES) has been defined by Taft as log (k/k_0), where k and k_0 are the rate constants for the acidic hydrolysis of a substituted ester and a reference ester, respectively [55]. This parameter represents the steric interactions. The MLR model showed an improvement in the predictive ability over the simple Offord's relationship.

A three-layer back-propagation network with a sigmoidal transfer function was designed in this work [53]. The three descriptors appearing in the MLR model were used as input parameters for the network generation. The signals from the output layer represent the electrophoretic mobilities of the peptides. Such a BPANN may be designed as 3-y-1 net to indicate the number of nodes in input, hidden, and output layers, respectively. The final BPANN model has an architecture of 3-4-1, with the optimized parameters of learning rate and momentum of 0.5 and 0.5, respectively. Inspection of the BPANN-calculated mobilities reveals a remarkable improvement for those peptides containing the highly charged amino acids of arginine, histidine, and lysine. For example, the deviation of –19.40% for the MLR-calculated value of KYK peptide should be compared with the 3.95% for the BPANN model. Also, for the relatively larger peptides, such as CGYGPKKKRKVGG and DRVYIHPFHLVIHN, the MLR deviations of –25.58% and –39.21%, respectively, should be compared with the deviations of –1.42% and –1.12% for the BPANN model. This results show that the structure-migration relationships for highly charged peptides follow nonlinear patterns. Despite the simplicity of the BPANN model developed by Jalali-Heravi and coworkers the average accuracy achieved by this model is ~2% [53].

To examine the robustness of this methodology, a 3-3-1 back-propagation artificial neural network model was developed using the same inputs as the previous model, which were the Offord's charge over mass term ($Q/M^{2/3}$), corrected steric substituent constant ($E_{s,c}$) and molar refractivity [54]. The data set consisted of 102 peptides with a larger range of size than that of the earlier report—up to 42 amino acid residues as compared to 14 amino acids in the initial study—which also included highly charged and hydrophobic peptides. The entire data set was obtained from the published result by Janini, Metral, and Isaaq [56]. The results of this model are compared with those obtained using multiple linear regressions model developed in this work and the multivariable model released by Janini et al. [56]. The present model exhibits better robustness than the MLR models in predicting CZE mobilities of a diverse data set at different experimental conditions. It is believed that the most important aim of these works was investigating the application of the generated models in simulation of the peptide maps of protein digests. Therefore, to test the utility of the models, the theoretical peptide maps of the digests of melittin, glucagons, and horse cytochrome C polypeptide and proteins were simulated.

To simulate an electropherogram, first, the ANN calculated electrophoretic mobilities were converted to migration times using the appropriate values for the experimental parameters as required by Eq. 9. For example, values of 37 and 30 cm were used for the total length of the column and injector-to-detector length, respectively. Also, a running voltage of 8 kV was used for this conversion. To simulate the peak of each theoretical fragment, it was assumed that the area for each peak is proportional to the number of peptide bonds. It is shown that, at 200 nm, the absorbance of a peptide is largely attributed to the peptide bonds, while the contribution of amino acids residues can be ignored [57, 58]. For the sake of convenience, the same peak width was used for each simulated peak in this work, which makes the peak height proportional to the number of peptide bonds.

Figure 6.13 demonstrates the experimental and BPANN simulated maps for the endoproteinase Lys-C digest of the peptide sequencing standard of melittin. The correct migration order of peptides and corresponding retention times agrees fairly well with the experimental electropherogram.

Next, glucagon, a polypeptide with 29 amino acid residues, was considered [54]. This polypeptide can be used as a control for proteolytic digestion, sequencing, and amino acid analysis. Janini and coworkers digested this protein with endoproteinase Glu-C with characteristic cleavage at the C-terminal of aspartic acid (D) and glutamic acid (E) residues [56]. Therefore, after a complete digestion, four fragments are expected for this protein. These fragments are listed in Table 6.7. Also, the values for the three descriptors, together with the calculated MLR and ANN values for the electrophoretic mobilities of glucagon fragments, are summarized in this table.

The simulated electropherogram is shown in Fig. 6.14. For comparison, the experimental and simulated electropherograms reported by Janini et al. are also shown in this figure. Inspection of Fig. 6.14 reveals an excellent matching between the line positions and a reasonable agreement between the relative heights of the experimental and simulated peaks. It seems that both models of multivariable and ANN overestimate the mobility for the FVQWLMNT fragment and, therefore, present a smaller value for the corresponding migration time. Validity of this conclusion depends on the accuracy of assignments of the peaks. However, one

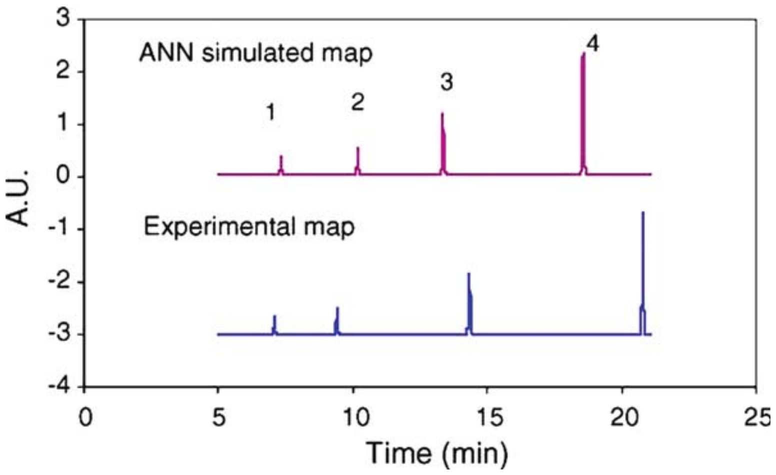


Fig. 6.13 Experimental and ANN simulated maps for the endoproteinase Lys-C digest of melittin

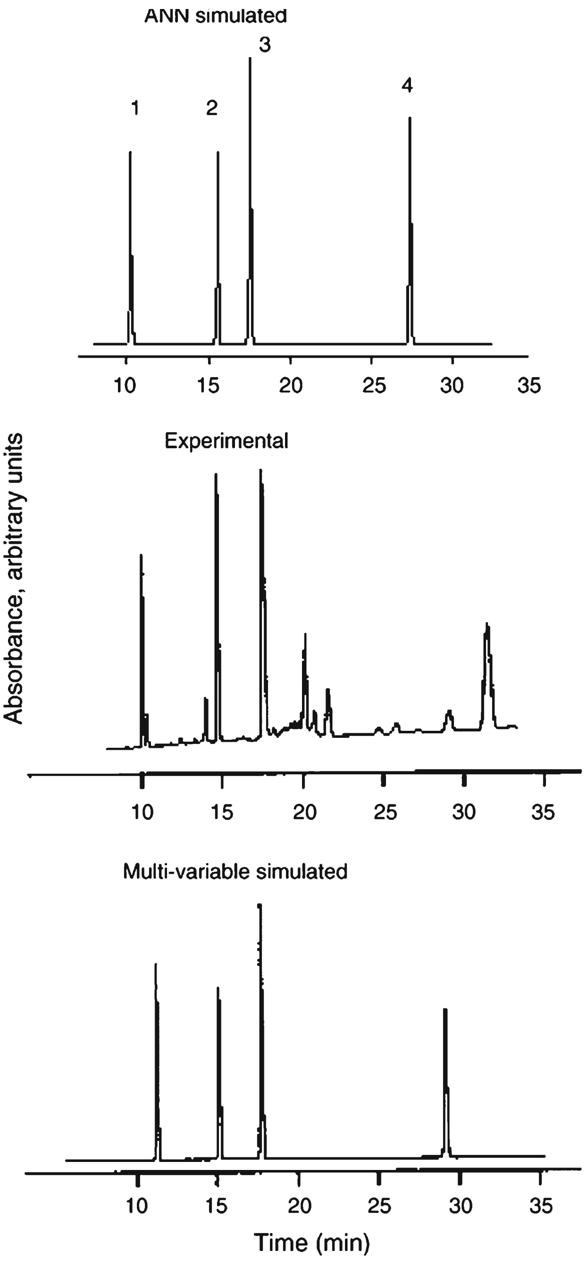


Fig. 6.14 Experimental, ANN and multivariable simulated electropherogram of glucagons

should consider the possibility of several shortcomings in the experiment, such as imperfect enzymatic digestion, impurity, and autolysis of the endoproteinase.

Finally, the relatively complex protein of horse cytochrome C with 104 amino acid residues was studied. This protein also was digested with endoproteinase

Lys-C with specificity of cleavage at the C-terminal of lysine residues [56]. The theoretical fragments of this protein, together with the values of the descriptors and corresponding MLR and ANN calculated mobilities, are demonstrated in Table 6.7. Figure 6.15 shows the experimental and ANN simulated electropherogram of cytochrome

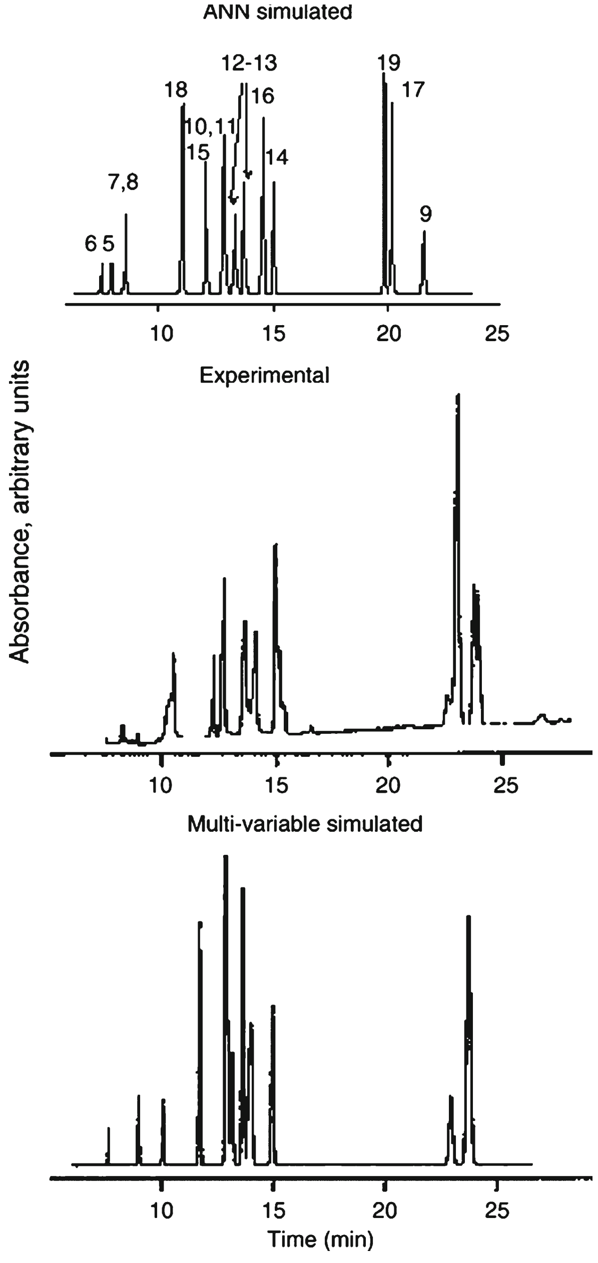


Fig. 6.15 Experimental, ANN and multivariable simulated electropherogram of cytochrome C

Table 6.7 Descriptor values together with MLR and ANN calculated mobilities of theoretical fragments of Lys-C digest of cytochrome ***C*** and Glu-C digest of glucagon

		Descriptors			MLR	ANN
No.	Peptide sequence	QM	$\boldsymbol{E}_{\mathrm{s,c}}$	MR	$\mu_{\mathrm{ef}} \times 10^5$	$\mu_{\mathrm{ef}} \times 10^5$
Glucagon digest						
1	SRRAQD	0.0338	–2.92	103.6	19.88	22.05
2	YSKYLD	0.0204	–4.32	127.1	14.60	14.41
3	HSQGTFTSD	0.0177	–4.18	123.6	13.56	12.75
4	FVQWLMNT	0.0081	–6.45	172.9	9.69	7.90
Cytochrome *C* digest						
5	GK	0.0530	–0.42	26.08	27.27	29.52
6	HK	0.0657	–1.28	48.84	32.02	31.39
7	NK	0.0450	–1.40	39.5	24.03	27.27
8	GGK	0.0450	–0.22	27.11	24.27	27.29
9	ATNE	0.0144	–1.93	48.2	12.32	10.18
10	GDVEK	0.0259	–2.91	68.8	16.60	17.67
11	GITWK	0.0257	–3.22	97.3	16.66	17.61
12	IFVQK	0.0249	–4.64	108.7	16.03	17.00
13	YIPGTK	0.0238	–3.26	103.3	15.97	16.48
14	MIFAGIK	0.0217	–5.17	124.0	14.79	15.02
15	CAQCHTVEK	0.0279	–4.14	144.4	17.64	18.86
16	TEREDLIAYLK	0.0223	–8.58	207.2	14.74	15.51
17	EETLMEYLENPK	0.0137	–8.42	224.3	11.68	10.92
18	TGPNLHGLFGRK	0.0322	–5.79	191.4	19.20	20.81
19	TGQAPGFTYTDANK	0.0135	–5.39	194.8	12.22	11.11

C. For comparison, the simulated electropherogram of this protein obtained by Janini et al. is included in Table 6.7. Inspection of Fig. 6.15 shows that the experimental electropherogram of cytochrome C has a striking similarity to the ANN simulated electropherogram. Almost each simulated peak has a counterpart in the experimental electropherogram, with good agreement between their migration times. The preliminary results are promising, but more research should be conducted in future to explore the use of a new series of sequence descriptors. Success in this step improves the capability of the models in simulating protein maps consisted of isomeric peptides.

4.4. Simulation of Spectra

The elucidation of the structure of a molecule requires the interpretation by the chemist of the collective spectral data derived from it. Nowadays, computer-based procedures have become a powerful tool to facilitate the interpretation of spectral data and, therefore, the enhancement of productivity. The neural network approach has attracted the attention of spectroscopists for establishing reliable correlations between different types of spectra (mass, ^{1}H-NMR, ^{13}C-NMR, etc.) and the chemical structure of the corresponding compounds. Among different analytical methods, nuclear magnetic resonance and mass spectrometry are powerful and versatile tools for the identification and structural analysis of complex compounds. Therefore, this section focuses on discussing the role of ANNs on simulation of mass and ^{13}C-NMR spectra.

Spectral simulation techniques for ^{13}C NMR spectroscopy can aid in the solution of complex structure elucidation problems and the verification of chemical shift assignment, particularly when suitable reference spectra are unavailable. Different basic approaches of ab initio, empirical, linear regression, and neural networks are used to calculate and predict the ^{13}C NMR chemical shift. One active area of research involves substituting linear regression with neural networks. Artificial neural networks have been reported for use in a few analytical chemical studies including ^{1}H NMR [59] and ^{13}C NMR spectroscopy [60–64]. To ensure the robustness and generality of an ANN model, various types of descriptors should be used as inputs. To develop more compact models, it is common to use selected subsets of descriptors instead of all possible descriptors. A PCAANN method for the calculation of ^{13}C chemical shifts of a number of lignin model compounds was developed [65]. The structure of these compounds is given in Fig. 6.16.

Lignins are the most abundant natural aromatic polymers on earth. The chemistry and analysis of these compounds has been the subject of much research, related to pulping, bleaching, and biodegradation . The chemical shifts for the carbon atoms in lignin model compounds fall in two distinct, separated subsets: carbons of the side chains (subset 1) and carbons in the benzene rings (subset 2). To prevent bias toward one type of carbon atom, only carbon atoms that are structurally unique are included in the development of the PCA. For the data set, 100 unique carbon atoms were identified. Of these, 73 carbon atoms were considered the training set, 20 carbon atoms were in the test set, and 7 carbon atoms were considered as control set. For the two distinct subsets, 30 of the unique carbons were in the first subsets, and the remaining 70 unique carbon atoms were in the benzene rings. A total of 62 descriptors were used for PCA investigation. Among these, five electronic descriptors show high loading values to cause largest effect on the ^{13}C chemical shifts of different carbons. These parameters are principally charge density surrounding a carbon center. The calculation of these descriptors is very easy and, nowadays, the tools for this type of calculation, that is, molecular packages such as MOPAC and HyperChem, are available in most laboratories [66, 67]. A BPANN model having three layers was created using these descriptors as its inputs. The specifications of the ANN models for the two subsets are summarized in Table 6.8.

This work shows that, in most cases, the results obtained by PCAANN are closer to the experimental values than those obtained using the ab initio method [65]. An example of the observed spectrum compared to the simulated one for the compound 1-(4-hydroxy-3,5-dimethoxyphenyl)ethanol as a control set is shown in Fig. 6.17. The RMS error for the prediction of this molecule is 2.759 ppm. The visual similarity between the two spectra is striking.

In another effort, the capabilities of the PCAANN algorithm has been studied in accurate simulation of ^{13}C NMR spectra of xanthones [68]. In this work, a

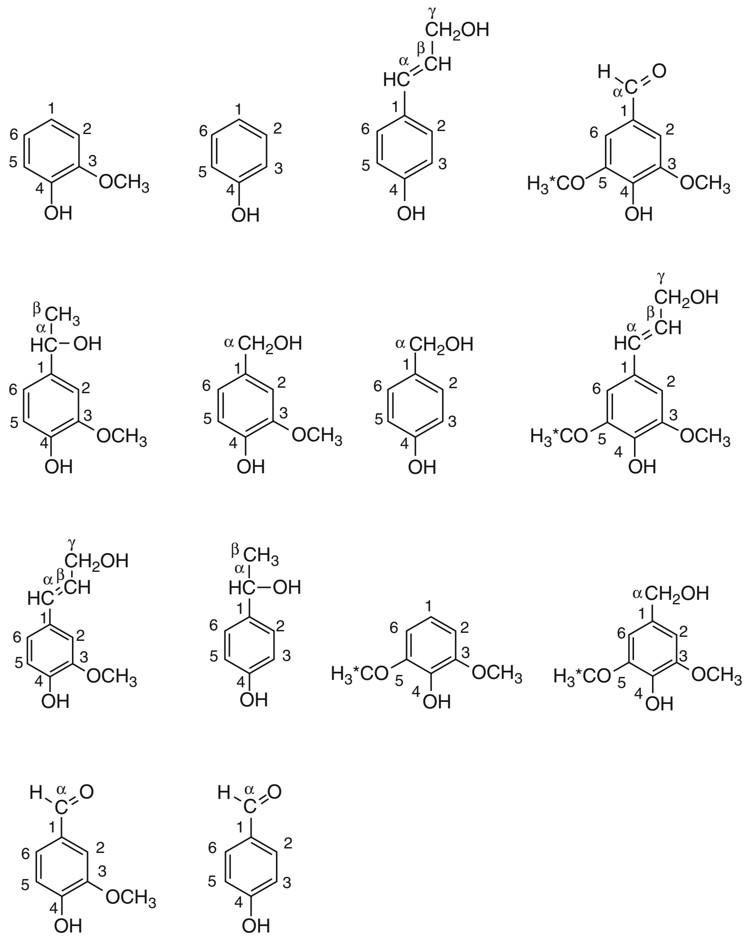


Fig. 6.16 The chemical structure of lignin model compounds

Table 6.8 Architecture of the ANNs and their specifications

	Subset 1	Subset 2
No. of nodes in the input layer	5	5
No. of nodes in the hidden layer	3	4
No. of nodes in the output layer	1	1
Momentum	0.4	0.6
Learning rate	0.6	0.4
Transfer function	Sigmoid	Sigmoid

PCA-ranking method is used for selecting the features affecting ^{13}C NMR chemical shifts for the first time. In this method, the descriptors were selected based on both high variances of the PCs and high correlations with the chemical shifts. Xanthones

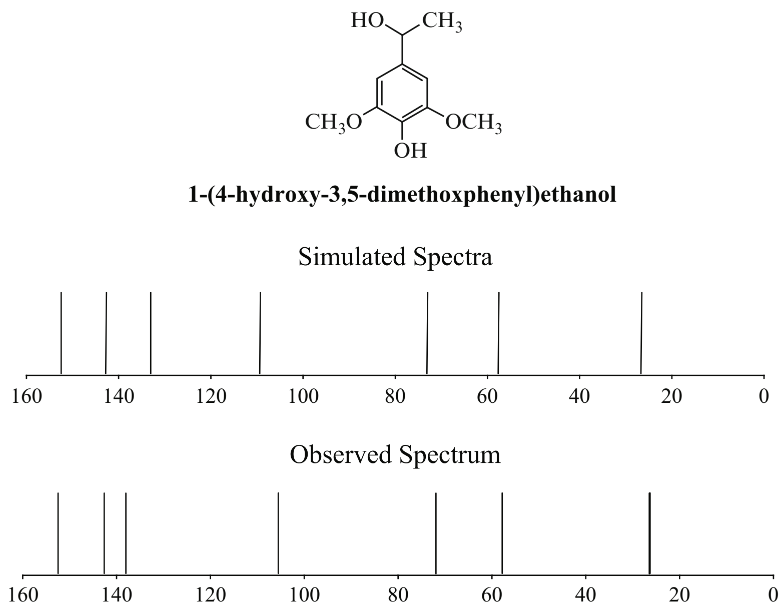


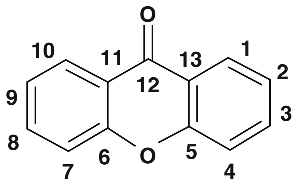
Fig. 6.17 Observed and simulated spectra of 1-(4-hydroxy-3,5-dimethoxyphenyl)ethanol

are secondary metabolites commonly occurring in some plants. These compounds are a group of heterocyclic molecules having a dibenzo-γ-pyrone carbon skeleton and differ from each other mainly in the extent and pattern of oxidation [69, 70].

The growing interest in these natural and synthetic compounds is due to their strong pharmacological and biological activities and their importance in chemotaxonomy. Pharmacological investigations have shown that xanthones have a wide range of bioactivities including antiinflammatory, antitumor, antibacterial, and antioxidant activities. Hydroxy- and methoxy-substituted xanthones are the most abundant natural and metabolic xanthones. Statistically 1,296 molecules of this type can exist. A total of 35 hydroxy- and methoxy-substituted xanthones containing 497 unique carbon centers were involved in this study. It should be noted that between the atoms having similar environments only one of them was used. Table 6.9 shows the compounds used in this work.

Thirty-two compounds were used as the calibration set for the formulation of the models and three compounds were used as the prediction set to examine the predictive ability of the models. To generate more suitable models, carbon atoms in the data set were divided into four subsets: subset 1, carbonyl carbon atoms; subset 2, central ring carbon atoms; subset 3, carbon atoms of the side rings except the fused carbon centers; and subset 4, methoxy groups carbon atoms. Different strategies, such as considering atom connectivity and distances related to different functional groups, were applied for choosing the subdivisions. Seven descriptors were selected by a PCA-ranking method and used as inputs for the ANN generation. In addition, to develop a general network, an indicator parameter was added for the atoms of

Table 6.9 List of xanthones used in reference [68]



Number	Substituent	Number	Substituent
1	3-OH	19	1,10-OH-3,7-OCH_3
2	2-OH	20	1,7-OH-7,8-OCH_3
3	1-OH	21	1,3,9-OH
4	4-OCH_3	22	1,8,10-OH-3,7,9-OCH_3
5	3-OCH_3	23	1-OH-3,7,8,9,10-OCH_3
6	2- OCH_3	24	1,10-OH-3,7,8,9-OCH_3
7	1-OCH_3	25	1,9-OH-3,10-OCH_3
8	3-OH-4-OCH_3	26	–
9	3-OCH_3-4-OH	27	7,8-OH-3-OCH_3
10[a]	2,3-OH-4-OCH_3	28	8-OH-2,7-OCH_3
11	2,3,4-OCH_3	29	4-OH
12	1,2-OH	30	2,3,4-OCH_3
13	1,2,3-OH	31	1,3,8-OH
14	1,3,9-OH	32	1,7,10-OH-3-OCH_3
15	1,3,7-OH	33*	1-OH-3,9,10-OCH_3
16	1,3,8,9-OH	34	1,8,10-OH-3,7,9-OCH_3
17	1,8,9-OH-3-OCH_3	35[a]	1,7,10-OH-3,8-OCH_3
18	1,3,10-OH		

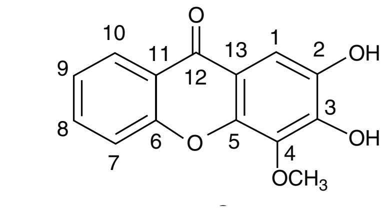
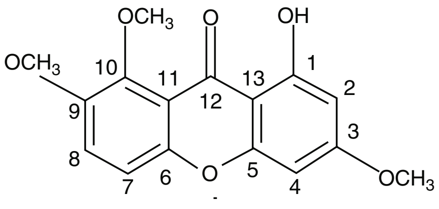
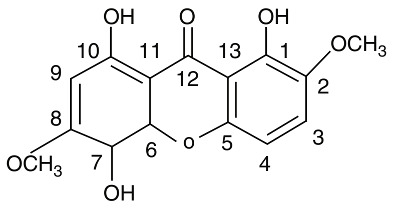
[a]Molecules included in the prediction set; remaining molecules are considered in the calibration set.

each subset in the data set. These indicator parameters were numbered 1 to 4 for corresponding subsets of 1 to 4, that is, for subset 1, 1; for subset 2, 2; and so forth. Therefore a total of eight parameters consisted of seven selected descriptors plus one indicator parameter (7 + 1) were used as inputs for the network generation. A back-propagation neural network with three layers was created using chemical shifts as the output layer. A high and similar R^2 value of 0.996, 0.993, and 0.998 for the calibration, test, and prediction sets, respectively, indicates the suitability and stability of the model in predicting the ^{13}C chemical shifts of xanthones.

To evaluate the predictive ability of the model, the chemical shifts of three molecules consisted of 45 carbon centers in the prediction set were calculated using the generated model as shown in Table 6.10. Also, a correlation coefficient of 0.998 and RMSE of 1.42 ppm for the prediction set indicates a good predictive ability for the ANN model. The simulated spectra of the three molecules, A, B, and C in the prediction set, are given in Fig. 6.18. An outstanding visual similarity exists between the simulated and observed spectra of all three molecules. The RMS errors for the prediction of these molecules were 1.11, 1.41, and 1.55 ppm, respectively.

Among different analytical methods, mass spectrometry is a powerful and versatile tool for the identification and structural analysis of complex compounds. The utility of mass spectroscopy arises from the fact that the ionization process generally produces

Table 6.10 Experimental and calculated values of chemical shifts for the molecules included in the prediction set

Prediction set	Carbon number	Experimental (ppm)	PCA-ranking-ANN predicted (ppm)	Δ(Exp. – Cal.)
 (A)	1	103.4	103.4	0.0
	2	144.1	144.3	–0.2
	3	144.9	147.6	–2.7
	4	135.3	134.9	0.4
	5	144.9	143.8	1.1
	6	155.4	154.7	0.7
	7	118.2	118.1	0.1
	8	134.5	134.5	0.0
	9	124.0	123.9	0.1
	10	125.8	127.1	–1.3
	11	120.7	121.1	–0.4
	12	174.8	174.4	0.4
	13	113.9	114.7	–0.8
	4-OCH_3	61.0	58.7	2.3
 (B)	1	162.7	162.3	0.4
	2	96.6	95.9	0.7
	3	166.0	166.7	–0.7
	4	91.7	92.4	–0.7
	5	156.4	156.3	0.1
	6	149. 9	150.8	0.9
	7	112.4	109.3	3.1
	8	121.0	120.9	0.1
	9	148.9	146.4	2.5
	10	147.7	147.1	0.6
	11	114.7	114.9	–0.2
	12	180.3	178.5	1.8
	13	103.0	102.7	0.3
	3-OCH_3	55.8	58.2	–2.4
	9-OCH_3	56.6	57.5	–0.9
	10-OCH_3	60.8	59.0	1.8
 (C)	1	147.9	149.2	–1.3
	2	140.4	142.7	–2.3
	3	124.1	122.8	1.3
	4	106.1	109.5	–3.4
	5	147.0	147.2	–0.2
	6	147.9	148.2	–0.3
	7	129.7	129.2	0.5
	8	160.0	157.5	2.5
	9	124.3	123.9	0.4
	10	146.4	148.93	–2.5
	11	121.12	120.7	0.4
	12	175.91	176.8	–0.9
	13	121.12	120.9	0.2
	2-OCH_3	56.7	57.5	–0.8
	8-OCH_3	56.7	57.6	–0.9

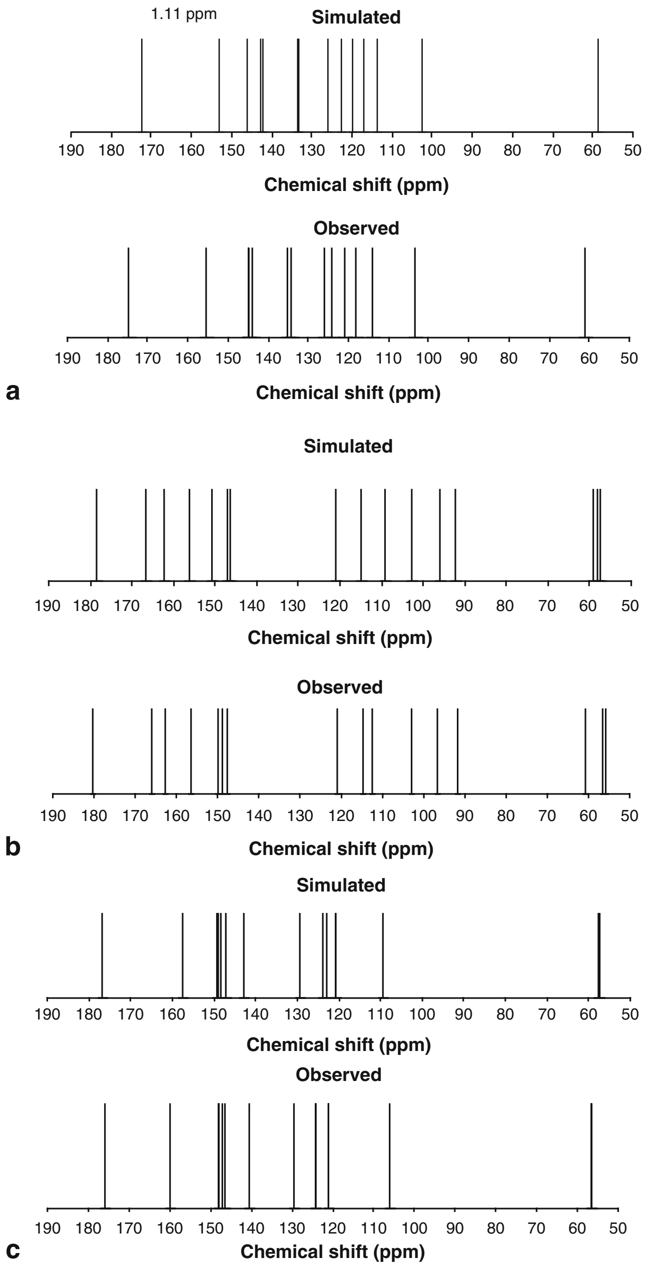


Fig. 6.18 Simulated and observed spectra of three molecules in the prediction set: (**a**) molecule A, (**b**) molecule B, and (**c**) molecule C

a family of particles whose mass distribution is characteristic of the parent species. Using the MLR technique for simulation of entire mass spectrum of an organic compound requires developing many linear equations, which is a time-consuming process. However, developing ANNs would solve this problem. It is intended that the inputs of the generated ANN be molecular structural descriptors and its outputs be mass spectral intensity at each *m*/*z* position.

Jalali-Heravi and Fatemi developed an ANN model to simulate the mass spectra of noncyclic alkanes and alkenes [71]. A collection of 262 organic compounds was chosen as data set, which is in the range of $C_{5–10}$ $H_{8–22}$. This data set consists of 117 noncyclic alkanes and 145 noncyclic alkenes. The compounds included in the data set are divided into groups according to the number of carbons of each compound and the type of molecule. The data set was randomly divided into two groups, a training set (236 compounds) and a prediction set (26 molecules). Two restrictions were applied for the selection of the *m*/*z* positions that would be used to simulate their intensities: (1) The intensities should be greater than 1% of TIC (as cutoff intensity); and (2) those *m*/*z* positions were considered in which more than 5% of molecules in the data set show a peak at them. A total of 44 *m*/*z* positions were chosen for which the intensities were predicted. A total of 75 topological descriptors were calculated for each compound. A stepwise deletion of terms procedure (Backward method) was used to select the important descriptors. It is found that 37 out of 75 descriptors are the most important ones for simulation of mass spectra [71]. The optimized parameters of the number of nodes in hidden layer, weights and biases learning rates momentum, and learning rate were 10, 0.2, 0.6, and 0.5, respectively. For the evaluation of the prediction power of the network, trained ANN was used to simulate the mass spectra of molecules included in the prediction set. Table 6.11 shows the correlation coefficient of the plots of the observed values against the ANN and MLR predicted values of TIC of all *m*/*z* positions of each molecule.

Figures 6.19 and 6.20 compare the experimental and predicted mass spectra of 4-ethyl-2-octene and 2,6-dimethylheptane as representatives of the noncyclic alkenes and alkanes, respectively. It is obvious from these figures that the predicted values of the *m*/*z* positions as well as their intensities are in good agreement with the observed values. Comparison of the simulated mass spectra of these compounds with those obtained by the experiment indicates that the model is able to predict accurately the main peaks of the mass spectra of these molecules. However, some differences between the observed and predicted values of *m*/*z* positions exist when either the intensity of the peak is low or the number of items of data at that position is small.

5. Notes

1. A characteristic difference between an expert system and a computer program is the distinct separation of the knowledge from the program code.
2. ANNs are a compact group of connected, ordered in layer elements able to process information. ANN-based approaches have the advantages of self-learning and abil-

Table 6.11 The molecules included in the prediction set, with their correlation coefficients of plots of observed TIC% against the calculated ones

Number	Compound	R_{ANN}	R_{MLR}	R_{mean}[a]
1	4-Methyl-1-pentene	0.91	0.69	0.54
2	2-Methylpentane	0.96	0.81	0.52
3	5-Methyl-1,4-hexadiene	0.86	0.58	0.10
4	4-Octene	0.93	0.91	0.42
5	2,3,3-Trimethylpentane	0.91	0.73	0.59
6	2,6-Dimethyl-3-heptane	0.76	0.83	0.62
7	5-Methyl-3-ethylhexane	0.88	0.90	0.54
8	2,2-Dimethylheptane	0.87	0.92	0.59
9	4-Ethyl-2-octene	0.89	0.84	0.53
10	2,6-Dimethyloctane	0.75	0.67	0.52
11	2-Methylbutane	0.92	0.88	0.56
12	3-Methyl-2-pentene	0.87	0.74	0.37
13	3,4-Dimethyl-1-pentene	0.82	0.76	0.55
14	2,2-Dimethylpentane	0.96	0.92	0.60
15	3-Ethylhexene	0.88	0.67	0.41
16	2,6-Dimethylheptane	0.97	0.88	0.55
17	3.5-Dimethylheptane	0.85	0.73	0.59
18	3-Methyloctane	0.90	0.85	0.59
19	2-Methyl-5-ethylheptane	0.90	0.88	0.59
20	2-Methyl-1,3-pentadiene	0.81	0.75	0.11
21	5-Methyl-1-hexene	0.82	0.85	0.58
22	5,5-Dimethyl-1-hexene	0.77	0.83	0.57
23	2-Methylhexane	0.81	—	0.44
24	2,2,4-Trimethyl-3-hexene	0.82	0.86	0.39
25	3-Methylpentane	0.80	—	0.59
26	2,3,4-Trimethylpentane	0.96	0.45	0.49

[a]The mean values of the correlation coefficient between ANN simulated spectrum of a molecule with all experimental MSw spectra of the molecules included in the prediction set.

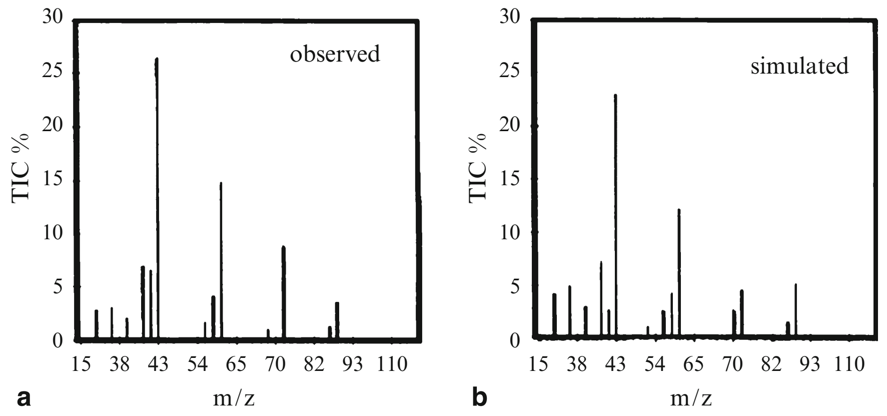


Fig. 6.19 The observed and ANN simulated mass spectra of 4-ethyl-2-octene: observed and simulated

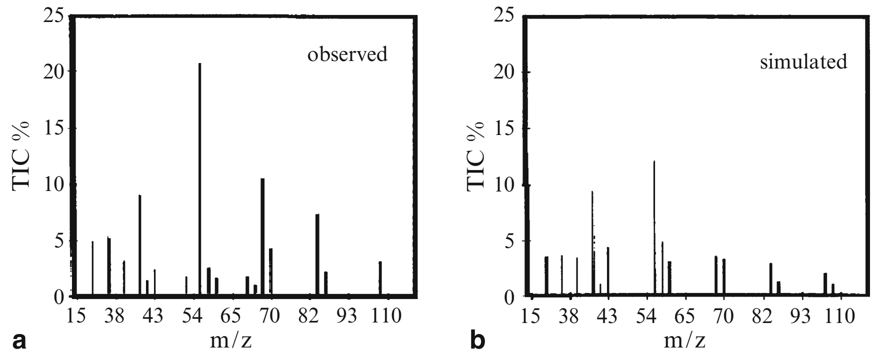


Fig. 6.20 The observed and ANN simulated mass spectra of 2,6-dimethylheptane: observed and simulated

ity to model complex data without need for a detailed understanding of the underlying phenomena. ANNs represents a modeling technique especially for data sets having nonlinear relationships, which are frequently encountered in analytical chemistry.

3. There are two approaches for training the networks: supervised and unsupervised. The aim in supervised learning is to predict one or more target values from one or more input variables. In this case, both inputs and outputs should be known. Unsupervised ANNs, or self-organizing maps, are good at finding relationships among complex sets of data. In unsupervised training, the network is provided with inputs but not the desired outputs.
4. A feedforward structure has no connection back from the output to the input neurons and does not keep a record of its previous output values. In contrast, feedback architecture has connections from output to input neurons.
5. A typical feedforward neural network with back-propagation has three layers: the input, the hidden, and the output layers. The input layer neurons receive data from a data file. The output neurons provide ANN's response to the input data. Hidden neurons communicate only with other neurons.
6. To prevent the overtraining in neural network approaches, the calculated mean square errors for the training and prediction sets should be plotted against the number of iterations. The training of the network should stop when the MSE starts to increase for the prediction set.
7. The genetic algorithm-kernel partial least square method can be considered a nonlinear feature selection method. The descriptors selected by GA-KPLS can be used as inputs for ANNs.
8. After training the network, it should be evaluated. The methods of leave-one-out cross-validation and leave-multiple-out cross-validation can be applied for investigating the consistency and robustness of the generated model.

9. The weight update functions (WUFs) play an important role in the reliability of the models using ANN. The Levenberg-Marquardt algorithm was shown to be the best WUF for predicting the $p\mathrm{IC}_{50}$ of heparanase inhibitors.
10. In the case of the MLR, the mean effect of each descriptor can be considered as measure of its role in predicting the $p\mathrm{IC}_{50}$ of the inhibitors. For the neural network, the sensitivity analysis can be incorporated to determine the importance or effect of the input variables on the network output.
11. The five comprehensive descriptors of the dipole moment—the highest occupied molecular orbital, the partial charge of the most negative atom, the partial charge of the most positive hydrogen, molecular mass and van der Waals volume—have successfully been used as inputs for developing ANNs for predicting retention behavior of organic compounds in different stationary phases.
12. Comparison of the results for relative response factors indicates the superiority of neural networks over that of MLRs and demonstrates a nonlinear behavior for RRFs of all types of GC detection systems, such as TCD, FID, Ar-PID, Kr-PID, and ECD.
13. The key parameter for separation of peptides, especially in low ionic strength buffers, is their electrophoretic mobilities. This parameter can be converted to migration time and a CZE electropherogram can be simulated using a Gaussian function. Therefore, calculation or prediction of this parameter is very useful in peptide mapping studies.
14. The PCA-ranking method can be used for selecting the features affecting ^{13}C NMR chemical shifts. In this method, the descriptors are selected based on both high variances of the PCs and high correlations with the chemical shifts.

References

1. McCulloch WS, Pitts W (1943) A statistical consequence of the logical calculus of nervous nets. Bull Math Biolophys 5:115–113.
2. McCulloch WS, Pitts W (1947) The limiting information capacity of a neuronal link. Bull Math Biolophys 9:127–147.
3. Hebb DO (1949) The organization of behavior. Wiley, New York.
4. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Nat Acad Sci 79:2554–2567.
5. Haykin S (1994) Neural network. Prentice-Hall, Englewood Cliffs, NJ.
6. Zupan J, Gasteiger J (1999), Neural networks in chemistry and drug design. Wiley-VCH, Weinheim.
7. Bose NK, Liang P (1996), Neural networks, fundamentals. McGraw-Hill, New York.
8. Anker SL, Jurs PC (1992) Application of neural networks in structure-activity relationships. Anal Chem 64:1157–1165.
9. Hagan MT, Demuth HB, Beal M (1996) Neural network design. PWS Publishing, Boston.
10. Zupan J, Gasteiger J (1993) Neural networks for chemists, an introduction. VCH, Weinheim.
11. Hopke PK, Song X (1997) Source apportionment of soil samples by the combination of two neural networks based on computer. Anal Chim Acta 348:375–386.
12. Lippmann RP (1987) IEEE ASSP (April 4).
13. Tusar M, Zupan J (1990) Software development in chemistry 4. In: Gasteiger J (ed) Neural networks,. Springer, Berlin, pp. 363–376.

14. Kohonen T (1982) Analysis of a simple self-organizing process. Biol Cybernetics 43:59–69.
15. Kohonen T (1988) Self-organization and associate memory. Springer, Berlin.
16. Zupan J (1989) Algorithms for chemists. Wiley, Chichester, UK, pp. 257–262.
17. Todeschini R, Consonni V, Pavan M (2002) Dragon software, version 2.1, via pisani, 13-20124 Milan, Italy.
18. Jalali-Heravi, M, Parastar F (2000) Use of artificial neural networks in a QSAR study of anti-HIV activity for a large group of HEPT derivatives. J Chem Inf Comput Sci 40:147–154.
19. Luco JM, Ferretti FH (1997) QSAR based on multiple linear regression and PLS methods for the anti-HIV activity of a large group of HEPT derivatives. J Chem Inf Comput Sci 37:392–401.
20. Tanaka H, Takashima H, Ubasawa M, Sekiya K, Nitta I, Baba M, Shigeta S, Walker RT, Clercq ED, Miyasaka T (1992) Synthesis and antiviral activity of deoxy analogs of 1-(2-Hydroxyethoxy)methyl]-6-(phenylthio)thymine (HEPT) as potent and selective anti-HIV-1 agents. J Med Chem 35:4713–4719.
21. Hansch C, Muir R M, Fujita T, Maloney PP, Geiger F, Streich M (1963) The correlation of biological activity of plant growth regulators and chloromycetin derivatives with Hammett constants and partition coefficients. J Am Chem Soc 85:2817– 2824.
22. Hansch C, Hoekman D, Gao H (1996) Comparative QSAR: toward a deeper understanding of chemicobiological interactions. Chem Rev 96:1045–1075.
23. Katritzky AR, Labanov VS, Karelson M (Copyright 1994–1995) CODESSA 2.0, Comprehensive descriptors for structural and statistical analysis. University of Florida, Gainesville.
24. Jalali-Heravi M, Kyani A (2007) Application of genetic algorithm-kernel partial least square as a novel nonlinear feature selection method: activity of carbonic anhydrase II inhibitors. Eur J Med Chem 42:649–659.
25. Agrawal VK, Bano S, Supuran CT, Khadikar PV (2004) QSAR study on carbonic anhydrase inhibitors: aromatic/heterocyclic sulfonamides containing 8-quinoline-sulfonyl moieties, with topical activity as antiglaucoma agents. Eur J Med Chem 39:593–600.
26. Melagraki G, Afantitis A, Sarimveis H, Igglessi-Markopoulou O, Supuran CT (2006) QSAR study on para-substituted aromatic sulfonamides as carbonic anhydrase II inhibitors using topological information indices. Bioorg Med Chem 14:1108–1114.
27. Clare BW,. Supuran CT (1999) Carbonic anhydrase inhibitors. Part 61. Quantum chemical QSAR of a group of benzenedisulfonamides. Eur J Med Chem 34:463–474.
28. Jalali-Heravi M, Garkani-Nejad Z (2001) Prediction of electrophoretic mobilities of sulfonamides in capillary zone electrophoresis using artificial neural networks. J Chromatogr A 927:211–218.
29. Vlodavsky I, Goldshmidt O, Zcharia E, Metzger S, Chajek-Shaulb T, Atzmon R, Guatta-Rangini Z, Friedmann,Y (2001) Molecular properties and involvement of heparanase in cancer progression and normal development. Biochimie 83:831–839.
30. Bernfield M, Götte M, Woo Park P, Reizes O, Fitzgerald ML, Lincecum J, Zako, M (1999) Functions of cell surface heparin sulfate proteoglycans. Ann Rev Biochem 68:729–777.
31. Vlodavsky I, Friedmann Y (2001) Molecular properties and involvement of heparanase in cancer metastasis and angiogenesis. J Clin Invest 108:341–347.
32. Courtney SM, Hay PA, Buck RT, Colville CS, Porter DW, Scopes DIC, Pollard FC, Page MJ, Bennett JM, Hircock ML, McKenzie EA, Stubberfield CR, Turner PR (2004) 2,3-Dihydro-1,3-dioxo-1H-isoindole-5-carboxylic acid derivatives: a novel class of small molecule heparanase inhibitors. Bioorg Med Chem Let 14:3269–3273.
33. Courtney SM, Hay PA, Buck RT, Colville CS, Phillips DJ, Scopes DAC, Pollard FC, Page MJ, Bennett JM, Hircock ML, McKenzie EA, Bhaman M, Felix R, Stubberfield CR, Turner PR (2005) Furanyl-1,3-thiazol-2-yl and benzoxazol-5-yl acetic acid derivatives: novel classes of heparanase inhibitor. Bioorg Med Chem Let 15:2295–2299.
34. Jalali-Heravi M, Asadollahi-Baboli M, Shahbazikhah P (2007) QSAR study of heparanase inhibitors activity using artificial neural networks and Levenberg-Marquardt algorithm. Eur J Med Chem (in press).

35. Jalali-Heravi M, Fatemi MH (1998) Prediction of flame ionization detector response factor using an artificial neural network. J Chromatogr A 825:161–169.
36. Anker LS, Jurs PC, Edwards PA (1990) Quantitative structure-retention relationship studies of odor-active aliphatic compounds with oxygen-containing functional groups. Anal Chem 62:2676–2684.
37. Jalali-Heravi M, Parastar F (2000) Development of comprehensive descriptors for multiple linear regression and artificial neural network modeling of retention behaviors of a variety of compounds on different stationary phases. J Chromatogr A 903:145–154.
38. Kollie TO, Poole CF, Abraham MH, Whiting, GS (1992) Comparison of two free energy of solvation models for characterizing selectivity of stationary phases used in gas-liquid chromatography. Anal Chim Acta 259:1–13.
39. Jalali-Heravi M, Fatemi MH (2001) Artificial neural network modeling of Kováts retention indices for noncyclic and monocyclic terpenes. J Chromatogr A 915:177–183.
40. Jalali-Heravi M, Garakani-Nejad Z (2002) Use of self-training artificial neural networks in modeling of gas chromatographic relative retention times of a variety of organic compounds. J Cromatogr A 945:173–184.
41. Wentworth WE, Helias, N, Zlatkis A, Chen ECM, Stearns SD (1998) Multiple detector responses for gas chromatography peak identification. J Chromatogr A 795:319–347.
42. Jalali-Heravi M, Kyani A (2004) Use of computer-assisted methods for the modeling of the retention time of a variety of volatile organic compounds: a PCA-MLR-ANN approach. J Chem Inf Comput Sci 44:1328–1335.
43. Jalali-Heravi M, Fatemi MH (2000) Prediction of thermal conductivity detection response factors using an artificial neural network. J Chromatogr A 897:227–235.
44. Jalali-Heravi M, Garakani-Nejad Z (2002) Prediction of relative response factors for flame ionization and photoionization detection using self-training artificial neural networks. J Chromatogr A 950:183–194.
45. Jalali-Heravi M, Noroozian E, Mousavi M (2004) Prediction of relative response factors of electron-capture detection for some polychlorinated biphenyls using chemometrics. J Chromatogr A 1023:247–254.
46. Link AJ, Eng J, Schieltz DM, Carmack E, Mize GJ, Morris DR, Arvik BM, Yates JR (1999) Direct analysis of protein complexes using mass spectrometry. Nat Biotechnol 17:676–682.
47. Wasburn MP, Wolters D, Yates JR (2001) Large-scale analysis of the yeast proteome by multidimensional protein identification technology. Nat Biotechnol 19:242–247.
48. Grossman PD, Colburn JC, Lauer HH (1989) A semiempirical model for the electrophoretic mobilities of peptides in free-solution capillary electrophoresis. Anal Biochem 179, 28–33.
49. Offord RE (1996) Electrophoretic mobilities of peptides on paper and their use in the determination of amide groups. Nature 211:591–593.
50. Compton BJ (1991) Electrophoretic mobility modeling of proteins in free zone capillary electrophoresis and its application to monoclonal antibody microheterogeneity analysis. J Chtomatogr A 599:357–366.
51. Cifuentes A, Poppe H (1997) Behavior of peptides in capillary electrophoresis: effect of peptide charge, mass and structure. Electrophoresis 18:2362–2376.
52. Janini GM, Mertal CJ, Issaq HJ, Muschik GM (1999) Peptide mobility and peptide mapping in capillary zone electrophoresis: experimental determination and theoretical simulation. J Chromatogr A 848:417–433.
53. Jalali-Heravi M, Shen Y, Hassanisadi M, Khaledi MG (2005) Prediction of electrophoretic mobilities of peptides in capillary zone electrophoresis by quantitative structure-mobility relationships using the Offord model and artificial neural networks. Electrophoresis 26:1874–1885.
54. Jalali-Heravi M, Shen Y, Hassanisadi M, Khaledi MG (2005) Artificial neural network modeling of peptide mobility and peptide mapping in capillary zone electrophoresis. J Chromatogr A 1096:58–68.
55. Taft Jr., RW (1956) In: NewmanMS (ed) Organic chemistry. Wiley, New York.

56. Janini GM., Metral CJ, Issaq HJ (2001) Peptide mapping by capillary zone electrophoresis: how close is theoretical simulation to experimental determination. J Chromatogr A 924:291–306.
57. Scopes RK (1974) Measurement of protein by spectrophotometry at 205 nm. Anal Biochem 59:277–282.
58. Herold M, Ross GA, Grimm R, Heiger DN (1996) In: Altria KD (ed) Capillary electrophoresis guidebook: principles, operation, and applications, methods in molecular biology. Humana Press, Totowa, NJ.
59. Aires-de-Sousa J, Hemmer MC, Gasteiger, J (2002) Prediction of ^{1}H NMR chemical shifts using neural networks. Anal Chem 74:80–90.
60. Ball JW, Anker LS, Jurs PC (1991) Automated model selection for the simulation of carbon-13 nuclear magnetic resonance spectra of cyclopentanones and cycloheptanones. Anal Chem 63:2435–2442.
61. Jalali-Heravi M, Mousavi M (1995) Simulation of ^{13}C NMR. spectra of nitrogen-containing aromatic compounds. Aust J Chem 48:12671275.
62. Meiler J, Will M (2001) Automated structure elucidation of organic molecules from ^{13}C NMR spectra using genetic algorithms and neural networks. J Chem Inf Comp Sci 41:1535–2546.
63. Meiler J, Maier W, Will M, Meusinger R (2002) Using neural networks for ^{13}C NMR chemical shift prediction—comparison with traditional methods. J Mag Reson 157:242–252.
64. Meiler J, Will M (2002) Genius: a genetic algorithm for automated structure elucidation from ^{13}C NMR spectra. J Am Chem Soc 124:1868–1870.
65. Jalali-Heravi M, Masoum S, Shahbazikhah P (2004) Simulation of ^{13}C nuclear magnetic resonance spectra of lignin compounds using principal component analysis and artificial neural networks. J Mag Reson 171:176–185.
66. HyperChem, available at www.hyper.com/products/evaluation.html.
67. MOPAC, available at www.psc.edu/general/software/package/mopac/mopac.html.
68. M, Jalali-Heravi P, Shahbazikhah BS, Zekavat M Ardejani (2007) Principal component analysis-ranking as a variable selection method for the simulation of ^{13}C nuclear magnetic resonance spectra of xanthones using artificial neural networks. QSAR Comb Sci (in press).
69. Peres V, Nagem TJ (1997) Trioxygenated naturally occurring xanthones. Phytochemistry 44:191–214.
70. Peres V, Nagem TJ, Faustino de Oliveira F (2000) Tetraoxygenated naturally occurring xanthones. Phytochemistry 55:683–710.
71. Jalali-Heravi M, Fatemi MH (2000) Simulation of mass spectra of noncyclic alkanes and alkenes using artificial neural network. Anal Chim Acta 415:95–103.

Chapter 7
Application of Artificial Neural Networks for Decision Support in Medicine

Brendan Larder, Dechao Wang, and Andy Revell

Abstract The emergence of drug resistant pathogens can reduce the efficacy of drugs commonly used to treat infectious diseases. Human immunodeficiency virus (HIV) is particularly sensitive to drug selection pressure, rapidly evolving into drug resistant variants on exposure to anti-HIV drugs. Over 200 mutations within the genetic material of HIV have been shown to be associated with drug resistance to date, and complex mutational patterns have been found in HIV isolates from infected patients exposed to multiple antiretroviral drugs. Genotyping is commonly used in clinical practice as a tool to identify drug resistance mutations in HIV from individual patients. This information is then used to help guide the choice of future therapy for patients whose drug regimen is failing because of the development of drug resistant HIV. Many sets of rules and algorithms are available to predict loss of susceptibility to individual antiretroviral drugs from genotypic data. Although this approach has been helpful, the interpretation of genotypic data remains challenging. We describe here the development and application of ANN models as alternative tools for the interpretation of HIV genotypic drug resistance data.

A large amount of clinical and virological data, from around 30,000 patients treated with antiretroviral drugs, has been collected by the HIV Resistance Response Database Initiative (RDI, www.hivrdi.org) in a centralized database. Treatment change episodes (TCEs) have been extracted from these data and used along with HIV drug resistance mutations as the basic input variables to train ANN models. We performed a series of analyses that have helped define the following: (1) the reliability of ANN predictions for HIV patients receiving routine clinical care; (2) the utility of ANN models to identify effective treatments for patients failing therapy; (3) strategies to increase the accuracy of ANN predictions; and (4) performance of ANN models in comparison to the rules-based methods currently in use.

Keywords Artificial neural networks, decision support, HIV infection, clinical response, antiretroviral therapy

1. Introduction

The evaluation of artificial neural networks (ANNs) in medicine has increased steadily over the last decade in many fields. ANNs have been developed as tools to aid clinical decision making by addressing problems of clinical diagnosis, disease staging, image analysis, and survival prediction in a wide range of medical fields, including oncology [1–4] cardiology and hematology [5–8], and critical care [9–11]. It is not difficult to imagine why this is the case. The increasing use of patient (and pathogen) genetic information in clinical diagnostics and therapeutic decision making has considerably increased the complexity of the issues. As a consequence, large amounts of complex biological information with many variables often have to be assimilated and processed if optimal diagnostic and therapeutic decisions are to be made. These situations ideally lend themselves to the application of ANNs, which are good decision-making systems in situations where data can be complex, relatively imprecise, and not amenable to the development of rules or algorithms. To date, the application of ANNs to decision making in the infectious disease field has been relatively uncommon. One of the first examples where this is being pioneered is in the treatment of human immunodeficiency virus (HIV) infection. This chapter initially reviews the application of ANNs in a number of clinical areas and then focuses on the potential use of ANNs in the HIV therapeutic field.

As indicated, ANNs have been used for decision making in a number of areas of medicine. One area in which ANNs have been applied particularly extensively is oncology, largely because of the often complex nature of cancer etiology, development, and treatment, coupled with the importance of early and accurate diagnosis. ANNs have been developed to aid cancer diagnosis, disease staging, and prognosis. This has been the topic of a number of reviews [12–15]. PAPNET is one of the best-known, commercially available ANN-based systems actually in clinical practice to date. This system gained FDA approval for clinical use in 1996 and remains one of only a very few medical computer-based applications that have been approved so far. PAPNET is used as a diagnostic tool for assisted screening of Pap (cervical) smears. It has been demonstrated to be more sensitive to the detection of false negatives than conventional repeated manual microscopic examination of Pap smears for abnormal cervical cells [16], which reduces the need for costly biopsies. However, it is worth noting that PAPNET analysis is also associated with an increase in false positives [17]. ANNs have also been extensively evaluated as tools to aid screening for prostate cancer, for which prostate-specific antigen (PSA) levels are currently widely used as a somewhat imperfect diagnostic indicator. In particular, ANNs that incorporate multiple variables were developed in an effort to improve the prediction of biopsy outcome [18, 19]. The predictive power of these ANNs was found to be significantly better than conventional methods of assessing the need for biopsy; and it appears that, like PAPNET, screening for prostate cancer with ANN models could significantly reduce the number of unnecessary biopsies. Further evaluation of ANNs using a variety of input variables has shown them to be highly accurate for the diagnosis, staging, imaging, and prognosis of prostate cancer; and it seems likely that they will become important tools in the clinical management of this disease in the future [3].

ANNs have also been evaluated as diagnostic or prognostic tools for many other cancers, including skin, breast, lung, bladder and liver cancer, and leukemia (reviewed in [14]). In terms of image and signal analysis, there is an ever-increasing list of examples where the pattern recognition capacity of ANNs has been applied to aid analysis of radiographs, ECTs, MRIs, CTs, ECGs, magnetic resonance spectra (MRS), and the like (reviewed in [20, 21]).

We recently investigated the application of ANN technology in the field of infectious diseases, where a major problem is the emergence of drug-resistant pathogens. In particular, we used ANN to investigate HIV drug resistance, one of the most critical threats to successful management of HIV infection. Our preliminary investigations of the complex genetic basis of HIV drug resistance led to the identification of previously unidentified mutations within the HIV protease gene (HIV protease activity is essential for viral replication) associated with drug resistance. This early success indicated that there might be considerable utility in ANN models as treatment decision-making support tools [22].

Using HIV as a model system, we describe here our approach to defining the utility of ANN models to aid in the prediction of clinical response to antiretroviral drugs. These drugs control HIV infection by suppressing HIV replication (reducing viral load) but do not eradicate the virus from the host. As such, HIV infection is a chronic disease that requires lifelong therapy to maintain suppression of replication and viral load in a patient. HIV infection is treated with combinations of antiretroviral drugs often referred to as HAART (highly active antiretroviral therapy). This typically comprises three antiretroviral agents from two drug classes that have different modes of action [23]. In many patients, the sequential accumulation of genetic mutations within the HIV genome that confer resistance to these agents leads to treatment failure within a few years [24]. The number of mutations that accumulate in a particular drug resistant strain and patterns or clustering of these mutations can be highly variable. Indeed, the evolution of drug-resistant HIV variants within a single host during exposure to antiretroviral therapy represents one of the most dynamic examples of viral adaptation described to date.

Currently, over 20 antiretroviral drugs are available from four drug classes, as well as a number of new drug candidates in clinical development. As more and more drugs become available and our understanding of the complexities of the genetic basis of HIV drug resistance increases, the selection of drugs following treatment failure is becoming increasingly challenging and important. Overlapping resistance profiles of drugs within the same drug class can lead to loss of susceptibility to multiple drugs and, so, severely limit the number of remaining active combinations. Techniques to assess loss of susceptibility to antiretroviral drugs in HIV infection (drug-resistance testing) as a means to guide the choice of new combinations of active agents have been increasingly incorporated into standard clinical care [23]. Two general approaches are taken to resistance testing: phenotyping and genotyping.

Phenotyping directly measures the susceptibility of HIV to a particular drug, while genotyping identifies the presence of mutations in the genetic material of the virus that are known to be associated with decreased drug susceptibility. Genotyping is most commonly used in routine clinical practice, primarily because it is quicker and less expensive. Many sets of rules or algorithms have been developed to interpret genotypic resistance data. However, with over 200 mutations affecting drug susceptibility, these methods generally do not address the complex relationship between genotype and response. At best, they provide an imperfect guide to whether HIV in a particular patient is broadly resistant or sensitive to individual drugs, and they cannot provide an indication of the likely degree of response of the patient to different combinations of agents. As such, the interpretation of the complex mutational patterns that commonly develop in HIV from patients failing antiretroviral therapy remains a challenge.

2. Materials

The iterative, nonlinear data processing achieved using ANNs accommodates multiple predictors and hidden interactions between variables to produce an appropriate result. This feature lends itself well to the interpretation of genotypic HIV drug-resistance data and the prediction of clinical response to new drug combinations. The HIV Resistance Response Database Initiative is a not-for-profit organization that aims to improve the clinical management of HIV infection by developing and making freely accessible a large clinical database and bioinformatic techniques to define the relationships between HIV resistance and virological response to treatment. The goal is to improve the interpretation of drug-resistance information and ultimately the quality of treatment decision making. The development of the database is an international collaborative initiative; data from more than 30,000 HIV patients have already been provided by a variety of private and public research groups. Our approach has therefore involved relating substantial HIV resistance data *directly* to the virological response of patients to different combinations of antiretroviral drugs in clinical practice by harnessing ANNs. We believe this approach can overcome many of the shortcomings of current HIV genotype interpretation techniques.

HIV causes illness by attacking cells of the immune system, known as *CD4 cells*, that are essential for protecting the body against infections and disease. As HIV disease progresses, the amount of virus in the bloodstream (viral load) increases and the number of CD4 cells (CD4 cell count) drops, leaving the host susceptible to infection by other pathogens. However, being HIV positive does not necessarily mean antiretroviral treatment is required immediately. Viral load and CD4 cell count are used to gauge how severe the infection has become and how well the immune system is continuing to work. This information is then used to decide when to initiate antiretroviral treatment. These indicators continue to be monitored and are used (particularly viral load) to trigger treatment changes whenever the virus develops resistance and begins to escape therapeutic control. Blood samples taken prior to initiation or change of therapy are also used to determine the genotype (commonly obtained by DNA sequence analysis of HIV genetic material). We collected these data plus viral load measurements taken at various times (up to 48 weeks) *after* the initiation of a new combination therapy. All these clinical and virological data are stored in a customized Oracle database. Additional information, such as the patient's previous treatments, is also collected where this is available.

3. Methods

We performed a series of analyses that have helped define the following: (1) the reliability of ANN predictions for HIV patients receiving routine clinical care; (2) the utility of ANN models to identify potentially effective treatments for patients failing therapy; (3) strategies to increase the accuracy of ANN predictions; and (4) performance of ANN models in comparison to the rules-based methods currently in use.

3.1. The Use of "Real-Life" Clinical Data to Model Treatment Response to Antiretroviral Agents

Initial power calculations were performed to estimate the sample size required to obtain a high predictive accuracy based on complex or simplified data input parameters [25]. Three-layer ANNs were constructed with the output of the models being follow-up viral load ($\log_{10}$ copies/ml) (*see* **Note 1**) on treatment. We found a database size of thousands of patient samples was probably required to obtain very high predictive accuracy [25]. It is also important, however, that the data sets are of good quality, contemporary, and encompass a wide range of treatments (drug combinations) and drug-resistance patterns.

Due to predefined eligibility criteria, clinical trial data by nature is subject to biases that do not necessarily reflect the "real-life" situation of HIV patients receiving routine clinical care. For ANN predictions to be useful in medicine, they must produce accurate predictions for individuals in a "real-life" setting. We assessed the accuracy of predictions by our ANN models for "real" patients by analyzing data from 351 patients from an HIV treatment cohort [26]. These patients had multiple genotypes, irregular viral load measurements, and had undergone treatment changes due to the emergence of drug-resistant HIV. Throughout our studies, the unit of data used for modeling consists of a treatment change episode (TCE). A TCE includes all the relevant data for a patient at or around the time when he or she was started on a new antiretroviral drug combination (Fig. 7.1). TCEs were selected from patients having a genotype up to 12 weeks before treatment change and baseline viral load up to 16, 12, or 8 weeks before treatment change. Follow-up viral load was within 4–40 weeks after treatment change. A total of 652, 602, and 539 TCEs, respectively, obtained using the three viral load windows, were used for neural network training (10% of each was partitioned for independent testing).

We found the model with the narrowest baseline viral load window (eight weeks) most accurately predicted viral load response ($r^2 = 0.55$, $p < 0.05$) (*see* **Note 2**). This model was used to predict response to 30 common antiretroviral regimens for each of 118 cases of actual treatment failure, TCEs where the viral load increased after a change to a new triple antiretroviral drug regimen guided by a genotype (3,540 evaluations). In 116 cases (98%), the model identified one or more regimens that were predicted to reduce viral load. Based on the 87% accuracy of prediction of viral load trajectory, this translates into 101 cases for which the model identified potentially beneficial treatment regimens for patients who experienced treatment failure in the clinic.

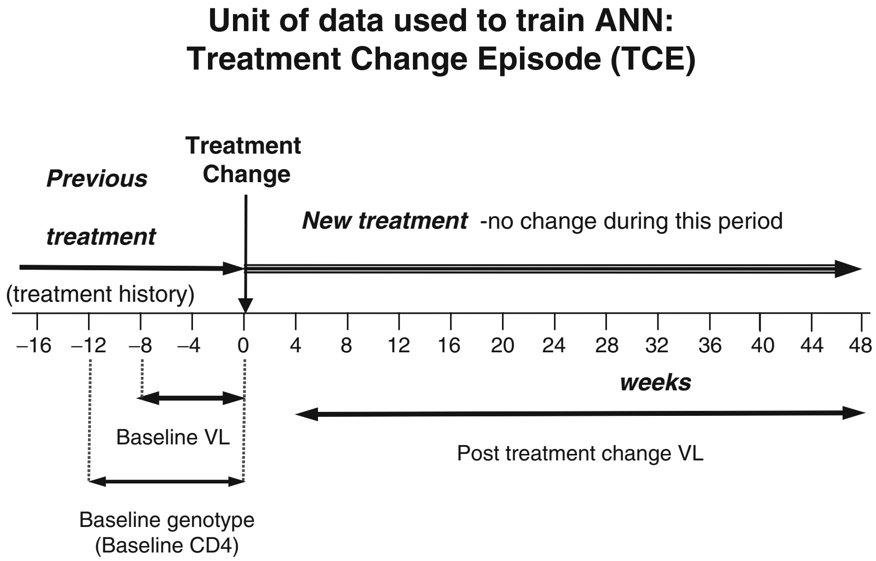


Fig. 7.1 Unit of data used to train ANN: treatment change episode (TCE). The ANN input training variables that comprise the treatment change episode. The basic elements of a TCE consist of the baseline resistance genotype, baseline and on-treatment viral loads, drugs in the new regimen, and the time of the on-treatment viral load. Other input variables, such as baseline CD4 cell count and treatment history, have been added to enhance the performance of ANN models

3.2. Identification of Effective Treatments for Patients Failing Salvage Therapy

Choosing an effective antiretroviral regimen for patients who have experienced multiple treatment failures, despite having their treatment changed according to current clinical practice guidelines, is particularly challenging. We examined the potential of our ANN models to identify potentially active alternative treatment regimens for highly treatment-experienced patients, using data from 511 patients from two clinical cohorts [27]. We obtained 747 TCEs with a genotype up to 12 weeks and a viral load up to 8 weeks before treatment change and a follow-up viral load within 4–40 weeks after treatment change. Thirteen ANN models were trained (using 690 randomly selected TCEs) and evaluated, using the remaining 57 TCEs as independent test sets. Then, 108 failure TCEs were identified (baseline viral load of at least four 4 logs and an increase in viral load following treatment change). The most accurate ANN model was used to predict response to 38 common treatment regimens for each failure case (4,104 individual virtual response predictions).

The correlation between the predicted and actual viral load change for the four ANN models gave a mean r^2 value of 0.65 and mean correct viral load trajectory prediction rate of 78%. The best model ($r^2 = 0.76$) identified one or more regimens that were predicted would reduce viral load in all 108 failure cases. The median actual change in viral load for these cases was +0.41 logs, whereas the predicted median change using the best regimens according to the ANN was −2.6 logs. These data suggest that the application of ANNs for the interpretation of genotypic resistance may be particularly relevant in the clinical management of extensively treated HIV-infected patients who have already been exposed to most of the available antiretroviral drugs (*see* **Note 3**).

3.3. Increasing the Accuracy of ANN Predictions

3.3.1. Size and Diversity of the Data Set

We examined the impact of the size and diversity of the data set on the accuracy of ANN predictions [28, 29]. ANN models were developed using data from (1) a single cohort (four models, 228 TCEs), (2) two cohorts (13 models, 747 TCEs), and (3) multiple sources (10 models, 1,581 TCEs). We also assessed the accuracy of predictions by "global" ANN models derived from data from over 200 clinics (10 "global models," 3,168 TCEs). All training and testing of data subsets was independent, randomly partitioned, and normalized.

We found the mean correlation (r^2) between the predicted and actual viral load change for the four ANN models developed from a single cohort was 0.65 and the mean correct trajectory prediction rate was 76%. These values were 0.62 and 78% for the 13 ANN models derived using pooled data from two cohorts, and 0.55 and

74% for the larger composite data set. The value for the composite data set was significantly lower than for the other two data sets.

For the "global" analysis, predictions by ten "local" ANN models trained with data from a single clinic (clinic A, 337 TCEs) were compared with those of the global models (including the 337 TCEs from clinic A). Forty and 38 TCEs from two single clinics (clinics A and B) were partitioned at random from the RDI database as test data sets. ANN predictions of viral load response were compared with the actual viral load responses in the two test sets (*see* **Note 4**). In this study we utilized the "committee average" method of harnessing the predictive power of the ANN. The predictions of ten ANN in a committee were average for each test TCE and the accuracy of the system then was assessed primarily by correlating the committee average predictions of follow-up viral load for each TCE with the actual follow-up viral.

We found the correlation between the committee average predicted and actual viral load gave r^2 values of 0.70 vs. 0.78 for test set A and of 0.23 vs. 0.07 for test set B, for global vs. local models, respectively. These correlations using the committee average were superior (in terms of r^2 values) to those of any individual ANN models. Similarly, the mean absolute difference between predicted and actual viral load was 0.55 vs. 0.49 log (ns) for test set A, and 0.66 vs. 0.97 log ($p < 0.05$) for test set B, for the global vs. local models, respectively. Therefore, global ANN models can perform as accurately as local models (trained with data from a single clinic) in predicting response for patients from that clinic. Moreover, global models appear superior to local models for making predictions for patients from other clinics, suggesting that they may be a more powerful way to exploit ANN as a generally available treatment decision-making tool (*see* **Note 5**).

3.3.2. Adjusting the Models for Additional Variables

It is possible that the accuracy of ANN predictions may be limited by preexisting minority populations of drug resistant variants that are not detected by the standard genotyping techniques used to define the baseline genotype of the clinical isolates in our database (typically, standard genotyping can detect minor variants only when they are present at levels of around 20-30% of the population). To address this possibility, we examined whether inclusion of previous treatment history information as a surrogate for minority mutant populations, as additional input variables in the training of ANN, may enhance the accuracy of viral load predictions by ANNs [30]. Similarly, we also examined the influence of genotype history [31] and adherence [32] variables on the accuracy of ANN predictions (*see* **Note 6**).

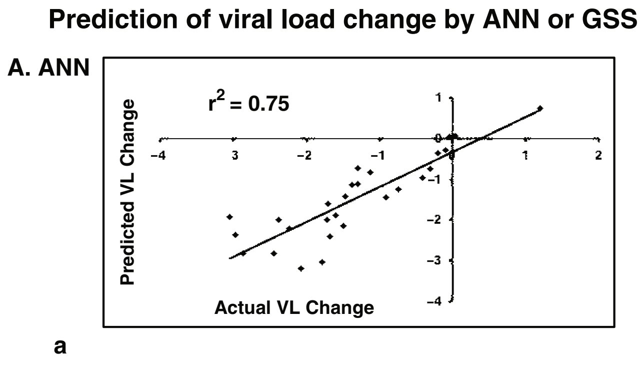
Basic ANN models were trained using 2,559 TCEs. Treatment history models were generated by training the basic models with four additional input variables related to prior exposure to antiretroviral therapy (any use of zidovudine, lamivudine, any nonnucleoside reverse transcriptase, and any protease inhibitor). Genotype history models were generated by training the basic models with 301 TCEs containing cumulative data from all available genotypes up to and including the baseline. The ANNs were then tested using the input variables of independent test TCEs, which produced a predicted viral load response that was compared to the actual response. Correlations between predicted and actual viral load change (r^2) for basic and treatment history models and basic and genotypic history models were 0.30 and 0.45 ($p < 0.01$), and 0.39 and 0.31 (ns), respectively.

To assess the impact of adherence variables, ANN models were trained using data from patients with low and high ($<$ or $\geq 90\%$) adherence (estimated by patient prescription refills). These models were tested using input variables from an independent data set. ANN models trained using data from highly adherent patients were significantly more accurate than those trained with data from less adherent patients ($r^2 = 0.29$ vs. 0.11, $p < 0.01$).

3.4. Comparison of ANN Predictions with Rules-Based Predictions Currently Used in Routine Clinical Care

Genotyping with rules-based interpretation is commonly used as a decision-making tool in routine clinical care of HIV-infected patients. Rules-based systems generally provide a three- category prediction, of sensitive, intermediate, or resistant, for each drug in a regimen. By allocating scores, for example 0, 0.5, and 1, to these classes of prediction, genotypic sensitivity scores can be generated for combinations of drugs. We compared the accuracy of viral load predictions derived using ANNs, with genotypic sensitivity scores (GSS) derived from commonly used rules-based interpretation systems (Stanford *[*HIVdb 4.1.2*]*, ANRS *[*v2004.09*]*, and Rega *[*v6.2*]*) to further define the clinical utility of ANNs in clinical decision-making for HIV infection [33].

Five basic ANN models were trained with data from 2,983 TCEs, using the following input variables: 55 HIV-resistance mutations, drugs in new regimen, baseline viral load, and time to follow-up. The accuracy of ANN predictions of the change in viral load for 27 randomly selected TCEs was assessed by comparison with determinations of the actual change in viral load ($r^2 = 0.75$). Changes in viral load based on the assumed impact on susceptibility as indicated by total and normalized GSS were also correlated with actual change in viral load for the test TCEs. The GSS was derived from the respective rules-based interpretation system and normalized relative to the number of drugs in each TCE. The ANN models were considerably more accurate predictors of virological response to combination therapy than rules-based GSS, explaining 75% of the variance in the change in viral load versus 13–29%. In addition, scatterplots indicated that ANNs were able to differentiate between regimens that GSS scores indicated would be equally effective (Fig. 7.2).



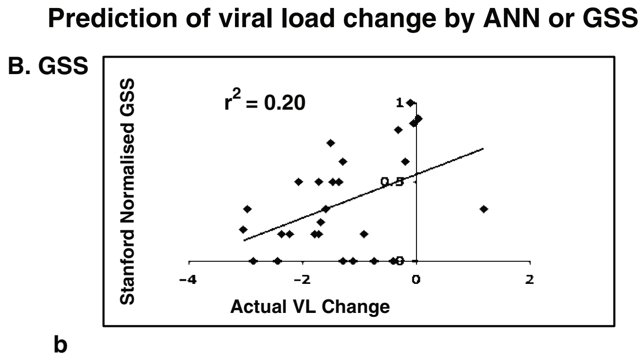


Fig. 7.2 Prediction of viral load changes by ANN or GSS. ANN models were trained with 2,983 TCEs then assessed for accuracy at predicting viral load with a test set of 27 independent TCEs. The results of the correlation between predicted and actual viral load changes are presented in A. The same test set of 27 independent TCEs was used to access the accuracy of mutation genotypic sensitivity scores in predicting actual viral load changes. B shows the results of a correlation between actual viral load changes and GSS. These scores were from the Stanford system (www.hivdb.stanford.edu), using version HIVdb 4.1.2. Each score was normalized relative to the number of drugs taken by a patient in the test TCE

4. Conclusions

Computerized decision support tools utilizing ANN are increasingly being incorporated into clinical practice. Our investigations of ANNs as tools to aid the choice of treatment of HIV patients failing antiretroviral therapy aims to facilitate the tailoring of drug therapy for the individual patient on the basis of genetic and clinical information (pharmacogenomics). On submission of the genetic code of HIV from an individual patient, together with specific clinical information, this interpretation system predicts which combination of drugs is likely to produce the best response in terms of overcoming viral resistance and reducing viral load. To our knowledge, this is the first application of ANNs as a strategy toward controlling the emergence of drug-resistant pathogens.

5. Notes

1. In all the studies outlined here, three-layer neural networks were constructed as previously described [22], although the output of the models was follow-up viral load ($\log_{10}$ copies/ml) rather than fold change in phenotype. Layer 1 consisted of input nodes, layer 2 was a hidden layer of intermediate nodes, and layer 3 was the output node. The models had full interconnection from the input nodes to the hidden nodes and to the output node.
2. Input training of the ANN models was sensitive to the timing of the viral load determination prior to treatment change. ANN modeling using the narrowest baseline viral load window gave the most accurate prediction of viral load response. This underlines the importance of the *quality* of the data set and on clinical application of the technology, the requirement for the clinician to be knowledgeable of the criteria under which the ANN can make accurate predictions.
3. ANN modeling identified potentially beneficial treatment regimens in all patients who experienced treatment failure following a treatment change guided by a genotype, predicting substantial reductions in viral load.
4. Throughout our studies, maximizing the amount of data in the ANN training sets has always been paramount, for there to be as wide a representation of different drug and mutation combinations as possible. This often meant limiting the size of independent test sets, although ideally, these should be between 5–10% of the training set. Another practical consideration is the number of ANN models developed per training set to use as the "committee"-averaged output. Although this has not been rigorously tested, we found that the development of ten models per committee is a reasonable compromise between "smoothing" the output and the CPU time required for training multiple models. ANN models trained using limited data from one clinical setting were surprisingly accurate in predicting responses to treatment from the genotype for other patients from the same cohort. Models developed using considerably more data from a variety of settings are not automatically more accurate, when tested with data from multiple settings. Preliminary analysis suggests that the accuracy of these ANN models may depend on the training and test data being similarly diverse. Although this does not necessarily imply similarity in terms of specific mutations or drugs but rather in terms of technical differences, such as race, nutrition, health status, access to health care, and the like.
5. Until ANN models have been used in prospective clinical trials to assess their accuracy predicting response, it is difficult to judge just how accurate a particular ANN needs to be. One must be mindful of the "noise" in the data sets caused by patient lack of adherence to their drug regimen and other factors. Nonadherence has been estimated to be as high as 20–30% in some treatment cohorts. In these circumstances, no matter how well trained, we would not expect a particular ANN to approach 100% accuracy. In fact, the best we could hope for is probably around 75% accuracy. Whether this translates into a trained ANN giving r^2 values of around 0.75 using test data sets from unrelated patients remains to be seen.
6. The choice of input variables for ANN training can obviously be critical for the final predictive accuracy of a model. We found that this is another area where a

trade-off occurs between what would be ideal and what is actually practical, both in terms of the size of the data sets and the type of data that can be collected. Given the typical size of the training data sets employed, we calculated the optimum number of input variables to be around 70-80 (data on file). The list of HIV drug-resistance mutations that have been used as input variables has been continually refined with different ANN models. Prevalence analyses were performed to select mutations considered the most significant, so that the dimensionality of genetic variation space could be reduced (as HIV is a highly variable virus, a comprehensive list of variant amino acids could run into the hundreds). Currently, 71 input variables are used to train "basic" models: 55 baseline genotypic mutations selected on the basis of frequency in the RDI database and from the literature (HIV reverse transcriptase codons: M41L, E44D, A62V, K65R, D67N, 69 insert, T69D/N, K70R, L74V, V75I, F77L, A98G, L100I, L101I/E, K103N, V106A, V108I, Y115F, F116Y, V118I, Q151M, V179D, Y181C, M184V, Y188C/L/H, G190S/A, L210W, T215Y/F, K219Q/E, P236L; HIV protease codons: L10F/I/R/V, K20M/R, L24I, D30N, V32I, L33F, M36I, M46I/L, I47V, G48V, I50V, I54V/L, L63P, A71V/T, G73S/A, V77I, V82A/F/T/S, I84V, N88S, L90M), drugs in the new combination regimen (14 drugs covering all those appearing in the training and test data sets: zidovudine, didanosine, stavudine, abacavir, lamivudine, tenofovir, efavirenz, nevirapine, indinavir, nelfinavir, ritonavir as a protease inhibitor booster, saquinavir, amprenavir, lopinavir), baseline viral load, and time to follow-up viral load. Undoubtedly, the inclusion of additional key input variables has improved and will continue to improve the accuracy of ANN predictions. Where previous treatment information has been available, this has been included in ANN models, but in an abbreviated format to approximate past experience (i.e., 4 variables, rather than the possibility of 14). Baseline CD4 cell count is another variable that can influence the way in which patients can respond to combination therapy. We recently found that inclusion of CD4 count as an additional variable can improve ANN predictions (data on file). Patient adherence to their drug regime is known to be a significant issue (*see* **Note 5**). However, it is beyond the scope of this modeling to predict whether a patient will actually take his or her prescribed medications. Our studies on adherence have, not surprisingly, served to underline the fact that ANN training is more successful if the data set is from a highly drug adherent clinical population. Thus, in the absence of comprehensive, reliable adherence data for a particular data set, it may be necessary to devise "censoring" methods to eliminate presumed nonadherent TCEs from training and test sets.

References

1. Ahmed FE (2005) Artificial neural networks for diagnosis and survival prediction in colon cancer. Mol Cancer 4:29–41.

2. Jerez JM, Franco L, Alba E, Llombart-Cussac A, Lluch A, Ribelles N, et al. (2005) Improvement of breast cancer relapse prediction in high risk intervals using artificial neural networks. Breast Cancer Res Treat 94:265–272.
3. Anagnostou T, Remzi M, Lykourinas M, Djavan B (2003) Artificial neural networks for decision-making in urologic oncology. Eur Urol 43:596–603.
4. Suzuki K, Li F, Sone S, Doi K (2005) Computer-aided diagnostic 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.
5. Baxt WG, Shofer FS, Sites FD, Hollander JE (2002) A neural computational aid to the diagnosis of acute myocardial infarction. Ann Emerg Med 39:366–373.
6. George J, Ahmed A, Patnaik M, Adler Y, Levy Y, Harats D, et al. (2000) The prediction of coronary atherosclerosis employing artificial neural networks. Clin Cardiol 23:453–456.
7. Zini G (2005) Artificial intelligence in hematology. Hematology 10:393–400.
8. Solomon I, Maharshak N, Chechik G, Leibovici L, Lubetsky A, Halkin H, et al. (2004) Applying an artificial neural network to warfarin maintenance dose prediction. Isr Med Assoc J 6:732–735.
9. Huang L, Yu P, Ju F, Cheng J (2003) Prediction of response to incision using the mutual information of electroencephalograms during anaesthesia. Med Eng Phys 25:321–327.
10. Fuller J J, Emmett M, Kessel JW, Price PD, Forsythe, JH (2005) A comparison of neural networks for computing predicted probability of survival for trauma victims. WV Med J 101:120–125.
11. Bent P, Tan HK, Bellomo R, Buckmaster J, Doolan L, Hart G, et al. (2001) Early and intensive continuous hemofiltration for severe renal failure after cardiac surgery. Ann Thorac Surg 71:832–837.
12. Papik K, Molnar B, Schaefer R, Dombovari Z, Tulassay Z, Feher J (1995) Application of neural networks in medicine: a review. Diagnostics and Medical Technology 1:538–546.
13. Zhu Y, Williams S, Zwiggelaar R (2006) Computer technology in detection and staging of prostate carcinoma: a review. Med Image Anal 10:178–199.
14. Lisboa PJ, Taktak AF (2006) The use of artificial neural networks in decision support in cancer: a systematic review. Neural Netw (Feb 13) (Epub in press).
15. Crawford ED (2003) Use of algorithms as determinants for individual patient decision making: national comprehensive cancer network versus artificial neural networks. Urology 62 (6) Suppl 1:13–19.
16. Koss LG, Sherman, ME, Cohen MB, Anes AR, Darragh TM, Lemos LB, et al. (1997) Significant reduction in the rate of false-negative cervical smears with neural network-based technology (PAPNET testing system). Hum Pathol 28:1196–1203.
17. Sherman ME, Schiffman MH, Mango LJ, Kelly D, Acosta D, Cason Z, et al. (1997) Evaluation of PAPNET testing as an ancillary tool to clarify the status of the "atypical" cervical smear. Mod Pathol10:564–571.
18. Babaian RJ, Fritsche H, Ayala A, Bhadkamkar V, Johnston DA, Naccarato W, et al. (2000) Performance of a neural network in detecting prostate cancer in the prostate-specific antigen reflex range of 2.5 to 4.0 ng/mL. Urology 56:1000–1006.
19. Zlotta AR, Remzi M, Snow PB, Schulman CC, Marberger M, Djavan B (2003) An artificial neural network for prostate cancer staging when serum prostate specific antigen is 10 ng./ml or less. J Urol 169:1724–1728.
20. Egmont-Petersen M, de Ridder D, Handels H (2002) Image processing with neural networks—a review. Pattern Recognition 35:2279–2301.
21. Sordo M (2002) Introduction to neural networks in healthcare. Open Clinical. [online] www.openclinical.org/docs/int/neuralnetworks011.pdf.
22. Wang D, Larder B (2003) Enhanced prediction of lopinavir resistance from genotype by use of artificial neural networks. J Infect Dis 188:653–660.
23. The Panel on Clinical Practices for Treatment of HIV Infection Convened by the Department of Health and Social Services. (2006) Guidelines for the use of antiretroviral agents in HIV-1 infected adults and adolescents, October 6, 2005. [online] http://aidsinfo.nih.gov/Content-Files/AdultandAdolescentGL.pdf.

24. Harrigan PR, Hogg RS, Dong WW, Yip B, Wynhoven B, Woodward J, et al. (2005) Predictors of HIV drug-resistance mutations in a large antiretroviral-naive cohort initiating triple antiretroviral therapy. J Infect Dis 191:339–347.
25. Wang D, De Gruttola V, Hammer S, Harrigan R, Larder B, Wegner S, et al., on Behalf of the HIV Resistance Response Database Initiative. (2002) A collaborative HIV resistance response database initiative: predicting virological response using neural network models. Antiviral Therapy 7:S96.
26. Wang D, Larder BA, Revell A, Harrigan R, Montaner J, on behalf of the HIV Resistance Response Database Initiative. (2003) A neural network model using clinical cohort data accurately predicts virological response and identifies regimens with increased probability of success in treatment failures. Antiviral Therapy 8:S112.
27. Larder BA, Wang D, Revell A, Lane C (2003) Neural network model identified potentially effective drug combinations for patients failing salvage therapy. 2nd IAS conference on HIV pathogenesis and treatment, Paris, July 13–16, poster LB39.
28. Larder BA, Wang D, Revell A, Harrigan R, Montaner J, Lane C (2004) Accuracy of neural network models in predicting HIV treatment response from genotype may depend on diversity as well as size of data sets. 11th conference on retroviruses and opportunistic infections, San Francisco, February 8–11, poster 697.
29. Revell A, Larder BA, Wang D, Wegner S, Harrigan R, Montaner J, Lane C (2005) Global neural network models are superior to single clinic models as general quantitative predictors of virologic treatment response. 3rd IAS conference on HIV pathogenesis and treatment. July 24–27, Rio de Janeiro, poster WePe12.6C04.
30. Wang D, Larder BA, Revell A, Harrigan R, Montaner J, Wegner S, Lane C (2005) Treatment history improves the accuracy of neural networks predicting virologic response to HIV therapy. BioSapiens-viRgil workshop on bioinformatics for viral infections, September 21–23, Caesar Bonn, Germany, poster 20.
31. Larder BA, Wang D, Revell A, Harrigan R, Montaner J,Wegner S, Lane C (2005) Treatment history but not previous genotype improves the accuracy of predicting virologic response to HIV therapy. 45th ICAAC, December 16–19. Washington, DC, poster H-1051.
32. Larder BA, Wang D, Revell A, Harrigan R, Montaner J, Wegner S, Lane C (2005) Treatment history and adherence information significantly improves prediction of virological response by neural networks. Antiviral Therapy 10:S57.
33. Larder BA, Revell A, Wang D, Harrigan R, Montaner J, Wegner S, Lane C (2005) Neural networks are more accurate predictors of virological response to HAART than rules-based genotype interpretation systems. 10th European AIDS conference/EACS, November 17–20, Dublin, poster PE3.4/13.

Chapter 8
Neural Networks in Building QSAR Models

Igor I. Baskin, Vladimir A. Palyulin, and Nikolai S. Zefirov

Abstract This chapter critically reviews some of the important methods being used for building quantitative structure-activity relationship (QSAR) models using the artificial neural networks (ANNs). It attends predominantly to the use of multilayer ANNs in the regression analysis of structure-activity data. The highlighted topics cover the approximating ability of ANNs, the interpretability of the resulting models, the issues of generalization and memorization, the problems of overfitting and overtraining, the learning dynamics, regularization, and the use of neural network ensembles. The next part of the chapter focuses attention on the use of descriptors. It reviews different descriptor selection and preprocessing techniques; considers the use of the substituent, substructural, and superstructural descriptors in building common QSAR models; the use of molecular field descriptors in three-dimensional QSAR studies; along with the prospects of "direct" graph-based QSAR analysis. The chapter starts with a short historical survey of the main milestones in this area.

Keywords Artificial neural networks, QSAR, back-propagation, learning, generalization

1. Introduction

The first application of artificial neural networks (ANNs) in the domain of structure-activity relationships dates back to the early 1970s. In 1971, Hiller et al. [1] reported on a study dealing with the use of perceptrons, the only type of artificial neural networks known at that time [2], to classify substituted 1,3-dioxanes as active or inactive with regard to their physiological activity. In the cited work, coded elements of chemical structures were projected onto the perceptron retina; the perceptron was trained using a set of compounds with known activities, and the trained neural network demonstrated good recognition ability on both the training and the test sets of compounds. This methodology was discussed in detail in

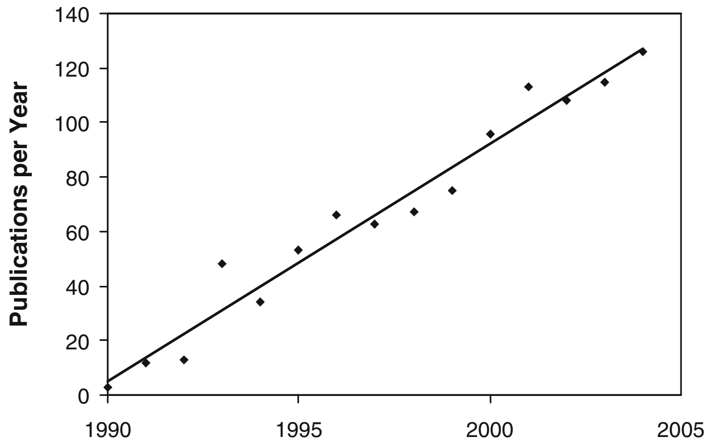


Fig. 8.1 Dependence of the number of papers published per year on the year of publication

another paper [3]. Nevertheless, the approach was not appreciated properly at that time; the articles remained almost unknown and have never been cited in any publication dealing with the use of neural networks in structure-activity studies.

The next stage of scientific development in this direction started in 1990 with the first publications of Aoyama, Suzuki, and Ichikawa dealing with the use of ANNs in QSAR studies [4, 5]. For the last 15 years, this approach to modeling structure-activity relationships has grown up and developed into a well-established scientific area with numerous ideas, theoretical approaches, and successful practical applications. Several relevant books [6, 7] and numerous review articles [8–28] are available. Figure 8.1 depicts the linear dependence of the number of papers published each year in this field on the year of publication. This linear trend demonstrates the steady growth of this scientific area, which now encompasses the use of artificial neural networks for predicting not only different types of biological activity of chemical compounds but also their physicochemical, ADME, biodegradability, and spectroscopic properties, as well as their reactivity. The aim of this paper is to review some important concepts and ideas accumulated in this field.

2. Methods, Discussion, and Notes

2.1. Neural Network Computational Methods Used in QSAR Studies

Applications of artificial neural networks in QSAR studies are characterized by a wide variety of architectures of neural networks as well as numerous approaches to represent chemical structures, preprocessing and selection of relevant descriptors, running the learning process, handling the predicted results, and so forth. All these notions characterize the *computational methods* used in this scientific area. The notion of the computational method is more specific than the notions of the neural network *architecture* and neural network *paradigm*. For example, we treat the genetic, the Bayesian, and the ordinary back-propagation neural networks as distinct computational methods, although they actually use the same architecture. Table 8.1 lists the main computational neural network methods with at least one application in QSAR studies. For each method, the table shows its name, a short identifier (used further in this chapter), the year of the first publication dealing with its use in QSAR studies, as well as the reference to such publication.

Table 8.1 Neural network methods used in QSAR studies

Name	Identifier	Year	Reference
Perceptron	Perceptron	1971	[*1*, *3*]
Back-propagation neural network	BPNN	1990	[*4*]
Autoassociative feedforward neural network	AAFFNN	1991	[*29*]
Self-organizing maps (Kohonen neural network)	SOM	1991	[*30*]
Counterpropagation neural network	CPNN	1992	[*31*]
Function-link neural network	FUNCLINK	1992	[*32*]
Neural device for direct structure-property correlation	NDDSPC	1993	[*33*, *34*]
Radial basis functions neural network	RBFNN	1993	[*35*]
Evolutionary algorithm for BPNN	Ev-BPNN	1993	[*36*]
Ensembles of BPNNs	Ens-BPNN	1993	[*37*]
Simulated annealing for BPNN	SA-BPNN	1995	[*38*]
Genetic algorithm for BPNN	GA-BPNN	1996	[*39*]
Principal component analysis for BPNN	PCA-BPNN	1996	[*40*]
Adaptive resonance theory 2-A	ART-2-A	1997	[*41*]
Bayesian regularized neural network	BRNN	1997	[*42*]
Probabilistic neural network	PNN	1997	[*43*]
Receptor-like neural network	RLNN	1997	[*44*]
Cascade-correlation neural network	CCNN	1998	[*45*]
Genetic algorithm for CPNNs	GA-CPNN	1999	[*46*]
Fuzzy adaptive resonance theory for mapping	FARTMAP	2000	[*47*]
Fuzzy neural network	FNN	2000	[*48*]
Recursive cascade correlation neural network	RCCNN	2001	[*49*]
Volume learning algorithm neural network	VLANN	2001	[*50*]
Comparative molecular surface analysis	CoMSA	2002	[*51*]
Genetic algorithm for RBFNN	GA-RBFNN	2002	[*52*]
Generalized regression neural network	GRNN	2002	[*53*]
Integrated SOM-fuzzy ARTMAP neural system	SOM-FARTMAP	2002	[*54*]
Particle swarms for BPNN	PS-BPNN	2002	[*55*]
Artificial ant colonies for BPNN	ANT-BPNN	2002	[*56*]
Hopfield neural network	HNN	2003	[*57*]
Genetic algorithm and principal component analysis for back-propagation neural network	PCA-GA-BPNN	2003	[*58*]
Variable structure neural net for pattern recognition	VSNNPR	2003	[*59*]
Niching particle swarms for BPNN	NPS-BPNN	2003	[*60*]
Genetic algorithm for self-organizing maps	GA-SOM	2004	[*61*]
Learning vector quantization	LVQ	2004	[*62*]

2.2. Tasks Performed by Neural Networks in QSAR Studies

In principle, ANNs can be used for solving any solvable task in computational mathematics, ranging from simple addition of binary numbers up to theorem proving. A special field in the computer science, *neuromathematics*, tackles such problems. In practice, ANNs are usually used for solving so-called ill-posed problems, for which numerous alternative solutions can be suggested, such as function approximation, pattern recognition, clustering, and data reduction. These problems exactly correspond to tasks performed by neural networks in an absolute majority of QSAR studies. Solving typical ill-posed problems involves some sort of *learning* or *training*, which can be *supervised*, *unsupervised*, or *reinforcement*. The last type of learning has not yet been used in QSAR/QSPR applications. Unsupervised learning analyzes the internal data structure by finding clusters and reducing dimensionality. In this paper, we discuss only supervised learning.

2.3. Supervised Learning

In the course of the supervised learning, a neural network tries to approximate experimental data by comparing its output values with the "desired" ones and minimizing the discrepancy by adjusting its internal parameters, such as connection weights and activation thresholds. If the experimental data are expressed by real numbers, the network performs *function approximation*, while for discrete, especially binary, values it performs *pattern recognition*. So, in statistical terms, it performs *regression* analysis and data *classification* (discriminant analysis), respectively. QSAR studies usually deal with regression analysis, while data classification is carried out in SAR studies. The rest of this chapter attends to the regression analysis performed by ANNs in QSAR studies.

2.3.1. Regression

The regression task is solved in almost all quantitative structure-property relationships (QSPR) and in a big part of structure-activity applications. The most widely used computational method for this purpose is BPNN (feedforward neural net trained with error back propagation, or back-propagation neural networks) or one of its derivatives (such as BRNN or GA-BPNN). The popularity of BPNN originates from its ability to be a "model-free mapping device" [63], which can approximate any nonlinear dependence of output variables (properties, activities) on the values of input variables (usually descriptors). Kolmogorov's superposition theorem [64] on the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, in conjunction with Kurková's results [65], presents the basis for this universal approximating ability of BPNNs.

So, the main advantage of using multilayer ANNs in QSAR studies lies in their ability to cope with nonlinearity in relationships, this ability requiring a hidden layer of neurons. Neural nets without hidden neurons, such as FUNCLINK, can hardly be considered as universal approximators, although some kinds of nonlinearity can be revealed by them due to nonlinear transforms of input variables (see the discussion in [66]). BPNNs approximate properties *globally*, since all hidden neurons are involved in production of neural-net output signal for all input vectors. In contrast, some other kinds of ANNs, such as CPNN or RBFNN, approximate properties *locally*, since only few neighboring (in descriptor space) hidden neurons are really involved in production of neural-net output. As a result, in spite of the ability of CPNNs and RBFNNs to approximate any nonlinear function with any precision (as a consequence of Kolmogorov's theorem), the number of hidden neurons required for attaining a given approximation error depends, in the general case, exponentially on data dimensionality. This is a consequence of the *dimensionality curse* [67], which is inherent to nonparametric kernel-based approximators. Since the number of hidden neurons should be much smaller than the number of data points, which is always very limited in QSAR studies, the approximating ability of BPNNs outperforms that of CPNNs and RBFNNs for data sets with not very low data dimensionality.

The global approximation character of BPNNs has some additional important implications for QSAR studies. If the compounds used for training and predictions are rather distinct from each other, one cannot expect good predictive performance of any statistical or neural-net model. However, if the compounds from the prediction set are similar to each other and the property to be predicted is measured for some of them, then it is possible to correct predictions for the remaining compounds from the prediction set by utilizing the measured property values of similar compounds without the need to retrain the neural network. This idea lies behind the associative neural networks (ASNN) concept [68], which combines global approximation performed by BPNNs with a local nearest-neighbors correction. In ASNN, the similarity of chemical structures is computed in the model space by correlating predictions made by different BPNN models. We implemented a different procedure in the NASAWIN program, which consists in correcting BPNN predictions by utilizing experimental property values of neighboring compounds, which maps onto the same cell in Kohonen neural network (SOM) trained using the same or another set of descriptors [59]. Raevsky et al. suggested correcting MLR (multiple linear regression, which performs global approximation) predictions with nearest-neighbor corrections, the similarity being determined by means of Tanimoto coefficients computed using substructure screens [69]. So, local correction to global structure-property approximations can be very useful, and it can efficiently be utilized by BPNNs.

The global approximating character of BPNNs has a direct connection to the distributed (holographic) character of information storage in human brain. As a consequence, the influence of an input signal on an output one is coded by means of several connection weights, and therefore the values of individual connection weights, taken separately, tell nothing about the influence of any input signal on any output one. In other words, weights provide nonlocal information on the influence of an input variable [70]. Consequently, individual connection weights cannot be interpreted separately from the other ones, and this has given rise to the myth of uninterpretability of neural network models.

2.3.2. Interpretation of Neural Network Regression Models

The problem of neural network *interpretation* has been addressed in a number of studies (see [70, 71] for relevant references). Aoyama and Ichikawa suggested using the partial differential coefficients of output value with respect to input parameters to analyze the relationship between inputs and outputs in neural network for each data point and demonstrated this method for QSAR data on biological activity of mitomycins [72]. Developing these ideas further, we suggested in a paper [73] to use the first (mean value) and the second (dispersion) distribution moments of the first and the second partial derivatives of the outputs with respect to inputs in neural networks over the whole training set for interpreting neural network QSAR/QSPR models. The use of such statistics makes it possible not only to obtain actually the same characteristics as for traditional "interpretable" statistical methods, such as the linear regression analysis, but also to reveal important additional information concerning the nonlinearity of QSAR/QSPR relationships. This approach was illustrated by interpreting BPNN models for predicting position of the long-wave absorption band of cyan dyes [73] and the acid hydrolysis rate constants of esters [74]. In both cases, the interpretations of neural network models appeared to be compatible with theoretical knowledge about the underlying physical and chemical phenomena.

Guha et al. have recently readdressed this problem in two papers [71, 75]. Guha, and Jurs [75] presented a method to measure the relative importance of the descriptors in QSAR neural network models based on a sensitivity analysis of descriptors. For all the reported data sets, the important descriptors revealed by this method appeared to correspond to the important descriptors in the linear model built using the partial least squares (PLS) technique. This means that the interpretability of neural network models is not reduced in comparison with traditional statistical linear approaches (exemplified by the PLS method), at least for interpretation methods based on ranking the relative importance of descriptors. Guha, Stanton, and Jurs [71] presented a methodology to carry out a detailed interpretation of the weights and biases of neural network QSAR models. It allows one to understand how an input descriptor is correlated with the predicted output property. The proposed methodology is based on the linearization of the transform functions in all computational (hidden or output) neurons. The data flow in the resulting network mimics that of the PLS analysis, with the latent variables of the latter corresponding to the hidden neurons in the network and the appropriate matrix elements corresponding to the connection weights in the neural network. This allowed the authors to develop interpretations of a neural network regression model similar in manner to the partial least squares interpretation method for linear models described by Stanton [76]. The method was tested on several data sets, the results of the interpretation method corresponded well to PLS interpretations for linear models using the same descriptors, and they were shown to be consistent with the generally accepted physical interpretations for these properties.

So, artificial neural networks (at least BPNNs) should no longer be regarded as uninterpretable "black boxes," and each QSAR work involving development of neural network models would benefit from application of interpretation procedures.

2.3.3. Methods for Controlling Generalization

Generalization means the ability of neural networks to predict the observed response variable for patterns not included in the training set. In other words, this is the ability to predict some biological activity or some other property of new chemical compounds. *Generalization error* can be estimated by computing the root-mean-square error of prediction on some external *validation* set. *Memorization* means the ability to reproduce the values of the response variable for patterns taken from the training set. *Memorization error* is the root-mean-square error of prediction on the training set. Evidently, in QSAR studies the primary concern should be to build neural network models that generalize better, that is, with the smallest generalization error. If, for some fixed data set, we gradually increase the complexity of some neural network (which is defined as the number of its adjustable parameters, i.e., connection weights and biases) by adding additional hidden neurons, the generalization error initially decreases but, after reaching the optimal network size, starts to increase, although the memorization error decreases all the time. The resulting neural network has bad generalization and good memorization ability. This phenomenon is called *overfitting*. Generally, the training of a neural network is a multistep iterative process. At the first phase of the training, the generalization error decreases but after reaching some point starts to increase, although the memorization error decreases all the time. Again, the resulting neural network model shows bad generalization but good memorization ability. This phenomenon is called *overtraining*. Therefore, one should exert every effort to prevent overfitting and overtraining. Both phenomena are thoroughly analyzed in papers [77–80] as applied to building QSAR models. The recipe to prevent overfitting, in accordance to these papers, lies in keeping the values of the parameter ρ, which is the ratio of the number of data points to the number of connections, higher than some threshold. This parameter, put forward originally by Andrea and Kalayeh [81], was analyzed in detail by Livingstone and coworkers [77, 78]. The overtraining can be avoided by means of "early stopping" of training after reaching the lowest generalization error as estimated using an additional validation data set [80]. It is also asserted by Tetko, Livingstone, and Luik [80] that overfitting and the associated "chance effects" are also prevented with the prevention of overtraining.

2.3.4. The Minimum Description Length Principle

To understand the essence of the aforementioned phenomena and assess the aforementioned results, consider the neural network learning from the statistical point of view [82]. According to the Bayesian theorem,

$$P(N \mid D) = \frac{P(D \mid N)P(N)}{\sum_N P(D \mid N)P(N)} \tag{1}$$

where N is a neural network model, D is a data set used for its training, $P(N \mid D)$ is the probability (i.e., validity) of the neural network model N trained with data D, $P(D \mid N)$ is the probability of data D to be explained by the neural network model N, and $P(N)$ is an a priori probability of the neural network model N. The best neural network model can be found by maximizing $P(N \mid D)$ or its logarithm:

$$\max_N \log P(N \mid D) \Rightarrow \max_N \{\log P(D \mid N) + \log P(N)\} \tag{2}$$

This is equivalent to

$$\min_N \{-\log P(D \mid N) - \log P(N)\} = \min_N \{I_D + I_N) \tag{3}$$

where I_D is the memorization error, expressed as the number of information bits necessary to account for all residuals on the training set, while I_N is the *model complexity*, expressed as the quantity of information needed for choosing the model N from the set with the a priori probability distribution $P(N)$. For multilayer neural networks, I_N can be taken as proportional to the number of connections and therefore to the number of hidden neurons. Equation 3 forms the basis of the minimum description length (MDL) principle originally put forward by Rissanen [83]. According to this principle, models with the shortest joint description of data and the model built on these data provide the lowest generalization error. Such models are characterized by the lowest error on the training set and the minimal possible number of adjustable model parameters.

The essence of the neural network training lies in the minimization of I_D. After the training, I_D becomes comparable or even lower than I_M. So, if the number of hidden neurons in BPNN exceeds some optimum, I_M starts to dominate over I_D, and the generalization error tends to be high. This explains the overfitting phenomenon and also opens the ways to finding the optimal number of connections W and hidden neurons H. It can be shown that for a simple case W can be estimated as [82]

$$W \sim \sqrt{Pd} \tag{4}$$

where P is the number of points in the training set, d is the number of input variables. More correct treatment would also include the dependence of W on some additional parameters, such as the dimensionality of d and the complexity of a function being approximated. In any case, it is not sufficient to use the simple ρ parameter as a criterion for the overfitting.

The MDL principle can also be used for explaining the overtraining phenomenon. The neural network training usually starts with all adjustable parameters initialized with random values close to zero. The complexity I_N of such model is also close to zero. In the course of training, I_N gradually increases. At the initial phase of the training, when the values of all connection weights are still small, the responses of all transfer functions in neurons lie in the linear range, and therefore the network approximates the linear part of relationships. The effective number of adjustable parameters at this linear phase does not exceeds that of a fully trained linear model, that is, approximately d. So, at the first phase, the effective number of adjustable parameters *gradually* increases from 0 to approximately d. After that, with the rise of the connection weight values, the responses of neurons become more and more nonlinear, the network starts learning nonlinearity, and the effective number of adjustable parameters *gradually* increases from d to approximately $H{\cdot}d$. At some moment, I_N starts to rise quicker than I_D decreases in the course of the training. This explains the overtraining phenomenon.

Several conclusions can be drawn concerning the overtraining phenomenon. First, although "early stopping" is an important procedure, which is usually necessary for preventing overtraining and overfitting, its application results in models with reduced nonlinearity. The bigger is the size of the network, the more linear are such models. Therefore, the "early stopping" should always be applied with caution. This "overlinearization" phenomenon can be prevented by applying the automatic weight elimination (AWE) algorithm [82]. Second, application of very fast algorithms for neural network training is not always the best choice, since they may quickly produce overtrained models without the possibility to avoid this.

2.3.5. Regularization of ANN Models

An alternative approach to preventing overtraining lies in the use of *regularization*, which is introduced by means of special functions that penalize the growth of the connection weights and, by doing that, limit the growth of the model complexity I_N. This results in the decrease of the generalization error. The regularization method initially was developed by Tikhonov and Arsenin as a general principle for solving ill-posed problems [84]. In this method, the influence of the penalty function is controlled by a special regularization parameter, the optimal value of which can be found either by assessing generalization using cross-validation or by means of the Bayesian approach through statistical analysis of connection weight distributions without the need to use cross-validation. So, the latter approach, called *Bayesian regularization*, appeared to be very useful for building QSAR models (see, for example, [85]), since it uses the whole data set for training and the resulting neural network models are reproducible. Two other advantages of Bayesian neural networks are the possibility to apply the procedure of automatic relevance determination for selecting descriptors and the possibility to use the distributions of the predictions to evaluate their uncertainty.

2.3.6. Ensembles of Neural Networks

One of the drawbacks of using BPNN in QSAR studies lies in irreproducibility of the resulting QSAR models. Starting with the initial random values of connection weights, they can produce different models. This problem can be solved by using an ensemble of neural networks models instead of a single one and averaging the prediction results. As a result, the models become more reproducible, the chance effects caused by initial randomization of weights are eliminated, and the memorization and generalization errors usually become lower. This methodology was initially applied to building QSAR models in 1993 by Tetko, Luik, and Poda [37]. The issue of using of the neural network ensembles in QSAR and QSPR studies was analyzed in detail by Agrafiotis, Cedeño, and Lobanov [86].

2.4. Handling Chemical Structures

2.4.1. Descriptor Selection

Because of the abundance of different descriptor selection approaches used in QSAR studies in conjunction with ANNs, we list them while applying some sort of classification. Depending on the use of activity values, descriptor selection can be unsupervised or supervised. The *unsupervised forward selection* procedure, which generates a subset of nonredundant descriptors with reduced multicollinearity, is described by Whitley, Ford, and Livingstone [87]. A sort of an *unsupervised backward elimination* procedure, which, starting with the whole descriptor set, leaves a subset of descriptors with pairwise correlations limited by some threshold value (although with multicollinearity) is implemented in the NASAWIN software [59].

Supervised descriptor selection procedures can be preliminary (carried out before running neural networks) or online. A *supervised preliminary forward selection* procedure based on the use of stepwise linear regression analysis procedures is implemented in NASAWIN [59]. It works as follows. The first chosen descriptor provides the best correlation with the target property, while each next selected descriptor provides the best correlation with the residual vector obtained at the previous step. The procedure is stopped after a fixed number of steps or at the lowest prediction error for the validation set. The main advantage of the algorithm lies in its very high efficiency (it can easily operate with millions of descriptors on very large databases), although the quality of the resulting models is slightly worse in comparison with the standard stepwise multiple linear regression technique. Nonetheless, the resulting suboptimal subset of descriptors appears to be sufficiently good as an initial guess for conducting not only regression but also classification SAR/QSAR studies using BPNNs.

Supervised online descriptor selection procedures, which are executed simultaneously with the ANN learning, can be NN-guided or controlling. The *supervised online NN-guided forward selection* procedures are performed by means of "growing" neural network architectures, such as cascade-correlation networks (CCNNs) for approximation [45]. The *supervised online NN-guided backward elimination* procedures are based on the use of various *pruning* algorithms for eliminating unnecessary weights and neurons from neural networks (see, for example, [88]).

The *supervised online controlling descriptor selection* procedures are usually implemented by means of stochastic combinatorial optimization algorithms, which optimize some measure of neural network model quality by finding optimal subsets of selected descriptors. Probably the first application of this kind of descriptor selection procedures in SAR/QSAR studies was made by Brinn et al. in 1993 using the "evolution algorithm" (Ev-BPNN) [36]. This was followed by the application of the simulated annealing (SA) algorithm for the same purpose [38]. The next important step was made in 1996 by So and Karplus [39] by introduction of the *genetic algorithm* (GA), which, being combined with different neural network architectures, has led to numerous computational methods actively used in QSAR studies: GA-BPNN ("genetic neural network") [39], GA-CPNN [46], GA-RBFNN [52], GA-SOM [61]. Several modern combinatorial optimization algorithms have also been applied in combination with BPNNs for descriptor selection in QSAR studies: the *artificial ant colonies* (ANT-BPNN) [56], the *particle swarms* (PS-BPNN) [55], and the *niching particle swarms* (NPS-BPNN) [60]. So, a big arsenal of methods is currently available for performing descriptor selection in QSAR studies.

2.4.2. Descriptor Preprocessing

Three principal types of data preprocessing are used in QSAR studies: scaling, nonlinear transform, and reduction of data dimensionality. While the scaling of input and output variables is a standard practice and does not deserve special consideration in this paper, the issue of the nonlinear transform is not trivial. Although one might argue that nonlinear transfer is not needed with neural networks, nevertheless our practice is the evidence of its benefit [89]. Two plausible explanations for this can be given. The first deals with the use of the early stopping procedure for preventing overtraining, which hampers the learning of strong nonlinearity, while the second explanation concerns our use of the preliminary descriptor selection procedure based on the use of stepwise linear regressions, which can partially account for nonlinearity due to nonlinearly transformed descriptors (like in the function-link neural networks).

Data dimensionality reduction is usually carried out in QSAR studies by means of principal component analysis (PCA). This gave rise to the PCA-BPNN [40] and PCA-GA-BPNN [58] computational methods, although combination with other neural network architectures is also possible. SOMs are also used for this purpose, like in SOM-FARTMAP [54]. We successfully used PLS latent variables for input into BPNNs [59]. However, several promising modern data preprocessing techniques, such as the independent component analysis (ICA) [90], are still not reported to have been used in QSAR studies in conjunction with ANNs.

2.4.3. Substituent Descriptors

A big part of QSAR studies deals with the analysis of congeneric sets of compounds with common scaffolds and different substituents. The classical Hansch-Fujita [91] and Free-Wilson [92] approaches imply the use of substituent constants, such as Hammett σ constants and lipophilic constants π, as well as variables indicating the presence of some structural features at some fixed scaffold positions, for deriving QSAR models. As applied to these substituent-based approaches, artificial neural networks have efficiently been used, in addition to building QSAR models, for solving two problems specific for this kind of descriptor: prediction of substituent constants and construction of QSAR models with correct symmetry properties.

2.4.3.1. Prediction of Substituent Constants

Conducting QSAR studies in the framework of the Hansch-Fujuta approach involves the use of substituent constants, which are tabulated for many but not all possible substituents. This limits the practical use of this approach to derivatives with simple substituents and issues the challenge of predicting the values of various substituent constants for any substituent. To address this problem, Kvasnička, Sklwnák, Pospichal applied BPNNs trained with specially designed functional group descriptors (occurrence numbers of some labeled subgraphs) to predict the values of inductive (σ_I) and resonance (σ_R) substituent constants [93]. A training set consisting of 66 substituents with tabulated values of both constants was used in their study. Several years later, we used BPNNs and quantum-chemical descriptors to build neural network models for predicting five substituent constants: σ_m, σ_p, F, R, and E_s [94]. Two BPNNs were used in this study. The first one, with four outputs for simultaneous prediction of σ_m, σ_p, F, and R, was trained using data on 144 substituents, while the second one, with a single output and trained with data on 42 substituents, was used for predicting E_s. Good predictive performance was achieved for all cases.

The problem was recently readdressed by Chiu and So [95, 96]. They used a big data set of 764 substituents for training BPNNs to predict four Hansch substituent constants: π, MR, F, and R [95]. The *E*-state descriptors were used for π and MR, while the aforementioned Kvasnička's graph-theoretical functional group descriptors were used for F and R. In the subsequent paper [96], the values of substituent constants predicted using these neural network models were successfully used for deriving QSAR models for HIV-1 reverse transcriptase and dihydrofolate reductase inhibitors. So, the subsequent use of ANNs for computing descriptors and conducting QSAR studies enabled the authors to obtain easily interpretable QSAR models with good predictive performance [96]. This pair of works may constitute the first example of the hierarchical approach to QSAR modeling, which seems to be very promising one.

2.4.3.2. Building QSAR Models with Correct Symmetry Properties

The next problem associated with the use of substituent descriptors lies in the necessity to construct QSAR models with correct symmetry properties. This means that, if the scaffold of some congeneric series of compounds is symmetric and contains topologically equivalent substituent positions, then the predicted values of any property, including biological activities, should not depend on the way nonequivalent substituents are assigned to them. For example, if positions 1 and 2 in some scaffold are topologically equivalent, then the assignments (R_1 = Cl, R_2 = Br) and (R_1 = Br and R_2 = Cl) designate the same compound, and QSAR models with correct symmetry properties should predict for them the same activity values.

This issue was analyzed by us [97]. To tackle the problem, we put forward an approach based on the application of ANNs to the training sets expanded by adding the copies of compounds with the same activity values but with permuted assignment of equivalent substituent positions (the learned symmetry concept). As the proof of the concept, the better predictive ability of resulting QSAR models, as compared with the performances of neural network models for nonexpanded sets, was demonstrated for the calcium channel blockers of 1,4-dihydropyridine type and hallucinogenic phenylalkylamines.

2.4.4. Substructural Descriptors

Among all other kinds of descriptors used in conjunction with ANNs, the substructural ones (i.e., the occurrence numbers of some subgraphs in molecular graphs) occupy a special place. As we proved earlier [98, 99], any molecular graph invariant (that is, any nonlocal property of a chemical compound) can be uniquely represented as (1) a linear combination of occurrence numbers of some substructures (fragments), both connected and disconnected, or (2) a polynomial on occurrence numbers of some connected substructures. Since, according to Kolmogorov's theorem [64], any polynomial can be approximated with a three-layer neural network, any property that is not very sensitive to stereoisomerism (such as a majority of physical properties) can be approximated by an output of a multilayer, globally approximating neural network taking occurrence numbers of connected substructures as its inputs. We used this as a guideline in our studies on predicting various physicochemical properties of organic compounds [89, 100–105].

The main shortcoming of the direct use of substructural descriptors in QSAR studies lies in the necessity to possess a sufficiently large database, in which all important fragments should be well-represented. That is why substructural descriptors are used predominantly for predicting properties, for which such databases are available, namely, the physicochemical, and toxicological ones. For an example of the use of fragmental descriptors in conjunction with multilayer neural networks and a large structurally heterogeneous data set for predicting mutagenicity, see [36]. On the other hand, if only small data sets are available, substructural descriptors can be very effective in mixture with physicochemical and quantum-chemical ones (for an example, see our work on mutagenicity of substituted polycyclic compounds [106]).

2.4.5. Superstructural Descriptors

Superstructural descriptors constitute a good alternative to substructural ones for building QSAR models for small data sets. They are computed by mapping molecular graphs into some molecular supergraph (for the whole data set), transferring their local atomic and bond descriptors to the corresponding supergraph elements, from which fixed-sized descriptor vectors are formed. This methodology can efficiently be combined with the use of ANNs, as was demonstrated in [107] for the cases of building neural network QSAR models for phospholipase inhibition by substituted indoles, dopamine receptor antagonistic activity of quinolinones, anti-HIV activity of benzylpyrimidines, and the ability of peptidyl trifluoromethyl ketones to prevent elastase-induced lung damage.

2.4.6. Molecular Field Descriptors for Three-Dimensional QSAR

Although computation of a big part of different molecular descriptors uses information on molecular geometry, the term *3D-QSAR* is traditionally associated with CoMFA and molecular field descriptors (i.e., the values of electrostatic, steric, lipophilic, and similar potentials computed at some points in space, e.g., at lattice nodes or on a molecular surface) involved in it [108]. We confine our discussion to the use of this kind of descriptors. Several approaches to using ANNs in 3D-QSAR studies have been developed (see the review in [21]). The main milestones in this field are Polanski's receptor-like neural network (RLNN) [44], which utilizes charges located on atoms as space-positioned descriptors; the volume learning algorithm neural network (VLANN) developed by Tetko, Kovalishyn, and Livingstone [50], which uses molecular field descriptors computed as some points in space; the comparative molecular surface analysis (CoMSA) put forward by Polanski, Gieleciak, and Bak [51], which uses molecular field descriptors computed on a molecular surface; and a molecular surface-based method developed by Hasegawa and coworkers [109], which is similar to CoMSA and also uses molecular surface descriptors. Despite all the differences among these four methods, they adopt actually the same basic two-stage strategy with molecular field potentials being mapped from their initial points in Cartesian space onto the cells of the Kohonen neural network at the first step, followed by the application of some regression technique, such as the committee of BPNNs in VLANN and PLS in the other three procedures, to correlate the potential values averaged over the cells with the target property at the second stage of this methodology. Consider, however, the mentioned approaches from the view point of data processing.

The specificity of the 3D-QSAR studies lies in the large number (typically, thousands) of spatially organized molecular field descriptors. So, the main two challenges in using ANNs are to sharply reduce the data dimensionality and to introduce some sort of nonlinearity into QSARs [21]. While the second task is tackled in VLANN by means of BPNNs, the first issue is addressed in all three approaches by means of using the Kohonen self-orginizing maps. One might suppose that it is the unique property of SOMs to reduce drastically the data dimensionality used in these applications. Surprisingly, but this appears not to be the case. The number of variables is actually reduced in these methods by means of their clustering with the help of SOMs. Furthermore, such reduction is performed inefficiently in comparison with the well-established CoMFA approach, since the number of cells in SOMs is always much greater than the number of latent variables in CoMFA PLS analysis. SOMs are known to be very effective tools for performing topology analysis of data sets and their neighborhood-conserving mapping onto the graphs of interconnected neurons in the competitive layer. The success in describing data by means of SOMs greatly depends on the adequacy of topologies embedded in the SOMs to that of the data being analyzed. This immediately poses the question as to whether the toroidal topology with four neighbors for each neuron, which is currently adopted in all SOM applications in SAR/QSAR studies, is adequate to represent the geometry of molecules and the topology of molecular fields around them. Would not the spherical topology of neurons be more adequate for self-organizing mapping of molecular fields? Is it so necessary to confine such mapping to any topology at all? The main shortcoming of using SOMs for representing molecular fields lies in the abstract nonphysical characters of such maps, which can hardly be understood by chemists and biologists outside the QSAR community. And, finally, is competitive learning needed at all to preprocess data for 3D-QSAR analysis? Our preliminary computational experiments indicate the feasibility of various alternative approaches to using ANNs in 3D-QSAR studies [110].

So, in spite of the existence of a whole series of remarkable applications of ANNs in the area of 3D-QSAR analysis, this field is still in its infancy. It waits for new ideas to be expressed, new methods to be developed, and new striking applications to be contributed.

2.4.7. Vector-Based and Graph-Based QSAR

All QSAR approaches considered in this paper so far are vector-based, since the descriptors in them should be represented as vectors of the same size for each of the chemical compounds belonging to the same data set. Almost all statistical procedures and ANNs were vector-based till recently. However, the most natural mathematical objects for representing the structures of chemical compounds are molecular graphs or the corresponding matrices of *variable size*. Since all isomorphic molecular graphs correspond to the same chemical compound, any structure-activity relationship function *should not depend on the numbering of nodes in molecular graphs* (or the permutation of rows and columns in the corresponding connection tables). A traditional approach to building QSAR models consists in computing vectors of molecular descriptors (graph *invariants*, whose values do not depend on the numbering of nodes in molecular graphs), which are usually chosen ad hoc, followed by the application of a vector-based statistical or ANN technique for finding QSAR models. As an obvious shortcoming of the traditional approach, the resulting models appear to be too biased and too dependent on the choice of a necessary descriptor set.

As an alternative to the traditional vector-based QSAR approach, an interesting challenge would be to build graph- or matrix-based QSAR models. To address this problem, in 1993, we developed a special neural device (NDDSPC) capable of constructing graph-based QSPR relationships for alkanes [33]. An advanced version of NDDSPC was used further for constructing a number of graph-based QSPRs and QSARs for heterogeneous sets of chemical compounds [34]. In 1995, Kireev proposed the graph-based ChemNet neural network, capable of establishing QSPRs [111]. The next contribution was made by Ivanciuc with his MolNet, which can also be used for building QSPR models [112].

By the end of the 1990s, the necessity of creating machine learning methods capable of handling variable-size structured data, such as chemical structures or biological sequences, was realized by computer scientists, and this led to important developments in this field (the eighth issue of the *Neural Networks* journal in 2005 completely deals with applications in this currently very hot scientific area). Two types of neural networks, probabilistic Bayesian networks and deterministic recursive neural networks, were developed to tackle this problem for the case of acyclic graphs (see [113] and references therein). Despite some limitations, such networks can be used for conducting QSAR studies for congeneric data sets with acyclic substituents. And, indeed, one such network, the recursive cascade correlation neural network (RCCNN), belonging to the aforementioned second type, has successfully been used by Micheli et al. for building QSAR models for benzodiazepines [49]. One can expect that, in the future, with further developments in this field, graph-based ANNs will be widely accepted in QSAR studies and maybe even revolutionize this area.

3. Conclusions

In this paper, we considered only a part of application fields of ANNs in QSAR studies. The untouched issues include the unsupervised learning (clustering, data dimensionality reduction, self-organizing maps, alignment optimization), classification (special pattern recognition neural network architectures, virtual screening with neural classificators, methods of rule extraction and knowledge integration, fuzzy neural networks), multiobjective learning, inverse task, QSCAR, and many other important topics. The question arises: What is the cause of such unprecedented creativity of researchers working in this field? Why are ANNs so popular in this research domain and their popularity grows from year to year (see Fig. 8.1).

ANNs are not the only "model-free mapping device capable of approximating any nonlinear function" [63] known to QSAR community. Some other machine learning approaches, such as the support vector machines, can do the same. In addition, the nonlinearity is not very important in many QSAR fields, and the differences in performance between ANN and non-ANN approaches are often marginal. The answer seems to be as follows. To develop a new statistical approach or, at least, deeply understand the related modern mathematical papers, it is necessary to be a good mathematician. On the other hand, many chemists and biologists can not only understand the structure and operations of neural networks but also be very creative in this field. Therefore, the naturalness, simplicity, and clarity of neural networks in comparison with many alternative machine learning approaches attracts many scientists to this area and promotes their high creativity in it. That is why one can expect that the rule of the linear growth of the number of ANN applications in QSAR and related areas, which was uncovered in the beginning of this chapter and shown in Fig. 8.1, will hold true at least in the nearest future.

References

1. Hiller SA, Glaz AB, Rastrigin LA, Rosenblit AB (1971) Recognition of physiological activity of chemical compounds on perceptron with random adaptation of structure. Dokl Akad Nauk SSSR 199:851–853.
2. Minsky M, Papert S (1969) Perceptrons. MIT Press, Cambridge, MA.
3. Hiller SA, Golender VE, Rosenblit AB, Rastrigin, LA, Glaz AB (1973) Cybernetic methods of drug design. I. Statement of the problem—the perceptron approach. Comp. Biomed. Res 6:411–421.
4. Aoyama T, Suzuki Y, Ichikawa H (1990) Neural networks applied to structure-activity relationships. J Med Chem 33:905–908.
5. Aoyama T, Suzuki,Y, Ichikawa H (1990) Neural networks applied to pharmaceutical problems. III. Neural networks applied to quantitative structure-activity relationship (QSAR) analysis. J Med Chem 33:2583–2590.
6. Zupan J, Gasteiger J (1999) Neural networks in chemistry. Wiley-VCH, Weinheim.
7. Devillers J (ed) (1996) Neural networks in QSAR and drug design. Academic Press, San Diego, CA.
8. Manallack DT, Livingstone DJ (1995) Neural networks and expert systems in molecular design. Methods and Principles in Medicinal Chemistry 3:293–318.
9. Winkler DA, Maddalena DJ (1995) QSAR and neural networks in life sciences. Series in Mathematical Biology and Medicine 5:126–163.
10. Maddalena DJ (1996) Applications of artificial neural networks to quantitative structure-activity relationships. Expert Opinion on Therapeutic Patents 6:239–251.
11. Anzali S, Gasteiger J, Holzgrabe U, Polanski J, Sadowski J, Teckentrup A, Wagener M (1998) The use of self-organizing neural networks in drug design. Persp Drug Disc Des 9-11:273–299.
12. Schneider G, Wrede P (1998) Artificial neural networks for computer-based molecular design. Progress in Biophysics and Molecular Biology 70:175–222.
13. Kovesdi I, Dominguez-Rodriguez MF, Orfi L, Naray-Szabo G, Varro A, Papp JG, Matyus P (1999) Application of neural networks in structure-activity relationships. Med Res Rev 19:249–269.
14. Manallack DT, Livingstone DJ (1999) Neural networks in drug discovery: have they lived up to their promise? Eur J Med Chem 34:195–208.

15. Ochoa C, Chana A, Stud M (2001) Applications of neural networks in the medicinal chemistry field. Curr Med Chem 1:247–256.
16. Terfloth L, Gasteiger J (2001) Neural networks and genetic algorithms in drug design. Drug Discovery Today 6:S102–S108.
17. Winkler DA, Burden FR (2002) Application of neural networks to large dataset QSAR, virtual screening, and library design. Methods in Mol Biol 201:325–367.
18. Halberstam NM, Baskin I I, Palyulin VA, Zefirov NS (2003) Neural networks as a method for elucidating structure-property relationships for organic compounds. Russ Chem Rev 72:629–649.
19. Kaiser KLE (2003) The use of neural networks in QSARs for acute aquatic toxicological endpoints. J Mol Struct (Theochem) 622:85–95.
20. Kaiser KLE (2003) Neural networks for effect prediction in environmental and health issues using large datasets. QSAR Comb Sci 22:185–190.
21. Livingstone D J, Manallack DT (2003) Neural networks in 3D QSAR. QSAR Comb Sci 22:510–518.
22. Niculescu SP (2003) Artificial neural networks and genetic algorithms in QSAR. J Mol Struct (Theochem) 622:71–83.
23. Novic M, Vracko M (2003) Artificial neural networks in molecular structures-property studies. Data Handling in Science and Technology 23:231–256.
24. Polanski J (2003) Self-organizing neural networks for pharmacophore mapping. Advanced Drug Delivery Reviews 55:1149–1162.
25. Taskinen J, Yliruusi J (2003) Prediction of physicochemical properties based on neural network modeling. Advanced Drug Delivery Reviews 55:1163–1183.
26. Winkler DA (2004) Neural networks as robust tools in drug lead discovery and development. Molecular Biotechnology 27:138–167.
27. Winkler DA, Burden FR (2004) Bayesian neural nets for modeling in drug discovery. Drug Discovery Today 2:104–111.
28. Shoji R (2005) The potential performance of artificial neural networks in QSTRs for predicting ecotoxicity of environmental pollutants. Curr. Comput-Aided Drug Des 1:65–72.
29. Livingstone DJ, Hesketh G, Clayworth D (1991) Novel method for the display of multivariate data using neural networks. J Mol Graph 9:115–118.
30. Rose VS, Croall IF, MacFie HJH (1991) An application of unsupervised neural network methodology (Kohonen topology-preserving mapping) to QSAR analysis. QSAR 10:6–15.
31. Peterson KL (1992) Counter-propagation neural networks in the modeling and prediction of Kovats indexes for substituted phenols. Anal Chem 64:379–386.
32. Liu Q, Hirono S, Moriguchi I (1992) Comparison of the functional-link net and the generalized delta rule net in quantitative structure-activity relationship studies. Chem Pharm Bull 40:2962–2969.
33. Baskin II, Palyulin VA, Zefirov NS (1993) Methodology of searching for direct correlations between structures and properties of organic compounds by using computational neural networks. Dokl Akad Nauk 333:176–179.
34. Baskin II, Palyulin VA, Zefirov NS (1997) A neural device for searching direct correlations between structures and properties of chemical compounds. J Chem Inf Comput Sci 37:715–721.
35. Lohninger H (1993) Evaluation of neural networks based on radial basis functions and their application to the prediction of boiling points from structural parameters. J Chem Inf Comput Sci 33:736–744.
36. Brinn M, Walsh PT, Payne MP, Bott B (1993) Neural network classification of mutagens using structural fragment data. SAR QSAR Env Res 1:169–210.
37. Tetko IV, Luik AI, Poda GI (1993) Applications of neural networks in structure-activity relationships of a small number of molecules. J Med Chem 36:811–814.
38. Sutter JM, Dixon SL, Jurs PC (1995) Automated descriptor selection for quantitative structure-activity relationships using generalized simulated annealing. J Chem Inf Comput Sci 35:77–84.

39. So S-S, Karplus M (1996) Evolutionary optimization in quantitative structure-activity relationship: an application of genetic neural networks. J Med Chem 39:1521–1530.
40. Viswanadhan VN, Mueller GA, Basak SC, Weinstein JN (1996) A new QSAR algorithm combining principal component analysis with a neural network: application to calcium channel antagonists. Network Science 2.
41. Domine D, Devillers J, Wienke D, Buydens L (1997) ART 2-A for optimal test series design in QSAR. J Chem Inf Comput Sci 37:10–17.
42. Sato K, Nakagawa J, Matsuzaki H (1997) Bayesian neural network approach to quantitative structure-activity relationships in carboquinones. Ann Rept Tohoku College of Pharmacy 44:187–193.
43. Kaiser KLE, Niculescu SP, McKinnon MB (1997) On simple linear regression, multiple linear regression, and elementary probabilistic neural network with Gaussian kernel's performance in modeling toxicity values to fathead minnow based on Microtox data, octanol/water partition coefficient, and various structural descriptors for a 419-compound dataset. In: Quantitative structure-activity relationships in environmental sciences, VII, Proceedings of QSAR 96, Elsinore, Denmark, June 24–28, 1996, pp. 285–297.
44. Polanski J (1997) The receptor-like neural network for modeling corticosteroid and testosterone binding globulins. J Chem Inf Comput Sci 37:553–561.
45. Kovalishyn VV, Tetko IV, Luik AI, Kholodovych VV, Villa AEP, Livingstone DJ (1998) Neural network studies 3. Variable selection in the cascade-correlation learning architecture. J Chem Inf Comput Sci 38:651–659.
46. Zupan J, Novic M (1999) Optimization of structure representation for QSAR studies. Anal Chim Acta 388:243–250.
47. Espinosa G, Yaffe D, Cohen Y, Arenas A, Giralt F (2000) Neural network based quantitative structural property relations (QSPRs) for predicting boiling points of aliphatic hydrocarbons. J Chem Inf Comput Sci 40:859–879.
48. Li, P, Cheng Y-Y (2000) Studies on quantitative structure-activity relationships of benzodiazepines using fuzzy neural networks. Gaodeng Xuexiao Huaxue Xuebao 21:1473–1478.
49. Micheli A, Sperduti A, Starita A, Bianucci AM (2001) Analysis of the internal representations developed by neural networks for structures applied to quantitative structure-activity relationship studies of benzodiazepines. J Chem Inf Comput Sci 41:202–218.
50. Tetko IV, Kovalishyn VV, Livingstone DJ (2001) Volume learning algorithm artificial neural networks for 3D QSAR studies. J Med Chem 44:2411–2420.
51. Polanski J, Gieleciak R, Bak A (2002) The comparative molecular surface analysis (CoMSA)—a nongrid 3D QSAR method by a coupled neural network and PLS system: predicting pK_a values of benzoic and alkanoic acids. J Chem Inf Comput Sci 42:184–191.
52. Patankar SJ, Jurs PC (2002) Prediction of glycine/NMDA receptor antagonist inhibition from molecular structure. J Chem Inf Comput Sci 42:1053–1068.
53. Mosier PD, Jurs PC (2002) QSAR/QSPR studies using probabilistic neural networks and generalized regression neural networks. J Chem Inf Comput Sci 42:1460–1470.
54. Espinosa G, Arenas A, Giralt F (2002) An integrated SOM-fuzzy ARTMAP neural system for the evaluation of toxicity. J Chem Inf Comput Sci 42:343–359.
55. Agrafiotis DK, Cedeno W (2002) Feature selection for structure-activity correlation using binary particle swarms. J Med Chem 45:1098–1107.
56. Izrailev S, Agrafiotis DK (2002) Variable selection for QSAR by artificial ant colony systems. SAR QSAR Env Res 13:417–423.
57. Arakawa M, Hasegawa K, Funatsu K (2003) Application of the novel molecular alignment method using the Hopfield neural network to 3D-QSAR. J Chem Inf Comput Sci 43:1396–1402.
58. Hemmateenejad B, Akhond M, Miri R, Shamsipur M (2003) Genetic algorithm applied to the selection of factors in principal component-artificial neural networks: application to QSAR study of calcium channel antagonist activity of 1,4-dihydropyridines (nifedipine analogs). J Chem Inf Comput Sci 43:1328–1334.

59. Baskin II, Halberstam NM, Artemenko NV, Palyulin VA, Zefirov NS (2003) NASAWIN—a universal software for QSPR/QSAR studies. In: Ford M et al. (eds) EuroQSAR 2002 designing drugs and crop protectants: processes, problems and solutions. Blackwell Publishing, pp. 260–263.
60. Cedeno W, Agrafiotis DK (2003) Application of niching particle swarms to QSAR and QSPR. In: Ford M et al. (eds) EuroQSAR 2002 designing drugs and crop protectants: processes, problems and solutions . Blackwell Publishing, pp. 255–259.
61. Bayram E, Santago P II, Harris R, Xiao Y-D, Clauset AJ, Schmitt JD (2004) Genetic algorithms and self-organizing maps: a powerful combination for modeling complex QSAR and QSPR problems. J Comp-Aided Mol Des 18:483–493.
62. Baurin N, Mozziconacci J-C, Arnoult E, Chavatte P, Marot C, Morin-Allory L (2004) 2D QSAR consensus prediction for high-throughput virtual screening. An application to COX-2 inhibition modeling and screening of the NCI. J Chem Inf Comput Sci 44:276–285.
63. Maggiora GM, Elrod DW, Trenary RG (1992) Computational neural networks as model-free mapping devices. J Chem Inf Comput Sci 32:732–741.
64. Kolmogorov AN (1957) On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. Dokl Akad Nauk SSSR 114:953–956.
65. Kurková V (1992) Kolmogorov's theorem and multilayer neural networks. Neural Networks 5:501–506.
66. Manallack DT, Livingstone DJ (1994) Limitations of functional-link nets as applied to QSAR data analysis. Quantitative Structure-Activity Relationships 13:18–21.
67. Stuetzle W, Mittal Y (1979) Some comments on the asymptotic behavior of robust smoothers. In: Gasser T, Rosenblatt M (eds) Smoothing techniques for curve estimation. Springer-Verlag, Helderberg.
68. Tetko IV (2002) Neural network studies, 4. Introduction to associative neural networks. J Chem Inf Comput Sci 42:717–728.
69. Raevsky OA, Trepalin SV, Trepalina HP, Gerasimenko VA, Raevskaja OE (2002) SLIPPER-2001—software for predicting molecular properties on the basis of physicochemical descriptors and structural similarity. J Chem Inf Comput Sci 42:540–549.
70. Féraud R, Clérot F (2002) A methodology to explain neural network classification. Neural Networks 15:237–246.
71. Guha R, Stanton DT, Jurs PC (2005) Interpreting computational neural network quantitative structure-activity relationship models: a detailed interpretation of the weights and biases. J Chem Inf Mod 45:1109–1121.
72. Aoyama, T, Ichikawa H (1992) Neural networks as nonlinear structure-activity relationship analyzers. Useful functions of the partial derivative method in multilayer neural networks. J Chem Inf Comput Sci 32:492–500.
73. Baskin II, Ait AO, Halberstam NM, Palyulin VA, Zefirov NS (2002) An approach to the interpretation of backpropagation neural network models in QSAR studies. SAR QSAR Env Res 13:35–41.
74. Halberstam NM, Baskin II, Palyulin VA, Zefirov NS (2002) Quantitative structure-conditions-property relationship studies. Neural network modeling of the acid hydrolysis of esters. Mendeleev Commun 12:185–186.
75. Guha R, Jurs PC (2005) Interpreting computational neural network QSAR models: a measure of descriptor importance. J Chem Inf Mod 45:800–806.
76. Stanton DT (2003) On the physical interpretation of QSAR models. J Chem Inf Comput Sci 43:1423–1433.
77. Manallack D, Livingstone DJ (1992) Artificial neural networks: application and chance effects for QSAR data analysis. Med Chem Res 2:181–190.
78. Livingstone DJ, Salt DW (1992) Regression analysis for QSAR using neural networks. Bioorg Med Chem Let 2:213–218.
79. Livingstone DJ, Manallack DT (1993) Statistics using neural networks: chance effects. J Med Chem 36:1295–1297.

80. Tetko IV, Livingstone DJ, Luik AI (1995) Neural network studies, 1. Comparison of overfitting and overtraining. J Chem Inf Comput Sci 35:826–833.
81. Andrea TA, Kalayeh H (1991) Applications of neural networks in quantitative structure-activity relationships of dihydrofolate reductase inhibitors. J Med Chem 34:2824–2836.
82. Ezhov AA, Shumsky SA (1998) Neurocomputing and its applications in economics and business. MIPhI, Moscow.
83. Rissanen J (1983) A universal prior for the integers and estimation by minimum description length. Annals of Statistics 11:416–431.
84. Tikhonov AN, Arsenin VY (1977) Solutions of ill-posed problems. Winston & Sons, Washington, DC.
85. Burden FR, Winkler DA (1999) Robust QSAR models using Bayesian regularized neural networks. J Med Chem 42:3183–3187.
86. Agrafiotis DK, Cedeño W, Lobanov VS (2002) On the use of neural network ensembles in QSAR and QSPR. J Chem Inf Comput Sci 42:903–911.
87. Whitley DC, Ford MG, Livingstone DJ (2000) Unsupervised forward selection: a method for eliminating redundant variables. J Chem Inf Comput Sci 40:1160–1168.
88. Tetko IV, Villa AEP, Livingstone DJ (1996) Neural network studies, 2. Variable Selection. J Chem Inf Comput Sci 36:794–803.
89. Artemenko NV, Baskin II, Palyulin VA, Zefirov NS (2003) Artificial neural network and fragmental approach in prediction of physicochemical properties of organic compounds. Russ Chem Bull 52:20–29.
90. Gustafsson MG. (2005) Independent component analysis yields chemically interpretable latent variables in multivariate regression. J Chem Inf Comput Sci 45:1244–1255.
91. Hansch C, Fujita T (1964) σ-ρ-π Analysis. A method for the correlation of biological activity and chemical structure. J Am Chem Soc 86:1616–1626.
92. Free SM Jr, Wilson JW (1964) A mathematical contribution to structure-activity studies. J Med Chem 7:395–399.
93. Kvasnička V, Sklenák Š, Pospichal J (1993) Neural network classification of inductive and resonance effects of substituents. J Am Chem Soc 115:1495–1500.
94. Baskin II, Keschtova SV, Palyulin VA, Zefirov NS (2000) Combining molecular modeling with the use of artificial neural networks as an approach to predicting substituent constants and bioactivity. In: Gundertofte K, Jørgensen FS (eds) Molecular modeling and prediction of bioactivity. Kluwer Academic/Plenum Publishers, New York, pp. 468–469.
95. Chiu T-L, So S-S (2004) Development of neural network QSPR models for Hansch substituent constants, 1. Method and validations. J Chem Inf Comput Sci 44:147–153.
96. Chiu T-L, So S-S (2004) Development of neural network QSPR models for Hansch substituent constants, 2. Applications in QSAR studies of HIV-1 reverse transcriptase and dihydrofolate reductase inhibitors. J Chem Inf Comput Sci 44:154–160.
97. Baskin II, Halberstam NM, Mukhina TV, Palyulin VA, Zefirov NS (2001) The learned symmetry concept in revealing quantitative structure-activity relationships with artificial neural networks. SAR QSAR Env Res 12:401–416.
98. Baskin II, Skvortsova MI, Stankevich IV, Zefirov NS (1994) Basis of invariants of labeled molecular graphs. Doklady Chemistry 339:231–234.
99. Baskin II, Skvortsova MI, Stankevich IV, Zefirov NS (1995) On the basis of invariants of labeled molecular graphs. J Chem Inf Comput Sci 35:527–531.
100. Baskin II, Palyulin VA, Zefirov NS (1993) Computational neural networks as an alternative to linear regression analysis in studies of quantitative structure-property relationships for the case of the physicochemical properties of hydrocarbons. Dokl Akad Nauk 332:713–716.
101. Artemenko NV, Baskin II, Palyulin VA, Zefirov NS (2001) Prediction of physical properties of organic compounds by artificial neural networks in the framework of substructural approach. Doklady Chemistry 381:317–320.
102. Artemenko NV, Palyulin VA, Zefirov NS (2002) Neural-network model of the lipophilicity of organic compounds based on fragment descriptors. Doklady Chemistry 383:114–116.

103. Zhokhova NI, Baskin II, Palyulin VA, Zefirov AN, Zefirov NS (2003) Fragmental descriptors in QSPR: flash point calculations. Russ Chem Bull 52:1885–1892.
104. Zhokhova NI, Baskin II, Palyulin VA, Zefirov AN, Zefirov NS (2003) Calculation of the enthalpy of sublimation by the QSPR method with the use of a fragment approach. Russ J Appl Chem 76:1914–1919.
105. Zhokhova NI, Baskin II, Palyulin VA, Zefirov AN, Zefirov NS (2004) Fragment descriptors in QSPR: application to magnetic susceptibility calculations. J Structural Chem 45:626–635.
106. Liubimova IK, Abilev SK, Gal'berstam NM, Baskin II, Paliulin VA, Zefirov NS (2001) Computer-aided prediction of the mutagenic activity of substituted polycyclic compounds. Biology Bull 28:139–145.
107. Radchenko EV, Baranova OD, Palyulin VA, Zefirov NS (2003) Non-linear molecular field topology analysis by means of artificial neural networks. In: Ford M et al. (eds) EuroQSAR 2002 designing drugs and crop protectants: processes, problems and solutions. Blackwell Publishing, pp. 317–318.
108. Cramer RD, Patterson DE, Bunce JD (1988) Comparative molecular field analysis (CoMFA), 1. Effect of shape and binding of steroids to carrier proteins. J Am Chem Soc 110:5959–5967.
109. Hasegawa K, Matsuoka S, Arakawa M, Funatsu K (2002) New molecular surface-based 3D-QSAR method using Kohonen neural network and three-way PLS. Comput Chem 26:583–589.
110. Baskin II, Tikhonova IG, Palyulin VA, Zefirov NS (2006) A combined approach to building 3D-QSAR models. In QSAR and Molecular Modelling in Rational Design of Bioactive Molecules, Proceedings of the European Symposium on Structure-Activity Relationships (QSAR) and Moleular Modelling, 15th, Istanbul, Turkey, Sept. 5-10, 2004/Eds by E. Aki (Şener) and I. Yalçin, pp. 123–124.
111. Kireev DB (1995) ChemNet: a novel neural network based method for graph/property mapping. J Chem Inf Comput Sci 35:175–180.
112. Ivanciuc O (2001) Molecular structure encoding into artificial neural networks topology. Roumanian Chem Quart Rev 8:197–220.
113. Baldi P, Rosen-Zvi M (2005) On the relationship between deterministic and probabilistic directed graphical models: from Bayesian networks to recursive neural networks. Neural Networks 18:1080–1086.

Chapter 9
Peptide Bioinformatics

Peptide Classification Using Peptide Machines

Zheng Rong Yang

Abstract Peptides scanned from whole protein sequences are the core information for many peptide bioinformatics research such as functional site prediction, protein structure identification, and protein function recognition. In these applications, we normally need to assign a peptide to one of the given categories using a computer model. They are therefore referred to as *peptide classification applications*. Among various machine learning approaches, including neural networks, peptide machines have demonstrated excellent performance in many applications. This chapter discusses the basic concepts of peptide classification, commonly used feature extraction methods, three peptide machines, and some important issues in peptide classification.

Keywords bioinformatics, peptide classification, peptide machines

1. Introduction

Proteins are the major components for various cell activities, including gene transcription, cell proliferation, and cell differentiation. To implement these activities, proteins function only if they interact with each other. Enzyme catalytic activity, acetylation, methylation, glycosylation, and phosphorylation are typical protein functions through binding. Studying how to recognize functional sites is then a fundamental topic in bioinformatics research. As proteins function only when they are bound together, the binding sites (functional sites) and their surrounding residues in substrates are the basic components for functional recognition.

Studying consecutive residues around a functional site within a short region of a protein sequence for functional recognition is then a task of peptide classification that aims to find a proper model that maps these residues to functional status. A peptide classification model can be built using a properly selected learning algorithm with similar principles to those used in pattern classification systems.

The earlier work was to investigate a set of experimentally determined (synthesized) functional peptides to find some conserved amino acids, referred to as *motifs*. Based on these motifs, biologists can scan any newly sequenced proteins to identify potentially functional sites prior to further experimental confirmation. In some cases, a functional site occurs between two residues in a substrate peptide, such as protease cleavage sites. A peptide with ten residues can be expressed as P_5-P_4-P_3-P_2-P_1-P_1'-P_2'-P_3'-P_4'-P_5' with a functional site located between P_1 and P_1'. The other eight residues—P_5, P_4, P_3, P_2, P_2', P_3', P_4', and P_5'— called the *flanking residues*, are also essential for functional recognition. In some situations, asymmetrical peptides may be used. For instance, tetra-peptides P_3-P_2-P_1-P_1' are used in trypsin cleavage site prediction [1]. Protein functional sites involve one residue for most posttranslational protein modifications; for instance, phosphorylation sites [2] and glycosyslation sites [3] occur at only one residue. In this situation, we can express a peptide with nine residues as P_4-P_3-P_2-P_1-P_0-P_1'-P_2'-P_3'-P_4', where the functional site is at residue P_0. The other eight residues are the flanking residues and are again essential for functional recognition. Note that the peptide size is commonly determined by substrate size necessary for chemical reactions. For example, trypsin protease cleavage activity requires four residue long peptides for proper recognition, while hepatitis C virus protease cleavage activity requires ten residue long peptides for proper recognition. The importance of studying protease cleavage sites can be explained by drug design for HIV-1 proteases. Figure 9.1 shows the principle for HIV-1 virus production. New proteases are matured and function through cleavage to produce more viruses. The nucleus of an uninfected cell can be infected by reverse transcriptase. To stop the infection of uninfected cells and stop maturation of new proteases, it is important to design inhibitors to prevent the cleavage activities that produce new protease and reverse transcriptase. Without knowing how substrate specificity is related to cleavage activity, it is difficult to design effective inhibitors.

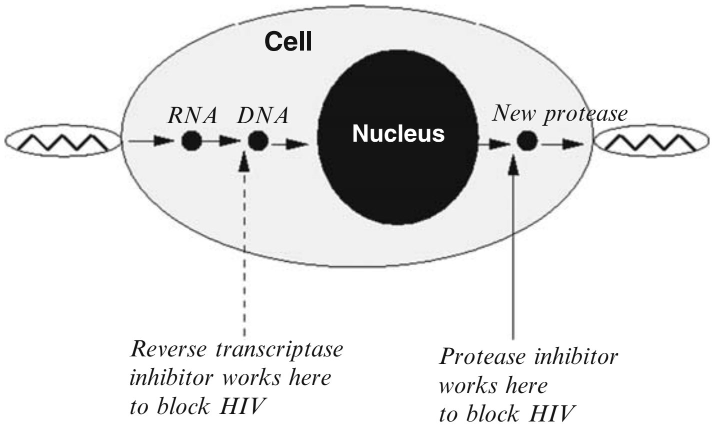

Cell
RNA
DNA
Nucleus
New protease
Reverse transcriptase inhibitor works here to block HIV
Protease inhibitor works here to block HIV

Fig. 9.1 Virus maturation process in the cell. The larger ellipse represents the cell and the smaller one represents the nucleus. Reproduced from [4] with permission

In protease cleavage site prediction, we commonly use peptides with a fixed length. In some cases, we may deal with peptides with variable lengths. For instance, we may have peptides whose lengths vary from one residue to a couple of hundreds residues long when we try to identify disorder regions (segments) in proteins [5]. Figure 9.2 shows two cases, where the shaded regions indicate some disorder segments. The curves represent the probability of disorder for each residue. The dashed lines show 50% of the probability and indicate the boundary between order and disorder. It can be seen that the disorder segments have variable lengths. The importance of accurately identifying disorder segments is related to accurate protein structure analysis. If a sequence contains any disorder segments, X-ray crystallization may fail to give a structure for the sequence. It is then critically important to remove such disordered segments in a sequence before the sequence is submitted for X-ray crystallization experiments. The study of protein secondary structures also falls into the same category as disorder segment identification, where the length of peptides that are constructs for secondary structures varies [6].

Because the basic components in peptides are amino acids, which are nonnumerical, we need a proper feature extraction method to convert peptides to numerical vectors. The second section of this chapter discusses some commonly used feature extraction methods. The third section introduces three peptide machines for peptide classification. The fourth section discusses some practical issues for peptide classification. The final section concludes peptide classification and gives some future research directions.

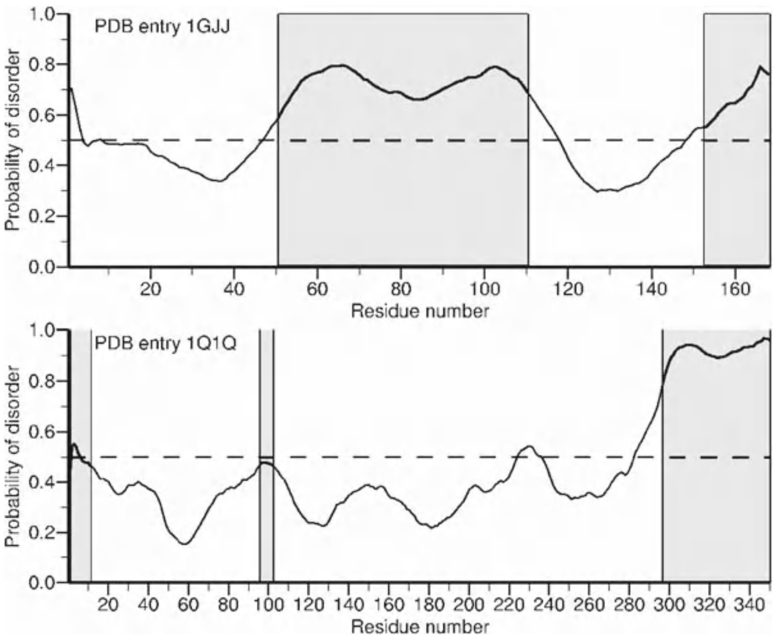


Fig. 9.2 The actual disorder segments and predictions. Reproduced from [7] with permission

2. Feature Extraction

Currently, three types of feature extraction methods are commonly used for converting amino acids to numerical vectors: orthogonal vector, frequency estimation, and bio-basis function methods.

The orthogonal vector method encodes each amino acid using a 20-bit long binary vector with 1 bit assigned a unity and the rest zeros [8]. Denoted by **s** a peptide, a numerical vector generated using the orthogonal vector method is $\mathbf{x} \in b^{20\times|\mathbf{s}|}$, where $b = \{0, 1\}$ and $|\mathbf{s}|$ is the length of **s**. The introduction of this method greatly eased the difficulty of peptide classification in the early days of applying various machine learning algorithms like neural networks to various peptide classification tasks. However, the method significantly expands the input variables, as each amino acid is represented by 20 inputs [9, *10*]. For an application with peptides ten residues long, 200 input variables are needed. Figure 9.3 shows such an application using the orthogonal vector method to convert amino acids to numerical inputs. It can be seen that data significance, expressed as a ratio of the number of data points over the number of model parameters, can be very low. Meanwhile, the method may not be able to properly code biological content in peptides. The distance (dissimilarity) between any two binary vectors encoded from two different amino acids is always a constant (it is 2 if using the Hamming distance or the square root of 2 if using the Euclidean distance), while the similarity (mutation or substitution probability) between any two amino acids varies [11–13]. The other limitation with this method is that it is unable to handle peptides of variable lengths. For instance, it is hard to imagine that this method could be used to implement a model for predicting disorder segments in proteins.

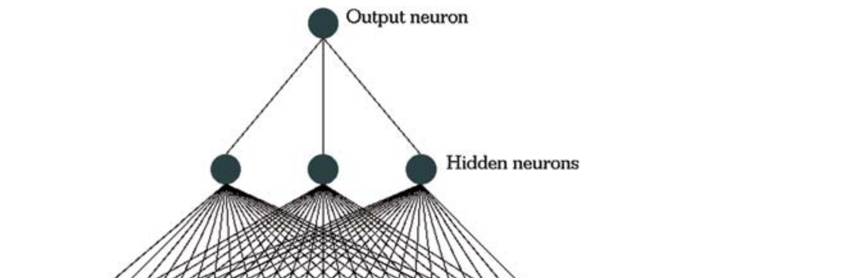


Fig. 9.3 An example of using the orthogonal vector for converting a peptide to numerical vector for using feedforward neural networks, where each residue is expanded to 20 inputs. Adapted from [14]

The frequency estimation is another commonly used method in peptide classification for feature extraction. When using single amino acid frequency values as features, we have the conversion as $\mathbf{S} \mapsto \mathbf{x} \in \Re^{20}$, meaning that a converted vector always has 20 dimensions. However, it has been widely accepted that neighboring residues interact. In this case, di-peptide frequency values as features may be used, leading to the conversion as $\mathbf{S} \mapsto \mathbf{x} \in \Re^{420}$. If tri-peptide frequency values are used, we then have the conversion as $\mathbf{S} \mapsto \mathbf{x} \in \Re^{8420}$. This method has been used in dealing with peptides with variable lengths for peptide classification, for instance, secondary structure prediction [15] and disorder segment prediction [5]. One very important issue of this method is the difficulty of handling the large dimensionality. For any application, dealing with a data set with 8,420 features is no easy task.

The third method, called the *bio-basis function*, was developed in 2001 [9, 10]. The introduction of the bio-basis function was based on the understanding that nongapped pairwise homology alignment scores using a mutation matrix are able to quantify peptide similarity *statistically*. Figure 9.4 shows two contours where one functional peptide (SKNYPIVQ) and one nonfunctional peptide (SDGNGMNA) are selected as indicator peptides. We use the term *indicator peptides*, because the use of some indicator peptides transform a training peptide space to an indicator or feature space for modeling. We then calculate the nongapped pairwise homology alignment scores using the Dayhoff mutation matrix [12] to calculate the similarities among all the functional/positive and these two indicator peptides as well as the similarities among all the nonfunctional/negative peptides and these two indicator peptides. It can be seen that positive peptides show larger similarities with the positive indicator peptide (SKNYPIVQ) from the left panel, while negative peptides show larger similarities with the negative indicator peptide (SDGNGMNA) from the right panel.

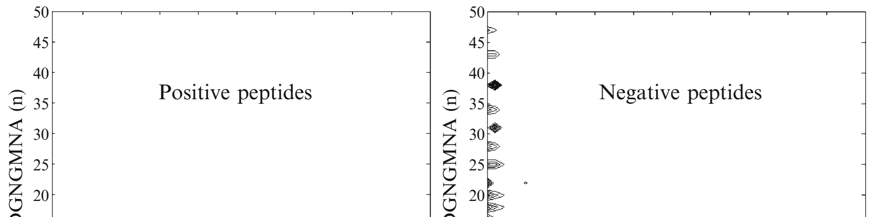


Fig. 9.4 Contours show that peptides with the same functional status demonstrate large similarity. Note that (p) and (n) stand for positive and negative

We denote by **s** a query peptide and by $\mathbf{r}_i$ the *i*th indicator peptide and each has *d* residue, the bio-basis function is defined as

$$\mathrm{K}(\mathbf{s}, \mathbf{r}_i) = \mathbf{exp}\left(\frac{\varphi(\mathbf{s}, \mathbf{r}_i) - \varphi(\mathbf{r}_i, \mathbf{r}_i)}{\varphi(\mathbf{r}_i, \mathbf{r}_i)} \right) \tag{1}$$

where $\varphi(\mathbf{s},\mathbf{r}_i) = \sum_{j=1}^{d} M(s_j, r_{ij})$. Note that s_j and r_{ij} are the *j*th residues in the query and indicator peptides, respectively. The value of $M(s_j, r_{ij})$ can be found from a mutation matrix. Figure 9.5 shows the Dayhoff mutation matrix [12]. From Eq. 1, we can see that $K(\mathbf{s}, \mathbf{r}_i) \leq 1$, because $\varphi(\mathbf{s}, \mathbf{r}_i) \leq \varphi(\mathbf{r}_i, \mathbf{r}_i)$. The equality occurs if and only if $\mathbf{s} = \mathbf{r}_i$. The basic principle of the bio-basis function is normalizing nongapped pairwise homology alignment scores. As shown in Fig. 9.6, a query peptide (**IPRS**) is aligned with two indicator peptides (**KPRT** and **YKAE**) to produce two nongapped pairwise homology alignment scores *a* ($\sum_1 = 24 + 56 + 56 + 36 = 172$) and *b* ($\sum_2 = 28 + 28 + 24 + 32 = 112$) respectively. Because $a > b$, it is believed that the query peptide is more likely to have the same functional status (functional or nonfunctional) as the first indicator peptide. After the conversion using the bio-basis function, each peptide **s** is represented by a numerical vector $\mathbf{x} \in \Re^m$ or a point in an *m*-dimensional feature space, where *m* is the number of indicator peptides. Note that $m = 2$ in Fig. 9.6.

The bio-basis function method has been successfully applied to various peptide classification tasks, for instance, the prediction of trypsin cleavage sites [9], the prediction of HIV cleavage sites [10], the prediction of hepatitis C virus protease cleavage sites [16], the prediction of the disorder segments in proteins [7, 17], the prediction of protein phosphorylation sites [18, 19], the prediction of the O-linkage sites in glycoproteins [20], the prediction of signal peptides [21], the prediction of factor Xa protease cleavage sites [22], the analysis of mutation patterns of HIV-1

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	40	24	32	32	16	36	28	28	28	24	28	32	36	32	24	36	36	32	8	20
C	24	80	12	12	16	20	20	24	12	8	12	16	20	12	16	32	24	24	0	32
D	32	12	48	44	8	36	36	24	32	16	20	40	28	40	28	32	32	24	4	16
E	32	12	44	48	12	32	36	24	32	20	24	36	28	40	28	32	32	24	4	16
F	16	16	8	12	68	12	24	36	12	40	32	16	12	12	16	20	20	28	32	60
G	36	20	36	32	12	52	24	20	24	16	20	32	28	28	20	36	32	28	4	12
H	28	20	36	36	24	24	56	24	32	24	24	40	32	44	40	28	28	24	20	32
I	28	24	24	24	36	20	24	52	24	40	40	24	24	24	24	28	32	48	12	28
K	28	12	32	32	12	24	32	24	52	20	32	36	28	36	44	32	32	24	20	16
L	24	8	16	20	40	16	24	40	20	56	48	20	20	24	20	20	24	40	24	28
M	28	12	20	24	32	20	24	40	32	48	56	24	24	28	32	24	28	40	16	24
N	32	16	40	36	16	32	40	24	36	20	24	40	28	36	32	36	32	24	16	24
P	36	20	28	28	12	28	32	24	28	20	24	28	56	32	32	36	32	28	8	12
Q	32	12	40	40	12	28	44	24	36	24	28	36	32	48	36	28	28	24	12	16
R	24	16	28	28	16	20	40	24	44	20	32	32	32	36	56	32	28	24	40	16
S	36	32	32	32	20	36	28	28	32	20	24	36	36	28	32	40	36	28	24	20
T	36	24	32	32	20	32	28	32	32	24	28	32	32	28	28	36	44	32	12	20
V	32	24	24	24	28	28	24	48	24	40	40	24	28	24	24	28	32	48	8	24
W	8	0	4	4	32	4	20	12	20	24	16	16	8	12	40	24	12	8	100	32
Y	20	32	16	16	60	12	32	28	16	28	24	24	12	16	16	20	20	24	32	72

Fig. 9.5 Dayhoff matrix. Each entry is a mutation probability from one amino acid (in rows) to another amino acid (in columns). Reproduced from [*14*] with permission

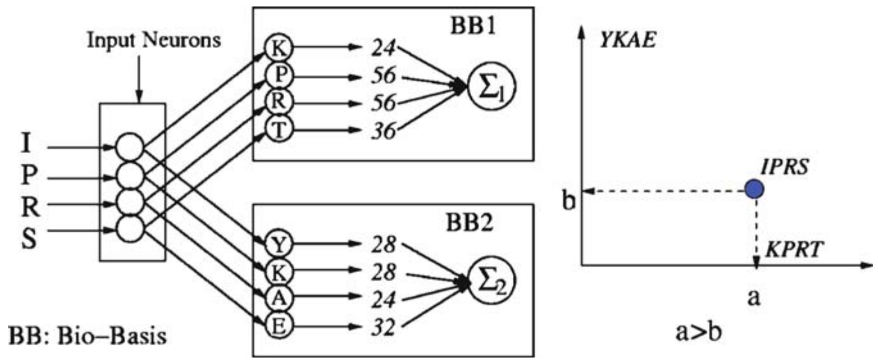


Fig. 9.6 The bio-basis function. As **IPRS** is more similar to **KPRT** than **YKAE**, its similarity with **KPRT** is larger that that with **YKAE**, see the right figure. Reproduced from [*14*] with permission

drug resistance [23], the prediction of caspase cleavage sites [24], the prediction of SARS-CoV protease cleavage sites, [25] and T-cell epitope prediction [26].

3. Peptide Machines

As one class of machine learning approaches, various vector machines are playing important roles in machine learning research and applications. Three vector machines—the support vector machine [27, 28], relevance vector machine [29], and orthogonal vector machine [30]—have already drawn the attention for peptide bioinformatics. Each studies pattern classification with a specific focus. The support vector machine (SVM) searches for data points located on or near the boundaries for maximizing the margin between two classes of data points. These found data points are referred to as the *support vectors*. The relevance vector machine (RVM), on the other hand, searches for data points as the representative or prototypic data points referred to as the *relevance vectors*. The orthogonal vector machine (OVM) searches for the orthogonal bases on which the data points become mutually independent from each other. The found data points are referred to as the *orthogonal vectors*. All these machines need numerical inputs. We then see how to embed the bio-basis function into these vector machines to derive peptide machines.

3.1. Support Peptide Machine

The support peptide machine (SPM) aims to find a mapping function between an input peptide **s** and the class membership (functional status). The model is defined as

$$y = f(\mathbf{s}, \mathbf{w}) \tag{2}$$

where **w** is the parameter vector, $y = f(\mathbf{s}, \mathbf{w})$ the mapping function, and y the output corresponding to the desired class membership $t \in \{-1, 1\}$. Note that –1 and 1 represent nonfunctional and functional status, respectively. With other classification algorithms like neural networks, the distance (error) between y and t is minimized to optimize **w**. This can lead to a biased hyperplane for discrimination. In Fig. 9.7, there are two classes of peptides, *A* and *B*. The four open circles of class *A* and four filled circles of the class *B* are distributed in balance. With this set of peptides, the true hyperplane separating two classes of peptides, represented as circles, can be found as in Fig. 9.7(a). Suppose a shaded large circle belonging to class *B* as a noise peptide is included, as seen in Fig. 9.7(b); the hyperplane (the broken thick line) is biased because the error (distance) between the nine circles and the hyperplane has to be minimized. Suppose a shaded circle belonging to class *A* as a noise peptide is included as seen in Fig. 9.7(c); the hyperplane (the broken thick line) also is biased. With theses biased hyperplanes, the novel peptides denoted by the triangles could be misclassified.

In searching for the best hyperplane, the SPM finds the set of peptides that are the most difficult training data points to classify. These peptides are referred to as *support peptides*. In constructing a SPM classifier, the support peptides are closest to the hyperplane within the slat formed by two boundaries (Fig. 9.7d) and are located on the boundaries of the margin between two classes of peptides. The advantage of using an SPM is that the hyperplane is searched through maximizing this margin. Because of this, the SPM classifier is robust; hence, it has better generalization performance than neural networks. In Fig. 9.7(d), two open circles on the upper boundary and two filled circles on the lower boundary are selected as support peptides. The use of these four circles can form the boundaries of the maximum margin between two classes of peptides. The trained SPM classifier is a linear

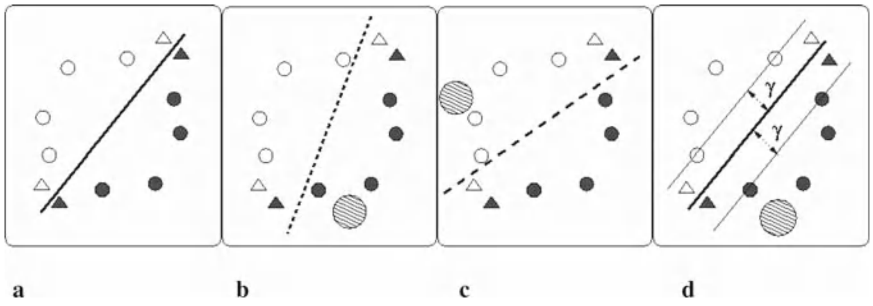


Fig. 9.7 **a** Hyperplane formed using conventional classification algorithms for peptides with a balanced distribution. **b** and **c** Hyperplanes formed using conventional classification algorithms for peptides without balanced distribution. **d** Hyperplane formed using SPM for peptides without balanced distribution. The open circles represent class *A*, the filled circles class *B*, and the shaded circle class *A* or *B*. The thick lines represent the correct hyperplane for discrimination and the broken thick lines the biased hyperplanes. The thin lines represent the margin boundaries. The γ indicates the distance between the hyperplane and the boundary formed by support peptides. The margin is 2γ. Reproduced from [*14*] with permission

combination of the similarity between an input peptide and the found support peptides. The similarity between an input peptide and the support peptides is quantified by the bio-basis function. The decision is made using the following equation, $y = \text{sign}\ \{\Sigma w_i t_i K(\mathbf{s}, \mathbf{r}_i)\}$, where t_i is the class label of the ith support peptide and w_i the positive parameter as a weight for the ith support peptide. The weights are determined by a SPM algorithm [31].

3.2. Relevance Peptide Machine

The relevance peptide machine (RPM) was proposed based on automatic relevance determination (ARD) theory [32]. In SPM, the found support peptides are normally located near the boundary for discrimination between two classes of peptides. However, the relevance peptides found by RPM are prototypic peptides. This is a unique property of RPM, as the prototypic peptides found by a learning process are useful for exploring the biological inside. Suppose the model output is defined as

$$y_i = \frac{\mathbf{1}}{1 + \exp(-\mathbf{k_i} \cdot \mathbf{w})} \tag{3}$$

where $\mathbf{k}_i = [K(\mathbf{s}_i, \mathbf{r}_1), K(\mathbf{s}_i, \mathbf{r}_2), \ldots, K(\mathbf{s}_i, \mathbf{r}_n)]^T$ is a similarity vector and $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$. The model likelihood function is defined as follows:

$$p(\mathbf{t} \mid \mathbf{w}) = \prod_{i=1}^{\mathbf{n}} y_i^{t_i}(1 - y_i)^{t_i} \tag{4}$$

Using ARD prior to computation can prevent overfitting the coefficients by assuming each weight follows a Gaussian

$$p(\mathbf{w} \mid \alpha) = \prod_{i=1}^{n} \mathrm{N}(0, \alpha_i^{-1}) \tag{5}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$. The Bayesian learning method gives the following posterior for the coefficients:

$$p(\mathbf{w} \mid \mathbf{t}, \boldsymbol{\alpha}) \propto |\boldsymbol{\alpha}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{w} - \mathbf{u})^{\mathrm{T}} \Sigma^{-1} (\mathbf{w} - \mathbf{u})\right] \tag{6}$$

where the parameters (covariance matrix $\boldsymbol{\Sigma}$, the mean vector $\mathbf{u}$, and the hyperparameters for weights $\mathbf{S}$), which can be approximated:

$$\Sigma = (\mathrm{K}^T \Lambda \mathrm{K} + \mathrm{S})^{-1} \tag{7}$$

$$\mathrm{u} = \Sigma \mathrm{K}^T \Lambda \mathrm{t} \tag{8}$$

$$\mathrm{S} = \mathrm{diag}(\alpha_1, \alpha_2, \alpha_\mathrm{n}) \tag{9}$$

$$\mathrm{K}=\begin{pmatrix} K(\mathbf{s}_1,\mathbf{r}_1) & K(\mathbf{s}_1,\mathbf{r}_2) & \cdots & K(\mathbf{s}_1,\mathbf{r}_n) \\ K(\mathbf{s}_2,\mathbf{r}_1) & K(\mathbf{s}_2,\mathbf{r}_2) & \cdots & K(\mathbf{s}_2,\mathbf{r}_n) \\ \vdots & \vdots & \vdots & \vdots \\ K(\mathbf{s}_n,\mathbf{r}_1) & K(\mathbf{s}_n,\mathbf{r}_2) & \cdots & K(\mathbf{s}_n,\mathbf{r}_n) \end{pmatrix} \tag{10}$$

and

$$\Lambda = \mathrm{diag}\{y_i(1 - y_i)\} \tag{11}$$

The marginal likelihood can be obtained through integrating out the coefficients:

$$p(\mathbf{t}\,|\,\boldsymbol{\alpha}) \propto |\Lambda^{-1} + \mathrm{KS}^{-1}\mathrm{K}^{\mathrm{T}}|^{-1/2} \exp\left(-\frac{1}{2}\mathrm{t}^{\mathrm{T}}(\Lambda^{-1} + \mathrm{KS}^{-1}\mathrm{K}^{\mathrm{T}})\mathrm{t}\right) \tag{12}$$

In learning, $\boldsymbol{\alpha}$ can be estimated in an iterative way:

$$\boldsymbol{\alpha}_i(\mathbf{t}+1) = \frac{\gamma_i(t)}{u_i^2(t)} \quad \forall i \in [1, n] \tag{13}$$

where

$$\gamma_i(t) = 1 - \alpha_i(t)\Sigma_{ii} \ \forall i \in [1, n] \tag{14}$$

A formal learning process of the RPM is then conducted. The following condition is checked during learning:

$$\alpha_{\mathrm{i}} > \theta \tag{15}$$

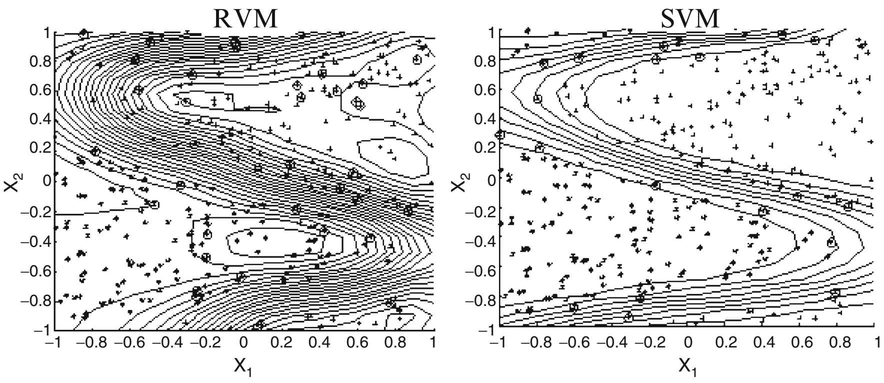


Fig. 9.8 How RPM and SPM select peptides as relevance or support peptides. Note that this is only an illustration. The curves are contours. The crosses represent one class of peptides and the stars the other. The circles represent the selected data points as the relevance peptides for the RPM and the support peptides for the SPM

where θ is a threshold. If the condition is satisfied, the corresponding expansion coefficient is zeroed. The learning continues until either the change of the expansion coefficients is small or the maximum learning cycle is approached [33].

Figure 9.8 shows how the RPM selects prototypic peptides as the relevance peptides, compared with the SPM, which selects boundary peptides as the support peptides.

3.3. *Orthogonal Peptide Machine*

First, we define a linear model using the bio-basis function as

$$\mathrm{y} = \mathrm{Kw} \tag{16}$$

where **K** is defined in Eq. 10. The orthogonal least square algorithm is used as a forward selection procedure by the orthogonal peptide machine (OPM). At each step, the incremental information content of a system is maximized. We can rewrite the design matrix **K** as a collection of column vectors ($\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_m$), where **k***i* represents a vector of the similarities between all the training peptides and the *i*th indicator peptide $\mathbf{r}_i$, $\mathbf{k}_i = [K(\mathbf{s}_1, \mathbf{r}_i), K(\mathbf{s}_2, \mathbf{r}_i), \ldots, K(\mathbf{s}_n, \mathbf{r}_i)]^{T.}$ The OPM involves the transformation of the indicator peptides ($\mathbf{r}_i$) to the orthogonal peptides ($\mathbf{o}_i$) to reduce possible information redundancy. The feature matrix **K** is decomposed to an orthogonal matrix and an upper triangular matrix as follows:

$$\mathrm{K} = \mathrm{OU} \tag{17}$$

where the triangular matrix **U** has ones on the diagonal line:

$$\mathrm{U} = \begin{pmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1,m-1} & u_{1m} \\ 0 & 1 & u_{23} & \cdots & u_{2,m-1} & u_{2m} \\ 0 & 0 & 1 & \cdots & u_{3,m-1} & u_{3m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & u_{m-1,m} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{18}$$

and the orthogonal matrix **O** is

$$\mathrm{O} = \begin{pmatrix} o_{11} & o_{12} & \cdots & o_{1m} \\ o_{21} & o_{22} & \cdots & o_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ o_{n1} & o_{n2} & \cdots & o_{nm} \end{pmatrix} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_m) \tag{19}$$

The orthogonal matrix satisfies

$$\mathrm{O}^T\mathrm{O} = \mathrm{H} \tag{20}$$

where $\mathbf{H}$ is diagonal with the elements h_{ii} as

$$h_{ii} = \mathbf{o}_i^T \mathbf{o}_i = \sum_{j=1}^{n} o_{ji}^2 \tag{21}$$

The space spanned by the set of orthogonal peptides is the same space spanned by the set of indicator peptides, and Eq. 16 can be rewritten as

$$\mathbf{y} = \mathbf{Kw} = \mathbf{OUw} = \mathbf{O}a \tag{22}$$

We can define an error model as follows:

$$\mathbf{e} = \mathbf{t} \ - \mathbf{y} \tag{23}$$

Suppose $\mathbf{e} \sim N(0, 1)$, the pseudoinverse method can be used to estimate $\mathbf{a}$ as follows:

$$\mathbf{a} = (\mathbf{O}^T\mathbf{O})^{-1}\mathbf{O}^T\mathbf{t} = \mathbf{H}^{-1}\mathbf{O}^T\mathbf{t} \tag{24}$$

Because $\mathbf{H}$ is diagonal,

$$\mathbf{H}^{-1} = \text{diag}\{h_{ii}^{-1}\} = \text{diag}\{(\mathbf{o}_i^T\mathbf{o}_i)^{-1}\} \tag{25}$$

The element in $\mathbf{a}$ is then

$$\alpha_i = \frac{\mathbf{o}^T \mathrm{t}}{\mathbf{o}^T \mathbf{o}} \tag{26}$$

The quantities $\mathbf{a}$ and $\mathbf{w}$ satisfy the triangular system

$$\mathbf{Uw} = \mathbf{a} \tag{27}$$

The Gram-Schmidt or the modified Gram-Schmidt methods are commonly used to find the orthogonal peptides then to estimate $\mathbf{w}$ [30, 34].

3.4. Case Study: Drug Resistance Data Analysis

HIV/AIDS is one of the most lethal and transmittable diseases, with a high mortality rate worldwide. The most effective prevention of HIV infection is to use a vaccine to block virus infection [34]. However, HIV vaccine is difficult to develop because of the expense and complexity in advancing new candidate vaccines. Although some efficient models and integrated HIV vaccine research enterprises have been proposed [36], there is little hope that an HIV vaccine would be developed before 2009 [37]. HIV is a type of retrovirus. It can enter uninfected cells to replicate itself. Inhibiting viral maturation and viral reverse transcription are then the major methods so far for treating HIV-infected patients. Two groups of inhibitors have since been developed. One group aims to stop protease cleavage activities and is referred to as the *protease inhibitor*. The other aims to stop reverse transcriptase cleavage activities and is referred to as the *reverse transcriptase inhibitor*.

However, HIV often develops resistance to the drugs applied. Drug-resistant mutants of HIV-1 protease limit the long-term effectiveness of current antiviral therapy [38]. The emergence of drug resistance remains one of the most critical issues in treating HIV-1-infected patients [39]. The genetic reason for drug resistance is the high mutation rate along with a very high replication rate of the virus. These two factors work together, leading to the evolution of drug-resistant variants and consequently resulting in therapy failure. The resistance to HIV-1 protease inhibitors can be analyzed at a molecular level with a genotypic or a phenotypic method [39, 40]. The genotypic method is used to scan a viral genome for some mutation patterns for potential resistance. As an alternative, the viral activity can be measured in cell culture assays. Yet another alternative, phenotypic testing is based on clinic observations, ithat is, directly measuring viral replication in the presence of increasing drug concentration.

Genotypic assays have been widely used as tools for determining HIV-1 drug resistance and for guiding treatment. The use of such tools is based on the principle that the complex impact of amino acid substitutions in HIV reverse transcriptase or protease on the phenotypic susceptibility or clinical response to the 18 available antiretroviral agents is observable [41]. Genotypic assays are used to analyze mutations associated drug resistance or reduced drug susceptibility. However, this method is problematic, because various mutations and mutational patterns may lead to drug resistance [39], and therefore the method occasionally fails to predict the effects of multiple mutations [42]. In addition, genotypic assays provide only indirect evidence of drug resistance [43]. Although HIV-1 genotyping is widely accepted for monitoring antiretroviral therapy, how to interpret the mutation pattern associated with drug resistance to make accurate predictions of susceptibility to each antiretroviral drug is still challenging [44]. Phenotypic assays directly measure drug resistance [43], where drug resistance can be experimentally evaluated by measuring the ratio of free drug bound to HIV-1 protease molecules. However, this procedure is generally expensive and time consuming [39, 42].

To deal with the difficulties encountered in genotypic assay analysis and phenotypic evaluation, a combination of inhibitor flexible docking and molecular dynamics simulations was used to calculate the protein-inhibitor binding energy. From this, an inhibitory constant is calculated for prediction [39]. Later, some statistical models were established for predicting drug resistance. For instance, a statistical model was proposed to analyze the viral and immunologic dynamics of HIV infection, taking into account drug-resistant mutants and therapy outcomes [45]. The factors causing the increase in CD4 cell count and the decrease in viral load from baseline after six months of HAART (highly active antiretroviral therapy) were analyzed. Note that CD4 stands for cluster of differentiation 4, which is a molecule expressed on the surface of T helper cells. In addition to the baseline viral load and CD4 cell count, which are known to affect response to therapy, the baseline CD8 cell count and resistance characteristics of detectable strains are shown to improve prediction accuracy. Note that CD8 stands for cluster of differentiation 8, which is a membrane glycoprotein found primarily on the surface of cytotoxic T cells. A logistic regression model was built for the prediction of the odds of achieving virology suppression after 24 weeks of HAART in 656 antiretroviral-naive patients starting HAART according to their week 4 viral load [46]. A regression model was built on a set of 650 matched genotype-phenotype pairs for the prediction of phenotypic drug resistance from genotypes [47]. As it was observed that the range of resistance factors varies considerably among drugs, two simple scoring functions were derived from different sets of predicted phenotypes. The scoring functions were then used for discrimination analysis [48].

In addition, machine learning algorithms were used. Decision trees as a method for mimicking human intelligence were implemented to predict drug resistance [43]. A fuzzy prediction model was built based on a clinical data set of 231 patients failing highly active antiretroviral therapy (HAART) and starting salvage therapy with baseline resistance genotyping and virological outcomes after three and six months [49]. In the model, a set of rules predicting genotypic resistance was initially derived from an expert and implemented using a fuzzy logic approach. The model was trained using the virological outcomes data set, that is, Stanford's HIV drug-resistance database (Stanford HIVdb). Expert systems were also used for this purpose [41]. Neural networks as a type of powerful nonlinear modelers were utilized to predict HIV drug resistance for two protease inhibitors, indinavir and saquinavir [50].

Other than using sequence information for prediction, a structure-based approach was proposed to predict drug resistance [42]. Models of WT complexes were first produced from crystal structures. Mutant complexes were then built by amino acid substitutions in the WT complexes with subsequent energy minimization of the ligand (a small molecule binding to a protein or receptor) and PR (protease receptor) binding site residues. A computational model was then built based on the calculated energies.

The data set studied was obtained from an online relational database, the HIV RT and protease database (http://hivdb.stanford.edu) [51]. The susceptibility of the data to five protease inhibitors were determined, saquinavir (SQV), indinavir (IDV), ritonavir (RTV), nelfinavir (NFV) and amprenavir (APV). We obtained 255 genotype-phenotype pairs for each of the protease inhibitor, except for APV, for which we obtained 112 pairs. Phenotypic resistance testing measures in-vitro viral replication of the wild type and the viral sample in increasing drug concentrations [52]. The resistance factor, defined as the ratio of 50% inhibitory concentration (IC_{50}) of the respective viral sample to IC_{50} of the nonresistant reference strain, reports the level of resistance as the fold change in susceptibility to the drug as compared to a fully susceptible reference strain. While genotypic resistance testing is done by scanning the viral genome for resistance-associated mutations. The major and minor mutations observed in a resistant HIV protease can be seen in Figure 1 in [53]. HIV protease is an aspartyl protease with 99 residues. In this study, we considered major and minor mutation sites, obtaining a window size of 20 residues for each drug. We divided the sample set into two classes by attaching to each peptide a label 1 or –1 for resistant (functional) and susceptible (nonfunctional), respectively. This division depended on whether the resistance factor of a sample exceeded the cutoff value or not. Based on previously published datasets and studies, we assumed the cutoff value for the resistant factor of each sample with respect to each protease inhibitor to be 3.5.

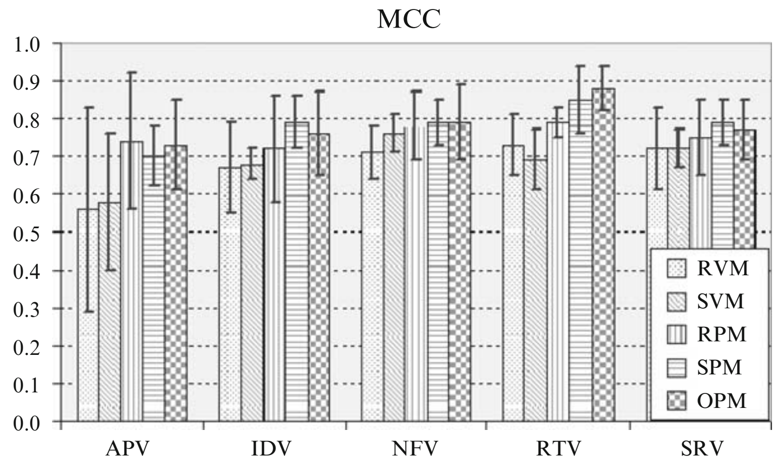


Fig. 9.9 Comparison between two vector machines and three peptide machines. The three peptide machines outperformed the two vector machines

We used Matthews correlation coefficient [54] for evaluation. Let TN, TP, FN, and FP denote true negative, true positive, false negative, and false positive, respectively. The definition of the Matthews correlation coefficient (MCC) is

$$\mathrm{MCC} = \frac{\mathrm{TN} \times \mathrm{TP}\text{-}\mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TN}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TP}+\mathrm{FP})}} \tag{28}$$

The larger the Matthews correlation coefficient, the better the model fits the data. If the value of the Matthews correlation coefficient is 1, it represents a complete correlation. If the value is 0, the prediction is completely random. If the value is negative, the prediction is on the opposite side of the target.

The evaluation is carried out based on the simulation of fivefold cross-validation for each drug and each algorithm. Figure 9.9 shows the comparison between vector machines and peptide machines. It can be seen that the peptide machines outperformed the vector machines. We also constructed neural networks models for this task, their performance is far worse than vector machines (data are not shown).

4. Practical Issues in Peptide Classification

In building a computer model for a real application, there are two important issues for model validation in addition to model parameter estimation. The first is how to evaluate a model. The second is how to select a model based on the evaluation. There are always some hyperparameters for determination when using neural networks or machine learning algorithms. For instance, the number of hidden neurons in feedforward neural networks is such a hyperparameter, and the determination of the optimal number of hidden neurons is not an easy task. Using the training data for the evaluation will not deliver meaningful result, as an optimization process can easily overfit the data, leading to poor generation capability. The generalization capability should measure how well a built model works with novel data.

Based on this requirement, we then need to consider how to use the available data for proper model validation. There are three commonly used methods for this purpose: cross-validation, resampling, and jackknife. All these methods use the same principle, that the validation data must not involve any process of model parameter estimation. This means that the available data must be divided into two parts. One is for model parameter estimation, which is commonly referred to as *training*. The other is for model evaluation (validation) and selection. The difference between these three methods is the strategy used for the division of a given data set.

4.1. Resampling

With the resampling method, we normally randomly sample a certain percentage of data for training and the rest for validation. Such a process is commonly repeated many times. Suppose there are *n* data points and we repeat the sampling process for *m* times. There will be *m* validation models, each of which has different training and validation data with some possible overlap. Note that all these validation models use the same hyperparameters; for instance, they all use *h* hidden neurons. The parameters of the *i*th validation model are estimated using the *i*th training data set with $k_i < n$ data points. The *i*th validation model is validated on the *i*th validation data set with $n - k_i$ data points. Because we use different training data sets each time, it is expected that the parameters of the *i*th validation model will be different from those of the *j*th validation model. The validation performance of the *i*th validation model is certainly different from that of the *j*th validation model as well. We denote by ω_i^{ϑ} the evaluated performance for the *i*th validation model. The evaluation statistic of the model with designed hyperparameters ϑ can follow

$$\mu_{\vartheta} = \frac{1}{m}\sum_{i=1}^{m}\omega_i^{\vartheta} \text{ and } \sigma_{\vartheta}^2 = \sum_{i=1}^{m}(\omega_i^{\vartheta} - \mu_{\vartheta})^2 \tag{29}$$

To determine the proper values for the hyperparameters so as to select a proper model, we can vary the values assigned to the hyperparameters. If we have *g* hyperparameters for selection, the selection will be taken in a *g*-dimensional space, where each grid is a combination of hyperparameters. Suppose we need to determine only the number of hidden neurons, we then have a series of μ_{ϑ} and σ^2_{ϑ}. The best model can be selected through

$$\boldsymbol{H} = \arg\max\{\boldsymbol{\mu}_v\} \sim \arg\min\{\boldsymbol{\sigma}_v^2\} \tag{30}$$

It should be noted that, for the resampling method, some data points may be used for multiple times in training or validation.

4.2. Cross-Validation

In cross-validation, we normally randomly divide a data set into m folds. Each fold contains distinctive data points. If we denote by Ω_i as the set of data points in ith fold, we have $\Omega_i \cap \Omega_j = \varphi$, meaning that two sets have no elements in common. Every time, we select one fold as the validation set and the remaining $m - 1$ folds are used as the training set for model parameters estimate. Such a process is repeated for m times, until each fold has been used for validation once. This means that there are m validation models. Again, all these validation models use the same hyperparameters. The ith validation model is trained using the folds except for the ith fold and validated on the ith fold. The parameters of the ith validation model also is different from those of the jth validation model, and the validation performance of different validation models will vary. Note that each data point ise validated only once. We denote by ω_i^ϑ the evaluated performance for the ith data point. The evaluation of the model with designed hyperparameters can follow

$$\mu_\vartheta = \frac{1}{m}\sum_{i=1}^{m}\omega_i^\vartheta \text{ and } \sigma_\vartheta^2 = \sum_{i=1}^{m}(\omega_i^\vartheta - \mu_\vartheta)^2 \tag{31}$$

Suppose we need to determine only the number of hidden neurons, we then have a series of μ_ϑ and σ_ϑ^2. The best model can be selected through the use of Eq. 30.

4.3. Jackknife

When data size is not too large, one commonly prefers using the jackknife (often called *leave-one-out cross-validation*) method. In using the jackknife method, we normally pick up one data point for validation and use the remaining data points for training. Such a process is repeated for n times until each data point has been exhausted for validation. This means that there are n validation models for n data points. Obviously, all these validation models use the same hyperparameters. The ith validation model is trained using all the data points except for the ith data point and validated on the ith data point. Equation 31 can be used for evaluation. The best model can be selected through the use of Eq. 30.

4.4. Blind Test

Model validation can help us evaluate models and select models with a proper setting of hyperparameters. However, such evaluation values cannot be regarded as the final model evaluation, which can be delivered to users. The evaluated performance on validation data may not be able to reflect the true performance, because the validation data has been used to tune hyperparameters. As a consequence, the evaluated performance on validation data commonly overestimates the true model accuracy.

Based on this understanding, a proper peptide classification methodology must contain a blind test stage. This means that, after model evaluation and model selection, we may need to test the "best" model using another data set, which has never been used for estimating model parameters and tuning model hyperparameters or selecting models [7].

4.5. Protein-Oriented Validation

A very important issue must be discussed here, especially for peptide classification. Many peptide classification tasks deal with functional site predictions. Within each protein sequence, there might be a few functional sites, such as protease cleavage sites, protein interaction sites, or protein posttranslational modification sites. Based on the substrate size, we can extract a peptide around one functional site. Such a peptide is commonly regarded as a functional peptide. To conduct proper peptide classification, we have to have a set of nonfunctional peptides. Each nonfunctional peptide has no functional site at the desired residue(s). We normally use a sliding window with a fixed length to scan a protein sequence from the N-terminal to the C-terminal, one residue by one residue to generate no-functional peptides. A commonly used method is to combine both functional and nonfunctional peptides to produce a data set. Suppose we use the cross-validation method, the data set is then randomly divided into *m* folds for cross-validation. One then builds *m* validation models for model evaluation and model selection. Now a question arises. When we use such kind of models for testing on novel whole protein sequences, the performance is commonly not as expected. This means that the model is some where overfitted. If model parameter estimation follows a proper procedure, the most probable problem is the data used for training and validation. We mentioned previously that we must maintain the independence of a validation data set from a training data set. When we check the method as described, we can clearly see two interesting facts. First, a training peptide and a validation peptide may come from the same protein. If one protein has conserved amino acids, the validation peptides generated this way may not be independent of the training peptides. Second, a training peptide and a validation peptide may have many identical residues if they are extracted from neighboring sliding widows.

Based on this analysis, we proposed to use protein-oriented validation [24]. Suppose we have *n* proteins, we may divide these *n* proteins into *m* folds. We then generate validation peptides using one fold and generate training peptides from the remaining folds. The validation models are constructed using the training peptides scanned from the sequences of the training proteins and verified on the validation peptides scanned from the sequences of the validation proteins.

4.6. Model Lifetime

We always have an important issue when using machine learning approaches for peptide classification, That is, if the model is correct forever. The answer is no, as a peptide data set collected at a certain time can be far less than complete. Based on this understanding, the models built using an incomplete set of peptides may not be able to generalize well forever. When new experimentally determined peptides have been collected, the exiting models must be corrected so that they can conduct classification tasks well. In this section, we investigate an interesting peptide classification topic, where we can use a model built on published peptides for peptide classification, and we can use a model built on both published peptides and newly submitted sequences in NCBI (www.ncbi.nih.gov) for peptide classification. We found there is a significant difference between the two.

The case studied in this session is about hepatitis C virus (HCV), which is a member of the flaviviridae family [55] and is the major agent of parenterally transmitted non-A/non-B hepatitis [56, 57]. The nomenclature of Schechter and Berger [1] is applied to designate the cleavage sites on the peptides, P_6-P_5-P_4-P_3-P_2-P_1-P_1'-P_2'-P_3'-P_4', the asymmetric scissile bond being between P_1 and P_1'. Two resources are available for modeling. First, research articles have published experimentally determined peptides, cleaved and noncleaved. Second, some databank like NCBI has collected many new submissions. Each submission contains a whole protein sequence with a number of experimentally determined cleavage sites. In terms of this, there are two strategies for modeling. If we believe that the published peptides are able to represent all the protease cleavage specificity, we can select indicator peptides from these published peptides for modeling. We refer to a model constructed this way as a *Type-I model*. In fact, viruses mutate very rapidly; hence, the published peptides may not be able to represent the cleavage specificity in the recent submissions to NCBI. We can then select more indicator peptides, which show more viral mutation information from the new sequences downloaded from NCBI to build a more informative model referred to as a *Type-II model*.

From published papers (data are not shown), we collected 215 experimentally determined peptides. They are referred to as *published peptides* in this study. Among them, 168 are cleaved, while 47 are noncleaved. Twenty-five whole protein sequences were downloaded from NCBI. They are aaa65789, aaa72945, aaa79971, aab27127, baa01583, baa02756, baa03581, baa03905, baa08372, baa09073, baa09075, baa88057, bab08107, caa03854, caa43793, cab46677, gnwvtc, jc5620, np_043570, np_671491, p26663, p26664, p27958, pc2219, and s40770. Within these 25 whole protein peptides (data are not shown) are 123 cleavage sites. The cleavage sites with notes of POTENTIAL, BY SIMILARITY, OR PROBABLE were removed.

First, we built Type-I models using the leave-one-out method. Each model is built on 214 published peptides and tested on 1 remaining published peptide. The models are then evaluated in terms of the mean accuracy, which is calculated on 215 leave-one-out testes. The best model is selected and tested on 25 new sequences downloaded from NCBI, which are regarded as the *independent* testing data. The simulation shows that the noncleaved, cleaved, and total accuracies are 77%, 77%, and 77%, respectively. Shown in Fig. 9.10 are the prediction results on five whole protein sequences, where the horizontal axis indicates the residues in whole protein sequences and the vertical axis the probability of positive (cleavage). If the probability at a residue is larger than 0.5, the corresponding residue is regarded as the cleavage site P1. The simulation shows that there were too many false positives. The average of false positive fraction was 27%, or 722 misclassified noncleavage sites.

Second, we built Type-II models. Each model is built and validated using all the published peptides (215) plus the peptides scanned from 24 newly downloaded sequences. In this run, the protein-oriented leave-one-out method is used. With this method, 1 of 24 new sequences is removed for validation on a model built using 215 published peptides and the peptides scanned from the remaining 23 new sequences. This is repeated 24 times and the performance is estimated on the predictions on 24 new sequences. The best model is selected and tested on the peptides on the remaining new sequence. These peptides are regarded as the *independent* testing peptides. The noncleaved, cleaved, and total accuracies are 99%, 83%, and 99%, respectively. It can be seen that the prediction accuracy has been greatly improved compared with the Type-I models. Shown in Fig. 9.11 is a comparison between the Type-I and the Type-II models. It shows that the Type-II models greatly outperformed the Type-I models in terms of the performance for noncleaved peptides. More important, the standard deviation in the Type-II models is much smaller, demonstrating high robustness.

Shown in Fig. 9.12 are the prediction results on five protein sequences using the Type-II models, where the horizontal axis indicates the residues in whole protein

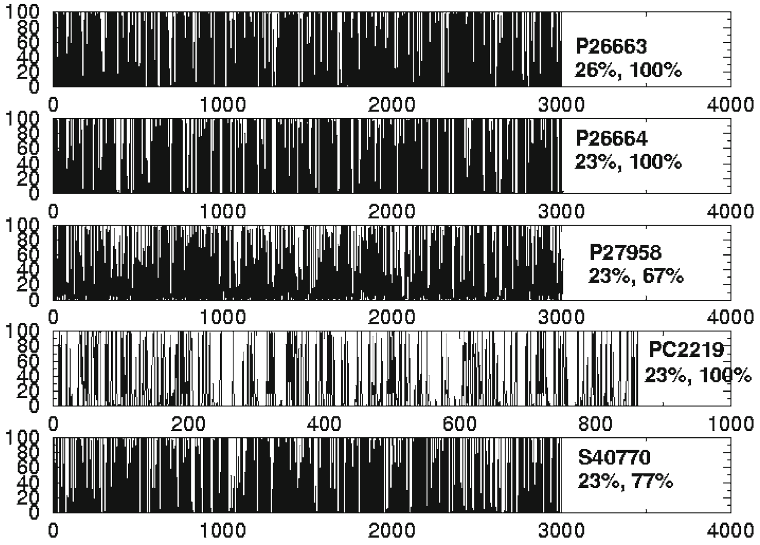


Fig. 9.10 The probabilities of cleavage sites among the residues for five whole protein sequences for the Type-I models. The horizontal axes indicate the residues in the whole sequences and the vertical axes the probabilities of positives (cleavage sites). The numbers above the percentages are the NCBI accession numbers and the percentages are the false-positive fractions and the true-positive fractions. Reproduced from [58] with permission

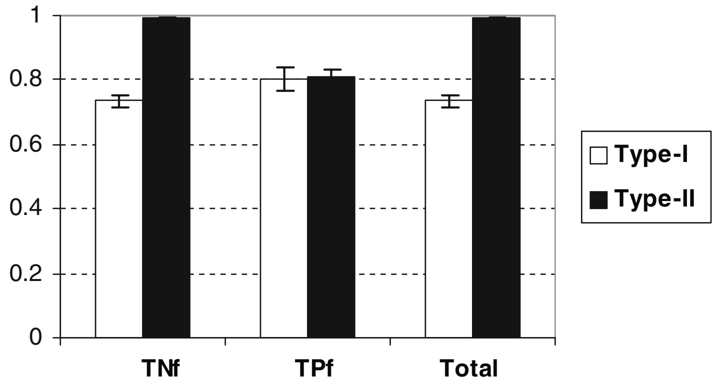


Fig. 9.11 A comparison between the Type-I and Type-II models. It can be seen that the Type-II models performed much better than the Type-I models in terms of the increase of specificity (true-negative fraction). Therefore, the false-positive fraction (false alarm) has been significantly reduced. TNf and TPf stand for true-negative fraction (specificity) and true-positive fraction (sensitivity). Reproduced from [58] with permission

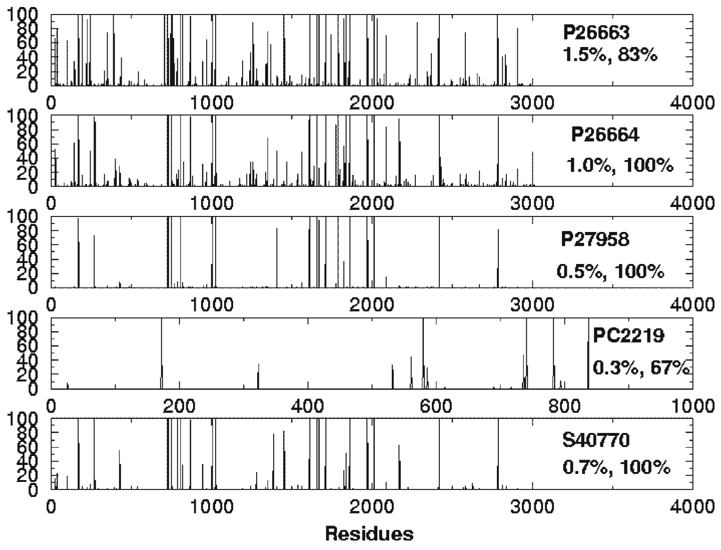


Fig. 9.12 The probabilities of cleavage sites among the residues for five whole protein sequences for the Type-II models. The horizontal axes indicate the residues in whole sequences and the vertical axes the probabilities of positives (cleavage sites). The numbers above the percentages are the NCBI accession numbers and the percentages are the false positive fractions and the true positive fractions. The simulation shows that the Type-II models demonstrated very small false-positive fractions compared with the Type-I models. It can be seen that the probability of positives are very *clean*. Reproduced from [58] with permission

sequences and the vertical axis the probability of positive (cleavage). The simulation shows fewer false positives than the Type-I models. The reason is that many of the 25 newly downloaded sequences were published after the reports with the published peptides. The published peptides may not contain complete information for all these 25 new sequences.

5. Summary and Future Work

This chapter introduced the state-of-the-art machines for peptide classification. Through the discussion, we can see the difference between the peptide machines and other machine learning approaches. Each peptide machine combines the feature extraction process and the model construction process to improve the efficiency. Because peptide machines use the novel bio-basis function, they can have biologically well-coded inputs for building models. This is the reason that the peptide machines outperformed the other machine learning approaches. The chapter also discussed some issues related with peptide classification, such as model evaluation, protein-oriented validation, and model lifetime. Each of these issues is important in terms of building correct, accurate, and robust peptide classification models. For instance, the blind test can be easily missed by some new bioinformatics researchers. Protein-oriented validation has not yet been paid enough attention in bioinformatics. There is also little discussion on model lifetime. Nevertheless, these important issues have been evidenced in this chapter.

We should note that there is no a priori knowledge about which peptide machine should be used. Research into the link between these three peptide machines may provide some important insights for building better machines for peptide classification. The core component of peptide machines is a mutation matrix. Our earlier work shows that the model performance using various mutation matrices varies [10]. It is then interesting to see how we can devise a proper learning method to optimize the mutation probabilities between amino acids during model construction.

References

1. Schechter I, Berger A (1968) On the active site of proteases, 3. Mapping the active site of papain; specific peptide inhibitors of papain. Biochem Biophys Res Comms 32:898.
2. Blom N, Gammeltoft S, Brunak S (1999) Sequence and structure based prediction of eukaryotic protein phosphorylation sites. J Mol Biol 294:1351–1362.
3. Kobata A (1984) The carbohydrates of glycoproteins. In: GinsburgV, Robins PW (eds) Biology of carbohydrates. Wiley, New York.
4. Yang ZR, Wang L, Young N, Trudgian D, Chou KC (2005) Machine learning algorithms for protein functional site recognition. Current Protein and Peptide Science 6:479–491.
5. Dunker AK, Obradovic Z, Romero P, Garner EC, Brown CJ (2000) Intrinsic protein disorder in complete genomes. Genome Inform Ser Workshop Genome Inform 11:161–171.

6. Baldi P, Pollastri G, Andersen C, Brunak S (2000) Matching protein beta-sheet partners by feedforward and recurrent neural networks. Proceedings of International Conference on Intelligent Systems for Molecular Biology, ISMB 8:25–36.
7. Yang ZR, Thomson R, McNeil P, Esnouf R (2005) RONN: use of the bio-basis function neural network technique for the detection of natively disordered regions in proteins. Bioinformatics 21:3369–3376.
8. Qian N, Sejnowski TJ (1988) Predicting the secondary structure of globular proteins using neural network models. J Molec Biol 202:865–884.
9. Thomson R, Hodgman TC, Yang ZR, Doyle AK (2003) Characterising proteolytic cleavage site activity using bio-basis function neural networks. Bioinformatics 19:1741–1747.
10. Yang ZR, Thomson R (2005) A novel neural network method in mining molecular sequence data. IEEE Trans on Neural Networks 16:263–274.
11. Altschul SF, Gish W, Miller W, Myers E, Lipman, DJ (1990) Basic local alignment search tool. J. Molec. Biol. 215:403–410.
12. Dayhoff MO, Schwartz RM, Orcutt BC 1978 A model of evolutionary change in proteins. matrices for detecting distant relationships. In: Dayhoff MO (ed) Atlas of protein sequence and structure, 5:345–358.
13. Johnson MS, Overington JP (1993) A structural basis for sequence comparisons—an evaluation of scoring methodologies. J Mol Biol 233:716–738.
14. Yang ZR (2004) Application of support vector machines to biology. Briefings in Bioinformatics 5:328–338.
15. Edelman J, White SH (1989) Linear optimization of predictors for secondary structure: application to transbilayer segments of membrane proteins. J Mole Biol 210:195–209.
16. Yang ZR, Berry E (2004) Reduced bio-basis function neural networks for protease cleavage site prediction. J Comput Biol Bioinformatics 2:511–531.
17. Thomson R, Esnouf R (2004) Predict disordered proteins using bio-basis function neural networks. Lecture Notes in Computer Science 3177:19–27.
18. Berry E, Dalby A, Yang ZR (2004) Reduced bio basis function neural network for identification of protein phosphorylation sites: comparison with pattern recognition algorithms. Comput Biol Chem 28:75–85.
19. Senawongse P Dalby AD, Yang ZR (2005) Predicting the phosphorylation sites using hidden Markov models and machine learning methods. J Chem Info Comp Sci 45:1147–1152.
20. Yang ZR, Chou KC (2004) Bio-basis function neural networks for the prediction of the O-linkage sites in glyco-proteins. Bioinformatics 20:903–908.
21. Sidhu A, Yang ZR (2006) Predict signal peptides using bio-basis function neural networks. Applied Bioinformatics 5(1):13-19.
22. Yang ZR, Dry J, Thomson R, Hodgman C (2006) A bio-basis function neural network for protein peptide cleavage activity characterisation. Neural Networks 19:401–407.
23. Yang ZR, Young N (2005) Bio-kernel self-organizing map for HIV drug resistance classification. Lecture Notes in Computer Science 3610:179–184.
24. Yang ZR (2005) Prediction of caspase cleavage sites using Bayesian bio-basis function neural networks. Bioinformatics 21:1831–1837.
25. Yang ZR (2005) Mining SARS-coV protease cleavage data using decision trees, a novel method for decisive template searching. Bioinformatics 21:2644–2650.
26. Yang ZR, Johanathan F (2005) Predict T-cell epitopes using bio-support vector machines. J Chem Inform Comp Sci 45:1142–1148.
27. Vapnik V (1995) The nature of statistical learning theory. Springer-Verlag, New York.
28. Scholkopf B (2000) The kernel trick for distances. Technical report, Microsoft Research, May.
29. Tipping ME (2001) Sparse Bayesian learning and the relevance vector machine. J Machine Learning Res 1:211–244.
30. Chen S, Cowan CFN, Grant PM (1991) Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans on Neural Networks 2:302–309.
31. Yang ZR, Chou KC (2003) Bio-support vector machines for computational proteomics. Bioinformatics 19:1–7.

32. MacKay DJ (1992) A practical Bayesian framework for backpropagation networks. Neural Computation 4:448–472.
33. Yang ZR (2005) Orthogonal kernel machine in prediction of functional sites in proteins. IEEE Trans on Systems, Man and Cybernetics 35:100–106.
34. Yang ZR, Thomas A, Young N, Everson R (2006) Relevance peptide machine for HIV-1 drug resistance prediction. IEEE Trans on Computational Biology and Bioinformatics (in press).
35. Putney, S (1992) How antibodies block HIV infection: paths to an AIDS vaccine. Trends in Biochem Sci 7:191–196.
36. Klausner RD et al. (2003) The need for a global HIV vaccine enterprise. Science 300:2036–2039.
37. Kathryn S (2003) HIV vaccine still out of our grasp. The Lancet Infectious Diseases 3:457.
38. Weber IT, Harrison RW (1999) Molecular mechanics analysis of drug-resistant mutants of HIV protease. Protein Eng 12:469–474.
39. Jenwitheesuk E, Samudrala R (2005) Prediction of HIV-1 protease inhibitor resistance using a protein-inhibitor flexible docking approach. Antivir Ther 10:157–166.
40. Gallego O, Martin-Carbonero L, Aguero J, de Mendoza C, Corral A, Soriano V (2004) Correlation between rules-based interpretation and virtual phenotype interpretation of HIV-1 genotypes for predicting drug resistance in HIV-infected individuals. J Virol Methods 121:115–118.
41. De Luca A, Cingolani A et al. (2003) Variable prediction of antiretroviral treatment outcome by different systems for interpreting genotypic human immunodeficiency virus type 1 drug resistance. J Infect Dis 187:1934–1943.
42. Shenderovich MD, Kagan RM, Heseltine PN, Ramnarayan K (2003) Structure-based phenotyping predicts HIV-1 protease inhibitor resistance. Protein Sci 12:1706–1718.
43. Brenwinkel N, Schmidt B, Walter H, Kaiser R, Lengauer T, Hoffmann D, Korn K, Selbig J (2002) Diversity and complexity of HIV-1 drug resistance: a bioinformatics approach to predicting phenotype from genotype. PNAS 99:8271–8276.
44. Sturmer M, Doerr HW, Staszewski S, Preiser W (2003) Comparison of nine resistance interpretation systems for HIV-1 genotyping. Antivir Ther 8:239–244.
45. Vergu E, Mallet A, Golmard JL (2002) The role of resistance characteristics of viral strains in the prediction of the response to antiretroviral therapy in HIV infection. J Acquir Immune Defic Syndr 30:263–270.
46. Smith CJ, Staszewski S et al. (2004) Use of viral load measured after 4 weeks of highly active antiretroviral therapy to predict virologic putcome at 24 weeks for HIV-1-positive individuals. J Acquir Immune Defic Syndr 37:1155–1159.
47. Beerenwinkel N, Daumer M et al. (2003) Geno2pheno: estimating phenotypic drug resistance from HIV-1 genotypes. Nucleic Acids Res 31:3850–3855.
48. Foulkes AS, De GV (2002) Characterizing the relationship between HIV-1 genotype and phenotype: prediction-based classification. Biometrics 58:145–156.
49. De Luca A, Vendittelli et al. (2004) Construction, training and clinical validation of an interpretation system for genotypic HIV-1 drug resistance based on fuzzy rules revised by virological outcomes. Antivir Ther 9:583–593.
50. Draghici S, Potter RB (2003) Predicting HIV drug resistance with neural networks. Bioinformatics 19:98–107.
51. Shafer RW, Stevenson D, Chan B (1999) Human immunodeficiency virus reverse transcriptase and protease sequence database. Nucleic Acids Res 27:348–352.
52. Walter H, Schmidt B, Korn K, Vandamme AM, Harrer T, Uberla K (1999) Rapid, phenotypic HIV-1 drug sensitivity assay for protease and reverse transcriptase inhibitors. J Clin Virol.13:71–80.
53. Sa-Filho DJ, Costa LJ, de Oliveira CF, Guimaraes APC, Accetturi CA, Tanuri A, Diaz RS (2003) Analysis of the protease sequences of HIV-1 infected individuals after indinavir monotherapy. J Clin Virology 28:186–202.
54. Matthews BW (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta 405:442–451.

55. Francki R, Fauquet C, Knudson D, Brown F (1991) Classification and nomenclature of virus: fifth report of the International Committee on Taxonomy of Viruses. Arch Virology 2:223s.
56. Choo Q, Kuo G, Weiner AJ, Overby L, Bradley D, Houghton M (1989) Isolation of a cDNA clone derived from a blood-borne non-A non-B viral hepatitis genome. Science 244:359–362.
57. Kuo G, Choo Q et al. (1989) An assay for circulating antibodies to a major etiologic virus of human non-A non-B hepatitis. Science 244:362–364.
58. Yang ZR (2006) Predicting hepatitis C virus protease cleavage sites using generalised linear indicator regression models. IEEE Trans on Biomedical Engineering 53:2119–2123.

Chapter 10
Associative Neural Network

Igor V. Tetko

Abstract An associative neural network (ASNN) is an ensemble-based method inspired by the function and structure of neural network correlations in brain. The method operates by simulating the short- and long-term memory of neural networks. The long-term memory is represented by ensemble of neural network weights, while the short-term memory is stored as a pool of internal neural network representations of the input pattern. The organization allows the ASNN to incorporate new data cases in short-term memory and provides high generalization ability without the need to retrain the neural network weights. The method can be used to estimate a bias and the applicability domain of models. Applications of the ASNN in QSAR and drug design are exemplified. The developed algorithm is available at http://www.vcclab.org.

Keywords Ensemble networks, memory, drug design, LIBRARY mode

1. The Biological Background to Associative Neural Networks

The understanding of brain function and, in particular, how the brain represents and uses information remains one of the main challenges of modern neurophysiology. The existence of two types of memory, short- and long-term memory, has been known for a long time [1]. The first type is produced by local neuronal circuits, and the second type of memory is encoded by the neural network connections. New information is first stored as short-term memory, and after some time, it is consolidated as long-term memory. The associative neural network (ASNN) method was inspired by these neurophysiologic properties of the brain and makes use of both types of memory. This provides high generalization ability and other properties of the approach considered in this paper.

2. Method

2.1. ASNN Combines Global and Local Models

The associative neural network [2–4] is a combination of a global regression or classifier learning method (GM) that is usually composed of an ensemble of neural networks and a local learning method (LM), such as k nearest neighbors (kNN) or Parzen window regression. The role of the GM is to learn the global properties of the analyzed task, while the LM is used to correct the bias of the global model. The main idea of ASNN is that the GM may not be flexible enough to capture the full diversity of the data set. Therefore the use of LM can refine the global model.

To some extent, the development of ASNN resembles drawing a picture by a painter. The artist first draws a global landscape of the picture, containing the sea, sky, or mountains. He or she uses big brushes to draw it then uses small brushes to add tiny but very important details, like birds or flowers. The order of drawing as well as the instruments used cannot be easily interchanged. The context of the global picture determines the placement of local elements; for example, the birds should fly in sky and mushrooms grow in forest. The GM and LM have an analogous relationship. The GM captures the overall variation of data, while the LM covers local variations of the model. Moreover, the use of the global context determines which local elements are allowed in each region of the global space.

2.2. Mathematical Formulation of the ASNN

Let us restrict our analysis to a regression problem. The extension for classification is straightforward. Consider an ensemble of *m* global models:

$$[\mathbf{ME}]_m = \begin{bmatrix} M_1 \\ \vdots \\ M_j \\ \vdots \\ M_m \end{bmatrix} \tag{1}$$

The prediction of a case $\mathbf{x}_i$, $i = 1, \ldots, N$ then corresponds to a vector of calculated values $z_i = z^i_j$

where $j = 1, \ldots, m$ is the index of the model within the ensemble:

$$[\mathbf{x}_i] \bullet [\mathbf{ME}]_M = [\mathbf{z}_i] = \begin{bmatrix} z_1^i \\ \vdots \\ z_j^i \\ \vdots \\ z_m^i \end{bmatrix} \tag{2}$$

A simple average

$$\overline{z}_i = \frac{1}{m} \sum_{j=1,\ldots,m} z_j^i \tag{3}$$

provides the ensemble response. A number of ensemble methods have been published [5–8]. Why is it beneficial to use an ensemble rather than the output from a single model? It is known that the expected error of a regression model can be decomposed as follows:

$$\mathrm{PE}(f_i) = (f_i - Y)^2 = (f^* - Y)^2 + (f^* - \overline{f})^2 + (f_i - \overline{f})^2 = E\varepsilon^2 + \mathrm{Bias}(f) + \mathrm{Var}(f_i) \tag{4}$$

where f_i is a value estimated by a model i, Y is a target value generated with noise ε, f^* is a true target value generated for a function with zero noise $\varepsilon = 0$, and $\bar{f}$ is the mathematical expectation calculated over the ensemble of models. Therefore, the expected error consist of three terms: $E\varepsilon^2$, which measures the amount of intrinsic noise in the data; variance $\mathrm{Var}(f_i)$; and model bias $\mathrm{Bias}(f)$. The ensemble average (provided regression models are not correlated) decreases the variance $\mathrm{Var}(f_i)$ of the results. However, the ensemble average does not decrease the bias of the model, which can be defined as a consistent deviation from the true model for some neighboring points.

If we can correctly detect some nearest neighbors of an analyzed point, then we can measure bias (consistent error) and use the estimated value of the bias to correct our prediction. The main question is how to determine the "true" nearest neighbors that have the same bias. To do this, the ASNN uses the global model.

The ASNN introduces a measure of proximity of two input cases i,j in the space of models as a correlation between vectors of their predicted values (or vector of residuals of predicted values):

$$r_{ij} = \text{correlation } (\mathrm{z}_i, \mathrm{z}_j) \tag{5}$$

This type of similarity measure was first introduced in 1995 [9] and formed the core of the efficient partition algorithm [10–12] developed to increase the speed and accuracy of neural network learning. Other similarity measures, such as the Spearman nonparametric rank correlation coefficient or (inverse) Euclidian distance, were recently analyzed elsewhere [13]. We define $r_{ij} = 0$ for negative values $r_{ij} < 0$. For each analyzed data case *i*, we can find *k* cases in the training set having the highest similarity r_{ij} to it. Let us designate such neighbors as $N_k(\mathbf{x}_i)$. The ASNN correct the ensemble value of $\overline{z}_i$ according to the formula is

$$\overline{z}_i' = \overline{z}_i + \tfrac{1}{k} \sum_{j \in N_k(\mathbf{x}_i)} \left(y_j - \overline{z}_j \right) \tag{6}$$

where y_i are the target values of the corresponding nearest neighbors. Since the variance of ensemble prediction $\overline{z}_i$ can be made small by increasing the number of neural networks in the ensemble, the difference $(y_i - \overline{z}_i)$ corresponds mainly to the bias of the neural network ensemble for the case $\mathbf{x}_i$. Thus, this formula explicitly corrects the bias of the analyzed case according to the observed biases calculated for the neighboring cases. Note that, if the nearest neighbors are selected by chance, the correction term will be random and will not correct the bias of the model. Moreover, it will increase its variance and thus decrease its prediction performance.

2.3. *Properties of the ASNN*

2.3.1. ASNN Estimates and Corrects the Bias of the GM

The ASNN may dramatically improve the prediction ability of the GM. The degree to which the ASNN can do this depends on the GM itself. The more biased the model, the greater improvement in its prediction ability can be achieved using the ASNN method. The unbiased model, of course, cannot be improved.

For example, an ensemble of 100 neural networks with one hidden neuron did not provide a good interpolation of $y = \mathbf{x}^2$ function for $\mathbf{x} = [-2, 2]$, where $\mathbf{x} = (x_1, x_2) = x_1 - x_2$ due to underfitting (Fig. 10.1a). The neural networks with two hidden neurons provided better interpolation of the function but still could not interpolate some regions of the function, particularly its extreme points $x = \{-1, 0, 1\}$. The ASNN correction using Eq. 6 significantly improved interpolation accuracy for both examples. The improvement was more spectacular for the more biased ensemble of neural networks with one hidden neuron.

A similar improvement in the performance of the ASNN was observed in a number of other studies. For example, an ensemble of 100 cascade correlation neural networks (CCNN) [14] was used to predict letter and satellite data sets from the UCI database [15], which are frequently used to benchmark machine learning methods (Table 10.1).

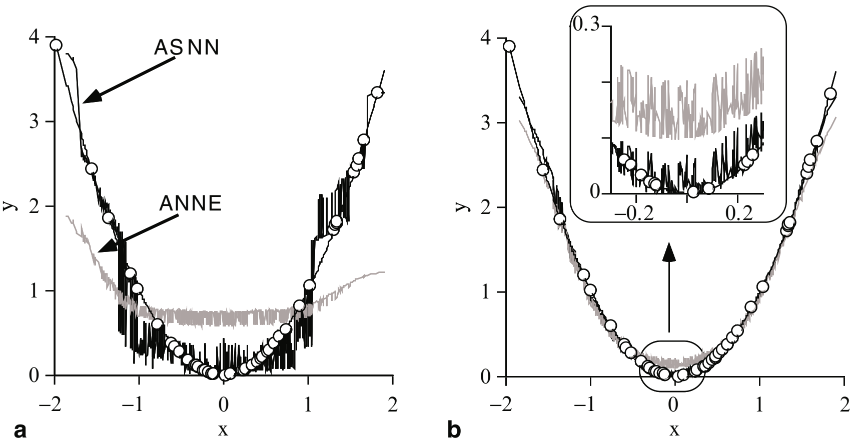


Fig. 10.1 Interpolation of the $y = x^2 = (x_1 + x_2)^2$ function with then artificial neural network ensemble (ANNE) composed of 100 networks and the associative neural networks. The training data points are shown as circles. **(a)** Neural networks with one hidden neuron show clear underfitting. **(b)** Neural networks with two hidden neurons provide a much better interpolation but have significant biases near the extreme values of the function. The ASNN corrects the biases of both ANNE

Table 10.1 Error rates on the UCI data set

Data set	Test set size	NN boosted	CCNN Ensemble	ASNN
UCI, letter	4,000	1.5%	4.1%	1.8%
UCI, satellite	2,000	8.1%	8.2%	7.8%

Note: Results of neural networks with 16-70-50-26 (letter) and 36-30-15-6 (satellite) neurons from ref [*16*].

The bias correction decreased the error rate of the ensemble by 0.5% and achieved the best published performance for the satellite set [3]. A greater improvement of the prediction ability in this study, however, was observed for the letter data set. The ensemble of CCNN had difficulty classifying 26 Latin letters and achieved error rate of only 4.1%, far less than the best result of 1.5% using the two-hidden layer neural network by Schwenk and Bengio [16]. The use of Eq. 6 decreased the error rate of the ensemble to 1.8% for this data set. Such a dramatic increase in the performance clearly indicates that the CCNN was not appropriate for the given task and the global model was biased. Contrary to the performance with letters, the prediction performance of the CCNN for the satellite data set was already nearly optimal, the solutions were relatively unbiased, and improvement in prediction ability was not so dramatic.

In some cases, the search for a less biased model may require considerable computer resources. For example, the development of the ALOGPS [4, 13, 17] program was done using the Levenberg-Marguardt [18] algorithm and 12,908 compounds from the PHYSPROP database [19]. In one cross-validation protocol, 50% of the compounds from the training set were used to develop an ensemble of 100 neural networks. At the end of training, the ensemble was used to predict the remaining 50% of the compounds. The use of neural networks with five hidden neurons required 2.5 days of CPU time (Athlon 800 MHz) and achieved RMSE = 0.53 for the training set. The training of neural networks with ten hidden neurons required 52 days on the same computer and achieved a RMSE = 0.51. ASNN achieved a RMSE = 0.49 and 0.48 using the ensemble with five and ten hidden neurons, respectively. The ASNN bias correction was performed in less than ten minutes of CPU time. This result indicates that neural networks both with five and ten hidden neurons were biased. The use of larger networks or neural networks with different architecture or the development of neural networks using different training algorithms could probably result in a method that would provide results similar to the ASNN. However, the search for such a method or neural network architecture may require considerable time and effort.

In studies to predict 1H NMR chemical shifts [20–22], the use of ASNN decreased error of the neural network ensemble from RMSE = 0.29 to 0.19 units for a test set of 952 compounds [21]. For some specific classes of compounds, such as nonaromatic *p*, the error decreased two times from RMSE = 0.39 to RMSE = 0.19. Smaller decreases of errors were calculated for other groups of compounds. The nonaromatic *p* compounds were underrepresented in the training set and included just 645 compounds, while other groups each contained more than 1,000 compounds. The problem of underrepresentation, which biased the neural network performance for the series with larger number of molecules, was corrected by the ASNN method. This is exactly the same behavior observed for the ASNN in Fig. 10.1b, in which the ASNN more remarkably improved predictions for underrepresented regions of the data set near the extreme values of the function.

The ASNN program was also used in electron-topological method [23] to analyze hydroxysemicarbazides as potential antitumor agents [24], structure-antituberculosis activity relationships study in a series of 5-(4-aminophenyl)-4-substituted-2, 4-dihydro-3h-1,2,4-triazole-3-thione derivatives [25], and to study inhibitors of cyclooxygenase-2 [26]. A comparative study of the ASNN approach with several other types of neural network methods was recently published in another work [27]. ASNN were also successfully used to classify EST sequences in bioinformatics [28]. Quantitative structure-property relationship studies of metal complexes with ionophores [29] demonstrated similar prediction ability between ASNN and support vector machines [30].

In summary, we can conclude that the ASNN decreases the bias of the GM, in particular for extreme or underrepresented parts of the data space. While the number of data points with extreme values can be small, such data (i.e., the most active inhibitors, most toxic compounds, etc.) are frequently the main targets of models. Therefore, correct prediction of their activity can be very important. Apart from improving prediction ability, the ASNN can be used as a diagnostic tool to estimate bias of the GM. The use of proper methodology may help create an unbiased GM. However, for a number of applications, particularly those involving large data sets, the ASNN provides fast and reliable solutions and can save considerable resources and effort when trying to improve the solution using traditional methods.

2.3.2. LIBRARY Correction

An interesting and useful feature of the ASNN method is its ability to incorporate new data and thus enhance prediction ability without the need to change the GM. Indeed, if new data become available, one can add them for similarity searching in Eq. 6 without the need to redevelop the GM. We term the new data LIBRARY and the procedure a LIBRARY or memory correction mode.

An example of the advantage of such correction was shown for prediction of 1H NMR spectra: the authors of [21] noticed that retraining of neural networks with all data did not improve the model compared to the LIBRARY mode. Successful applications of the ASNN LIBRARY mode were demonstrated in studies to predict lipophilicity [31–33] and aqueous solubility [34, 35] of chemical compounds. Table 10.2 indicates that the use of a LIBRARY correction increased the prediction ability of the ALOGPS program (developed using the ASNN method) approximately two times for the prediction of lipophilicity of neutral and charged chemical compounds from in-house data belonging to AstraZeneca [32] and Pfizer [31]. For comparison, the results of ACD log *P*/*D* programs for the same series are also shown. Note the low prediction performance of programs for blind prediction, particularly for the prediction of lipophilicity of charged compounds, log *D* at pH 7.4. This result indicated that compounds used for drug studies in pharmaceutical firms cover different chemical space than compounds available in the public domain. The use of the LIBRARY mode made it possible to dramatically improve prediction performance of the program by correcting the bias due to different chemical diversity of compounds in the training and in the test sets.

Note that, in the preceding example, the ALOGPS program was used in the LIBRARY mode to predict lipophilicity of charged compounds, log *D*, while it was originally developed to predict lipophicility of noncharged species only. Thus, the new data were incompatible in some aspects with the training set. Why were we able to achieve good results in the LIBRARY mode? The basic explanation is that both neutral and charged compounds share a number of same parameter-property dependencies. The prediction errors from the log *D* data using the basis model could be considered as a bias of the log *P* model. Starting from the GM as a base model and taking into account that some invariants are conserved both for log *P* and log *D* models, only a few corrections (compared to the development of a new model from scratch) are required to update the basic model to fit the new data.

Table 10.2 Mean average error of ALOGPS for several datasets of two pharmaceutical companies

Data set	Molecules	Blind prediction	LIBRARY	ACD log *P*/*D*	References
AstraZeneca, log *P*	2,569	0.60	0.46	0.86	[32]
AstraZeneca, log *D*	8,122	1.27	0.49	1.05	[32]
Pfizer, log *D*	17,341	0.92	0.43	0.97	[31]
BASF, log *P*	6,100	0.67[1]	0.36[a]	n/a	[34]

[a] Estimated.

Moreover, the number of data points used to achieve good results in the LIBRARY mode could be insufficient to build a completely new model, as demonstrated at Fig. 10.2. The neural network from Fig. 10.1b was used as an "old" basic model. The quadratic function $f_1(\mathbf{x}) = 0.5 \times x^2$ scaled by a factor of 0.5 was used to generate "new", incompatible data. The difference between both functions is accounted for by the multiplier in one of them and both functions are strongly correlated. The number of data points, 10, available for the second function was too small to develop a new model for it. An application of the GM from the nonscaled quadratic function produced large errors during prediction of 1,000 test data points generated by the scaled function. The use of 10 data cases as the LIBRARY in Eq. 6 dramatically improved the prediction ability of the neural networks and allowed reasonable approximation of the scaled function.

While a number of methods restrict the allowed range of the target variable, that is, the output values in neural networks are usually restricted to the range of values of the output layer function, the use of the ASNN method allows one to overcome this limit in the LIBRARY mode. Fig. 10.3 indicates that the neural network trained to predict the Gauss function for negative x values in the LIBRARY mode interpolated very different functions. The ASNN calculated negative values for the linear function example in the LIBRARY mode, although the sigmoid function $y = 1/(1 + e^{-x})$ defined on [0, 1] was used in the GM. Fig. 10.3c also demonstrates an example in which the use of the kNN method in input space provided lower prediction ability than the ASNN.

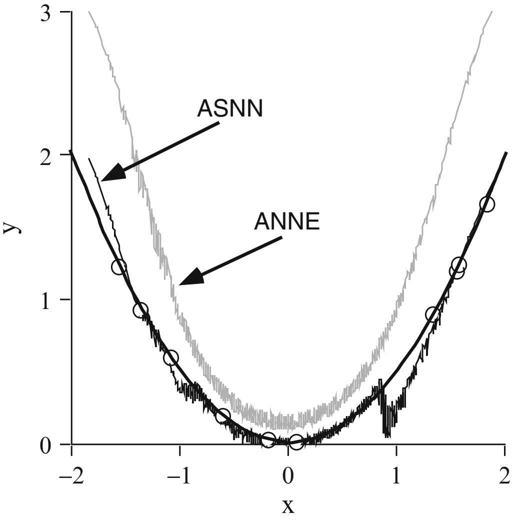


Fig. 10.2 The application of ASNN in the LIBRARY mode (memory correction). The original model (ANNE, gray line) was developed using data from the $y = x^2 = (x_1 + x_2)^2$ function shown in Fig. 10.1. The response of the ASNN in the LIBRARY mode after using ten points (circles) from the $y = 0.5x^2 = 0.5(x_1 + x_2)^2$ function is shown with a bold line

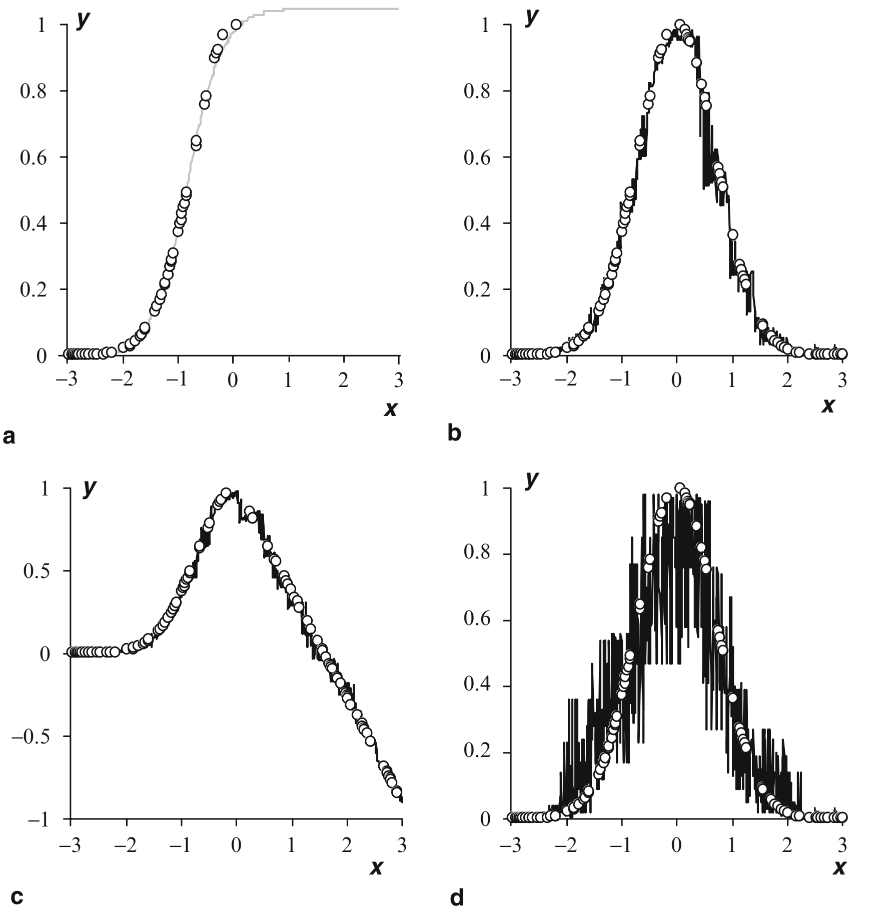


Fig. 10.3 (**a**) The gray line corresponds to the response of the ASNN trained with 50 cases (circles) that had only negative values, $\mathbf{x} = x_1 + x_2 < 0$ for the Gauss function approximation $y = \exp(-\mathbf{x}^2)$. The neural network provided good interpolation of function for the negative x values and flat response for the positive $\mathbf{x}$ values, $\mathbf{x} = x_1 + x_2 > 0$. The responses of the ASNN (black line) were changed if 50 cases (circles) from the Gauss (**b**) or linear (**c**) function were added to the ASNN memory. (**d**) The interpolation of the Gauss function using the k-nearest-neighbors method in the initial space of variables $\{x_1, x_2\}$. Reprinted with permission from [4], Copyright (2002) American Chemical Society

2.3.3. Applicability Domain of Models and Secure Data Sharing

The failure of existing methods to predict even simple physicochemical properties of compounds, such as log *P* and aqueous solubility [33, 35], may dramatically influence the drug development process and increase its cost. A development of methods using data from different companies may lead to new programs with better prediction ability. Unfortunately, pharmaceutical firms are unlikely to freely release such data in the public domain. The reason stems from the competitive nature of this business and the need of the private sector to protect the structures of those chemicals that have not been disclosed in patents or other publications. Therefore, there is an interest in developing methods for the safe exchange of chemical information and the problem of secure sharing of data is very important [36]. One possible method consists of sharing not molecules but their indices and descriptors, calculated from their molecular structure. However, theoretically, as few as one float value descriptor can be sufficient to disclose chemical structures [37]. Therefore, a different approach is required.

Another interesting and surprisingly closely related problem is estimating the accuracy of model predictions and their applicability domains. If the accuracy of a model is known or correctly estimated, when researches can decide whether the model is accurate enough to predict their compounds or whether they need to perform additional measurements to improve their model.

The ASNN provides a unified approach to address both these problems. The square of maximum correlation coefficient between test molecule and all molecules in the training set was used to estimate the accuracy of lipophilicity prediction of ALOGPS program [38, 39]. We called this coefficient the *property-based similarity*, *R*, of the target molecule to the training set. Fig. 10.4 shows that the accuracy of the

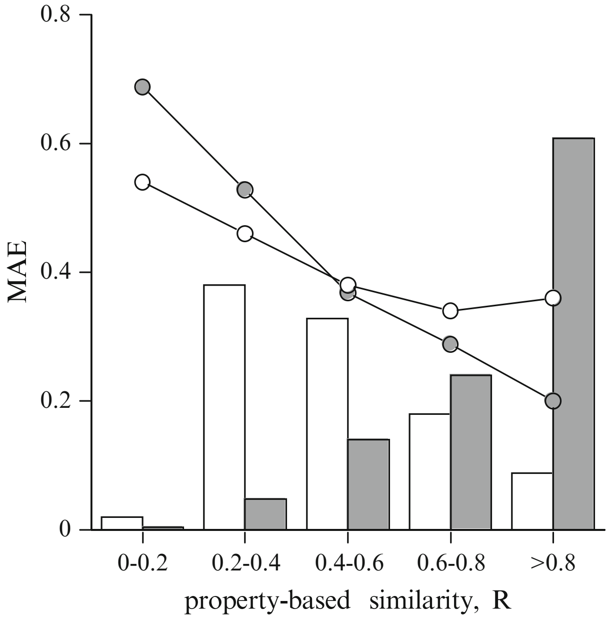


Fig. 10.4 Lines show mean absolute errors (MAE) of neural networks to predict molecules from the "nova" set (a set of molecules structurally different to the training set, see [38, 39]) as a function of property-based similarity, *R* [39]. The MAE for the blind and LIBRARY mode predictions are shown as white and grey circles, respectively. Histograms show the fraction of molecules for each range of *R* values

program developed using a subset of compounds (the so-called star set) from the PHYSPROP data set [19] can be reasonably estimated for the prediction of a chemically diverse set (the so-called nova, see the definition of both sets in [38 and 39]).

The use of the "nova" set in the LIBRARY mode changed the distribution of compounds as a function of the correlation coefficient but did not dramatically influence the accuracy of prediction for each range of *R*. A similar dependency of the accuracy of ALOGPS prediction as function of *R* was calculated elsewhere [40, 41]. We found that a fit to power function,

$$\mathrm{MAE}_{\mathrm{pred}} = 0.302 \times R^{-0.6} \tag{7}$$

where $\mathrm{MAE}_{\mathrm{pred}}$ is the predicted error and *R* is the property-based similarity can be used to analytically approximate the prediction accuracy for molecules from AstraZeneca (7,498 molecules) and Pfizer (8,750 compounds) *in-house* data sets. A similar method to estimate applicability domain of models to predict toxicity against *T. pyriformis* was also recently reported [42].

Since the property-based similarity *R* correctly estimates the accuracy of prediction, one can share molecules with confident and highly accurate predictions rather than use original molecules for model development. The surrogate data can be selected to be dissimilar to the original molecules and can be publicly shared without the danger of disclosing the original molecules [37, 39].

2.3.4. What Is Property-Based Similarity?

Let us again consider the analogy between the ASNN and the biological properties of neuronal networks in the brain. Let us explain the name of the method: associative neural networks. The problem of searching the nearest neighbors of the target compound can be considered as a search for the associations of the neural networks in memory. Indeed, basically this search answers the question, What are the most similar cases to the target compound in terms of a given property? Of course, whenever we answer this question ourselves, we always implicitly or explicitly assume some target property. For example, we can think about neighbor of a person in terms of his or her classmates, nationality, family, work friends, or house neighbors. The metric for each such neighborhood relation can be explicitly either assumed or indicated in the question. Thus, depending on the question, we weight different "attributes" of considered people to provide a correct answer (of course, some neighbors will be the same for different questions).

ASNN also finds its neighbors, that is, its associations, considering the property-based metric produced by the global model. Different target properties lead to the selection of different nearest neighbors. Fig. 10.5 illustrates that the nearest neighbors of biphenyl in lipophilicity and aqueous solubility space (Eq. 5) are very different, despite the same descriptors (E-state indices [43–45]) used to search the nearest neighbors in set of 12,908 compounds from refs [13, 17]. All analogs of biphynel in the log *S* space are more symmetric and contain two phenyl rings.

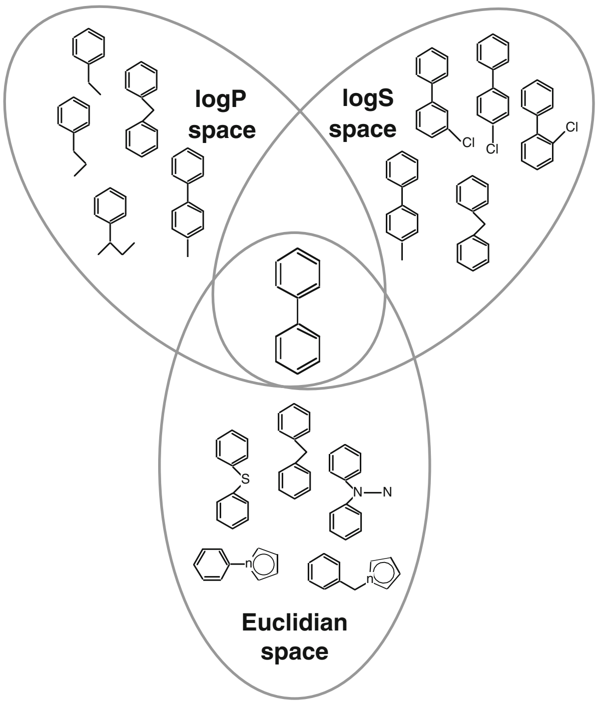


Fig. 10.5 The nearest neighbors of biphenyl calculated in the property-based lipophilicity (log *P*) and aqueous solubility (log *S*), and in the Euclidian space. The same set of 75 descriptors was used to calculate Euclidian distance and to develop log *P* and log *S* models. Only one molecule, diphenylmethane, is common to all three spaces. Reprinted from [41] Copyright (2006), with permission from Elsevier

Contrary to that, the analogs of symmetric biphenyl in the lipophilicity space also include nonsymmetric compounds containing just one phenyl ring.

To better understand the differences in the nearest neighbors for both properties one can consider the general solubility equation (GSE) of Jain and Yalkowsky [46]

$$\log S = 0.5 - 0.01(\mathrm{MP} - 25) - \log P \tag{8}$$

Apart from the melting point, aqueous solubility, log *S*, is linearly related to lipophilicity, log *P*. The melting point reflects symmetry and crystal packing of compounds, and it is very important for aqueous solubility (see, e.g., [35]) Therefore, the nearest neighbors of biphenyl in aqueous solubility space are more symmetric. The packing and thus symmetry are not so important in log *P* space and biphenyl's neighbors in this space are not symmetric.

We argue the search for "associations" given the ensemble of neural networks is analogous to the search of associations used by people. The search for an accurate answer for a given question begins by creating a prototype in the brain, which is

Table 10.3 Free and commercial programs based on the ASNN approach

No.	Name	WWW site	Description	Refs
1	ASNN	www.vcclab.org[a]	The ASNN program	[2–4]
2	ALOGPS 2.1	www.vcclab.org[a]	The ALOGPS program to predict lipophilicity and solubility of chemical compounds	[4, 13, 17]
3	SPINUS	www.chemie.uni-erlangen.de/services/spinus[b]	Prediction of 1H NMR chemical shifts	[20–22]
4	ADMET Modeler	www.simulations-plus.com	ADME/T models	
5	Trident	www.wavefun.com	Model builder for log *P*/*D* predictions	

[a]Interactive online calculation at the Virtual Computational Laboratory site [47, 48].
[b]Interactive online calculation at SPINUS site.

compared with memory patterns stored in the brain. The context of the question creates the metrics for the search, that is, activates one or another group of neurons and brain circuits, reflecting different features of the question. The memory patterns that are the most correlated with the prototype of the answer receive the highest speed of propagation in the brain, activating the target neurons responsible for the answer.

Another biologically motivated feature of the ASNN is its possibility to store new knowledge in local memory (LIBRARY) or the GM. The first type of storage allows ASNN to provide very fast adaptation to new requirements. This can be particular important if there are significant changes in conditions and the previous GM model requires some adjustment. However, if new data dominate, over time, the GM is redeveloped to incorporate them.

2.3.5. Commercially and Freely Available ASNN-Based Software

The ASNN method was first described in a 2001 article [2] followed by two publications in 2002 [3, *4*]. Since that time, a number of freely available and commercial software tools based on ASNN were developed, as summarized in Table 10.3.

3. Conclusions

The ASNN was inspired by the function of biological neurons in the brain. The use of the LM in ASNN can dramatically increase prediction ability of the GM and can easily adapt the GM to new data without the need to redevelop the GM.

The ASNN has been demonstrated as efficient tool to increase prediction ability of the GM and can be used as a diagnostic tool to measure bias of the GM. The ASNN can be also used to estimate the applicability domain of models and secure data sharing. Currently, the main applications of the method can be found in computational chemistry but ASNN can be applied in other fields of science.

4. Notes

1. In all the studies outlined here, apart from results shown in Table 10.1 and reference [3], three-layer neural networks were constructed as previously described [6]: Layer 1 consisted of input nodes, layer 2 was a hidden layer of intermediate nodes, and layer 3 was the output node. Usually three to five hidden neurons were sufficient to get good results with the ASNN approach (and many more neurons could be required to get similar performance with traditional ANN method). The logistic $f(x) = 1(1 + e^{-x})$ activation function was used both for the hidden and output nodes. The models had full interconnection from the input nodes to the hidden nodes and to the output node. The input and output values were normalized to a [0.2–0.8] interval.
2. The training of neural network was done using the so-called early stopping technique in combination with ensemble averaging (early stopping over an *ensemble*, or ESE [10–12]). Prior to each ANN analysis, the available data—the initial training set—was partitioned randomly into learning and validation data sets of equal size. In addition, if required by the protocol used (see, e.g., Table 10.1), test sets were also used. An ensemble of M = 64–100 ANNs was trained after using a random-number generator to initialize the weights of the nodes. The test sets never participated in the training and were used only to evaluate the final prediction ability of the network after termination of ANNE training. Since the partitioning of the data cases was done by chance, each data case appeared in the learning and validation sets an equal number of times (100 ± 10) on average. In a given analysis by a single ANN, each data case was included in either the learning or the validation set but never simultaneously. Following ensemble learning, it was thus possible to estimate statistical parameters for each data case from the initial training set, in that it would belong to both the learning and validation sets. The overall size of the learning and validation sets for ANNE was therefore equal to that of the initial training set (see Fig. 2 in [49]). Training of a particular network was terminated at that epoch when the RMSE for the validation set started to increase (early stopping point) rather than training to convergence for the learning set. Following training, the ANN weights were stored and used to estimate various statistical parameters of the ANN, as well as its performance for the learning, validation and test data (when available). Usually 2–3,000 epochs were used to train network with Levenberg-Marguardt [18] algorithm.
3. The ASNN used Parzen-window regression [50] to correct a final prediction of the target molecule using the errors of its nearest neighbors. The neural networks

calculated 64 output values for the analyzed molecule and molecules from the LIBRARY (i.e., training set or training set + new user-supplied data). These 64 values provided a new representation of each molecule in the space of models (Eq. 2). The pairwise Spearman rank correlations of molecules in the space of the models were calculated (Eq. 5). The molecules from the LIBRARY with highest correlations to the analyzed molecule were selected as the nearest neighbors of the target molecule. The parameters of the Parzen-window regression were optimized by minimization of the root mean squared error for the LIBRARY molecules.

4. Property-based similarity is the square of the maximum correlation (Eq. 5) of the target molecule to all molecules in the LIBRARY set with the exception of itself. In addition to the property-based similarity, we calculated its prediction error (the error between the estimated and calculated values) for each molecule from the LIBRARY set. For all molecules from the LIBRARY, we averaged errors with identical (using 0.01 precision) property-based similarities. The so-calibrated values were used to estimate the accuracy of prediction for new molecules.

Acknowledgment This study was supported by the Virtual Computational Chemistry Laboratory grant INTAS INFO-00363 and the GoBio BMBF grant 0313883. I thank Philip Wong for his useful comments and remarks.

Abbreviations ALOGPS: Artificial log *P* and log *S* program to predict lipophilcity and aqueous solubility [38, 39]; ASNN: Associative neural network [2–4]; BASF: Chemical company, www.basf.com.; CCNN: Cascade correlation neural network; CPU: Central processing unit; E-state: Electrotopological state indices [43, 44]; GM: Global model; kNN: k nearest neighbors; LIBRARY mode: An operational mode of the ASNN when new compounds are used to correct neural network ensemble predictions without changing neural network weights (see Eq. 6); LM: Local model; ESE: Early stopping over the ensemble [6, 10, 12]; log *D*: The same as log *P* but for ionized compounds (usually measured at a specific pH); log *P*: 1 Octanol/water partition coefficient; log *S*: Aqueous solubility of compounds; MAE: Mean absolute error; NMR: Nuclear magnetic resonance; PHYSPROP: Physical properties database [19]; “nova” set: Set of compounds with log *P* values in the PHYSPROP database that do not have reported experimental values in BioByte StarList (see [38, 39]); “star” set: Set of compounds with log P values in PHYSPROP database that have reported experimental values in BioByte StarList (see [38, 39]); QSAR: Quantitative structure-activity relationship studies; RMSE: Root mean squared error; UCI: University of California, Irvine; VCCLAB: Virtual Computational Chemistry Laboratory, www.vcclab.org [47, 48]

References

1. Fuster JM (1995) Memory in the cerebral cortex. MIT Press, Cambridge, MA,.
2. Tetko V (2001) Associative Neural Network, CogPrints Archive, cog00001441.

3. Tetko IV (2002) Associative neural network. Neural Process. Lett 16:187–199.
4. Tetko IV. (2002) Neural network studies, 4. Introduction to associative neural networks. J Chem Inf Comput Sci 42:717–728.
5. Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal 12:993–1001.
6. Tetko IV, Livingstone DJ, Luik AI (1995) Neural network studies, 1. Comparison of overfitting and overtraining. J Chem Inf Comput Sci 35:826–833.
7. Breiman,L (2001) Random forests. Machine Learning 45:5–32.
8. Tong W, Hong H, Fang H, Xie Q, Perkins R (2003) Decision forest: combining the predictions of multiple independent decision tree models. J Chem Inf Comput Sci 43:525–531.
9. Tetko IV, Villa AEP (1995) In Unsupervised and supervised learning: cooperation toward a common goal, In: ICANN'95, international conference on artificial neural networks NEURONIMES'95, Paris. EC2 & Cie, Paris, France, pp 105–110.
10. Tetko IV, Villa AEP (1997) Efficient partition of learning data sets for neural network training. Neural Networks 10:1361–1374.
11. Tetko IV, Villa AEP (1997) An efficient partition of training data set improves speed and accuracy of cascade-correlation algorithm. Neural Process Lett 6:51–59.
12. Tetko IV, Villa AEP (1997) An enhancement of generalization ability in cascade correlation algorithm by avoidance of overfitting/overtraining problem. Neural Process Lett 6:43–50.
13. Tetko IV, Tanchuk VY (2002) Application of associative neural networks for prediction of lipophilicity in ALOGPS 2.1 program. J Chem Inf Comput Sci 42:1136–1145.
14. Fahlman S, Lebiere C (1990) The cascade-correlation learning architecture. NIPS 2:524–532.
15. Blake EK, Merz C (1998) UCI repository of machine learning databases, available www.ics.uci.edu/~mlearn/MLRepository.html.
16. Schwenk H, Bengio Y (2000) Boosting neural networks. Neural Comput.12:1869–1887.
17. Tetko IV, Tanchuk VY, Villa AE (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.
18. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1994) Numerical recipes in C (2nd edn). Cambridge University Press, New York, p. 998.
19. The Physical Properties Database (PHYSPROP), Syracuse Research Corporation, available www.syrres.com, accessed December 20, 2006.
20. Binev Y, Aires-de-Sousa J (2004) Structure-based predictions of 1H NMR chemical shifts using feed-forward neural networks. J Chem Inf Comput Sci 44:940–945.
21. Binev, Y., Corvo, M, Aires-de-Sousa, J (2004) The impact of available experimental data on the prediction of 1H NMR chemical shifts by neural networks. J. Chem. Inf. Comput. Sci. 44:946–949.
22. Da Costa FB, Binev Y, Gasteiger J, Aires-De-Sousa J (2004) Structure-based predictions of H-1 NMR chemical shifts of sesquiterpene lactones using neural networks. Tetrahedron Letters 45:6931–6935.
23. Dimoglo AS, Shvets NM, Tetko IV, Livingstone DJ (2001) Electronic-topologic investigation of the structure-acetylcholinesterase inhibitor activity relationship in the series of N-benzylpiperidine derivatives. Quant Struct-Activ Rel 20:31–45.
24. Kandemirli F, Shvets N, Kovalishyn V, Dimoglo A (2006) Combined electronic-topological and neural networks study of some hydroxysemicarbazides as potential antitumor agents. J Mol Graph Model 25:33–36.
25. Kandemirli F, Shvets N, Unsalan S, Kucukguzel I, Rollas S, Kovalishyn V, Dimoglo A (2006) The structure-antituberculosis activity relationships study in a series of 5-(4-aminophenyl)-4-substituted-2,4-dihydro-3h-1,2,4-triazole-3-thione derivatives. A combined electronic-topological and neural networks approach. Med Chem 2:415–422.
26. Dimoglo A, Kovalishyn V, Shvets N, Ahsen, V (2005) The structure-inhibitory activity relationships study in a series of cyclooxygenase-2 inhibitors: a combined electronic-topological and neural networks approach. Mini Rev Med Chem 5:879–892.

27. Ajmani S, Tetko IV, Livingstone DJ, Salt D (2005) A comparative study of neural network architectures for QSAR., In: Aki(Sener) E, Yalcin I (eds) QSAR and molecular modelling in rational design of bioactive molecules, Computer Aided Drug Design and Development Society in Turkey, Istanbul, pp. 183–184.
28. Friedel CC, Jahn KH, Sommer S, Rudd S, Mewes HW, Tetko IV (2005) Support vector machines for separation of mixed plant-pathogen EST collections based on codon usage. Bioinformatics 21:1383–1388.
29. Tetko IV, Solov'ev VP, Antonov AV, Yao X, Doucet JP, Fan B, Hoonakker F, Fourches D, Jost P, Lachiche N, Varnek A (2006) Benchmarking of linear and nonlinear approaches for quantitative structure-property relationship studies of metal complexation with ionophores. J Chem Inf Model 46:808–819.
30. Vapnik VN (1998) Statistical leaning theory. Wiley, New York.
31. Tetko IV, Poda GI (2004) Application of ALOGPS 2.1 to predict log D distribution coefficient for Pfizer proprietary compounds. J Med Chem 47:5601–5604.
32. Tetko IV, Bruneau P (2004) Application of ALOGPS to predict 1-octanol/water distribution coefficients, logP, and logD, of AstraZeneca in-house database. J Pharm Sci 93:3103–3110.
33. Tetko IV, Livingstone DJ (2007) Rule-based systems to predict lipophilicity. In: Testa B, van de Waterbeemd H (eds) Comprehensive medicinal chemistry II: in silico tools in ADMET, vol. 5. Elsevier, Oxford, UK, pp 649–668.
34. Poda GI, Tetko IV, Rohrer DC (2005) Towards predictive ADME profiling of drug candidates: lipophilicity and solubility. In: 229th American Chemical Society national meeting and exposition, ACS, San Diego, CA, p. MEDI 514.
35. Balakin KV, Savchuk NP, Tetko IV (2006) In silico approaches to prediction of aqueous and DMSO solubility of drug-like compounds: trends, problems and solutions. Curr Med Chem 13:223–241.
36. Wilson EK (2005) Is safe exchange of data possible? Chem. Eng. News 83:24–29.
37. Tetko IV, Abagyan R, Oprea TI (2005) Surrogate data—a secure way to share corporate data. J Comput Aided Mol Des 19:749–764.
38. Tetko IV, Tanchuk VY (2005) ALOGPS (www.vcclab.org) is a free on-line program to predict lipophilicity and aqueous solubility of chemical compounds. In: 229th American Chemical Society national meeting and exposition, ACS, San Diego, CA pp. U608–U608.
39. Tetko IV (2005) Encoding molecular structures as ranks of models: a new secure way for sharing chemical data and development of ADME/T models. In 229th American Chemical Society national meeting and exposition, San Diego, CA, pp. U602–U602.
40. Tetko IV, Bruneau P, Mewes HW, Rohrer DC, Poda GI (2006) Can we estimate the accuracy of ADMET predictions? In: 232th American Chemical Society national meeting, San Francisco.
41. Tetko IV, Bruneau P, Mewes HW, Rohrer DC, Poda GI (2006) Can we estimate the accuracy of ADME-Tox predictions? Drug Discov Today 11:700–707.
42. Tetko IV (2006) In estimation of applicability domain of a model for toxicity against *T. pyriformis* using ALOGPS logP. Workshop on ranking methods, Verbania, Italy, October 2–3.
43. Kier LB, Hall LH (1990) An electrotopological-state index for atoms in molecules. Pharmaceutical Research 7:801–807.
44. Kier LB, Hall LH (1999) Molecular structure description: the electrotopological state. Academic Press, London, p. 245.
45. Hall LH, Kier LB (1995) Electrotopological state indices for atom types—a novel combination of electronic, topological, and valence state information. J Chem Inf Comput Sci 35:1039–1045.
46. Jain N, Yalkowsky SH (2001) Estimation of the aqueous solubility I: application to organic nonelectrolytes. J Pharm Sci 90:234–252.
47. Tetko IV, Gasteiger J, Todeschini R, Mauri A, Livingstone D, Ertl P, Palyulin VA, Radchenko EV, Zefirov NS, Makarenko AS, Tanchuk VY, Prokopenko VV (2005) Virtual computational chemistry laboratory—design and description. J Comput-Aided Mol Des 19:453–463.
48. Tetko IV (2005) Computing chemistry on the web. Drug Discov Today 10:1497–1500.

49. Tetko IV, Villa AE, Aksenova TI, Zielinski WL, Brower J, Collantes ER, Welsh WJ (1998) Application of a pruning algorithm to optimize artificial neural networks for pharmaceutical fingerprinting. J Chem Inf Comput Sci 38:660–668.
50. Härdle W (1990) Smoothing techniques with implementation in S. Springer-Verlag, New York.

Chapter 11
Neural Networks Predict Protein Structure and Function

Marco Punta and **Burkhard Rost**

Abstract Both supervised and unsupervised neural networks have been applied to the prediction of protein structure and function. Here, we focus on feedforward neural networks and describe how these learning machines can be applied to protein prediction. We discuss how to select an appropriate data set, how to choose and encode protein features into the neural network input, and how to assess the predictor's performance.

Keywords Feedforward neural networks, protein structure, secondary structure, overfitting, performance estimate

1. Introduction

1.1. Scope

Our goal is to introduce the reader to the use of neural networks and, in particular, to the use of feedforward neural networks (NNs) in protein structure and function prediction. NNs are very popular among computational biologists that have to deal with extremely complex phenomena and very noisy data, since NNs can solve classification and regression tasks without needing much prior knowledge of the problem and they are tolerant to errors. Indeed, both supervised (e.g., feedforward and recurrent) and unsupervised (e.g., Kohonen maps) NNs have been applied to a vast number of problems: from detection of secondary structures (SSs; see Przybylski and Rost [1] for a review) to prediction of posttranslational modifications [2–4]; from identification of disordered regions [5] to prediction of metal binding sites [6, 7]; from assignment of subcellular localization [8–10] to separation of proteins into functional classes [11]; as well as many others. Developing an NN-based predictor requires addressing several issues, such as deciding which type of NN to use, choosing the NN architecture, and selecting the cost and activation functions. However, most of these decisions are not specific to the use of NNs for biological problems but rather of more general interest; they are discussed in several textbooks and review papers (see, for example, [12]). Here, we prefer to discuss those aspects of NN development more directly related to the task of predicting protein structural and functional features. In particular, we focus on dataset selection, sample labeling, input feature encoding, and performance assessment. Our analysis concerns methods that use information from the protein sequence only, but most of it holds valid for any learning algorithm that aims to predict protein structural or functional features. In the Introduction, we provide the reader with the (very) essential biological background and NN basics. The Materials section describes protein sequence, structure, and function databases that are widely used in computational biology for developing new predictors. Finally, in the Methods section, we show how to develop an NN that predicts a protein structural or functional feature. As an example, we analyze the case of SS prediction.

1.2. Proteins

1.2.1. What Are Proteins?

Proteins absolve many important functions in living organisms: from catalysis of chemical reactions to transport of nutrients, from recognition to signal transmission. They are polypeptide chains formed by a unique combination of 20 amino acids (aa) [13]. These constituent molecules are characterized by a conserved region (backbone) and a variable region (side chain), where the side chain confers different physicochemical properties to each aa (in terms of, for example, size, charge, hydrophobicity). Proteins are synthesized by living cells starting from genes, most often stored in DNA molecules in the form of nucleic acid sequences (exceptions are, for example, RNA viruses that use RNA as genetic information). DNA is copied into RNA and RNA sequences are translated into proteins with the help of large RNA-protein complexes called *ribosomes*. Most proteins, when exiting the ribosome machinery, fold into a stable three-dimensional structure and are translocated to their cellular compartment, where they absolve their function. Some others, called *natively unstructured*, are unable to fold under physiological conditions and remain disordered. Still, they are implicated in a large number of functional tasks, such as molecular recognition, acetylation, and glycosylation [14] (in a few reported cases, they have been shown to become structured when binding to a substrate, i.e., another protein [15]). Finally, a small number of proteins (e.g., prions) have been observed in more than one stable three-dimensional structure, with different conformations being responsible for different functions (or malfunctions) [16].

1.2.2. Experimental Protein Structure Determination

No matter if unique or not, folded or disordered, the three-dimensional conformation is a key element for determining the capability of a protein to interact with its environment and perform a specific function. This is why, over the last decades, a great effort has been devoted to experimental protein structure determination.

Experimental methods that solve protein structure at atomic resolution include, most prominently, X-ray crystallography and NMR. Both methods have strengths and weaknesses: X-ray crystallography often produces high-quality structures but provides a mostly static representation of the protein, with possible artifacts coming from crystal (i.e., nonnative) contacts [17]; NMR offers a view of the structure of a protein in solution and even a glimpse at its dynamic but lacks a robust resolution measure, thus making difficult to assess the quality of the obtained structures [18]. In other words, where we are able to tell a reliable X-ray structure from a less reliable one, making the same distinction for NMR-derived structures is more difficult. Recently, cryoelectron microscopy has also been used to produce atomic resolution protein structures. Although experimental techniques have continued to improve over the last years and high-throughput production of protein structure is now becoming a reality thank to the structural genomics projects [19], our knowledge of the protein structural universe appears to be still very limited if compared to the number of known protein sequences. The latter have dramatically increased with the advent of genome [20] and environmental [21, 22] sequencing. In face of more than 5 million protein sequences found in TrEMBL (see the Materials section and Table 11.1 later), the Protein Data Bank (PDB) [23], that is, the repository of all publicly available structures, contains about 50,000 entries (statistics from April 2008). Most important, protein classes that have proved tough to treat through experimental methods are considerably underrepresented in the PDB. A classic example is membrane proteins, estimated to constitute from 20 to 30% of all proteins and accounting for less than 1% of the PDB. More, even in cases for which experimental determination of a structure goes smoothly, the average cost per structure is still high [24].

Computational methods can help reduce the gap between known sequences and known structures. NNs, in particular, can be instrumental for the prediction of features such as [1] secondary structure, aa solvent accessibility, aa flexibility, and intrachain aa contacts. These predictions can be used in combination with techniques such as homology modeling (for a recent review see Petrey and Honig [25] and see Jacobson and Sali [26] for the impact of homology modeling on drug design) and fold recognition [27] to produce full three-dimensional structural models or, when these latter methodologies cannot be applied, as a primary source of structural information to assist experimental studies.

1.2.3. Experimental Analysis of Protein Function

Although very important, structural knowledge is often in insufficient for determining the function of a protein. Indeed, similar or same three-dimensional structures can absolve completely different functions and the same function can be performed by different three-dimensional scaffolds [28]. This is due to at least two factors: on the one hand, function and in particular biochemical function (e.g., enzymatic activity) is often carried out at a local level by a few key aa (i.e., small structural differences account for big functional differences); on the other hand, different environmental conditions (such as subcellular localization, temperature, or the tissue the protein is expressed in) can lead to different functions (for a review, see Whisstock and Lesk [29]). Numerous experimental assays exist that predict function directly (that is, not necessarily using structural knowledge) and the information obtained is stored in several public databases. Arguably, the most important of them is Swiss-Prot [30] (a subset of TrEMBL), comprising more than 350,000 manually annotated protein entries (April 2008), compiled from available structural and functional experimental data. Still, Swiss-Prot contains ten times less entries than TrEMBL; also, for several Swiss-Prot entries, we have only a very partial functional knowledge. For this reason, protein function prediction has recently taken center stage in computational biology.

In conclusion, the gap between known sequences and structurally and functionally annotated proteins keeps widening. As a consequence, although experimental techniques for protein structure and function prediction continue to improve, computational methods in these areas are badly needed. NN-based methods can be (and have been widely) used for this purpose.

1.2.4. The Concept of Homology

Homology is a concept central to most of computational biology and a factor to be reckoned with when building a data set for the development of a NN-based method. Proteins, as we know them today (i.e., in contemporary organisms), were not all invented from scratch. According to evolutionary theory, organisms originated from a single common ancestor and diverged gradually over time. During the course of evolution, the ancestral genes propagated through the newly emerging species, thus generating copies that gradually diverged from their relatives by independently accumulating mutations over time). At times, novel genes were "discovered" and also transmitted to later species. As a consequence of this process, most contemporary genes and corresponding proteins have multiple relatives (called *homologs*) across, as well as within (from gene duplication events), species.

Close homology between two proteins can usually be detected from sequence similarity. Given two proteins and an alignment between them (obtained, for example, using PSI-BLAST [31]), if sequence similarity exceeds a certain threshold, we can safely assume that it reflects a common evolutionary origin for the two proteins. In practice, >30% sequence identity on a region longer than 100 aa can be considered a strong indication of homology. An important consequence of the evolutionary relationship between two proteins and the reason why homology is relevant to our discussion on prediction methods is that homologous proteins have a similar structure and, often, similar function. When developing a NN for the prediction of a protein feature, the existence of these correlations is a factor to be taken into account (see Sections 3.3.2 and 3.3.3).

1.3. NN Basics

For the general aspects of NNs, such as learning algorithms and cost functions, we refer the reader to Wu [12]. Here, we introduce a few concepts and definitions crucial for the Methods section. Although a minimal NN has only two layers of nodes (input and output; e.g., perceptrons), the most widely used are slightly more complicated, comprising at least three layers: input, hidden (one or more), and output (Fig. 11.1).

The different layers are connected through junctions in a serial way: input nodes connect to hidden nodes (I-H junctions in Fig. 11.1) and hidden nodes connect to output nodes (H-O junctions in Fig. 11.1). The most common NNs are fully connected, with each node in a layer connected to all nodes in the next layer. The layer constituents (nodes) and the junctions are nothing more than sets of numerical

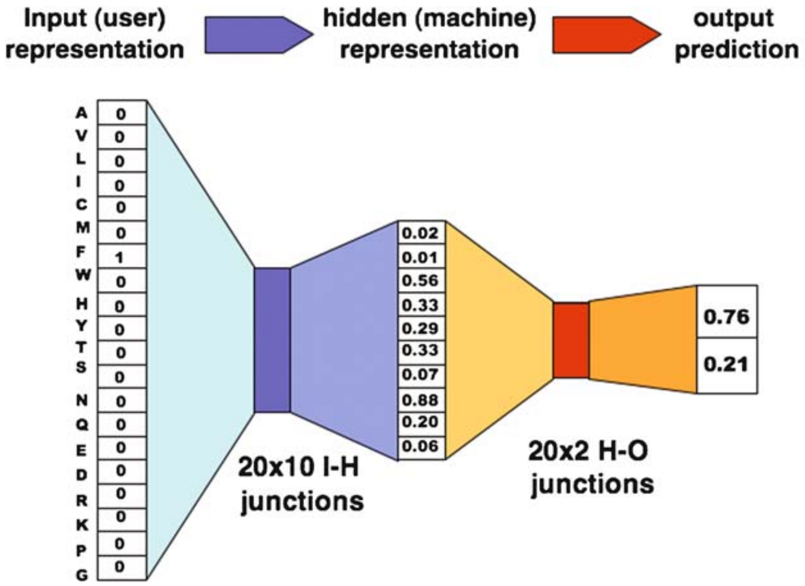


Fig. 11.1 A feedforward NN that predicts an aa class based solely on the aa type, using sparse encoding. In this example, the input consists of a phenylalanine residue (F); accordingly, the input vector is formed by 19 zeros and by a single 1 value corresponding to the F node. The input is translated into a new 10 hidden node representation via 20 × 10 input-hidden (I-H) junctions. Finally, 20 × 2 output-hidden (O-H) junctions predict the hidden representation to belong to one of two output classes (positive, P or negative N)

values. While the layers' nodes describe different representations of the input data (either as given by the programmer, input nodes, or as calculated by the NN, hidden and output nodes), the junctions are the free parameters that have to be optimized during training. In a feedforward NN, the input node values are always provided by the programmer and incorporate the mathematical representation of the input samples (i.e., the examples given to the NN to learn a particular prediction task). The hidden nodes constitute a nonhuman interpretable set of numbers, each generated by the NN as a combination of input node and input-hidden junction values. The hidden layer is, at the same time, the way multilayer NNs learn very complicated classifications and patterns and the reason why NNs are often referred to as *black boxes*. In brief, the NN transforms the programmer choice's input (input node values) into a new representation (hidden node values), which makes the classification of the input samples in the output space easier (where the output space is the one defined by the output nodes). The output layer is formed by values that represent the NN response to the initial input or the predicted classification of the input samples. In a supervised NN, the output values are used by the learning algorithm to determine how the NN free parameters (i.e., the junctions) have to be changed to improve the NN performance. In back propagation, the NN update starts with the hidden-output junctions and ends with the input-hidden junctions (hence it goes "back"-ward). The way this can be achieved is outlined in [12, 32].

2. Materials

2.1. Databases

Countless databases contain information about protein sequence, structure, and function. Here, we focus on a few of the most popular ones (Table 11.1).

2.1.1. Sequences

NCBI GenBank, the EMBL nucleotide sequence database, and the DNA Databank of Japan (DDBJ) are the world's largest repositories of DNA sequences. They are linked and exchange data on a regular basis. They currently (April 2008) contain more than 80 million sequences. Automatic translation (and automatic annotation) for the subset of all coding sequences is found in TrEMBL (~5 million proteins). Swiss-Prot [30] is a manually annotated database containing information about protein function, domain structure, posttranslational modifications, and the like, comprising ~350,000 TrEMBL sequences. Annotations are generally derived from the literature. TrEMBL and Swiss-Prot have recently been incorporated into the UniProt [33]. Several other databases contain information about protein families, domains, and functional sites; several of them are integrated into InterPro [34].

Table 11.1 Some popular databases used by computational biologists for developing their prediction methods

Database/method	URL
GenBank	www.ncbi.nlm.nih.gov/Genbank
EMBL nuc.sq.dtb.	www.ebi.ac.uk/embl
DDBJ	www.ddbj.nig.ac.jp
Swiss-Prot&TrEMBL [30]	ca.expasy.org/sprot
UniProt [33]	www.ebi.uniprot.org/index.shtml and mirror sites
UniRef	ftp://ftp.expasy.org/databases/uniprot/uniref
PDB [23]	www.rcsb.org/pdb/Welcome.do
PDB non-red. Sets	www.rcsb.org/pdb/clusterStatistics.do
CD-HIT [38]	http://bioinformatics.ljcrf.edu/cd-hi
UniqueProt [44]	http://cubic.bioc.columbia.edu/services/uniqueprot
EVA [42] unique set	http://cubic.bioc.columbia.edu/eva/res/weeks.html
BLAST/PSI-BLAST [31]	www.ncbi.nlm.nih.gov/BLAST
BLASTClust	www.ncbi.nlm.nih.gov/Web/Newsltr/Spring04/blastlab.html
SCOP [35]	http://scop.mrc-lmb.cam.ac.uk/scop
CATH [36]	http://cathwww.biochem.ucl.ac.uk/latest/index.html
GO [37]	www.geneontology.org/index.shtml

2.1.2. Structures

The protein data bank [23] is the primary source of information for protein (plus DNA and RNA) three-dimensional structure. It contains all publicly available protein structures obtained by X-ray, NMR, or electron microscopy techniques (~50,000, of which ~18,000 are unique at 95% sequence identity). Protein structures have been the object of several classification attempts. The most used schemes are SCOP [35] and CATH [36], which classify PDB proteins in a hierarchical fashion, starting with overall secondary structure composition (only alpha-helices, only beta strands, mixed alpha-beta) and ending with closely related homologous proteins.

2.1.3. Functions

As said in the Introduction, the proteins part of Swiss-Prot [30] are annotated for known functional and structural features. The Gene Ontology (GO [37]) project describes the function of gene products according to three predefined classification schemes, which encompass biological processes, cellular components, and molecular functions.

2.1.4. Nonredundant and Sequence-Unique Databases

Redundancy in sequence databases is a very common phenomenon. Redundancy reduction can aim at two distinct goals: removing duplicate information (i.e., very similar sequences) thus effectively reducing the number of sequences one has to deal with or creating a sequence-unique database for developing a *de novo* prediction method (i.e., a method that predicts a protein feature when no homologous template is available). A number of nonredundant databases are publicly available, as well as programs that can be used to create task-tailored unique data sets. The NCBI nr nucleotide and protein databases reduce redundancy at a very basic level by merging into a single entry proteins with the exact same sequence. UniProt maintains UniRef90 and UniRef50, two databases built using the CD-HIT algorithm [38] and clustering all sequences in UniProt at 90% and 50% sequence identity, respectively (e.g., no two sequences in UniRef90 share sequence identity $>90\%$). The nrdb90 [39] takes the union of several well-known databases (such as Swiss-Prot [30], TrEMBL, GenBank, and PDB [23]) and reduces redundancy at 90% identity.

Sequence-unique sets differ from generic nonredundant sets, in that they are meant to maximally reduce the presence of homologous proteins (proteins that may have correlated structural and functional features). In practice, no approach can guarantee that two proteins in a data set are not homologous, but several available methods manage to successfully remove the most trivial relationships in a set. It has also to be noted that criteria to define *uniqueness* can vary according to the feature under consideration. Different thresholds or different measures may be needed to define a sequence-unique set with respect to, say, structure [40] or subcellular localization [41]. As far as structure is concerned, the PDB periodically updates several "unique" sets (at different levels of sequence identity), created by using CD-HIT [38] or BLASTClust. The EVA server [42] maintains a continuously updated set of sequence-unique PDB chains (no pair of proteins in this set has HSSP-value [40, 43] above 0; note that, in this latter case, "uniqueness" has been specifically defined as related to structural similarity). Programs that can be used to create personalized sequence-unique data sets are, for example, CD-HIT [38] and UniqueProt [44].

3. Methods

The aim of this section is to show how NNs can be used to predict protein structural and functional features (Fig. 11.2). As an example, we take the prediction of secondary structure (SS) elements in water-soluble globular proteins. We describe how relevant data can be extracted from the available databases, discuss the steps that have to be taken to avoid overfitting, review encoding schemes for the most used protein sequence features, and, finally, analyze ways to estimate the NN performance. Most of the issues we discuss are not specific to the prediction task under consideration (i.e., SS) and their understanding can help in developing NNs that attempt to predict any protein structural or functional feature.

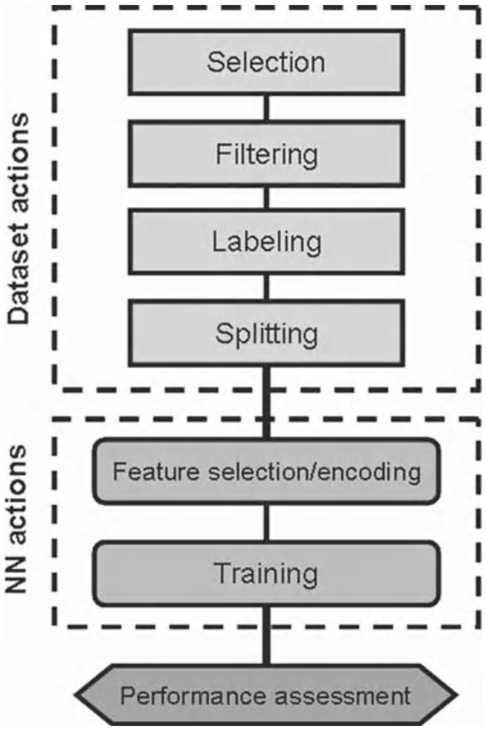


Fig. 11.2 Developing an NN for predicting protein features. First, we have to create a data set by selecting data from available databases and filtering out those occurrences that cannot be reliably labeled for the features we want to predict. After the remaining data have been labeled, we can proceed in splitting the data set into several folds (three minimum) to be able to train and assess the performance of the NN. Once the data sets are defined, we have to decide which sequence (or structure) features are relevant for the classification (or regression) problem under study and how to encode them in the NN input (see Fig. 11.4). Next, we can train our NN and finally assess its performance, taking advantage of the data set splitting (see Fig. 11.5)

3.1. What Is Secondary Structure?

Under physiological conditions, most proteins are folded into a stable three-dimensional structure. Although whole three-dimensional protein structures are considerably diverse, most of them can be seen as combinations of only a few recurrent local three-dimensional motifs, usually referred to as SS. The most common among them are alpha-helices and beta-strands (Fig. 11.3).

These structural motifs can be described by different properties. For example, helices and strands have characteristics backbone dihedral angle conformations (clearly visible in the so-called Ramachandran plot [45]). Energetically, alpha-helices are stabilized by backbone hydrogen bonds between amino acids separated by three or four positions along the chain (Fig. 11.3). In contrast, beta-strands are stabilized by interactions with other strands and form super-SSs known as beta sheets (Fig. 11.3); this allows them to bridge amino acids that may be very distant along the protein sequence. Predicting alpha helices and beta strands is attractive since they are ubiquitous, and most three-dimensional structures can be described as assemblies of these elements. Indeed, in the past, accurate predictions of helices and strands have led to important advances in database sequence search, fold recognition, and de novo three-dimensional structure prediction (see, for example, [1, 25, 46]).

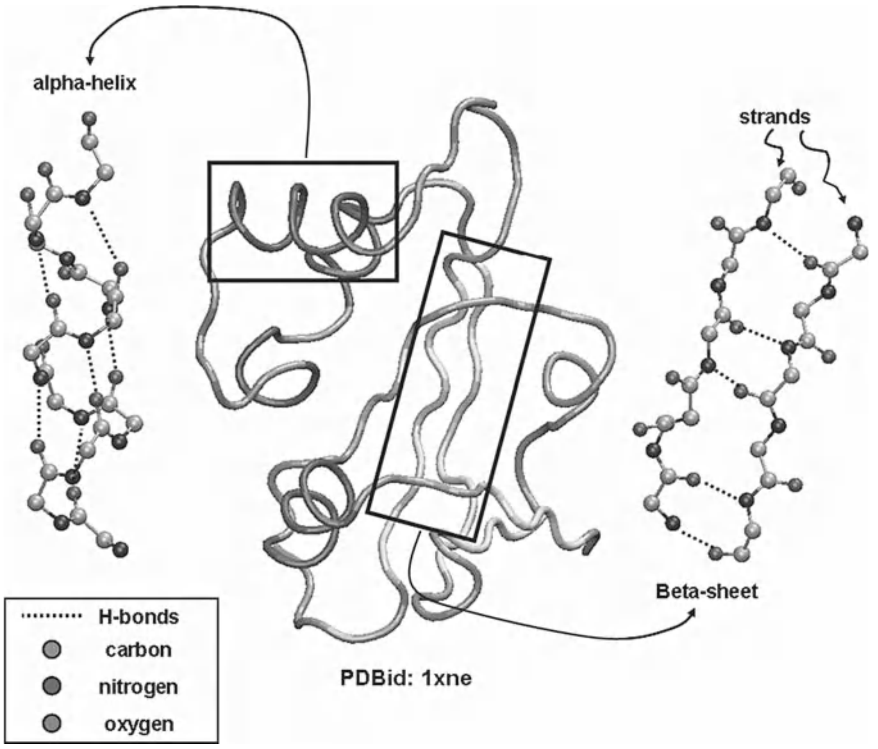


Fig. 11.3 Center (tube representation of the protein backbone): We highlight alpha-helices and beta-strands on the backbone structure of a protein of unknown function (PDBid:1xne). Left (ball and stick representation of the backbone atoms only, balls = atoms and sticks = covalent bonds; omitting hydrogen atoms): Alpha helices are stabilized by hydrogen bonds (H-bonds) between the amide and carbonyl groups of the protein backbone. Right (same representation as in left): beta strands are stabilized by interstrand H-bonds between the amide and carbonyl groups of the protein backbone

3.2. The Prediction Task

Although recently there have been attempts to predict SS in water-soluble globular proteins by using an increasing number of classes (helix, strand, turn, bend, etc.) [47, 48], most available methods predict SS on a three-class basis (helix, strand, other). Here, for the sake of simplicity, we adopt an even simpler two-state classification. So, according to our definition, a residue can be in either of two states: SS (helix or strand) or non-SS (other).

3.3. Data Set

3.3.1. Selecting Protein Sequences from the PDB

For developing an NN that predicts SS in water-soluble globular proteins, we need a database with reliable annotations for helices and strands. In general, when predicting structural features, the choice falls on the PDB [23]; that is, unless the specific feature we are interested in turns out to be underrepresented in the PDB to such an extent that more data are needed. If this is the case (and it is not for SS of water-soluble globular proteins, as we will see), we may want to also rely on lower resolution data (although this implies some risks, as shown for the use of low-resolution data in transmembrane helix predictions [49]). As previously said, the PDB contains almost 50,000 protein structures however, not all structures are good for inclusion in our data set. First, we want to remove all proteins that are not water soluble and globular, such as membrane and coiled-coil proteins (<1% of the PDB). Second, we need to make a decision about the resolution of the structures part of our data set. In particular, given that SS is best assigned based on atomic-level structural information, we may want to consider only structures having high resolution. This would, however, automatically exclude all NMR structures, constituting about one seventh of the PDB or ~5,000 structures. In fact, as mentioned in the Introduction, these structures have no well-defined associated resolution value. Still, for some NMR structures that may have unreliable atomic coordinates, many others are likely to produce reliable SS annotation. So, including or not including NMR structures depends critically on the size of the original, unfiltered database. In general, considering only PDB structures with resolution lower than 3.0 Å (note that, for the way resolution is measured, low-values indicate high resolution) would return ~40,000 structures (i.e., almost all X-ray structures in the PDB); a slightly more conservative threshold, 2.5 Å, would still provide ~34,000 proteins (for up-to-date PDB statistics go to www.rcsb.org/pdb/static.do?p=general_information/ pdb_statistics/index.html). In the case of SS, each of these proteins would provide from tens to thousands of samples (since one residue = one sample), with a good balance between positives and negatives (with about 50% of all residues being found in helices or strands

[1]). Although the upcoming sections show that these numbers do not correspond to the actual number of proteins that can be used for training (due to the need of splitting the data set into several folds and the need to reduce redundancy or create a sequence-unique set), we will see that the PDB is large enough for developing an SS predictor that uses hundreds of nodes for describing the NN input (Section 3.6).

3.3.2. Avoiding Overfitting

NN, like other learning algorithms, may suffer from under- or overfitting. Both phenomena cause the NN to underperform on new, previously unseen data (i.e., data not contained in the training set) and both are closely related to the number of free parameters (NFP) contained in the NN. If the NFP is too low, the NN may be unable to efficiently classify the training data (underfitting); on the other hand, if the ratio between NFP and the number of training samples is high, the NN ends up fitting the training data so well as to eventually hamper its capability of generalizing to new ones (overfitting). Underfitting can be prevented by steadily increasing the NFP (typically, by increasing the number of hidden nodes) until improvement of the NN performance becomes insignificant. Avoiding overfitting is not as easy; indeed, just looking at the performance of the NN on the training set will not provide us the information we need on the diminishing generalization power of the NN. In general, having a low ratio between the NFP and the number of training samples ($<1/10$) helps but cannot guarantee that there is absolutely no overfitting. In computational biology, the most popular way to tackle this problem is to use "stop training" (other techniques exist, such as weight decay and Bayesian learning). Stop training consists of splitting the data set into two folds, training on the first and using the second (cross-training set, from now on) to check if and when the NN enters an overfitting regime. Training is stopped when it becomes clear that the performance on the cross-training set starts deteriorating. It is important to stress that, even when performing stop training, the ratio between NFP and the number of training samples should not be $\geqslant 1/10$. In fact, in this case, due to the considerably sparse sampling of the input space, new (cross-training) samples often fall far from the "known" training samples, and hence it will be intrinsically difficult for the NN to predict them correctly (i.e., to generalize to new data). In conclusion, to obtain a close-to-optimal performance of the NN we need both to keep under control the ratio between NFP and training samples and to perform stop training. Finally, note that, for a correct evaluation of the NN performance, the two sets introduced here are not sufficient. As we will see in Section 3.7, performance evaluation calls for a minimum threefold split of the original data set.

Two other important issues concern the relative size of the training and cross-training sets and their protein composition. Concerning the size, we have to satisfy two conflicting constraints. On the one hand, we would like to train the NN on as many samples as possible. On the other, the cross-training set needs to be large enough to well represent our sample population. In practice, the cross-training set is usually taken as a fraction of the training set (1/3, 1/5, 1/10, etc., see also Section 3.7). More complicated is the issue of which proteins to assign to each of the two data sets (data set composition). Since the cross-training set has to simulate a set of data the NN has not seen before, a first obvious requirement is that it shares no sequence with the training set. But is this enough? As discussed in the Introduction, proteins are grouped into families (according to their evolutionary origin) and proteins in the same family (homologs) share many structural and functional features. So, the question is whether it is a good idea to perform stop training on a set that contains proteins homologous to some of the training sequences. The answer really depends on what we are trying to predict. In general, we have to keep in mind that the composition of the cross-training set relative to the training set should mirror as closely as possible the composition of new unannotated data relative to the training set. As a consequence, if we believe that the range of applicability of our method will extend to the prediction of proteins with a known annotated homolog (i.e., a homolog for which the feature we intend to predict in the target protein is known), it is correct to include in the cross-training set proteins homologous to the training sequences. For example, it has been shown that subcellular localization is poorly conserved in homologous proteins [41]. Hence, predicting such a feature is of value even when a homolog of known localization is available. On the other hand, many structural features (such as SS) are found to be significantly correlated in homologous proteins, with the structure of the homolog often providing the best guess for the target's one (although predictions based on machine learning approaches can still be useful in regions in which the similarity between the two homologs is low). In practice, most NN methods in computational biology are optimized for producing de novo predictions, or predictions on proteins with no known annotated homolog. This is a realistic case; in fact, 20–30% of all open reading frames in every newly full-sequenced genome constitutes such proteins [50, 51]. Now, if our goal is to predict protein features de novo, we have to build the cross-training set so that none of its proteins has a close homolog in the training set. The way this is achieved is described in the following section.

3.3.3. Sequence-Unique Sets

Reduction of homology redundancy in a data set is achieved by creating sequence-unique sets. The term *sequence uniqu*e refers to the property of these sets of having a single representative (unique) for each protein sequence family or group of homologous proteins defined through sequence similarity. In other words, if the cross-training set is unique with respect to the training set, then none of its proteins has a close relative in the training set (and vice versa). So, using such a set should provide a good estimate for the performance of the method in a situation in which no homolog of the protein being predicted is available. This estimate eventually constitutes a lower bound for the performance in cases in which the target protein has detectable homology to a protein in the training set. So much for the presence of interset homologs. How instead should intraset homology be treated? Training with more than one representative for each protein family is somehow similar to jittering. Jittering consists in increasing the number of training samples by creating artificial samples that are small variations of the real ones. This is exactly the effect of adding homologs to the training set, with the advantage that homologs are real data and, thus, the class they belong to is generally known (i.e., they do not represent noise). Jittering samples instead, being virtual data, are often noisy with respect to class assignment. In conclusion, using groups of homologous proteins in training may be advisable. However, it is important to realize that some protein families are larger than others and that, additionally, protein databases are biased toward particular families. As a consequence, just adding all members of the families represented in the training set can lead to a biased predictor (i.e., biased toward the families with more members). One way to solve this problem is to balance the presence of homologs among the different families, by limiting the maximal number of members allowed for each family. Similar considerations hold and are even more relevant when taking into consideration proteins that are part of the cross-training set. Since it is difficult to determine whether and how the bias observed in present databases reflects an actual bias in the protein space, including several family representatives in the cross-training set may lead to erroneous estimates of the method's performance. Since there is no clear advantage in adding homologs to the cross-training set, our suggestion is not adding them at all.

In the Materials section, we discussed where to find ready-to-use sequence-unique sets and programs that can be used to create ad hoc sequence-unique sets (Section 2.1.4).

Let us now go back to the problem of predicting SS. As said previously, by taking into consideration all PDB structures having resolution lower than 3 Å, we retrieve about 40,000 structures. If we now make this data set sequence unique (see Section 2.1.4), we are left with about 3,500 proteins. This is the set we have to use to generate the training and cross-training data.

3.4. Data Labeling

Another fundamental step in the process of building a prediction method is labeling the samples. Problems related to sample labeling may be of a different nature. On the one side, the databases computational biologists rely on are not devoid of errors (i.e., misannotations) [52, 53]. This problem can be mitigated by picking the most reliable annotations within a database, when this kind of information is available. For example, Swiss-Prot includes both experimentally annotated proteins (more reliable) and proteins annotated by computational means (less reliable).

The next step is to decide what kind of classification we want to adopt. In general, this choice is a trade-off between the desire to have an accurate feature definition (many classes) and the need to retrieve enough samples for each feature class. Here, as said previously, we opt for a two-class labeling of the data (SS, non-SS). How are we going to assign our samples to one class or the other? For SS, assignment has to be done at the residue level (in other cases, e.g., when predicting function or subcellular localization, the assignment is done on a per-protein basis). SS can be defined in many different ways, and as a consequence, different criteria may provide different data labeling results. The most popular automatic methods for SS assignments are DSSP [54] and STRIDE [55] (for other methods, see [1]). DSSP assigns a residue to one of eight classes based on the evaluation of backbone atoms interaction energies; the classes are H = alpha helix, B = residue in an isolated beta bridge, E = extended strand, G = 3-helix (3/10 helix), I = 5-helix (pi helix), T = hydrogen bonded turn, S = bend, and *other*. STRIDE utilizes backbone angles and empirically derived energies for the hydrogen bonds. If we were to use, say, DSSP, we would reduce the eight classes to two: SS (H and E) and non-SS (all the others); note that classes G (3/10 helices), I (pi helices), and B (beta-bridges) could also be considered as part of the SS class. By running DSSP on all proteins of our data set, we can attach an SS or non-SS label to each of the residues.

3.5. Encoding a Protein Sequence into the NN Input

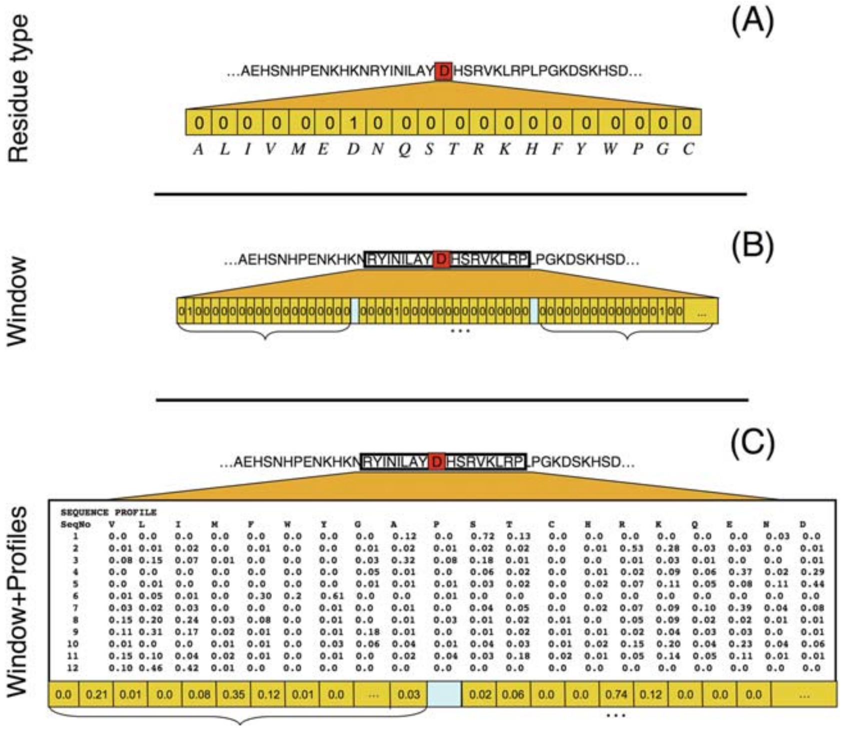
The primary sequence of a protein of length L can be seen as a series of L letters extracted from the natural aa 20-character alphabet. In the following paragraphs, we want to discuss how information from the protein sequence can be encoded into the input of an NN to predict some structural or functional feature (here, SS). The first thing we need to decide is what kind of primary sequence information we want to use; the second is how we are going to feed it into the NN input. It is difficult to overemphasize the importance of these two steps; indeed, much of the success of the method depends on these decisions.

One crucial aspect of encoding is that similar vectors must represent similar input samples (according to any a priori idea of similarity that we may have for the samples), while dissimilar vectors must stand for dissimilar samples. This may appear as trivial advice; however, it is not that difficult to introduce unwanted vector similarities (or correlations) between unrelated input samples (see, for example, C- and N-terminal aa encoding in Section 3.5.2).

3.5.1. Amino Acid Type

What determines the propensity of an aa to be in a SS conformation? The most trivial answer is the residue side chain (i.e., the aa type). Indeed, the conformational space accessible to the backbone of each aa depends critically on its side chain, with different aa having different constraints. Since one way to define an SS is by specifying the backbone conformation, it is reasonable to think that different aa types correspond to different SS propensities. This is confirmed by a simple analysis of aa relative frequencies in SS elements [56].

There are several ways in which the aa type can be translated into a numeric entry for an NN. One of the most used in computational biology is the so-called sparse or one-hot encoding. It consists in storing the aa type for a single sequence position into a 20-element vector, whose elements represent the 20 possible aa side chains. The vector is ordered, so that each aa is assigned a given vector element. For example, to enter into the NN an aspartate for a given sequence position (aa one letter-code D, Fig. 11.4A), we need to build the following vector:



SEQUENCE PROFILE

SeqNo	V	L	I	M	F	W	Y	G	A	P	S	T	C	H	R	K	Q	E	N	D
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.12	0.0	0.72	0.13	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.0
2	0.01	0.01	0.02	0.0	0.01	0.0	0.0	0.01	0.02	0.01	0.02	0.02	0.0	0.01	0.53	0.28	0.03	0.03	0.0	0.01
3	0.08	0.15	0.07	0.01	0.0	0.0	0.0	0.03	0.32	0.08	0.18	0.01	0.0	0.0	0.01	0.03	0.01	0.0	0.0	0.01
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.01	0.0	0.06	0.02	0.0	0.01	0.02	0.09	0.06	0.37	0.02	0.29
5	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.01	0.01	0.01	0.03	0.02	0.0	0.02	0.07	0.11	0.05	0.08	0.11	0.44
6	0.01	0.05	0.01	0.0	0.30	0.2	0.61	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.03	0.02	0.03	0.0	0.0	0.0	0.01	0.0	0.01	0.0	0.04	0.05	0.0	0.02	0.07	0.09	0.10	0.39	0.04	0.08
8	0.15	0.20	0.24	0.03	0.08	0.0	0.01	0.0	0.01	0.03	0.01	0.02	0.01	0.0	0.05	0.09	0.02	0.02	0.01	0.01
9	0.11	0.31	0.17	0.02	0.01	0.0	0.01	0.18	0.01	0.0	0.01	0.02	0.01	0.01	0.02	0.04	0.03	0.03	0.0	0.01
10	0.01	0.0	0.0	0.01	0.01	0.0	0.03	0.06	0.04	0.01	0.04	0.03	0.01	0.02	0.15	0.20	0.04	0.23	0.04	0.06
11	0.15	0.10	0.04	0.02	0.01	0.0	0.01	0.0	0.02	0.04	0.03	0.18	0.02	0.01	0.05	0.14	0.05	0.11	0.01	0.01
12	0.10	0.46	0.42	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 11. 4 Encoding information from the protein sequence into an NN. (**A**) Sparse encoding for the aa type: We use 20 nodes, 1 for each of the 20 natural aa. Each residue is assigned a specific position in the vector that is fixed and does not change from sample to sample. For an aspartate D, the input is a vector with 19 zeros and a single one value corresponding to the D element (or node). (**B**) Sparse encoding for the aa type of neighboring residues. In this case, if we consider a window of size *N* around the central D residue, we have a vector with 20 × *N* elements (or *N* vectors with 20 elements each). Note that the light shaded separators in the figure do not represent nodes; they are separators that we added to show that the final input vector is obtained by joining the *N* vectors that code for each single position within the window. (**C**) Encoding evolutionary profiles for neighboring residues using frequency of occurrence in multiple sequence alignments. The input node values are no longer zeros and ones but the frequencies themselves

0, 0,…, 1,…, 0, where 1 corresponds to the D node and 0 to the nodes coding for the other aa. Note that the vector looks the same no matter what is the position of D along the sequence. Also, this particular choice of values, 1 and 0, has no special meaning; we could as well assign 0 to the D node and 1 to the others or use 100 instead of 1. The only important thing is that the notation be consistent throughout all the samples. Sparse encoding has proven to be efficient and is the most widely used scheme for feeding an NN with aa type information. In particular, it allows a straightforward transition from aa type to evolutionary profile encoding, as discussed in the next paragraph. One problem, though, is that it is very expensive in terms of the number of input nodes used. For entering the aa type at a single sequence position, we have to use 20 nodes; and if we want to use information from neighboring residues as well (see Section 3.5.2), we easily need hundreds of nodes. This is a problem, since it significantly increases the NFP of the NN and hence the risk of overfitting. A less-expensive type of encoding is obtained by using a truly binary representation for the aa alphabet. Under this scheme, by allowing more than one node to be equal to 1 at the same time (acceptable inputs are, e.g., 1, 0, 0, 0, 0; 0, 1, 0, 0, 0; 1, 1, 0, 0, 0; 1, 0, 1, 0, 0), all we need to describe the 20 aa are five nodes. However, this type of representation may be less efficient than sparse encoding (see, for nucleotide encoding, [57]). A different approach consists in redefining the aa alphabet. Indeed, aa can be grouped into classes of "similarity" according to their physicochemical properties. A classification comprising four classes may include, for example, hydrophobic, aromatic, polar, and charged. By using this new alphabet (or a similar one), we would need only four input nodes for each sequence position, thus trading the loss of part of the information contained in the 20-letter alphabet for a smaller NFP. This scheme can also be easily extended for the use with evolutionary profiles. Finding the optimal aa alphabet (i.e., the most efficient in terms of information content) is a challenge that has been addressed by several groups (see, for example, [58–60]).

3.5.2. Windows: Using Information from the aa Neighborhood

The probability that a certain aa adopts an SS conformation depends not only on its individual propensity for a particular SS but also on that of its neighbors. Indeed, the conformational space that is accessible to the backbone of a dipeptide or a tripeptide depends on the interactions between the constituent aa. Another argument in favor of using information from neighboring aa is that SS are not formed by isolated aa. Indeed, helices and strands are constituted of a minimum of three or four and can actually span tens of aa (with helices longer on average than strands). Thus, we assume that, for an aa to be in SS, it is necessary that at least some of its neighbors, preceding or following, have a good SS propensity; this propensity should be reflected by the aa composition at those positions. We can enter this information into the NN by simply extending the notation devised for the single position input. If we are using sparse encoding, we create an ordered vector with $20 \times (2\,w + 1)$ elements (with w, the window half-length, which extends left and right to the aa whose SS we want to predict, Fig. 11.4B). In other words, each residue in the window takes up 20 input nodes. When using windows, there is one additional detail to take into account: *N*- and *C*-terminal residues. In fact, suppose we use a three-residue window (w = 1) and the residue we are trying to predict (let us call it residue *i*) is the first *N*-terminal residue; what values are we going to enter into the 20 nodes meant to contain information about the virtual position *i* – 1 (virtual because it extends beyond the actual protein sequence)? One possible choice would be to leave the 20 nodes empty (i.e., all zero); however, in so doing we would be using the nodes that are meant to represent the residue type, for a completely different task (i.e., to tell the NN that the central residue in the window is *N*- or *C*-terminal). As we said, it is always better not to introduce unwanted correlations between the input features if this can be avoided with little harm. A safer, if slightly more parameter-expensive procedure, is to add one node per position, which takes the value 0 when the encoded position corresponds to an actual residue, and 1 when it corresponds to a virtual one. In practice, this is the most common way *N*- and *C*-terminal residues are encoded in NN predictors.

3.5.3. Using Evolutionary Information

As mentioned in the Materials section (Section 1.2.4), given a protein, we can generally find a certain number of sequences that are homologous to it. Now, given a multiple alignment of sequences belonging to the same protein family, we find that different positions along the alignment have different degrees of aa conservation among the aligned proteins. Since homologous proteins often share similar structural and functional features, knowledge of the particular aa mixture allowed at a specific position can provide important information for many prediction tasks. In fact, the residue we see at a given position in a protein sequence can be regarded as one of the possible occurrences that are compatible with the conservation of the protein structure and function in that family. As a consequence, it is generally a big advantage to be able to include evolutionary information in the input of an NN. Historically, the use of this information in the NN was first introduced for predicting SS, improving performance considerably [61]. Since then, it has been applied to a large number of prediction tasks, often proving to be a fundamental ingredient for obtaining better performance.

To use evolutionary information in an NN, it is first necessary to produce a list of homologs for the protein being predicted. The most popular program that performs this task, by searching available sequence databases (such as Swiss-Prot [30] or TrEMBL), is PSI-BLAST [31]. In addition to providing a list of homologues (if they exist) PSI-BLAST also outputs the pairwise sequence alignments, a position-specific scoring matrix that represents the aa substitution scores at each position in the protein family, and the frequency of occurrence of the 20 aa at each given position in the alignment. These data can be used to enter information from the protein family into an NN. The first and perhaps most straightforward approach consists in directly entering into the NN the values of the PSI-BLAST position-specific substitution matrix.

When using sparse encoding, this translates into feeding each of the 20 nodes representing the aa type at a given position with the values found in the column of the substitution matrix that corresponds to that position (i.e., the PSI-BLAST calculated scores for the aa in the original sequence to mutate into one of the other 19 aa or be conserved). Another very popular strategy consists in using the frequency of occurrence of the aa at each sequence position. The frequencies (normalized to range, e.g., between 0 and 1) feed a 20-element ordered vector (see Fig. 11.4C).

It is important to realize that the significance of quantities such as the calculated substitution scores or the aa frequencies in a family depends on the evolutionary spread of the proteins and the number of sequences found in the family. To address the first issue, PSI-BLAST reports weighted frequencies, balancing the profile by reducing the weight of very similar sequences. The second problem (dependency of profile significance on the number of aligned sequences) can be addressed by entering into the NN the explicit number of aligned sequences (e.g., by using a sparse encoding binning scheme). In this way, we ask the NN to learn how to assign a different weight to conserved (or variable) positions in profiles generated by different number of sequences. In general, when developing a method that relies on evolutionary information, it is always important to report performance estimate as a function of the amount of evolutionary information available (e.g., as a function of the number of aligned sequences).

3.6. Final Decisions on Input Features, NN Architecture, and Training

3.6.1. Building the NN

At this point, we have all the elements we need to build a first simple NN for predicting protein SS. Given a position in a protein sequence, we can define, for example, a window of length 19 ($w = 9$) and, for each position in the window, extract from the PSI-BLAST output the aa occurrence frequencies; finally, we can assign each of these values to an NN input node (overall, we need $19 \times 20 = 380$ nodes). Additionally, we want to use one node for each position in the window to account for *N*- and *C*-terminal aa. Hence, overall, the input nodes amount to $18 \times 21 + 20 = 398$ (note that the central aa does not need the additional node since it cannot be virtual). How do we choose the number of hidden nodes (NHN)? The overall NFP in our NN is given by 398 times the NHN, plus the NHN times the number of output nodes. Say, we choose to have only one output node (0 = non-SS, 1 = SS). Since our data set includes about 3,500 proteins, most of them providing hundreds of samples, we expect the overall number of training samples to be on the order of 10^6. As mentioned before, to reduce the risk of overfitting, the ratio between the NFP and the number of training samples should be lower than ~1/10 (even when performing stop training); hence,

$$(398 \times \mathrm{NHN} + \mathrm{NHN} \times 1)/10^6 < 1/10$$

hence, approximately,

$$NHN < 10^6/4 \times 10^3 \text{ or } NHN < 250$$

So, in principle, any number between 2 and 250 may be used.

3.6.2. Balanced Training

Often, in a classification task, the different classes are distributed unevenly in the training sets, and as consequence, the less populated classes (fewer samples) generally suffer from poorer performance. One way to counter the data set imbalance is to perform "balanced training" [61]. Balanced training consists of presenting the NN with the same number of samples from each of the prediction classes at each epoch. This implies either a resampling of the less-populated classes or discarding samples from the more-populated ones (or both). For example, in the classic three-class SS prediction problem (helix, strand, and other), strands are generally under-represented in the training set. It has been observed that using balanced training can help improve the performance on strands [61].

We now move on to discuss performance estimate for an NN. As we will see, the way performance is estimated also affects the way we have to split our data set.

3.7. Performance Estimate

3.7.1. Hold Out

In the section on how to avoid overfitting (Section 3.3.2), we discussed the notion of stop training and the necessity to introduce a cross-training set. In this section, we want to address the issue of performance evaluation and the need to introduce one more data set: the validation or test set. It is essential to realize that the performance of an NN-based prediction method can be correctly evaluated on an unseen data set only if the NN parameters are frozen. In other words, the choice of all NN parameters has to be done before any performance assessment is performed. Having said this, it is clear that two sets (training and cross-training) do not suffice. In fact, we use training for producing different models (or sets of NN parameters) and cross-training for choosing what we believe is the optimal model (Fig. 11.5). In other words, cross-training is an integral part of the parameter optimization process; hence, the necessity to introduce a third set, which is meant to be left untouched ("unseen") until all parameters have been fixed. This set is subject to the same restrictions discussed for the cross-training set. First, it must not contain exact copies of proteins found either in the training or in the cross-training set. Second, if what we wish to evaluate is the method's performance on *de novo* predictions, it should be sequence unique with respect to the previous two sets. Third, it must be representative of the sample population. It is interesting to

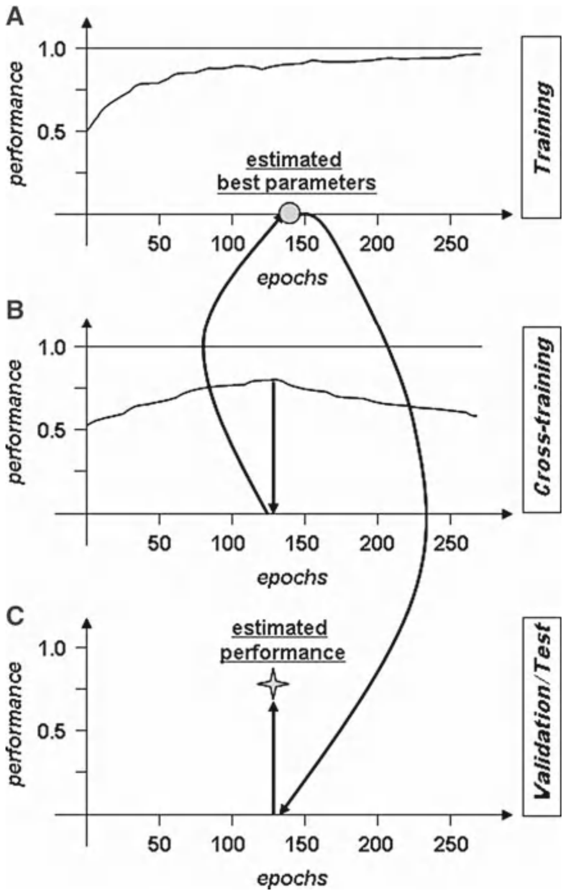


Fig. 11.5 NN training, cross-training and validation/test sets. **A** Training is performed to saturation; **B** cross-training on a set of "unseen" data (see text) is used to estimate the epoch (i.e., the training parameters), granting the best generalization (least overfitting); **C** validation/test is performed on a third set of data, seen during neither training nor cross-training. Arrows are used to describe the protocol: from B we identify the best generalizing epoch; then, we pick from A the corresponding parameters (junction values); finally, we run the NN with those parameters once on the validation/test set. The performance value obtained on this last set is our estimate of the NN performance

note that mistakes (nonuniqueness, composition bias) in defining training, cross-training, and validation sets have quite different consequences. In fact, mistakes in selecting the training sets can cause the method to be suboptimal and hence to underperform; mistakes in selecting the validation set most often cause an overestimate of the method's performance.

In conclusion, our original data set has to be split into three folds (Fig. 11.5): training, cross-training, and validation sets. The first is used to learn the classification problem; the second to decide when the training process should be stopped (this, as previously said, has to be considered part of the training process); the third to test the method's performance. This methodology is usually referred to as *hold -out* (meaning that one set is held out from the training process and saved for validation purposes).

3.7.2. Cross-Validation

Another way, similar to hold out, to fairly evaluate an NN is to use cross-validation. In N-fold cross-validation, the original data set is split into N subsets. $N - 2$ subsets are used for training, one each for cross-training and validation. In this case, the NN is trained N times (with N typically equals 5 or 10) until all subsets have been used at least once as a training, cross-training, and validation set. The estimate of the NN performance is taken as the average performance over the N validation sets. The advantage of cross-validation over hold out is that it allows using most of the available samples for training. On the other hand, it is time-consuming (training has to be performed N times). For these reasons, cross-validation is used mostly when the number of available samples is relatively low.

3.7.3. Measures

Several measures have been devised to assess the performance of NN. For classification problems, it is useful to first define a few basic quantities: true positives (TP) or correctly predicted positive samples (in our case, TP are, e.g., SS samples); true negatives (TN) or correctly predicted negative samples (non-SS samples); false positives (FP) or samples that we predict as positives but are in fact negatives; false negatives (FN) or samples that we predict as negatives but are positives. The true positive rate and the false positive rate (TPR and FPR, respectively) are defined by the following equations:

$$\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP} + \mathrm{TN}) \tag{1}$$

$$\mathrm{TPR} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN}) \tag{2}$$

TPR can be plotted against FPR to create a receiver operating characteristic (ROC) curve (Fig. 11.6). ROC curves indicate what is the fraction of correctly predicted positives (relative to all positives) as a function of the fraction of incorrectly predicted negatives (relative to all negatives). A ROC curve is obtained by setting a threshold T that separates predicted positives and predicted negatives on the output of the NN. By sliding T over the entire NN output range, we can generate a performance curve on the ROC plane.

As an example, assume that the NN produces output values between 1 and 0 and that the intended output is 1 for a positive and 0 for a negative. For $T = 1$, the number of predicted positives (i.e., predictions with a score >1) is null and so are TP and FP. Therefore, from Eqs. 1 and 2, TPR and FPR equal 0 (Fig. 11.6). As we lower the threshold, the number of TP and FP increases. The last point on the ROC

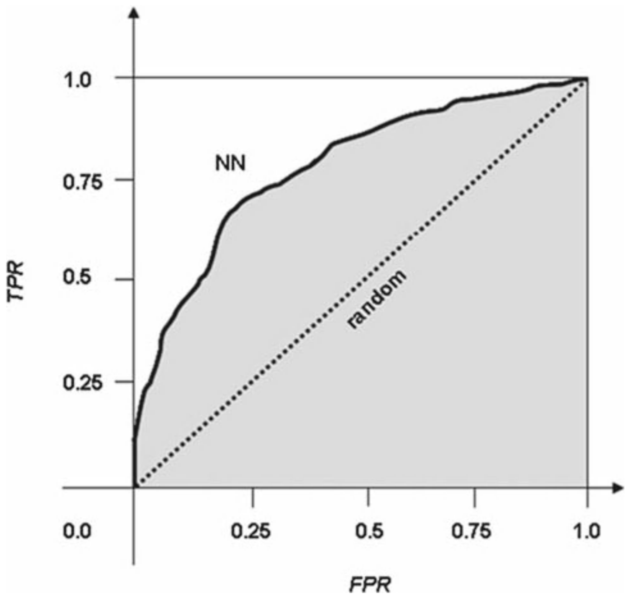


Fig. 11.6 ROC plots. The ROC is a curve in the FPR-TPR space (FPR = false positive rate, TPR = true positive rate). The ROC of a random predictor (i.e., a method that predicts true positives and false positives at the same rate) is represented by the diagonal connecting the points (0, 0) and (1, 1) (dashed line). An NN predictor is expected to be better than random (thick line). The area under the ROC curve (AUC) is a measure of the performance of the NN (shaded area). For a random predictor, AUC = 0.5

curve is obtained by choosing $T = 0$. In this case, no negatives are predicted (i.e., TN = 0 and FN = 0) and from Eqs. 1 and 2, we have TPR = 1 and FPR = 1 (Fig. 11.6). ROC curves can be used to compare a method with a random baseline or other methods that address the same prediction task. A very popular performance measure is the area under the ROC curve (AUC): the higher is the AUC, the better the method (with the maximal AUC equaling 1). In some circumstances, we may be interested in comparing two predictors, not on the entire range of TPR and FPR, but for example, only at low FPR (i.e., we may want our method to predict only a few positives but with very high accuracy). In this case, we can restrict the AUC comparison at values below a certain FPR (say, 0.05) or pick a single FPR value and compare the two predictors based on the corresponding TPR. In general, we always expect a method to be better than random. The performance of a random predictor, that is, a predictor that generates true positives and false positives at the same rate, is represented in the ROC space by a straight diagonal line (Fig. 11.6, dashed line). So, for our method to be better than random, its AUC has to be higher than 0.5 or at least higher than the diagonal line in some FPR range.

Other popular measures are the so-called Q measures. A generic Q_k measure reads:

$$\mathrm{Q_k = 100 \times \Sigma_{(i=1,k)} C_i / N} \quad (3)$$

where k is the number of classes being predicted, C_i is the number of correctly predicted samples in that class, and N is the total number of samples (over all classes). For example, if we classify residues as SS or non-SS (two classes), we have to calculate a Q_2. Maximal and minimal values for Q_k are 100 and 0, respectively (independent of the number of classes k).

It is important to remember that, in the presence of a strong imbalance in the population of the different classes (in terms of number of samples), using Q measures can be inappropriate. For example, if in a two-state classification task, only 1% of the samples are labeled as positives, Q_2 by and large reflects the method's performance on the negatives. For example, a method predicting all samples to be negatives has $Q_2 = 99$. One solution is to introduce a per-class accuracy, which (for positives) is defined as ACC = TP/(TP + FP). This is usually reported together with the TPR, also called *coverage* or *recall* in this context.

Additional measures commonly used for performance assessment in computational biology include, for regression problems, correlations. More, specific measures are often defined for each particular prediction task (for the case of SS, see [1]).

In conclusion, it has to be stressed that no measure can be singled out as the best; in fact, what is best depends on the task and the data sets at hand.

3.7.4. Comparison between Different Methods

A few issues have to be taken into account when comparing two methods that address the same prediction task. First, for the comparison to be fair, it is necessary that the two methods be compared on the same data set. This data set has to represent as well as possible the protein feature being predicted and must contain proteins that were not used for training either of the two methods. Second, as previously said, it is necessary to make sure that the presence of homologous proteins does not bias the results of the assessment. As we already learned, this is usually achieved by ensuring that the test set is sequence unique and sequence unique with respect to the training sets. For methods assessed on structural data (such as SS), the easiest way to satisfy all these conditions is to perform the evaluation on proteins whose structure was solved after both methods were developed and that are sequence unique with respect to proteins previously present in the PDB. This is what experiments like EVA [42] and LiveBench [62] tried to implement.

4. Notes

1. Redundancy across sets is not good. Unless you are trying to predict features that are not very well conserved among homologs, make your training, cross-training, and test sets sequence unique with respect to each other, that is, no protein in one set should be homologous to any in the other sets (see Sections 2.1.4, 3.3.2, and 3.3.3).

2. Redundancy within the training set sometimes is useful. Keeping some redundancy within the training set may help the NN learn better. You must be careful, though, to avoid overtraining on large families, because this may bias your predictor in a way that does not reflect the bias in the whole protein sequence space but rather the one found in your data set. It is not advised to keep redundancy within the cross-training and test sets (see Section 3.3.3).
3. Evolutionary information is relevant. When possible, include evolutionary information in the input of your NN (e.g., in the form of profiles obtained from family alignments; see Section 3.5.3).
4. Balanced training is useful. If the classes you are trying to predict are unevenly populated, balanced training may improve the performance on the less-populated classes (see Section 3.6.2).
5. Cross-validation is needed. Use hold out or cross-validation in your method's performance assessment; this allows you to avoid gross overestimates of your method's performance (see Sections 3.7.1 and 3.7.2).
6. Compare different methods on the same set of "unseen" data. To compare different methods, test the methods on the same data set and make sure the data set does not contain proteins used in training any of the methods compared (see Section 3.7.4). For redundancy reduction with respect to the training sets, read the advice in notes 1 and 2.

Acknowledgements Thanks to Hans-Erik G. Aronson (Columbia) for computer assistance; thanks to Dariusz Przybylski (Columbia) for important discussions and very useful comments on the manuscript. This work was supported by Grants U54-GM072980 and U54 GM75026-01 from the National Institutes of Health (NIH) and Grant NIH/NLM R01-LM07329-01 from the NIH and the National Library of Medicine.

Abbreviations aa: amino acids; AUC: area under the ROC curve; FN: false negative; FP: false positive; FPR: false-positive rate; NFP: number of free parameters; NHN: number of hidden nodes; NN: feedforward neural network; PDB: protein data bank; ROC: receiver operating characteristics; SS: secondary structure; TN: true negative; TP: true positive; TPR: true-positive rate

References

1. Przybylski D, Rost B (2006) Predicting simplified features of protein structure. In: Lengauer T (ed) Bioinformatics: from genomes to therapies. Wiley-VCH.
2. Blom N, Hansen J, Blaas D, Brunak S (1996) Cleavage site analysis in picornaviral polyproteins: discovering cellular targets by neural networks Protein Sci 5:2203–2216.
3. Nielsen H, Engelbrecht J, Brunak S, von Heijne G (1997) A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites Int J Neural Syst 8:581–599.

4. Nielsen H, Brunak S, von Heijne G (1999) Machine learning approaches for the prediction of signal peptides and other protein sorting signals Protein Eng 12:3–9.
5. Li X, Romero P, Rani M, Dunker AK, Obradovic Z (1999) Predicting protein disorder for N-, C-, and internal regions. Genome inform ser workshop. Genome Inform. 10:30–40.
6. Sodhi JS, Bryson K, McGuffin LJ, Ward JJ, Wernisch L, Jones DT (2004) Predicting metal-binding site residues in low-resolution structural models J Mol Biol 342:307–320.
7. Passerini A, Punta M, Ceroni A, Rost B, Frasconi P (2006) Identifying cysteines and histidines in transition metal binding sites using support vector machines and neural networks Proteins: Structure, Function and Bioinformatics 65:305–316.
8. Nair R, Rost B (2003) Better prediction of sub-cellular localization by combining evolutionary and structural information Proteins 53:917–930.
9. Emanuelsson O, Nielsen H, Brunak S, von Heijne, G (2000) Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. J Mol Biol 300:1005–1016.
10. Reinhardt A, Hubbard T (1998) Using neural networks for prediction of the subcellular location of proteins Nucleic Acids Res 26:2230–2236.
11. Jensen LJ, Gupta R, Blom N, Devos D, Tamames J, Kesmir C, Nielsen H, Staerfeldt HH, Rapacki K, Workman C, Andersen CA, Knudsen S, Krogh A, Valencia A, Brunak S (2002) Prediction of human protein function from post-translational modifications and localization features. J Mol Biol 319:1257–1265.
12. Wu CH (1997) Artificial neural networks for molecular sequence analysis Comput Chem 21:237–256.
13. Creighton TE (1993) Proteins: structure and molecular properties. WH Freeman, New York.
14. Dunker AK, Brown CJ, Lawson JD, Iakoucheva LM, Obradovic Z (2002) Intrinsic disorder and protein function Biochemistry. 41:6573–6582.
15. Dunker AK, Cortese MS, Romero P, Iakoucheva LM, Uversky VN (2005) Flexible nets. The roles of intrinsic disorder in protein interaction networks Febs J 272:5129–5148.
16. Soto C, Estrada L, Castilla J (2006) Amyloids, prions and the inherent infectious nature of misfolded protein aggregates Trends Biochem Sci 31:150–155.
17. Carugo O, Argos P (1997) Protein-protein crystal-packing contacts Protein Sci 6:2261–2263.
18. Snyder DA, Bhattacharya A, Huang YJ, Montelione GT (2005) Assessing precision and accuracy of protein structures derived from NMR data Proteins 59:655–661.
19. Brenner SE (2001) A tour of structural genomics Nat Rev Genet 2:801–809.
20. Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, Bult CJ, Tomb JF, Dougherty BA, Merrick JM, et al. (1995) Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. Scienc. 269:496–512.
21. Venter JC, Remington K, Heidelberg JF, Halpern AL, Rusch D, Eisen JA, Wu D, Paulsen I, Nelson KE, Nelson W, Fouts DE, Levy S, Knap AH, Lomas MW, Nealson K, White O, Peterson J, Hoffman J, Parsons R, Baden- Tillson H, Pfannkoch C, Rogers YH, Smith H. O (2004) Environmental genome shotgun sequencing of the Sargasso Sea. Science 304:66–74.
22. Tringe SG, Rubin EM (2005) Metagenomics: DNA sequencing of environmental samples. Nat Rev Genet 6:805–814.
23. Berman HM, Battistuz T, Bhat TN, Bluhm WF, Bourne PE, Burkhardt K, Feng Z, Gilliland GL, Iype L, Jain S, Fagan P, Marvin J, Padilla D, Ravichandran V, Schneider B, Thanki N, Weissig H, Westbrook JD, Zardecki C (2002) The Protein Data Bank. Acta Crystallogr D Biol Crystallogr 58:899–907.
24. Chandonia JM, Brenner SE (2006) The impact of structural genomics: expectations and outcomes. Science 311:347–351.
25. Petrey D, Honig B (2005) Protein structure prediction: inroads to biology. Mol Cell 20: 811–819.
26. Jacobson M, Sali A (2004) Comparative protein structure modeling and its applications to drug discovery Annual Reports in Medicinal Chemistry 39:259–276.
27. Godzik A (2003) Fold recognition methods. Methods Biochem Anal 44:525–546.
28. Watson JD, Laskowski RA, Thornton JM (2005) Predicting protein function from sequence and structural data. Curr Opin Struct Biol 15:275–284.

29. Whisstock JC, Lesk AM (2003) Prediction of protein function from protein sequence and structure. Q Rev Biophys 36:307–340.
30. Boeckmann B, Bairoch A, Apweiler R, Blatter MC, Estreicher A, Gasteiger E, Martin MJ, Michoud K, O'Donovan C, Phan I, Pilbout S, Schneider M (2003) The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res 31:365–370.
31. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman D J (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25:3389–3402.
32. Rost, B. (2003) Neural networks predict protein structure: hype or hit? In: Frasconi P (ed) Artificial intelligence and heuristic methods for bioinformatics. IOS Press, Amsterdam, pp. 34–50.
33. Wu CH, Apweiler R, Bairoch A, Natale DA, Barker WC, Boeckmann B, Ferro S, Gasteiger E, Huang H, Lopez R, Magrane M, Martin MJ, Mazumder R, O'Donovan C, Redaschi N, Suzek B (2006) The Universal Protein Resource (UniProt): an expanding universe of protein information. Nucleic Acids Res 34:D187–D191.
34. Mulder NJ, Apweiler R, Attwood TK, Bairoch A, Bateman A, Binns D, Bradley P, Bork P, Bucher P, Cerutti L, Copley R, Courcelle E, Das U, Durbin R, Fleischmann W, Gough J, Haft D, Harte N, Hulo N, Kahn D, Kanapin A, Krestyaninova M, Lonsdale D, Lopez R, Letunic I, Madera M, Maslen J, McDowall J, Mitchell A, Nikolskaya AN, Orchard S, Pagni M, Ponting CP, Quevillon E, Selengut J, Sigrist CJ, Silventoinen V, Studholme DJ, Vaughan R, Wu CH (2005) InterPro, progress and status in 2005. Nucleic Acids Res 33:D201–D205.
35. Andreeva A, Howorth D, Brenner SE, Hubbard TJ, Chothia C, Murzin AG (2004) SCOP database in 2004: refinements integrate structure and sequence family data. Nucleic Acids Res 32:D226–D229.
36. Pearl F, Todd A, Sillitoe I, Dibley M, Redfern O, Lewis T, Bennett C, Marsden R, Grant A, Lee D, Akpor A, Maibaum M, Harrison A, Dallman T, Reeves G, Diboun I, Addou S, Lise S, Johnston C, Sillero A, Thornton J, Orengo C (2005) The CATH Domain Structure Database and related resources Gene3D and DHS provide comprehensive domain family information for genome analysis. Nucleic Acids Res 33:D247–D251.
37. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 25:25–29.
38. Li W, Jaroszewski L, Godzik A (2001) Clustering of highly homologous sequences to reduce the size of large protein databases. Bioinformatics 17:282–283.
39. Holm, L, Sander C (1998) Removing near-neighbour redundancy from large protein sequence collections. Bioinformatics 14:423–439.
40. Rost B (1999) Twilight zone of protein sequence alignments. Protein Eng 12:85–94.
41. Rost B, Liu J, Nair R, Wrzeszczynski KO, Ofran Y (2003) Automatic prediction of protein function. Cell Mol Life Sci 60:2637–2650.
42. Koh IY, Eyrich VA, Marti-Renom MA, Przybylski D, Madhusudhan MS, Eswar N, Grana O, Pazos F, Valencia A, Sali A, Rost B (2003) EVA: Evaluation of protein structure prediction servers. Nucleic Acids Res 31:3311–3315.
43. Sander C, Schneider R (1991) Database of homology-derived protein structures and the structural meaning of sequence alignment. Proteins 9:56–68.
44. Mika, S, Rost B (2003) UniqueProt: Creating representative protein sequence sets. Nucleic Acids Res 31:3789–3791.
45. Ramachandran GN, Ramakrishnan C, Sasisekharan V (1963) Stereochemistry of polypeptide chain configurations. J Mol Biol 7:95–99.
46. Dunbrack RL Jr (2006) Sequence comparison and protein structure prediction. Curr Opin Struct Biol 16:374–384.
47. Pollastri G, Przybylski D, Rost B, Baldi P (2002) Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins 47:228–235.

48. Karchin R, Cline M, Mandel-Gutfreund Y, Karplus K (2003) Hidden Markov models that use predicted local structure for fold recognition: alphabets of backbone geometry. Proteins 51:504–514.
49. Chen CP, Kernytsky A, Rost B (2002) Transmembrane helix predictions revisited. Protein Sci 11:2774–2791.
50. Siew N, Fischer D (2003) Analysis of singleton ORFans in fully sequenced microbial genomes. Proteins 53:241–2451.
51. Siew N, Fischer D (2003) Twenty thousand ORFan microbial protein families for the biologist? Structure 11:7–9.
52. Kyrpides NC, Ouzounis CA (1998) Errors in genome reviews. Science 281:1457.
53. Iyer LM, Aravind L, Bork P, Hofmann K, Mushegian AR, Zhulin IB, Koonin EV (2001) Quod erat demonstrandum? The mystery of experimental validation of apparently erroneous computational analyses of protein sequences. Genome Biol 2:51.
54. Kabsch W, Sander C (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22:2577–2637.
55. Frishman D, Argos P (1995) Knowledge-based protein secondary structure assignment. Proteins 23:566–5579.
56. Chou PY, Fasman GD (1974) Conformational parameters for amino acids in helical, beta-sheet, and random coil regions calculated from proteins. Biochemistry 13:211–222.
57. Demeler B, Zhou GW (1991) Neural network optimization for E. coli promoter prediction. Nucleic Acids Res 19:1593–1599.
58. Fan, K, Wang W (2003) What is the minimum number of letters required to fold a protein? J Mol Biol 328:921–926.
59. Wang J, Wang W (1999) A computational approach to simplifying the protein folding alphabet. Nat Struct Biol 6:1033–1038.
60. Chan HS (1999) Folding alphabets. Nat Struct Biol 6:994–996.
61. Rost B. Sander C (1993) Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol 232:584–599.
62. Rychlewski L. Fischer D (2005) LiveBench-8: the large-scale, continuous assessment of automated protein structure prediction. Protein Sci 14:240–245.

Chapter 12
The Extraction of Information and Knowledge from Trained Neural Networks

**David J. Livingstone, Antony Browne, Raymond Crichton
Brian D. Hudson, David C. Whitley, and Martyn G. Ford**

Abstract In the past, neural networks were viewed as classification and regression systems whose internal representations were incomprehensible. It is now becoming apparent that algorithms can be designed that extract comprehensible representations from trained neural networks, enabling them to be used for data mining and knowledge discovery, that is, the discovery and explanation of previously unknown relationships present in data. This chapter reviews existing algorithms for extracting comprehensible representations from neural networks and outlines research to generalize and extend the capabilities of one of these algorithms, TREPAN. This algorithm has been generalized for application to bioinformatics data sets, including the prediction of splice junctions in human DNA sequences, and cheminformatics. The results generated on these data sets are compared with those generated by a conventional data mining technique (C5) and appropriate conclusions are drawn.

Keywords Bioinformatics, cheminformatics, rule extraction, C5, TREPAN, data mining, *M*-of-*N* rule

1. Introduction

The increasing availability of large amounts of both biological and chemical data has led to the need for methods to "mine" the data, that is to say, to seek information and perhaps even knowledge that the data may contain. The expectation is that the data may contain information or knowledge that the original experiments were not necessarily designed to find. This process has become known as *data mining*. Data mining can be described as the process of discovering *previously unknown* dependencies and relationships in data sets.

In the past, most data mining has been performed using symbolic artificial intelligence data mining algorithms such as C4.5 [1], C5 (a faster variant of C4.5 with higher predictive accuracy) or CART [2]. As exemplified in the other chapters in this book, artificial neural networks (ANN) have been applied very successfully to model many types of biological and chemical data. They have not been considered data mining tools, however, since they are viewed as "black boxes." This is because (most) neural networks do not use easily interpretable representations such as rules, for example, Boolean rules, or decision trees. The model in an ANN is the connections and associated weights between the artificial neurons, and extracting with clarity information and knowledge from this form of representation can be difficult. For some applications, this lack of explicability does not matter, but for others the facility to explain is important. It would be advantageous to be able to extract representations, such as decision trees and rules, which are comprehensible forms of knowledge representation, from trained neural networks for the following reasons:

- To provide explanations, which can be crucial for the acceptance of intelligent systems, to users of a system.
- To improve the generalization ability of neural networks by using an extracted set of rules to predict where in the data space the neural network may perform badly. This will suggest regions of the data space where new data should be collected and used to retrain the network.
- To automate and thereby overcome the knowledge acquisition bottleneck for symbolic AI systems. Here, a neural network can be used to generate a rule base for incorporation into the knowledge-based system.

However, probably the most important reason for extracting decision trees and rules from neural networks is to facilitate *data mining* (more precisely referred to as *knowledge extraction*). As Craven and Shavlik [3] observe, "a (learning) system may discover salient features in the input data whose importance was not previously recognized." In this case, a fundamental question is, Why bother extracting decision trees from neural networks, when symbolic data mining techniques such as ID3 and C4.5/C5 exist? Clearly, data mining with neural networks is worthwhile only if some increase in performance is observed, such as

- The decision tree or rule set extracted by the neural network technique gives a better fit to test data than symbolic techniques.
- The decision tree or rule set extracted by the neural network technique is smaller than that produced by symbolic techniques, with the neural network–based technique giving an equivalent (or superior) fit to test data than symbolic techniques.
- The decision tree or rule set extracted by the neural network technique is represented in a more comprehensible way. For some domains, particular ways of representing the extracted knowledge may be more comprehensible than others.

Obviously, a trained neural network has learned interesting relationships inherent in the data set it was trained on. However, as noted earlier, these relationships are encoded as weight vectors within the trained neural network, which are difficult to interpret (particularly because of the nonlinearities present in the network transfer functions). Statistical techniques exist to analyze the weight vectors representing trained networks (for a review of statistical techniques, see Bullinaria [4], or for alternative methods, see [5]), but such techniques can be difficult to interpret. Structures, such as rules and decision trees, are a more comprehensible way of presenting information. Because of the desire to extract comprehensible representations from the weight vectors in trained neural networks, researchers developed techniques to extract easily interpretable structures such as symbolic if-then rules and decision trees from neural networks once they have been trained. Gallant's connectionist expert systems [6] and matrix controlled inference engine [7] are two early models where expert system rules are extracted from a neural network. Many other rule extraction techniques followed, mostly designed to extract rules from multilayer perceptrons (MLPs) [*8*–24], with a smaller number being designed for application to Kohonen networks [25], recurrent networks [26, 27], and radial basis function networks [28].

The differences among these approaches to rule extraction can be categorized [29, 30] as decompositional and pedagogical. Decompositional approaches "look inside" the network itself and, therefore, involve analyzing weights between units to extract rules. Some decompositional approaches require specialized restricted weight modification algorithms [31]. Others require specialized network architectures, such as an extra hidden layer of units with staircase activation functions [32]. Pedagogical approaches treat the network as a "black box" (i.e., they do not "look inside" the network) and extract rules by observing the relationship between the network's inputs and outputs. Because of this, such algorithms are general purpose in nature and can be applied to any feedforward network architecture [33]. Other neural network–based knowledge extraction techniques do not extract rules or decision trees but enable a series of hypotheses to be tested against a trained neural network. One is generalized effect (GE) analysis [34, 35]. This technique looks for relationships between the inputs of the neural network and can detect whether pairs of inputs are being treated in a similar way by the network. In this way, GE analysis detects whether there is a systematic relationship between pairs of inputs, but it is currently unclear how such a technique can be applied to mine biological data sets.

Parallel to the development of techniques for extracting Boolean rules from trained neural networks has been the synthesis of corresponding techniques for extracting fuzzy rules. Examples of this approach include the models developed by Hayashi [36], Halgamuge and Glesner [37], Carpenter and Tan [38], and Mitra and Hayashi [39]. For some problems, fuzzy rules may be more comprehensible, and it may be that the number of rules extracted may be lower than those extracted by a technique that extracts "crisp" rules.

The rule extraction algorithms just referenced are meant to be applied at the end of the training of a network. Once extracted, the rules are fixed: There is no modification on the fly, unless the rules are reextracted (starting anew) after further training of the network. On the other hand, CLARION [40] is an agent that can extract and modify rules dynamically. Such a technique is useful if an explanation of *how the network learns to solve the problem over time* is helpful in understanding the data set being mined.

Early techniques for data mining from feedforward neural networks trained with back propagation include the SUBSET algorithm [41]. This searches initially for sets of weights containing a single link or connection of sufficient (positive) value to guarantee that the bias on the unit being analyzed is exceeded. The approach of Fu [42] proposed an exhaustive search–based algorithm to extract conjunctive rules from MLPs. The validity interval analysis technique developed by Thrun [43] uses a generate-and-test procedure and is similar to sensitivity analysis, in that it characterizes the output of the trained neural network by systematically varying the input patterns and examining the changes in the network's output classification. An alternative to the if-then form of rules, the *M*-of-*N* form, may be more suitable for applications in bioinformatics, as it confers superior understanding of the problem domain. Rules in this form state, "If *M* of the *N* conditions a_1, a_2, ..., a_m are true, then the conclusion *b* is true." It is argued [41] that some concepts can be better expressed in such a form, and use of this representation also helps avoid the combinatorial explosion in tree size found with if-then rules.

A four-step procedure is used to extract such rules, by first grouping similarly weighted links, eliminating insignificant groups, then forming rules from the remaining groups through an exhaustive search. The following steps are performed:

1. With each output node, form groups of similarly weighted links.
2. Set the link weights of all group members to the average of the group.
3. Eliminate any group that does not significantly affect the output value.
4. Optimize the biases (thresholds) of all output nodes, using the back-propagation algorithm, while holding links weights constant.
5. Form a rule for each output node, using the weights and threshold of the rule.
6. If possible, create an *M*-of-*N* rule.

A flexible data-mining algorithm operating on neural networks and using the *M*-of-*N* rule representation would therefore be useful. The TREPAN algorithm [44], which does not require a specialized neural network architecture or training, meets these requirements. This algorithm is so flexible that it is not restricted to feedforward neural networks but can be applied to other systems, including standard statistical classifiers. Moreover, TREPAN builds an *M*-of-*N*–based decision tree that represents the function that the neural network has learned by recursively partitioning the input space.

The TREPAN algorithm draws query instances by taking into account the actual distribution of training data instances in the problem domain. For real-valued input features, it uses kernel density estimates to generate a model of the underlying distribution of the data, while for discrete-valued input features, it uses empirical distributions. This model of the training data is used in a generative manner to draw instances of input vectors for presentation to the network. The TREPAN decision tree is built in a best-first manner. As each node is added to the decision tree, the algorithm tries to maximize the gain in the fidelity of the extracted decision tree to the network it is trying to model. It achieves this by looking at the significance of the distributions at consecutive levels of the tree, using the Kolmogorov-Smirnoff test for real-valued features and the Chi-squared test for discrete-valued features. Stopping criteria can be selected so that the size of the returned tree can be controlled by the user, aiding comprehensibility.

The TREPAN algorithm has been shown to have applicability for problems at the interface of biology and chemistry. For a protein coding problem, for example, TREPAN produced better performance on a novel test set, giving 61% accuracy versus 55% for C4.5. It also gave simpler, more comprehensible decision trees, with 5 internal nodes and 14 features versus 53 internal nodes and 53 features for C4.5. The features are the number of input variables from the data set incorporated into the decision tree. TREPAN therefore shows considerable promise as a data-mining tool since, in principle, it is able to extract easily interpreted rules from any trained model. Such a model may be mathematical, statistical, or indeed, a trained neural network. The following sections examine the development and application of the TREPAN algorithm to ANN trained on biological and chemical data.

2. The TREPAN Algorithm

The original implementation of the TREPAN algorithm was restricted to extracting decision trees from MLPs precluding the use of more general neural networks or other classification methods. The algorithm has been recoded in the Matlab [45] programming environment to be compatible with any relevant classifier. In this form, the tool can be linked to Matlab neural networks. Such networks may be produced using the NETLAB [46] neural networks toolbox or the proprietary Matlab neural networks toolkit. The architecture is such that the tool may be used with other black box models, such as hidden Markov models (HMMs) [47] and support vector machines (SVMs) [48].

The classifier is used simply as an *oracle* that takes a pattern to be classified as input and produces a classification as output. Apart from the oracle itself, the algorithm requires a specification of the features (variables) involved in the problem and the data on which the oracle was trained. The original generalization of the algorithm was restricted to nominal input variables and a single binary output. Given the abundance of DNA and protein sequence data, this covered a wide range of bioinformatics applications. One ouput of a generalized TREPAN is a decision tree, as shown in Fig. 12.1. The topology of the tree is represented in terms of nodes, parents, and children and leaves and forks. Binary splitting occurs at each of the nodes, so that each node has either zero or two children. A node with no children is a *leaf*; a node with two children is a *fork*.

A second output is a set of *M*-of-*N* rules, where the integer *N* is the total number of conditions and the integer *M* is the number of those conditions that need to be satisfied for the rule to apply. Further details of the generalized algorithm can be found in the paper by Browne et al. [49] and a Matlab program can be obtained from the website of the Centre for Molecular Design (www.port.ac.uk/research/cmd/software).

A common limitation of the decision tree methods is that, as the tree grows, the number of training cases reaching a node decreases, until there are inadequate examples to expand the tree further. Small sample sizes therefore restrict the amount of information that can be extracted from a particular data set. Generalized TREPAN overcomes this by generating additional input patterns by sampling the

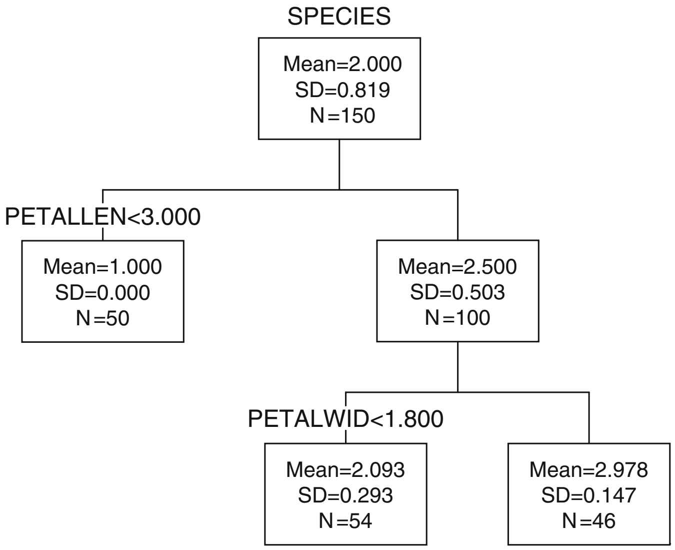


Fig. 12.1 Tree diagram of the well-known Fisher Iris data. The first binary split is on the variable petal length with all of the cases of species 1 being classified in the left-hand leaf. The right-hand node has two children, which correctly split most of the other two species on petal width

empirical distributions of the original training data. These additional, generated training cases are used alongside the original training data to expand the tree.

3. The Problems

To illustrate the power of the generalized TREPAN approach, a small number of biological and chemical problems have been chosen and analyzed. In these examples, the solution is either known or may be expected, so that the results from the algorithm may be verified. The data consist of either nominal or real values. The algorithm available for download from the Centre for Molecular Design website handles classification problems using only a binary output; a version that will operate with real-number (continuous variable) output will be released in the future.

3.1. Substance P Binding to NK1 Receptors

This data set represents the binding of substance *P* to NK1 receptors. Substance *P* is an 11- residue peptide (Fig. 12.2) involved in the neurobiology of pain and depression. The development of small-molecule antagonists of the substance *P*

H-Arg Pro Lys Pro Gln Gln Phe Phe Gly Leu Met-NH2

Fig. 12.2 The structure of substance *P*

(SP)-preferring tachykinin NK1 receptor during the past decade represents an important opportunity to exploit these molecules as novel therapeutic agents [50].

The data set attempts to use the measured biological activities of various analogues to identify the positions at which changes in stereochemistry affect the binding strength and, hence, elucidate possible binding sites. Wang et al. [51] generated data for 512 analogues of the basic peptide by systematic replacements of *L* with *D* amino acids. The data set is expressed as *D*/*L* amino acids at the respective positions in the sequence (note that Gly-9 and Met-11 were not changed in the original experiment). Here, *L* is the natural amino acid and *D* is the unnatural one, so $L \rightarrow D$ changes should be indicators of the importance of each position in relation to its biological activity, with *L* good and *D* bad. The problem is to identify the positions in the sequence where changes in the stereochemistry give rise to significant differences in the biological activity.

3.2. Identification of Splice Junction Sites in Eukaryotic DNA

This example involves the prediction of splice junction sites in human DNA sequences. The subject is important because the ability to identify the location of these sites is crucial to gene finding programs. These programs attempt to predict whether a particular DNA sequence contains a gene coding for a protein and, if so, to predict the likely gene product. The problem has been extensively studied using a variety of methodologies [52–55] and the results have been compared [56].

Splice junctions are the positions at which, after primary transcription of the DNA into RNA, the noncoding regions (introns) of a gene are excised to form edited mRNA. This is eventually used to provide the template for protein translation. The problem divides into two distinct problems, as the exon/intron, or donor, sites are composed of very different sequences than the intron/exon, or acceptor, sites. Donor sites are nearly always located immediately preceding a GT sequence. Likewise, acceptor sites immediately follow an AG sequence. Thus, GT and AG pairs within a DNA sequence are markers for potential splice junction sites. The task is therefore to identify which of these potential sites correspond to real sites and hence to make predictions of likely genes and gene products.

The data set used here is that prepared by Thanaraj [57]. This is a "clean" data set of 641 sequences from human genes, each of which has been confirmed by experiment, using expressed sequence tags. In the data set, there are 567 donor and 637 acceptor sites, which have been used as the training set positives in the calculations that follow. The window used for the donor junctions was a nine base-pair sequence starting at position –6 and ending at position +3 (where the GT marker is at positions +1 and +2). For the acceptors, a 12 base-pair window was used, from position –11 to position +1 (where the AG marker is at positions –2 and –1).

Negative training set examples were generated using recommended procedures [58]. These rely on the observation that GT or AG markers lying close to a real splice junction are much less likely to be another real junction, and so false sites can be generated by taking sequences close to confirmed real sites. Using this method, false sites were generated from both the upstream and downstream sides of the real sites.

3.3. Distinguishing Drugs from Leads

This data set comprises a set of 137 druglike molecules classified into drugs or leads [59]. For each of these a set of seven descriptors, analogous to those in the original paper, was calculated using Cerius-2 [60]. These parameters are *A*log*P*, MW, MR, *N* donors, *N* acceptors, *N* rot bonds, and the number of Lipinski violations. This data set is interesting as the intention is to derive useful "rules of thumb" in the same spirit as the Lipinski rules [61]. Indeed, as these rules are inherently cast as an *M*-of-*N* rule (i.e., if two of four conditions are met, the compound is unlikely to be bioavailable), the formalism of the TREPAN rules may be very suitable for this type of problem.

3.4. Analysis of a Molecular Dynamics Trajectory

This data set concerns the analysis of conformational data using statistical techniques. The data are derived from molecular dynamics simulations of the antidiabetic agent rosiglitazone (Fig. 12.3). These simulations were performed for a period of 5 ns, using standard techniques. The data themselves are the dihedral angles of each of the eight flexible torsion angles T1–T8. A sample structure was taken every 1 ps leading to 5,000 data points. Each of these conformations was classified as

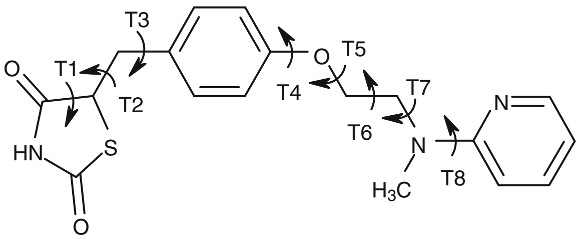


Fig. 12.3 The molecular structure of rosiglitazone with the eight torsion angles (T1–T8) used for conformation analysis

either a folded or an extended structure, based on the measured distance between the two ends of the molecule; from the simulation carried out, the conformations were divided approximately 50:50 between folded and extended.

3.5. Inhibition of HIV-1 Protease

The final example is a standard QSAR data set [62]. This comprises a set of 48 inhibitors of HIV-1 protease. These compounds were categorized into low ($pIC_{50} <$ 8) and high (pIC_{50} >8) inhibitory activity against the enzyme. The *X* block is a set of 14 parameters described in [62]. The original paper utilized a three-component PLS model with an R^2 of 0.91 and a Q^2 of 0.85. The highest loadings of the parameters of the model were X9, X11, X10, and X13, all of which were positive.

4. Knowledge Extracted

4.1. Substance P Binding to NK1 Receptors

A series of MLPs with different architectures were trained, eventually settling on an 18:3:1 network architecture as giving the best generalization of the data. The training set used 90% of the data with the remaining 10% reserved for testing. Training and testing sets were selected to give equal weightings of actives and inactives in both sets (about 3:1 active:inactive). Five separate networks were trained and, for each network, a decision tree was extracted using TREPAN. As there are only 512 possible examples in this small data set, no sampling procedure was used, and all the training data were used to generate the tree. As seen in Table 12.1, the trees showed high fidelity to the networks from which they were generated that is, they replicated the outputs of the network well. In addition, the trees retained good predictions of the actual activities. The mean accuracies for the five trained MLPs were 88.3% and 83.3% for the training and test sets, respectively. The corresponding accuracies for the decision trees extracted from them were 84.5% and

Table 12.1 Decision tree using TREPAN

Network	Fidelity	True +	True –	False +	False –
1	95.5 %	30	12	1	1
2	97.7 %	27	16	0	1
3	90.9 %	31	9	2	2
4	88.6 %	29	10	2	3
5	86.4 %	29	9	2	4
6	88.6 %	30	9	3	2

*H-Arg-Pro-Lys-***Pro**-*Gln*-**Gln-Phe-Phe**-*Gly*-**Leu**-*Met-*NH_2

Fig. 12.4 Structure of substance *P* with important residues for binding shown in bold

```
p5=G
    p3=C or p3=T => NEGATIVE
        p3=A
            p2=G => POSITIVE
            p2=A
                p4=A or p4=G => POSITIVE
                p4=C or p4=T => NEGATIVE
            p2=C
                p4=A => POSITIVE
                else => NEGATIVE
            p2=T
                p6=A or p6=G => NEGATIVE
                p6=C or p6=T => POSITIVE
        p3=G
            p4=T => NEGATIVE
            p4=C
                p6=T => POSITIVE
                else => NEGATIVE
```

Fig. 12.5 Part of the C5 decision tree for donors

86.4%. Interestingly, the equivalent results using the C5 algorithm are 87% for the training set and 79.6% for the test set.

The data were summarized as a single *M*-of-*N* rule, which is easy to understand in the context of biological sequences. The rule says that, if four of five conditions are met, then the compound will be active, where the five conditions are $P4 = L$, $P6 = L$, $P7 = L$, $P8 = L$, $P10 = L$. This rule is consistent with positions identified as important for substance *P* binding using the FIRM analysis of Young and Sacks [63], as shown in Fig. 12.4, where the important residues are highlighted in bold.

This rule is more concise than the decision trees extracted using C5. In particular, this rule does not include positions 1 and 3, which are involved in the C5 decision tree and are redundant.

4.2. Identification of Splice Junction Sites in Eukaryotic DNA

To determine the architecture of the MLPs to be used, the training set was split into training, validation, and test sets in the ratio 50:25:25. Using techniques based on the magnitude and distribution of the validation set errors [64], network architectures using a single hidden layer of ten units were selected for both the donor and acceptor problem. Once selected, the full training set was used to train five networks using the specified architecture.

C5 gave complex decision trees for both the donor and acceptor cases. A small part of the C5 decision tree for donors is shown in Fig. 12.5, the full tree extending

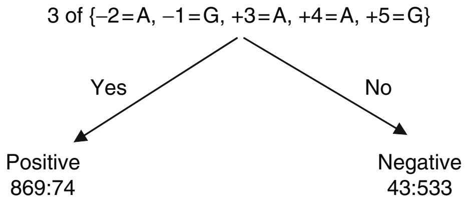


Fig. 12.6 The TREPAN *M*-of-*N* rule for donors. The number of false positives and false negatives predicted by the rule are 74 and 43, respectively

to over two pages of text. The corresponding *M*-of-*N* rule from TREPAN is shown in Fig. 12.6.

C5 gave 94% accuracy for the donor training set and 87% accuracy for the acceptor training set, whereas the results for Trepan averaged over the five ANN were 92% and 80%, respectively. The test set performances were 92% and 85% for C5 and 91% and 81% for TREPAN. Therefore, TREPAN produces comparable results to C5 but the *M*-of-*N* rules are considerably simpler than the C5 decision trees in both cases. Moreover, the rules agree with the generally accepted consensus sequence for donor sites [65]. Similar results were obtained for acceptor sites [49].

4.3. Distinguishing Drugs from Leads

The C5 algorithm was able to classify only 69% of the training examples correctly. An ANN with three hidden units gave a similar performance. Application of the TREPAN method reduced this performance to ~65% but produced a very simple *M*-of-*N* rule, involving just two descriptors, molecular weight and MR. The corresponding tree from C5 invoked molecular weight, then rule-of-5 violations, then the number of acceptors. Both methods give sensible trees consistent with the findings of the original paper, but the TREPAN tree shows the advantage of the *M*-of-*N* formalism. The 1-of-2 rule derived is essentially an OR condition, something a binary split method such as C5 cannot achieve.

4.4. Analysis of a Molecular Dynamics Trajectory

In this case, the network training procedure resulted in a network with four hidden units. The C5 algorithm produces highly accurate classifications of the conformations but does this using a very complicated decision tree, as shown in Fig. 12.7. By contrast, the TREPAN tree, shown in Fig. 12.8, is straightforward. Essentially, this tree shows how the molecule adopts a folded conformation. First, the molecule must fold about one of the central torsion angles, in this case T5. In addition, it needs required values for two outer torsion angles, T2 and T7. This second rule

```
T5 <= 269 [Mode: extended]
    T5<=52 [Mode: extended]
        T7<=185 [Mode: extended]=> extended
        T7>185 [Mode: folded]
            T6<=75 [Mode: extended]=>folded
            T6>75 [Mode: extended]
                T5<=41 [Mode: folded]
                    T8<=249 [Mode: folded] =>folded
                    T8>249[Mode: extended] => extended
                T5>41[Mode: extended] => extended
T5 <= 52 [Mode: extended]
    T6<= 73 [Mode: extended]
        T8 <= 242 [Mode: extended]
            T5 <= 7 [Mode: extended]
                T8 <=22 [Mode: extended]=>extended
                T8>22 [Mode: folded] => folded
            T5 > 7 [Mode: extended] => extended
        T8 > 242 [Mode: extended] =>extended
    T6>73 [Mode: extended] => extended
T5 > 269 [Mode: folded] => folded
```

Fig. 12.7 The C5 decision tree for the classification of the conformations of rosiglitazone

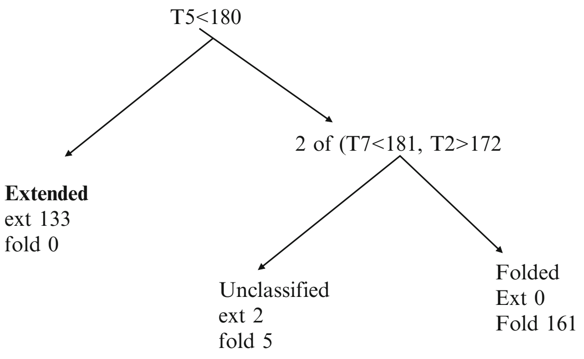


Fig. 12.8 The Trepan *M*-of-*N* rules for the classification of the conformations of rosiglitazone

again shows the value of the *M*-of-*N* formalism. In this case the 2-of-2 rule is analogous to a logical AND operation.

4.5. Inhibition of HIV-1 Protease

The C5 tree gives a simple splitting on the values of X11 and X13. Both of these have high loadings in the original PLS model. The optimal neural network was found to have two hidden units. The TREPAN tree derived from this ANN gave a primary split in a 1-of-2 rule (analogous to a logical OR) based on X13 and X9, also a variable with a high loading in the PLS model. The TREPAN rule, however, also requires two other variables (X1, X6) to achieve the same performance as the C5 tree. In this case, the TREPAN rule is more complicated than the C5 decision tree, although it achieves the same performance. It is, however, readily comprehensible and preserves the features of the original model.

5. Discussion

The five examples presented here cover a range of applications taken from bioinformatics and cheminformatics problems. In all cases, it was possible to train artificial neural networks to fit useful models, but these, of course, unveiled little knowledge about the underlying problem. A data-mining tool, TREPAN, was shown to unravel the complexities of these neural network models to produce accurate and comprehensible rules that compare well with a popular, well-established method, C5. Furthermore, the rules are simpler. For example, the splice junction donor problem can be expressed as a single *M*-of-*N* rule in contrast to the rather complicated C5 decision tree. Moreover, this rule agrees with a generally accepted sequence for donor sites. This simplicity is also clearly seen in the substance *P* example, where, once again, a single *M*-of-*N* rule is sufficient to summarize the data. This is compared to the C5 decision tree, which has 12 decision points. There is an added advantage, compared with the C5 decision trees, that the rules are straightforward to code in any software that might be developed to provide a bioinformatics tool.

A problem with symbolic decision tree methods is that they are based on binary splits. Most decision tree extraction algorithms have trouble expressing OR relationships. Such algorithms express the extracted knowledge as "IF . . . THEN . . . *else*" statements (the nodes of the tree) combined with logical ANDs (the links). The combination of neural networks and TREPAN has the advantage that OR statements can be easily represented in the extracted knowledge structure. Sampling is an important part of the TREPAN algorithm. This is the process whereby examples are generated, consistent with the distributions of the real data set. The objective of the algorithm is to represent the behavior of the neural network, rather than just predict the results for a test set.

Another very positive conclusion from this work is that the flexibility of the TREPAN algorithm enables its application to other neural network based and statistical classifiers, such as support vector machines. In addition, as the algorithm inspects only the input-output behavior of the system, it is ideally suited for application to ensembles of diverse neural net classifiers [66], using techniques such as boosting [*67*, *68*], bagging [69], and stacking [70], potentially with the application of different ensemble combination methods [71].

6. Conclusions

Because of their utility in fitting models to data, probably where nonlinearity is involved, artificial neural networks have become popular tools in the analysis of biological and chemical data, as evidenced by the other chapters in this book. A major drawback, however, is their lack of transparency. Because of this, considerable effort has been applied to the development of techniques designed to unravel the complexity of models encoded by ANNs. The key requirements are accuracy, comprehensibility, and simplicity. A number of methods have been demonstrated to go some way toward addressing these requirements and one in particular, as discussed at length in this review, shows much promise. The combination of neural networks with an algorithm to extract knowledge from the trained networks potentially offers the "best of both worlds" to biologists and chemists attempting to both make predictions on their data and understand them; that is, it is possible to combine the generalization accuracy of neural networks with the comprehensibility of the knowledge extraction method applied to them. As well as facilitating knowledge discovery, the user obtains understanding of how the network performs its task. This should lead to both more confidence in and better acceptance of the application of these techniques. The extraction of decision trees from trained neural networks is an important addition to the data-mining toolkit of knowledge extraction techniques.

7. Notes

1. TREPAN is a generalized implementation of the algorithm described by Dr. Mark Craven of the University of Wisconsin, Madison [41]. The software is available for download from www.cmd.port.ac.uk.
2. The program is written for the Matlab environment and therefore requires a functional Matlab setup. The Matlab statistics toolbox is also required. Matlab can be obtained from the MathWorks, Inc. [45].
3. The program requires an "oracle." This must be a Matlab function that returns a binary classification based on the input data. Usually, the oracle is a trained artificial neural network but can be any "black-box" classifier, such as a support vector machine.
4. It is usual to assess the performance of TREPAN using the same training data as that used to train the ANN. It is important to note the fidelity of the TREPAN tree to the original network. Therefore, the predictions of the TREPAN tree should bear a close relationship with those of the ANN, regardless of how good the predictions are with respect to the actual activity values (the accuracy).
5. Since the external network may require its data in a different format than that of TREPAN, care needs to be taken with respect to the input file formats for each.
6. The protocols for training and selecting the ANNs are fully described in [49].

7. The main determinant of the time taken to perform the analysis is the number of possible splits that need to be considered. Since this is much higher for real-valued input data, these analyses take considerably longer than those that use only nominal or integer input data.

Acknowledgement This chapter is dedicated to our dear friend and colleague, Martyn, who passed away after a brave fight with cancer on June 7, 2007.

References

1. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Mateo, CA.
2. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth, Belmont, CA.
3. Craven MW, Shavlik JW (1994) Using sampling and queries to extract rules from trained neural networks. In: Proc. of the 11th international conference on machine learning. Morgan Kaufmann, San Mateo, CA, pp. 37–45.
4. Bullinaria JA (1997) Analysing the internal representations of trained neural networks. In: Browne A (ed) Neural network analysis, architectures and algorithms. Institute of Physics Press, Bristol, UK, pp. 3–26.
5. Browne A (1997) Neural network analysis, architectures and algorithms. Institute of Physics Press, Bristol, UK.
6. Gallant SI (1998) Connectionist expert systems, Communications of the ACM 31:152–169.
7. Gallant SI, Hayashi Y (1990) A neural network expert system with confidence measurements. IPMU:562–567.
8. Saito K, Nakano R (1988) Medical diagnostic expert system based on PDP model. In: Proc. of IEEE international conf. on neural networks, pp. 255–262.
9. Shavlik J, Towell G (1989) An approach to combining explanation-based and neural learning algorithms, Connection Science 1:233–255.
10. Baba K, Enbutu I, Yoda M (1990) Explicit representation of knowledge acquired from plant historical data using neural networks. Neural Networks. 3:155–160.
11. Bochereau L, Boutgine P (1990) Extraction of semantic features and logical rules from multilayer neural networks. In: International joint conference on neural networks, Washington, DC, vol. 2, pp. 579–582.
12. Goh TH (1993) Semantic extraction using neural network modelling and sensitivity analysis. In Proc. international joint conf. on neural networks, Nagoya, Japan, pp.1031–1034..
13. McMillan C, Mozer M, Smolensky P (1993) Dynamic conflict resolution in a connectionist rule-based system. In: Proc. of the 13th IJCAI, pp.1366–1371.
14. Yeung D, Fong H (1994) Knowledge matrix: AN explanation and knowledge refinement facility for a rule induced neural network. In: Proc. 12th national conf. on artificial intelligence, vol. 2, pp. 889–894.
15. Yoon B, Lacher R (1994) Extracting rules by destructive learning. In: Neural networks, 1994. IEEE world congress on computational intelligence, pp. 1766–1771.
16. Sethi I, Yoo J (1994) Symbolic approximation of feedforward networks. In: Gesema E, Kanal L (eds) Pattern recognition in practice, IV: multiple paradigms, comparative studies and hybrid systems. North-Holland, Amsterdam, pp. 313–324.
17. Fletcher G, Hinde C (1995) Using neural networks as a tool for constructive rule based architectures. Knowledge Based Systems 8:183–187.
18. Thrun SB (1995) Extracting rules from artificial neural networks with distributed representations. In: Tesauro G, Touretzky D, Leen T (eds) Advances in neural information processing systems MIT Press, San Mateo, CA, pp. 505–512.

19. Benitez J, Castro J, Requina JI (1997) Are artificial neural networks black boxes? IEEE Trans Neural Networks 8:1156–1164.
20. Taha I, Ghosh J (1997) Evaluating and ordering of rules extracted from feedforward networks. In: Proc. IEEE international conf. on neural networks, pp. 408–413.
21. Ampratwum CS, Picton PD, Browne A (1998) Rule extraction from neural network models of chemical species in optical emission spectra. In: Proc. workshop on recent advances in soft computing, pp. 53–64.
22. Maire F (1999) Rule extraction by backpropagation of polyhedrons. Neural Networks, 12:717–725.
23. Ishikawa M (2000) Rule extraction by successive regularization. Neural Networks 13:1171–183.
24. Setiono R (2000) Extracting μ-of-n rules from trained neural networks. IEEE Trans Neural Networks 11:512–519.
25. Ultsch A, Mantyk R, Halmans G (1993) Connectionist knowledge aquisition tool CONKAT. In: Hand J (ed) Artificial intelligence frontiers in statistics AI and statistics, vol. III, Chapman and Hall, London, pp. 256–263.
26. Giles C, Omlin C (1993) Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent networks. Connection Science 5:307–328.
27. Giles C, Omlin C (1993) Rule refinement with recurrent neural networks. In: Proc. IEEE international conf. on neural networks, pp. 801–806.
28. McGarry K, Wermter S, MacIntyre J (1999) Knowledge extraction from radial basis function networks and multi layer perceptrons. In: Proc. international joint conf. on neural networks (Washington, DC), pp. 2494–2497.
29. Andrews R, Tickle AB, Golea M, Diederich J (1997) Rule extraction from trained artificial neural networks. In: Browne A (ed.) Neural network analysis, architectures and algorithms. Institute of Physics Press, Bristol, UK, pp. 61–100.
30. Tickle, A, Maire, F, Bologna, G, Andrews, R, Diederich J (2000) Lessons from past, current issues, and future research directions in extracting knowledge embedded in artificial neural networks. In: Wermter S, Sun R (eds) Hybrid neural systems. Springer-Verlag, Berlin, pp. 226–239.
31. Shavlik J (1994) Combining symbolic and neural learning. Machine Learning 14:321–331.
32. Bologna G (2000) Rule extraction from a multilayer perceptron with staircase activation functions. In: Proc. international joint conf. on neural networks, Como, Italy, pp. 419–424.
33. Craven MW, Shavlik JW (1997) Understanding time series networks. Int J Neural Syst 8:373–384
34. Browne A (1998) Detecting systematic structure in distributed representations. Neural Networks 11:815–824.
35. Browne A, Picton P (1999) Two analysis techniques for feed-forward networks. Behaviormetrika 26:75–87.
36. Hayashi Y (1991) A neural expert system with automated extraction of fuzzy if-then rules and its application to medical diagnosis. In: Lippmann R, Moody J, Touretzky D (eds) Advances in neural information processing systems, vol. 3. Morgan Kaufmann, San Mateo, CA.
37. Halgamuge S.K, Glesner M (1994) Neural networks in designing fuzzy systems for real world applications. Fuzzy Sets and Systems 65:1–12.
38. Carpenter G, Tan, A.H. (1995) Rule extraction: From neural architecture to symbolic representation. Connect. Sci 7:3–27.
39. Mitra S, Hayashi Y (2000) Neuro-fuzzy rule generation: survey in a soft computing framework. IEEE Trans Neural Networks 11:748–768.
40. Sun R, Peterson T (1998) Autonomous learning of sequential tasks: experiments and analyses. IEEE Trans Neural Networks 9:1217–1234.
41. Towell G, Shavlik JW (1993) The extraction of refined rules from knowledge based neural networks. Machine Learning 31:71–101.
42. Fu L (1994) Rule generation from neural networks. IEEE Trans Systems, Man and Cybernetics, 24:1114–1124.

43. Thrun SB (1994) Extracting provably correct rules from neural networks. Technical report IAI-TR-93–5, Institut fur Informatik III Universitat Bonn.
44. Craven MW, Shavlik JW (1997) Understanding time series networks. Int J Neural Syst 8:373–384.
45. Matlab. The Mathworks Inc., Natick, MA, www.mathworks.com/products/matlab.
46. Nabney IT (2002) NETLAB: algorithms for pattern recognition. Springer, Heidelberg, www.ncrg.aston.ac.uk/netlab.
47. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE 77:257–286.
48. Vapnik V (1995) The nature of statistical learning theory. Springer, New York.
49. Browne A, Hudson BD, Whitley DC,Ford MG, Picton P (2004) Biological data mining with neural networks: implementation and application of a flexible decision tree extraction algorithm to genomic domain problems. Neurocomputing 57:275–293.
50. Rupniak, NM, Kramer MS (1999) Discovery of the antidepressant and anti-emetic efficacy of substance P receptor (NK1) antagonists. Trends Pharmacol Sci 20:485–490.
51. Wang, JX, DiPasquale, AJ, Bray, AM, Maeji NJ, Geysen, HM (1993) Study of stereo-requirements of Substance P binding to NK1 receptors using analogues with systematic D-amino acid replacements. Biorg. Med. Chem. Lett., 3:451–456.
52. Kulp D, Haussler D, Reese MG, Eeckman FH (1996) A generalized hidden Markov model for the recognition of human genes in DNA. In: Proc. ISMB-96. AAAI/MIT Press, St. Louis, pp. 134–142.
53. Salzberg S, Chen X, Henderson J, Fasman K (1996) Finding genes in DNA using decision trees and dynamic programming. In: Proc. ISMB-96. AAAI/MIT Press, St. Louis, pp. 201–210.
54. Yada T, Hirosawa M (1996) Gene recognition in cyanobacterium genomic sequence data using the hidden Markov model. In: Proc. ISMB-96. AAAI/MIT Press, St. Louis, pp. 252–260.
55. Ying X, Uberbacher EC (1996) Gene prediction by pattern recognition and homology search. In: Proc. ISMB-96. AAAI/MIT Press, St. Louis, pp. 241–251.
56. Burset, M, Guigo R (1996) Evaluation of gene structure prediction programs. Genomics 34:353–367.
57. Thanaraj TA (1999) A clean data set of EST-confirmed splice sites from Homo sapiens and standards for clean-up procedures. Nucleic Acids Res 27:2627–2637.
58. Thanaraj TA (2000) Positional characterization of false positives from computational prediction of human splice sites. Nucleic Acids Res 28:744–754.
59. Oprea TI, Davis AM, Teague SJ, Leeson PD (2001) Is there a difference between leads and drugs? A historical perspective. J Chem Inf Comp Sci 41:1308–-1315.
60. Cerius-2. MSI Inc., San Leandro, CA.
61. Lipinski CA, Lombardo F, Dominy BW, Feeney PJ (2001) Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Adv Drug Deliv Res 46:3–26.
62. Kiralj R, Ferreira MMC (2003) A priori molecular descriptors in QSAR: a case of HIV-1 protease inhibitors, I. The chemometric approach J Mol Graph Mod 21:435–448.
63. Young S, Sacks S (2000) Analysis of a large, high-throughput screening data using recursive partitioning. In: Gundertofte K, Jorgensen FS (eds) Molecular modelling and prediction of biological activity. Kluwer Academic/Plenum Press, New York, pp. 149–156.
64. Manallack DT, Tehan BG, Gancia E, Hudson BD, Ford MG, Livingstone DJ, Whitley DC, Pitt WR (2003) A consensus neural network based technique for identifying poorly soluble compounds. J Chem Inf Comput Sci 43:674–679.
65. Watson JD, Hopkins NH, Roberts JW, Argetsinger J, Weiner A (1987) Molecular biology of the gene (4th edn). Benjamin Cummings, Menlo Park, CA.
66. Sharkey AJC, Sharkey NE, Chandroth GO (1996) Neural nets and diversity. Neural Computing and Applications 4:218–227.
67. Drucker H, Schapire R, Simard P (1993) Boosting performance in neural networks. Int J Pattern Recogn 7:705–719..
68. Schapire RE (1990) The strength of weak learnability. Mach Learn 5:197–227.

69. Breiman L (1996) Bagging predictors. Mach Learn 26:123–140.
70. Wolpert DH (1992) Stacked generalization. Neural Networks 5:241–259.
71. Yang S, Browne A, Picton P (2002) Multistage neural network ensembles. In: Proc. 3rd international workshop on multiple classifier systems, lecture notes in computer science, vol. 2364. Springer, Heidelberg, pp. 91–97.

Index