Neural Networks in Chemistry

By Johann Gasteiger* and Jure Zupan*

The capabilities of the human brain have always fascinated scientists and led them to investigate its inner workings. Over the past 50 years a number of models have been developed which have attempted to replicate the brain's various functions. At the same time the development of computers was taking a totally different direction. As a result, today's computer architectures, operating systems, and programming have very little in common with information processing as performed by the brain. Currently we are experiencing a reevaluation of the brain's abilities, and models of information processing in the brain have been translated into algorithms and made widely available. The basic building-block of these brain models (neural networks) is an information processing unit that is a model of a neuron. An artificial neuron of this kind performs only rather simple mathematical operations; its effectiveness is derived solely from the way in which large numbers of neurons may be connected to form a network. Just as the various neural models replicate different abilities of the brain, they can be used to solve different types of problem: the classification of objects, the modeling of functional relationships, the storage and retrieval of information, and the representation of large amounts of data. This potential suggests many possibilities for the processing of chemical data, and already applications cover a wide area: spectroscopic analysis, prediction of reactions, chemical process control, and the analysis of electrostatic potentials. All these are just a small sample of the great many possibilities.

1. Introduction

In many areas where complex information needs to be processed--from the stock-market to medical diagnosis and chemistry—people are suddenly talking about "neural networks". Neural networks appear to be a new secret weapon to combat a multitude of problems. In order to illustrate this, Figure 1 shows the dramatic rise in the number of publications on the use of neural networks in chemistry. What makes neural networks so attractive? Are they really a panacea for information processing ?

Neural networks were originally developed as models of information processing within the brain. This explains some of their fascination: The human brain has phenomenal processing power, far beyond that even of supercomputers today. Obviously the human brain processes information in a completely different way from that of today's conventional computers, which are constructed along the lines of the "von Neumann" architecture. A von Neumann computer works through a program (an algorithm) step by step, that is, sequentially.

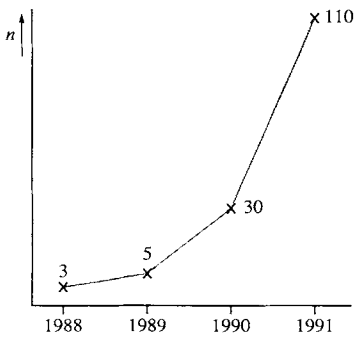


Fig. 1. Increase in the number of publications on the use of neural networks in chemistry over the period 1988-1991.

In contrast the human brain operates largely in parallel: incoming information is channeled through many processing units simultaneously. This can be demonstrated by the "100 Step Paradox": We know from neurophysiology that a nerve cell or neuron recovers approximately one millisecond after firing. On the other hand we also know that the human brain is able to perform intelligent processes, such as recognizing a friend's face or reacting to some danger, in approximately one tenth of a second. Therefore the brain is able to perform difficult tasks in less than 100 sequential steps. This small number of steps is of course insufficient to solve such complex problems, so we conclude that many tasks must be performed simultaneously and in parallel.

Artificial neural networks are nowadays usually implemented as software packages which run on conventional von Neumann computers and merely simulate parallel processing. True parallel processing is only possible on appropriate hardware (Transputers) and is still rare today. The software approach permits the use of the same program for quite different knowledge domains. The same algorithm can be used to study the relationships between chemical structure and the infrared spectrum, to simulate a tennis match, or to predict stock-market trends.

In conventional programming one tries to solve problems by finding a *problem-specific algorithm* where every instruction is tailored to exactly the task at hand. Alternatively one may solve the problem by using an expert system, which distinguishes strictly between the knowledge specific to the task and the mechanism to draw conclusions and make decisions. Neural network algorithms, however, do not belong to

[*] Prof. Dr. J. Gasteiger
Organisch-chemisches Institut
der Technischen Universität München
Lichtenbergstrasse 4, D-W-8046 Garching (FRG)
Prof. Dr. J. Zupan
National Institute of Chemistry
Hajdrihova 19, SLO-61115 Ljubljana (Slovenia)

any particular knowledge domain, but may be used generally to solve certain *classes of problems* from a wide variety of fields.

No longer is it the sequence of instructions that is specific to a task. It is the type of information that is fed into the neural network and the way it is represented there that tailor a study involving a neural network to the task at hand.

Neural networks may be used to solve the following problem types:

- Classification: An object, characterized by various properties, is assigned to a particular category.
- Modeling: Neural networks can output both binary and real values. Through this feature, by combining certain experimental results for an object we can arrive at other properties for it. Statistical methods produce such relationships by using an explicit mathematical equation. Neural networks however are able to express such relationships implicitly; this is especially useful in cases where an explicit equation can not be set up.
- Association: Neural networks can be used for the task of comparing information because they are able to store information of similar kinds. Thus they are able to recognize that two pictures of a face depict the same person, even when one of the pictures is distorted (autoassociation). Furthermore, they can be used for associative tasks where one object has a particular relation to another object (heteroassociation).
- Mapping: Complex information can be transformed into a simpler representation (e.g., projection onto a plane) while preserving all essential information.

In this introductory article we shall first introduce the basic features of the various types of neural networks before giving an overview and selected examples of the application of neural nets in the field of chemistry.[1, 2]

2. Neurons and Networks

In this review the term neural networks always refers to "artificial neural networks", because these were developed in order to emulate the biological neural networks of the human brain. However for simplicity the epithet "artificial" is omitted here.

2.1. A Model of a Neuron

Neural networks consist of subelements, the neurons, which are connected together to form a network. The artificial neuron is supposed to model the functions of the biological nerve cell. Although there are at least five physiologically distinct types of nerve cell, we need only present one type here (Fig. 2), since we discuss only the basic structure of a neuron; the physiological processes—and the chemical processes[3] that cause them—cannot be examined in more detail.

The nerve's cell body possesses a large number of branches, known as dendrites, which receive the signals and pass them on to the cell body. Here the signals are accumulated, and when a particular threshold limit has been exceeded, the neuron "fires". An electrical excitation is transmitted across the axon. At its end each axon has contact with the dendrites of the neighboring neurons; this contact point is called the

Johann Gasteiger was born in Dachau in 1941. He studied chemistry at Munich and Zurich universities with a doctoral thesis (1971) on "Mechanistic Investigations into Reactions in the Cyclooctatetraene System" under R. Huisgen. In his post-doctorate studies (1971–1972) he carried out ab initio calculations on carbanions under A. Streitwieser, Jr. at the University of California in Berkeley. In 1972 he transferred to the Technical University of Munich where he developed, along with a group led by I. Ugi, a prototype for a synthesis-planning program. He received his "habilitation" in chemistry in 1978 with a study on models and algorithms for investigating chemical reactions and reactivity. Since 1989 he has been professor at the Technical University of Munich. His main area of research is the development of methods and computer programs for predicting reactions and planning syntheses, for evaluating and simulating mass spectra, and for the three-dimensional modeling of molecules. From 1987–1991 he was the project manager for the Fachinformationszentrum Chemie in the implementation of a reaction database based on ChemInform, and in 1991 was awarded the Gmelin–Beilstein memorial medal from the German Chemical Society for his achievements in the field of computer chemistry.

Jure Zupan was born in Ljubljana, Slovenia in 1943. He studied physics at the University of Ljubljana, where he received his doctorate in 1972 under D. Hadzi with a thesis on the energy bands in boronitrides. Until 1973 he worked at the Josef Stefan Institute in the field of quantum chemistry and the magnetic properties of ceramic materials. Since 1974 he has led a group at the Institute for Chemistry in Ljubljana. His areas of work include chemometrics, artificial intelligence, and the development of expert systems and algorithms for chemical applications. In 1982 he was visiting professor at Arizona State University/USA, in 1988 at the Vrije Universiteit in Brussels, and 1990–1992 at the TU München in Garching. For his research work (approximately 150 original publications, 2 monographs, and 3 books) he received the highest Slovenian award for research in 1991. He obtained his habilitation in 1975 and has been Professor of Chemometry at the University of Ljubljana since 1988.

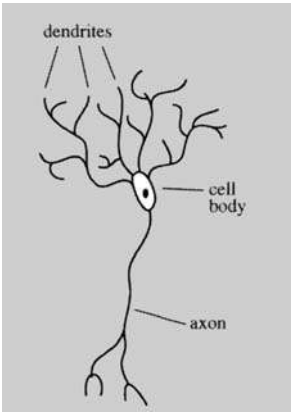


Fig. 2. Much simplified scheme of a nerve cell. The number of dendrites and the number of branches in the dendrites are much higher in reality.

synapse. Neurons are linked with each other across these synapses.

The synapses, however, also present a barrier that alters the intensity of the signal during transmission. The degree of alteration is determined by the synaptic strength. An input signal of intensity x_i has an intensity of s_i after crossing synapse i of strength w_i [Eq. (a), Fig. 3]. The synaptic strength may change, even between one impulse and the next.

$$s_i = w_i x_i \qquad \text{(a)}$$

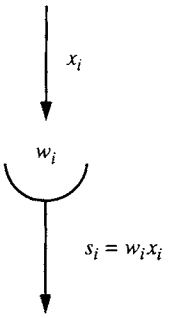


Fig. 3. Transformation of an input signal x_i on passage through a synapse of strength w_i.

Each neuron has a large number of dendrites, and thus receives many signals simultaneously. These m signals combine into one collective signal. It is not yet known exactly how this net signal, termed *Net*, is derived from the individual signals.

For the development of artificial neurons the following assumptions are made:

1. The *Net* is a function of all those signals that arrive at the neuron within a certain time interval, and of all the synaptic strengths.
2. The function is usually defined as the sum of the signals s_i, which in turn is given by the product of the input signals x_i $(i = 1, \ldots m)$ and the synaptic strengths w_i $(i = 1, \ldots m)$, now referred to as weights [Eq. (b)]. Figure 4 shows the model of a neuron as developed up to this point.

$$Net = s_1 + s_2 + \ldots s_i + \ldots s_m = w_1 x_1 + w_2 x_2 + \ldots w_i x_i + \ldots w_m x_m \qquad \text{(b)}$$

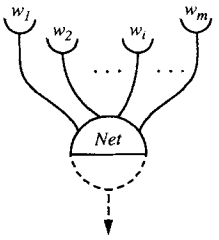


Fig. 4. First stage of a model of a neuron.

The net signal *Net* is, however, not yet the signal that is transmitted, because this collective value *Net* can be very large, and in particular, it can also be negative. It is especially the latter property that cannot be a good reflection of reality. A neuron may fire or not, but what is the meaning of a negative value? In order to attain a more realistic model, the value of *Net* is modified by a transfer function. In most cases a sigmoid function, also known as a logistic or Fermi function, is used. With this transfer function the range of values for the ouput signal *out* [Eq. (c)] is restricted to between zero and one, regardless of whether *Net* is large or small or negative.

$$out = \frac{1}{1 + e^{-(\alpha Net + \vartheta)}} \qquad \text{(c)}$$

Most important, we now have a nonlinear relationship between input and output signals, and can therefore represent nonlinear relationships between properties, a task which can often only be carried out with difficulty by statistical means. Moreover, in α and ϑ we now have two parameters with which to influence the function of the neuron (Fig. 5).

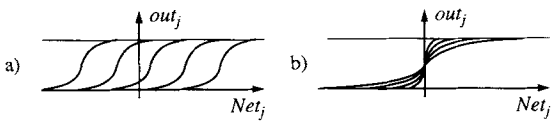


Fig. 5. Influence of α (a) or ϑ on the output signal out_j, defined as in Equation (c).

The transfer function completes the model of the neuron. In Figure 6a the synaptic strengths, or weights w are still depicted as in Figure 4; in the following figures they will no longer be shown, as in Figure 6b, but must of course still be used.

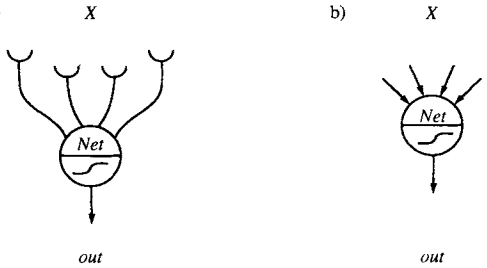


Fig. 6. Complete model of a neuron a) with and b) without explicitly defined synapse strengths w.

Symbols and Conventions

The literature on neural networks uses a confusing array of terminologies and symbols. In order that the reader may better compare the individual networks, a standard nomenclature will be followed throughout this article:

- Magnitudes that consist of a single value (scalar magnitudes) will be represented by lower-case letters in italics x_i. (The only exception is *Net* which is capitalized in order to distinguish it from the English word net. Moreover, though *Net* and *out* are symbols for single pieces of data, they are written with three letters so as to be more easily readable.)
- Data types which consist of several related values (vectors or matrices) are symbolized by a capital letter in bold italics: $\boldsymbol{X}$.
- An input object that is described by several single data (e.g., measured values from sensors) will thus be represented by $\boldsymbol{X}$, whereas the individual values are given by $x_1, x_2, \ldots x_m$. A single input value from this series is specified with the index i, thus x_i. A single neuron from one group (layer) of n neurons will be labeled with the index j; the whole of the output signals from these n neurons will be denoted $\boldsymbol{Out}$ $(out_1, out_2, \ldots out_n)$: The output signal from any one individual thus has the value out_j.
- In a layer of n neurons receiving m input data there are $n \times m$ weights that are organized into a matrix $\boldsymbol{W}(w_{11}, w_{12} \ldots w_{nm})$. A single weight from this matrix will be labeled w_{ji}.
- If there are more than one input objects, they will be distinguished by the index s: thus $\boldsymbol{X}_s$; the individual data will then be x_{si}.
- In a multilayered network the various layers will be labeled with the superscript l, e.g., out_j^l.
- Iterations in a neural network are characterized by the superscripts t, which are written in parentheses, e.g., $\boldsymbol{W}^{(t)}$.

2.2. Creating Networks of Neurons

The 100-step paradox teaches us that the advantage of the human brain stems from the parallel processing of information. The model of a neuron that we have just presented is very simple, but even much more complicated models do not provide any great degree of increased performance. The essential abilities and the flexibility of neural networks are brought about only by the interconnection of these individual arithmetic units, the artificial neurons, to form networks.

Many kinds of networking strategies have been investigated; we shall present various network models and architectures in the following sections. Since the most commonly applied is a layered model, this network architecture will be used to explain the function of a neural net.

In a layered model the neurons are divided into groups or layers. The neurons of the same layer are not interconnected, but are only linked to the neurons in the layers above and below. In a single-layered network all the neurons belong to one layer (Fig. 7). Each neuron j has access to all input data $\boldsymbol{X}$ $(x_1, x_2, \ldots x_i, \ldots x_m)$ and generates from these an output value which is specific to this neuron, out_j.

In Figure 7 the input units are shown at the top. They do not count as a layer of neurons because they do not carry out any of the arithmetic operations typical of a neuron, namely

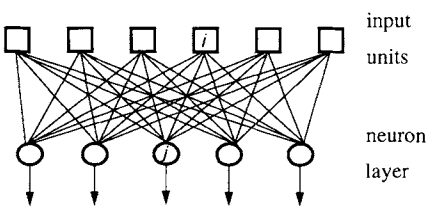


Fig. 7. Neural network with input units (squares) and one layer of active neurons (circles).

the generation of a net signal *Net*, and its transformation by a transfer function into an output signal *out*. In order to distinguish them from neurons, which are represented as circles in the following diagrams, input units will be shown as squares.

The main function of the input units is to distribute input values over all the neurons in the layer below. The values that arrive at the neurons are different, because each connection from an input unit i to a neuron j has a different weight w_{ji}, representing a specific synaptic strength. The magnitudes of the weights have to be determined by a learning process, the topic of Section 4.

The output value out_j of a neuron is determined by Equations (d) and (e), which are generalizations of Equations (b)

$$Net_j = \sum_{i=1}^{m} w_{ji}\, x_i \tag{d}$$

$$out_j = \frac{1}{1 + e^{-(\alpha\, Net_j + \vartheta)}} \tag{e}$$

and (c). The index j covers all n neurons and the index i all m input values.

In a single-layered network the output signals out_j of the individual neurons are already the output values of the neural network.

Equations (d) and (e) suggest a more formal representation for the neuron and the neural network. The input values can be interpreted as a vector $\boldsymbol{X}$ $(x_1, x_2, \ldots x_i, \ldots x_m)$ that is transformed by the matrix of weights $\boldsymbol{W}$ with elements w_{ji} and the transfer function into the vector of output values $\boldsymbol{Out}$ $(out_1, out_2, \ldots out_j, \ldots out_n)$ (Fig. 8).

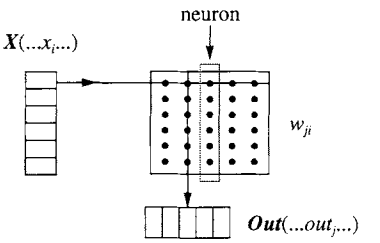


Fig. 8. Matrix representation of a one-layered network, which transforms the input data $\boldsymbol{X}$ into the output data $\boldsymbol{Out}$ by using the weights w_{ji}.

Each neuron represents a column in the matrix in Figure 8. In this matrix representation it is emphasized that every input value is fed into every neuron. The implementation of the layered model as algorithms is also realized in the matrix representation.

A single layer of neurons is known as a perceptron model and offers only limited flexibility as yet for the transformation of input values into output values. These limitations can be overcome by using several single layers in succession.

In a multilayered model the architecture chosen usually connects all the neurons of one layer to all the neurons in the layer above and all the neurons in the layer below. Figure 9 shows a two-layered neural network. (As we mentioned previously, the input units do not count here, because they are not neurons but serve only to distribute the input values across the neuron layer below.) The network user cannot

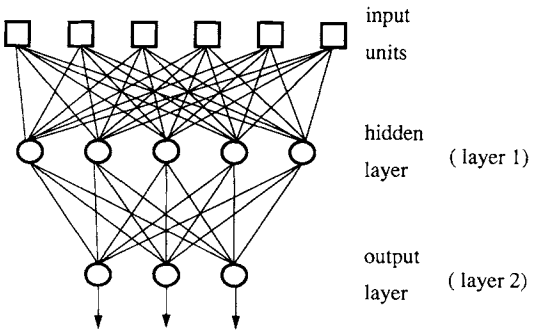


Fig. 9. Neural network with input units and two layers of active neurons.

access the first layer of neurons, which is therefore known as a hidden layer; the neurons in it are called inner neurons. The output values $\boldsymbol{Out}^1$ of the first layer of neurons are the input values $\boldsymbol{X}^2$ of the second layer of neurons. Thus each neuron in the upper layer passes its output value on to every neuron in the layer below. Because of the different weights w_{ji} in the individual connections (synapses) the same output value $\boldsymbol{Out}^1 = \boldsymbol{X}^2$ has a different effect on each individual neuron [Eq. (d)]. The result of the neural network as a whole is only given by the last layer in the network (here $\boldsymbol{Out}^2$). Figure 10 shows a two-layer network in matrix notation.

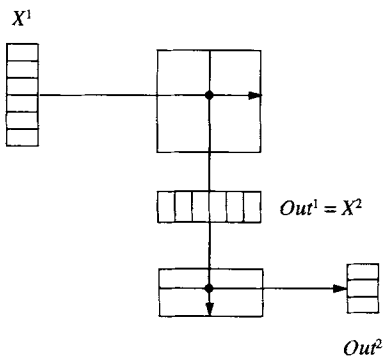


Fig. 10. Matrix representation of a two-layered network.

3. Historical Perspective

The dramatic increase in interest in neural networks does much to disguise the fact that people have been developing them for nearly 50 years.[4] As early as the forties McCulloch and Pitts[5, 6] developed a model of a neuron as a logical threshold element with two possible states. They were forced to leave open the question of how neurons learn, to which Hebb (1949) suggested an answer.[7] This Hebb learning rule states that the synaptic strength of a neuron varies proportionally with the activity in front of and behind the synapse.

Rosenblatt[8] organized a number of neurons in such a way that they corresponded essentially to a one-layered network (Fig. 7), and named this formation a perceptron.

The original enthusiasm for the development of models of biological nervous systems began to wane in the sixties for various reasons. The results with the networks of that time were hardly relevant to solving real-world problems and thus not very encouraging. With the advent of the computer, interest shifted more and more to problem-solving by directly programmed methods. The buzz-word invented for this, "artificial intelligence", shows only too clearly what was expected of this technology, although it was clear from the outset that the method chosen had very little to do with the processing of information as performed by the human brain.

Not surprising therefore that research into neural networks suffered a heavy blow from none other than Marvin Minsky, one of the main proponents of artificial intelligence. In 1969 together with Papert[9] Minsky published a very severe but probably justified criticism of the network models of the time. They showed in their theoretical study that perceptrons—at least as they were being developed at that time—offer only limited possibilities. Moreover they speculated that extending the architecture of perceptrons to more layers would not bring about any significant improvement in results. As a consequence of this criticism from so influential a personage as M. Minsky, research funding for the modeling of biological nervous systems became virtually unavailable.

In the years that followed very little work was done on neural network models; however, even at this time some important advances were made. The work of Albus,[10] Amari,[11] Grossberg,[12] Kohonen,[13] von der Malsburg,[14] and Widrow and Hoff[15] all deserves mention.

A decisive new impetus was given in 1982 by a physicist, Hopfield,[16] who was able to show that network models of binary neurons correspond formally to spin systems, and can be manipulated by the methods already developed for them.

The greatest boost for the application of neural networks then came through the publication of the "back-propagation" algorithm for learning in multilayered models, introduced by Rumelhart, Hinton, and Williams.[17, 18] Even though the back-propagation algorithm had been suggested earlier,[19] credit must go to the research group on "Parallel Distributed Systems"[20] for bringing it to the attention of the public.

Despite all the successes in the development of models for neural information processing, we must clearly acknowledge the fact that we are still very far from an understanding of the way the human brain works. The capacities of artificial neural networks must still be reckoned very rudimentary in comparison with the biological networks they are supposed to emulate. Nevertheless even these elementary models of neural networks have shown new ways of processing information. It is precisely to the possibilities they are opening up, especially in the area of chemistry, that we wish to devote ourselves in this article.

4. Architectures and Learning Processes

Over the years a whole series of research groups have developed their own characteristic artificial neural networks. Some of these models are closer to their biological counterparts than others; some seem to emulate certain processes in the human brain very well, whereas others bear only a slight similarity to their biological forbears.

Our primary concern here however is not the faithfulness to reality of each neural network model. We wish rather to

show which kinds of problems can be processed with the various models, and what capacities these neural networks offer to information processing. An artificial neural network can still have great significance for information processing, even if it bears but little relationship to biological systems.

As we mentioned in the introduction, neural networks can be applied to a whole series of problems: classification, modeling, association, and mapping. Each neural network is more or less suitable for handling these problem types; each has its own characteristic strengths and weaknesses. We wish expound upon this here and to develop some feeling for which network model is best applied to a particular task.

Three elements essentially characterize every neural network model:

1. the arithmetic operation in a neuron
2. the architecture of the net, that is, the way the individual neurons are connected
3. the learning process that adapts the weights so that the correct result is obtained.

The model of a neuron mentioned in Section 2 is only one of many, albeit a very common one. Even the organization of neurons into layers need not necessarily be that described in Section 2. The learning process, though, is very tightly coupled to the architecture of the neural network. There are essentially two types of learning processes: learning with or learning without instruction ("supervised" and "unsupervised" learning).

In supervised learning the neural net is presented with a series of objects. The input data $\boldsymbol{X}$ from these objects is given to the net, along with the expected output values $\boldsymbol{Y}$. The weights in the neural net are then adapted so that for any set p of known objects the output values $\boldsymbol{Y}$ conform as closely as possible to the expected values $\boldsymbol{Y}$ (Fig. 11).

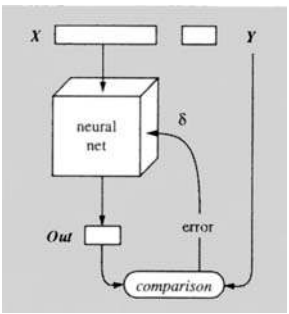


Fig. 11. Supervised learning. A comparison with expected values yields an error δ, which determines whether further adjustment cycles are necessary.

In unsupervised learning the input data are passed repeatedly through the network until it has stabilized, and until the input values map an object into certain areas of the network (Fig. 12).

4.1. The Hopfield Model

The American physicist Hopfield brought new life into neural net research in 1982 with his model.[16] He pointed out analogies between neural networks and spin systems and

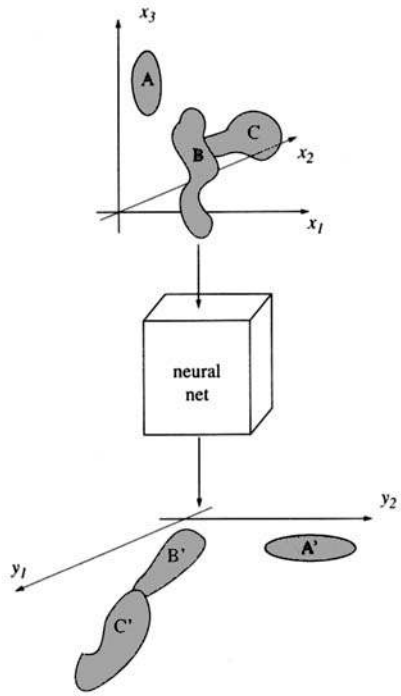


Fig. 12. Unsupervised learning.

was thereby able to apply a whole series of mathematical methods from theoretical physics to research into neural networks. In addition to this he is also responsible for the introduction of nonlinear transfer functions.

The Hopfield net performs one of the most interesting functions of the human brain: it can *associate*. This means that stored pictures (or any other kind of complex information that can be represented as a multidimensional vector or matrix) can be recognized even from partial information or distorted versions of the original picture. For example, one could recognize one particular face from a collection of faces, even if only the eyes and nose are shown.

The Hopfield net is a one-layered model that has exactly the same number of neurons as input nodes. Because every input node is connected to every neuron (Fig. 7), $m \times m$ weights must be determined for m input nodes. The original Hopfield model works with bipolar input data ($+1$ or -1) (f). The net result in neuron j is attained by the multiplication

$$x_i \begin{cases} +1 \\ -1 \end{cases} \qquad \text{(f)}$$

of all input signals x_i by the weights w_{ji} for that neuron, as in Equation (d). The transfer function here is a simple step function (Fig. 13) as may be produced by the sign of the value.

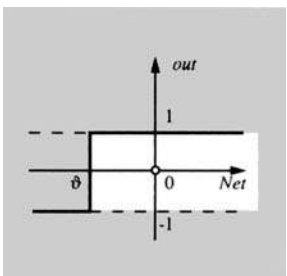


Fig. 13. A step function as a transfer function.

The output signal of a neuron, out_j, in a Hopfield net is given in Equation (g), where as already mentioned, x_i can only assume the values $+1$ and -1.

$$out_j = \text{sign}(Net_j) = \text{sign} \sum_{i=1}^{m} w_{ji}\, x_i \qquad \text{(g)}$$

In order to understand the learning process in Hopfield networks we introduce an example at this point: We require a Hopfield net to learn the four pictures in Figure 14. Each

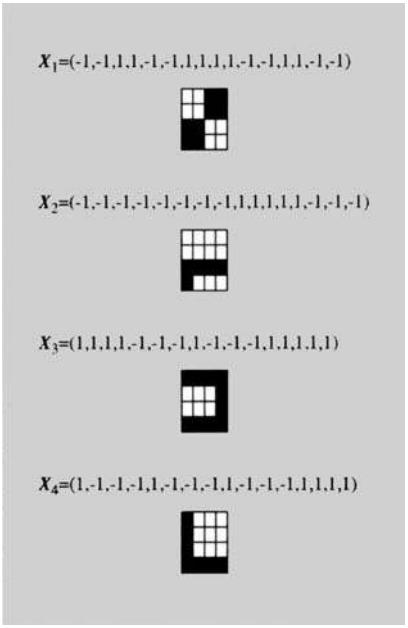


Fig. 14. Four pictures X_1, X_2, X_3, and X_4 used to train a Hopfield network.

picture consists of 4×4 fields (pixels) that are either black ($+1$) or white (-1), so that each picture may be represented as a $4 \times 4 = 16$ dimensional vector. Figure 15 shows the architecture of the corresponding Hopfield net and the input of the first picture with values represented by -1 (for a white pixel) and $+1$ (for a black one). In the picture used for training, the output values are exactly equal to the input values.

The weights w in a Hopfield net do not have to be derived from a costly, iterative learning process, but can be calculated directly from the pictures: If p pictures are presented to

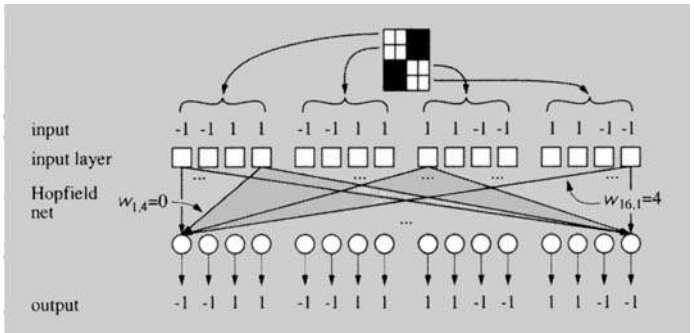


Fig. 15. Architecture of the Hopfield net complete with input and output of the first picture of Figure 14. The given w_{ji} values are determined from Equation (h) for the four pictures of Figure 4.

the Hopfield net, then the weights w_{ji} for the neuron j may be derived from the input values of the individual pictures s and thus $\boldsymbol{X}_s$ according to (h) and (i). This means that the weight

$$w_{ji} = \sum_{s=1}^{p} x_{sj}\, x_{si} \quad \text{for} \quad j \neq i \qquad \text{(h)}$$

$$w_{ji} = 0 \quad \text{for} \quad j = i \qquad \text{(i)}$$

w_{ji} increases by 1 if, in a particular picture, the pixels j and i are either both black or both white, and decreases by 1 if, in that picture, the ith and jth pixels are of different colors. The more pictures there are in which pixels j and i match, the greater is the weight w_{ji}. The 4×4 pictures from Figure 14 are stored accordingly in a Hopfield net consisting of 16 neurons with 16 weights in a weight-matrix of dimensions 16×16.

We should first test whether a Hopfield net has stabilized. For this purpose an input vector (e.g., one of the four pictures from Fig. 14) is fed into the Hopfield net, and the output values derived from Equation (g). These output values are compared to the input values. If they are equal then the process can terminate, otherwise the output values are fed into the net again as new input values and the process is repeated (Fig. 16). If after a few cycles one receives as output the same values as the original input, then one can say that the net has stabilized. Of course a Hopfield net is not intended simply to reproduce its own input values as output; this is just a stability test.

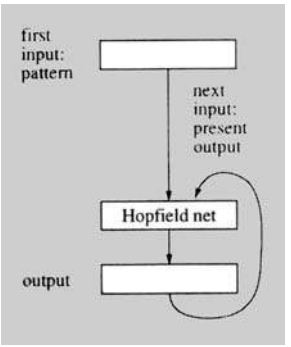


Fig. 16. Testing a Hopfield net for stability.

The true value of a Hopfield net becomes evident in the retrieval of the original stored data from incomplete or distorted data; we can for instance retrieve an original picture in the Hopfield net from a blurred or spoiled picture.

We shall investigate this phenomenon with the four pictures from Figure 14 and a Hopfield net in which they are stored in the form of 16×16 weights. We produce distorted pictures by altering a certain number of pixels in the original pictures, that is, by changing fields from black to white and vice versa. In Figure 17 we have shown the results obtained when each picture is altered by two, five, or indeed even thirteen pixels.

One can see that with a distortion of two pixels the original pictures are retrieved correctly after only 1–2 iterations. When five fields are altered (a noise level of 31%!), the

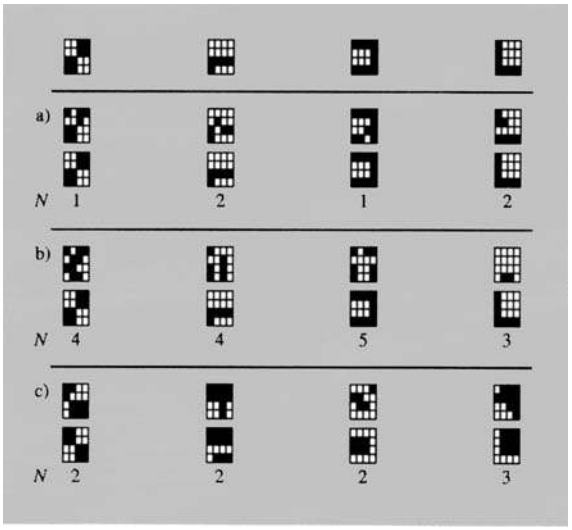


Fig. 17. Search for original pictures (showed at the top) stored in a Hopfield network after input of pictures with a) two, b) five, and c) thirteen altered fields. N is the number of iterations required to arrive at the result (bottom line in a–c) when presented with the erroneous input (top line in a–c).

original pictures are still found successfully, but only after 3–5 iterations. This is not always so; we have observed cases with a five-pixel distortion where the wrong picture is output, or the right picture in negative form (black and white pixels swapped), or an oscillation occurs between two patterns that represent none of the stored pictures, so that the operation has to be aborted.

If thirteen pixels in the pictures are altered (81% distortion) the original pictures are retrieved after 2–3 iterations as negatives. How does this come about? After all, significantly more than half the fields have been altered in color. It is remarkable that the negative of the original picture is returned, and not the negative of some other picture.

Thus we have seen that as simple a model as a Hopfield net can still reproduce one of the human brain's more interesting faculties, namely the power of association. The Hopfield net does however have one significant drawback: The number of patterns (pictures, vectors) that can be stored is severely limited. In order to store more pictures an increased number of neurons is required; thus the size of the weight matrix grows very rapidly.

4.2. An Adaptive Bidirectional Associative Memory

With the Hopfield net we have shown that a neural network has the power to associate. An adaptive bidirectional associative memory (or ABAM)[21] can do this as well. We shall now show, however, that an ABAM is able, in addition, to combine patterns. An ABAM is, like a Hopfield net, a one-layered network. The number of output neurons n is however usually much lower than the number of input units m.

In order to simplify the notation, and because in the case of ABAMs the meanings of input and output are apt to blur, in the following sections we refer to an input vector as $\boldsymbol{X}$ and the output values as $\boldsymbol{Y}$. Thus we always observe pairs of input ($\boldsymbol{X}_s$) and output ($\boldsymbol{Y}_s$) values (the index s characterizes the individual input values and their corresponding outputs):

$$\boldsymbol{X}_s = (x_{s1}, x_{s2}, \ldots x_{sj}, \ldots x_{sm})$$
$$\boldsymbol{Y}_s = (y_{s1}, y_{s2}, \ldots y_{si}, \ldots y_{sn})$$

Given a series of such pairs $\{\boldsymbol{X}_s, \boldsymbol{Y}_s\}$ where it is known which $\boldsymbol{Y}_s$ value is to be expected from a particular $\boldsymbol{X}_s$ value, the weights are determined in a supervised learning process.

First starting values are calculated for the weights according to Equation (j). The weight matrix is now no longer square, as in the Hopfield net, but rather rectangular. It has dimensions $m \times n$.

$$w_{ji} = \sum_{s=1}^{p} x_{sj}\, y_{si} \qquad \text{(j)}$$

The basic idea of learning with an ABAM net is that one can multiply an $m \times n$ matrix in two different ways: the standard way by multiplying by an m-dimensional vector, which results in an n-dimensional vector, or in the alternative transposed form, by multiplying by an n-dimensional vector to give an m-dimensional vector (Fig. 18).

$$\{\boldsymbol{X}^{(0)}, \boldsymbol{Y}^{(0)}\} \longrightarrow \boldsymbol{W}^{(0)}$$
$$\boldsymbol{X}^{(0)}\boldsymbol{W}^{(0)} \longrightarrow \boldsymbol{Y}^{(1)}$$
$$\boldsymbol{Y}^{(1)}\boldsymbol{W}^{(0)\mathrm{T}} \longrightarrow \boldsymbol{X}^{(1)}$$
$$\{\boldsymbol{X}^{(1)}, \boldsymbol{Y}^{(1)}\} \longrightarrow \boldsymbol{W}^{(1)}$$
$$\vdots \longrightarrow$$
$$\boldsymbol{X}^{(t)} = \boldsymbol{X}^{(0)}$$
$$\boldsymbol{Y}^{(t)} = \boldsymbol{Y}^{(0)}$$

Fig. 18. The learning process in an adaptive bidirectional associative memory (ABAM).

The learning process is as follows: An initial weight matrix $\boldsymbol{W}^{(0)}$ is calculated from the predefined pairs $\{\boldsymbol{X}^{(0)}, \boldsymbol{Y}^{(0)}\}$. From $\boldsymbol{X}^{(0)}$ and $\boldsymbol{W}^{(0)}$ the output values $\boldsymbol{Y}^{(1)}$ are produced, which do not as yet correspond to the goal values. Therefore these $\boldsymbol{Y}^{(1)}$ values are multiplied by the transposed matrix $\boldsymbol{W}^{(0)\mathrm{T}}$ to give a set of $\boldsymbol{X}^{(1)}$ values. Applying Equation (j) to the pairs $\{\boldsymbol{X}^{(1)}, \boldsymbol{Y}^{(1)}\}$ we calculate a new weight matrix $\boldsymbol{W}^{(1)}$. This process is repeated until the pair $\{\boldsymbol{X}^{(t)}, \boldsymbol{Y}^{(t)}\}$ corresponds to the predefined values $\{\boldsymbol{X}^{(0)}, \boldsymbol{Y}^{(0)}\}$.

In the procedure the $\boldsymbol{Y}$ values are calculated in the usual way. Here, too, the individual x and y values are bipolar $(+1, -1)$ [Eq. (k) and (l)].

$$Net_j = \sum_{i=1}^{m} w_{ji}\, x_i \qquad \text{(k)}$$

$$y_i = \mathrm{sign}(Net_j) \qquad \text{(l)}$$

As with the Hopfield net we also use simple pictures here as an illustration of how an ABAM may be applied. Pictures made of 5×5 fields, where each field may each be black $(+1)$ or white (-1), will serve as input information. These can

thus be represented by a 25-dimensional vector, so that 25 input units will be required. We use only five such pictures and identify them on the output side by a 5-dimensional vector. Each of the five patterns (pictures) is assigned one of the five positions in the vector, thus in the case of the first picture the first position is 1 and all the others are zero (10000), the second picture is represented by (01000) etc. (We are actually dealing with bipolar values +1 and −1 here, but for clarity's sake we use binary notation (1,0) in the following sections). Figure 19 shows the five patterns and their identifications. The ABAM has a 25×5 architecture, and thus 25×5 weights.

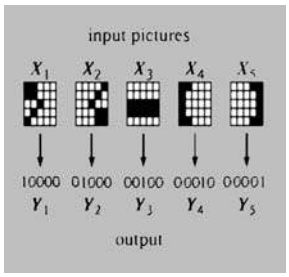


Fig. 19. Five pictures used to train the ABAM, their symbols, and vector representations.

An ABAM was trained with these five pictures and the five-dimensional vector (consisting of four zeros and one 1 in each case). Then the ABAM's ability to associate distorted pictures with stored undistorted ones was put to the test. All possible 1-pixel errors were generated; in the case of five pictures each of 5×5 pixels this makes a total of 125 different distorted pictures. The ABAM identified the correct undistorted output picture in all cases.

Next the ability of the ABAM to combine individual pieces of information was tested: that is, can it recognize, when presented with pictures made of two patterns, which combination of two pictures it is, despite previously having been presented with the patterns only singly? Figure 20 shows all ten permutations of pairs for the five patterns, with the answer from the ABAM written underneath as a five-bit vector. In eight cases the ABAM returns a 1 in exactly the right two positions. Thus, for example, for the first combined pattern, which consists of the patterns 1 and 2, it returns the answer (11000), meaning that first and second patterns are contained in it. In the cases of the other two pattern combinations (1 + 3 and 3 + 5) the ABAM also recognizes which individual patterns are present in the combination, but makes the mistake of identifying a third pattern. If we generate these three-pattern combinations, as demonstrated in the bottom row of Figure 20, they differ very little from the two-pattern combinations (1 pixel difference). The ABAM's answers in these two cases are therefore not very far from the truth.

The ABAM's ability to combine two patterns has been discussed here in more detail for two reasons: Firstly, the ability to recognize that a piece of information is a combination of other pieces of information is, of course, an important capability of the human brain. Therefore if the ABAM can do this too, we have reproduced an important aspect of

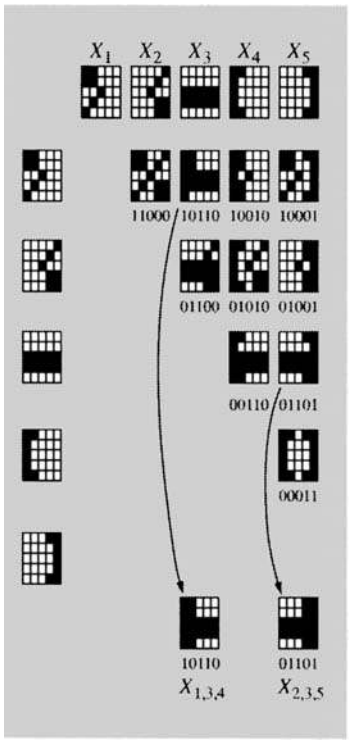


Fig. 20. All combinations of two pictures from Figure 19. The two cases where three bits were activated erroneously are compared with the combination of three pictures drawn in the bottom row (see text for details).

biological, neural information processing. Secondly, the ability to combine single pieces of information is required in many problem tasks in chemistry, especially in the study of relationships between structure and spectral data: If a neural network learns this relationship from pairs of spectra and structures, it should be able to derive all the component substructures from a new spectrum, even if it has never "seen" this particular combination of substructures previously.

4.3. The Kohonen Network

4.3.1. Principles

T. Kohonen[22, 23] developed a neural network which of all models has the greatest similarity to its biological counterpart.[24] This is especially true of the way in which the brain processes sensory signals.

There is a broad strip of tissue in the cerebral cortex which specializes in the perception of touch and is known as the somatosensory cortex. Particular areas in this tissue are responsible for particular parts of the body; parts of the body that carry the most sensory receptors are assigned correspondingly large, contiguous areas, whereas areas of skin which are supplied with relatively few sensory nerves are assigned only small areas, even if the part of the body in question is actually very large by comparison. In addition, neighboring parts of the body are assigned neighboring regions in the somatosensory cortex, so that for the sense of touch there is a contorted mapping of the body surface onto the brain (Fig. 21).

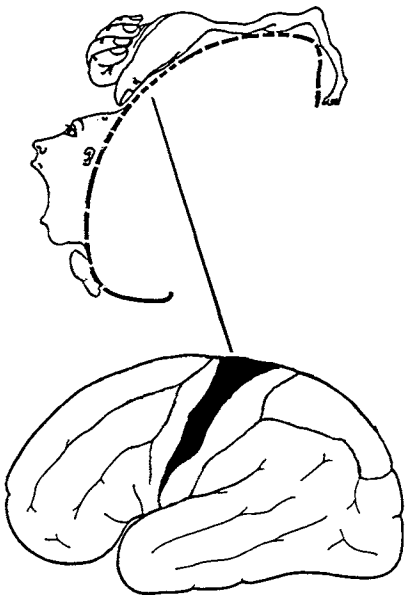


Fig. 21. Map of the human body (top) in the somatosensory cortex of the brain (bottom).

Kohonen introduced the concept of a "self-organized topological feature map" that is able to generate such mappings. Briefly, these are two-dimensional arrays of neurons that reflect as well as possible the topology of information, that is, the relationships between individual pieces of data and not their magnitude. Using the Kohonen model we are able to create a *mapping* of multidimensional information onto a layer of neurons, that preserves the essential content of the information (relationships). The whole process therefore represents a kind of abstraction. Figure 22 shows the two-dimensional organization of the neurons in a Kohonen net.

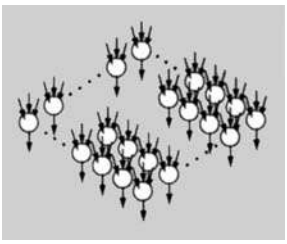


Fig. 22. Two-dimensional assignment of the neurons in a Kohonen network.

In this context mapping of information means that the similarity of a pair of signals is expressed in the proximity or "neighborhood relation" of the neurons activated by them: the more alike two signals are, the closer together the neurons they activate should lie. However, as we are talking here about topological and not Euclidean distance, a neuron in a square array (Fig. 23a) has eight neighbors in the first "sphere", since the neuron has eight direct neighbors. In a Kohonen network where the neurons are arrayed in a square the spheres of neighborhood grow through the network as shown in Figure 23b.

We must take the discussion on the topology of the Kohonen net a little further before we can proceed to the learning algorithm because the topology is the decisive factor in a Kohonen net. If every neuron is to have the same number of neighbors, a square, planar array is poorly suited to the task because the neurons at the edges have fewer neighbors than those in the center of the net (Fig. 23c).

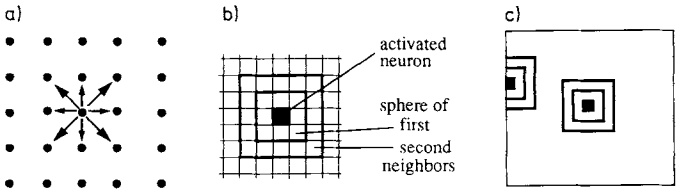


Fig. 23. Proximity relations of the neurons in a Kohonen network. a) First sphere of neighbors; b) growth of the spheres; c) neurons at the edge of the network.

From a square or rectangular array of elements we can nevertheless easily construct a topology in which each element has exactly the same number of neighbors. In order to do this we must wrap the surface around on itself and connect ("glue together") its opposite edges. Thus from the surface we produce first a cylinder and then a torus, as shown in Figure 24.

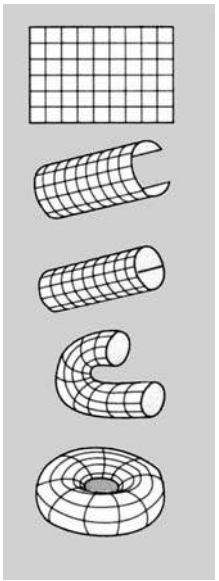


Fig. 24. The transformation of a rectangular array of neurons into a torus in which each neuron has the same number of neighbors.

In a torus each element has the same number of neighbors: eight in the first circle, 16 in the second etc. Of course it is not easy to display in its entirety a mapping onto a torus. We shall therefore continue to represent Kohonen networks as

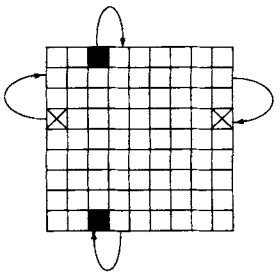


Fig. 25. Representation of the surface of a torus on a plane. The topmost edge wraps onto the bottom, and the left edge wraps onto the right.

flat surfaces in the knowledge that if one arrives at one of the edges the surface carries on over to the opposite edge (Fig. 25). Both the filled-in squares are therefore immediate neighbors. The same is true in the horizontal axis for those fields marked by crosses.

The topology of the Kohonen net has been discussed a little more fully here in order to understand the examples better (see Sections 10.1 and 10.2).

4.3.2. The Learning Process

Learning in a Kohonen net is a competitive process: all the neurons compete to be stimulated by the input signal. The input signal is an object that is described by m single values and can therefore be interpreted as a point in an m-dimensional space, projected onto a plane. Only one single neuron is finally chosen as the "best" ("winner takes all"). Different criteria are applied to find the "best" neuron—the central neuron c. Very often the neuron whose weights are most similar to the input signal $\boldsymbol{X}_k$ is chosen [Eq. (m)].

$$out_c \leftarrow \min \left\{ \sum_{i=1}^{m} (x_{ki} - w_{ji})^2 \right\} \qquad \text{(m)}$$

For this central neuron c, the weights w_{jc} are corrected so that the output value becomes even more similar to the input signal. The weights on the other neurons are also corrected, albeit proportionally less the further they are from the strongly stimulated (central) neuron. Here it is the topological distance which counts, that is, which sphere of neighborhood (as reckoned outwards from the central neuron) includes the neuron under consideration (compare Fig. 23).

Then the process is repeated with the next input data. Every object (that is, every point in the m-dimensional space) stimulates a very specific neuron, so that every object is assigned a definite point in the Kohonen net.

A simple example, the mapping of the surface of a sphere onto a Kohonen net, will explain the workings and the results of the Kohonen net in more detail (Fig. 26). The spherical surface was divided into eight sectors, as shown in Figure 26b; a point on this surface is defined by its three coordinates (x, y, z values). An array of $15 \times 15 = 225$ neurons was used to make a Kohonen net, which therefore had three input units distributing their data over 225 neurons. Thus a total of $3 \times 225 = 675$ weights had to be determined.

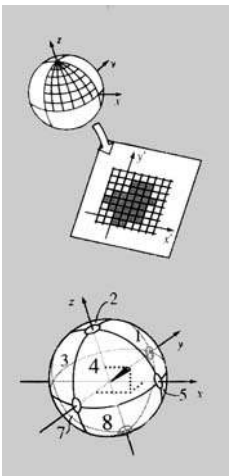


Fig. 26. The representation of the surface of a sphere in a Kohonen network. The sphere's surface is divided into eight sectors. Each point on the surface of the sphere is characterized by its assignment to one of these sectors. The point marked in the sphere at the bottom belongs to sector number 4.

Two thousand points were chosen at random from the spherical surface, and their sets of x, y, and z coordinates used severally to train the Kohonen net. For the graphical representation of the net that had developed after the learning of these 2000 points, each point was identified by the number of the spherical sector from whence it came. (This information was not used in the learning process, however, but only at the end for identifying the points.) Because 2000 points have to be mapped onto 225 fields there are several points mapped onto each field. As it turns out, however, at the end of the learning process only points from the same sector of the sphere arrive at the same field, and points from neighboring regions of the spherical surface are projected onto neighboring fields. Figure 27 shows the resulting Kohonen net.

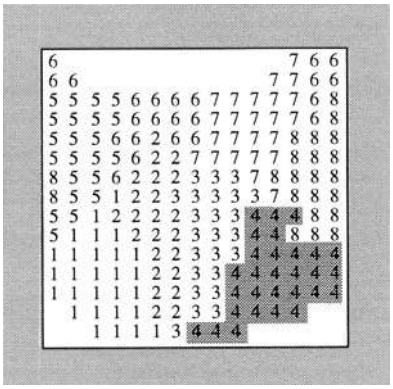


Fig. 27. Result of the projection of a spherical surface onto a Kohonen network.

Fields with the same number, that is, points from the same sector of the sphere, form contiguous areas in the Kohonen net. It must be remembered that the surface shown here actually forms a torus (Fig. 24); therefore points on the left-hand edge are continued over onto the right hand edge (Fig. 25). Thus the two fields on the left-hand edge marked with the number 8 do indeed have a direct connection to the remaining fields with the number 8. Some fields (neurons) in the Kohonen net were not assigned any points from the spherical surface. That the spherical surface was mapped onto the Kohonen net while preserving the neighborhood relations between the points on the sphere may also be seen from the details of the Kohonen net as shown in Figure 28. Sectors on the sphere which meet at lines of longitude or at the equator are also neighbors in their projection on the Kohonen net, and hold whole borders in common. In several regions four fields meet together; these regions or points correspond to the points where the coordinate axes break through the spherical surface. For example at the circle in Figure 28 (bottom row, center) the sectors 1,2,3 and 4 converge. This corresponds to the "North pole" of the sphere.

The mapping of a sphere onto a Kohonen net has been explained in this detail to show how a simple three-dimensional body can be mapped onto a plane. This illustrates well the way a Kohonen net functions as a topology-preserving

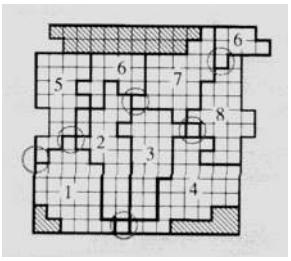


Fig. 28. Areas on the Kohonen map of Figure 27 where four sectors meet, highlighted by circles.

map and lays the foundation for a good understanding of the application of Kohonen networks to examples from the field of chemistry.

4.4. Back-Propagation

The majority of neural net applications uses the "back-propagation" algorithm. This algorithm does not represent any particular kind of network architecture (a multilayered net is generally used) but rather a special learning process. Although this method was not introduced until 1986 by Rumelhart, Hinton, and Williams[17] it quickly gained widespread popularity and contributed decisively in the eventual triumph of neural networks. A few years ago a study carried out on publications about neural networks in chemistry showed that in 90% the back-propagation algorithm was used.[2]

The attraction of learning through back-propagation stems from the fact that adjustments to the neural net's weights can be calculated on the basis of well-defined equations. Nevertheless this procedure for correcting errors has very little in common with those processes responsible for the adjustment of synaptic weights in biological systems.

The back-propagation algorithm may be used for one- or multilayered networks and is a supervised learning process. The input data are passed through the layers; the output data of a layer l, $\boldsymbol{Out}^l$ form the input data $\boldsymbol{X}^{l+1}$ of the layer $l+1$. The results for the input data should eventually be delivered by the final layer. This will, however, not be the case at first. The output data, $\boldsymbol{Out}^{last}$, of the final layer are therefore compared with the expected values $\boldsymbol{Y}$ and the error determined. This error is now used to correct the weights in the output layer. Next the weights in the penultimate layer are corrected with regard to the error from the final layer. Thus the error is fed back layer by layer, from bottom to top, and used to correct the weights at each level (Fig. 29). The error therefore flows counter to the direction of the input data; hence the name back-propagation, or loosely formulated, "error-correction in reverse".[25] In the following sections we demonstrate the basic features of the back-propagation algorithm. For a detailed explanation we refer the reader to the literature.[1, 17, 18, 25]

The back-propagation algorithm is intended to change the weights until the error in the output values $\boldsymbol{Out}$ is minimized; that is, they must correspond as closely as possible to the provided values $\boldsymbol{Y}$.

In the case of the last layer the error can be determined directly because the value $\boldsymbol{Y}$, which is the expected output value, is known. The weight adjustments for the final layer,

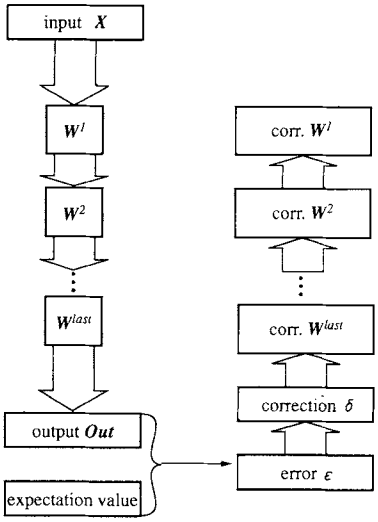


Fig. 29. The learning process of the back-propagation algorithm. The weights are corrected by feeding the errors back into the network.

Δw_{ji}^{last}, are determined by deriving the error ε [Eq. (n)] according to the individual weights. The chain rule leads to Equations (o) and (p).

$$\varepsilon = \sum_{j=1}^{m} (y_j - out_j^{last})^2 \tag{n}$$

$$\delta_j^{last} = -\partial\varepsilon/\partial Net_j^{last} \tag{o}$$

$$\Delta w_{ji}^{last} = \eta\, \delta_j^{last} out_j^{last-1} \tag{p}$$

In the hidden layers the error is not directly known because it is not known what output values, $\boldsymbol{Out}^l$, should be produced by those layers. At this point we make the assumption that the error from the layer below was distributed evenly over all the weights in the layer above. This assumption enables the error in one layer to be calculated from the error in the layer below (which has just been calculated). This is the basis of the back-propagation algorithm: The error is carried back through the individual layers (hence "back-propagation of errors") thus enabling us to correct the weights in each layer.

All in all we derive the closed form (q) for correcting the weights of a single layer. Here η is a parameter, the learning

$$\Delta w_{ji}^{l} = \eta \sum_{k=1}^{r} \delta^{l+1} w_{jk}^{l+1} f'(Net_j^l) out_i^{l+1} \tag{q}$$

rate, that determines how rapidly a neural net learns. Usually a value between 0.1 and 0.9 is chosen for it. Frequently when correcting the weights, the changes in weights from the cycle before (previous) are also taken into account. This is accomplished by extending the Equation (q) by the contribution $\mu \Delta w_{ji}^{l(previous)}$. The parameter μ is the momentum term. It determines the extent to which the previous weight changes are taken into account; it gives the learning process a certain capacity for inertia. The smaller μ is, the quicker previous changes in weights are forgotten. It can be shown that the sum of η and μ should be about 1.[26]

5. Other Neural Network Models

Apart from those mentioned in Section 4, a number of different processes for analyzing data with neural networks are still available. We wish briefly to mention only two more.

The counterpropagation network consists of a Kohonen layer combined with a look-up table in order to give answers.[27]

The *a*ssociative *m*emory *s*ystem (AMS) was developed as a model of information processing in the cerebellum, which is primarily responsible for processing motoric stimuli.[10] The AMS model has been used to predict chemical reactivity data, in particular the polar reactivity of bonds.[28]

6. Overview of Applications in Chemistry

A great many approaches to problems from diverse branches of chemistry have already been investigated by applying neural networks—from very general tasks such as structure-spectrum relationships, chemical reactivity, secondary and tertiary structures of proteins and process monitoring, to such specific questions as the classification of energy levels in the curium atom and the recognition and classification of aerosol particle distributions, as well as the relationship between the physical structure and the mechanical properties of polyethylene–terephthalate fibers.

This diversity underlines the fact that neural networks represent quite general, broadly applicable problem-solving methods. The final, concrete application is determined only by the nature of the data fed into the neural network.

This article is intended to convey some feeling for the kinds of problems best approached with neural networks, so that the reader will be able to decide whether these methods can be used for his particular problem. The goal is therefore to demonstrate the capabilities of the methods and to give incentive to further applications, rather than to give a comprehensive overview of all the work that has gone before. For this reason only a selection of typical neural network applications in chemistry will be given in the following sections; these will not be organized by application area but rather by problem type, that is, according to whether the task in question is one of classification, modeling, association, or mapping.

There is, of course, a whole series of methods that can be used as alternatives to neural networks—and which have been used successfully for many years. Many tasks for which neural networks are used today could be solved equally well by using statistical and pattern-recognition methods such as regression analysis, clustering methods, and principal component analysis. In many cases these methods should be able to deliver results that are every bit as good. What is unfortunately lacking in most papers is a comparison between the capabilities of neural networks and those of more established methods. Comparisons such as these might bring out the advantages of neural networks more. These advantages include the following:

- the mathematical form of the relationship between the input and output data does not need to be provided
- neural networks are also able to represent nonlinear relationships.

On the other hand, the application of neural networks requires exactly the same care in the formulation of the problem, the representation of information, the selection of data, and the division of data into training and test sets as is necessary for pattern-recognition and statistical methods. It must be stated clearly that the quality of the results obtained from neural networks depends crucially on the work and trouble invested in these subtasks.

Essential aspects must be considered before deploying any kind of neural network.

- What is the nature of the task under consideration? Classification, modeling, association, or mapping?
- Which kind of learning process should be chosen? Supervised or unsupervised learning? Is there a set of expected results available for the objects, or must the structure of the information first be found?

In Table 1 the various methods are correlated with problem types and learning processes to make it easier to decide which neural network method should be used for a specific task.

Table 1. Possibilities for applying neural networks.

	Hopfield network	ABAM	Kohonen network	Back-propagation
classification		x	x	x
modeling				x
association	x	x		x
mapping			x	
learning process [a]	unsup.	unsup. + sup.	unsup.	sup.

[a] unsup. = unsupervised learning, sup. = supervised learning (see Section 4).

In chemistry most problems are of the classification or modeling types. This is one of the reasons why multilayered networks trained by the back-propagation algorithm predominate. The results of a survey of those neural network applications in chemistry that appeared before the end of 1990 show that over 90% of studies used the back-propagation algorithm.[2] Hopfield networks were used twice, an adaptive bidirectional associative memory only once, and a Kohonen net coupled with the counterpropagation algorithm likewise only once. This distribution need not hold true in future, however, since ABAM and Kohonen networks as well as the counterpropagation algorithm all offer possibilities that have by no means been exhausted.

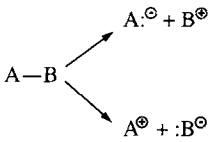
If we consider only the multilayered model with the back-propagation algorithm, we still have a broad spectrum for the complexities of the networks used: from networks with only 20 weights up to some with 40000, in one case even 500000 weights! The number of data used for training a net should, in the case of the back-propagation algorithm, be at least as large as the number of weights. This rule was by no means always followed. The number of times the training data-set was passed through the net (number of epochs) also varied considerably: from 20 up to 100000 iterations. As may be imagined, the training times increased greatly in proportion with the number of weights and the number of iterations, from a few minutes on a personal computer up to hours on a Cray supercomputer.

7. Classification

Classification problems are one of the most common areas of application for neural networks. An object is first characterized by various measurements and on that basis assigned to a specific category. Alternatively it may be found not to have a particular characteristic or not to belong to the class in question. Thus the output data are binary in nature: a feature is either present or it is not; the object either belongs to a certain class or it does not. The input data that characterize the object can be binary in nature but can also be real values (measured data). Classification is a traditional domain for the use of statistical and pattern-recognition methods. Neural networks have the advantage that they can still be applied in cases where the relationships between the object data and the classes to which they are to be assigned are highly complex. Neural networks are suitable even for relationships that cannot or can only barely be expressed in terms of explicit equations. In the following sections we show examples from several branches of chemistry where either a single condition is to be diagnosed (Section 7.1) or a single category is to be chosen from a series of others (Section 7.5), or alternatively where an object is to be assigned to several classes simultaneously (Sections 7.2–7.4).

7.1. Chemical Reactivity

The chemist derives his knowledge about the reactivity of bonds and functional groups from a great many observations of individual reactions. How can this process be transferred to a neural network? Let us look at the problem of polar bond breaking (Scheme 1), the first step in many organic reactions.[29]



Scheme 1. Polar bond cleavage.

The reactivity is characterized very roughly in this case: namely whether a bond is easy or hard to break heterolytically. For this a single output neuron that is set to one if the bond may be easily broken and to zero if the bond is difficult to break will suffice. We still have to characterize the polar bond breaking by parameters that serve to influence the process. For this purpose a number of energetic and electronic effects were used: bond dissociation energy (BDE), difference in total charge Δq_{tot}, difference in π-charge Δq_π, difference in σ-electronegativity $\Delta\chi_\sigma$, σ-polarity Q_σ, polarizability of bonds α_b, and the degree of resonance stabilization $R^\pm$ of the charges that arise from polar bond breaking. Values for these quantities were calculated by empirical methods.[30–34] For these seven parameters, seven units are required into which are fed the (real) values of the individual quantities. A hidden layer of three neurons completes the architecture for this study (Fig. 30).

A data set that consisted of 29 aliphatic compounds containing 385 bonds was created. As each bond may be broken

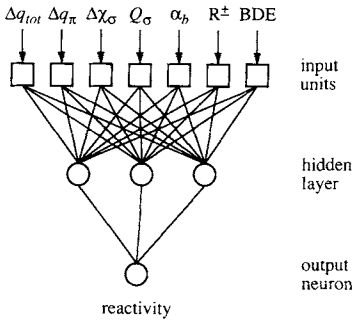


Fig. 30. Architecture and input parameters for a neural network to predict the breaking of bonds. For details see text.

heterolytically in two directions (see Scheme 1), there was a total of 770 possible bond breaks. From these, 149 heterolyses of simple bonds were chosen and 64 of them used to train the net by the back-propagation algorithm; the remaining 85 bond-breaks were used to test the net (the division into training and test data sets will be explained in Section 10.1). Figure 31 shows a selection of molecules from the data set, in which those bonds classified as breakable or unbreakable have been marked.

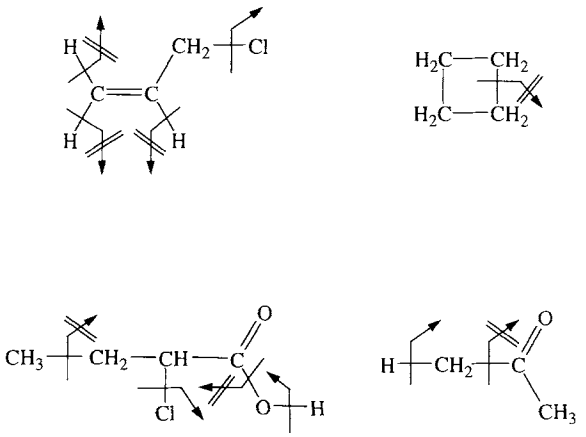


Fig. 31. Selection of structures from the training set indicating which bonds are easy (arrow) or difficult (arrow crossed through) to break heterolytically. The direction of the arrows indicate to which atom the electron pair will be shifted in bond cleavage (that is, which atom will receive the negative charge).

After 1300 cycles (epochs) the net had correctly learned all 64 bond breaks from the training data set. Then the bond breaks from the test data set were passed through the already trained neural net. These 85 bond breaks, about which the net had hitherto received no information, were also classified correctly. The division of the bonds into those easily and those not easily broken was predicted by the net exactly the same way as the chemist had determined them. The net had therefore learned the relationship between electronic and energetic parameters and polar bond-breaking.

The net was then ready to be applied to compounds that were contained neither in the training nor the test data sets. It even made correct predictions about bond types which contained atoms that were not used at all in training. In Figure 32 the predicted reactive bonds of a structure which was not used to train the net are shown. The dissociation of

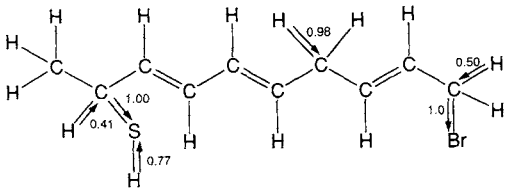


Fig. 32. Bond cleavages predicted by the neural network for a structure which was not used in training. The direction of the arrows indicate the shifts of electron pairs, the values the predicted probability for heterolysis.

a bromide ion and a thiol group, both in allyl position, were found to be especially reactive, as were the removal of a proton from the central allylic position and from the thiol group. The allylic positions at the ends of the system were estimated to be less acidic, whereas the position at which the bromine atom can still function in an inductively stabilizing way receives a higher acidity. Thus all results correspond closely to chemical experience.

It is remarkable that the reactivity of the SH group was correctly judged, even though the training data set did not contain a single structure with a sulfur atom. This is made possible by the use of those electronic and energetic parameters that contain the influence of the atom in general form in the techniques used for calculation of the inputs. Thus even a type of atom which does not occur in the current training set can be taken into consideration, provided that it is contained in the calculation process.

This neural network is able to predict which bonds can easily be broken in a polar manner for a wide range of aliphatic structures.

7.2. Process Control

For many chemical processes the relationship between process data and control parameters can be represented, if at all, only by nonlinear equations. It is therefore very hard to model these processes and to predict their behavior. It therefore comes as no surprise that neural networks are being used intensively for tasks in process control.[35-41] This includes not only the classification of certain events (yes/no decisions), but also the modeling of control parameters (prediction of a real value).

An example in which the task is to choose between membership of different classes will serve to illustrate the possible applications.[36] The goal was to derive individual failure modes from six different sensor values for a continually stirred reaction vessel in which an exothermic reaction was in progress.

The following measurements were taken (cf. Fig. 33): 1) The outlet concentration of the educt C_e, 2) the reactor temperature T_r, 3) the volume of the contents of the reactor V_r, 4) the outlet flowrate FR_p, 5) the temperature of the cooling water T_c, and 6) the flowrate of the cooling water, FR_c. These six parameters were to be used as "symptoms" to diagnose several failure modes of the reactor. Malfunction can be caused by incorrect inlet concentration of the educt C_{eo}, inlet temperature T_e, and inlet flow-rate FR_e. If any of these measurements deviates by more than 5 percent from the normal value, a failure has occurred in the reactor.

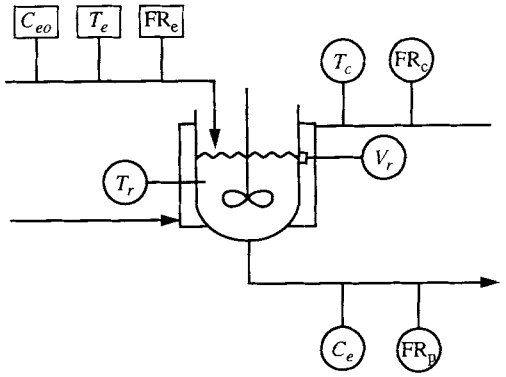


Fig. 33. Diagram of a reactor tank showing the six measured values (C_e, T_r, V_r, FR_p, T_c, FR_c) and the state variables that give rise to a failure mode (C_{eo}, T_e, FR_e).

Each one of the failure modes affects almost all the symptoms, that is, all six measured values, so that a specific failure mode can not be diagnosed directly from a single measurement. Moreover, it is possible for the failure modes to occur not just singly but simultaneously, and for the effects of simultaneous failure modes to compensate in the sensor values, or contrarily to amplify each other synergetically.

Which network architecture was chosen in this case? Because six real measured values are to be input, six input units are required. Likewise the number of output neurons chosen was six; for each of the three crucial inlet parameters (C_{eo}, T_e, and FR_e) one neuron for a deviation over the normal value and one for a deviation below it. Thus if any of the six failure modes occurs the corresponding output neuron should also be activated. Five neurons in a hidden layer complete this multilayered net (Fig. 34).

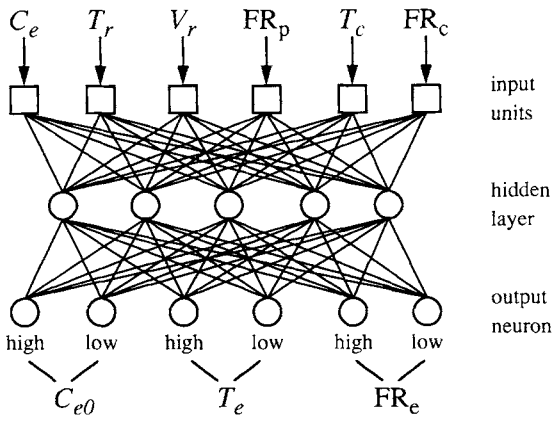


Fig. 34. Neural network for diagnosing failure modes of the chemical reactor depicted in Figure 33.

Accordingly for this neural network $6 \times 5 + 5 \times 6 = 60$ weights were to be determined. Twelve individual failure modes were deliberately produced, and the network trained by using the sensor data measured from them on application of the back-propagation algorithm. The network so trained was able to identify sensor data from the reactor running normally (data which were not used in the training process) as undisturbed behavior. Furthermore, when four multiple

failure modes were produced the neural net was likewise able to derive these correctly from the sensor data taken at the time.

Neural networks will certainly become very important in the area of process control. It should also be possible to take a neural network that has been trained on a specific process, hard-code this onto a chip, and build this chip into the control process.

7.3. The Relationship between Structure and Infrared Spectra

In the examples from Sections 7.1 and 7.2, rather simple neural networks with relatively few weights were used. For the following application a considerably larger network with almost 10000 weights was developed.

Modern structure elucidation is based on spectroscopic methods. However, as the relationships between the structure of an organic compound and its spectroscopic data are too complex to be captured by simple equations, there are a multitude of empirical rules. The enormous amount of spectroscopic data thus available fulfills an important precondition for the training of neural networks. The first steps to store the relationships between structure and spectroscopic data in neural networks have already been taken. Nevertheless, as we shall see, there is still much development to be done in this area.

Munk et al.[42] investigated to what extent conclusions could be drawn by a neural network about the substructures contained in a particular compound from an infrared spectrum. The range of an infrared spectrum between 400 and 3960 cm^{-1} was divided into 256 intervals, and each of these intervals assigned to an input element. If an absorption band was found in such an interval, its intensity was fed into the input element. The neural network had 36 output neurons, each of which was responsible for one of 36 different functional units (primary alcohol, phenol, tertiary amine, ester etc.). If a functional unit was present in the compound under investigation, then the corresponding neuron received the value one, otherwise the value zero. Further, an intermediate layer containing 34 hidden neurons was used, which required $265 \times 34 + 34 \times 36 = 9928$ weights to be determined for this multilayered network. The basic procedure and the architecture of the neural network are sketched in Figure 35.

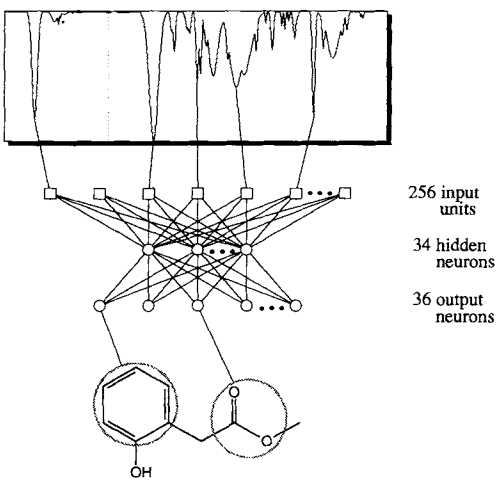


Fig. 35. Neural network to learn the relationships between the infrared spectrum of a compound and the substructures present in it.

To determine the weights of the neural network, 2499 infrared spectra together with their structures, which were broken down into their functional units, were learned through the back-propagation algorithm. Then 416 spectra were used to test the predictive ability of the net. A single cycle through all spectra required 10 min CPU time on a VAX 3500; for a training session with many cycles (called epochs—typically 100 epochs were necessary) a Cray supercomputer was used.

For each functional group the quality of the results was measured by a number, named the A50 value. This value represents the precision at 50% information return, that is, the precision with which a functional group can be ascertained when the threshold value is set to the mean of the distribution curve.

Figure 36 shows a typical result, in this case for primary alcohols. The threshold value here lay at an output value of

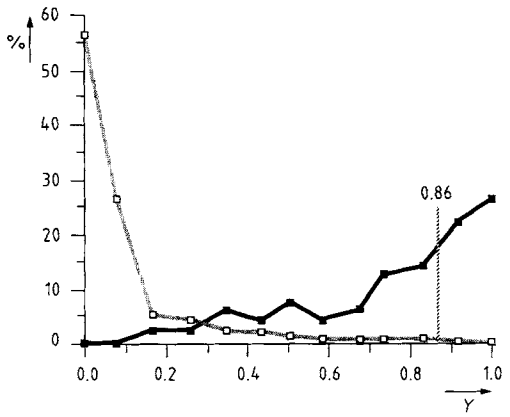


Fig. 36. Percent distribution of the output values Y of the neural network for primary alcohols. The solid line represents compounds which are primary alcohols, the shaded line all other compounds. The mean of the output values for primary alcohols is 0.86.

0.86. At this value 132 of the 265 primary alcohols contained in the training set are correctly identified, but 34 compounds are incorrectly classified as primary alcohols as well. The A50 value for this group is therefore $132/(132 + 34) = 79.5\%$. This value was still deemed to be good. Thirty of the 36 functional groups were able to be predicted with similar or better precision.

The results from this network, which contained a hidden layer, were compared with classification capabilities of a network without a hidden layer.[43] The hidden layer brought about a considerable improvement in results.

This experiment has, of course, not solved the problem of resolving the relationships between infrared spectrum and structure. For the most part, work concentrated on a few important functional groups, and the molecular skeleton was ignored. Even for those groups the predictions were only of moderate quality; to recognize 132 out of 265 primary alcohols in addition to 34 false assignments is disappointing. If one sets the threshold value even higher one can determine to a high degree of certainty whether a functional group is

present or absent, but a large range of compounds remains for which no reliable predictions can be made. In an automatic structure–elucidation system, however, these kinds of predictions allow a considerable reduction of the search space. Herein lies the value of these results.

This is not the last word on the relationship between structure and infrared data. Further experiments should attempt to classify vibrations of the skeleton. This would, however, necessitate a change in the way structures are coded.

7.4. The Relationship between Structure and Mass Spectra

The relationships between mass spectra and structure are even more complex than those between infrared spectra and structure. Nevertheless, this problem also has already been approached with neural networks.

In this case, too, a multilayered network containing a hidden layer was trained with the back-propagation algorithm.[44] The mass spectra were described by 493 features; these included the logarithms of the intensity of the peaks between m/z 40 and 219, the logarithms of the neutral losses between m/z 0 and 179, autocorrelation sums, modulo-14 values, etc. The values for these 493 spectral characteristics were fed into the same number of input units.

Here too the structure of an organic compound was characterized by 36 substructures which, however, differed partly from those used in the study on infrared spectra. Thirty-six output neurons were needed for this. As the number of neurons in the hidden layer was 80, $493 \times 80 + 80 \times 36 = 42320$ weights had to be determined.

Correspondingly larger data sets were investigated: with 31926 mass spectra for training and 12671 mass spectra for testing. With such large data sets and a network with so many weights, the learning process (the back-propagation algorithm was also used here) required a great deal of time. One epoch (that is the process of passing all 32000 spectra once through the net) needed 6 h on a HP 9000/370 or SUN-4 workstation. Typically 50 epochs were needed; thus training alone required two weeks computing time on a high-performance workstation.

The results of the classification from the fully trained neural network, MSnet, were compared with results from STIRS.[45] STIRS, from the group led by McLafferty, is a powerful expert system for determining the presence of functional groups from mass spectra.

The classification results from MSnet were somewhat better than those from STIRS. MSnet offers a few additional advantages however: 1) A probability value can be given for the assignment of a compound to a particular class. 2) Not only the presence but also the absence of a functional group can be diagnosed. 3) The computing time required for queries to MSnet is two orders of magnitude lower than for STIRS.

The following point must be emphasized: Even though the training of a neural network may require a great deal of CPU time, a fully trained neural network can make predictions in minimal time.

In order to satisfy as general a requirement as representing or learning the relationship between molecular structure and spectroscopic data for the entire domain of organic chemistry, we must confront a fundamental problem, the statistical distribution of data. For example, the data set of 32000 compounds contains 33 phthalic acid esters, which give a very characteristic peak at m/z 149. However, most of the spectra which have a peak at m/z 149 happen not to be phthalic acid esters because there are only very few of them in the total set. As a consequence phthalic acid esters are not recognized.

In this paper[44] an interesting attempt is made to overcome this general problem: a hierarchy of neural networks is proposed (Fig. 37). A preliminary network first undertakes

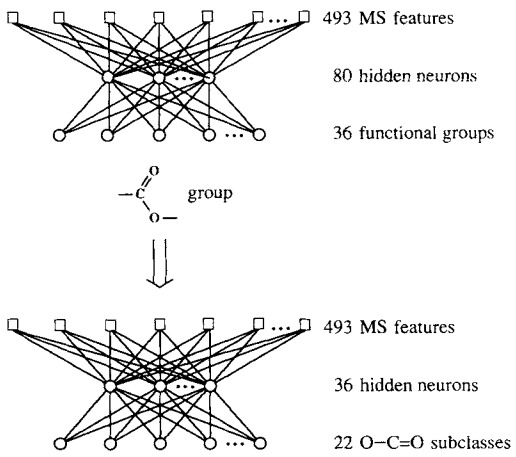


Fig. 37. Hierarchy of neural networks for deriving substructures from mass spectra.

the partitioning according to the most important functional groups, while specialized neural networks carry out further refinements of the compound classes. Thus a special network was developed which divided compounds containing the O–C=O group into 22 subclasses (saturated esters, aromatic esters, lactones, anhydrides etc.).

This idea of using a hierarchy of neural networks could prove very useful in other problem areas.

7.5. Secondary Structure of Proteins

In the example from the previous section we were still dealing with a rather simple network architecture. We now move on to describe cases with a rather extensive coding of the input data. The neural network involved is correspondingly complex, with a large number of weights.

To gain a deeper insight into the physiological characteristics of proteins one must know their secondary structure. For this reason there has been no scarcity of attempts to derive the secondary structure of proteins from their primary structures (that is, from the sequence of amino acids). Chou and Fasman[46] introduced a method which is much used today to decide from the sequence of amino acids what secondary structure the individual parts of a protein will assume. This procedure is able to predict with 50–53% accuracy for the individual amino acids in a protein whether they take part in an α-helix, β-sheet or an irregular coiled struc-

ture.[47] Over the past few years a series of papers have appeared in quick succession[48-56] on applications of neural networks to predict the secondary or even tertiary structure of particular sections of a protein from the sequence of amino acids. Our intention is to demonstrate the principal procedure based on the work of Qian and Sejnowski.[48] Most of the other investigations[49-56] have chosen a very similar strategy.

Both the Chou and Fasman method[46] and the application of neural networks are based on the assumption that the amino acid (AA) itself and its immediate surroundings (that is, the amino acids immediately before and after it in the sequence) decide which secondary structure it will adopt.

In order to take the dependence of the secondary structure on the sequence into account, the amino acid under examination will be fed into the net along with the six amino acids preceding and the six following it in the sequence. Thus from the amino acid sequence, a "window" of 13 amino acids is extracted every time. This window is pushed in steps of single AAs down the entire length of the amino acid sequence, so that each individual AA in turn may find itself in the center.

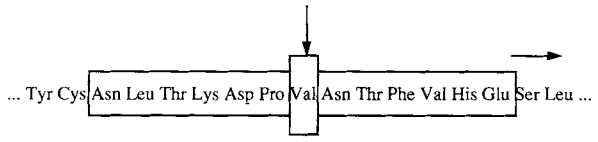


Fig. 38. Section ("window") of 13 amino acids from a protein sequence. The window helps to determine the secondary structure in which the amino acid in question (in this case valine) finds itself.

How are the individual amino acids coded? For each AA in the window of 13 AAs a bit vector of length 21 is used. For each of the 20 naturally occurring amino acids a particular position in the bit vector is reserved (e.g., the 14th bit is set to 1 when the AA proline is present). A last bit is required in order to mark that the window is at the beginning or end of the protein and thus that there are no more AAs at one of its ends. Thirteen input units are needed for the size of the window and 21 for the identity of the AA, thus $13 \times 21 = 273$ input units altogether; each feeds only one bit, a zero or a one, into the network.

The network had three output neurons: one for the presence of an α-helix, one for a β-sheet, and one for a coiled structure. After experimenting with between 0 and 80 hidden neurons, a hidden layer of 40 neurons was chosen as optimal for the net, which meant that $273 \times 40 + 40 \times 3 = 11040$ weights were to be determined. Again in this case the back-propagation learning algorithm was chosen. The overall architecture is pictured in Figure 39.

The network was trained with 106 proteins containing a total of 18105 amino acids. The efficacy of the net was tested with 15 additional proteins, which comprised 3250 AAs in total. A prediction accuracy of 62.7% was achieved.

Thus one could say with 62.7% certainty whether an amino acid was part of an α-helix, a β-sheet or a coiled structure. This is a noticeable improvement over traditional methods for predicting secondary structures but still leaves something to be desired. It is therefore understandable that this area is being researched so actively.[48-56]

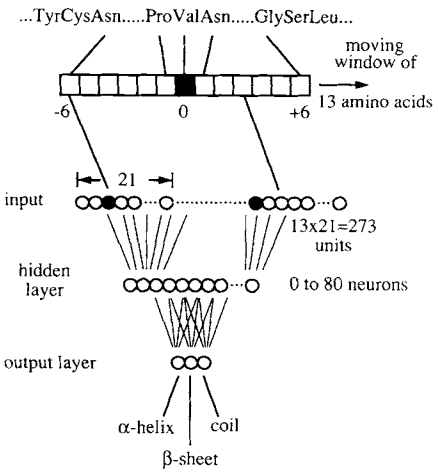


Fig. 39. Neural network for deriving the secondary structure of a protein from the sequence of amino acids.

7.6. Summary

These applications from very different branches of chemistry serve to underline the broad range of possibilities which neural networks offer for classification. In every one of the examples portrayed, a multilayered network was used that was trained by the back-propagation learning process.

The number of neurons in the hidden layer is usually determined by a systematic series of experiments. With too few neurons a problem can not be correctly learned; the larger the number of neurons the smaller the error in learning, but the longer the training times. Too many neurons and too long a training period can lead to another problem: overtraining. This means that a neural net has been trained for so long that, although it can reiterate the training set without error, it may give bad predictions when faced with new data. Multilayered networks usually have many weights, and therefore many degrees of freedom in adapting to a data set. This presents the danger that one might fall into an erroneous local minimum in training, thus reducing the ability to make predictions based on new data.

The complexity of the neural net (mainly the number of weights and the number of training data used) can bring with it a large increase in the training time. One should not be discouraged too much by long training times, however, because ideally a network needs to be trained only once. Once it has finished learning, predictions on new data can be made very rapidly, because these data only have to be passed through the fully trained net once.

In the case of classification problems it is desirable to obtain values of one or zero in the output neurons so that an object belongs unambiguously to a category or not. In reality values between one and zero are obtained, and decisions are made on the basis of threshold values (e.g., 0.8 or 0.2) whether an object belongs to that category. These numeric output values can also be interpreted as probabilities, and as such may be used in a decision support system.

The numerical output clearly points to the transition to modeling problems, which are the subject of the next section. In modeling tasks it is desirable to obtain function values, (i.e., real data). These can be produced by taking those val-

ues between zero and one and transforming them through mathematical functions.

In summary, the most important task when implementing a neural network is to find a suitable representation for the input and output data. The hierarchical ordering of multiple networks (Section 7.4) and the movable window from Section 7.5 demonstrates that the technology sets no limits to the imagination.

8. Modeling

We have already seen that even in the case of classification a neural network returns values between zero and one, that is, a continuum of values. It is also possible, however, to train a neural network with real expectation values and to use the output values in their real magnitudes, just as one might normally calculate the value of a function from a series of variables. This task of using data about an object (compound, reaction, spectrum) and deriving other characteristics of the object from them has commonly come to be called "modeling", and is what we shall use the term to mean in the following sections. The neural network therefore assumes the task of using input variables (data) from an object to find the value of some dependent characteristic (or even several of them). A neural network offers the advantage of not requiring an explicit formulation of the relationship as a mathematical equation. It expresses this relationship implicitly in the connections between the individual neurons.

8.1. HPLC Analysis

A simple example shall serve to represent the potential of this technique in analytical chemistry.

In the HPLC analysis of Spanish wines, the dependence of the separation of the components (expressed as the selectivity factor SF) on the ethanol content (10, 20, or 30%) and the pH value (5.0, 5.5, or 5.6) of the liquid phase was determined. These nine experimental points were fitted to a quadratic equation with standard modeling techniques.[57] The result is given in Equation (r), where x_1 = % ethanol and x_2 = pH value. This functional relationship is also represented in Figure 40 in the form of lines of the same selectivity factor.

$$SF = 0.018\,x_1^2 - 1.42\,x_2^2 + 0.015\,x_1\,x_2 - 0.995\,x_1 + 16.16\,x_2 - 31.86 \qquad (r)$$

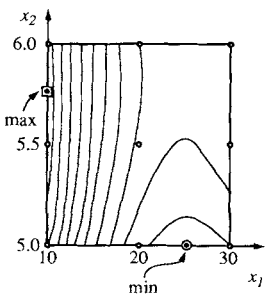


Fig. 40. HPLC analysis of Spanish wines: Shown is the dependence of the selectivity factor SF on the ethanol content x_1 and on the pH value x_2 of the liquid phase. The curved arrows highlight the maximum (left) and minimum SF (bottom).

The same nine experimental data were built into a neural network with two input units (one for the ethanol content and one for the pH value), one output neuron (for the selectivity factor), and six neurons in a hidden layer, which was trained with the back-propagation algorithm.[58]

Values for the ethanol content and the pH were fed into the network and the results entered into the diagram in Figure 41. Here too, just as in Figure 40, lines of the same selectivity factor were drawn in. A comparison between Figures 40 and 41 shows that both standard modeling and neural network techniques arrive at quite similar results. In particular, the positions for the minimum and the maximum values of the selectivity factor are very similar.

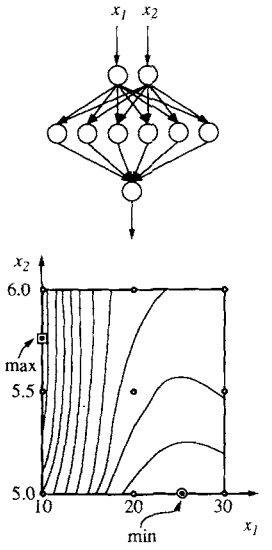


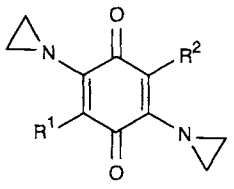
Fig. 41. The results obtained from a neural network (shown at the top) on the selectivity of the HPLC analysis of Spanish wines: dependence of SF on the ethanol content and on the pH value. See also Figure 40.

The advantage of the neural network is clear: in statistical modeling the mathematical form of the functional relationship (here a quadratic equation) had to be given explicitly. With the neural network this is not necessary; it finds the relationship by itself, implicitly, by assigning appropriate weights.

8.2. Quantitative Structure–Activity Relationships (QSAR)

The search for quantitative structure–activity relationships (QSARs) is one of the most important application areas for modeling techniques. A great deal of trouble and effort has been invested, especially in the prediction of pharmacological and biological data. It is therefore all the more surprising that as yet very few studies have been published which employ neural networks to provide quantitative relationships between structure and biological activity.[59, 60] A typical study shall briefly be described here.

For this study a data set that had already been investigated with statistical modeling techniques (a multilinear regression analysis) was deliberately chosen in order to compare the performance of a neural network against a standard method from the field of QSAR. The data set comprised 39 *para*-quinones, some of them anticarcinogenic (Scheme 2).



Scheme 2. *para*-Benzoquinones containing two aziridine substituents, a class of compounds which includes some anticarcinogenic members. R^1 and R^2 are, for instance, CH_3, C_6H_5, etc.

The influence of the substituents R^1 and R^2 was described by six physicochemical parameters: the contribution of R^1 or of both substituents to the molar refractivity index, MR_1 and $MR_{1,2}$, their contribution to the hydrophobicity, π_1 and $\pi_{1,2}$, the substituent constants of the field effect (F), and the resonance effect (R). Accordingly for a description of the influences of the substituents, six input units were required (Fig. 42).

The neural network was intended to yield the minimum effective dosage of the medication for a single injection. This minimum effective dosage is given by the amount of substance (as $\lg 1/c$) that leads to a 40% increase in length of life. A single neuron was present in order to output the value of $\lg 1/c$. A hidden layer of 12 neurons completed the network architecture (Fig. 42).

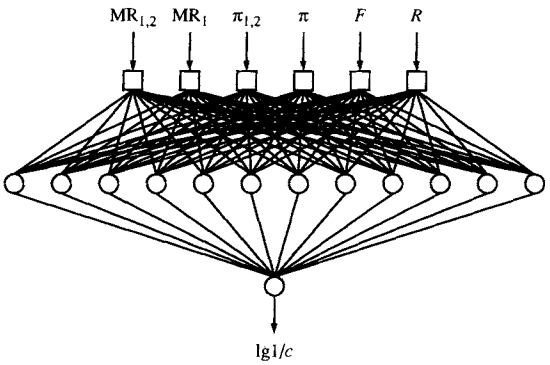


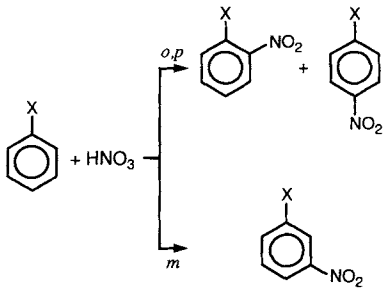
Fig. 42. Neural network for predicting the anticarcinogenic activity of *para*-benzoquinones.

Thirty-five benzoquinones were used to train the multilayered network with the back-propagation algorithm. The values for $\lg 1/c$ obtained from the network were compared with the results calculated from an equation determined by multilinear regression analysis. In 17 cases the results from the neural network were better, for 6 more or less as good, and for 12 worse. In other words, the results from the neural network are significantly better. Nevertheless this problem can be solved quite well by a linear method that leaves little room for improvement by a neural network. In the case of QSAR problems containing nonlinear relationships more might be gained by using neural networks.

8.3. Chemical Reactivity

Whereas in Section 7.1 we were satisfied with a yes/no decision on chemical reactivity (does a bond break easily in a polar manner or not), here we wish to make a quantitative statement about the behavior of a chemical reaction.

The electrophilic aromatic substitution of monosubstituted benzene derivatives can lead in principle to three isomers: *ortho*, *meta*, and *para* products (Scheme 3). The dependence

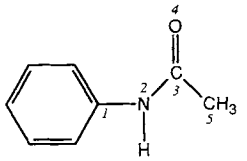


Scheme 3. Isomeric distribution in electrophilic aromatic substitution.

of the isomer distribution on the nature of the substituents X is a classical problem in organic chemistry. Basically the substituents may be divided into two classes: electron-donating substituents (inductive or mesomeric), which prefer to direct into *o*- and *p*-positions, and mesomeric electron acceptors which steer towards the *m*-position. The factors that determine the *o*/*p* ratio are both steric and electrostatic in nature.

In one study[61] into the product ratios for the nitration of monosubstituted benzene derivatives, the amounts of *ortho* and *para* products were combined. Accordingly, one output neuron was used for the $o + p$ content and a second for the percentage of *m*-isomer. As already mentioned, the distribution of product is determined by the nature of the substituent, especially by the electronic effects which it produces. In order to represent this, two codings of the input information were tested. In the first attempt the partial atomic charges on the six carbon atoms of the benzene ring as they are calculated in the semi-empirical quantum mechanical program suite MOPAC[62] according to Mulliken population analysis were used as the six inputs. In addition an intermediate layer with 10 hidden neurons brought the total number of weights to be determined to $6 \times 10 + 10 \times 2 = 80$.

As an alternative method the structure of the substituent was represented directly in the form of a connection table. This had dimensions of 5×5; each row contained first the atomic number of the atom in question then the index of the atom itself followed by the index of the adjacent atom further away from the ring, the order of this bond, and the formal charge on the atom. For each atom of the substituent except a hydrogen atom a new row was used. The first row reflects the situation at the atom which is directly bonded to the ring. With each successive row a further step is taken into the substituent (cf. Fig. 43). If the substituent had fewer than five non-hydrogen atoms then the rest of the 25 entries were filled with zeros. If it had more than five heavier atoms, those atoms which were further away from the point of attachment of the substituent on the ring (atom 1) were left out. Figure 43 explains this coding by connection table for the example of acetanilide. For this coding $5 \times 5 = 25$ input units were required, the intermediate layer had five neurons, and the network thus $25 \times 5 + 5 \times 2 = 135$ weights.



atomic number	relevant bond: reference atom	adjacent atom	bond order	charge
7	2	1	1	0
6	3	2	1	0
8	4	3	2	0
6	5	3	1	0
0	0	0	0	0

Fig. 43. Example illustrating the representation of monosubstituted benzene derivatives by a connection table of the substituent.

The network was trained with 32 monosubstituted benzene derivatives and the back-propagation algorithm; it was then tested with 13 further benzene derivatives. In this example the enormous number of 100000 epochs or training cycles was necessary before the error in the training data set had been reduced to a sufficient degree.

Of the two forms of coding with their networks, the second version—that is, the input of the substituent by means of a connection table—clearly produced the better results. Only the percentages of the *meta* products (Table 2) are necessary

Table 2. Results of attempts at predicting the amount of *meta*-product in the nitration of monosubstituted benzene derivatives. The size of the error in the prediction is quoted [%].

Method	Training data (32 compounds)	Test data (13 compounds)
network based on charge	5.2	19.8
network based on connection table	0.3	12.1
CAMEO	18.0	22.6
estimate by chemists	–	14.7

to judge the quality of the results. The training data set was able to be learned down to an average error of 0.3% for the *m*-isomer content. For the network obtained from the charge values the average error was 5.2%. With the test data of 13 compounds that the network had not seen before, the connection table coding gave an average error of 12.1% in predicting the percentage of *m*-product. For the atomic charge representation the error was noticeably higher at 19.8%

The results from these two neural networks were compared to values produced by CAMEO,[63] an expert system for predicting reactions. They were in every respect better than those produced by CAMEO. Finally the 13 monosubstituted benzene derivatives were given to three organic chemists to predict the expected percentage of *m*-product from the nitration. The values they gave were averaged; this gave an error of 14.7%. The chemists were thus better than the neural network with charge-coding and better than CAMEO, but were beaten by the neural network with the connection-table input!

Though the results for predicting the product ratios from the nitration of monosubstituted benzene derivatives with a neural network based on the coding of the substituents by a 5×5 connection table are very encouraging, they do deserve more detailed comment. First, it is not surprising that the coding by the partial charges on the six ring atoms was rather unconvincing. Apart from the known inadequacies of Mulliken population analysis, the ground-state charge distribution is only one of the factors (not even the most important electronic effect) that influences the product ratios in electrophilic aromatic substitution. For these reasons the charge values can not represent the benzene derivatives sufficiently well to explain the product ratios from nitration.

As satisfactory as its results may be, the coding of the benzene derivatives by a 5×5 connection table can not provide an explanation for the effects responsible for regioselectivity, nor for the influence of the reaction conditions. Nor can it explain the product ratios from di- and polysubstituted benzene derivatives. To do this one must choose a different representation for the benzene compounds, which is indeed feasible.[64] The substituent should not be described globally. Rather its influence on the individual positions on the aromatic ring should be represented directly: for each position on the ring a value for the resonance effect, the local electrostatic potential, and the steric effect should be given. Because this represents the influence of a substituent at every single position on the ring, we can extend inferences from monosubstituted benzene derivatives to di- and polysubstituted compounds. Then we can make predictions about isomer distributions in further substitution. In addition, it is also possible to take account of the influence of the medium by providing a further input unit for the concentration of sulfuric acid. The network is trained with this alongside the descriptors for the influences of the substituents.[64]

8.4. Summary

Modeling tasks are very common in many branches of chemistry. This opens up a wide area for the application of neural networks. At present this area is dominated almost exclusively by multilayered networks and the back-propagation algorithm. This need not be the case. Other neural network architectures, especially the counterpropagation algorithm,[27] can certainly be used just as well for creating models.

The deployment of neural networks for modeling (that is, for predicting some characteristic of an object from a series of other parameters or measurements from that object) should always be considered carefully in relation to the use of statistical methods. If one has a relatively clear idea which variables influence the feature sought, and if there is some largely linear relationship between them, then traditional methods, such as multilinear regression analysis clearly offer advantages: these methods are faster and require less computing time, they give measures for the quality of the discovered relationship, and, above all, the equation derived from the statistical modeling allows a clear interpretation of the individual effects of the feature sought.

Neural networks should be used, however, if it is presumed that nonlinear relationships exist between the dependent and the independent variables, and if it is not possible to specify exactly which parameters influence the characteristic under investigation.

Whether the choice falls on statistical methods or neural networks, the success of a study depends crucially on the selection of the data set, on the representation of the information, and on the methods employed to validate the results. Moreover, when implementing neural networks, the following points are of crucial importance:

- the selection of a homogenous data set for training (e.g., by using experimental design techniques or a Kohonen network – see Section 10.1)
- the partitioning of the data set into training and test data sets
- the selection of suitable parameters as input data to describe the objects.

9. Association

The capacity to associate is closely related to the recognition of similarities between different objects. Neural networks can also be conceived of as memories, because they store the information they have learned from the training data in the form of weights or "synaptic strengths". Should a new object be fed into the neural network whose input data resemble those of an object used in training, some network architectures are able to recognize this and yield the stored object from the training phase as output.

This ability of neural networks to associate, to recognize similarities in given data, has been exploited very little in chemistry up to now. Even the two studies presented here are essentially only feasibility studies. The basic solution to a problem is demonstrated here on small, simply structured data sets. For a real-world application however, larger sets of more complex data would have to be investigated.

9.1. Adjustment of Base Lines

Systematic deviations from the base line may be observed in many spectra and could be caused by impurities, the influence of solvents, or problems with technical equipment. Five different types of base line (normal, ascending, descending, concave, convex) were stored in a Hopfield network as simple black pixel-patterns in a matrix consisting of 6×20 points.[65] Then a simple, simulated spectrum which contained a convex base line was fed into the thus trained Hopfield network. After three iterations the pattern for this baseline was output by the network (Fig. 44).

Afterwards therefore, a background correction could be undertaken for this spectrum. We have already pointed out in this review that a real-world application would need a far higher resolution; thus the base lines would have to be represented by a larger pixel matrix (e.g., 20×250). However, because of the basic limitations on the storage capacity of Hopfield networks, and in order to make good predictions, 2×10^7 matrix elements would need to be checked. It only makes sense to carry out this task by hardware implementations of parallel neural networks.

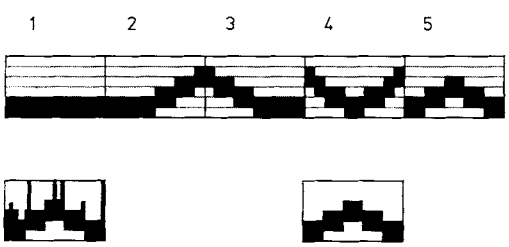


Fig. 44. Five types of base lines of a spectrum (top), a much simplified spectrum with a concave baseline (bottom left) and the concave base line recognized by the neural network after three iterations (bottom right).

Because of the storage and calculation problems presented by Hopfield networks, the same task was also investigated with a Hamming network.[65] A Hamming network,[66] which requires far less memory capacity, was in fact also able to be solve the problem. However, in this case too, only a 7×20 matrix was used to represent the five types of base line; an application which uses experimental data with a high resolution has yet to appear.

9.2. Identification of Spectra

Another feasibility study was made into how slightly altered UV spectra (e.g., through the influence of solvents) might be recognized.[67] Prototypes of UV spectra were represented by a sequence of points (pixels) in two 10×10 fields; the first field containing the band maxima was placed at the input of an adaptive bidirectional associative memory (ABAM), and the second field, comprising the tailing portions, was placed at the output (Fig. 45a). Because of the bidirectional nature of the ABAM the terms input and output are of course purely arbitrary.

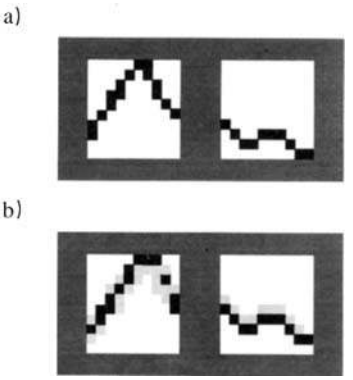


Fig. 45. Grid representation of an ultraviolet spectrum (a) and fuzzy coded forms of this spectrum (gray dotted fields) (b).

The ABAM was trained with five model UV spectra coded in this way. For testing fuzzy codings and slightly altered spectra were used (cf. Fig. 45 b). The recall capability of the ABAM was heavily dependent on the choice of various network parameters; however, in the end a set of parameters was found that was able to recall all five spectra learned from the input of its distorted counterpart.

10. Mapping

Many chemical phenomena are influenced simultaneously by a whole series of factors and depend on many parameters. These parameters can be seen as coordinates in a multidimensional space. Individual observations then represent

points in this multidimensional space. To visualize the structure of this multivariate information in order to make visible the essential relationships between the individual data points, the dimensionality of the space has to be reduced to such a degree that it can be represented graphically and thus analyzed by the human eye. Such a mapping, say onto a two-dimensional plane, should preserve the essential relationships between the individual data points as far as possible.

In Section 4.3 we saw how a three-dimensional space, the surface of a sphere, could be mapped by a Kohonen network onto the surface of a torus, and the neighborhood relationships between the points on the sphere were kept intact. In the following sections we present two examples for the mapping of multidimensional chemical information by a Kohonen network.

10.1. Chemical Reactivity

We wish once more to examine the data set introduced in Section 7.1, which classified a series of single bonds in aliphatic molecules according to whether they were easy or difficult to break heterolytically[29] (cf. Figure 31). Each of the bonds was characterized by seven electronic and energetic parameters, such as difference in charge and polarizability, and bond dissociation energy (cf. Figure 30). The rupture of a bond is thus represented by a point in a seven-dimensional space.

The question now is, are the points with reactive bonds in this seven-dimensional space separated from those of the nonreactive bonds? Moreover, if this is the case, can this separation be retained in a map onto a two-dimensional surface by means of a Kohonen network?

A Kohonen network consisting of 11×11 neurons was trained with the 149 bonds; the values of the seven electronic and energetic factors were used as input data (see Fig. 30).

The results are shown in Figure 46. Indeed the reactive bonds do find themselves predominantly in certain neurons, and the nonreactive bonds in different neurons. Additionally the neurons of the reactive bonds form a contiguous part of the Kohonen network.[29]

It may be concluded from this that the chosen parameters characterize the heterolysis of a bond well (because reactive and nonreactive bonds are separate) and that a Kohonen network is able to preserve this separation even when mapping it onto a surface. It should be emphasized once again that the Kohonen network learns without supervision, and therefore that the information on the reactivity of a bond was not used during the learning process.

Yet another conclusion may be drawn from Figure 46. Bond cleavages that activate the same neuron ought also to contain more or less the same reactivity information; the interaction of the seven parameters ought to produce a similar net effect in reactivity. Accordingly, if the relationships between reactivity and the electronic and energetic parameters are to be investigated further, it is sufficient to choose a single bond out of the many that activate the same neuron. At the same time, at least one bond should be chosen from each occupied neuron in order to cover as much as possible of the reactivity spectrum. In fact the data set that was learned by the back-propagation algorithm (see Section 7.1) was chosen with these considerations in mind. Thus two different neural network models were used: a Kohonen network for the selection of the data set and a multilayered neural network, trained by the back-propagation algorithm, for the task of classification.[29]

The Kohonen method can therefore be used to make balanced selections of data sets for investigations with statistical or pattern-recognition methods, or even used with other neural networks.

10.2. Electrostatic Potential

The electrostatic potential surrounding a molecule has a decisive influence on many physical, chemical, and biological properties. Electrostatic potentials are analyzed in detail, especially in investigations into the interactions between substrate and receptor, but also in studies into chemical reactivity.

If one moves a probe charge, for example a positive unit charge in the form of a point, around a molecule, for every point in space the electrostatic potential may be determined by quantum mechanical or classical electrostatic methods.

Scheme 4 shows a three-dimensional molecular model of 3-chloro-3-methylbutan-1-ol, and Figure 47 the electrostatic potential that a positive point charge experiences on the van der Waals surface of this molecule. The magnitude of the

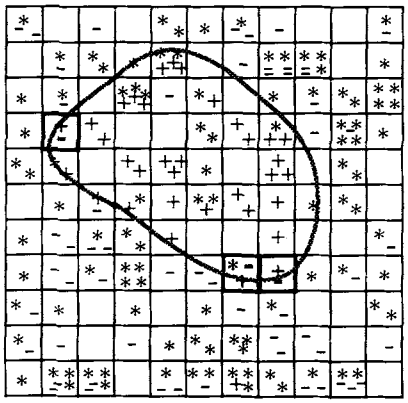
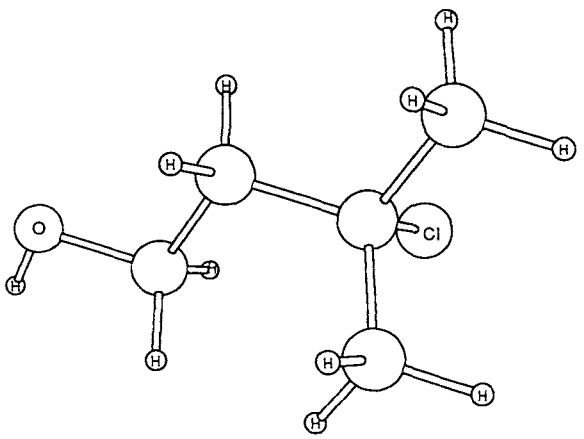


Fig. 46. A Kohonen network that maps polar bond rupture characterized by seven electronic and energetic parameters. + indicates a reactive bond, − a nonreactive bond, and * a bond whose reactivity was not decided.



Scheme 4. Molecular model of 3-chloro-3-methylbutan-1-ol.

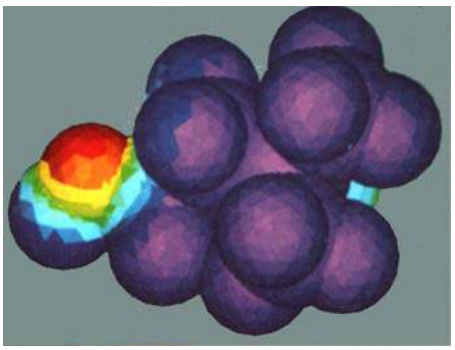


Fig. 47. Electrostatic potential on the van der Waals surface of 3-chloro-3-methylbutan-1-ol. Red regions possess a negative potential, and thus attract a positive charge; blue and violet regions repel this charge.

electrostatic potential is translated into a color code: Strongly negative values of the electrostatic potential (positions to which a positive charge is attracted—that is, nucleophilic positions) are represented by red. Strongly positive values (positions from which a positive charge is repelled—that is, electrophilic positions) are marked in blue or violet. The intermediate values are represented by continuous blends of color. The electrostatic potential was calculated in the classical way by summing the Coulomb interactions of the probe charge with the atomic charges as calculated by the PEOE procedure.[30, 31]

Figure 47 represents a parallel projection of the distribution of the electrostatic potential on the van der Waals surface of the molecule onto the plane of the graphics screen. Of course, only that part of the electrostatic potential that happens to be visible to the observer can be shown. Thus for example, the chlorine atom can barely be seen in Figure 47. A complete view of the distribution of the electrostatic potential can only be obtained by taking a series of such mappings from different observation points. The more complex the form of the van der Waals surface and the variation in electrostatic potential, the more mappings are required. For this reason it will be very difficult to obtain a complete impression of the distribution of the electrostatic potential and the relationships between all electrophilic and nucleophilic centers.

The inadequacies of a parallel projection onto a plane (monitor screen) led us to look for other projection methods that are capable of representing the essential aspects of the potentials on a molecular surface in a single mapping.[68] A Kohonen network is one such projection method.

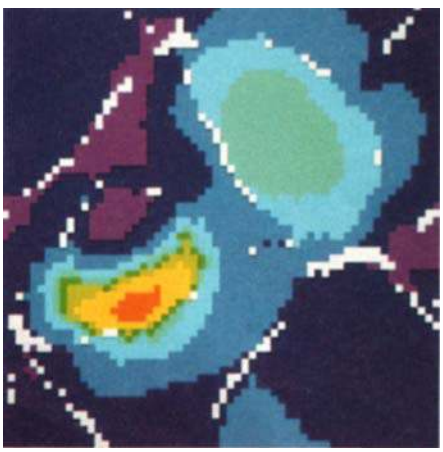


Fig. 48. Kohonen network of the electrostatic potential in Figure 47.

Figure 48 shows the projection of the electrostatic potential of 3-chloro-3-methylbutan-1-ol from Figure 47 onto a Kohonen network. To obtain this mapping, 20000 points were chosen at random from the molecular surface and a Kohonen network of 60×60 neurons was trained with the x, y, and z coordinates of each point. After each point had been passed through the network the neurons were examined to see which points were assigned to which neurons.

Points of the same or similar potential values were indeed found to be located in the same or neighboring neurons. The electrostatic potential on the van der Waals surface of a molecule had therefore been mapped onto the Kohonen map, and the neighborhood relationships on the van der Waals surface were largely preserved intact.

We remind the reader once more that in the case of a Kohonen network the mapping takes place onto a torus, that is onto a surface without beginning or end (cf. Figure 24). The map in Figure 47 can therefore be shifted up, down, to the left or to the right.

A comparison between Kohonen maps of electrostatic potentials of different molecules reflects the essential similarities of the electronic properties on the surface of the molecules, that is, at the places where molecules come into direct contact with their environment. The electrostatic potential on the surface of a molecule is a decisive factor in the interaction of a substrate with a biological receptor. Kohonen maps of electrostatic potentials of different molecules that are bound to the same receptor should therefore indicate certain similarities.

Indeed, it could be shown that Kohonen maps of the electrostatic potential of compounds that bind to the muscarinic receptor have common characteristics. Similarities are also perceived in the Kohonen maps of electrostatic potentials of molecules that bind to the nicotinic receptor, but these characteristics are different from those for molecules bound to the muscarinic receptor.[68]

Kohonen maps show in a single picture essential characteristics of electrostatic potentials. These characteristics are apparently responsible for the binding of a substrate to a biological receptor.

11. Summary and Outlook

Although the development of computational models of information processing in the human brain can look back on a history of almost 50 years, it is only in the last four to five years that neural networks have been applied widely to problem solving. New, efficient algorithms have paved the way for a multitude of applications, and the use of neural networks is still increasing—also in the field of chemistry—by leaps and bounds.

The models of neural networks and their potential applications of classification, modeling, association, and mapping are as diverse as the capabilities of the human brain are varied. The potential in neural networks for the processing of chemical information is very far from being exhausted.

In chemistry the task is often to assign objects to certain categories or to predict the characteristics of objects. This

accounts for the dominance of the back-propagation algorithm. A whole series of other neural network models exists, however, which could be applied successfully to the field of chemistry. This should be explored more widely in future.

Undoubtedly, many of the problems that have been approached with neural networks could also have been solved with statistical or pattern-recognition methods. Neural networks, however, offer capacities which exceed the possibilities of traditional methods of data analysis. Of special importance is the fact that the relationship between input and output data need not be specified in mathematical form, but is derived from the data themselves and represented implicitly. This enables the modeling even of nonlinear relationships.

The use of neural networks still requires much experimentation; guidelines to arrive as quickly as possible at a viable solution to a problem become apparent only gradually. Of crucial importance to the successful application of a neural network is the strategy for data representation; the better the chemical information to be examined is coded, the easier and better the problem may be solved.

The cooperation between our two working teams depended crucially on support from the Bundesministerium für Forschung und Technologie of the Federal Republic of Germany and from the Slovenian Research Ministry. Particularly the provision of a guest-professorship for Jure Zupan at the Technical University of Munich within the framework of BMFT project 08G 3226 has enabled us to develop a productive scientific exchange. We would like to thank especially our co-workers A. Fröhlich, P. Hofmann, X. Li, J. Sadowski, K.-P. Schulz, V. Simon, and M. Novic, who undertook with us the first steps into the Terra Nova of applying neural networks to chemical problems.

Received: July 20, 1992 [A 899 IE]
German version: *Angew. Chem.* **1993**, *105*, 510
Translated by Dr. P. Brittain, Garching (FRG)

[1] A textbook introduction to neural networks will appear shortly: J. Zupan, J. Gasteiger, *Neural networks for Chemists: A Textbook*, VCH, Weinheim, **1993**.
[2] For a detailed overview of neural network applications in Chemistry up to 1990 see J. Zupan, J. Gasteiger, *Anal. Chim. Acta* **1991**, *248*, 1–30.
[3] K. J. Skinner, *Chem. Eng. News* **1991**, *69* (40), 24–41.
[4] A highly recommended collection of publications reflecting milestones in the development of neural networks, each with a short commentary, may be found in *Neurocomputing: Foundations of Research* (Eds: J. A. Anderson, E. Rosenfeld), MIT Press, Cambridge, MA, USA, **1988**.
[5] W. S. McCulloch, W. Pitts, *Bull. Math. Biophys.* **1943**, *5*, 115–133.
[6] W. Pitts, W. S. McCulloch, *Bull. Math. Biophys.* **1947**, *9*, 127–147.
[7] D. O. Hebb, *The Organization of Behaviour*, Wiley, New York, **1949**.
[8] F. Rosenblatt, *Psych. Rev.* **1958**, *65*, 386–408.
[9] M. Minsky, S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, USA, **1969**.
[10] J. S. Albus, *Brain, Behavior and Robotics*, BYTE Books, Peterborough, USA, **1981**.
[11] S.-I. Amari, *Biol. Cybernetics* **1977**, *26*, 175–185.
[12] S. Grossberg, *Biol. Cybernetics* **1976**, *23*, 121–134.
[13] T. Kohonen, *IEEE Trans. Comput.* **1972**, *C-21*, 353–359.
[14] C. von der Malsburg, *Kybernetik* **1973**, *14*, 85–100.
[15] B. Widrow, M. E. Hoff, *IRE ESCON Conv. Rec.* **1960**, 96–104.
[16] J. J. Hopfield, *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558.
[17] D. E. Rumelhart, G. E. Hinton, R. J. Williams in *Parallel Distributed Processing, Vol. 1* (Eds.: D. E. Rumelhart, J. L. McClelland, and the PDP Research Group), MIT Press, Cambridge, MA, USA, **1986**, pp. 318–362.
[18] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Nature* **1986**, *333*, 533–536.
[19] P. J. Werbos, Dissertation, Applied Mathematics, Harvard University, **1974**.
[20] *Parallel Distributed Processing, Vol. 1 + 2* (Ed.: D. E. Rumelhart, J. L. McClelland and the PDP Research Group), MIT Press, Cambridge, MA, USA, **1986**.
[21] B. Kosko, *Appl. Opt.* **1987**, *26*, 4947–4960.
[22] T. Kohonen, *Biol. Cybernetics* **1982**, *43*, 59–69.
[23] T. Kohonen, *Self-Organization and Associative Memory*, Springer, Berlin, **1988**.
[24] H. Ritter, T. Martinetz, K. Schulten, *Neuronale Netze – Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*, Addison-Wesley, Bonn, **1990**.
[25] W. Kirchner, *c't* **1990**, *11*, 248.
[26] X. Li, J. Gasteiger, J. Zupan, unpublished results.
[27] R. Hecht-Nielsen, *Appl. Opt.* **1987**, *26*, 4979.
[28] K.-P. Schulz, P. Hofmann, J. Gasteiger in *Software-Entwicklung in der Chemie 2* (Eds.: J. Gasteiger), Springer, Heidelberg, **1988**, pp. 181–196.
[29] V. Simon, J. Gasteiger, J. Zupan, *J. Am. Chem. Soc.*, submitted.
[30] J. Gasteiger, M. Marsili, *Tetrahedron* **1980**, *36*, 3219–3228.
[31] J. Gasteiger, H. Saller, *Angew. Chem.* **1985**, *97*, 699–701; *Angew. Chem. Int. Ed. Engl.* **1985**, *24*, 687–689.
[32] M. G. Hutchings, J. Gasteiger, *Tetrahedron Lett.* **1983**, *24*, 2541–2544.
[33] J. Gasteiger, M. G. Hutchings, *J. Chem. Soc. Perkin Trans. 2* **1984**, 559–564.
[34] J. Gasteiger, M. Marsili, M. G. Hutchings, H. Saller, P. Löw, P. Röse, K. Rafeiner, *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 467–476.
[35] J. C. Hoskins, D. M. Himmelblau, *Comput. Chem. Eng.* **1988**, *12*, 881–890.
[36] V. Venkatasubramanian, R. Vaidyanathan, Y. Yamamoto, *Comput. Chem. Eng.* **1990**, *14*, 699–712.
[37] K. Watanabe, I. Matsuura, M. Abe, M. Kubota, D. M. Himmelblau, *AIChE J.* **1989**, *35*, 1803–1812.
[38] P. Bhagat, *Chem. Eng. Prog.* **1990**, *86*(8), 55.
[39] V. Venkatasubramanian, K. Chan, *AIChE J.* **1989**, *35*, 1993–2002.
[40] L. H. Ungar, B. A. Powell, S. N. Kamens, *Comput. Chem. Eng.* **1990**, *14*, 561–572.
[41] N. Bhat, T. J. McAvoy, *Comput. Chem. Eng.* **1990**, *14*, 573–583.
[42] M. E. Munk, M. S. Madison, E. W. Robb, *Mikrochim. Acta* **1991**, II, 505–514.
[43] E. W. Robb, M. E. Munk, *Mikrochim. Acta* **1990**, I, 131–155.
[44] B. Curry, D. E. Rumelhart, *Tetrahedron Comput. Methodol.* **1990**, *3*, 213–238.
[45] F. W. McLafferty, D. B. Stauffer, *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 245–252.
[46] P. Y. Chou, G. D. Fasman, *Biochemistry* **1974**, *13*, 222–241.
[47] W. Kabsch, C. Sander, *FEBS Lett.* **1983**, *155*, 179–182.
[48] N. Qian, T. J. Sejnowski, *J. Mol. Biol.* **1988**, *202*, 865–884.
[49] L. H. Holley, M. Karplus, *Proc. Natl. Acad. Sci. USA* **1989**, *86*, 152–156.
[50] D. G. Kneller, F. E. Cohen, R. Langridge, *J. Mol. Biol.* **1990**, *214*, 171–182.
[51] H. Bohr, J. Bohr, S. Brunak, R. M. J. Cotterill, B. Lautrup, L. Norskov, O. H. Olsen, S. B. Petersen, *FEBS Lett.* **1988**, *241*, 223–228.
[52] H. Bohr, J. Bohr, S. Brunak, R. M. J. Cotterill, F. Fredholm, B. Lautrup, O. H. Olsen, S. B. Petersen, *FEBS Lett.* **1990**, *261*, 43–46.
[53] H. Andreassen, H. Bohr, J. Bohr, S. Brunak, T. Bugge, R. M. J. Cotterill, C. Jacobsen, P. Kusk, B. Lautrup, *J. Acquired Immune Defic. Syndr.* **1990**, *3*, 615.
[54] J. D. Bryngelson, J. J. Hopfield, S. N. Sothhard, *Tetrahedron Comput. Methodol.* **1990**, *3*, 129–141.
[55] M. S. Friedrichs, P. G. Wolynes, *Tetrahedron Comput. Methodol.* **1990**, *3*, 175–190.
[56] G. L. Wilcox, M. Poliac, M. N. Liebman, *Tetrahedron Comput. Methodol.* **1990**, *3*, 191–211.
[57] J. Zupan, F. X. Rius, *Anal. Chim. Acta* **1990**, *239*, 311–315.
[58] M. Tusar, F. X. Rius, J. Zupan, *Mitteilungsblatt der GDCh-Fachgruppe „Chemie – Information – Computer (CIC)"* **1991**, *19*, 72–84.
[59] T. Aoyama, Y. Suzuki, H. Ichikawa, *J. Med. Chem.* **1990**, *33*, 905–908.
[60] T. Aoyama, Y. Suzuki, H. Ichikawa, *J. Med. Chem.* **1990**, *33*, 2583–2590.
[61] D. W. Elrod, G. M. Maggiora, R. G. Trenary, *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 477–484.
[62] MOPAC, A General Molecular Orbital Program Package V4.0, QCPE.
[63] A. J. Gushurst, W. L. Jorgensen, *J. Org. Chem.* **1988**, *53*, 3397–3408.
[64] A. Fröhlich, J. Gasteiger, unpublished results.
[65] M. Tusar, J. Zupan in *Software Development in Chemistry 4* (Ed.: J. Gasteiger), Springer, Berlin, **1990**, p. 363–376.
[66] R. P. Lippmann, *IEEE ASSP Magazine* **1987**, *April*, 4–22.
[67] M. Otto, U. Hörchner in *Software Development in Chemistry 4* (Ed.: J. Gasteiger), Springer, Berlin, **1990**, p. 377–384.
[68] J. Gasteiger, X. Li in *Software Development in Chemistry 7* (Ed.: D. Ziessow], Springer, Berlin, **1993**; X. Li, J. Sadowski, C. Rudolph, J. Gasteiger, unpublished results.