Part IV
Applications

9 General Comments on Chemical Applications

learning objectives:

- the types of problems to which neural networks can be applied
- multivariate (multiple input) and multiresponse (multiple output) systems
- how to choose between supervised and unsupervised learning
- classification of objects into categories and hierarchies of categories
- why modeling is simpler with neural nets than with classical techniques ... and how that very simplicity can be a disadvantage
- how mapping can be used in chemistry
- how neural nets can be used in process feedback and control systems
- the crucial role of data representation in neural network applications
- what is a moving window approach
- overview of the examples presented in Chapters 10 to 20

9.1 Introduction

In chemistry, as in all natural sciences, we would like to learn new methods that can improve, shorten, or bring new insight into old ways of handling experimental data. Part IV will show how different neural network architectures and learning strategies can be applied to some of the problems encountered in the everyday practice in chemistry.

While we can't show all possible applications, we will discuss certain **types** of problems that can be dealt with by neural networks.

The neural network approach is basically a method for handling multivariate and multiresponse data. *Multivariate* data are used to describe **one** object with **several** variables: for example, analysis of air samples for the pollutants NO_x, SO_2 and CO would comprise three-variate data.

If such multicomponent data are studied with respect to several factors (responses) such a problem is generally also described as *multiresponse* (Figure 9-1).

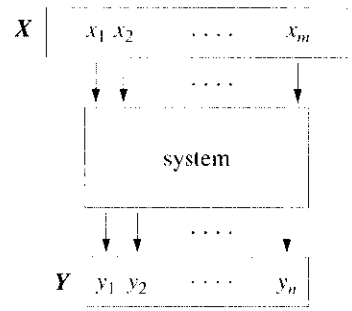

Figure 9-1: Multivariate input $\boldsymbol{X}$ producing multiresponse output $\boldsymbol{Y}$.

The relation between a multivariate object $\boldsymbol{X}_s = (x_{s1}, x_{s2}, ..., x_{sm})$ and the putative factors $\boldsymbol{Y}_s$ $(y_{s1}, y_{s2}, ..., y_{sn})$ can be written in the following way:

$$(y_{s1}, y_{s2}, ..., y_{sn}) = \boldsymbol{A}\,(x_{s1}, x_{s2}, ..., x_{sm}) \qquad (9.1)$$

Here, $\boldsymbol{A}$ is a $(m \times n)$-variate matrix that linearly transforms vector $\boldsymbol{X}_s$ into vector $\boldsymbol{Y}_s$. The index s, according to the convention of Section 1.4, indicates different samples: $s = 1, 2, ..., r$.

Equation (9.1) is the linear (and therefore simplest) version of the multivariate multiresponse problem ($\boldsymbol{A}$ can actually be a very complex operator).

In many applications, $\boldsymbol{A}$ is not known; it may be that all you have is a set of carefully collected m-variate data $\{\boldsymbol{X}_s\}$ accompanied by a set of n-variate responses $\{\boldsymbol{Y}_s\}$. And in some applications, even the responses $\{\boldsymbol{Y}_s\}$ are not known and have still to be figured out. From such sets of carefully measured multivariate data, conclusions are sought that are valid for unknown samples.

The key to finding these answers is that the sought information is already hidden in the multivariate data. For example, an infrared spectrum measured with a resolution of 1 cm^{-1} between 4000 and 200 cm^{-1} comprises 3800 intensity points; these data contain the complete information about the structure of the compound – say, which 15 groups make up the molecule, out of a pool of 200 possibilities. The essence of this 3800-variate 200-response problem is to find out how to confirm or exclude any of the 200 functional groups based on each of the 3800-point spectra (Figure 9-2).

Problems of this type are usually not solved by *ab initio* (theoretical) calculations; instead, they require various statistical, pattern recognition, and, as we want to show, neural network methods.

As has been stressed previously, learning can be either supervised or unsupervised. In supervised learning, the system must adapt itself to yield the known correct answers to all query objects in the training set, while unsupervised learning just maps the objects according to some internal criterion, such as similarity, into the "virtual" area defined by the architecture of the network.

The choice of a supervised or an unsupervised approach depends on the problem and the data available to solve it. In both cases, objects with known answers are needed. In supervised learning, the answers are directly used to influence the learning system, i.e., to calculate changes in the weights of the neural network; in unsupervised learning, the answers are needed only to identify and label the output neurons.

The basic question to consider when deciding on the learning method is: do you want to force the system to adapt itself to an already selected representation of objects and classes, or do you want to keep your options open? An unsupervised neural network method is more flexible due to its many possible outputs. Thus, unsupervised learning can be used as a screening step, allowing you to inspect the behavior of the "response space". After enough knowledge is accumulated, you might be able to switch to supervised learning.

For supervised learning, the multivariate objects should be divided into three sets:

- a training set
- a control set for determining when to stop training (see overtraining, Section 8.6.3), and
- a test set for checking the achieved predictive ability.

For unsupervised learning, we don't need a control set, since learning has to continue until the network stabilizes. (This is achieved in a Kohonen network when the neighborhood of the corrections shrinks to a single neuron.)

The next decisive factor for the choice of the appropriate learning strategy is the number of available objects. The number of objects within reach for all three sets of data (training, control, and test) is very relevant to the efficiency of the method. A supervised learning procedure can take tens or even hundreds of thousands of epochs (an epoch is one pass through all the objects in the input set). Furthermore, for each object, the corrections of thousands of weights might be required.

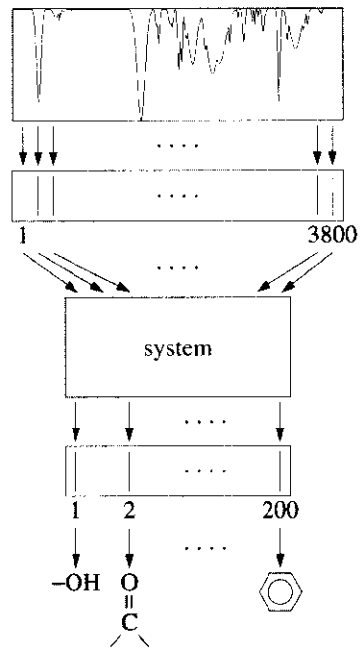

Figure 9-2: A 3800-variate spectrum is input to the system, and a 200-variate output points to specific chemical fragments.

Hence, the efficiency of the learning process is influenced by the number of variables for each object (which defines the size of the input layer), the number of weights in the hidden layers, and the number of objects.

9.2 Classification

One of the simplest and most frequently used operations in handling complex multivariate data is *classification*, which may sort objects into a simple set of categories, or into a hierarchy of sub-classes within classes (Figure 9.3).

A one-level classification is made, for example, in finding the type of secondary structure of a protein: whether a particular amino acid is in an α-helix, β-sheet, or coil structure. Even such a simple three-category classification can be handled by a hierarchical classification scheme: first, whether there is any defined structure around the particular amino acid, and second, if a structure is confirmed, to decide whether the structure is an α-helix or β-sheet (Figure 9-3) (see Chapter 17).

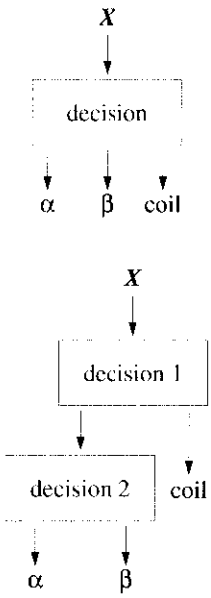

Figure 9-3: One-level and hierarchical classification.

More often, hierarchical classification involves many levels of decisions, classifying an object into the proper group, sub-group, category, sub-category, or class: for example, the prediction of different structural features in an unknown compound on the basis of its spectrum, or the selection of the proper chromatographic method for a given analysis.

In many chemical applications, the object belongs to several **different classes simultaneously**, so that the result of the classification is a product of two or more decisions.

The spectrum-structure correlation problem is a typical multidecision classification: the "object" is the spectrum, while the output is a list of fragments (classes) present in the structure that produced the spectrum. Figure 9-4 shows the hierarchy for deciding which of the five structural fragments A, B, C, D, and E are present in a structure represented by its spectrum. On the first level, a four-category decision is made: whether either, both or neither of the fragments A and B are present in the structure ("∅" = $\overline{A} \wedge \overline{B}$ = neither; "AB" = $A \wedge B$ = A and B).

For each of theses decisions, an analogous second-level decision is made regarding the fragments C and D, while at the final level a two-category decision about E is made.

Figure 9-4 shows one of many possible decision hierarchies that can be set up to solve the problem.

Usually, when faced with the task of setting up a classification hierarchy for a complex decision, you should begin with a statistical study of the relationship between the representation of objects and their class membership. You should consult the literature on this matter (Section 9.8: Massart et al., *Chemometrics*, or Zupan, *Algorithms for Chemists*).

For our purpose, it is enough to say that setting up the hierarchy of decisions has to be done **before** the learning procedure is started; if the final results are not satisfactory, it is advisable to look at the representation of the objects and/or the way the decision hierarchy was set up, change one or both, and try the entire procedure again.

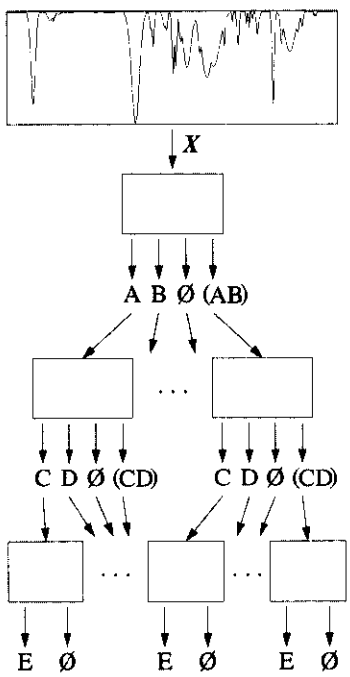

Figure 9-4: Hierarchical classification of an infrared spectrum according to five classes of functional groups, A, B, C, D, and E. A means that only group A is present, in (AB) both groups are present, in ∅ both groups are absent.

9.3 Modeling

Some classifications systems produce a **discrete** answer, such as yes or no, or an integer identifying the input object with one of several classes. However, *modeling* requires a system that yields a **continuous** answer for each input. In modeling, a relatively small number of data are used to build a model that can give predictions for **all** possible objects. Curve-fitting (Figures 8-13, 8-15, 9-5) falls into this category; in most applications, it is a one-variable/one-response procedure:

$$y = \mathrm{f}(x, a, b, ..., p) \tag{9.2}$$

Here, the function f contains a set of unknown parameters a, b, ..., p (in the case of polynomial models, the number and types of terms are chosen to reproduce the degree of curvature anticipated to be in the data). These parameters have to be determined to minimize the discrepancy between the set of experimentally determined answers $\{y_s^{experimental}\}$ and the set of answers obtained by the model $\{y_s^{model}\}$ at the **same** values of x. The following criterion is often used:

$$\left(y_s^{experimental} - y_s^{model}\right)^2 \rightarrow minimum \tag{9.3}$$

Modeling can also be a multivariate/one-response problem, or even a multivariate (x)/multiresponse (y) problem; in the latter case, we need to find a **different** model for **each** response y_i, i.e., a different function form f_i with a different parameter set.

In any kind of modeling, a tacit assumption is always made:

Small changes of input will cause **small changes in output**.

("Chaotic" systems, in which this requirement is relaxed, require specialized techniques that are beyond the scope of this book.)

By its very nature, modeling always requires supervised learning; therefore two types of neural networks are generally applicable: the back-propagation and the counter-propagation methods. In modeling, the data are always composed of two parts. The first part is the representation of the object $\boldsymbol{X}_s$:

$$\boldsymbol{X}_s = (x_{s1}, x_{s2}, ..., x_{sj}, ..., x_{sm}) \tag{9.4}$$

and the second one is the answer or target, $\boldsymbol{Y}_s$:

$$\boldsymbol{Y}_s = (y_{s1}, y_{s2}, ..., y_{sj}, ..., y_{sn}) \tag{9.5}$$

that is expected from the object $\boldsymbol{X}_s$.

In multiresponse problems, neural networks have a major advantage over classical modeling by analytical functions:

In neural networks the targets $\boldsymbol{Y}_s$ $(y_{s1}, y_{s2}, ..., y_{sj}, y_{sn})$ may be multi-dimensional!

What does this mean? Simply that a **multiresponse** answer can be modeled without bothering with different analytical functions or coefficients (Figures 9-6 and 9-7).

Classical models require a separate predefined analytical function for each response, while neural networks can model a multiresponse vector without any *a priori* knowledge!

Of course, as a wise man said, for every silver lining there's a cloud; in classical modeling, the coefficients of the model function can often be given a physical interpretation (e.g., the virial coefficients of a gas). There is no such thing as the coefficient of a given variable in the neural network approach because **all** variable values are shared among **all** weights (albeit not to the same extent). Great effort has been expended (so far, with little success) to find methods that would

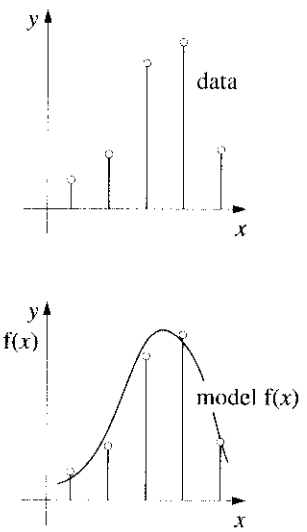

Figure 9-5: One-variable/one-response model.

$$y_1 = f_1(x_1, ..., x_m, a_{11}, a_{12}, ..., a_{1p})$$
$$y_2 = f_2(x_1, ..., x_m, a_{21}, a_{22}, ..., a_{2r})$$
$$\vdots$$
$$y_n = f_n(x_1, ..., x_m, a_{n1}, a_{n2}, ..., a_{ns})$$

The functions
$f_1(\cdot), f_2(\cdot), ..., f_n(\cdot)$
are valid.

All parameters
a_{ij}
must be determined.

Figure 9-6: Classical modeling.

associate specific weights or paths through the network with given variables or combinations of them.

In many applications, especially when the experimental error is comparatively large, we can simplify the above-mentioned proportionality between input and output changes. In such cases, a lookup table (Figure 9-8) is adequate, even though it gives the same answer for a range of inputs.

Modeling is mainly used for objects having a few (one to ten) variables, which produces neural networks much smaller and computationally less demanding than those used for classification problems.

In the selection of a learning strategy, the distribution of the objects within the variable space has to be considered. If the objects for training are few and more or less evenly spaced (some experimental techniques make this possible), then the back-propagation method is suitable. On the other hand, if the objects are plentiful and scattered irregularly through the variable space, they should first be reduced to cover the space evenly and then the counter-propagation method is used. (The reduction of objects can be carried out either by a Kohonen network (Chapter 6, and the examples in Chapters 10 and 11), or with a statistical evaluation of the intervals of the variables.)

Basically, a counter-propagation method used for modeling needs more objects than back-propagation.

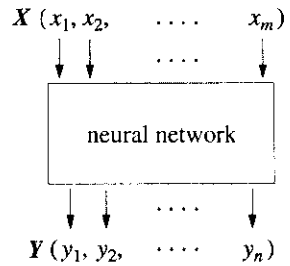

Figure 9-7: Modeling with neural networks.

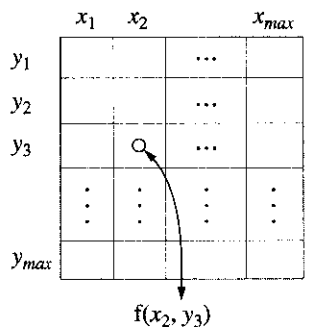

Figure 9-8: A lookup table acting as a model.

9.4 Mapping

Mapping is the transformation of an m-dimensional space into a space of lower (often 2 or 3) dimension (see Section 6.4) in order to display some features that cannot be shown in a higher-dimensional space. All kinds of mappings can be achieved with Kohonen networks.

To some extent, mapping is similar to the clustering or classification of objects. The difference lies not so much in the method itself as in the interpretation of the results. In classification, we want to identify which cluster or class a given object belongs to, while in mapping we focus on the entire map and, in effect, **derive** the clusters from the data.

The essence of mapping with neural networks is this:

> The objects are represented by the **coordinates** (topological position) of the central neuron (Section 6.3) and **not** by the **values** of the output.

The questions that mapping can answer are: what does the map as a whole look like, how many distinguishable areas can be identified on the map, what are the shapes of these areas, how can the map's features be correlated with the objects from which the map was obtained.

A mapping can be either linear or nonlinear, as illustrated in Figure 9-9.

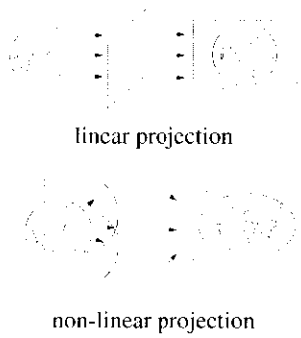

Figure 9-9: Mapping the world globe into a plane.

One of the useful applications of mapping involves a preprocessing method (usually called "experimental design"), by which we usually want to choose the most appropriate objects (experiments) from an available set, for example, selecting spectra for spectrum-structure correlations, or eliminating redundant (or overlapping) combinations of parameters to be used in a series of experiments. In the latter case, we would monitor how often each neuron in the plane is excited by an object (Figure 9-10), e.g., a set of experimental parameters. All but one of the objects that excite or fire the same neuron should be discarded. (This is also useful for selecting the objects to be used for training a back-propagation net; see Section 11.4).

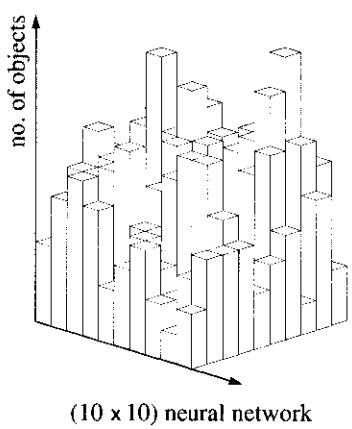

Figure 9-10: Distribution of objects that excite specific neurons on the mapping plane.

9.5 Associations; Moving Window

Among neural network applications not yet mentioned, the following should be used more often in chemistry:

- auto- and hetero-associations
- prediction of time-dependent events

The problem of *association* is to find a target object associated with an input, even if the input is corrupted or incompletely known, for example, when identifying the baseline type in a recorded spectrum, or the type of spectrometer malfunction responsible for a particular bad spectrum. In a sense, then, auto- and hetero-associations can be regarded as classifications.

In both examples, neither the patterns $\boldsymbol{X}$ (spectra) nor the target responses $\boldsymbol{Y}$ (baselines or malfunction types) can be defined exactly. For example, it is easy to see whether the baseline of an infrared spectrum is concave or convex, but hard to describe the difference mathematically. It's even worse in the second example, where the "target" may be anything from a badly prepared sample to drift in the electronic circuits.

Because associations can be learned only through examples, i.e., through input-target pairs, we are restricted to neural networks capable of performing supervised learning.

Investigation of *time-dependent processes* is one of the main topics in process control research (see Section 16.4). Here, the user wants a model that will predict the behavior of a multiresponse system based on a series of data taken over time.

In time-dependent modeling, the input and output variables are basically the same: the only difference is that the input consists of **present and past** values of process variables, while the output predicts the **future** values of the same variables (Figure 9-11, for more details see Chapter 16, particularly Table 16-1). The consecutive sets of variables taken as one input vector and the consecutive sets of future values on the output side are called the *past* and the *future horizons of learning*.

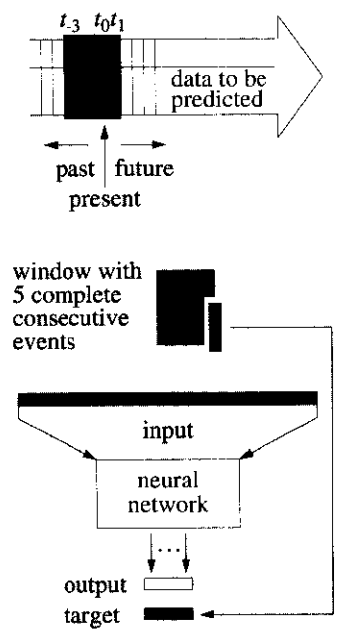

Figure 9-11: Present, past and future data as input/output pairs in time-dependent models.

In principle, these horizons can be of arbitrary length, but in process control the future horizon usually covers only one time step. The *moving window* shown in Figure 9-11 encompasses five consecutive time events; the past horizon is of length four: the input contains all process variables at time t_{-3}, t_{-2}, t_{-1}, and t_0 (the current value), while the process variables beginning with t_1 (in the future horizon) are taken as targets. The "future" data used for training are obtained either from theoretical models or from actual past observations.

Let's construct a vector $\boldsymbol{P}_t$ containing the values at time t of some process variables, say x_1, the flow rate; x_2, the input temperature and x_3, the output temperature:

$$\boldsymbol{P}_t = (x_{1t}, x_{2t}, ..., x_{mt}) \tag{9.6}$$

Then, taking into account **three** consecutive steps in the process, the input vector $\boldsymbol{X}$ looks like this:

$$\boldsymbol{X} = (x_{1t}, x_{2t}, \ldots, x_{mt}, x_{1,t+1}, x_{2,t+1}, \ldots, x_{m,t+1}, x_{1,t+2}, \ldots) \qquad (9.7)$$
$$\boldsymbol{P}_t \qquad \boldsymbol{P}_{t+1} \qquad \boldsymbol{P}_{t+2}$$

The corresponding target vector $\boldsymbol{Y}$ can consist of the fourth vector $\boldsymbol{P}_{t+3}$:

$$\begin{aligned} \boldsymbol{Y} &= (x_{1,t+3}, x_{2,t+3}, \ldots, x_{m,t+3}) \\ \boldsymbol{Y} &= \boldsymbol{P}_{t+3} \end{aligned} \qquad (9.8)$$

For making a **complete model** of a process, **all** process variables must be trained on the output side; but not all variables in the process have equal influence on the final results, and only few are of interest (say, the yield of the product). These significant variables are the ones we choose for training.

The above-described *moving window* procedure is used in many kinds of problems; the variable sets under the window all shift **left** by one position (equivalent to shifting the window **right**), so that the oldest one is moved all the way out (and discarded), and what was the first "future" value becomes the "present" (time = zero) value.

This technique can be used for any problem that deals with a sequence of events or objects, from predicting environment parameters in chemical process plants, to deducing the secondary structure of a protein based on the sequence of amino acids.

We hope that these examples have prepared you to believe that:

> The applicability of neural networks is limited only by your imagination.

9.6 Overview of the Examples in Chapters 10 to 20

The diversity and number of applications of neural networks in chemistry has increased dramatically in the last few years. This is reflected in the number of publications dealing with the use of neural networks to solve chemical problems:

In the rest of the book we will present some of these applications. With such a large number to choose from, we can necessarily give only a fragmentary and personal overview.

year	no. of publications
1988	3
1989	5
1990	20
1991	105
1992	290
1993	441
1994	498
1995	743
1996	855
1997	927

Table 9-1: Number of publications on neural networks in chemistry per year.

In the rest of the book we will present some of these applications. With such a large number to choose from, we can necessarily give only a fragmentary and personal overview.

While, in general, different types of problems require different neural network architectures and different learning strategies, it turns out that 90% of the problems described until now in the chemical literature have used one-hidden-layer neural networks, and the back-propagation learning strategy.

One of our goals is to encourage you to think about the **problem** first and then select the proper neural network, rather than trying to bend all your problems to the one neural network method you happen to be familiar with.

See Table 9-2 for some hints about the kinds of problems that can be dealt with using different neural network learning methods.

Our examples cover the area as broadly as possible and illustrate the diversity of potential applications. For example:

- various chemical disciplines
  - analytical chemistry (Chapters 10, 12, 18),
  - organic chemistry (Chapters 11, 13, 14, 18, 19),
  - pharmaceutical and biochemistry (Chapters 13, 17, 19, 20),
  - chemical engineering and chemical industry (Chapters 15 and 16).

problem \ strategy	back-propagation	counter-propagation	Kohonen network
classification	*	*	*
modeling	*	(*)	
mapping	(*)	(*)	*
association	*	*	
moving window	*	*	

Table 9-2: Neural network learning strategies and their application to different types of problems. An asterisk indicates whether this learning method can be applied to the indicated problem type.

- various types of applications
- different neural network learning methods
- different sizes of neural network architectures and datasets

Table 9-3 summarizes the essential features of the examples in the following Chapters.

chapter	problem	type of problem	size of the network	method
10	origin of olive oils	classification + mapping	medium	BPE + KL
11	bond reactivity	classification + mapping	small	KL + BPE
11	reaction classification	classification	medium	KL
12	HPLC separation	modeling	small	BPE
13	QSAR	modeling	small	BPE
13	QSAR	classification + modeling	medium	KL + BPE
13	QSAR	variable selection + modeling	medium	GA + CP
14	electrophilic aromatic substitution	modeling	small	BPE
15	paint coat recipe	modeling	small	BPE
16	fault detection, process control	classification + modeling	small	BPE + CP + MW
17	protein structure	classification	large	BPE + MW
18	infrared spectrum/structure correlation	classification + mapping	large	BPE + KL
18	infrared spectrum simulation	classification + modeling	large	CP
19	molecular surfaces	mapping	large	KL
20	chemical libraries	classification	large	KL

Table 9-3: Main features of the examples in Chapters 10 – 20; BPE: back-propagation of error, KL: Kohonen learning, CP: counter-propagation, MW: moving window, GA: genetic algorithm.

That's why we have intentionally not arranged the chapters according to the subdisciplines of chemistry. Rather, we want to help you develop an understanding and an eye for the types of problems –

classification, modeling, mapping – that can be tackled by neural networks.

> Before attempting to design a neural network solution to a problem, you should first **classify** the problem and only then attempt to identify the type of neural network and the learning method most appropriate to solve it.

We start with classification problems (Chapters 10 and 11) and then turn our attention to modeling problems (Chapters 12 – 16). However, Chapter 16 also contains a classification problem and classification problems are the main theme again in Chapters 17 and 18 (these were placed so late in the book because they involve such large network architectures and datasets). Chapter 19 deals with a mapping problem, where we will show that **artificial neural networks** can help us understand **biological neural networks**.

Mapping problems are also mentioned in Chapters 10, 11, 18, and 20. This brings up another important point:

> Quite often, a certain application contains aspects of several different types of problems and should therefore be attacked with **several different neural network methods simultaneously**.

In Chapters 10 – 20 we do exactly that – we look at many of the problems using multiple neural network methods: multilayer networks with learning by back-propagation of errors (BPE), counter-propagation network (CP), Kohonen learning (KL), and the moving window input scheme (MW).

A (perhaps the) most important issue in any application of neural networks is the representation of information that is fed into the network and/or obtained from it. Since neural networks are such general-purpose problem-solvers, it is all the more important to be very careful in choosing an appropriate representation of the information that goes into and leaves it. We therefore put strong emphasis on the proper representation of chemical information. The representation chosen has to be adapted to the problem.

As a case in point, the problem of how to represent the structures of organic compounds is addressed – and solved in different ways – in the examples shown in Chapters 11, 13, 14, 17, 18, 19, and 20.

Because of the overall importance of structure representation in many chemical applications we have added a special chapter (Chapter 21) that deals with this problem and collects the various methods that have been used in the different chapters.

Bernard Widrow stressed the importance of proper data representation this way:

"The three most important issues that must be addressed in the development of neural networks are:

1. representation
2. representation
3. representation"

9.7 Essentials

Classification

Either supervised or unsupervised learning may be used in creating a classification engine; in the former, one-hidden-layer neural network architectures with back-propagation learning are used, and Kohonen networks with the latter.

Modeling

Modeling applies only to systems in which changes in the output are proportional to the changes in the input. Modeling always requires supervised learning, and therefore mainly two types of neural networks are applicable: the back-propagation and the counter-propagation methods.

Neural net models are simpler to implement than with classical methods, because of having no need for adjustable parameters; but they also lack the potential for physical interpretation that those parameters possess.

Mapping

The essence of mapping is reduction of dimensionality. The objects are represented by the **coordinates of the position** of the "central" neuron and **not** by the **values of the output**.

9.8 References and Suggested Readings

9-1. D. L. Massart et al., Eds., *Chemometrics Tutorials*, Elsevier, Amsterdam, NL, 1990.

9-2. S. N. Deming and S. L. Morgan, *Experimental Design: A Chemometrics Approach*, Elsevier, Amsterdam, NL, 1987.

9-3. D. L. Massart, B. G. M. Vandeginste, L. M. C. Buydens, S. De Jong, P. J. Lewi and J. Smeyers-Verbeke, *Handbook of Chemometrics and Qualimetrics: Part A*, Elsevier, Amsterdam, NL, 1997.

9-4. B. G. M. Vandeginste, D. L. Massart, L. M. C. Buydens, S. De Jong, P. J. Lewi and J. Smeyers-Verbeke, *Handbook of Chemometrics and Qualimetrics: Part B*, Elsevier, Amsterdam, NL, 1998.

9-5. K. Varmuza and H. Lohninger, "Principal Component Analysis of Chemical Data", in *PCs for Chemists*, Ed.: J. Zupan, Elsevier, Amsterdam, NL, 1990, pp. 43 – 64.

9-6. K. Varmuza, *Pattern Recognition in Chemistry*, Springer-Verlag, Berlin, FRG, 1980.

9-7. J. Zupan and J. Gasteiger, "Neural Networks: A New Method for Solving Chemical Problems or Just a Passing Phase?", *Anal. Chim. Acta* **248** (1991) 1 – 30.

9-8. J. Zupan, *Algorithms for Chemists*, John Wiley, Chichester, UK, 1989.

9-9. J. Gasteiger and J. Zupan, "Neuronale Netze in der Chemie", *Angew. Chem.* **105** (1993) 510 – 536; "Neural Networks in Chemistry", *Angew. Chem. Int. Ed. Engl.* **32** (1993) 503 – 527.

9-10. J. Zupan, "Neural Networks in Chemistry", in *Encyclopedia of Computational Chemistry*, Eds.: P. v. R. Schleyer, N. L. Allinger, T. Clark, J. Gasteiger, P. A. Kollman, H. F. Schaefer III and P. R. Schreiner, Wiley, Chichester, UK, 1998, pp. 1813 – 1827.

9-11. D. J. Livingstone, G. Hesketh and D. Clayworth, "Novel Method for the Display of Multivariate Data Using Neural Networks", *J. Mol. Graphics* **9** (1991) 115 – 118.

9-12. B. Kocjancic and J. Zupan, "Application of a Feed-Forward Artificial Neural Network as a Mapping Device", *J. Chem. Inf. Comput. Sci.* **37** (1997) 985 – 989.

9-13. M. Novic and J. Zupan, "Computer-Aided Identification", in *Encyclopedia of Analytical Science*, Vol. 2, Ed.: A. Townshend, London, Academic Press, UK, 1995, pp. 817 – 825.

9-14. J. Zupan, M. Novic and I. Ruisanchez, "Tutorial: Kohonen and Counterpropagation Artificial Neural Networks in Analytical Chemistry", *Chemom. Intell. Lab. Syst.* **38** (1997) 1 – 23.