12 HPLC Optimization of Wine Analysis

learning objectives:

- an outline of the HPLC method of analysis
- a classical method for modeling (predicting the effectiveness of) the mobile phase of an HPLC column
- modeling the mobile phase with a neural network
- comparison of two different neural nets for this modeling task

12.1 The Problem of Modeling

The identification of individual components in a complex mixture is very common in analytical chemistry; but before the components can be identified and quantified, they first have to be separated. Chromatographic methods presently play a dominant role as separation techniques. High Performance Liquid Chromatography (HPLC) is a widely used method in which the mixture is distributed between a stationary solid or liquid phase on a solid support, and a mobile liquid phase applied at high pressure. The various components are distributed between the two phases to different extents and thus are separated.

Modeling is quite commonly applied in chemistry to a number of areas or problems, particularly to all kinds of optimization from complex procedures composed of many activities, to recipes for given products. To optimize a process or a recipe, or to find the best-fitting function to a number of experimental points, a model has to be found first; after this, the optimization procedure is performed using the response surface of the model as the basis for finding the best solution.

> Without a model, optimization does not lead to a general solution of the problem.

A model tries to describe in mathematical terms the response of a system to appropriate stimuli. Classical modeling is based on a relatively small number of variables – usually fewer than ten – and one response. If **more** responses from the **same set** of variables are required, then a **separate model** must normally be built **for each response** when classical methods are used (see Section 9.3). However, neural networks are able to do both one-response and multiresponse modeling.

As a first example of modeling, we will demonstrate both methods: classical modeling and modeling by neural networks. By comparing all steps in both procedures from the selection of data to the checking of the results, we can appreciate the differences and the advantages and disadvantages of both.

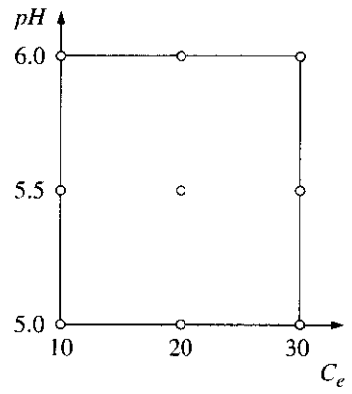

Figure 12-1: A full two-variable three-level experimental (factorial) design scheme. Circles represent properties of nine different liquid phases leading to nine chromatograms.

12.2 Modeling the Mobile Phase for HPLC by a Standard Method

In Chapter 10, we built a system that can identify the geographical origins of more than 500 Italian olive oils, based on analyses for eight fatty acids. (Such extensive analytical work is not unusual where products for human consumption must be checked.)

In this chapter we will follow an example worked out by the group of Professor Rius of the Analytical Laboratory of the Chemistry Department of the University Rovira i Virgili of Tarragona (see Section 12.5). They were faced with setting up an analytical procedure for routine analysis of wine samples. Since hundreds of identical analyses have to be carried out each day, it is important to automate the procedure and reduce the time needed for each analysis.

The chemical analysis was supposed to be carried out by HPLC. The chief figure of merit is the *performance factor* of an analysis, which includes two things: first, how well the components are separated on the column, and how long it takes for the entire analysis to be carried out.

It is evident that the performance factor is a compromise between two features: separability and time. Both features are influenced by

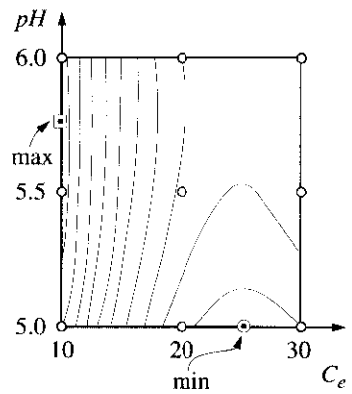

Figure 12-2: Model surface of the performance factor *PF* obtained by a standard modeling technique.

the selection of the mobile phase: if it passes through the column too quickly, then some components may not be well separated; if it passes through too slowly, then the time per analysis may be too great.

The first goal in setting up the analytical procedure is to make a model for the mobile phase, specifically for the performance factor as a function of its properties. Then we can select the most appropriate mobile phase for a given analysis.

In this example, only two properties of the mobile phase will be considered: the concentration of ethanol, C_e, and the acidity of the mobile phase (its *pH*). Experience has shown that C_e should be between 10 and 30%, while the *pH* should be between 5.0 and 6.0.

In order to build a model, we have to have some data, whether we will be using a classical method or a neural network. To obtain the data, a classical two-variable three-level experimental design is used, as shown in Figure 12-1. Nine pairs of variables (C_e, *pH*) are selected, and *HPLC* chromatograms are made for each; then a performance factor *PF* is assigned to each pair (Table 12-1).

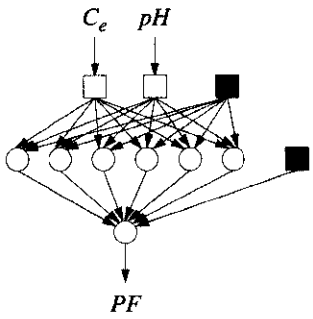

Figure 12-3: The (2 x 6 x 1) neural network used in modeling the HPLC mobile phase.

no.	C_e [%]	*pH*	*PF*
1	10	5.0	6.08
2	20	5.0	2.42
3	30	5.0	2.10
4	10	5.5	7.31
5	20	5.5	3.00
6	30	5.5	3.13
7	10	6.0	7.06
8	20	6.0	3.72
9	30	6.0	3.37

Table 12-1: The data for building a model of the HPLC process.

Standard modeling assumes a quadratic polynomial for the performance factor:

$$PF = ax^2 + by^2 + cxy + dx + ey + f \qquad (12.1)$$

where x stands for the concentration of the ethanol, C_e, and y stands for the *pH*. The parameters a, b, c, d, e, and f are obtained by any standard modeling or optimization technique. Setting up the model by SIMPLEX optimization leads to the following set of coefficients:

$$a = 0.018 \quad b = -1.42 \quad c = 0.0145$$
$$d = -0.995 \quad e = 16.16 \quad f = -31.86 \qquad (12.2)$$

Using the above parameters in (12.1) gives an expression for the performance factor *PF* over the entire measurement space, as shown in Figure 12-2. This model maximizes *PF* at $C_e = 10.2\%$ and $pH = 5.7$.

12.3 Modeling the Mobile Phase for HPLC by a Neural Network

In the next step, we will use the same data (Table 12-1) for building a neural net with one hidden layer employing back-propagation learning strategy. (Back-propagation is chosen for learning because building a model **always** involves supervised learning.)

Evidently, the network should have two neurons for input and one for output. The entire training set involves only 27 different numbers, i.e., nine input vectors each having two input values, and one target. The number of weights trained by the back-propagation strategy should, at least in principle, **not exceed** this number.

A few small neural networks with three, four, five, and six neurons in the hidden layer were constructed and the model trained with the nine input vectors and their targets; the result were compared with the classical model. It turns out that the best model is the (2 x 6 x 1) network, with (2 + 1) x 6 + (6 + 1) x 1 = 25 weights (Figure 12-3).

The comparison of the neural network model with the model obtained by optimization is shown in Figure 12-4. The differences are small indeed; the optimum in the neural network model is at $C_e = 10\%$ and $pH = 5.6$. The overall agreement between their responses is very good.

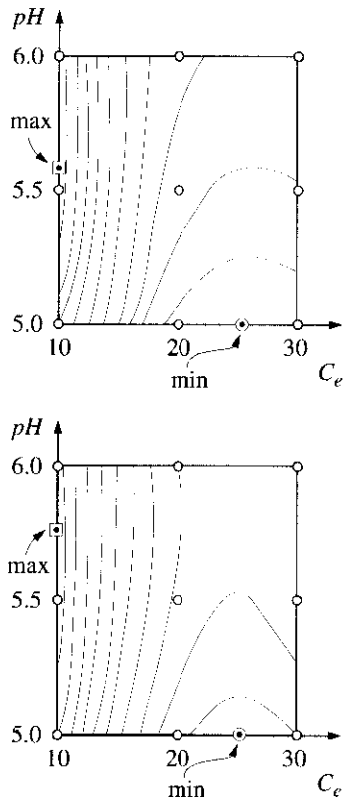

Figure 12-4: Above: the model obtained by the neural network using back-propagation learning; below: the model obtained by the classical modeling technique.

> Note that no hypothesis about the model function was necessary for constructing the neural network model.

In the classical model, we had to guess a model function. However, this does have an advantage: the parameters of the final

model indicate how the response depends on the particular variables. For example, the parameters (12.2) in combination with Equation (12.1) indicate that concentration of ethanol and the *pH* act in the quadratic and linear terms oppositely to each other, which apparently causes the coefficient c of the mixed term to be relatively small. Such evidence of how individual variables influence the measured quantity is very important; unfortunately, this information is missing from the neural network model.

12.4 Comparison of Networks with Identical Architectures

First, let us take a closer look at the weights of the neural network. We did two training runs of the same network with the same set of nine input vectors and targets, but with different initial weights. This leads to two completely different networks (as judged by the final weights of each network, Table 12-2). However, **the maps produced** by these different networks **are almost identical** as can be seen in Figure 12-5.

This means that the (2 x 6 x 1) network is too large for the problem, or to be more precise, the network is too large for the amount of data we have at disposal to train it. This fact becomes evident if one counts the number of weights in the network on Figure 12-3. This (2 x 6 x 1) network has, including the biases, in the hidden layer, i.e., in the layer between the input and the hidden nodes 3 x 6 = 18 weights and in the output layer 3 x 1 = 3 weights. Together there are 21 weights in the network. For nine input vectors (Table 12.1) these are far too many, and it should not come as a surprise that virtually any new random set of weights at the beginning of learning will yield a different set of weights after learning ended (the weights stabilized). This result is as anticipated because the problem of modeling the mobile phase for HPLC is a relatively simple case and will therefore not have many local minima or maxima but will have a response surface that is relatively smooth. The response surfaces yielded by both completely different networks are almost identical (Figure 12-5).

The smallest error-backpropagation network with one hidden layer that one can create for nine input objects represented by 2-dimensional input vectors has 5 nodes in the layout (2 x 2 x 1) as shown in Figure 12-6. Adding the bias weights this (2 x 2 x 1) network

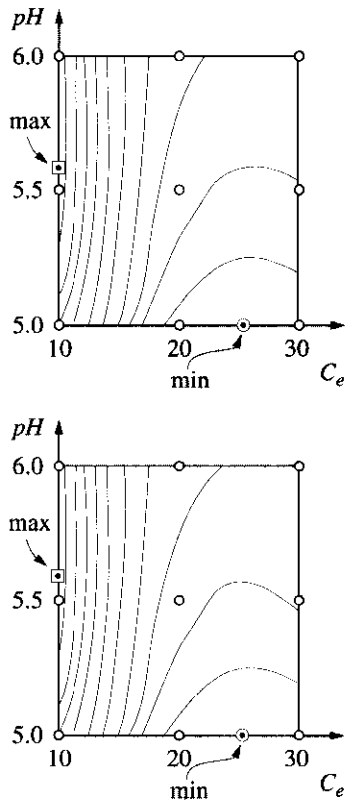

Figure 12-5: Identical maps obtained by two (2 x 6 x 1) networks having completely different weights.

has 9 weights. Someone might expect that the small network would now always converge to the same set of weights. We have tried more than 50 different randomization and found eight different sets of weights. For the novice to the field the result might come somewhat unexpected. There are eight different sets of weights, however, all are composed of the same set of absolute figures! Due to the symmetrical nature of the normalized input vectors these eight different weight sets turn out to be two groups of four sets with identical weights, the only difference being that the input nodes are changing the places (Table 12.3). The changes of places and signs of weights in these "different" sets are made in such a manner that the resulting outputs are always the same. Although seemingly different, the eight neural network models nevertheless always yield surface maps identical to those in Figure 12-5.

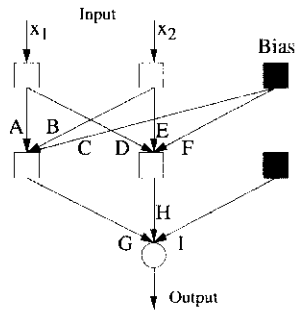

Figure 12-6: The smallest possible two layer error-back-propagation network for the problem of nine 2-dimensional input vectors.

The reader can check the results exposed in this Chapter by visiting the site on the web with the address

http://www2.ccc.uni-erlangen.de/ANN-book/

> In order to obtain numbers comparable to the experimental values, the inputs and outputs of a neural network must be properly scaled.

		network 1 input			network 2 input		
w_{ji}^1		$i=1$	2	bias	$i=1$	2	bias
$j=$	1	−5.49	0.30	−0.15	−4.37	0.97	−0.78
	2	0.36	3.89	−0.93	1.17	−3.21	−0.81
	3	0.85	0.67	−0.19	−1.25	0.11	−1.15
	4	−2.85	−0.61	−0.68	6.58	−0.60	−0.41
	5	5.44	−0.32	0.15	1.45	1.64	−1.87
	6	−0.95	−1.16	−1.59	−1.45	0.46	−0.16
w_{ji}^2		$j=1$			$j=1$		
$i=$	1	4.51			3.26		
	2	2.88			−1.85		
	3	0.39			0.84		
	4	1.76			−4.60		
	5	−4.47			1.84		
	6	0.03			0.60		
	bias	−0.03			1.82		

Table 12-2: Weights w_{ji}^l of two (2 x 6 x 1) networks giving very similar maps.

	weights on the first hidden neuron			weights on the second hidden neuron			weights on the output neuron		
network	w_{i1}^1 $i=1$	2	bias	w_{i2}^1 $i=1$	2	bias	w_{i1}^2 $i=1$	2	bias
1	A	B	−C	D	E	−F	−G	H	I
2	A	B	−C	−D	−E	F	−G	−H	I
3	−A	−B	C	D	E	−F	G	H	−I
4	−A	−B	C	−D	−E	F	G	−H	I
5 ≈ 1	D	E	−F	A	B	−C	H	−G	I
6 ≈ 2	−D	−E	F	A	B	−C	−H	−G	I
7 ≈ 3	D	E	−F	−A	−B	C	H	G	−I
8 ≈ 4	−D	−E	F	−A	−B	C	−H	G	−I

Table 12-3: Weights, w_{ji}^1, of eight "different" (2 x 2 x 1) neural network models which all yield the same response surface shown in Figure 12-5.

12.5 References and Suggested Readings

12-1. J. Zupan and F. X. Rius, "XYZ, A Program for Modeling, Validation, Prediction and Graphic Display of Data", *Anal. Chim. Acta* **239** (1990) 311 – 315.

12-2. M. Tusar, X. F. Rius and J. Zupan, "Neural Networks in Chemistry", *Mitteilungsblatt der GDCh-Fachgruppe "Chemie–Information–Computer" (CIC)* **19** (1991) 72 – 84.

12-3. D. L. Massart, B. G. M. Vandeginste, S. N. Deming, Y. Michotte and L. Kaufmann, *Chemometrics: A Textbook*, Elsevier, Amsterdam, NL, 1988.

12-4. S. N. Deming and S. L. Morgan, *Experimental Design: A Chemometrics Approach*, Elsevier, Amsterdam, NL, 1987.

12-5. K. W. C. Burton and G. Nickless, "Optimization via Simplex", *Chemom. Intell. Lab. Syst.* **1** (1987) 135 – 149.

12-6. S. N. Deming and L. R. Parker, "A Review of Simplex Optimization in Analytical Chemistry", *Crit. Rev. Anal. Chem.* (Sept. 1978) 187 – 202.

12-7. M. Tusar, J. Zupan and J. Gasteiger, "Neural Networks and Modelling in Chemistry", *J. Chim. Phys.* **89** (1992) 1517 – 1529.