Zbl. Bakt. 284, 516–539 (1996)

Quantitative Analysis of Multivariate Data Using Artificial Neural Networks: A Tutorial Review and Applications to the Deconvolution of Pyrolysis Mass Spectra

ROYSTON GOODACRE, MARK J. NEAL, and DOUGLAS B. KELL

Institute of Biological Sciences, University of Wales, Aberystwyth, Dyfed, UK

Received April 29, 1994 · Accepted January 11, 1996

Summary

The implementation of artificial neural networks (ANNs) to the analysis of multivariate data is reviewed, with particular reference to the analysis of pyrolysis mass spectra. The need for and benefits of multivariate data analysis are explained followed by a discussion of ANNs and their optimisation. Finally, an example of the use of ANNs for the quantitative deconvolution of the pyrolysis mass spectra of *Staphylococcus aureus* mixed with *Escherichia coli* is demonstrated.

Introduction

Multivariate data consist of the results of observations of many different characters (variables) for a number of individuals (objects) (73, 74). Each variable may be regarded as constituting a different dimension, such that if there are *n* variables each object may be said to reside at a unique position in an abstract entity referred to as *n*-dimensional hyperspace. This hyperspace is necessarily difficult to visualise, and the underlying theme of multivariate analysis (MVA) is thus *simplification* (22) or dimensionality reduction, which usually means that we want to summarise a large body of data by means of *relatively* few parameters, preferably the two or three which lend themselves to graphical display, with minimal loss of information.

In the case of spectroscopy, variables are usually represented by properties such as the absorbance at particular wavelengths (e.g. ref. 73). A spectral technique which seems ideally suited to analysis by multivariate methods is pyrolysis mass spectrometry (PyMS). Pyrolysis is the thermal degradation of complex molecules in a vacuum which causes their cleavage to smaller, volatile fragments separable by a mass spectrometer (80) on the basis of their mass-to-charge ratio (m/z). Almost all biological materials will produce pyrolytic degradation products such as methane, ammonia, water, methanol and H_2S, whose $m/z < 50$, and fragments with $m/z > 200$ are rarely analytically important for bacterial discrimination (8) unless very special conditions are employed (98); the analytically useful multivariate data are then constituted by a set of 150 normalised intensities versus m/z in the range 51 to 200.

Conventionally, at least within microbiology and biotechnology, because PyMS has been used as a taxonomic aid (8, 43, 55, 64, 71, 80), the reduction of the multivariate data generated by the PyMS system (and indeed of those generated by other arrays of sensors; e.g. gas chromatography (70), spectroscopic methods (74), and nuclear magnetic resonance (67) is normally carried out using principal components analysis (PCA; 21, 22, 30, 34, 56, 74). This is a well-known technique for reducing the dimensionality of multivariate data whilst preserving most of the variance, and so is an excellent technique for observing the *natural* relationships between multivariate samples. Whilst it does not take account of any groupings in the data, neither does it require that the populations be normally distributed, i.e. it is a non-parametric method (in addition, it permits the loadings of each of the m/z ratios on the principal components to be determined, and thus the extraction of at least some chemically significant information). The closely-related canonical variates analysis (CVA) technique then separates the samples into groups on the basis of the principal components and some *a priori* knowledge of the appropriate number of groupings (70, 109). Provided that the data set contains "standards" (i.e. type or centro-strains) it is evident that one can establish the closeness of any unknown samples to a known organism, and thus effect the identification of the former, a technique termed 'operational fingerprinting' by *Meuzelaar* et al. (80). An excellent example of the discriminatory power of the approach is the demonstration (44) that one can use it to distinguish *E. coli* strains which differ only in the presence or absence of single antibiotic-resistance plasmids. However, only rarely has the chemical basis for any such differences either been sought or found.

Analyses of the above type fall into the category of "unsupervised learning", in which the relevant multivariate algorithms seek "clusters" in the data (30). This allows the investigator to group objects together on the basis of their perceived closeness in the *n*-dimensional hyperspace referred to above. Such methods, then, although in some sense quantitative, are better seen as qualitative since their chief purpose is merely to *distinguish* objects or populations. More recently, a variety of related but much more powerful methods, most often referred to within the framework of chemometrics, have been applied to the "supervised" analysis of multivariate data. In these methods, of which multiple linear regression (MLR), partial least squares regression (PLS) and principal components regression (PCR) are the most widely used, one seeks to relate the multivariate spectral inputs to the concentrations of target determinands, i.e. to generate a quantitative analysis, essentially *via* suitable types of multidimensional curve fitting or regression analysis (14, 15, 17, 74–76, 79). Although non-linear versions of these techniques are increasingly available (e.g. 35, 63, 67, 100, 110, 111, 112), the usual implementations of these methods are linear in scope. A related approach to chemometrics, which is inherently nonlinear, however, is the use of (artificial) neural networks (ANNs) (see below).

For a given analytical system there are some patterns (e.g. mass spectra) which have desired responses which are known (i.e. the concentration of target determinands). These two types of data (the representation of the objects and their responses in the system) form pairs which for the present purpose are called inputs and targets. The goal of supervised learning is to find a *model* or *mapping* that will correctly associate the inputs with the targets (Fig. 1).

Thus the basic idea in these supervised learning techniques is that there are minimally 4 data sets to be studied, as follows. The "training data" consist of (i) a matrix of *s*

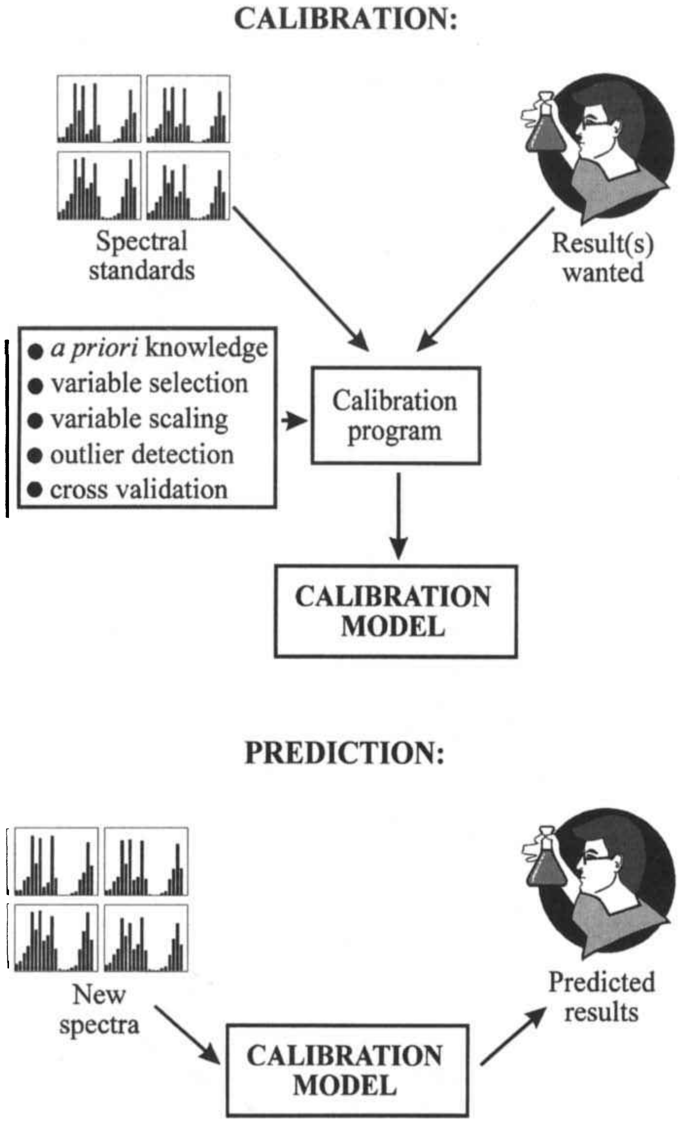


Fig. 1. The process of multivariate calibration *via* supervised learning consists of two stages. Calibration: establishing a CALIBRATION MODEL for later prediction of "results" (e.g., amount of determinand) from "spectra" by matching spectral standards and the known results wanted from a set of calibration samples (the training set) in some sort of supervised learning calibration program; this may be by using a neural network simulation, or with a program that performs multiple linear regression, partial least squares or principal component regression. Some background knowledge may be used in the formation of the model such as *a priori* knowledge, variable selection and scaling, outlier detection and cross validation.
Prediction: converting instrumental data for new samples into predictions of wanted results (e.g., in terms of determinand concentration) using the above previously established CALIBRATION MODEL in the computer program.

rows and *n* columns in which *s* is the number of objects and *n* the number of variables (these may be the absorbance at particular wavelengths, or in the present case the normalised ion intensities at a particular m/z; Fig. 1), and (ii) a second matrix, again consisting of *s* rows and typically 1 or two columns, in which the columns represent the variable(s) whose value(s) it is desired to know (these are the result(s) wanted; Fig. 1) and which for the training set have actually been determined by some existing, "benchmark" method. This variable may be the concentration of a target determinand, and is always paired with the patterns in the same row in (i). The "test data" also consist of two matrices, (iii) and (iv), corresponding to those in (i) and (ii) above, but the test set contains different objects. As the name suggests, this second pair is used to test the accuracy of the system; alternatively they may be used to cross-validate the model. That is to say, after construction of the model using the training set (i, ii) the test data (iii) (these may be new spectra; Fig. 1) are then "passed" through the calibration model so as to obtained the model's prediction of results. These may then be compared with the known, expected responses (iv).

As in all other data analysis techniques, these supervised learning methods are not immune from sensitivity to badly chosen initial data (113). Therefore the exemplars for the training set *must* be carefully chosen; the golden rule is "garbage in – garbage out". An excellent example of an unrepresentative training set was discussed some time ago on the BBC television programme *Horizon;* a neural network was trained to attempt to distinguish tanks from trees. Pictures were taken of forest scenes lacking military hardware and of similar but perhaps less bucolic landscapes which also contained more-or-less camouflaged battle tanks. A neural network was trained with these input data and found to differentiate most successfully between tanks and trees. However, when a new set of pictures was analysed by the network, it failed to distinguish the tanks from the trees. After further investigation, it was found that the first set of pictures containing tanks had been taken on a sunny day whilst those containing no tanks were obtained when it was overcast. The neural network had therefore thus learned simply to recognise the weather! We can conclude from this that the training and tests sets should be carefully selected to contain representative exemplars encompassing the appropriate variance over all relevant properties for the problem at hand.

It is also imperative that the objects fill the sample space. If a neural net is trained with samples in the concentration range from 0 to 50% it is unlikely to give a accurate estimates for samples whose concentrations are greater than 50%, that is to say the network is unable to *extrapolate*. Furthermore for the network to provide good *interpolation* it needs to be trained with a number of samples covering the desired concentration range (47).

Artificial neural networks

ANNs are an increasingly well-known means of uncovering complex, non-linear relationships in multivariate data, whilst still being able to map the linearities. ANNs can be considered as collections of very simple "computational units" which can take a numerical input and transform it, usually *via* summation, into an output (see 1, 2, 6, 20, 24, 28, 40, 59, 60, 66, 78, 87, 88, 93, 97, 102 and 103 for excellent introductions; and 4, 7, 9–12, 18, 23, 25, 29, 37, 42, 45, 47, 52, 65, 69, 77, 85, 89, 92, 96, 99, 105 and 114 for applications in analytical chemistry and microbiology).

The relevant principle of "supervised" learning in ANNs is that, as with the multivariate calibration described above, they take numerical inputs (the training data, which are usually multivariate) and transform them into "desired" (known, predetermined) outputs. The input and output nodes may be connected to the "external world" and to other nodes within the network (for a diagrammatic representation see Fig. 2). The way in which each node transforms its input depends on the so-called "connection weights" (or "connection strength") and "bias" of the node, which are modifiable. The output of each node to another node or the external world then depends on both its

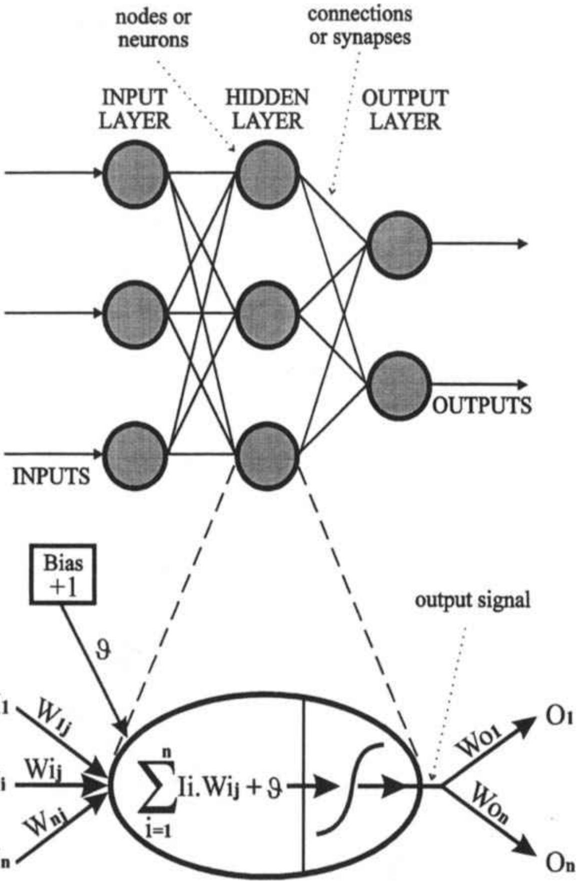


Fig. 2. A neural network consisting of 3 inputs (data for PyMS actually consisted of 150 inputs/masses) and 2 outputs (for the PyMS study this was a single node which represented the %*S. aureus*) connected to each other by 1 hidden layer consisting of 3 nodes (for PyMS this was actually 8). In the architecture shown, adjacent layers of the network are fully interconnected although other architectures are possible. One of the nodes in the hidden layer is given in more detail showing the information processing by node. An individual node sums its input (the Σ function) from nodes in the previous layer, including the bias (ϑ), transforms them *via* a "sigmoidal" squashing function, and outputs them to the next node to which it is linked *via* a connection weight.

weight strength, bias and on the weighted sum of all its inputs, which are then transformed by a (normally non-linear) weighting function referred to as its activation, threshold or squashing function. As with other supervised learning methods, the great power of neural networks stems from the fact that it is possible to "train" them. One can acquire sets of multivariate data (which may be pyrolysis mass spectra) from standard materials of known identities and train ANNs using these identities as the desired outputs. Training is effected by continually presenting the networks with the "known" inputs and outputs and modifying the connection weights between the individual nodes and the biases, typically according to some kind of back-propagation algorithm (93), until the output nodes of the network match the desired outputs to a stated degree of accuracy. The trained ANNs may then be exposed to unknown inputs (i. e. spectra) and they and will immediately provide the globally optimal best fit to the outputs.

Provided the ANN gives the correct results for "unknown" data (i. e. data unseen by the calibration system, but known by the operator) it may be said to have "generalised". The operator can now be confident that when *genuine* unknown spectra are passed through the neural network, the predictions will be accurate and precise, provided of course that these spectra bear some relationship to the training set.

The so-called *false-sample problem* often poses a major stumbling block for the routine application of these numerical techniques for the analysis of multivariate data (27). This occurs, for example, when an operator makes a calibration model to quantify substance *a* in *b*, but then tests the model on (say) *c* mixed with *b*. It should be obvious that the neural network calibration model is now being asked to extrapolate beyond the knowledge domain on which it was trained, and will thus fail to give an accurate prediction of the determinand. As well as exploiting appropriate software methods (e. g. statistical ones) to detect whether a test sample falls within the domain of validity of the training set, it is also possible to include suitable "false" samples and corresponding "dummy" output variables in the training set; this practice should allow the neural network to detect widely different samples (53). In this sense, then, analysis with ANNs of this type can fairly be regarded within the framework of the multivariate calibration approach as outlined in Figure 1.

The following summarises the fundamental nomenclature and the rudimentary mathematical concepts that are used to describe and analyse ANN processing.

The processing units

ANNs employ processing nodes (neurons or units), connected using abstract interconnections (connections or synapses). Connections each have an associated real value, termed the weight (w_i), that scales the input (i_i) passing through them (Fig. 2); this also includes the bias (ϑ), which also has a modifiable weight. Nodes sum the signals feeding to them *(Net)*:

$$Net = i_1 w_1 + i_2 w_2 + i_3 w_3 + \ldots + i_i w_i + \ldots + i_n w_n =$$

$$= \sum_{i=1}^{n} i_i w_i + \vartheta$$

Activation functions

The sum of the scaled inputs and the node's bias, are then scaled to lie between 0 and +1 (or sometimes between –1 and +1) by an activation function to give the nodes output *(Out);* this scaling is typically achieved using a logistic "squashing" (or sigmoidal) function:

$$Out = \frac{1}{(1 + \exp^{-Net})}$$

It is widely thought that a continuously differentiable sigmoidal squashing function (93, 97, 102) is most appropriate, although some advantage may accrue to the use of a linear activation function on the output nodes.

Neural network topology

ANN topologies, or architectures, are formed by organising nodes into layers (also termed fields or slabs) and linking these layers of neurons with modifiable weighted interconnections. A diagrammatic representation of a neural network consisting of 3 inputs and 2 outputs connected to each other by 1 hidden layer consisting of 3 nodes is shown in Figure 2. In the fully connected topology shown each of the 3 nodes in the input layer is connected to the 3 in the hidden layer, by 9 connection weights, which in turn are connected to the 2 output nodes, by a further 6 connection weights. In addition, there is also a bias (extra node), which always has an activation level of +1, which is connected to nodes in the hidden and output layers (but not the input layer) *via* modifiable weighted connections (5 in the example shown in Fig. 2). Such an architecture can be written as a 3-3-2 ANN, and is commonly referred to as a fully interconnected feedforward multilayer perceptron.

Other architectures are possible such as direct linear feed through, where in addition to the above the nodes in the input layer is also connected directly to the output layer. There are many other topologies where not all the nodes between layers are connected, the connections may be chosen or random.

It is known from the statistical literature that better predictions can often be obtained when only the most relevant input variables are considered (81, 90, 95). Therefore neural networks that prune larger networks are an active area of study (e.g., 33, 57, 18, 68, 84, 91, 104). It is also possible to grow neural networks from small ones (16, 32, 36, 82, 83).

The most widely used neural network topology is one that is fully connected (Fig. 2) and where the input and output nodes are connected *via* a single hidden layer. One reason that this architecture is so attractive for the quantitative analysis of multivariate spectral data is that it has been shown mathematically (26, 39, 61, 62, 108) that a neural network consisting of only one hidden layer, with an arbitrarily large number of nodes, can learn any, arbitrary (and hence non-linear) mapping of a continuous function to an arbitrary degree of accuracy. It is the presence of this hidden layer which permits the nonlinear mapping, since similar networks lacking a hidden layer can only effect a multivariate *linear* mapping (93). In addition, such ANNs are widely considered to be relatively robust to noisy data, such as those which may be generated by mass spectrometry or gas chromatography.

The above 3-3-2 architecture is rather simple and the question arises "How does one choose the number of nodes in the hidden layer?" For pyrolysis mass spectra there are 150 m/z values and thus 150 nodes in the input layer. It is important not to have too many nodes in the hidden layer(s) because this may allow the neural network to learn by example only and not to generalise (5). We have found that a suitable rule of thumb is that good generalisation often comes from using a number of nodes in the hidden layer that approximates to the natural logarithm of the number of nodes in the input layer. Thus with 150 input nodes this equates to 8. For a single determinand (output node) this architecture would be represented as a 150-8-1 ANN. However, it is noteworthy that during our work we have found that fewer nodes in the hidden layer (indeed often even zero) can be used to quantify PyMS data successfully.

Preparation of data

As mentioned previously it is of paramount importance to have the correct exemplars in the training and test sets. It is necessary also that these samples fill the sample space. Neural networks are very bad at extrapolating and to ensure good interpolation they need to be trained with samples equally-spaced over the desired concentration range; for a range of binary mixtures, it has been found, using PyMS, that 11 samples spaced every 10% will allow the network to generalise well (47).

Once the data used to train the ANN are collected, and the concentrations of the determinand ascertained using "wet chemistry", they are split into two data sets in which one half of the pair are the inputs (stimulus) of the network (for present purposes this would be the pyrolysis mass spectra normalised to the total ion count) and the other are the known or expected responses (i. e. the concentrations of the determinand[s]).

At this stage it is important to determine whether the training and/or test sets contain any outlying samples; these are usually detected using principal components analysis. It is known that if outliers are included in the construction of a calibration model then inaccuracies in the predictions from new multivariate data using the model are likely to occur (74).

Some of the m/z values may be omitted from the training data, a practice termed "pruning". It is unlikely however that the operator will know *a priori* which masses to remove so at least for typical back-propagation neural networks it is best to start with the intensities from all 150 masses.

The input and output nodes are next normalised between 0 and +1. It has been found (47, 52) that network generalisation is improved if the output layer is scaled to exploit less than the full range of the normalised scale, and the optimum for PyMS appears typically to be between +0.1 and +0.9. We have also found on occasion that scaling input nodes *individually*, i. e. over their own range rather than over the whole range encompassed in the entire input space, has improved learning rates dramatically (>100 fold) (52, 86).

Training the neural network

The first step is to choose the algorithm to be used for training the ANN. There are numerous algorithms available, and indeed the list of new ones expands continually. The most commonly employed is the standard back-propagation (BP) algorithm (93, 106, 107). Other algorithms which we have exploited include stochastic back-propagation, also termed learning by pattern (78), quick propagation (31), and Weigend weight elimination (104). The following describes training ANNs with the standard back-propagation algorithm.

Before training commences the connection weights are set to small random values, including the weights connecting the bias to the hidden and output layers (102). Next the stimulus pattern is applied to the network, which is allowed to run until an output is produced at each output node. The differences between the actual output and that expected, taken over the entire set of patterns, are fed back through the network in the reverse direction to signal flow (hence back-propagation) modifying the weights as they go. This process is repeated until a suitable level of error is achieved (93, 97, 102).

For any given ANN, set of connection weight values, and training set there exists an overall RMS error of prediction. An error surface can be constructed by using one dimension in a multidimensional space to represent each connection weight, and an additional one for the RMS error. The BP algorithm performs gradient descent on this error surface by modifying each weight in proportion to the gradient of the surface at its location. Two constants, *learning rate* and *momentum,* control this process; for standard BP a learning rate of 0.1 and a momentum of 0.9 often give the best results. Learning rate scales the magnitude of the step down the error surface taken after each complete calculation in the network (epoch), and momentum acts like a low pass filter, smoothing out progress over small bumps in the error surface by remembering the previous weight change.

It is known that gradient descent can sometimes cause networks to get "stuck" in a depression in the error surface should such a depression exist. These are termed "local minima" (97, 102). However, it has been found empirically that local minima are seldom problematic for larger networks dealing with problems such as those presently under discussion, since the chance of encountering a multidimensional depression that is bounded in every dimension is relatively small.

One complete calculation in the network is called an epoch. This is equivalent to one complete pass through all the training data, calculating for *each member* of the training set. For a 150-8-1 ANN topology, trained with the standard back-propagation algorithm (93, 106, 107), where the weights are updated after all the training data are seen, one epoch represents 1217 connection weight updatings (1200 weights between the input and hidden layer (150 × 8), 8 weights between the hidden layer and the output node (8 × 1), and 9 weights from the bias to the 8 nodes in the hidden layer and the single output node) and a recalculation of the root mean squared (RMS) error between the true and desired outputs over the entire training set. In contrast, one epoch for an ANN trained using the stochastic back-propagation (78) would also incluce the weight updatings after *each* of the training pairs is passed through the neural network.

Stability and convergence

During training a plot of the RMS error versus the number of epochs represents the "learning curve", and may be used to estimate the extent of training. Training may be said to have finished when the network has found the lowest RMS error. Provided the network has not become stuck in a local minimum, this point is refered to as the global minimum on the error surface.

It is known (45, 52, 93, 102) that neural networks can become over-trained. An over-trained neural network has usually learnt perfectly the stimulus pattern it has seen but can not give accurate predictions for unseen stimuli, and it is no longer able to generalise. For ANNs accurately to learn the concentrations of determinands in biological systems networks must obviously be trained to the correct point. It is therefore imperative that ANNs should be trained several (perhaps many) times to ascertain whether they converge reproducibly. In addition, a superior method to reveal when the neural network will best generalise is to use the (or a) test set to cross-validate the model. During training the network may be interrogated with new stimulus patterns so as to generate outputs at the output node. The stimuli used may be pyrolysis mass spectra, whose determinand concentration is known to the operator but not to the neural network. The error between the output seen and that expected may then be calculated thus allowing a second learning curve for the test set to be drawn. Training is stopped when the RMS error on the test or cross-validation data is lowest.

Table 1. Some parameters which one may vary during the production of a feedforward back-propagation neural network calibration model to improve learning/convergence and generalisation

1. Number of hidden layers:

   One is thought sufficient for most problems.
   More give a big increase in computational load.

2. Number of nodes in hidden layer:

   Rule of thumb says *ln* (number of inputs).

3. Architecture:

   Fully interconnected feedforward net is most common.
   Many others exist such as adaptive resonance theory, Boltzmann machine, direct linear feedthrough, Hopfield networks, Kohonen networks.

4. Number of exemplars in training set:

   Need enough to fill parameter space and to allow generalisation.
   When fewer are used then the network can "store" all the knowledge.

5. Number of input variables:

   Those that do not contribute positively to discrimination may impair generalisation and are best removed by pruning the input data.

6. Scaling of input and output variables:

   Individual scaling on inputs improves learning speed dramatically.
   There is a need to leave headroom, especially on the output layer.

7. Updating algorithm:

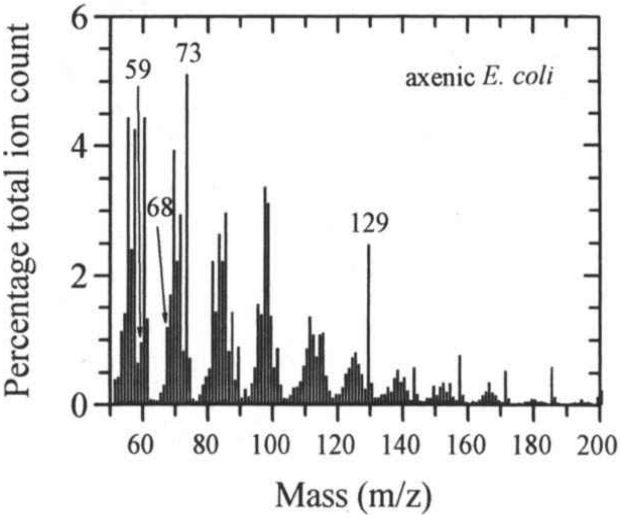
   There are many variants on the original "vanilla-flavoured" back-propagation (BP) – most of which give small but worthwhile improvements. Others include radial basis functions, quick-prop, stochastic BP, Weigend weight eliminator.
   Standard back-propagation (93) is still the most popular.

8. Learning rate and momentum:

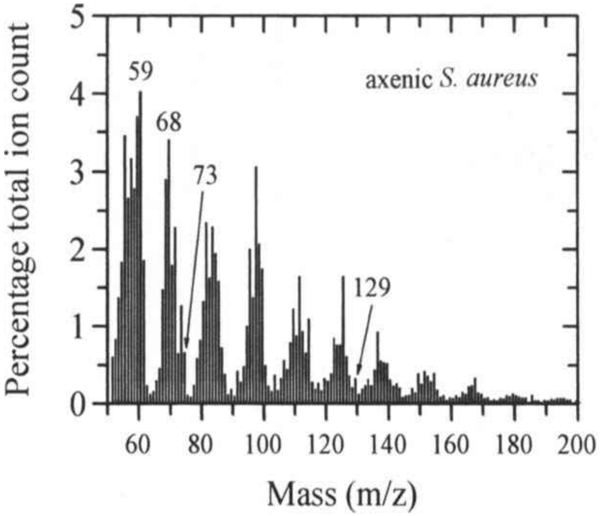
   Need to be carefully chosen so that the net does not get stuck in local minima nor "shoot" off in the wrong direction when encountering small bumps on the error surface.
   For standard BP a learning rate of 0.1 and a momentum of 0.9 are best.

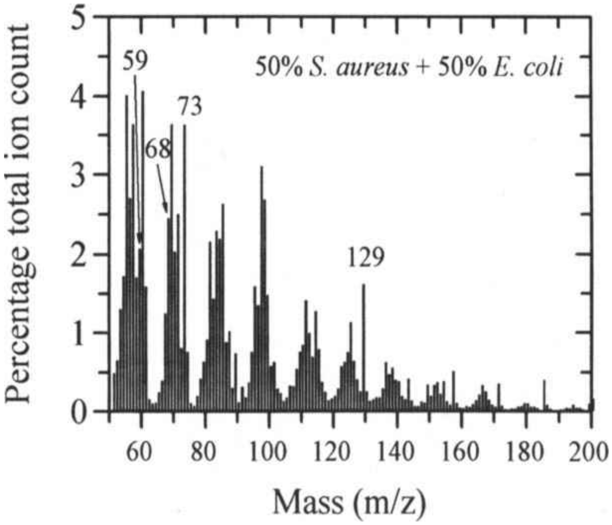
9. Stability:

   Best to reserve some of the training data for cross-validation.



Percentage total ion count
6
4
2
0
59
73
68
129
axenic *E. coli*
60
80
100
120
140
160
180
200
Mass (m/z)


axenic *S. aureus*
59
68
73
129
Percentage total ion count
0
1
2
3
4
5
60
80
100
120
140
160
180
200
Mass (m/z)


50% *S. aureus* + 50% *E. coli*
59
68
73
129
Percentage total ion count
0
1
2
3
4
5
60
80
100
120
140
160
180
200
Mass (m/z)