Journal of Computer-Aided Molecular Design, 11 (1997) 135–142
ESCOM

J-CAMD 381

Data modelling with neural networks: Advantages and limitations*

D.J. Livingstone[a,]**, D.T. Manallack[b] and I.V. Tetko[c]

[a]*Centre for Molecular Design, University of Portsmouth, Halpern House, 1-2 Hampshire Terrace, Portsmouth PO1 2QF, U.K.*
[b]*Chiroscience Ltd., Cambridge Science Park, Milton Road, Cambridge CB4 4WE, U.K.*
[c]*Institute of Bioorganic and Petroleum Chemistry, Ukrainian Academy of Sciences, Murmanskaya 1, Kiev-660, 253660 Ukraine*

Received 28 August 1996
Accepted 15 September 1996

Keywords: Overfitting; Overtraining; Data display; Chance effects; Nonlinear relationships; Variable selection

Summary

The origins and operation of artificial neural networks are briefly described and their early application to data modelling in drug design is reviewed. Four problems in the use of neural networks in data modelling are discussed, namely overfitting, chance effects, overtraining and interpretation, and examples are given of the means by which the first three of these may be avoided. The use of neural networks as a variable selection tool is shown and the advantage of networks as a nonlinear data modelling device is discussed. The display of multivariate data in two dimensions employing a neural network is illustrated using experimental and theoretical data for a set of charge transfer complexes.

Introduction

Artificial neural networks (ANN) are computer systems which attempt to model the way the brain works. They have considerable appeal in that they appear to be able to make decisions, like humans, based on complex, noisy, irrelevant and/or partial information [1]. Although they appear to be a recent phenomenon, the origins of ANN predate the earliest electronic computers [2]; however, a critical comment on perceptrons published in 1969 [3] had the effect of stopping virtually all research funding into ANN until the early 1980s, a period referred to as the 'dark age' in the introduction to the book by Eberhart and Dobbins [1]. The last 10 years or so have seen an explosion of interest in neural network research, along with an ever increasing array of practical applications in fields as diverse as the control of nuclear reactors and the grading of pork for fat content [4]. That such systems have a significant economic impact can be seen from a recent posting to the usenet newsgroup comp.ai.neural-nets, in which it was claimed that the use by an airline of a neural network to forecast passenger demand led to annual savings of U.S.$ 140.000.000.

The comparison of ANN with the human brain may have led to some of the 'hype' that surrounds them but, while it is true that ANN may be considered to be just algorithms that are designed to solve particular problems, the analogy with the brain is a very useful way to describe the construction and operation of networks. The basic building blocks of ANN, like biological neural networks, are neurons and these artificial neurons simulate some of the functions of biological neurons. Artificial neurons receive input signals, process these signals so as to decide whether to 'fire' and produce an output signal as a result of this processing [5]. The connections between these neurons and the type of processing that they do is often referred to as a neural network paradigm, and one of the most popular used in data modelling consists of layers of artificial neurons in which each neuron in a particular layer is connected to every neuron in the next layer. This is variously known as a multilayer perceptron (MLP) or feed-forward back-propagation (BP) network, since the data used to train the network is presented at the first (input) layer and then feeds forward through the hidden layer(s) to produce a response at the output layer. All of the networks discussed here are of this form, although in one, the ReNDeR network, the signals of interest are produced in a hidden layer.

*This paper is based in part on a presentation given at the 14th Molecular Graphics and Modelling Society Conference, held in Cairns, Australia, August 27–September 1, 1995.

**To whom correspondence should be addressed at: ChemQuest, Cheyney House, 19-21 Cheyney Street, Steeple Morden, Herts SG8 0LP, U.K.

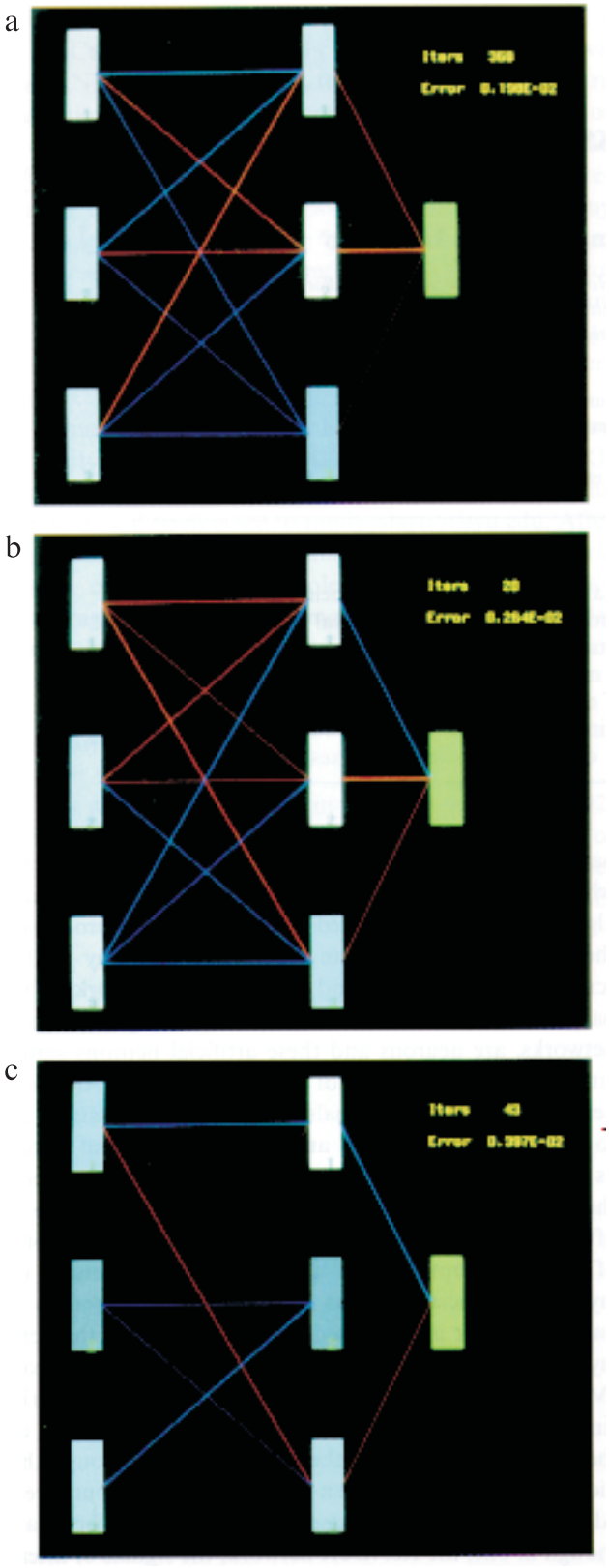

Fig. 1. Pictorial representation of three different neural networks trained to reproduce log P values from α, μ and E. Positive connections are coded in red and negative connections in blue. The magnitude of the connection weights is indicated by the intensity of the colour. The connections are summarised as the first three entries in Table 1.

Use of an artificial neural network involves training and this is carried out in much the same way that biological neural networks train. The connections between the artificial neurons are adjustable parameters with a sign and a magnitude and training involves adjustment of these connection strengths (weights) until some desired output (target) signals are produced. A common form of training involves starting the network with random values for the connection weights, presentation of the data and calculation of output signals. The output values are compared with the targets and the weights are adjusted, hence the use of the term 'back propagation', since the output errors are computed at the output layer and the weights are adjusted backwards through the network to the input layer [6].

Quantitative models

The earliest applications of ANN in the production of quantitative models involved the use of both classified [7] and continuous response data [8,9]; in other words, the equivalent of linear discriminant analysis (LDA) and multiple linear regression analysis (MLR). One of the intriguing features of these models is that they appeared to provide a better explanation of the data (higher correlation coefficients, for example) than the corresponding statistical methods, while not requiring the same transformations of the input terms (e.g. square terms) or the use of 'extra' variables such as indicators. A clue to the reason for this improved performance lies in the term 'linear' in LDA and MLR. ANN are not restricted to the generation of linear models, and the combination of a large set of connection weights and nonlinear transfer functions in a BP network allows models of any complexity to be fitted between the response and the descriptors [10].

Enthusiasm for this new approach to data modelling led to the creation of some neural networks in which there were more connections between the neurons than there were data points in the set being modelled. Since each connection weight is an adjustable parameter, comparable, say, to the regression coefficients in a linear regression model, the apparently good performance of these networks raised a suspicion that they may be overfitting the data. As we began to examine the performance of ANN as a data modelling tool, it became clear that they may also be susceptible to chance effects, in much the same way that statistical methods are [11], and that it was possible to overtrain a network. The question of interpretation also arises; is it possible, for example, to identify important variables (and their contributions) by an examination of the weights of a trained network? We have thus set out to examine these four important features of ANN as a data modelling tool: overfitting, chance effects, overtraining and interpretation.

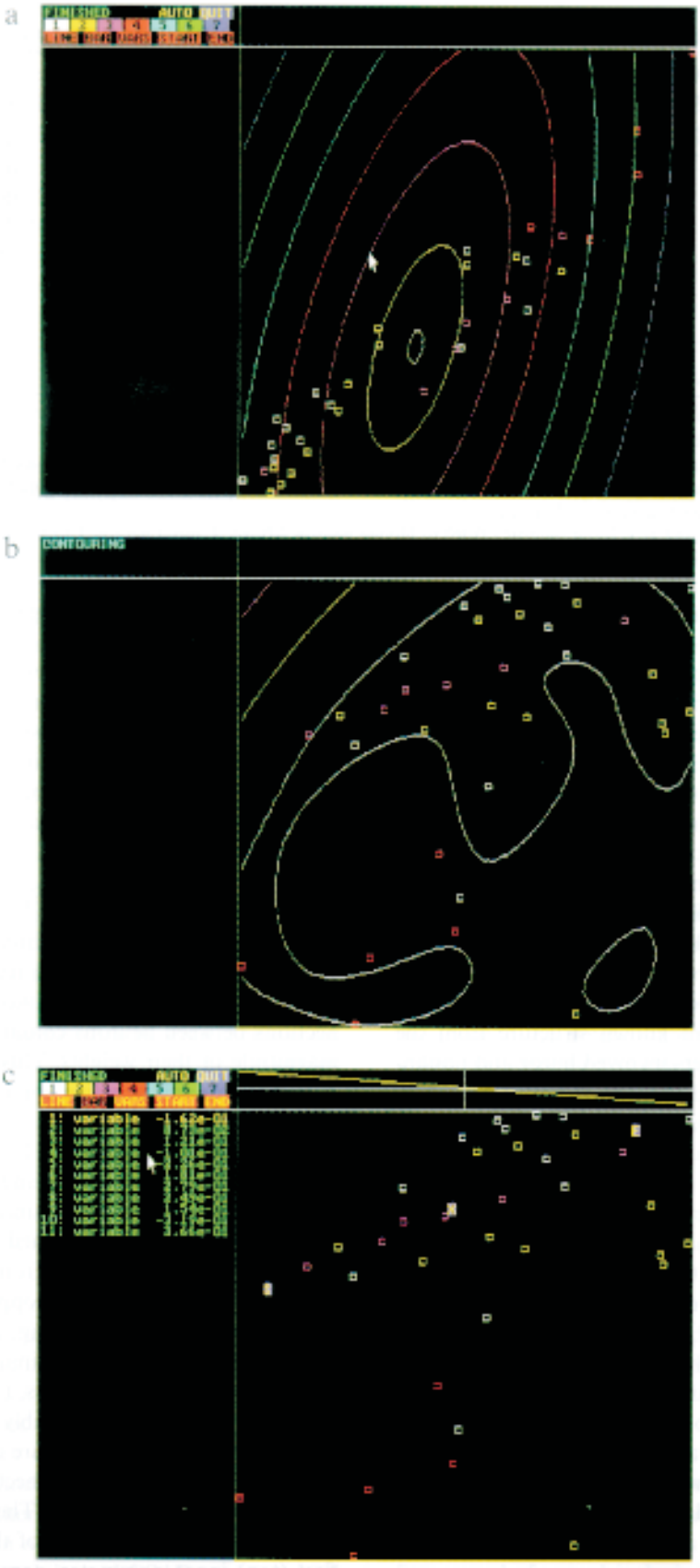

Fig. 2. 2D displays of the charge-transfer data produced by ReNDeR (see text for details). The line diagram shown above the plot in c shows how CMR changes over the line indicated by the cursor points (S for the start of the diagram, E marks the end and X marks the position of the vertical line).

TABLE 1
SUMMARY OF THE INPUT-HIDDEN LAYER CONNECTIONS FOR 3-3-1 NETWORKS TRAINED[a] ON A LINEAR DATA SET

Network[b]	Fitting error ($\times 10^{-2}$)	Cycles trained	Connection no.[c]								
			1	2	3	4	5	6	7	8	9
1	0.198	368	B	R	(B)	B	R	(B)	R	(B)	(B)
2	0.264	20	R	(R)	R	R	(R)	B	B	(B)	(B)
3	0.397	43	B	–	R	–	(B)	(B)	(R)	B	(R)
4	0.208	37	R	R	B	B	B	R	B	B	R
5	0.208	67	B	B	R	R	R	B	(B)	R	–
6	0.197	100	B	R	R	–	B	–	R	B	B
7	0.182	200	–	R	–	R	(R)	(R)	–	B	(R)
8	0.208	38	R	B	–	B	(R)	R	B	R	–
9	0.207	50	R	B	R	–	(R)	(B)	B	(R)	B
10	0.178	365	R	(B)	(B)	(R)	R	(R)	B	–	(R)

The sign of the connection weights is indicated as red (R) for positive and blue (B) for negative (as shown for the first three networks in Fig. 1) with the magnitude indicated at three levels, e.g. B (high), (B) (intermediate), – (low).

[a] See Ref. 14. Networks were trained with conjugate gradient minimisation using the BIOPROP simulator for a maximum of 500 cycles of training. The training terminated when the change in fitting error fell below the default value and in all cases this was before the iteration limit was reached.

[b] The first three networks are shown in Figs. 1a, b and c.

[c] Network connections are numbered as 1: input neuron 1 (top) to hidden neuron 1 (top); 2: input neuron 1 to hidden neuron 2 (middle); 3: input neuron 1 to hidden neuron 3 (bottom); and so on, ending with 9: input neuron 3 to hidden neuron 3.

Experiments with random numbers showed that both discriminant networks [12] and regression networks [13] do suffer from chance correlations, and guidelines were proposed, based on the ratio of data points to network connections (ρ), to minimise the possibility of such chance effects. Andrea and Kalayeh, who first proposed the use of the parameter ρ [9], showed that there were upper and lower limits for ρ in the data sets they analysed. Values of ρ lower than 1.8 resulted in a network that would 'memorise' the data, while networks with ρ values greater than 2.2 contained an insufficient number of connections for training to be successful. The real test of performance of any data model, of course, is found in prediction and since this cannot be tested with random numbers we selected several data sets of known structure from the literature [14]. These data sets involved linear and nonlinear relationships, with and without indicator variables, and it was shown that decreasing ρ, by increasing the number of hidden layer neurons, led to models that performed well in fitting but poorly in prediction; similar results, in terms of prediction, have been reported by Campbell and Johnson for abductive networks [15]. This is a characteristic of overfitted ANN (but see below), i.e., the model performs well on the training set data but poorly in cross-validation*. Interestingly, all four ANN models gave lower cross-validated R^2 values than the corresponding leave-one-out R^2 value for the regression equation, at all values of ρ. The expectation when these experiments were set up was that there would be a particular value, or range of values, of ρ that would give a compromise between good fitting and good prediction, but at which the network might be expected to work as well as a regression model.

The network performance was 'best', in terms of similarity with the multiple linear regression results, for the linear data set. This set consisted of measured log P and calculated values of polarisability (α), dipole moment (μ) and energy of the highest occupied molecular orbital (E) for a set of 37 diverse compounds [16], giving the following regression equation:

$$\log P = 0.412\alpha - 0.359\mu + 0.384E - 7.115 \qquad (1)$$

One of the networks trained from these data is shown in Fig. 1a. As described in the legend to the figure, this pictorial representation of trained networks has the connections between neurons colour-coded for the sign and magnitude of their weights. This particular network has both positive and negative connections between each of the input parameters and the three hidden layer neurons and has a fitting error of 0.198×10^{-2}. Networks were created using the BIOPROP simulator (see Experimental) and training was continued until the default change in network error limit was reached. Figures 1b and c show two more trained networks from this data set; in both cases network training had stopped because the network error was no longer improving, in other words, the network had reached a local minimum. This is another quite typical feature of BP networks, i.e., they have a tendency to find local minima, presumably because the fitting error surface has a complex structure due to the large number of adjustable parameters (connections), and these minima are often very close in value. The network shown in Fig. 1b has positive values for all of the connections from the first (top) input parameter, negative values for the connections from the last parameter and both positive and negative connections for the second input descriptor. The

*Both leave-one-out and leave-N-out (where N is approximately 10% of the data) strategies were employed and these gave comparable results.

third network, shown in Fig. 1c, has both positive and negative connections for the first and third descriptors but purely negative connections for the second. Fitting errors for these two networks were 0.264×10^{-2} and 0.397×10^{-2}, respectively which, although higher than for the network shown in Fig. 1a, are still quite respectable solutions and thus may have been chosen as the final result for network training on this data set. Table 1 shows a summary of the connections for 10 networks fitted to these data, including the networks shown in Figs. 1a, b and c, where it may be seen that quite different connection patterns occur even for networks with the same fitting errors (4, 5, 8 and 9). Although the neural networks shown here could not be expected to be as simple to interpret as Eq. 1, since the three hidden neurons provide multiple paths to the output, this does demonstrate the complexity of neural network models, even in fitting such a simple linear example. A quote from Smolensky [17], cited in the book *Intelligent Hybrid Systems* [18], nicely describes this: 'Each connection represents a soft constraint; the knowledge contained in the system is the set of all such constraints. If two units have an inhibitory connection, then the network has the knowledge that when one is active the other ought not to be. Any of the soft constraints can be overridden by others, they have no implications singly; they only have implications collectively.'

The work reported previously [12–14] and the networks shown here demonstrate that ANN may suffer from chance effects and overfitting and that interpretation of the fitted ANN models is not straightforward. What of the question of overtraining? One of the difficulties in the construction of ANN models is the choice of an appropriate stopping point for training, since it is always possible to continue to reduce the error in a neural network by repeated adjustment of the network weights. The danger in this is that the network will simply fit a more complex model to that particular training set and, as the model fits the training data better, it becomes less able to generalise. As Table 1 shows, given sufficient complexity in architecture, a network is capable of giving quite different solutions to even such a simple three-parameter linear problem. A training strategy which is quite well known in the neural network field is called 'early stopping'. This method involves the use of a 'control' set to monitor network training and training is continued (using a training set as for other training strategies) until the control set error begins to rise, hence 'early stopping'. Although the control set is not used in the actual construction of the network (i.e. to adjust connection weights), it is used to monitor the performance of the network and thus this approach requires a further set to function as an independent test set. Application of this technique to both artificial structured data sets and the real QSAR sets examined previously [14] showed that it was possible to produce trained networks that performed well both in fitting and in prediction [19]. An interesting feature of these experiments was that the use of the early stopping criteria appeared to prevent overfitting. The network performance was unaffected by the number of hidden layer neurons, and hence connections, employed in the networks and thus early stopping seems to avoid the problems of both overtraining and overfitting.

Experimental

The neural networks shown in Fig. 1 and reported in Table 1 were computed using the BIOPROP package (Laboratory of Biodynamics, University of California, Berkeley, CA, U.S.A.) running on a Silicon Graphics Indigo2 workstation. The displays were produced by an in-house (SmithKline Beecham, Welwyn, Herts, U.K.) modification of the code. The BIOPROP package has the limitation that it is only possible to construct three-layer networks, but it does have the advantage that it contains a script language for the automation of network experiments (see Ref. 14 for an example script).

The networks used for the variable selection experiments were created using a neural network program written in Borland C++ (v. 3.0), running on a 486 PC clone. The networks were trained using the SuperSAB algorithm, which has been shown to be faster than simple descent. Further details of the network implementation can be found in Refs. 19 and 21. The training protocol involved partition of the data into two sets, a training set and a control set. The training set was used by the network program to adjust the weights of the network and, in general, the network error decreased during training until it reached a constant error (or one that changed by a very small amount) – this point is designated S3. During training the prediction error of the network was assessed by calculation of output values for the control set and it was found that during training the prediction error decreased, reached a minimum (point S1) and then began to increase. A further point, S2, is found for the combined control and training set error. Both S1 and S2 points are found within fewer training cycles than the training minimum error S3 (see Refs. 19 and 21 for illustration of this protocol).

The network displays shown in Fig. 2 were produced by the ReNDeR demonstrator program (AEA Technology, Oxon, U.K.), running under DOS on a 486 PC clone.

Variable selection

One of the most common problems encountered in data modelling is the choice of independent variables for inclusion in the model. Various strategies may be adopted, for example forward inclusion, backward elimination or best subset calculations, in the case of multiple linear regression models, but these all suffer from various disad-

vantages. Perhaps one of the major disadvantages of any variable selection method based on an underlying linear regression model is the fact that the form of the model, in this case linear, is specified in advance. If the data can be fitted well by a linear model, then this type of variable selection may well be appropriate; if the data are better fitted by some nonlinear model, then this may be used as the underlying model for the variable selection process but the problem, of course, rests in the construction of the 'correct' nonlinear model. As stated above, ANN allow models of any complexity to be fitted between the response and descriptors and, furthermore, these need not be specified in advance. A neural network properly (i.e., without overfitting or overtraining) fitted to a data set should make use of the best nonlinear model as dictated by the data itself, and thus a variable selection procedure applied to the ANN model might be expected to extract the most relevant set of variables, at least in terms of modelling the response variable, which is usually the interest in variable selection.

There are a number of variable selection strategies that may be applied to neural network models, known in the neural network field as 'pruning' techniques, since they generally aim to prune unnecessary connections between neurons and hence some of the neurons themselves if all their connections are severed. We have examined the performance of five different pruning algorithms, using both artificially structured data sets (ASDS) and 'real' QSAR data sets, including those mentioned above [21]. Two of these techniques are based on the direct analysis of neuron weights ('magnitude'-based), while the other three take account of a change in network error following the elimination of connections ('error'-based). All five methods gave comparable results for the ASDS, but differences became apparent when they were applied to the real data sets. It was shown that one of the magnitude-based techniques and one of the error-based methods gave the best results for the analysed examples, but that without further systematic investigation of a larger number of data sets it was not possible to say which is the best pruning method.

As an example of the utility of network pruning, we have applied one of the magnitude-based methods (developed by Tetko [21,22]) to a simple chemical system involving the formation of charge-transfer complexes between monosubstituted benzenes and a common electron acceptor, 1,3,5-trinitrobenzene. This data set consists of experimental measurements of complex formation for 35 compounds and physicochemical descriptors for those compounds calculated by molecular mechanics and semi-empirical quantum mechanics [23]. The initial data set contained 58 physicochemical parameters from which the following subset of 11 were selected on the basis of their individual correlations with the formation constant values:

CMR, ClogP, Ehomo, P3, Mux, Sn(1), Sn(2), P1, Fe(4), Mu, Sn(3)

These descriptors are calculated molar refraction (CMR) and log P (ClogP), the energy of the highest occupied molecular orbital (Ehomo), principal ellipsoid axes (P3 and P1), the x component and the magnitude of the dipole moment (Mux and Mu), nucleophilic superdelocalizability for particular ring carbon atoms (Sn()) and frontier electron density for ring carbon 4 (Fe(4)). The order in which the parameters are listed is the order in which they were selected; thus, CMR had the highest correlation with the charge-transfer substituent constant (κ), ClogP the next highest, and so on, with Sn(3) having the lowest correlation with κ of the set of 11.

The network architecture employed was 11:5:1, that is, 11 input neurons for the 11 physicochemical properties, 5 hidden neurons and a single output neuron for κ. The network also employed a bias neuron on the input and hidden layers (see Experimental) and each network was computed 400 times from random starting weights. Table 2 shows how the trained networks may be used to estimate the importance of input variables by a comparison of the results of two magnitude-based methods at early stopping points (S1 and S2) and at the end of network training (S3) – see the Experimental section for details. Variable 11 (Sn(3)) is identified by both techniques at all

TABLE 2
COMPARISON OF THE TETKO [21,22] AND WIKEL [24] METHODS OF PARAMETER SELECTION AT TWO EARLY STOPPING POINTS (S1 AND S2) AND AT COMPLETE TRAINING (S3)

	CMR	ClogP	Ehomo	P3	Mux	Sn(1)	Sn(2)	P1	Fe(4)	Mu	Sn(3)
Tetko Method											
S1	0	0	35	11	18	44	77	40	17	28	130
S2	0	0	26	12	18	49	75	42	11	28	139
S3	0	0	19	4	22	77	65	34	4	16	159
Wikel Method											
S1	0	0	36	9	21	64	73	42	20	20	115
S2	0	0	29	11	20	68	75	41	10	19	127
S3	0	0	20	6	15	82	61	46	5	8	157

The entries indicate the number of networks (from a total of 400 trained) in which that parameter had the lowest sensitivity (least importance).

TABLE 3
NETWORK FITTING RESULTS AND ELIMINATED PROPERTIES FOR THE CHARGE-TRANSFER DATA SET (STOPPING POINT S2)

Parameter											q^2 (S1)	q^2 (S2)
1	2	3	4	5	6	7	8	9	10	11		
											0.95 ± 0.01	0.95 ± 0.009
										x	0.95 ± 0.009	0.95 ± 0.008
					x					x	0.95 ± 0.009	0.95 ± 0.008
					x	x				x	0.95 ± 0.008	0.95 ± 0.008
		x			x	x				x	0.95 ± 0.008	0.95 ± 0.008
		x			x	x	x			x	0.94 ± 0.009	0.94 ± 0.009
		x			x	x	x		x	x	0.94 ± 0.007	0.94 ± 0.008
		x			x	x	x	x	x	x	0.94 ± 0.006	0.94 ± 0.007
		x		x	x	x	x	x	x	x	0.90 ± 0.007	0.89 ± 0.009
		x	x	x	x	x	x	x	x	x	0.90 ± 0.006	0.90 ± 0.009
	x	x	x	x	x	x	x	x	x	x	0.32 ± 0.03	0.30 ± 0.03

Eliminated properties are marked by 'x'. The correlation coefficients (q^2) and their 95% confidence intervals were calculated by leave-one-out cross-validation. Parameters are numbered in the same order as in the text and in Table 2.

three stopping points as the variable with the lowest sensitivity. The next least sensitive variable is found to be 7 (by both methods) at stopping points S1 and S2 and variable 6 at stopping point S3. Although these sensitivity results give some indication of the next variables to remove, the networks were reconstructed, after the elimination of variable 11, and retrained, after which variable 6 was identified as the least sensitive at all three stopping points by both pruning methods. The results of the complete variable selection are shown in Table 3, along with the R^2 value (and standard error) for each network at the early stopping points, S1 and S2. The first row of the table shows the results for a network which contained all 11 properties, the next row shows the results when variable 11 (Sn(3)) was omitted, then variable 6 (Sn(1)), then variable 7 and so on. Elimination of variable 11 in the first step is the expected result, since this variable had the lowest individual correlation with κ, but the next step in the procedure shows that a variable with a higher correlation than four others (7, 8, 9 and 10) is eliminated. Variable 7 (Sn(2)) is the next to be dropped and then, surprisingly, the variable with the third highest correlation with κ (Ehomo). The three 'best' variables chosen by the networks (CMR, ClogP and P3) give a three-term multiple linear regression equation with an R^2 of 0.92; the best three-term equation (found by forward-stepping linear regression) for CMR, ClogP and Ehomo has an R^2 of 0.95 [23]. There is clearly some underlying nonlinear relationship between κ and the physicochemical descriptors and it is this nonlinearity which is presumably responsible for the different order in which variables are selected by the network, although it is possible that collinearity or multicollinearities amongst the descriptors also contribute. A nonlinear model involving the first few parameters chosen by the networks may give a better fit to the κ data than the multiple linear regression equations.

Data display

The lower dimensional display (two or three dimensions) of a multivariate data set, although perhaps not normally considered as data modelling, is an important tool in the construction of QSAR models [20,25]. Standard methods such as principal component analysis (PCA) and nonlinear mapping (NLM) may be used to produce such displays [26], but there is also an interesting neural network technique known as ReNDeR (Reversible Nonlinear Dimension Reduction) which shows some promise [27]. This network operates by 'squeezing' the information in a data set through a bottleneck hidden layer of two or three neurons to reproduce the same values at the output layer as are fed into the input layer. The network consists of five layers* of neurons; an input layer, a first hidden layer (coding), a central layer of two or three neurons, a third hidden layer (decoding) and the output layer. The input and output layers contain the same number of neurons, one for each physicochemical property, and the coding and decoding layers are the same size, usually (but not necessarily) with fewer neurons than the input layer. Coordinates for the lower dimensional display are produced at the central layer of two or three neurons and the encoding and decoding layers allow a nonlinear transformation of the data points from the original high-dimensional data space. Examples have shown that this method can produce displays which are as good, or better, than PCA or NLM [27,28], but an interesting feature of the technique is the possibility of reversing the mapping from two or three dimensions back to the multidimensional space. Because the network is

*The neural network literature is variable on this point. Since the input neurons simply act as distributors of the input signals, they are sometimes not referred to as a layer and thus this would be a four-layer network.

symmetrical, and since all the connection weights are known for a trained network, it is possible to create mapping error contours by projecting a grid of points from the 2D space into the original ND space and then back to 2D. These points 'miss' by varying amounts as they return to the 2D display and from this a set of mapping error contours can be overlaid on the plot [28,29]. Figure 2 shows some ReNDeR displays for the charge-transfer data set, i.e. 2D maps derived from an original 11-dimensional data space. Figure 2a shows the display before any network training has been carried out; the error contours are symmetrical on the picture and increase rapidly from a minimum error contour of 5%. The second part of the figure shows the display after some 50 000 training cycles and here it can be seen that the majority of the points lie within a 1% error contour and thus this is a quite satisfactory 2D picture of the relationships between the compounds in the 11D data space. Figure 2c illustrates that it is possible to see how one of the input variables changes as the cursor is moved over a particular part of the 2D plot. In this case it is the behaviour of the first variable in the set (CMR) and it is clear how such information can be useful when attempts are made to define the properties of, for example, a cluster of active compounds.

Conclusions

Data modelling with neural networks is certainly not 'an answer to the maiden's prayer', but neural networks do offer a number of advantages over some of the more traditional methods of data modelling and should be viewed as a useful adjunct to these techniques. Neural networks can suffer from overfitting and overtraining, but these problems may be avoided by the selection of a suitable architecture and by the use of a training set/control set protocol such as that shown here. They are susceptible to chance effects, as are presumably other 'supervised learning' methods, and the interpretation of network data models is not straightforward. A major advantage of the use of neural networks for data modelling is that they are able to fit complex nonlinear models and these models do not have to be specified in advance, unlike other nonlinear modelling techniques. Neural networks as a variable selection tool may give a quite different insight into a data set, as shown here for the set of charge-transfer data. Finally, neural networks may be used to 'view' high-dimensional data, with the advantage that it is possible to move back from the 2D display to the original multivariate data space.

Acknowledgements

The authors are grateful to Dr. M.R. Saunders (SmithKline Beecham) for modifications to the BIOPROP code.

References

1 Eberhart, R.C. and Dobbins, R.W., Neural Network PC Tools, Academic Press, San Diego, CA, U.S.A., 1990, p. 1.
2 McCulloch, W.C. and Pitts, W., Bull. Math. Biophys., 5 (1943) 115.
3 Minsky, M. and Papert, S., Perceptrons, MIT Press, Cambridge, MA, U.S.A., 1969.
4 Livingstone, D.J., Data Analysis for Chemists: Applications to QSAR and Chemical Product Design, Oxford University Press, Oxford, U.K., 1995, p. 183.
5 Manallack, D.T. and Livingstone, D.J., In Van de Waterbeemd, H. (Ed.) Advanced Computer-Assisted Techniques in Drug Design, VCH, Weinheim, Germany, 1994, pp. 293–318.
6 Livingstone, D.J. and Salt, D.W., In Dean, P.M. (Ed.) Molecular Similarity in Drug Design, Blackie Academic & Professional, Glasgow, U.K., 1995, pp. 187–214.
7 Aoyama, T., Suzuki, Y. and Ichikawa, H., J. Med. Chem., 33 (1990) 905.
8 Aoyama, T., Suzuki, Y. and Ichikawa, H., J. Med. Chem., 33 (1990) 2583.
9 Andrea, T.A. and Kalayeh, H., J. Med. Chem., 34 (1991) 2824.
10 Salt, D.W., Yildiz, N., Livingstone, D.J. and Tinsley, C.J., Pestic. Sci., 36 (1992) 161.
11 Topliss, J.G. and Edwards, R.P., J. Med. Chem., 22 (1979) 1238.
12 Manallack, D.T. and Livingstone, D.J., Med. Chem. Res., 2 (1992) 181.
13 Livingstone, D.J. and Manallack, D.T., J. Med. Chem., 36 (1993) 1295.
14 Manallack, D.T., Ellis, D.D. and Livingstone, D.J., J. Med. Chem., 37 (1994) 3758.
15 Campbell, J.L.E. and Johnson, K.E., Can. J. Chem., 71 (1993) 1800.
16 Lewis, D.F.V., J. Comput. Chem., 10 (1989) 145.
17 Smolensky, P., Artif. Intel. Rev., 1 (1987) 95.
18 Goonatilake, S. and Khebbal, S. (Eds.) Intelligent Hybrid Systems, Wiley, Chichester, U.K., 1995.
19 Tetko, I.V., Livingstone, D.J. and Luik, A.I., J. Chem. Inf. Comput. Sci., 35 (1995) 826.
20 Livingstone, D.J., Data Analysis for Chemists: Applications to QSAR and Chemical Product Design, Oxford University Press, Oxford, U.K., 1995, pp. 65–89.
21 Tetko, I.V., Villa, A.E.P. and Livingstone, D.J., J. Chem. Inf. Comput. Sci., 36 (1996) 794.
22 Tetko, I.V., Tanchuk, V.Yu., Chentsova, N.P., Antonenko, S.V., Poda, G.I., Kukhar, V.P. and Luik, A.I., J. Med. Chem., 37 (1994) 2520.
23 Livingstone, D.J., Evans, D.A. and Saunders, M.R., J. Chem. Soc. Perkin Trans. II, (1992) 1545.
24 Wikel, J.H. and Dow, E.R., Bioorg. Med. Chem. Lett., 3 (1993) 645.
25 Livingstone, D.J., Methods Enzymol., 203 (1991) 613.
26 Hudson, B., Livingstone, D.J. and Rahr, E., J. Comput.-Aided Mol. Design, 3 (1988) 55.
27 Livingstone, D.J., Hesketh, G. and Clayworth, D., J. Mol. Graph., 9 (1991) 31.
28 Livingstone, D.J., In Devillers, J. (Ed.) Neural Networks in QSAR and Drug Design, Academic Press, London, U.K., 1996, pp. 157–176.
29 Livingstone, D.J., In Sanz, F., Giraldo, J. and Manaut, F. (Eds.) QSAR and Molecular Modelling: Concepts, Computational Tools and Biological Applications, J.R. Prous, Barcelona, Spain, 1995, pp. 18–26.