Analysis of Linear and Nonlinear QSAR Data Using Neural Networks

David T. Manallack,*,† Dianne D. Ellis, and David J. Livingstone

SmithKline Beecham Pharmaceuticals, The Frythe, Welwyn, Herts, AL6 9AR U.K.

Received February 15, 1994®

The use of feed forward back propagation neural networks to perform the equivalent of multiple linear regression has been examined using artificial structured data sets and real literature data. Their predictive ability has been assessed using leave-one-out cross-validation and training/test set protocols. While networks have been shown to fit data sets well, they appear to suffer from a number of disadvantages. In particular, they have performed poorly in prediction for the QSAR data examined here, they are susceptible to chance effects, and the relationships developed by the networks are difficult to interpret. This investigation reports results for one particular form of artificial neural network; other architectures and applications, however, may be more suitable.

Introduction

The use of neural networks in the field of chemistry has grown considerably[1] since their first applications were described for process control[2] and protein secondary structure prediction.[3,4] Applications have now been found for the analysis of spectra, prediction of chemical reactivity, electrostatic potentials, and quantitative structure–activity relationships (QSAR) (see ref 5). Networks have been able to perform established computing tasks as well as tackling complex problem solving using their pattern recognition abilities.

One of the most interesting uses of neural networks in chemistry is their application in structure–property correlations (SPC) and QSAR (see reviews 1, 6–8). A considerable amount of research has been performed to refine the methodology, including the development of improved network architectures and training algorithms.[9–16]

One of the earliest recognized problems using neural networks for QSAR data analysis was the phenomenon of over-fitting.[9] Using a series of dihydrofolate reductase inhibitors, Andrea and Kalayeh[9] suggested that the ratio of the number of compounds to network connections (defined as the parameter ϱ‡) should be greater than 1.0, and in their study they found that the optimal value for ϱ was 2.0. Models with a ϱ value greater than 2.2 performed poorly in predictions, presumably because they lacked sufficient connections to be able to develop the relevant rules. Models employing ϱ values less than 1.8, on the other hand, appeared to overfit the data since they explained the training set well but were poor in prediction.

Our own investigations into the use of neural networks for QSAR data analysis have concentrated primarily on the problem of chance effects.[8,10,12] These concerns were raised from some early uses of networks for QSAR where the number of connections in the network far exceeded the number of compounds under consideration.[17,18] It was possible that the presence of too many connections in a network may not only allow chance correlations to occur but may also result in overfitting. To investigate this we used random numbers as input data for a series of simulated QSAR data analyses using networks. The results demonstrated that the networks were able to train to successfully reproduce the values of a random target. The network was in effect, "memorizing" these data, and guidelines were suggested to minimize chance effects and optimize the performance of the technique. These guidelines, however, were generated using random numbers. This is, of course, a nonideal situation as real data are structured and the predictive nature of a trained network cannot be assessed using random numbers. This report describes our investigations into the use of neural networks to perform multiple linear regression (MLR) using structured data files and four previously published QSAR studies. Our intentions for this work were 2-fold. Firstly, we hoped to provide guidelines for QSAR data analysis using neural networks based on the properties of structured data and real data. These guidelines should be more relevant than those developed using random numbers.[10,12] Our second aim was to assess the extent to which neural networks are able to carry out regression analysis.

Methods

The data sets used in this work comprised artificial "structured" data and "real" data abstracted from the literature as described below:

Structured Data Sets. All structured data sets were generated to have four input variables (independent variables) and a target value (dependent variable). While we have used the term "structured" here, this indicates a relationship between the input variables and the target. Unlike real data sets, there are virtually no relationships between the independent variables themselves. The target was generated from the input variables to have a specific structure (e.g., linear or inclusion of a quadratic term, etc.). In some cases not all the input variables were used to generate the target, and where an indicator variable was required, this was created using a separate set of random numbers (see below). In addition, "noise" was included by adding a small random number to the target so as to generate data sets which did not have a perfect correlation between the dependent and independent variables, i.e., the networks would have the opportunity to perform

† Present address: Chiroscience, No. 283 Cambridge Science Park, Milton Road, Cambridge, CB4 4WE U.K. Telephone: 0438 782102. Fax: 0438 782550.

‡ The character rho is different than in previous publications due to a new PostScript character set.

® Abstract published in *Advance ACS Abstracts,* September 15, 1994.

0022-2623/94/1837-3758$04.50/0

better or worse than MLR. The four independent variables consisted of random numbers scaled between 0.2 and 0.8 generated using the RS1 data analysis package (BBN software, Staines, UK). Following the generation of the target variable this was then scaled between 0.2 and 0.8. Output values from the BIOPROP package[19] fall in the range 0.0−1.0; thus, scaling the target between 0.2 and 0.8 allows the network to extrapolate beyond the scaled range. For each of the four experiments listed below, five data files were generated consisting of 65 cases (i.e., compounds). The first 50 cases were used for training and cross-validation purposes, and the remaining 15 were used as a test set. Results described here are the average of the five files for each of the following four simulated QSAR scenarios.

Linear. All four variables were used to generate the target variable. Noise was added to keep the relationship between the four independent variables and the target to an R^2 of about 0.85, i.e.,

$$\text{target} = V_1 \cdot w + V_2 \cdot x + V_3 \cdot y + V_4 \cdot z + \text{noise}$$

where V_1, V_2, V_3, V_4 are the four independent variables; w, x, y, z are the coefficients; and noise is a random number.

Indicator. Three variables plus an indicator (1 or 0) were employed to generate the target. The files used for network training did not, however, include the indicator variable. One feature of networks is their ability to replicate the brain's capacity for pattern recognition. Information processing by networks has demonstrated that they can learn complex "nonlinear" relationships for problem solving, and this is seen as one of their major advantages. The omission of the indicator variable in this experiment is intended to investigate this claim with particular regard to the analysis of QSAR data sets.[8] To maintain the input of four independent variables, a separate column of random numbers was used in place of the indicator variable. Noise was added to keep the relationship between the indicator and three independent variables and the target to an R^2 of about 0.85, i.e.,

$$\text{target} = V_1 \cdot w + V_2 \cdot x + V_3 \cdot y + \text{indicator} \cdot z + \text{noise}$$

[The indicator was generated by using a separate set of 65 random numbers ranged between 0.0 and 1.0. If a number fell below 0.5, the indicator was given a value of 0.0, and if the number was 0.5 or greater, then the indicator was given a value of 1.0.]

Quadratic. The target was generated using three variables which included the square of one of the variables. For training the quadratic term was not included (i.e., only the original column of random numbers and not the square of this column). To keep the number of input variables to four, an additional two sets of random numbers were included for network training, i.e.,

$$\text{target} = V_1 \cdot w + (V_1)^2 \cdot x + V_2 \cdot y + \text{noise}$$

Quadratic plus Indicator. The target was generated using four variables which includes the square of one of the variables and an indicator variable. For training purposes the indicator was not used, and to keep four inputs, an additional column of random numbers was included for input to network training. Noise was added to keep the relationship between the indicator and three independent variables, and the target to an R^2 of about 0.83,

$$\text{target} = V_1 \cdot w + (V_1)^2 \cdot x + V_2 \cdot y + \text{indicator} \cdot z + \text{noise}$$

[The indicator was generated as above.]

Table 1. MLR Results Using Real QSAR Data Sets[a]

no.	equation	n	R^2	s	F	cross-validated R^2
	Linear[22]					
	$\log P_{expt} = 0.40\alpha_{calc} - 0.46\mu + 0.33E(\text{HOMO}) - 6.06$	37	0.826	0.6956	52.10	
1	$\log P_{expt} = 0.412\alpha_{calc} - 0.359\mu + 0.384E(\text{HOMO}) - 7.115$	37	0.847	0.609	60.8	0.784
	Indicator[21]					
	$\log(1/C) = 0.45\pi + 1.05I - 0.48\text{MR}_Y$[b]	38	0.929	0.264	17.1	
2	$\log(1/C) = 0.424\pi + 1.090I - 0.495\text{MR}_Y + 3.374$	38	0.931	0.259	151.9	0.916
3	$\log(1/C) = 0.424\pi + 0.165\text{MR}_Y + 3.494$	38	0.835	0.393	88.87	
	Quadratic[20]					
	$\log(1/D_{40}) = -1.40R_m^2 - 0.42R_m + 0.71\text{p}K_a + 0.39$	50	0.828	0.25	67.9	
4	$\log(1/D_{40}) = -1.42R_m^2 - 0.43R_m + 0.70\text{p}K_a + 0.36$	50	0.821	0.252	70.2	0.789
5	$\log(1/D_{40}) = -1.06R_m + 0.73\text{p}K_a + 0.02$	50	0.709	0.317	57.23	
	Quadratic plus Indicator[21]					
	$\log(1/C) = 0.82\pi'_3 - 0.11\pi'^2_3 - 0.97\text{MR}_Y + 0.91I + 4.47$	34	0.878	0.343	13.3	
6	$\log(1/C) = 0.84\pi'_3 - 0.11\pi'^2_3 - 0.97\text{MR}_Y + 0.96I + 4.40$	34	0.837	0.420	37.1	0.781
7	$\log(1/C) = 0.34\pi'_3 - 0.03\text{MR}_Y + 4.47$	34	0.430	0.759	11.7	

[a] Equations 1, 2, 4, 6 represent our own work repeating the MLR analyses reported previously. In these examples, small differences were found in the coefficients and constants of the original equations to our own calculations. These often small differences may be due to computer rounding errors or to typographical errors in the original paper (data tables were carefully checked against the originals to avoid errors). [b] The constant value was not documented; presumably an omission.

QSAR Data Sets. The real QSAR data sets chosen for this study represent a number of different, commonly encountered, QSAR models.[20-22] These examples were chosen because they represented a particular form of QSAR model (linear, linear with indicator, quadratic, etc.), data was available in the original report and the data set was of a 'useful' size. Table 1 lists each of the four data sets along with the originally published MLR equations and cross-validated correlation coefficients calculated using a leave-one-out (LOO) procedure. These four examples contain variously structured data sets ranging from linear to a quadratic equation including an indicator variable.

Neural Network Implementation. Neural networks were constructed using the BIOPROP program[19] which employs a command language allowing the automation of network simulation (input/output/training/saving, etc.) by the use of script files. All networks

Chart 1

```
!       EXAMPLE BIOPROP TRAINING SCRIPT (Non-Interactive Session)
! Comments are placed after a "!" symbol
! Training of the dataset.  Effect of changing the number of hidden layer units


set ninputs  4              ! number of input units  = 4
set noutputs 1              ! number of outputs units  = 1
set $data filename          ! $data = file containing input/target information
set #maxiter 5000           ! maxiter = maximum number of training cycles



openf output_results_file   ! open file to contain output results
log logfile.dat             ! a file to log the commands used
readp $data 1               ! open file containing input/target information
printf $data                ! print filename into output_results_file

! loop to alter the number of hidden layer units from 1 to 10

for #hidden 1 10

 set #bestcorr 0            ! initialize value of variable bestcorr

 set nhidden #hidden        ! set number of hidden layer units to variable
!                            #hidden

! loop to initialize the weights 5 times (i.e training for each hidden layer will run
! through this loop to initialize the connection weights each time)

   for #init 1 5            ! loop 5 times to initialize weights
!                            except first run through

      if #init gt 1.0
        initwts             ! initwts initializes the connection weights to
                            ! small random values
      endif

! TRAIN  The following steps train the network keeping a record of the
! best results.  Pertubation of the connection weights is performed as well as
! restarting the networks to find the global minimum.
   trainc #maxiter
   testf $data 0 1
! corr[1] is a variable (internal) for the correlation between the output and target
   if corr[1] gt #bestcorr
    set #bestcorr corr[1]
   endif

! SMALL PERTUBATION OF THE CONNECTION WEIGHTS  (i.e shake 0.2)

   shake 0.2
! TRAIN
   trainc #maxiter
   testf $data 0 1

   if corr[1] gt #bestcorr
    set #bestcorr corr[1]
   endif

! MODERATE PERTUBATION OF CONNECTION WEIGHTS

   shake 1.5
! TRAIN
   trainc #maxiter
   testf $data 0 1

   if corr[1] gt #bestcorr
    set #bestcorr corr[1]
   endif

! SMALL PERTUBATION OF THE CONNECTION WEIGHTS

   shake 0.2
! TRAIN
   trainc #maxiter
   testf $data 0 1

   if corr[1] gt #bestcorr
    set #bestcorr corr[1]
   endif

! LARGE PERTUBATION OF THE CONNECTION WEIGHTS

   shake 3.0
! TRAIN
   trainc #maxiter
   testf $data 0 1

   if corr[1] gt #bestcorr
    set #bestcorr corr[1]
   endif

! SMALL PERTUBATION OF THE CONNECTION WEIGHTS

   shake 0.2
! TRAIN
   trainc #maxiter
   testf $data 0 1

   if corr[1] gt #bestcorr
    set #bestcorr corr[1]
   endif

   next                 ! end of loop for #init to initialize weights

! square the best result to give the correlation coefficient

 pow #b #bestcorr 2.0               ! variable #b contains square of #bestcorr

 printf  hidden layer units #hidden  R^2  #b     ! print correlation coefficient
!                                                  to output_results_file

next            ! end of loop for incrementing number of hidden layer units

closef          ! close the output_results_file

log             ! close logging of session

quit            ! finish BIOPROP session
```

were trained using the back propagation (BP) method using a conjugate gradient minimizer. Traditional BP usually employs a simple gradient descent technique[23] to reach convergence. The conjugate gradient method used by BIOPROP provides faster network training compared to traditional BP methods. Hertz and coworkers[24] provide a theoretical discussion of the implementation of the conjugate gradient method to neural network training. Several detailed descriptions of the general theory behind neural networks have already been published, and these can be found in refs 9 and 23–25.

Training Protocols. One problem that has been encountered during network training is that they can fall into so called "local minima". Attempts to overcome this problem include perturbation of the weights connecting the units in the network, followed by further training, or reinitialization, of the connection weights and retraining. A simple check on the total error will indicate which minimum is the lowest. The training procedures employed here used both approaches to try and locate the global minimum (Chart 1). While there can be no certainty that the final model computed is the global minimum (if indeed one exists), it is felt that this approach was sufficiently exhaustive. At the completion of training, the target and output results from the network were compared to provide a correlation coefficient.

Neural networks were constructed employing a single hidden layer and one output unit to perform the equivalent of regression analysis. For each data set the number of hidden units was varied to examine the effect of changing ϱ on the correlation coefficient. This involved a hidden layer containing 1 unit up to a maximum of 16 corresponding to a minimum ϱ value of about 0.6. In three of the four QSAR data sets (both structured and real), either an indicator variable or a squared term or both were present in the original equations. These "nonlinear" terms were not included for network training. The removal of these properties was aimed at allowing the network to exploit its ability to develop "nonlinear" relationships without these being specifically stated.

Cross-Validation. In addition to simple training procedures, cross-validation was used to monitor predicitive performance. Both leave-one-out (LOO) and leave-*N*-examples-out (*cf.* ref 26) procedures were conducted. For the random number structured data sets LOO was not performed as the computation time required was excessive (n.b. the results presented here are the average of five separate data sets). A leave-*N*-out procedure was employed setting *N* to 5 (i.e., leave-10%-out) while the real data sets were split into 10 approximately equal groups. In the case of the four real QSAR studies, choice of the compounds for each group was based on hierarchical clustering of the compounds as described by the physicochemical parameters. Hierarchical clustering was performed using the multivariate statistics package ARTHUR (Infometrix, Inc., Seattle, WA). A similarity level was chosen which split the compounds into four clusters, and representatives from each were selected to create the 10 groups for testing. Training and testing were continued until each group had been left out (once) for test purposes.

Training/Test Sets. Another, and perhaps more useful, method of determining the predictive ability of a network is to leave out a number of compounds for test purposes. For the structured data files, "compounds" **51–65** were used for testing. Hierarchical cluster analysis was used to select test compounds from the real QSAR data sets (Tariq Andrea, personal communication). Typically between 40 and 50% of the compounds were removed from the data sets to create test sets. A similarity level was used which split the compounds into x clusters, where x is the number of test compounds to be chosen. Representatives of each cluster were then removed for testing. As one might expect, these clusters often contain more than one compound, for example Figure 1 shows a section of the dendrogram for the quadratic data set. Network training for these data gave some unusual results (see below) so that a second training/test set was generated as indicated in the figure. A simple comparison of the actual target values (dependent variables) and the network outputs provided a correlation coefficient for these test sets.

Results and Discussion

Our previous report on the use of neural networks for QSAR data analysis gave guidelines based on results using random numbers.[12] In that study the data sets used had 50 cases (i.e., 50 compounds) of five random variables comprising four independent variables and one dependent variable. At ϱ values of less than 2.0 the correlation coefficients obtained were above 0.74. This clearly illustrated that, since the random numbers were not related to one another, the networks were able to fit complex but meaningless relationships between "independent" and "dependent" variables. In order to provide more relevant guidelines, and to assess the performance of networks as "regression engines", we report the following results.

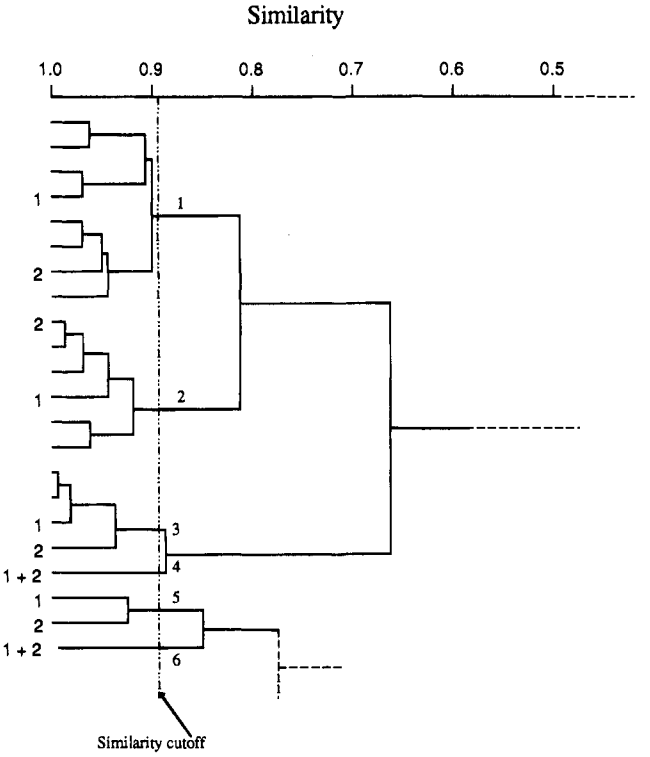

Figure 1. Diagram illustrating section of a dendrogram produced by cluster analysis for the real quadratic QSAR data set.[20] Compounds are clustered according to their similarity (six clusters are shown at the indicated similarity cutoff), and membership of the two test sets are shown (1 or 2).

Table 2. Averaged Correlation Coefficient of Structured Data Files Using MLR[a]

structure	R^2 (first 50 cases)	R^2 (all 65 cases)
linear	0.847	0.854
indicator	0.837	0.843
quadratic	0.843	0.849
quadratic plus indicator	0.823	0.830

[a] Results are the average of five data files for each structural type.

Structured Data Files. The relationship between the independent variables and the target variable of the structured data sets was verified using traditional MLR. Table 2 shows the average correlation coefficient of the five data sets for each of the four data structures examined. These results demonstrate that the 50 training compounds have an R^2 value of approximately 0.84 which does not differ significantly to the results for the entire data sets containing 65 cases (i.e., the 50 training compounds plus 15 test set compounds).

Linear. Figure 2 illustrates that network training using a single hidden layer unit (ϱ = 7.14) gave a correlation coefficient equivalent to MLR (0.847). As hidden layer units were added (i.e., ϱ decreases), the correlation coefficient gradually increased from this level to a value above 0.9. Predictive performance, on

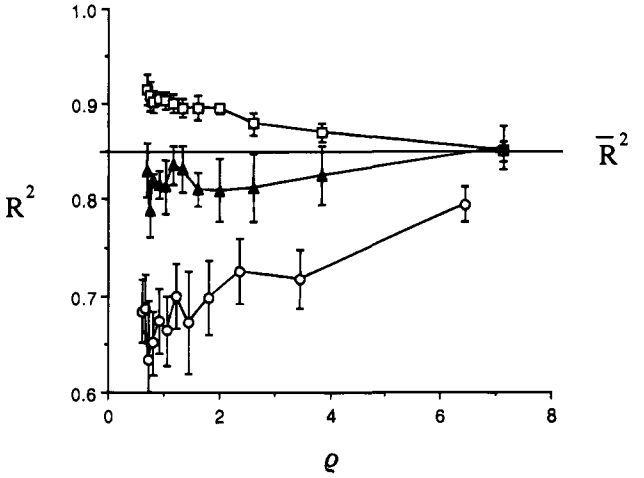

Figure 2. Plot illustrating the effect of changing ϱ on the correlation coefficient for the linear structured data sets. Each point is the average of five data sets, and the standard error of the mean is given by the error bars. Training for the 50 cases is shown as the top curve (□). Two curves show the predictive performance of the network using a leave-N-out (○) cross-validation procedure (N = 10%) and the results of the 15 test cases (▲). The averaged correlation coefficient (first 50 cases) for the five data sets determined using MLR (Table 2) is represented as the horizontal line indicated with $\bar{R}^2$.

the other hand, as measured by cross-validation and the training/test set procedure decreased as ϱ decreased. The apparent improvement in fitting as ϱ decreases is in accord with our previous findings with random numbers.[12] The advantage of using structured data sets such as these is that the predictive ability of the trained networks may be assessed and, as we suspected,[12] performance decreases with increasing connections. It was interesting to note that the leave-N-out cross-validation results gave considerably lower correlation coefficients than the 15 test compounds.

Indicator. Network training for the indicator data sets demonstrated a steady increase in the correlation coefficient from 0.511 to a value approaching 0.9 as ϱ decreased (Figure 3). At a ϱ value of 1.0 the network had matched the correlation coefficient obtained using MLR. The results from the leave-N-out cross-validation and the 15 test set compounds fell below 0.5 and decreased steadily as more hidden layer units were added. These results show that given sufficient connections the network was able to perform adequately in training to match the required target. The results testing the predictive ability of the network clearly demonstrate that it is doing very poorly. This is due to the network being unable to determine the value of the indicator for the test compounds. These indicators were generated using a separate column of random numbers and as such no clues (i.e., interproperty correlations) are built in to the data set for the network to develop learning rules.

Quadratic. Like the results obtained for the linear data sets, the correlation coefficient for network training began at the value obtained by MLR and increased steadily to a value above 0.9 as ϱ decreased (Figure 4). For the 15 test compounds the network performed reasonably well in prediction, however, the leave-N-out cross-validation results showed poor predictive ability. Good test set predictions by these networks is presumably due to the fact that they have learned the quadratic

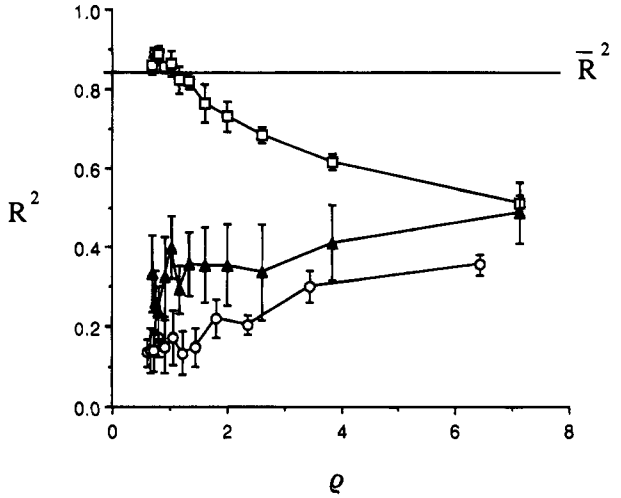

Figure 3. Plot illustrating the effect of changing ϱ on the correlation coefficient for the indicator structured data sets. Each point is the average of five data sets, and the standard error of the mean is given by the error bars. Training for the 50 cases is shown as the top curve (□). Two curves show the predictive performance of the network using a leave-N-out (○) cross-validation procedure (N = 10%) and the results of the 15 test cases (▲). The averaged correlation coefficient (first 50 cases) for the five data sets determined using MLR (Table 2) is represented as the horizontal line indicated with $\bar{R}^2$.

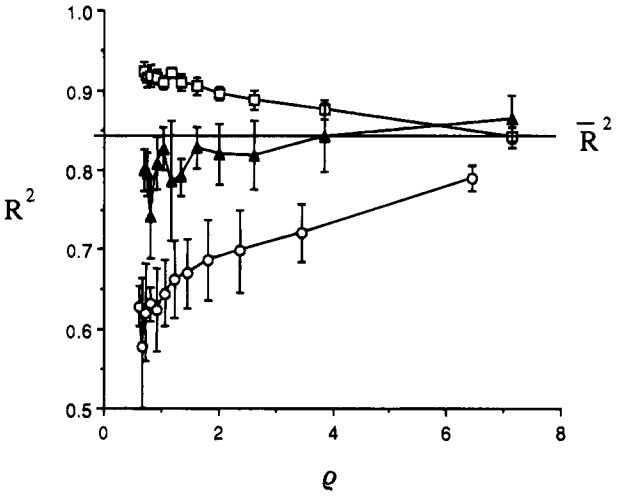

Figure 4. Plot illustrating the effect of changing ϱ on the correlation coefficient for the quadratic structured data sets. Each point is the average of five data sets and the standard error of the mean is given by the error bars. Training for the 50 cases is shown as the top curve (□). Two curves show the predictive performance of the network using a leave-N-out (○) cross-validation procedure (N = 10%) and the results of the 15 test cases (▲). The averaged correlation coefficient (first 50 cases) for the five data sets determined using MLR (Table 2) is represented as the horizontal line indicated with $\bar{R}^2$.

predictive rule, in the case of the linear data sets the networks have learned additive rules.

Quadratic plus Indicator. Network training increased from a value of 0.6 for one hidden unit to values above that obtained using MLR as ϱ decreased (Figure 5). The network performed extremely poorly in prediction for both cross-validation and the 15 test set compounds. No doubt the poor performance is again due to the network being unable to predict the value of the indicator which was generated from a separate set of random numbers.

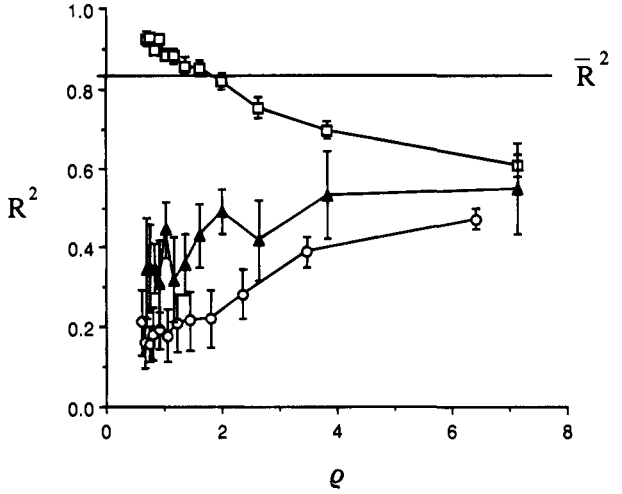

Figure 5. Plot illustrating the effect of changing ϱ on the correlation coefficient for the quadratic plus indicator structured data sets. Each point is the average of five data sets and the standard error of the mean is given by the error bars. Training for the 50 cases is shown as the top curve (□). Two curves show the predictive performance of the network using a leave-*N*-out (○) cross-validation procedure (N = 10%) and the results of the 15 test cases (▲). The averaged correlation coefficient (first 50 cases) for the five data sets determined using MLR (Table 2) is shown as the horizontal line indicated with $\bar{R}^2$.

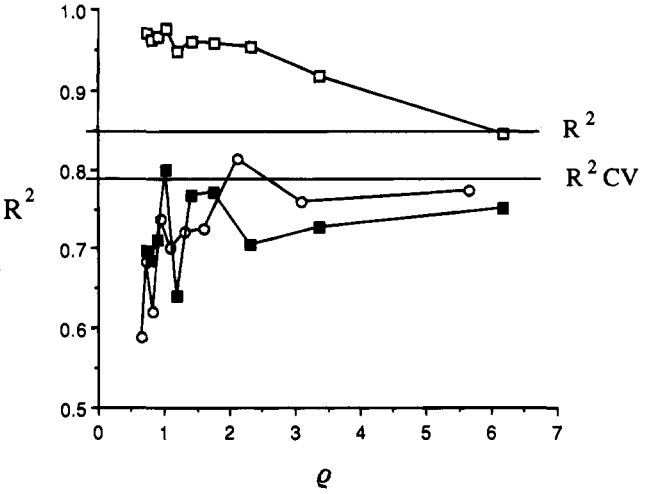

Figure 6. Plot showing the results of multiple linear regression using traditional statistics and neural networks for the "linear" QSAR data set.[22] The correlation coefficient has been plotted against ϱ. The top curve represents training of the data using networks employing a 3-n-1 architecture with the data scaled between 0.2 and 0.8 (□). Testing the predictive performance of the networks used cross-validation employing both LOO (■) and leave-*N*-out (○) procedures (N was approximately 10%). The horizontal lines represent the results obtained using traditional statistics listed in Table 1. The cross-validation result using traditional statistics used a LOO procedure.

Fitting random number data sets using neural networks is easy because the data contains no "real" structure. This is why structured data sets were examined. The structural data sets, however, suffer from the problem that although they contain the artificially imposed relationships between the input variables and the target, they do not contain the "natural" relationships that occur between the input variables in a real data set. This study has therefore chosen a number of real QSAR data sets to examine.

Real QSAR Examples. For each of the real QSAR data sets, network training demonstrated that the correlation coefficient increases with the addition of hidden layer units (Figures 6–9). In only two cases did the networks result in correlation coefficients higher than those obtained using MLR (Figures 6, 9, eqs 1, 6, Table 1). In the linear example, ϱ values less than 6.0 provided R^2 values exceeding that using MLR, while the quadratic plus indicator data set exceeded the MLR correlation coefficient at ϱ values less than 3.0. It should be remembered that in our training procedures the squared terms and indicator variables were not provided as input to network training. The networks have therefore found the rules associated with the "nonlinear" relationships of the input properties to the target data. To emphasize these results, the MLR equations were also generated without the square or indicator terms (equations, 3, 5, 7, Table 1), and these values have been indicated on Figures 7–9. As can be seen from these figures, in all three cases the networks have performed well at ϱ values less than 8.0 and far exceed the fit using MLR. One major difference between the structured and real data sets is that indicator variables from real QSAR studies are used to mark compounds from different structural classes or containing particular structural features and as such represent "real data". The purpose of our investigations has been to examine the claims that neural networks are able to

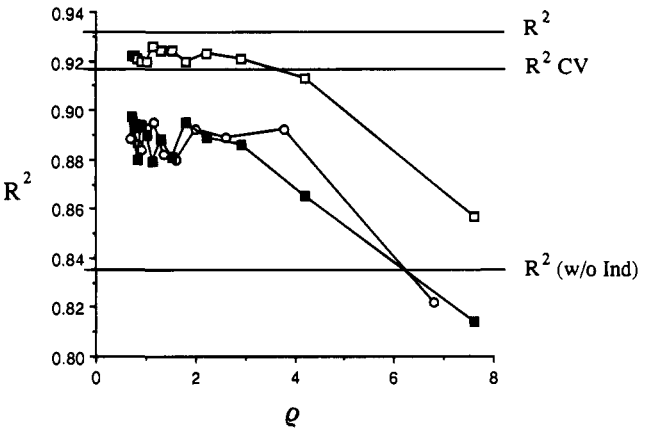

Figure 7. Plot showing the results of multiple linear regression using traditional statistics and neural networks for the "indicator" QSAR data set.[21] The correlation coefficient has been plotted against ϱ. The top curve represents training of the data using networks employing a 2-n-1 architecture with the data scaled between 0.2 and 0.8 (□). Testing the predictive performance of the networks used cross-validation employing both LOO (■) and leave-*N*-out (○) procedures (N was approximately 10%). The horizontal lines represent the results obtained using traditional statistics listed in Table 1. In addition, a line has been drawn showing the correlation coefficient in which the indicator variable was omitted. The cross-validation result using traditional statistics used a LOO procedure.

perform statistical tasks better than traditional techniques, and thus we have withheld indicator variables and quadratic terms from the neural networks. We have not trained networks which have included these terms since our expectation is that they would perform well at low values of ϱ (i.e., similar to the linear "real" case).

Although the training results outperformed MLR in two cases, this does not give any indication of the

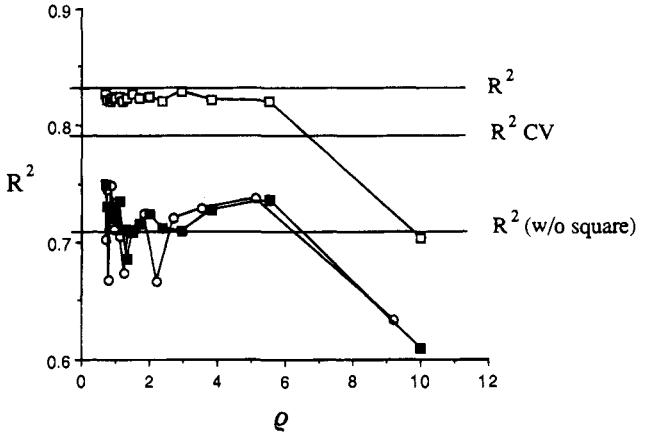

Figure 8. Plot showing the results of multiple linear regression using traditional statistics and neural networks for the "quadratic" QSAR data set.[20] The correlation coefficient has been plotted against ϱ. The top curve represents training of the data using networks employing a 2-n-1 architecture with the data scaled between 0.2 and 0.8 (□). Testing the predictive performance of the networks used cross-validation employing both LOO (■) and leave-N-out (○) procedures ($N = 10\%$). The horizontal lines represent the results obtained using traditional statistics listed in Table 1. In addition, a line has been drawn showing the correlation coefficient in which the quadratic variable was omitted. The cross-validation result using traditional statistics used a LOO procedure.

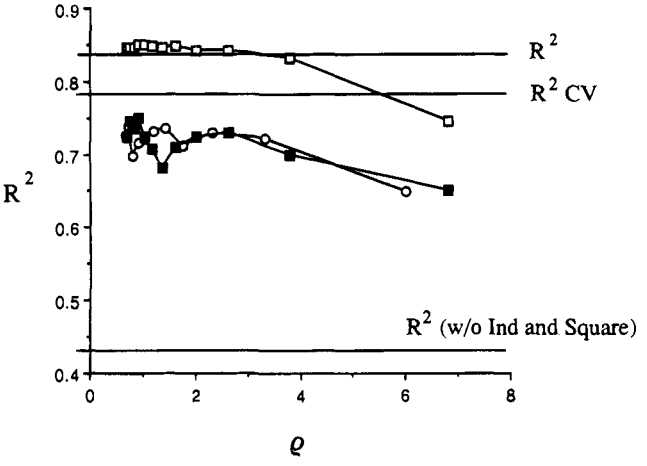

Figure 9. Plot showing the results of multiple linear regression using traditional statistics and neural networks for the "quadratic plus indicator" QSAR data set.[21] The correlation coefficient has been plotted against ϱ. The top curve represents training of the data using networks employing a 2-n-1 architecture with the data scaled between 0.2 and 0.8 (□). Testing the predictive performance of the networks used cross-validation employing both LOO (■) and leave-N-out (○) procedures (N was approximately 10%). The horizontal lines represent the results obtained using traditional statistics listed in Table 1. In addition, a line has been drawn showing the correlation coefficient in which the indicator and quadratic variables were omitted. The cross-validation result using traditional statistics used a LOO procedure.

predictive ability of the network. As for the structured data sets, cross-validation provides some clues to predictive power, giving an indication of how well the network predicts when N cases are left out of the analysis. In this study we have performed two cross-validation experiments where $N = 1$ and $N =$ approximately 10%. These results are shown in Figures 6–9. Both methods gave similar results, and in only the linear case did correlation coefficients rise above

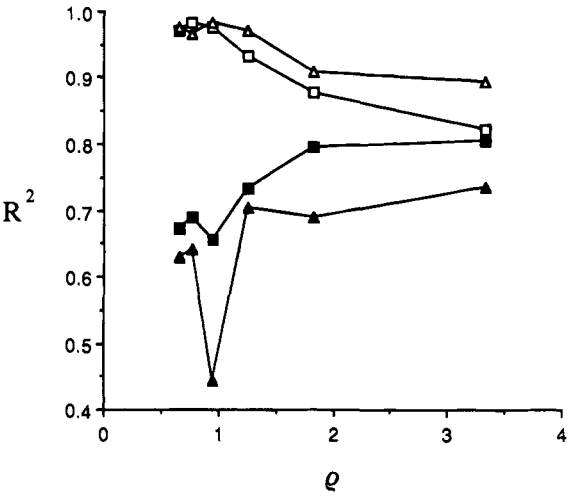

Figure 10. Diagram showing the training and test set results using neural networks for the "linear" data set. Two trials were conducted using separate sets of 17 test compounds removed from the data set. Results are shown as the correlation coefficient plotted against ϱ for training, trail 1 (□) and trial 2 (△). Using these trained networks, predictions of activity were made for the relevant test set which are also shown as the correlation coefficient, trial 1 (■) and trial 2 (▲).

that obtained using MLR. The linear results also differed from the nonlinear examples as the cross-validated R^2 increased with additional hidden layer units. In all cases the behavior of the cross-validated results was variable at low values of ϱ. Thus, although the networks performed well in training, they did poorly in prediction as measured by cross-validation. Arguments may be put forward which suggest that these results are a consequence of over-training and thus highlight a shortcoming of the use of networks to perform MLR. Conversely, networks are able to fit the more complex data sets without the need to specify nonlinear terms and appear to predict better than the regression models without these variables.

A more effective way of testing predictive ability is to use a train/test set procedure. The generation of test and training sets requires careful consideration as both sets should be representative of the entire data set. Cross-validation, on the other hand, leaves every compound in the set out once for testing, and unless the data set is particularly well-behaved, in terms of the disposition of compounds in parameter space, cross-validation will select some very unsuitable (i.e., outlier) compounds. In this work, hierarchical cluster analysis was used to choose representative test compounds. To illustrate this, Figure 1 shows a section of a dendrogram highlighting the compounds chosen for testing. Where more than one compound was present in a cluster, the choice of a representative test compound was arbitrary. Results for both the training and test sets using a network regression are shown in Figures 10–13. The linear example gives comparable results to the structured linear data sets (Figure 2) in that for trials 1 and 2 the correlation coefficient increases as the number of hidden units is incremented (Figure 10). Similarly, the trend for the test results is a reduction in correlation coefficient as ϱ decreases.

Training for the indicator and quadratic plus indicator data sets also shows a gradual increase in R^2 as ϱ decreases (Figures 11 and 13). The test results are more erratic and in a number of places the test results rose

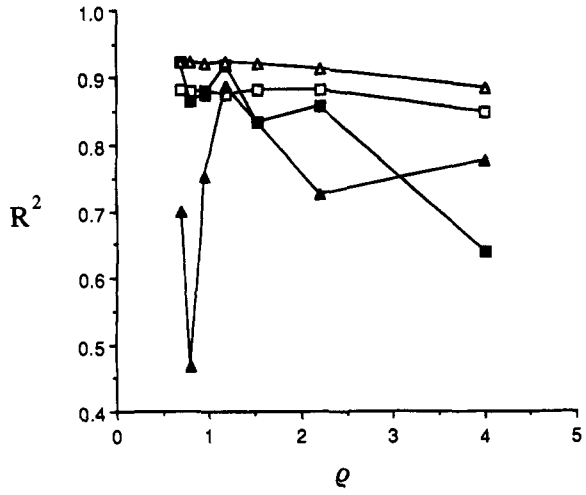

Figure 11. Diagram showing the training and test set results using neural networks for the "indicator" data set. Two trials were conducted using separate sets of 18 test compounds removed from the data set. Results are shown as the correlation coefficient plotted against ϱ for training, trial 1 (□) and trial 2 (△). Using these trained networks, predictions of activity were made for the relevant test set which are also shown as the correlation coefficient, trial 1 (■) and trial 2 (▲).

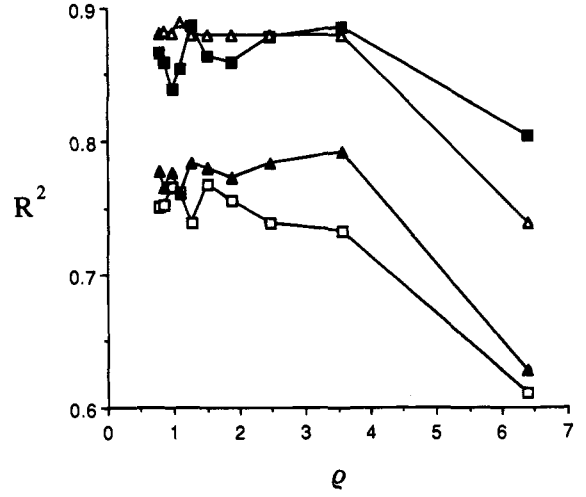

Figure 12. Diagram showing the training and test sets results using neural networks for the "quadratic" data set. Two trials were conducted using separate sets of 18 test compounds removed from the data set. Results are shown as the correlation coefficient plotted against ϱ for training; trial 1 (□) and trial 2 (△). Using these trained networks, predictions of activity were made for the relevant test set which are also shown as the correlation coefficient, trial 1 (■) and trial 2 (▲).

above the levels of the training set. Interestingly, for the first trial set for the quadratic case, the test results far exceeded the R^2 values for the training set (Figure 12). This result was indeed surprising, and the experiment was repeated following a check of the relevant BIOPROP script file to ensure that the training and test results had not been swapped. In addition, a second training and test set were created to again repeat the experiment. This second trial behaved as expected with the test results falling below the training set (Figure 12). To further investigate this result we examined plots of predicted (derived using eq 4, Table 1) and observed values of the biological activity used for the quadratic case marking the test compounds for trials 1 and 2, respectively (Figures 14, 15). At first sight, the plots appeared to show that the test compounds chosen

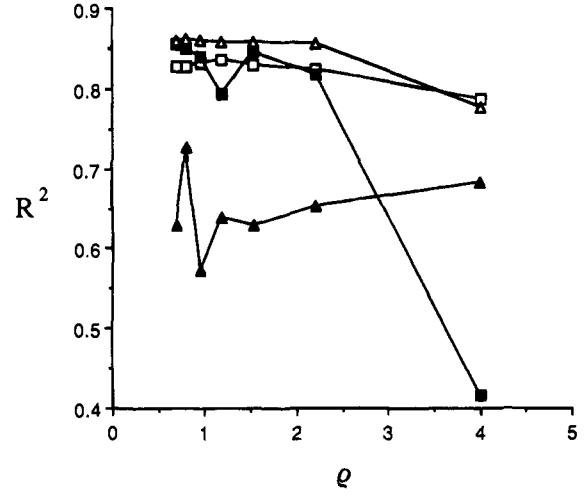

Figure 13. Diagram showing the training and test set results using neural networks for the "quadratic plus indicator" data set. Two trials were conducted using separate sets of 14 test compounds removed from the data set. Results are shown as the correlation coefficient plotted against ϱ for training, trial 1 (□) and trial 2 (△). Using these trained networks predictions of activity were made for the relevant test set which are also shown as the correlation coefficient, trial 1 (■) and trial 2 (▲).

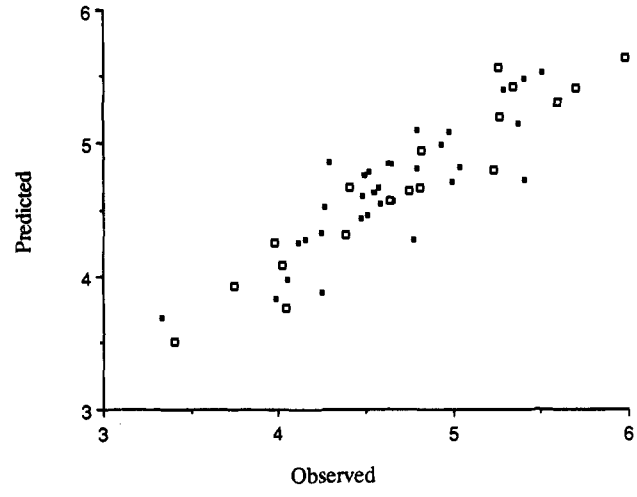

Figure 14. Plot of predicted against observed biological activities for the real quadratic QSAR example derived using eq 4, Table 1. Compounds represented with a (□) symbol was used as test compounds for trial 1 and the remaining compounds (■) comprised the training set.

were satisfactory representatives of the entire data set. On closer inspection, however, it can be seen that for trial 1, the test compounds fall closer to the line of perfect prediction. As this test set does not include some of the "outliers" that are present in the other set this may explain the improved prediction results. A check using MLR was carried out on both training and test sets. Regression equations were derived for each of the 32 training set compounds, and the activity was predicted for the 18 test cases (Table 3). As can be seen, the test set prediction for trial 1 gave a correlation coefficient of 0.905, considerably higher than the correlation coefficient of the training set. In accord with the network results, the test set prediction for trial 2 gave a correlation coefficient lower than that of the training set. The equations derived from the training sets are shown in Table 3 where it can be seen that the regression coefficients are considerably different.

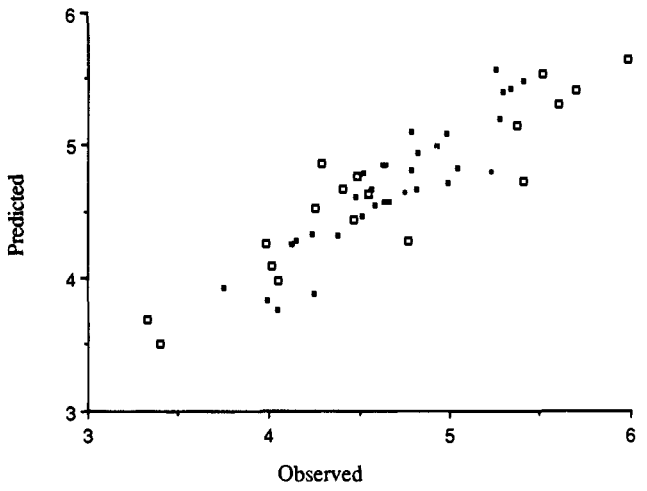

Figure 15. Plot of predicted against observed biological activities for the real quadratic QSAR example derived using eq 4, Table 1. Compounds represented with a (□) symbol were used as test compounds for trial 2 and the remaining compounds (■) comprised the training set.

Table 3. MLR Results for the Quadratic Training/Test Data Sets

trial	equation	n	R^2	s	F
1 training	$\log(1/D_{40}) = -1.16R_m^2 - 0.46R_m + 0.66pK_a + 0.58$	32	0.730	0.26	25.3
1 test[a]		18	0.905		
2 training	$\log(1/D_{40}) = -1.60R_{m^2} - 0.16R_m + 0.51pK_a + 1.65$	32	0.861	0.17	58.1
2 test[a]		18	0.800		

[a] The test set correlation coefficients were calculated by comparing predicted results from the training set MLR equation with observed results.

Conclusions

The results that we have found for both the structured data sets and the real QSAR sets are broadly in agreement with what we found using random number data.[10,12] That is to say, increasing the number of connections in a network (i.e., decreasing ϱ) leads to higher correlation coefficients, and this may be why it appeared that networks were out-performing traditional methods in earlier reports. The advantage of using structured data and real data is that it is possible to test predictions. The predictive ability of the networks was generally poor compared with how well they performed in fitting.

Two of the most important aspects of quantitative structure–activity relationship studies is that they are able to explain biological activity in a series of compounds and to predict activity in either test compounds or novel targets. This has particular importance in the pharmaceutical and agrochemical industries where a QSAR may be used to assign synthetic priorities and thus directly reduce costs. Consequently, we have directed some effort to an assessment of the predictive ability of neural networks. Leave-one-out cross-validation is an obvious way to check predictions but is not necessarily the best; leave-N-out is more realistic but introduces the complication of compound choice. The training/test protocol is preferred, but we have demonstrated the difficulty in the adequate choice of such sets, even using a multivariate technique such as cluster analysis.

The advantages of using neural networks to fit QSAR data are that the functional form of the relationship does not have to be specified and that indicator variables are not required in order to combine subsets of compounds in one equation. The disadvantages of neural networks are that it is difficult to assess importance/contribution of individual terms, and there is no equivalent to "fit" statistics as in regression. There are dangers of chance effects, and it is not easy to identify these in advance; in addition, it appears that networks perform poorly in prediction. Of course, one aspect not discussed here is the computation time required to set up and train these neural networks. As might be expected, the network results took considerably longer to obtain than MLR (for example, some of the cross-validation experiments took several hours to complete on a Silicon Graphics 4D/35 workstation).

Choice of network architecture (number of hidden units) is clearly critical and, from the data sets examined here, appears to be dataset dependent. In our previous work[10,12] we have shown there are critical ϱ values below which chance effects become a problem, but it is not possible to give general guidelines for the choice of ϱ values for the analysis of real data sets.

It appears to us that the disadvantages of neural networks outweigh their advantages, at least for the type of networks (feed forward back propagation) that we have examined here. This is not to say that other network architectures or applications other than MLR may not prove of value in the investigation of the complex relationships between chemical structure and biological activity. For example, a neural network has been shown to provide a novel method for the low-dimensional display of multivariate data,[27] and this has been used with some success in the investigation of QSAR's.[27,28]

Acknowledgment. The authors wish to thank David Salt, Martyn Ford, Martin Saunders, Brian Bond, Martin Johnson, Tariq Andrea, David Winkler, and John Davies for their helpful discussions.

Supplementary Material Available: The QSAR data used to produce Table 1 representing the four "real" QSAR datasets (7 pages). Ordering information is given on any current masthead page.

References

(1) Gasteiger, J.; Zupan, J. Neural Networks in Chemistry. *Angew. Chem., Int. Ed. Engl.* **1993**, *32*, 503–527.
(2) Hoskins, J. C.; Himmelblau, D. M. Artificial Neural Network Models of Knowledge Representation in Chemical Engineering. *Comput. Chem. Eng.* **1988**, *12*, 881–890.
(3) Qian, N.; Sejnowski, T. J. *Predicting* the Secondary Structure of Globular Proteins Using Neural Network Models. *J. Mol. Biol.* **1988**, *202*, 865–884.
(4) Bohr, H.; Bohr, J.; Brunak, S.; Cotterill, R.; Lautrup, B.; Norskov, L.; Olsen, O.; Petersen, S. Protein Secondary Structure and Homology by Neural Networks. *FEBS Lett.* **1988**, *241*, 223–228.
(5) Zupan, J.; Gasteiger, J. *Neural Networks for Chemists*; VCH Publishers: Cambridge, 1993.
(6) Lacy, M. E. Neural Network Technology and its Application in Chemical Research. *Tetrahedron Comput. Methodol.* **1990**, *3*, 119–128.
(7) Zupan, J.; Gasteiger, J. Neural Networks: A New Method for Solving Chemical Problems or Just a Passing Phase. *Anal. Chim. Acta* **1991**, *248*, 1–30.
(8) Manallack, D. T.; Livingstone, D. J. Neural Networks—A Tool for Drug Design. In *Chemometrics in Molecular Design*; Van de Waterbeemd, H., Ed.; VCH Weinheim, in press.
(9) Andrea, T. A.; Kalayeh, H. Applications of Neural Networks in Quantitative Structure-Activity Relationships of Dihydrofolate Reductase Inhibitors. *J. Med. Chem.* **1991**, *34*, 2824–2836.
(10) Manallack, D. T.; Livingstone, D. J. Artificial Neural Networks: Applications and Chance Effects for QSAR Data Analysis. *Med. Chem. Res.* **1992**, *2*, 181–190.

(11) Salt, D. W.; Yildiz, N.; Livingstone, D. J.; Tinsley, C. J. The Use of Artificial Neural Networks in QSAR. *Pestic. Sci.* **1992**, *36*, 161–170.
(12) Livingstone, D. J.; Manallack, D. T. Statistics Using Neural Networks—Chance Effects. *J. Med. Chem.* **1993**, *36*, 1295–1297.
(13) Ajay. A Unified Framework For Using Neural Networks to Build QSARs. *J. Med. Chem.* **1993**, *36*, 3565–3571.
(14) Tetko, I. V.; Luik, A. I.; Poda, G. I. Applications of Neural Networks in Structure-Activity Relationships of a Small Number of Molecules. *J. Med. Chem.* **1993**, *36*, 811–814.
(15) Wikel, J. H.; Dow, E. R. The Use of Neural Networks For Variable Selection in QSAR. *BioMed. Chem. Lett.* **1993**, *3*, 645–651.
(16) Ghoshal, N.; Mukhopadhayay, S. N.; Ghosal, T. K.; Achari, B. Quantitative structure-activity relationship studies using artificial neural networks. *Ind. J. Chem.* **1993**, *32B*, 1045–1050.
(17) Aoyama, T.; Suzuki, Y.; Ichikawa, H. Neural Networks Applied to Quantitative Structure-Activity Relationship Analysis. *J. Med. Chem.* **1990**, *33*, 2583–2590.
(18) Aoyama, T.; Suzuki, Y.; Ichikawa, H. Neural Networks Applied to Structure-Activity Relationships. *J. Med. Chem.* **1990**, *33*, 905–908.
(19) BIOPROP is available from Steven Muskal, Laboratory of Biodynamics, University of California, Berkeley, CA 94709. e-mail: "smuskal@sb1.cchem.berkeley.edu".
(20) Denny, W. A.; Atwell, G. J.; Cain, B. F. Potential Antitumour Agents. 32. Role of Agent Base Strength in the Quantitative Structure-Antitumour Relationships for 4′-(9-Acridinylamino)-methanesulfonanilide Analogues. *J. Med. Chem.* **1979**, *22*, 1453–1460.
(21) Coats, E. A.; Genther, C. S.; Dietrich, S. W.; Guo, Z.-r.; Hansch, C. Comparison of the Inhibition of Methotrexate-Sensitive and -Resistant *Lactobacillus casei* Cell Cultures with Purified *Lactobacillus casei* Dihydrofolate Reductase by 4,6-Diamino-1,2-dihydro-2,2-dimethyl-1-(3-substituted-phenyl)-*s*-triazines. Use of Quantitative Structure-Activity Relationships in Making Inferences about the Mechanism of Resistance and the Structure of the Enzyme in Situ Compared with the Enzyme in Vitro. *J. Med. Chem.* **1981**, *24*, 1422–1429.
(22) Lewis, D. F. V. The Calculation of Molar Polarizabilities by the CNDO/2 Method: Correlation with the Hydrophobic Parameter, LogP. *J. Comput. Chem.* **1989**, *10*, 145–151.
(23) Rumelhart, D. E.; McClelland, J. L. *Parallel Distributed Processing, Vol. 1*; MIT Press: Cambridge, MA, 1988.
(24) Hertz, J.; Krogh, A.; Palmer, R. G. *Introduction to the Theory of Neural Computation*; Addison-Wesley Publishing Co.: Santa Fe, CA, 1991.
(25) Katz, W. T.; Snell, J. W.; Merickel, M. B. Artificial Neural Networks. *Methods Enzymol.* **1992**, *210*, 610–636.
(26) Weinstein, J. N.; Kohn, K. W.; Grever, M. R.; Viswanadhan, V. K.; Rubinstein, L. V.; Monks, A. P.; Scudiero, D. A.; Welch, L.; Koutsokos, A. D.; Chiausa, A. J.; Paull, K. D. Neural Computing in Cancer Drug Development: Predicting Mechanism of Action. *Science* **1992**, *258*, 447–451.
(27) Livingstone, D. J.; Hesketh, G.; Clayworth, D. Novel Method for the Display of Multivariate Data Using Neural Networks. *J. Mol. Graph.* **1991**, *9*, 115–118.
(28) Good, A. C.; So, S.-S.; Richards, W. G. Structure-Activity Relationships from Molecular Similarity Matrices. *J. Med. Chem.* **1993**, *36*, 433–438.