ELSEVIER

Journal of Statistical Planning and Inference 46 (1995) 93–104

journal of statistical planning and inference

The maximum likelihood neural network as a statistical classification model

David Faraggi, Richard Simon*

Biometric Research Branch, National Cancer Institute, 6130 Executive Blvd., Room 739, Rockville, MD 20852, USA

Received 30 August 1993, revised 22 July 1994

Abstract

Neural networks have received considerable attention in recent years. This development has been pursued primarily by non-statisticians. Consequently many statistical tools and concepts have not been utilized in this development and great claims for neural networks have sometimes been made without comparisons to standard statistical procedures. In this paper we utilize the input–output relationship associated with a simple feed-forward neural network as the basis for a non-linear multivariate classifier. A statistical model for the data is defined based on a logistic likelihood function. Neural network parameters are estimated using the method of maximum likelihood instead of the back-propagation technique often used in the neural network literature. An extension for the multinomial case is presented. These maximum likelihood based models can be compared using readily available techniques such as the likelihood ratio test and the Akaike criterion (1973). We provide empirical comparisons of this network approach with standard logistic regression for both the binomial and multinomial cases.

Key words: Classification; Feed-forward neural networks; Logistic likelihood; Maximum likelihood

1. Introduction

The problem of classifying observations into two or more groups based on covariates has received much attention in the statistical literature. Many methods of classification have been developed including linear and non-linear discriminant analysis (Cacoullos, 1973), nearest neighbor analysis (Johnson, 1967), cluster analysis (Cormack, 1971; Hartigan, 1975) and classification trees (Breiman et al., 1984).

* Correspondence address: Biometric Branch, Department of Health and Human Services, National Institute of Health, Methesda, MD 20892, USA. E-mail: rich@brb.nci.nih.gov. Fax: 301-402-0560.

0378-3758/95/$09.50 © 1995—Elsevier Science B.V. All rights reserved.
SSDI 0378-3758(94)00093-X

Another approach is to model the probability of belonging to a specific group as a function of the covariates. Logistic models are most commonly used (Cox and Snell, 1989).

Recently neural networks (Rumelhart et al., 1986) have received considerable attention by non-statisticians for problems of classification and prediction. Applications include areas such as speech recognition (Sejnowski and Rosenberg, 1987), diagnostic image analysis (DaPonte and Sherman, 1991), clinical diagnosis (Mann and Brown, 1991), macromolecule sequence analysis (Holley and Karplus, 1989) and prediction of mechanism of action for new cancer drugs (Weinstein et al., 1992).

Growing attention is given to neural networks by statisticians. White (1989) developed some limiting properties of the back-propagation technique. Perlovsky and McManus (1991) used the normal mixture model to formulate the neural network and estimated its parameters by minimizing an entropy measure of classification using the EM algorithm. Gish (1993) took a probabilistic view of neural networks to derive the maximum likelihood estimates for the binary classification case. Nychka et al. (1992) used neural network non-linear regression models to estimate the dominant Lyapunov exponent and compared it with thin plate splines. Geman et al. (1992) discussed the variance/bias trade-off for neural networks. Ripley (1993) presented the basic ideas of the neural network and provided some comparisons with standard statistical techniques. Recently, Cheng and Titterington (1994) reviewed neural networks from a statistical point of view.

In this paper we develop a general statistical classification model based on the non-linear function of the covariates associated with a single hidden layer feed-forward neural network. We begin by reviewing single hidden layer feed-forward neural networks and the back-propagation algorithm commonly used for training. We then introduce a generalization of a two-class model using the non-linear function obtained from the neural network topology to replace the linear functional employed in the logistic model. Our approach is analogous to the generalization of linear model to the class of additive models (Hastie and Tibshiriani, 1986) where the linear function $\boldsymbol{x\beta}$ is replaced by a sum of univarate smoothers. The parameters of the resulting model are determined by maximizing the likelihood function instead of back-propagation. We then extend the results to multinomial classification. We conclude by providing some empirical comparisons, using simulated data. The neural network classification model is compared with linear and quadratic logistic models for binomial and multinomial examples.

2. Neural networks

An example of a single hidden layer feed-forward neural network is shown schematically in Fig. 1. For each case there are inputs $x_1, \ldots, x_k$ representing the covariates. A constant 1 is also taken as one of the inputs. Each input node is connected to all of

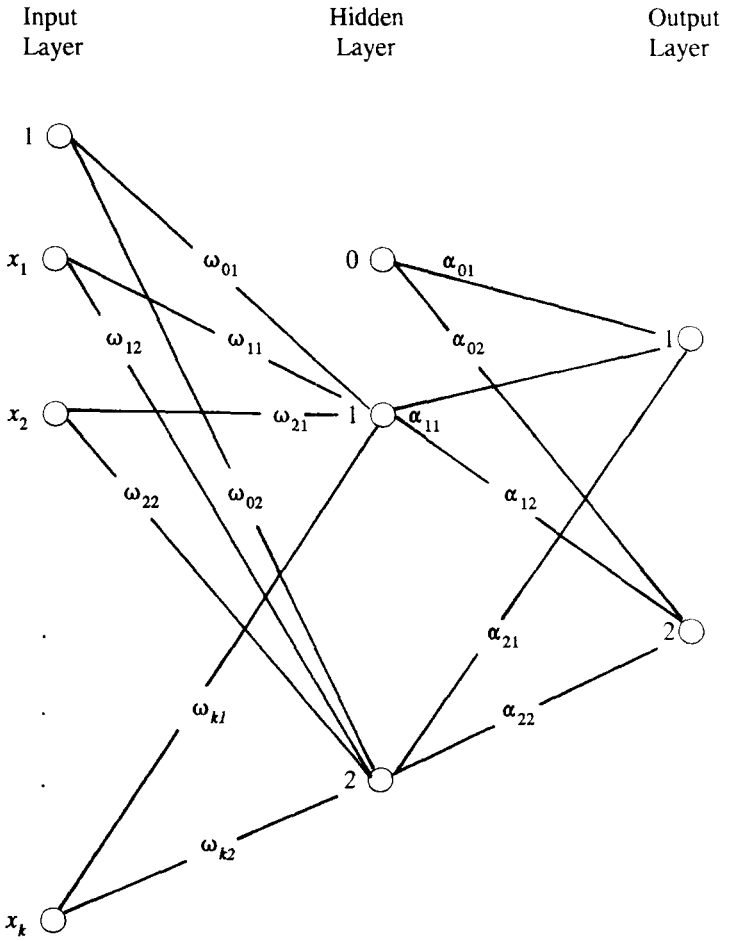

Fig. 1. Single hidden layer neural network with two hidden nodes and two outputs.

the nodes in the 'hidden layer' of the network. The hidden layer is the set of nodes between the input and output nodes. H denotes the number of nodes in the hidden layer. Each hidden node is connected to all of the output nodes. There is a special zeroth hidden node that is connected to the output nodes but not the input nodes. It's role will be described below.

Consider a vector of covariates $x_i=(1, x_{i1}, x_{i2}, \ldots, x_{ik})$ for the ith case as an input to the network. The outputs of the network are $g(x_i, \theta)$ where $g(\cdot)$ is a vector of R outputs defined by the network topology and θ is a vector of unknown parameters. The input to the hth hidden unit in the hidden layer, $(h=1, \ldots, H)$, is a linear projection of the input vector x_i, as $z_{ih}=x_i \omega_h$, where $\omega_h=(\omega_{0h}, \omega_{1h}, \omega_{2h}, \ldots, \omega_{kh})'$. The output of hidden unit h with input z_{ih} is $f(z_{ih})=1/(1+\exp(-z_{ih}))$ where $f(\cdot)$ is called a squashing

function. The output from node r in the output layer ($r=1,\ldots,R$) is a linear projection of the output of the hidden nodes, plus a constant. The weights ω_{0h} multiplied by fixed input '1' and the weights between the zeroth hidden node and the output nodes are called bias terms. These bias terms are analogous to the constant term in simple regression. They ensure that the output is invariant to scale changes of the input variables and provide an unrestricted range to the output of the network. We omit the squashing function from the output layer for reasons discussed below. Hence the functional representation of the output from the rth node in the output layer in a single hidden layer network with H hidden units for a given input vector $\boldsymbol{x}_i$ is

$$g_r(\boldsymbol{x}_i,\boldsymbol{\theta}_r)=\alpha_{0r}+\alpha_{1r}[1/(1+e^{-x_i\omega_1})]+\alpha_{2r}[1/(1+e^{-x_i\omega_2})]+\ldots$$
$$+\alpha_{Hr}[1/(1+e^{-x_i\omega_H})], \tag{1}$$

where $\boldsymbol{\theta}_r=(\boldsymbol{\omega}_1',\boldsymbol{\omega}_2',\ldots,\boldsymbol{\omega}_H',\alpha_{0r},\alpha_{1r},\alpha_{2r},\ldots,\alpha_{Hr})'$. The total number of parameters in (1) is $(k+1)H+(H+1)$. Define $\boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{\theta})=[g_1(\boldsymbol{x}_i,\boldsymbol{\theta}_1),\ldots,g_R(\boldsymbol{x}_i,\boldsymbol{\theta}_R)]$ to be the row vector of R outputs for a single input $\boldsymbol{x}_i$. Thus the total number of parameters in the network is $(k+1)H+(H+1)R$. Also define $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\theta})=[\boldsymbol{g}(\boldsymbol{x}_1,\boldsymbol{\theta})',\ldots,\boldsymbol{g}(\boldsymbol{x}_n,\boldsymbol{\theta})']'$ an n by R output matrix for all n observations ($i=1,\ldots,n$). A single hidden layer network with $H=1$ and the identity squashing function $f(z)=z$ reduces the neural network function $g(\cdot)$ to the usual linear function of the input variables.

Back-propagation (Rumelhart et al., 1986) is a commonly used algorithm for training the network; that is, for estimating the network parameters. The back-propagation procedure is a gradient descent method to minimize the total squared error $\sum_i\sum_r(y_{ir}-g_r(\boldsymbol{x}_i,\boldsymbol{\theta}_r))^2$, where y_{ir} is the rth response of the ith case. The training begins by choosing an arbitrary starting $\hat{\boldsymbol{\theta}}^{(0)}$. The first step moves from $\hat{\boldsymbol{\theta}}^{(0)}$ in the direction of the gradient of $\sum_r(y_{1r}-g_r(\boldsymbol{x}_1,\hat{\boldsymbol{\theta}}_r^{(0)}))^2$. The second step moves from $\hat{\boldsymbol{\theta}}^{(1)}$ in the direction of the gradient of $\sum_r(y_{2r}-g_r(\boldsymbol{x}_2,\hat{\boldsymbol{\theta}}_r^{(1)}))^2$. This repeats for many cycles through the data until the network is trained, i.e., the estimates converge. Since the network function is non-linear with many parameters involved, the error surface may be highly convoluted and contain many local optima. Numerous improvements and modifications to the basic back-propagation method have been proposed. For further discussion of neural networks and back-propagation see, for example, Wasserman (1989).

3. Maximim likelihood neural network classification

3.1. The binary case

Consider the case of binary classification based on a set of covariates. The well-known logistic model assumes that $\log[p/(1-p)]=\boldsymbol{x}_i\boldsymbol{\beta}$, $i=1,\ldots,n$ where $\boldsymbol{x}_i=(1,x_{i1},\ldots,x_{ik})$, $\boldsymbol{\beta}=(\beta_0,\beta_1,\ldots,\beta_k)'$, n is the number of observations and $p\,(1-p)$ is

the probability that an observation belongs to class 1(0). It is easy to show that the log likelihood for the logistic model can be written as

$$\log L(\boldsymbol{\beta}) = \boldsymbol{y}'\boldsymbol{x}\boldsymbol{\beta} - \mathbf{1}_n' \log[\mathbf{1}_n + \exp(\boldsymbol{x}\boldsymbol{\beta})], \tag{2}$$

where $\boldsymbol{y}$ is an n by 1 vector of the binary classifications, $\boldsymbol{x} = (\boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_n')'$ and $\mathbf{1}_n$ is an n by 1 vector of ones. The log operation in (2) denotes an element by element natural log of a vector. Consider replacing the linear combination of the covariates and the parameters $\boldsymbol{x}\boldsymbol{\beta}$ that appears in the log likelihood (2) with the output vector of the network $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta})$ that is obtained from (1) for a single output network ($R = 1$). As we replace $\boldsymbol{x}\boldsymbol{\beta}$ that maps $\Re^n \to \Re^{k+1}$ with the output of the network we omit the squashing function that is usually applied to the output of the network so that the output $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta})$ will be on the whole real line and not squashed to (0, 1). We obtain

$$\begin{aligned} \log L(\boldsymbol{\theta}) &= \boldsymbol{y}'\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) - \mathbf{1}_n' \log[\mathbf{1}_n + \exp(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}))] \\ &= \sum_i y_i g_1(\boldsymbol{x}_i, \boldsymbol{\theta}) - \sum_i \log[1 + \exp(g_1(\boldsymbol{x}_i, \boldsymbol{\theta}))]. \end{aligned} \tag{3}$$

3.2. The multinomial case

Assume now that there are s classes. The multinomial logistic model can be written as $\log[p_w/(1-p_w)] = \boldsymbol{x}_i\boldsymbol{\beta}_w$ for $w = 1, \ldots, (s-1)$ and $p_s = 1 - p_1 - \cdots - p_{s-1}$, where p_w is the probability of class w and $\boldsymbol{\beta}_w$ is an unknown vector of coefficients associated with class w. Assume that the n observations take m different values $\boldsymbol{x}_v$ ($v = 1, \ldots, m$). Let $n_l(\boldsymbol{x}_v)$ denote the frequency of $\boldsymbol{x}_v$ in class l ($l = 1, \ldots, s$). The likelihood function can be written as

$$L = \prod_{l=1}^{s} \prod_{v=1}^{m} [p_l(\boldsymbol{x}_v)]^{n_l(\boldsymbol{x}_v)}, \tag{4}$$

where $p_w(\boldsymbol{x}_v) = \exp(\boldsymbol{x}_v\,\boldsymbol{\beta}_w)/[\sum_{l=1}^{s} \exp(\boldsymbol{x}_v\,\boldsymbol{\beta}_l)]$, $\boldsymbol{\beta}_w = (\beta_{0w}, \beta_{1w}, \ldots, \beta_{kw})'$, and $\boldsymbol{\beta}_s = \mathbf{0}$. Note that this multinomial model permits all parameters to vary from one class to another. For further discussion of this model see, for example, Seber (1984).

The log likelihood can be written in matrix form. Define $\boldsymbol{y}$ to be an n by $(s-1)$ matrix where the ith row of $\boldsymbol{y}$ has one in the wth column if the observation belongs to class w and zeros elsewhere. If an observation belongs to the sth class the row in $\boldsymbol{y}$ that corresponds to this observation will have all zeros in it. The log likelihood can be written as

$$\log L(\boldsymbol{\beta}) = \operatorname{trace}(\boldsymbol{y}'\boldsymbol{x}\boldsymbol{\beta}) - \mathbf{1}_n' \log\{\mathbf{1}_n + \exp[(\boldsymbol{x}\boldsymbol{\beta})\mathbf{1}_{(s-1)}]\}, \tag{5}$$

where $\boldsymbol{x} = (\boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_n')'$, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_{(s-1)})$, $\mathbf{1}_n$ and $\mathbf{1}_{(s-1)}$ are vectors of ones of sizes n and $(s-1)$, respectively. Binary classification is a special case of (5) with $s = 2$. As in the binomial case we replace the linear relation between the covariates and the

parameters ($x\beta$) in the log likelihood (5) by the $R = s-1$ outputs of the neural network. In (5) $x\beta$ is an n by $(s-1)$ matrix, hence, it can be replaced by the output matrix $g(x, \theta)$ described in Section 2 for $(s-1)$ outputs. We get

$$\log L(\theta) = \operatorname{trace}[y'g(x,\theta)] - \mathbf{1}_n' \log\{\mathbf{1}_n + \exp[(g(x,\theta))\mathbf{1}_{(s-1)}]\}. \tag{6}$$

In an effort to begin to provide some empirical experience with these classifiers we have studied the following examples. We provide simulated examples where the functions generating the data are known and can be used in the evaluation of the performance of the different models. The Newton–Raphson iterative procedure is used to maximise the log-likelihood function and obtain the maximum likelihood estimates. Gauss (1993) maximum likelihood software was used to obtain the estimates. Both first and second derivatives were approximated numerically. To have reasonable assurance that we found the global maximum rather than a local one we used multiple starting values to initiate the search. In the examples where over 30 parameters had to be estimated, 100 different randomly selected initial values were used. Between 300–500 iterations were needed for each set of starting values to reach convergence.

4. Examples

4.1. The binomial case

To compare the neural network model with the logistic model 100 data points were generated using two functions:

I. $\log[p/(1-p)] = -0.5 + 2x_1 - x_2 + 0.3x_1^2 - 0.8x_2^2 + 0.6x_1x_2 + \varepsilon,$

II. $\log[p/(1-p)] = 1.5 + 2.3\log(x_1) - 0.6\sqrt{x_2} - 1.7(1/x_3) + 0.3x_4^2$
$$-3.2[\log(x_1)]\sqrt{x_2} + 0.8[\log(x_1)][1/x_3] - 1.9[1/x_3]x_4^2 + \varepsilon,$$

where the x's for the first function were uniformly selected from a hyper-plane [0, 1] and for the second function $x_1 \sim \mathrm{U}(0,5)$, $x_2 \sim \mathrm{U}(0,3)$, $x_3 \sim \mathrm{N}(0,49)$, $x_4 \sim \mathrm{N}(0,1)$. We used $\varepsilon \sim \mathrm{N}(0, \sigma^2)$ and a variety of noise levels were added to the models so that the effect of the increased noise levels on model performance could be studied. This approach permitted us to evaluate the potential over-fitting of the neural network models (Geman et al., 1992).

The potential for over-fitting for neural networks is well known. Neural network users often split the sample into a training set and a test set in order to avoid over-fitting. However, when the true function that generates the data is known, as in our simulation study, we can determine the 'true' classification of an observation as that determined by the specific function when no noise is added to the model. That is,

if the function without noise is denoted $p(x, 0)$, then the 'true' class of a case with covariate vector x_i is round$(p(x_i, 0))$ where round$(p(x_i, 0))=1$ if $p(x_i, 0) \geqslant 0.5$ and 0 otherwise. Introducing different noise levels to the 'true' relationship changes the 'true' class to the 'observed' class round$(p(x_i, \varepsilon_i))$.

We generated ten different 'observed' vectors each of size 100 by resampling the noise from the normal distribution. Because the search for the global maximum is very computer intensive, we limited our simulation study to ten replicates. However, the results indicated that the same pattern displayed in the table for the averages was obtained in most replicates. This provides additional assurance that the results displayed are accurate even with the limited number of replications performed.

For all models the estimates of the parameters are the maximum likelihood estimators. Once the estimates were obtained they were used to predict the binary response. If $\hat{p}(x_i, \hat{\theta})$ was $\geqslant 0.5$ for case x_i then the predicted class for the ith observation was taken to be one, otherwise it was predicted to be zero. Comparing predicted classes to the 'true' classes, we obtain a measure of the extent to which the models fit or over-fit the data. This measure is an independent measure of the misclassifications and thus we avoid the less efficient approach of evaluating the performance of the network on an independent noisy data set.

We chose two functions for data generation. The first is a quadratic function. Hence, for such data a quadratic logistic regression model will obviously produce the best possible results. The quadratic model for this case will serve as the reference model. The second function is non-linear in the covariates but linear in the parameters. For data generated with this function both the linear and quadratic regression models are obviously not correct. However, they are evaluated as models that are commonly used by statisticians as approximations to the unknown functions and are compared with the neural network models.

4.1.1. The first function

Table 1 summarizes the results obtained using the first function when different amounts of noise were added to the simulated data. The first row in the table indicates the variance of the $N(0, \sigma^2)$ generating the noise added to the data.

Two models were used to fit the data for this example. The quadratic model which is the correct model for this case except for the lack of a normal noise term. The second model was the neural network model with two nodes in the hidden layer (NN$(H=2)$). The second column of Table 1 shows the number of parameters in the models. The entries in Table 1 are the average number of misclassifications of the predicted categories as compared to the 'true' classifications when no noise was added. We provide also the standard error of these average misclassifications (in parenthesis).

Table 1 indicates that when the noise level is low to moderate ($\sigma^2 \leqslant 1$) the correct model (the quadratic model) performs well. The average number of misclassifications is at the most 5.7 compared with the binary response when no noise was added. When the noise level was increased to $\sigma^2=4$ (average of 34.8 changes from the 'true' to the

Table 1
Misclassifications for the first function

Model	Number of parameters	σ^2			
		0.01[a]	0.1	1	4
Quadratic	6	1.1[b]	1.4	5.7	16.5
		(0.28)[c]	(0.31)	(1.4)	(2.3)
NN ($H=2$)	9	1.1	2.1	7.0	14.8
		(0.28)	(0.23)	(1.5)	(1.5)

[a] The σ^2 values of 0.01, 0.1, 1 and 4 correspond to averages of 1.4, 7.1, 24.5 and 34.8 changes in classification compared to no-noise case, respectively.
[b] Mean.
[c] Standard error.

Table 2
Misclassifications for the second function

Model	Number of parameters	σ^2		
		0	0.25[a]	1
Linear	5	15	16.6[b]	19.4
			(0.9)[c]	(1.5)
Quadratic	15	11	14.9	17.5
			(0.7)	(0.8)
NN ($H=2$)	13	8	11.2	15.3
			(0.9)	(0.8)
NN ($H=3$)	19	4	9.9	11.7
			(0.6)	(1.2)

[a] The σ^2 values of 0, 0.25 and 1 correspond to averages of 0, 4.8 and 12.0 changes in classification compared to no-noise case, respectively.
[b] Mean.
[c] Standard error.

'observed' binary response) the ability of the quadratic model to predict the correct binary response decreased (average of 16.5 misclassifications). The performance of the neural network model with two nodes in the hidden layer is very similar to the performance of the quadratic model.

4.1.2. The second function

Table 2 summarizes the results for the data obtained using the second function with variable amounts of added noise. The layout of the table is identical to Table 1. For this case all the models that are presented in Table 2 are approximations to the true model. In practice the correct model is generally unknown and the model used

approximations the true relationship. For this example both neural network models outperformed the linear and quadratic models. This is especially apparent for the case when $\sigma^2 = 0$. The addition of 10 parameters (from the linear to the quadratic model) reduced the number of misclassifications from 15 to 11 (27%). On the other hand the neural network model with two nodes in the hidden layer which has 13 parameters reduced the number of misclassifications to 8 (47% reduction from the linear model). Addition of four more parameters and a different model (19 parameters for the neural network with three hidden nodes) reduced the number of misclassifications was reduced to 4 (64% reduction from the linear model).

4.2. *The multinomial case*

For the multinomial case function II was used with different coefficients for different classes. We set $\boldsymbol{\beta}_1 = (1.5, 2.3, -0.6, -1.7, 0.3, -3.2, 0.8, -1.9)'$ and $\boldsymbol{\beta}_2 = (-0.5, -1, 2, 0.5, 1, -1, 3, 2.5)'$ i.e., the number of groups in this example was set to $s = 3$. We used the same distribution assumptions on the x's as above. One hundred x's were generated. With the defined $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ as above $\boldsymbol{x}_i\boldsymbol{\beta}_1$ and $\boldsymbol{x}_i\boldsymbol{\beta}_2$ were determined for all observations. $p_{iw}^{\mathrm{T}} = \exp(\boldsymbol{x}_i\boldsymbol{\beta}_w)/[1 + \exp(\boldsymbol{x}_i\boldsymbol{\beta}_1) + \exp(\boldsymbol{x}_i\boldsymbol{\beta}_2)]$ the 'true' probability was determined for $w = 1, 2$ and $i = 1, \ldots, 100$. Finally $p_{i3}^{\mathrm{T}} = 1 - p_{i1}^{\mathrm{T}} - p_{i2}^{\mathrm{T}}$ was calculated. The 'true' classification of the ith observation was determined by the class associated with the largest p_{il}^{T}, $l = 1, 2, 3$. As before, we repeated the procedure 10 times with noise added. $p_{iw} = \exp(\boldsymbol{x}_i\boldsymbol{\beta}_w + \varepsilon_i)/[1 + \exp(\boldsymbol{x}_i\boldsymbol{\beta}_1 + \varepsilon_i) + \exp(\boldsymbol{x}_i\boldsymbol{\beta}_2 + \varepsilon_i)]$ was determined for $w = 1, 2$ and $p_{i3} = 1 - p_{i1} - p_{i2}$, $i = 1, \ldots, 100$. The 'observed' class of the ith observation was determined as the class associated with the largest p_{il}, $l = 1, 2, 3$. The 10 'observed' sets were used in the estimation procedure for all models. With the estimated parameters the classes were predicted. For case $\boldsymbol{x}_i$ the class associated with the largest $\hat{p}_{il}(\boldsymbol{x}_i, \boldsymbol{\theta})$ was predicted, $l = 1, 2, 3$.

Table 3 summarizes the results for the multinomial example when $\varepsilon \sim \mathrm{N}(0, 1)$. At this noise level there were on the average 14.2 changes in 'observed' classification compared with the 'true' classification. Table 3 provides the average number of misclassifications, the standard error of the average and the maximized log likehood. It indicates that the neural network models with more than two nodes in the hidden layer produced fewer misclassifications than linear or quadratic models.

5. Discussion

The purpose of this paper is to develop maximum likelihood neural network based models for the general classification problem. Using maximum likelihood rather than back-propagation to minimize squared error is analogous to the use of the logistic model to relate a scalar $x\beta$ to the probability of response rather than use of a linear

Table 3
Misclassifications and maximum log likelihood results obtained for the second function — the multinomial case

Model	q	Average # of misclassification from 'true'	MAX L (lik)
Linear	10	24.6[a]	−71.3
		(0.9)[b]	(1.4)
Quadratic	30	21.8	−48.9
		(1.0)	(2.3)
NN ($H=2$)	16	25.2	−52.8
		(0.7)	(2.8)
NN ($H=3$)	23	18.8	−38.8
		(1.0)	(2.7)
NN ($H=4$)	30	15.6	−29.1
		(1.0)	(3.1)
NN ($H=5$)	37	15.6	−25.4
		(1.5)	(2.8)

[a] Standard error.
[b] Mean.

model with binary response data. It is important to emphasize that in our simulation study although the noise added to the functions was generated from the normal distribution it only caused changes in the discrete response from the 'true' response to the 'observe' response. The likelihood functions were based on either the binary or multinomial logistic model. When the response is continuous the neural network non-linear function can be used in a non-linear regression type setting with the appropriate likelihood maximized.

Since the logistic function satisfies the regularity conditions under which the maximum likelihood estimators exist (Cramer, 1946), the maximum likelihood estimators of the neural network parameters are consistent, asymptotically normal and asymptotically efficient. Hence, if the relationship between class probabilities and covariates satisfies the neural network logistic models described in Section 3, then as the amount of data increases, the estimated model will converge to the true model. Furthermore, Gallant and White (1991) showed that the neural network model with sufficient number of hidden nodes will approximate any square integrable function to any degree of accuracy. Hence, the maximum likelihood estimates of the neural network with sufficient number of hidden nodes will asymptotically approach the correct discriminant boundary assuming that the true discriminant boundary function is square integrable.

This neural network generalization of logistic discrimination using maximum likelihood parameter estimates has several potential advantages. It permits the topology of a single hidden layer feed-forward network to be used for representing the relationship between predictors and classes. It does this in the context of a specific

statistical model for the data and hence standard statistical methods for evaluating estimators, selecting submodels and evaluating predictors are available. Because the number of parameters in neural network models increase rapidly with the number of input covariates and hidden nodes, over-fitting of the data may result. Standard likelihood ratio tests and Akaike's criterion can be used to select parsimonious models. This can be generalized to structured forward selection or backward elimination procedures for the selection of input variables. Other approaches are also possible, such as sample splitting, cross-validation or the bootstrap. When interpretation of which covariates are 'important' is not of concern, use of the first several principle components of the covariates is an alternative approach to dimensionality reduction (Koutsoukos et al., 1994). Finally, we use the Newton–Raphson iteration on the entire training data set simultaneously rather than back-propagation type procedures using one observation at a time (or a few at a time). This avoids the problem of training patterns (Newman, 1991) where the parameter estimates depend on the sequence in which the data are presented.

Neural network classification, as introduced here, will not necessarily provide better results than standard statistical approaches. The results will depend on the unknown model from which the data arise and the amount of data available. Our simulation results suggest that in some cases the neural network output functions provide a good approximation to the true but unknown relation between covariates and classes. For function I, the neural network performed about as well as the true quadratic model. When the true model II was unknown, the neural network model outperformed both the linear and quadratic models. We believe that neural network based non-linear functions should be considered by statisticians for classification problems, along with other statistical approaches.

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In: B.N. Patrov and F. Csaki, Eds., *Proc. 2nd Internat. Symp. on Information Theory*, Akademia Kiado, Budapest, 267–281.

Breiman, L., J.H. Friedman, R.A. Olshen and C.J. Stone (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Cacoullos, T. (Ed.) (1973). *Discriminant Analysis and Applications*. Academic Press, New York.

Cheng, B. and D.M. Titterington (1994). Neural networks: a review from a statistical perspective (with discussion). *Statist. Sci.* **9**, 2–54.

Cormack, R.M. (1971). A review of classification. *J. Roy. Statist. Soc. Ser. A* **134**, 321–367.

Cox, D.R. and E.J. Snell (1989). *Analysis of Binary Data*, 2nd ed. Chapman and Hall, London.

Cramer, H. (1946). *Mathematical Methods of Statistics*. Princeton Univ. Press, Princeton, NJ.

Dapontc, J.S. and P. Sherman (1991). Classification of ultrasonic image texture by statistical discriminate analysis of neural network. *Comput. Med. Image. Graph.* **15**, 3–9.

Gallant, A.R. and H. White (1991). On learning the derivatives of an unknown mapping with multilayer feedforward networks. *Neural Networks* **5**, 129–138.

Gauss (1993). *Maximum Likelihood Gauss Applications*. Aptech Systems, Inc.

Geman, S., E. Bienstock and R. Doursat (1992). Neural networks and the bias/variance dilemma. *Neural Comput.* **4**, 1–58.

Gish, H. (1993). Maximum likelihood training of neural networks. In: D.J. Hand, Ed., *Artificial Intelligence Frontiers in Statistics*. Chapman and Hall, London.

Hartigan, J.A. (1975). *Clustering Algorithms*. Willy, New York.

Hastie, T. and R. Tibshiriani (1986). Generalized additive models. *Statist. Sci.* **1**, 297–318.

Holley, L.H. and M. Karplus (1989). Protein structure prediction with a neural network. *Proc. Natl. Acad. Sci. USA* **86**, 152–156.

Johnson, S.C. (1967). Hierarchical clustering schemes. *Psychometrika* **32**, 241–254.

Koutsoukos, A.D., L.V. Rubinstein, D. Faraggi, R.M. Simon, S. Kalyandrug, J.N. Weinstein, K.W. Kohn and K.D. Paull (1994). Discrimination techniques applied to the NCI in vitro anti-tumor drug screen: predicting biochemical mechanism of action. *Statist. Medicine* **13**, 719–730.

Mann N.H.I. and M.D. Brown (1991). Artificial intelligence in the diagnosis of low back pain. *Orthop. Clin. North Am.* **22**, 303–314.

Newman D.S. (1991). Simulation experiments for neural network learning. Computer science and statistics. In: E.M. Keramidas and S.M. Kaufman, Eds., *Proc. 23rd Symp. on the Interface*.

Nychka, D., S. Ellner, A.R. Gallant and D. McCaffrey (1992). Finding chaos in noisy systems. *J. Roy. Statist. Soc. Ser.* B **2**, 399–426.

Perlovsky, L.I. and M.M. McManus (1991). Maximum likelihood neural networks for sensor fusion and adaptive classification. *Neural Networks* **4**, 89–102.

Ripley, B.D. (1993). Statistical aspects of neural networks. In: O.E. Brandorff-Nielsen, J.L. Jensen and W.S. Kendall, Eds., *Network and Chaos — Statistical and Probabilistic Aspects*. Chapman and Hall, London.

Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986). Learning internal representations by error propagation. In: *Parallel Distributed Processing*. Vol. 1, MIT Press, Cambridge, MA; 318–362.

Seber, G.A.F. (1984). *Multivariate Observations*. Willey, New York.

Sejnowski, T.J. and L.R. Rosenberg (1987). Parallel networks that learn to pronounce English text. *Complex Systems* **1**, 145–168.

Wasserman, P.D. (1989). *Neural Computing Theory and Practice*. Van Nostrand Reinhold, New York.

Weinstein, J.N., K.W. Kohn, M.R. Grever, V.N. Viswanadhan, L.V. Rubinstein, A.P. Monks, D.A. Scudiero, L. Welch, A.D. Koutsoukos, A.J. Chiausa and K.D. Paull (1992). Neural computing in cancer drug development predicting mechanism of action. *Science* **258**, 447–451.

White, H. (1989). Some asymptotic results for learning in single hidden-layer feedforward network models. *J. Amer. Statist. Assoc.* **84**, 1003–1013.