Neural Networks and Related Methods for Classification

by B. D. RIPLEY

Department of Statistics, University of Oxford,
1 South Parks Road, Oxford OX1 3TG, UK.

Email: `ripley@stats.ox.ac.uk`

To be read to the Royal Statistical Society Research Section on 20 October 1993.

This paper is available in PostScript, and its datasets, by anonymous ftp from

`markov.stats.ox.ac.uk (192.76.20.1)`

in directory `pub/neural/papers`. See the `README` for details.

Neural Networks and Related Methods for Classification

by B. D. RIPLEY

Department of Statistics, University of Oxford[*]

10 December 1992; revised 23 July 1993

Summary

Feed-forward neural networks are now widely used in classification problems, whereas non-linear methods of discrimination developed in the statistical field are much less widely known. A general framework for classification is set up within which methods from statistics, neural networks, pattern recognition and machine learning can be compared. Neural networks emerge as one of a class of flexible non-linear regression methods which can be used to classify via regression. Many interesting issues remain, including parameter estimation, the assessment of the classifiers and in algorithm development.

Keywords: NEURAL NETWORKS; CLASSIFICATION; SEMI-PARAMETRIC REGRESSION; NON-LINEAR DISCRIMINANTS; PROJECTION PURSUIT REGRESSION; GENERALIZATION

1 Introduction

Neural networks have been developed rapidly since around 1985, and are now used widely. The subject area is not well-defined, being distorted by funding considerations, but is dominated by two areas of study:

- feed-forward networks, also known as multi-layer perceptrons, used for classification (and less often for regression and function approximation, for example in control; Miller *et al.*, 1990).
- symmetric recurrent networks, known as attractor neural networks (ANNs) or Hopfield nets, used as associative memories.

In both areas the biological connections were originally seminal but are now much weaker, and the emphasis is on solving application problems rather than on gaining insights into neurobiology. The second field is closer to biology and will not be discussed further in this paper: Amit (1989) is an excellent introduction. Another area of increasing activity is the use of recurrent neural networks to predict time series (e.g. Weigend & Gerschenfeld, 1993).

There are a number of wide-ranging introductions to the field of neural networks, of which Hertz *et al.* (1991) is very suitable for a statistical audience. I have recently written an extensive review paper (Ripley, 1993) setting neural networks in a statistical context with examples. Weiss & Kulikowski (1991) contrast neural networks approaches with those of statistical pattern recognition and machine learning. One view that I have heard independently expressed several times is that that the main impact of neural networks has

[*] 1 South Parks Road, Oxford OX1 3TG, UK. Email: `ripley@stats.ox.ac.uk`

been to revitalize the field of pattern recognition, and the material in this paper could certainly be seen as part of that subject.

Classification and *discrimination* are used in a number of closely related senses. Consider the following real life examples:

- Grading Danish bacon rashers
- Botanical field guide key to species identification
- Distinguishing male and female crabs of two species (Campbell and Mahon, 1974)
- Recognizing symbols on hand-drawn maps (Hjort, 1986)
- Predicting occurrence of tsetse flies in Zimbabwe (Ripley, 1993)

In each we are given measurements (features) from a space $\mathcal{X}$ for each object, and asked to assign a label from a set $\mathcal{C}$ of K classes. We will usually think of $\mathcal{X}$ as a Euclidean space and of each dimension as a feature, but the framework includes discrete components and more complicated structures (such as whole images and point patterns of variable size). (Although we do not consider this point further until section 8, readers should be aware that the major consideration in most pattern recognition problems is to select the 'right' features to make up $\mathcal{X}$.)

We may want a definite classification, or we may want some measure of belief in a possible classification, for example a probability distribution over $\mathcal{C}$. To allow for this additional information, we suppose that we are asked to supply a response from a space $\mathcal{Y}$. Typically this will be $[0,1]^K$, but with ordered classes (as for bacon graded 1 to 4) we might return a latent continuous score. In any case, a *classifier* is a map $f\colon \mathcal{X} \to \mathcal{Y}$.

A classifier has to be learned (estimated) from a collection of p examples $(\mathbf{X}_i, Y_i)$ of features and the desired responses. The example response is usually a definite classification, but it need not be, especially when the responses are elicited from experts summarizing past experience.

There can be two rather distinct purposes in learning a classifier. The most obvious is to classify future observations, as in all five examples. However, for the crabs and tsetse flies we could be interested in the form of the classifier *per se*, that is in what combinations of the features discriminate between the classes. Only for predictive classification is a 'black-box' representation of f useful. (However, it is sometimes possible to approximate the 'black-box' in a more interpretable way, for example as a set of decision rules. See §4.)

Problems of this general class (and often the very same examples) have been studied in the fields of pattern recognition, machine learning and neural networks as well as multivariate statistics. Although the precise methods of estimating the classifier f differ, the issues of model selection and assessment are common to all. In this paper we aim to lay out what is common and what is distinctive in the many approaches, to allow a synthesis of the best features of all. Some reviewers have commented that the paper contains surprisingly little about neural networks, but this represents a misunderstanding of the area; we do not expect a paper on regression to concentrate on linear least-squares, and similarly the important developments in neural networks are not about the networks *per se* (§2) but how they are used. In such a fast developing field (emphasised by how recent the references are) I will surely have missed many things (or omitted them for lack of space) which the discussion will help to fill in.

2 Feedforward neural networks

Feed-forward neural networks provide a flexible way to generalize linear regression functions. We start with the simplest but most common form with one hidden layer as shown in figure 1. The input units just provide a 'fan-out' and distribute the inputs to the 'hidden'

units in the second layer. These units sum their inputs, add a constant (the 'bias') and take a fixed function ϕ_h of the result. The output units are of the same form, but with output function ϕ_o. Thus

$$y_k = \phi_o\left(\alpha_k + \sum_j w_{jk}\,\phi_h\Big(\alpha_j + \sum_i w_{ij}\,x_i\Big)\right) \tag{1}$$

The 'activation function' ϕ_h of the hidden layer units is almost always taken to be the logistic function

$$\ell(z) = \frac{\exp(z)}{1+\exp(z)}$$

and the output units are linear, logistic or threshold units. (The latter have $\phi_o(x) = I(x > 0)$.)

We can eliminate the biases α_i by introducing an input unit which is always at $+1$ and feeds every other unit. (This is the same idea as adding a constant column to the design matrix to include the intercept in a regression.) The function f is then parameterized by the set of weights w_{ij}, one for every link in the network (or zero for links which are absent).

Such networks have a considerable history, including an original biological motivation, which is explained in Hertz *et al.* (1991). However, they can equally be seen as a way to parameterize a fairly general non-linear function from $\mathcal{X}$ to $\mathcal{Y}$. Such networks *are* rather general: Cybenko (1989), Funahashi (1989), Hornik *et al.* (1989) and later authors have shown that neural networks with linear output units can approximate any continuous function f uniformly on compact sets, by increasing the size of the hidden layer. This implies approximation of measurable functions in measure and L_p. A short constructive proof (essentially that given by Jones, 1990) can be based on the denseness of trigonometric polynomials on a compact set in $\mathbb{R}^d$, the fact that a term of the form $\prod_{i=1}^d \cos(\omega_i x_i + \psi_i)$ can be expressed as a sum of terms of the form $\cos(\alpha + \beta^T\mathbf{x})$, and the approximation of one-dimensional cosines by linear combinations of logistics, a one-dimensional version of the result. Note that since logistics can be approximated by step functions, the approximation results still apply with threshold units in the hidden layer.

Barron (1993a, b) and Jones (1992) show that (for smooth enough f) the L_2 approximation is $O(N^{-1/2})$ where N is the number of hidden units. A heuristic reason why projection methods might work well with modest numbers of hidden units is that the first stage allows a projection onto a subspace of $\mathcal{X}$ of much lower dimensionality, within which the approximation can be performed.

Kůrková (1992) even has an approximation result for networks with two hidden layers and linear output units in which all the weights except those leading to the outputs are fixed, so the function is linearly parameterized.

These approximation results show that one hidden layer is sufficient, but it may be more parsimonious to use fewer hidden units connected in two or more hidden layers. Further, (1) does not include linear functions as a special case, although they can be approximated by scaling down all the weights w_{ij} and up all the weights w_{jo}, using the fact that the centre of the logistic function is linear. It may be preferable to include direct 'skip-layer' connections from the inputs to the outputs, obtaining

$$y_k = \phi_o\left(\alpha_k + \sum_i w_{ik}x_i + \sum_j w_{jk}\,\phi_j\Big(\alpha_j + \sum_i w_{ij}\,x_i\Big)\right) \tag{2}$$

The approximation results apply to linear output units, but clearly they also apply to logistic output units if the component functions f_k are bounded away from zero and one. As we shall see, for classification it *may* be unimportant to approximate well at the extremes.

The approximation results are in general nonconstructive, and in practice the weights have to be chosen to minimize some fitting criterion, for example least squares

$$E = \sum_p \|t^p - y^p\|^2$$

where t^p is the target and y^p the output for the pth example pattern. Statisticians may feel unhappy about using least squares to fit a function f_k with target values $0, 1$. Other measures have been proposed, including 'maximum likelihood' (Hinton, 1989; Spackman, 1992; van Ooyen & Nienhuis, 1992; in fact minus the logarithm of a conditional likelihood, see §6.1) or equivalently the Kullback–Leibler divergence (Solla *et al.*, 1988; Bridle, 1990), which amount to minimizing

$$E = \sum_p \sum_k \left[t_k^p \log \frac{t_k^p}{y_k^p} + (1 - t_k^p) \log \frac{1 - t_k^p}{1 - y_k^p} \right] \tag{3}$$

Another form is considered in section 5.

The forms (1) and (2) and their multi-layer extensions are attractive in that the derivatives of a fit criterion E with respect to the weights can be calculated recursively from output to input using the chain rule, a procedure known as *back-propagation*. This can be extended to calculating the Hessian (Bishop, 1991a, 1992; Buntine & Weigend, 1993). For the Hessian the forms are more complex, but simplify somewhat for one hidden layer (including networks with skip-layer connections).

2.1 Algorithms

The classic algorithm of neural networks (also known as *back-propagation* or the *generalized delta rule*) is to take fixed steps in the direction of steepest descent in minimizing the criterion E,

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{4}$$

As the step control η is unchanged throughout the procedure, this has been seen as a form of stochastic approximation. Often 'momentum' is used, which can be seen as applying exponential smoothing to (4) leading to

$$\Delta w_{ij} = -(1-\alpha)\eta \frac{\partial E}{\partial w_{ij}} + \alpha(\Delta w_{ij})_{\text{old}} = -\eta' \frac{\partial E}{\partial w_{ij}} + \alpha(\Delta w_{ij})_{\text{old}}$$

The fit criterion E will in all cases be a sum over terms E^p for each example pattern. Rather than present all patterns, compute $\partial E^p / \partial w_{ij}$ for each, sum and then adjust the weights, we could adjust the weights after every pattern (presented in systematic order or randomly), a form even closer to stochastic approximation and in which momentum is desirable. (This is often known as the 'on-line' algorithm, in contrast to the 'batch' version.)

In my view too much attention has been spent in the literature on analogies with stochastic approximation (e.g. White, 1989a,b), which is not often used as a practical procedure in other problems. The problem is one of least-squares or of minimizing a criterion close to least squares. As such it is amenable to general unconstrained optimization techniques (Fletcher, 1987; Gill *et al.*, 1981). For realistic numbers of weights (up to 1000) quasi-Newton methods work well. For larger problems the storage of the approximate Hessian can be too demanding, and conjugate gradient methods or the limited-memory BFGS quasi-Newton method (Gill *et al.*, 1981, p. 150) can be used. (Shanno, 1990 reviews modern developments in limited-memory optimization.) In my experience these methods work much better on hard fitting problems than those based on empirically chosen variants of steepest descent.

There are practical difficulties in minimizing E, since experience shows local minima to be very common. For fully connected networks there will be a number of equivalent sets of parameter values giving the same function f by symmetric interchanges of the hidden units, a mild lack of identifiability. In practice we find many minima with similar values of E corresponding to very different internal structures in the 'black-box'. (These have been discovered by the *multi-start* algorithm, that is running the optimizer several times from randomly chosen starting sets of weights.) There has been some interest in global optimization procedures such as simulated annealing or a noisy variant of stochastic approximation (Kushner, 1987; White, 1989a; Styblinski & Tang, 1990; Gelfand & Mitter, 1991) but these seem impracticable for routine use.

With the classic algorithm it is necessary to decide when to stop, and computational considerations may necessitate stopping before other algorithms reach a local minimum. It is common practice to stop when a high proportion (often 100%) of the training set is classified correctly, for example when 95% of the training set has output within distance 0.05 of the target. Such stopping rules can change quite dramatically the success rate on a test set by avoiding over-fitting, and are a major potential for 'tuning' in comparative experiments. Another suggestion is to have a validation set (a test set used to tune the method but not to assess performance), and stop when the error rate on the validation set starts to increase. (In my experience this often stops far too soon, before the minimum rate on the validation set is achieved on the path of weights taken in minimizing E.)

The computation of both $f(\mathbf{x})$ and $\partial E/\partial w_{ij}$ can be performed in almost any vector/matrix manipulation package, and macros have been written in many such packages to 'simulate' neural networks. Neural network simulator packages are often special-purpose matrix manipulation packages, with graphical output facilities. For large problems more efficient code in a compiled language is needed. There are many freely-available implementations of neural network procedures.

3 Two-class classification problems

For simplicity, let us assume for this section that there are just two classes labelled 0 and 1. This is a quite common case (predicting presence/absence) as well as providing some considerable theoretical simplifications.

The decision-theoretic approach to classification is to assume a joint distribution for the features $\mathbf{X} \in \mathcal{X}$ and the class indicator $Y \in \{0, 1\}$, which we can specify by prior probabilities π_0, π_1 for the class and class-specific distributions of features $p(\mathbf{x} \mid y)$. The Bayes rule is then to choose the class with the higher posterior probability $p(y \mid \mathbf{x})$. Rather than look at the ratio $p(1 \mid \mathbf{x})/p(0 \mid \mathbf{x})$ let us first look at the difference

$$p(1 \mid \mathbf{x}) - p(0 \mid \mathbf{x}) = 2p(1 \mid \mathbf{x}) - 1 = 2[E(Y \mid \mathbf{X} = \mathbf{x}) - 0.5]$$

which shows that the classifier can be found by thresholding $f(\mathbf{x}) = E(Y \mid \mathbf{X} = \mathbf{x})$ at 1/2. Of course $\operatorname{logit} f(\mathbf{x}) = \log p(1 \mid \mathbf{x})/p(0 \mid \mathbf{x})$, recovering the log of the natural ratio, and suggesting considering f on logit scale. (It is easy to extend the discussion here to include general loss functions and 'doubt' and 'outlier' options, but for brevity these are omitted.)

In practice f will be an unknown function from $\mathcal{X}$ to $[0, 1]$, and so is an ideal candidate for approximation by a neural network with a single logistic output unit. This is the common practice, using least squares as the fitting criterion. For pure classification we are only interested in knowing whether $f(\mathbf{x}) > 0.5$, so an accurate approximation to f is only needed in the centre of its range. Note that our decision theory framework assumes that the densities are known, and uncertainties in parameters of the densities will affect the Bayes rule. For the time being we will ignore these and take a 'plug-in' approach. (See also §6.4.)

If we are prepared to make assumptions of normality and equal covariance matrices for $P(\mathbf{x} \mid y)$, it is well known that

$$f(\mathbf{x}) = \ell(\alpha + \beta^T \mathbf{x}) \tag{5}$$

that is, logistic discrimination (as in Anderson, 1982). This can be fitted by maximum likelihood, which corresponds to fitting a neural network without a hidden layer using (3) (e.g. McLachlan, 1992). Thus the neural network model can be seen as a perturbation of logistic discrimination which allows for general mean functions $E(Y \mid \mathbf{X} = \mathbf{x})$. Note that if the model is complex enough to allow us to classify the whole training set correctly, we can then drive E to zero by scaling up the weights to the output units, so the existence difficulties in parameter estimation for logistic regression (e.g. Lessafre & Albert, 1989) reappear with a vengeance.

Other ways of fitting a more general function f can be derived from the recent statistical literature. Consider the classification of rock crabs (Campbell & Mahon, 1974). There are five measurements on the carapaces of the animals. Rather than perform linear discrimination on the raw measurements, we might want to include ratios and so perform the analysis on log scale. Ideally, the classification procedure would itself choose the right scale. To do so we can consider a regression of the form

$$y = \alpha + \sum_{j=1}^{p} g_j(x_j) \tag{6}$$

for smooth but unknown functions g_j (Friedman & Silverman, 1989; Hastie & Tibshirani, 1990). If these are parameterized via splines with fixed knots, say, we have

$$y = \alpha + \sum_{j=1}^{p} \sum_{k=1}^{df_j} \beta_{jk} \phi_{jk}(x_j) \tag{7}$$

The smoothing procedure need not choose parameters by least squares, but if it does we have linear regression in an extended space $\mathcal{X}$ of features spanned by the functions $\phi_{jk}(x_j)$. We could include a logistic output unit and have

$$y = \ell\Big(\alpha + \sum_{j=1}^{p} g_j(x_j)\Big) \tag{8}$$

and fit by maximum likelihood, obtaining a generalized additive model (Hastie & Tibshirani, 1990). This idea applies to all the forms for f below.

These forms still do not allow interactions between the features in $\mathcal{X}$. Perhaps the simplest way to allow interactions is to allow linear combinations (projections):

$$y = \alpha + \sum_{j=1}^{r} g_j(\alpha_j + \beta_j^T \mathbf{x}) \tag{9}$$

which is *projection pursuit regression* (PPR: Friedman & Stuetzle, 1981). This is often written in the alternative form

$$y = \alpha + \sum_{j=1}^{r} \gamma_j \phi_j(\alpha_j + \beta_j^T \mathbf{x})$$

for normalized (zero mean, unit variance) smooth functions ϕ_j. Equation (9) has the same approximation properties as the single-hidden-layer neural network (by the proof sketched in section 2; *cf.* Diaconis & Shahshahani, 1984; Jones, 1987). Indeed, they are both based on projection directions, and in one dimension each can approximate the other. Zhao & Atkeson (1992) have an $O(N^{-1/2})$ approximation result for PPR over a larger class of functions that the analogous result (Barron, 1993a) for neural networks. Theoretical and empirical comparisons between PPR and neural networks are given by Hwang *et al.* (1992a,b, 1993).

Multivariate adaptive regression splines (MARS, Friedman, 1991) allows for interactions more explicitly by

$$y = \alpha + \sum_{m=1}^{M} \beta_m \prod_{k=1}^{K_m} \phi_{km}(x_{\nu(k,m)}) \tag{10}$$

where

$$\phi_{km} = [x - t_{km}]_+ \qquad \text{and} \qquad \phi_{k,m+1} = [t_{km} - x]_+$$

and t_{km} is an observed value of $x_{\nu(k,m)}$. The *degree* is the largest K_m, the specific choices of functions being made to allow fast least-squares fitting algorithms. Breiman (1991) has another class chosen with similar aims.

Another way to produce a flexible class of discrimination functions would be to parameterize $p(\mathbf{x} \mid y)$ as a mixture of a fixed set $\phi_j(\mathbf{x})$ of densities on $\mathcal{X}$, with weights p_{yj}. Then

$$p(\mathbf{x})[p(1 \mid \mathbf{x}) - p(0 \mid \mathbf{x})] = \sum_j [\pi_1 p_{1j} - \pi_0 p_{0j}] \phi_j(\mathbf{x}) \tag{11}$$

is a linear combination of the basis density functions, and could be estimated. As $p(\mathbf{x})$ is unknown, this cannot be used directly to estimate the discriminant from the training sample, but it has the same sign as $f - 1/2$ so can be used for classification. (Hall & Wand, 1988, suggest estimating (11) by kernel smoothing.)

This also suggests parametrizing f or logit f as a linear combination of basis functions. For a one-dimensional $\mathbf{x}$ splines are a natural choice. For higher dimensions *radial basis functions* or RBFs (Powell, 1987; Broomhead & Lowe, 1988; Moody & Darken, 1989; Poggio & Girosi, 1990,; Musavi *et al.*, 1992) have been advocated. These are approximations of the form

$$y = \alpha + \sum_j \beta_j G(\|\mathbf{x} - \mathbf{x}_j\|) \tag{12}$$

for prespecified centres $\mathbf{x}_j$. Girosi & Poggio (1990) and Park & Sundberg (1991) show approximation properties for this class, and Poggio & Girosi (1990) explore a number of issues in fitting. Examples of G proposed include the Gaussian $G(r) = \exp -r^2/2$ and the multiquadric $G(r) = \sqrt{(c^2 + r^2)}$. Alternatively we can use multi-dimensional splines (Wahba, 1990) to parameterize f.

Another approach to fitting $f(\mathbf{x}) = E(Y \mid \mathbf{X} = \mathbf{x})$ would be to use non-parametric regression (e.g. Härdle, 1990, 1991), using kernel functions or nearest neighbours. On the other hand, we could use the analogous density estimation techniques to estimate $p(\mathbf{x} \mid y)$ and compute the discriminant from $\pi_1 \hat{p}(\mathbf{x} \mid 1)/\pi_0 \hat{p}(\mathbf{x} \mid 0)$. This approach has been considered in monographs by Hand (1982) and Coomans and Broeckaert (1986). Estimating densities in high dimensions is notoriously difficult, and to use them for discrimination we are interested in regions where the densities of the two classes are comparable, usually in the tails of each. Methods of bandwidth selection, perhaps adaptively, should be tailored to the application. (Despite the extensive literature on density estimation, there appears to be very little known for this problem.) Altogether it appears to be better to estimate the classifier directly. The approach of using a mixture distribution to represent a high-dimensional density has perhaps been unfairly neglected in the statistical literature, with exceptions (e.g. Roeder, 1990). The ideas have been seen as part of the neural network field: Sprecht (1990a, b) and Tråvén (1991).

4 Space-partition methods

Perhaps the most non-parametric method of all is that of k-nearest neighbours, in which a majority vote is taken amongst the k nearest examples in $\mathcal{X}$ to the new unclassified example.

This can be justified either via nearest-neighbour density estimation, or as using the nearest-neighbour non-parametric regression of Y on $\mathbf{X}$, but amounts to assuming local constancy of f and calculating a piecewise constant approximation. There are a number of practical issues, including the choice of a metric in $\mathcal{X}$ (which also applies to multi-dimensional kernel methods), the computation of fast searches for neighbours, and the desirability of replacing the example patterns by a 'representative' set of points in $\mathcal{X}$. The latter is the aim of *Learning Vector Quantization* (Kohonen, 1990), which is considered to be a neural network method.

One idea I have found rather fruitful (for example in the tsetse fly example in Ripley, 1993) is to use other non-linear discriminant methods (especially classification trees) to help select the metric in $\mathcal{X}$, for instance omitting variables which do not appear in the tree, and weighting by their prominence if they do appear.

Classification trees also partition the space $\mathcal{X}$ into locally constant regions, often hypercubes parallel to the feature axes. There are a number of subtly different schemes for estimating trees, of which the best known to statisticians will be CART (Breiman *et al.*, 1984), but perhaps the most influential is Quinlan's *ID3* (Quinlan, 1979, 1983, 1986) from its seminal role in the machine learning community. Variants include those of Ciampi *et al,* (1987), Loh & Vanichsetakul (1988), Chou *et al.* (1989), Crawford (1989), Chan & Bao (1991), Chou (1991), Gelfand & Delp (1991), Gelfand *et al.* (1991), Buntine (1992), Clark & Pregibon (1992) and Fayyad & Irani (1992). The idea is to choose repeatedly a feature (or combination of features) and to split the space $\mathcal{X}$ on that value. The methods differ in allowing multiway splits or forcing binary splits and in how the best split is computed. More importantly, they differ in how to stop growing the tree and how to prune it to avoid over-fitting.

Some tree-construction methods allow linear or Boolean combinations of variables to define a split. Guo & Gelfand (1992) even allow a simple neural network at each node to select a non-linear combination of the feature variables there.

A tree can be expressed as a set of decision rules, one leading to each leaf. In his developments of *ID3*, now called *C4.5* Quinlan (1990, 1993) simplifies these rules to increase generalization. Rules can also be extracted from neural networks (e.g. Gallant, 1993, Chapter 17). Other, direct, methods of inducing decision rules are considered in the field of machine learning. They do have the considerable advantage of interpretability and so are particularly suitable for understanding the differences between the classes (see, for example, King *et al.*, 1992). They can also incorporate prior logical information much more easily than the methods described here.

Tree-based classifiers have been related to neural nets by Brent (1991) and Sethi (1990,1). They had essentially the same idea. Suppose we have binary tree with t splits and so $t + 1$ leaves. The splits compute and threshold t functions of $\mathbf{x}$. Think of a network in which the first hidden layer of t nodes computes the split functions, and the second hidden layer has a node for each of the $t + 1$ paths to a leaf which ANDs the (possibly inverted) outputs of the nodes on the path. The output layer then ORs leaves with the same class.

5 More than two classes

In this section we will specialize to a target set $\mathcal{Y} \subset \mathbb{R}^K$, and let $C \in \{1, \dots, K\}$ denote the true class. For each $\mathbf{x} \in \mathcal{X}$ the classifier gives $y = f(\mathbf{x}) \in \mathcal{Y}$. We then have to decide how to turn y into a classification. For each class k the natural target for y_k is one on that class, zero elsewhere, so we might want $f_k(\mathbf{x})$ to estimate $E(Y_k \mid \mathbf{X} = \mathbf{x}) = p(C = k \mid \mathbf{X} = \mathbf{x})$ where Y_k is the indicator of membership of class k. In that case the (approximate) Bayes rule is to choose the class k with the largest value of $f_k(\mathbf{x})$. As this depends only on the relative magnitudes of the components, it is invariant under any increasing transformation of f. Thus we can establish a general principle to choose the largest component of the classifier. Another idea (which is used in most of the neural network packages I have studied) is to

choose the exact classification (probability one on class k, zero on the rest) which is nearest in the Euclidean space $\mathcal{Y}$ to the observed y. In some schemes, 'doubt' is reported if y is not closer than some pre-specified tolerance (such as 0.2) to any exact classification. With the conventional targets, the nearest target *does* correspond to the largest component, but this scheme does not correspond to the Bayes rule with 'doubt' option, which declares 'doubt' if no component exceeds a threshold.

If we regard the standard neural-network procedure with two classes as non-linear logistic regression, the natural extension to more than two classes is a log-linear model for the posterior probabilities, which gives rise to

$$p(k \mid \mathbf{x}) = \frac{\exp f_k(\mathbf{x}; \hat{\theta})}{\sum_j \exp f_j(\mathbf{x}; \hat{\theta})} \tag{13}$$

and (conditional) negative log-likelihood

$$E = \sum_k t_k \log \frac{t_k}{p(k \mid \mathbf{x})} \tag{14}$$

Here f can be any flexible family of multivariate functions such as neural networks with *linear* output units. (As (14) is not least-squares, this procedure is less suitable for families such as MARS which are designed for fast least-squares fitting.) In the neural network literature the log-linear approach is often called *softmax* after Bridle (1989, 1990).

Note that this differs from the commonest approach using *logistic* output units, which can be seen as fitting separate logistic models for each class versus the rest, without imposing consistency conditions on the posterior probabilities.

The component functions f_k can be estimated as individual regressions, or the function f by multivariate regression. The difference will only appear in the more general forms of regression of section 3, in which we have the choice of using the same basis functions for each regression or not. Feedforward neural networks are naturally expressed with a common set of intermediate values in the hidden layer, but of course the weights could be chosen with a pattern of zeros so that each output has its own set of hidden units. For projection pursuit regression we could have any of

$$\begin{aligned}
y_k &= \alpha_k + \sum_{j=1}^{r} \gamma_{jk} \phi_j(\alpha_j + \beta_j^T \mathbf{x}) \\
y_k &= \alpha_k + \sum_{j=1}^{r} \gamma_{jk} \phi_{jk}(\alpha_j + \beta_j^T \mathbf{x}) \\
y_k &= \alpha_k + \sum_{j=1}^{r} \gamma_{jk} \phi_{jk}(\alpha_{jk} + \beta_{jk}^T \mathbf{x})
\end{aligned}$$

the last corresponding to completely separate regressions. The S-PLUS implementation uses the first, which is the most parsimonious and hence the most useful. The second does allow the idea of using projections to pick a r-dimensional flat (translated subspace) in $\mathcal{X}$, and approximating within that flat.

For two classes linear regression gives the sample linear discriminant if the classes are properly represented in the training sample. We can think of the linear regression as the best (least-squares) linear approximation to the posterior probabilities. As a principle of classifier design this has been used (Duda & Hart, 1973; Devijver & Kittler, 1982; Fukunaga, 1990) under the name of *minimum (mean) square error* classifiers. Unlike the linear discriminant, that procedure classifies by the nearest target or equivalently the largest component. The difference is minor with two classes; the linear combination is the same, but the threshold in linear discrimination may differ from 0.5 (although usually neglibly, Ripley & Hjort, 1994 §3.3).

In unpublished work, Breiman & Ihaka (1984) showed how to use regression to find the linear classifier for more than two classes. Their idea is to replace the targets y by a $(K-1)$-dimensional score $\theta(y)$ and to fit by least squares in the space of θ-scores. (Think of putting the three targets for a three-class problem around the circumference of a circle in $\mathbb{R}^2$.) Then if we optimize over the choice of scores as well as the parameters in the regressions, the sample linear discriminant is recovered up to computable scale factors. What is not made clear in their work is that this process is equivalent to performing a linear discriminant analysis in the space of fitted values from the regressions of the class indicators on $\mathbf{x}$ (Ripley & Hjort, 1994).

The motivation of the Breiman-Ihaka approach was to use semi-parametric rather than linear regression. We can set up a flexible semi-parametric classifier, by applying the linear classifier to the space of fitted values, or by optimizing over the scores whilst fitting the function $f(\mathbf{x})$ by least-squares. (If the regression is linear in y, such as a neural network with linear output units, the optimization over scores can be done without re-fitting the regression.)

This approach allows us to approximate optimally the discriminant in $J < K-1$ dimensions using the first J discriminants. This can be used to present the data on the first few generalized discriminant variables, and that can be revealing (see §6.3 and figure 10). However, there can be difficulties with non-linear methods fitting the training set extremely well, and hence the information in the residuals reflects the stopping rule rather than a genuine pattern of lack of fit. Even when the lack of fit is genuine, it can be dominated by outliers, and a robust linear discriminant analysis applied to the fitted values can be helpful (Ripley & Hjort, 1994).

These ideas have been independently taken up by Hastie *et al.* (1992a, b), who extend them to penalized regression methods such as additive models with smoothing spline smoothers.

6 Issues in function-fitting

The following issues apply to all the methods that compute and maximize over discriminant functions.

6.1 Parameter estimation

'Maximum likelihood' estimation of parameters in classifier functions $f(\mathbf{x};\theta)$ is mentioned frequently in the literature. We need to consider the likelihood a little more carefully. That usually used for generalized linear models corresponds to fixing the samples from $\mathcal{X}$ and only considering the uncertainty in the responses. For discrimination it is more natural to consider either that $(\mathbf{X}_i, Y_i)$ are random samples from $p(\mathbf{x}, y)$, *mixture sampling*, or that the numbers in each class are fixed, so that *separate samples* are taken from each $p(\mathbf{x} \mid y)$. It is known (e.g. McLachlan, 1992) that these sampling schemes reduce to the same maximum likelihood problem *provided*

$$p(\mathbf{x}, y) = p_{\theta}(y \mid \mathbf{x}) p(\mathbf{x})$$

that is that the marginal distribution of the $\mathbf{X}_i$'s is independent of the parameters. This is *not* the case for the normal model for linear discrimination. The maximum likelihood method (3) of fitting the logistic (5) is based on the conditional likelihood of Y given $\mathbf{X}$ and so may be inefficient relative to estimating the parameters in the joint distribution using the full likelihood. McLachlan, (1992, §8.5) summarizes work on assessing this inefficiency. This work is asymptotic, and for classification the effect of parameter estimation is asymptotically

negligible, so the results are about fine distinctions. In practice we are a very long way from asymptotics, and the performances can show important differences.

The Fisher (1936) sample linear discriminant is fully efficient for the normal model, and may be calculated via linear regression. However, the logistic form has wider validity and may be more efficient under other models for $p(\mathbf{x}, y)$. Thus the regression approach gives a fully efficient estimator of the classifier under the equal-covariance matrix normal model, but there are no claims that this extends to the non-linear versions.

6.2 Degree of smoothness and complexity

The aim is to achieve the best classifier on the whole set $\mathcal{X}$, not just on the training set, and with a flexible fitting procedure for f there is the ever-present danger of over-fitting. Stopping rules, as we have seen, are an *ad hoc* way to avoid this. In section 7 we will consider the general problem of doing well over all of $\mathcal{X}$; here we look at ways to choose a smooth f.

One way to ensure that f is smooth is to restrict the class of estimates, for example by using a limited number of spline knots or radial basis functions. Another way is *regularization* (Poggio & Girosi, 1990; Bishop, 1991b, 1993) in which the fit criterion is altered to

$$E + \lambda C(f)$$

for example, with a penalty C on the second derivatives of f such as

$$\frac{1}{p} \sum_{i,o} \frac{\partial^2 y_o}{\partial x_i^2}$$

Such norms are used in the derivation of splines and RBF networks. *Weight decay*, specific to neural networks, uses as penalty the sum of squares of the weights w_{ij}. (This only makes sense if the inputs are rescaled to range about $[0, 1]$ to be comparable with the outputs of internal units.)

These ideas can be seen as generalizations of ridge regression and shrinkage methods, and like them derived from a Bayesian perspective, by taking a prior proportional to $\exp -\lambda C$. In the neural network field this has been taken up by MacKay (1992) following Gull (1989). However, they appear to regard the choice of the parameter λ as a model-choice, and so select a maximum of its posterior, and plug the estimate in. A wider ranging discussion is given by Buntine & Weigend (1991) who advocate a hyperprior on λ and integration rather than maximization. For weight decay, the prior corresponds to independent Gaussian distributed weights of mean zero and variance $\sigma_w^2 = 2/\lambda$. Then MacKay fixes σ_w^2 whereas Buntine & Weigend integrate over a non-informative hyperprior. At least in this case, replacing the average over the posterior for σ_w^2 by the integrand evaluated at its maximum seems to be a rather poor approximation.

An issue related to the degree of smoothing is how many terms to allow in the model; for instance how many hidden units in a neural network or how many functions g_j in projection pursuit regression (9). These do not necessarily control the smoothness (which in PPR is primarily controlled by the smoothness of the functions) as much as the dimensionality of the approximation.

A number of techniques have been proposed to select the model complexity. In PPR the number of terms in (9) is controlled by fitting additional terms one at a time *keeping all other parameters fixed*, then using simultaneous fitting and backwards selection. The method of *cascade correlation* (Fahlman & Lebiere, 1990) is similar in adding hidden units one at a time, but omits the simultaneous fitting (although this could easily be done). In fact, the hidden units are each added in a new layer, connected to all earlier units, whereas in PPR a new term is fitted to residuals from the existing fit. Other constructive techniques are described by Gallant (1993, Chapter 10).

Another approach is to penalize model complexity as in Mallow's C_p and Akaike's *AIC*. One difficulty with such criteria for neural-network models is to quantify the appropriate complexity, which is not simply the number of parameters (Moody, 1992; Murata *et al.*, 1991; Amari & Murata, 1993; Barron, 1993b).

Many methods ultimately resolve their free parameters for model smoothness and complexity by some form of cross-validation, either using a separate validation set or by dividing (and rotating) the training set. The objective used in cross-validation is often least-squares, even though the actual goal is classification for which it is more important to know the discriminant well near its threshold(s). My impression is that the use of cross-validation ideas in these non-linear and highly parameterized problems is not fully understood; in practice the variability of cross-validation estimates causes difficulties.

6.3 Robust fitting

It is common practice to find that a small number of the example patterns are wrongly classified, and thus scored as one rather than zero or *vice-versa*. For example, in the iris data (Anderson, 1935; Fisher, 1936) the non-linear discriminant analysis given in Ripley (1993) throws doubt on the classification of 3 of the 150 examples.

Whereas the maximum error that can be made is one, the fitting methods will usually fit the training set very accurately, and so the error will be many times the residual standard error. From the least-squares point of view it is essential to use a robust criterion if over-fitting is to be avoided. From the point of view of logistic regression we have the possibility of 'bit-flip' errors, for which resistant methods have been proposed (Pregibon, 1982; Copas, 1988; Carroll & Pederson, 1993).

Most of the discussion of robustness has been for problems with small numbers of parameters compared to the number of data points, using asymptotic expansions, and it would be very helpful to have proven methods for semi-parametric models. The issues are rather different, as a mis-classification can have only local effect on the fitted surface, and outliers in the $\mathcal{X}$ space can have only very limited leverage. The major impact of errors in the training set is on the choice of model complexity and on the assessment of the true performance level.

6.4 The predictive Bayesian approach

In the Bayesian approach parameter estimation is not an issue in itself; we need to compute $p(y \mid \mathbf{x}, \mathcal{T})$ for the training set $\mathcal{T} = \{(\mathbf{X}_i, Y_i), i = 1, \dots, p\}$ and hence compute the classifier. Here $\mathbf{x}$ is an arbitrary future observed pattern. Suppose we have parameters θ in the model for $p(\mathbf{x}, y; \theta)$. Then

$$p(\theta \mid \mathcal{T}) \propto p(\theta) p(\mathcal{T} \mid \theta) = p(\theta) \prod_{i=1}^{p} p(\mathbf{x}; \theta) f_{y_i}(\mathbf{x}; \theta)$$

from which we can deduce

$$p(y \mid \mathbf{x}, \mathcal{T}) = \int p(\theta, y \mid \mathbf{x}, \mathcal{T})\, d\theta = \int p(y \mid \mathbf{x}; \theta) p(\theta \mid \mathcal{T})\, d\theta$$

This approach is sometimes called the predictive approach (Aitchison & Dunsmore, 1975). It may appear computationally prohibitively expensive for neural network models for $p(y \mid \mathbf{x}; \theta)$, but Markov chain Monte Carlo methods have recently been used by Neal (1992, 1993) to compute the predictive distribution in a small (regression) example. As in section 6.2 we may also use integration over hyperparameters in $p(\theta)$, which will affect the formula for $p(\theta \mid \mathcal{T})$ and may need a further level of Monte Carlo integration.

Even when finding the full predictive distribution is computationally prohibitive, we can use the theory to suggest useful hints. For example, when fitting neural networks

we often encounter local minima. The theory suggests the correct way to proceed is to average the predictive probabilities given by the solutions, weighted by the probability of the peak around $p(\hat{\theta} \mid \mathcal{T})$, and that can be approximated by the frequency with which that solution occurs in randomly started optimization runs. (A version of this idea for trees is implemented by Buntine, 1992).

7 Assessing generalization

The real test of how a classifier f performs is its behaviour over the whole of the feature space $\mathcal{X}$, typically measured by the average misclassification rate

$$P(f(\mathbf{X}) \neq \arg\max_k P(C = k \mid \mathbf{X}, \mathcal{T}))$$

averaged over the unconditional distribution $p(\mathbf{x}) = \sum \pi_k p(\mathbf{x} \mid k)$, although we may also wish to consider some form of 'worst-case' behaviour over $\mathbf{x} \in \mathcal{X}$. Following the psychological literature this idea is referred to as *generalization ability*.

Of course, asymptotically in the number p of example patterns, we would expect the parameter estimation to be irrelevant provided f is estimated in a finite-dimensional space, or in a space growing sufficiently slowly with p, and theorems to this effect are given by White (1989b, 1992), although as he states that at least one hundred times as many example patterns as parameters are needed for these results to be relevant, they seldom will be. Geman *et al.* (1992) discuss this in the general context of non-parametric regression, but point out (pp. 33, 44) the irrelevance of asymptotic theory to most practical pattern-recognition problems, in which we have to establish a compromise between bias and variance in estimating the true classifier. Stone (1982) has a general treatment of the rates of approximation which can be achieved in non-parametric regression. Donoho & Johnstone (1989) consider a rather restricted class of two-dimensional functions, and show that projection-based and local (kernel-based) approximation methods have domains in which each is much better suited than the other, so there will be no universally best methods. We might hope that asymptotic theory would allow us to approximate $P(C = k \mid \mathbf{x}, (\mathbf{x}_i, y_i), i = 1, \dots, p)$ by averaging $P(C = k \mid \mathbf{x}, \theta)$ locally about optimal fitted θ (Buntine & Weigend, 1991), but this seems a forlorn hope given the difficulty of maximization and many good local minima found in practice. For large enough training sets we can apply asymptotic theory as usual. (Buntine & Weigend say 'the sample size should be at least some factor of the number of weights in the network'—I presume they meant multiple rather than factor.) Other forms of approximation are those of Rissanen (1987) and Wallace & Freeman (1987).

In the remainder of this section we sketch a method to tackle the scaling of the generalization ability of sub-optimal procedures with the size of the training set, which can be used to provide sufficient bounds on the size of the training set required. Barron has developed complexity measures (surveyed in Barron, 1993b) based on this method.

Consider a two-class classification problem, and let f be the correct classification function to be learned from p examples. For any function $\phi\colon \mathcal{X} \to \{0, 1\}$ define

$$\begin{aligned} g(\phi) &= P(\phi(\mathbf{X}) \neq f(\mathbf{X})) \\ g_p(\phi) &= \#(\phi(\mathbf{x}_i) \neq f(\mathbf{x}_i), i = 1, \dots, p)/p \end{aligned}$$

where $\mathbf{X}$ is a random element of $\mathcal{X}$ from a pre-specified distribution. Thus $g(\phi)$ is the misclassification rate for classifier ϕ, and $g_p(\phi)$ is the apparent misclassification rate on the training set. We know that $g_p(\hat{f})$ will be negatively biased for $g(\hat{f})$, but we can use a worst-case bound on their difference.

Consider a training set randomly chosen from the same distribution as the test cases. We have (Blumer *et al.*, 1989; Abu-Mostafa, 1989; Vapnik, 1982)

$$\delta = P(\max_\phi |g_p(\phi) - g(\phi)| > \epsilon) \leq 4\left[(2p)^{d_{VC}} + 1\right] e^{-\epsilon^2 p/8} \qquad (15)$$

where $d < \infty$ is the Vapnik–Chervonenkis dimension of the space of exactly representable binary functions of the classifications of p examples. (There are other similar bounds, some linear in ϵ.) Thus

$$|g(\hat{f}) - g_p(\hat{f})| < \epsilon$$

with high probability if the right hand side of (15) is small, which it will be for large p. (This can be translated to a sufficient bound on p of order $(8d/\epsilon)\log(13/\epsilon)$.) In that case, if we can do well on the training set, we can also have confidence that usually we will do well on future examples. Note that the bound is independent of the distribution over $\mathcal{X}$. There are also necessary bounds on p of order $d/32\epsilon$ when $g_p(\hat{f}) = 0$, that is the training examples are fitted exactly.

Baum & Haussler (1989) applied these ideas to feedforward networks with M *threshold* nodes and W weights. They showed that

$$d \le 2W \log_2(eM)$$

and for a single-hidden-layer network with N inputs and H nodes in the hidden layer (M also includes output nodes) that

$$d \ge 2\lfloor H/2 \rfloor N \approx W$$

for large H. Thus the size of the training set needs to be of order W/ϵ to have a high probability of success rate at most ϵ worse on the test set than on the training set.

This is an example of Valiant's (1984) concept of *PAC-learning*, where *PAC* stands for 'probably almost correct'. The idea is to seek a procedure which with arbitrarily high probability $1-\delta$ has misclassification rate less than ϵ and takes polynomial time in the size of the problem and $1/\delta, 1/\epsilon$. We have the stark result (Judd, 1990, corollary 22)[1]

> 'Networks cannot generalize.'

the apparent contradiction being resolved by the restriction on the class of classifiers that the Baum–Haussler network can learn, and the absence of an explicit algorithm for training such networks. At present positive applications of *PAC* are confined to functions f which can be represented by the network and to networks of threshold units. An empirical study (Cohn & Tesauro, 1992) shows error rates far below the upper bounds even in such networks. Baum (1990) gives theoretical support that the bound over all distributions of examples is too conservative. For neural networks with logistic units the VC dimension has only very recently been shown to be finite (Macintyre & Sontag, 1993) and no bounds are known.

8 Examples

Our three examples illustrate some of the intricacies of non-linear classification. They illustrate two extremes of the range of example problems. Few of the computations took longer than 2 minutes on a Sun SparcStation IPC, the exceptions being some of the cross-validation runs and the larger MARS and neural network experiments on the full sonar data which took up to 20 minutes each.

Performance is assessed on separate test sets. It would be possible to assess performance by more sophisticated techniques from the training set, for example using estimated posterior probabilities (Fukunaga, 1990) or bootstrapping (Weiss & Kulikowski, 1991). However, the comparisons would be distorted by biases in these techniques which appear to differ considerably between types of classifiers.

[1] Judd has a technical condition on *loading* which some reviewers see as restrictive.

8.1 Synthetic data

The test set is illustrated in figure 2. Each population is an equal mixture of two two-dimensional normally distributed populations, and the two populations are equally likely. The boundary of the Bayes' rule is shown, calculated from the known population densities.

The decision boundaries found for a training set of size 250 are shown in figures 3–6. The error rates were estimated using a further sample of size 1000 and are given in the table. Even with this size of test set the quoted rates have a standard error of about 1%. This example is rather noisy, and this favours the smoothing methods over the space-partition methods. Notice the improvement that editing (Devijver & Kittler, 1982) gives for the nearest-neighbour rule; unfortunately editing is only particularly effective in small numbers of dimensions with large training sets.

Figure 4 shows the dramatic effect of weight decay (set at $\lambda = 5 \times 10^{-3}$) in smoothing the discriminant function, and this is reflected in considerably better performance in the table. Note that for this training set PPR appears to fit rather better than neural networks, and that the non-differentiability of the function fitted by MARS shows clearly in figure 6.

method	**error rate %**
Bayes rule	8.0
linear discriminant	10.8
logistic discriminant	11.4
quadratic discriminant	10.2
1-nearest neighbour	15.0
3-nearest neighbour	13.4
5-nearest neighbour	13.0
ditto after multiedit	8.3
neural network with 3 hidden units	9.4
ditto without weight decay	11.1
neural network with 6 hidden units	9.5
PP regression (r = 2)	8.6
MARS (degree = 1)	9.3
MARS (degree = 2)	9.6
classification tree	10.1
LVQ (12 representatives)	9.5

As far as possible the parameters (such as the number of LVQ representatives) were chosen automatically or the defaults were used; in some cases I did some preliminary testing on a further validation set of size 250.

8.2 Sonar data

Gorman & Sejnowski (1988) published an early application of neural networks, which is one of White's (1989b) examples of

> '... solutions to problems that had previously withstood conventional attacks as well as quick and reliable solutions to problems that had previously yielded comparably effective solutions grudgingly.'

The data are sonar signals bounced off a metal cylinder (a mine, 111 signals) and off a roughly cylindrical rock (97 signals). The signals are energies in each of 60 wavebands. The patterns were ordered in time as the boat moved over the objects, and hence also by incidence angle. The data were divided into training and test sets of size 104 each using a cluster analysis to ensure as even matching as possible. (The training set is split 49/55, the test set 62/42, which violates the sampling assumptions of many of the methods.)

Gorman & Sejnowski fitted a 60–24–2 neural net (coding the outputs for the two classes separately, so this is a fully-connected network with 24 units in the hidden layer) which has 1514 parameters, and found an average test-set error rate of 10.8%.

The mean responses for the two classes are shown in figure 7. This suggests that the responses are smooth, as confirmed by figure 8, and so the 60 channels of data are not independent. I averaged them in bands of 3, to make 20 composite channels. The results are given in the table, in column (a) for all 60 variables and column (b) for 20 variables. This training and test set are artificially similar, so the results were repeated for training and test sets made up of alternate samples from the dataset, shown in columns (c) (60 variables) and (d) (20 variables).

method	**errors out of 104**			
	(a)	(b)	(c)	(d)
linear discriminant	25	20	28	20
nearest neighbour	9	8	21	17
ditto on rescaled vars	4	6	17	20
neural network with 2 hidden	17	21	19	23
neural network with 6 hidden	12	20	17	20
neural network with 12 hidden	11	20	15	19
PP regression (r = 2 or 3)	29	24	23	17
MARS (degree = 1)	30	23	21	21
MARS (degree = 2)	30	20	26	27
classification tree	29	21	28	26

The second set of nearest-neighbour results were obtained by rescaling each channel to unit within-group variance on the training set. The results for neural network are averages over 5 sets of starting weights. Skip-layer connections were included, and weight decay and entropy fitting were used in the training. The number of weights is large, exceeding the number of examples except for 20 variables and 2 hidden units. The extraordinary behaviour of the large nets in this example (with 60 variables and 12 hidden units there are 805 weights) seems to reflect a need to approximate the nearest-neighbour classifier, rather than finding any real structure in the data.

We then used the first 60% of each run of the boat to train, the rest to test, to see if any real generalization had been achieved:

method	**errors out of 84**
linear discriminant	39
nearest neighbour	42
1-nn on rescaled vars	41
neural network with 2 hidden	48,50,37,36,36
PP regression (r = 3)	36
MARS (degree = 1)	36
classification tree	29

which clearly it had not been. The five neural net runs are given in decreasing order of fit to the training data. These results are for 20 variables, but similar results were obtained for all 60 channels. The second problems is harder in that it involves a larger degree of extrapolation (in so far as this is pertinent in such high-dimensional spaces).

8.3 Forensic glass

This set of data was collected by B. German on 214 fragments of glass, and taken from Murphy & Aha (1992). Each has a measured refractive index and composition (weight percent of oxides of Na, Mg, Al, Si, K, Ca, Ba and Fe). The fragments were originally

classed as seven types, but some are infrequent or absent, and I re-grouped them as window float glass (70), window non-float glass (76), vehicle window glass (17) and other (22), omitting headlamp glass which seemed rather distinct in some preliminary linear discriminant analysis. The dataset was split randomly, resulting in a training set of size 89 and a test set of size 96.

The results are given in the table. The neural network results are averaged over 5 runs. They show both log-linear (softmax) models, separate logistic models fitted via (3) and and least-square fits with linear output units. The PPR and MARS results used least-squares fitting and post-processing by linear discriminant analysis. As some errors are more common and perhaps less serious than others, in the second figure confusion between the two types of window glass is allowed.

method	**error rate %**	
linear discriminant	41	22
nearest neighbour	26	17
neural network with 2 hidden	38	17
neural network with 6 hidden	33	16
ditto logistic outputs	39	17
ditto via LDA	41	19
PP regression ($r = 2, 5$)	40	19
MARS (degree = 1)	37	17
MARS (degree = 2)	31	19
classification tree	28	15
LVQ (11 representatives)	31	17

There is little to choose between the non-linear methods, most of which show slight but useful gains over linear discrimination. It seems that the Breiman-Ihaka approach is inferior to log-linear models in this example (and in most others of my experience) but results of log-linear PPR and MARS are unavailable from lack of suitable software.

Figures 9 and 10 show linear and generalized (via MARS) discriminants for this dataset.

9 Challenges

Our examples are perhaps a little unrepresentative in that linear methods do moderately well, and the gains by non-linear methods are less spectacular than in, for example, the tsetse fly problem of Ripley (1993). The gains are however worthwhile, and in our examples achieved for rather modest amounts of computation. It seems to me that the statistical community has not thought enough about how to use (well) large amounts of computation. Most datasets are very expensive to collect relative to both human and computer costs of analysis, and we ought to be thinking how best to spend 24 hours of workstation time on classification problems. (Some approaches appear to need years of computer time, including Monte Carlo versions of a full Bayesian analysis in realistic problems.)

On the other hand, training data are rarely plentiful and often expensive to classify, so it is important for estimation to be statistically efficient. Here classification has been studied much less than regression, not least because asymptotics are not very relevant, and more insight is needed. The value of modelling the whole distribution $p(\mathbf{x}, y)$ rather than $p(y \mid \mathbf{x})$ alone remains unclear, except where unclassified examples are readily available.

The assessment of performance remains problematical, not least because the training set may not be fully representative of the class of patterns to be classified. For example, in symbol recognition styles of symbols change with time, and updating methods for the parameters can be very helpful. It is far too common in the literature for classifiers to be tuned to look good on the test set, yet tuning non-linear methods is crucial to their success.

What is not in doubt is that there are very many classification problems in which non-linear methods can considerably out-perform classical methods, and as automated data collection increases such problems will continue to proliferate. Neurobiologists and psychologists are also looking for insights into the success of humans in solving complex pattern recognition problems apparently effortlessly.

Acknowledgements

I am grateful to Andrew Barron, Chris Bishop, Wray Buntine, Jerry Friedman, Trevor Hastie, Nils Hjort, Steve Muggleton, Radford Neal, Rob Tibshirani and Bob Williamson for enlightening conversations and comments on this subject, as well as for pointing me to as yet unpublished work. Ross Ihaka supplied a copy of Breiman & Ihaka (1984), and Rob Tibshirani kindly provided the Hastie–Tibshirani S and Ratfor code for MARS and optimal scoring.

References

Abu-Mostafa, Y. S. (1989) The Vapnik–Chervonenkis dimension: information vs complexity in learning. *Neural Computation* **1**, 312–317.

Aitchison, J. and Dunsmore, I. R. (1975) *Statistical Prediction Analysis.* Cambridge: Cambridge University Press.

Amari, S.-I. and Murata, N. (1993) Statistical theory of learning curves under entropic loss criterion. *Neural Computation* **5**, 140–153.

Amit, D. J. (1989) *Modeling Brain Function.* Cambridge: Cambridge University Press.

Anderson, E. (1935) The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, **59**, 2–5.

Anderson, J. A. (1982) Logistic discrimination. In *Handbook of Statistics 2: Classification, Pattern Recognition and Reduction of Dimensionality* eds P. R. Krishnaiah and L. Kanal, Amsterdam: North Holland, pp. 169–191.

Barron, A. R. (1993a) Universal approximation bound for superpositions of a sigmoid function. *IEEE Trans. Inform Theory* **39**, 1–16.

Barron, A. R. (1993b) Approximation and estimation bounds for artificial neural networks. *Machine Learning* (to appear).

Baum, E. B. (1990) The perceptron algorithm is fast for non-malicious distributions. *Neural Computation* **2**, 248–260.

Baum, E. B. and Haussler, D. (1989) What size net gives valid generalization? *Neural Computation* **1**, 151-160.

Bishop, C. (1991a) A fast procedure for retraining the multilayer perceptron. *Int. J. Neural Systems* **2**, 229–236.

Bishop, C. (1991b) Improving the generalization properties of radial basis function neural networks. *Neural Computation* **3**, 579–588.

Bishop, C. (1992) Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Computation* **4**, 494–501.

Bishop, C. (1993) Curvature-driven smoothing: a learning algorithm for feedforward networks. *IEEE Trans. Neural Networks*.

Blumer, A., Ehrenfeucht, A., Haussler, A. and Warmuth, M. (1989) Learnability and the Vapnik-Chervonenkis dimension. *J. Assoc. Comput. Mach.* **36**, 926–965.

Breiman, L. (1991) The Π–method for estimating multivariate functions from noisy data. *Technometrics* **33**, 125–160.

Breiman, L. and Ihaka, R. (1984) Nonlinear discriminant analysis via ACE and scaling. *Technical Report #40* U.C. Berkeley Statistics Department.

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984) *Classification and Regression Trees.* Monterey, CA: Wadsworth and Brooks/Cole.

Brent, R. P. (1991) Fast training algorithms for multilayer neural nets. *IEEE Trans. Neural Networks* **2**, 346–354.

Bridle, J. S. (1989) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neuro-computing: Algorithms, Architectures and Applications* eds F. Fougleman-Soulie and J. Hérault. New York: Springer.

Bridle, J. S. (1990) Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems 2 (Denver, 1989)* (ed D. S. Touretzky) pp. 211–217. San Mateo, CA: Morgan Kaufmann.

Broomhead, D. S. and Lowe, D. (1988) Multivariable functional interpolation and adaptive networks. *Complex Systems* **2**, 321–355.

Buntine, W. L. (1992) Learning classification trees. *Statistics and Computing* **2**, 63–73.

Buntine, W. L. and Weigend, A. S. (1991) Bayesian back-propagation. *Complex Systems* **5**, 603–643.

Buntine, W. L. and Weigend, A. S. (1993) Calculating second derivatives on feed-forward networks. *IEEE Trans Neural Networks*.

Campbell, N. A. and Mahon, R. J. (1974) A multivariate study of variation in two species of rock crab of genus *Leptograpsus*. *Aust. J. Zool.* **22**, 417–425.

Carroll, R. J. and Pederson, S. (1993) On robustness in the logistic regression model. *J. Roy. Statist. Soc. B* **55**, 693–706.

Chan, C. and Bao, J. (1991) On the design of a tree classifier and its application to speech recognition. *Int. J. Pattern. Recogn. Artif. Intell.* **5**, 677-692.

Chou, P. A. (1991) Optimal partitioning for classification and regression trees. *IEEE Trans. PAMI* **13**, 340–354.

Chou, P. A., Lookabaugh, T. and Gray, R. M. (1989) Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Inf. Theory* **35**, 299–315.

Ciampi. A., Chang, C.-H., Hogg, S. and McKinney, S. (1987). Recursive partitioning: a versatile method for exploratory data analysis in biostatistics. In *Biostatistics* (ed I. B. McNeil and G. J. Umphrey). Dordrecht: Reidel.

Clark, L. A. and Pregibon, D. (1992) Tree-based models. In *Statistical Models in S* (eds J. M. Chambers and T. J. Hastie) pp. 377–419. Pacific Grove, CA: Wadsworth and Brooks/Cole.

Cohn, D. and Tesauro, G. (1992). How tight are the Vapnik-Chervonenkis bounds? *Neural Computation* **4**, 249–269.

Coomans, D. and Broeckaert, I. (1986) *Potential Pattern Recognition.* Letchworth: Research Studies Press.

Copas, J. B. (1988) Binary regression models for contaminated data (with discussion). *J. Roy. Statist. Soc. B* **50**, 225-265.

Crawford, S. L. (1989) Extensions to the CART algorithm. *Int. J. Man-Machine Studies* **31**, 197–217.

Cybenko, G. (1989) Approximation by superpositions of a sigmoidal function. *Math. Control Systems Signals* **2**, 303–314.

Devijver, P. A. and Kittler, J. V. (1982) *Pattern Recognition. A Statistical Approach.* Prentice-Hall, Englewood Cliffs.

Diaconis, P. and Shahshahani, M (1984) On non-linear functions of linear combinations. *SIAM J. Sci. Statist. Comput.* **5**, 175–191.

Donoho, D. L. and Johnstone, I. M. (1989) Projection-based approximation and a duality with kernel methods. *Ann. Statist.* **17**, 58–106.

Duda, R. O. and Hart, P. E. (1973) *Pattern Classification and Scene Analysis.* New York: Wiley.

Fahlman, S. E. and Lebiere, C. (1990) The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2 (Denver, 1989)* (ed D. S. Touretzky) pp. 524–532. San Mateo, CA: Morgan Kaufmann.

Fayyad, U. M. and Irani, K. B. (1992) On the handling of continuous-valued attributes in decision tree generation. *Machine Learning* **8**, 87–102.

Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics* **7**, 179–188.

Fletcher, R. (1987) *Practical Methods of Optimization.* Chichester: Wiley.

Friedman, J. H. (1991) Multivariate adaptive regression splines (with discussion). *Annals of Statistics* **19**, 1–141.

Friedman, J. H. and Silverman, B. W. (1989) Flexible parsimonious smoothing and additive modelling (with discussion). *Technometrics* **31**, 3–39.

Friedman, J. H. and Stuetzle, W. (1981) Projection pursuit regression. *J. Amer. Statist. Soc.* **76**, 817–823.

Fukunaga, K. (1990) *Introduction to Statistical Pattern Recognition.* Second edition. London: Academic Press.

Funahashi, K. (1989) On the approximate realization of continuous mappings by neural networks. *Neural Networks* **2**, 183–192.

Gallant, S. L. (1993) *Neural Network Learning and Expert Systems.* Cambridge, MA: The MIT Press.

Gelfand, S. B. and Delp, E. J. (1991) On tree structured classifiers. In *Artificial Neural Networks and Statistical Pattern Recognition. Old and New Connections.* (eds I. K. Sethi and A. K. Jain) pp. 51–70. Amsterdam: North Holland.

Gelfand, S. B. and Mitter, S. K. (1991) Recursive stochastic algorithms for global optimization in $\mathbb{R}^d$. *SIAM J. Control Opt.* **29**, 999-1018.

Gelfand, S. B., Ravishankar, C. S. and Delp, E. J. (1991) An iterative growing and pruning algorithm for classification tree design. *IEEE Trans. PAMI* **13**, 163–174.

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

Gill, P. E., Murray, W. and Wright, M. H. (1981) *Practical Optimization.* London: Academic Press.

Girosi, F. and Poggio, T. (1990) Networks and the best approximation property. *Biological Cybernetics* **63**, 169–176.

Gorman, R. P., and Sejnowski, T. J. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks* **1**, 75–89.

Gull, S. F. (1989) Developments in maximum entropy data analysis. In *Maximum Entropy and Bayesian Methods, Cambridge, 1988* (ed J. Skilling) pp. 53–71. Dordrecht: Kluwer.

Guo, H. and Gelfand, S. B. (1992) Classification trees with neural network feature extraction. *IEEE Trans. Neural Networks* **3**, 923–933.

Hall, P. J. and Wand, M. P. (1988) On non-parametric discrimination using density differences. *Biometrika* **75**, 541–547.

Hand, D. J. (1982) *Kernel Discriminant Analysis.* Letchworth: Research Studies Press.

Härdle, W. (1990) *Applied Nonparametric Regression.* Cambridge: Cambridge University Press.

Härdle, W. (1991) *Smoothing Techniques with Implementation in S.* New York: Springer-Verlag.

Hastie, T. J. and Tibshirani, R. J. (1990) *Generalized Additive Models.* London: Chapman and Hall.

Hastie, T, Tibshirani, R. and Buja, A. (1992a) Flexible discriminant analysis. Submitted.

Hastie, T, Buja, A. and Tibshirani, R. (1992b) Penalized discriminant analysis. Submitted.

Hertz, J., Krogh, A. and Palmer, R. G. (1991) *Introduction to the Theory of Neural Computation.* Redwood City: Addison-Wesley.

Hinton, G. E. (1989) Connectionist learning procedures. *Artificial Intelligence* **40**, 185–234.

Hjort, N. L. (1986) *Notes on the Theory of Statistical Symbol Recognition.* Norwegian Computing Center Report 778.

Hornik, K., Stinchcombe, M. and White, H. (1989) Multilayer feedforward networks are universal approximators. *Neural Networks* **2**, 359–366. Reprinted in White (1992).

Hwang, J.-N., Lay, S.-R., Maechler, M., Martin, D. and Schimert, J. (1993) Regression modeling in back-propagation and projection pursuit learning *IEEE Trans. Neural Networks* to appear.

Hwang, J.-N., Li, D., Maechler, M., Martin, D. and Schimert, J. (1992a) A comparison of projection pursuit and neural network regression modeling. In *Advances in Neural Information Processing Systems 4* eds J. E. Moody, S. J. Hanson and R. P. Lippmann, San Mateo, CA: Morgan Kaufmann, pp. 1159–1166.

Hwang, J.-N., Li, D., Maechler, M., Martin, D. and Schimert, J. (1992b) Projection pursuit learning networks for regression. *Engineering Applications Artificial Intelligence* **5**, 193-204.

Jones, L. K. (1987) On a conjecture of Huber concerning the convergence of projection pursuit regression. *Ann. Statist.* **15**, 880–882.

Jones, L. K. (1990) Constructive approximations for neural networks by sigmoidal functions. *Proceedings of the IEEE* **78**, 1586–1589.

Jones, L. K. (1992) A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Ann. Statist.* **20**, 608–613.

Judd, J. S. (1990) *Neural Network Design and the Complexity of Learning.* Cambridge, MA: The MIT Press.

King, R. D., Muggleton, S., Lewis, R. A. and Sternberg, M. J. E. (1992) Drug design by machine learning: the use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydorfolate reductase. *Proc. Nat. Acad. Sci.* **89** 11322–11326.

Kohonen, T. (1990) The self-organizing map. *Proc. IEEE* **78**, 1464–1480. Reprinted in Lau (1992).

Kůrková, V. (1992) Kolmogorov's theorem and multilayer neural networks. *Neural Networks* **5**, 501–506.

Kushner, H. (1987) Asymptotic global behaviour for stochastic approximation and diffusions with slowly decreasing noise effects: global minimization via Monte Carlo. *SIAM J. Cont. Opt.* **47**, 169–185.

Lau, C. ed. (1992) *Neural Networks; Theoretical Foundations and Analysis.* New York: IEEE Press.

Lessafre, E. and Albert, A. (1989) Partial separation in logistic discrimination. *J. Roy. Statist. Soc. B* **51**, 109–116.

Loh, W. and Vanichsetakul, N. (1988) Tree-structured classification via generalized discriminant analysis. *J. Amer. Statist. Assoc.* **83**, 715–728.

Macintyre, A. and Sontag, E. D. (1993) Finiteness results for sigmoidal "neural" networks. In *Proc. 25th Annual Symp. Theory Computing, San Diego, 1993*.

MacKay, D. J. C. (1992) A practical Bayesian framework for backprop networks. *Neural Computation* **4**, 448–472.

McLachlan, G. J. (1992) *Discriminant Analysis and Statistical Pattern Recognition.* New York: Wiley.

Miller, W. T. III, Sutton, R. S. and Werbos, P. J. eds (1990) *Neural Networks for Control.* Cambridge, MA: The MIT Press.

Moody, J. E. (1992) The *effective* number of parameters: an analysis of generalization and regularization in non-linear learning systems. In *Advances in Neural Information Processing Systems 4* eds J. E. Moody, S. J. Hanson and R. P. Lippmann, San Mateo, CA: Morgan Kaufmann, pp. 847–854.

Moody, J. and Darken, C. (1989) Fast learning in networks of locally-tuned processing units. *Neural Computation* **1**, 281–294.

Murata, N., Yoshizawa, S. and Amari, S. (1991) A criterion for determining the number of parameters in an artifical neural networks model. In *Artificial Neural Networks* eds T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Amsterdam: North Holland, pp. 9–14.

Murphy, P. M., & Aha, D. W. (1992). *UCI Repository of Machine Learning Databases* [Machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science. Available by anonymous ftp from `ics.uci.edu` in directory `pub/machine-learning-databases`.

Musavi, M. T., Ahmed, W., Chan, K. H., Faris, K. B. and Hummels, D. M. (1992) On the training of radial basis function classifiers. *Neural Networks* **5**, 595–603.

Neal, R. (1992) Bayesian training of backpropagation methods by the hybrid Monte Carlo method. *Technical report CRG-TR-92-1* Dept of Computer Science, University of Toronto.

Neal, R. (1993) Bayesian learning via stochastic dynamics. In *Advances in Neural Information Processing Systems 5*, eds C. L. Giles, S. J. Hanson, and J. D. Cowan, San Mateo, CA: Morgan Kaufmann, pp. 475–482.

van Ooyen, A. and Nienhuis, B. (1992) Improving the convergence of the back-propagation algorithm. *Neural Networks* **5**, 465–471.

Park, J and Sandberg, I. W. (1991) Universal approximation using radial-basis-function networks. *Neural Computation* **3**, 246–257.

Poggio, T. and Girosi, F. (1990) Networks for approximation and learning. *Proc. IEEE* **78**, 1481–1497. Reprinted in Lau (1992).

Powell, M. J. D. (1987) Radial basis functions for multivariable interpolation: a review. In *Algorithms for Approximation* (eds J. C. Mason and M. G. Cox) pp. 143–167. Oxford: Clarendon Press.

Pregibon, D. (1982) Resistant fits for some commonly used logistic models with medical applications. *Biometrics* **38**, 485–498.

Quinlan, J. R. (1979) Discovering rules by induction from large classes of examples. In *Expert Systems in the Microelectronic Age*, (ed D. Michie). Edinburgh: Edinburgh University Press.

Quinlan, J. R. (1983) Learning efficient classification procedures and their application to chess end-games. In *Machine Learning* (eds R. S. Michalski, J. G. Carbonell and T. M. Mitchell) pp. 463–482. Palo Alto: Tioga.

Quinlan, J. R. (1986) Induction of decision trees. *Machine Learning* **1**, 81–106.

Quinlan, J. R. (1990) Decision trees and decision making. *IEEE Trans. SMC* **20**, 339–346.

Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Ripley, B. D. (1993) Statistical aspects of neural networks. In *Networks and Chaos – Statistical and Probabilistic Aspects* (eds O. E. Barndorff-Nielsen, J. L. Jensen and W. S. Kendall), pp. 40–123. London: Chapman & Hall.

Ripley, B. D. and Hjort, N. L. (1994) *Pattern Recognition and Neural Networks – A Statistical Approach.* Cambridge: Cambridge University Press.

Rissanen, J. (1987) Stochastic complexity (with discussion). *J. Roy. Statist. Soc. B* **49**, 223–239.

Roeder, K. (1990) Density estimation with confidence sets exemplified by superclusters and voids in galaxies. *J. Amer. Statist. Assoc.* **85**, 617–624.

Sethi, I. K. (1990) Entropy nets: from decision trees to neural networks. *Proc. IEEE* **78**, 1605–1613. Reprinted in Lau (1992).

Sethi, I. K. (1991) Decision tree performance enhancement using an artificial neural network implementation. In Sethi & Jain (1991) pp. 71–88.

Shanno, D. F. (1990) Recent advances in numerical techniques for large-scale optimization. In Miller *et al.* (1990) pp. 171–178.

Solla, S. A., Levin, E. and Fleisher, M. (1988) Accelerated learning in layered neural networks. *Complex Systems* **2**, 625–639.

Spackman, K. A. (1992) Maximum likelihood training of connectionist models: comparison with least-squares back propagation and logistic regression. *Proceedings of the Annual 15th Symposium on Computer Applications in Medical Care, Nov. 1991*, pp. 285–289. New York: IEEE Press.

Sprecht, D. F. (1990a) Probabilistic neural networks. *Neural Networks* **3**, 109–118.

Sprecht, D. F. (1990b) Probabilistic neural networks and the polynomial Adaline as complementary techniques for classification. *IEEE Trans. Neural Networks* **1**, 111–121.

Stone, C. J. (1982) Optimal global rates of convergence for nonparametric regression. *Ann. Statist.* **10**, 1040–1053.

Styblinski, M. A. and Tang, T.-S. (1990) Experiments in non-convex optimization: stochastic approximation and simulated annealing. *Neural Networks* **3**, 467–484.

Tråvén, H. G. C. (1991) A neural network approach to statistical pattern classification by “semiparametric” estimation of probability density functions. *IEEE Trans. Neural Networks* **2**, 366–377.

Valiant, L. G. (1984) A theory of the learnable. *Comm. Assoc. Comput. Mach.* **27**, 1134–1142.

Vapnik, V. N. (1982) *Estimation of Dependences based on Empirical Data.* Berlin: Springer-Verlag.

Wahba, G. (1990) *Spline Models for Observational Data.* Philadelphia: SIAM.

Wallace, C. S. and Freeman, P. R. (1987) Estimation and inference by compact encoding (with discussion). *J. Roy. Statist. Soc. B* **49**, 240–265.

Weigend, A. S. and Gershenfeld, N. A. eds (1993). *Predicting the Future and Understanding the Past: a Comparison of Approaches.* Redwood City: Addison-Wesley.

Weiss, S. M. and Kulikowski, C. A. (1991) *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems.* San Mateo, CA: Morgan Kaufmann.

White, H. (1989a) Learning in artificial neural networks: a statistical perspective. *Neural Computation* **1**, 425–464. Reprinted in White (1992).

White, H. (1989b) Some asymptotic results for learning in single hidden layer feedforward networks. *J. Amer. Statist. Assoc.* **84**, 1008-1013. Reprinted in White (1992).

White, H. (1992) *Artificial Neural Networks: Approximation and Learning Theory.* Oxford: Basil Blackwell.

Zhao, Y. and Atkeson, C. G. (1992) Some approximation properties of projection pursuit learning networks. In *Advances in Neural Information Processing Systems 4* eds J. E. Moody, S. J. Hanson and R. P. Lippmann, San Mateo, CA: Morgan Kaufmann, pp. 936–943.

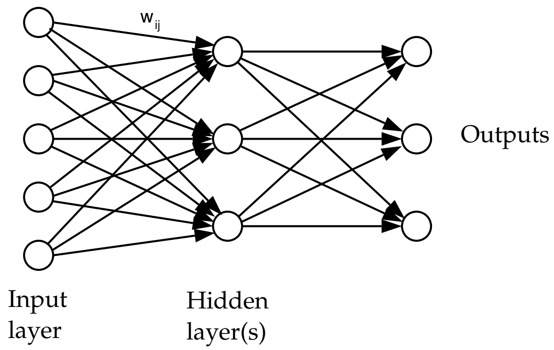

Figure 1: A generic feed-forward neural network. Normally all units in one layer are connected to all in the next layer, as shown.

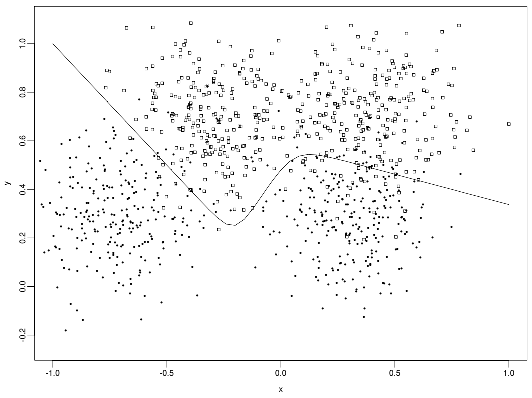

Figure 2: A two-dimensional synthetic dataset with the boundary of the Bayes rule. This is the test set of size 1000. The two classes are shown by filled discs and open squares.

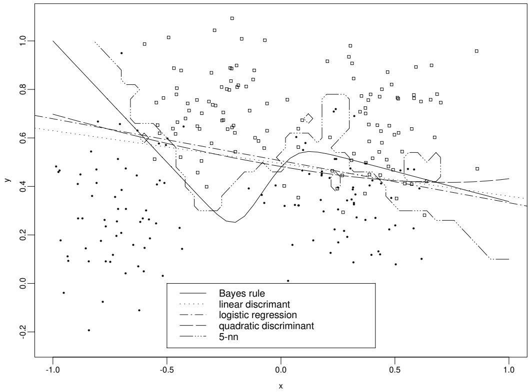

Figure 3: The training set with the Bayes rule, linear, logistic and quadratic discriminants and the 5-NN rule.

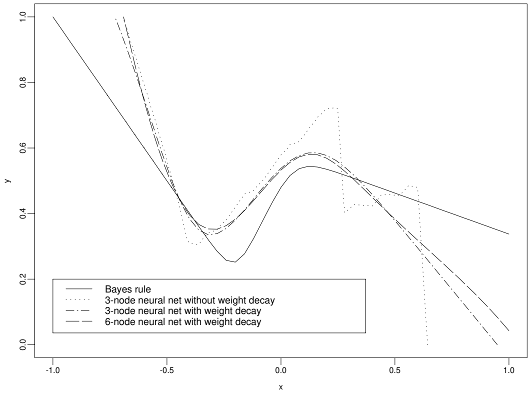

Figure 4: The boundaries for neural networks with three and six hidden units. Note how using weight decay smooths the fitted surface.

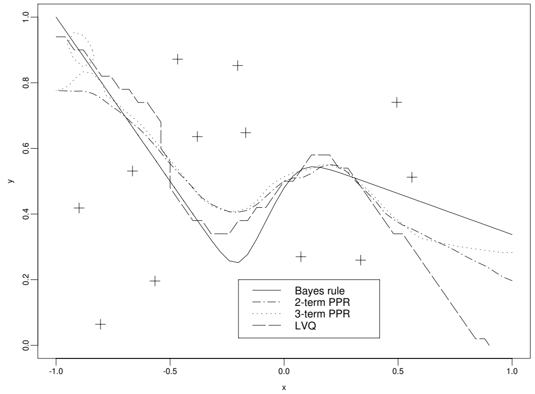

Figure 5: The boundaries for projection pursuit regression with two and three projection terms. Also shown are the 12 ‘codebook’ vectors chosen for learning vector quantization and its decision boundary.

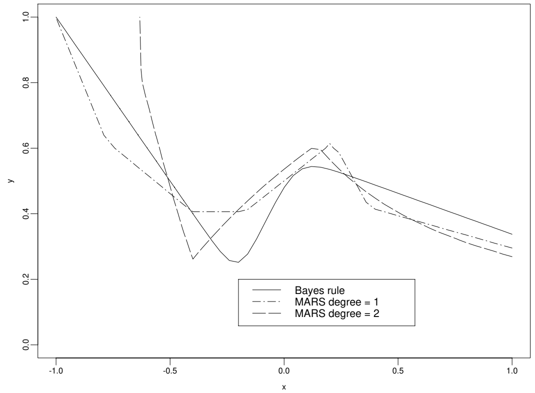

Figure 6: The boundaries for MARS with degree 1 and 2.

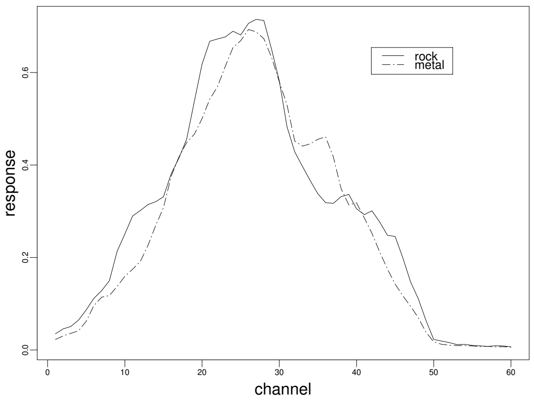

Figure 7: Mean response for the sonar data.

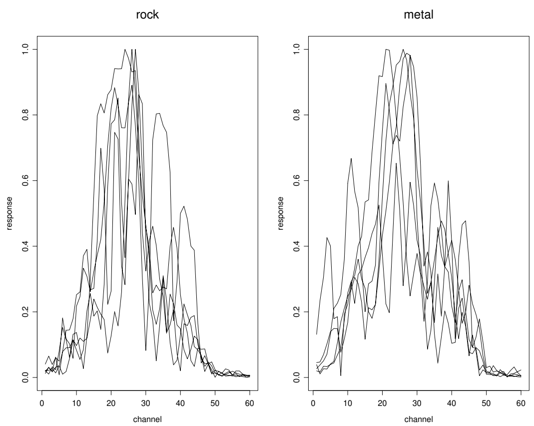

Figure 8: Five randomly chosen response curves for each class of the sonar data.

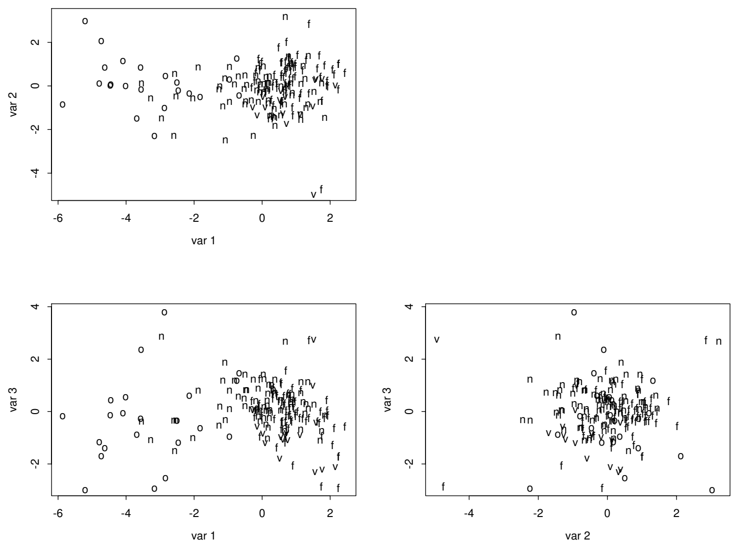

Figure 9: Discriminant plots for the first three linear discriminants for the forensic glass data. The coding is v for vehicle glass, f for window float glass, n window non-float glass and o other.

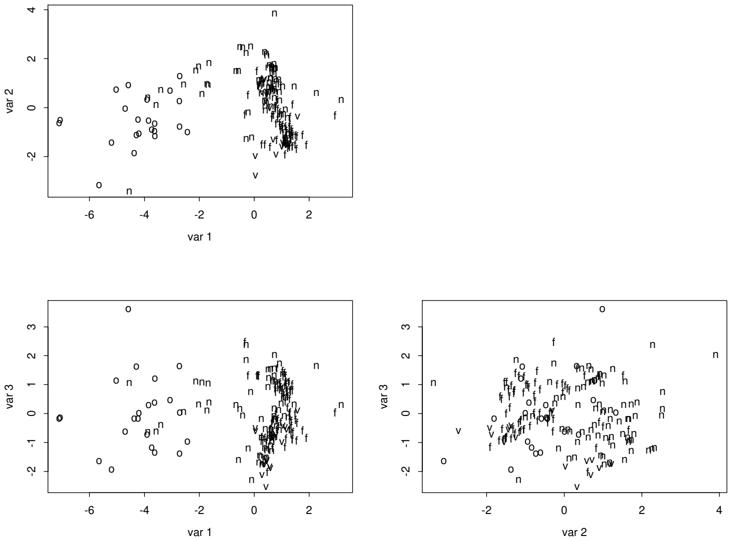

Figure 10: Generalized discriminant plots from MARS with degree=1. Note how the second variable more effectively separates the two types of window glass.

Data Availability

Copies of this paper in PostScript and of the main data files are available by anonymous ftp:

```
local% ftp markov.stats.ox.ac.uk    (192.76.20.1)
Connected to markov.stats.
220 markov.stats FTP server (SunOS 4.1) ready.
Name (markov.stats.ox.ac.uk:ripley): anonymous
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> cd pub/neural/papers
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> get RSS.ps.Z
200 PORT command successful.
150 Binary data connection for RSS.ps.Z (127043 bytes).
226 Binary Transfer complete.
local: RSS.ps.Z remote: RSS.ps.Z
ftp> get RSS.data.Z
200 PORT command successful.
150 Binary data connection for RSS.data.Z (73029 bytes).
226 Binary Transfer complete.
local: RSS.data.Z remote: RSS.data.Z
ftp> quit
221 Goodbye.
local% uncompress RSS.ps.Z RSS.data.Z
local% lpr -s RSS.ps
local% sh RSS.data
shar: Extracting "README" (8932 characters)
shar: Extracting "synth.data" (7500 characters)
shar: Extracting "synth.test" (32000 characters)
shar: Extracting "sonar.sets" (1529 characters)
shar: Extracting "sonar.data" (89227 characters)
shar: Extracting "sonar20.data" (33904 characters)
shar: Extracting "glass.train" (5101 characters)
shar: Extracting "glass.test" (5402 characters)
shar: End of shell archive.
```

The file `RSS.data` is a shar archive containing the training and test sets for the synthetic example, the sonar data and the glass data.

The synthetic data set has mixtures of normal distributions with centres $(-0.7, 0.3)$ and $(0.3, 0.3)$ in class 0 and $(-0.3, 0.7)$ and $(0.4, 0.7)$ in class 1, with variances 0.03.

A number of freely available neural network packages, including a version of my own C code, are available in directory `pub/neural`. My S package for neural networks is in directory `pub/S`.