Classification using splines

Smarajit Bose

Department of Statistics, Ohio State University, 141 Cockins Hall, 1958 Neil Ave., Columbus, OH 43210, USA

Received July 1994; revised December 1995

Abstract

In a J class classification problem with data of the form: (y_n, x_n), $n = 1,\ldots,N$, where $y_n \in \{1,\ldots,J\}$ and $x_n = (x_{1n},\ldots,x_{Mn})$, linear discriminant analysis produces estimated class boundaries which are linear in $x_1,\ldots,x_M$. In this paper, a method is developed which estimates the conditional class probabilities in a function space which is bigger than the linear function space. The decision rule based on those estimated conditional class probabilities can have very nonlinear class boundaries. The method projects the conditional class probabilities onto a space spanned by cubic splines, and, hence, is called classification using splines (CUS). This new method seems to achieve comparable and in some cases lower misclassification error rates than existing methods like CART or the back-propagation neural network classifier.

Keywords: Classification; Nonlinear class boundaries; Cubic splines; Cross-validation; Neural network; CART

1. Introduction

The classification problem in statistics is known as the problem of pattern recognition in the area of computer science. In the last few years, a method of pattern recognition, broadly called the neural network method, has regained interest among researchers, in both statistics and computer science. The method was developed in the process of modeling the human learning procedures. This method uses a network system, similar to some models of the human nervous system and, hence, was given the name the neural network method. Different architectures of the network have been developed along with interesting training schemes for estimating the related parameters (Lippmann, 1987; Lapedes and Farber, 1988; Barron and Barron, 1988; etc.). The most commonly used back-propagation classifier seems to achieve very

low misclassification error rates in many problems. For the remainder of this paper, the neural network method will refer to the back-propagation classifier.

Very recently, researchers have attempted to clarify the mathematical logic behind the success of the method, which is mostly based on ad hoc principles. The simple neural network method has undergone a number of modifications which are not yet mathematically justified. Justifying the success and finding the limitations of this method does not seem to be an easy task. To date there is not much literature along these lines. Efforts have been made to evaluate statistical aspects of the neural network method in Bose (1992); Ripley (1994) and Cheng and Titterington (1994).

This paper presents a statistical method which has some similarities with the neural network method, but can be justified theoretically and is computationally more feasible. It seems to achieve misclassification error rates comparable to those of the neural network method and other statistical methods such as CART (Breiman et al., 1984).

The organization of this paper is as follows. In Section 2, the problem is described, existing methods are discussed and the motivation for classification using splines (CUS) is presented. The theoretical formulations of CUS can be found in Section 3. Section 4 deals with computational aspects of the problem and presents the details of the CUS algorithm. The method is illustrated with the help of a few examples and its performance is compared with other existing methods in Section 5. Finally, Section 6 provides a summary and additional remarks. The mathematical framework behind the CUS algorithm has been given in the appendix.

2. The problem of classification

In the classification problem, measurements x_i, $i = 1,\ldots,M$, are taken on a single individual (or object), and the individuals are to be classified into one of J classes on the basis of these measurements. It is assumed that J is finite, and the measurements x_i are random observations from these classes.

A training sample is available which has data in the form $(\boldsymbol{x}_n, y_n)$, $n = 1,\ldots,N$, where y_n is the class label of the nth sample, $y_n \in \{1,\ldots,J\}$, $\boldsymbol{x}_n = (x_{1n},\ldots,x_{Mn})$ is the vector of measurements taking values in an M-dimensional space $\mathscr{X}$.

Based on the training sample we desire to find a decision function $d(\boldsymbol{x}_n) : \mathscr{X} \to \{1,\ldots,J\}$ for classifying the individuals. In other words, d will provide a partition, say $\mathscr{X}_1, \mathscr{X}_2, \ldots, \mathscr{X}_J$, of $\mathscr{X}$, where $\mathscr{X}_j$ corresponds to the jth class, $j = 1,\ldots,J$, and measurements belonging to $\mathscr{X}_j$ will be classified as coming from the jth class.

A misclassification occurs when a decision rule d assigns an individual (based on its measurement vector) to class i when it is actually coming from class $j \neq i$. Misclassification error rate,(MER) defined as the proportion of individuals wrongly classified, can be calculated by the formula

$$MER = \frac{1}{N}\sum_{n=1}^{N} 1\{d(\boldsymbol{x}_n) \neq y_n\}.$$

A good classifier d tries to achieve the lowest possible MER in a given problem.

2.1. The optimal Bayes rule

The optimum rule $d(\boldsymbol{x})$ which minimizes MER is the following:

$$d(\boldsymbol{x}) = j_0, \quad \text{where } j_0 = \operatorname{argmax}_j p(j \mid \boldsymbol{X}). \quad \text{for } j = 1,\ldots,J,$$

and $p(j \mid \boldsymbol{x})$ is the conditional class probability for the jth class given the measurement vector $\boldsymbol{x}$. In other words, an individual should be assigned to that class which has the maximum probability, given the vector of measurements $\boldsymbol{x}$. This rule is known as the Bayes rule (Anderson, 1984). In the case there is more than one class with maximum posterior probability $p(\cdot \mid \boldsymbol{x})$, the individual may be assigned to any of these classes.

If the distributions $p(\boldsymbol{x} \mid j)$ of the measurements $\boldsymbol{x}$ for each class, and the probabilities π_j for selecting class j, are known, the posterior probabilities $p(j \mid \boldsymbol{x})$ can be calculated using the Bayes formula:

$$p(j \mid \boldsymbol{x}) = \frac{\pi_j p(\boldsymbol{x} \mid j)}{\sum_k \pi_k p(\boldsymbol{x} \mid k)}, \quad j = 1,\ldots J.$$

Typically, however, π_j and $p(\boldsymbol{x} \mid j)$, $j = 1,\ldots,J$, are unknown. If one assumes some specific distributions for the $p(\boldsymbol{x} \mid j)$ and π_j, the classification rule will be based on these calculated posterior probabilities.

A detailed overview on traditional statistical approaches can be found in Duda and Hart (1973) or Fukunaga (1972). A recent discussion on these approaches is also available in James (1985). A discussion on the approaches based on artificial intelligence techniques and their relationship to other classification techniques can be found in Chandrasekaran and Goel (1988).

2.2. Discriminant analysis

In the conventional method of discriminant analysis for solving classification problems, it is assumed that the measurement vectors $\boldsymbol{x}$ in each class follow a multivariate normal distribution. The class probabilities π_j, $j = 1,\ldots,J$ are estimated by $\hat{\pi}_j = n_j/n$, n_j being the number of observations from the class j in the training sample and $n = \sum_j n_j$. If it is further assumed that the covariance matrices of the measurements in each class, $\Sigma_1, \Sigma_2, \ldots, \Sigma_J$ are the same, it turns out that the regions created by the Bayes rule are separated by boundaries which are linear in $x_1,\ldots,x_M$. This method of linear discriminant analysis will not be adequate when the true class boundaries are very nonlinear. For example, if two classes are separated by a circle, linear discriminant analysis will not produce good results.

When the conventional assumption that $\Sigma_1 = \Sigma_2 = \cdots = \Sigma_J$ is dropped, and covariance matrices for the different classes are estimated individually, the Bayes decision rule gives quadratic boundaries. This method of quadratic discriminant analysis is still not adequate for approximating highly nonlinear class boundaries. For instance, in Example 1 of Section 5, quadratic discriminant analysis achieves a misclassification error rate of 10.4% (an improvement compared to 13.6% error made

by linear discriminant analysis). However, CUS achieves an error rate close to the optimal misclassification error rate of 6.67% in this simulated experiment.

2.3. Existing nonlinear methods for classification

The inadequacy of linear or quadratic discriminant analysis for the purpose of classification in many examples made it necessary to look for approaches with the ability of approximating highly nonlinear class boundaries. One statistical technique, classification and regression trees, (CART: Breiman et al., 1984), seems to work well in many examples. The method produces a binary tree classifier which performs a set of tests on each individual, and assigns them to different classes.

However, CART also has limitations. When the true class boundaries cannot be approximated well by the union of a finite number of hyper-rectangles, the misclassification error rate will be high. For example, if two classes are separated by a circle, CART will not perform well.

Apart from the neural network method used in computer science, there exist many other nonlinear methods, for instance, a nonlinear discriminant analysis method based on ACE (alternating conditional expectations, Breiman and Friedman, 1985), proposed by Breiman and Ihaka (1984), or Flexible Discriminant Analysis by Hastie et al. (1994), which is based on nonparametric regression methods such as MARS (multivariate adaptive regression splines, Friedman, 1991) or BRUTO (Hastie, 1989).

2.4. CUS: classification using splines

Instead of assuming specific distributions for x and using them to calculate the conditional class probabilities $p(j \mid x)$, one can try to estimate $p(j \mid x)$, $j = 1,\ldots,J$, directly from the training sample and classify the sample cases according to the Bayes rule based on the estimates. If the estimates are close to the true conditional class probabilities, misclassification error rate should be close to the optimal misclassification error rate.

We have developed a method of classification using splines (CUS). With this method the true conditional class probabilities are projected into a class of smooth functions spanned by spline functions of the vector x. The method does not assume any distribution for the measurements in a class and can produce very nonlinear class boundaries. The method is based on nonlinear regression with cubic splines. Details of this method are described in Sections 3 and 4.

3. A new method for classification: CUS

3.1. Theoretical motivation of CUS

Consider random variables X and Y, where X takes values in an M-dimensional space $\mathcal{X}$ and Y takes values in the set $\{1,\ldots,J\}$. Assume the densities $f(x \mid Y = j)$, $j = 1,\ldots,J$, are continuous.

Let $L_2(X)$ be the class of all functions $\phi(X)$ such that $E\phi^2(X) < \infty$.

Define the functions ψ_j, $j = 1,\ldots,J$, as

$$\psi_j(Y) = \begin{cases} 1 & \text{if } Y = j, \\ 0 & \text{otherwise.} \end{cases}$$

Let

$$L = \sum_{j=1}^{J} E\{\psi_j(Y) - \phi_j(X)\}^2, \quad \text{where } \phi_j(X) \in L_2(X),\ j = 1,\ldots,J.$$

Then L is minimized by $\phi_j^*(X) = E\{\psi_j(Y) \mid X\}$, $j = 1,\ldots,J$.

Since $E\{\psi_j(Y) \mid X\} = p(j \mid X)$, the functions $\phi_j^*(X)$, $j = 1,\ldots,J$ are the conditional class probabilities, and these determine the optimal classification rule. From a training set, we can estimate the $p(j \mid X)$, $j = 1,\ldots,J$, by projecting the $\psi_j(Y)$ onto a suitable subspace of $L_2(X)$. Then the decision rule can be based on the estimated $\tilde{p}(j \mid X)$ rather than the $p(j \mid X)$. If the estimates are close to the original functions then the misclassification error rate of the decision rule thus obtained will be close to the optimal misclassification error rate.

With finite data, the problem of minimizing L based on the sample reduces to a regression of the indicator variables $\psi_1,\ldots,\psi_J$ onto a subspace generated by some functions of $X_1,\ldots,X_N$. The choice of subspace plays an important role in obtaining good estimates of $p(j \mid X)$, $j = 1,\ldots,J$.

The approach of estimating the posterior class probabilities was considered by Villalobos and Wahba (1983). They proposed a method based on maximum penalized log likelihood estimation using multivariate thin plate splines. The idea was attractive, however, they reported that the computation cost goes up rapidly for higher dimensions. A simpler alternative is discussed in the following section.

3.2. *The additive subspace*

The additive subspace $\mathscr{L}$ of $L_2(X)$ consists of $\phi(X)$ which are of the form

$$\phi(X) = \sum_{m=1}^{M} \phi_m(X_m),$$

where $E\phi_m^2(X_m) < \infty$, for $m = 1,\ldots,M$. The additive space is a very popular choice in nonparametric regression problem (Stone, 1985; Stone and Koo, 1986; Breiman, 1993; Hastie and Tibshirani, 1990), mainly because the contribution of each individual predictor variable can be explored separately. At the same time, it avoids the problem known as the curse of dimensionality. This problem arises in higher dimensions. As the dimension M of x increases, the sample size required to achieve the same degree of accuracy as in lower dimensions, increases at a very high rate. The method proposed by Villalobos and Wahba (1983) faced this problem.

If we choose the additive space for projecting each $p(j \mid X)$, the estimates $\hat{p}(j \mid X)$ are of the form

$$\hat{p}(j \mid X) = \sum_{m=1}^{M} \phi_{jm}(X_m), \quad j = 1, \ldots, J.$$

Note that we are interested in finding a good decision rule, not the $p(j \mid X)$ themselves. If the estimated functions are close to $p(j \mid X)$, the decision rule should be close to the optimal Bayes rule. There is an obvious drawback in the additive model: interactions between the predictor variables are not allowed and, hence, the estimation will not be satisfactory where $p(j \mid X)$ is highly nonadditive. Even then, the decision rule might be close to the optimal decision rule if $P(\text{argmax}_j\, p(j \mid X) \neq \text{argmax}_j\, \tilde{p}(j \mid X))$ is small.

It may be desirable that the estimated functions satisfy the basic rules of probability functions. The additivity condition can easily be enforced by including an intercept term in the regression, however, for computational ease, we do not enforce the positivity condition. This does not pose any problem for correctly identifying the maximum posterior probability as long as the estimated functions are close enough to the actual probabilities.

3.3. *Asymptotic behaviour of CUS*

There are other advantages of using the additive subspace. There are many known results in the field of nonparametric additive regression, which uses the additive subspace. The consistency and the rates of convergence of additive regression were derived in Stone (1985). We can extend those results to prove that the misclassification error rate of CUS converges in probability to the optimal misclassification error rate under similar conditions as required in additive regression. Details and proofs can be found in Bose (1992).

4. Computational aspect of CUS

The class of functions used in CUS is spanned by the cubic splines. This is one of the most popular classes used in additive regression. The problem we have described in the previous sections reduces to a number of additive regressions depending on the number of classes, with the same basis functions. Cubic splines are pieces of cubic polynomials joined together at the knots. The functions have continuous second derivatives, so they are sufficiently smooth (for a detailed discussion on splines see de Boor, 1978). We also put the restriction that the functions are linear in the tails to reduce the high variability near the end points following the suggestion made by Stone and Koo (1986) and Breiman (1993).

The guidelines for related issues, such as the number of initial knots, knot placements, etc., are given in Breiman (1993). We discuss these guidelines in more detail in the subsequent sections. The method is based on an idea suggested by Smith

(1982). The method starts with many knots and then reaches a smaller dimensionality by the backward deletion method. The dimensionality is selected by cross-validation.

After the estimates $\tilde{p}(j \mid \boldsymbol{x}) = \sum_{m=1}^{M} \phi_{jm}(x_m)$, $j = 1,\ldots,J$, are obtained using the selected model, each measurement vector, $\boldsymbol{x}_n$ is classified into the group j which maximizes $\tilde{p}(j \mid \boldsymbol{x}_n)$. This is also done on a test set (if available), which is not used for the training process. The misclassification error rates are calculated for the training set (resubstitution error) and the test set (test set error). These errors measure the performance of the method. The resubstitution error measures the goodness of fit for the training data, whereas the test set error indicates how the method will perform for classifying future data.

4.1. Selection of basis and parameters

For data with univariate x_n, $n = 1,\ldots,N$, K knots $t_1 < \cdots < t_K$ are placed on the x-axis in the range $[\min(x_n),\max(x_n)]$. The space $\mathcal{S}_0(\boldsymbol{t})$, $\boldsymbol{t} = (t_1,\ldots,t_K)$, of cubic splines is defined as follows: $\phi(x) \in \mathcal{S}_0(\boldsymbol{t})$ iff

(i) on each (t_k, t_{k+1}), $\phi(x)$ is a cubic polynomial,

(ii) $\phi(x)$ has continuous derivatives up to second order.

It is easy to see that $\mathcal{S}_0(\boldsymbol{t})$ is a $(K+4)$-dimensional space.

The two sets of functions most commonly used as bases for the class of cubic splines are the power basis and the B-spline basis. The power basis for univariate splines consists of functions $1, x, x^2, x^3, (x - t_k)_+^3$, $k = 1,\ldots,K$. The B-spline basis consists of functions $B_k(\boldsymbol{x})$ which are linear combinations of the functions in the power basis such that $B_k(\boldsymbol{x})$ has support $[t_k, t_{k-4}]$, $k = 1,\ldots,K$ (except near the end knots). We use the power basis in the CUS algorithm since addition or deletion of knots is computationally more convenient with this basis. To make the extrapolated spline fit outside the range $[\min(x_n),\max(x_n)]$ linear, Breiman (1993) suggested to take $t_1 = x_1$, where $(x_1,\ldots,x_N)$ are ordered, put an additional knot (t_{K+1}) at x_N, and impose the conditions

$$\phi''(x_1) = \phi'''(x_1-) = 0 \quad \text{for the left tail,} \tag{1}$$

$$\phi''(x_N) = \phi'''(x_N+) = 0 \quad \text{for the right tail.} \tag{2}$$

The restricted space $\mathcal{S}(\boldsymbol{t})$ is $K+1$ dimensional. A convenient power basis for this space can be constructed by using the functions $1, x, (x - t_k)_+^3$, $k = 1,\ldots,(K+1)$. Conditions (1) are automatically satisfied while conditions (2) are imposed during estimation.

For CUS with multivariate $\boldsymbol{x}$, we start with $K+1$ knots placed on each x-coordinate where $K+1$ is reasonably large. Then, we use the power basis for the additive subspace, which is the collection of the functions $1, x_m, (x_m - t_{km})_+^3$, $k = 1,\ldots,(K+1)$; $m = 1,\ldots,M$, and impose conditions (2). The $\psi_j(Y)$ are then regressed on the span of this basis. The entire procedure can be implemented using modified Gaussian sweep algorithm, which and some other details are provided in the appendix.

There are two important questions that need to be answered:

- How many knots are to be used?

- Where should the knots be placed?

We address these questions below.

We point out that there are many choices available for knot positions, basis functions, etc. Our approach is heavily influenced by Breiman (1993) which showed after an extensive simulation study that the method described below performed as well as the other alternatives that were considered. With his recommended choices, CUS achieved very reasonable misclassification errors. However, it would be interesting to see if the performance of CUS could be improved by using different basis functions or a different knot placement strategy.

4.2. Number and placement of initial knots

One has to be careful working with a large number of knots since $(X'X)$ may become singular. This problem can be tackled by carefully placing the knots. The knot placement algorithm we describe below has never encountered this problem in the numerous experiments we have performed.

Usually, the knots are placed on order statistics. Breiman reports that equispaced knots in the range of the X-variable works equally well, if not better, under most circumstances. However, this strategy performs poorly when the distribution of the X-variable has large gaps between some successive points. After running several simulation experiments, Breiman (1993) suggested an alternative method which we use for our purpose. This knot placement algorithm takes into account a result involving $\frac{1}{9}$th power of the underlying densities by Agrawal and Studden (1980). The range from min (x_{mn}) to max (x_{mn}) is divided into L equal bins $B_1,\ldots,B_L$ where $L = \text{int}(5N^{1/3})$. Let c_l be the number of data points in the B_l bin. Then the bins are combined to produce K intervals $I_1,\ldots,I_K$ such that N_k is nearly constant for $k = 1,\ldots,K$, where

$$N_k = \sum_{B_l \subseteq I_k} c_l^{1/9}.$$

N_k can be made nearly constant by minimizing $\text{Var}(N_k)$, using a dynamic programming algorithm. The $(K+1)$ end points of the intervals are chosen as the knots. If there are only a few clusters of $\{x_{mn}\}$ such that not enough bins will have $c_l > 0$, the algorithm reduces the number of knots. This seems to work well for estimation purposes (for a detailed discussion see Breiman, 1993).

Now comes the issue of deletion of knots. Two important questions are:

- Which knots should be deleted?
- When to stop deleting knots?

These questions are discussed in the next section.

4.3. The backward deletion procedure

The X matrix is modified for the regressions where the columns are formed by the functions in the power basis. Therefore, each knot corresponds to a column of the modified X matrix in the regression. The knot which leads to the smallest increment

in the misclassification error seems to be an obvious choice for deletion. However, the misclassification error criterion is too discrete in nature and, hence, is not an attractive choice. Instead, we concentrate our attention on the estimation procedure since it is directly related to the performance of CUS in terms of classification. The deletion criterion used is the same as in additive regression. The knot whose deletion leads to the smallest increment in the residual sum of squares is deleted. In this case, there are J regressions involving ψ_j, $j = 1,\ldots,J$, and the residual sum of squares we consider is the sum of the residual sum of squares from all J regressions.

It should also be noted that at each step we consider the same knot as a candidate for deletion in each regression, and look at the increment in the overall residual sum of squares, since we want to project all the $p(j \mid x)$ onto the same class of functions. At each step, we consider each of the knots one by one as possible candidates for deletion, calculate the increment in overall residual sum of squares, then delete the one for which the increment is the smallest.

During the deletion procedure, a linear basis function (x_m) may be deleted. However, to maintain the linearity of the left-hand tail, the linear term should not be considered as a candidate for deletion until all knots corresponding to that variable has been deleted. The end knot $t_{m,K+1}$ (at x_{mN} in the beginning) maintains the linearity at the right-hand tail. Therefore, one also has to be careful about deleting $t_{m,K+1}$ at each stage. It turns out that if $t_{m,K+1}$ is deleted, the linearity at the tails can be guaranteed by putting constraints on the coefficients of the remaining basis functions. The deletion procedure is described in more detail in the appendix where we present the algorithm.

The next question is when to stop. We use cross-validation to select the dimensionality. However, to reduce computation, the knot placement in the cross-validation is not changed. In v-fold cross-validation, we divide the data at random into v groups of approximately equal size. We then set aside the groups one at a time as test data, use the rest of the data for training, delete the knots one at a time, and note the misclassification error rate using the test data for each dimensionality. We pool the misclassification error rates from different test sets to get an overall misclassification error rate for each dimensionality. The dimensionality with the least misclassification error rate is selected. After the dimensionality is selected, we work with the complete data and continue backward deletion until we reach the chosen dimensionality. Then we calculate $\hat{p}(j \mid x)$ and define the decision rule for classification according to them.

It should be noted that starting with different numbers of initial knots, we might get different results. Since the knots will be placed at different locations, the cross-validated misclassification error rates for different dimensionalities and consequently, the selected dimensionality and variables in the final model can be different. Hence, it will be interesting to see how the results may vary for a given problem by applying CUS several times with different numbers of initial knots. The number of initial knots which produces the least training set misclassification error rate (or least cross-validated error rate which is also available) can be used to select the final model. However, in the examples reported in the next section the test set errors were not too sensitive on number of initial knots used. The range of the test set errors for

different numbers of initial knots in each of the example was less than 1%. More details are available in Bose (1992).

4.4. Iterating CUS

The estimates $\hat{p}(j \mid x), 1 \leq j \leq J$, obtained by CUS are the projections of the functions $p(j \mid x)$ into the subspace generated by additive splines. An interesting idea, originally suggested by Breiman, was to consider the additive subspace generated by (say) the first $J - 1$ of these estimated functions (because of the additivity restriction only $J - 1$ of them are independent). This new subspace will trivially contain the estimated functions. Therefore, the projections of $p(j \mid x)$ into the new subspace are expected to be at least as close as the $\hat{p}(j \mid x)$. The other attractive feature of this new subspace is that it will indirectly have some interactions between original predictor variables. This is the result of considering nonlinear smooth functions (additive cubic splines in CUS) of the estimated functions $\hat{p}(j \mid x), 1 \leq j \leq J$, which are of the form $\sum_{m=1}^{M} \phi_{jm}(x_m)$, $j = 1, \ldots, J$. Details of the motivation, computational aspects and similarity of this approach with the neural network method can be found in Bose (1992).

This approach computationally means replacing the original predictor variables by the estimated functions, and applying CUS on the revised dataset. It has been illustrated in Bose (1992) that if we decide to iterate until the estimated functions cease to change, we end up overfitting the training set. However, in most cases, a second iteration of CUS, yielded equivalent or lower test-set misclassification error rate. This prompted us to consider two iterations of CUS in our illustrations in the next section.

5. Experimental results

In this section, we use a few simple examples to illustrate the performance of CUS. For assessing the accuracy of CUS in these examples, we also report the results obtained by discriminant analyses, CART, and the neural network method. An extensive simulation experiment can be found in Bose (1992). We have selected three examples from that experiment. These examples collectively present a summary of the findings of the larger experiment. The first two examples are simulated, where we have used a training set of 500 cases and a test set of 3000 cases. Such a large test set was used to obtain reliable estimates of the prediction errors for each dataset. For these examples, we can theoretically calculate the true probabilities that we are trying to estimate. Using these theoretical probabilities the optimal rule or the optimal misclassification error rates can be derived for both training and test sets. A low optimal misclassification error rate means the classes are well separated whereas high optimal misclassification error rate indicates a lot of overlap among the classes. For the simulated datasets, this error is reported for training and test sets as part of the results. We also report the mean absolute errors for estimating the posterior class probabilities $p(j \mid x)$, $j = 1, \ldots, J$, for CUS and the neural network method.

The neural network method provides estimates of these probabilities at the output nodes.

The parameters for each method play an important role in this type of experiment. A thorough discussion is available in Bose (1992). The performance of CART did not seem to be heavily dependent on any particular parameter. For CUS, the only important parameter is the number of initial knots. The performance of the neural network method is heavily dependent on the number of hidden nodes. Therefore, we have experimented with different number of initial knots for CUS and different number of hidden nodes for the neural network method. The best results achieved by different methods are reported in this paper; the detailed results are available in Bose (1992).

For comparing the cpu times used by different methods, we may only restrict our attention to the training time. After the training phase, classification of future cases is almost instantaneous for these methods. With the development of faster computers, high cpu time may not be regarded as a disadvantage for a method, nevertheless if comparable accuracy can be achieved by another method using significantly less cpu time, the second method will be more attractive in terms of application. The training time that has been reported for each method below are cpu times on an IBM RS-6000.

Dataset 1. This is a two-dimensional problem with two classes. We have used mixture of 3 different bivariate normal distributions for each class distribution. The plot for this dataset is shown in Fig. 1. The details are as follows:

$$p(\boldsymbol{x}\,|\,1) = 1/3\sum_{i=1}^{3} N_2(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \qquad p(\boldsymbol{x}\,|\,2) = 1/3\sum_{i=4}^{6} N_2(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

$$\boldsymbol{\mu}_1 = (0,0), \qquad \boldsymbol{\mu}_2 = (-2,-3), \qquad \boldsymbol{\mu}_3 = (2,-1),$$

$$\boldsymbol{\mu}_4 = (3,-4), \qquad \boldsymbol{\mu}_5 = (1,-3), \qquad \boldsymbol{\mu}_6 = (4,-3),$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}, \qquad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix},$$

$$\boldsymbol{\Sigma}_4 = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}, \qquad \boldsymbol{\Sigma}_5 = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}, \qquad \boldsymbol{\Sigma}_6 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

The results in Table 1 shows the inadequacy of discriminant analysis for achieving nearly optimal misclassification error rate in the test set. The quadratic discriminant analysis works better than the linear version, but the other methods achieve nearly optimal error rates. The neural network method has a slight edge, but the other methods achieve comparable results with considerably less training time.

The second iteration of CUS, reduces the error in estimating $p(j\,|\,\boldsymbol{x})$ considerably; in fact, its estimation error is lower than the neural network method. However, that did not result in lower misclassification error rate which confirms our

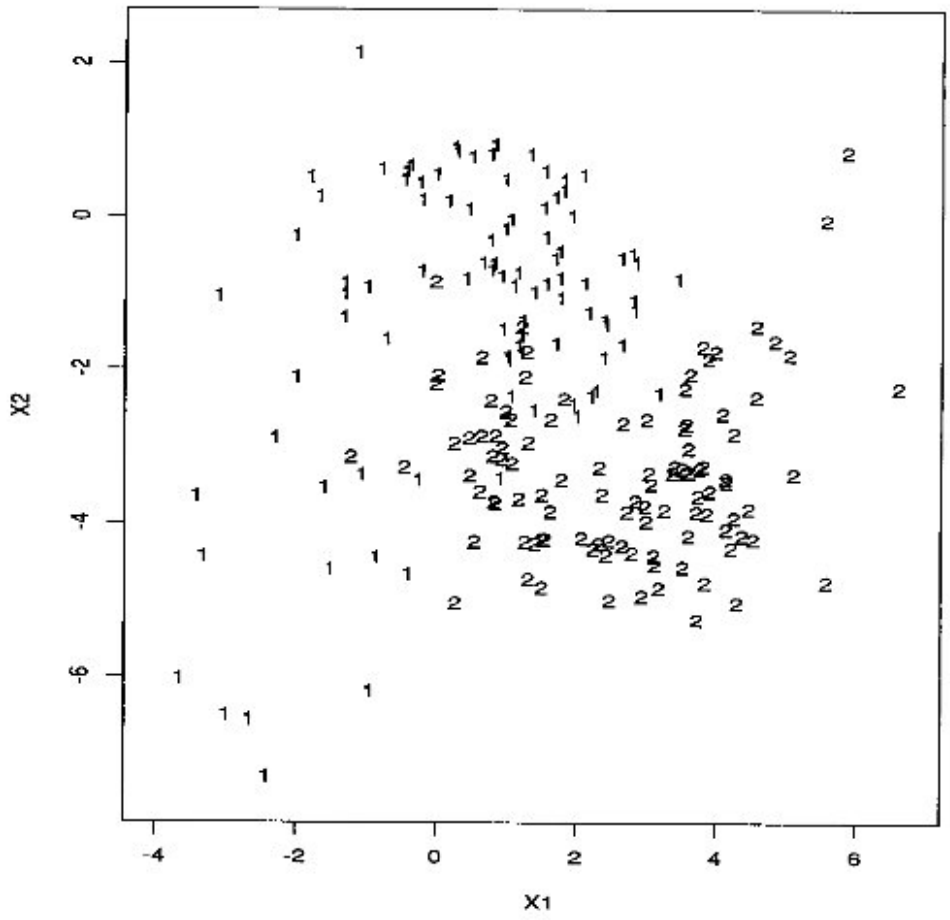

Fig. 1. The classes are represented by the corresponding numbers.

Table 1
Misclassification error rates for dataset 1

Method	Parameter or type	Misclassification error rate (%)		Training time	Error in estimating $p(j \mid x)$
		Training set	Test set		
Optimal		5.8	6.67		
Discriminant analysis	Linear	16.8	13.60	.18	
	Quadratic	11.8	10.40	.22	
CART		6.2	7.37	9.81	
Neural network	Number of nodes = 20	4.2	6.97	234.96	0.0917
CUS	Number of knots = 12	6.8	7.37	7.33	0.1492
CUS (second iteration)	Number of knots = 12	6.6	7.40	13.50	0.0651

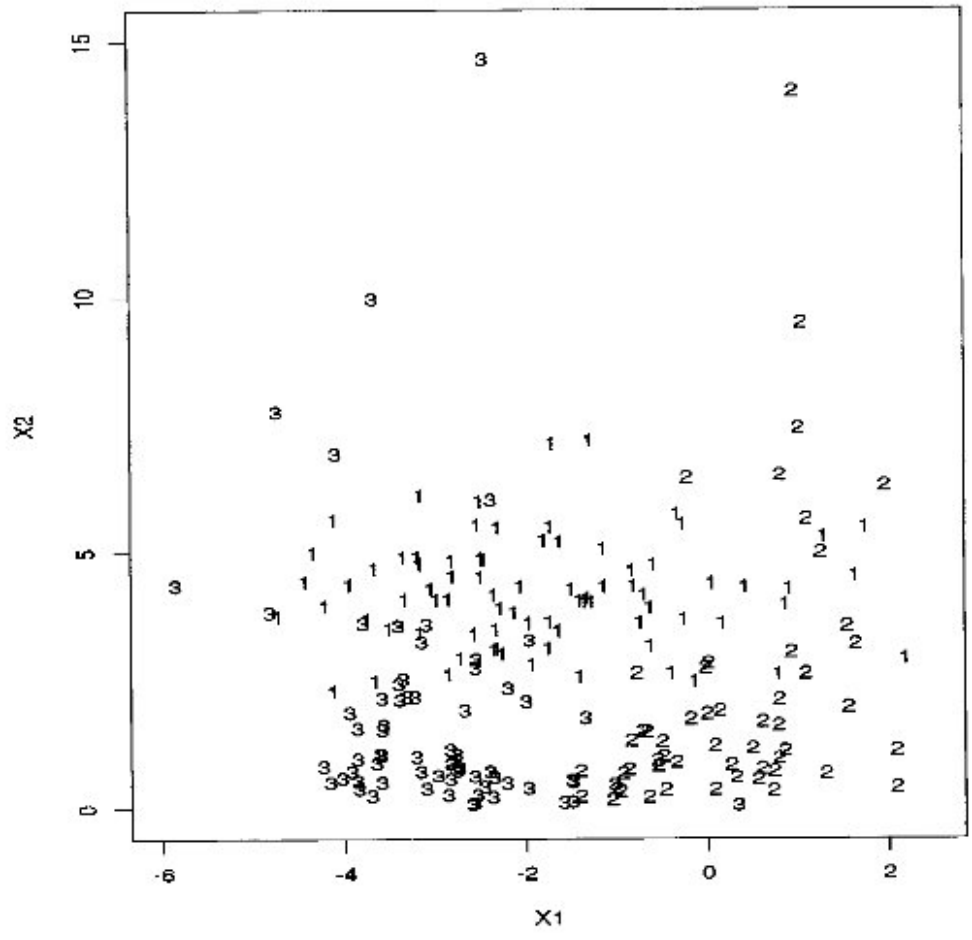

Fig. 2. The classes are represented by the corresponding numbers.

view that low misclassification error rates might be achievable in some problems even if the simple additive model cannot adequately approximate the true posterior probabilities.

Dataset 2. This is an example with three classes in six dimensions. The distribution for the measurement vector in the first class is multivariate normal. For the other two classes, $X_1, X_2', X_3, \ldots, X_6$ follow multivariate normal distribution, and $X_2 = \exp(X_2')$. The first two dimensions for each class has been shown in Fig. 2. The details are as follows:

$$p(\boldsymbol{x} \mid 1) = N_6(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \qquad \boldsymbol{\mu}_1 = (-1, 2, .5, .5, .25, .25),$$
$$p(\boldsymbol{x} \mid 2) = G(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2), \qquad \boldsymbol{\mu}_2 = \mathbf{0},$$
$$p(\boldsymbol{x} \mid 3) = G(\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3), \qquad \boldsymbol{\mu}_3 = (-2, 0, 1, 1, .5, .25).$$

$\boldsymbol{\Sigma}_1 = \boldsymbol{I}_6$, the six-dimensional identity matrix,

Table 2
Misclassification error rates for dataset 2

Method	Parameter or type	Misclassification error rate (%)		Training time	Error in estimating $p(j \mid x)$
		Training set	Test set		
Optimal		22.0	23.13		
Discriminant analysis	Linear	37.2	40.27	0.21	
	Quadratic	23.4	26.57	0.29	
CART		26.0	33.83	38.88	
Neural network	Number of nodes = 5	25.0	29.07	138.54	0.1578
CUS	Number of knots = 5	25.0	29.07	14.97	0.1501
CUS (second iteration)	Number of knots = 5	24.6	29.20	20.26	0.1248

$\Sigma_{2ij} = 0.5^{|i-j|}$, $1 \le i,j \le 6$, where Σ_{kij} is the (i,j)th element of Σ_k, $\Sigma_{3ij} = c_{ij}(0.5)^{|i-j|}$, $1 \le i,j \le 6$, where

$$c_{ij} = \begin{cases} 1 \text{ if } i = j, \\ -1 \text{ if } i \neq j \text{ and } i = 2 \text{ or } j = 2, \\ 0 \text{ otherwise.} \end{cases}$$

G is such that X_1, $\log X_2$, X_3, X_4, X_5 and X_6 have joint multivariate normal distribution.

Since the class boundaries are nearly elliptical in this problem, Table 2 confirms that the quadratic discriminant analysis can outperform the other methods in this example. Due to this simple structure, the second iteration of CUS does not achieve lower misclassification error rate even though its estimation error is still low compared to the neural network method and the first iteration of CUS. The misclassification error rates are not very close to the optimal rates, but the methods perform equally well except CART has a slightly higher error rate.

Dataset 3. *(Vowel recognition data)* This has been previously analyzed by Lee (1989). Prof. Richard Lippmann has kindly provided the dataset. In her Master's thesis, Lee tested several contemporary methods including the neural network method and CART on this dataset.

This dataset consists of 10 classes of 2-dimensional measurement vectors. This was created by Peterson and Barney (1952) by a spectographic analysis of vowels in words formed by "h", followed by a vowel and then followed by a "d". There were 67 people who spoke the words and the first two formant frequencies of 10 vowels were split into two sets, resulting in a training set consisting of 338 cases and a test set consisting of 333 cases. These formants are the two lowest resonant frequencies of a speaker's vocal tract. The scatter-plot of the dataset is given in Fig. 3.

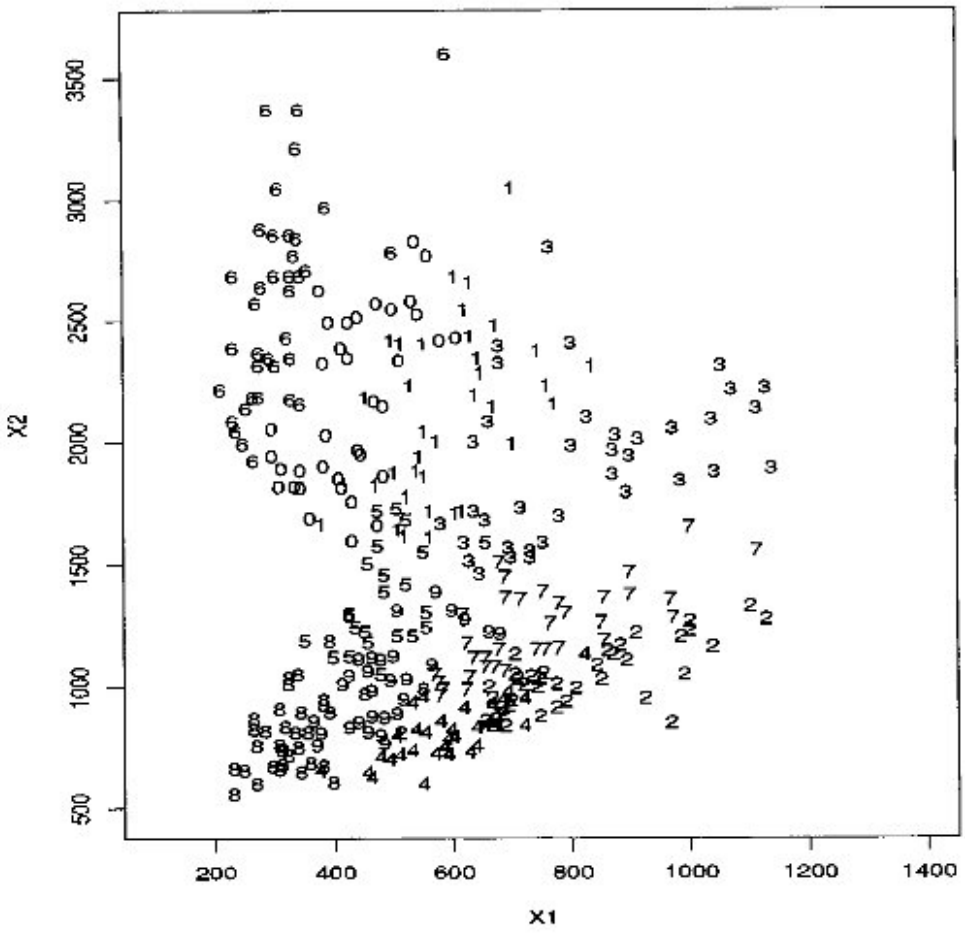

Fig. 3. Vowel Data; The Classes are represented by different numbers.

Lee (1989) reported error rates of 21% for both the training and the test sets when she used the neural network method with 50 hidden nodes. The result we achieved with the neural network method is even better and uses fewer nodes. This example highlights all the interesting points that we have discussed in this paper. Table 3 shows that quadratic discriminant analysis definitely improves the performance of the linear version. The neural network method can achieve even lower misclassification error rates, however, the training time is long. CART and CUS are quick alternatives which can achieve lower misclassification error rates if we compare them to the rates achieved by linear discriminant analysis. However, for examples with complicated boundaries like this the second iteration of CUS can improve the error rates. It performs equally well with considerably less training time if we compare its performance with the neural network method.

From these results and other simulation experiments reported in Bose (1992), it appears that it might be possible to achieve the power of the neural network method, at least for some problems, with much simpler mathematical models that are currently being used in Statistics.

Table 3
Misclassification error rates for dataset 3

Method	Parameter or type	Misclassification error rate (%)		Training time
		Training set	Test set	
Discriminant analysis	Linear	25.5	29.00	0.21
	Quadratic	21.3	19.82	0.24
CART		17.75	23.72	3.84
Neural network	Number of nodes = 20	16.27	18.62	483.97
CUS	Number of knots = 8	26.92	23.72	5.49
CUS (second iteration)	Number of knots = 6	16.86	18.92	57.33

5.1. *Estimated class boundaries for CUS*

The boundary estimated by CUS in Example 1 is shown in Fig. 4. The classifiers do not provide the boundaries themselves. The decision rule is based on the posterior class probabilities, which can be derived after training. Calculating the boundaries from the decision rule will require solving very nonlinear systems of equations. Instead, we have used the "interp" and "contour" functions of the statistical package "S" for the purpose of interpolation and plotting the boundaries, respectively. Hence, the boundaries in the plot are further approximations to the estimated class boundaries. At least we get an idea from the plot of how the estimated boundaries might look. We have only included the boundaries for CUS and did not compare estimated boundaries for different methods. Fig. 1 does suggest an optimal boundary similar to the estimated boundary in Fig. 4.

6. Discussion

Motivated by the success of the neural network method in many discrimination problems, we have developed a useful statistical technique CUS, which has a simple and interpretable structure but is capable of achieving competitive misclassification error rates in some examples that we have presented. CUS has another attractiveness that it requires very little training time particularly when compared against the neural network method. The computational gain is actually much more than it appears in Section 5 since CUS also performs model selection as a part of the training process. Finding the best architecture for the neural network method requires repeating the slow training scheme several times for different network models.

From our experience, we believe that CUS has the potential to become a very useful statistical tool in situations when the traditional discriminant analysis methods are not adequate (for example, when the class boundaries are not linear or quadratic). CART has been used as an alternative in such situations in the last decade. CUS

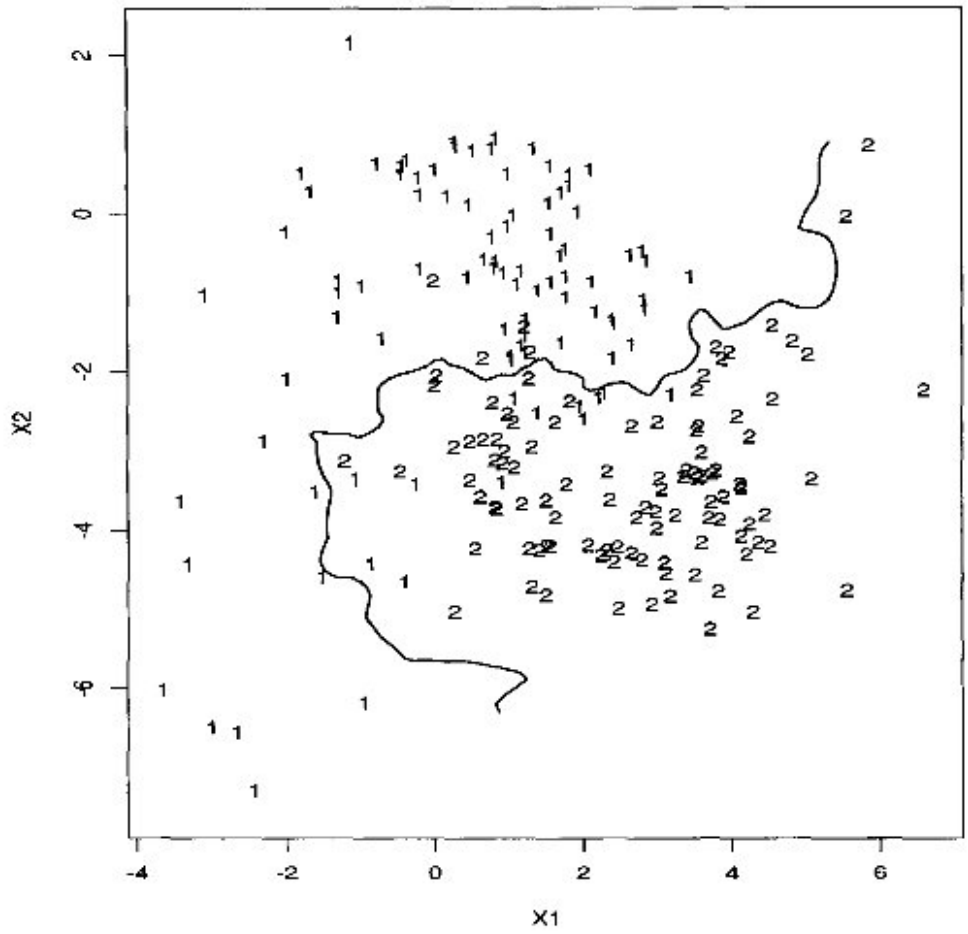

Fig. 4. The estimated boundaries for Dataset 1.

seems to be competitive with CART in the examples we have tried, and does not require considerably more training time. When the real class boundaries are very complex (possibly involve high order of interactions), CUS or CART can not achieve misclassification error rates comparable to the neural network method. A second iteration of CUS (as described in Section 4.4) seems to be helpful in such situations. For simpler problems when the boundaries are close to additive, the second iteration does not seem to improve the misclassification error rates. However, the effectiveness of CUS should be evaluated through additional simulations (and perhaps involving other existing classification methods) in a future study.

Appendix. The CUS algorithm

Given the preceding discussions, this method might seem to be computationally expensive, particularly since the dimensionality of the basis is selected by cross-

validation. However, the algorithm developed is remarkably fast. Most of the computations are Gaussian sweeps and the time it takes to estimate the $p(j|x)$, $j = 1,\ldots,J$ and to classify a large number of cases coming from several classes, number of measurements M being very high, is of the magnitude of a couple of minutes of CPU time in an IBM RS/6000.

A.1. Mathematical framework behind the algorithm

We reproduce the framework given by Breiman (1993). The restricted spline functions for any of the x-variable is of the form

$$h(x) = \sum_{k=1}^{K+1} \beta_k (x - t_k)_+^3 + \alpha + \nu x.$$

Let $x_{(N)}$ be $\max_n x_n$. The condition that $h(x)$ has to be linear to the right of $x_{(N)}$ (which requires the second and third derivatives to vanish beyond $x_{(N)}$) imposes the constraints:

$$\sum_{k=1}^{K+1} \beta_k = 0 \quad \text{and} \quad \sum_{k=1}^{K+1} \beta_k (x - t_k) = 0 \quad \text{for} \quad x \geq x_{(N)}.$$

The knot t_{k+1} at $x_{(N)}$ is an artificial knot to cancel out an x^3 term to the right of $x_{(N)}$. In particular, for $x = x_{(N)}$ we get the following constraints

$$\beta_{K+1} = -\sum_{1}^{K} \beta_k$$

and

$$\sum_{1}^{K} (x_{(N)} - t_k)\beta_k = 0. \tag{A.1}$$

If the knot t_{k+1} is deleted, the condition of linearity to the right of t_K can be imposed by adding another constraint

$$\sum_{1}^{K} \beta_k = 0. \tag{A.2}$$

During the deletion process, if the linearity of the fitted functions has to be maintained beyond the rightmost undeleted knots, we need to solve the least-square problems under one or two linear constraints. These constraints have to be satisfied for each of the x-variables.

Breiman (1993) described in detail how to solve least-squares problem with such constraints by using the Gaussian sweep operator. The algorithm he described has been modified for our purpose.

Let T denote the matrix where the (k,m)th element of T indicates the position of the kth knot on the mth x-variable; $k = 1,\ldots,K$, $m = 1,\ldots,M$.

We form the F matrix, where for $n = 1,\ldots,N$,

$$F_{n,k+(m-1)K} = (x_{mn} - t_{km})_+^3, \quad k = 1,\ldots,K;\ m = 1,\ldots,M,$$
$$F_{n,m+MK} = x_{mn}, \quad m = 1,\ldots,M,$$
$$F_{n,MK+M+1} \equiv 1.$$

We form the symmetric SS matrix of order $D \times D$, where $D = MK + M + 1$, and

$$SS_{ij} = \sum_n F_{ni}F_{nj}, \quad i,j = 1,\ldots,D.$$

This matrix will be augmented by M rows and columns for the (A.1) constraints, another M rows and columns for the (A.2) constraints and another J columns for J classes.

The new rows and columns will have all zero elements except for $m = 1,\ldots,M$, and $k = 1,\ldots,K$.

$$SS_{k+(m-1)K,D+m} = \max x_{mn} - t_{k,m},$$
$$SS_{k-(m-1)K,D+M-m} = 1$$

and finally for $j = 1,\ldots,J$, and $k = 1,\ldots,D$,

$$SS_{k,D+2M-j} = \sum_{n=1}^{N} F_{nk}\psi_j(y_n),$$

where $\psi_j(y_n)$ is the indicator function corresponding to the jth class. The SS matrix is kept symmetric by assigning the same values as above to the corresponding elements. Therefore, the SS matrix looks like

$$SS = \begin{bmatrix} S & Z & U & V \\ Z' & 0 & 0 & 0 \\ U' & 0 & 0 & 0 \\ V' & 0 & 0 & 0 \end{bmatrix},$$

where S is a $D \times D$ submatrix, Z corresponds to the constraints (A.1), U corresponds to the constraints (A.2) and V corresponds to the J classes.

It turns out that by sweeping the diagonal elements of SS matrix up to the $(D + M, D + M)$ element, we get the regression coefficients of the J regressions under constraints (A.1) in the last J columns.

During deletion, if a variable is deleted, an inverse sweep of the corresponding diagonal element will produce the regression coefficients of the new model at the last J columns of the SS matrix. Similarly, if an end knot is deleted, we sweep the corresponding diagonal element between $(D+M+1, D+M+1)$ and $(D+2M, D+2M)$ to impose constraints (A.2).

An inverse sweep on the diagonal element $MK + m$, $m = 1,\ldots,M$, which corresponds to the deletion of the linear term x_m is allowed only if all the knots on x_m have been already deleted. At any stage, the increment in overall RSS, which will

be caused by deletion of a variable, can be computed without an actual sweep by the formula :

$$-\sum_{j=1}^{J} SS^2_{k,D+2M+j}/SS_{k,k},$$

where k corresponds to the index associated with the variable.

Therefore, the sweep is performed only when a variable is actually deleted. A variable can be selected for deletion by minimizing the above formula over k.

The proofs can be found in Breiman (1993).

Acknowledgements

The author thanks his Ph.D. adviser, Prof. Leo Breiman for many helpful ideas and discussions.

References

Agrawal, G.G. and W.J. Studden, Asymptotic integrated mean square error using least square and minimizing splines, *The Ann. Statist.*, **8** (1980) 1307–1325.

Anderson, T.W., *An introduction to multivariate statistical analysis* (Wiley, New York, 1984).

Barron, A.R. and R.L. Barron, Statistical learning networks: a unifying view, in: *1988 Symp. on the Interface: Statistics and Computer Science* (1988) 192–203.

Bose, S., A method for estimating nonlinear class boundaries in the classification problem and comparison with other existing methods, Ph.D. Dissertation. (Dept. of Statistics, Univ. of California, Berkeley, CA, 1992).

Breiman, L., Fitting additive models to regression data: diagnostics and alternating views, *Comput. Statist. Data Anal.*, **15** (1993) 13–46.

Breiman, L. and J.H. Friedman, Estimating optimal transformations for multiple regression and correlation (with discussion), *J. Amer. Statist. Assoc.*, **80** (1985) 580–619.

Breiman, L., J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and regression trees* (Wadsworth & Brooks, Monterey, 1984).

Breiman, L. and R. Ihaka, *Nonlinear discriminant analysis via scaling and ace*, Technical Report 40, (Dept. of Statistics, University of California, Berkeley, CA, 1984).

Chandrashekharan, B. and A. Goel, From numbers to symbols to knowledge structures: artificial intelligence prospectives on the classification task, *IEEE Trans. Systems, Man Cybernet.*, **18** (1988) 415–424.

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

de Boor, C., *On uniform approximation by splines* (Springer, New York, 1978).

Duda, R. and P. Hart, *Pattern classification and scene analysis* (Wiley, New York, 1973).

Friedman, J.H., Multivariate adaptive regression splines (with discussion), *The Ann. Statist.*, **19** (1991) 1–141.

Fukunaga, K., *Introduction to statistical pattern recognition* (Academy Press, New York, 1972).

Hastie, T., Discussion of Flexible parsimonious smoothing and additive modeling by J. H. Friedman and B. W. Silverman, *Technometrics*, **31** (1989) 3–39.

Hastie, T. and R. Tibshirani, *Generalized additive models* (Chapman & Hall, New York, 1990).

Hastie, T., R. Tibshirani and A. Buja, Flexible discriminant Analysis, *J. Amer. Statist. Assoc.*, **89** (1994) 1255–1270.

James, M., *Classification algorithms* (Wiley, New York, 1985).

Lapades, A. and R. Ferber, How neural network works, in: D. Anderson (Ed.), *Neural information processing systems* (American Institute of Physics, New York, 1988) 442–456.

Lee, Y., *Adaptive modules in pattern recognition systems*, Master's Thesis (Dept. of Computer Science, MIT, Cambridge, MA, 1989).

Lippman, R.P., An introduction to computing with neural nets, *IEEE ASSP Magazine*, **4** (1987) 4–22.

Petersen, G.E. and H.L. Barney, Control methods used in a study of vowels, *The J. Acoust. Soc. Amer.*, **24** (1952) 175–185.

Ripley, B.D., Neural networks and related methods for classification (with discussion), *J. Roy. Statist. Soc.*, **56** (1994) 409–456.

Smith, P.L., Curve fitting and modeling with splines using statistical variable selection techniques, NASA Report 166034. (Langley Research Center, Hampla, VA, 1982).

Stone, C.J., Additive regression and other nonparametric models, *The Ann. Statist.*, **13** (1985) 689–705.

Stone, C.J. and C.-Y. Koo, Additive splines in statistics, in: *Proc. Statistical Computing Section* (American Statistical Association, Washington, DC, 1986) 45–48.

Villalobos, M.A. and G. Wahba, Multivariate thin plate spline estimates for the posterior probabilities in the classification problem, *Commun. Statist. Theory Methods*, **12** (1983) 1449–1479.