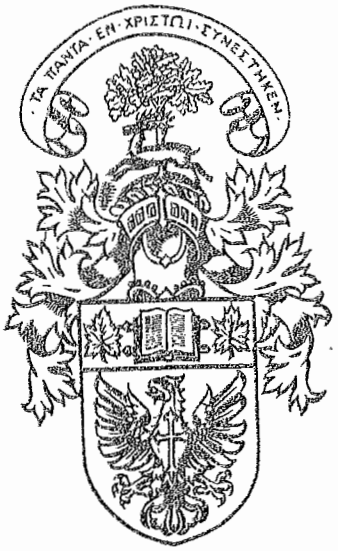


FACULTY OF BUSINESS

RESEARCH AND WORKING PAPER SERIES

THE APPLICATION OF MONOTONICITY CONSTRAINTS TO THE BACK PROPAGATION NEURAL NETWORK TRAINING ALGORITHM

By

Norman P. Archer

and

Shouhong Wang

Faculty of Business
McMaster University
Hamilton, Ontario
Canada L8S 4M4

WORKING PAPER NO. 343

THE APPLICATION OF MONOTONICITY CONSTRAINTS TO THE BACK PROPAGATION NEURAL NETWORK TRAINING ALGORITHM

By

Norman P. Archer

and

Shouhong Wang

Faculty of Business
McMaster University
Hamilton, Ontario
Canada L8S 4M4

WORKING PAPER NO. 343

THE APPLICATION OF MONOTONICITY CONSTRAINTS TO THE BACK PROPAGATION NEURAL NETWORK TRAINING ALGORITHM

by

Norman P. Archer

Shouhong Wang

Faculty of Business
McMaster University
Hamilton, Ontario
Canada L8S 4M4

THE APPLICATION OF MONOTONICITY CONSTRAINTS TO THE BACK PROPAGATION NEURAL NETWORK TRAINING ALGORITHM

Abstract

When statistical data are used in supervised training of a neural network employing the back propagation least mean square algorithm, the behavior of the classification boundary during training is often unpredictable. This research suggests the application of monotonicity constraints to the back propagation learning algorithm. When the training sample set is pre-processed by a linear classification function, this can improve neural network performance and efficiency in classification applications where the feature vector is related monotonically to the pattern vector. This technique can be applied to any classification problem which possesses monotonicity properties, such as managerial pattern recognition problems and others.

THE APPLICATION OF MONOTONICITY CONSTRAINTS TO THE BACK PROPAGATION NEURAL NETWORK TRAINING ALGORITHM

1. Introduction

A pattern recognition machine or classification decision function may be divided into two parts [1] : a feature extractor and a class selector. The feature extractor is a transformation $\mathbf{Y}=\phi(\mathbf{X})$ which transforms a pattern vector $\mathbf{X}$ $(x_1, x_2 \ldots x_m)$ describing an object into a feature vector $\mathbf{Y}$ $(y_1, y_2 \ldots y_n)$. Based on $\mathbf{Y}$, the class selector then selects a class $c_i \epsilon C$, where $C=\{c_1, c_2 \ldots c_k\}$ is a set of goal classes to which $\mathbf{X}$ may belong. Techniques used for managerial pattern recognition or classification can be classified according to the amount of knowledge available to generate the structure of the classification model [2]. Starting from the "glass box" end and working towards the "black box" end of the spectrum, these techniques include frame-based or rule-based expert systems [3], hierarchical classification [4], Bayes rule [5], discriminant analysis [6], and neural networks [7]. However, classification techniques other than the neural network technique are not applicable to many pattern recognition problems which are "black box problems" due to a lack of knowledge about distribution and classification data structures and, most important, potentially complex classification region boundaries.

In neural network classification, a layered network [7] is commonly used [8]. One of the most popular supervised learning algorithms is the back-propagation least mean square error learning

algorithm (BPLMS) [9] which integrates two fundamental learning algorithms: the least mean square error algorithm [10] and the back-propagation algorithm [7].

A neural network model almost always guarantees good separation in training data sets [9], but has mixed results for classification accuracy of test data sets [11][12]. It is commonly accepted that the behavior of the neural network training process depends heavily upon the training set itself [13]. For a given training sample the classification boundary generated by a neural network is relatively unpredictable, especially in the case of small sample size. This behavior is particularly critical when we are classifying statistical data. In this case, general learning which is only based on training sample data is usually not satisfactory since the neural network generates relatively arbitrary classification boundaries in solving these problems. Methods must therefore be developed that regulate classification boundary generation behavior, based on generic characteristics of certain classes of pattern recognition problems.

Neural network researchers are painfully aware of unpredictability in statistical data classification, and attempts have been made to improve available algorithms. One approach deals with the learning vector quantization model [14], based on the nearest-neighbor method. In this method the number of processing units in the input domain is predetermined. Each unit has a predetermined d-element reference vector, and each unit is associated with one of the classes of the input samples. The learning process is used to update the unit. Another model dealing with statistical classification data is the probabilistic neural network [15, 16]. The probabilistic

neural network is actually a parallel processing device for statistical functions. The hidden nodes (pattern units in the probabilistic neural network model) assume a form of kernel function with a pre-defined smoothing parameter. The learning process sets the weights vector directly and connects the neural network nodes properly. Both of the above models apply statistical concepts directly to neural networks. As discussed above, the major limitation of these two methods is that there are strong assumptions necessary about certain parameters, and the selection of those parameters influences the results for a particular problem. Therefore, the validation of these models is basically empirical, based upon benchmarking studies.

This paper reports on research which tries to exploit fully the adaptive nature of neural networks, and utilizes a type of generic problem domain knowledge, (i.e. monotonicity), to improve neural network learning ability in generating classification boundaries. In the models developed in the present research there are no assumptions about the statistical properties of the sample data set. Instead, more general knowledge about classification is consolidated in the neural network through the learning process. The argument is that the information (or prior knowledge) required in the present neural network model plays a rational role but is less restrictive than that required for other classification techniques.

In this research, a simple two layer, single output neural network with a sigmoid activation function and the BPLMS learning algorithm (see Figure 1), is investigated. This can be generalized to any two layer neural network with more than one output node, which is

a composite of neural networks of this type. Furthermore, arbitrary decision regions can be generated by neural networks with only one hidden layer [17][18].

*** Insert Figure 1 about here ***

2. Classification Boundary and Monotonicity Constraint

2.1. Classification Boundary

The neural network shown in Figure 1 is a typical two class classifier. Usually, the values of the x_i's are normalized to [0, 1] for computational convenience. The output value y of every sample point is also in [0, 1]. Hence, the **X**-y space is a functional cube. As an example, if **X** is two-dimensional, the functional cube can be depicted in three dimensions (see Figure 2). At step t the neural

*** Insert Figure 2 about here ***

network corresponds to the y-surface $y=\phi(\mathbf{X})$. Suppose that the cut-off y score for separating the two classes is 0.5. The boundary at step t is then the intersection line of the y-surface and the plane y=0.5.

For convenience, the following notation is used in this paper.

h : the number of hidden nodes;

$\mathbf{U}_i$: weight vector of the inputs into the ith hidden node,

$\mathbf{U}_i=(w_{i0}, w_{i1}, \ldots. w_{im})$, i=1...h,

where w_{ir} (r=0...m) is the weight at the connection from

the rth input node to the ith hidden node;

v_i : weight at the connection from the ith hidden node to the (single) output node;

With this notation, the function $y=\phi(\mathbf{X})$ represented by the neural network with a sigmoid activation function can be expressed as:

$$y = \{ 1 + \exp [-(v_0 + \sum_{i=1}^{h} v_i (1 + \exp (-\mathbf{U}_i\mathbf{X}'/T))^{-1})/T]\}^{-1}.$$

To simplify this discussion, temperature T will be set to 1, then

$$y = \{ 1 + \exp [-(v_0 + \sum_{i=1}^{h} v_i (1 + \exp (-\mathbf{U}_i\mathbf{X}'))^{-1})]\}^{-1}.$$

Problems of neural network unpredictability in classification originate from at least two sources:

(1) Given a certain training data set, the classification results generated by the individual neural network may differ unpredictably in many ways. Classification using the standard neural network with the BPLMS learning algorithm is based on the assumption that each input/output pair of the training sample data truly and accurately reflects the relationship between the pattern vector and the corresponding class. This condition is open to question in many practical applications. First, most real world sample data may contain measurement errors in their pattern vector values; namely, what we observe from a sample point is $\mathbf{Y}=\phi(\hat{\mathbf{X}})$ rather than $\mathbf{Y}=\phi(\mathbf{X})$, where $\hat{\mathbf{X}}$ is the measured $\mathbf{X}$.

(2) According to artificial intelligence theory [19], the dimensionality of a pattern vector is never complete for an object in the real world. This argument is expecially true for managerial problems since a formal description of the pattern vector basically provides only partial information about a person, a company, or a business event. From this standpoint, what we can observe from a sample point is $\hat{\mathbf{Y}}=\phi(\mathbf{X})$ rather than $\mathbf{Y}=\phi(\mathbf{X})$, where $\hat{\mathbf{Y}}$ is the real outcome of an event, with some causal factors (unobserved or "latent" factors) not included in **X**, even if **X** is accurately measured. Under these circumstances it is no surprise that, in some cases, two training sample points with the same (accurately measured) **X** have totally opposite classifications. Obviously, these kinds of facts pose difficulties during the training process. Therefore, the practical application of neural networks to pattern recognition with statistical data should be more concerned with appropriate treatment of statistical fluctuations than with simply pursuing 100% correct classification of the training sample set.

2.2. Monotonicity Constraints

This research was initially directed to the solution of managerial pattern recognition problems, which usually involve two classes. There is a large variety of managerial classification problems, e.g. prediction of corporate bankruptcy [20], credit extension decisions [21] [22] [23], and assessment of strategic planning [24]. A wide range of managerial decision problems involving pattern classification can be considered from the point of view of

utility or preference analysis [25]. For instance, determining the classification of observations into two classes is equivalent to deciding "which is preferred". One characteristic of a utility function is that the assumption of monotonicity is usually valid [14]. This can be defined as

$$[\ x_1 > x_2\ ] \Longleftrightarrow [\ \phi(\mathbf{X}|x_j=x_1) > \phi(\mathbf{X}|x_j=x_2)\ ] \quad .$$

The above expresses a monotonically increasing function. For our purposes, monotonic decrease is conceptually the same as monotonic increase, but with the inequality on the left hand side reversed. In terms of a two class classification problem if, other things being equal, the larger (or the smaller) value a variable x_j has, the larger the classification score y becomes, then y is monotonic in x_j . Given a managerial classification problem, prior knowledge about monotonicity is usually available. For example, in the creditworthiness classification case, income and assets are monotonic in the creditworthiness score; namely, other things being equal, the more the better.

The monotonicity property is not limited to managerial problems. For example, in an oil spill identification problem [26], during chemical testing of a spill sample, a spectrum is run on the sample, and the distance between its particular spectral line and that of the suspect sample is calculated as a pattern variable of the sample. Typically, this distance increases monotonically as the suspect and spill spectra become dissimilar.

If a monotonicity constraint is imposed on neural network learning, then y will vary monotonically with the corresponding

pattern variable. This will in turn result in an improvement in the predictability of boundary generation; that is, the shape of the boundary will always be consistent with the monotonicity property. However, given a training sample of statistical data, completely separating the training set into two sets may require a very complex y-surface to generate the boundary. This may violate monotonicity and not necessarily meet the objective of having good prediction capabilities, since such a boundary would be strongly affected by statistical fluctuations. In order to reduce the impact of such statistical fluctuations, a filter is needed to pre-process the statistical data. A linear classification function is a good choice, based on two related considerations: First, a linear classification function generates a good approximation of a "true" classification in dealing with statistical data [27]. Secondly, the linear classification function $y=\sum_{j=0}^{m} b_j x_j$ is monotonic in x_j for all j, since $\partial y/\partial x_j$=constant for all j. In the case of a pattern recognition problem which is monotonic, a linear classification function is superior to other distribution estimation techniques which may violate monotonicity and result in decision making conflicts.

2.3. A Proposed Heuristic for Improving The BPLMS Neural Network Training Algorithm

Assume that the BPLMS neural network training algorithm is subjected to a monotonicity constraint, and is trained with a sample that was pre-processed with classification scores obtained from a linear classification function. The classification generated boundary

would then be close to linear, after training were completed. However, our goal is not to simply map the linear classifier's result to a neural network result, but to improve on it. If we lack knowledge about the underlying distribution functions, the criterion which can be used to evaluate classification performance is a low misclassification rate for the training sample.

In practice, if the linear boundary has a large misclassification rate, and a cluster of misclassified points is observed, a heuristic can be used to modify the training sample set. This includes identification and verification of clusters, and making trade-offs between the potential gain from corrected misclassifications and the potential risk of introducing new misclassifications due to the modifications. In this case, vector analysis [28] is helpful, especially in the case of high dimensionality pattern vectors. A suggested vector analysis approach of finding the clusters is as follows.

(1) Pre-define a small number ξ (e.g. 0.05). If a misclassified point s has $|\zeta_s - 0.5| > \xi$, where ζ_s is the linear classification function value of s, then it is ignored in further training since it is considered to be an "outlier" in the sense that it is a random observation well beyond the likely region of the true boundary.

(2) Define set $(S_{mis})_{c_r}$ (r=1,2) such that $s \epsilon (S_{mis})_{c_r}$ if s is a misclassified point from class c_r and $|\zeta_s - 0.5| < \xi$. In the example of Figure 3, $(S_{mis})_{c_1}$=(A, B, C).

(3) Let $(\omega_j)_s$ be the angle between the vector direction of s and axis x_j (j=1...m-1), and then

$\cos(\omega_j)_s = (x_j)_s / ||X_s||$. For each $s\epsilon(S_{mis})_{c_r}$ (r=1,2), there is an image $(\omega_j)_s$ on each ω_j (j=1...m-1) axis.

(4) Screen the ω_j axes from 0 to $\pi/2$. Those misclassified points which belong to the same class and are contiguous will comprise clusters $(S^g_{mis})_{c_r}$, where g is the cluster number. In the example of Figure 3, where m=2, {A, B, C}, {F}, and {D, E} are three such clusters.

The trade-off between the potential gain from the corrected misclassifications and the potential risk of introducing new misclassifications can be made by vector analysis, as follows.

(1) Define set $(S^g_{cor})_{c_r}$ such that $s\epsilon(S^g_{cor})_{c_r}$ if s is in class c_r, which is correctly classified by the linear classification function, $|\zeta_s - 0.5| < \xi$, and $(\omega_j)_s$ falls into the range $[\ \min(\omega_j)_{(S^g_{mis})_{c_{\sim r}}}\ ,\ \max(\omega_j)_{(S^g_{mis})_{c_{\sim r}}}\ ]$, where $\min(\omega_j)_{(S^g_{mis})_{c_{\sim r}}}$ and $\max(\omega_j)_{(S^g_{mis})_{c_{\sim r}}}$ are the extreme values of the ω_j's of the misclassified points in the cluster $(S^g_{mis})_{c_{\sim r}}$, and where $c_{\sim r}$ is the class of points not included in c_r.

In the example of Figure 3, $(S^1_{cor})_{c_2}$={G}.

(2) If the total number of points in $(S_{mis})_{c_{\sim r}}$ is significantly greater than the total number of points in $(S_{cor})_{c_r}$ (r=1,2), this means that the modification of the misclassified points in $(S_{mis})_{c_r}$ would introduce fewer new misclassifications if learning is completed on the modified training sample set.

(3) Ignore $s\epsilon(S_{cor})_{c_r}$ (r=1,2) ({G} in the example) in further training. Train the neural network with the new training data set. If training is now completed, the misclassification rate will be no greater than before.

Based on a properly modified training sample set, the neural network will readily generate a boundary with regulated shape. This kind of adaptive problem is difficult if not impossible to solve by other classification methods.

*** Insert Figure 3 about here ***

3. Monotonic Function Model

Two issues related to the problems of standard neural network learning have been discussed thus far. First, the standard BPLMS learning algorithm makes a neural network's boundary behavior unpredictable. In order to ensure a classification result which is more consistent with true boundaries, it has been suggested that monotonicity should be imposed during the learning process. Second, if monotonicity is imposed during learning, the original training sample set may not be suitable, because the training is often in

conflict with monotonicity when classifying statistical data. Hence, pre-processing of the training sample based on a linear classification boundary is suggested. This section will summarize a learning model, called the Monotonic Function Model, for training neural networks using a modified BPLMS algorithm in classifying statistical data under monotonicity conditions.

3.1. Monotonic Condition for Neural Networks

If $y=\phi(\mathbf{X})$ is monotonic in x_j , then $\partial y/\partial x_j \geq 0$. Using the notation in Figure 1, and a sigmoid activation function, this condition is generalized as follows.

$$\partial y/\partial x_j = \partial\{[1+\exp(-(v_0+\sum_{i=1}^{h} v_i(1+\exp(-\mathbf{U}_i\mathbf{X}'))^{-1}))]^{-1}\}/\partial x_j$$

$$= y(1-y) \sum_{i=1}^{h} \partial [v_i(1+\exp(-\mathbf{U}_i\mathbf{X}'))^{-1}] / \partial x_j$$

$$= y(1-y) \sum_{i=1}^{h} w_{ij} v_i \exp(-\mathbf{U}_i\mathbf{X}')[1+\exp(-\mathbf{U}_i\mathbf{X}')]^{-2}.$$

Thus,

$$\partial y/\partial x_j \geq 0 \iff \sum_{i=1}^{h} w_{ij} v_i \exp(-\mathbf{U}_i\mathbf{X}')[1+\exp(-\mathbf{U}_i\mathbf{X}')]^{-2} \geq 0 \quad \text{for all } j,$$

provided y is monotonic in x_j .

The following algorithm meets these general necessary and sufficient conditions. A derivation of this algorithm is given in the Appendix.

For all j provided y is monotonic in x_j do:

(1) For i=1...h, define set I so that $i \in I$ if $w_{ij}v_i < 0$;

(2) Compute

$$Q_1 = 0.25 \sum_i w_{ij} v_i \quad \text{for those } i \epsilon I,$$

where 0.25 is the maximum of

$$\exp(-\mathbf{U}_i\mathbf{X}')[1+\exp(-\mathbf{U}_i\mathbf{X}')]^{-2};$$

(3) Compute

$$Q_2 = \sum_i w_{ij} v_i A_i \quad \text{for those } i \notin I,$$

where $A_i = \min([\exp(-\sum_p w_{ip})(1+\exp(-\sum_p w_{ip}))^{-2}],$

$$[\exp(-\sum_q w_{iq})(1+\exp(-\sum_q w_{iq}))^{-2}]\},$$

w_{ip}'s and w_{iq}'s are elements of $\mathbf{U}_i$,

and all $w_{ip}<0$ and all $w_{iq}\geq 0$;

(4) If $Q_1 + Q_2 \geq 0$, then y is monotonic,

else monotonicity is violated;

End-do.

During training for a sample point, if the monotonic condition would be violated due to a large change of weights, the learning rate is decreased for that sample point. Because our model does not change the principle of error hill climbing, the adaptive nature of the algorithm does not have to be changed. Note that the initialized neural network with its weights of uniform (0,1) random numbers has the monotonicity property. Experiments with this algorithm have shown that, when the proper training sample set (see sections 2.3. and 3.2.) is used, the monotonic condition is rarely violated.

3.2. Training Sample Set

In section 2.2. a heuristic training scheme based on the pre-process result of a linear classification function was suggested. Here, the procedure to find a proper training sample set is briefly summarized.

(1) Pre-process the initial sample data set with a linear classification function. Normalize the classification score on [0,1] such that a point on the linear boundary has score 0.5. Pre-define a small number ξ.

(2) Based on the pre-process result and ξ, and using vector analysis (Section 2.3), try to find a new training sample set such that correctly classifying it will result in a lower misclassification rate.

(3) If the new training set is found, then goto step (5); otherwise, reduce ξ.

(4) If $\xi \approx 0$, then STOP; otherwise, goto step (2).

(5) Train the neural network on the new training sample set, while maintaining monotonicity. If successful, then a new classification result is obtained, which has an overall misclassification rate which is no greater than the linear classifier on the initial sample set;
otherwise reduce ξ, and goto step (2).

3.3. Example Applications of the Monotonic Function Neural Network Model

3.3.1. An Example of Simulated Data

In this subsection an experiment with simulated data is described which applies neural network classification with monotonic conditions and a pre-processed training sample set. For this experiment, the pattern vector dimension was two, so results could be displayed on a plot. The size of the data set was 40 (20 in each class). An initial sample data set was generated by simulation. The true classification boundary was a cosine shape (see Figure 4). The linear discriminant analysis (LDA) technique was employed to pre-process the sample data. The LDA classification accuracy for the 40 sample points was 33/40 (82.5%). After transformation of the LDA result to the classification scores, ξ was determined by the training sample algorithm (Section 3.2.) as 0.04. Four misclassified points were within this tolerance. However, one of the previously correctly classified points was now incorrectly classified, resulting in a net gain of three properly classified points. Thus a new training sample set with 36 sample points was used to train the neural network, with 5 hidden nodes, under the monotonic condition. After 6400 learning sweeps, the final classification boundary was obtained, which is closer to the true boundary than the LDA result in most regions of the pattern space, resulting in an overall accuracy of 90% (Figure 4). This compares with standard BPLMS neural network training on the original data set, where no convergence was observed after 65000 learning sweeps.

*** Insert Figure 4 about here ***

3.3.2. An Example of Real Data

In the last subsection , simulated data were used for testing the MF model. The advantage of simulated data is that the optimal boundary is known and can be used to evaluate the performance of the model, as shown. However, the above method is not applicable for real problems. A widely acknowledged testing method is cross-validation [17], a form of the jackknife method [29] which is commonly used in statistics. In this method, a training data set is randomly selected from the available sample. After training the classifier, the remainder of the observations are used to test the classifier. Usually, the sample is equally divided into two subsets, and each of them serves as training data set and testing data set in turn in two trials.

The data used in this experiment came from the Alpha TV Commercial data bank published by Green [30]. The author used a subset of the data bank to conduct a linear discriminant analysis of two classes described as follows:

Class 1: 78 respondents who selected the Alpha brand of radial tires;

Class 2: 174 respondents who did not select the Alpha brand;

and the pattern variables:

1. whether Alpha was the brand selected in the respondent's last purchase of replacement tires;
2. pre-exposure interest in Alpha radial tires;
3. post-exposure believability of the Alpha commercial;
4. post-exposure interest in Alpha radial tires.

Using the 252 data points, linear discriminant analysis gives the result that (52+131)/252 or 72.6 percent of the sample is correctly classified, although the difference in the locations of the two class centroids was highly significant (F=17.98, with 4 and 247 degrees of freedom [30, p.179]). Hence, as pointed out by the author, the separation effected by the linear discriminant function is not good from a practical point of view. However, the MF model can improve on this result.

The classification case being discussed is a typical managerial pattern recognition problem. The assumption about monotonic relationships between selection of Alpha radial tires and the four pattern variables, including last brand purchased, pre-exposure interest, post-exposure believability, and post-exposure interest is valid. The Monotonic Function model is therefore applicable, and an experiment was designed as follows. Every other point of each class was selected for one subset S_1, and the remainder was selected for S_2, such that each subset contained 39 points from class 1 and 87 points from class 2. In the first trial S_1 was the training data set and S_2 was the test data set, and in the second trial the roles of the two subsets were switched.

Because this was a high dimensionality problem, it is not possible to depict the results on a two dimension graph. Nevertheless, the clustering phenomenon was well observed in the vector analysis. In the two trials, ξ=0.01 and ξ=0.03 were applied respectively to obtain proper training data sets, beginning with the LDA result. Neural networks with 15 hidden nodes were employed, and

the learning rate was set at 0.1. The results obtained in this experiment are compared with the LDA results in Table 1. The experiment shows that the classification result of the MF model improves on the LDA method, although the results of the MF model were based on a limited number of trial and error processes in the determination of ξ.

*** Insert Table 1 about here ***

4. Discussion

4.1. Efficiency

A well-known problem with the standard BPLMS algorithm is the significant computational time required to reach a convergent result [15]. One may expect that, on average, the more complex y-surface desired, the more iteration time would be required for the neural network learning process [17] [31]. In the MF model developed in this research, one of the underlying principles is to reduce the unnecessary, even harmful, complexity of the y-surface by reducing the impact of statistical fluctuations on the location of the classification boundary. As a consequence, the neural network generates the y-surface with its simple monotonic features easily, given the proper training data set. This can be explained by an example. Suppose we have two sample data points which have the same pattern vector **X** but with opposite outcomes of c_1 and c_2 respectively. From the statistical point of view, this phenomenon is the result of statistical fluctuations. Applying the standard BPLMS algorithm,

these results will never be classified. Let us change the example slightly so that the two sample points have marginally different pattern vector values, say, $\mathbf{X}$ and $\mathbf{X}\pm$ respectively. Then, the standard BPLMS algorithm will probably take a very long time to generate a boundary to separate the two points. However, the MF model deals with this problem with considerably more finesse. It first pre-processes the data fully, taking statistical considerations of both conflict points into account and obtaining a proper training data set. The point which is furthest from the pre-determined linear boundary would be ignored during further learning, but would still have its impact consolidated in the pre-process boundary result. The neural network then learns the proper training data set which is not subject to undue statistical fluctuation, and the generated y-surface will have a less complex (and monotonic) form. In the case where there is a great deal of overlap from sample data sets, this MF model approach is usually much more efficient than the standard BPLMS method, as shown in the example.

4.2. Validity of Monotonicity

Sometimes, the relationship between the initially defined x_j (j-th component of $\mathbf{X}$) and y may not be strictly monotonic. However, if the inflection point(s) is (are) known, it is possible to decompose x_j into two or more dimensions in order to obtain a monotonic relationship. This is illustrated in Figure 5.

*** Insert Figure 5 about here ***

Note that, if the slope of the boundary at the inflection point is not very large, the classification result will not be very sensitive to the accuracy of the inflection point's location (Figure 5). The important point here is that information about an inflection point, even if imprecise, still makes it possible to efficiently apply linear functions to obtain an approximate classification result. As well, the monotonic constraints on neural network learning behavior tend to further improve classification results.

4.3. Robustness of the MF Model

When developing a model dealing with statistical data, robustness is often used to evaluate the model from the statistical point of view. Robustness signifies insensitivity to small deviations from the assumed underlying situation, including randomness, independence, distribution functions, etc. [32]. Although it is difficult to prove the present models' robustness directly since statistical tools are hard to apply, the robustness of the MF model can be compared to the LDA statistical model results. The MF model developed in the present research is designed to further reduce misclassification rates from the LDA classification technique. Given the fact that there exists a large amount of literature to explain the robustness of the LDA (e.g. [33]), robustness arguments regarding the present basic model may be stated as follows: the robustness of the present model is at least as good as the LDA in terms of low misclassification rates on training data sets. The limited results obtained so far indicates that there is also an improvement relative to LDA in test data sets.

LDA is used as a pre-processing tool in the proposed algorithm primarily because of considerations involving robustness. Actually, there are a number of linear classifiers (e.g. the perceptron) which are able to pre-process statistical data, although some of them are less well grounded theoretically than the LDA technique.

4.4. Generalization of The Model

The application of the Monotonic Function Model described above may be extended readily to k>2 problems. Suppose that we have no other knowledge except for the statistical data set consisting of discrete sample points. The linear function classifier [34][5] would seem to be the only feasible method to develop a starting "prototype" in k>2 classification. There would be (k-1) or k(k-1)/2 linear classification functions for the k class problem, depending upon the separability criteria [34][5]. Given this approximate classification result the neural network model with its highly adaptive nature could be used to improve on this result as in the k=2 case. The idea of extending the k=2 model to the k>2 case is then straightforward; that is, given a k>2 problem, use a linear classifier to obtain the approximate classification result, then employ neural network techniques to reduce misclassifications. Two types of neural network structures can be developed; those with (k-1) output nodes and those with k(k-1)/2 output nodes, respectively, according to the separability criteria. Each output node in the neural network then corresponds to one linear function classifier in the beginning prototype.

REFERENCES

[1] T. Y. Young and T. W. Calvert, Classification, Estimation and Pattern Recognition. New York: American Elsevier Publishing Company, Inc., 1974.

[2] W. J. Karplus, "The Spectrum of Mathematical Models," Perspectives in Computing, Vol.3, No.2, pp. 4-13, 1983.

[3] J. Martin and S. Oxman, Building Expert Systems: A Tutorial. Englewood Cliffs, New Jersey: Prentic Hall, 1988.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees. Belmont, California: Wadsworth International Group, 1984.

[5] D. J. Hand, Discrimination and Classification. New York: John Willey & Sons, 1981.

[6] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," Annals of Eugenics, Vol. 7, pp. 179-188, 1936.

[7] F. Rosenblatt, Principles of Neurodynamics: Perceptions and the Theory of Brain Mechanisms. Washington D.C.: Spartan Books, 1962.

[8] R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, Vol. 2, pp. 4-22, 1987.

[9] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Volume 1: Foundations. Cambridge, Massachusetts: A Bradford Book, The MIT Press, 1986.

[10] G. Widrow and M. E. Hoff, "Adaptive Switching Circuits," Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4, 1960, pp. 96-104.

[11] S. Lehar and J. Weaver, "A Developmental Approach to Neural Network Design," in Proc. IEEE First International Conference on Neural Networks, Vol. 2 (San Diego, CA), 1987, pp. 97-105.

[12] B. O'Reilly, "Computers That Think Like People," Fortune, Vol. 119, pp. 90-93, 1989.

[13] F. F. Soulie, P. Gallinari, Y. L. Cun, and S. Thiria, "Evaluation of Network Architectures on Test Learning Tasks," in Proc. IEEE First International Conference on Neural Networks, Vol. 2 (San Diego, CA), 1987, pp. 653-660.

[14] T. Kohonen, G. Barna, and R. Chrisley, "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies," in Proc. IEEE International Conference on Neural Networks, Vol. 1 (San Diego, CA), 1988, pp. 61-68.

[15] D. F. Specht, "Probabilistic Neural Networks," Neural Networks, Vol. 3, No. 1, pp. 109-118, 1990a.

[16] D. F. Specht, "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," IEEE Transactions on Neural Networks, Vol. 1, No. 1, pp. 111-121, 1990b.

[17] R. P. Lippmann, "Pattern Classification Using Neural Networks," IEEE Communications Magazine, Vol. 27, No. 11, pp. 47-64, 1989.

[18] G. Cybenko, "Approximation by Superpositions of A Sigmoidal Function," Mathematics of Control, Signals, and Systems, Vol. 2, No. 4, pp. 303-314, 1989.

[19] E. Rich, Artificial Intelligence. New York: McGraw-Hill Book Company, 1983.

[20] E. I. Altman, "Financial Ratios, Discriminant Analysis and The Prediction of Corporate Bankruptcy," The Journal of Finance, Vol. 28, No. 4, pp. 589-609, 1968.

[21] J. J. McGrath, "Improving Credit Evaluation with a Weighted Application Blank," Journal of Applied Psychology, Vol. 44, No. 5, pp. 325-328, 1960.

[22] C. C. Greer, "Deciding to Accept or Reject a Marginal Retail Credit Application," Journal of Retailing, Vol. 43, No. 4, pp. 44-53, 1968.

[23] J. H. Myers and E. W. Forgy, "The Development of Numerical Credit Evaluation Systems," Journal of the American Statistics Association, Vol. 58, No. 303, pp. 799-806, 1963.

[24] V. Ramanujam, N. Venkatraman, and J. C. Camillus, "Multi-Objective Assessment of Effectiveness of Strategic Planning: A Discriminant Analysis Approach," Academy of Management Journal, Vol. 29, No. 2, pp. 347-372, 1986.

[25] R. H. Keeney and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs. New York: John Wiley & Sons, 1976.

[26] Y. T. Chien and T. J. Killeen, "Computer and Statistical Considerations for Oil Spill Identification," in Handbook of Statistics 2: Classification, Pattern Recognition and Reduction of Dimensionality, P. R. Krishnaiah and L. N. Kanel Eds., New York: North-Holland, 1982, pp. 651-671.

[27] R. A. Eisenbeis, "Pitfalls in the Application of Discriminant Analysis in Business, Finance, and Economics," The Journal of Finance, Vol. 32, No. 3, pp. 875-900, 1977.

[28] T. Kohonen, Self-Organization and Associative Memory. Berlin: Springer-Verlag, 1988.

[29] B. Efron, The Jackknife, the Bootstrap and Other Resampling Plans. Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics, 1982.

[30] P. E. Green, Analyzing Multivariate Data. Hinsdale, Illinois: The Dryden Press, 1978.

[31] A. Wieland and R. Leighton, "Geometric Analysis of Neural Network Capabilities," in Proc. IEEE First International Conference on Neural Networks, Vol. 3 (San Diego, CA), 1987, pp. 385-392.

[32] P. J. Huber, Robust Statistics, New York: John Wiley & Sons, 1981.

[33] P. A. Lachenbruch, Discriminant Analysis. New York: Hafner Press, 1975.

[34] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis. New York: John Wiley & Sons, 1973.

APPENDIX

Monotonic Condition In The MF Model

This Appendix is a derivation of the monotonic condition in the MF model. The numbers in brackets to the left correspond to the numbered sections of the algorithm in Section 3.1.

(1) Let

$$\sum_{i=1}^{h} w_{ij}\, v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2} =$$

$$\sum_{i\in I} w_{ij}\, v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2} +$$

$$\sum_{i\notin I} w_{ij}\, v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}$$

where I is a set such that, if $i\in I$, $w_{ij}v_i<0$, that is,

$$\sum_{i\in I} w_{ij}\, v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}<0 \ ;$$

and if $i\notin I$ for $w_{ij}v_i\geq 0$, that is,

$$\sum_{i\notin I} w_{ij}\, v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}\geq 0 \ ;$$

(since $\exp(-\mathbf{U}_i\mathbf{X}')\ [\ (1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2} > 0$).

(2) Monotonicity $\Longleftrightarrow$ $\sum_{i=1}^{h} w_{ij}\, v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2} \geq 0$,

and each term in the sum is independent. Thus,

$$\min \sum_{i\in I} w_{ij} v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}$$

$$+ \min \sum_{i\notin I} w_{ij} v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2} \geq 0$$

is a necessary and sufficient condition for monotonicity.

Denote $\min \sum_{i\in I} w_{ij} v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}$ and

$\min \sum_{i\notin I} w_{ij} v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}$ as Q_1 and Q_2 ,

respectively.

$\exp(-\mathbf{U}_i\mathbf{X}')\ [\ (1 + \exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}$ has a maximum of 0.25 at $\mathbf{U}_i\mathbf{X}'=0$ (because at least one point $\mathbf{X}=\underline{0}$, and this maximum occurs at that point), hence

$$Q_1 = 0.25 \sum_{i\in I} w_{ij} v_i \ .$$

(3) We now need to find $Q_2 = \min \sum_{i\notin I} w_{ij} v_i \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}$

(when $w_{ij}v_i \geq 0 \quad \forall\, i$).

Let $R_i = \exp(-\mathbf{U}_i\mathbf{X}')[(1+\exp(-\mathbf{U}_i\mathbf{X}'))]^{-2}$. R_i is a function of the quantity $\sum_{r=0}^{m} w_{ir}x_r$, since $R_i = \exp(-\sum_{r=0}^{m} w_{ir}x_r)[(1+\exp(-\sum_{r=0}^{m} w_{ir}x_r))]^{-2}$.

The behavior of R_i as a function of $\sum_{r=0}^{m} w_{ir}x_r$ is shown in Figure 6.

*** Insert Figure 6 about here ***

Since $0 \le x_r \le 1$, we have

$$0 \le \sum_r w_{ir} x_r \le \sum_r w_{ir} \qquad \forall w_{ir} \ge 0 \text{ , and}$$

$$\sum_r w_{ir} \le \sum_r w_{ir} x_r \le 0 \qquad \forall w_{ir} < 0.$$

The extreme points are P_1 (x_r=1 when $w_{ir}<0$, and x_r=0 when $w_{ir} \ge 0$) and

P_2 (x_r=1 when $w_{ir} \ge 0$, and x_r=0 when $w_{ir}<0$).

Note that the two extreme points P1 and P2 of the function R_i depend on the specific $\sum_r w_{ir}$ values, and either point could be the minimum.

Thus,

$$Q_2 = \min \sum_{i \notin I} w_{ij} v_i \exp(-\mathbf{U}_i \mathbf{X}')[(1+\exp(-\mathbf{U}_i \mathbf{X}'))]^{-2}$$

$$= \sum_{i \notin I} \min\, w_{ij} v_i \exp(-\mathbf{U}_i \mathbf{X}')[(1+\exp(-\mathbf{U}_i \mathbf{X}'))]^{-2}$$

(since each independent term ≥ 0)

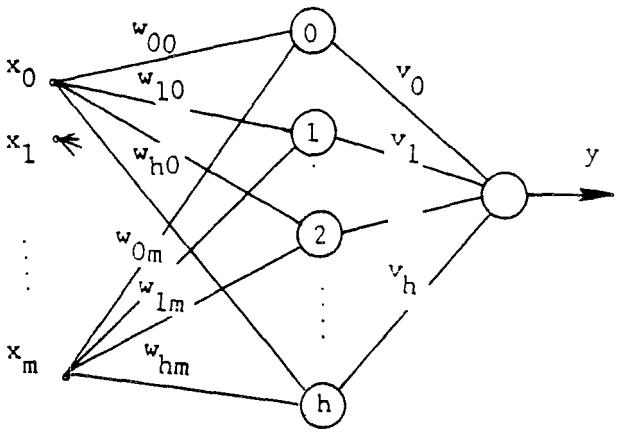
$$= \sum w_{ij} v_i \, (\min \exp(-\mathbf{U}_i \mathbf{X}')[(1+\exp(-\mathbf{U}_i \mathbf{X}'))]^{-2})$$

$$= \sum w_{ij} v_i \, A_i$$

where A_i is the minimum value of P1 and P2.

(4) If $Q_1 + Q_2 \ge 0$ then $\sum_{i=1}^{h} w_{ij} \, v_i \exp(-\mathbf{U}_i \mathbf{X}')[(1+\exp(-\mathbf{U}_i \mathbf{X}'))]^{-2} \ge 0$,

that is, y is monotonic; else monotonicity is violated.



$$U_i = (w_{i0}, w_{i1} \cdots w_{im}) \quad i=0,1 \ldots h$$

Figure 1. Two Layer, Single Output Neural Network

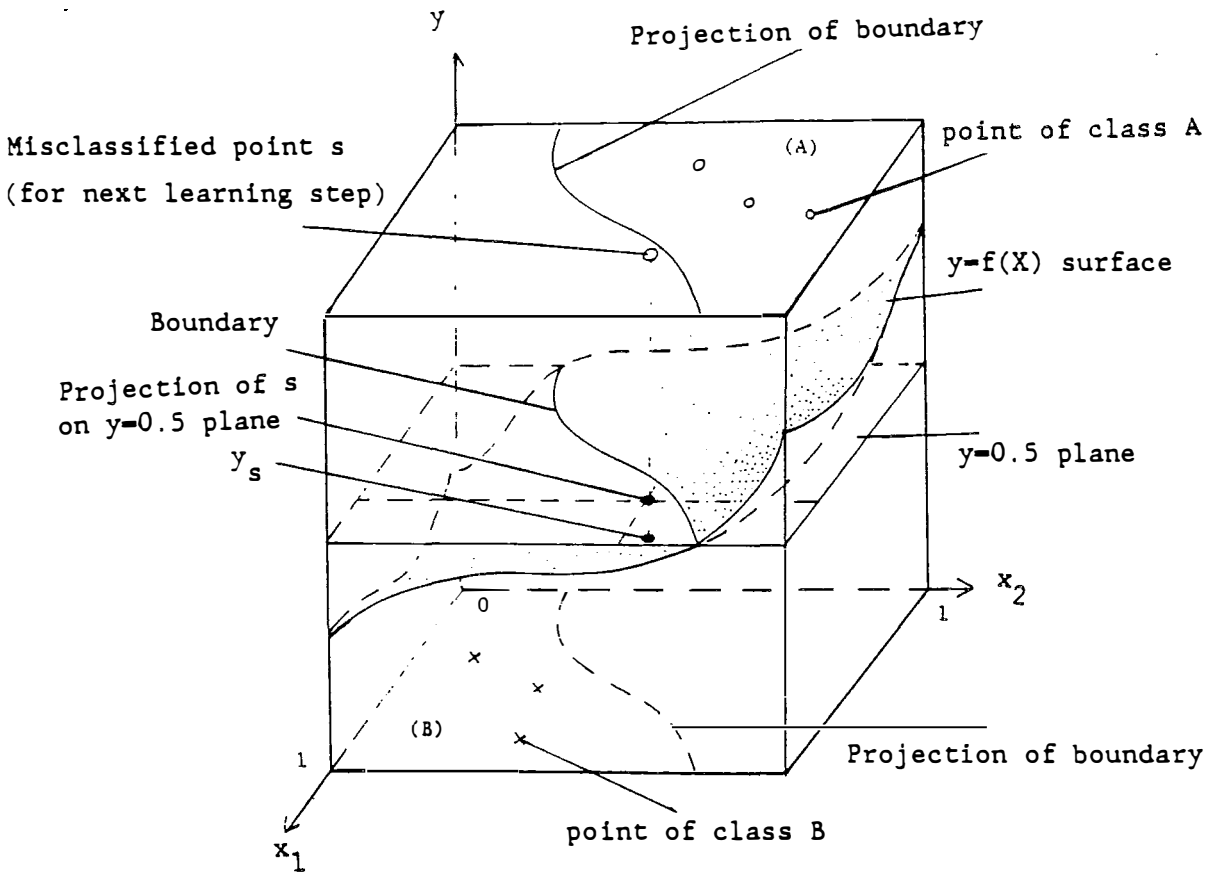
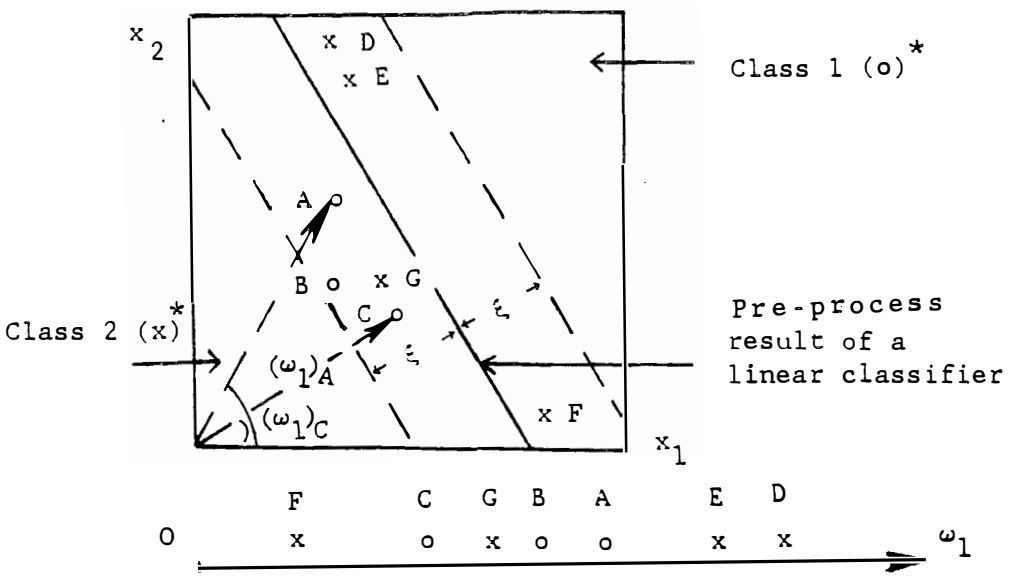


Figure 2. Neural Network Functional Cube



* Irrelevant sample points not shown.

Figure 3. Vector Analysis for Training Sample Set

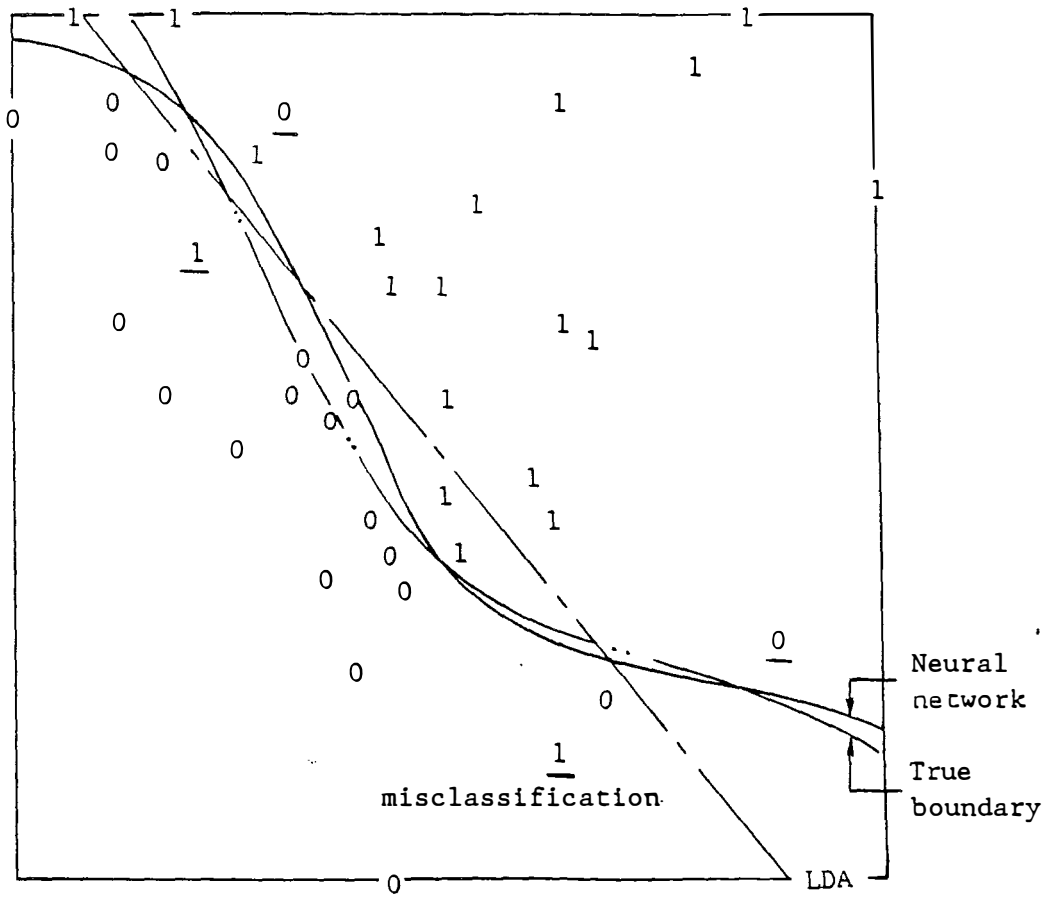


Figure 4. Experimental Result for the Monotonic Function Neural Network Classifier

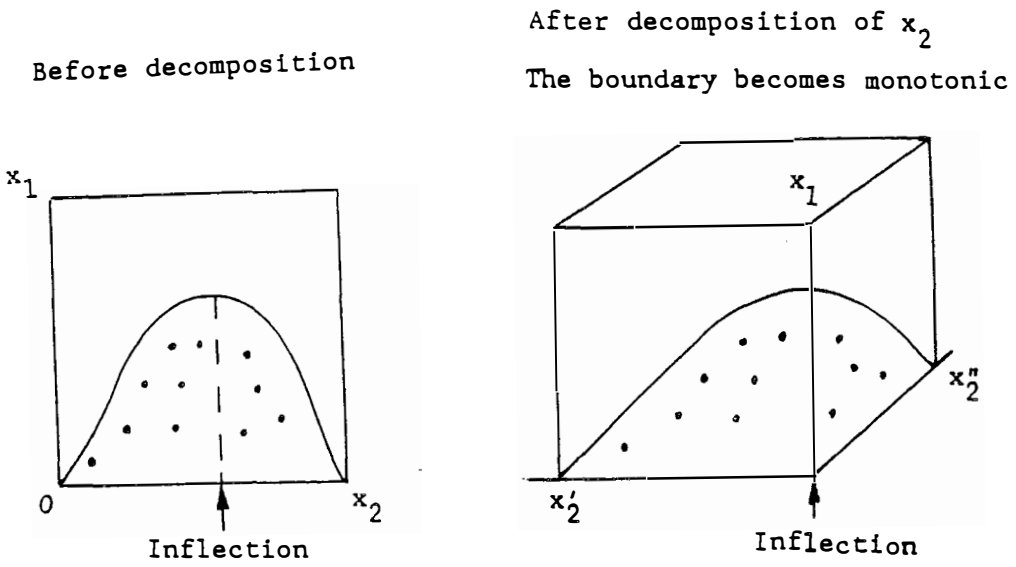


Figure 5. Decomposition of a Pattern Dimension

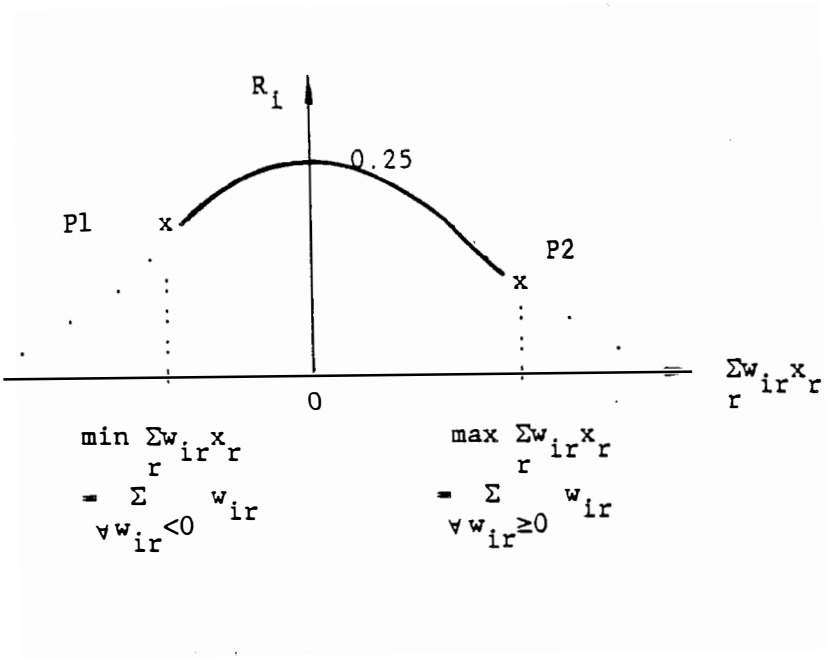


Figure 6. The Behavior of R_i as a Function of $\sum_{r=0}^{m} w_{ir} x_r$

		LDA	MFM	Reduced misclassification
Train on the original data set (252 points) and test on the same data set		(52+131)/252 = 72.6 %		
Trial 1. Train on subset S1 (126 points)	Test on S1	(26+67)/126 = 73.8 %	(26+77)/126 = 81.7 %	(10/33)= 30.3 %
	Test on S2	(24+71)/126 = 75.4 %	(20+76)/126 = 76.2 %	(1/31)= 3.2 %
	Overall	74.6 %	79.0 %	
Trial 2. Train on subset S2 (126 points)	Test on S1	(31+56)/126 = 69.0 %	(25+67)/126 = 73.0	(5/39)= 12.8 %
	Test on S2	(26+58)/126 = 66.7 %	(24+71)/126 = 75.4 %	(11/42)= 26.2 %
	Overall	67.9 %	74.2 %	
Overall		71.3 %	76.6 %	

Table 1. A Comparison of the Percentage Correctly Classified by the MF Model and the LDA Method on the Alpha TV Commercial Study Data [30]

Faculty of Business
McMaster University

WORKING PAPERS - RECENT RELEASES

314. Harish C. Jain and Rick D. Hackett, "Employment Equity in Canada: Public Policy and a Survey", November, 1988.

315. Harish C. Jain, "The Recruitment and Selection of Visible Minorities in Canadian Police Organizations: 1985-1987", November, 1988.

316. Thomas E. Muller, "The Two Nations of Canada vs. The Nine Nations of North America: A Cross-Cultural Analysis of Consumers' Personal Values", November, 1988.

317. Min S. Basadur, Mitsuru Wakabayashi and George B. Graen, "Identifying Creative Problem Solving Style", December, 1988.

318. Robert G. Cooper and Elko J. Kleinschmidt, "New Product Success Factors: A Comparison of "Kills" versus Successes and Failures", December, 1988.

319. P. Ho and Ali R. Montazemi, "Evaluating Decision Strategies For Stock Market Investment", January, 1989.

320. Isik Zeytinoglu, "Part-Time and Occasional Teachers In Ontario's Elementary School System", January, 1989.

321. G. John Miltenburg and T. Goldstein, "Developing Production Schedules Which Balance Part Usage and Smooth Production Loads In Just-In-Time Production Systems", February, 1989.

322. Winston H. Mahatoo, "Background Information for Marketing Visitors to the People's Republic of China", February, 1989.

323. Min Basadur, Mitsuru Wakabayashi, George B. Graen, "Attitudes Towards Divergent Thinking Before and After Training: Focusing Upon the Effect of Individual Problem Solving Styles, March, 1989.

324. Tom E. Muller, "Canada's Aging Population and Projected Changes in Value Orientations and the Demand for Urban Services", March, 1989.

325. Robert G. Cooper, "Stage-Gate Systems: A New Tool for Managing New Products", March 1989.

326. Harish C. Jain, "Racial Minorities and Affirmative Action/ Employment Equity Legislation in Canada", April, 1989.

327. Christopher K. Bart, "Controlling New Products in Large Diversified Firms: A Presidential Perspective", April, 1989.

328. Robert G. Cooper, "Compressing the New Product Time Cycle", May, 1989.

329. Danny I. Cho and Mahmut Parlar, "A Survey of Maintenance Models", July, 1989.

330. Min Basadur, Mitsuru Wakabayashi and Jiro Takai, "Training Effects on Japanese Managers' Attitudes Toward Divergent Thinking", July, 1989.

331. Robert F. Love, "Properties of Ordinary and Weighted Sums of Order p", July, 1989.

332. Robert F. Love, "Floor Layouts Using a Multi-Facility Location Model", July, 1989.

333. Thomas E. Muller, "Staying Ahead of the Consumer: Signals About the Future from North America's Aging Population", September, 1989.

334. Robert G. Cooper, "New Products: What Distinguishes the Winners", November, 1989.

335. Robert G. Cooper and Elko J. Kleinschmidt, "Firms' Experiences Using a Formal New Product Process", January, 1990.

336. Joseph B. Rose and Gary N. Chaison, "Fortune and Misfortune: Union Growth in Canada and the United States", January, 1990.

337. Peter J. Sloane and Harish C. Jain, "Use of Equal Opportunities Legislation and Earnings Differentials: A Comparative Study".

338. John Medcof, "The Probabilistic Contrast Model and PEAT".

339. Peter Banting, "Supplying the Samurai".

340. Joseph B. Rose and Gary N. Chaison, "New Directions and Divergent Paths: The North American Labor Movements in Troubled Times".

341. Peter M. Banting and David L. Blenkhorn, "Developing and Managing Japanese and U.S. OEM -- Canadian Autoparts Supplier Relationships in the 1990s."

342. R.G. Cooper and E.J. Kleinschmidt, "New Products: The Key Factors in Success".