RSRE
MEMORANDUM No. 4346

ROYAL SIGNALS & RADAR ESTABLISHMENT

IMPROVING UPON STANDARD PATTERN CLASSIFICATION ALGORITHMS BY IMPLEMENTING THEM AS MULTI-LAYER PERCEPTRONS

Author: M D Bedworth

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
R S R E MALVERN,
WORCS.

RSRE MEMORANDUM No. 4346

Royal Signals and Radar Establishment
Memorandum: 4346

Improving upon Standard Pattern Classification Algorithms by Implementing them as Multi–Layer Perceptrons.

Mark D. Bedworth
Speech Research Unit, SP4,
Royal Signals and Radar Establishment,
St. Andrews, Great Malvern, England

12th December 1989

Abstract

The multi–layer perceptron (MLP) is a type of adaptive layered network often used as a pattern classifier. In more recent literature, MLPs are compared with simpler classification techniques using common datasets. We select two of these simple static pattern classification algorithms and briefly review the relevant techniques. After introducing a modest set of evaluation databases, the performance of the standard classifiers and MLPs are assessed. A technique for implementing the two standard classifiers as MLPs is presented and this novel approach is used to automatically design a 'good' set of initial weights for the MLP networks. Encouraging experimental results for these hybrid techniques are shown for illustration.

[illegible] pattern recognition, speech [illegible]
networks [illegible]

1 Introduction

Although pattern classification techniques have existed for many years; a recent addition, the multi-layer perceptron, has attracted widespread interest. The multi-layer perceptron (MLP) is a type of adaptive feed-forward layered network, trained by example with a labeled set of training vectors. Many pattern classification experiments using these networks can be found in the literature, including speech and image pattern processing tasks [6, 10, 11].

In early experiments, comparisons were rarely made between MLPs and other standard classifiers using a common database. Although the **potential** recognition performance of a suitable MLP network is superior to simpler techniques, there exists no guarantee that the *nonlinear* learning algorithm will find an optimum solution (although a locally optimal solution would often be considered adequate). In some cases the final performance of the trained MLP on the set of test patterns is **inferior** to the other techniques; many random weight starts being required before a network with high enough performance is found (often tens or even hundreds of runs).

We introduce a method for initialising the weights of MLP networks to produce reasonable performance **before** training. These starting values should be close to a suitable optimum; random valued weights are more likely to be distant from such an optimum (a situation where nonlinear optimisation algorithms are known to behave unpredictably).

2 Nearest class mean classifiers

The nearest class mean classifier assigns the class, ω, of the nearest class mean $\underline{M}_j$ to the unknown input vector, $\underline{P}_i$

$$\omega(\underline{P_i}) = \omega(\underline{M}_j) \text{ if } \Delta(\underline{P}_i, \underline{M}_j) < \Delta(\underline{P}_i, \underline{M}_k) \forall k \neq j, \tag{1}$$

The distance metric, Δ, is often the Euclidean distance Δ_E,

$$\Delta_E(\underline{P}_i, M_j)^2 = \sum_l (P_{il} - M_{jl})^2. \tag{2}$$

In the following text, "Euclidean distance to class mean classifier" will be abbreviated to NCM.

The Mahalanobis distance Δ_M would usually give a higher classification rate, although there needs to be sufficient training data available to reliably estimate the covariance matrices, Σ_j, of the distributions of each class j

$$\Delta_M(\underline{P}_i, \underline{M}_j)^2 = (\underline{P}_i - \underline{M}_j)\Sigma^{-1}(\underline{P}_i - \underline{M}_j)^T \tag{3}$$

The abbreviation MDCM will be used in the following sections to denote a "Mahalanobis distance to class mean classifier".

A full explanation of exemplar classifiers and associated techniques can be found in [7].

		input	number of	number of
	number	vector	training	test
	of classes	size	patterns	patterns
Vowel formant	10	2	338	333
Vowel spectrum	11	54	154	154
Isolated word	8	760	160	160
Radar	5	10	1360	1230

3 The four test problems

The four test problems used in this paper consist of two speaker independent vowel recognition tasks, a speaker dependent isolated word recognition and a radar signal recognition problem.

The two classification methods outlined above were applied to the four test problems selected for the study. In additiou, the nearest neighbour (NN) and K-nearest neighbour (KNN) were also applied to the data.

The NN technique assigns to the unknown pattern the class of the nearest training data point rather than class mean, and KNN takes votes among the K nearest training data points before assigning a class label. All reasonable values for K were tried and the performance of the classifer with the value of K giving the highest score **on the test set** was reported. Euclidean distance was used for the NN and KNN techniques.

3.1 Vowel formant data

The data consisted of the frequencies of the first and second formants of ten vowel sounds spoken by 76 speakers, both male and female [12]. The data was divided into a training set (338 labeled frequency pairs) and a test set (333 labeled examples).

The frequency of the first formant was between 200Hz and 1200Hz, the frequency of the second formant between 500Hz and 3500Hz, the values for each formant were scaled to lie between 0 and 1. The scaling factors were estimated from the training set - the values in the test set were, of course, scaled using the same factors. The identity of each of the vowel sounds was identified by the letters "a" through "j".

As can be seen in figure 1 the distributions of the 10 classes overlap to a large extent making this a difficult problem for automatic pattern recognition algorithms.

classifier	training set performance	test set performance
NCM (Euclidean)	67.75	71.47
NCM (Mahalanobis)	78.40	80.18
NN	100.00	76.58
KNN ($k = 5$)	82.25	81.08

Table 1: The performance of the standard classifiers on the vowel formant data.

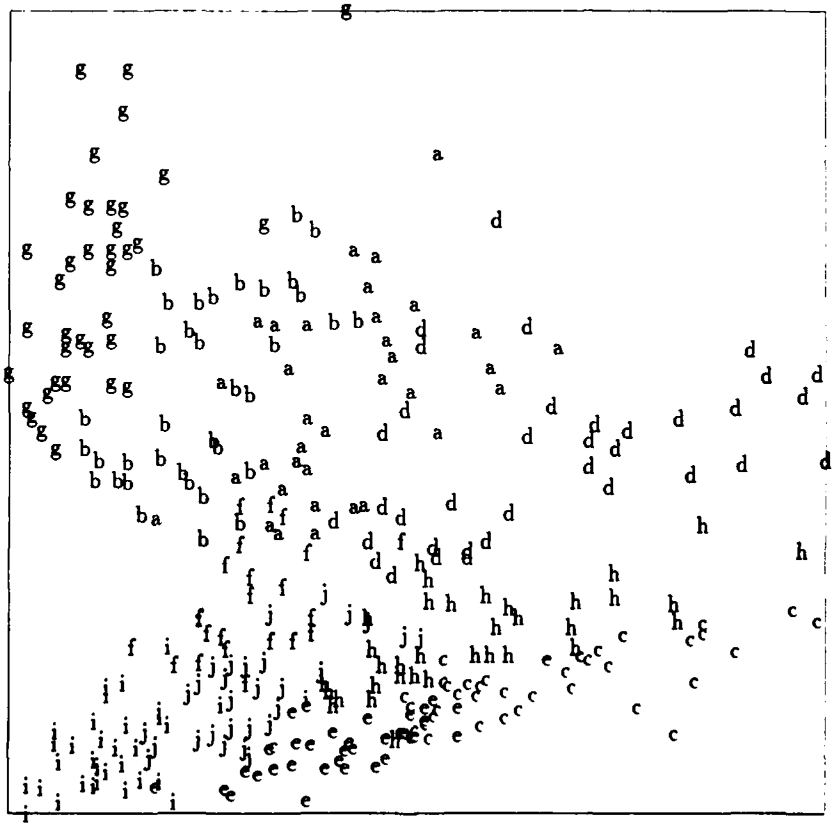

Figure 1: The training data for the vowel formant task. The distributions for each of the 10 classes (labeled "a" through "j") are illustrated (with the frequencies of the first and second formants on horizontal and vertical axes respectively).

3.2 Vowel spectrum data

The task was to recognize one of 11 quasi-stationary vowel sounds by their spectral shape. The input pattern was a 54 channel spectrum of an averaged portion of the steady state vowel of 11 training words {'head','heed','hod','hud','heard','hood','had','hard','hid','who'd','hoard'}. Figure 2 shows the spectra for the first 11 training patterns.

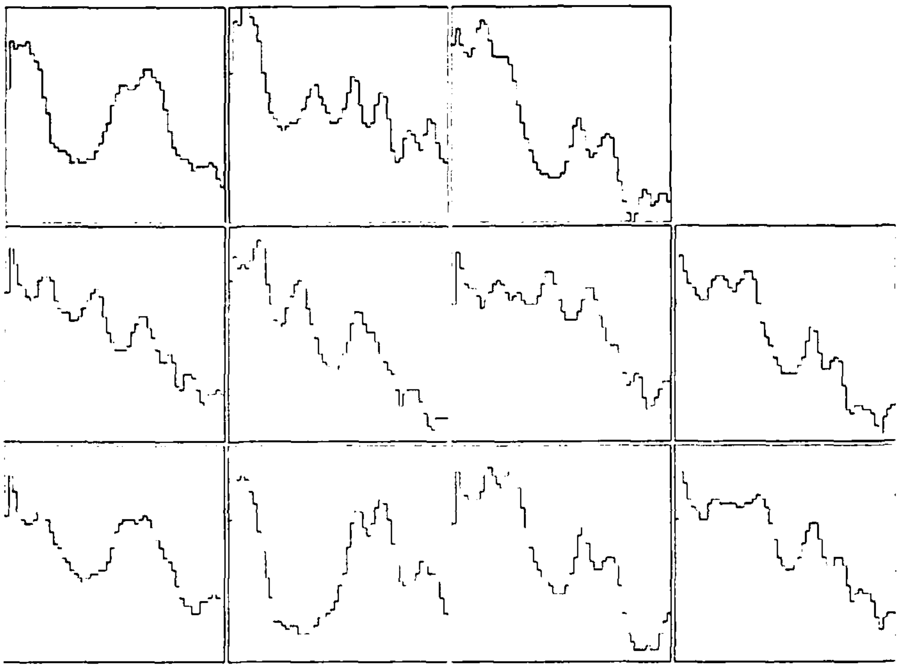

Figure 2: The spectra of the first 11 patterns from the training file. Each spectrum has 54 frequency values (horizontal axis).

The database consisted of the 11 vowels from 28 speakers (14 speakers in the training set and 14 in the test set; both sets containing male and female speakers). This problem has been studied in detail by others, see [10].

There are no results available for the Mahalanobis distance to class mean technique as there was insufficient training data to estimate the covariance matrices of the within class distributions of the 11 vowel spectra.

3.3 Isolated word recognition data

The problem is a small scale speaker-dependent speech recognition application. The task was to discriminate between the confusable 'EE' sounds from the English alphabet i.e. {'B', 'C', 'D', 'E', 'G', 'P', 'T', 'V'}. A filter bank analysis of the isolated utterances of a consistent speaker (using the JSRU hardware channel vocoder [8]) produced an average spectral density in each of nineteen frequency channels (spanning ≈ 200Hz to ≈ 4kHz) every 20ms. The values were scaled to lie in the interval [0, 1]. The utterances were between 500ms and 800ms in length and were aligned in a 40 frame window so that the start of the word was in the third time frame, see figure 3. Other aspects of this problem have been studied elsewhere, see [1, 14].

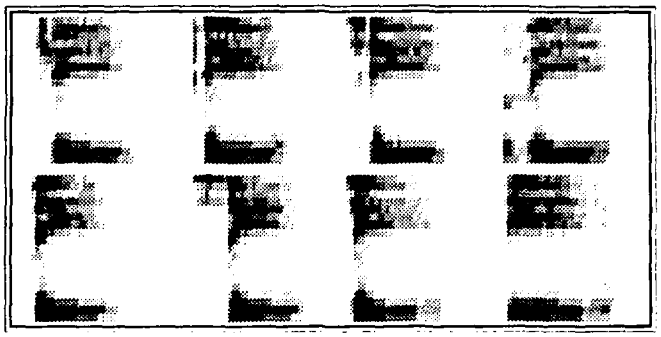

Figure 3: Eight patterns from the training set of the isolated "EE" word recognition task. Frequency is on the vertical axis and time on the horizontal axis. Each of the utterances is aligned within a 40 frame window (not shown).

There are no results available for the Mahalanobis distance to class mean technique as there was insufficient training data to estimate the covariance matrices of the within class distributions of the 8 'EE' words.

classifier	training set performance	test set performance
NCM (Euclidean)	77.92	66.88
NCM (Mahalanobis)	n/a	n/a
NN	100.00	71.43
KNN ($k = 3$)	90.26	74.03

Table 2: The performance of a the standard classifiers on the vowel spectrum data.

classifier	training set performance	test set performance
NCM (Euclidean)	91.25	85.63
NCM (Mahalanobis)	n/a	n/a
NN	100.00	90.63
KNN ($k = 5$)	97.50	93.75

Table 3: The performance of the standard classifiers on the isolated word data.

3.4 Radar data

In earlier work this task was analysed using a number of pattern recognition methods [4]. The original data consisted of over 25,000 radar signals of five classes. Each input vector consisted of 26 features of the radar signal (the feature set was designed for a different pattern classification algorithm).

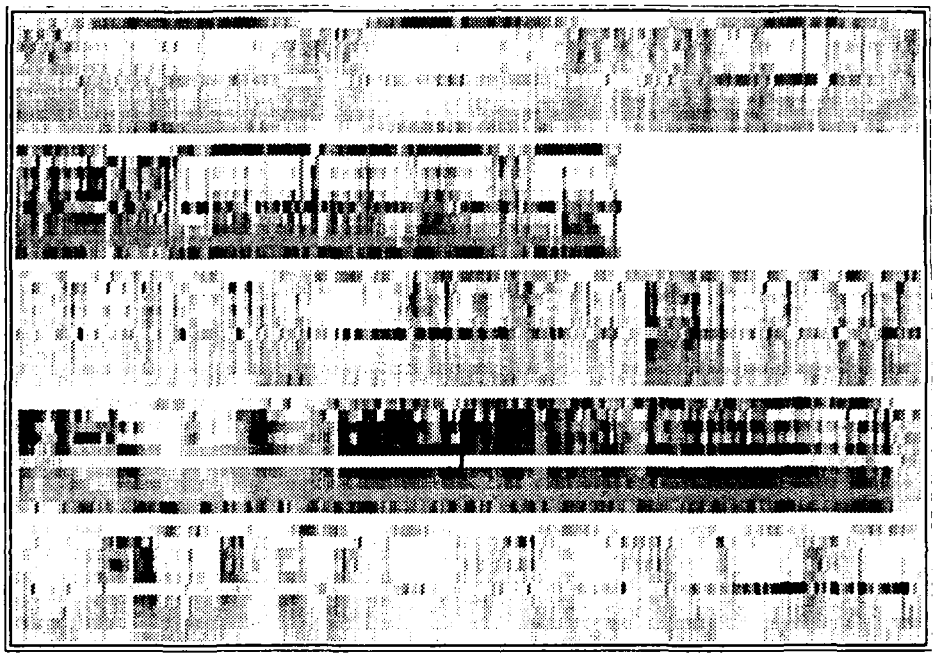

Figure 4: Every tenth input vector from the radar classification training set. Each vertical slice represents the 10 features in the input pattern. Each horizontal block shows the training patterns associated with one of the five classes.

For the purposes of this study the data was subsampled by taking every tenth vector from the training and test datasets (leaving 1360 and 1230 vectors in the training and test datasets respectively). In addition, the 10 features with the highest variance ratio (the ratio between the variance of within class means to the mean of the within class variances for each input unit) were extracted, the 16 features with the lowest variance ratio being discarded.

4 Summary of standard results

The following table summarises the performance of the best of the standard techniques applied to each of the four test problems. The best technique was the one which gave the highest score on the test data.

5 The multi–layer perceptron

Rumelhart *et al* [13] describe a computational network, the multi-layer perceptron (MLP), which consists of a set of elementary processing units linked by weighted connections. The transfer functions of each processing unit and the values of the weights linking the units determine the specific behaviour of the network. An input vector representing a pattern to be recognised is applied to some of the units (input units) and an output vector collected at a set of output units, representing the result of the classification. During training the network can exploit the hidden units (which are not exposed directly to either the input or the output) to encode distributed internal representations in patterns of weight values. The network is taught by example with a training set, and attempts to minimize the error between its own output vector and the desired target vector for each of the training patterns.

Most application experiments with MLPs have used layered networks of so–called logistic units (as described in the original paper). The technique is of course applicable to any feed forward network composed of processing units with suitably differentiable transfer functions [3], we will use networks of non–logistic units in a later section.

The processing units in the usual MLP have a scalar product fan–in and logistic nonlinearity:-

$$O_{pj} = \frac{1}{1 + e^{-I_{pj}}}, \tag{4}$$

and

$$I_{pj} = \sum_i O_{pi} w_{ij}, \tag{5}$$

where O_{pj} is the output of the jth unit for pattern p, I_{pj} is the net input to the same unit and w_{ij} is the weight linking the ith unit to the jth unit.

The training algorithm, error back propagation (EBP), estimates the partial derivatives of an error measure with respect to each of the weights. This error is reduced by iteratively updating the weight values using one of a number of nonlinear optimisation techniques. For the experiments reported here the conjugate gradients nonlinear optimisation strategy was used [14]. This is essentially a parameterless method unlike the more commonly used accelerated gradient descent technique. A preliminary investigation of the use of gradient descent with the data sets shown here indicated that a significant amount of trial and error was necessary to arrive at good values for the learning parameters (η and α). Although final performance was in some instances good, the total training time was much greater than for networks trained using conjugate gradients.

The performance of the network is taken to be the proportion of patterns for which the output vector was closer (in Euclidean distance) to the target vector of the correct class than to that of any other class.

The error reported in the results is the normalised error

$$\mathcal{E} = \sqrt{\frac{\sum_{pi}(O_{pi} - T_{pi})^2}{\sum_{pi}(T_{pi} - \bar{T}_i)^2}}. \qquad (6)$$

A normalised error of 1.0 can be produced by predicting in the mean - for the problems considered here a normalised error of less than 0.5 on the training set would indicate good performance.

6 Summary of MLP results

A number of standard MLP networks (scalar product fan-in and logistic nonlinearity, fully layer connected) were trained on the four training sets above.

The number of hidden units was varied between 0 (connections directly between input and output) up to 50. Where necessary several random weight starts were used with the same network, giving, on average, 20 network runs for each problem. The weight values were typically initialized to values distributed uniformly in the interval $[-1, 1]$. Conjugate gradients was used throughout to update the weight values.

For some runs the training proceeded by first exposing the network to a subset of the data and then continuing with the full training set (incremental training). This was found to give superior final performance in most cases.

The following table details the performance of the **best** MLP trained in this way for the four test problems. The best network was chosen as the network with the highest performance **on the test data**.

The training time for that network was the CPU time (in minutes) required to produce the set of weight values using a software simulation (written in PASCAL) on a VAXstation 3200. It does not include the time used to find a reasonable number of hidden units, the best size for the initial weights or for multiple weight starts: if these times are included then the individual training times should be multiplied by up to ten.

It is noted that the final performance of the MLP networks on the tasks above is not consistently better than the nearest neighbour results.

	training set	test set
classifier	performance	performance
NCM (Euclidean)	52.35	50.81
NCM (Mahalanobis)	58.90	59.92
NN	100.00	54.47
KNN ($k = 5$)	81.62	56.59

Table 4: The performance of the standard classifiers on the radar data.

		training set	test set
	technique	performance	performance
Vowel formant	KNN ($k = 5$)	82.25	81.08
Vowel spectrum	KNN ($k = 3$)	90.26	74.03
Isolated word	KNN ($k = 5$)	97.50	93.75
Radar	NCM (Mahalanobis)	58.90	59.92

Table 5: The performance of the best of the standard classification algorithms on the four test problems.

	training set	test set	training set	test set	training
	error	error	performance	performance	time
Vowel formants	0.521626	0.594303	84.91	78.98	80
Vowel spectrum	0.444854	0.812647	83.12	70.78	95
Isolated word	0.000000	0.212789	100.00	98.13	2000
Radar	0.693767	0.838835	63.13	60.55	1800

Table 6: The performance of MLP classifier networks on the four test problems.

7 Nearest class mean MLP

Prior knowledge can be incorporated into MLP networks in a number of ways (for example structured training data [5]). The most direct method would be to automatically design the network topology, the type of processing element and the values of the weights. The prior knowledge might be that of a human expert, a rule based expert system, or some standard classification algorithm. In view of the performance figures for MLPs and nearest neighbour classifiers detailed above, the goal of building the prior knowledge of a nearest neighbour classifier into a MLP was undertaken [2].

For a nearest neighbour classifier the decision surface is piecewise linear and is composed of segments of the perpendicular bisectors of the prototype vectors with disimilar class labels. Some, but not necessarily all, of the bisectors will form the decision surface.

This decision surface can be implemented in a MLP with one hidden layer of scalar-product logistic processing units. If the decision regions are convex (as for a class mean classifier), then the structure of the network is particularly simple.

The weights leading from the ith input unit to the jth hidden unit are

$$w_{ij} = \psi(P_i - Q_i), \tag{7}$$

where $\underline{P}$ is the pattern on the positive side of the boundary and $\underline{Q}$ the pattern on the negative side of the boundary ($\omega^+(j)$ =class of $\underline{P}$ and $\omega^-(j)$ =class of $\underline{Q}$). ψ is a constant which approximates the logistic nonlinearity to a step function at the scale of the problem in hand. The bias on the jth hidden unit is

$$\theta_j = -\frac{\psi}{2}\sum_i (P_i^2 - Q_i^2), \tag{8}$$

and the equation of the decision boundary is

$$\underline{X}^T(\underline{P} - \underline{Q}) - \frac{1}{2}(\underline{P}^T\underline{P} - \underline{Q}^T\underline{Q}) = 0. \tag{9}$$

The number of hidden units will be

$$\mathcal{N}_{hidden} = \frac{1}{2}\sum_c(\sum_p \omega(p) = c).(\sum_p \omega(p) \neq c). \tag{10}$$

For a Euclidean distance to class mean classifier (for which it is guaranteed that the decision regions are convex) the number of hidden units is

$$\mathcal{N}_{hidden} = \frac{1}{2}C(C - 1), \tag{11}$$

where C is the number of classes.

If the decision regions are convex then the correct output is obtained with the logical AND of the relevant hidden units (assuming the hidden nonlinearities approach a step function).

To AND the binary valued hidden unit outputs the weights from the hidden units to the output unit are

$$\begin{aligned} w_{jk} &= \kappa \text{ if } \omega(k) = \omega^{+}(j) \\ w_{jk} &= -\kappa \text{ if } \omega(k) = \omega^{-}(j) \\ w_{jk} &= 0 \text{ otherwise} \end{aligned} \tag{12}$$

and the bias is

$$\theta_k = (\sum_{j|w_{jk}=\kappa} \kappa) - \frac{\kappa}{2}, \tag{13}$$

where κ is some arbitrary value which causes the nonlinearity to be equivalent to a step function at the scale of the particular problem (a value of about 1 or 2 seems to be a reasonable starting point for the tasks reported in this paper).

Although the decision boundary is piecewise linear and is composed wholly of perpendicular bisectors, not all perpendicular bisectors form part of the decision surface. The technique above tends to generate networks with more hidden units than necessary. For this reason all those hidden units which do not change the classification rate of a nearest class mean classifier MLP can be removed. This can be done automatically whilst the weight values are being calculated.

A single pass algorithm was used for speed and simplicity; *i.e.* starting with the full set of hyperplanes, discard the first and test the classifier - if the performance is not worse then discard it otherwise retain it; move to the next hyperplane and so on. A mulipass algorithm would probably reduce the number of hidden units (although not significantly) at the cost of increased computation. This might be worthwhile when the number of classes rises above about 15 (with 105 hidden units in the unpruned network): it will almost certainly be necessary when the number of classes is more than, say, 20 (190 hidden units in the unpruned network). The reduction in network size obtained by pruning obviously depends on the problem. For the networks reported here the reduction was typically about 50%: (the number of hidden units for the four networks reduced from 45 → 15, 55 → 22, 28 → 13 and 10 → 8).

8 Adaptive NCM/MLP results

The above technique was tried on the four test problems. In each case the constants κ and ψ were found by trying two or three values and testing the resultant network. A reasonable value for ψ is the value which scales the average distance between the class means to from about 1 to about 10 (biased towards a smaller value to allow adjustment of the weights during training). The technique seems to be robust to large variations in κ and ψ: up to an order of magnitude making little or no qualitiative difference after training.

In each case the network (with redundant hidden units pruned out with the one pass algorithm) was further trained using conjugate gradients with the full training dataset. At first just optimising the first layer of weights, then after convergence optimising all the weights (total training time is given).

	training set error	test set error	training set performance	test set performance	training time
2-15-10	0.854842	0.850803	68.93	72.37	1
2-15-10	0.463723	0.637717	87.87	79.28	30

Table 7: The performance of a pruned MLP network set up using nearest class mean after further training on the vowel formant data.

Figures 5 and 6 show the decision regions and the training data for the vowel formant MLP network after being set up as a NCM classifier, and after training using conjugate gradients optimisation.

The position of the decision boundary was found by scanning the input domain on a coarse grid (100 × 100) and recording the classification. The locations of the 10 class means are labelled with the same letters as the data points in figure 1. The lower case characters show the locations of the training data patterns which were incorrectly classified by the network; the correctly classified training patterns are shown simply as dots. Note how the softness of the nonlinearities has created spurious decision areas between classes "f", "e" and "h".

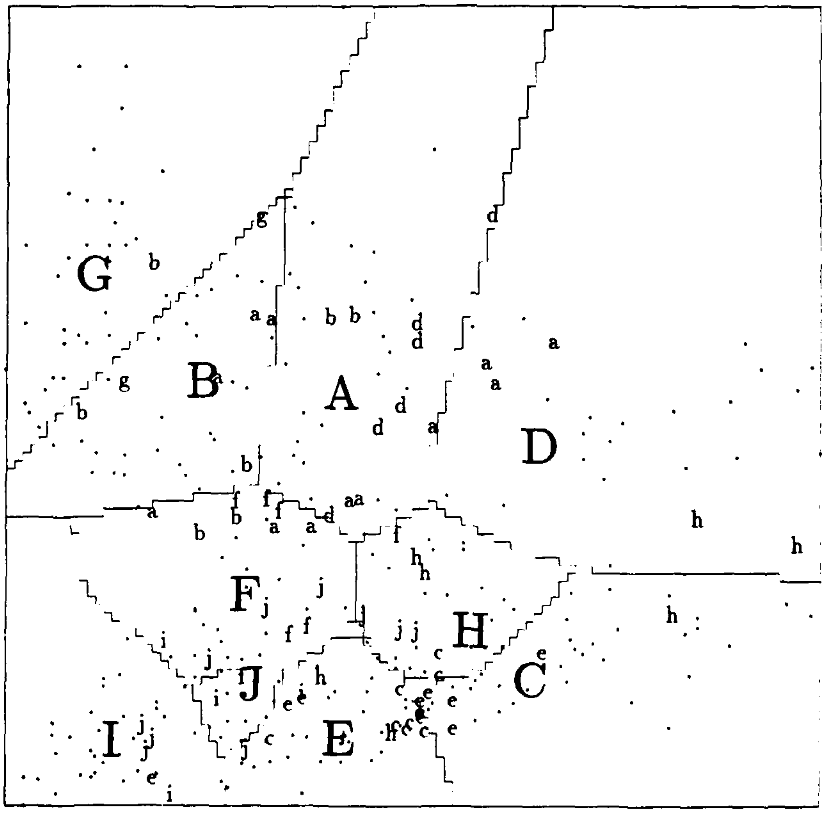

Figure 5: The location of the decision surface for the vowel formant MLP network after initialisation to a NCM classifier.

In figure 6 the decision regions and incorrectly classified data points for the trained MLP are shown. Note how the decision regions have distorted to capture the corresponding training data points (the region for class "h" which extends to the right being a good example): the number of misclassified training data points has been reduced.

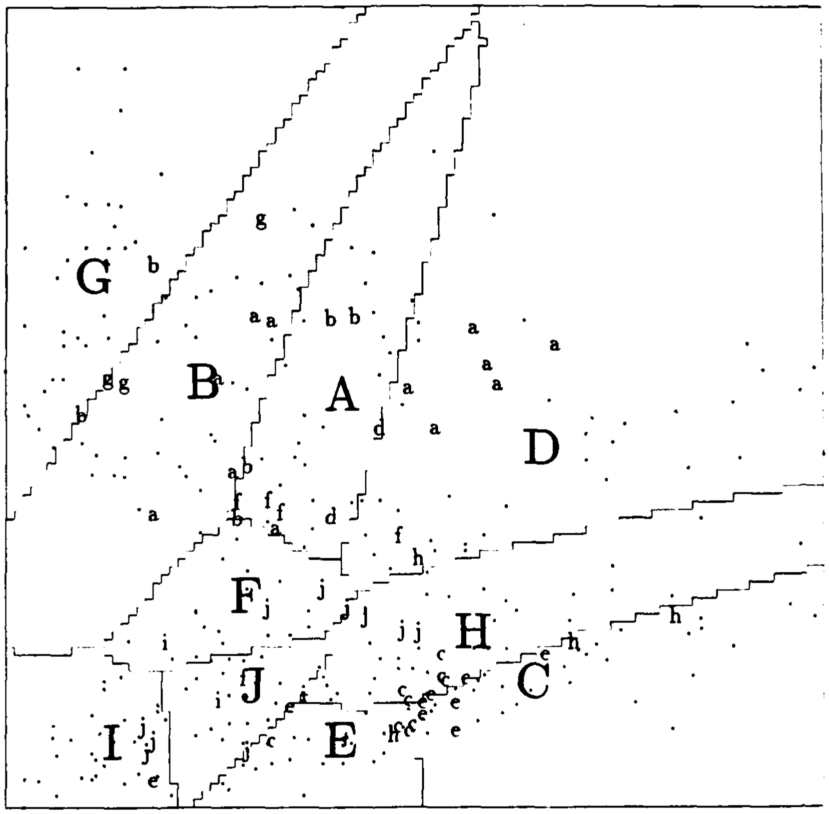

Figure 6: The location of the decision surface for the same vowel formant MLP network as the previous figure after training.

	training set error	test set error	training set performance	test set performance	training time
54-22-11	0.712390	0.824429	77.92	66.23	3
54-22-11	0.000001	0.702706	100.00	78.57	340

Table 8: The performance of a pruned MLP network set up using nearest class mean after training on the vowel spectrum data.

	training set error	test set error	training set performance	test set performance	training time
760-13-8	0.625548	0.691672	91.88	86.25	40
760-13-8	0.000000	0.244520	100.00	97.50	1000

Table 9: The performance of a pruned MLP network set up using nearest class mean after further training on the isolated word data.

	training set error	test set error	training set performance	test set performance	training time
10-8-5	0.905779	0.911956	52.43	50.57	1
10-8-5	0.724895	0.795895	70.79	64.15	840

Table 10: The performance of a pruned MLP network set up using nearest class mean after further training on the radar data.

9 Mahalanobis distance to class mean MLP

In the above method the decision boundary was constructed by the seperating hyperplanes implemented by the logistic, scalar product hidden units. The decision boundary was explicitly modelled. The complementary technique is to explicitly model the distributions and to implicitly model the decision boundary.

The MLP is no longer a homogeneous network of scalar product logistic units. The first layer of hidden units is linear with scalar product fan-in

$$O_{pj} = I_{pj}, \tag{14}$$

where

$$I_{pj} = \sum_i O_{pi} w_{ij}. \tag{15}$$

The second hidden layer consists of gaussian units with a radial fan-in

$$O_{pj} = \epsilon^{-I_{pj}}, \tag{16}$$

and

$$I_{pj} = \sum_i (O_{pi} - w_{ij})^2. \tag{17}$$

The output layer is of standard logistic units

$$O_{pj} = \frac{1}{1 + \epsilon^{-I_{pj}}}, \tag{18}$$

with

$$I_{pj} = \sum_i O_{pi} w_{ij}. \tag{19}$$

The first layer of hidden units constitute C linear transformations of the input vector (one transformation for each class) into a vector of the same rank. The transformation to the yth set of hidden units is

$$\underline{W}_y, \text{ where } \underline{W}_y \underline{W}_y^T = \Sigma_y \tag{20}$$

Σ_y is the covariance matrix of the distribution of the yth class. The set of transformations $\underline{W}_y$ transform the distributions of class y into a isotropic distributions with equal variance. The scaling factor which decides the absolute value of this variance was manually adjusted - up to five values were tried for each problem (the time required was included in the training time).

The weights in the next layer are the reference points for the inverse exponential (Gaussian) hidden units. The weights are initialised to the transformed means of the class distributions

$$\underline{w'_y}_y = \underline{W'}_y \underline{\mu}_y. \tag{21}$$

The final layer of weights sharpens up the output of the radial units by using a logistic nonlinearity. The weights are initialised to ξ for weights corresponding to hidden units and output units associated with the same class and zero for other connections. The bias on each output unit is $-\xi/2$ (i.e. a single input AND gate).

10 Mahalanobis distance to class mean MLP results

There was insufficient data to train the vowel spectrum task and the isolated word recognition task using this technique as the method requires an estimate of the covariance matrices for the distributions of each class.

After the network was initialised the values of the weights in the first two layers (corresponding to the means and covariances of the class distributions) were adjusted using the conjugate gradients method.

After convergence all the weights in the network were optimized (this might cause the test set performance to fall as a consequence of overfitting of the training data).

	training set error	test set error	training set performance	test set performance	training time
2-20-10-10	0.721804	0.719310	78.40	80.18	1
2-20-10-10	0.585536	0.604757	79.88	79.28	4
2-20-10-10	0.537536	0.608371	83.43	77.78	50

Table 11: The performance of a MLP network set up using a Mahalanobis distance to class mean classifier after further training on the vowel formant data.

	training set error	test set error	training set performance	test set performance	training time
10-50-5-5	0.945202	1.010487	59.92	58.90	2
10-50-5-5	0.718861	0.824044	71.03	62.60	100
10-50-5-5	0.650472	0.816935	77.13	62.68	710

Table 12: The performance of a MLP network set up using a Mahalanobis distance to class mean classifier after further training on the radar data.

11 A note on the arbitrary constants

It will have been noted that there are a number of problem specific parameters in the above techniques. This is normally an unsatisfactory situation which often results in wasted time while suitable values are found. In this case the inclusion of these parameters is not quite as bad as at first sight - there are two reasons for this:

1. The actual values do not seem to have a dramatic effect on the final results, the techniques still work with the parameters set up to an order of magnitude away from the optimum value. The values for the runs above were found with only a few trials (the time required for this has been included in the total training time).
2. The amount of time needed to set up the network and test it is minimal, a large number of trial parameter values can be evaluated in just a few minutes.

12 Summary

Four test problems were analysed using standard pattern classification techniques aswell as multi-layer perceptron networks. Nearest neighbour classifiers were found not to give significantly worse performance than MLPs when used on the test problems presented here. MLPs do have advantages over nearest neighbour classifiers in the recognition phase as they are, perhaps, more suitable for fast and effecient parallel hardware implementations.

One of the least desirable aspects of MLPs, namely the slow and unreliable training phase, was partially replaced by a technique for implementing *one of two* standard classifiers as a MLP network. This original method of incorporating prior knowledge into a multi-layer perceptron is used to set up the network topology, the type of processing unit and the initial values for the weights.

The performance of the best of the 'standard' classifiers (KNN and MDCM), the best 'standard' MLP and the MLPs trained from Euclidean distance to class mean (MLP/NCM) and Mahalanobis distance to class mean (MLP/MDCM) is outlined below.

Dataset	KNN	MDCM	MLP	MLP/NCM	MLP/MDCM
Vowel formant (train)	82.25	78.40	84.91	87.87	79.88
Vowel formant (test)	81.08	80.18	78.98	79.28	79.28
Vowel spectrum (train)	90.26	n/a	83.12	100.00	n/a
Vowel spectrum (test)	74.03	n/a	70.78	78.57	n/a
Isolated word (train)	97.50	n/a	100.00	100.00	n/a
Isolated word (test)	93.75	n/a	98.13	97.50	n/a
Radar (train)	81.62	58.90	63.13	70.79	77.13
Radar (test)	56.59	59.92	60.55	64.15	62.68

Table 13: The performance of the various techniques on the four test problems.

As can be seen from the table, the final performance of initialised networks is better than the best of the standard MLPs except in the case of the isolated word experiment (where the difference is probably not significant). Training times for all but one of the runs were reduced by about a half (the isolated word MLP set up as NCM actually took 3 times longer). It should be remembered that the results for the initialised networks are for the **first** network to be trained; the standard MLP results are the best of a number of runs.

13 Conclusions

The results of using the techniques for initialising MLP networks on the four test problems were very encouraging; both an increase in performance and a considerable reduction in **total** training time was observed in most cases.

The techniques clearly will not work when the class distributions are unsuitable for class mean classifiers (annular distributions being an extreme case). However, for those problems for which simple classifiers exhibit reasonable performance, valuable gains are available.

Although more experimental results with a wider range of problems and a statistical analysis of the significance of the results is clearly required; the relationship between MLPs and other pattern classifiers is shown to be worthy of further work, both theoretical and experimental.

An extension of the methods to include classifiers which use many prototype points per class (nearest neighbour techniques) is possible. The size of the networks are likely to be very large unless some preprocessing of the prototype points is undertaken (using some clustering algorithm).

These techniques now form a valuable addition to our collection of pattern processing algorithms and has been used successfully on a few applications.

References

[1] M. D. Bedworth and D. Lowe, "Fault Tolerance in Multi-Layer Perceptrons: A Preliminary Study", SP4 Research Note No. 59 (1988) (Royal Signals and Radar Establishment, Malvern).

[2] M. D. Bedworth, "Multi-Layer Perceptrons as Adaptive Nearest Neighbour Classifiers", SP4 Research Note 70 (1988) (Royal Signals and Radar Establishment, Malvern).

[3] M. D. Bedworth and J. S. Bridle, "Using Error Back Propagation: Some Alternatives to Logistic Networks", SP4 Research Note 75 (1988) (Royal Signals and Radar Establishment, Malvern).

[4] M. D. Bedworth, J. S. Bridle and D. Lowe, "A Comparison of Various Classifiers on a Radar Problem", SP4 Research Note 76 (1988) (Royal Signals and Radar Establishment, Malvern).

[5] D. G. Bounds and M. D. Bedworth, "Structured Training of Multi-Layer Perceptrons: NETspeak Trained on Children's Reading Books", RIPRREP/1000/41/89 (1989) (National Electronics Research Initiative in Pattern Recognition, RSRE, Malvern).

[6] H. Bourlard and C. J. Wellekens, "Multilayer Perceptrons and Automatic Speech Recognition", Proc. of IEEE 1st International Conference on Neural Networks (1987) (Philips Research Laboratory, Brussels).

[7] P. A. DeVijver and J. Kittler, "Pattern Recognition: A Statistical Approach", Prentice/Hall International Press (1982).

[8] J. N. Holmes, "The JSRU channel vocoder", IEE PROC., 127, 53-60, (1980).

[9] N. A. McCulloch, M. D. Bedworth and J. S. Bridle, "NETspeak: A Multi-Layer Perceptron that can Read Aloud", RIPRREP/1000/4/87 (1987) (National Electronics Research Initiative in Pattern Recognition, RSRE, Malvern).

[10] N. A. McCulloch and W. Ainsworth, "Speaker Independent Vowel Recognition Using a Multi-Layer Perceptron", Proc. Speech '88 7th FASE Symposium, p.851-859 (1988) (National Electronics Research Initiative in Pattern Recognition, Malvern).

[11] S. M. Peeling, R. K. Moore and M. J. Tomlinson, "The Multi-layer Perceptron as a Tool for Speech Pattern Processing Research", Proc. IoA Autumn Conf on Speech and Hearing (1986) (Royal Signals and Radar Establishment, Malvern).

[12] G. E. Peterson and H. L. Barney, "Control Methods used in the Study of Vowels", Journal of the Acoustical Society of America, Vol.24 No.2, pp.175-184 (1952) (Bell Telephone Laboratories, Inc., Murray Hill, New Jersey).

[13] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation", ICS Report 8506 (1985) (Institute for Cognitive Science, University of California, San Diego).

[14] A. R. Webb, David Lowe and M. D. Bedworth, "A Comparison of Nonlinear Optimisation Strategies for Feed-Forward Adaptive Layered Networks", RSRE Memo 4157 (1988) (Royal Signals and Radar Establishment, Malvern).

DOCUMENT CONTROL SHEET

Overall security classification of sheet .... UNCLASSIFIED ....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification, eg (R), (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference Memo 4346	3. Agency Reference	4. Report Security Classification UNCLASSIFIED

5. Originator's Code (if known) 7784000	6. Originator (Corporate Author) Name and Location ROYAL SIGNALS & RADAR ESTABLISHMENT ST ANDREWS ROAD, GREAT MALVERN WORCESTERSHIRE WR14 3PS
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title

IMPROVING UPON STANDARD PATTERN CLASSIFICATION ALGORITHMS BY IMPLEMENTING THEM AS MULTI-LAYER PERCEPTRONS

7a. Title in Foreign Language (in the case of Translations)

7b. Presented at (for Conference Papers): Title, Place and Date of Conference

8. Author 1: Surname, Initials BEDWORTH M D	9a. Author 2	9b. Authors 3, 4 ...	.0 Date 1989.12	pp. ref. 23

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution Statement

UNLIMITED

Descriptors (or Keywords)

Continue on separate piece of paper

Abstract

The multi-layer perceptron (MLP) is a type of adaptive layered network often used as a pattern clasifier. In more recent literature, MLPs are compared with simpler classification techniques using common datasets. We select two of these simple static pattern classification algorithms and briefly review the relevant techniques. After introducing a modest set of evaluation databases, the performance of the standard classifiers and MLPs are assessed. A technique for implementing the two standard classifiers as MLPs is presented and this novel approach is used to automatically design a 'good' set of initial weights for the MLP networks. Encouraging experimental results for these hybrid techniques are shown for illustration.