Efficient training and improved performance of multilayer perceptron in pattern classification

B.B. Chaudhuri*, U. Bhattacharya

Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203, B.T. Road, Calcutta-700 035, India

Received 17 August 1998; accepted 11 April 2000

Abstract

In pattern recognition problems, the convergence of backpropagation training algorithm of a multilayer perceptron is slow if the concerned classes have complex decision boundary. To improve the performance, we propose a technique, which at first cleverly picks up samples near the decision boundary without actually knowing the position of decision boundary. To choose the training samples, a larger set of data with known class label is considered. For each datum, its k-neighbours are found. If the datum is near the decision boundary, then all of these k-neighbours would not come from the same class. A training set, generated using this idea, results in quick and better convergence of the training algorithm. To get more symmetric neighbours, the nearest centroid neighbourhood (Chaudhuri, Pattern Recognition Lett. 17 (1996) 11–17) is used. The performance of the technique has been tested on synthetic data as well as speech vowel data in two Indian languages.

Keywords: Backpropagation algorithm; Multilayer perceptron; Faster training; Pattern classification

1. Introduction

A serious drawback of backpropagation algorithm (BPA) [12] for the training of conventional multilayer perceptron (MLP) is its slow rate of convergence. Several modifications [1,2,7–9,11,13] of the original BPA have already been suggested either

to improve the convergence or to improve the performance over the original algorithm. In a number of such modifications [1,8,11,13], search techniques other than the conventional gradient descent method are proposed. Another set of modifications [2,7,9] has considered dynamic adaptation of learning parameters based on certain heuristics. Improvement in training performance for some specific applications is reported for each of these modified algorithms.

However, none of the modifications is capable of delivering satisfactory performance for all problems, in general. Moreover, such modified BP algorithms usually involve more computations per iteration and also need a priori choices of certain additional parameters. In particular, the conjugate gradient training algorithm usually converges to the nearest minimum significantly faster than conventional BPA, but still can be slow and also there is no guarantee that the achieved minimum is global [10]. Thus, the search for an approach to speed-up its convergence and/or for the improvement of general performance of the trained network still remains important. In this paper, we shall demonstrate that an intelligent selection of training patterns can considerably improve the speed and efficiency of the training of MLP network. Although we consider only the conventional BPA for the simulation results, the proposed approach is, in principle, applicable to other training algorithms as well.

Our idea is to analyze the available set of prototypes and, chose those for network training which are near the decision boundaries of the pattern classes. It will be shown that patterns near the decision boundary can be cleverly picked even without knowing the actual decision boundary or the class conditional density function. It will also be shown that these patterns can lead to quick and better convergence of network training. Moreover, the extra computational load for the selection of training patterns becomes insignificant because the training set so obtained is much smaller in size than the set of randomly selected training samples.

A similar approach is found in somewhat different context namely, statistical learning theory. Here it is observed that the patterns closest to the decision boundary are hard-to-learn and these are the patterns, which determine the optimal separating hyperplane [16]. The training patterns, which lie on the margin, at some distance from the optimal hyperplane, are critical and these are called support vectors. However, in practice, separating hyperplane may not exist, if there are large overlaps among the classes. In such situations, patterns lying on the margins provide a set of candidate support vectors, which are used for the initialization of the support vector machine learning algorithm [14,17]. On the other hand, patterns, which are away from the margin, do not have significant role in the optimization of the classifier and hence they are not included in the training set. Thus, a relatively small number of support vectors result in better generalization ability with less computational load.

Somewhat related but different approaches of exploiting training patterns in the context of MLP training are also reported in the literature. Drago and Ridella [5] as well as Weymaere and Martens [18] proposed a method for optimum weight initialization of an MLP network. In [5] a scale factor is determined on the basis of the network response to the whole training set which is then used to obtain the initial scaled random weights. The method in [18] consists of a clustering of training data

followed by a network construction algorithm. The cluster centres are used to obtain the initial weights. Both approaches employ all training patterns, while our approach is to choose only a subset near the pattern class boundary.

Strand and Jones [15] developed a procedure in which the network is initially partially trained by some of the patterns. Those patterns that are handled well by the partially trained network are put in a queue, while the troublesome patterns are used for further learning. To prevent the network from forgetting what it has already learned, cases are retrieved periodically from the queue. On the other hand, Davis and Hwang [4] first train the network using the complete training set. This trained network identifies the patterns near the border and these are used for further training. These two approaches try to exploit the importance of some patterns over others like us, but they chose them through MLP training while we chose them before training and in a different way. Moreover, since we work on a smaller subset of patterns, the training is speeded up.

The proposed approach has been described in Section 2. Results of applying this approach to both synthetic and real-life data has been described in Section 3. The synthetic data sets were prepared from normal distributions while the real-life data set consists of formant frequencies of vowel speech sounds in two Indian languages. It is assumed that the probability distribution of the multivariate data is unknown to the classifier.

2. Proposed approach

In the conventional backpropagation method [12], the components of connection weight vector of an MLP are modified during learning in a direction opposite to the direction of the gradient of the mean-square error surface. This weight modification involves two learning parameters, called the learning rate and the momentum factor. The rate of convergence of the training largely depends on the choices of these learning parameters. However, dynamic adaptation of the learning rate [2,7] often improves the convergence performance.

During training of MLP network, it is important that the training patterns are selected so that the connection weights move over time to optimize the network response to all the pattern classes. The efficiency of such a network can be improved through careful preprocessing and/or proper selection of the training inputs. Estimates of the optimal number of input patterns needed for the best training performance depends on the problem. Intuitively, it seems that the larger and more complex the boundary of a pattern class is, the more examples will be required to train the network.

An aspect of designing a good training set is its size. Usually, a smaller training set takes less time for the convergence but too small a training set may cause overfitting which affects the performance. As a rule of thumb, the minimum number of training patterns per class may be computed as twice the number of connection weights of the MLP. Of course, training set having size twice the minimum one (i.e., four times the number of connection weights of the MLP) is a better choice.

Another equally important issue is the distribution of training patterns in the input space. It can be understood that a good training set should satisfy the following criteria.

1. The separating hypersurface between each pair of classes should be properly reflected in the training set.
2. If any of the classes has more than one connected component, then criteria 1 should be satisfied for each of these clusters.

The above observations come from an idea about how MLP works for pattern classification. During training phase, the weights learn how to minimize the misclassification of the training set. Since minimization of misclassification needs accurate knowledge of class boundary (for two classes C_1 and C_2, the boundary denotes the hypersurface where the distribution of C_1 equals that of C_2), the MLP essentially tries to learn the decision boundary during training phase. It is, therefore, natural that the patterns near the boundary are more important than patterns near the core (e.g. centroid) of the class. In Section 3 we shall demonstrate that the experimental results support this view.

If the pattern classes are of convex shape in the feature space, conventional BPA may lead to quick and accurate training, and a special choice of training patterns is not necessary. But for overlapped and meshed classes with non-convex boundary, which is a more practical and general situation, it is better to look for the above criteria.

It is easy to know the decision boundary provided discriminant function for each class or the class conditional probability distributions are known in parametric form. In that case, however, there is no need of using MLP tools since simple discriminant analysis (in case of known discriminant function) or Bayes' classifier (in case of known class conditional distribution) [6] can optimally classify the pattern. Thus, MLP approach is useful when the class boundary information is not precisely known. In such a situation our problem lies in developing a way of choosing patterns near the boundary without knowing the boundary precisely.

For this purpose we consider k neighbours of each pattern in the training set. For the convenience of explanation, let there be two classes C_1 and C_2. Let P be a pattern from class C_1. If all the k neighbours of P also belong to C_1, then it is highly likely that P lies in the interior of class C_1 and not near its boundary with C_2. The situation is illustrated in Fig. 1 for $k = 6$. On the other hand, for a pattern like Q which is near the boundary, the number of neighbours for classes C_1 and C_2 are nearly equal. In other words, for a pattern near to the boundary between two classes, its neighbours come from the neighbouring classes, not from only one class. This is the basic idea of choosing training patterns near the class boundaries without knowing the actual boundary positions. Let us now describe the idea more explicitly. If out of k neighbours of a pattern P belonging to class C_i, k_i neighbours belong to class C_i where $k_1 \leq k_i \leq k_2$ then P is chosen as a pattern near the boundary of C_i. The value of k_2 determines how far inside the class C_i we shall go to pick the patterns (to be considered as patterns near the boundary), while k_1 is used to reject stray patterns from C_i which are embedded in the patterns of neighbouring class. We may choose

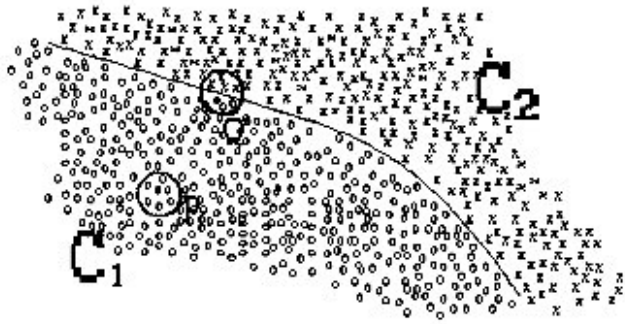

Fig. 1. Selection of training patterns.

these two values so that $k_1 < k/2$ and $k_2 > k/2$. For example, in the case when $k = 6$, we may take $k_1 = 2\ (< 3)$ and $k_2 = 5\ (> 3)$. Of course, instead of constant values, the choice of these two quantities may be data dependent. For example, if the data in class C_1 is much dense than in class C_2 then for a datum of class C_2 near the boundary, out of k neighbours, it is likely to get more neighbours from classes C_1 than from C_2. Conversely, for a datum of class C_1 near the boundary, it is likely to get more neighbours from class C_1 than from C_2. This lack of balance will affect the selection of data if k_1 and k_2 are the same for both the cases. Instead, we can make k_1 and k_2 dependent on the local densities of the classes which is reflected by the average distance of the neighbours belonging to a particular class.

The choice of k should depend on the dimension of the pattern space. If d is the dimension then $k = 2^d + 2$ is a good choice. Thus, in 2-D we have $k = 6$. This is the number of Voronoi neighbours a point can have on an average, in 2-D space if the point pattern is generated by Poisson process.

There are several ways of defining the neighbourhood of a point. The most popular one is the nearest neighbourhood, which takes the first k-ranked points according to Euclidean distance as neighbours. More symmetrical neighbourhood is the Voronoi neighbourhood but in a high dimensional ($d > 3$) space, it is computationally very inefficient. Recently, Chaudhuri [3] proposed a computationally efficient symmetrical neighbourhood, called nearest centroid neighbourhood (NCN), which may be used here.

The nearest centroid neighbours can be computed in an iterative way as follows. Let P be a point whose k neighbours should be found in a set of points S_0. These k neighbours are such that (a) they are as near to P as possible and (b) their mean or centroid is also as near to P as possible. For the candidate P, its first nearest centroid neighbour is its nearest neighbour. In Fig. 2, N_1 is the first nearest centroid neighbour of P. Given $k-1$ nearest centroid neighbours, the centroid of these $k-1$ neighbours and each other data in the pattern space is computed and its distance from P is measured. The kth neighbour is the datum for which the centroid is nearest to P. In the figure, Q is the second nearest neighbour of P, but it is not the second nearest centroid neighbour. Rather, the second NCN should be a point in the opposite direction (and with equal distance) of the first neighbour N_1 with respect to P so that

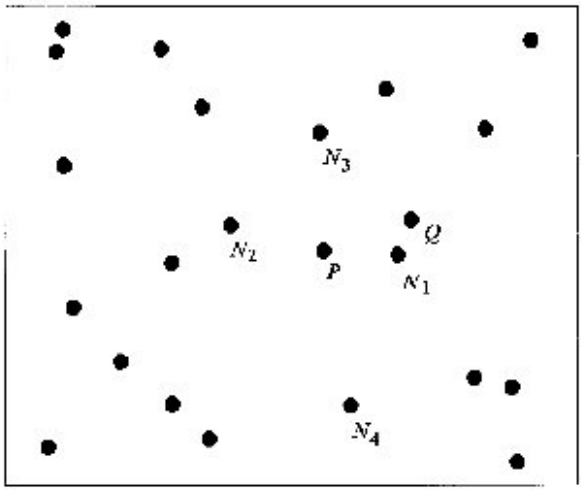

Fig. 2. An example of nearest centroid neighbours (the NC neighbours are denoted as N_i).

the centroid is minimally away from P. Here, N_2 is the second NCN of P. Because of the centroid criterion, the chosen neighbouring points lie all around P. Also, P being nearest to the centroid, acts as the unbiased representative of its neighbours. This kind of neighbourhood selection is free from user-specified parameters and also the computational complexity is of the same order as that of the k-nearest neighbours.

The basic steps of the NCN algorithm are given below.

Step 1: (Initialization) $S \Leftarrow S_0 - P$; $T = \phi$; $j = 0$. Find the nearest neighbour (in terms of the Euclidean distance) of P in S. Let it be R (resolve the tie arbitrarily). Make $T \Leftarrow R$; $S \Leftarrow S - R$.

Step 2: $j \Leftarrow j + 1$; For each point $Q \in S$ find the centroid M of points in $T \cup Q$. Choose the point as Q_0 for which M is nearest to P in terms of Euclidean distance. In case of a tie choose as Q_0 the point that is farthest from the neighbour chosen in the previous iteration.

Step 3: Make $T \Leftarrow T \cup Q_0$; $S \Leftarrow S - Q_0$;

If $j = k$ return. [T is the subset of k NC neighbours of P.] Else, go to step 2.

In this way we choose the training patterns from approximately the boundary region between classes. Note that this approach does not need any a priori parametric knowledge of data distribution or exact decision boundary and is computationally efficient for data in high dimensional space.

Now let us recall the overall procedure in an algorithmic way:

Step 1: Start with a sufficiently large number of training samples from each class. Call this as the initial training set (T_I).

Compute $k = 2^d + 2$; Select k_1 and k_2 such that $k_1 < k/2$ and $k_2 > k/2$.

Step 2: Obtain k nearest centroid neighbours (NCN) for each member in the set T_I. (The NCN algorithm is given before.)

Step 3: Analyze the class-membership of all the k members of NCN of a sample (say, s) in T_I. Let s belong to class C_i and let k_i be the number of members from C_i in the set of k NCNs. If $k_1 \leq k_i \leq k_2$, then s is selected in the final training set T_F. Otherwise, s is excluded from the final training set.

Repeat the step for all samples in T_I.

Step 4: The final training set, T_F obtained at the end of Step 3, is a small subset of T_I. Use T_F for training of the MLP by conventional backpropagation algorithm.

To see the effectiveness of our approach we consider classification problems of data generated using normal distributions. We shall also consider the problem of vowel speech signal recognition in two Indian languages, namely Telugu and Bengali.

3. Experimental results

As stated before, initially we consider several synthetic data sets. Since these data sets are generated using a class of parametric distribution, it is convenient to change the class distributions in a controlled way and judge the effect of the change systematically. Later on, we have considered the realistic problem of vowel data classification.

Our synthetic data consists of randomly generated points in 2-D and 5-D with normal distribution. That the data is generated from a parametric distribution is not transparent to the BPA of MLP. For simplicity, we have considered only two-class and three-class problems.

Two-class Problem on simulated data: In this case, each class consists of two subclasses. Let the two classes be denoted by C_1 and C_2. Then $C_1 = \{x: x \in N(\mu_{11}, \sigma_{11}) \cup N(\mu_{12}, \sigma_{12})\}$ and $C_2 = \{x: x \in N(\mu_{21}, \sigma_{21}) \cup N(\mu_{22}, \sigma_{22})\}$ where, $\mu_{11} = (30,30)$, $\sigma_{11} = (5,5)$; $\mu_{12} = (50,50)$, $\sigma_{12} = (7,7)$ and $\mu_{21} = (50,30)$, $\sigma_{21} = (7,5)$; $\mu_{22} = (30,50)$, $\sigma_{22} = (5,7)$. It is easy to verify that there is considerable overlap between the two classes C_1 and C_2. Mixture of two normal distributions for each class is supposed to make the decision boundary more complex. The forms of decision regions, as obtained by Bayesian approach, are shown in Fig. 3. The architecture of the MLP, used for this problem consists of two input nodes, five hidden nodes and two output nodes. The values for the learning parameters are chosen as $\eta = 1.5$ and $\alpha = 0.7$. We will consider three different training sets and their constructions are described in the following. The first of these training sets consists of 250 random samples from each of the four subclasses (two subclasses from each class). From this first set, two other training sets, consisting of samples from near the centre and samples from near the boundary, are constructed. In the first of these two sets, 25 patterns from each subclass are selected such that these are the nearest from the respective means. In the other set, 25 patterns from each subclass which lie near the boundaries of the respective classes, as obtained by the proposed technique (with $k_1 = 2$, $k_2 = 4$), are selected. Usually, the number of patterns, belonging to different subclasses, are different in the last training set but we

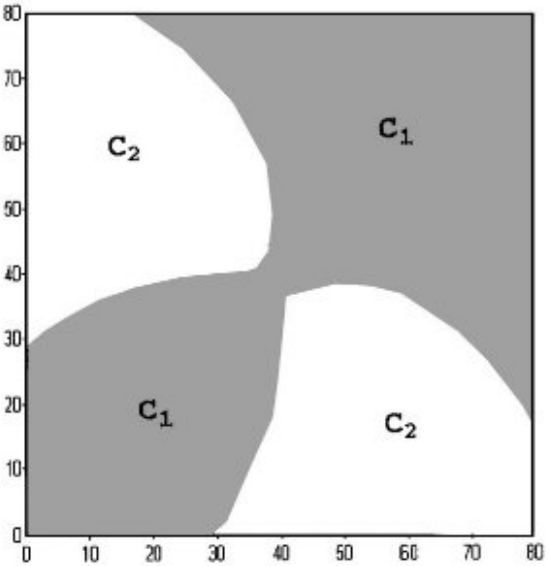

Fig. 3. Two-class problem: decision regions are obtained by Bayesian approach.

Table 1
Simulation results on two-class problem

Approach for selection of training samples	Training epochs (#)	System error		Misclassification (%)	
		Training set	Test set	Training	Test[a]
Random	5000	0.154528	0.097862	13.5	17.75 (0.0085)
Around mean values	149	0.000828	0.216255	0.00	25.80 (0.0098)
Proposed approach	158	0.000778	0.092648	0.00	9.95 (0.0067)
Bayes Error	—	—	—	—	9.27 (0.0065)

[a]Values in parenthesis indicate the standard error of the reported result.

randomly select only 25 such patterns from each subclass. The training of the network is continued till the system error on the training set becomes less than 0.001 or it completes a maximum number (5000) of training sweeps, whichever occurs first.

After each training set trains the MLP, a test set consisting of 500 random patterns from each of the four subclasses is presented to the network. The test patterns are subject to classification and the average misclassification error is computed. The results are shown in Table 1 where the errors during training are also presented.

Three-class problem on simulated data: In this case, three classes have been considered such that each pair of two classes has considerable overlap. Let these classes be denoted by C_1, C_2 and C_3. Then $C_1 = \{x: x \in N(\mu_1, \sigma_1)\}$, $C_2 = \{x: x \in N(\mu_2, \sigma_2)\}$

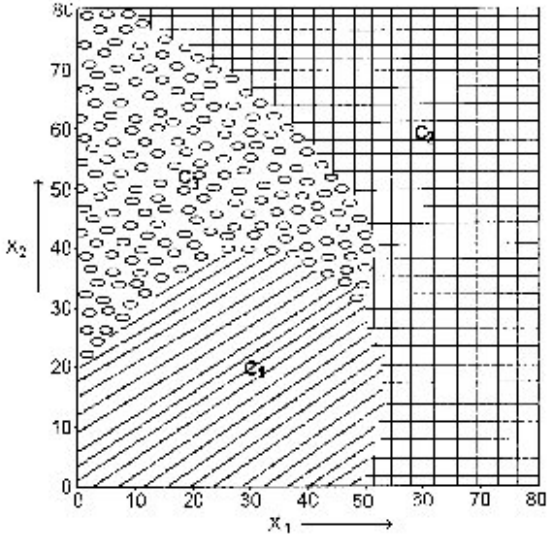

Fig. 4. Three-class problem: decision regions are obtained by Bayesian approach.

and $C_3 = \{x: x \in N(\mu_3, \sigma_3)\}$. Moreover, we choose $\mu_1 = (30,30)$, $\sigma_1 = (5,5)$; $\mu_2 = (60,60)$, $\sigma_2 = (7,9)$ and $\mu_3 = (30,45)$, $\sigma_3 = (9,5)$. The forms of decision regions, as obtained by Bayesian approach, are shown in Fig. 4. The architecture of MLP chosen for this classification task consists of two input nodes, 10 hidden nodes and three output nodes. The values for the learning parameters are chosen as $\eta = 0.9$ and $\alpha = 0.7$. In this case, the training sets are formed in the following way.

As in the previous case of two-class problem, three training sets are formed to compare the training performance. The first one consists of 300 random samples from each of the three classes. The second set is a small subset of the first one and this consists of 30 random samples from each of the three classes, which are nearest to the respective means. The third training set, which is another small subset of the first one, is constructed as follows. All the three possible pairs of classes are considered. For each of these pairs, samples from the first training set which lie near the respective boundaries are determined using the proposed technique (with $k_1 = 2$, $k_2 = 4$) and 15 such samples are selected randomly in the third training set corresponding to each member class of the pair. Thus, the third training set consists of 30 samples from each of the three classes and they are selected according to the proposed technique.

The training of the network has been continued till the system error on the training set becomes less than 0.001 or it completes a maximum number (5000) of training sweeps, whichever occurs first. Next, a test set of 1500 samples, taking 500 random patterns from each of the three classes, is presented to the trained network for classification, as in the two-class problem. The results are summarized in Table 2.

Higher-dimensional problem on simulated data: In the above, we considered problems in two dimensions only. This is because of its ease in pictorial demonstration.

Table 2
Simulation results on three-class problem

Approach for selection of training samples	Training epochs (#)	System error		Misclassification (%)	
		Training set	Test set	Training	Test[a]
Random	5000	0.107318	0.09994	3.89	15.73 (0.0094)
Around mean values	96	0.000965	0.12986	0.00	21.20 (0.0106)
Proposed approach	111	0.000979	0.04321	0.00	8.53 (0.0072)
Bayes Error	—	—	—	—	7.20 (0.0067)

[a]Values in parenthesis indicate the standard error of the reported result.

However, the proposed approach works as well in higher-dimensional problems. Now we will consider a two-class problem in the five-dimensional space, each class consisting of two subclasses. Let the two classes be denoted by C_1 and C_2. Then $C_1 = \{x: x \in N(\mu_{11}, \sigma_{11}) \cup N(\mu_{12}, \sigma_{12})\}$ and $C_2 = \{x: x \in N(\mu_{21}, \sigma_{21}) \cup N(\mu_{22}, \sigma_{22})\}$ where, $\mu_{11} = (40,40,40,40,40)$, $\sigma_{11} = (12,12,12,12,12)$; $\mu_{12} = (50,50,50,50,50)$, $\sigma_{12} = (15,15,15,15,15)$ and $\mu_{21} = (50,40,50,40,50)$, $\sigma_{21} = (15,12,15,12,15)$; $\mu_{22} = (40,50,40,50,40)$, $\sigma_{22} = (12,15,12,15,12)$. Thus, the present problem in five-dimension is a two-class problem but each class is generated from the mixture distributions of two normal populations.

The architecture of the MLP, used for the above problem consists of five input nodes, five hidden nodes and two output nodes. The values for the learning parameters are chosen as $\eta = 0.8$ and $\alpha = 0.5$. Three training sets, as before, are constructed. The first one consists of 250 random samples from each of the four subclasses (two subclasses in each class). The second set is a small subset of the first one. It consists of 25 patterns from each of the four subclasses, such that these belong to the first training set and also are nearest to the respective means. The third training set is constructed from among the samples in the first training set following the proposed technique (with $k_1 = 25$ and $k_2 = 40$). In this third training set only 25 samples from each of the four subclasses are considered.

The training of the network is continued till the system error on the training set becomes less than 0.001 or it completes a maximum number (5000) of training sweeps, whichever occurs first. A test set consisting of 500 random patterns from each of the four subclasses is presented to the trained network for classification. The results of classification and the training statistics are shown in Table 3.

From Tables 1–3, it is noted that the accuracy of classification is improved by the proposed approach. Although the speed of training is comparable with the case when the training samples are collected from around the respective class means, the performance is much improved by the proposed approach. Also, the speed is greatly improved by this approach compared to the conventional approach (where the training patterns are chosen at random).

Vowel recognition problem on real-life speech vowel data: We applied the proposed scheme for the vowel speech signal recognition of two Indian languages namely

Table 3
Simulation results on five-dimensional classification problem

Approach for selection of training samples	Training epochs (#)	System error		Misclassification (%)	
		Training set	Test set	Training	Test[a]
Random	5000	0.005033	0.105446	11.00	34.75 (0.0106)
Around mean values	146	0.000095	0.241688	0.00	42.25 (0.0110)
Proposed approach	178	0.000197	0.092682	0.00	26.12 (0.0098)
Bayes Error	—	—	—	—	25.00 (0.0097)

[a]Values in parenthesis indicate the standard error of the reported result.

Telugu and Bengali. At first, we briefly discuss the procedure for preparation of vowel data.

The recordings of speech data were made inside an empty auditorium with an AKAI tape recorder. The spectrographic analysis has been done on Kay Sonograph Model 7029-A which is a standard audio frequency spectrum analyser. It produces a permanent record of the spectra of any complex waveform, in the range of 5 Hz–16 kHz.

The procedure for preparation of vowel data requires the determination of vowel boundaries and the boundary of the steady-state formant frequencies. The steady state of vowel is that part on the record in which all formants lie parallel to the time axis. The transition is depicted by the inclined formant patterns. The exact point of inflection is difficult to locate in the records but an approximate location can be found by tracing the central line for each formant band. Once these points are located for all available formants, the steady state of the vowel is taken to be the shortest horizontal span for all the formants. It is well known that the first three formant frequencies carry most of the information regarding vowel identification.

The formant frequencies are measured from the base line (i.e., zero-line) of the spectrogram to the central line of formant bands where the formant is in a steady state. The scale used for this measurement is derived from the calibrated tune of 500 Hz recorded on each and every spectrogram. One small division of the scale is equal to 20 Hz. Manual checking of 5% sample obtained by this scale revealed that the formant frequencies have been recorded within an accuracy of 10 Hz. In a few cases, when the vowel hardly reaches a stable state, the congruence of on-glide and off-glide has been taken as the steady state.

Segmentation of Telugu vowels: A number of discrete phonetically balanced speech samples for the Telugu vowels in *consonant–nucleus vowel–consonant* (CNC) form were selected. The CNC combination is taken because the form of consonants connected to a vowel is responsible for influencing the role and quality of vowels. For the present study, 800 Telugu words, uttered by three male speakers in the age group of 25–30 years were recorded for training.

Ten vowels considered in this experiment are /u/,/u:/,/o/,/ɔ/,/a/,/ə/,/e/,/e:/,/i/,/i:/. Table 4 represents the average formant frequencies for first three formants. Here,

Table 4
Average formant frequencies of Telugu vowels

Phonetic symbol	F_1 (Hz)	F_2 (Hz)	F_3 (Hz)
/u/	370	1066	2500
/u:/	348	923	2543
/o/	476	1133	2630
/o:/	486	1000	2540
/ɔ/	606	1473	2420
/a/	710	1240	2400
/e/	517	1796	2633
/e:/	470	1883	2657
/i/	365	2116	2757
/i:/	304	2095	2565

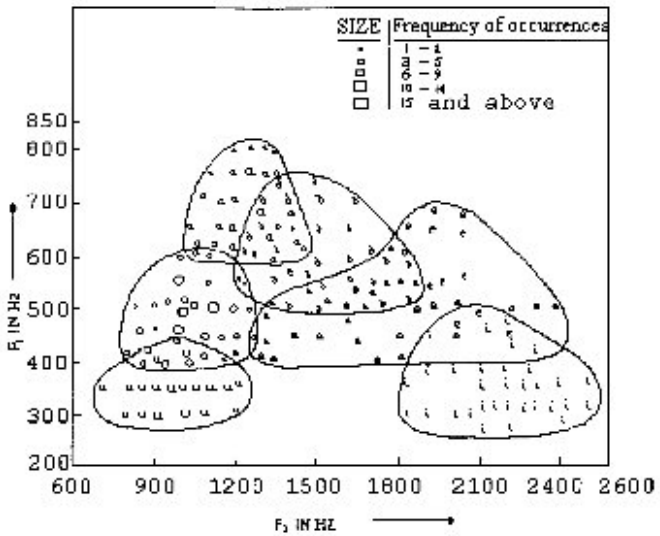

Fig. 5. Telugu vowels in the F_1–F_2 plane.

similarly uttered vowels /u/ − /u:/, /o/ − /o:/, /e/ − /e:/ and /i/ − /i:/ which differ only in duration have been paired together. The first two formant frequencies F_1 and F_2 are plotted for the resulting six different vowels in Fig. 5. The vowel /æ/ is excluded from the present classification task because the number of samples were so small that an effective selection of samples and the subsequent training of MLP could not be undertaken.

The architecture of the MLP, used for this problem consists of three input nodes (corresponding to three formant frequencies), four hidden nodes and six output nodes (corresponding to six classes). The values for the learning parameters are chosen as

Table 5
Training statistics on segmentation of Telugu vowels

Approach for selection of training samples	Training epochs (#)	System error		Misclassification (%)	
		Training set	Test set	Training	Test[a]
Random	1000	0.3263453	0.473593	8.83	42.75 (0.017)
Around mean values	198	0.0384125	0.768453	2.00	49.38 (0.018)
Proposed approach	211	0.0396978	0.205426	2.00	34.75 (0.017)

[a]Values in parenthesis indicate the standard error of the reported results.

$\eta = 0.5$ and $\alpha = 0.3$. Three training sets, as before, are constructed. In the first such set, 100 patterns from each of the six classes are selected randomly. The second set consists of 25 patterns from each of the six classes such that these belong to the first set and are nearest to the respective class mean positions. The third training set is constructed from the first one according to the proposed technique (with $k_1 = 3$ and $k_2 = 7$). In this training set we keep only 25 patterns from each class randomly and the rest patterns are ignored.

The training of the network is continued till the system error on the training set becomes less than 0.05 or a maximum number (1000) of training sweeps is completed, whichever occurs first. A test set, consisting of the samples generated by 15 speakers, is presented to the trained network for classification. The training statistics and results of classification are given in Table 5.

Segmentation of Bengali vowels: For the present study, a sample of 350 commonly spoken Bengali words in CNC form were selected. Ten male and ten female educated and phonetically conscious speaker uttered these words. Of all these records, data for three male speakers have been chosen on the basis of good and clear spectrographs. The age of the speakers varied between 30 and 55 years. Table 6 represents the average formant frequencies for first three formants. The first two formant frequencies F_1 and F_2 are plotted for different vowels in Fig. 6.

The architecture of the MLP chosen for this classification task consists of three input nodes, six hidden nodes and seven output nodes. The values for the learning parameters are chosen as $\eta = 0.5$ and $\alpha = 0.3$. In this case, the training sets are formed in the following way.

Three training sets are constructed as in the previous cases. Training samples have been chosen only from the first speaker. The first such training set consists of 40 samples from each of the seven classes. Both of the second and third training sets consist of 15 samples from each of the seven classes. The test set consists of all the samples available from all the three speakers. The training has been continued till the system error on the training set becomes less than 0.05 or a maximum number (1000) of training sweeps is completed, whichever occurs first. The training statistics and the classification results are given in Table 7.

In these practical data too, effects similar to the simulated data were observed, as evident from the results in Tables 5 and 7. The percentage of misclassification is

Table 6
Average formant frequencies of Bengali vowels

Phonetic symbol	F_1 (Hz)	F_2 (Hz)	F_3 (Hz)
/ u /	327	935	2198
/ o /	438	1015	2308
/ ∂ /	626	1095	2391
/ a /	695	1326	2424
/ æ /	681	1663	2320
/ e /	374	1935	2410
/ i /	304	2095	2565

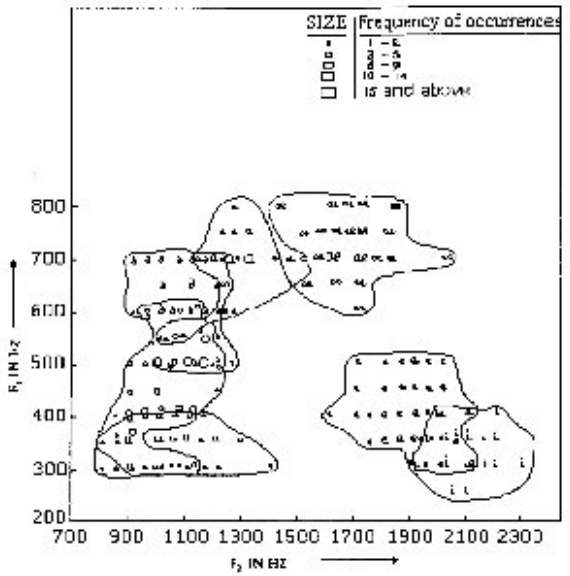

Fig. 6. Bengali vowels in the F_1–F_2 plane.

significantly reduced in the proposed approach. Also, the rate of convergence of training by the proposed approach is much faster than the conventional approach (i.e., when the training patterns chosen at random).

4. Conclusion

To speed-up the rate of convergence of BP algorithm we proposed a method for an intelligent selection of training samples. This method of speeding-up does not involve any modification of the original BP algorithm. This method works well in

Table 7
Training statistics on segmentation of Bengali speech vowel data

Approach for selection of training samples	Training epochs (#)	System error		Misclassification (%)	
		Training set	Test set	Training	Test[a]
Random	1000	0.1967545	0.491435	13.93	40.85 (0.026)
Around mean values	121	0.0399488	0.697582	1.50	43.43 (0.026)
Proposed approach	126	0.0475878	0.153487	0.00	30.29 (0.025)

[a]Values in parenthesis indicate the standard error of the reported results.

complicated cases where the classes are not easily separable by drawing a few hyperplanes. If the classes are overlapping or they are meshed, a training set formed according to the proposed approach can improve the performance (both speed of training, in terms of number of training sweeps, and rate of true classification of the trained network) of an MLP network considerably. The computational load for the selection of the training samples does not reduce the importance of this approach because due to much smaller training set, the computation time required for each sweep is significantly less. The proposed approach has been tested on several problems but due to limited space only four (two on simulated data and two on real-life data) have been presented here. The approach can be used for other training algorithms of MLP network as well.

Acknowledgements

The authors sincerely acknowledge the valuable suggestions of the reviewers, which have helped a great deal to improve the quality of this paper.

References

[1] S. Becker, Y.L. Cun, Improving the convergence of backpropagation learning with second order methods, Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufman, 1988, pp. 29–37.
[2] U. Bhattacharya, S.K. Parui, Self-adaptive learning rates in backpropagation algorithm improve its function approximation performance, Proceedings IEEE International Conference on Neural Networks, Australia, IEEE, San Diego, December 1995, pp. 2784–2788.
[3] B.B. Chaudhuri, A new definition of a neighbourhood of a point in multi-dimensional space, Pattern Recognition Lett. 17 (1996) 11–17.
[4] D.T. Davis, J.N. Hwang, Attentional focus training by boundary region data selection, International Joint Conference on Neural Networks, Baltimore, MD, 1992.
[5] G.P. Drago, S. Ridella, Statistically controlled activation weight initialization (SCWAI), IEEE Trans. Neural Networks 3 (1992) 627–631.
[6] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[7] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (1988) 295–307.
[8] E.M. Johansson, F.U. Dowla, D.M. Goodman, Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method, Int. J. Neural Systems 2 (1992) 291–301.
[9] J.E. Kruschke, J.R. Movellan, Benefits of gain: speeded learning and minimal hidden layers in backpropagation networks, IEEE Trans. Systems Man Cybernet. 21 (1991) 273–280.
[10] T. Masters, Practical Neural Network Recipes in C + +, Academic Press, San Diego, 1993.
[11] D.J. Montana, I. Davis, Training feedforward networks using genetic algorithms, Mach. Learning (1990) 762–767.
[12] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I: Foundations, MIT Press, Cambridge, MA, 1986, pp. 318–362.
[13] S. Singhal, I. Wu, Training multilayer perceptrons with the extended Kalman algorithm, Proceedings ICASSP, Galsgow, Scotland, May 1989.
[14] A.J. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, Introduction to large margin classifiers, in: A.J. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 1999, pp. 1–28.
[15] E.M. Strand, W.T. Jones, An adaptive pattern set strategy for enhancing generalization while improving backpropagation training efficiency, International Joint Conference on Neural Networks, Baltimore, MD, 1992.
[16] V.N. Vapnik, A.Ja. Cherronenkis, Theory of Pattern Recognition, Nauka, Moscow, 1974.
[17] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[18] N. Weymaere, J.P. Martens, On the initialization and optimization of multilayer perceptrons, IEEE Trans. Neural Networks 5 (1994) 738–751.

B.B. Chaudhuri received his B.Sc (Hons), B. Tech and M. Tech degrees from Calcutta University, India in 1969, 1972 and 1974, respectively, and Ph.D. Degree from Indian Institute of Technology, Kanpur in 1980. He joined Indian Statistical Institute in 1978 where he served as the Project Co-ordinator and Head of National Nodal Center for Knowledge Based Computing. Currently, he is the head of Computer Vision and Pattern Recognition Unit of the institute. His research interests include Pattern Recognition, Image Processing, Computer Vision, Natural Language Processing and Digital Document Processing including OCR. He has published about 200 research papers in reputed International Journals and has authored the books entitled "Two Tone Image Processing and Recognition" (Wiley Eastern, 1993) and "Object Oriented Programming: Fundamentals and applications" (Prentice-Hall, 1998). He was awarded Sir J.C. Bose Memorial Award for best engineering science-oriented paper in 1986, M.N. Saha Memorial Award (twice) for best application-oriented papers in 1989 and 1991, the Homi Bhabha Fellowship award in 1992 for OCR of the Indian Languages and computer communication for the blind, Dr. Vikram Sarabhai Research Award in 1995 for his outstanding achievements in the fields of Electronics, Informatics and Telematics and C. Achuta Menon Prize in 1996 for computer-based Indian language processing. He worked as a Leverhulme visiting fellow at Queen's University, U.K. in 1981–1982, a visiting scientist at GSF, Munich and guest faculty at the Technical University of Hannover during 1986–1988 and again in 1990–1991. He is a Senior member of IEEE, member secretary (Indian Section) of International Academy of Sciences, Fellow of International Association of Pattern Recognition, Fellow of National Academy of Sciences (India), Fellow of Institution of Electronics and telecommunication Engineering (India) and Fellow of the Indian National Academy of Engineering. He is serving as associate editor of Pattern Recognition, Pattern Recognition Letters (Elsevier Sciences) and VIVEK as well as guest editor of a special issue of Journal IETE on Fuzzy Systems.

U. Bhattacharya received his B.Sc. (Hons.), M.Sc. and M.Phil. degrees, respectively, in the years 1986, 1989 and 1990 from Calcutta University, India. He obtained his Post-Graduate Diploma in Computer Applications in 1991 from the Regional Computer Center, Calcutta, India. He joined the Indian Statistical Institute, Calcutta in 1991 as a Research Scholar. Currently, he is a programmer in the Computer Vision and Pattern Recognition Unit of the Institute. He was awarded the Young Scientist Award by the Indian Science Congress Association in 1995. He is a member of IEEE. He visited the Institute of Biomathematics and Biometry, GSF, Munich during the year 1998–1999. His research interests include Pattern Recognition, Image Processing and Artificial Neural Networks.