

A Piecewise Linear Network Classifier

Abdul A. Abdurrah, Michael T. Manry, Jiang Li, Sanjeev S. Malalur and Robert G. Gore

Abstract— A piecewise linear network is discussed which classifies N -dimensional input vectors. The network uses a distance measure to assign incoming input vectors to an appropriate cluster. Each cluster has a linear classifier for generating class discriminants. A training algorithm is described for generating the clusters and discriminants. Theorems are given which relate the network's performance to that of nearest neighbor and k -nearest neighbor classifiers. It is shown that the error approaches Bayes error as the number of clusters and patterns per cluster approach infinity.

I. INTRODUCTION

Feedforward neural nets with the universal approximation property [1,2] mimic Bayes discriminants [3,4] and have been successfully used for many classification tasks. However, training time is slow, and convergence of the classification error to Bayes error has not been shown. Such convergence theorems do exist for nearest neighbor classifiers (NNC) and k -nearest neighbor classifiers (k-NNC) [5, 6], which also have the advantage of being easy to design. However, the NNC and k-NNC are rarely used because it is very time-consuming to apply them. Piecewise linear networks have long been used for function approximation and classification [7-11] where speed of operation and simplicity are very important. One approach is to train a multilayer perceptron (MLP) having piecewise linear activations [12, 13]. This approach is useful in hardware implementations, and results in a continuous approximation. However, if we are willing to give up continuous approximation, a simpler piecewise linear network can be devised.

In this paper we develop a discontinuous piecewise linear network classifier (PLNC), based upon the work in [14]. The structure, operation and training of the PLNC are described in section II. In section III the estimation of Bayesian *a-posteriori* probabilities by the PLNC is discussed, along with its relation to the NNC and k-NNC. Simulations and performance comparisons are presented in section IV.

II. PIECEWISE LINEAR NETWORK CLASSIFIER

A. Structure and Operation

As shown in figure 1, the PLNC has input elements in the first layer, the hidden units in the second and outputs in the third. The N -dimensional input vector \mathbf{x}_f has elements $x_f(n)$ with means and standard deviations $\mu(n)$ and $\sigma(n)$ respectively where $1 \leq n \leq N$. First, \mathbf{x}_f is normalized as

$$x_f(n) \leftarrow [x_f(n) - \mu(n)] / \sigma(n) \quad (1)$$

The normalized $(N+1)$ -dimensional vector \mathbf{x} is formed by augmenting \mathbf{x}_f as

$$\mathbf{x} = (\mathbf{x}_f^T : 1)^T \quad (2)$$

The hidden layer consists of K clusters, each having an N -dimensional mean vector \mathbf{m}_c , where $1 \leq c \leq K$. A Euclidean distance measure $d(\cdot)$ is used in clustering. Each cluster also has a weight matrix \mathbf{A}_c of dimension N_c by $(N+1)$, where N_c is the number of classes in the classification problem.

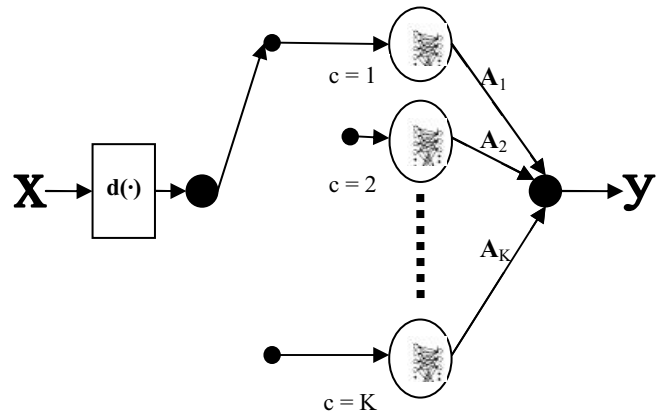


Fig. 1. PLNC Structure

Given an input vector \mathbf{x} , we find cluster index c such that $d(\mathbf{x}, \mathbf{m}_c)$ is minimized. Then we form the output vector \mathbf{y} as

$$\mathbf{y} = \mathbf{A}_c \cdot \mathbf{x} \quad (3)$$

The estimate of the correct class i_c is given by

$$i_c' = \arg \max_i [y_i] \quad (4)$$

where y_i is the i^{th} element of output vector \mathbf{y} and $1 \leq i \leq N_c$.

B. Training

A classification problem typically involves a feature space

Manuscript received January 31, 2007

A. A. Abdurrah, M. T. Manry and S. S. Malalur are with the Department of Electrical Engineering, Image Processing and Neural Networks Lab, University of Texas at Arlington, Arlington, TX 76019-0016 USA (e-mail: abdulaziz.abdurrah@uta.edu, manry@uta.edu, smalalur@uta.edu).

R. G. Gore is with Lockheed Martin, Mission Systems, Fort Worth, TX 76101, USA.

J. Li is with ECE Department and VMASC at Old Dominion University, Norfolk VA 23529, USA (e-mail: JLi@odu.edu).

with numerous feature vectors or samples that have to be assigned various class labels. In supervised learning, the training dataset includes the class label, i_c , of each of the N_v feature vectors. The label is transformed into an N_c -dimensional target vector \mathbf{t} such that

$$t_{pi} = \begin{cases} +b & i = i_c(p) \\ -b & \text{otherwise} \end{cases} \quad (5)$$

where $1 \leq p \leq N_v$, and b is a positive integer.

Before the network is used for classification, it has to be trained, which involves designing the weight matrices \mathbf{A}_c and the cluster mean vectors \mathbf{m}_c . Clustering of the N_v input vectors \mathbf{x}_p is performed using the self-organizing map (SOM) [15].

Elaborating on (3), the elements of the output vector \mathbf{y} , are calculated as

$$y_{pi} = \sum_{n=1}^{N+1} a_{cin} \cdot x_{pn} \quad (6)$$

where a_{cin} denotes an element from the weight matrix \mathbf{A}_c matrix belonging to row i and column n . The mean-squared error (MSE) for the c^{th} cluster is then given by

$$E_c = \frac{1}{N_v(c)} \sum_{q=1}^{N_v(c)} \sum_{i=1}^{N_c} \left[t_{qi} - \sum_{n=1}^{N+1} a_{cin} x_{qn} \right]^2 \quad (7)$$

where $N_v(c)$ is the number of feature vectors in cluster c and t_{qi} is the desired output which could be $+b$ or $-b$. The error gradient with respect to a_{cjm} is given by

$$\frac{\partial E}{\partial a_{cjm}} = -2 \frac{1}{N_v(c)} \sum_{q=1}^{N_v(c)} \left[t_{qj} - \sum_{n=1}^{N+1} a_{cjn} x_{qn} \right] x_{qm} \quad (8)$$

On further simplification we get

$$\frac{\partial E}{\partial a_{cjm}} = -2 \frac{1}{N_v(c)} \left[\left(\sum_{q=1}^{N_v(c)} t_{qj} x_{qm} \right) - \left(\sum_{q=1}^{N_v(c)} \sum_{n=1}^{N+1} a_{cjn} x_{qn} x_{qm} \right) \right] \quad (9)$$

Equation (9) can be written in terms of auto-correlation and cross-correlation elements, r and c respectively, as

$$\frac{\partial E}{\partial a_{cjm}} = -2 \left[c(j, m) - \sum_{n=1}^{N+1} a_{cjn} r(n, m) \right] \quad (10)$$

Equating the gradient in (10) to zero, in order to minimize the error, yields

$$c(j, m) = \sum_{n=1}^{N+1} a_{cjn} r(n, m) \quad (11)$$

The set of equations in (11) is solved using the Output Reset (OR) method [16-18] and a Schmidt based linear equation solver [19-20]. After the training is complete, the PLNC's \mathbf{m}_c vectors and \mathbf{A}_c matrices are fixed and can be used to classify validation data.

III. NEURAL NETWORKS AND THE BAYES OPTIMAL DISCRIMINANT FUNCTION

A. Neural networks and Bayes Discriminant

Over the past several years, neural networks have been used for many tasks, including classification and function approximation. There have been many useful theoretical results concerning their capabilities including the following [4] theorem.

Theorem: When neural net classifiers are trained to minimize the mean-squared error (MSE), the MSE approaches a constant value plus the expected squared error between the neural net output and Bayes discriminant, as the number of training patterns approaches infinity. Specifically,

$$\lim_{N_v \rightarrow \infty} \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{N_c} \left[t_{pi} - y_i(\mathbf{x}_p) \right]^2 = \sum_{i=1}^{N_c} \mathbb{E} \left[(b_i(\mathbf{x}) - y_i(\mathbf{x}))^2 \right] + a \quad (12)$$

where a is a constant, independent of p , t_{pi} is defined in (5), and $y_i(\mathbf{x}_p)$ is the i^{th} output of the network. The Bayes discriminant $b_i(\mathbf{x})$, is the probability that the i^{th} class is correct, given \mathbf{x} , which is written as $P(i|\mathbf{x})$.

The above theorem, however, leaves room for some doubts:

- 1) The neural network's probability of error is not bounded or shown to converge to Bayes probability of error.
- 2) The mean-square error in the theorem treats positive and negative errors the same. However, it is good if the correct class discriminant is larger than desired, but bad if it is smaller [16-18]. This problem leads to sub-optimal networks.
- 3) The theorem makes no use of the neural network's structure. It applies equally well to any discriminant designed by minimizing the MSE.

B. Nearest Neighbor and the PLN Classifiers

In the NNC, an input feature vector is assigned to the class of its nearest training sample, usually determined via the Euclidean distance. The feature space is divided into convex regions or clusters each having a specific class label. Specifically, the distance measure performs a Voronoi tessellation of the N -dimensional input feature space.

Now we consider the relationship between the PLNC and NNC. As K approaches infinity, the convex Voronoi cells in the feature space get smaller in volume and the optimal decision boundaries in each cluster become linear. Hence, each cluster can have its own linear discriminant and overall, a more complex decision boundary is achieved. Therefore, for a given value of K , the PLNC should perform better than the NNC. We begin to address this idea in the following lemmas.

Lemma 1: If a PLNC and NNC have the same distance measure and identical cluster mean vectors, the PLNC has at

least as good a performance as the NNC.

Proof: Since the PLNC's augmented input vector \mathbf{x} includes the constant 1; the PLNC's output vector \mathbf{y} can have a 1 for the same class as picked by the NNC. Other class outputs can be 0.

Suppose that the c^{th} NNC cluster belongs to class i . If we would like the PLNC's c^{th} cluster to always map patterns to class i , the elements of the N_c by $(N+1)$ matrix \mathbf{A}_c' are defined as

$$a_c'(m,n) = \begin{cases} 1 & m = i_c \text{ and } n = N+1 \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

When a PLNC's \mathbf{A}_c matrix initially performs worse than the NNC for a given c^{th} cluster, replace \mathbf{A}_c with \mathbf{A}_c' as defined in (13).

Lemma 2: If a PLNC and a NNC have the same distance measure and identical cluster mean vectors, then as the number of clusters, K and patterns, N_v approach infinity,

$$\mathbf{P}_{\text{eB}} \leq \mathbf{P}_{\text{e(PLNC)}} \leq \mathbf{P}_{\text{e(NNC)}} \leq 2 \cdot \mathbf{P}_{\text{eB}} \quad (14)$$

where $\mathbf{P}_{\text{e(PLNC)}}$, $\mathbf{P}_{\text{e(NNC)}}$ and \mathbf{P}_{eB} respectively denote the PLNC, NNC and Bayes probabilities of error.

Proof: As K approaches infinity, we know that [5, 6]

$$\mathbf{P}_{\text{eB}} \leq \mathbf{P}_{\text{e(NNC)}} \leq 2 \cdot \mathbf{P}_{\text{eB}} \quad (15)$$

Using lemma 1 and (15) yields the result of (14).

C. Convergence of the PLNC Error Probability

We now wish to be more specific and evaluate the average probability of error for the PLNC, as the amount of training data increases. First, recall the following result describing the convergence of the k-NNC probability of error, $\mathbf{P}_{\text{e(k-NNC)}}$. Here k is the number of nearest training vectors or samples to the test sample \mathbf{x} .

Lemma 3 [5, 6]: As k and (N_v/k) approach infinity, the k-NNC can be viewed as an attempt to estimate the *a-posteriori* probabilities from the training samples. Under this condition, k-NNC becomes optimal and

$$\lim_{k, \frac{N_v}{k} \rightarrow \infty} \mathbf{P}_{\text{e(k-NNC)}} = \mathbf{P}_{\text{eB}} \quad (16)$$

We now investigate the relationship between Bayes discriminant and the PLNC in more detail. We start by observing that the classification error for the c^{th} cluster of a PLNC is given by (5). Let us take into consideration only the feature vectors that have been assigned class label i . In this case, the classification error would be given by

$$E_{c,i} = \frac{1}{N_v(c)} \sum_{q=1}^{N_v(c)} \left[t_{qi} - \sum_{n=1}^{N+1} a_{cin} x_{qn} \right]^2 \quad (17)$$

The partial derivative of the error with respect to the weight matrix elements, a_{cim} , is given by:

$$\frac{\partial E}{\partial a_{cim}} = \frac{-2}{N_v(c)} \sum_{q=1}^{N_v(c)} \left[t_{qi} - \sum_{n=1}^{N+1} a_{cin} x_{qn} \right] x_{qm} \quad (18)$$

Before going any further, we consider how letting K and

$N_v(c)$ approach infinity effects the range and variability of the input feature vectors \mathbf{x} . Assuming the feature space is a bounded compact subset of \mathbb{R}^N , each cluster falls within a cell of the Voronoi tessellation. As K and $N_v(c)$ simultaneously increase towards infinity, the maximum radius of each cluster decreases towards zero. The range of values the elements of a feature vector in a cluster can take decreases so much that they become constant, i.e, variability within a cluster approaches zero. Within a given cluster, as \mathbf{x} becomes independent of the pattern number p , we can replace x_{pm} with mean vector element m_n in (18). We also equate the gradient to zero, so as to minimize the MSE. Equation (18) reduces to

$$\frac{-2}{N_v(c)} \sum_{q=1}^{N_v(c)} \left[t_{qi} - \sum_{n=1}^{N+1} a_{cin} m_n \right] m_m = 0 \quad (19)$$

which becomes

$$\sum_{q=1}^{N_v(c)} \left[t_{qi} - \sum_{n=1}^{N+1} a_{cin} m_n \right] = 0 \quad (20)$$

Since the terms a_{cin} and m_n are constants, we can replace the sum over n by the single constant $a_{ci(N+1)}$ which yields

$$\sum_{q=1}^{N_v(c)} \left[t_{qi} - a_{ci(N+1)} \right] = 0 \quad (21)$$

This can be written as

$$\sum_{q=1}^{N_v(c)} t_{qi} = N_v(c) \cdot a_{ci(N+1)} \quad (22)$$

which yields

$$a_{ci(N+1)} = \frac{\sum_{q=1}^{N_v(c)} t_{qi}}{N_v(c)} \quad (23)$$

Without loss of generality, we can represent t_{qi} as

$$t_{qi} = \delta(i_c(q) - i) \quad (24)$$

where δ is the Kronecker delta function. In turn (23) becomes

$$a_{ci(N+1)} = \frac{\sum_{q=1}^{N_v(c)} \delta(i_c(q) - i)}{N_v(c)} \quad (25)$$

$\delta(i_c(q) - i)$ equals 1 if $i_c(q)$ equals i , so, the numerator summation equals the number of training vectors in cluster c that belong to class i . This number could be represented by $N_v(c, i)$, hence reducing (25) to

$$a_{ci(N+1)} = \frac{N_v(c, i)}{N_v(c)} \quad (26)$$

which converges to the *a-posteriori* probability $\mathbf{P}(i|\mathbf{x})$.

For a given cluster c , $a_{ci(N+1)}$ is largest for the class i which has the most input vectors in the cluster. This is precisely how the k-NNC makes decisions. So, each PLNC cluster with $N_v(c)$ members emulates a k-NNC decision with $k = N_v(c)$. Summarizing the results above, we have the following

lemma.

Lemma 4: As K and $N_v(c)$ approach infinity, the output of a PLNC approximates the a -posteriori probability functions of the class labels, given the input vector. Under this condition, the PLNC hence becomes optimal and

$$\lim_{K, N_v(c) \rightarrow \infty} P_{e(\text{PLNC})} = P_{eB} \quad (27)$$

IV. SIMULATION RESULTS AND COMPARISONS

In this section, we compare the PLNC with the NNC, MLP and a piecewise linear MLP (PLMLP) [12] using different data sets. Each data set consists of a training file and a validation file.

A. PLNC vs NNC

Here, we compare the PLNC to a NNC having the same number of cluster mean vectors. Three data sets are used for the comparison.

- 1) *grng* [21] – This geometric shape recognition data set has four shapes having different degrees of rotation, scaling, translation, and oblique distortions. The shapes are ellipse, triangle, quadrilateral, and pentagon. Each image is 64 x 64 pixels. The input vectors contain 16 ring-wedge features. For this comparison, the NNC and PLNC, both use the same set of cluster center vectors for classification. In fig. 2, the NNC performs poorly because it is forced to use the same clusters as the PLNC, for this example only.
- 2) *gong* [22] – The raw data consists of images from hand printed numerals collected from 3,000 people by the Internal Revenue Service. Images are 32 by 24 binary matrices, which are scaled to remove character size variation. The feature set contains 16 elements. The 10 classes correspond to the Arabic numerals. For this comparison, the PLNC and NNC are allowed to design and use their own sets of cluster center vectors. The results of this comparison are shown in fig. 3. The linear classifier in each PLNC cluster allows the PLNC to outperform the NNC.
- 3) *comf18* [23] – This data set consists of texture features corresponding to an image segmentation problem. Each segmented region is separately histogram equalized to 20 levels. Then the joint probability density of pairs of pixels separated by a given distance and a given direction is estimated. We use 0, 90, 180, 270 degrees for the directions and 1, 3, and 5 pixels for the separations. The density estimates are computed for each classification window. For each separation, the co-occurrences for the four directions are folded together to form a triangular matrix. From each of the resulting three matrices, six features are computed: angular second moment, contrast, entropy, correlation, and the sums of the main diagonal and the first off diagonal. This results in 18 features for each classification

window. For this comparison too, the PLNC and NNC are allowed to design and use their own set of cluster center vectors. The results of this comparison are shown in fig. 4. As in the previous examples, the PLNC outperforms the NNC.

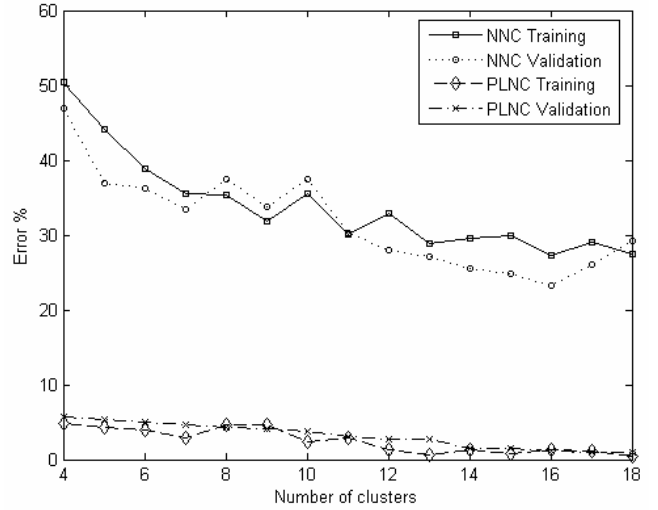


Fig. 2. Comparison of PLNC and NNC for shape recognition data

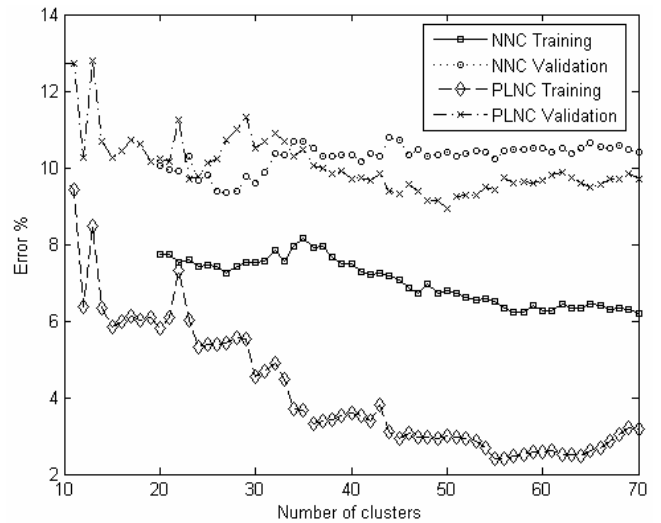


Fig. 3. Comparison of PLNC and NNC for numeral recognition problem

B. PLNC vs MLP vs PLMLP

Here, we compare the PLNC to a sigmoidal MLP and to a PLMLP [12]. Unlike the PLNC, the PLMLP produces continuous mappings. Two data sets are used for the comparison.

- 1) *F17C* – This prognostics data set consists of parameters that are available in the basic health usage monitoring system (HUMS) by Bell Helicopter Textron. The data was obtained from the M430 flight load level survey conducted in Mirabel Canada. Each input vector contains 17 elements. The 39 classes represent different

flight maneuvers like taking off, landing, turning right or left etc. The results of this comparison are shown in fig. 5. In this example, the PLNC outperforms both the MLP and PLMLP.

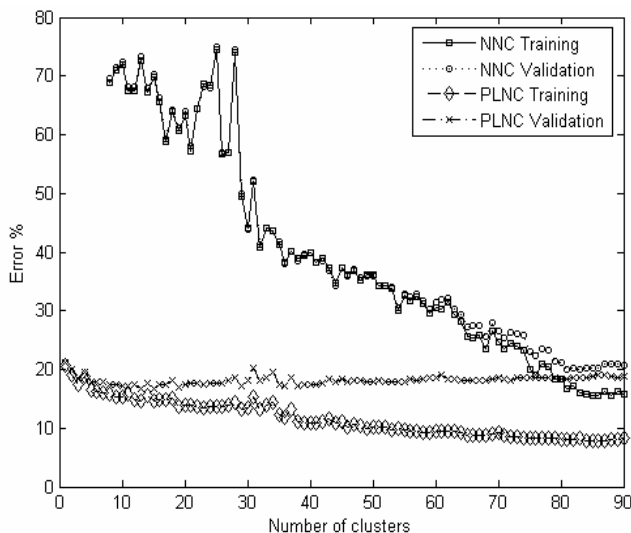


Fig. 4. Comparison of PLNC and NNC for segmentation problem

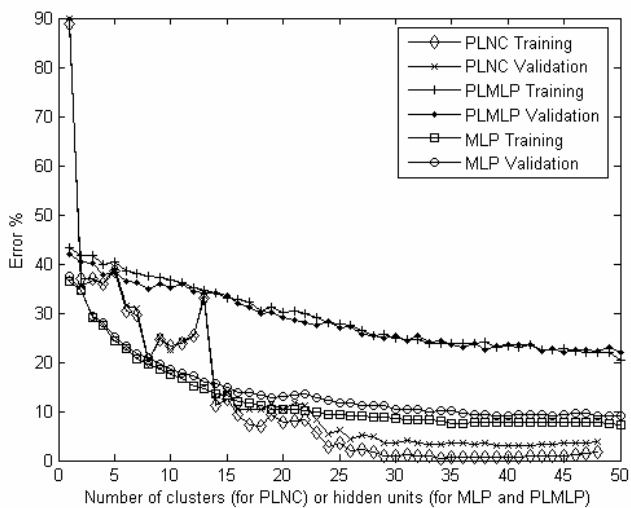


Fig. 5. Comparison of PLNC, PLMLP and MLP for prognostics

- 2) *speech* - Speech samples are first pre-emphasized and are converted into frequency domain by taking their DFT. They are then passed through Mel filter banks and the inverse DFT is applied on the output to get Mel-Frequency Cepstrum Coefficients (MFCC). Each of MFCC(n), MFCC(n)-MFCC(n-1) and MFCC(n)-MFCC(n-2) would have 13 features, which results in a total of 39 features. Each class corresponds to a phoneme. The results of this comparison are shown in fig. 6. The PLNC outperforms the PLMLP and the MLP for larger networks.

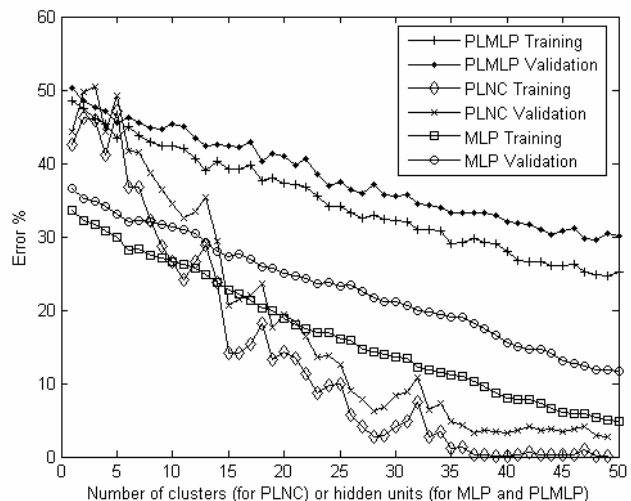


Fig. 6. Comparison of PLNC, PLMLP and MLP for phoneme recognition problem

C. Comparisons Based On the Numbers of Multiplies

Here, we re-do the plots of subsection B, so that the three full trained networks are compared based upon the numbers of multiplies required to process one input vector. For the sigmoidal MLP it is assumed that two multiplies are required to evaluate the activation. The PLMLP, in contrast, requires only one multiply for calculating the activation.

Figures 7 and 8 show that a PLNC requires fewer multiplications to obtain the same classification error as compared to the MLP and PLMLP.

V. CONCLUSIONS

In this paper we have described a piecewise linear network classifier (PLNC) and given a training algorithm for it. This algorithm has fast convergence speed because sets of linear equations are solved in each cluster. Using existing theorems for the NNC and k-NNC, we have proved that the output of the PLNC approximates a Bayes optimal discriminant when trained to minimize the mean square error (MSE). Using several data sets, we have shown that the PLNC often performs better than equivalent NNCs, MLPs, and PLMLPs. It can also be stated that the PLNC always trains faster than the MLP and often faster than the NNC, which usually requires a lot more clusters. Future work will compare various alternative methods for training the linear classifiers. PLN pruning techniques [14] will also be extended to the PLNC.

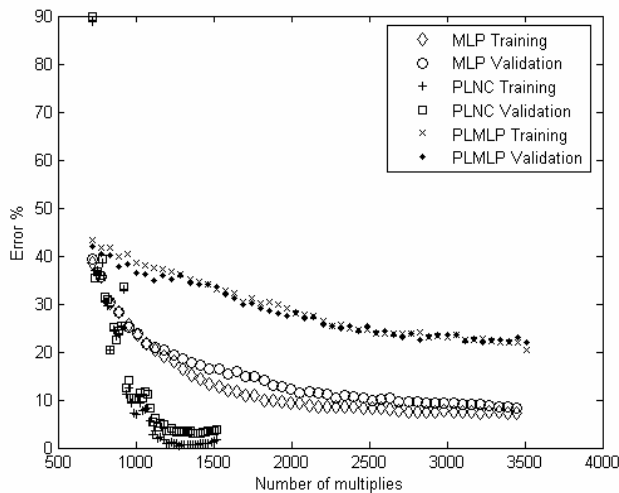


Fig. 7. Comparison of PLNC, PLMLP and MLP for prognostics data, based on number of multiplies

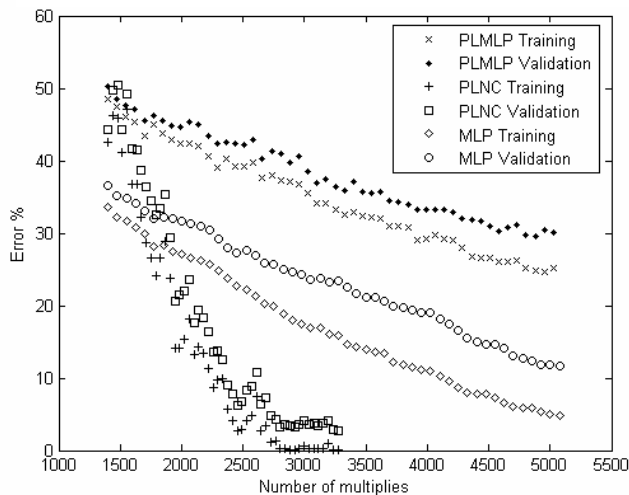


Fig. 8. Comparison of PLNC, PLMLP and MLP for phoneme recognition problem, based on number of multiplies

REFERENCES

- [1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359-366.
- [2] K. Hornik, M. Stinchcombe, and H. White, "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks," *Neural Networks*, vol. 3, 1990, pp. 551-560.
- [3] Michael D. Richard and Richard P. Lippman, "Neural Network Classifiers estimate Bayesian *a-posteriori* probabilities," *Neural Computation*, vol. 3, no. 4, pp. 461-483, 1991.
- [4] Dennis W. Ruck, Steven K. Rogers, Matthew Kabrisky, Mark E. Oxley and Bruce W. Suter, "The Multilayer Perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans Neural Networks*, TNN-1(4):296-298, 1990.
- [5] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, 1990.
- [6] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, John Wiley & Sons, 2nd ed., 2001.
- [7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.
- [8] D. B. Fogel, "An information criterion for optimal neural network selection," *IEEE Trans. Neural Networks*, vol. 2, no. 5, pp. 490-497, Sept.1991.
- [9] J. H. Friedman, "Multivariate adaptive regression splines," *Annals of Statistics*, vol. 19, no. 1, pp. 1-141, 1991.
- [10] S. Subbarayan, K. Kim, M. T. Manry, V. Devarajan and H. Chen, "Modular neural network architecture using piecewise linear mapping," *30th Asilomar Conference on Signals, Systems & Computers*, vol. 2, pp. 1171-1175, Nov. 1996.
- [11] W. Li, J.-N. Lin, and R. Unbehauen, "Canonical representation of piecewise polynomial functions with nondegenerate linear domain partitions," *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, no. 8, pp. 838-848, Aug. 1998.
- [12] D.R. Hush and B. Horne, "Efficient algorithms for function approximation with piecewise linear sigmoidal networks," *IEEE Trans. Neural Networks*, Vol. 9, No. 6, pp. 1129-1141, 1998.
- [13] E.F. Gad, A.F. Atiya, S. Shaheen, A. El-Dessouki, "A new algorithm for learning in piecewise-linear neural networks," *Neural Networks 13* (2000), pp. 485-505.
- [14] H. Chandrasekaran, J. Li, W.H. Delashmit, P.L. Narasimha, C. Yu and M.T. Manry, "Convergent Design of Piecewise Linear Neural Networks", *NeuroComputing*, vol. 70, 2007, pp. 1022-1039.
- [15] T. Kohonen, *Self-Organization and Associative Memory*, 2nd ed., Springer-Verlag, 1987.
- [16] L-M Liu, M.T. Manry, F. Amar, M.S. Dawson, and A.K. Fung, "Iterative Improvement of Image Classifiers Using Relaxation," *Conference Record of the Twenty-Eighth Annual Asilomar Conference on Signals, Systems, and Computers*, vol. 2, 10/31/94 to 11/2/94, pp.902-906.
- [17] Jiang Li, Michael T. Manry, Li-Min Liu, Changhua Yu, and John Wei, "Iterative Improvement of Neural Classifiers", *Proceedings of the Seventeenth International Conference of the Florida AI Research Society*, May 2004, pp. 700-705.
- [18] R.G. Gore, Jiang Li, Michael T. Manry, Li-Min Liu, Changhua Yu, and John Wei, "Iterative Design of Neural Network Classifiers through Regression". *International Journal on Artificial Intelligence Tools*, Vol. 14, Nos. 1&2 (2005) pp. 281-301.
- [19] F. J. Maldonado, M. T. Manry, Tae-Hoon Kim, "Finding optimal neural network basis function subsets using the Schmidt procedure", *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 444 - 449, July 2003.
- [20] F. J. Maldonado, M. T. Manry, "Optimal Pruning of Feed-forward Neural Networks Based upon the Schmidt Procedure," *The 36th Asilomar Conference on Signals, Systems, & Computers '02*, pp. 1024 - 1028.
- [21] H. C. Yau, M. T. Manry, "Iterative Improvement of a Nearest Neighbor Classifier," *Neural Networks*, Vol. 4, 1991, pp. 517-524.
- [22] W. Gong, H. C. Yau, and M. T. Manry, "Non-Gaussian Feature Analyses Using a Neural Network," *Progress in Neural Networks*, vol. 2, 1994, pp. 253-269.
- [23] R.R. Bailey, E. J. Pettit, R. T. Borochoff, M. T. Manry, and X. Jiang, "Automatic Recognition of USGS Land Use/Cover Categories Using Statistical and Neural Network Classifiers," *Proceedings of SPIE OE/Aerospace and Remote Sensing*, April 12-16, 1993, Orlando Florida.