

Convergent Design of A Piecewise Linear Neural Network

Hema Chandrasekaran and Michael T. Manry
 Department of Electrical Engineering
 The University of Texas at Arlington
 Arlington, TX 76019
 e-mail : manry@uta.edu

Abstract

A piecewise linear neural network (PLNN) is discussed which maps N -dimensional input vectors into M -dimensional output vectors. A convergent algorithm for designing the PLNN from training data is described. The design algorithm is based on a variation of backtracking algorithm known as the 'branch and bound' method. The performance of the PLNN is compared with that of a multilayer perceptron (MLP) of equivalent size. The results show that the PLNN is capable of performing as well as an equivalent MLP.

I. Introduction

Piecewise linear mappings are appealing because of their simplicity. Even though many piecewise linear mappings[1-4] have been described earlier, absolute convergence of the mean square error at every iteration has never been insisted upon. In the proposed paper, we describe a piecewise linear neural network (PLNN) in which (1) each input vector is assigned to the appropriate cluster using a distance measure, and (2) the input vector is multiplied by a matrix to generate an output vector. The network structure and initialization are described in section II. The training algorithm is discussed in section III. Numerical results are given in section IV.

II. Network Structure and Initialization

A. Notation and Basic Operation

The PLNN consists of

- N_c , N -dimensional cluster center vectors \mathbf{m}_k where $1 \leq k \leq N_c$,
- an M by $(N + 1)$ dimensional matrix \mathbf{W}_{gl}
- N_c matrices $\mathbf{W}_{pwl}(k)$ of dimension M by $(N + 1)$
- a distance measure $d(\cdot)$, and
- $2N$ parameters $\mu(i)$, $\sigma(i)$ for $1 \leq i \leq N$.

The network structure is shown in figure 1. In a trained network, each input vector \mathbf{x} is processed as follows:

1. The input vector \mathbf{x} is augmented as $(\mathbf{x}^T:1)^T$.
2. The input vector elements are normalized as $x'(i) = (x(i) - \mu(i))/\sigma(i)$ where $\mu(i)$, $\sigma(i)$ are mean and standard deviation respectively of the input vector elements.
3. Input vector \mathbf{x} is assigned to the k th cluster such that $d(\mathbf{x}', \mathbf{m}_k) = \min_n d(\mathbf{x}', \mathbf{m}_n)$ using the distance measure $d(\cdot)$.
4. The output of the network is $\mathbf{y} = \mathbf{W}_{gl} \cdot \mathbf{x} + \mathbf{W}_{pwl}(k) \cdot \mathbf{x}'$

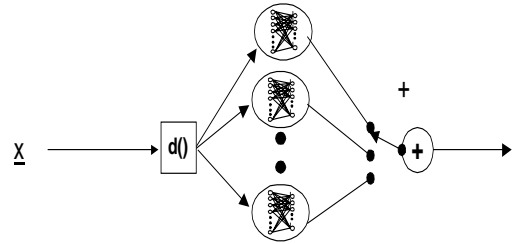


Fig. 1 Piecewise Linear Neural Network Structure

In the following subsections we summarize steps in the initialization of our training algorithm.

B. Global Mapping

Our goal here is to obtain a preliminary linear mapping between input and output vectors. Subtracting this linear mapping from the training data prevents the network modules from wasting clusters on linear mapping. This in turn, improves the efficiency of the network. The input feature vectors \mathbf{x}_p are normalized as in section II. The global linear mapping matrix \mathbf{W}_{gl} is extracted at this stage. Modified target vectors are found as

$$\mathbf{t}_p' = \mathbf{t}_p - \mathbf{W}_{gl} \cdot \mathbf{x}_p.$$

C. Modified Distance Measure

Training data sometimes contain random or less useful features. In this case, the Euclidean distance measure gives equal importance to all features, which leads to many more clusters and a more complex network than is necessary. The weighted distance measure developed in [4] mitigates this problem and leads to more efficient clusters. Such a distance measure is given by

$$d(x'_p, m_k) = \sum_{i=1}^N w(i) \cdot [x'_p(i) - m_k(i)]^2 \quad (1)$$

where $w(i)$ is the weight assigned to the i th feature's squared error.

D. Initial Clustering

Initially we start with one cluster centered at the input mean vector. This center is perturbed slightly and cluster is split into two clusters. The mean vectors of the two clusters are recalculated after reassigning all the pattern vectors to either of the two clusters using modified distance measure and nearest neighbor clustering.

E. Clusterwise Linear Regression

In each iteration, after a new cluster has been added, each cluster is fitted with a piecewise linear mapping. The mean square mapping error $E_k(j)$ for the j th output and k th cluster is

$$E_k(j) = \sum_{p \in S(k)} [t'_p(j) - y_p(j)]^2 \quad (2)$$

where $y_p(j)$ is the cluster output, $S(k)$ is the set of all patterns that are members of cluster k . The linear mapping weights are found by minimizing the error function $E_k(j)$ for each output by the conjugate gradient algorithm.

III. Network Training

The initialized network has two modules and a weighted distance measure. In each iteration, a new cluster is added by splitting one of the existing clusters. Each cluster is then fitted with a piecewise linear mapping. This method of clustering falls under the category of hierarchical clustering [5,6] utilizing a *divisive* method. Once the specified number of modules have been added, the network is pruned to obtain a compact configuration. In this section we describe the process of adding useful modules and removing less useful modules.

A. Adding a New Module

Our goal here is to find the best module to split, and to optimally split it into two modules, such that the forecast training error shows the maximal decrease. In each iteration, one module is split and the number of modules increases by one. The number of modules at the end of the n th iteration is $N_c(n)$. The mapping error for each module is calculated as

$$E_k = \sum_{j=1}^M E_k(j) \quad 1 \leq k \leq N_c(n) \quad (3)$$

Next, the clusters are ranked in descending order of their mapping errors as $E_{\pi(1)}, E_{\pi(2)}, \dots, E_{\pi(N_c(n))}$ so $E_{\pi(k)} \geq E_{\pi(m)}$ for $k < m$. Thus, cluster number $\pi(k)$ has the k th largest training error. For the k th cluster, the average distance of a pattern vector to its center vector is calculated as

$$\delta(k) = \frac{1}{N_v(k)} \sum_{p \in S(k)} \sum_{i=1}^N w(i) \cdot [x'_p(i) - m_k(i)]^2 \quad (5)$$

Here $N_v(k)$ is the total number of patterns belonging to cluster k .

Our goal is to find the best split of a cluster, such that E_k shows the largest decrease. The following steps are involved in the process of adding a new module:

1. The cluster $\pi(k)$, $1 \leq k \leq 4$ is picked as a candidate for splitting.
2. Now we need to optimally identify the two new cluster centers within the cluster with index $\pi(k)$. Using sequential leader clustering with a threshold of $\delta(\pi(k))$, a maximum of four child clusters are formed. Let the child clusters be denoted by $\pi(k,1)$, $\pi(k,2)$, $\pi(k,3)$, $\pi(k,4)$. We also accumulate the statistics for the child clusters. Specifically, the auto-correlation matrices $\mathbf{R}(k,1)$, $\mathbf{R}(k,2)$, $\mathbf{R}(k,3)$, $\mathbf{R}(k,4)$, cross-correlation matrices $\mathbf{C}(k,1)$, $\mathbf{C}(k,2)$, $\mathbf{C}(k,3)$, $\mathbf{C}(k,4)$, and the mean vectors $\mathbf{m}_{\pi}(k,1)$, $\mathbf{m}_{\pi}(k,2)$, $\mathbf{m}_{\pi}(k,3)$, $\mathbf{m}_{\pi}(k,4)$ for all the child clusters are accumulated.
3. The mean vectors $\mathbf{m}_{\pi}(k,u)$, $\mathbf{m}_{\pi}(k,v)$, $u \neq v$, $1 \leq u, v \leq 4$, are chosen as possible new cluster centers. The statistics of non-candidate child clusters are assigned to the nearest candidate child cluster, u or v .
4. The two candidate child clusters u and v are linearly fitted and their mapping errors E_u and E_v are calculated. This combination is accepted as a possible candidate pair, if $E_u + E_v$ is less than $E_{\pi(k)}$.
5. Steps 3 and 4 are repeated till we try all pairs of child clusters. Finally, we select that pair (u,v) such that $\min_{u,v} (E_u + E_v)$ is obtained.

- Now we have one more cluster than in the previous iteration. All the pattern vectors are reassigned. If, in the process of reassigning, we encounter a small cluster with less than $(N+1)$ members, then the cluster is deleted and we skip to step 8; otherwise each cluster is fitted with a piecewise linear mapping and the mapping error is calculated as

$$E = \sum_{k=1}^{N_c(n)} E_k \quad (6)$$

- If the mapping error decreases, we save the new configuration and update the bound on the mapping error as

$$E_{\min} = E \quad (7)$$

- We now backtrack the network to the configuration that existed in step 1. Steps 2 through 7 are repeated for parent clusters $\pi(2)$, $\pi(3)$, $\pi(4)$. Only the configuration having the lowest value of E_{\min} is saved.
- Finally the configuration that leads to the lowest actual bound on the mapping error is retained. This method of finding the optimal placement of a new hyperplane can be termed a *limited branch and bound method* [7] which is a special variation of the backtracking algorithm.
- If the new module results in a decrease in E , the next iteration proceeds by repeating steps 2 through 8. Otherwise the algorithm terminates.

B. Eliminating a Module

Even though the training algorithm tries to add as many modules as specified by the user, this does not necessarily lead to a compact PLNN with an optimal set of center vectors. Our goal here is to identify and eliminate the least useful module in a completely trained network without significantly degrading mapping error performance.

- For each input vector \mathbf{x}_p , identify two nearest neighbor modules with indices α, β .
- Compute the two mapping errors for \mathbf{x}_p as $E_{\alpha}^p, E_{\beta}^p$ if it were assigned to modules α and β respectively.
- Let k be the index of the module to be potentially eliminated. Set $E_k = 0$.

$$E_{\alpha} = E_{\alpha} + \sum_{p \neq k} E_{\beta}^p, \quad 1 \leq \alpha \leq N_c, \quad \alpha \neq k.$$

In the above equation, the mapping error for cluster α increases because it attracts some members from cluster k which is to be eliminated.

- Compute E'_k , the mapping error for the configuration without module k , as

$$E'_k = \sum_{\alpha=1, \alpha \neq k}^{N_c} E_{\alpha}$$

- Repeat steps 3, 4, and 5 for $1 \leq k \leq N_c$. Find the smallest E'_k . If $E'_k < E_{\min}$, eliminate that module. On the other hand, we could still eliminate module k , if E'_k is not significantly worse than E .
- Now, after assigning all patterns of the k th cluster to their next nearest neighbor, fit the clusters with a new piecewise linear mapping and save the configuration.

This procedure is repeated till a compact network is achieved. The results of pruning completely trained PLNNs are shown in figure 8 and figure 9.

IV. Efficiency

When the trained network is employed to process one input vector into an output vector, it requires $2 \cdot N \cdot N_c + N \cdot M$ multiplies. In contrast a single hidden layer fully connected MLP requires $N \cdot (N_h + M) + N_h \cdot M$ multiplies and N_h non-linear sigmoidal function calculations. Through equating the number of multiplies required to process an input vector, we find the equivalent MLP neural network for comparison purposes as

$$N_h = \frac{2 \cdot N \cdot N_c}{N + M + 2} \quad (8)$$

Here we have assumed that each sigmoidal function calculation takes as much time as two multiplies.

V. Numerical Results

In this section we describe our simulation procedure and discuss the results. Figures 2 through 7 depict PLNN training and testing performance. In addition, the MLP training and testing errors are also shown. The number of modules N_c required to adequately generalize any data file, is the point on the knee of the PLNN testing curve. If the PLNN testing curve does not show prominent upward trend, then the number of modules required is that point, where the PLNN testing curve starts to flatten out. Then an equivalent MLP structure is found using equation (8).

The PLNN training algorithm requires five passes through the training data file per iteration. In every iteration one module is added till the training algorithm terminates. MLP network trained with OWO-HWO [8] algorithm requires two passes through the training data per iteration. Let the number of modules read off the PLNN testing curve be N_c . Then the number of hidden units in an equivalent MLP given by (8) is N_h . The MLP is trained for $3 \cdot N_c$ iterations. In the following figures, an iteration of the PLNN training is equal to 3 iterations of MLP training. So, every third iteration of the MLP training is plotted alongside one PLNN iteration.

A. Data Sets With One Output

X : This training and testing data sets come from a chaotic time series generated by the Mackey-Glass delay-difference equation[9] with $a = 0.2$, $b = 0.1$, and $\tau = 17$. The desired output is the sample at time $T+6$. The data set consists of 4 inputs and 1 output and contains 2654 training patterns and 1806 testing patterns. The PLNN training and testing curves are plotted in figure 2. The training and testing performances of an MLP with $N_h = 6$ equivalent to a PLNN with $N_c = 5$, are also shown. From figure 2, it is clear that PLNN trains and generalizes as well as the MLP for this data set.

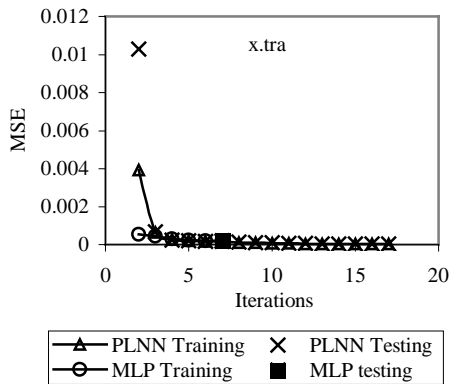


Figure 2

Hwang-t3 : The "hwang-t3" data set contains artificial data derived from a real valued function of two variables, used by J.N.Hwang and others to test non-parametric regression methods[10]. The function y is defined to be

$$y = 42.659 \cdot (0.1 + x_1 \cdot (0.05 + x_1^4 - 10 \cdot x_1^2 \cdot x_2^2 + 5 \cdot x_2^4))$$

The functions are all real-valued, and are defined on the two-dimensional domain $[0,1] \times [0,1]$. There are 8170 training patterns, 5130 testing patterns with 2 inputs and 1 output.

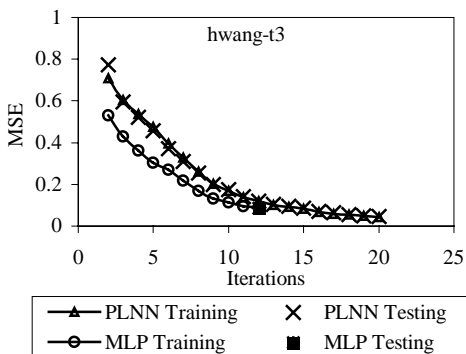


Figure 3

The PLNN training and testing curves are shown in figure 3. Also shown are the performance of an MLP with $N_h = 8$, equivalent to a PLNN with $N_c = 10$. We observe that the PLNN trains and tests as well as the MLP.

Puma-8nm : This file is part of the DELVE data set and contains 4918 training patterns and 3274 testing patterns. The data is artificial and describes forward kinematics of a robot arm. The inputs include angular positions, velocities and torques of the robot arm. The output is the angular acceleration of one of the robot arm's links. The data set is characterized by 8 inputs, 1 output, high non-linearity, and medium noise. The MLP has $N_h = 14$, equivalent to a PLNN with $N_c = 9$. Figure 4 clearly illustrates that the PLNN performs as well as the MLP both during training and testing.

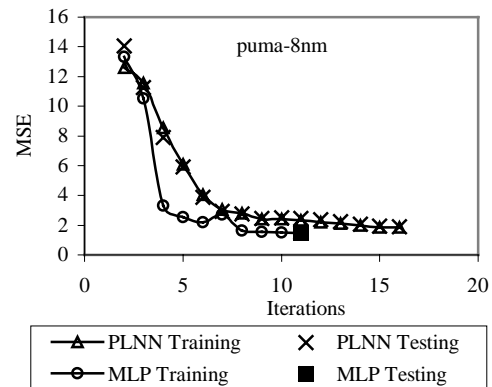


Figure 4

B. Data Sets With Multiple Outputs

Twod : The data set *Twod* contains simulated data based on models from backscattering measurements. The data set has 8 inputs and 7 outputs, 1768 training patterns and 1000 testing patterns. The inputs consist of eight theoretical values of backscattering coefficient parameters at V and H polarizations and four incident angles. The outputs are the corresponding values of permittivity, upper surface height, lower surface height, normalized upper surface correlation length, normalized lower surface correlation length, optical depth and single scattering albedo which has a joint uniform pdf [11]. The MLP has the structure 8-8-7 equivalent to a PLNN with $N_c = 8$. Figure 5 depicts PLNN training and testing performances, MLP training and testing performances. In this case we note, MLP performs slightly better than the PLNN.

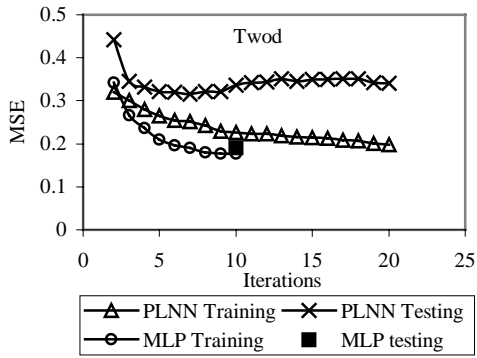


Figure 5

Build3 : This file is part of the Proben1 benchmarking data sets. The data set represents the problem of predicting the hourly consumption of electrical energy, hot water, and cold water based on the date, time of day, outside temperature, outside air humidity, solar radiation, and wind speed. There are 8 inputs, and 3 outputs and 2104 training patterns and 1707 testing patterns in this data set. The MLP has the structure 8-12-3 equivalent to a PLNN with $N_c = 10$. Figure 6 clearly illustrates that the PLNN performs as well as the MLP both during training and testing.

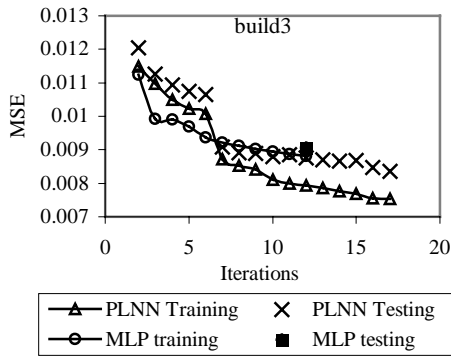


Figure 6

F17 : The third multiple-output example is the data set *F17*, which contains 2823 training patterns and 1922 testing patterns for onboard *Flight Load Synthesis* (FLS) in helicopters. There are 17 inputs and 9 outputs for each pattern. This data was obtained from the M430 flight load level survey conducted in Mirabel Canada in early 1995 by Bell Helicopter Textron of Fort Worth. The MLP has the structure 17-18-9 equivalent to that of a PLNN with $N_c = 12$. Figure 7 clearly illustrates that the PLNN performs as well as the MLP both during training and testing.

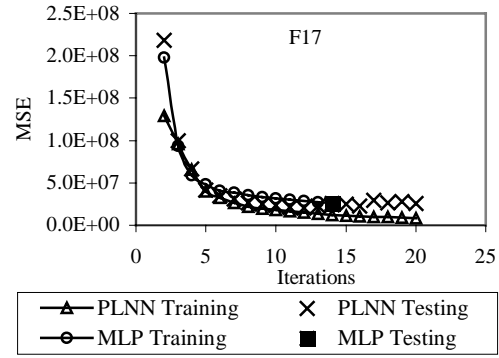


Figure 7

C. Pruning Results

The results of eliminating one least useful module at a time is plotted along with the PLNN training in figures 8 and 9. The results illustrate that as we compact and retrain the network, the resulting configuration is an improvement over the original configuration.

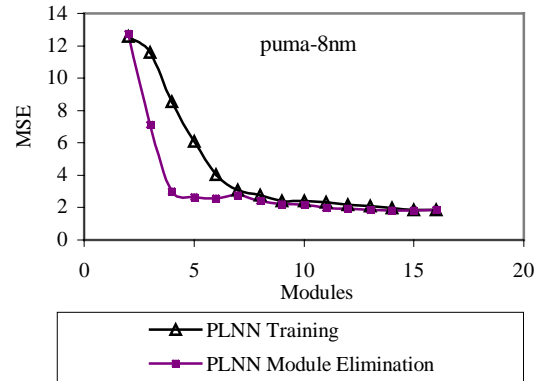


Figure 8

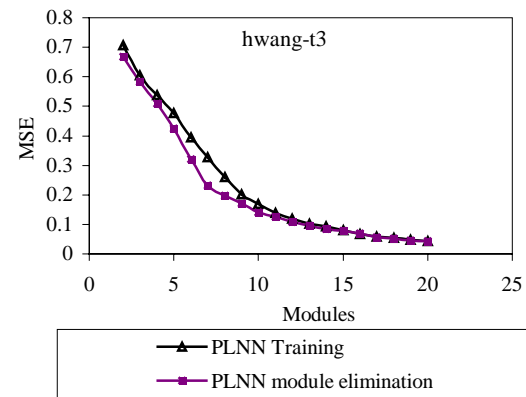


Figure 9

V. Conclusions

In this paper we have described a piecewise linear neural network and given a convergent design algorithm for it. An effective pruning algorithm for the network has been given. Using several benchmark data sets, we have shown that the PLNN usually performs as well as an equivalent MLP.

Acknowledgement

This work was funded by the Advanced Technology Program of the state of Texas under grant 003656-063.

References

- [1] M.Mattavelli, E.Amaldi, and J.Vesin, "A Perceptron-Based Approach to Piecewise Linear Modeling with an Application to Time Series", *Lecture Notes in computer science - ICANN'97*, No. 1327, 1997.
- [2] J.C.Principe and L.Wang, "Non-linear Time Series Modeling with Self-Organization Feature Maps", *Proceedings of the 1995 IEEE Workshop*, pp 11-21.
- [3] J.N.Lin and R.Unbehauen, "Canonical Piecewise-Linear Networks", *IEEE Transactions on Neural Networks*, Vol.6, No. 1, Jan 1995.
- [4] S.Subbarayan, K.Kim, M.T.Manry, V.Devarajan and H.Chen, "Modular Neural Network Architecture using Piecewise Linear Mapping", *Thirtieth Asilomar Conference on Signals, Systems & Computers*, Vol. 2, pp 1171-1175, Nov 1996.
- [5] R.Gnanadesikan, *Methods for Statistical Data Analysis of Multivariate Observations*, John Wiley & Sons, Inc. 1997.
- [6] H.Späth, *Cluster Analysis Algorithms for Data reduction and Classification of Objects*, Ellis Horwood Limited, 1980.
- [7] T.C.Hu, *Combinatorial Algorithms*, Addison-Wesley Publishing Company, Inc. 1982
- [8] H.H.Chen, M.T. Manry and Hema Chandrasekaran , "A Neural Network Training Algorithm Utilizing Multiple sets of Linear Equations", *Neurocomputing* Vol. 25 No.1-3, pp 55-72, April 1999.
- [9] A.Lapedes and R.Farber, "Nonlinear signal processing using neural networks: prediction and system modeling", *Technical Report LA-UR-87-2662*, Los Alamos National Laboratory, Los Alamos, NM, 1987.
- [10] Hwang, J.-N., Lay, S.-R., Maechler, M., Martin, R. D., and Schimert, J. "Regression modeling in back-propagation and projection pursuit learning", *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 342-353, 1994.
- [11] A.K.Fung, Z.Li, and K.S.Chen, "Backscattering from a randomly rough dielectric surface", *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 30, No. 2, pp. 356-369, 1992.