Output Weight Optimization for the Multi-Layer Perceptron

by

M.T. Manry¹, Xiujun Guan¹, S.J. Apollo², L.S. Allen³, W.D. Lyle³, and W. Gong⁴

¹Department of Electrical Engineering, University of Texas at Arlington, Arlington, Texas 76019

²General Dynamics, P.O. Box 748, Fort Worth, Texas 76101

³Mobil Research and Development Corp., 13777 Midway Road, Dallas, Texas 75244

⁴Radiology Dept., U.T. Southwestern Medical Center, Dallas, Texas 75235

Abstract

In a previous paper, a fast method for designing multi-layer perceptron (MLP) neural networks was introduced, in which linear equations are solved for the output weights. Here, the method is motivated by an analysis of the MLP, based on polynomial basis functions (PBFs). A conjugate gradient solution to the output weight equations is introduced. A mutation technique is described, which can be used to improve hidden unit weights. The output weight optimization (OWO) technique is extended to classification networks, which have nonlinear output unit activations. In several examples, it is seen that OWO is significantly faster than back-propagation (BP).

I. Introduction

MLP networks have been widely applied in the fields of pattern recognition and signal and image processing. The most critical problem is the long training time required. In this paper, we extend the design technique of Barton [1]. In section II, we introduce a polynomial basis function model of the MLP, and use it to justify the technique. In section III, we describe a conjugate gradient solution method, which works much better than other approaches, especially when the sets of linear equations are ill-conditioned. The technique’s efficiency, relative to back-propagation (BP), is described. The method is extended to the case of classification networks having sigmoidal output units. In section IV, examples are given in which OWO training is compared with BP.

II. Basis Vector Representation

Here, we introduce a matrix-vector representation [2] for trained MLP network outputs, in terms of their inputs. The activation of the mth hidden unit in the network, assuming that it is analytic, can be modeled as a power series with integer degree p(m) [2],

\[ \phi^{(m)} = \sum_{n=0}^{p(m)} D(m,n) X^{(m)}(m)^n \]  

(1)

\[ X^{(m)}(m) = \sum_n \phi^{(n)}(m) w(m,n) + \theta(m) \]  

(2)

where D(m,n) is the power series coefficient for the nth degree term in the mth unit, X^m(n) and \( \phi^{(n)}(m) \) denote the net input and activation output [3] respectively for the mth unit’s kth input training vector, and \( N_{m,n} \) denotes the number of training patterns. w(m,n) is the weight from the nth unit to mth unit and \( \theta(m) \) is an additive bias for mth unit, as shown in Fig. 1. The ith output unit net function [3] of an MLP can be characterized as the weighted sum of the basis vectors

\[ f(x) = \sum_{i=1}^{N} w^{(i,m)} \phi^{(m)} \]  

(3)

where \( N_s = 1 + N + N_h \) and \( N_h \) denotes the number of
To minimize training error.

(2) Randomizing the hidden unit weights insures that the rows of C are linearly independent.

(3) If P is the minimum network degree required for solving (1) with an acceptable amount of error, and if the network has \((L-N-1)\) hidden units, then we can solve for a weight vector \(W_o\) which produces this acceptable amount of error.

III. Fast Learning in Mapping Networks

In this section, we describe the basic OWO algorithm for designing mapping networks, propose some improvements to it, and extend the technique to classification networks.

A. Basic Algorithm

The training error for the \(i\)th output unit can be written as

\[
E(i) = \sum_{p=1}^{N_o} \left[ T_p - f_p(i) \right]^2
\]

(5)

where \(f(x)\) is the vector of output net functions, \(X = [x_{10}, x_{11}, \ldots, x_{1N}, x_{20}, x_{21}, \ldots, x_{2N}, \ldots, x_{m0}, x_{m1}, \ldots]\).

From Eq. (4), we see that the PBF model naturally divides the MLP into three components; the output weight vector \(W_o\), the matrix \(C\) whose rows correspond to hidden unit PBFs, and the vector \(X\) which stores information about the inputs. Several facts are clear.

(1) The output unit net functions are linear functions of the output weight vector \(W_o\), so linear equations can be solved...
possible to use the conjugate gradient approach [5] to minimize $E(i)$. This approach minimizes computer storage, theoretically converges to the exact solution in $N_u$ iterations for quadratic error functions such as $E(i)$, will work even if the set of equations has an infinite number of solutions, and can be used for iterative improvement of the solution when there are numerical problems such as round-off error.

The OWO MLP network design technique is summarized as follows.

1. Initialize all weights and thresholds as small random numbers in the usual manner.
2. Calculate the augmented input vector elements $\phi^p(m)$, from the original training vectors and hidden unit outputs.
3. Calculate the cross- and auto-correlations $T(m)$ and $r(k,m)$.
4. Minimize $E(i)$ for $i$ between 1 and $N_u$.

At this point we can use the weights or train them further, using regular BP. It should be noted that some investigators have tried to speed up training by applying conjugate gradient optimization to all of the network weights [6]. This approach is not entirely appropriate because the objective function is not quadratic (unlike ours), and careful time-consuming line searches are required [6]. We will analyze the efficiency of OWO, relative to back-propagation (BP).

In practical applications, the nets resulting from OWO may not be adequate. Some training of the hidden unit weights may be required. In addition to applying BP training to an OWO-initialized net, it is possible to apply pseudo-genetic mutation algorithms. In mutation algorithms [7], an existing weight set has small random numbers added to it. The new network is kept if its performance is better than the old one's, and discarded if it performs worse. In each iteration of our version of mutation, the hidden unit weights are updated as

$$w(i,m) \leftarrow w(i,m) + w(i,m) \tau n_i(i,m)$$

$$\theta(i) \leftarrow \theta(i) + \theta(i) \tau n_\theta(i)$$

where $n_i(i,m)$ and $n_\theta(i)$ denote Gaussian random numbers with standard deviation $\sigma$, and $\tau$ denotes a small positive number between 0 and 1. The output weights are recalculated using OWO, and the training error $E$ is found.

### C. Extension to Classification Networks

For classification or decision making networks, according to Ruck et al [8], MLPs used for classification, can closely approximate optimal Bayes discriminant functions which are bounded between 0 and 1. Using sigmoid output units constrains the MLP outputs to be in the required range, and may improve performance. Therefore, it is desirable to extend the OWO procedure to this case. This cannot be done directly, since the sigmoidal output layer activations are not a linear function of the output weights. Our approach to this is as follows. For each training pattern, we obtain the desired sigmoidal outputs of +1 or 0 for each unit. Then, we back up to the net function and choose desired net outputs of +5 or -5 respectively. We now have training data for a mapping network, which has linear output layer activations. After training via OWO, as in subsection B, we re-connect the sigmoidal operators to the output layer.

### D. Computational Load

Our goal here is to find the numbers of multiplications required for one iteration of BP, through all of the training data, and one iteration of OWO. Let $N_h$ and $N_w$ denote the numbers of hidden units and all the weights respectively. For both techniques, the number of multiplies used in evaluating the net functions is $N_hN_w$. For BP, the numbers of multiplies required for evaluating delta functions is $N_w[2N_h + (N-2)N_h]$, and the number required for updating the weights and thresholds is $N_w[2N_h + N_b + N_m]$. The total number of multiplies required for one BP iteration is then

$$N_{bp} = N_w[4N_h - (N-3)N_h - (N-3)N_w]$$

In OWO, the number of multiplications required for calculating the autocorrelations and cross correlations is...
The gradient calculations require $N_3 N_d$ multiplies. The calculation of $\beta$ requires three times that. Then, the $\alpha$ calculation, weight updates, and direction vector calculations each require $N_d N_a^2$ multiplies. The total number of multiplies required for one OWO iteration is then

$$N_{\text{owo}} = \frac{N_d N_a (N_a^2 + 2 N_a + 1)}{2} + N_d N_a$$

(10)

The first two of the six input variables correspond to $x$ and $y$, which are the horizontal and vertical distances in meters from the particle beam to the center of the SSC cross section. The second two variables are $x'(t)$ and $y'(t)$. The last two variables are the time of flight and its time derivative. The output of the polynomial is the predicted value of $x$ one revolution in the future. We generated 200 training vectors and 200 testing vectors from a dynamic map, keeping only one of the desired outputs (the distance $x$). The training results from BP, ordinary OWO, and OWO mutation are displayed in Fig. 2. For both training algorithms, the initial random weights had a standard deviation of 2. The learning factor for BP was .05, and the gain factor $r$ in Eq. (17) and Eq. (18), for OWO training, was .05. For the networks trained via OWO, the desired output and actual output are shown in Fig. 3 for testing data. It seems that the trained network can work well for both training data and testing data. It is clear that OWO works better than BP and that OWO with mutation works better than OWO without mutation.

IV. Training Examples

A. Mapping Example

Six high degree polynomials in six variables, collectively called the dynamic map, can theoretically be used to model the dynamics of particle beams in accelerators such as the superconducting super collider (SSC) [9].

![Figure 2. Training of SSC Network](image)

The shapes had varying amounts of rotation, scaling, translation, and oblique distortion. From each of the 200 training images and 200 testing images, 16 ring-wedge energy feature [10] were calculated. The MLP network topology was 16-20-12-4. For the OWO training, one iteration of OWO was used, followed by BP. The BP training approach included a momentum term and an adaptively determined learning factor [18], and is much faster than conventional BP. The training results are shown.

B. Classification Examples

As a classification example, we applied OWO and BP training to networks used for shape recognition. The raw images contained one of four geometric shapes; ellipse, triangle, quadrilateral, or pentagon. The raw images contained one of four geometric shapes; ellipse, triangle, quadrilateral, or pentagon. The raw images contained one of four geometric shapes; ellipse, triangle, quadrilateral, or pentagon.
in Fig. 4. For this example, it is clear that OWO has provided a useful initialization of the MLP.

V. Conclusions

We have shown that a PBF model of the MLP gives some insight as to how the previously described OWO training method works. One iteration of OWO requires approximately as many multiplications as one iteration of BP. A conjugate gradient solution to the output weight equations has allowed us to train much larger networks than previously possible. A mutation algorithm has been demonstrated which allows for the improvement of hidden unit weights. We have extended the OWO training procedure to classification networks, having nonlinear output layers. In the design of mapping networks, OWO training can lead to good networks in one iteration. In the design of classification networks, OWO provides a useful initial network, that can be improved via BP training.

Acknowledgement

This work was supported by a grant from Mobil Research and by NASA under Grant NAGW-3091.

References