

Iterative Improvement of Neural Classifiers

Jiang Li, Michael T. Manry, Li-Min Liu, Changhua Yu, and John Wei

Department of Electrical Engineering, University of Texas at Arlington
Arlington, Texas 76019.
manry@uta.edu

Abstract

A new objective function for neural net classifier design is presented, which has more free parameters than the classical objective function. An iterative minimization technique for the objective function is derived which requires the solution of multiple sets of numerically ill-conditioned linear equations. An enhanced feedforward network training algorithm is derived, which solves linear equations for output weights and reduces a separate error function with respect to hidden layer weights. The design method is applied to networks used to classify aerial survey imagery from remote sensing and to networks used to classify handprinted numeral image data. The improvement of the iterative technique over classical design approaches is clearly demonstrated.

Introduction

Two commonly used neural network classifiers are the functional link neural network (FLNN) (Pao 1989) and the multilayer perceptron (MLP) (Rumelhart & McClelland 1988). The MLP and FLNN approximate the general Bayes discriminant (Ruck *et al.* 1990; Wan 1990). FLNNs and MLPs are designed by minimizing the standard training error,

$$E = \sum_{i=1}^{N_c} E(i) \quad (1)$$

where N_c is the number of classes and $E(i)$, the mean-squared error for the i th output is

$$E(i) = \frac{1}{N_v} \sum_{p=1}^{N_v} [t_p(i) - y_p(i)]^2 \quad (2)$$

Here $t_p(i)$ denotes the i th desired output for the p th input pattern, $y_p(i)$ denotes the i th observed output for the p th input pattern, and N_v denotes the total number of training patterns. In this paper, we assume that $t_p(i_c) = b$ and $t_p(i_d) = -b$ where i_c denotes the correct class number for the current training pattern, i_d denotes any incorrect class number for that pattern, and b is a positive constant. If

$$i_c = \arg \max_i y_p(i), \quad (3)$$

This work was funded by NASA under Grant NAGW-3091, by the NSF under grant IRI-9216545, by EPRI under grant RP 8030-09, and by the Advanced Technology Program of the state of Texas under grant 003656-0129-2001.

we say the network classifies the current pattern correctly. Otherwise, a classification error is counted.

For neural network classifiers having linear output layer activations, the i th output can be written as

$$y_p(i) = \sum_{j=1}^{N_u} w_o(i, j) X_p(j) \quad (4)$$

where output weight $w_o(i, j)$ denotes the weight connecting the j th unit to the i th output unit, and N_u denotes the number of units feeding signals to the i th output unit. $X_p(j)$ denotes the j th basis function for the p th pattern. In the FLNN, $X_p(j)$ may represent a multinomial combination of the N inputs x_n whereas in the MLP, $X_p(j)$ may represent an input vector, the constant "1" or a sigmoidal activation for a hidden unit. Both the MLP and the FLNN can be designed by solving linear sets of equations. A conjugate gradient (CG) solution to these sets of equations has been given that works, even though the equations are ill-conditioned (Pao 1989; Barton 1991; Sartori & Antsaklis 1991; Manry *et al.* 1992; Dawson *et al.* 1992).

The error function E in (1) is too restrictive in at least two ways if a classifier is being designed. First, if each individual output vector has a different constant bias add to it, E could be increased or decreased, but this has no effect on the classification error. Second, if an output has the correct sign but a magnitude larger than b , E will increase while the classification error will be unaffected or decrease. In order to take advantage of these facts, we have developed the Output Reset (OR) algorithm, which uses a relaxation approach in combination with CG. First, we develop OR as a method for updating only neural network weights connected directly to output units. Then OR is integrated into a training algorithm that also modifies MLP hidden weights. Examples are given to show that the OR algorithm, combined with enhanced OWO-HWO training, results in greatly improved classifiers.

The Output Reset Algorithm

We first introduce OWO training and then derive the OR algorithm for the fixed basis function case. For the MLP, this corresponds to keeping the hidden weights, which feed into the hidden units, constant.

Output Weight Optimization

Applying OWO to the three layer, fully connected MLP, the basis functions are defined as

$$X_p(k) = \begin{cases} x_p(k) & \text{for } 1 \leq k \leq N \\ 1 & \text{for } k = N + 1 \\ O_p(k - N - 1) & \text{for } N + 2 \leq k \leq N + N_h + 1 \end{cases} \quad (5)$$

where $O_p(j)$ is the j th hidden unit output activation for the p th pattern. $O_p(N + 1) = 1$ to handle the hidden unit and output unit biases. Substituting (4) into (2), the mean-squared error for the i th output can be rewritten as

$$E(i) = \frac{1}{N_v} \sum_{p=1}^{N_v} [t_p(i) - \sum_{k=1}^{N_u} w_o(i, k) X_p(k)]^2 \quad (6)$$

where $N_u = N + N_h + 1$. Taking the gradient of $E(i)$ with respect the output weights, we get

$$g(m) = \frac{\partial E(i)}{\partial w_o(i, m)} = -2[c(i, m) - \sum_{k=1}^{N_u} w_o(i, k) \cdot r(k, m)] \quad (7)$$

where $1 \leq m \leq N_u$. The cross-correlation $c(i, m)$ and auto-correlation $r(k, m)$ are defined as

$$c(i, m) = \sum_{p=1}^{N_v} t_p(i) \cdot X_p(m) \quad (8)$$

$$r(k, m) = \sum_{p=1}^{N_v} X_p(k) \cdot X_p(m) \quad (9)$$

Setting $g(m)$ to zero, we get

$$\sum_{k=1}^{N_u} w_o(i, k) \cdot r(k, m) = c(i, m) \quad 1 \leq m \leq N_u \quad (10)$$

For each value of i we have a set of N_u equations in N_u unknowns. Since those linear equations are generally ill-conditioned, the conjugate gradient approach can be utilized to get the output weights which minimize $E(i)$.

OWO is only adequate for generating a useful initial network, after the hidden weights have been initialized. Note that the hidden weights are not updated in OWO training .

Derivation of OR Algorithm

In the OR algorithm we (1) give each output vector the specific bias which minimizes E , and (2) set the desired output equal to the actual output when the output has the correct sign but is larger than b in magnitude. The error function E can be modified as

$$E' = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{N_c} [t'_p(i) - y_p(i)]^2 \quad (11)$$

where $t'_p(i) = t_p(i) + a_p + d_p(i)$ and where $d_p(i)$ is a function of p and i to be defined later. Our goal is to find a_p , $d_p(i)$ and $y_p(i)$ that minimize E' , under the following conditions:

1. The difference $|t'_p(i_c) - t'_p(i_d)|$ must be larger than or equal to $2b$. Without this condition, E' can be minimized by setting the network weights and the difference $|t'_p(i_c) - t'_p(i_d)|$ to zero.
2. Each change made to a_p , $d_p(i)$ and $t'_p(i)$ (through changes in the network weights), must reduce E' or keep it unchanged.

Using these two conditions the following three methods can be used to decrease E' .

Method 1. Changes to a_p In order to minimize E' with respect to a_p , it is sufficient that the first derivative of E' with respect to a_p be zero, yielding

$$a_p = \frac{1}{N_c} \sum_{i=1}^{N_c} [y_p(i) - t_p(i) - d_p(i)] \quad (12)$$

After adding a_p to each desired output for the p th pattern, the distances between the correct class's output and the others are the same as before. Therefore, the classification performance remains the same and condition (1) is satisfied. Since a_p is specifically found to minimize E' , condition (2) is also satisfied.

Method 2. Changes to $d_p(i)$ Ignoring condition (1), $d_p(i)$ can be found such that the term $[t_p(i) + a_p + d_p(i) - y_p(i)]^2$ is zero, yielding $d_p(i) = y_p(i) - t_p(i) - a_p$, which satisfies condition (2). However, in order to satisfy condition (1), we modify $d_p(i)$ such that $d_p(i_c) \geq 0$, $d_p(i_d) \leq 0$. In summary,

- a If $y_p(i_c) \geq t_p(i_c) + a_p$, then choose $d_p(i_c) = y_p(i_c) - t_p(i_c) - a_p$
- b If $y_p(i_d) \leq t_p(i_d) + a_p$, then choose $d_p(i_d) = y_p(i_d) - t_p(i_d) - a_p$
- c Otherwise, choose $d_p(i_c) = 0$ and $d_p(i_d) = 0$

Method 3. Changes to $y_p(i)$ In (6), $t_p(i)$ is replaced by $t'_p(i)$ and the error E' is minimized with respect to the output weights using OWO. This requires the substitution of $t'_p(i)$ for $t_p(i)$ in (8). The hidden weights are not updated in this method.

Three Output Reset Algorithms

Three OR algorithms are developed by combining the different methods given above.

- Basic algorithm: The detailed steps are as follows:
 - Find and store the desired outputs $t_p(i)$
 - For each input vector \mathbf{x}_p
 - * Calculate the network outputs $y_p(i)$ using the current network weights
 - * Find $t'_p(i)$, using one or more methods from Subsection .
 - * Accumulate the auto- and cross-correlations needed in Method 3 for the OWO algorithm, where $t'_p(i)$ replaces $t_p(i)$ in (8).
 - Find the output weights $w_o(i, j)$ using OWO. Go back to the previous step for another iteration, if desired.

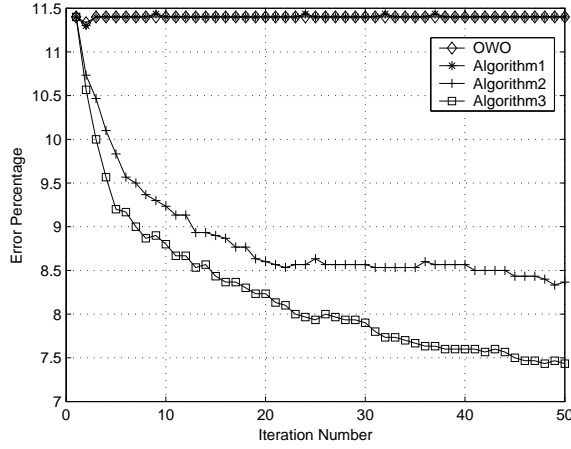


Figure 1: Performances comparison among algorithms

In algorithm 1, the desired outputs $t'_p(i)$ are found using Method 1, with $d_p(i)$ set to zero. In algorithm 2, $t'_p(i)$ is found using Method 2, with a_p set to zero. In algorithm 3, we initially use Method 1 to find $t'_p(i)$ with $d_p(i)$ set to zero. Then we alternate between Method 1 and Method 2 for three iterations.

Performance Comparisons of The Three Algorithms

In this section we illustrate training performances of the three proposed algorithms. The generalization capabilities of the algorithms will be discussed later.

We run these algorithms on a data set generated from handprinted numeral images using methods described by Gong (Gong, Yau, & Manry 1994). These images were collected by the Internal Revenue Service from 3,000 people. Each digitized image belongs to one of the ten classes representing numerals 0 to 9. We randomly chose 300 characters from each class to generate 3,000 training patterns. An image scaling algorithm is used to remove size variation in characters. Each image has 16 features.

A fully connected three layer feedforward neural network with sigmoid activation function for the hidden layer was chosen. The output layer has linear activation functions. We chose ten hidden units for the network. The three OR algorithms and the OWO algorithm alone have been utilized to train the networks yielding the results in Fig. 1. In the figure, the horizontal axis represents the iteration number while the vertical axis corresponds to the classification error percentage for each iteration.

It is observed from Fig. 1 that Algorithms 2 and 3 generate considerably lower classification error percentage than OWO training alone. Algorithm 1 does not show any significant improvement. The third algorithm produces the lowest error. In the remainder of this paper, algorithm 3 is referred to as the OR algorithm. Note that only weights in the output layer are being updated at this point.

OR Enhanced MLP Training

In this section, we discuss the integration of OR into more advanced MLP training algorithms, in which all weights are subject to optimization. There are many well-developed training algorithms, including the Back Propagation (BP), Conjugate Gradient (CG) and Levenberg-Marquardt (LM) algorithms. Training error can be further decreased when OR is used in most algorithms.

We are interested in an algorithm called OWO-HWO (Yu & Manry 2002), which can be used in the training of feed-forward neural networks such as the MLP. In OWO-HWO, we alternately modify output weights and hidden unit weights to reduce the training error. Since the output units have linear activation functions, in this paper the OWO procedure is used to obtain output weights by solving linear equations, whereas the HWO is utilized to calculate the hidden weight changes by minimizing a mean-square error between the desired and the actual net function.

Review of OWO-HWO

We are given a set of N_v training patterns $(\mathbf{x}_p, \mathbf{t}_p)$ where the p th input vector \mathbf{x}_p and p th desired output vector \mathbf{t}_p have dimension N and N_c , respectively. A three layer, fully connected MLP networks with sigmoid activation function for the hidden layer is used. For the p th pattern, the j th hidden unit net and activation functions are

$$net_p(j) = \sum_{k=1}^{N+1} w(j, k) \cdot x_p(k) \quad (13)$$

$$O_p(j) = f(net_p(j)) = \frac{1}{1 + \exp(-net_p(j))} \quad (14)$$

the i th observed output is

$$y_p(i) = \sum_{k=1}^{N+1} w_{oi}(i, k) \cdot x_p(k) + \sum_{j=1}^{N_h} w_{oh}(i, j) \cdot O_p(j) \quad (15)$$

where $w_{oi}(i, k)$ and $w_{oh}(i, j)$ are weights connecting to the i th output unit from the k th input and j th hidden unit respectively. The output weights $w_{oi}(i, k)$ and $w_{oh}(i, j)$ can be found using the OWO method. In the HWO procedure, the hidden weights $w(j, k)$ are updated by minimizing a separate error function for each hidden unit. For the j th hidden unit and p th pattern, the desired net function $net_{pd}(j)$ is constructed as (Scalero & Tepedelenlioglu 1992)

$$net_{pd}(j) \cong net_p(j) + Z \cdot \delta_p(j) \quad (16)$$

Z is the learning rate and $\delta_p(j)$ is the delta function of the j th hidden unit and is defined as

$$\delta_p(j) = f'(net_p(j)) \sum_{i=1}^{N_c} \delta_{po}(i) w_o(i, j) \quad (17)$$

where $\delta_{po}(i)$ is the delta function of the i th output layer,

$$\delta_{po}(i) = t_p(i) - y_p(i) \quad (18)$$

The hidden weights are updated as

$$w(j, k) \leftarrow w(j, k) + Z \cdot e(j, k) \quad (19)$$

where $e(j, k)$ is the hidden weight change. With the basic operations and (16-18), we can use the following equation to solve for the changes in the hidden weights,

$$net_{pd}(j) + Z \cdot \delta_p(j) \cong \sum_{k=1}^{N+1} [w(j, k) + Z \cdot e(j, k)] \cdot x_p(k) \quad (20)$$

we obtain

$$\delta_p(j) \cong \sum_{k=1}^{N+1} e(j, k) \cdot x_p(k) \quad (21)$$

Before solving (21) in the least squares sense, an objective function (Yu & Manry 2002) for the j th hidden unit is defined as

$$E_\delta(j) = \sum_{p=1}^{N_v} \left[\delta_p(j) - \sum_{k=1}^{N+1} e(j, k) x_p(k) \right]^2 f'(net_p(j)) \quad (22)$$

which is minimized with respect to $e(j, i)$ using the conjugate gradient method which is similar as we did in OWO thus we obtained the hidden weights change $e(j, k)$, we then update the hidden weights by performing (19).

Algorithm Description for OR Combined MLP Training

Using both OR and OWO-HWO, we construct the following algorithm,

1. Initialize all weights and thresholds as small random numbers in the usual manner. Pick a value for the maximum number of iterations, N_{it} . Set the iteration number i_t to 0.
2. Increment i_t by 1. Stop if $i_t > N_{it}$.
3. Pass the training data through the network. For each input vector, calculate the hidden unit outputs $O_p(j)$. If $i_t = 1$, i.e., in the first iteration, accumulate the cross- and auto-correlation $c(i, m)$ and $r(k, m)$ as in (8-9). Otherwise, if $i_t > 1$, use OR algorithm to change the desired output $t_p(i)$ to $t'_p(i)$ for each pattern and accumulate the cross-correlation $c(i, m)$ as in (8) with $t_p(i)$ replaced by $t'_p(i)$.
4. Using OWO, Solve linear equations for the output weights $w_{oi}(i, k)$ and $w_{oh}(j, k)$, and calculate E .
5. If E decreases, go to Step 8. Otherwise, if E increases, modify Z as $Z = 0.5 \cdot Z$, reload the previous best hidden weights and go to Step 8.
6. Make a second pass through the training data. Calculate the hidden weight changes using HWO with $t'_p(i)$ in place of $t_p(i)$ in (4).
7. Calculate the learning factor using the method described by Magoulas (Magoulas, Vrahatis, & Androulakis 1999).
8. Update the hidden unit weights.
9. Go to Step 2.

Theory

Theorem 1. The Output Reset (OR) algorithm leads to convergence of the error function E' .

This is clear from the facts that all three methods above satisfy condition (2).

In the following derivation, x denotes the input feature vector to be classified. The Bayes discriminant function is defined as $g_x(i) = P(C_i|x)$, which is the probability that the input pattern x belongs to the class C_i . An error function can be defined as

$$e \equiv 4b^2 \sum_{i=1}^{N_c} \int_S [F'_x(i) - g_x(i)]^2 p(x) dx \quad (23)$$

where S is the set of all possible input patterns. $F'_x(i)$ is defined as

$$F'_x(i) \equiv \frac{1}{2b} y'_p(i) + \frac{1}{2} \quad (24)$$

where $y'_p(i) = y_p(i) - a_p - d_p(i)$. Here $p(x)$ is the joint probability density function of the random input vector x . The error function E' is rewritten from equation (11) as

$$E' = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{N_c} [t_p(i) - y'_p(i)]^2 \quad (25)$$

Theorem 2. As N_v increases, the error function E' approaches e plus a constant, i.e.

$$\lim_{N_v \rightarrow \infty} E' = e + C \quad (26)$$

Proof. Taking the limit of $E'/4b^2$, as the number of input pattern is large,

$$\lim_{N_v \rightarrow \infty} \frac{E'}{4b^2} = \lim_{N_v \rightarrow \infty} \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{N_c} \left[\frac{1}{2b} (t_p(i) - y'_p(i)) \right]^2 \quad (27)$$

Using equation (24) and the definition of $t_p(i)$,

$$\begin{aligned} \lim_{N_v \rightarrow \infty} \frac{E'}{4b^2} &= \lim_{N_v \rightarrow \infty} \frac{1}{N_v} \left(\sum_{x \in S_1} [(1 - F'_x(1))]^2 \right. \\ &\quad + \sum_{j \neq 1} (0 - F'_x(j))^2 + \dots \\ &\quad + \sum_{x \in S_{N_c}} [(1 - F'_x(N_c))]^2 \\ &\quad \left. + \sum_{j \neq N_c} (0 - F'_x(j))^2 \right) \end{aligned}$$

where S_i denotes the set of input vectors x belonging to the i th class. By following the derivation of Ruck (Ruck *et al.* 1990) for the multiclass case, we obtain equation (26), with the constant C defined as

$$C = 4b^2 \sum_{i=1}^{N_c} \int_S g_x(i) (1 - g_x(i)) p(x) dx \quad (28)$$

From Theorem 1 and Theorem 2, we conclude that if the MLP is large enough, is trained properly and the training set is large enough, the neural net is a good approximation to a Bayes classifier.

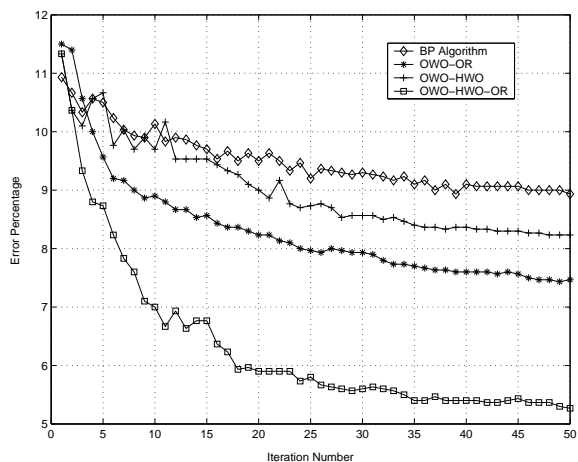


Figure 2: Training performances for handwritten data

Simulations and Discussions

In this section we compare the learning and the generalization capabilities of MLPs trained using BP, OWO-HWO, and OWO-HWO-OR. In the first iteration of BP, all of the output weights are initialized once by OWO in a linear output activation MLP network. Then sigmoid output layer activations are added before BP training. In the OWO-HWO algorithm, the weights in the hidden layers are dynamically updated by HWO while the output weights are also updated by OWO. While in the OWO-HWO-OR algorithm, we embedded the OR algorithm to the OWO-HWO to modify the desired output. Simulations are performed on the handprinted numeral data set used earlier, a remote sensing data set from an aerial survey.

Handprinted Data Set

The handprinted data set contains a training and a testing data set of the same size. In performing these experiments, the training data set is used for "training" and the testing data set is used to "testing". We first compare the "Learning" capabilities of the three algorithms. We run 50 iterations for each algorithm and plot their error percentages vs iteration number in Fig. 2. It is observed that the OWO-HWO-OR algorithm performs best. Algorithm 3 ranks second and the BP algorithm ranks last. The learning capability of the proposed algorithm is clearly illustrated.

In order to test the generalization capabilities of the three algorithms, we run each algorithm 150 times on training data with different random initial weights and apply the trained network to testing data. Generally, one could use 10-fold cross-validation with a statistical test (*t*-test) to examine the performance of an algorithm. However, since different initial MLP weights produce different classification errors after training, we can alternately evaluate a training algorithm by examining the classification error statistics for networks trained with different initial weights. Consequently, we plot the histograms of the training and testing results for all three algorithms, for several examples. In Fig. 3, the plots show 150 training results for the three algorithms with various ini-

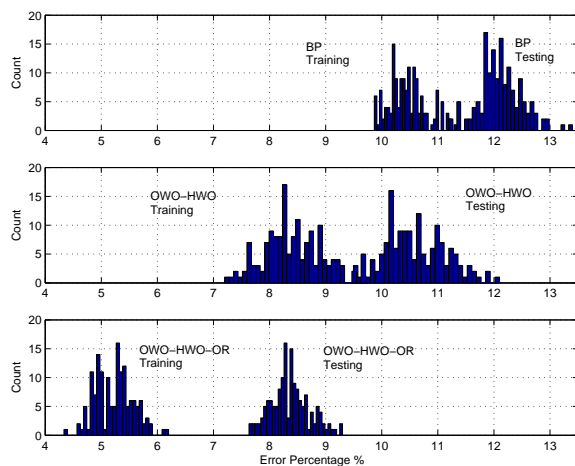


Figure 3: Training and testing performances comparison for handwritten data

Table 1: The Mean and Standard Deviation of Training and Testing Results of the Three Algorithms for Handprinted Numeral Data

Algorithm Applied	Training Error % mean(std)	Testing Error % mean(std)
BP	10.462(0.3334)	12.1218(0.3812)
OWO-HWO	8.4062(0.5028)	10.5987(0.5479)
OWO-HWO-OR	5.1227(0.3328)	8.3673(0.3429)

tial weights, and 150 testing results corresponding to the trained networks. The horizontal axis of each figure represents the classification error percentage, while the vertical axis corresponds to the number of times that the classification error percentage falls into a given interval. These results clearly demonstrate that the OWO-HWO-OR algorithm is the best one for both the training and testing cases. OWO-HWO ranks second and the BP algorithm ranks last. The proposed algorithm shows not only its powerful learning capability but also its excellent generalization potential for this complicated multiple class case. Table 1 lists the means and standard deviations of the training and testing results for each algorithm. Note that the OWO-HWO algorithm improves both the training and testing performances as compared to BP. The classification error means of BP are 10.462% and 12.1218% for training and testing respectively, while the counterparts for the OWO-HWO algorithm are 8.4062% and 10.5987%. Unfortunately, the error percent standard deviations increase for OWO-HWO. For example, the standard deviation for training increases from 0.3334% for BP to 0.5028% for OWO-HWO. By combining OWO-HWO with the OR algorithm, however, the means as well as the standard deviations of error percentages decrease, and the standard deviation for testing (0.3429%) is even lower than that of BP (0.3812%).

Aerial Survey Data Set

This data set is generated from an aerial survey and contains 12392 patterns. The survey images consist of two sets of black and white aerial images of the Dallas, Texas, area.

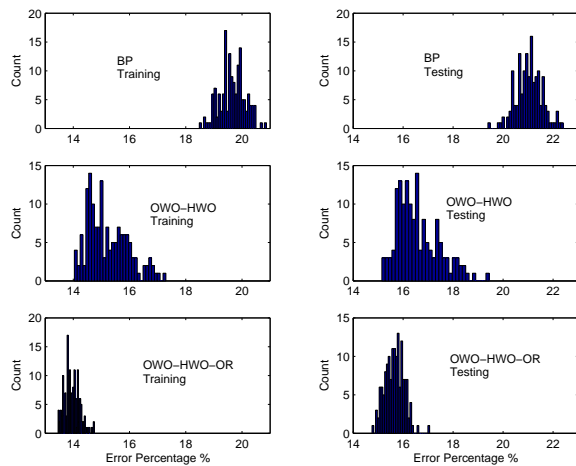


Figure 4: Training and Testing performances comparison for aerial survey data

Seventy-one images with various resolutions were used for this test. Regions of four land use/cover types were identified in the images: urban areas, fields or open grassy land, trees (forested land), and water such as lakes or rivers. Delineation and labelling of examples of each land use/cover type was done manually in softcopy. There are 18 features for each classification window (Bailey *et al.* 1993). The aerial survey data set is randomly split into two parts-the first part contains approximately two-thirds of the total patterns. The second part represents the other one-third of the population. In performing these experiments, the first part is used for "training" and the second part is used for "testing". Both parts of the data sets are chosen so that they represent the same relative proportion of the various classes as does the entire data set.

We again run each algorithm 150 times on the training data with different random initial weights and apply the trained network to the testing data. Fig. 4 shows the histograms of the classification error percentage of training and testing for all three algorithms. The left column of the figure presents the training histogram with 150 different initial weight sets. The right column lists the testing results corresponding to the trained network on the left. It is the same as before that OWO-HWO outperforms BP and OWO-HWO-OR is the best among the three algorithms. It is also noted that the standard deviation of the training and testing results for the OWO-HWO-OR is the smallest.

Conclusion

In this paper, we propose some basic improvements to the objective function used in training FLNN and MLP classifiers. The method is analyzed in detail. The classifier design equations have been solved using the numerically stable OWO and HWO approaches. The OR algorithm, which is a type of relaxation algorithm, has been detailed. It allows us to iteratively improve our initial solution to the design equations. In this paper, the OR approach is used to design MLP classifiers. It can also be used to improve the

performance of FLNNs and other classifiers having weights connected to the output nodes. Compared to conventionally-designed MLPs, the OR-designed networks have proved superior for complicated classification problems. Additional details of the OR algorithm are discussed in a journal version of the paper, which is currently under preparation.

References

- Bailey, R. R.; Pettit, E. J.; Borochoff, R. T.; Manry, M. T.; and Jiang, X. 1993. Automatic recognition of USGS land use/cover categories using statistical and neural network classifiers. *Proceeding of SPIE OE/Aerospace and remote sensing* 12–16.
- Barton, S. 1991. A matrix method for optimizing a neural network. *Neural Computation* 3(3):450–459.
- Dawson, M. S.; Olvera, J.; Fung, A.; and Manry, M. 1992. Inversion of surface parameters using fast learning neural networks. *Proceeding of IGARSS '92* 11:910–912.
- Gong, W.; Yau, H. C.; and Manry, M. T. 1994. Non-gaussian feature analyses using a neural network. *Progress in Neural Networks* 2:253–269.
- Magoulas, G. D.; Vrahatis, M. N.; and Androulakis, G. S. 1999. Improving the convergence of the backpropagation algorithm using learning adaptation methods. *Neural Computation* 11:1769–1796.
- Manry, M. T.; Guan, X.; Apollo, S. J.; Allen, L. S.; Lyle, W. D.; and Gong, W. 1992. Output weight optimization for the multilayer perceptron. *Conference Record of the Twenty-Sixth Annual Asilomar Conference on Signals, Systems, and Computers* 1:502–506.
- Pao, Y. H. 1989. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley.
- Ruck, D. W.; Rogers, S. K.; Kabrisky, M.; Oxley, M. E.; and Suter, B. W. 1990. The multilayer perceptron as an approximation to a Bayes optimal discriminate function. *IEEE Trans. on Neural Network* 1(4):296–298.
- Rumelhart, D. E., and McClelland, J. L. 1988. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press. Foundations.
- Sartori, M., and Antsaklis, P. J. 1991. A simple method to derive bounds on the size and to train multilayer neural networks. *IEEE Transactions on Neural Networks* 2(4):467–471.
- Scalero, R. S., and Tepedelenlioglu, N. 1992. A fast new algorithm for training feedforward neural networks. *IEEE Transactions on Signal Processing* 40(1):202–210.
- Wan, E. A. 1990. Neural network classification: A bayesian interpretation. *IEEE Trans. on Neural Network* 1(4):303–305.
- Yu, C., and Manry, M. T. 2002. A modified hidden weight optimization algorithm for feed-forward neural networks. *Thirty-Sixth Asilomar Conference on Signals, Systems & Computers* 1:1034–1038.