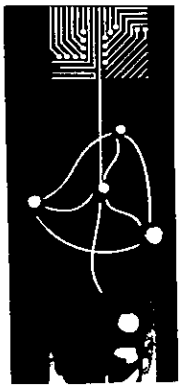

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

OPTIMAL ESTIMATE TRAINING 2 (OET-2)

Alexandre Pinto Alves da Silva[1]
Victor Hugo Quintana[2]
Germano Lambert Torres[1]

1 - ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ
2 - UNIVERSITY OF WATERLOO - CANADA

Abstract - A supervised learning algorithm for multilayer perceptrons (Optimal Estimate Training 2 - OET2) has been developed to overcome shortcomings of the generalized delta rule with backward error propagation. OET2 is particularly useful for real-world pattern recognition problems where very large training sets are necessary.

1. INTRODUCTION

The majority of the applications of ANNs have employed semi-linear feed-forward nets (multilayer perceptrons), using the generalized delta rule with backward error propagation for the learning process [1]. However, it is almost impossible to treat real-world problems utilizing this learning rule. The principal shortcomings of the generalized delta rule with backpropagation of error are as follows:

- it generally takes a long time to converge;
- it does not scale well (usually the training time grows exponentially as the number of neurons increase);
- it requires a choice of good parameters for the learning process (learning rate, constant of the momentum term and slope of the activation function); and
- as a steepest-descent method it could stop at a local minimum.

Despite many attempts to improve the performance of the generalized delta rule with backward error propagation [2, 3], and to find an alternative to it [4, 5], there is still no efficient and reliable method to train a multilayer perceptron. The main problem is the high nonlinearity of the error function.

The Functional Link Net (FLN) [6], a high-order net, has been tried with more success than a multilayer architecture. The basic idea is to avoid the necessity of hidden layers by performing a nonlinear transformation of the input pattern (enhancement of the original input pattern) before it is supplied to the input layer of the network. Without hidden layers, the learning process can be pursued using the delta rule instead of the generalized delta rule. Consequently, the FLN takes less time to converge.

Assuming that the input pattern enhancement is done properly, it has been shown that the FLN can also achieve more accurate results. However, besides the fact that the number of new inputs can increase considerably, depending on the number of original inputs and independent training patterns, it is not an easy task to define a general procedure that will guarantee the effectiveness and efficiency of the input pattern enhancement.

In order to overcome the problems mentioned above, a new method for loading information into a multilayer perceptron is proposed in this paper. The idea is to use successive quadratic approximations for the error function, until getting into the proximity of the global minimum.

2. A SUPERVISED LEARNING TECHNIQUE FOR VERY LARGE TRAINING SETS

Reference [7] presents a comparison study between Optimal Estimate Training (OET) and the generalized delta rule with backward error propagation. The task is to restore a data bit stream corrupted by a diffusive point-spread function. The results obtained are an indication that OET can be much faster and much more accurate than the other technique.

OET is a supervised learning technique for multilayer perceptrons. The sets of input and output patterns are represented in matrix form. An optimum nonlinear mapping that associates input training patterns to output training patterns is found by successively solving linear systems using direct methods, which

in fact is a procedure based on nonlinear least-squares methods (local approximation of the error function by a quadratic function, and the exact minimization of such an approximate function). The Moore-Penrose pseudoinverse is solved explicitly in different steps of the OET. An important feature of this technique is that the scaling problem is very well characterized. Based on the complexity of the operations involved ($O[r.m^2]$, where r is the number of input training patterns and m their dimension), it is easy to estimate how the OET learning rate will scale with network size.

3. OPTIMAL ESTIMATE TRAINING

The general idea of OET can be summarized using Figure 1. In Figure 1, $h_{in,j}$ (j = 0, 1, ..., m) is the set of input variables ($h_{in,0}$ represents an additional input with constant value equal to 1); w_{ji} is the set of interconnection weights leading from element j to element i, and $h_{out,i}$ represents the output of neuron i.

As usual, the propagation of the inputs is produced by a summation of the weighted inputs, i.e.,

$$z_{out,i} = \sum_{j=0}^{m} h_{in,j} w_{ji} \qquad (1)$$

The value of $h_{out,i}$ is calculated by the application of an activation function,

$$h_{out,i} = f(z_{out,i}) \qquad (2)$$

To perform the OET, a desired output value, $h_{d\text{-}out,i}$, is propagated back through the activation function and the corresponding desired intermediate result, $z_{d\text{-}out,i}$, is obtained. To execute this operation, it is necessary to employ a continuous invertible function such as the hyperbolic tangent. This function is suitable for the OET since the desired outputs can be positive or negative. Afterwards, the set of inputs and desired intermediate results can be interrelated to calculate optimal estimates for the set of interconnection weights, in a least-squares sense.

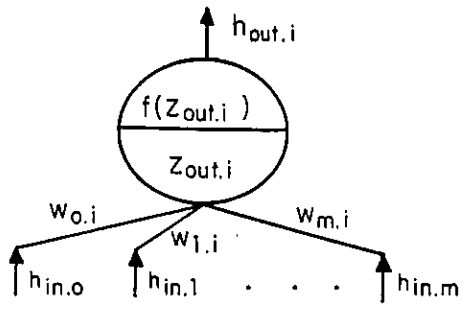

Figure 1: Representation of a neuron.

The procedure for training a network with one hidden layer (Figure 2) is now described. The extension of this training procedure for more than one hidden layer is straightforward.

The following notation is used in this paper:

$H_{d\text{-}out}$ → r by n desired output matrix;

H_{in} → r by m+1 input matrix;

$Z_{d\text{-}out}$ → $H_{d\text{-}out}$ propagated back through the output layer (r by n matrix); the i^{th} column of $Z_{d\text{-}out}$ is denoted by $\underline{z_{d\text{-}out}}^{i}$, and the j^{th} column of $Z_{d\text{-}out}^{t}$ is denoted by $\underline{z_{d\text{-}out}}^{j}$;

W_{out} → m+1 by n matrix of interconnection weights linking the hidden layer with the output layer; the i^{th} column of W_{out} is denoted by $\underline{w_{out}}^{i}$;

$H_{d\text{-}hid}$ → r by m+1 matrix; the j^{th} column of $H_{d\text{-}hid}^{t}$ is denoted by $h_{d\text{-}hid}^{j}$;

$Z_{d\text{-}hid}$ → $H_{d\text{-}hid}$ propagated back through the hidden layer (r by m+1 matrix); the k^{th} column of $Z_{d\text{-}hid}$ is denoted by $\underline{z_{d\text{-}hid}}^{k}$;

W_{hid} → m+1 by m+1 matrix of interconnection weights linking the input layer with the hidden layer; the k^{th} column of W_{hid} is denoted by $\underline{w_{hid}}^{k}$;

Z_{hid} → produced by multiplying H_{in} and W_{hid} (r by m+1 matrix);

H_{hid} → Z_{hid} propagated forward through the hidden layer (r by m+1 matrix);

Z_{out} → produced by multiplying H_{hid} and W_{out} (r by n matrix);

H_{out} → Z_{out} propagated forward through the output layer (r by n matrix);

r → number of input/output patterns in the training set;

m+1 → number of input neurons (which is the same as the number of neurons in the hidden layer); and

n → number of output neurons.

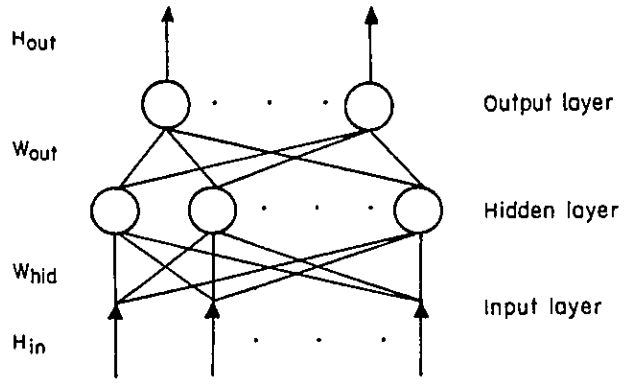

Figure 2: Fully connected feed-forward net with one hidden layer (input layer processing elements are data conduits)

The network training steps are [7]:

1) Calculate an initial value for W_{out} (this would be the final value for W_{out} if hidden layers were not used):

$$W_{out} = H_{in}^{-} Z_{d-out} \quad (3)$$

where

$$Z_{d-out} = \tanh^{-1}(H_{d-out}),$$

and

$$H_{in}^{-} = (H_{in}^{t} H_{in})^{-1} H_{in}^{t},$$

assuming that r is greater than m+1 (overdetermined system). The symbol + represents the generalized inverse or Moore-Penrose pseudoinverse.

2) Obtain H_{d-hid} that would produce Z_{d-out} given W_{out},

• if n = m+1,

$$H_{d-hid}^{t} = (W_{out}^{t})^{-1} Z_{d-out}^{t}, \quad (4)$$

assuming that W_{out}^{t} is non-singular;

• if n > m+1,

$$H_{d-hid}^{t} = (W_{out}^{t})^{+} Z_{d-out}^{t}, \quad (5)$$

$$(W_{out}^{t})^{+} = (W_{out} W_{out}^{t})^{-1} W_{out}, \quad (6)$$

overdetermined case;

and

• if n < m+1,

$$H_{d-hid}^{t} = (W_{out}^{t})^{+} Z_{d-out}^{t}, \quad (8)$$

$$(W_{out}^{t})^{+} = W_{out} (W_{out}^{t} W_{out})^{-1}, \quad (9)$$

underdetermined case.

3) Normalize the elements of H_{d-hid} to ensure that they will be situated within the output range of the hidden layer's activation functions,

$$H_{d-hid} = H_{d-hid} 0.99 / |\lambda|, \quad (10)$$

where λ is the element of H_{d-hid} with the largest magnitude. A multiplication factor, say 0.99, is used to obtain finite values for Z_{d-hid}, and to get advantage of the hyperbolic tangent nonlinearity.

4) Produce Z_{d-hid} using the inverse hyperbolic tangent,

$$Z_{d-hid} = \tanh^{-1}(H_{d-hid}) \quad (11)$$

5) Given H_{in} and Z_{d-hid}, calculate W_{hid},

$$W_{hid} = H_{in}^{-} Z_{d-hid} \quad (12)$$

6) Obtain the actual intermediate output for the hidden layer,

$$Z_{hid} = H_{in} W_{hid}, \quad (13)$$

and

$$H_{hid} = \tanh(Z_{hid}) \quad (14)$$

7) Recalculate W_{out},

$$W_{out} = H_{hid}^{-} Z_{d-out}, \quad (15)$$

where

$$H_{hid}^{-} = (H_{hid}^{t} H_{hid})^{-1} H_{hid}^{t}. \quad (16)$$

The network training is finished at step 7, i.e., once the matrices of interconnection weights W_{hid} and W_{out} are optimally calculated in the least-squares sense.

To obtain the ANN output for any set of inputs, the following steps are performed:

1) Combine H_{in} and W_{hid} to produce Z_{hid},

$$Z_{hid} = H_{in} W_{hid} \quad (17)$$

2) Treat Z_{hid} with the activation function (hyperbolic tangent),

$$H_{hid} = \tanh(Z_{hid}) \quad (18)$$

3) Combine H_{hid} and W_{out} to produce Z_{out},

$$Z_{out} = H_{hid} W_{out} \quad (19)$$

4) Treat Z_{out} with the activation function to generate H_{out} (the output of the network),

$$H_{out} = k \tanh(Z_{out}), \quad (20)$$

where k = 1.0101 (output normalization factor (1/0.99)).

4. OPTIMAL ESTIMATE TRAINING 2 (OET2)

In this section, the OET algorithm is extended in order to overcome its major drawbacks. Three new features are incorporated to the OET algorithm: numerical stability, input enhancement, and bounded interconnection weights.

4.1 Numerical Stability

Although the OET gives very promising results, some numerical problems with the matrix inversions of equations (4) and (16) are reported in [7]. These problems should have been expected because the conventional least-squares solution via normal equation is intrinsically prone to ill-conditioning problems. It can be shown that the condition number $c(H^t H)$ is the square of $c(H)$. This means that if H is not very well conditioned, $H^t H$ is ill-conditioned, and this may badly affect the final results, without any warning.

In order to improve the accuracy of the Shepanski's learning process, a QR (orthogonal) decomposition method could be used. There are several ways to define an orthogonal transformation. Since the patterns to be used in the training process of an ANN may be encountered sequentially, the Givens transformation is potentially suitable because it works by rows. With respect to this aspect, the Givens method is a better choice than the sequential algorithms based on the Kalman filter, which are also sensitive to numerical problems. An iterative method, such as the Preconditioned Conjugate Gradient (PCG), could be used to overcome ill-conditioning problems in the least-squares solution via a normal equation. However, the PCG method becomes more efficient than direct methods only when $H^t H$ has a sparse structure and a very large dimension (say, $> 10^3$), which is not usually the case since H_{in} is generally a dense matrix.

It is important to be aware of the cases in which H_{in} is not assumed to be sufficiently well conditioned. It could occur that during the computation, perturbations caused in H_{in} due to round-off errors replace H_{in} by a rank-deficient matrix. Notice that with OET there is a restriction related to the columns of the input matrix. If the values associated to a certain input channel can be obtained (approximately) as a linear combination of the corresponding values of other input channels, H_{in} will be rank-deficient ($H_{in}^t H_{in}$ singular). This arbitrary instability can be avoided using a stabilization method to detect the set of linearly independent columns [8]. A new input matrix can be formed by considering only the linearly independent columns (redundant information is discarded). A stabilization method shall also be applied in case of rank deficiency in W_{out}^t and/or H_{hid}, which must be treated as being of full rank.

4.2 Input Enhancement

A major limitation in the Shepanski's training technique is the necessity of having the same number of processing elements in the input and hidden layers. To overcome this shortcoming, an enhanced input representation scheme can be applied when it is necessary to increase the number of neurons in the hidden layer. Using the idea of facilitating the learning process by increasing the extent of nonlinearity, with a nonlinear transformation of the input pattern [6], the problem is solved in a better way than just including "dummy" input channels.

A strategy for input enhancement, similar to the one suggested in [9], is adopted in this work. The values of the most statistically correlated pairs of input channels, considering only the ones contained in the new full rank input matrix, are multiplied to form new entries. The appearance of a specific input in several products is not permitted. Limited diversity of information in noisy environments is not a safety policy. With this proposed technique, the necessary increase of the enhanced inputs is expected to be less than using the FLN because there is a corresponding increase in the number of hidden neurons.

4.3 Bounded Interconnection Weights

Interconnection weight bounds can be applied when it would be necessary to decrease the number of neurons in the hidden layer. Using OET, the number of neurons in the hidden layer cannot be decreased without reducing the dimensionality of the input patterns. An improvement on the ANN's generalization capability, produced by decreasing the number of hidden units (and, consequently, decreasing the number of interconnections), is expected to be obtained alternatively by reducing its interconnection weight bounds. This can be intuitively understood by noting that when weight bounds are applied, the total squared sum of errors can be controlled, allowing the training procedure to appropriately constrain the decision regions created by the network.

In fact, when an ANN is trained for classification or function approximation purposes, it is necessary to avoid the training data overfitting in order to obtain good generalization. This can be achieved by making a search for the simplest ANN (the one with less neurons and interconnections) which provides the minimum error rate for the testing set being used. When training with an iterative procedure like backpropagation, an alternative to avoid an exhaustive search for a good network architecture is to use cross-validation [10]. The idea is to interrupt the learning process as soon as the generalization for the test set begins to deteriorate.

"Easy learning" should also be avoided, i.e., when an input-output mapping can be established by assigning large weights (in comparative terms) to a small percentage of the total number of interconnections (e.g., when some features strongly characterize certain classes). Without noticing that "easy learning" is happening, one could be induced to reduce the ANN architecture. Low knowledge encoding redundancy caused by "easy learning" is highly undesirable in very noisy environments. Once more, interconnection weight bounds can be used to avoid "easy learning" and to get the benefits of knowledge encoding redundancy. A more uniform distribution of knowledge across the ANN is obtained by limiting the interconnection weight magnitudes.

The combination of input enhancement and bounded interconnection weights is not employed in this work because of the associated increase in the computational burden. However, this is a possibility that deserves investigation.

4.4 Proposed Training Algorithm

The proposed procedure for training a feed-forward network with one hidden layer is presented next. The encoding algorithm steps are as follows:

1) Determine an input matrix H_{in} (r X m),formed by m linearly independent input pattern features of the original H_{in} matrix.

2) Calculate an initial W_{out}:

$$H_{in} W_{out} = Z_{d-out} \tag{21}$$

$$\min_{\underline{w}^i_{out}} \left\| H_{in}\, \underline{w}^i_{out} - \underline{z}^i_{d-out} \right\|_2, \quad \text{for } i = 1, \ldots, n \tag{22}$$

assuming r greater than m.

3) Given W_{out}, obtain $H_{d\text{-}hid}$ that produces $Z_{d\text{-}out}$:

$$W^t_{out} H^t_{d-out} = Z^t_{d-out} \tag{23}$$

- if n ≥ m then:

$$\min_{\underline{h}^j_{d-hid}} \left\| W^t_{out}\, \underline{h}^j_{d-hid\,t} - \underline{z}^j_{d-out\,t} \right\|_2, \quad \text{for } j = 1, \ldots, r \tag{24}$$

- if n < m then:

$$\min \left\| \underline{h}^j_{d-hid\,t} \right\|_2, \ \text{s.t.,}\ W^t_{out}\, \underline{h}^j_{d-hid\,t} = \underline{z}^j_{d-out\,t}, \quad \text{for } j = 1, \ldots, r \tag{25}$$

4) Normalize the elements of $H_{d\text{-}hid}$ to assure that they will be located within the output range of the hidden layer's activation functions (hyperbolic tangent),

$$H_{d-hid} = H_{d-hid}\, 0.99 / |\lambda| \tag{26}$$

where λ is the element of $H_{d\text{-}hid}$ with the largest magnitude.

5) Calculate $Z_{d\text{-}hid}$ using the inverse hyperbolic tangent,

$$Z_{d-hid} = \tanh^{-1}(H_{d-hid}) \tag{27}$$

6) Given H_{in} and $Z_{d\text{-}hid}$, calculate W_{hid},

$$H_{in} W_{hid} = Z_{d-hid} \tag{28}$$

$$\min_{\underline{w}^k_{hid}} \left\| H_{in}\, \underline{w}^k_{hid} - \underline{z}^k_{d-hid} \right\|_2, \ \text{s.t.,}\ \left| w^k_{hid,l} \right| \le Bound1, \quad \text{for } k = 1, \ldots, m \text{ and } l = 1, \ldots, m, \tag{29}$$

where $w_{hid.l}{}^k$ is a compoment of $\underline{w}_{hid}{}^k$.

7) Obtain the actual intermediate output for the hidden layer,

$$Z_{hid} = H_{in} W_{hid}, \tag{30}$$

and

$$H_{hid} = \tanh(Z_{hid}) \tag{31}$$

8) Recalculate W_{out},

$$H_{hid} W_{out} = Z_{d-out} \tag{32}$$

$$\min_{\underline{w}^i_{out}} \left\| H_{hid}\, \underline{w}^i_{out} - \underline{z}^i_{d-out} \right\|_2, \ \text{s.t.,}\ \left| w^i_{out,l} \right| \le Bound2, \quad \text{for } i = 1, \ldots, n \text{ and } l = 1, \ldots, m. \tag{33}$$

where $w_{out.l}{}^i$ is a component of $\underline{w}_{out}{}^i$.

4.5 Finding Bounds

To find a good pair of bounds the following heuristic procedure is employed:

1) Train the ANN (one hidden layer) with no bounds. This allows maximum network capacity for representing nonlinear mappings.

2) Keep Bound2 set to a very large number. Set Bound1 approximately to the maximum value that produces a solution for W_{hid} which has all elements equal to this value.

3) Improve the ANN capacity of producing nonlinear mappings by gradually raising Bound1, while leaving Bound2 set to a very large number.

a) If this procedure immediately increases the error rate for the test set, then try an ANN without hidden layers. Otherwise, go to b).

b) Continue to raise Bound1 until the error rate for the test set starts to increase. Then, go to step 4. If the ANN generalization ability keeps improving until returning to the situation described in step 1, then go to c).

c) Apply an input enhancement scheme to increase the extent of nonlinearity allowed by the ANN.

4) Keep the best Bound1 value, and lower Bound2 to check if the capacity for representing nonlinearities is greater than necessary. Lower Bound2 until the error rate starts to increase again. The training process is finished.

Numerically stable methods for solving the linear least-squares problems (22), (24), (25), (29) and (33) are described in [8].

CONCLUSIONS

This paper has presented a learning algorithm, Optimal Estimate Training 2 (OET2), for multilayer perceptrons. OET2 is very appropriate for large training sets. Its execution time varies linearly with the number of patterns and quadratically with the dimension of the input patterns. OET2 does not require any choice of learning parameters. OET2 has proved to be orders of magnitude faster than the backpropagation algorithm in practical applications [11-14].

The following topics deserve further research effort:

- Inclusion of adaptation capacity in the OET2 algorithm. Row weighting in the orthogonal decomposition of the input matrix (H_{in}) can be used. The efect of past data can be reduced by gradually increasing the weighting as new data are acquired.

- Application of column weighting in the orthogonal decomposition of the input matrix (H_{in}) during the ANN training. The inverses of the variances of the input variables can be used as weighting factors. This procedure can improve the ANN generalization ability.

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams: "Learning internal representations by error propagation in parallel distributed processing", in *Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, 1986, pp. 318-362.

[2] L.-W. Chan and F. Fallside: "An adaptive training algorithm for back propagation networks", *Computer Speech and Language*, Vol. 2, 1987, pp. 205-218.

[3] R.A. Jacobs: "Increased rates of convergence through learning rate adaptation", *Neural Networks*, Vol. 1, 1988, pp. 295-307.

[4] S.E. Fahlman and C. Lebiere: "The cascade-correlation learning architecture", in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky (Ed.), Morgan Kaufmann, 1990, pp. 524-532.

[5] W.T. Miller III, F.H. Glanz, and L.G. Kraft III: "CMAC: an associative neural network alternative to backpropagation", *Proceedings of the IEEE*, Vol. 78, Oct. 1990, pp. 1561-1567.

[6] Y.-H. Pao: *Adaptive Pattern Recognition and Neural Networks*, Addison Wesley, 1989.

[7] J.F. Shepanski: "Fast learning in artificial neural systems: multilayer perceptron training using optimal estimation", *Proc. IEEE 2nd Intern. Conf. Neural Nets*, San Diego, Jul. 1988, Vol. 1, pp. 465-472.

[8] C.L. Lawson and R.J. Hanson: *Solving Least Squares Problems*, Prentice-Hall, 1974.

[9] Y.-H. Pao and D.J. Sobajic: "Autonomous feature discovery for critical clearing time assessment", *Proc. 1st Symp. Expert Syst. Applications Power Syst.*, Stockholm-Helsinki, Aug. 1988.

[10] G.E. Hinton: "Using cross-validation to avoid overfitting", presented at *Neural Network Workshop*, Niagara-on-the-Lake, May 1990.

[11] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang: "Solving data acquisition and processing problems in power systems using a pattern analysis approach", *IEE Proceedings Part C: Generation, Transmission and Distribution*, Vol. 138, July 1991, pp. 365-376.

[12] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang: "Neural networks for topology determination of power systems", *Proc. 1st International Forum on Applications of Neural Networks to Power Systems* (IEEE Catalog Number: 91TH0374-9), Seattle, July 1991, pp. 297-301.

[13] A.P. Alves da Silva, A.M. Leite da Silva, J.C.S. de Souza, and M.B. Do Coutto Filho: "State forecasting based on artificial neural networks", *Proc. 11th Power Systems Computation Conference*, Avignon, August 1993, pp. 461-467.

[14] L.E. Borges da Silva, A.P. Alves da Silva, G. Lambert Torres, and G. Roy: "An alternative neural network training algorithm for rreal time control", *Proc. IEEE International Symposium on Industrial Electronics*, Santiago, May 1994, pp. 72-76.