Two-phase Backpropagation

George M. Georgiou **Cris Koutsougeras**
Center for Automation and Autonomous Complex Systems
Computer Science Department, Tulane University
New Orleans, LA 70118

June 5, 1991

Abstract

A Two-phase Backpropagation algorithm is presented. In the first phase the directions of the weight vectors of the first (and possibly the only) hidden layer are constrained to remain in directions suitably chosen by pattern recognition, data compression, or speech and image processing techniques. Then, the constraints are removed and the standard Backpropagation algorithm takes over to further minimize the error function. The first phase swiftly situates the weight vectors in a good position (relatively low error) which can serve as the initialization of the standard Backpropagation. The generality of its application, its simplicity, and the shorter training time it requires, make this approach attractive.

1 Introduction

Backpropagation (BP) (Werbos, 1974; Rumelhart et al., 1986) has been established as one of the most common techniques to train feed-forward neural networks that perform a variety of tasks. Being a general-purpose algorithm, BP does not make explicit use of any information which could be extracted from the training set by traditional methods, e.g. Principal Components. Instead of expecting BP to discover or find substitutes for such knowledge, one could embed it in the BP algorithm to begin with. Intuitively, this will save time and improve the convergence properties of the algorithm. The Two-phase BP algorithm of this paper incorporates in BP information which is in the form of directions of hyperplanes in the input space.

Usually the techniques for solving problems in areas like pattern classification, data compression, and speech and image processing, call for finding certain directions in the input space, such as discriminant vectors (Duda and Hart, 1973), Principal Components, and constraint or oriented Principal Components (Kung et al., 1990), upon which the data vectors are to be projected for further analysis. In this paper we collectively call these directions *important directions.* The BP algorithm has been successfully used to solve many of the same problems. The weight vectors of the units in the first hidden layer of a trained BP network can be thought as being the important directions upon which the data vectors are projected. For example, when a BP network is used as a classifier, the weight vectors of the first hidden layer assume the role of the linear discriminants in traditional classifiers. There is evidence that sometimes the important directions found by BP are closely related to those found by traditional techniques. In (Cottrell and Baldi, 1988), for example, it is found that the weight vectors of the (single) hidden layer span the same subspace as the Principal Components corresponding to the largest eigenvalues.

2 The Two-phase Backpropagation Algorithm

The object of the algorithm is to train a feedforward network which consists of one or more hidden layers. Phase 1 of the algorithm first calls for the calculation of the k important directions, using an appropriate method, e.g. Principal Components for image compression (Gonzales and Wintz, 1977) and Fisher's Linear Discriminants[1] for multiple classes classification (Johnson and Wichern, 1988). These directions are denoted by the normalized vectors $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$. Each column vector $\mathbf{d}_j = (d_{j1}, d_{j2}, \ldots, d_{jn})^t$ represents a hyperplane of dimension $n-1$ passing through the origin of the n-dimensional input space.

All weights and thresholds in the network are updated as in the standard BP (Rumelhart et al., 1986), except the weights w_{ji} in the first hidden layer which are updated in a special way: Denoting the weight vector of unit j, $1 \leq j \leq k$, in the first hidden layer by $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jn})^t$, we require that $\mathbf{w}_j$ remains in the direction of $\mathbf{d}_j$. In order to achieve this, weight w_{ji} is written as

$$w_{ji} = m_j \, d_{ji}, \quad \text{where } m_j \text{ is a scalar variable.} \tag{1}$$

Next, we treat the m_j's as free variables, instead of the w_{ji}'s, and we update the m_j's according to a steepest gradient descent rule, instead of the w_{ji}'s. Keeping each $\boldsymbol{d}_j$ constant during training, indeed $\mathbf{w}_j$ remains in the direction of $\mathbf{d}_j$.

To find the update rule Δm_j, we follow the notation and method of (Rumelhart et al., 1986). For input pattern $\mathbf{x}_p$ we have

$$\Delta_p m_j = -\eta \, \frac{\partial E_p}{\partial m_j} = -\eta \, \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial m_j}$$

Also,

$$\delta_{pj} \overset{\text{def}}{=} -\frac{\partial E_p}{\partial net_{pj}} = f'(net_{pj}) \sum_k \delta_{pk} w_{kj}$$

The remaining factor is calculated as follows,

$$\frac{\partial net_{pj}}{\partial m_j} = \frac{\partial(\sum_i w_{ji} x_{pi} + \theta_j)}{\partial m_j} = \frac{\partial(m_j \sum_i d_{ji} x_{pi} + \theta_j)}{\partial m_j} = \sum_i d_{ji} x_{pi}$$

Now the update rule for m_j is complete:

$$\Delta_p m_j = \eta \, \delta_{pj} \sum_i d_{ji} x_{pi} = \eta f'(net_{pj}) \sum_k \delta_{pk} w_{kj} \sum_i d_{ji} x_{pi} \tag{2}$$

Where η is the learning rate, θ_j is the threshold, and f is the activation function. The subscript k runs over the units of the next layer. This update rule can be extended to include a momentum term (Rumelhart et al., 1986).

In Phase 2, the w_{ji}'s, as well as all the other weights and thresholds, are updated according the standard BP.

Summarizing, the Two-phase BP algorithm can be described with the following steps:

1. Calculate the k normalized important directions $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$.

2. Initialize weights. Assign random values to the weights (except the ones in the first hidden layer), thresholds, and m_j's.

[1]These are an example of a non-orthogonal set of important directions.

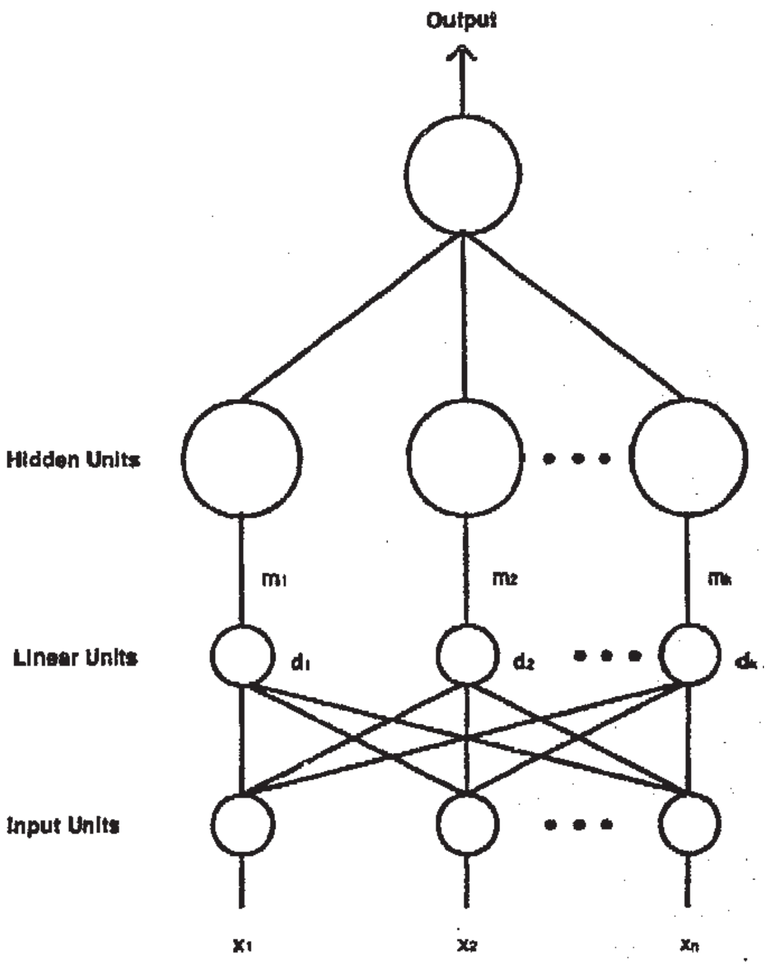

Figure 1: An equivalent network during Phase 1

3. Begin updating the weights and thresholds of the net as in rum, except the weights w_{ji} of the first hidden layer which are updated according to (1) and (2). Training is continued until no significant reduction in the error level is made in a certain number of epochs. The exact criterion is open to investigation.

4. Using the weights and thresholds at the conclusion of step 3 as the initial weights, train the network using the standard BP.

Steps 1, 2, and 3 comprise Phase 1, and step 4 Phase 2. An equivalent network for Phase 1 is shown in Figure 1. For simplicity, no thresholds (in the hidden and output units), only one output unit, and only one hidden layer are shown. The weight vectors of the linear units are frozen to $\boldsymbol{d_1}, \boldsymbol{d_2}, \ldots, \boldsymbol{d_k}$, while the m_j's are adjusted. For an input vector $\boldsymbol{x_p}$, the output of the linear unit j is simply the inner product $\sum_i^k d_{ji}x_{pi} = \boldsymbol{d_j} \cdot \boldsymbol{x_p}$, which, not surprisingly, is the last term of (2). The output of the linear units can saved in memory so that the inner products are not recalculated in each epoch. In Phase 2 each hidden unit subsumes the corresponding linear unit: The hidden unit j has weights $w_{ji}, 1 \leq j \leq k$ and $1 \leq i \leq n$, which are directly connected to the input units.

3 Neural Networks that extract Principal Components and related directions

The traditional methods for finding important directions usually involve information in the form of means, covariance matrices, and eigenvectors. The use of such information in the neural net context may be criticized on the grounds that neurons use information which is not available to them through a connection. Recently single-layer neural networks have been developed that extract important directions, most notably the Principal Components, in an iterative and unsupervised manner (Oja, 1989; Sanger, 1989; Kung et al., 1990). The basis of these networks is the single neuron model introduced by Oja (Oja, 1982) which extracts the (normalized) principal component (the one that corresponds to the largest eigenvalue) of a stationary random process. For practical purposes the random process is approximated by a finite training set which is repeatedly presented as input to the neuron during training.

Once the single-layer net is trained (unsupervised), i.e. the important directions are found, with the appropriate connections, it can serve as the first hidden layer of a feed-forward network to be further trained by the Two-phase algorithm (supervised). The Two-phase algorithm will try, and succeed as far as our experiments suggest, to efficiently find a solution in which the directions of the final weight vectors in the first hidden layer will be close to their corresponding important directions. Thus, the Two-phase algorithm provides a general way for supervised training of neural networks that recruit unsupervised trained networks.

4 An Example

This example is a binary classification problem. The data are classes II (Iris setosa) and III (Iris verginica) from the well known Fisher's Iris data set (Fisher, 1936). Each class has 50 real-valued 4-dimensional data vectors. Since this a classification problem, we used a discriminant technique, namely that of Foley and Sammon (Foley and Sammon, Jr., 1975), to find discriminant vectors. Figure 2 shows typical[2] trajectories of the error function versus epochs for the Two-phase algorithm (Phase 1 and Phase 2) and the standard BP. One hidden layer with two hidden units (corresponding to two discriminant vectors) was used. The same learning rate (in the range 0.05-0.1) and momentum constant (in the range 0.1-0.3) were used for all the weights, thresholds, and m_j's in each run. At the moment of switching to the standard BP, that is the beginning of Phase 2, the error consistently dropped abruptly to a low level, significantly outperforming the regular BP. No disruption of the training has been observed at the moment of switching. Figure 3 shows the trajectories of Phase 2 had it started at different epochs.

Switching to Phase 2 when the error dropped less than 0.4 in the last 5 epochs (usually happened between epoch 40 and 50) and using the formula

$$\phi = \arccos\left(\frac{\mathbf{w}_i \cdot \mathbf{w}_f}{|\mathbf{w}_i| \cdot |\mathbf{w}_f|}\right)$$

it was found that, on the average, the direction of the final weight vector $\mathbf{w}_f$ in Phase 2, deviated only less than $\phi = 10$ degrees from the direction of the initial weight vector $\mathbf{w}_i$ which is parallel to its corresponding discriminant vector.

[2]Based on more than 100 runs with frequent changes in the learning rate and momentum. For comparison, Phase 1 is shown to continue even after Phase 2 has started.

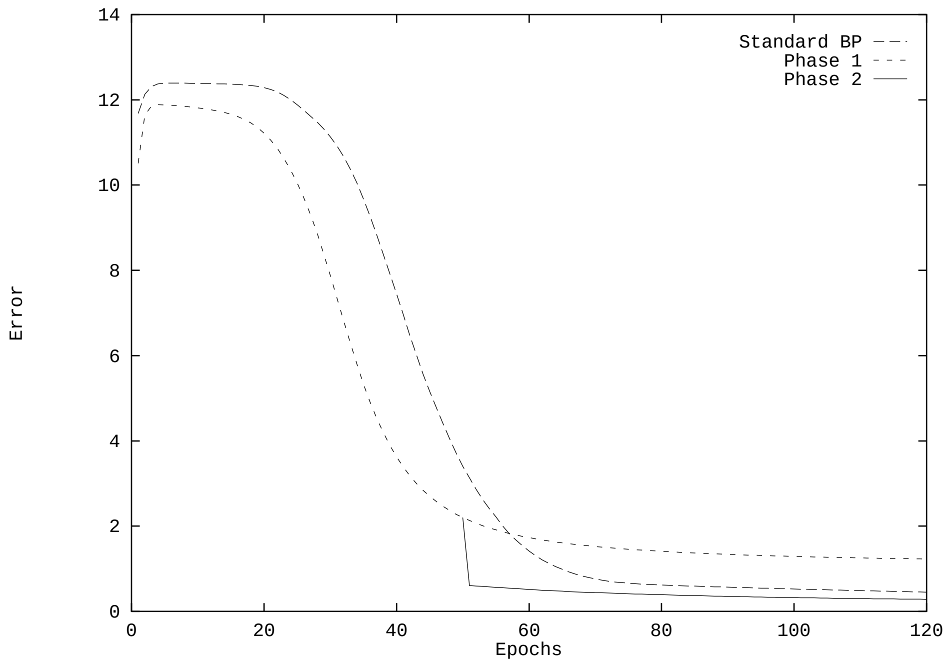

Figure 2. The Error functions vs Epochs

5 Discussion

In Phase 1 the adjustable variables, i.e. the degrees of freedom, of the network are always less than those in the regular BP[3]. Using the standard BP algorithm in a network with one hidden layer with k units and l output units, there are $l(k+1)+k(n+1)$ adjustable variables, whereas in Phase 1 of the algorithm there are only $l(k+1)+2k$. Thus, the reduction is by a factor F of:

$$F(k,l,n) = \frac{l(k+1)+k(n+1)}{l(k+1)+2k}$$

Figure 4 shows a graph of factor F vs k, for $l=1$ and $n=10,20,30$. It shows that for larger input dimension n, one has substantially less variables to adjust in Phase 1 than in the regular BP, thus substantially speeding up each epoch. Thus, the Two-phase algorithm promises to be even more effective for high dimensional applications. In comparing the speed of the Two-phase Algorithm and the standard BP, one has to consider the cost of finding the important directions in Phase 1, but this usually involves procedures such as finding eigenvectors which can be carried out in a small fraction of the time required by BP.

Having less variables to adjust, not only contributes to computational savings, but it can also have an impact on the convergence properties of the network. It has been observed that in many cases when more hidden units, i.e. more degrees of freedom, are added to the net, the more time it takes for the network to be trained (Lippman and Gold, 1987; Kolen and Pollack, 1990). By reducing the degrees of freedom (Phase 1), one expects the network to converge faster, but almost

[3]The trivial exception is when then input dimension is 1, where the adjustable variables are the same.

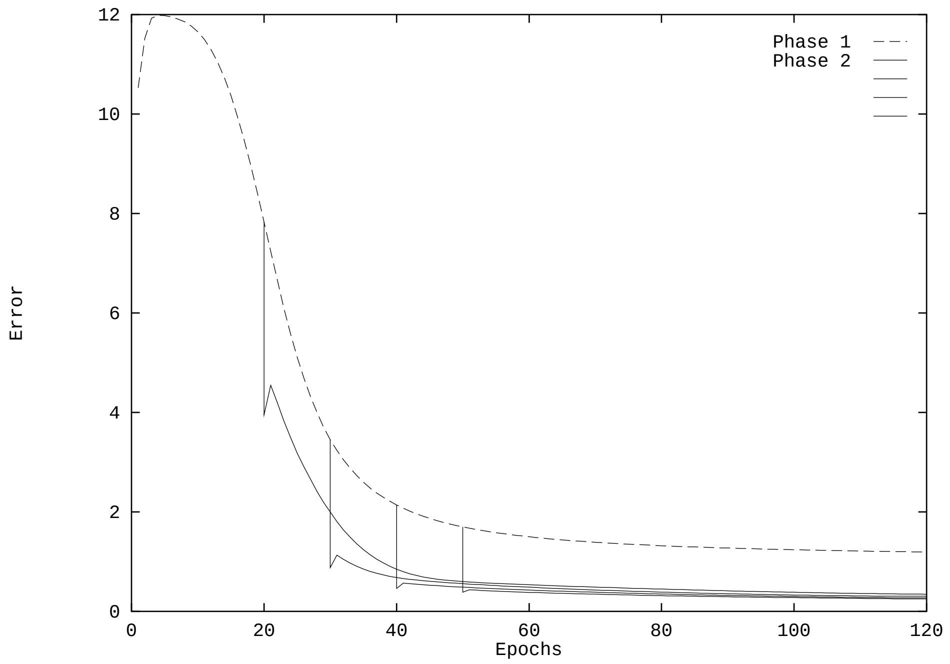

Figure 3. Switching to Phase 2 at different times

certainly not to the best solution achievable by the standard BP since the weight space in Phase 1 is a subspace of the weight space in the standard BP (Phase 2). Based on the above, the Two-phase algorithm can be viewed as doing the following: Phase 1 positions the weights so that the error is somewhere in a deep valley of the error surface (since additional information from the training set was used) and Phase 2 guides it to the bottom.

When to switch to Phase 2 from Phase 1? If the switch is done early when the weights and thresholds are drastically changing, then most likely at the end of Phase 2 the directions of the weight vectors will deviate substantially from the important directions, and thus the important directions are not actually being "used" to reach the solution. The best time to switch seems to be when marginal or no improvement in the error level is detected (see Figure 3).

With the important directions at hand, a software implementation of traditional BP can be transformed to the Two-phase BP by changing a few lines in the code.

References

Cottrell, G. W. and Baldi, P. (1988). Principal components analysis of images via back propagation. In *Proc. SPIE Int. Soc. Opt. Eng.*, volume 1001, (pp. 1070–1077).

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* N.Y.: Wiley.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, *7, Part II*, 179–188.

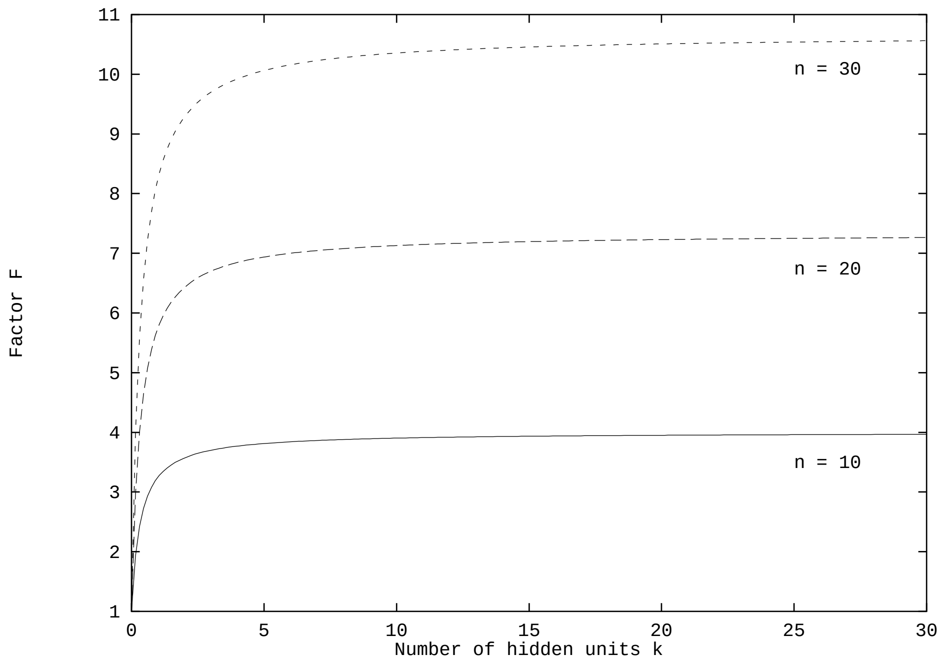

Figure 4. Factor by which the adjustable variables are reduced during Phase 1

Foley, D. H. and Sammon, Jr., J. W. (1975). An optimal set of discriminant vectors. *IEEE Transactions on Computers*, *C-24*(3), 281–289.

Gonzales, R. C. and Wintz, P. (1977). *Digital Image Processing.* Reading, MA: Addison-Wesley.

Johnson, R. A. and Wichern, D. W. (1988). *Applied Multivariate Statistical Analysis* (2nd ed.). Englewood Cliffs: Prentice Hall.

Kolen, J. F. and Pollack, J. B. (1990). Backpropagation is sensitive to initial conditions. *Complex Systems*, *4*, 269–280.

Kung, S. Y., Diamantaras, K. I., and Taur, J. S. (1990). Neural networks for extracting pure/-constrained/oriented principal components. In *Proc. 2nd Int. Workshop on SVD and Signal Processing*, Kingston, RI.

Lippman, R. P. and Gold, B. (1987). Neural classifiers useful for speech recognition. In *1st Int. Conference on Neural Networks*, (pp. IV417–IV426)., San Diego. IEEE.

Oja, E. (1982). A simplified neuron model. *J. Math. Biology*, *15*, 267–273.

Oja, E. (1989). Neural networks, principal components, and subspaces. *Int. Journal of Neural Systems*, *1*(1), 61–68.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group

(Eds.), *Parallel Distributed Processing: Volume 1: Foundations* (pp. 318–362). Cambridge: MIT Press.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer feedforward neural network. *Neural Networks*, *2*(6), 459–473.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.