Fast and Efficient Learning Algorithms for Feed-forward Neural Networks

Nicolaos B. Karayiannis
Department of Electrical Engineering
University of Houston
Houston, Texas 77204-4793
USA
email: Karayiannis@UH.EDU

Anastasios N. Venetsanopoulos
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario
CANADA M5S 1A4
email: anv@dsp.toronto.edu

Abstract
This paper presents a generalized criterion proposed for training feed-forward neural networks and evaluates the performance of the resulting gradient-descent-based learning algorithm. This paper also presents revisions, improvements, and extensions of the family of ELEANNE algorithms, proposed for training multilayered neural networks. The comparison of the proposed algorithms with their original version and gradient-descent-based algorithms indicated that the proposed revisions of the algorithms improve significantly their convergence, reduce the occurrence of local minima during the training process, and make the algorithms less sensitive to the parameters provided by the user.

1 Introduction
Most areas of neural network applications are currently in the transition from state-of-the-art paradigms to large-scale systems for real-world applications. This last category includes applications of neural networks in speech processing, image processing and computer vision, pattern classification and recognition, system control, and robotics. As the neural network field moves toward maturity, it became clear that the learning algorithms used to train “toy” networks to perform relatively simple tasks may be completely incapable of training larger networks in more complex problems. Slow convergence and long training times are the disadvantages often mentioned when neural networks are compared with other competing techniques. There is a cross-fertilization between advances in neural network theory and applications. Certainly, applications of neural networks benefit from theoretical developments in neural network research. In return, problems arising in neural network applications provide useful feedback to researchers in the area. For example, the development of large-scale systems for real-world applications emphasized the importance of certain issues, such as fast and efficient learning, generalization and scaling properties of various neural architectures, and learning without local minima.

Feed-forward neural networks are frequently trained by the error back propagation (EBP) algorithm, which was derived by minimizing the quadratic error criterion using the gradient descent method [1]. Although the impact of this algorithm in neural network theory and applications was undoubtedly enormous, it is widely recognized that it suffers from slow convergence. The development of fast and efficient learning algorithms for neural networks has been a subject of considerable interest over the past few years. The popularity of multilayered neural networks motivated several researchers to focus on heuristic techniques for accelerating the EBP algorithm and the development of alternative learning algorithms. Accelerated versions of the EBP algorithm were attempted by adapting the learning rate during training [2], [3], or by using various other heuristics to improve the convergence of the algorithm [4], [5].

The development of fast learning algorithms for multilayered neural networks was attempted by considering alternative criteria to the frequently used quadratic error criterion. In an attempt to derive fast learning algorithms, several researchers suggested that feed-forward neural networks with binary outputs can be trained by maximizing the relative entropy of the expected output with respect to the estimate provided by the network [6], [7], [8]. It was experimentally verified that the gradient-descent-based learning algorithm resulting from
the maximization of the relative entropy, known as the QuickProp or quick back propagation (QBP) algorithm, converges faster than the conventional EBP algorithm [8]. It was proposed recently that feed-forward neural networks can be trained by appropriately relaxing the quadratic error criterion during training [9]. The resulting generalized training criterion led to a variety of fast and efficient learning algorithms, including the fast back propagation (FBP) algorithm [9], [10], [11], [12].

The development of alternative learning algorithms for multilayered neural networks was also attempted by modifying the optimization strategy and/or employing adaptation rules other than the gradient descent. Parker presented an interesting step toward the development of second-order learning algorithms for neural networks [13]. In an attempt to develop second-order learning algorithms for feed-forward neural networks, Watrous considered various methods for approximating the second-order derivatives of the objective function iteratively [14]. Becker and le Cun attempted to improve the convergence of back propagation learning by using an approximation of the Newton method [15]. In an attempt to develop second-order learning algorithms for feed-forward neural networks, Kollas and Anastassiou proposed an analytically complicated and computationally demanding algorithm as an efficient alternative to gradient-descent-based algorithms [16]. Singhal and Wu suggested that the training of a multilayered neural network can be interpreted as an identification problem for a nonlinear dynamic system which can be solved by using the extended Kalman algorithm [17]. Puskorius and Feldkamp proposed a learning algorithm for feed-forward neural networks based upon a decoupled extended Kalman filter [18]. Karayiannis proposed an alternative optimization strategy, which provided the basis for the development of a family of Efficient LEarning Algorithms for Neural NEtworks (ELEANNE) [9]. This formulation resulted in efficient learning algorithms for multilayered neural networks which combine the convergence properties of second-order optimization methods and the computational simplicity of the gradient descent method [9], [12], [19].

Although there has been a significant progress in the area of fast and efficient learning algorithms, it seems that researchers in the area are hesitant to employ the new training tools in applications. As a result of the popularity of the EBP algorithm, sometimes multilayered neural networks are referred to as backpropagation networks. The inevitable result of such an identity in perception is that the inefficiencies of this learning algorithm create a negative impression about an otherwise powerful neural architecture. Since the natural tendency of researchers to use well established techniques fails to fully justify the resistance toward the new developments, the only explanation seems to be that none of the algorithms proposed as alternatives to the EBP algorithm emerged as a clear winner. In some cases, the additional computational burden and implementation complexity is not really balanced by a marginal improvement in convergence. Other algorithms are used only by their creators, since they are not well-defined and easily reproducible. In conclusion, experimental evidence that a learning algorithm converges slightly faster than the EBP algorithm is not sufficient for its acceptance as a serious competitor. A new learning algorithm will be widely accepted only if it succeeds in training tasks where the EBP algorithm and its derivatives fail or their performance is unacceptable.

In an attempt to evaluate various training criteria existing in the literature, this paper compares the performance of the EBP, QBP, and FBP algorithms used to perform a nontrivial task. After the publication of the ELEANNE family in [12], several researchers applied the algorithms in difficult training tasks and pointed out some of their shortcomings. This feedback motivated the reexamination of the algorithms proposed for training multilayered neural networks, which resulted in the revisions and improvements presented in this paper [20], [21].

2 Training Feed-forward Neural Networks

Consider the feed-forward neural network with \( n_i \) inputs, \( n_o \) output units, and one layer of \( n_h \) hidden units, shown in Figure 1. Assuming that the input of the network is formed by \( x_{j,k}, j = 1, 2, \ldots, n_i \), the outputs of the hidden units are \( h_{j,k} = \rho(h_{j,k}) = \rho(v_j^h x_k) \), where \( x_k = [x_{0,k}, x_{1,k}, x_{2,k}, \ldots, x_{n_h,k}] \), \( x_{0,k} = 1 \forall k = 1, 2, \ldots, m \), and \( \rho(x) = \tanh(x) \). Let \( \hat{x}_k \) be the \( (n_h + 1) \times 1 \) vector formed as \( \hat{x}_k = [h_{0,k}, h_{1,k}, h_{2,k}, \ldots, h_{n_h,k}] \), and \( \hat{x}_0,k = 1 \forall k = 1, 2, \ldots, m \). The output of the network is formed by the elements \( \hat{y}_{i,k} = \sigma(\hat{y}_{i,k}) = \sigma(w_i^h \hat{x}_k) \), where \( \sigma(\cdot) \) is a continuous, differentiable everywhere function. If the output of the network is analog, \( \sigma(x) = x \). If the output of the network is binary and the states of the output units are \( \pm 1, \sigma(x) = \tanh(x) \).

Feed-forward neural networks can be trained to carry out certain tasks by appropriately creating a training set, that is, a set of output-input pairs \( (y_{k}, x_k) \), \( k = 1, 2, \ldots, m \), also called examples or associations. For a given application, the training set is determined by an external teacher. This kind of learning is called supervised learning.
Feed-forward neural networks are trained by adjusting their synaptic weights in such a way that when the input of the network is \( x_k \), its response is the corresponding output \( y_k \). This is usually achieved by minimizing a discrepancy measure \( D_k \) between the desired output \( y_k \) and the actual response \( \hat{y}_k \) for all the output-input pairs in the training set. If the network consists of \( n_0 \) output units, the discrepancy between the desired output \( y_k \) and the actual output \( \hat{y}_k \) can be measured by

\[
D_k = \sum_{i=1}^{n_0} \phi(e_{i,k}) = \sum_{i=1}^{n_0} \phi(y_{i,k} - \hat{y}_{i,k})
\]

where \( \phi(.) \) is a positive definite, convex, continuous and differentiable everywhere function. The minimization of the total discrepancy measure

\[
D = \sum_{k=1}^{m} D_k = \sum_{k=1}^{m} \sum_{i=1}^{n_0} \phi(y_{i,k} - \hat{y}_{i,k})
\]

is frequently based on the sequential minimization of \( D_k, k = 1, 2, \ldots, m \). The training of a feed-forward neural network is usually performed in a sequence of adaptation cycles. Each adaptation cycle involves the adaptation of all the synaptic weights of the network with respect to all the associations which form the training set.

The training of neural networks is traditionally based on the minimization of the objective function

\[
E = \sum_{k=1}^{m} E_k = \frac{1}{2} \sum_{k=1}^{m} \sum_{i=1}^{n_0} (y_{i,k} - \hat{y}_{i,k})^2
\]

If the output of the network is binary, the network can alternatively be trained by maximizing the relative entropy criterion \([6],[7],[8]\). If the binary outputs of the network take the values \( \pm 1 \), the network can be trained by maximizing \([12],[22]\)

\[
H = \sum_{k=1}^{m} H_k = -\frac{1}{2} \sum_{k=1}^{m} \sum_{i=1}^{n_0} [\phi_-(e_{i,k}) + \phi_+(e_{i,k})]
\]

where

\[
\phi_{\pm}(e_{i,k}) = \frac{1}{2} (1 \pm y_{i,k}) \ln \left[ \frac{1 \pm y_{i,k}}{1 \pm \hat{y}_{i,k}} \right]
\]

The maximization of (4) is equivalent with the minimization of the discrepancy measure \( D \), defined in (2), with \( \phi(e_{i,k}) = \phi_-(e_{i,k}) + \phi_+(e_{i,k}) \).

According to a generalized training criterion proposed recently, a feed-forward neural network can be trained by minimizing the cost function \([9]\)

\[
G(\lambda) = \sum_{k=1}^{m} \sum_{i=1}^{n_0} [\lambda \phi_2(e_{i,k}) + (1 - \lambda) \phi_1(e_{i,k})]
\]

where \( \phi_2(e_{i,k}) = \frac{1}{2} e_{i,k}^2 \) and \( \lambda \) decreases from 1 to 0 during the training. If the outputs of the network are binary and take the values \( \pm 1 \),

\[
\phi_1(e_{i,k}) = y_{i,k} (y_{i,k} - \hat{y}_{i,k}) = 1 - y_{i,k} \hat{y}_{i,k}
\]

It was suggested that the value of \( \lambda \) may be determined in each adaptation cycle from the total error at that point, according to the following rule \([9],[10],[11],[12]\)

\[
\lambda = \lambda(E) = \exp(-\mu / E^2)
\]

where \( \mu \) is a positive real number. Although the rule (8) is not unique, it guarantees that \( \lambda \approx 1 \) during the initial adaptation cycles, i.e., when the total error \( E \) is large, and \( \lambda \) approaches zero as the total error decreases.

During the initial stage of training, i.e., when \( \lambda = 1 \), the resulting algorithm is based on the minimization of the error \( G(1) = E = \frac{1}{2} \sum_{k=1}^{m} \sum_{i=1}^{n_0} (y_{i,k} - \hat{y}_{i,k})^2 \). The synaptic weights of the network trained on the basis of the generalized criterion are updated by penalizing error.

During the last stage of training, i.e., when \( \lambda = 0 \), the resulting algorithm is based on the minimization of \( G(0) = \sum_{k=1}^{m} \sum_{i=1}^{n_0} y_{i,k} (y_{i,k} - \hat{y}_{i,k}) \), or equivalently the maximization of \( C = \sum_{k=1}^{m} \sum_{i=1}^{n_0} y_{i,k} \hat{y}_{i,k} \). \( C \) is a measure of similarity between the expected response and the response provided by the network. During this stage of training, the synaptic weights of the network trained on the basis of the generalized criterion are updated by rewarding success.

During the intermediate stage of training, i.e., when \( 0 < \lambda < 1 \), the synaptic weights of the network are updated by partly penalizing error and partly rewarding success.
3 Gradient-Descent-Based Algorithms

A variety of learning algorithms for multilayered neural networks can be obtained by applying the gradient descent method on the training criteria presented above [12]. The update equations for each algorithm can be derived by sequentially minimizing the corresponding discrepancy measure $D_k$ for $k = 1, 2, \ldots, m$. The update equation for the synaptic weights $w_{p,k}$ is obtained as follows:

$$w_{p,k} = w_{p,k-1} + \alpha \frac{\partial}{\partial w_{p,k}} D_k$$

If a network with analog output is trained by minimizing the quadratic error criterion,

$$\epsilon_{p,k}^q = \epsilon_{p,k}^q = y_{p,k} - \hat{y}_{p,k}$$

If the output of the network is binary,\footnote{Equation for the synaptic weights}

$$\epsilon_{p,k}^q = \begin{cases} (1 - \hat{y}_{p,k})(y_{p,k} - \hat{y}_{p,k}) & \text{if } D = E \\ \epsilon_{p,k}^r & \text{if } D = -H \\ \epsilon_{p,k}^g(\lambda) = (1 - \hat{y}_{p,k})(y_{p,k} - \lambda \hat{y}_{p,k}) & \text{if } D = G(\lambda) \end{cases}$$

It can also be shown that the synaptic weights $v_{pq}$ can be updated through the equation

$$v_{pq} = v_{pq-1} + \alpha \frac{\partial}{\partial v_{pq}} D_k$$

where

$$\epsilon_{p,k}^h = (1 - \hat{h}_{p,k}) \sum_{i=1}^{n_a} \epsilon_{i,k} \cdot u_{ip}$$

The minimization of the quadratic error and the maximization of the relative entropy lead to the EBP and QBP algorithms, respectively [1], [8]. The minimization of the generalized training criterion resulted in the FBP algorithm [9], [10], [11], [12].

The convergence of the gradient-descent-based algorithms presented above depends on the output error $\epsilon_{p,k}^o$, which is propagated back through the network during the training process. When $\lambda = 1$,

$$\epsilon_{p,k}^q = \epsilon_{p,k}^q(1) = \begin{cases} (1 + \hat{y}_{p,k})\epsilon_{p,k}^2 & \text{if } y_{p,k} = +1 \\ -(1 - \hat{y}_{p,k})\epsilon_{p,k}^2 & \text{if } y_{p,k} = -1 \end{cases}$$

When the estimate $\hat{y}_{p,k}$ is close to its target $y_{p,k}$, the adaptation of the synaptic weights is dominated by the term $\epsilon_{p,k}^2 = (y_{p,k} - \hat{y}_{p,k})^2$. This term makes the convergence of the EBP algorithm very slow after the initial adaptation cycles. It can easily be verified that if $\lambda = 0$,

$$\rho_{p,k}^q(0) = \begin{cases} (1 + \hat{y}_{p,k})\epsilon_{p,k} & \text{if } y_{p,k} = +1 \\ -(1 - \hat{y}_{p,k})\epsilon_{p,k} & \text{if } y_{p,k} = -1 \end{cases}$$

This latter equation indicates that when the value of $\lambda$ approaches zero, the adaptation of the synaptic weights is determined by the error $\epsilon_{p,k} = y_{p,k} - \hat{y}_{p,k}$.

Finally, consider the output error which corresponds to the FBP and QBP algorithms after the initial adaptation cycles, i.e., when each $\hat{y}_{p,k}$ approaches its target $y_{p,k}$. If $y_{p,k} = 1$ and $\hat{y}_{p,k} \in [0, 1]$, then $1 \leq (1 + \hat{y}_{p,k}) < 2$. The definition of $\rho_{p,k}^q(0)$ in (15) indicates that

$$\epsilon_{p,k}^q \leq \epsilon_{p,k}^q(0) < 2 \epsilon_{p,k}^q$$

It can similarly be shown that this latter inequality is also valid in the case where $y_{p,k} = -1$ and $\hat{y}_{p,k} \in [-1, 0]$. This observation reveals a similarity between the FBP and QBP algorithms after the initial adaptation cycles, and also explains the fast convergence of both algorithms. The crucial difference between these two algorithms is that during the initial adaptation cycles the FBP algorithm is based on the minimization of the quadratic function (3). The possibility of formulating an alternative generalized criterion by combining the quadratic error and the relative entropy criteria has been discussed elsewhere [11], [12].

4 Improved ELEANNE Algorithms

The development of the family of ELEANNE algorithms was based on an alternative optimization strategy proposed for training feed-forward neural networks [9]. This strategy is based on the adaptation of the synaptic weights that connect the hidden and the output layers using a second-order method, followed by the adaptation of the other sets of synaptic weights on the basis of the gradient descent method. This choice is justified by the observation that the effect of the synaptic weights on the training process decreases as the corresponding connections are separated from the output of the network by one or more layers of nonlinear hidden units. This latter argument is also supported by the fact that any insignificant fluctuation of these synaptic weights is absorbed by the sigmoid nonlinearities of the hidden units.

In order to simplify the analysis which follows, assume that the multilayered neural network shown in Figure 1 consists of two single-layered neural networks; the upper
network with synaptic weights \( w_{pq} \), and the lower network with synaptic weights \( v_{pq} \). Clearly, the upper network can be regarded as a single-layered network whose inputs are formed by the outputs of the hidden units. Assuming that \( x_k \) is the input of the lower network, the input of the upper network is the temporary input pattern \( h_k \). The adaptation of the synaptic weights of the upper network is based on the observation that \( E = \sum_{i=1}^{n_v} E_i \), where \( E_i = E_{i,m} = \sum_{k=1}^{n_h} \phi(e_{i,k}) = E_{i,m-1} + \phi(e_{i,m}) \). The network can be trained with respect to the associations \((y_i, x_k), k = 1, 2, \ldots, m-1\) by minimizing \( E_{i,m-1} \). The weight vectors can be updated with respect to the new association \((y_m, x_m)\) by minimizing \( E_{i,m} = E_{i,m-1} + \phi(e_{i,m}) \). Suppose that \( w_i \) has been updated with respect to \((y_i, x_k), k = 1, 2, \ldots, m-1\) by minimizing \( E_{i,m-1} \), resulting in the estimate \( w_i = w_{i,m-1} \). The \( i \)th row of the weight matrix \( W \) is then updated with respect to \((y_m, x_m)\) by minimizing \( E_{i,m} \) through the update equation \([9], [12], [19]\)

\[
 w_{i,m} = w_{i,m-1} - \alpha \frac{\partial E_{i,m}}{\partial w_i} |_{w_i = w_{i,m-1}} \tag{17}
\]

where \( \partial E_{i,m} / \partial w_i \) is the gradient of \( E_{i,m} \) with respect to \( w_i \), \( H_{i,m} \), the Hessian matrix, and \( \alpha \) is a positive real number, called the learning rate.

It can be shown that the gradient of \( E_{i,m} \) with respect to \( w_i \), \( H_{i,m} \), the Hessian matrix, and \( \alpha \) is a positive real number, called the learning rate.

If the output of the network is binary, \( E_{i,k}^\delta = \begin{cases} (1 - \hat{y}_{i,k}^2)(y_i,k - \hat{y}_{i,k}) & \text{if } D = E \\ y_i,k - \hat{y}_{i,k} & \text{if } D = -H \end{cases} \)

The Hessian matrix \( H_{i,m} \) is of the form \([20], [21]\)

\[
 H_{i,m} = \sum_{k=1}^{m} c_{i,k} \hat{h}_k \hat{h}_k^* \tag{20}
\]

If a network with analog output is trained by minimizing the quadratic error, \( c_{i,k} = 1 \forall i = 1, 2, \ldots, n_v \). If the output of the network is binary, \( c_{i,k} = \begin{cases} (1 - \hat{y}_{i,k}^2)^2 & \text{if } D = E \\ 1 - \hat{y}_{i,k}^2 & \text{if } D = -H \end{cases} \)

4.1 Recursive Inversion of the Hessian

The Hessian matrix \((20)\) can also be written as \( H_{i,m} = H_{i,m-1} + c_{i,m} \hat{h}_m \hat{h}_m^* \), where \( c_{i,k} \geq 0 \). Therefore, the inverse \( P_{i,m} = H_{i,m}^{-1} \) can be evaluated recursively using the matrix inversion lemma as \( P_{i,m} = P_{i,m-1} - \delta_{i,m} P_{i,m-1} \hat{h}_m \hat{h}_m^* P_{i,m-1} \), where \( \delta_{i,m} = c_{i,m}(1 + c_{i,m} \hat{h}_m^* P_{i,m-1} \hat{h}_m)^{-1} \). Since there is no guarantee that the matrix \( H_{i,m} \) is nonsingular, the recursive evaluation of \( P_{i,m} \) may be unstable. This problem can be overcome by recursively evaluating \( P_{i,m} = (\eta I + H_{i,m})^{-1} \), where \( \eta \) is a positive constant close to zero and \( I \) is the identity matrix. Using this approach, the inverse \( P_{i,m} = (H_{i,m})^{-1} \) can be evaluated by \([9], [12], [19]\)

\[
 P_{i,m} = P_{i,m-1} - \delta_{i,m} P_{i,m-1} \hat{h}_m \hat{h}_m^* P_{i,m-1} \tag{22}
\]

where \( \delta_{i,m} = c_{i,m}(1 + c_{i,m} \hat{h}_m^* P_{i,m-1} \hat{h}_m)^{-1} \) and \( P_{i,0} = (1/\eta) I \).

The use of the learning algorithms resulting from this approach in complex training tasks revealed some disadvantages of this approach. In practice, even the choice of a positive \( \eta \) very close to zero may lead to an unstable recursion for \( P_{i,m} \) and force the resulting learning algorithm to oscillatory behavior. On the other hand, increasing the parameter \( \eta \) for the sake of stability affects the convergence of the resulting learning algorithm.

4.2 Compensation for the Initialization Process

The development of stable learning algorithms that are not strongly affected by the initialization scheme presented above was attempted by compensating the inverse Hessian matrix for the arbitrary initialization of its recursive evaluation \([20], [21]\). The recursive evaluation of \( P_{i,k} \) is initialized by

\[
 P_{i,1} = P_{i,0} - \delta_{i,1} P_{i,0} \hat{h}_1 \hat{h}_1^* P_{i,0} \tag{23}
\]

where \( \delta_{i,1} = c_{i,1}(1 + c_{i,1} \hat{h}_1^* P_{i,0} \hat{h}_1)^{-1} \) and \( P_{i,0} = P_{i,0} = (1/\eta) I \). In order to compensate for the use of \( P_{i,0} \), \( P_{i,1} \) is replaced in future recursions by \( P_{i,1} = P_{i,1} + \theta_{1,1} P_{i,0} \)

\[
 (P_{i,1} + \theta_{1,1} P_{i,0}) (c_{i,1} \hat{h}_1 \hat{h}_1^*) = I \tag{24}
\]

where the compensation coefficient \( \theta_{1,1} \) is selected in such a way that

\[
 (P_{i,1} + \theta_{1,1} P_{i,0}) (c_{i,1} \hat{h}_1 \hat{h}_1^*) = I \tag{24}
\]

In general, \( P_{i,k} \) is evaluated in terms of \( P_{i,k-1} = P_{i,k-1} + \theta_{i,k-1} P_{i,0} \) by

\[
 P_{i,k} = P_{i,k-1} - \delta_{i,k} P_{i,k-1} \hat{h}_k \hat{h}_k^* P_{i,k-1} \tag{25}
\]
where \( \delta_{i,k} = c_{i,k} (1 + c_{i,k} \hat{b}_k^* P_{i,k-1} \hat{h}_k)^{-1} \). In order to compensate for the use of \( P_{i,0} \), \( P_{i,k} \) is replaced by \( P_{i,k} = P_{i,k} + \theta_{i,k} P_{i,0} \), where \( \theta_{i,k} \) is selected in such a way that

\[
(P_{i,k}^* + \theta_{i,k} P_{i,k-1}) (c_{i,k} \hat{h}_k \hat{h}_k^*) = I
\]

(26)

The compensation coefficient \( \theta_{i,k} \) which satisfies this latter condition is given by [20], [21]

\[
\theta_{i,k}^{-1} = c_{i,k} (\hat{b}_k^* P_{i,k-1} \hat{h}_k) [1 + c_{i,k} (\hat{h}_k^* P_{i,k-1} \hat{h}_k)]
\]

(27)

### 4.3 Improved ELEANNE 5

The ELEANNE 5 was developed for neural networks with nonlinear output units on the basis of the following strategy: An adaptation cycle consists of \( n_0 \) stages, corresponding to the output units. Each stage consists of the adaptation of the corresponding row of the weight matrix \( W \) and all the synaptic weights with nonlinear output units on the basis of the following condition is given by

\[
\frac{\partial}{\partial w_{pq}^{\text{est}}} J(Y_{i,k} - Y_{i,k}) = 0
\]

(20)

The inverse Hessian is then updated with respect to the synaptic weights \( v_{pq} \) of the lower network by sequentially considering the \( m \) associations

\[
C_{i,k} = \sum_{k=1}^{m} c_{i,k} \hat{h}_k \hat{h}_k^* = \sum_{k=1}^{m} c_{i,k} \hat{h}_k \hat{h}_k^*
\]

During the initialization of the inverse Hessian, \( P_{i,k} \) is replaced by its current estimate \( P_{i,k} \). The weight vector \( w_i \) of the upper network is updated with respect to \( (y_k, x_k) \) by

\[
w_{i,k} = w_{i,k-1} + \alpha c_{i,k}^* P_{i,k} \hat{h}_k
\]

(28)

where \( c_{i,k}^* \) is defined by (19). The synaptic weights of the lower network \( v_{pq} \) can be updated with respect to the association \( (y_k, x_k) \) by minimizing \( D_k \) using the gradient descent method. According to (1), \( D_k = D_k^{\text{est}} + D_k^{\text{est}} \), where

\[
D_k^{\text{est}} = \sum_{i=1}^{n_0} \phi(y_k - y_i; k) - D_k^{\text{est}} \phi(e_{i,k}) = \phi(y_k - y_i; k)
\]

Since only \( w_i \) has been updated, the synaptic weights \( v_{pq} \) can be updated by minimizing \( D_k^{\text{est}} \) as follows [9], [12], [19]

\[
v_{pq,k} = v_{pq,k-1} + \alpha \hat{h}_k^* \hat{h}_k^* x_k
\]

(29)

where \( \hat{h}_k^* \) is defined in (18) and (19) if the output of the network is analog and binary, respectively. The synaptic weights \( v_{pq} \) of the lower network can be updated by
\[ v_{p,k} = v_{p,k-1} + \alpha \epsilon_{p,k}^h x_k \] (32)

where \( \epsilon_{p,k}^h = (1 - h_{p,k}^2) \sum_{i=1}^{n_{ip}} c_{i,k}^p w_{ip} \).

5 Experimental Results

The algorithms under consideration were used to train a feed-forward neural network with one layer of hidden units to separate two nested spirals in a 2-D space. This is a nontrivial classification task, which has been used as a benchmark in evaluating the convergence of learning algorithms [4], [23]. Some researchers reported that gradient-descent-based learning algorithms failed to converge when tested on this training set. Nevertheless, even in cases where these algorithms were successful, the number of adaptation cycles required for convergence was tremendous [23].

In the first set of experiments, the training set consisted of 100 associations, formed by assigning the 50 points belonging to each of the nested spirals to two classes. A network with two inputs, one nonlinear output unit and one layer of 10 hidden units was trained using the EBP, the QBP, and the FBP algorithms initialized randomly. Table I summarizes the average number of adaptation cycles required for convergence to the total error value \( E_0 = 10^{-2} \), evaluated over five trials, each starting with a different set of synaptic weights generated by a random number generator producing numbers between \(-0.5\) and \(+0.5\). According to Table I, the FBP and QBP algorithms both converge considerably faster than the EBP. As the value of the learning rate \( \alpha \) increases, the FBP algorithm converges faster than the QBP algorithm.

Table I. A comparison of gradient-descent-based learning algorithms

\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
Algorithm & EBP & QBP & FBP \\
\hline
& \( E_0 = 10^{-2} \) & \( E_0 = 10^{-2} \) & \( E_0 = 10^{-2} \) \\
\hline
1 \times 10^{-2} & 241,600 & 54,600 & 62,800 \\
2 \times 10^{-2} & 100,720 & 22,460 & 26,240 \\
3 \times 10^{-2} & 73,740 & 20,780 & 20,620 \\
4 \times 10^{-2} & 64,175 & 14,925 & 10,800 \\
\hline
\end{tabular}
\end{center}

The double spiral problem was also used for evaluating the improved ELEANNE algorithms. A network with two inputs, one nonlinear output unit and one layer of 10 hidden units was trained by maximizing the relative entropy using the QBP algorithm, the original ELEANNE 7 and the improved ELEANNE 7 algorithms. Table II shows the number of adaptation cycles required for convergence when the network was trained with various values of \( \alpha \) and \( \eta \). According to Table II, the number of adaptation cycles required for convergence decreased as the value of \( \eta \) decreased. When the value of \( \eta \) was fixed, the number of adaptation cycles required for convergence decreased as the learning rate increased. Tables II indicates that the improved algorithms proposed in this paper converge faster than their original versions, are less sensitive to the initialization parameter \( \eta \), and are not susceptible to local minima. In addition, the proposed algorithms converge significantly faster than the corresponding gradient-descent-based algorithms. The comparison of the best results for each algorithm indicated that the training times required by the improved ELEANNE algorithms (tEL) and the Back Propagation algorithms (tBP) are related by \( t_{EL}/t_{BP} \approx 1/54 \) [20], [21].

Table II. A comparison of improved ELEANNE 7 with original ELEANNE 7 and QBP

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
Algorithm & Learning Rate (\( \alpha \)) & \( 1 \times 10^{-2} \) & \( 5 \times 10^{-2} \) & \( 1 \times 10^{-2} \) \\
\hline
QBP & & & & \\
Original & & & & \\
ELEANNE 7 (\( \eta = 1.0 \)) & & & & \\
Improved & & & & \\
ELEANNE 7 (\( \eta = 0.1 \)) & & & & \\
Improved & & & & \\
ELEANNE 7 (\( \eta = 0.1 \)) & & & & \\
\hline
\end{tabular}
\end{center}

6 Conclusions

The development of fast and efficient learning algorithms for feed-forward neural networks was achieved by using training criteria other than the quadratic error and/or by employing update rules more sophisticated than the gradient descent. The experiments indicated that the convergence of gradient-descent-based algorithms can be accelerated by maximizing the relative entropy criterion or by relaxing the quadratic error criterion during training. It was also experimentally verified that the improved ELEANNE algorithms perform complex training tasks faster than their original versions and much faster than gradient-descent-based algorithms.

7 References


