Optimization for Training Neural Nets

Etienne Barnard

Abstract—**Various techniques of optimizing criterion functions to train neural-net classifiers are investigated. These techniques include three standard deterministic techniques (variable metric, conjugate gradient, and steepest descent), and a new stochastic technique. It is found that the stochastic technique is preferable on problems with large training sets and that the convergence rates of the variable-metric and conjugate-gradient techniques are similar.**

I. Introduction

PERHAPS the most useful result of the recent resurgence in research on neural nets is the creation of a family of powerful, robust classifiers. Whereas classifiers which create higher-order (nonlinear, nonquadratic) decision boundaries have conventionally been plagued by the combinatorial explosion (polynomial classifiers [1]) or the requirement that an excessive number of learning samples be stored (the nearest-neighbor classifier), neural-net classifiers can calculate such boundaries under realistic constraints. Thus, it is becoming increasingly clear that there are situations where neural-net classifiers are an attractive alternative to conventional techniques [2]–[5].

One important characteristic of neural-net classifiers is that their training usually requires iterative techniques. The back-propagation (BP) classifier, for instance, is trained by optimizing a transcendental criterion function. (This procedure will be explained in more detail later.) Since such a function can almost never be minimized analytically, optimization has to proceed iteratively. The number of optimization variables equals the number of free parameters in the specification of the neural net. This is typically equal to the number of "weights" which connect the neurons to one another—often as many as a few thousand. Thus, the optimization problem is rather involved; this has given neural-net training a reputation for being very slow. (In one of the first papers on neural-net classifiers [6], a relatively simple two-class experiment is described which required several thousand iterations to train, and Waibel *et al.* [3] describe a three-class problem which required more than 20 000 iterations.)

Thus, much research has been directed at improving the learning speed of neural-net classifiers. One approach studies alternative neural-net algorithms which might have the same classification performance as, for instance, BP, with faster training capabilities. Examples of such classifiers are the counterpropagation [7] and adaptive-clustering [4] classifiers. A different way to improve learning speed is to maintain the same basic classifier but to change the criterion function to facilitate iterative optimization [8], [9]. Fahlman [8], for instance, proposes adding a constant to the gradient used for back-propagation; he argues that this improves the learning rate since it ensures that more is learned on every iteration. (Incidentally, it is important to note that such modifications might adversely affect the performance of the trained classifier. Adding a constant to the derivative of the sigmoid function, for instance, leads to a function that asymptotically increases linearly. This is expected to compromise the error rate of the classifier at the position of the minimum when the various density functions overlap, since outliers are now weighted more heavily. An analogous effect has been described [4] for linear classifiers.) A third option is to use standard neural nets and criterion functions and to improve the technique whereby this function is optimized. It is this approach which is studied in more detail here.

When back-propagation was introduced, [6], it was proposed that the criterion function be optimized using gradient descent. The training of the Kohonen net [10] can similarly be viewed as gradient descent with a least-mean-squares criterion function. However, it was soon realized that more efficient training techniques can be employed; the "momentum" term [6] is the most popular example of such improvement. In this approach, not only the gradient but also the previous weight change is used to update the weight vector. Various researchers have shown the usefulness of the momentum approach; its efficacy can be understood by analogy with the quadratic case, where it improves the minimization rate by creating a second-order dynamic system (i.e., a system whose time behavior is characterized by two poles in the frequency domain). The second-order dynamic system can be shown to have much better asymptotic convergence properties than the first-order system which describes gradient descent without momentum.

The momentum algorithm is characterized by two parameters (the "learning rate" and the "momentum constant"), whose values are critical to the success of the optimization process. Although it is not hard to derive optimal values for these constants in the quadratic case, no equivalent result for the more general case—which almost always obtains for neural-net classifiers—exists. A large number of heuristic algorithms which adapt these parameters according to the training set and the status of the neural net have therefore been proposed (see, for instance, [11]–[13]). These heuristics are usually derived in ad hoc fashion, so that neither optimality nor robustness is ensured.

One way of overcoming these limitations is to utilize the body of knowledge available from conventional numerical analysis and control theory. Unconstrained optimization,

Manuscript received January 15, 1991; revised October 31, 1991.

The author is with the Department of Electronics and Computer Engineering, University of Pretoria, 0002, Pretoria, South Africa.

IEEE Log Number 9105457.

which is required for the training we have discussed, is one of the most mature disciplines within numerical mathematics [14] and supplies a wealth of applicable knowledge. In Section II we describe some important possible approaches in more detail. It should be noted that in using these techniques, we are probably giving up any semblance of biological plausibility. Since we are mostly concerned with the engineering aspects of neural nets and since there is any case much reason to doubt the direct biological relevance of algorithms such as BP [15], this is a small price to pay. Use of conventional optimization techniques for neural-net training has been proposed by various authors; see e.g., [16]–[18].

To use conventional optimization techniques, it is necessary that the training problem be described by a deterministic criterion function. This function is usually obtained by selecting a fixed set of training samples, and defining the total criterion function to be the sum of the criterion functions of the training samples. When this function is optimized iteratively, each iteration requires the computation of the contributions from all of the training samples (to calculate e.g. the value of the criterion function or its gradient). Thus, for large training sets, much computation has to be performed for every update of the weights. In the pattern-recognition literature, this is known as the batch mode for training a classifier. It might be preferable to have a technique which updates the weights after every presentation of a training sample. If this is done, each presentation of a training sample immediately leads to an update of the weight vector, so that updates involving samples which occur late in the training set have the benefit of training from earlier samples. Consequently, a single pass through the training set could possibly lead to much more improvement of the classifier than one iteration using the total criterion function. Such ideas lead us to consider techniques which have conventionally been associated with control and stochastic systems: every training sample is viewed as a random sample from an unknown distribution, and training of the classifier is viewed as the nondeterministic optimization of a criterion function which depends on this distribution. In Section III immediate-update algorithms (or sequential algorithms, as they are called) are discussed, and a new algorithm which implements stochastic optimization efficiently is introduced.

In Section IV we present simulation results which compare four optimization techniques for two different neural-net algorithms on a number of classification problems. This comparison allows us to understand the relative merits and problems of stochastic optimization and deterministic approximation. We also use the results in Section IV to analyze the relative efficiencies of three different deterministic optimization techniques on neural-net problems. Section V summarizes our conclusions, and outlines possible future work.

It should be noted that the issue of local and global minima is not addressed in this research. That problem is logically separate from the local-optimization problem which is studied here, and might be of less importance than has been supposed [19], for the following reason. Experience seems to indicate that the local minima generally correspond to duplication of function by certain neurons. If this is so, the problem of local minima can always be overcome by having a slightly larger number of neurons than the absolute minimum required if global optimization had been possible.

II. Deterministic Minimization of Criterion Functions

We will now introduce a general framework within which optimization of neural-net criterion functions can be discussed, and then study some specific deterministic optimization algorithms within this context.

The output of the neural net is described by a vector of neuron activities $\boldsymbol{z}$; for a given input sample s, the output $\boldsymbol{z} = \boldsymbol{z}_s$ is determined by the set of weights (which, for future convenience, are described by a vector $\boldsymbol{w}$), and the sample vector $\boldsymbol{x}_s$. (The way that $\boldsymbol{z}_s$ is obtained from $\boldsymbol{w}$ and $\boldsymbol{x}_s$ is determined by the specific neural-net model used.) For sample s there is a certain "desired" output vector $\boldsymbol{d}_s$, which typically indicates the class to which this sample belongs. The purpose of training is to adapt the $\boldsymbol{w}$ so that $\boldsymbol{z}_s$ approximates $\boldsymbol{d}_s$ when $\boldsymbol{x}_s$ is presented as input, for all s. We therefore introduce a criterion function, $E(\boldsymbol{z}_s, \boldsymbol{d}_s)$, which is small if and only if $\boldsymbol{z}_s$ and $\boldsymbol{d}_s$ are approximately equal, and train the classifier by minimizing this criterion function with respect to $\boldsymbol{w}$.

Since training requires the simultaneous minimization of the criterion functions of a number of samples, the contributions from the individual samples should somehow be combined. To use deterministic minimization, we form a single "total" criterion function by summing the contribution from all samples; that is, the function minimized is

$$E = \sum_s E(\boldsymbol{z}_s, \boldsymbol{d}_s). \tag{1}$$

As was mentioned in Section I, a number of optimization techniques have been employed for this purpose; for the reason described there we concentrate on conventional optimization techniques in this section. Of these techniques, it is generally felt [14] that the variable-metric (or "quasi-Newton") techniques [20] are most efficient for the unconstrained optimization of an arbitrary function. The variable-metric techniques estimate the inverse of the Hessian matrix of the criterion function iteratively, and update the weight vector under the assumption that the criterion function is locally approximately quadratic. Specifically, the Hessian matrix of a quadratic function $E = (1/2)\boldsymbol{w}^t \boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}\boldsymbol{w} + c$ is $\boldsymbol{A}$, and its minimum is located at $\boldsymbol{w}_m = -\boldsymbol{A}^{-1}\boldsymbol{b}$. Since $\boldsymbol{b}$ can be approximated from the current value of $\boldsymbol{w}$ and the gradient of E (using $\nabla E = \boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}$), knowledge of the inverse Hessian is sufficient to allow us to calculate the position of the minimum of E in the quadratic approximation. Because of the nonquadratic nature of the criterion function, $\boldsymbol{w}_m$ thus calculated will not correspond to the true minimum, and this procedure has to be repeated, starting at $\boldsymbol{w}_m$. (In practice this procedure is modified somewhat by the use of line searches, to improve the robustness of the algorithm. That is, if the current optimal weight vector is $\boldsymbol{w}$ and the quadratically optimal estimate is $\boldsymbol{w}_m$, a one-dimensional search for the minimum value along the line passing through these two points is performed.)

Although the variable-metric algorithms are very powerful, they do have certain disadvantages. Their main problem is that they require storage proportional to the square of the

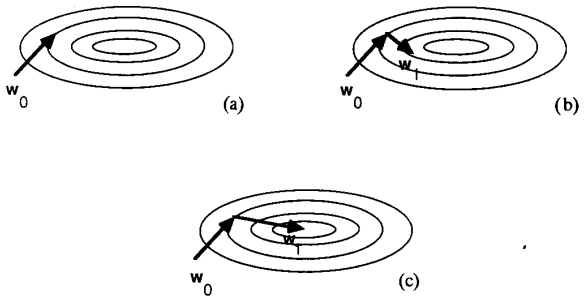

Fig. 1. (a) Location of the minimum along the direction of $\boldsymbol{w}_0$(b) Subsequent minimization along gradient direction. (c) Minimization along direction conjugate to $\boldsymbol{x}_0$.

number of optimization variables (since the inverse Hessian must be stored). For neural-net classifiers, which often employ a large number of weights, this disadvantage can be significant. In addition, the number of computations required for every iteration is also proportional to N^2(where N is the number of weights); in practice, this is often the dominant component of the computation time required.

For unconstrained minimization problems which involve a large number of variables, conjugate-gradient algorithms [21] are commonly used. These algorithms also optimize a criterion function by assuming that it is locally quadratic, but now approximation of the inverse Hessian matrix is not attempted. Instead, the guiding idea here is to maintain "conjugacy" of the successive search directions as we now explain. In Fig. 1, the contours of constant function value are shown for a two-dimensional slice through a quadratic function. Say we have located the minimum along the direction $\boldsymbol{w}_0$ indicated by the arrow in Fig. 1(a), and we now want to choose a new direction along which to search for a one-dimensional minimum. In Fig. 1(b) the new search direction, $\boldsymbol{w}_1$, is chosen to point in a direction opposite to the gradient at the position of the current minimum; it is clear that it will be necessary to minimize along $\boldsymbol{w}_0$ again after the minimum along $\boldsymbol{w}_1$ has been located. The search direction in Fig. 1(c), on the other hand, was chosen to ensure that the minimum along $\boldsymbol{w}_0$ is maintained (that is, all points along $\boldsymbol{w}_1$ share the property that their value cannot be decreased by moving in the direction of $\boldsymbol{w}_0$). This obviates the need for another search along $\boldsymbol{w}_0$. This example suggests a reasonable criterion for the choice of a new search direction, namely that it should not interfere with the minima found along the previous search directions. It turns out that there is a simple algorithm for constructing such a direction out of the gradient and the previous search direction when the function to be minimized is quadratic [21]. This algorithm requires only $O(N)$ computation per iteration and storage, and is known as the conjugate gradient algorithm.

It is indicative of the analytical complexity of nonlinear optimization that no general results on the relative merits of these two types of algorithms exist, despite their popularity in practical use. Because the variable-metric methods accumulate more knowledge of the error function, one expects that they will generally require fewer iterations to find the minimum than conjugate-gradient techniques; this is indeed what is observed in most benchmarks (see, e.g., Gill *et al.* [14]). however, we know of no characterization which will enable one to decide which method is most efficient computationally in a given situation. Thus, for a particular type of problem, it is necessary to empirically compare the two types of algorithms to see whether the increased rate of convergence of variable-metric methods outweighs its additional computational and storage requirements. For the case of neural-net criterion functions, such an empirical comparison will be performed in Section IV.

III. Optimization as a Stochastic Problem

If we view the training samples as random samples from an unknown distribution, the training problem is the minimization of the expected value E_i of the criterion function, where the expectation value is to be computed with respect to the unknown distribution. In the controls literature this problem is known as stochastic approximation. It was solved conceptually by Robbins and Munroe [22]. The Robbins–Munroe procedure can be understood as a stochastic version of gradient descent [1]: from a sample vector $\boldsymbol{x}_s$ we calculate a sampled gradient $\boldsymbol{g}_s$ and then adjust the optimization variables $\boldsymbol{w}$ in a direction opposite to this gradient. Thus, $\boldsymbol{w}$ is adapted according to

$$\boldsymbol{w} \rightarrow \boldsymbol{w} - \alpha \boldsymbol{g}_s, \tag{2}$$

where α is a positive parameter. The choice of α is of extreme importance. For deterministic techniques, one attempts to choose such "step size" parameters so that the criterion function takes on its (locally) minimal value along the search direction after the update (2). This is not appropriate for stochastic techniques, since the sampled criterion function is not the same as the expectation value of the criterion function. Thus, the purpose of each iteration is now to accumulate information about the position of the minimum, rather than to move directly toward that minimum, and α should be chosen accordingly. In fact, it is easy to see that α should not be constant: initially, when the weight vector is far from its optimal value, α should be large, so that $\boldsymbol{w}$ is driven toward the location of the minimum as quickly as possible. As $\boldsymbol{w}$ starts to converge, however, α should be decreased, so that outliers do not drive it away from the minimum which it is approaching.

It can be shown that, under fairly general conditions, algorithms such as the one described by (2) converge to a minimum of E as long as

1) $\sum_1^\infty \alpha_n = \infty$ (where n counts the number of sample presentations and α_n is the value of α after n iterations) and
2) $\sum_1^\infty \alpha_n^2 < \infty$.

Beyond these constraints it is very hard to determine how α should be chosen for fast, assured convergence; even these constraints are obviously not practical, since they refer to asymptotic quantities.

This problem of choosing the step size appropriately is a major hurdle for the efficient use of techniques derived from stochastic approximation for the training of neural nets. This is exactly the same problem researchers have to face when they try to find appropriate values for the learning rate and

momentum parameters when performing BP learning in the sequential training mode. Thus, an understanding of stochastic approximation leads to useful theoretical insights, but not a practical solution.

Extended Kalman estimation provides a different approach to optimizing a function which is only known through random samples. Consider the following situation: we have a system which is described by a state vector $\boldsymbol{w}(n)$, where n is a discretized time parameter. This state vector is not measurable, but is known to evolve according to the relationship

$$\boldsymbol{w}(n+1) = \boldsymbol{A}\boldsymbol{w}(n) + \boldsymbol{S}\boldsymbol{x}(n), \tag{3}$$

where $\boldsymbol{x}(n)$ is the input to the system, and $\boldsymbol{A}$ and $\boldsymbol{S}$ are known matrices. We wish to estimate $\boldsymbol{w}(n)$ by observing the output $\boldsymbol{d}(n)$ of the system; it is known that $\boldsymbol{d}$ is related to $\boldsymbol{w}$ by

$$\boldsymbol{d}(n) = \boldsymbol{T}\boldsymbol{w}(n) + \boldsymbol{r}(n). \tag{4}$$

Here $\boldsymbol{r}(n)$ is the observation noise (of which we know the statistics), and $\boldsymbol{T}$ is a known matrix. The Kalman algorithm supplies a solution to this problem, i.e., an estimate for $\boldsymbol{w}(n)$ which is optimal in a well-defined sense. Although this algorithm is directly applicable only when the input-state and state-output relationships are linear (as assumed in (3) and (4)), it can also be applied to nonlinear problems by linearizing the problem around the current estimated state of the system. This approach to nonlinear estimation is known as extended Kalman filtering.

The power of Kalman estimation is obtained at a price: the algorithm involves a fairly large amount of computation for every update. In its basic form, O(N^2) calculations have to be performed to update the state estimate, and the storage requirements are also O(N^2), where N is the dimensionality of the state vector.

To apply this algorithm to the training of neural nets, Singhal and Wu [23] suggested that we identify the state variable $\boldsymbol{w}$ with the weight vector, the input $\boldsymbol{x}$ with the measured input sample, and the output $\boldsymbol{d}$ with the desired neural-net output. Equations (3) and (4) become

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) \tag{5}$$

and

$$\boldsymbol{d}(n) = \boldsymbol{z}(n) + \boldsymbol{r}(n), \tag{6}$$

where $\boldsymbol{z}(n)$ is the actual output of the neural net when presented with the input $\boldsymbol{x}(n)$. Since $\boldsymbol{z}(n)$ is a known nonlinear function of the weights $\boldsymbol{w}(n)$, the extended Kalman algorithm can be applied.

For this application the computational requirements of the extended Kalman algorithm can be particularly severe, since updating of the weight vector is performed after every presentation of a sample. Thus, if N is large, the $O(N^2)$ calculations which have to be performed for these updates can easily outweigh the advantages that fast convergence with the Kalman algorithm can offer. For example, if $N = 1000$, we can perform approximately 1000 iterations of a system which has linear computational requirements in the time that it takes to perform one iteration with the extended Kalman algorithm. (This should be contrasted with the computational requirements of variable-metric methods: since those methods update the weight vector only after a pass through the entire training sets, the $O(N^2)$ computation required for them is not such an important issue.)

We therefore now introduce a practical sequential ("stochastic") algorithm which requires only $O(N)$ storage and computation. This algorithm is based on the observation that there is a characteristic length scale, L, which is traversed on every iteration as one approaches the minimum, and that this length scale generally changes rather slowly. Thus, if we can approximate L at every stage during the minimization procedure, we can always adjust the step-size parameter(s) appropriately.

To estimate L, we accumulate a large number of samples (how large will be discussed below), and form a deterministic criterion function as the sum of the individual criterion functions. One line search along the gradient of the deterministic function is then performed; the distance in weight space traversed by this line search provides an estimate of L.

Using this accumulated set of samples we also compute the average length, l, of the sample gradient vectors at the current position in weight space. An appropriate step size for an algorithm such as (2) is then

$$\alpha = L/l. \tag{7}$$

Note that this ensures that the average adjustment to the weight vector will have the correct length scale, but that the actual change on the presentation of any sample will still be proportional to the gradient computed from the sample. This is preferable to forcing every weight change to have the same norm, since the information contained in the norm of the gradient is retained.

Three issues which have to be addressed are how large a sample set should be used to estimate L and l, how often α should be updated, and the appropriate way to terminate this algorithm. The sample size should be large enough to ensure that the estimates are sufficiently accurate, but the larger it is, the more storage and computation are required for this step. In typical pattern-recognition applications a fixed training set is used, so that this compromise must already be addressed when the training set is selected. Since the trade-offs are equivalent, a suitable training-set size is also a suitable sample-set size for the estimation procedure. We can therefore estimate l and L using the entire training set.

The frequency with which α is updated is dictated by the following dichotomy: if updating takes place too seldomly, the estimates of l and L might become inaccurate; on the other hand, if we update these parameters too often we loose the computational advantages of the stochastic method. Our compromise is to set a fixed upper limit to the number of sample presentations between every update, and to force a new update when this limit is exceeded or whenever it is found that $\langle E \rangle$ is no longer decreasing. The upper limit is chosen to ensure that the amount of computation required for the deterministic updating phase is less than the computation involved in stochastic learning as long as progress towards the minimum of E is being made; for a fixed training set, it

has been determined empirically that approximately 20 passes through the training set constitute an appropriate upper limit.

It is clear that the optimal value for this parameter is problem dependent: as problem complexity increases, the error function becomes more complicated in weight space, and more frequent updates of L are required. Thus, the dimensionality of the input space may be used as an indicator of the update frequency, since a high-dimensional input space may be an indicator of problem complexity. Fortunately, the method is not very sensitive to the value chosen, and a fixed (problem-independent) upper limit has been found to suffice.

The termination problem is the following: as we converge towards the optimal $\boldsymbol{w}$, the estimate for L which is calculated by the line-search procedure will tend to be dominated by the corrections in the transition from stochastic to deterministic operation. That is, after a sequence of stochastic updates with step size α_n the estimated $\boldsymbol{w}$ will be stochastically perturbed by a quantity proportional to this step size (because the estimate is based on random sampling). The deterministic estimate for L will therefore include a contribution which is caused by the correction of this stochastic fluctuation. This contribution will dominate as the procedure converges to the optimal $\boldsymbol{w}$. Once we are this close to the optimal $\boldsymbol{w}$, however, deterministic techniques should converge very quickly. In practical problems, we therefore always terminate our learning procedure with a few iterations of deterministic search (i.e., using a fixed set of samples as above), to eliminate the effects of overestimating L. However, in the tests described in Section IV such a terminating step is not included, since this step will typically employ one of the other techniques which are compared with this stochastic method in Section IV. Thus, the efficacy of the termination process can be determined by examining the performance of those techniques.

This basic algorithm is embellished with some standard techniques from numerical analysis, for instance the retention of an optimal estimate for $\boldsymbol{w}$ at all times (to restart in case some unexpected aspect of the energy function throws the optimization procedure off track) and the use of a proven line-search algorithm for the deterministic phase of the algorithm. These precautions ensure the robustness of our stochastic algorithm; its efficiency, however, will depend strongly on the validity of the assumption that there is a meaningful, stable length scale which can be estimated with a deterministic line search. Thus, as with the deterministic techniques, it is possible to know whether this technique performs well on a given class of problems only by investigating it empirically.

IV. Simulation Results

To compare the various optimization procedures on neural-net training problems, we used three benchmark data sets. The first set is the standard "exclusive-or" benchmark, which consists of four two-dimensional training samples from two classes, as shown in Fig. 2. This benchmark has become standard in neural-network literature, because it cannot be solved by a simple linear classifier. It is unsatisfactory from a pattern-recognition point of view, however, since there is no contiguity relationship between samples in the same class.

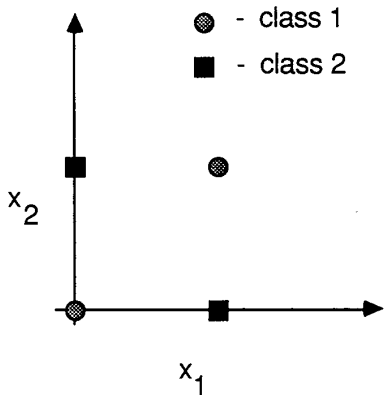

Fig. 2. The exclusive-or classification problem.

Thus, the ability of the classifier to "generalize" (so that a relatively small number of classifier parameters describe many training samples), which is usually an important property of classifiers, is not tested by the exclusive-or problem. The other two data sets are more realistic in this sense. The second set is the artificial set shown in Fig. 3. It consists of 383 two-dimensional samples from three classes. The third data set was generated from the Fourier transform of artificial aircraft data, as is discussed in more detail elsewhere [4], [24]. There were three classes of aircraft, and 630 32–dimensional training samples from each class were employed, for a total of 1890 samples.

Two different neural-net algorithms were used to classify these samples, namely standard three-layered BP [6] and the adaptive-clustering (AC) classifier [4]. The criterion functions used in conjunction with these classifiers were also standard: the sum-of-squares criterion function for BP and the perceptron criterion function (with safety margin) for AC. In our comparison we compare the actual values of the criterion function as a function of the number of iterations performed with the various techniques, since we are interested in the ability of the various algorithms to minimize such a function. The performance of the classifier on a separate test set would be more appropriate as a measure of the overall performance of the classifier. However, this performance also depends on such factors as the nature of the criterion function used, and our purpose is to compare the optimization techniques in isolation.

We compared the performance of four optimization techniques on these data sets and neural-net algorithms. Three of the techniques are deterministic, namely the BFGS variable-metric technique [20] and conjugate-gradient method with restarts [21] (these two techniques are based on the algorithms described in Section II), and the steepest-descent technique. The last method minimizes the criterion function by 1) computing the gradient at the current weight value, 2) performing a line search in the direction of the gradient to locate the local minimum in this direction, and 3) using this as the new weight value for step 1. This method provides a useful baseline reference as the most crude gradient-based optimization algorithm. It can also be viewed as an upper limit for the performance of standard back-propagation [6]

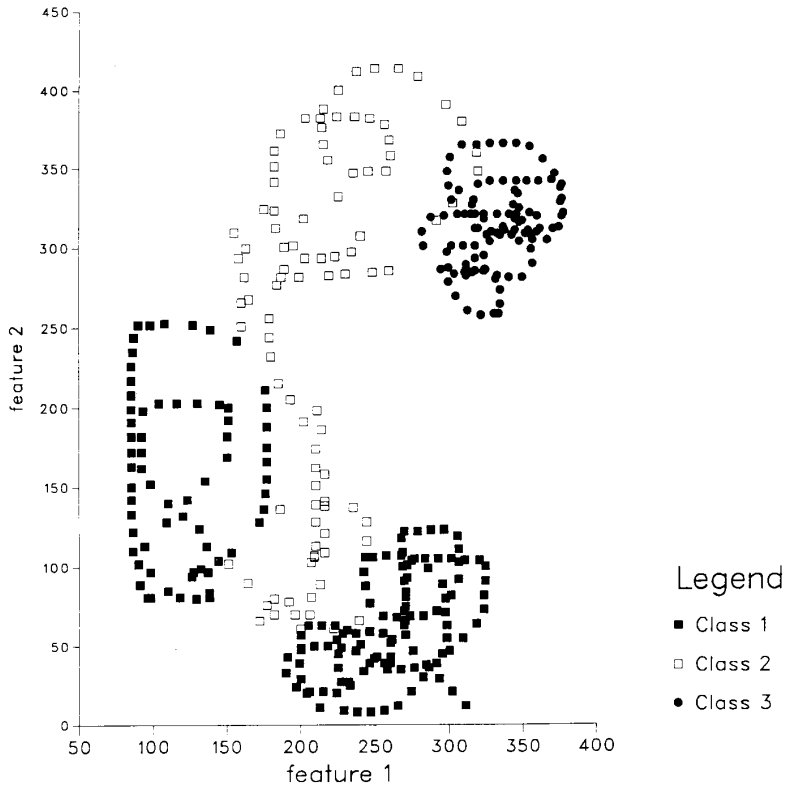

Fig. 3. Artificial two-dimensional data set used in optimization comparison.

TABLE I
SIZES OF NEURAL NETS USED IN COMPARATIVE STUDIES

Problem	Size of Input Layer	Size of Hidden Layer	Size of Output Layer
Exclusive Or	3	4	2
Artificial	3	6	3
Aircraft	33	10	3

in "batch" mode, since it dynamically adjusts the effective step size during line searches. (Standard "on-line" and "batch-mode" implementations of back-propagation are not included in our comparison, since results obtained are so sensitive to the values of learning parameters chosen. It is clear, however, that on most problems such fixed learning rates will lead to inferior performance, and both the published literature [3], [8] and our informal experience on the data sets described above confirm this expectation.) The fourth optimization algorithm employed was the stochastic method described in Section III.

For all problems the sizes of the second (hidden) layers of the neural nets were chosen to give satisfactory classification performance. The values used are listed in Table I. Also shown are the sizes of each input layer (one more than the number of features) and each output layer (equal to the number of classes).

In parts (a), (b), and (c) of Fig. 4, the results obtained with, respectively, the BP classifier on the exclusive-or, artificial two-dimensional, and aircraft data sets are shown. Since the convergence of BP depends on the random initial values of the weight vector, all BP results represent the average of five trials with different initial weights. (The random initial weights were uniformly distributed in the interval $[-1, 1]$, and the input features were all scaled to lie in the interval $[0, 1]$.) In Fig. 4(a) we see that both the BFGS and the conjugate-gradient method lead to fast minimization of the criterion function when the exclusive-or data are employed, whereas the other two methods were less successful. That these differences are indeed significant can be deduced from the fact that the standard deviation of each of the points in Fig. 4(a) is approximately 0.02. The slow convergence of the steepest-descent method is caused by the well-known drawbacks [25] of such methods when the condition number of the Hessian of the criterion function is large on the average. The stochastic technique, on the other hand, fares badly because the length scale L is estimated from only four samples, and is thus inaccurate. This can be deduced from the results on the other two BP problems. When the artificial two-dimensional set is employed (Fig. 4(b)), the stochastic technique already converges approximately twice as fast as the BFGS algorithm, which performs second best, and on the aircraft data set (which has the largest number of samples) the stochastic technique converges at a rate approximately five times as high as the rate of the best deterministic technique. For these two problems the standard deviations on each of the data points were on

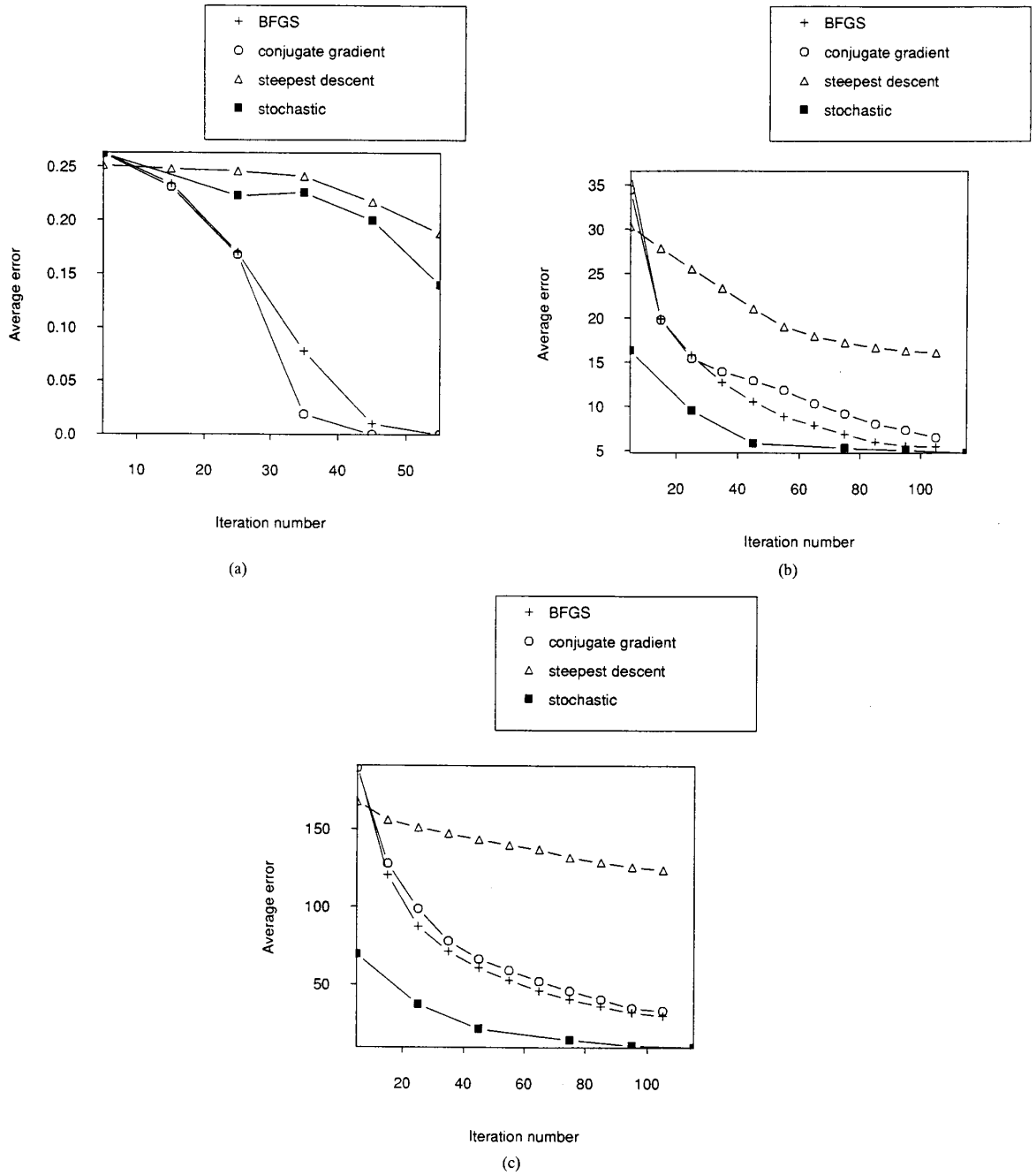

Fig. 4. Comparative optimization performance of various techniques when back-propagation training is used on (a) the exclusive-or problem, (b) the artificial two-dimensional problem, and (c) the aircraft problem.

the order of 0.3 and 1.3, respectively, implying that all the observed differences are statistically significant.

Of the deterministic techniques, the variable-metric technique consistently performs best, as expected. However, the difference between the variable-metric and conjugate-gradient techniques is always relatively small, and as the complexity of the problem increases, this difference decreases. These results indicate that, if a deterministic technique is to be used, the additional computation and storage required for the BFGS method are probably not justified, especially if the neural net is large. The steepest-descent algorithm never performs competitively; this confirms the observation [26] that BP criterion functions have large condition numbers in realistic applications.

The exclusive-or problem could not be implemented sensibly on the AC classifier, since a sufficient number of pro-

totypes to make this nonlinear classification possible (namely three) render the optimization procedure trivial. We therefore tested only the artificial two-dimensional and aircraft data sets on this classifier. Results are shown in Fig. 5. (Note that the vertical axes of Figs. 4 and 5 are not directly comparable, since the different criterion functions penalize the same errors differently.) Since the AC algorithm does not start from random initial weights, the results shown in Fig. 5 represent the outcome of a single trial. On the two-dimensional data set (Fig. 5(a)), the steepest-descent algorithm again performs poorly, with the performances of the other three algorithms being comparable. As before, the stochastic technique outperforms the deterministic techniques on the aircraft data, with the BFGS algorithm performing somewhat better than the conjugate-gradient technique, and the steepest-descent algorithm converging only very slowly.

Thus, the relative performances of the stochastic and deterministic algorithms are approximately the same when this neural net is used as when BP is used. For this classifier there seems to be somewhat more reason to prefer the BFGS algorithm to the conjugate-gradient technique, although the advantage is still not compelling.

We also compared the approximate computational complexities of the various algorithms by comparing their execution time on the same system (a Sun 4/260) using similar implementations compiled on the same complier (the gnu-c compiler). (Note that such a statistical comparison is required because all the algorithms include steps which are executed only if certain conditions are met by the data at a given iteration, which implies that the actual computation on every iteration is not constant.) Average execution times per iteration for the BP classifier on the artificial and aircraft data sets are listed in Table II. It is unfortunately impossible to ensure that the implementations of all algorithms are exactly equally efficient; the values in Table II should therefore be taken as no more than an approximate guide of the relative complexities. It is clear that on problems such as those considered in Table II there is not a large difference between the computation times for the various algorithms considered. This is because the calculation of the gradient of the criterion function consumes most of the computation time, and all algorithms have this process in common. Thus, it is fair to compare these algorithms on the basis of results such as those shown in Fig. 4 and Fig. 5 (as long as the training set is large enough for the gradient calculation to dominate over the update computation).

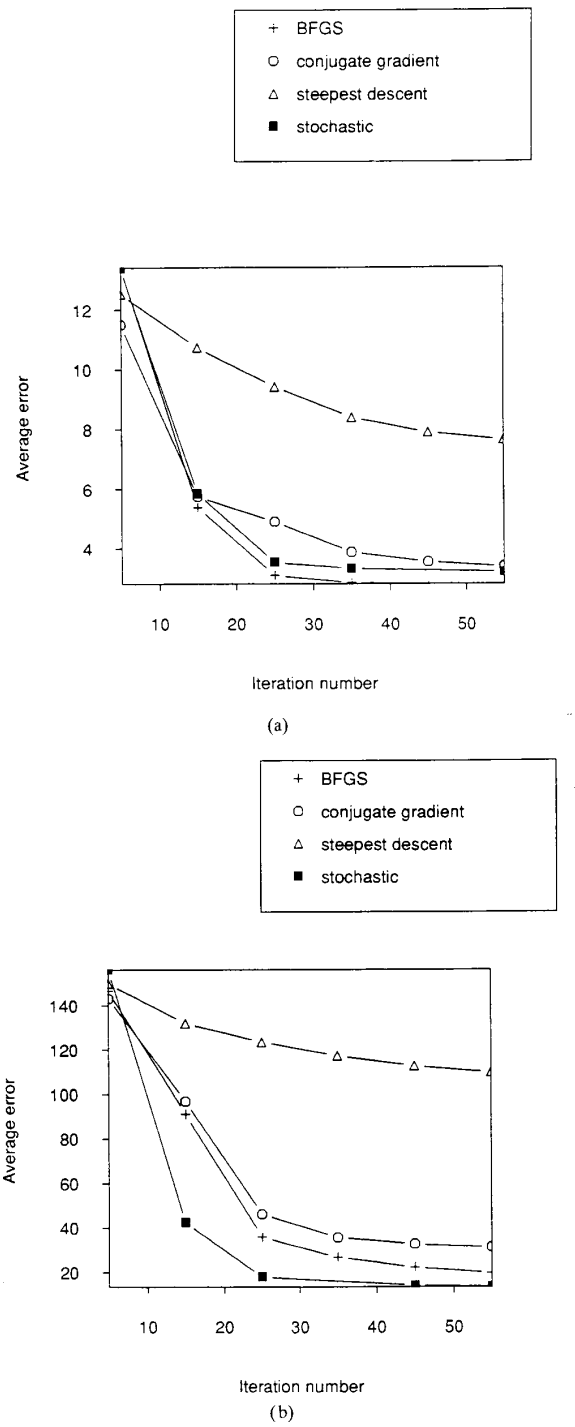

Fig. 5. Comparative optimization performance of various techniques when adaptive-clustering training is used on (a) the artificial two-dimensional problem (b) the aircraft problem.

V. Conclusion

We have seen that a large number of optimization techniques have been proposed for neural-net training. The most important distinction between the different types of algorithms rests on whether they perform in sequential or batch mode. Methods from the latter category have the advantage that they can utilize the accumulated wisdom regarding numerical optimization techniques; however, the fact that they update the neural net only after a complete sweep through the training set is an important disadvantage. For this reason we introduced an optimization technique which operates in sequential mode, yet does not require the choice of any arbitrary parameters or extensive computation after every presentation of a training sample.

TABLE II
AVERAGE EXECUTION TIMES FOR A SINGLE ITERATION OF BP USING DIFFERENT OPTIMIZATION TECHNIQUES ON TWO DATA SETS

Optimization Technique	Average Time per Iteration on Artificial Data (s)	Average Time per Iteration on Aircraft Data (s)
BFGS	0.34	11.8
Conjugate Gradient	0.27	8.4
Gradient Descent	0.30	8.5
Stochastic	0.33	9.5

Our simulation results indicate two important conclusions: the sequential algorithm introduced in Section III is the most efficient of the techniques we studied when the number of training samples becomes large, and the additional expense of variable-metric techniques over the conjugate-gradient technique with restarts does not seem to be warranted, especially if a BP classifier is implemented. It should be noted that the efficiency of the sequential algorithm derives from a reduced number of iterations for convergence, which more than compensates for the fact that each iteration takes somewhat longer than the same for the fastest deterministic techniques (see Table II).

This research suggests a number of interesting extensions. It might be useful to design a sequential algorithm, for instance, which uses a more sophisticated update equation than (2). If for example, a term similar to the "momentum" term of BP is added to the update equation, there is one additional parameter which has to be estimated during the deterministic estimation phase; if this parameter can be estimated accurately, a speedup similar to the difference between the conjugate-gradient algorithm and the steepest-descent algorithm might be obtained. A different extension worth considering is the application of the sequential algorithm to certain problems where extended Kalman filtering has traditionally been used, namely those problems where the amount of computation between sample presentations must be limited.

REFERENCES

[1] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* New York: Wiley, 1973.

[2] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar signals," *Neural Networks,* vol. 1, no. 1, pp. 75–90, 1988.

[3] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang," Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech, Signal Process.,* vol. 37, pp. 328–339, Mar. 1989.

[4] E. Barnard and D. Casasent, "A comparison between criterion functions for linear classifiers, with an application to neural nets," *IEEE Trans. Syst., Man, Cybern.,* vol. 19, pp. 1030–1041, Sept./Oct. 1989.

[5] D. J. Burr, "Experiments on neural net recognition of spoken and written text," *IEEE Trans. Acoust., Speech, Signal Process.,* vol. 36, pp. 1162–1168, July 1988.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel Distributed Processing,* Cambridge MA: MIT Press, 1986, vol. I, ch. 8, pp. 318–362.

[7] R. Hecht-Nielsen, "Counterpropagation networks," *Appl. Opt.,* vol. 26, no. 23, pp. 4979–4984, 1 Dec. 1987.

[8] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," *Proc. 1988 Connectionist Models Summer School* Carnegie Mellon University, Pittsburgh, PA, June 1988, pp. 38–51.

[9] S. A. Solla, E. Levin, and M. Fleisher, "Accelerated learning in layered neural networks," Tech. Rep. ATT Bell Laboratories, 1988.

[10] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with neural nets: Benchmarking studies," in *Proc. Conf. Neural Networks* San Diego, CA, July 1988, pp. I-61–I-68.

[11] J. P. Cater, "Successfully using peak learning rates of 10 (and greater) in back-propagation networks with the heuristic learning algorithm," *Proc. Conf. Neural Networks* (San Diego, CA), July 1987, pp. II–645–II-651.

[12] T. P. Vogl, J. K. Mangis, A. K. Zigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Bio. Cybern.,* vol. 59, pp. 257–264, Sept. 1988.

[13] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks,* vol. 1, no. 4, pp. 295–308, 1988.

[14] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization.* London: Academic Press, 1981.

[15] A. I. Selverston, "A consideration of invertebrate central pattern generators as computational databases," *Neural Networks,* vol. 1, no. 2, pp. 109–118, 1988.

[16] R. L. Watrous, "Learning algorithms for connectionist networks: applied gradient methods of non-linear optimization," in *Proc. Conf. Neural Networks* (San Diego, CA), July 1987, pp. II-619–II-627.

[17] A. Lapedes and R. Farber, "How neural nets work," in *Neural Information Processing Systems,* D. Z. Anderson, Ed. Denver, CO: AIP, 1988, pp. 442–456.

[18] P. J. Werbos, "Backpropagation: Past and future," in *Proc. Int. Conf. Neural Networks* San Diego, CA, July 1988, pp. I-343–I-354.

[19] M. L. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry* (expanded edition). Cambridge, MA: MIT Press, 1988.

[20] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

[21] M. J. D. Powell, "Restart procedures for the conjugate gradient method," *Mathematical Programming,* vol. 12, pp. 241–254, Apr. 1977.

[22] H. Robbins and S. Munroe, "A stochastic approximation method," *Ann. Math. Statist.,* vol. 22, no. I, pp. 400–407, 1951.

[23] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended kalman algorithm," in *Proc. ICASSP* (Glasgow, Scotland), May 1989.

[24] E. Barnard and D. Casasent, "Image processing for image understanding with neural nets," in *Proc. Int. Joint Conf. Neural Networks* (Washington, DC), June 1989, pp. I-111–I-116.

[25] G. Strang. *Introduction to Applied Mathematics.* Cambridge, MA: Wellesley-Cambridge, 1986.

[26] A. J. Owens and D. L. Filkin, "Efficient training of the back propagation network by solving a system of stiff ordinary differential equations," in *Proc. Int. Joint Conf. Neural Networks* (Washington, DC), June 1989, pp. I-381–II-386.

Etienne Barnard received the B.Eng degree in electronics from the University of Pretoria, South Africa, the degrees B.Sc.(Hons) and M.Sc. in physics from Witwatersrand University, South Africa, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA. He is currently Assistant Professor of Electronics and Computer Engineering at the University of Pretoria.

His research interests include pattern recognition, with applications to speech and image understanding, neural networks, and computer-generated holography. He has acted as consultant to industry in all of these fields. Dr. Barnard is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS.