PARALLEL TANGENT LEARNING ALGORITHM FOR TRAINING ARTIFICIAL NEURAL NETWORKS

by

Ali A. Ghorbani and Virendra C. Bhavsar

TR93-075 April 1993

Faculty of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3

Phone: (506) 453-4566
Fax: (506) 453-3566

Parallel Tangent Learning Algorithm for Training Artificial Neural Networks

Ali A. Ghorbani and Virendra C. Bhavsar

Parallel/Distributed Processing Group

Faculty of Computer Science

University of New Brunswick

Fredericton, N.B., Canada E3B 5A3

Phone:+1 (506)453-4566, Fax:+1 (506)453-3566

Email: ghorbani@vermis.cs.UNB.ca bhavsar@UNB.ca

April 22, 1993

Key words: Artificial Neural Networks, Backpropagation, Deflecting Gradient, Gradient Descent, Multilayer Feedforward Networks, Parallel Tangent.

Abstract

A modified backpropagation training algorithm using deflecting gradient technique is proposed. Parallel tangent(Partan) gradient is used as a deflecting method to accelerate the convergence. This method can also be thought as a particular implementation of the method of conjugate gradient. Partan gradient consists of two phases namely, climbing through gradient and accelerating through parallel tangent. Partan overcomes the inefficiency of zigzagging in the conventional backpropagation learning algorithm by deflecting the gradient through acceleration phase. The effectiveness of the proposed method in decreasing the rate of convergence is investigated by applying it to four learning problems with different error surfaces. It is found through simulation that regardless of the degree of the complexity of the problems used, the Partan backpropagation algorithm shows faster rate of convergence to the solution. In particular, for the exclusive-or problem its convergence time is approximately five times faster than that of standard backpropagation, whereas about two times faster rate of convergence is obtained for Encoder/Decoder, Binary-to-local, and Sonar problems.

1 Introduction

The commonly-used error backpropagation(BP) training algorithm offers a simple way to learn arbitrary complex decision boundaries. It is an iterative gradient descent algorithm designed to train multilayer feedforward networks of sigmoid nodes by minimizing the mean square error between the actual output of the network and the desired output. Despite its popularity and effectiveness, its convergence, however, tends to be extremely slow. The main objective of this paper is to incorporate an acceleration technique into the BP algorithm for achieving faster rates of convergence.

Over the last couple of years, many new acceleration techniques have been developed to speedup the rate of convergence in the backpropagation training algorithm. The most popular of these strategies is to include a momentum term in the weight updating phase [15]. Other techniques include:

- Global learning rate adaptation with different variations, where proper values for learning rates and momentum factors are chosen to optimize the backpropagation training algorithm[2, 9, 12].

- Local learning rate adaptations, wherein independent learning rates are used for every connection and optimal learning rates for every weight is found[10, 6, 7]. One variation is to setup schedule to change the step length as the network is learning [10].

In this paper, we propose a global error gradient adaptation technique. Other acceleration techniques can also be incorporated in this method to further improve the rate of convergence.

The outline of the paper is as follows. In the following section, gradient descent method is summarized. In Section 3, the concept of parallel tangent gradient is described and an algorithm for minimizing a differentiable function by using the parallel tangent scheme is given. The error backpropagation learning algorithm is briefly outlined in Section 4 using our notations. Subsequently, the effect of incorporating Partan in backpropagation algorithm and a new learning algorithm based on these two schemes are given. In Section 6, we describe the simulations performed in this study and analyze the results of applying the Partan backpropagation and conventional backpropagation on four different problems. Finally, the conclusions of present study are given.

2 Gradient Descent

The gradient is a simple method for optimizing multidimensional problems [8, 14, 16, 17]. Although this method often poses problems of convergence, it is by far the most widely applied, because of its simplicity. The vector of first partial derivatives of a function $f(x_1, x_2, \ldots, x_n)$, as the n-dimensional vector,

$$\nabla f(\mathrm{x}) = [\frac{\partial f}{\partial x_1}(\mathrm{x}), \frac{\partial f}{\partial x_2}(\mathrm{x}), \frac{\partial f}{\partial x_3}(\mathrm{x}), ..., \frac{\partial f}{\partial x_n}(\mathrm{x})]^T$$

defines a direction called the gradient direction, which is the basis of all gradient algorithms. This vector points in the direction of the greatest rate of increase of function f. Optimum gradient results when the gradient direction of each point is followed until a local minimum is reached [17]. When the method is used for minimizing a differentiable function of several variables, it is sometimes called the steepest descent method.

In general, the gradient algorithm takes a point $p_i \in S \subset E^n$ and computes a new point $p_{i+1} \in S \subset E^n$, where S represent an arbitrary set and E represents the Euclidean space. The new point is defined by making

$$p_{i+1} = p_i + \eta \mathrm{s}$$

where, $\eta > 0$ for minimization or $\eta < 0$ for maximization. Further, p_i is the origin of the line, s is the gradient vector,$\nabla f(p_i)$, determining the direction and η is the step-size parameter to be estimated. If η is small, the gradient path is approximated very closely, but convergence will be slow. For large η, convergence will initially be very fast, but the algorithm will oscillate about the optimum.

It is interesting to note that in principle this method will not reach the optimum in a finite number of steps, because the steps shorten as the point is approached. However, the optimum can be approached as closely as desired, and if the starting point is not too near the major axis, the neighborhood of the optimum is attained rapidly.

The method of steepest descent usually works quite well during stages of the optimization process [14]. However, as an optimum point is approached, the method usually behaves poorly, where small orthogonal steps are taken (zigzagging phenomena). To illustrate the zigzagging phenomena, let us consider an objective function with concentric ellipsoidal contours as shown in Fig. 1. If the initial point for a gradient search happens to be precisely on one of the axes

of the system of ellipses, the gradient line will pass right through the optimum(peak) and the search will be over in one descent (ascent). Otherwise the search will follow a zigzag course such as the one from p_0 to p_2 to p_3 to p_4,etc.(in order to be consistent with the convention adopted in the next section, after p_0 the next point is denoted as p_2, instead of p_1).

It can be seen that the crooked path is bounded by two straight lines which intersect at the optimum. This suggests that the search from point p_3 be conducted, not in the gradient direction toward p_4, but along the straight line from p_0 through p_3. In this way, the peak, p^* would be located after three steps: first from p_0 to p_2 along the gradient at p_0, then from p_2 to p_3 along the gradient at p_2, and finally from p_3 along the line through p_0 and p_3. This is the two dimensional version of a method which accelerates along a ridge and usually is called gradient parallel tangents(gradient Partan) [16].

To accelerate the convergence of steepest descent based learning algorithm(i.e., delta rule and generalized delta rule, error backpropagation learning algorithm), we propose the use of parallel tangent(Partan) gradient. This technique improves the speed of training artificial neural networks by a large factor, as demonstrated in this paper.

3 Parallel Tangent(Partan) Gradient

The parallel tangent technique combines many desirable properties of the simple gradient methods [8, 16]. It makes use of the geometric properties of the quadratic objective functions and is best suited and works well with elliptical contours. Even when the contours are not precisely elliptical and for more general cases and nonquadratic problems, the technique has certain ridge-following properties which make it attractive.

Parallel tangent has many forms and gradient Partan is one form which amounts to a multidimensional extension of the accelerated gradient method. This technique represents a distinct improvement over the method of steepest descent. The Partan method is not really intended for two dimensions, though in one of our experiment it was quite useful and well suited.

Figure 2 shows a schematic diagram of general parallel tangent . Note that the points have been numbered such that the odd-numbered ones (i.e., p_3, p_5, p_7, p_9, etc.) are the result of a climb(gradient search), whereas the even-numbered ones following p_2 (that is, p_4, p_6, p_8, etc.)

are obtained by acceleration. In other words, the even numbered point p_{2k} is determined by acceleration from P_{2k-4} through $p_{2k-1}, k = 2, 3, ..., N$, i.e.,

$$p_{2k} = \Omega(p_{2k-1}, p_{2k-4})$$

where Ω is the acceleration function. Acceleration is the process of taking the minimum point on the line connecting p_{2k-1} and P_{2k-4}.

A procedure for minimizing a differentiable function of several variables is given in Figure 3. This procedure starts with an arbitrary starting point and searches for an optimum using a positive termination scalar, ε. Starting at point p_0, point p_2 is found by a standard gradient step. After that, the optimization is continued for n iterations and then restarted with a standard gradient descent. Each step consists of a standard gradient descent search followed by an acceleration.

4 Backpropagation Algorithm

In this section we briefly review the backpropagation algorithm and introduce our notations for use in the following section. In the backpropagation training algorithm the gradient descent method is used to minimize the difference between the actual output state vector of the network and the target state vector(criterion function) in a multilayer feedforward neural network. Let L represent the number of layers in the network, N_ℓ represent the number of nodes in layer ℓ where $\ell = 1, 2, 3, \ldots, L$, and n_ℓ represent an arbitrary node in layer ℓ, with $n_\ell = 1, 2, 3, \ldots, N_\ell$. Let $W^{\ell,\ell+1}_{n_\ell,n_{\ell+1}}$ represent the connection strength between node n_ℓ in layer ℓ and node $n_{\ell+1}$ in layer $\ell + 1$. Let $net^\ell_{n_\ell}$ and $A^\ell_{n_\ell}$ represent the net-input and activation values of node n_ℓ in layer ℓ, respectively. Let E represent the global error of the network and K represent the number of training examples required for the network to generalize. Finally let $D^{L,k}_{n_\ell}$ represent the n_ℓ-th pattern of desired output of k-th training example at the output layer(L-th layer).

The objective function to be minimized is the global error of the network which is defined as

$$E = \frac{1}{2}\sum_{k=1}^{K} E_k = \sum_{k=1}^{K}\sum_{n_\ell=1}^{N_\ell} (A^{L,k}_{n_\ell} - D^{L,k}_{n_\ell})^2$$

where

$$net^{\ell}_{n_\ell} = \sum_{n_{\ell-1}=0}^{N_{\ell-1}} W^{\ell-1,\ell}_{n_{\ell-1},n_\ell} A^{\ell-1}_{n_{\ell-1}},$$

and

$$A^{\ell}_{n_\ell} = f(net^{\ell}_{n_\ell}).$$

The most frequently used activation function f is the logistic function

$$f(x) = \frac{1}{1+e^{-x}}$$

or its symmetric version

$$f(x) = tanh(x).$$

The learning is the process of modifying the weights in order to decrease the criterion function, E, which is a function of all $W^{\ell,\ell+1}_{n_\ell,n_{\ell+1}}$ weights. The process looks for a solution weight vector, W^*, that satisfies the minimum distance criterion. Backpropagation learning algorithm adjusts all connection strengths(weights) of a multilayer feedforward network after each epoch (where an epoch consists of presentation of a subset or the whole set of the training examples) based on the negative direction of the error gradient,

$$\Delta W^{\ell,\ell+1}_{n_\ell,n_{\ell+1}} = -\eta \frac{\partial E}{\partial W^{\ell,\ell+1}_{n_\ell,n_{\ell+1}}}$$

where η is the learning rate factor. The partial error derivative of the hidden units are computed by the chain rule, leading to the generalized delta rule, as described in [15]. To accelerate the convergence, a momentum term can be added in the adjustment of coupling strengths. In this case, a constant α is used as the relative contribution of the previous change of the coupling strength. Thus, incorporating a momentum in the above weight updating rule, the weights are updated according to the following rule:

$$\Delta W^{\ell,\ell+1}_{n_\ell,n_{\ell+1}}(s+1) = -\eta \frac{\partial E}{\partial W^{\ell,\ell+1}_{n_\ell,n_{\ell+1}}(s)} + \alpha \Delta W^{\ell,\ell+1}_{n_\ell,n_{\ell+1}}(s),$$

where α stands for the momentum factor and s represents the epoch number.

In practice, the backpropagation algorithm has proved to be a suitable method in computing a weight vector that enables the network to perform certain input-output mappings [15]. Standard backpropagation algorithm (generalized delta rule) has the reputation of being very slow.

It suffers from major drawbacks which are associated with steepest descent technique(explained above). Several acceleration techniques have been proposed to overcome this major deficiency [7, 10, 13]. In the next Section, we propose a modified backpropagation training algorithm using deflecting gradient technique.

5 Partan Backpropagation

Backpropagation training algorithm teaches a network iteratively. The learning takes place through the optimization of a criterion function which is defined to be the sum of the criterion functions of the training examples. In other words, in order to properly train the network, an objective function which is the result of the contributions of all training samples, is simultaneously minimized.

The parallel tangent technique can be incorporated in the backpropagation algorithm. The acceleration function, Ω, will be used to navigate safely without too much inefficient zigzagging. In fact, parallel tangent overcomes the difficulties of zigzagging by *deflecting* the gradient. It can be used with both online (incremental) and batch (cumulative) weight adjustment strategies. Batch update strategy uses the total gradient which means that the weights are updated only after all patterns in the training set have passed through the network. The online update scheme uses the partial gradients rather than the total gradient. It travels over the surface in a direction determined by a single input to the system and does not take into account that another pattern may influence the network in a completely opposite direction.The general framework of the new training technique is defined as follows:

WHILE $(error >= \varepsilon)$ **DO**
  CALL *Forward-Pass* procedure;
  CALL *Backward-Pass* procedure;
  IF (*Odd Points*)
    CALL *Acceleration* procedure;
  CALL *Weight-Update* procedure;
END

A proposed detailed algorithm for Partan backpropagation is given in Figure 4. In this

procedure we are recommending two different step-sizes(learning rates). One learning rate for the *climbing* through gradient, η, and the other for *accelerating* through parallel tangent, μ, are used. We also recommend the use of variable step-sizes, specially for the acceleration step-size.

The Partan-backprop algorithm starts with an arbitrary weight vector, $\mathbf{w}_0$, and finds weight vector $\mathbf{w}^*$ (see Fig. 4) by a standard gradient descent (backpropagation) step. After that, n iterations are carried out. Each cycle consists of a gradient descent search followed by an acceleration. The procedure will stop whenever the termination point is reached. After n iterations, one has the choice of either continuing the cycle of backpropagation search and acceleration or starting over again. In the Figure 4, we have presented the latest choice (start over).

6 Simulation

In order to evaluate the performance of the Partan and conventional backpropagation learning algorithms, simulations are carried out on four learning problems: exclusive-or(xor), 8-bit encoder-decoder, binary-to-local [10], and sonar classification[5]. These problems are chosen because, they possess different error surfaces and collectively, represent an environment that is suitable to determine the effect of the proposed learning algorithm. As stated previously, the experiments are carried out with both conventional and Partan backpropagation using total gradient (batch update) so that the weights are updated only after all the patterns in the training set have been presented to the networks. For both algorithms, the backpropagation procedure was used to calculate the partial derivatives of the error with respect to each weight.

The network architectures are predetermined, specifying the number of hidden units, the step sizes ηs and μs, the number of patterns in the training set, and the convergence criterion ε, which was set so that the average error per pattern in the training set is bellow some threshold. For conventional backpropagation networks, we have selected the architectures (multilayer networks) and parameters(gain and momentum rates) that resulted in good performance. The same parameters and architectures are used for Partan backprop to exhibit its superiority over the conventional backprop. Ideal parameters and architectures for Partan may even show faster rate of convergence. For example, when the Partan parameters are tuned for the xor problem, its convergence is almost five times faster than conventional.

At the start of each simulation, the weights are initialized to random values between $+\tau$ to $-\tau$. Since the backpropagation algorithm is sensitive to different starting points, we carried out our simulation with various runs starting from different random initialization for the weights of the network. At the end of each epoch, the weights of the network were updated. For each algorithm twenty simulations were attempted, some of which have converged to local minima and failed to reach the solution criteria. The statistics on the number of epochs includes those attempts that were successful and reached optimum.

6.1 Exclusive-Or Problem

The network architecture used for solving this problem consists of two input units, two hidden units, and one output unit. The training is considered complete when the cumulative error is below 0.1. The conventional backpropagation method using a $\eta = 5.0/\textit{fan-in}$ and $\alpha = 0.9$ required on average 86.22 (standard deviation σ=25.36) epochs through the four input vectors while the modified method(Partan backprop) using an $\eta = 5.0/\textit{fan-in}$ and an $\mu = 0.9$ without using momentum in the backprop procedure, required on average 45.76 epochs($\sigma = 17.25$). Thus, Partan backprop shows faster rate of convergence to the solution. When learning parameters tuned carefully the Partan backprop exhibit even faster convergence, on average 17.38 epochs (σ=5.85).

Exclusive-Or Problem								
	Learning Parameters				Number of Trails		Number of Epochs	
Algorithm	$\eta/\textit{fan-in}$	α	μ	$\pm\tau$	Attempted	failed	Average	Standard Dev.
Conventional	5.0	0.9	0.0	0.5	25	2	86.22	25.36
Partan	5.0	0.0	0.9	0.5	25	1	45.76	17.25
Partan	8.0	0.0	0.9	2.0	25	3	17.38	5.85

6.2 Encoder/Decoder Problem

For this task, the network architecture consisted of eight inputs which were connected to three hidden units which were connected to eight output units. The input to the network consists of eight data line in which one of the data line is one(i.e., $00000001, 00000010, \ldots, 10000000$). The

output of the network is the same as input. This is an example of auto-associative networks by which a set of orthogonal input vectors are mapped to a set of orthogonal output vectors through a small set of hidden units [1] . This network produces target outputs exactly similar to the inputs. Thus, the network is to learn an encoding of a 8-bit input vector into a 3-bit pattern and its decoding into a 8-bit output vector. Solution to the problem occurs when the cumulative errors for one epoch is less than 0.2. On this problem, the Partan backpropagation exhibited a significant improvement in the rate of convergence. It shows approximately 50% of decrease in the number of epochs over the conventional backpropagation. With tuned parameters more improvement can be expected.

Encoder/Decoder Problem								
Algorithm	Learning Parameters				Number of Trails		Number of Epochs	
	$\eta/fan\text{-}in$	α	μ	$\pm\tau$	Attempted	failed	Average	Standard Dev.
Conventional	2.0	0.5	0.0	1.0	25	–	1475.64	207.77
Partan	2.0	0.0	0.5	1.0	25	–	927.44	59.12

6.3 Binary-To-Local Problem

The network architecture used for this problem consisted of three input units, two units in the first hidden layer, eight units in the second hidden layer, and eight output units. The input vector represent a value between zero to seven and the output vectors components are all zero except the one that is indicated by the input vector and is set to one(i.e., input vector $[100]^T$ produces the output vector $[00010000]^T$). This problem is called binary-to-local because, an appropriate network structure maps a binary representation of a number into a local representation. This network is believed to produce an error surface with sharp ravines [10]. Thus it is chosen as an interesting test-bed to experiment the ridge-following property of Partan backpropagation. The training is considered complete when the error for one epoch is less than 0.5. The conventional backpropagation with an $\eta = 0.8/fan\text{-}in$ and $\alpha = 0.5$ represent on average $1166.38(\sigma = 320.80)$ epochs to reach the optimum where as the Partan method with an $\eta = 0.8/fan\text{-}in$ and $\mu = 0.5$, required on average $708.88(\sigma = 68.98)$ epochs.

Binary-to-Local Problem								
Algorithm	Learning Parameters				Number of Trails		Number of Epochs	
	$\eta/fan\text{-}in$	α	μ	$\pm\tau$	Attempted	failed	Average	Standard Dev.
Conventional	0.8	0.5	0.0	0.3	25	1	1166.38	320.80
Partan	0.8	0.0	0.5	0.3	25	1	708.88	68.98

6.4 SONAR, Mines vs. Rocks Problem

This is the last task that is used in our experiments. The training and test sets each consisted of 104 patterns [5]. Each input pattern is a set of 60 numbers in the range of 0.0 to 1.0. The network architecture that is used for this problem consisted of 60 input units, 24 hidden units, and 2 output units. This architecture is reported by Gorman and Sejnowski [5] to be the best architecture that produces good results. Solution to this problem occurs when the error for one epoch becomes less than 0.1.

SONAR, Mines vs. Rocks Problem								
Algorithm	Learning Parameters				Number of Trails		Number of Epochs	
	$\eta/fan\text{-}in$	α	μ	$\pm\tau$	Attempted	failed	Average	Standard Dev.
Conventional	0.5	0.9	0.0	0.3	25	-	892.08	27.73
Partan	0.5	0.0	0.9	0.3	25	1	486.25	37.99

7 Conclusion

In this paper, we have proposed the use of a deflecting gradient method in order to increase the rate of convergence in the backpropagation learning algorithm. The parallel tangent (Partan) gradient is used as a deflecting method to accelerate the convergence to the solution. We have demonstrated through simulation that the proposed method decreases the rate of convergence by a large amount. The modification in the backpropagation algorithm amounts to computing a new point that is closer to the optimum than the point computed by steepest descent procedure.

The new algorithm uses two learning rates, one for gradient search and the other for accelerating through parallel tangent. When backpropagation algorithm is used for gradient search the momentum does not improve the rate of convergence in parallel tangent technique.

With the Partan method, the initial approach as well as the final convergence are faster. In all the problems we studied so far, the convergence of Partan was faster than conventional backpropagation. The most desirable property of Partan backpropagation, however, is its strong global convergence characteristics. Each step of the process is at least as good as steepest descent; the additional move(acceleration) to p_{i+1} provides further decrease of the objective function.

References

[1] Ackley, D. H., Hinton. G. E., and Sejnowski. T. J. 1985. A learning algorithm for Boltzmann machines, *Cognitive Science* **9**, 147-169.

[2] Darken, C. and Moody, J. 1991. Note on Learning Rate Schedules for Stochastic Optimization, In *Neural Information Processing Systems* 3, Lippmann R. P., Moody J. E. and Touretzky D. S.(editors), pp. 832-838, Morgan Kaufmann, San Mateo, CA.

[3] Denker J., et al. 1987. Large Automatic Learning, Rule Extraction, and Generalization, em Complex Systems **1**, 877-922.

[4] Eaton, Harry A. C. and Olivier, Tracy L. 1992. Learning Coefficient Dependence on Training Set Sizes, *Neural Networks*, **5**, 283-288.

[5] Gorman, R. P., and Sejnowski, T. J. 1988. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets, *Neural Networks* **1**, 75-89.

[6] Fahlman, S. 1987. An empirical study of learning in backpropagation networks (CMU-CS-88-162), Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

[7] Fahlman S. E. and Lebiere C. 1990. The Cascade-Correlation Learning Architecture, *Neural Information Processing 2*, pp. 524-532. Morgan Kaufmann, San mateo, CA.

[8] Gottfried B. S. and Weisman J. 1973. *Introduction to optimization theory*, Prentice-Hall, Englewood Cliffs, N.J.

[9] Hertz, J., Krogh, A., and Palmer, R. G. 1991. *Introduction to theory of neural computation*, Addison-Wesley, Redwood City, CA.

[10] Jacobs, R. A. 1988. Increased Rates of Convergence Through Learning Rate Adaption, *Neural Networks* **1**, 295-307.

[11] Hecht-Nielsen R. 1989. Theory of Backpropagation Neural Networks, In *Proceeding of the International Joint Conference on Neural Networks*, pp. 593-605, Washington D.C.

[12] Kramer A. H. and Vincentelli A. S. 1989. Efficient parallel learning algorithms for neural networks, In *Neural Information Processing Systems*1, Touretzky D. S.(editor), pp. 40-48, Morgan Kaufmann, San Mateo, CA.

[13] Leonard, J. and Kramer, M. A. 1990. Improvement of the Backpropagation Algorithm for Training Neural Networks, *Computer Chem. Engng.* **14**(3), 337-341.

[14] Luenberger, D. G. 1973. *Introduction to linear and nonlinear programming,* Addison Wesley, Reading, MA.

[15] Rumelhart D. E., Hinton G. E., and Williams R. J. 1986. Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstuctures of Cognition* **I**, 318-362, MIT press, Cambridge, MA.

[16] Wilde D. J. and Beightler C. S. 1967. *Foundations of optimization*, Prentice-Hall, Englewood Cliffs, N.J.

[17] Wismer D. A. and Chattergy R. 1978. *Introduction to Nonlinear Optimization*, Elsevier North-Holland, Amsterdam.

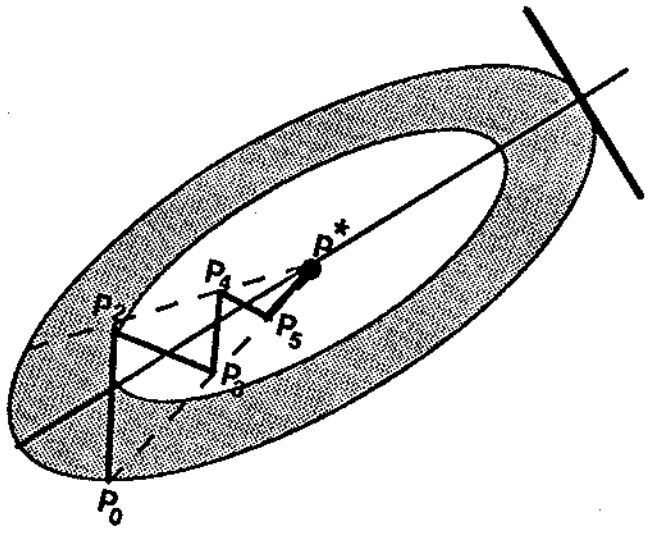

Figure 1. Zigzagging in elliptical contour.

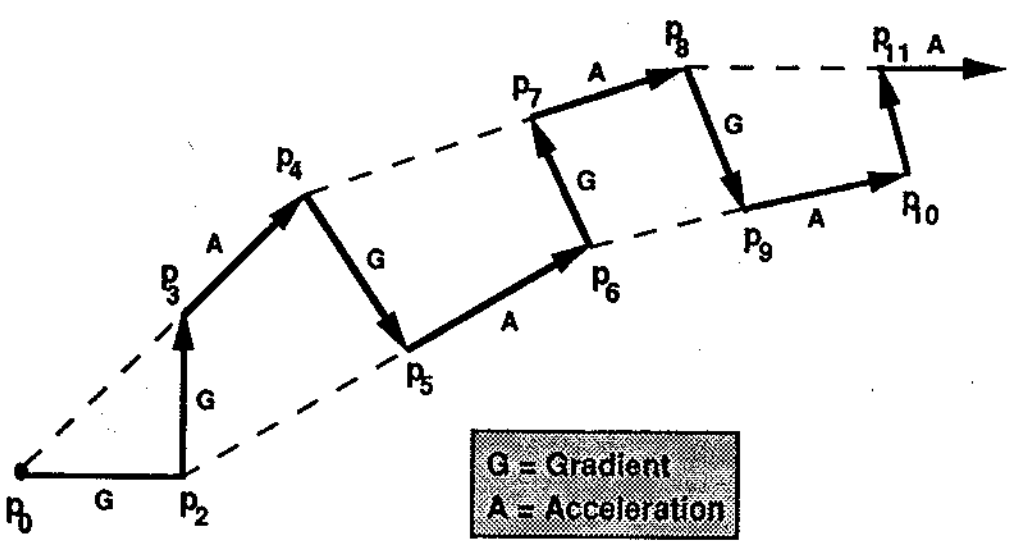

Figure 2. Parallel Tangent Gradient.

PROCEDURE *Partan* (input[$\mathbf{p}_0$, ε], output[$\mathbf{p}^*$]);
$\mathbf{p}_0$ is the starting point vector and ε is a termination scalar which is chosen to be greater than zero. $\mathbf{p}^*$ is the optimum vector. η and μ are appropriate step size factors(learning rates) for climbing and acceleration, respectively. $\nabla f(\mathbf{p}_i)$ is the gradient of the criterion function f at point $\mathbf{p}_i$. Γ represents the function that is used to compute a proper step size. n represents the number of independent variables.

BEGIN
$\mathbf{p}^* = \mathbf{p}_0$;
REPEAT
$\mathbf{p}_k = \mathbf{p}_0 = \mathbf{p}^*$;
$\mathbf{p}^* = \mathbf{p}_k - \eta \nabla f(\mathbf{p}_k)$;
FOR $i = 1$ to n **DO**
BEGIN
$\eta = \Gamma(\eta, \nabla f(\mathbf{p}^*), n)$;
$\mathbf{s} = \mathbf{p}^* - \eta \nabla f(\mathbf{p}^*)$;
$\delta = \mathbf{s} - \mathbf{p}_k$;
$\mathbf{p}_k = \mathbf{p}^*$;
$\mathbf{p}^* = \mathbf{s} + \mu\delta$;
$\mu = \Gamma(\mu, \delta, n)$;
END;
UNTIL $(\|\mathbf{p}_0 - \mathbf{p}^*\| < \varepsilon)$;
RETURN ($\mathbf{p}^*$);
END.

Figure 3. Parallel Tangent Gradient Optimization Algorithm.

PROCEDURE *Partan-backprop* (input[$\mathbf{I}, \mathbf{D}, \mathbf{w}_0, \varepsilon$], output[$\mathbf{w}^*$]);
I and **D** are the input and desired output vectors, respectively. $\mathbf{w}_0$ is the starting point, $\mathbf{w}^*$ is the optimum weight vector, and ε is a termination scalar which is chosen to be greater than zero. η and μ are appropriate step size factors(learning rates) for climbing and acceleration, respectively. ∇_{w_i} is the gradient of the criterion function f at point $\mathbf{w}_i$. n represents the number of independent variables and M is number of iterations(epochs). Γ represents the function that is used to compute a proper step sizes. *backprop* is the standard backpropagation procedure that returns the gradient of the criterion function at a given point and the amount of existing error.

```
BEGIN
    j = 1;
    w* = w0;
    REPEAT
      wk = w0 = w*;
      CALL  backprop(input[I, wk, ε ,D], output [∇wk,error]);
      w* = wk − η∇wk;
      FOR  i = 1 to n  DO
      BEGIN
        CALL  backprop(input[I, w*, ε ,D], output [∇w*,error]);
        IF  (error < ε)  RETURN(w*),  EXIT;
        η = Γ(η, ∇w*, n);
        s = w* − η∇w*;
        δ = s − wk;
        wk = w*;
        w* = s + μδ;
        μ = Γ(μ, δ, n);
      END;
      j = j + 1;
    UNTIL  (error < ε OR j >= M)
    RETURN(w*);
END.
```

Figure 4. Parallel Tangent Backpropagation Learning Algorithm.