Computer Speech and Language (1987) 2, 205–218

An adaptive training algorithm for back propagation networks

L.-W. Chan and F. Fallside

Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, U.K.

Abstract

The effect of the coefficients used in the conventional back propagation algorithm on training connectionist models is discussed, using a vowel recognition task in speech processing as an example. Some weaknesses of the use of fixed coefficients are described and an adaptive algorithm using variable coefficients is presented. This is found to be efficient and robust in comparison with the fixed parameter case, to give fast near optimal training and to avoid trial and error choice of fixed coefficients. It has also been successfully used in a vision processing application.

1. Introduction

As is well known, the applicability of connectionist models has been much improved by the introduction of the back propagation algorithm by Rumelhart, Hinton & Williams (1985). In this, Appendix I, the current weight change $\Delta w_{ij}(n)$ for the weight $w_{ij}(n)$ is made proportional to the gradient in the energy-weight space $\partial E/\partial w_{ij}$ as

$$\Delta w_{ij}(n) = -\eta \partial E/\partial w_{ij} \tag{1}$$

where η is a positive constant known as the "learning rate". This gradient descent method can be enhanced by introducing a "momentum term" from the previous weight change as

$$\Delta w_{ij}(n) = -\eta \partial E/\partial w_{ij} + \alpha \Delta w_{ij}(n-1) \tag{2}$$

where α is another constant. This has the effect of damping oscillations in the training of weights.

This algorithm has been widely used in pattern processing, see, for example, Rumelhart & McClelland (1986). One feature of its use is the sensitivity of training time to the numerical values chosen for η and α. Various choices have been given. For a range of problems Rumelhart & McClelland (1986) quotes ranges of $0{\cdot}05 \leqslant \eta \leqslant 0{\cdot}75$ and $0 \leqslant \alpha \leqslant 0{\cdot}9$. Alternatively, switching between values of coefficients during training has been reported. For example, Cottrell, Munro & Zipser (1987) used $\eta = 0{\cdot}25$ for the first

0885-2308/87/030205+14 $03.00/0

© 1987 Academic Press Limited

10^3 iterations and $\eta = 0{\cdot}01$ for the next 5×10^4 iterations with $\alpha = 0$; and Plaut, Nowlan & Hinton (1986) for a different problem used $\eta = 0{\cdot}0005$, $\alpha = 0{\cdot}5$ for the first 25 iterations and $\eta = 0{\cdot}07$, $\alpha = 0{\cdot}99$ for the remaining iterations. This switching gives improvements in learning speed but the method is problem-dependent and needs numerical tuning for each problem.

Since training in connectionist models can be very slow and the wrong choice of coefficients in the back propagation algorithm can make training exceedingly slow, it is important to look for ways to optimize the choice of coefficients. In this paper we describe an algorithm which employs continuously adaptive coefficients. We have used it in two problem areas, the recognition of vowel sounds in speech for which results are given in this paper and in an image recognition problem (Chan & Fallside, 1986). In both cases it gives a significant improvement over fixed coefficients and is relatively insensitive to the initial values chosen for the coefficients.

The paper first discusses the vowel recognizer system and some of the effects of the choice of η and α on training time, then describes the adaptive algorithm and then gives comparative results for the vowel recognition problem.

2. Vowel recognizer example

This uses data from previous work in a study of vowel recognition using a Boltzmann machine (Prager, Harrison & Fallside, 1986), and in a comparison of the performance of a Boltzmann machine and a back propagation network (Prager & Fallside, 1987). The data comprised 44 vowel spectra from 11 vowels. The speech data is preprocessed by a Fast Fourier Transform and logarithmic compression of the spectra.

The back propagation network has three layers: an input layer with 128×16 "binary" units (this samples the compressed spectra at 128 frequency values up to 5 kHz and allocates it one of 16 amplitude levels); a hidden layer of 59 units and an output layer of eight units. Four of the outputs specify the ordinate and four the co-ordinates of a target vowel in an approximate vowel quadrilateral. The input layer is only connected to the hidden layer and the hidden layer only to the output layer; there are no interconnections within a layer.

The training was started with small random connecting weights. In order to compare the effect of different parameters on the rate of convergence, all the trainings were carried out by using the same initial random connecting weights. In each cycle the 44 vowel spectra were presented in turn to the network and the global cost function E and δ's were evaluated (Appendix 1). Updating of the weights was carried out at the end of each cycle.

3. Some effects of fixed coefficients on training behaviour

3.1. General

For a given input to the net, the cost function defines a surface or energy landscape in the weight space. From equation (1), if the learning rate η is small, the weight vector, viewed as a point in this space, will follow the surface down a line of steepest descent. However, its movement will be correspondingly slow and to speed it up a larger value of η is used. On the other hand, a large value of η may cause oscillations across both sides of ravines. To dampen these oscillations down, the momentum term with coefficient α can be

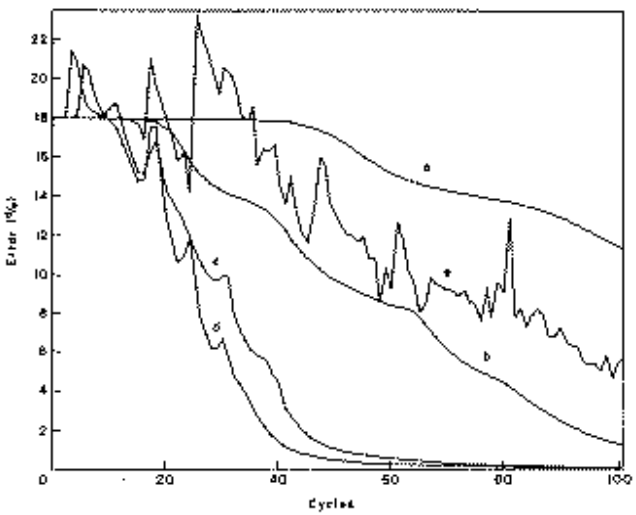

Figure 1. Training with fixed coefficients, $\alpha = 0{\cdot}05$. (a) $\eta = 0{\cdot}02$, (b) $\eta = 0{\cdot}05$, (c) $\eta = 0{\cdot}1$, (d) $\eta = 0{\cdot}2$, (e) $\eta = 0{\cdot}5$.

Broadly, η acts as a "gain" coefficient and α as a "damping" coefficient. Thus, as would be expected when η and α are small, the training behaviour can be speeded up by increasing η, and counteracting oscillations by increasing α. An example of this is shown in Fig. 1 for the vowel recognizer, where, for a fixed $\alpha = 0{\cdot}05$, increasing values of η are used from $\eta = 0{\cdot}02$ to $\eta = 0{\cdot}5$. As can be seen, training is progressively speeded up but oscillations eventually arise. For large η their effect is to slow down training again; in this case there is an optimum choice of about $\eta = 0{\cdot}2$.

Similarly, the value of the damping term α can be found out in the same way to optimize the learning behaviour. It is observed that a large momentum usually tends to speed up the learning when η is small. However, beyond a critical value of η, a large momentum would have a negative effect. Examples of this are shown in Fig. 2 with $\eta = 0{\cdot}2$ increasing α over the range 0·05, 0·2, 0·9 progressively improves the response. However, on increasing η to 0·3, the response deteriorates when α is large.

We see that the training behaviour is sensitive to these coefficients. Although it can be improved by trial and error adjustment of the coefficient values, this adjustment is a slow procedure.

3.2. Some particular effects

Some particular effects of the coefficients on training behaviour are as follows:

Ravines—when η is small, the trajectory would always find its shortest path down the ravines. However, with the increase of η ravines become a problem; fine adjustment of the learning rate is needed, otherwise the trajectory will oscillate from side to side of a

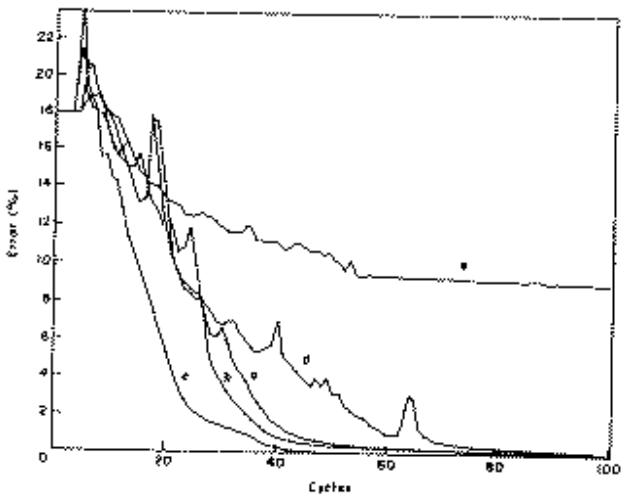

Figure 2. Training with fixed coefficients:
(a) $\eta = 0{\cdot}2$, $\alpha = 0{\cdot}05$,
(b) $\eta = 0{\cdot}2$, $\alpha = 0{\cdot}2$,
(c) $\eta = 0{\cdot}2$, $\alpha = 0{\cdot}9$,
(d) $\eta = 0{\cdot}3$, $\alpha = 0{\cdot}05$,
(e) $\eta = 0{\cdot}3$, $\alpha = 0{\cdot}9$.

ravine, causing slow training. This can be offset by lowering η or by including a momentum term with an appropriate damping factor.

Plateaux—when the search approaches the global minimum along a flat plateau, the small gradient causes very small weight changes and further training becomes very slow, resulting in a long slow tail.

Effective learning rate—during the long tail mentioned above we notice that in the filter of equation (2), when

$$\Delta w_n \rightarrow \Delta w_{n-1} \tag{3}$$

then

$$\Delta w_n \rightarrow \frac{-\eta}{1-\alpha}\nabla E \tag{4}$$

and thus, the effective rate is enhanced by $1/(1-\alpha)$; examples of this are shown in Fig. 3 where the magnitude of $-\Delta w_n/\nabla E \times \eta$ is plotted *vs.* n. It can be seen that after initial training, the steady state of $1/(1-\alpha)$ is approached. It indicates that when η is small, a momentum term with α slightly less than 1 is found to be most beneficial.

Large training rate—large values of η can lead to the build-up of large weight values which tends to give overshoot or even to lock-up units fully on or off, resulting in a very slow movement of the trajectory.

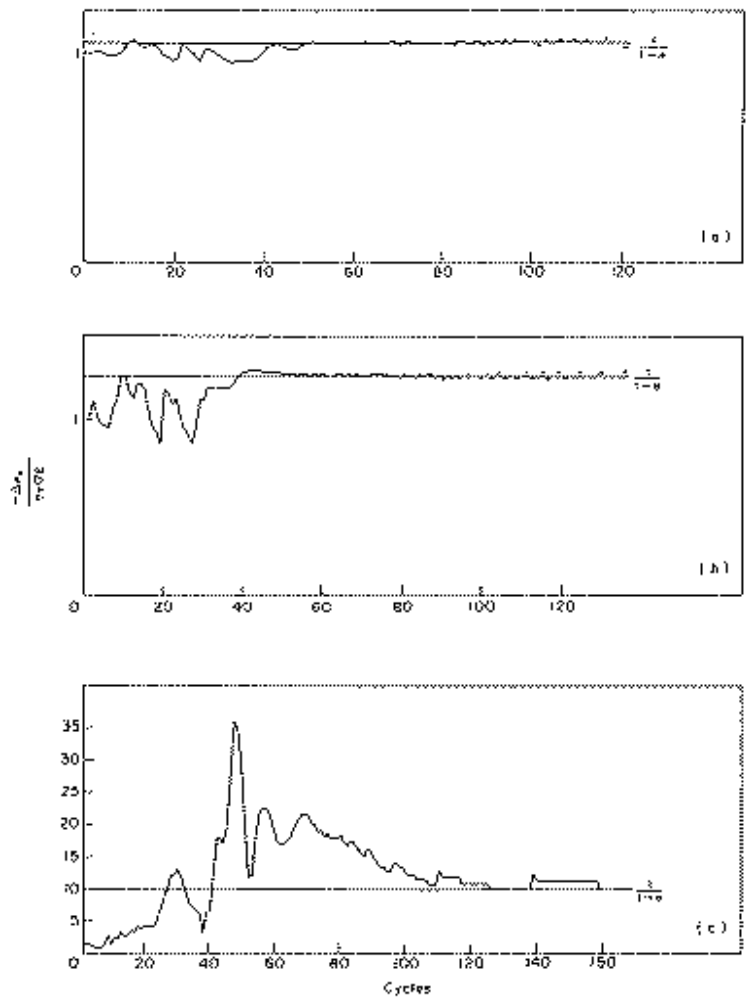

Figure 3. Approach to steady state learning rate, $\eta = 0{\cdot}2$:
(a) $\alpha = 0{\cdot}05$, (b) $\alpha = 0{\cdot}2$, (c) $\alpha = 0{\cdot}9$.

Large momentum term—a momentum term not only offsets oscillations across both sides of ravines but increases the effective rate. However, too large a value will dominate the weight update and the resulting updating direction can deviate far away from the steepest gradient, especially when a steep slope is followed by a gentle slope. The large

deviations can cause the resulting trajectory to move uphill rather than down. Besides, a large effective rate would have the same drawback as a large η. This is demonstrated in Fig. 2c.

3.3. *Descriptions of training by angles in weight space*

Training is dominated by the shape of the energy contour in the weight space. While we do not seek to evaluate this, useful information can be learned about it from the directions of the local gradient vector ∇E_n and the weight updates Δw_{n-1} and Δw_n in the vector version of equation (2).

$$\Delta w_n = -\eta \nabla E_n + \alpha \Delta w_{n-1} \tag{5}$$

Two angles are of interest, θ_n the angle between the current gradient $-\nabla E_n$ and the previous update Δw_{n-1}, given by

$$\cos\theta_n = \frac{-\nabla E_n \cdot \Delta w_{n-1}}{\|\nabla E_n\| \|\Delta w_{n-1}\|} \tag{6}$$

and ϕ_n the angle between successive weight updates, Δw_{n-1} and Δw_n

$$\cos\phi_n = \frac{\Delta w_n \cdot \Delta w_{n-1}}{\|\Delta w_n\| \|\Delta w_{n-1}\|} \tag{7}$$

These angles are shown in Fig. 4. θ_n gives a direct indication of the nature of the energy surface during the training trajectory and $\theta_n - \phi_n$ gives an indication of the smoothing produced by the momentum term.

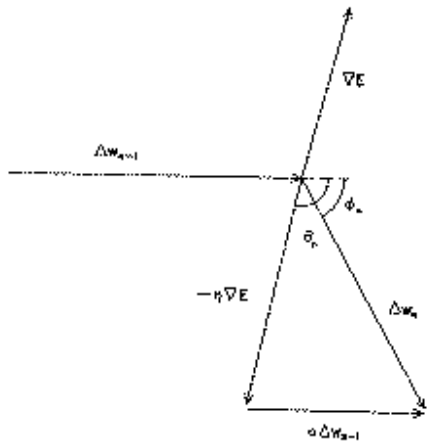

Figure 4. Angles θ_n and ϕ_n in the energy landscape.

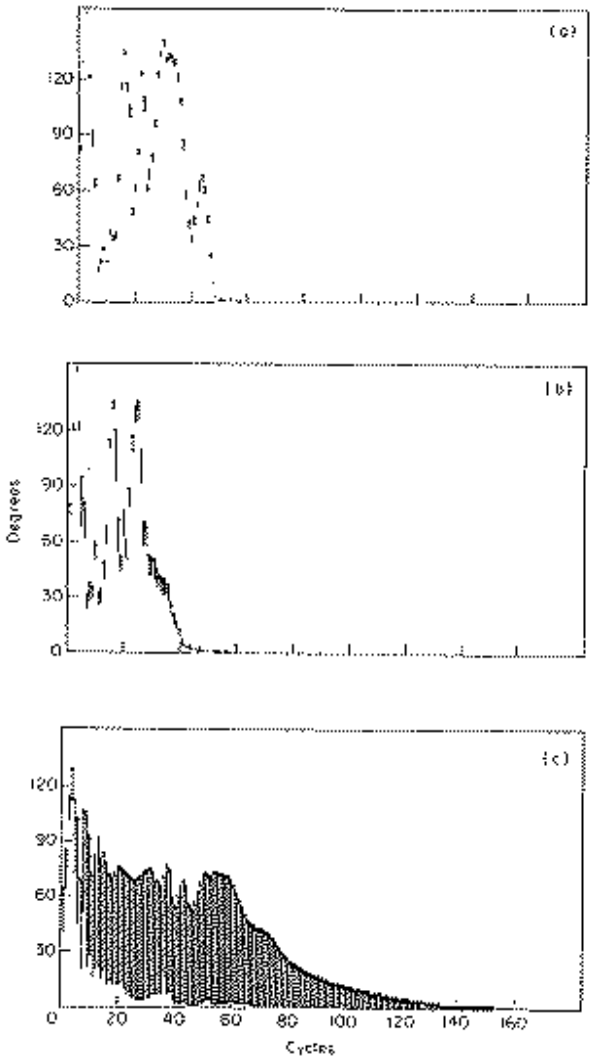

Figure 5. Behaviour of the angles θ_n (indicated by ×) and ϕ_n (indicated by o) during training, $\eta = 0.2$.
(a) $\alpha = 0.05$, (b) $\alpha = 0.2$, (c) $\alpha = 0.9$.

An example of the behaviour of these angles during the training sequences of Fig. 2, for $\eta = 0{\cdot}2$ and $\alpha = 0{\cdot}05$, $0{\cdot}2$ and $0{\cdot}9$ are shown in Fig. 5. For the smallest value of α, the difference in angles $(\theta_n - \varphi_n)$ is small—this indicates that updating is dominated by the gradient term, but as α is increased, the difference $(\theta_n - \varphi_n)$ increases and φ_n is much reduced in magnitude and variation. this shows that the momentum term is smoothing or damping the variations in Δw_n caused by the gradient term.

4. An adaptive algorithm for the training coefficients

In establishing an adaptive algorithm, a number of properties are important: **robustness**—it should not be too problem-dependent; **efficacy**—it should achieve an improvement in training speed; **insensitiveness** to numerical values of coefficients; **computational efficiency**—avoidance of significant additional computational and storage requirements.

The choice of adaptive algorithm established here attempts to eradicate two specific weaknesses of the fixed coefficient algorithm: oscillations across the walls of a ravine and the long tail caused by small gradients in plateaux.

4.1. Adaptation of learning rate coefficient η

Here we look for an expression which will detect the arrival at the wall of a ravine or at a plateau and which will adjust the learning rate accordingly.

From Fig. 6(a) we see that if $90° \leqslant \theta_n \leqslant 270°$ arrival at a ravine wall is likely and η should be decreased to reduce resulting oscillations between walls. Also from Fig. 6(b) we see that if $\theta_n \to 0°$ or $360°$ arrival at a plateau is likely and η should be increased to counteract the small gradient.

One function which satisfies both these requirements is

$$\eta_n = \eta_{n-1}\left(\tfrac{1}{2} + \tfrac{1}{2}\cos\theta_n\right) \tag{8}$$

as shown in Fig. 7.

4.2. Adaptation of momentum coefficient α

Here we try to improve the training performance further by adapting the momentum term in addition to η. The adaptation is to avoid domination of the weight update by the momentum term and to follow the broad trend seen in Section 3.1; that α should be proportional to η for optimum training behaviour.

As a result we choose

$$\alpha_n = \lambda_n \eta_n \tag{9}$$

with

$$\lambda_n = \lambda_0 \frac{\|\nabla E_n\|}{\|\Delta w_{n-1}\|} \tag{10}$$

and

(11)

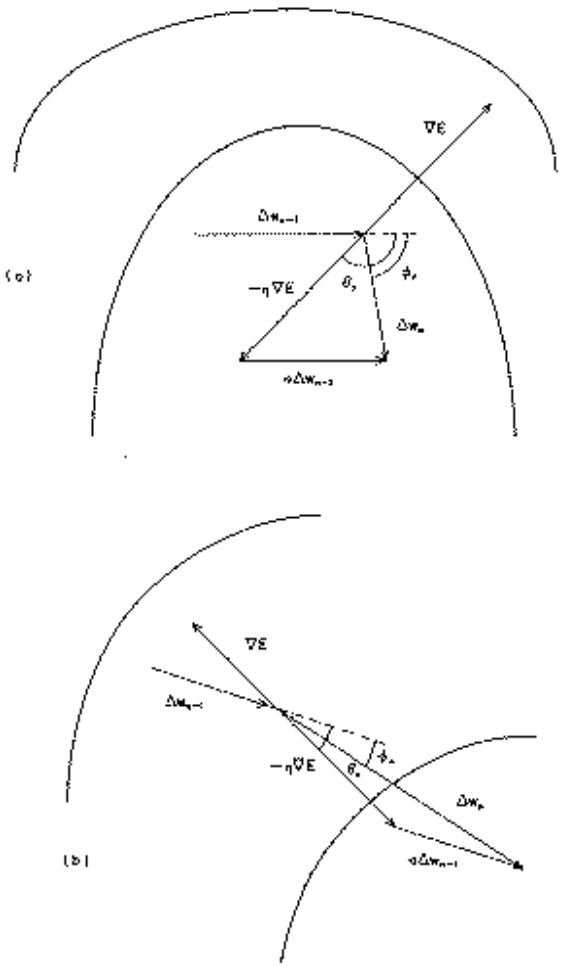

Figure 6. Angles in the energy landscape (a) approaching ravine walls, (b) approaching plateaux.

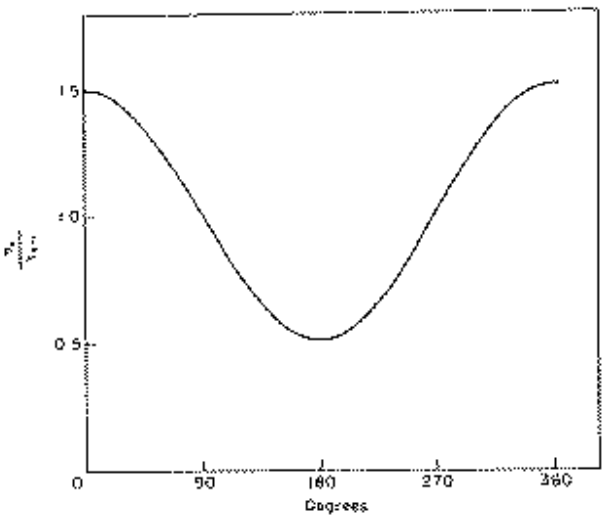

Figure 7. Adaptive learning rate function $\eta_n(\theta_n)$.

This ensures that $\lambda_n < \|\nabla E_n\|/\|\Delta w_{n-1}\|$.

If the momentum term has $\lambda_n \approx \|\nabla E_n\|^2/\|\nabla E_{n-1}\|^2$ the algorithm becomes that of Fletcher & Reeve (1964), see, for example, Scales (1985) for a discussion of this. This was tried but found not to offer an improvement in training.

Thus, finally, the algorithm adopted is

$$\Delta w_n = \eta_n(-\nabla E_n + \lambda_n \Delta w_{n-1}) \tag{12}$$

If the step size is small relative to the energy surface, the fraction $\|\nabla E_n\|/\|\Delta w_{n-1}\|$ shows little variations, and hence λ_n shows little variations. Thus, adaptation of momentum alone offers little improvement if η is fixed at a small value. On the other hand, when η is adapted there would be sudden changes in the gradient of the energy surface. So the adaptation of the momentum term can serve the purpose of regulating the actual amount of damping.

We note now that we have exchanged the choice of η and α in the fixed coefficient case for the choice of η_0 and λ_0. However, we find that training is relatively insensitive to these, as discussed in the next section.

5. Results

Training using the algorithm of equations (8)–(12) has been applied successfully to the vowel recognition problem and to the image recognition problem (Chan & Fallside, 1986). Typical results for the vowel recognition problem are shown in Fig. 8. Here we see that when η_0 is varied over two orders of magnitude and α_0 over one order, the responses are consistently good and relatively insensitive to the choices of η_0 and α_0 whereas responses in the fixed parameter case, Figs 1 and 2, vary significantly and become unacceptable for such variations in η and α. By comparing Figs 1, 2 and 8, the responses of the adaptive method always fall near to the optimal response found for the fixed parameter case.

If η_0 and α_0 are assigned very large values, nails can be locked up in the first cycle. To prevent this locking up, a **binary search technique** can be used. If the cost function E_{n+1} is higher than the previous one (or exceeds it by a certain tolerance, say 1%), we backtrack a step and reduce η_n by half. This technique not only enhances the rapid reduction of any unreasonably high value of η_0 but also prevents overshooting and stabilizes the overall convergency.

We see that the algorithm has successfully reduced oscillations at the walls of ravines and removed the slow tail behaviour. The first of these is seen in Fig. 8 and in the plot of angles w_n of Fig. 9; by comparison with Fig. 5, θ_n is consistently held in the range $0° < \theta_n < 90°$. The second of these is seen in Fig. 8 where the slow tails have effectively disappeared in comparison with the fixed coefficient cases. As shown in Fig. 10, during the main part of the response $\eta_n \simeq 0{\cdot}8$, which is near to the optimal value 0·2, obtained by trial and error. When the global minimum is approached, η_n increases rapidly to compensate for the small gradient and hence gives a reasonable step size for training.

In our updating scheme all training samples have been presented before the weights are updated. This scheme minimizes the overall energy space of all training patterns.

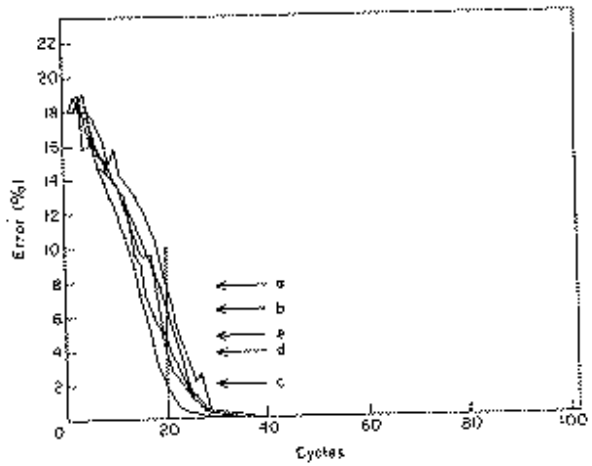

Figure 8. Training with the adaptive algorithm
(a) $\eta_0 = 0{\cdot}05$, $\alpha_0 = 0{\cdot}9$.
(b) $\eta_0 = 0{\cdot}5$, $\alpha_0 = 0{\cdot}9$.
(c) $\eta_0 = 5{\cdot}0$, $\alpha_0 = 0{\cdot}9$.
(d) $\eta_0 = 0{\cdot}5$, $\alpha_0 = 0{\cdot}5$.
(e) $\eta_0 = 5{\cdot}0$, $\alpha_0 = 0{\cdot}5$.

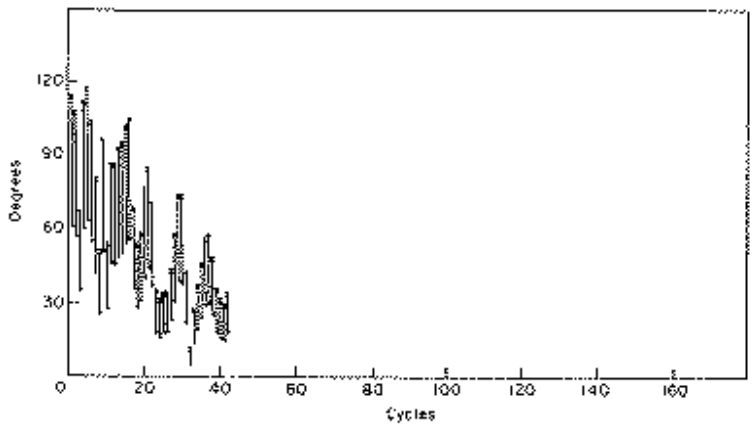

Figure 9. Behaviour of the angles θ_n (indicated by ×) and ϕ_n (indicated by o) during training for adaptive learning.

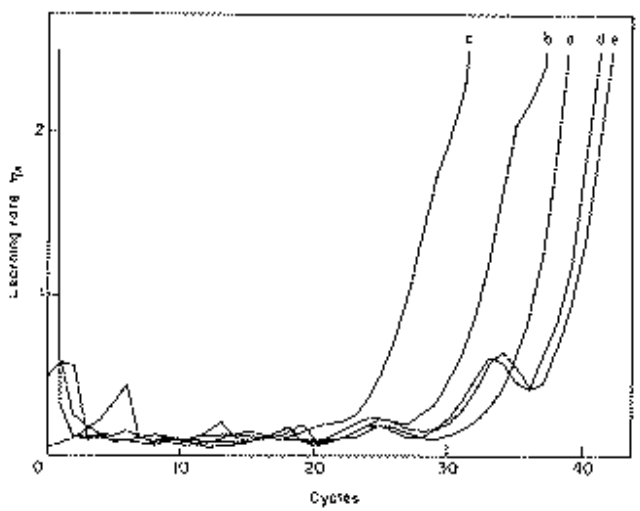

Figure 10. Behaviour of learning rate η_n during training for adaptive algorithm
(a) $\eta_0 = 0{\cdot}05$, $\alpha_0 = 0{\cdot}9$,
(b) $\eta_0 = 0{\cdot}5$, $\alpha_0 = 0{\cdot}9$,
(c) $\eta_0 = 5{\cdot}0$, $\alpha_0 = 0{\cdot}9$,
(d) $\eta_0 = 0{\cdot}5$, $\alpha_0 = 0{\cdot}5$,
(e) $\eta_0 = 5{\cdot}0$, $\alpha_0 = 0{\cdot}5$.

Any conflicting and correlating patterns can be identified at this stage. However, if the number of training patterns is large, this updating scheme may not be feasible. To overcome this problem, we can divide the training data into subsets such that the distribution of each subset is similar to the distribution of all training patterns. In this case, the energy space of each subset can be approximated to the global energy space and hence the angle θ. For example, if the training patterns contain m speeches from n vowel sounds, we can construct subsets containing all n vowels in the same proportion. Each subset is trained in alternative cycles. Thus, the adaptive learning algorithm is still applicable.

6. Conclusions

An adaptive algorithm has been introduced for the training of back propagation networks. It has been applied to two separate pattern processing problems and found to improve training speed and behaviour. It is much more robust to choice of numerical values than is the fixed parameter case, and this in itself can save much trial and error in the choice of values. It appears to successfully avoid oscillatory behaviour in training, caused by oscillations across the walls of ravines, and to eliminate the long slow tails caused by small gradients in plateaux.

The additional computational load of calculating $\|\Delta w_{n-1}\|$, $\|VE\|$ and the dot product $\Delta w_{n-1} \cdot VE$ is insignificant and no extra storage is required over the fixed coefficient case.

Acknowledgement

One of the authors (L.-W. Chan) is supported by a grant from King's College, Cambridge.

References

Chan, L.-W. & Fallside, F. (1986) Connectionist Models and Geometric Reasoning. To be published in *Proc. U.K. Conf. Geometric Reasoning*.
Cottrell, G. W., Munro, P. & Zipser, D. (1987) Image Compression by Back Propagation: An Example of Extensional Programming. *ICS Report 8702*, Institute for Cognitive Science, University of California.
Fletcher, R. & Reeves, C. M. (1964) A rapidly convergent descent method for minimization. *Computer Journal*, 6, 163.
Pinal, D. C., Newlan, S. J. & Hinton, G. E. (1986) Experiments on Learning by Back Propagation. *Technical Report CMU-CS-86-126*. Carnegie-Mellon University.
Prager, R. W. & Fallside, F. (1987) A Comparison of the Boltzmann Machine and the Back Propagation Network as Recognisers of Static Speech Patterns. *Computer Speech and Language*. 2.
Prager, R. W., Harrison, T. D. & Fallside, F. (1986) Boltzmann Machines for Speech Recognition. *Computer Speech and Language*, 1, 3–27.
Rumelhart, D. E. & McClelland, J. L. (eds) (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Bradford Books/MIT Press.
Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1985) Learning Internal Representations by Error Propagation. *ICS Report 8506*. Institute for Cognitive Science, University of California.
Scales, L. E. (1985) *Introduction to Non-linear Optimization*. Macmillan.

Appendix 1

In error propagation networks, the output activity (or the state s_j) of each unit is determined by the activities of the units connected to it from a previous layer and the connecting weights between them, i.e.

$$s_i = F\left(\sum_i w_{ij} s_i + \text{bias}_j\right)$$

where

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

State values are passed from the input layer to succeeding layers until the output layer is reached, forming the feedforward process.

During training, the states of the output units s_j are compared with the target ones t_j and a cost function E is formed,

$$E = \frac{1}{2}\sum_p \sum_j (t_j - s_j)^2$$

This is minimized by the steepest gradient descent method, giving

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

which leads to

$$\Delta w_{ij} = \eta \delta_j s_i \quad \text{and} \quad \Delta \text{bias}_j = \eta \delta_j$$

(This bias term can be treated as a normal weight by assuming that it is a connection to an "on" unit of unit value.)

where

$$\delta_j = s_j(1 - s_j)(t_j - s_j) \quad \text{for output units,}$$

and

$$\delta_i = s_i(1 - s_i)\sum_k \delta_k w_{ki} \quad \text{for hidden units.}$$

In this case δ_k's are propagated backward from upper layers to lower layers. These forward and backward processes are repeated in cycles until the output states match the corresponding target states within a minimum or very small E.