THE "MOVING TARGETS" TRAINING ALGORITHM

Richard Rohwer
Centre for Speech Technology Research, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, SCOTLAND

Abstract

A simple method for training the dynamical behavior of a neural network is derived. It is applicable to any training problem in discrete-time networks with arbitrary feedback. The algorithm resembles back-propagation in that an error function is minimized using a gradient-based method, but the optimization is carried out in the hidden part of state space either instead of, or in addition to weight space. A straightforward adaptation of this method to feedforward networks offers an alternative to training by conventional back-propagation. Computational results are presented for some simple dynamical training problems, one of which requires response to a signal 100 time steps in the past.

1 Introduction

This paper presents an algorithm for training the dynamical behavior of a discrete-time neural network model. In some ways this method is similar to back-propagation. It is based on the calculation of the gradient of an error measure; the error measure is a sum of squares; and the algorithm is intended for pattern classification. But this algorithm is very different in other ways. It applies to networks with arbitrary connectivity (including full feedback), not just feedforward networks. It is intended for pattern completion as well as pattern classification, and makes no formal distinction between the two. It is applicable to dynamical patterns in fully recurrent networks and presents a radically altered framework for treating static patterns in feedforward networks.

The central idea is to treat hidden nodes as target nodes with variable training data. These "moving targets" are varied during the minimization process. Werbos [12] used the term "moving targets" to describe the qualitative idea that a network should set itself intermediate objectives, and vary these objectives as information is accumulated on their attainability and their usefulness for achieving overall objectives. The (coincidentally) like-named algorithm presented here can be regarded as a quantitative realization of this qualitative idea.

The literature contains several temporal training algorithms based on minimization with respect to the weights. These methods are reasonably successful as long as they do not have to respond to distant temporal correlations; it must be possible (in principle) to specify the desired output in terms of the inputs and outputs of the recent past. This type of method includes the straightforward extension of the back-propagation method to back-propagation through time [10], the methods of Rohwer and Forrest [8], Pearlmutter

[5], and the forward propagation of derivatives [13, 4]. A careful comparison of moving targets with back-propagation in time and teacher forcing appears in [14].

The training problem is formalized in section 2, and the general scope of the formalism is noted. The algorithm is presented in section 3, and an analogous algorithm for feedforward algorithms is indicated. Computational results are given in 4. The precise form of the algorithm used for these calculations is stated, including the details of the minimization algorithm, which is loosely based on conjugate gradient concepts.

2 Notation and statement of the training problem

Consider a neural network model with feedback as a dynamical system in which the dynamical variables x_{it} change with time according to a dynamical law given by the mapping

$$\left.\begin{array}{rcll} x_{it} & = & \sum_j w_{ij} f(x_{j,t-1}) & i > 0 \\ x_{0t} & = & \text{bias constant} & \end{array}\right\} \tag{1}$$

unless specified otherwise. The *weights* w_{ij} are arbitrary parameters representing the connection strength from node j to node i. f is an arbitrary differentiable function, usually taken to be the logistic:

$$f(x) = 1/(1 + e^{-x}). \tag{2}$$

Let us call any given variable x_{it} the *"activation"* on node i at time t. It represents the total input into node i at time t. Let the *"output"* of each node be denoted by $y_{it} = f(x_{it})$. Let node 0 be a *"bias node"*, assigned a positive constant activation so that the weights w_{i0} can be interpreted as activation thresholds.

In normal back-propagation, a network architecture is defined which divides the network into input, hidden, and target nodes. The moving targets algorithm makes itself applicable to arbitrary training problems by defining analogous concepts in a manner dependent upon the training data, but independent of the network architecture. Let us call a node-time pair an *"event"*. To define a training problem, the set of all events must be divided into three disjoint sets, the *input* events I, *target* events T, and *hidden* events H. For every input event $(it) \in I$, we require *training data* X_{it} with which to overrule the dynamical law (1) using

$$x_{it} = X_{it} \quad (it) \in I. \tag{3}$$

(The bias events $(0t)$ can be regarded as a special case of input events.) For each target event $(it) \in T$, we require training data X_{it} to specify a desired activation value for event $(0t)$. No notational ambiguity arises from referring to input and target data with the same symbol X because I and T are required to be disjoint sets. The training data says nothing about the hidden events in H.

Considerable flexibility in the definition of the training problem arises from the fact that the time dimension can be used to separate the three classes of events from each other. Suppose, for example, that a sequence of inputs is to be recognized as a particular phoneme by having a particular node "turn on" whenever the sequence goes by. If it

were necessary to specify a target value for this node at every time step, then one would have to make some rather arbitrary decisions about what target value to assign when only some of the data needed to recognize the phoneme has been presented. For instance, Watrous [11] uses various functions to interpolate between 0 and 1 during the course of a sequence. The moving targets method allows the node to be classified as hidden during these ambiguous times, and as a target otherwise.

In general, different sequences will result if the dynamical law (1) is applied to different initial conditions x_{i0}. If the network is always initialized to a particular state, then these initial events are classified as inputs. If no such training data exists, they can be classified as hidden events, in which case the training algorithm will generate values for them which should be used when the network is run. If they are given target values, the network will seek an optimal compromise between the given values and others which might be needed to obtain the desired future behavior.

3 The "moving targets" method

Like back-propagation, the moving targets training method uses (arbitrary) gradient-based minimization techniques to minimize an *"error"* function such as the *"output deficit"*

$$E_{\text{od}} = \tfrac{1}{2} \sum_{(it)\in T} \{y_{it} - Y_{it}\}^2, \tag{4}$$

where $y_{it} = f(x_{it})$ and $Y_{it} = f(X_{it})$. A modification of the output deficit error gave the best results in numerical experiments. However, the most elegant formalism follows from an *"activation deficit"* error function:

$$E_{\text{ad}} = \tfrac{1}{2} \sum_{(it)\in T} \{x_{it} - X_{it}\}^2, \tag{5}$$

so this is what we shall use to present the formalism.

The basic idea is to treat the hidden node activations as variable target activations. Therefore let us denote these variables as X_{it}, just as the (fixed) targets and inputs are denoted. Let us write the computed activation values x_{it} of the hidden and target events in terms of the inputs and (fixed and moving) targets of the previous time step. Then let us extend the sum in (5) to include the hidden events, so the error becomes

$$E = \tfrac{1}{2} \sum_{(it)\in T\cup H} \left\{ \sum_j w_{ij} f(X_{j,t-1}) - X_{it} \right\}^2 . \tag{6}$$

This is a function of the weights w_{ij}, and because there are no x's present, the full dependence on w_{ij} is explicitly displayed. We do not actually have desired values for the X_{it} with $(it) \in H$. But any values for which weights can be found which make (6) vanish would be suitable, because this would imply not only that the desired targets are attained, but also that the dynamical law is followed on both the hidden and target nodes. Therefore let us regard E as a function of both the weights and the "moving targets" $X_{it}, (it) \in H$.

This is the essence of the method. The derivatives with respect to all of the independent variables can be computed and plugged into a standard minimization algorithm.

The reason for preferring the activation deficit form of the error (5) to the output deficit form (4) is that the activation deficit form makes (6) purely quadratic in the weights. Therefore the equations for the minimum,

$$dE/dw_{ij} = \partial E/\partial w_{ij} = 0, \tag{7}$$

form a linear system, the solution of which provides the optimal weights for any given set of moving targets. Therefore these equations might as well be used to define the weights as functions of the moving targets, thereby making the error (6) a *function of the moving targets alone.*

At this point the hidden activations x and the weights w have undergone a kind of role reversal. In the usual formulation of back propagation, the error depends on the activations and the weights, but the weights are the only independent variables because the activations depend on the weights in a complicated way involving recursion. In the moving targets formulation, the error again depends on activations and weights, but the activations are the only independent variables because the weights depend on the activations in a relatively simple way involving a system of linear equations.

The derivation of the derivatives with respect to the moving targets is spelled out in ([14]). The result is:

$$\frac{dE}{dX_{as}} = \sum_i \chi_{i,s+1} e_{i,s+1} w_{ia} f'_{as} - \chi_{as} e_{as}, \tag{8}$$

where

$$\chi_{it} = \begin{cases} 1 & (it) \in T \cup H \\ 0 & (it) \notin T \cup H \end{cases} \tag{9}$$

$$e_{it} = \sum_j w_{ij} f(X_{j,t-1}) - X_{it}, \tag{10}$$

$$f'_{it} = \left.\frac{df(x)}{dx}\right|_{x=X_{it}}, \tag{11}$$

and

$$w_{ij} = \sum_k \left(\sum_t \chi_{it} X_{it} Y_{k,t-1} \right) M^{(i)-1}_{kj}, \tag{12}$$

where $M^{(a)-1}$ is the inverse of $M^{(a)}$, the correllation matrix of the node outputs defined by

$$M^{(a)}_{ij} = \sum_t \chi_{at} Y_{i,t-1} Y_{j,t-1}. \tag{13}$$

In the event that any of the matrices M are singular, a pseudo-inversion method such as singular value decomposition [7] can be used to define a unique solution among the infinite number available.

The only subtlety in the derivation is the realization that because the weights are defined by (7), it is necessary only to calculate the partial derivative of the error with respect to the moving targets:

$$\frac{dE}{dX_{as}} = \sum_{ij} \frac{\partial E}{\partial w_{ij}} \frac{dw_{ij}}{dX_{as}} + \frac{\partial E}{\partial X_{as}} \tag{14}$$

$$= \frac{\partial E}{\partial X_{as}}. \tag{15}$$

Note also that (12) calls for a separate matrix inversion for each node. However if the set of input nodes remains fixed for all time, then all these matrices are equal.

The basic ideas used in the moving targets algorithm can be applied to feedforward networks to provide an alternative method to back-propagation. The hidden node activations for each training example become the moving target variables. The calculation involves a different matrix inversion problem for each layer of weights. The algorithm may be particularly suitable for for networks with many layers. Back-propagation has difficulty with these networks because the back-propagated errors diminish as each layer is traversed. The feed-forward version of moving targets also has the conceptual appeal of directly varying the internal representations. This makes it straightforward to initialize the internal representations using prior knowledge or preprocessing methods such as cluster analysis.

The moving targets method for feedforward nets is analogous to the method of Grossman, Meir, and Domany [2] for networks with discrete node values. Birmiwal, Sarwal, and Sinha [1] have developed an algorithm for feedforward networks which incorporates the use of hidden node values as fundamental variables and a linear system of equations for obtaining the weight matrix. Their algorithm differs from the feedforward version of moving targets mainly in the (innessential) use of a specific minimization algorithm which discards most of the gradient information except for the signs of the various derivatives. Heileman, Georgiopoulos, and Brown [3] also have an algorithm which bears some resemblance to the feedforward version of moving targets.

4 Computational Results

A set of numerical experiments performed with the activation deficit form of the algorithm 5 is reported in [14]. Some success was attained, but greater progress was made after changing to a quartic output deficit error function with temporal weighting of errors:

$$E_{\text{quad}} = \tfrac{1}{4} \sum_{(it)\in T} (1.0 + at)\{y_{it} - Y_{it}\}^4. \tag{16}$$

Here a is a small positive constant. The quartic function is dominated by the terms with the greatest error. This combats a tendency to fail on a few infrequently seen state

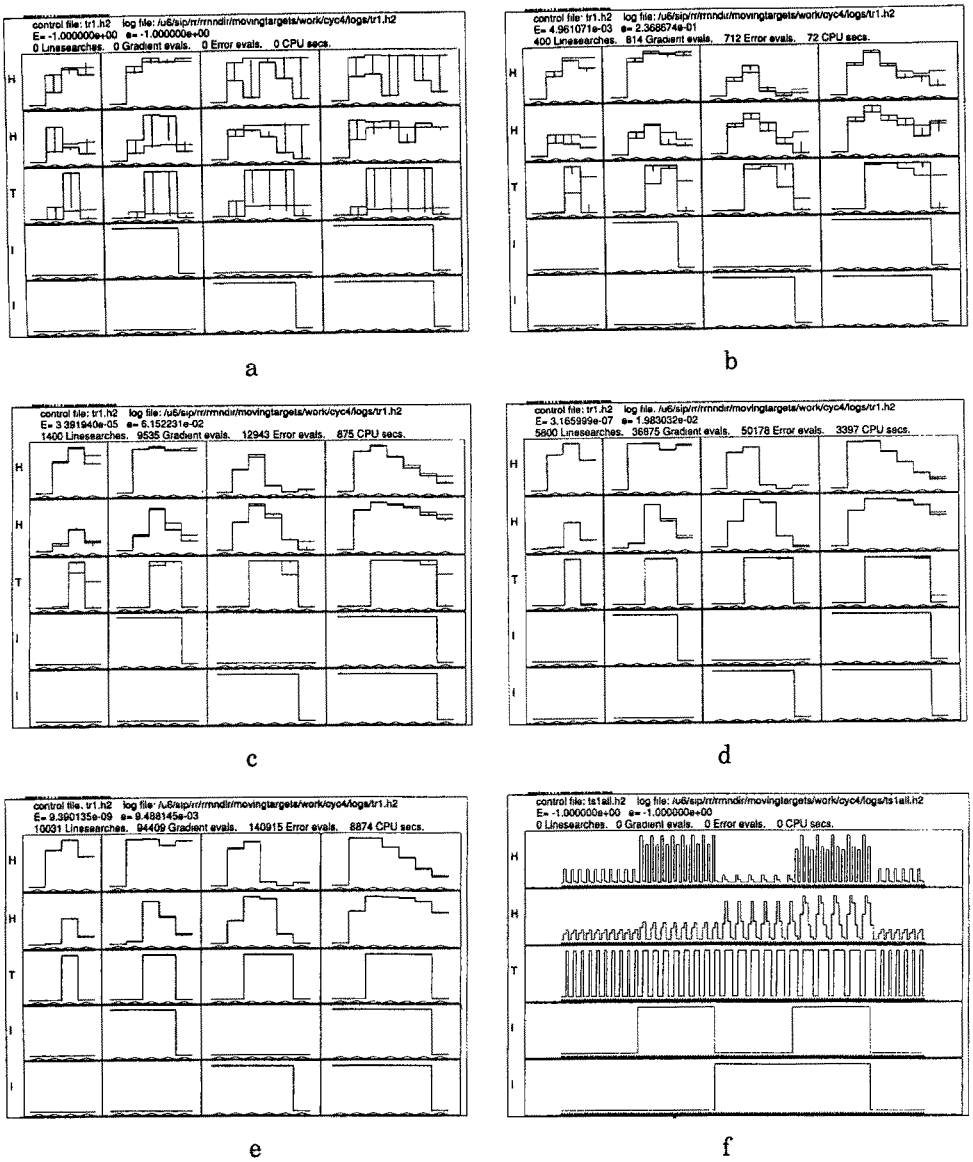
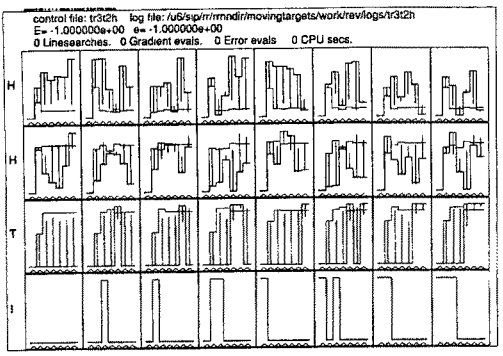
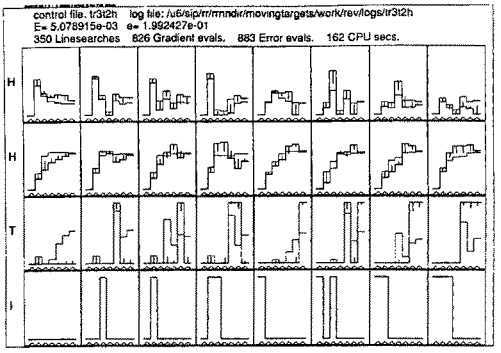


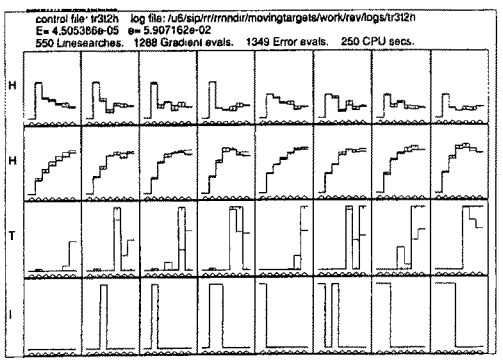
Figure 1. Switching between 4 limit cycles under input control. Time proceeds left to right. Node type (Input, Target, Hidden) indicated in left margin. Heavy connected line shows Y_{it}; input values for input nodes, target values for target nodes, and moving target values for hidden nodes. On target and hidden nodes, heavy lines at midpoints of timesteps show discrepencies between (fixed or moving) target values, and values computed by the network from the desired state on the previous timestep. These discrepencies contribute to the error function. Trajectory of network state given the initial conditions only is shown by the light connected line. Training progresses from (a) to (e). Behaviour of trained network over extended time period shown in (f). Inputs regulate pulse width.



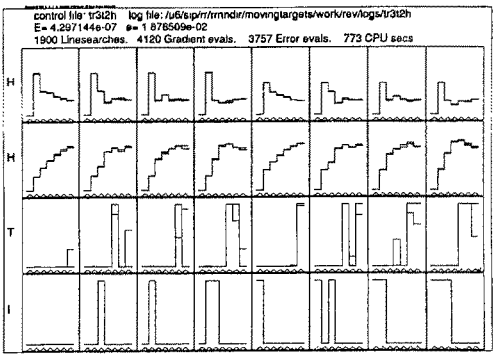
a



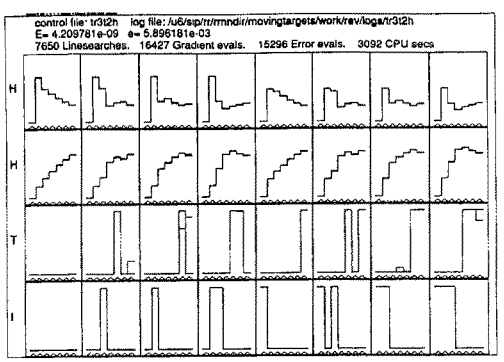
b



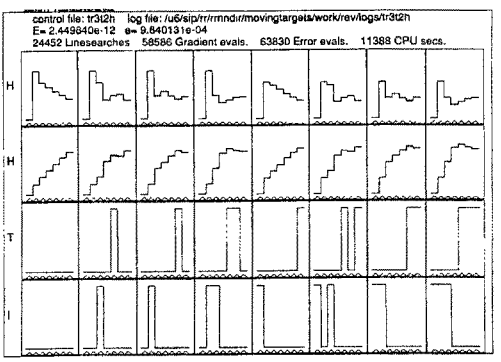
c



d



e



f

Figure 2. Sequence reversal. Interpretation of plots as in figure 1. Network is trained to temporally reverse each of the 8 possible length-3 sequences of bits. Training proceeds from (a) to (f).

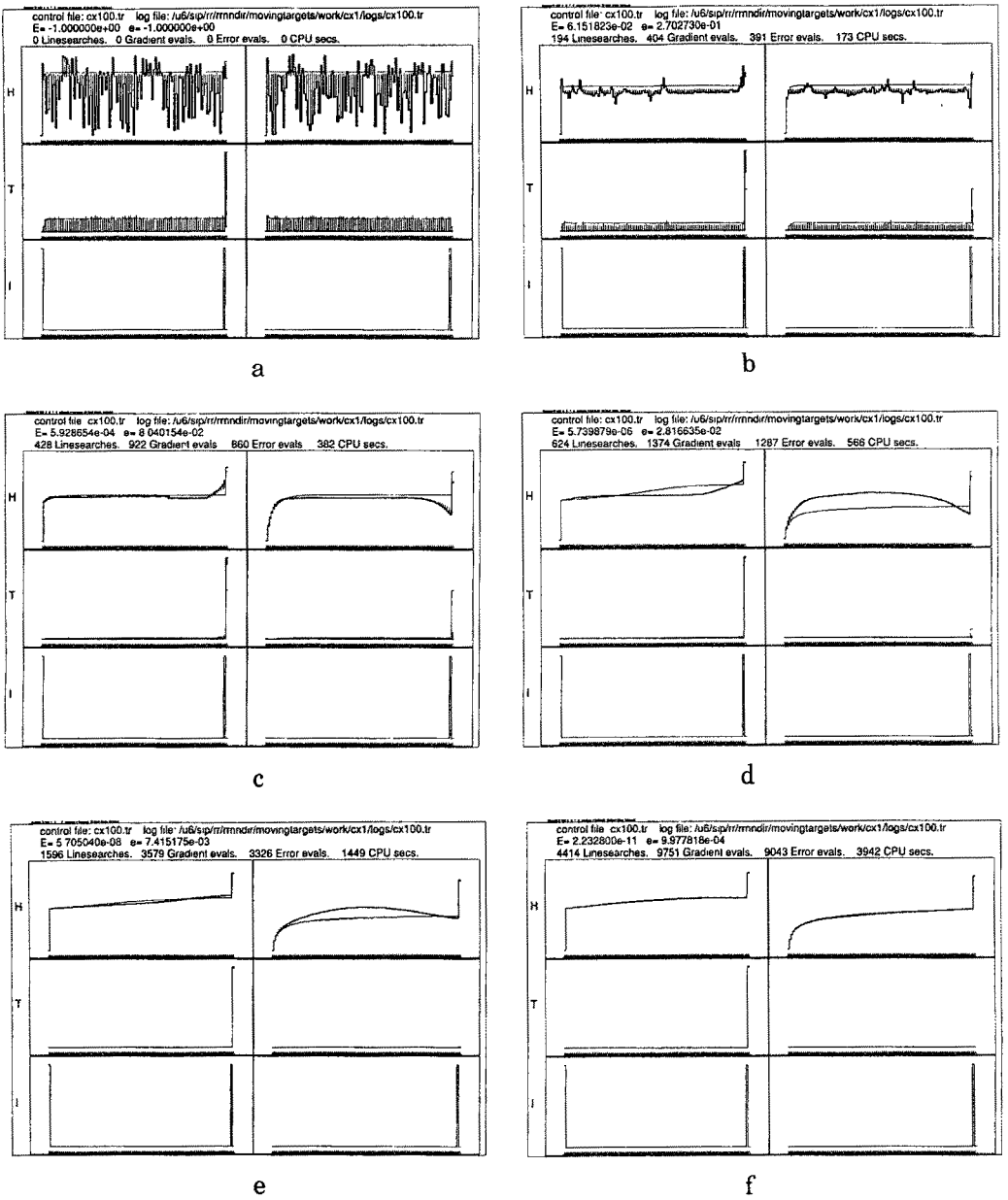


Figure3. Response to temporally distant stimulus. Interpretation of plots as in figure 1. Network learns to fire target node on second input pulse, even though first pulse is 100 time steps in the past. Training proceeds from (a) to (f).

transitions in order to gain unneeded accuracy on a large number of similar, low-error state transitions. Some kind of weighting on the training data might be a better solution. The temporal weighting encourages the algorithm to focus first on late-time errors, and then work back in time. This appears to help with occasional local minimum difficulities.

Further progress was made by altering the minimization algorithm. Originally the conjugate gradient algorithm [7] was used, with a linesearch algorithm from Fletcher [6]. The new algorithm might be called "curvature avoidance". The change in the gradient with each linesearch is used to update a moving average estimate of the absolute value of the diagonal components of the Hessian. The linesearch direction is taken to be the component-by-component quotient of the gradient with these curvature averages. Were it not for the absolute values, this would be an unusual way of estimating the conjugate gradient. The absolute values are used to discourage exploration of directions which show any hint of being highly curved. The philosophy is that by exploring low-curvature directions first, narrow canyons are entered only when necesary.

The performance of the new minimization algorithm on the activation deficit error function has not been tested.

The progress of training on three simple problems is illustrated in the figures. The problem in Figure 1 is to learn to 4 different limit cycles and switch between them in response to the binary number represented on the 2 input nodes. Specifically, the input controls the pulse width of a pulse generator. In Figure 2, a network learns to temporally reverse a 3-bit sequence presented on the input node. At time step 4, the 2 hidden nodes carry an encoding of which of the 8 possible sequences was present. The problem in figure 3 is to respond to the second of 2 input pulses separated by 100 time steps. This demonstrates that the network can learn to notice very distant temporal correllations in circumstances where this is necessary.

5 Acknowledgements

This work was supported by the UK Information Engineering Directorate/ Science and Engineering Research Council as part of the IED/SERC Large Scale Integrated Speech Technology Demonstrator Project (SERC grants D/29604, D/29611, D/29628, F/10309, F/10316), in which Marconi Speech and Information Systems are the industrial partner. This work was also supported by ESPRIT Basic Research Action 3207 ACTS.

References

[1] K. Birmiwal, P. Sarwal, and S. Sinha, "A new Gradient-Free Learning Algorithm", Tech. report, Dept. of EE, Southern Illinois U., Carbondale, (1989).

[2] T. Grossman, R. Meir, and E. Domany, "Learning by Choice of Internal Representations", Dept. of Electronics, Weizmann Institute of Science, Rehovot, Israel, (1988).

[3] G. L. Heileman, M. Georgiopoulos, and A. K. Brown, "The Minimal Disturbance Back Propagation Algorithm", Tech. report, Dept. of EE, U. of Central Florida, Orlando, (1989).

[4] G. Kuhn, "Connected Recognition with a Recurrent Network", to appear in proc. NEUROSPEECH, 18 May 1989, as special issue of Speech Communication, v. 9, no. 2, (1990).

[5] Pearlmutter, B., "Learning State Space Trajectories in Recurrent Neural Networks", Proc. IEEE IJCNN 89, Washington D. C., Publ. IEEE TAB Neural Network Committee., p. II-365, (1989).

[6] R. Fletcher, Practical Methods of Optimization, v1, Wiley, (1980).

[7] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes, The Art of Scientific Computing*, Cambridge, (1986).

[8] R. Rohwer and B. Forrest, "Training Time Dependence in Neural Networks" Proc. IEEE ICNN, San Diego, p. II-701, (1987).

[9] R. Rohwer and S. Renals, "Training Recurrent Networks", in *Neural Networks from Models to Applications*, L. Personnaz and G. Dreyfus, eds., I.D.S.E.T., Paris, p. 207, (1989).

[10] Rumelhart, D., Hinton, G. and Williams, R., "Learning Internal Representations by Error Propagation" in *Parallel Distributed Processing*, v. 1, MIT, (1986).

[11] R. L. Watrous, "Phoneme Discrimination using Connectionist Networks", Tech. Report, Dept. of Computer Science, U. Penn., submitted to J. Acoustical Soc. of America, (1989).

[12] P. Werbos, *Energy Models and Studies*, B. Lev, Ed., North Holland, (1983).

[13] R. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", ICS Report 8805, UC San Diego, (1988).

[14] R. Rohwer, "The 'Moving Targets' Training Algorithm", to appear in Proc. DANIP, GMD Bonn, J. Kinderman and A. Linden, Eds., 24-25 April, 1989.