UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

AD-A217 733

1. REPORT NUMBER	2. GOVT A
ARO 25515.1-LS	N/A

4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED
Gain Modification Enhances High Momentum Backward Propagation	Technical
	6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)
Charles M. Bachmann	DAAL03-88-K-0116

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Department of Physics and Center for Neural Science, Brown University, Prov., R. I., 02912	N/A

11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE
U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709	November 30, 1989
	13. NUMBER OF PAGES 52 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Backward Propagation,
Gain Modification
Momentum, (jhd)
Effective Time-Dependent Step-Constant

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We present a backward propagation network which simultaneously modifies the gain parameters and the synaptic weights. Gain modification is shown to enhance the improvement in convergence rate obtained by high momentum in standard synaptic backward propagation. These improvements occur without degrading the generalization capabilities of the final solutions obtained by the network.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Gain Modification Enhances High Momentum Backward Propagation

Charles M. Bachmann[1]
Physics Department
and Center for Neural Science
Brown University
Providence, R. I. 02912

November 8, 1989

[1]This work was supported in part by the National Science Foundation, the Army Research Office, and the Office of Naval Research.

Contents

Chapter 1

Theoretical Development of Gain Modification

1.1 Introduction

We present a backward propagation network which simultaneously modifies the gain parameters and the synaptic weights. The additional complexity is minimized by the fact that the error signal for modification of the gain of a neuron is proportional to the ordinary error signal for the incoming synaptic weights to the neuron. For a given neuron, the proportionality factor is the reciprocal of its gain. Thus, only the ordinary error signal must be propagated. The input to a synapse, which multiplies the error signal in the standard modification rule for synapses, is replaced by the net input to the cell in the gain modification rule. We also demonstrate that our algorithm can be viewed as a gradient descent in rescaled synaptic vectors with effective time-dependent step-constants which depend on both the magnitude of the gains and the magnitude of the ordinary synaptic vectors. In this paper, we show that the effect of gain modification by this algorithm can be used to enhance the improvements in convergence rate obtained by the use of high momentum in ordinary synaptic modification. These improvements occur without degrading the generalization capabilities of the final solutions obtained by the network.

1.2 Review of Notation for Standard Backward Propagation

We begin with a short summary of the notation which we have used for backward propagation in our previous work (Bachmann, 1988) [1]. The notation differs somewhat from that used by Rumelhart, Hinton, and Williams (1986) [6] and Werbos (1988) [7]. For simplicity, we consider a three-layer network. A typical network is illustrated in figure 1.1.

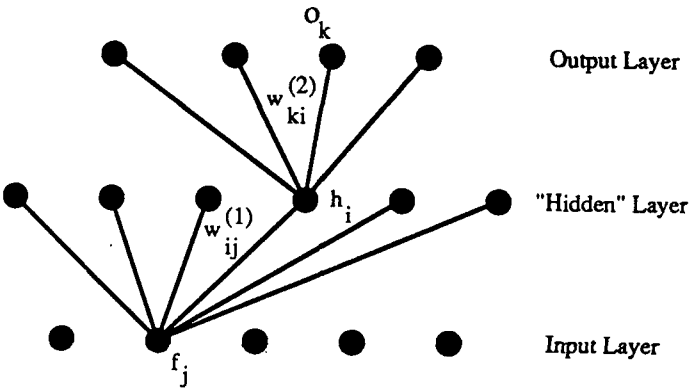

Figure 1.1: The o_k label the output units, the h_i label the "hidden units", and the f_j label the input units. Only some of the connections are shown. Superscripts on synaptic weights denote layer index.

The feedforward equations are defined by:

$$x_i^s = \sum_{j=1}^{N_1} w_{ij}^{(1)} f_j^s + \phi_i^{(1)} \tag{1.1}$$

$$h_i^s = \psi(x_i^s; \lambda_i^{(1)}) \tag{1.2}$$

$$y_k^s = \sum_{i=1}^{N_2} w_{ki}^{(2)} h_i^s + \phi_k^{(2)} \tag{1.3}$$

$$o_k^s = \psi(y_k^s; \lambda_k^{(2)}), \tag{1.4}$$

o_k^s is the output of the kth unit in the output layer, h_i^s is the output of the ith unit in the hidden layer, and f_j^s is the value of the jth input. s denotes

the pattern index. Additionally, recall that y_k^s is the total input to the kth output unit, and x_i^s is the total input to the ith hidden unit. $\phi_i^{(1)}$ is the bias for the ith cell in the hidden layer, and $\phi_k^{(2)}$ is the bias for the kth cell in the output layer. $\psi(z;\lambda)$ is the sigmoid input-output function defined by:

$$\psi(z;\lambda) = \frac{1}{1+e^{-\lambda z}} \tag{1.5}$$

We have introduced explicitly the gains $\lambda_k^{(2)}$ of the kth cell in the last layer and $\lambda_i^{(1)}$ for the ith cell in the hidden layer. [1] [2] ξ_s, the energy of pattern s, is given by:

$$\xi_s = \sum_{k=1}^{N_2} \frac{1}{2}(o_k^s - \tau_k^s)^2. \tag{1.6}$$

A pattern gradient, which is used to modify the synaptic weights following the presentation of each pattern to the network, is computed from the energy ξ_s. The backward propagation method is thus only an approximation to gradient descent since the true Liapunov function is really:

$$\begin{aligned} \xi &= \sum_{s=1}^{m} \xi_s \\ &= \frac{1}{2}\sum_{s=1}^{m}\sum_{k=1}^{N_2}(o_k^s - \tau_k^s)^2 \end{aligned} \tag{1.7}$$

To make a connection with Rumelhart's delta error signal notation (Rumelhart, 1988) [6], we note that by defining partial derivatives with respect to the "net", or net input to a cell, we have[3]:

$$\begin{aligned} \delta_{sk}^{(2)} &= -\frac{\partial \xi_s}{\partial y_k^s} \\ &= -\frac{\partial \xi_s}{\partial o_k^s}\frac{\partial o_k^s}{\partial y_k^s} \\ &= -\lambda_k^{(2)}(o_k^s - \tau_k^s)o_k^s(1-o_k^s) \end{aligned} \tag{1.8}$$

[1] In Rumelhart's original model there is no gain parameter λ ($\lambda = 1$). Although initially all gains are set to 1, our proposed model allows the gains to vary individually.

[2] Hopfield (1984) [3] has used a gain parameter in the continuous version of his model to study the effect of changing the character of the nonlinearity of the input-output function.

[3] We have added superscripts to the error signals for greater clarity.

$$\begin{aligned}
\delta_{si}^{(1)} &= -\frac{\partial \xi_s}{\partial x_i^s} \\
&= -\sum_{k=1}^{N_2} \frac{\partial \xi_s}{\partial o_k^s}\frac{\partial o_k^s}{\partial y_k^s}\frac{\partial y_k^s}{\partial h_i^s}\frac{\partial h_i^s}{\partial x_i^s} \\
&= -\lambda_i^{(1)} h_i^s(1-h_i^s)\sum_{k=1}^{N_2}(o_k^s - \tau_k^s)\lambda_k^{(2)} o_k^s(1-o_k^s)w_{ki}^{(2)}. \qquad (1.9)
\end{aligned}$$

This allows one to write:

$$\delta_{si}^{(1)} = \lambda_i^{(1)} h_i^s(1-h_i^s)\sum_{k=1}^{N_2}\delta_{sk}^{(2)} w_{ki}^{(2)} \qquad (1.10)$$

The synaptic modifications are proportional to the negative gradient:

$$\begin{aligned}
\Delta_s(w_{ki}^{(2)}) &= -\eta\frac{\partial \xi_s}{\partial w_{ki}^{(2)}} \\
&= -\eta\frac{\partial \xi_s}{\partial y_k^s}\frac{\partial y_k^s}{\partial w_{ki}^{(2)}} \qquad (1.11) \\
\Delta_s(w_{ij}^{(1)}) &= -\eta\frac{\partial \xi_s}{\partial w_{ij}^{(1)}} \\
&= -\eta\frac{\partial \xi_s}{\partial x_i^s}\frac{\partial x_i^s}{\partial w_{ij}^{(1)}} \qquad (1.12)
\end{aligned}$$

Therefore, with the above definitions for $\delta_{sk}^{(2)}$ and $\delta_{si}^{(1)}$, equations 1.11 and 1.12 become: [4]

$$\begin{aligned}
\Delta_s(w_{ki}^{(2)}) &= \eta\delta_{sk}^{(2)} h_i^s \qquad (1.13) \\
\Delta_s(w_{ij}^{(1)}) &= \eta\delta_{si}^{(1)} f_j^s. \qquad (1.14)
\end{aligned}$$

Note that equation 1.10 defines the backward propagation of the error signals. Recall also that in standard backward propagation, a "momentum" term is often used at each step of the modification procedure in the gradient descent search for the minimum. Heuristically, the momentum term can

[4] As in Rumelhart et. al. (1986) [6], the biases are modified by the same procedure; however, the input for biases is defined to be unity.

be viewed as a means of increasing the step-constant when the curvature of the energy surface is low and several successive modifications have the same sign (Jacobs, 1988) [5]. The momentum term consists of adding a small amount proportional to the previous modification, so that the actual modification at time step t for the nth layer of synaptic connections is:

$$\Delta_{s(t)}(w_{ki}^{(n)}) = -\eta \frac{\partial \xi_{s(t)}}{\partial w_{ki}^{(n)}} + \kappa \Delta_{s(t-1)}(w_{ki}^{(n)}) \tag{1.15}$$

where s(t) denotes the index of the pattern presented at time step t and κ is a positive constant less than 1. One can also write this in a slightly different form:

$$\Delta_{s(t)}(w_{ki}^{(n)}) = -\eta \sum_{\nu=0}^{t} \frac{\partial \xi_{s(\nu)}}{\partial w_{ki}^{(n)}} \kappa^{t-\nu}. \tag{1.16}$$

This form is more suggestive in showing that the use of a momentum term is equivalent to a discrete approximation to a temporal integral average with an exponentially decaying kernel. The kernel has the effect of emphasizing the influence of the patterns most recently presented, assigning exponentially less weight to those patterns presented earlier in time.

1.3 Simultaneous Modification of Gain Parameters and Synapses by a Backward Propagation Algorithm

In this section, we consider the possibility of modifying the gain parameters and the synaptic weights simultaneously. To accomplish this, we have formulated a backward propagation procedure which modifies the gain parameters in the network in a manner similar to the method used for the synaptic weights. The procedure can take advantage of quantities already calculated in the ordinary backward propagation procedure for the synaptic weights, thus minimizing the additional complexity. The error signal for the gain of a particular neuron is proportional to the ordinary synaptic error signal for the incoming synaptic weights connected to the neuron. The proportionality factor is just the reciprocal of the neuronal gain. Additionally, the input to a particular synapse, which multiplies the error signal in

the standard synaptic modification formula, is replaced by the net input to the neuron for the modification of its gain.

We note in passing that other work has been done on gain modification procedures, for example Kruschke (1988) [4]; however, Kruschke's procedure does not use the gradient with respect to the gain. Rather, gains are modified in the context of a competitive learning scheme, which is combined with backward propagation modification of the synaptic weights. In contrast, our scheme uses a backward propagation procedure for both gains and weights, incorporating respectively the gradients with respect to the gains and the weights.

To derive our model, we begin by defining rescaled error signals $\gamma_{sk}^{(2)}$ and $\gamma_{si}^{(1)}$ in terms of the error signals for the synaptic weights in equations 1.8 and 1.9:

$$\begin{aligned}
\gamma_{sk}^{(2)} &\equiv \frac{1}{\lambda_k^{(2)}}\delta_{sk}^{(2)} \\
&= -\frac{1}{\lambda_k^{(2)}}\frac{\partial \xi_s}{\partial y_k^s} \\
&= -\frac{1}{\lambda_k^{(2)}}\frac{\partial \xi_s}{\partial o_k^s}\frac{\partial o_k^s}{\partial y_k^s} \\
&= -(o_k^s - \tau_k^s)o_k^s(1 - o_k^s) \qquad (1.17)
\end{aligned}$$

$$\begin{aligned}
\gamma_{si}^{(1)} &\equiv \frac{1}{\lambda_i^{(1)}}\delta_{si}^{(1)} \\
&= -\frac{1}{\lambda_i^{(1)}}\frac{\partial \xi_s}{\partial x_i^s} \\
&= -\frac{1}{\lambda_i^{(1)}}\sum_{k=1}^{N_2}\frac{\partial \xi_s}{\partial o_k^s}\frac{\partial o_k^s}{\partial y_k^s}\frac{\partial y_k^s}{\partial h_i^s}\frac{\partial h_i^s}{\partial x_i^s} \\
&= -h_i^s(1 - h_i^s)\sum_{k=1}^{N_2}(o_k^s - \tau_k^s)\lambda_k^{(2)}o_k^s(1 - o_k^s)w_{ki}^{(2)}. \qquad (1.18)
\end{aligned}$$

Given these definitions, we may write a backward propagation equation for the rescaled error signals $\gamma_k^{(2)}$ and $\gamma_i^{(1)}$ by combining equations 1.17 and 1.18. Alternatively, we could have derived this formula by simply replacing $\delta_{sk}^{(2)}$

by $\lambda_k^{(2)}\gamma_{sk}^{(2)}$ and $\delta_{si}^{(1)}$ by $\lambda_i^{(1)}\gamma_{si}^{(1)}$ in equation 1.10:

$$\gamma_{si}^{(1)} = h_i^s(1-h_i^s)\sum_{k=1}^{N_2}\lambda_k^{(2)}\gamma_{sk}^{(2)}w_{ki}^{(2)} \tag{1.19}$$

If we observe that:

$$\frac{\partial o_k^s}{\partial y_k^s} = \lambda_k^{(2)}o_k^s(1-o_k^s) \tag{1.20}$$

then we have:

$$\begin{aligned}\frac{\partial o_k^s}{\partial \lambda_k^{(2)}} &= y_k^s o_k^s(1-o_k^s)\\ &= y_k^s\frac{1}{\lambda_k^{(2)}}\frac{\partial o_k^s}{\partial y_k^s}\end{aligned} \tag{1.21}$$

The same line of reasoning also allows us to write:

$$\frac{\partial h_i^s}{\partial \lambda_i^{(1)}} = x_i^s\frac{1}{\lambda_i^{(1)}}\frac{\partial h_i^s}{\partial x_i^s} \tag{1.22}$$

Therefore, we may write:

$$\begin{aligned}\Delta_s\lambda_k^{(2)} &= -\alpha\frac{\partial \xi_s}{\partial \lambda_k^{(2)}}\\ &= -\alpha\frac{\partial o_k^s}{\partial \lambda_k^{(2)}}\frac{\partial \xi_s}{\partial o_k^s}\\ &= -\alpha y_k^s\frac{1}{\lambda_k^{(2)}}\frac{\partial o_k^s}{\partial y_k^s}\frac{\partial \xi_s}{\partial o_k^s}\\ &= -\alpha y_k^s\frac{1}{\lambda_k^{(2)}}\frac{\partial \xi_s}{\partial y_k^s}\end{aligned} \tag{1.23}$$

$$\begin{aligned}\Delta_s\lambda_i^{(1)} &= -\alpha\frac{\partial \xi_s}{\partial \lambda_i^{(1)}}\\ &= -\alpha\frac{\partial h_i^s}{\partial \lambda_i^{(1)}}\frac{\partial \xi_s}{\partial h_i^s}\\ &= -\alpha x_i^s\frac{1}{\lambda_i^{(1)}}\frac{\partial h_i^s}{\partial x_i^s}\frac{\partial \xi_s}{\partial h_i^s}\\ &= -\alpha x_i^s\frac{1}{\lambda_i^{(1)}}\frac{\partial \xi_s}{\partial x_i^s}\end{aligned} \tag{1.24}$$

where we have used equations 1.21 and 1.22 to write the derivatives with respect to the gains in equations 1.23 and 1.24 in terms of derivatives with respect to the net inputs y_k^s and x_i^s to the kth cell in the output layer and the ith cell in the hidden layer respectively. Using the definitions in equations 1.17 and 1.18, we can recast equations 1.23 and 1.24 as:

$$\Delta_s \lambda_k^{(2)} = \alpha \gamma_{sk}^{(2)} y_k^s \tag{1.25}$$

$$\Delta_s \lambda_i^{(1)} = \alpha \gamma_{si}^{(1)} x_i^s \tag{1.26}$$

For comparison with equations 1.13 and 1.14 describing synaptic modification, we note that in equations 1.25 and 1.26 depicting gain modification, the ordinary error signals defined in equations 1.8 and 1.9 have been replaced by the rescaled error signals defined in equations 1.17 and 1.18, and the signal input to a particular incoming synapse (either h_i^s or f_j^s depending on the layer) has been replaced by the corresponding total integrated potential to the cell, i.e. y_k^s or x_i^s depending on the layer. Since the rescaled error signals, γ, are proportional to the ordinary synaptic error signals, δ, we need only propagate the ordinary error signals δ and obtain the rescaled error signals locally by dividing by the cell gain. Equivalently, we can view the error signal as a quantity which initiates changes in the input-output characteristics of the neuron at the same time that it modifies the incoming synapses to the neuron. From this perspective, the modification rule for a synapse couples the ordinary error signal and the input to the incoming synapse with coupling constant η, while the gain modification rule couples the ordinary error signal and the net input to a cell with coupling constant $\frac{\alpha}{\lambda_i^{(n)}}$. The only difference is the strength of the coupling, and, in fact, since the gain parameter is a more sensitive parameter than the synaptic weights, we choose the initial coupling for the gains, α, to be an order of magnitude smaller than that of the synapses in the simulations described here. [5] Alternatively, we will show in the next section that one can view simultaneous gain and synaptic modification as a gradient descent in rescaled synapses with effective step-constants which are dependent on the gain and magnitude of the ordinary synaptic vectors and are direction-dependent in the

[5] The gains are all set to one initially. Therefore, the order of magnitude of the initial coupling is determined by α. Even though the gains are chosen to be initially the same, symmetry is still broken by choosing the initial synaptic weights randomly in a small hypercube.

synaptic vector space. From this perspective, the effective step-constant is now also time-dependent through its dependence on the gain and the magnitude of the ordinary synaptic vectors.

1.4 Gain Modification Viewed as an Effective Time-Dependent Step-Constant

In developing our theoretical arguments, it will be easiest to consider the effects of gain modification on ordinary backward propagation without momentum. It will be seen that the addition of momentum will not alter the development here. To simplify matters, we adopt a notation which includes synaptic weight vectors and biases in the same vector. We also change the notation to a slightly more general form than that used in sections 1.2 and 1.3. For the ith cell in the nth layer, [6] we define:

$$\begin{aligned} \vec{W}_i^{(n)} &= (\vec{w}_i^{(n)}, \phi_i^{(n)}) \\ \vec{F}^{(n),s} &= (\vec{f}^{(n),s}, 1) \end{aligned} \tag{1.27}$$

where the input to layer n, $\vec{f}^{(n),s} = \vec{o}^{(n-1),s}$, is just the output vector of the previous layer. [7] The jth component of $\vec{w}_i^{(n)}$, $w_{ij}^{(n)}$, then is just the connection from the jth cell in the (n-1)th layer to the ith cell in the nth layer. This allows us to write the output for pattern s of the ith cell in the nth layer as:

$$o_i^{(n),s} = \frac{1}{1 + e^{-\lambda_i^{(n)} \vec{W}_i^{(n)} \cdot \vec{F}^{(n),s}}}. \tag{1.28}$$

In the case of nonmodifiable gains set equal to one, it is apparent that we may view the norm of the vector $\vec{W}_i^{(n)}$ as the gain. This follows by rewriting equation 1.28 as:

$$o_i^{(n),s} = \frac{1}{1 + e^{-\lambda_i^{(n)} |\vec{W}_i^{(n)}| \hat{W}_i^{(n)} \cdot \vec{F}^{(n),s}}}. \tag{1.29}$$

[6] The input layer is defined to have the index n = 0.

[7] To compare the notation for neuron activation used earlier for a three layer system, $\vec{f}^s$, $\vec{h}^s$, $\vec{o}^s$, with the current notation, we see that the inputs to the network would be $\vec{f}^{(1),s} = \vec{o}^{(0),s} = \vec{f}^s$, hidden unit outputs would be $\vec{f}^{(2),s} = \vec{o}^{(1),s} = \vec{h}^s$, and the last layer output would be $\vec{f}^{(3),s} = \vec{o}^{(2),s} = \vec{o}^s$.

Here $\hat{W}_i^{(n)}$ is a unit vector in the direction of $\vec{W}_i^{(n)}$. This leads us to ask if modifying the gains, $\lambda_i^{(n)}$, is truly advantageous. To answer this question, we will need to define rescaled synaptic vectors according to:

$$\vec{u}_i^{(n)} \equiv \lambda_i^{(n)} \vec{W}_i^{(n)} \tag{1.30}$$

Note that with this definition equation 1.28 becomes:

$$o_i^{(n),s} = \frac{1}{1 + e^{-\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s}}}. \tag{1.31}$$

Since $\vec{u}_i^{(n)}$ is just a rescaling of the synaptic vector $\vec{W}_i^{(n)}$, we can ask how this vector changes when we modify $\vec{W}_i^{(n)}$ and $\lambda_i^{(n)}$ separately. In particular, we will find it useful to consider the changes in $\vec{u}_i^{(n)}$ along the direction of $\vec{u}_i^{(n)}$ and those in the hyperplane perpendicular to $\vec{u}_i^{(n)}$. We denote these two components by $(\Delta_s \vec{u}_i^{(n)})_{\parallel}$ and $(\Delta_s \vec{u}_i^{(n)})_{\perp}$. We also define $\hat{v}_i^{(n)}$ to be a unit vector in the direction of $(\Delta \vec{u}_i^{(n)})_{\perp}$ in the hyperplane orthogonal to $\hat{u}_i^{(n)}$. If the changes in $\lambda_i^{(n)}$ and $\vec{W}_i^{(n)}$, and thus in $\vec{u}_i^{(n)}$, are sufficiently small, then we may write:

$$\Delta_s \vec{u}_i^{(n)} = \Delta_s (\lambda_i^{(n)} \vec{W}_i^{(n)}) \approx \lambda_i^{(n)} \Delta_s \vec{W}_i^{(n)} + \vec{W}_i^{(n)} \Delta_s \lambda_i^{(n)}. \tag{1.32}$$

The change $\Delta_s \vec{W}_i^{(n)}$ can be rectified into two orthogonal components: [8]

$$\begin{aligned} \Delta_s \vec{W}_i^{(n)} &= -\eta \nabla_{\vec{W}_i^{(n)}} \xi^s \\ &= -\eta (\hat{W}_i^{(n)} (\hat{W}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s) + \hat{v}_i^{(n)} (\hat{v}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s)) \end{aligned} \tag{1.33}$$

Note that $\Delta_s \lambda_i^{(n)}$ is just:

$$\Delta_s \lambda_i^{(n)} = -\alpha \frac{\partial \xi^s}{\partial \lambda_i^{(n)}} \tag{1.34}$$

However, we can write $\frac{\partial \xi^s}{\partial \lambda_i^{(n)}}$ in terms of the component of $\nabla_{\vec{W}_i^{(n)}} \xi^s$ along $\hat{W}_i^{(n)}$. To see this, observe that we may write:

$$\begin{aligned} \frac{\partial \xi^s}{\partial \lambda_i^{(n)}} &= \frac{\partial \xi^s}{\partial (\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s})} \frac{\partial (\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s})}{\partial \lambda_i^{(n)}} \\ &= \frac{\partial \xi^s}{\partial (\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s})} \mid \vec{W}_i^{(n)} \mid \hat{W}_i^{(n)} \cdot \vec{F}^{(n),s} \end{aligned} \tag{1.35}$$

[8] Here we neglect momentum.

At the same time, note that:

$$\begin{aligned}
\hat{W}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s &= \frac{\partial \xi^s}{\partial \mid \vec{W}_i^{(n)} \mid} \\
&= \frac{\partial \xi^s}{\partial (\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s})} \frac{\partial (\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s})}{\partial \mid \vec{W}_i^{(n)} \mid} \\
&= \frac{\partial \xi^s}{\partial (\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s})} \lambda_i^{(n)} \hat{W}_i^{(n)} \cdot \vec{F}^{(n),s} \qquad (1.36)
\end{aligned}$$

From equations 1.35 and 1.36, we obtain:

$$\frac{\partial \xi^s}{\partial \lambda_i^{(n)}} = \frac{\mid \vec{W}_i^{(n)} \mid}{\lambda_i^{(n)}} \hat{W}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s \qquad (1.37)$$

Having derived equation 1.37, we can now combine it with equations 1.33 and 1.34 to rewrite equation 1.32 as:

$$\begin{aligned}
\Delta_s \vec{u}_i^{(n)} \approx -\eta \lambda_i^{(n)} \{ \; & (1 + \frac{\alpha}{\eta} (\frac{\mid \vec{W}_i^{(n)} \mid}{\lambda_i^{(n)}})^2) (\hat{u}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s) \hat{u}_i^{(n)} \\
& + (\hat{v}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s) \hat{v}_i^{(n)} \; \}, \qquad (1.38)
\end{aligned}$$

where we have used the fact that $\hat{u}_i^{(n)} = \hat{W}_i^{(n)}$. This allows us to write:

$$\begin{pmatrix} (\Delta \vec{u}_i^{(n)})_{\parallel} \\ (\Delta \vec{u}_i^{(n)})_{\perp} \end{pmatrix} \approx -\eta \lambda_i^{(n)} \begin{pmatrix} 1 + \frac{\alpha}{\eta} (\frac{\mid \vec{W}_i^{(n)} \mid}{\lambda_i^{(n)}})^2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{u}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s \\ \hat{v}_i^{(n)} \cdot \nabla_{\vec{W}_i^{(n)}} \xi^s \end{pmatrix} \qquad (1.39)$$

We can recast this equation in terms of gradients with respect to $\vec{u}_i^{(n)}$ by converting the gradients with respect to $\vec{W}_i^{(n)}$. In doing so, we note one subtlety about the gradients which were used to derive equation 1.39. The gradient, $\nabla_{\vec{W}_i^{(n)}} \xi^s$, which appears in ordinary backward propagation (equation 1.33, and therefore in equation 1.39, is a vector of partial derivatives evaluated at constant $\lambda_i^{(n)}$. We could denote this explicitly by writing $(\nabla_{\vec{W}_i^{(n)}} \xi^s)_{\lambda_i^{(n)}}$. Similarly, in the modification of the gains (equation 1.34), $\frac{\partial \xi_s}{\partial \lambda_i^{(n)}}$ is calculated as a partial derivative to be taken at constant $\vec{W}_i^{(n)}$ which we could write as $(\frac{\partial \xi_s}{\partial \lambda_i^{(n)}})_{\vec{W}_i^{(n)}}$. In fact, equation 1.37 is an equation relating

$(\frac{\partial \xi_s}{\partial \lambda_i^{(n)}})_{\vec{W}_i^{(n)}}$ to $(\nabla_{\vec{W}_i^{(n)}} \xi^s)_{\lambda_i^{(n)}}$. Because the gradient of the energy with respect to $\vec{W}_i^{(n)}$, $(\nabla_{\vec{W}_i^{(n)}} \xi^s)_{\lambda_i^{(n)}}$, is calculated for fixed $\lambda_i^{(n)}$, we may easily express it in terms of the gradient with respect to $\vec{u}_i^{(n)}$, $(\nabla_{\vec{u}_i^{(n)}} \xi^s)_{\lambda_i^{(n)}}$. We see that the jth component of the two gradients are related by:

$$\begin{aligned} (\frac{\partial \xi_s}{\partial W_{ij}^{(n)}})_{\lambda_i^{(n)}} &= (\frac{\partial u_{ij}^{(n)}}{\partial W_{ij}^{(n)}})_{\lambda_i^{(n)}} (\frac{\partial \xi_s}{\partial u_{ij}^{(n)}})_{\lambda_i^{(n)}} \\ &= \lambda_i^{(n)} (\frac{\partial \xi_s}{\partial u_{ij}^{(n)}})_{\lambda_i^{(n)}} \end{aligned} \tag{1.40}$$

We write this compactly as:

$$(\nabla_{\vec{W}_i^{(n)}} \xi^s)_{\lambda_i^{(n)}} = \lambda_i^{(n)} (\nabla_{\vec{u}_i^{(n)}} \xi^s)_{\lambda_i^{(n)}}. \tag{1.41}$$

This allows us to rewrite equation 1.39 as:

$$\begin{pmatrix} (\Delta_s \vec{u}_i^{(n)})_{\parallel} \\ (\Delta_s \vec{u}_i^{(n)})_{\perp} \end{pmatrix} \approx -\eta (\lambda_i^{(n)})^2 \begin{pmatrix} 1 + \frac{\alpha}{\eta} (\frac{|\vec{W}_i^{(n)}|}{\lambda_i^{(n)}})^2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{u}_i^{(n)} \cdot \nabla_{\vec{u}_i^{(n)}} \xi^s \\ \hat{v}_i^{(n)} \cdot \nabla_{\vec{u}_i^{(n)}} \xi^s \end{pmatrix} \tag{1.42}$$

We see that a backward propagation algorithm which simultaneously modifies the synaptic weights and the gains is equivalent to a gradient descent in the rescaled synapes $u_{ij}^{(n)}$ with time-dependent step-constants. In the direction of the rescaled synaptic vector, the step-constant now has a quadratic dependence on the magnitudes of the gain and the ordinary synaptic vector, and in the hyperplane perpendicular to the direction of the rescaled synaptic vector, the step-constant has a quadractic dependence only on the magnitude of the gain. From equation 1.42, we obtain:

$$\eta_{\parallel} = \eta (\lambda_i^{(n)})^2 + \alpha \mid \vec{W}_i^{(n)} \mid^2 \tag{1.43}$$

$$\eta_{\perp} = \eta (\lambda_i^{(n)})^2. \tag{1.44}$$

The fact that the step-constants are positive-definite ensures that we always take steps in the direction opposite the gradient; therefore, we can classify this approach as a gradient descent algorithm. Momentum can be added to the above argument without loss of generality, since it simply adds

a small amount proportional to the most recent change; the only difference is that now the step-constants are time-dependent.

We note that the derivation of our model, resulting in equation 1.42, depended on the approximation in equation 1.32 that we take sufficiently small steps in changing the ordinary synapses and gains. If the synapses grow to be very large, it is possible that this approximation may break down. However, depending on our choices of η and α, the approximation will be valid for at least part of the evolution of the network and certainly during the early development. In a continuum model, the derivation would be exact, provided that we replace the single pattern energy ξ_s with the total energy over all patterns $\xi = \sum_s \xi_s$. For such a continuum model, we would replace equation 1.32 with:

$$\frac{d\vec{u}_i^{(n)}}{dt} = \frac{d(\lambda_i^{(n)}\vec{W}_i^{(n)})}{dt} = \lambda_i^{(n)}\frac{d\vec{W}_i^{(n)}}{dt} + \vec{W}_i^{(n)}\frac{d\lambda_i^{(n)}}{dt}. \tag{1.45}$$

Similarly, we would replace equation 1.33 with:

$$\begin{aligned}\frac{d\vec{W}_i^{(n)}}{dt} &= -\eta\nabla_{\vec{W}_i^{(n)}}\xi \\ &= -\eta(\hat{W}_i^{(n)}(\hat{W}_i^{(n)}\cdot\nabla_{\vec{W}_i^{(n)}}\xi) + \hat{v}_i^{(n)}(\hat{v}_i^{(n)}\cdot\nabla_{\vec{W}_i^{(n)}}\xi))\end{aligned} \tag{1.46}$$

and equation 1.34 with:

$$\frac{d\lambda_i^{(n)}}{dt} = -\alpha\frac{\partial\xi}{\partial\lambda_i^{(n)}}. \tag{1.47}$$

In equations 1.35, 1.36, and 1.37, we would need to sum both sides of the equation over the pattern index s, to obtain the appropriate relationships for the total energy ξ. In the end, we would obtain an exact nonlinear differential equation of the form:

$$\begin{pmatrix} (\frac{d\vec{u}_i^{(n)}}{dt})_{\parallel} \\ (\frac{d\vec{u}_i^{(n)}}{dt})_{\perp} \end{pmatrix} = -\eta(\lambda_i^{(n)})^2 \begin{pmatrix} 1+\frac{\alpha}{\eta}(\frac{|\vec{W}_i^{(n)}|}{\lambda_i^{(n)}})^2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{u}_i^{(n)}\cdot\nabla_{\vec{u}_i^{(n)}}\xi \\ \hat{v}_i^{(n)}\cdot\nabla_{\vec{u}_i^{(n)}}\xi \end{pmatrix} \tag{1.48}$$

from which we would obtain the effective time-dependent step-constants in equations 1.43 and 1.44.

1.5 Impact of the Effective Time-Dependent Step-Constant

In the previous section, we derived the form of the effective step-constant and determined that $\eta_{\|}$ grows quadratically as a function of the magnitudes of the ordinary synaptic vector and the gain and that $\eta_{\perp}$, the step-constant in the direction of change of $\hat{u}_i^{(n)}$, grows quadratically with the gain. It may be particularly important in the early stages of synaptic development when $\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s}$ is small and $o_i^{(n),s}$ can be approximated linearly as: [9]

$$\begin{aligned} o_i^{(n),s} &= \frac{1}{1+e^{-\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s}}} \\ &\approx \frac{1}{2-\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s}} \\ &\approx \frac{1}{2}(1+\frac{\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s}}{2}) \end{aligned} \tag{1.49}$$

In this regime, therefore, the error signals can be approximated as polynomial functions of $\vec{u}_i^{(n)} \cdot \vec{F}^{(n),s}$, and the the quadratic dependence of the step-constants on $\mid \vec{W}_i^{(n)} \mid$ and $\lambda_i^{(n)}$ is likely to be significant. [10]

In the next chapter, we will demonstrate that the use of high momentum leads to shorter convergence time. When gain modification is combined with high momentum, there is further improvement in convergence time. Empirical results suggest that when high momentum is used shorter convergence times result because the synapses develop more rapidly. When gain modification is added to high momentum, the rate of development of the effective synapses, $u_{ij}^{(n)}$, is accelerated further.

[9] The initial synaptic weights are small: between -0.1 and 0.1, and the connectivity of the networks we considered in the next chapter was small. Additionally, the input patterns were confined to the unit disk. Therefore, we can expect that this approximation will be valid at the beginning of synaptic development.

[10] Furthermore, note that the ordinary error signals, as shown earlier, are explicitly dependent on the gains. This may also play a significant role in the time evolution of the network.

Chapter 2

Experimental Benchmarks for Gain Modification and Momentum

2.1 Momentum and the Concentric Circle Paradigm

We describe below some benchmarks for the concentric circle paradigm. In this paradigm, we train a backward propagation network with two inputs representing the x and y coordinates of a point in the unit disk. Within the unit disk, there are two pattern classes separated by a circular boundary at $r = \frac{1}{\sqrt{2}}$. This divides the class into two regions, an inner disk and an outer annulus, both of area $\frac{\pi}{2}$. For the single output unit, the target output for patterns in the outer annulus is 1 and for patterns in the inner disk is 0. In the simulations described below, the network had one hidden layer of 6 hidden units, and the network was trained on 40 patterns randomly selected from the unit disk. For each setting of the parameters, we repeated the experiment 90 times starting the weights at different random points in a small hypercube; each initial weight was chosen randomly betweeen 0.1 and -0.1. Additionally, a different randomly selected set of 40 patterns was used in each experiment. In all experiments the step-constant in the first layer of connections was two times the size of the step constant in the second layer. We have found that this accelerates the learning procedure, a fact

which is consistent with what Becker and Lecun (1988) [2] have observed. In their work, they determined that the step constant should be scaled by a factor of $\frac{1}{\sqrt{N_c}}$, where N_c, for a given layer of synapses, is the number of connections in that layer onto a neuron in the next layer of neurons. If this scaling is a general principal, then, for our particular network architecture, we should have increased the step-constant in the first layer of synapses by a factor of $\sqrt{3}$. However, by choosing a factor of 2, we are at least approximately close to what Becker and LeCun found to be ideal for the binary input paradigm which they investigated. As of yet, we have not studied in great detail the scaling of step-constants as a function of the number of synaptic connections in a layer. However, for consistency, in the simulations described in the next section, we chose α, the step-constant for gain modification, to be twice as large for neurons in the hidden layer compared to its value in the output layer.

In one set of experiments (series ab), we took the step-constant to be $\eta = 0.4$. The momentum was varied by increments of 0.1 from 0.0 to 0.9, and 90 experiments were performed at each value of momentum. The mean and standard deviation of convergence time are plotted for each value of momentum in figure a.1 of appendix a. Only those experiments converging within 20000 epochs [1] [2] were used to compute the means and standard deviations. From the graph, it is apparent that the mean convergence time decreases with increasing momentum. The large error bars, however, are somewhat misleading and tend to underemphasize the rather dramatic improvement in convergence time at high momentum. The size of the errors is typically comparable to the mean and is due to the fact that there are a number of trials for which the convergence time is several times larger than for the majority of the trials. Given the limited number of trials, 90, this tends to make the error bars large and to mask the improvement of the central cluster of trials. With increasing momentum, the central cluster of convergence times becomes narrower and moves toward shorter times. We illustrate this point in a series of graphs for series ab, figures a.4- a.13, ordered by momentum, in which we plot a histogram of number of trials converging vs. convergence time.

[1] An epoch is defined to be a complete presentation of the data set sequentially. However, we modify the network following the presentation of each pattern.

[2] In our experiments, we defined the convergence of a network to occur when the network output was within 0.1 of the target output for all training patterns.

It is interesting to note that at low momentum, there is evidence of at least two distributions in the convergence times. These two clusters gradually move to shorter convergence times and merge to form a single cluster at low convergence time.

Figure a.2 of appendix a shows the fraction of trials converging within 20000 epochs. Over most of the range of κ, that is for κ between 0.0 and 0.8, the percentage of trials converging is in the range of$\approx 89 - 98\%$. At very high momentum, however, $\kappa = 0.9$, the percentage of trials converging drops to 79%. At the same time, the amount of generalization by the network is not momentum dependent. This is readily apparent in figure a.3 of appendix a, in which we plot the mean and standard deviation of the fraction of 5000 randomly selected test patterns which were correctly identified by the network after the network converged. We also graph the fraction which could be correctly identified by the network's "best guess", that is the output which the network was closest to, 0 or 1. Means and standard deviations are depicted only for those trials which actually converged within 20000 epochs. As indicated by the small error bars, the generality of the solutions varies only weakly across trials for a given value of momentum. Furthermore, as we have noted, the mean generality is constant across momentum. We expect, therefore, that solution generality should not have a strong dependence on convergence time.

We can conclude also that the length of convergence time has very little to do with the generality of the solution reached for this paradigm; we infer this from the fact that the error bars are small in the generalization graph in figure a.3. It is interesting to ask, then, which features might distinguish the network solutions with long convergence time from those with short convergence time. In appendix b, we have produced a scatter plot of the mean and standard deviation (over the neurons in a particular solution) of the magnitude of synaptic vectors vs. convergence time for series ab. As defined in the previous chapter, each of these vectors includes both the ordinary synapse and the bias. For each layer, there is a set of 3 graphs, ordered in increasing momentum, one at zero momentum ($\kappa = 0$), one at intermediate momentum ($\kappa = 0.5$), and one at high momentum ($\kappa = 0.9$). The graphs for layer 2 are in figures b.1- b.3, and those for layer 1 are in figures b.4-b.6. We note that as the momentum is increased, the longer convergence times move toward the cluster of convergence times at shorter times, this region of the graph becomes more dense, and itself moves

simultaneously closer to the origin along the time axis. As a consequence, as the momentum is increased and the number of trials converging within the first several thousand iterations becomes more dense, we observe that, as a population, the synaptic norms rise more rapidly as a function of convergence time.

2.2 Gain Modification Combined with High Momentum Synaptic Modification

We have seen from the histograms of number of trials converging vs. convergence time for series ab in appendix a, that high momentum can dramatically improve the convergence rate for the majority of the trials. With this in mind, we combined the gain modification procedure described earlier with high momentum synaptic modification. We chose α, the gain modification step-constant, to be an order of magnitude smaller than the synaptic modification step-constant, η, since the gain is, in general, a more sensitive parameter. In series ag, we examined three different cases: no momentum (sag3), high momentum (sag2), and high momentum with gain modification (sag4,sag5,sag6). In each run, we performed 500 experiments. We summarize the parameters chosen for these runs: [3]

Run Number	η	κ	α
sag3	0.3	0.0	0.000
sag2	0.3	0.8	0.000
sag5	0.3	0.8	0.020
sag4	0.3	0.8	0.038
sag6	0.3	0.8	0.060

In figure c.1 and c.2 of appendix c, we plot the percentage of trials converging vs. convergence time on two different time scales for the runs listed above. Detailed histograms of sag2, sag3, and sag4, comparing the three cases on two different time scales, the first 6000 epochs (figures c.3-c.5), and the full 20000 epochs (figures c.6-c.8), are also provided in appendix c.

[3] As noted earlier, $\eta \rightarrow 2\eta$ and $\alpha \rightarrow 2\alpha$ for the first layer of synapses and the gains of the hidden units respectively. The values for η and α in the table are those values used in the second layer of synapses and the gains of the output units respectively.

Examining the graphs in figures c.1 and c.2, we see that without momentum or gain modification, 80% of the simulations converge within $\approx$ 10000 epochs. High momentum clearly leads to much more rapid convergence, for we obtain the 80% level within $\approx$ 1500 epochs. When gain modification is added to high momentum, we achieve the 80% level within $\approx$ 500 epochs. This is 3 times faster than high momentum without gain modification and 20 times faster than the bare algorithm. This is a tantalizing result, for on a more complex problem where longer convergence times are more probable, reaching the 80% level in a third of the time (compared with high momentum alone) could be quite significant. Asymptotically, the runs in which gain modification was combined with high momentum (sag4, sag5, sag6) also achieved the highest percentage of trials converging, $\approx$ 95-98%. For high momentum without gain modification, the asymptotic convergence rate was slightly lower at $\approx$ 93%, and for the bare algorithm, even less at $\approx$ 90%.

A comparison of the synapses obtained in these three cases is illustrated in the graphs in appendix d. Figures d.1, d.2, and d.3 graph the the synaptic vector norm [4] in layer 2 vs. convergence time. Figure d.1 is the no momentum case (sag3), figure d.2 the high momentum case (sag2), and figure d.3 the high momentum with gain modification case (sag4). As a population there is a trend toward a longer synaptic norm in layer 2 as time increases when the bare algorithm is used (figure d.1). This increase in synaptic norm occurs more rapidly when high momentum (figure d.2) is added to the bare algorithm. When we combine high momentum with gain modification, (figure d.3) there is some depression of the size of the synaptic norm. In this case, however, the length of the effective synaptic vector, or rescaled synaptic vector, depicted in figure d.4 in appendix d, develops much more rapidly than in the high momentum case without gain modification. This is due to the fact that the gains in this layer as a population are typically $\approx$ 1.4-2.5 (see figure d.5 of appendix d where the gains are plotted as a function of synaptic norm for layer 2). This seems to speed the process of learning, since solutions with longer synaptic norms are achieved without the requirement of the synapses' becoming large; a single multiplicative factor, the gain, hastens the change of the length of the synaptic vector.

[4] As before, this vector includes both the ordinary synaptic vector and the bias.

In layer 1, a similar trend obtains (figures d.6- d.9). [5] As in layer 2, longer effective synapses in layer 1 are achieved more rapidly by modifying the gain parameter. When we compare figures d.6 and d.7, we see that the increase in $E(\mid \vec{W}_i^{(1)} \mid)$ occurs more rapidly as a function of convergence time for the majority of the population as we go from no momentum to high momentum. [6] However, the synaptic norms (figure d.8) are not significantly depressed in the case of high momentum with gain modification; this leads to rescaled synaptic vector norms which are much larger than the other two cases (see figure d.9). [7] In figure d.8, the ordinary synapses are probably not significantly depressed in layer 1 because the step-constant was twice as large for layer 1, as we noted earlier.

The mean gain is plotted versus mean synaptic vector norm in figure d.10. The gains are principally between 1.0 - 2.0 in this layer and act to accelerate the development of the length of the effective synaptic vectors.

Rapid development of the synaptic vector norms appears to be the factor which decreases the convergence time when high momentum is used, and this effect is further enhanced when gain modification is used with high momentum. We note also for completeness that the use of gain modification did not in any way degrade the generality of the solutions obtained.

2.3 Future Directions

In the work described above, we have shown how gain modification can enhance the improvements in convergence rate obtained with high momentum in synaptic modification for a simple concentric circle paradigm. For completeness, we plan to do simulations in which gain modification is used without momentum. In addition, further analysis and reduction of the data already obtained is in progress. We plan to establish similar benchmarks for higher dimensional problems, for instance for higher dimensional concentric hyperspheres. This will allow a comparison of how the improvements obtained with gain modification scale with dimensionality. Finally, we will establish benchmarks for gain modification for more complex problems such

[5] As before error bars represent the standard deviation with respect to the neurons in the layer.

[6] E is the expected value, or mean.

[7] Note that the scale of the graph in figure d.9 is different from the previous graphs.

as parity.

2.4 Acknowledgement

We wish to acknowledge Professor Leon N Cooper, who first suggested the idea of modifying gains in a backward propagation network and to thank Nathan Intrator, Dr. Christopher Scofield, David Ward, and Michael Perrone for a number of useful discussions and suggestions concerning this work.

Bibliography

[1] C. M. Bachmann, *"Generalization and the Backward Propagation Neural Network"*, ARO Technical Report, Physics Dpt. & Ctr. for Neural Science, Brown University, Providence, R.I., Jan. 14, 1988.

[2] S. Becker, Y. LeCun, *Improving the Convergence of Back-Propagation Learning with Second Order Methods*, in **Proceedings of the 1988 Connectionist Models Summer School**, June 17-26, 1988, Carnegie Mellon University, eds. D. Touretzky, G. Hinton, T. Sejnowski, p. 29- 37, Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[3] J. J. Hopfield, *"Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons",Proc.Natl.Acad. Sci. USA, Vol.81, pp.3088-3092, May 1984, Biophysics*

[4] J. K. Kruschke, *Creating Local and Distributed Bottlenecks in Hidden Layers of Back-Propagation Networks* in Proceedings of the 1988 Connectionist Models Summer School, June 17-26, 1988, Carnegie Mellon University, eds. D. Touretzky, G. Hinton, T. Sejnowski, Morgan Kaufmann Publishers, San Mateo, CA, 1989, pp. 120-126.

[5] R. A. Jacobs, *"Increased Rates of Convergence Through Learning Rate Adaptation"*, Neural Networks, Vol. 1, pp. 295-307, 1988.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *"Learning Internal Representations by Error Propagation"*, in **Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol.1: Foundations**, ed. D.E. Rummelhart, J.L. McClelland,The MIT Press, Cambride Mass., 1986, pp.318-362.

[7] P. J. Werbos, *"Generalization of Backpropagation with Application to a Recurrent Gas Market Model"*, Neural Networks, Vol. 1, 1988, pp. 339-356.

Appendix a
Graphs of series ab: Convergence Properties as a Function of Momentum

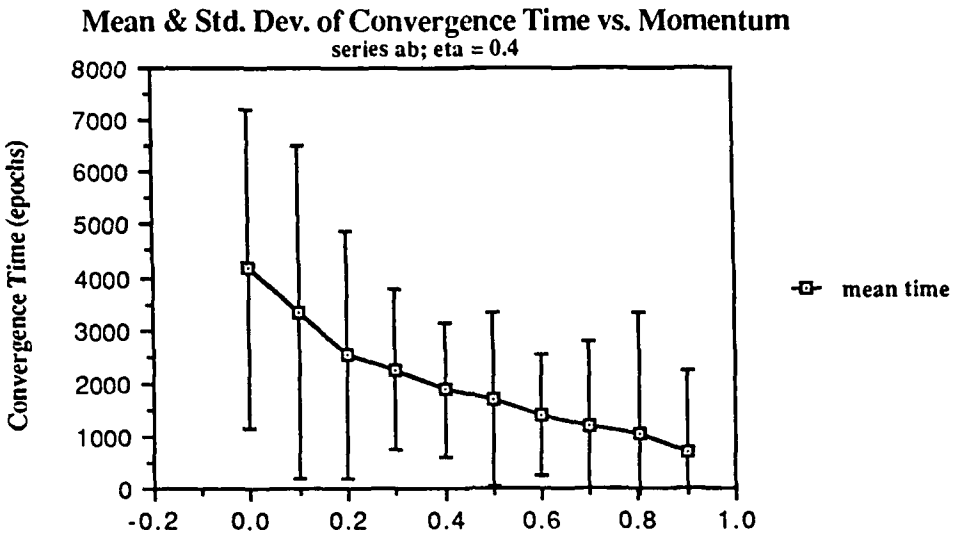

Fig. a.1 momentum

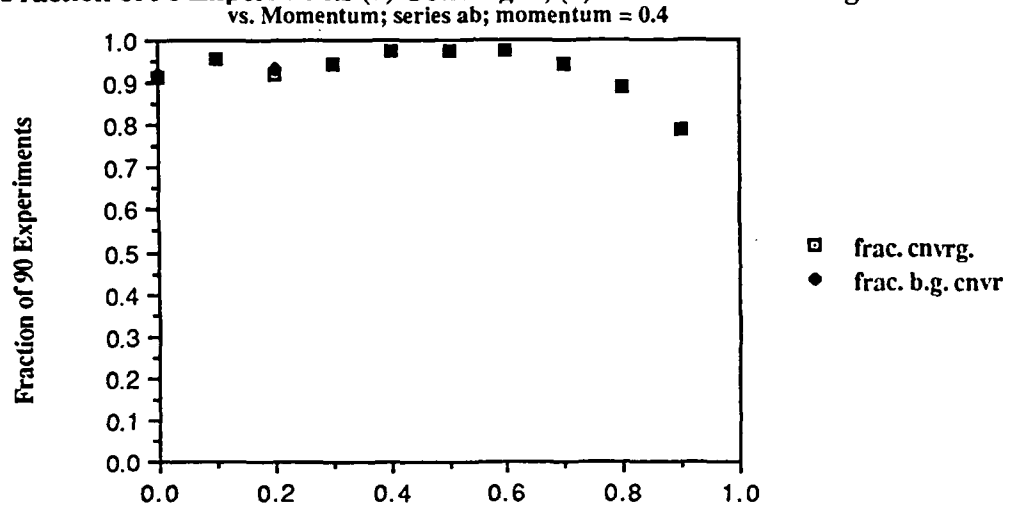

Fig. a.2 momentum

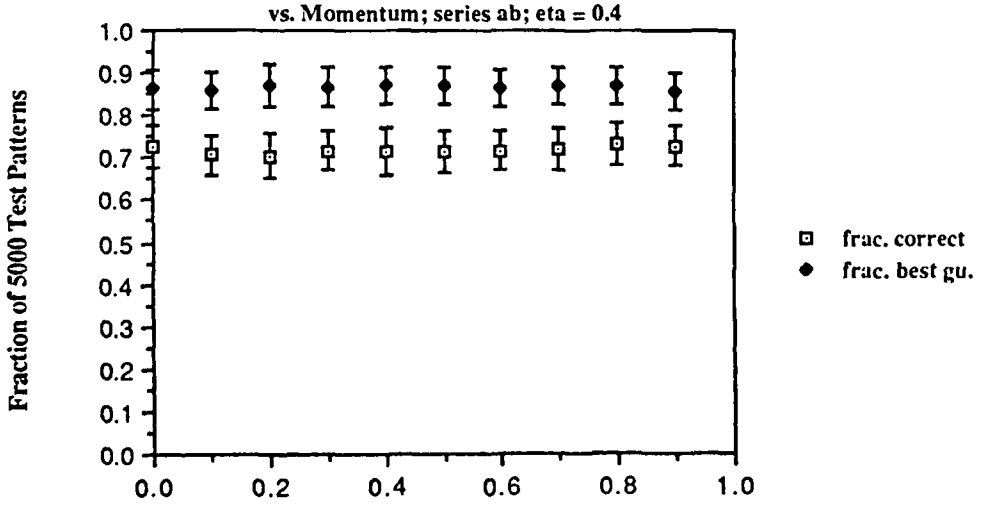

Fig. a.3 momentum

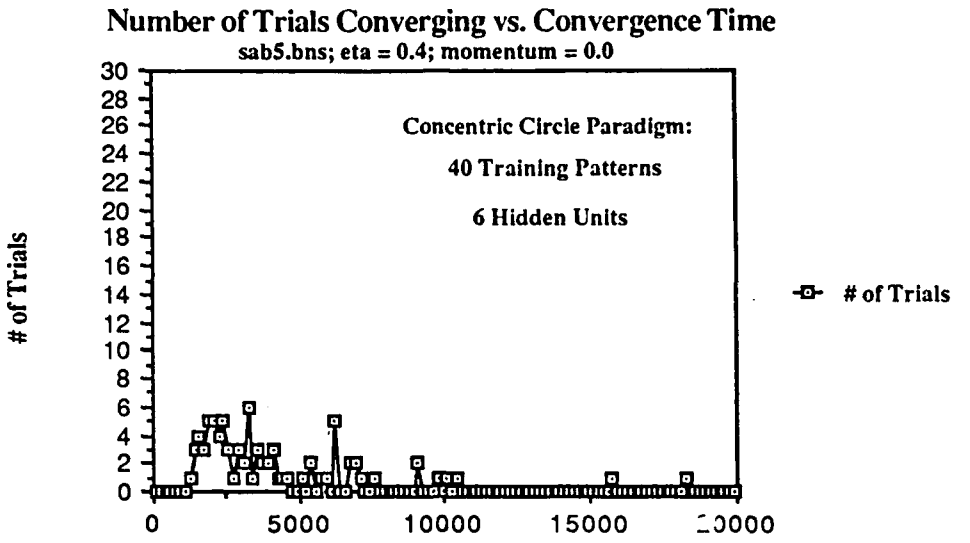

Fig. a.4

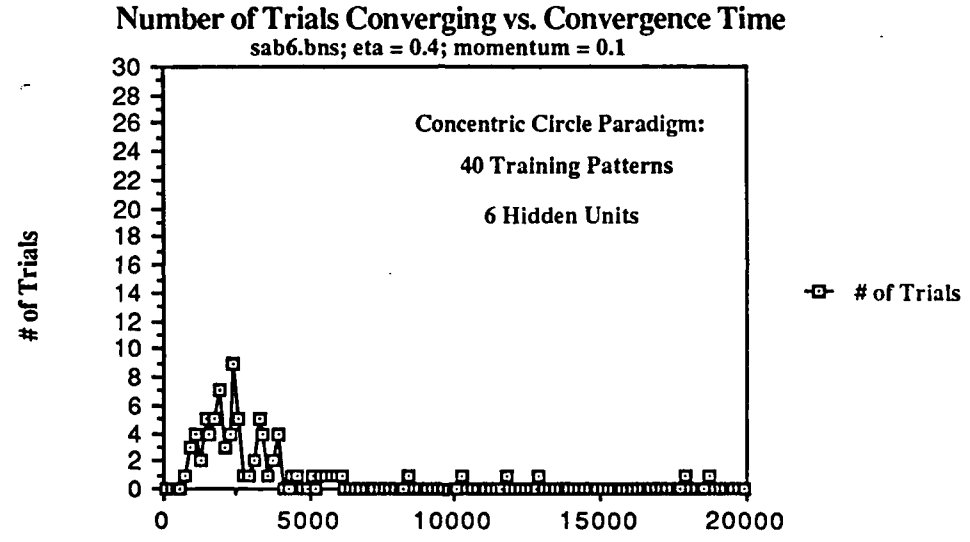

Fig. a.5

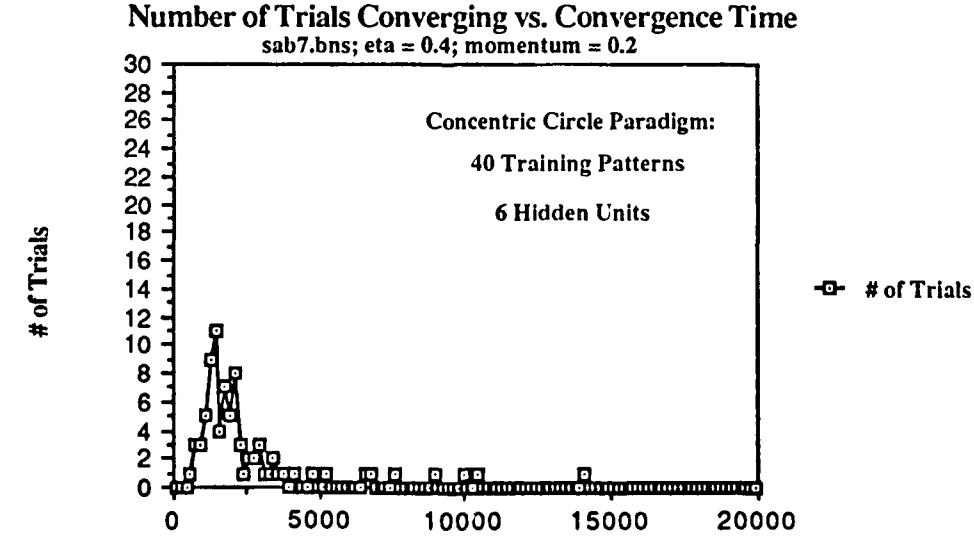

Fig. a.6

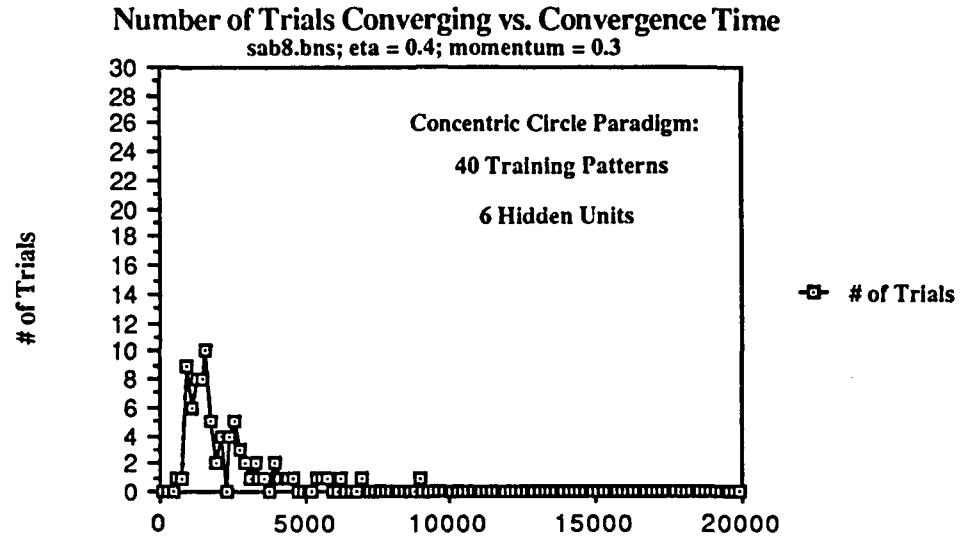

Fig. a.7

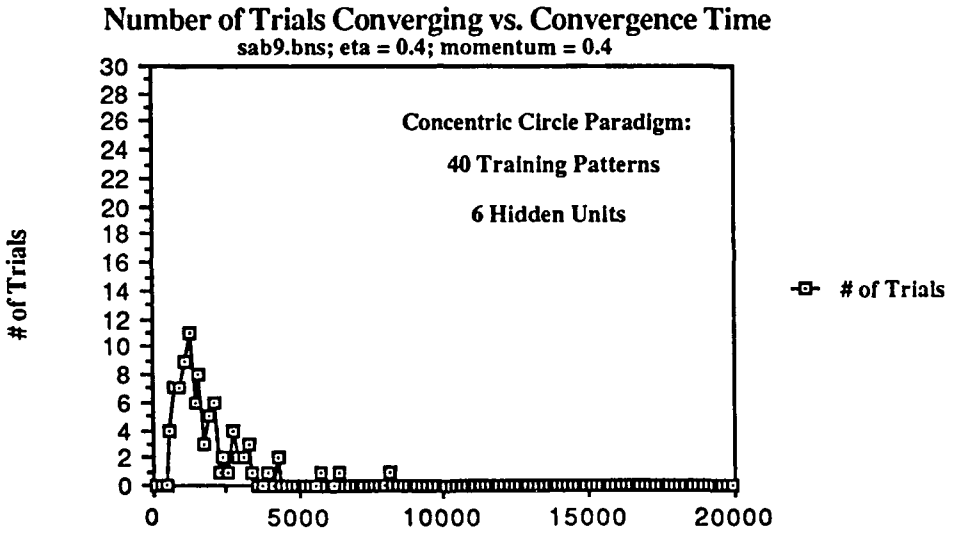

Fig. a.8

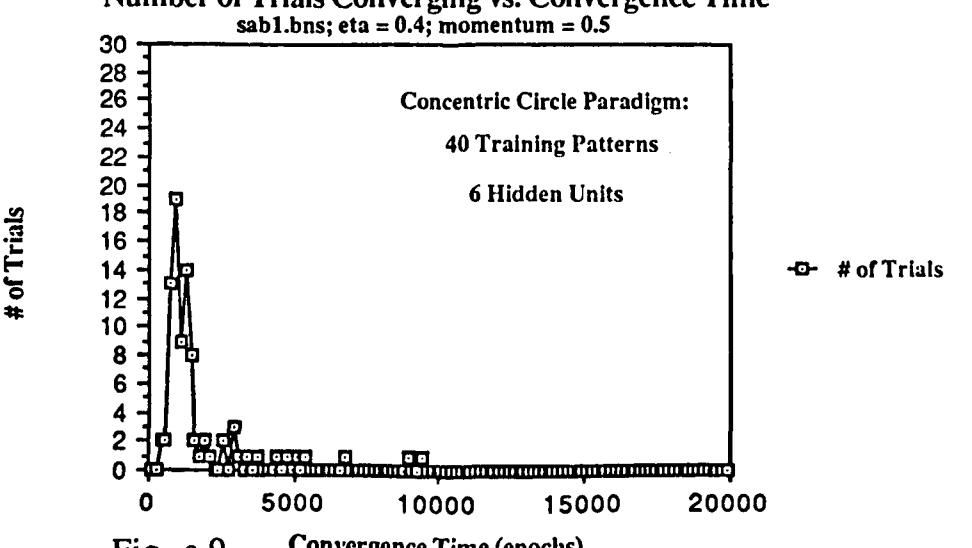

Fig. a.9

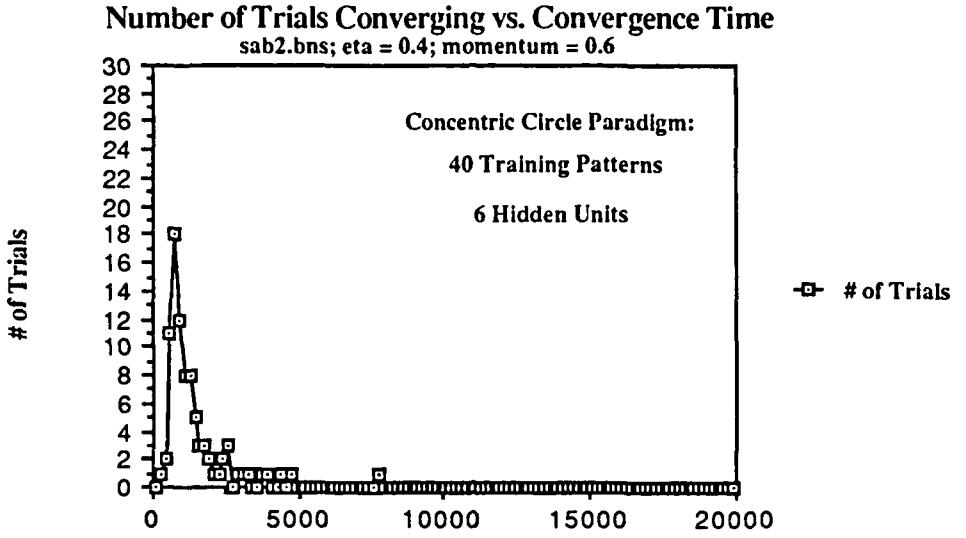

Fig. a.10

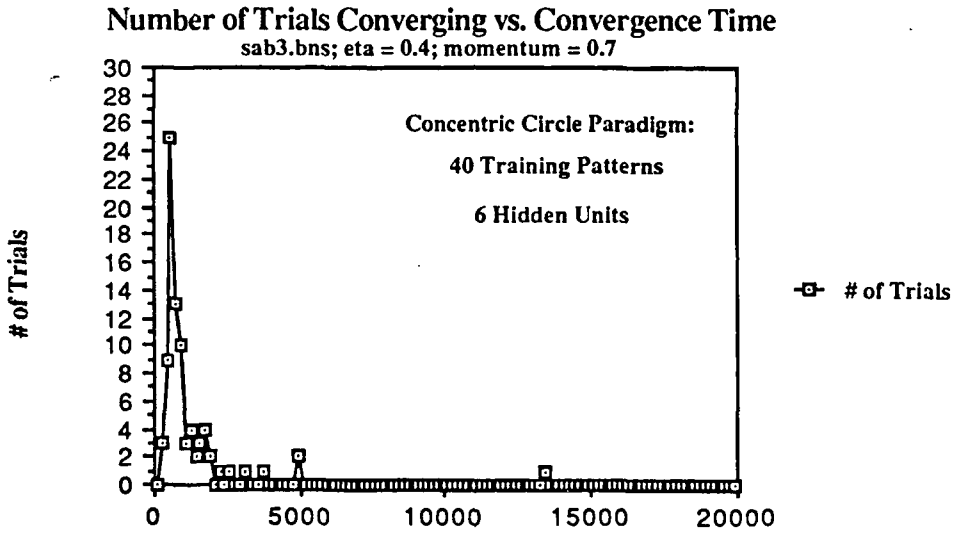

Fig. a.11

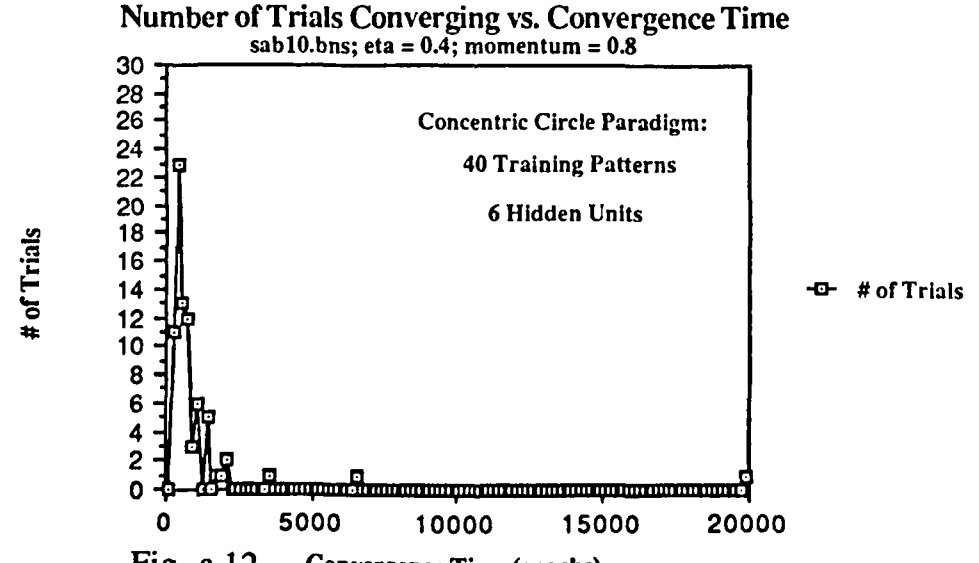

Fig. a.12

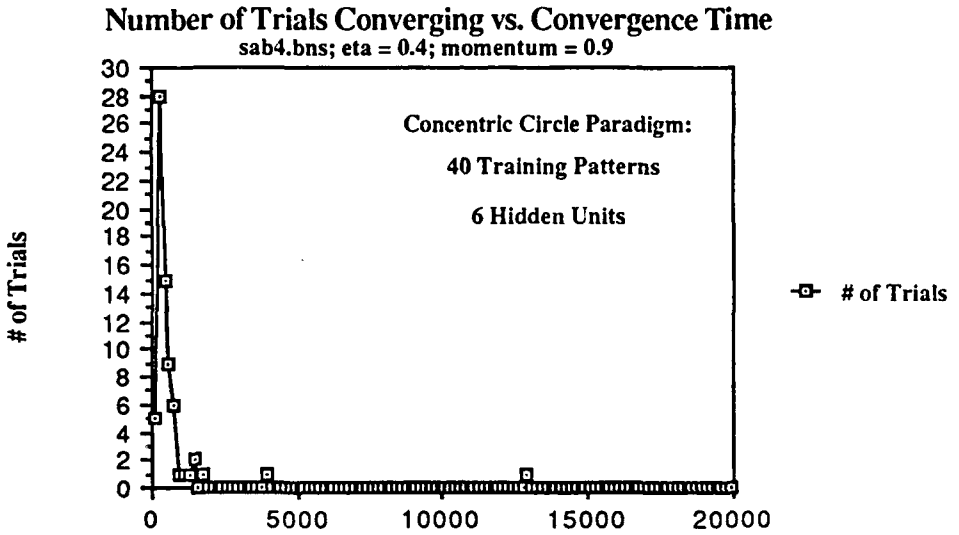
Number of Trials Converging vs. Convergence Time
sab4.bns; eta = 0.4; momentum = 0.9
# of Trials
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
0
5000
10000
15000
20000
Concentric Circle Paradigm:
40 Training Patterns
6 Hidden Units
# of Trials
Convergence Time (epochs)

Fig. a.13

Appendix b
Graphs of series ab: Magnitude of Synaptic Vectors vs. Convergence Time

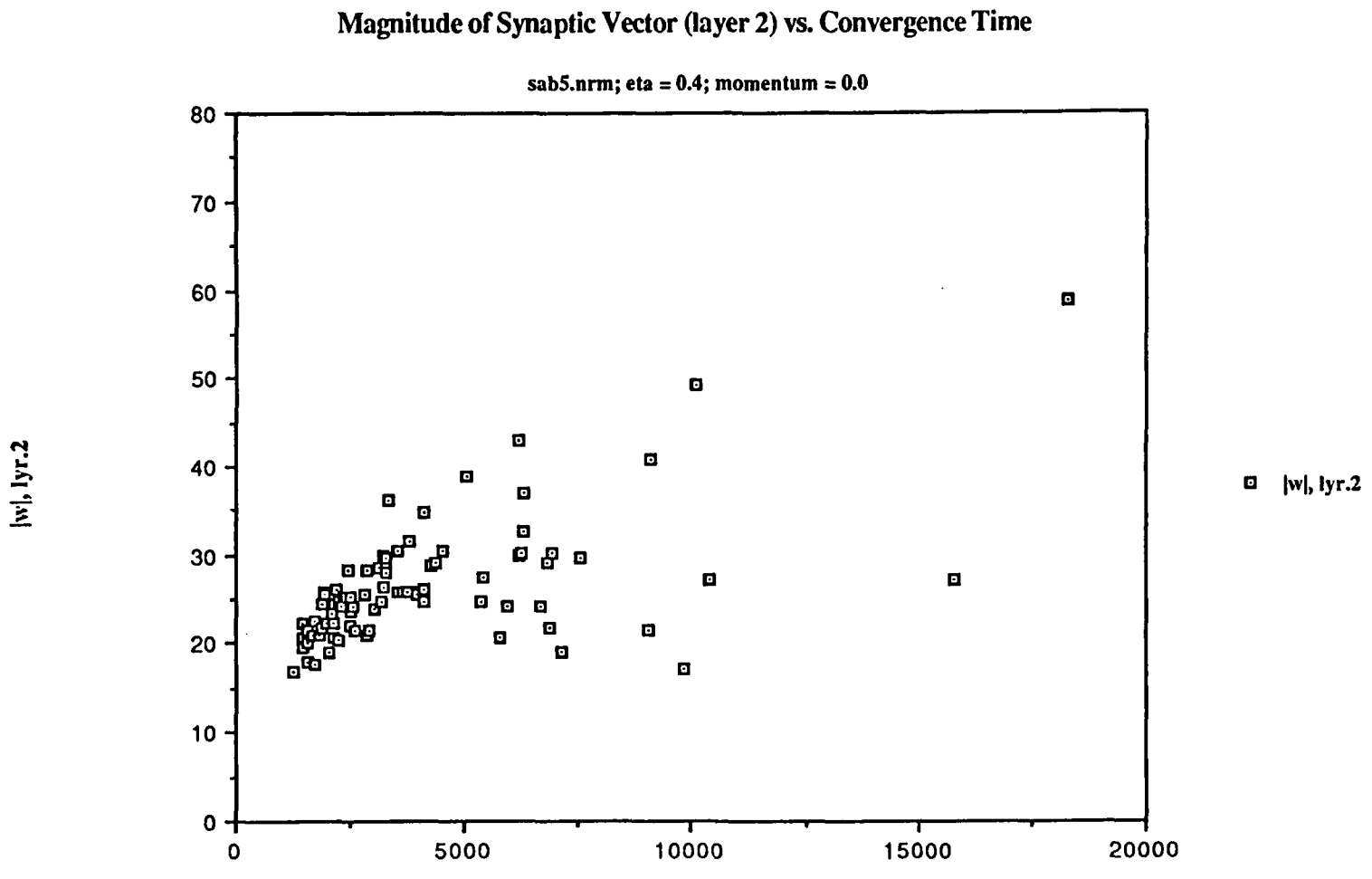
Magnitude of Synaptic Vector (layer 2) vs. Convergence Time
sab5.nrm; eta = 0.4; momentum = 0.0
|w|, lyr.2
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
|w|, lyr.2

Fig. b.1

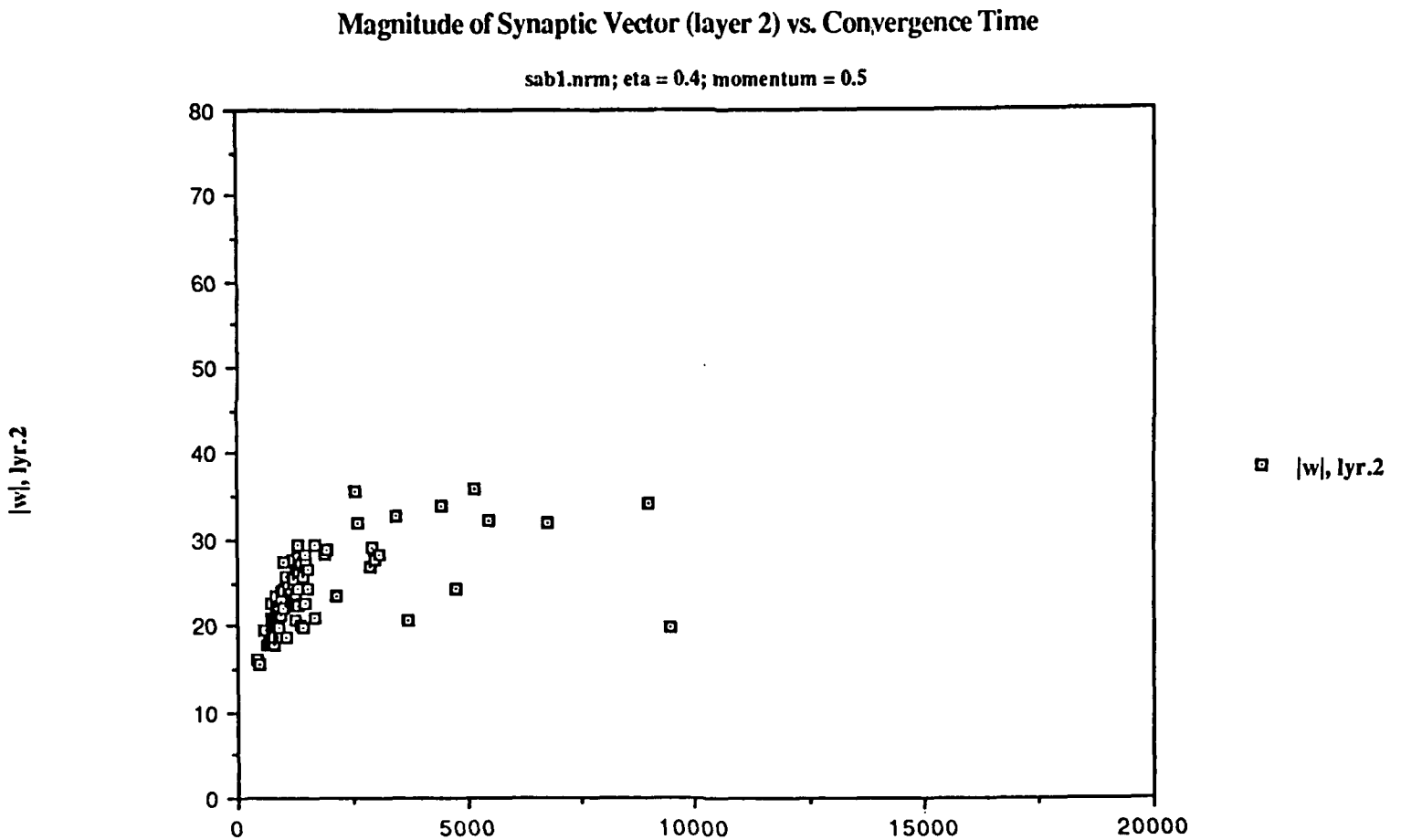
Magnitude of Synaptic Vector (layer 2) vs. Convergence Time
sab1.nrm; eta = 0.4; momentum = 0.5
|w|, lyr.2
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
|w|, lyr.2
Convergence Time (epochs)

Fig. b.2

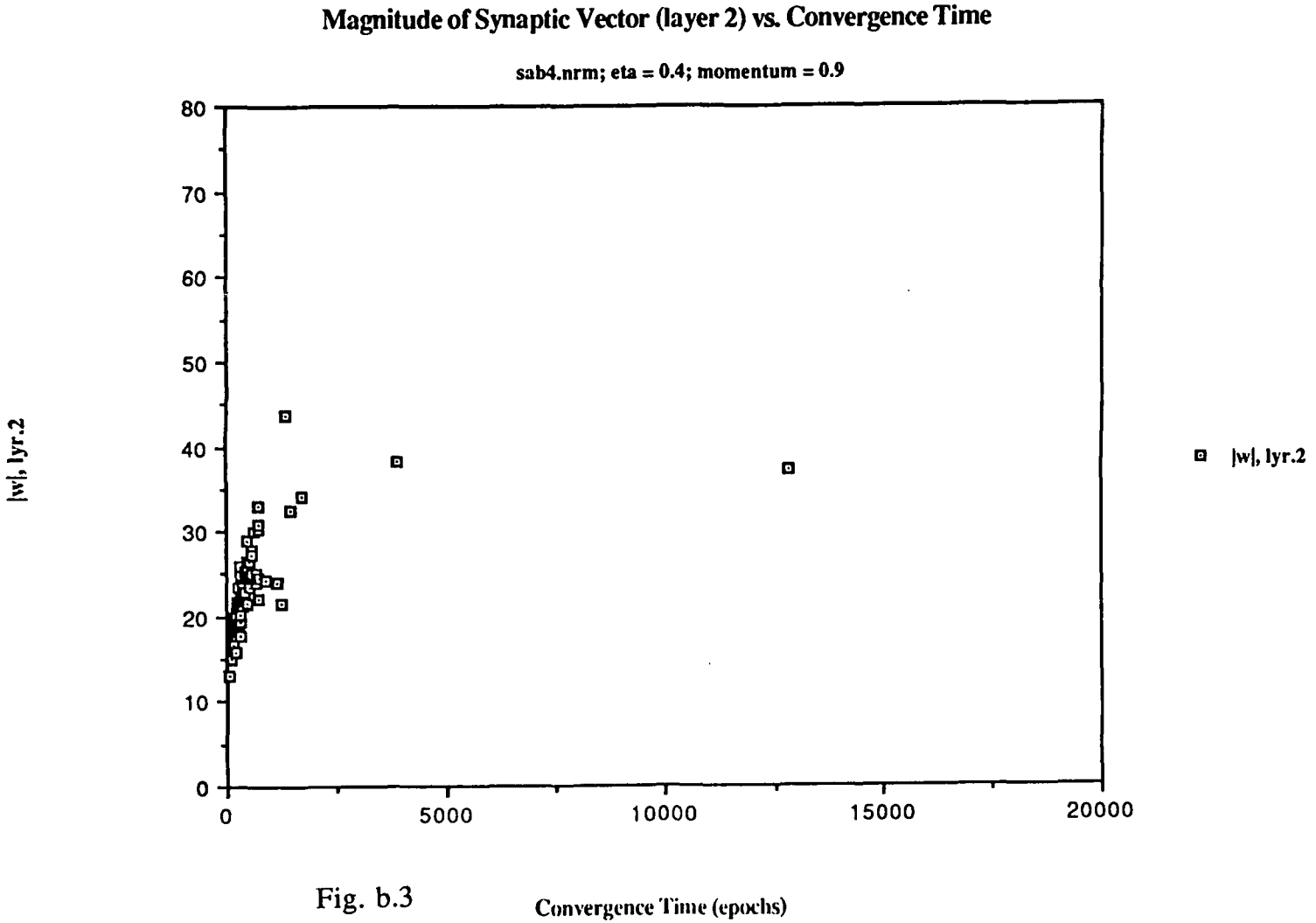
Magnitude of Synaptic Vector (layer 2) vs. Convergence Time
sab4.nrm; eta = 0.4; momentum = 0.9
|w|, lyr.2
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
|w|, lyr.2
Convergence Time (epochs)

Fig. b.3

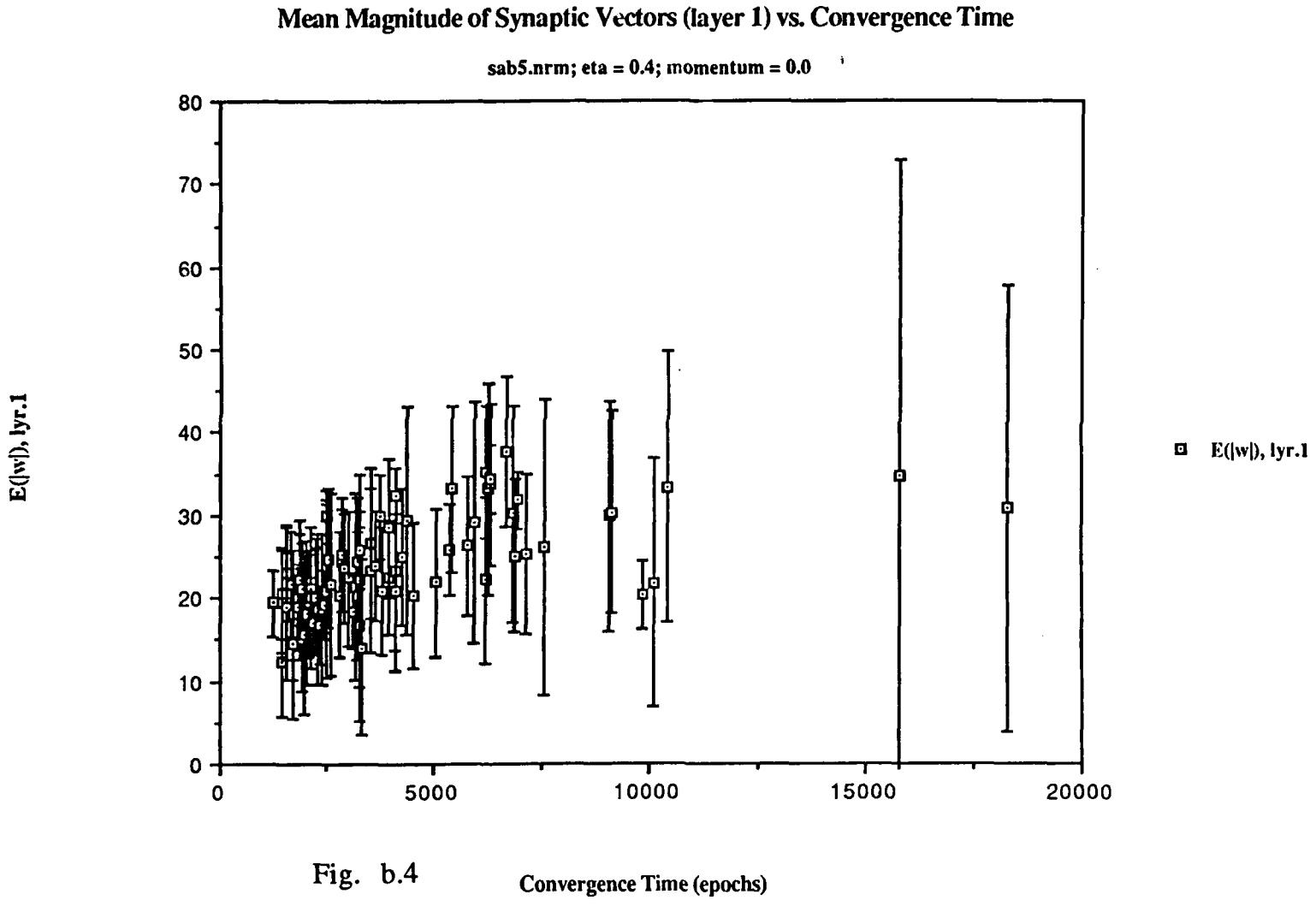
Mean Magnitude of Synaptic Vectors (layer 1) vs. Convergence Time
sab5.nrm; eta = 0.4; momentum = 0.0
E(|w|), lyr.1
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
E(|w|), lyr.1

Fig. b.4

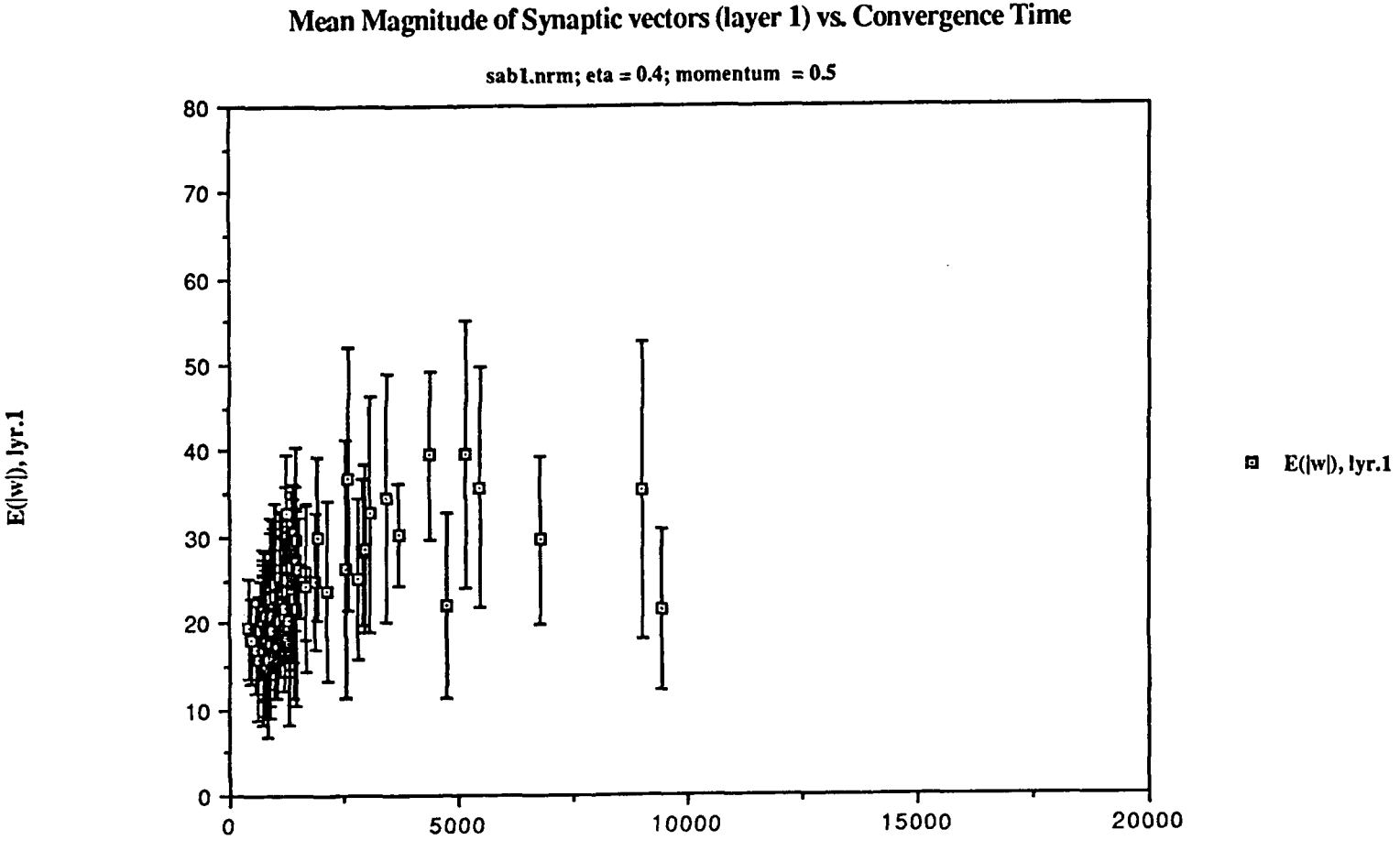
Mean Magnitude of Synaptic vectors (layer 1) vs. Convergence Time
sab1.nrm; eta = 0.4; momentum = 0.5
E(|w|), lyr.1
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
E(|w|), lyr.1

Fig. b.5

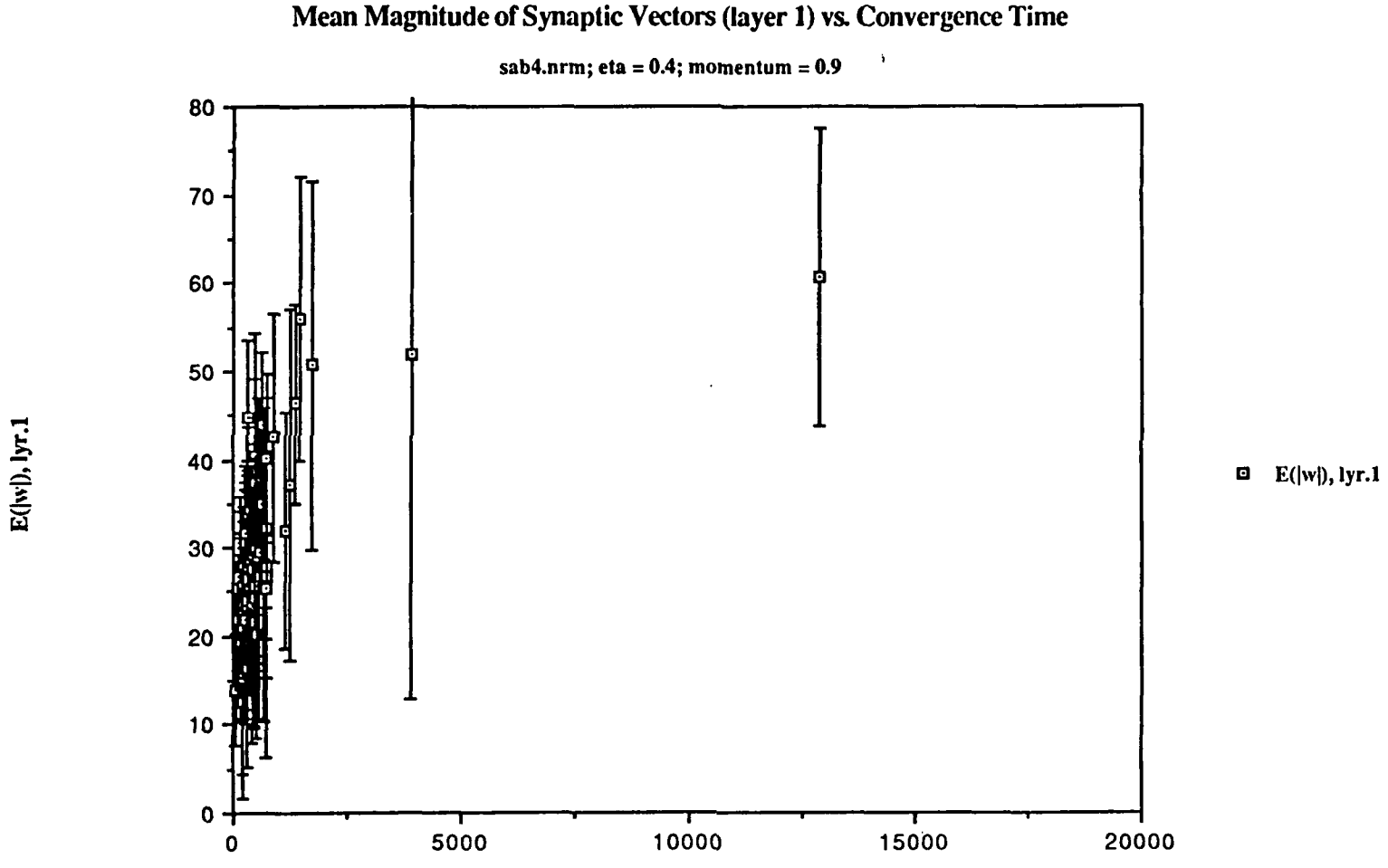
Mean Magnitude of Synaptic Vectors (layer 1) vs. Convergence Time
sab4.nrm; eta = 0.4; momentum = 0.9
E(|w|), lyr.1
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
E(|w|), lyr.1

Fig. b.6

Appendix c
Graphs of series ag: Convergence Properties for (1)No Momentum, (2) High Momentum, (3) High Momentum with Gain Modification

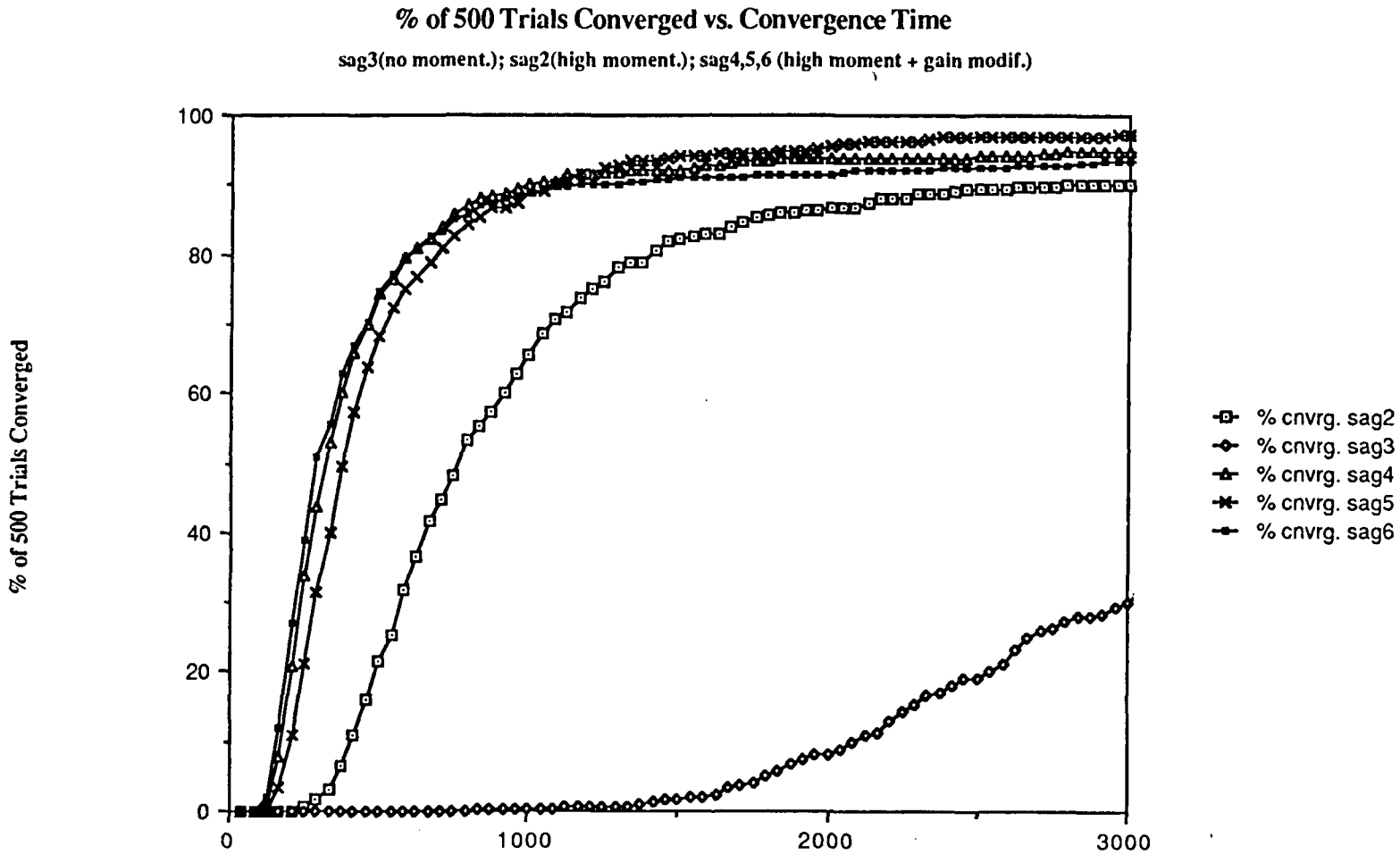
% of 500 Trials Converged vs. Convergence Time
sag3(no moment.); sag2(high moment.); sag4,5,6 (high moment + gain modif.)
% of 500 Trials Converged
100
80
60
40
20
0
0
1000
2000
3000
% cnvrg. sag2
% cnvrg. sag3
% cnvrg. sag4
% cnvrg. sag5
% cnvrg. sag6
Convergence Time (epochs)

Fig. c.1

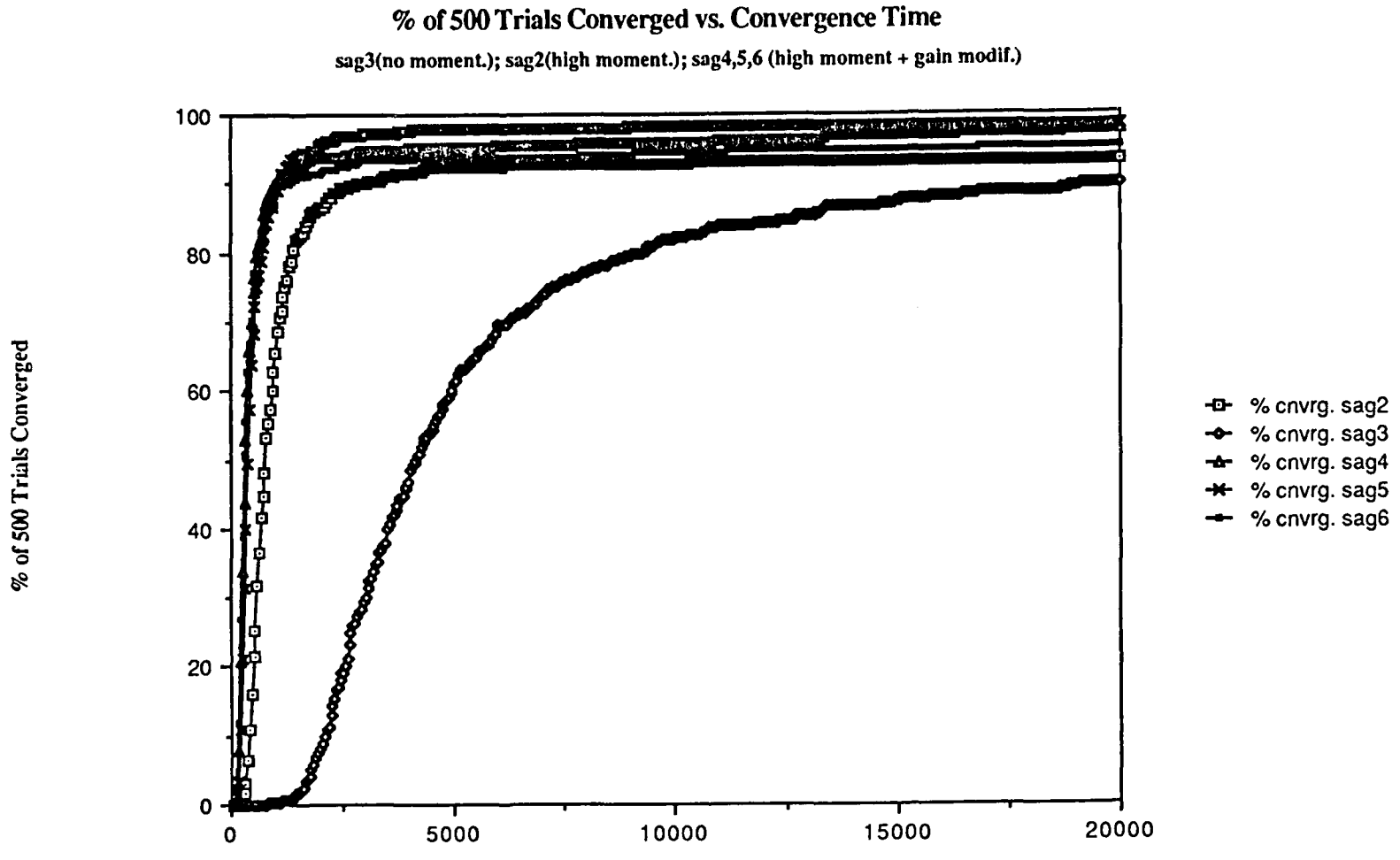
% of 500 Trials Converged vs. Convergence Time
sag3(no moment.); sag2(high moment.); sag4,5,6 (high moment + gain modif.)
% of 500 Trials Converged
100
80
60
40
20
0
0
5000
10000
15000
20000
Convergence Time (epochs)
% cnvrg. sag2
% cnvrg. sag3
% cnvrg. sag4
% cnvrg. sag5
% cnvrg. sag6

Fig. c.2

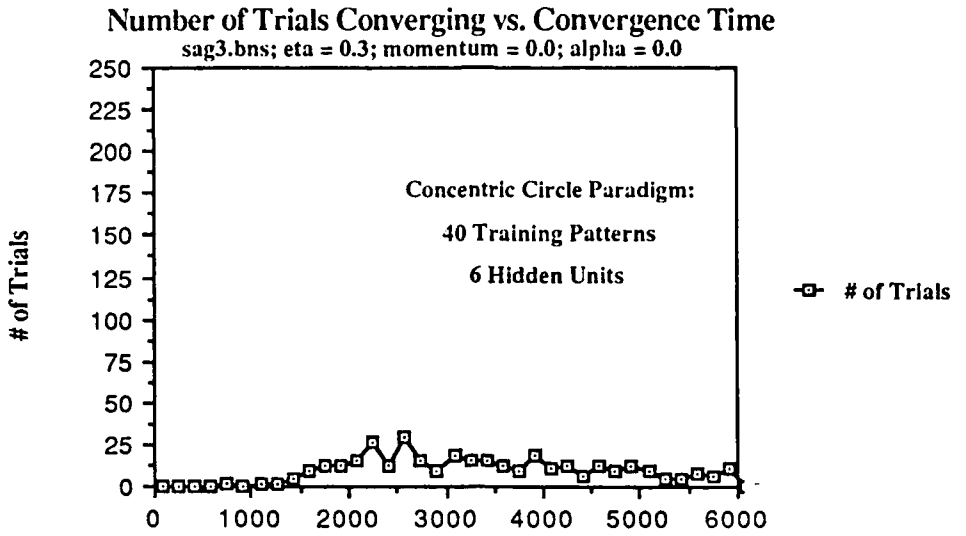

Fig. c.3

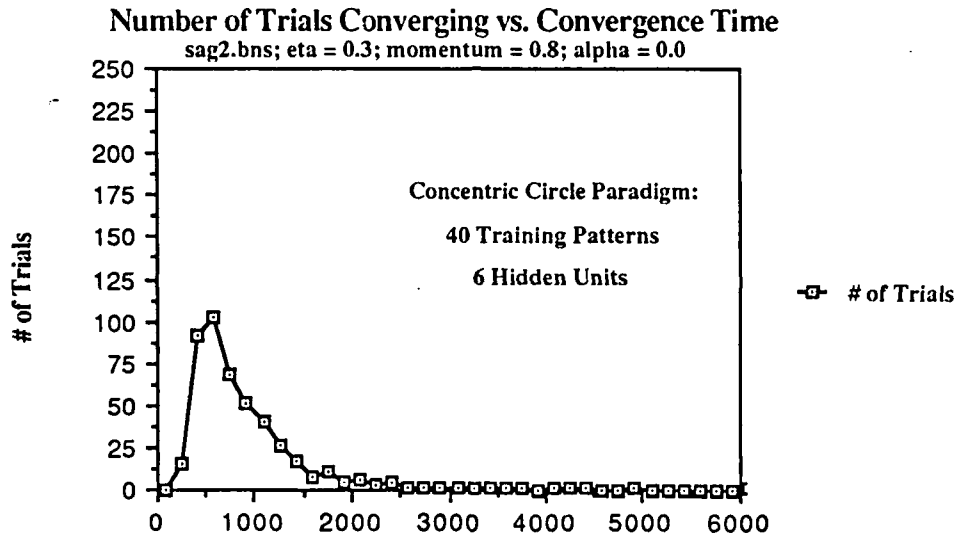

Fig. c.4

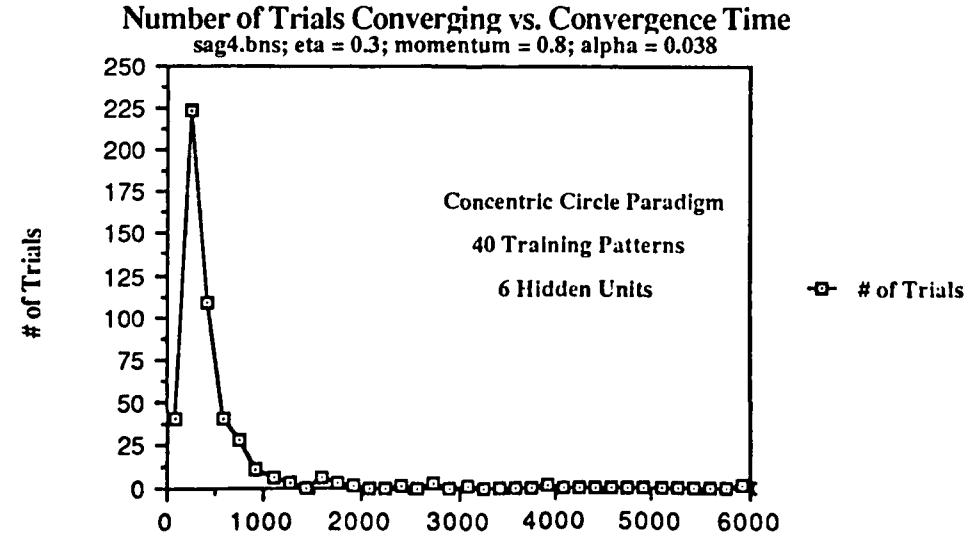

Fig. c.5

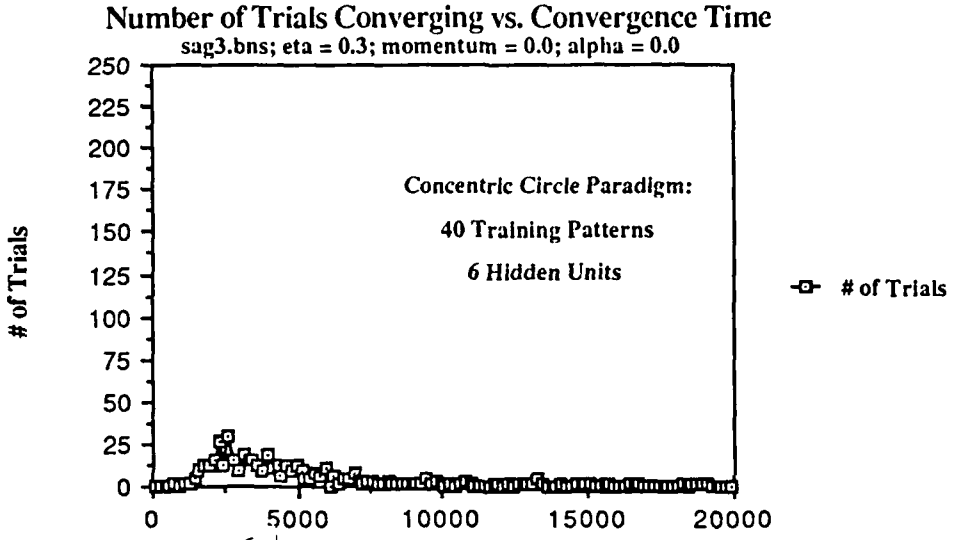

Fig. c.6

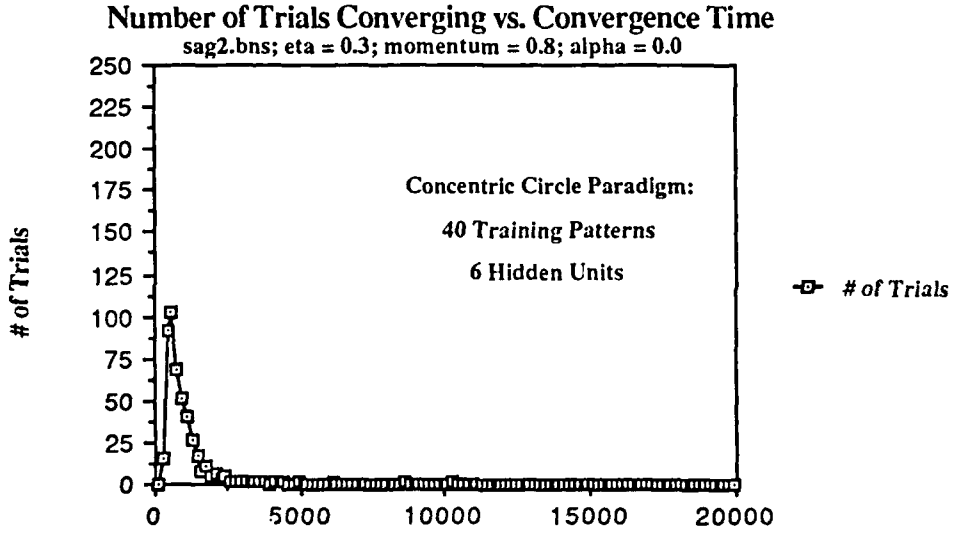

Fig. c.7

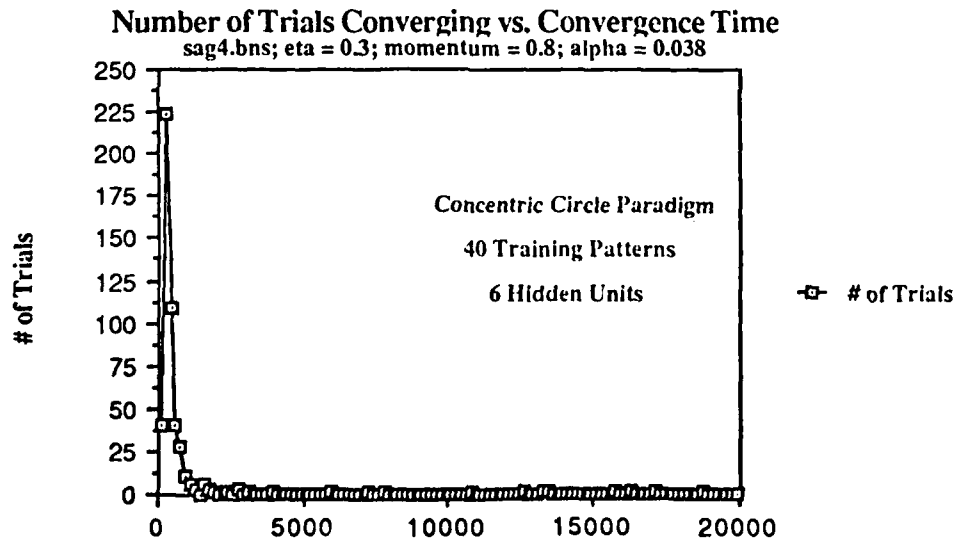

Fig. c.8

Appendix d
Graphs of series ag: Magnitude of Synaptic Vectors vs. Convergence Time

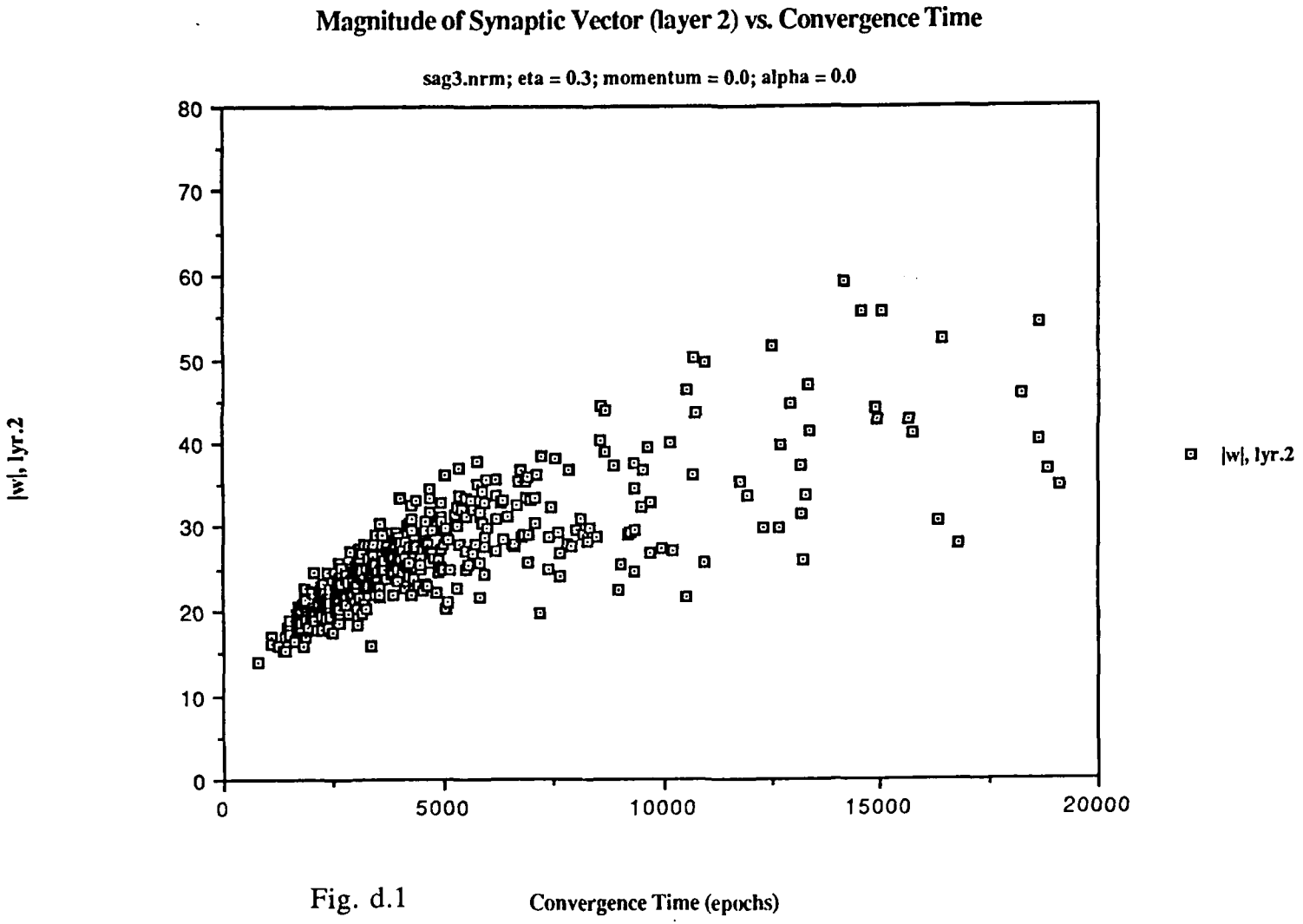
Magnitude of Synaptic Vector (layer 2) vs. Convergence Time
sag3.nrm; eta = 0.3; momentum = 0.0; alpha = 0.0
|w|, lyr.2
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
|w|, lyr.2
Convergence Time (epochs)

Fig. d.1

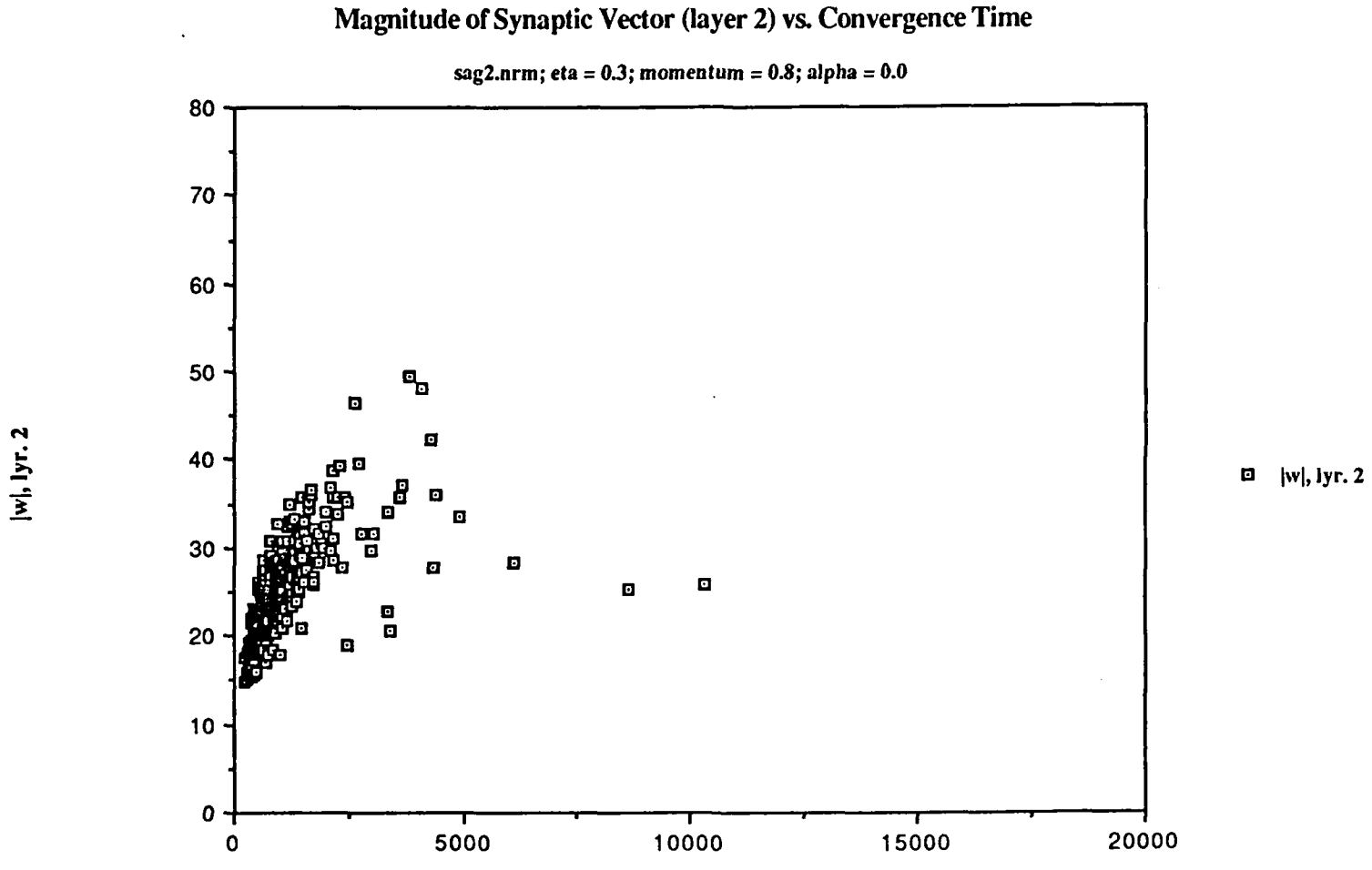
Magnitude of Synaptic Vector (layer 2) vs. Convergence Time
sag2.nrm; eta = 0.3; momentum = 0.8; alpha = 0.0
|w|, lyr. 2
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
|w|, lyr. 2
Convergence Time (epochs)

Fig. d.2

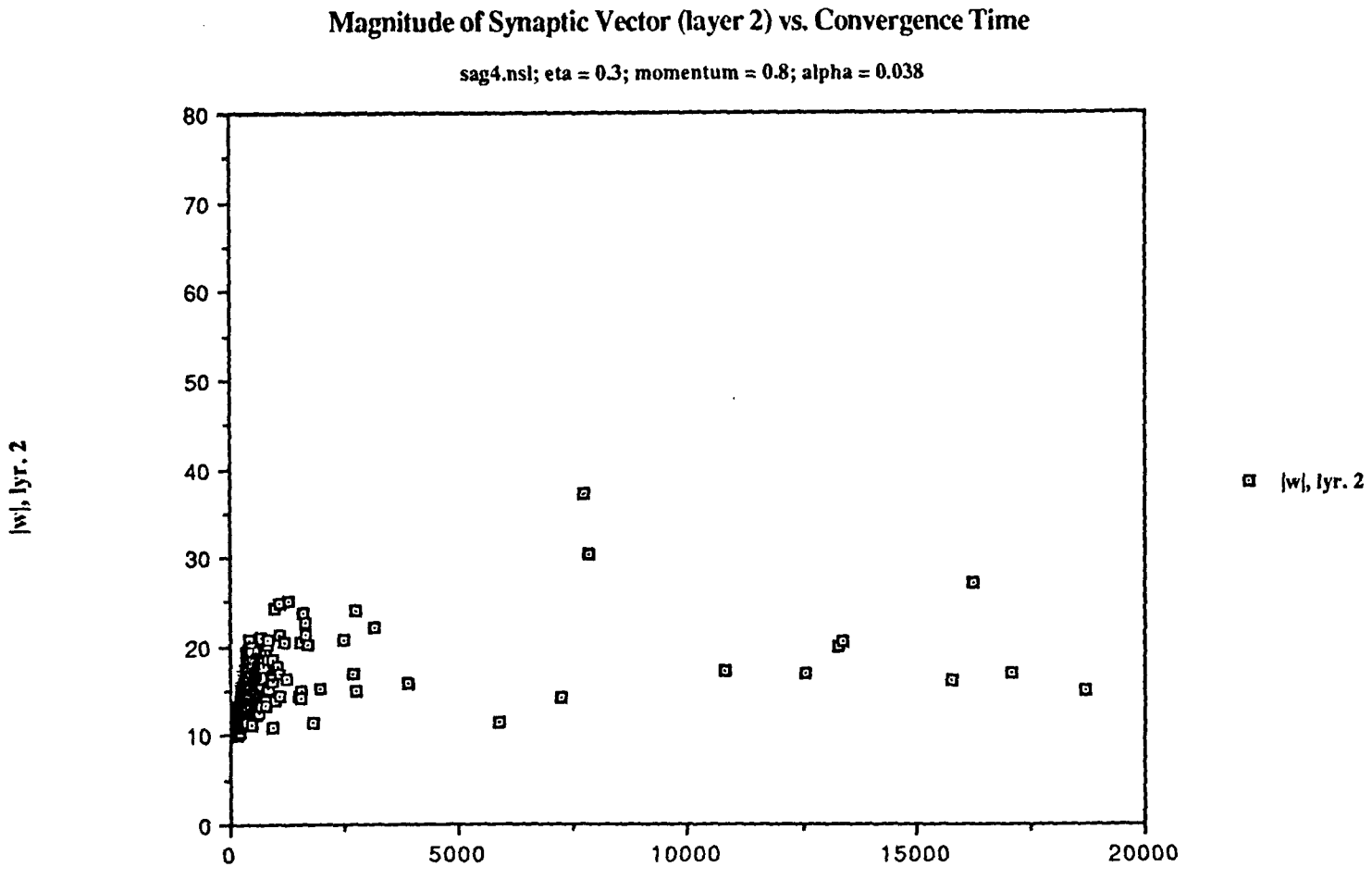
Magnitude of Synaptic Vector (layer 2) vs. Convergence Time
sag4.nsl; eta = 0.3; momentum = 0.8; alpha = 0.038
|w|, lyr. 2
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
|w|, lyr. 2

Fig. d.3

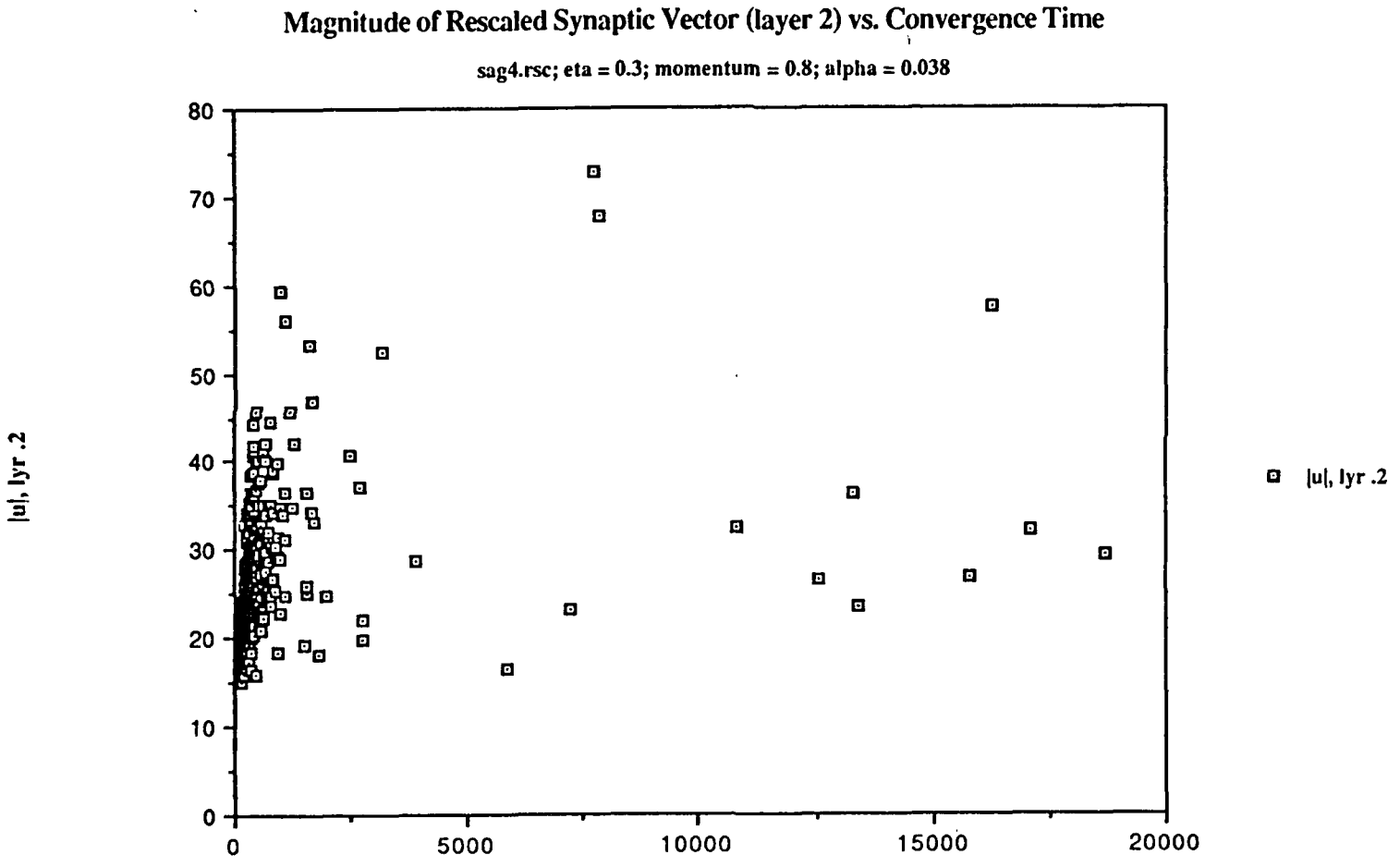
Magnitude of Rescaled Synaptic Vector (layer 2) vs. Convergence Time
sag4.rsc; eta = 0.3; momentum = 0.8; alpha = 0.038
|u|, lyr .2
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
|u|, lyr .2

Fig. d.4

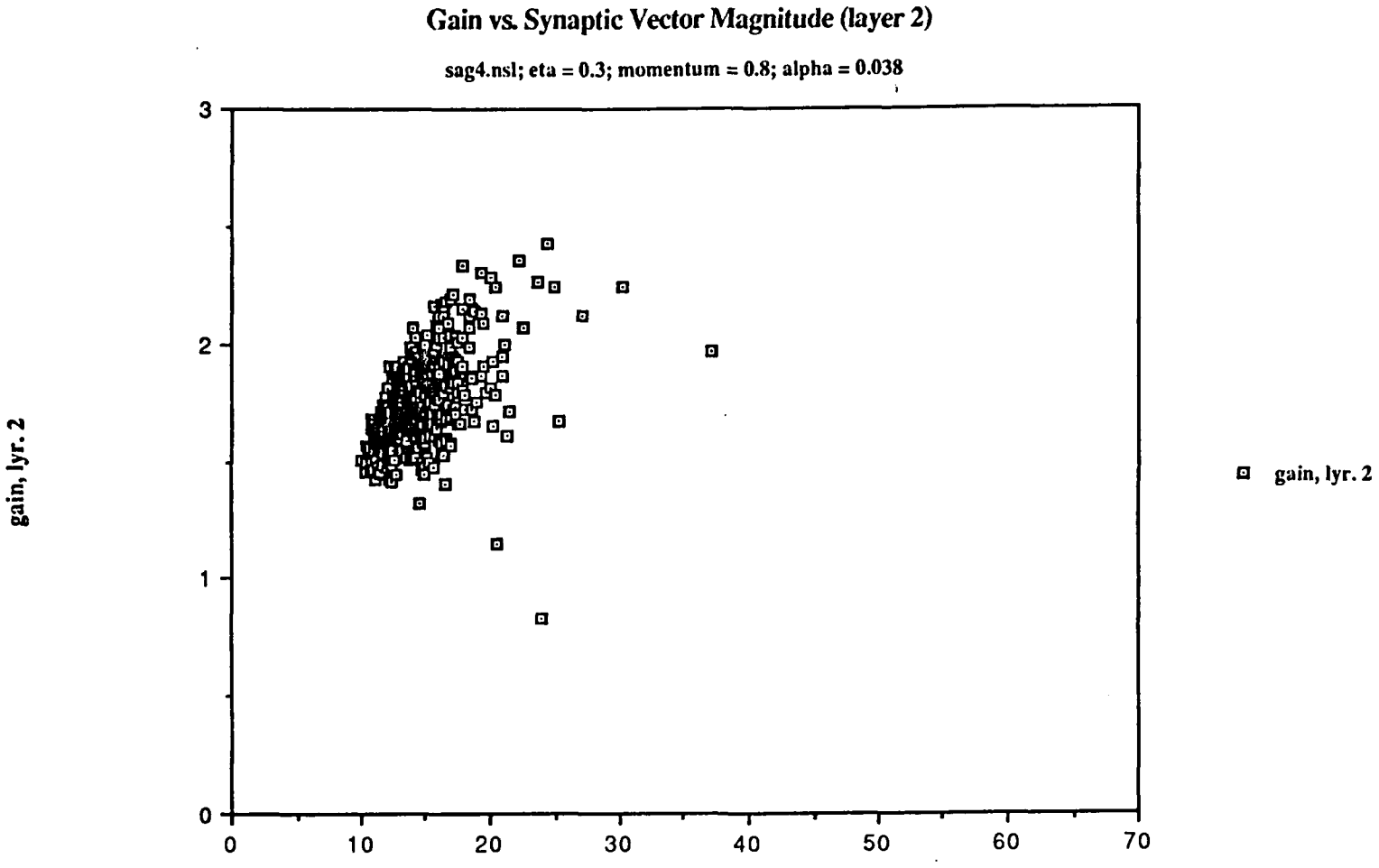
Gain vs. Synaptic Vector Magnitude (layer 2)
sag4.nsl; eta = 0.3; momentum = 0.8; alpha = 0.038
gain, lyr. 2
3
2
1
0
0
10
20
30
40
50
60
70
|w|, lyr. 2
gain, lyr. 2

Fig. d.5

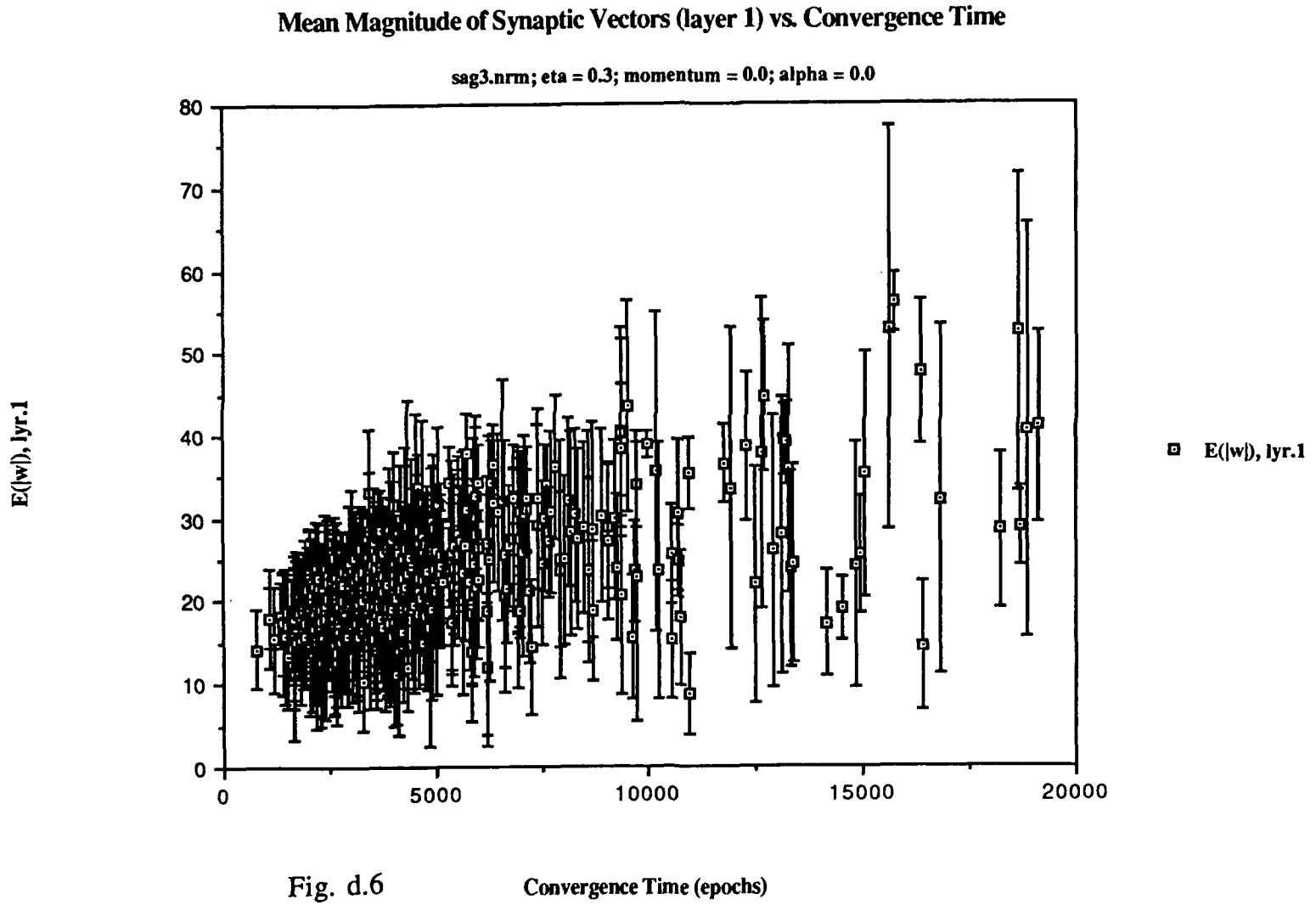
Mean Magnitude of Synaptic Vectors (layer 1) vs. Convergence Time
sag3.nrm; eta = 0.3; momentum = 0.0; alpha = 0.0
E(|w|), lyr.1
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
E(|w|), lyr.1

Fig. d.6

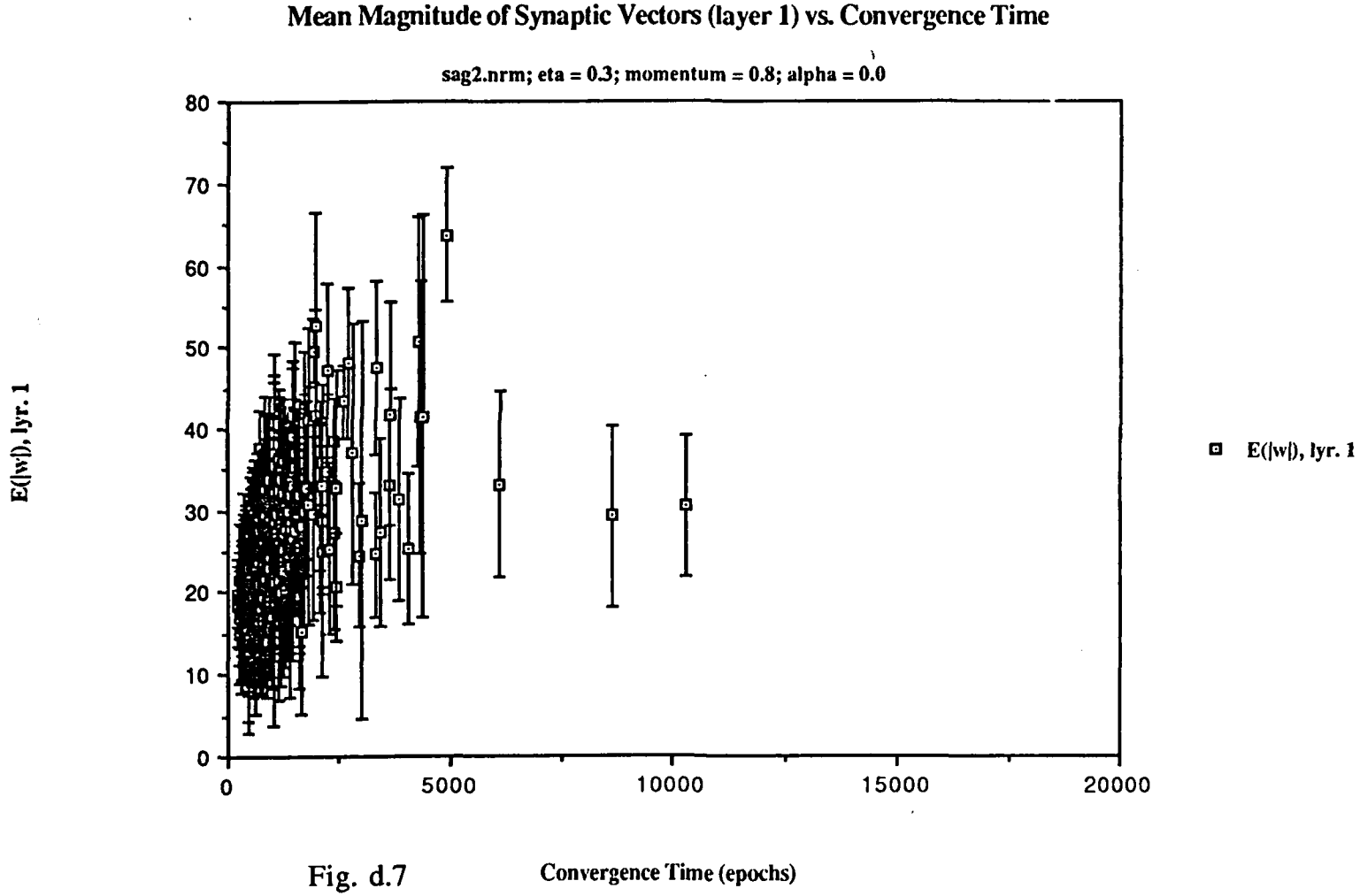
Mean Magnitude of Synaptic Vectors (layer 1) vs. Convergence Time
sag2.nrm; eta = 0.3; momentum = 0.8; alpha = 0.0
E(|w|), lyr. 1
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
E(|w|), lyr. 1

Fig. d.7

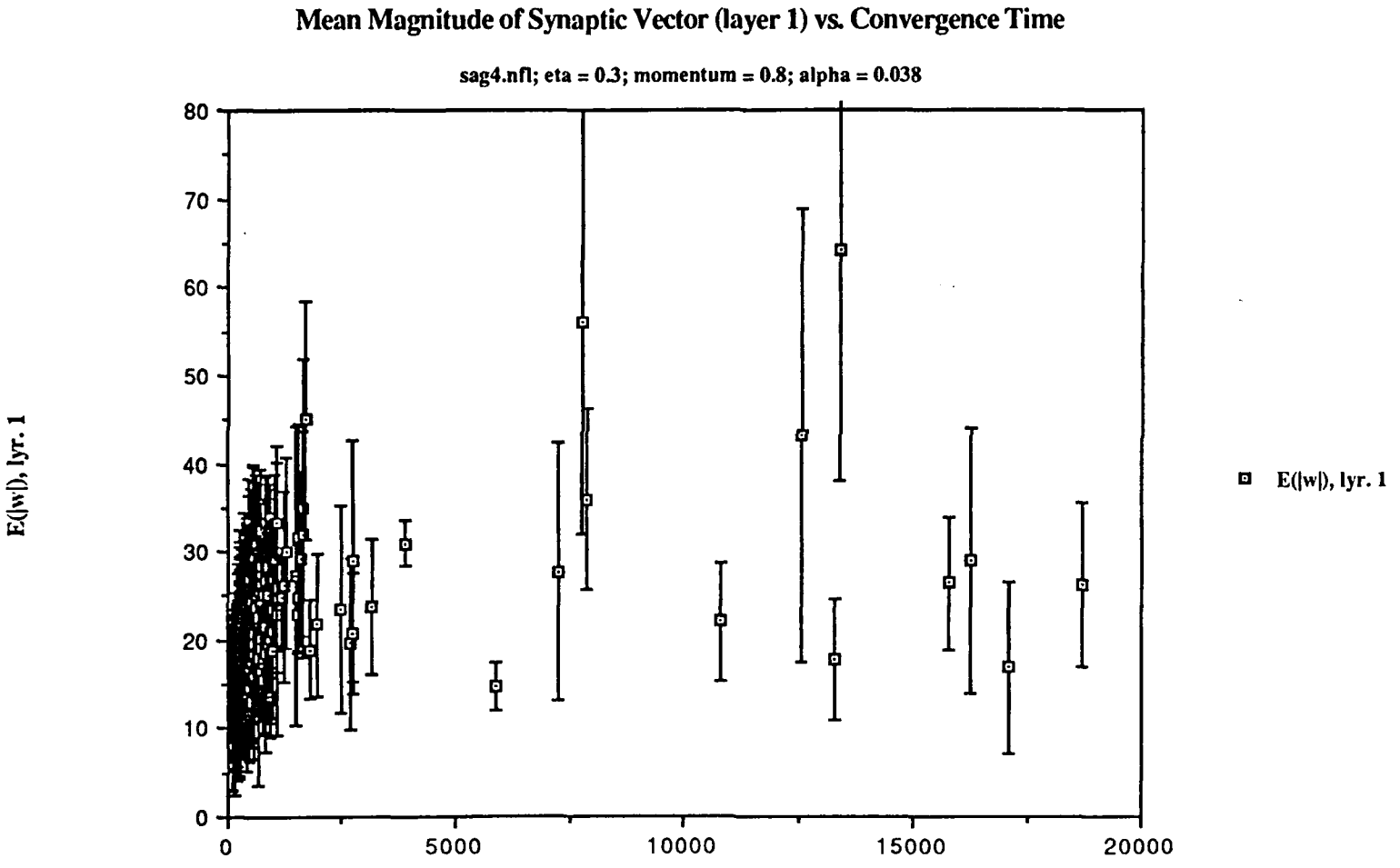
Mean Magnitude of Synaptic Vector (layer 1) vs. Convergence Time
sag4.nfl; eta = 0.3; momentum = 0.8; alpha = 0.038
E(|w|), lyr. 1
80
70
60
50
40
30
20
10
0
0
5000
10000
15000
20000
Convergence Time (epochs)
E(|w|), lyr. 1

Fig. d.8

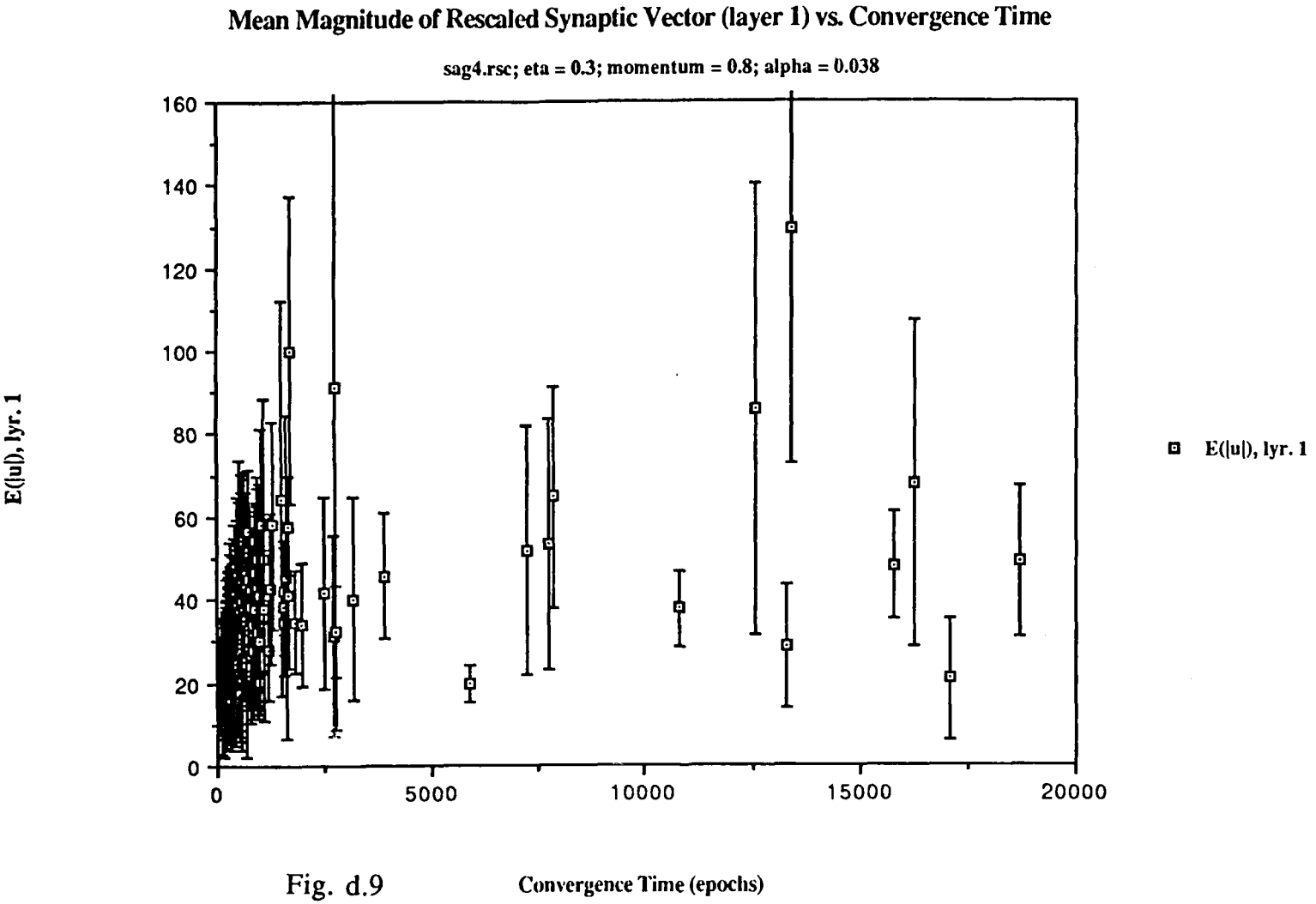
Mean Magnitude of Rescaled Synaptic Vector (layer 1) vs. Convergence Time
sag4.rsc; eta = 0.3; momentum = 0.8; alpha = 0.038
E(|u|), lyr. 1
160
140
120
100
80
60
40
20
0
0
5000
10000
15000
20000
Convergence Time (epochs)
E(|u|), lyr. 1

Fig. d.9

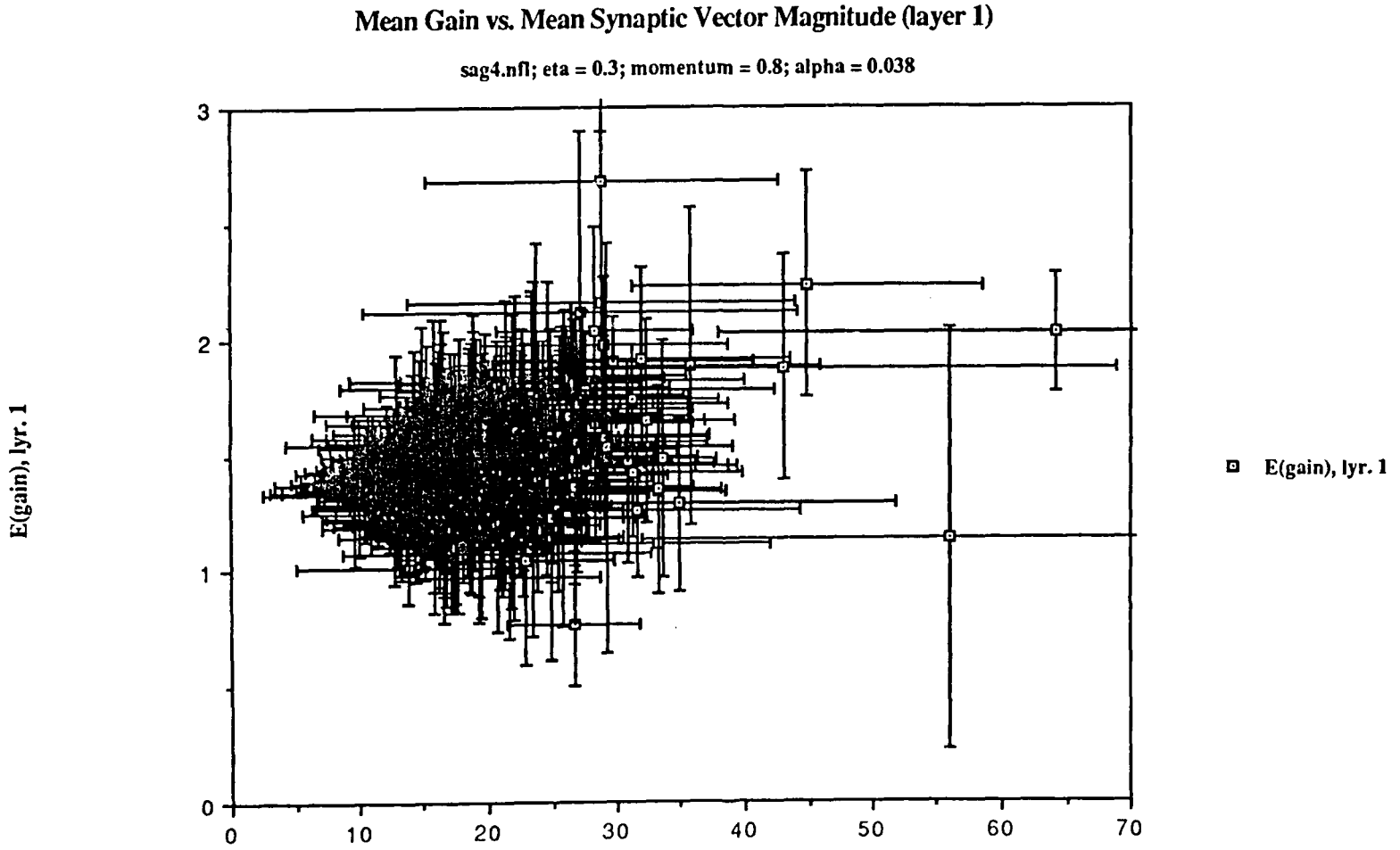
Mean Gain vs. Mean Synaptic Vector Magnitude (layer 1)
sag4.nfl; eta = 0.3; momentum = 0.8; alpha = 0.038
E(gain), lyr. 1
E(|w|), lyr. 1
0
1
2
3
0
10
20
30
40
50
60
70
E(gain), lyr. 1

Fig. d.10