Biologically Plausible Error-driven Learning using Local Activation Differences: The Generalized Recirculation Algorithm

Randall C. O'Reilly

Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
oreilly+@cmu.edu

July 1, 1996

Neural Computation, 8:5, 895-938, 1996

Abstract

The error backpropagation learning algorithm (*BP*) is generally considered biologically implausible because it does not use locally available, activation-based variables. A version of BP that can be computed locally using bi-directional activation *recirculation* (Hinton & McClelland, 1988) instead of backpropagated error derivatives is more biologically plausible. This paper presents a generalized version of the recirculation algorithm (*GeneRec*), which overcomes several limitations of the earlier algorithm by using a generic recurrent network with sigmoidal units that can learn arbitrary input/output mappings. However, the *contrastive-Hebbian learning* algorithm (*CHL*, a.k.a. *DBM* or mean field learning) also uses local variables to perform error-driven learning in a sigmoidal recurrent network. CHL was derived in a stochastic framework (the Boltzmann machine), but has been extended to the deterministic case in various ways, all of which rely on problematic approximations and assumptions, leading some to conclude that it is fundamentally flawed. This paper shows that CHL can be derived instead from within the BP framework via the GeneRec algorithm. CHL is a symmetry-preserving version of GeneRec which uses a simple approximation to the midpoint or second-order accurate Runge-Kutta method of numerical integration, which explains the generally faster learning speed of CHL compared to BP. Thus, all known fully general error-driven learning algorithms that use local activation-based variables in deterministic networks can be considered variations of the GeneRec algorithm (and indirectly, of the backpropagation algorithm). GeneRec therefore provides a promising framework for thinking about how the brain might perform error-driven learning. To further this goal, an explicit biological mechanism is proposed which would be capable of implementing GeneRec-style learning. This mechanism is consistent with available evidence regarding synaptic modification in neurons in the neocortex and hippocampus, and makes further predictions.

Introduction

A long-standing objection to the error backpropagation learning algorithm (*BP*) (Rumelhart, Hinton, & Williams, 1986a) is that it is biologically implausible (Crick, 1989; Zipser & Andersen, 1988), principally because it requires error propagation to occur through a mechanism different from activation propagation. This makes the learning appear non-local, since the error terms are not locally available as a result of the propagation of activation through the network. Several remedies for this problem have been suggested, but none are fully satisfactory. One approach involves the use of an additional "error-network" whose job is to send the error signals to the original network via an activation-based mechanism (Zipser & Rumelhart, 1990; Tesauro, 1990), but this merely replaces one kind of non-locality with another, activation-based kind of non-locality (and the problem of maintaining two sets of weights). Another approach uses a global reinforcement signal instead of specific error signals (Mazzoni, Andersen, & Jordan, 1991), but this is not as powerful as standard backpropagation.

The approach proposed by Hinton and McClelland (1988) is to use bi-directional activation *recirculation* within a single, recurrently connected network (with symmetric weights) to convey error signals. In order to get this to work, they used a somewhat unwieldy four-stage activation update process that only works for auto-encoder networks. This paper presents a generalized version of the recirculation algorithm (*GeneRec*), which overcomes the limitations of the earlier algorithm by using a generic recurrent network with sigmoidal units that can learn arbitrary input/output mappings. The GeneRec algorithm shows how a general form of error backpropagation, which computes essentially the same error derivatives as the Almeida-Pineda (*AP*) algorithm (Almeida, 1987; Pineda, 1987b, 1987a, 1988) for recurrent networks under certain conditions, can be performed in a biologically plausible fashion using only locally available activation variables.

GeneRec uses recurrent activation flow to communicate error signals from the output layer to the hidden layer(s) via symmetric weights. This weight symmetry is an important condition for computing the correct error derivatives. However, the "catch-22" is that GeneRec does not itself preserve the symmetry of the weights, and when it is modified so that it does, it no longer follows the same learning trajectory as AP, even though it is computing essentially the same error gradient. The empirical relationship between the derivatives computed by GeneRec and AP backpropagation is explored in simulations reported in this paper.

The GeneRec algorithm has much in common with the *contrastive Hebbian learning* algorithm (*CHL*, a.k.a. the mean field or deterministic Boltzmann machine (DBM) learning algorithm), which also uses locally available activation variables to perform error-driven learning in recurrently connected networks. This algorithm was derived originally for stochastic networks whose activation states can be described by the Boltzmann distribution (Ackley, Hinton, & Sejnowski, 1985). In this context, CHL amounts to reducing the distance between two probability distributions that arise in two phases of settling in the network. This algorithm has been extended to the deterministic case through the use of approximations or restricted cases of the probabilistic one (Hinton, 1989b; Peterson & Anderson, 1987), and derived without the use of the Boltzmann distribution by using the continuous Hopfield energy function (Movellan, 1990). However, all of these derivations require problematic assumptions or approximations, leading some to conclude that CHL is fundamentally flawed for deterministic networks (Galland, 1993; Galland & Hinton, 1990).

It is shown in this paper that the CHL algorithm can be derived instead as a variant of the GeneRec algorithm, which establishes a more general formal relationship between the BP framework and the deterministic CHL rule than previous attempts (Peterson, 1991; Movellan, 1990; LeCun & Denker, 1991). This relationship means that all known fully general error-driven learning algorithms that use local activation-based variables in deterministic networks can be considered variations of the GeneRec algorithm (and thus, indirectly, of the backpropagation algorithm as well). An important feature of the GeneRec-based derivation of CHL is that the relationship between the learning properties of BP, GeneRec, and CHL can be more clearly understood. Another feature of this derivation is that it is completely general with respect to the activation function used, allowing CHL-like learning rules to be derived for many different cases.

CHL is equivalent to GeneRec when using a simple approximation to a second-order accurate numerical integration technique known as the *midpoint* or second-order Runge-Kutta method, plus an additional symmetry preservation constraint. The implementation of the midpoint method in GeneRec simply amounts to the incorporation of the sending unit's plus-phase activation state into the error derivative, and as such, it amounts to an on-line (per pattern) integration technique. This method results in faster learning by reducing the amount of interference due to independently computed weight changes for a given pattern. This would explain why CHL networks generally learn faster than otherwise equivalent BP networks (e.g., Peterson & Hartman, 1989; Movellan, 1990). A comparison of optimal learning speeds for all variants of GeneRec and feedforward and AP recurrent backprop on four different problems is reported in this paper. The results of this comparison are consistent with the derived relationship between GeneRec and AP backpropagation, and with the interpretation of CHL as a symmetric, midpoint version of GeneRec, and thus provide empirical support for these theoretical claims.

The finding that CHL did not perform well at all in networks with multiple hidden layers ("deep" networks), reported by Galland (1993), would appear to be problematic for the claim that CHL is performing a fully general form of backpropagation, which can learn in deep networks. However, I was unable to replicate the Galland (1993) failure to learn the "family trees" problem (Hinton, 1986) using CHL. In simulations reported below, I show that by simply increasing the number of hidden units (from 12 to 18), CHL networks can learn the problem with 100% success rate, in a number of epochs on the same order as backpropagation. Thus, the existing simulation evidence seems to support the idea that CHL is performing a form of backpropagation, and not that it is a fundamentally flawed approximation to the Boltzmann machine as has been argued (Galland, 1993; Galland & Hinton, 1990).

Given that the GeneRec family of algorithms encompasses all known ways of performing error-driven learning using locally-available activation variables, it provides a promising framework for thinking about how error-driven learning might be implemented in the brain. To further this goal, an explicit biological mechanism capable of implementing GeneRec-style learning is proposed. This mechanism is consistent with available evidence regarding synaptic modification in neurons in the neocortex and hippocampus, and makes further predictions.

Introduction to Algorithms and Notation

In addition to the original recirculation algorithm (Hinton & McClelland, 1988), the derivation of GeneRec depends on ideas from several standard learning algorithms, including: feedforward error backpropagation (BP) (Rumelhart et al., 1986a) with the cross-entropy error term (Hinton, 1989a); the Almeida-Pineda (AP) algorithm for error backpropagation in a recurrent network (Almeida, 1987; Pineda, 1987b, 1987a, 1988); and the contrastive Hebbian learning algorithm (CHL) used in the Boltzmann machine and deterministic variants (Ackley et al., 1985; Hinton, 1989b; Peterson & Anderson, 1987). The notation and equations for these algorithms are summarized in this section, followed by a brief overview of the recirculation algorithm in the next section. This provides the basis for the development of the generalized recirculation algorithm presented in subsequent sections.

Feedforward Error Backpropagation

The notation for a three-layer feedforward backprop network uses the symbols shown in table 1. The target values are labeled t_k for output unit k, and the pattern-wise sum is dropped since the subsequent derivations do not depend on it. The cross-entropy error formulation (Hinton, 1989a) is used because it eliminates an activation derivative term in the learning rule which is also not present in the recirculation algorithm.

Layer (index)	Net Input	Activation
Input (s)	—	s_i = stimulus input
Hidden (h)	$\eta_j = \sum_i w_{ij} s_i$	$h_j = \sigma(\eta_j)$
Output (o)	$\eta_k = \sum_j w_{jk} h_j$	$o_k = \sigma(\eta_k)$

Table 1: Variables in a 3 layer backprop network. $\sigma(\eta)$ is the standard sigmoidal activation function $\sigma(\eta) = 1/(1 + e^{-\eta})$.

The cross-entropy error is defined as:

$$E = \sum_k t_k \log\ o_k + (1 - t_k) \log(1 - o_k) \tag{1}$$

and the derivative of E with respect to a weight into an output unit is:

$$\begin{aligned} \frac{\partial E}{\partial w_{jk}} &= \frac{dE}{do_k} \frac{do_k}{d\eta_k} \frac{\partial \eta_k}{\partial w_{jk}} \\ &= \left[\frac{t_k}{o_k} - \frac{(1 - t_k)}{(1 - o_k)} \right] \sigma'(\eta_k) h_j \\ &= (t_k - o_k) h_j \end{aligned} \tag{2}$$

where $\sigma'(\eta_k)$ is the first derivative of the sigmoidal activation function with respect to the net input: $o_k(1 - o_k)$, which is canceled by the denominator of the error term. In order to train the weights into the hidden units, the impact a given hidden unit has on the error term needs to be determined:

$$\begin{aligned} \frac{\partial E}{\partial h_j} &= \sum_k \frac{dE}{do_k} \frac{do_k}{d\eta_k} \frac{\partial \eta_k}{\partial h_j} \\ &= -\sum_k (t_k - o_k) w_{jk} \end{aligned} \tag{3}$$

which can then be used to take the derivative of the error function with respect to the input to hidden unit weights:

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial h_j} \frac{dh_j}{d\eta_j} \frac{\partial \eta_j}{\partial w_{ij}} \\ &= -\sum_k (t_k - o_k) w_{jk} \sigma'(\eta_j) s_i \end{aligned} \tag{4}$$

which provides the basis for adapting the weights.

Almeida-Pineda Recurrent Backpropagation

The AP version of backpropagation is essentially the same as the feedforward one described above except that it allows for the network to have recurrent (bidirectional) connectivity. Thus, the network is trained to settle into a stable activation state with the output units in the target state, based on a given input pattern clamped over the input units. This is the version of BP that the GeneRec algorithm, which also uses recurrent connectivity, approximates most closely. The same basic notation and cross-entropy error term as described for feedforward BP are used to describe the AP algorithm, except that the net input terms (η) can now include input from any other unit in the network, not only those in lower layers.

Layer	Phase	Net Input	Activation
Input (s)	—	—	s_i = stimulus input
Hidden (h)	-	$\eta_j^- = \sum_i w_{ij}s_i + \sum_k w_{kj}o_k^-$	$h_j^- = \sigma(\eta_j^-)$
	+	$\eta_j^+ = \sum_i w_{ij}s_i + \sum_k w_{kj}o_k^+$	$h_j^+ = \sigma(\eta_j^+)$
Output (o)	-	$\eta_k^- = \sum_j w_{jk}h_j^-$	$o_k^- = \sigma(\eta_k^-)$
	+	—	$o_k^+ = t_k$

Table 2: Equilibrium network variables in a three layer network having reciprocal connectivity between the hidden and output layers with symmetric weights ($w_{jk} = w_{kj}$), and phases over the output units such that the target is clamped in the plus phase, and not in the minus phase. $\sigma(\eta)$ is the standard sigmoidal activation function.

The activation states in AP are updated according to a discrete-time approximation of the following differential equation, which is integrated over time with respect to the net input terms[1]:

$$\frac{d\eta_j}{dt} = -\eta_j + \sum_i w_{ij}\sigma(\eta_i) \tag{5}$$

This equation can be iteratively applied until the network settles into a stable equilibrium state (i.e., until the change in activation state goes below a small threshold value), which it will provably do if the weights are symmetric (Hopfield, 1984), and often even if they are not (Galland & Hinton, 1991).

In the same way that the activations are iteratively updated to allow for recurrent activation dynamics, the error propagation in the AP algorithm is also performed iteratively. The iterative error propagation in AP operates on a new variable y_j which represents the current estimate of the derivative of the error with respect to the net input to the unit, $\frac{\partial E}{\partial \eta_j}$. This variable is updated according to:

$$\frac{dy_j}{dt} = -y_j + \sigma'(\eta_j)\sum_k w_{kj}y_k + J_j \tag{6}$$

where J_j is the externally "injected" error for output units with target activations. This equation is iteratively applied until the y_j variables settle into an equilibrium state (i.e., until the change in y_j falls below a small threshold value). The weights are then adjusted as in feedforward BP (4), with y_j providing the $\frac{\partial E}{\partial \eta_j}$ term.

Contrastive Hebbian Learning

The contrastive-Hebbian learning algorithm (CHL) used in the stochastic Boltzmann machine and deterministic variants is based on the differences between activation states in two different phases. As in the AP algorithm, the connectivity is recurrent, and activation states (in the deterministic version) can be computed according to (5). As will be discussed below, the use of locally-computable activation differences instead of the non-local error backpropagation used in the BP and AP algorithms is more biologically plausible. The GeneRec algorithm, from which the CHL algorithm can be derived as a special case, uses the same notion of activation phases.

The two phases of activation states used in CHL are the *plus* phase states, which result from both input and target being presented to the network, and provide a training signal when compared to the *minus* phase activations, which result from just the input pattern being presented. The equilibrium network variables (i.e., the states after the iterative updating procedure of (5) has been applied) for each phase in such a system are labeled as in table 2.

[1]It is also possible to incrementally update the activations instead of the net inputs, but this limits the ability of units to change their state rapidly, since the largest activation value is 1, while net inputs are not bounded.

The CHL learning rule for deterministic recurrent networks can be expressed in terms of generic activation states a (which can be from any layer in the network) as follows:

$$\frac{1}{\epsilon}\Delta w_{ij} = (a_i^+ a_j^+) - (a_i^- a_j^-) \tag{7}$$

where a_i is the sending unit and a_j is the receiving unit. Thus, CHL is simply the difference between the pre and post-synaptic activation coproducts in the plus and minus phases. Each coproduct term is equivalent to the derivative of the energy or "harmony" function for the network with respect to the weight:

$$E = \sum_j \sum_i a_j a_i w_{ij} \tag{8}$$

and the intuitive interpretation of this rule is that it decreases the energy or increases the harmony of the plus-phase state and vice-versa for the minus-phase state (see Ackley et al., 1985; Peterson & Anderson, 1987; Hinton, 1989b; Movellan, 1990, for details).

The Recirculation Algorithm

The original Hinton and McClelland (1988) recirculation algorithm is based on the feedforward BP algorithm. The GeneRec algorithm, which is more closely related to the AP recurrent version of backpropagation, borrows two key insights from the recirculation algorithm. These insights provide a means of overcoming the main problem with the standard backprop formulation from a neural plausibility standpoint, which is the manner in which a hidden unit computes its own error contribution. This is shown in (3). The problem is that the hidden unit is required to access the computed quantity ($\frac{\partial E}{\partial o_k}$), which depends on variables at the output unit only. This is the crux of the non-locality of error information in backpropagation.

The first key insight that can be extracted from the recirculation algorithm is that (3) can be expressed as the difference between two terms, each of which look much like a net-input to the hidden unit:

$$\frac{\partial E}{\partial h_j} = -\left(\sum_k t_k w_{jk} - \sum_k o_k w_{jk}\right) \tag{9}$$

Thus, instead of having a separate error-backpropagation phase to communicate error signals, one can think in terms of standard activation propagation occurring via reciprocal (and symmetric) weights that come from the output units to the hidden units. The error contribution for the hidden unit can then be expressed in terms of the difference between two net-input terms. One net-input term is just that which would be received when the output units had the target activations t_k clamped on them, and the other is that which would be received when the outputs have their feed-forward activation values o_k.

In order to take advantage of a net-input based error signal, Hinton and McClelland (1988) used an auto-encoder framework, with two pools of units: *visible* and *hidden*. The visible units play the role of both the input layer and the output layer. Each training pattern, which is its own target, is presented to the visible units, which then project to a set of hidden units, which then feed back to the same visible units. The input from the hidden units changes the state of the visible units, and this new state is then fed through the system again, hence the name *recirculation* (see figure 1). As a result, the visible units have two activation states, equivalent to t_k (or s_i) and o_k, and the hidden units have two activation states, the first of which is a function of $\eta_j^* = \sum_k t_k w_{kj}$, which can be labeled h_j^*, and the second of which corresponds to h_j.

The second key insight from the recirculation algorithm is that instead of computing a difference in net-inputs in (9), one can use a difference of activations via the following approximation to (4):

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= -(\eta_j^* - \eta_j)\sigma'(\eta_j)t_k \\ &\approx -(h_j^* - h_j)t_k \end{aligned} \tag{10}$$

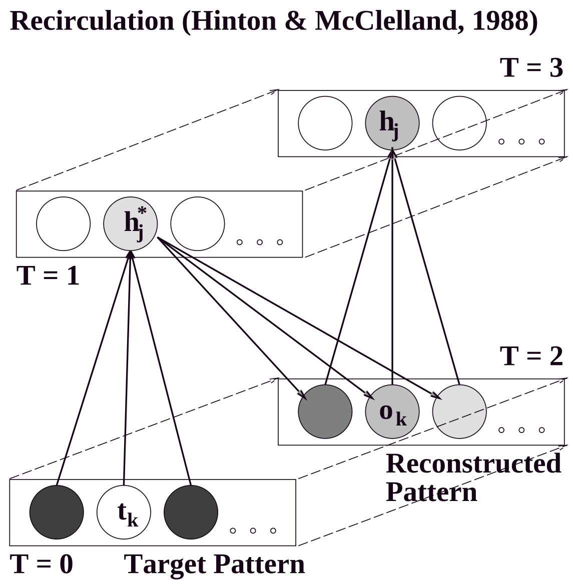

Figure 1: The recirculation algorithm, as proposed by Hinton and McClelland (1988). Activation is propagated in four steps ($T = 0 - 3$). $T = 0$ is the target pattern clamped on the visible units. $T = 1$ is the hidden units computing their activation as a function of the target inputs. $T = 2$ is the visible units computing their activations as a function of the hidden unit state at $T = 1$. $T = 3$ is the hidden unit state computed as a function of the reconstructed visible unit pattern.

The difference between activation values instead of the net-inputs in (10) can be used since Hinton and McClelland (1988) imposed an additional constraint that the difference between the reconstructed and the target visible-unit states (and therefore the difference between η_j^* and η_j) be kept small by using a "regression" function in updating the visible units. This function assigns output state (computed at time $T = 2$) as a weighted average of the target output and the activation computed from the current net input from the hidden units[2]:

$$o_k = \lambda t_k + (1 - \lambda) f(\eta_k) \tag{11}$$

Thus, the difference in a hidden-unit's activation values is approximately equivalent to the difference in net-inputs times the slope of the activation function at one of the net-input values ($\sigma'(\eta_j)$), as long as the linear approximation of the activation function given by the slope is reasonably valid. Even if this is not the case, as long as the activation function is monotonic, the error in this approximation will not affect the sign of the resulting error derivative, only the magnitude. Nevertheless, errors in magnitude can lead to errors in sign over the pattern-wise sum.

Since the difference of activations in (10) computes the derivative of the activation function implicitly, one can use the resulting learning rule for any reasonable (monotonic) activation function[3]. This can be important for cases where the derivative of the activation function is difficult to compute. Further, the activation variable might be easier to map onto the biological neuron, and it avoids the need for the neuron to compute its activation derivative.

Note that due to the above simplification, the learning rule for the recirculation algorithm (based on (10)) is the same for both hidden and output units, and is essentially the delta-rule. This means that locally available activation states of the pre and postsynaptic units can be used to perform error-driven learning, which avoids

[2] Note that Hinton and McClelland (1988) used linear output units to avoid the activation function derivative on the output units, whereas cross-entropy is being used here to avoid this derivative. Thus, the function $f(\eta_k)$ in (11) will either be linear or a sigmoid, depending on which assumptions are being used.

[3] Note that the output units have to use a sigmoidal function in order for the cross-entropy function to cancel out the derivative.

the need for a biologically troublesome error backpropagation mechanism that is different from the normal propagation of activation through the network.

Phase-Based Learning and Generalized Recirculation

While the recirculation algorithm is more biologically plausible than standard feedforward error backpropagation, it is limited to learning auto-encoder problems. Further, the recirculation activation propagation sequence requires a detailed level of control over the flow of activation through the network and its interaction with learning. However, the critical insights about computing error signals using differences in net input (or activation) terms can be applied to the more general case of a standard three-layer network for learning input to output mappings. This section presents such a *generalized recirculation* or *GeneRec* algorithm, which uses standard recurrent activation dynamics (as used in the AP and CHL algorithms) to communicate error signals instead of the recirculation technique. Instead of using the four stages of activations used in recirculation, GeneRec uses two activation phases as in the CHL algorithm described above. Thus, in terms of activation states, GeneRec is identical to the deterministic CHL algorithm, and the same notation is used to describe it.

The learning rule for GeneRec is simply the application of the two key insights from the recirculation algorithm to the AP recurrent backpropagation algorithm instead of the feedforward BP algorithm, which was the basis of the recirculation algorithm. If the recurrent connections between the hidden and output units are ignored so that the error on the output layer is held constant, it is easy to show that the fixed point solution to the iterative AP error updating equation (6) (i.e., where $\frac{dy_j}{dt} = 0$ for hidden unit error y_j) is of the same form as feedforward backpropagation. This means that the same recirculation trick of computing the error signal as a difference of net input (or activation) terms can be used:

$$\begin{aligned} y_j^{\infty} &= \sigma'(\eta_j) \sum_k w_{kj} y_k^{\infty} \\ &\approx \sigma'(\eta_j) \left(\sum_k w_{kj}(t_k - o_k) \right) \\ &\approx \sigma'(\eta_j) \left(\sum_k w_{kj} t_k - \sum_k w_{kj} o_k \right) \end{aligned} \qquad (12)$$

Thus, assuming constant error on the output layer, the equilibrium error gradient computed for hidden units in AP is equivalent to the difference between the GeneRec equilibrium net input states in the plus and minus phases. Note that the minus phase activations in GeneRec are identical to the AP activation states. The difference of activation states can be substituted for net input differences times the derivative of the activation function by the approximation introduced in recirculation, resulting in the following equilibrium unit error gradient:

$$y_j^{\infty} \approx h_j^{+} - h_j^{-} \qquad (13)$$

Note that while the hidden unit states in GeneRec also reflect the constant net input from the input layer (in addition to the output-layer activations that communicate the necessary error gradient information), this cancels out in the difference computation of (13). However, this constant input to the hidden units from the input layer in both phases can play the role of the regression update equation (11) in recirculation. To the extent that this input is reasonably large and it biases the hidden units towards one end of the sigmoid or the other, this bias will tend to moderate the differences between h_j^{-} and h_j^{+}, making their difference a reasonable approximation to the differences of their respective net inputs times the slope of the activation function.

While the analysis presented above is useful for seeing how GeneRec equilibrium activation states could approximate the equilibrium error gradient computed by AP, the AP algorithm actually performs iterative updating of the error variable (y_j). Thus, it would have to be the case that iterative updating of this single variable is equivalent to the iterative updating of each of the activation states (plus and minus) in GeneRec, and then taking the difference. This relationship can be expressed by writing GeneRec in the AP notation. First, we define two components to the error variable y_j, which are effectively the same as the GeneRec plus and minus phase net inputs (ignoring the net input from the input units, which is subtracted out in the end anyway):

$$\frac{dy_j^+}{dt} = -y_j^+ + \sum_k w_{kj} t_k \tag{14}$$

$$\frac{dy_j^-}{dt} = -y_j^- + \sum_k w_{kj} \sigma(\eta_k) \tag{15}$$

Then, we approximate the fixed point of y_j with the difference of the fixed points of these two variables:

$$\begin{aligned} y_j^\infty &\approx \sigma'(\eta_j)(y_j^{\infty +} - y_j^{\infty -}) \\ &\approx h_j^+ - h_j^- \end{aligned} \tag{16}$$

which can be approximated by the subtraction of the GeneRec equilibrium activation states as discussed previously.

Unfortunately, the validity of this approximation is not guaranteed by any proof that this author is aware of. However, there are several properties of the equations that lend some credibility to it. First, the part that is a function of t_k on the output units, y_j^+, is effectively a constant, and the other part, y_j^-, is just the activation updating procedure that both GeneRec and AP have in common. Further, given that the fixed point solutions of the GeneRec and AP equations are the same when recurrent influences are ignored, and the pattern of recurrent influences is given by the same set of weights, it is likely that the additional effects of recurrence will be in the same direction for both GeneRec and AP. However, short of a proof, these arguments require substantiation from simulation results comparing the differences between the error derivatives computed by GeneRec and AP. The results presented later in the paper confirm that GeneRec computes essentially the same error derivatives as AP in a three layer network (as long as the weights are symmetric). This approximation deteriorates only slightly in networks with multiple hidden layers, where the effects of recurrence are considerably greater.

As in the recirculation algorithm, it is important for the above approximation that the weights into the hidden units from the output units have the same values as the corresponding weights that the output units receive from the hidden units. This is the familiar symmetric weight constraint, which is also necessary to prove that a network will settle into a stable equilibrium (Hopfield, 1984). We will revisit this constraint several times during the paper. However, for the time being, we will assume that the weights are symmetric.

Finally, virtually all deterministic recurrent networks including GeneRec suffer from the problem that changes in weights based on gradient information computed on equilibrium activations might not result in the network settling into an activation state with lower error the next time around. This is due to the fact that small weight changes can affect the settling trajectory in unpredictable ways, resulting in an entirely different equilibrium activation state than the one settled into last time. While it is important to keep in mind the possibility of discontinuities in the progression of activation states over learning, there is some basis for optimism on this issue. For example, in his justification of the deterministic version of the Boltzmann machine (DBM) Hinton (1989b) supplies several arguments (which are substantiated by a number of empirical findings) justifying the assumption that small weight changes will generally lead to a contiguous equilibrium state of unit activities in a recurrent network.

To summarize, the learning rule for GeneRec that computes the error backpropagation gradient locally via recurrent activation propagation is the same as that for recirculation, having the form of the delta-rule. It can be stated as follows in terms of a sending unit with activation a_i and a receiving unit with activation a_j:

$$\frac{1}{\epsilon}\Delta w_{ij} = a_i^-(a_j^+ - a_j^-) \tag{17}$$

As shown above, this learning rule will compute the same error derivatives as the AP recurrent backpropagation procedure under the following conditions:

- Iterative updating of the error term (y_j) can be approximated by the separate iterative updating of the two activation terms (h_j^+ and h_j^-) and then taking their difference.
- The reciprocal weights are symmetric ($w_{jk} = w_{kj}$).
- Differences in net inputs times the activation function derivative can be approximated by differences in the activation values.

The Relationship Between GeneRec and CHL

The GeneRec learning rule given by (17) and the CHL learning rule given by (7) are both simple expressions that involve a difference between plus and minus phase activations. This raises the possibility that they could somehow be related to each other. Indeed, as described below, there are two different ways in which GeneRec can be modified that, when combined, yield (7).

The GeneRec learning rule can be divided into two parts, one of which represents the derivative of the error with respect to the unit (the difference of that unit's plus and minus activations, $a_j^+ - a_j^-$), and the other which represents the contribution of a particular weight to this error term (the sending unit's activation, a_i^-). It is the phase of this latter term which is the source of the first modification. In standard feedforward or AP recurrent backpropagation, there is only one activation term associated with each unit, which is equivalent to the minus-phase activation in the GeneRec phase-based framework. Thus, the contribution of the sending unit is naturally evaluated with respect to this activation, and that is why a_i^- appears in the GeneRec learning rule.

However, given that GeneRec has another activation term corresponding to the plus-phase state of the units, one might wonder if the derivative of the weight should be evaluated with respect to this activation instead. After all, the plus phase activation value will likely be a more accurate reflection of the eventual contribution of a given unit after other weights in the network are updated. In some sense, this value *anticipates* the weight changes that will lead to having the correct target values activated, and learning based on it might avoid some interference.

On the other hand, the minus phase activation reflects the *actual* contribution of the sending unit to the current error signal, and it seems reasonable that credit assignment should be based on it. Given that there are arguments in favor of both phases, one approach would be to simply use the average of both of them. Doing this results in the following weight update rule:

$$\frac{1}{\epsilon}\Delta w_{ij} = \frac{1}{2}(a_i^- + a_i^+)(a_j^+ - a_j^-) \tag{18}$$

This is the first way in which GeneRec needs to be modified to make it equivalent to CHL. As will be discussed in detail below, this modification of GeneRec corresponds to a simple approximation of the *midpoint* or second-order accurate Runge-Kutta integration technique. The consequences of the midpoint method for learning speed will be explored in the simulations reported below.

The second way in which GeneRec needs to be modified concerns the issue of symmetric weights. In order for GeneRec to compute the error gradient via reciprocal weights, these weights need to have the same value (or at least the same relative magnitudes and signs) as the forward-going weights. However, the basic GeneRec learning rule (17) does not preserve this symmetry:

$$a_i^-(a_j^+ - a_j^-) \neq a_j^-(a_i^+ - a_i^-) \tag{19}$$

While simulations reported below indicate that GeneRec can learn and settle into stable attractors without explicitly preserving the weight symmetry, a symmetry-preserving version of GeneRec would guarantee that the computed error derivatives are always correct.

One straightforward way of ensuring weight symmetry is simply to use the average (or more simply, the sum) of the weight changes that would have been computed for each of the reciprocal weights separately, and apply this same change to both weights. Thus, the symmetric GeneRec learning rule is:

$$\begin{aligned} \frac{1}{\epsilon}\Delta w_{ij} &= a_i^-(a_j^+ - a_j^-) + a_j^-(a_i^+ - a_i^-) \\ &= (a_j^+ a_i^- + a_j^- a_i^+) - 2a_j^- a_i^- \end{aligned} \tag{20}$$

Note that using this rule will not result in the weights being updated in the same way as AP backpropagation, even though the error derivatives computed on the hidden units will still be the same. Thus, even the symmetry preserving version of GeneRec is not identical to AP backpropagation. This issue will be explored in the simulations reported below.

If both the midpoint method and symmetry preservation versions of GeneRec are combined, the result is the CHL algorithm:

$$\begin{aligned} \frac{1}{\epsilon}\Delta w_{ij} &= \frac{1}{2}\left[(a_i^+ + a_i^-)(a_j^+ - a_j^-) + (a_j^+ + a_j^-)(a_i^+ - a_i^-)\right] \\ &= (a_i^+ a_j^+) - (a_i^- a_j^-) \end{aligned} \tag{21}$$

Note that LeCun and Denker (1991) pointed out that CHL is related to a symmetric version of the delta rule (i.e., GeneRec), but they did not incorporate the midpoint method, and thus were only able to show an approximation that ignored this aspect of the relationship between CHL and GeneRec.

The above derivation of CHL is interesting for several reasons. First, it is based on error backpropagation (via GeneRec), and not some kind of approximation to a stochastic system. This eliminates the problems associated with considering the graded activations of units in a deterministic system to be expected values of some underlying probability distribution. For example, in order to compute the probability of a given activation state, one needs to assume that the units are statistically independent (see Hinton, 1989b; Peterson & Anderson, 1987). While Movellan (1990) showed that CHL can be derived independent of the Boltzmann distribution and the concomitant mean-field assumptions, his derivation does not apply when there are hidden units in the network.

Further, the consequences of the relationship between CHL as derived from GeneRec and standard error backpropagation (i.e., that CHL uses the faster midpoint integration method and imposes a symmetry-preservation constraint) should be apparent in the relative learning properties of these algorithms. Thus, this derivation might explain why CHL networks tend to learn faster than equivalent backprop networks. Finally, another advantage of a derivation based on the BP framework is that it is sufficiently general as to allow CHL-like learning rules to be derived for a variety of different activation functions or other network properties.

The Midpoint Method and the GeneRec Approximation to it

As was mentioned above, the use of the average of both the minus and plus phase activations of the sending unit in the GeneRec learning rule corresponds to an approximation of a simple numerical integration

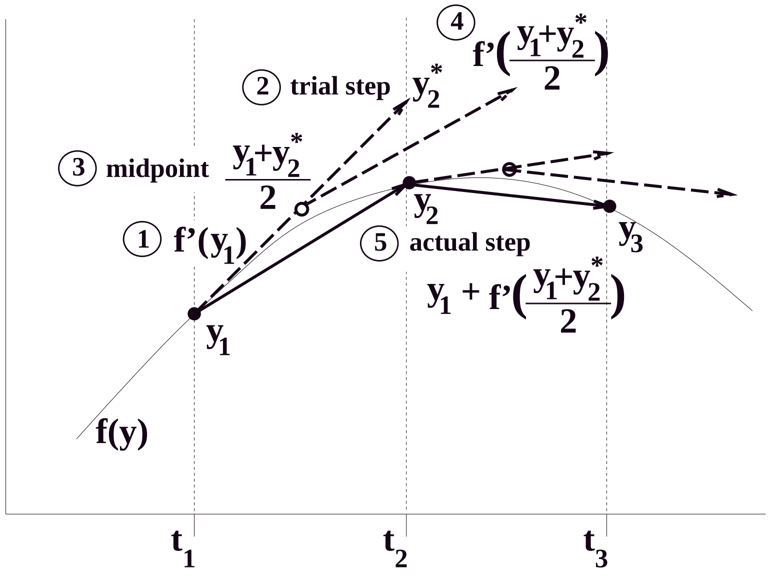

Figure 2: The midpoint method. At each point, a trial step is taken along the derivative at that point, and the derivative re-computed at the midpoint between the point and the trial step. This derivative is then used to take the actual step to the next point, and so on.

technique for differential equations known as the midpoint or second-order accurate Runge-Kutta method (Press, Flannery, Teukolsky, & Vetterling, 1988).

The midpoint method attains second-order accuracy without the explicit computation of second derivatives by evaluating the first derivatives twice and combining the results so as to minimize the integration error. It can be illustrated with the following simple differential equation:

$$\frac{dy(t)}{dt} = f'(y(t)) \tag{22}$$

The simplest way in which the value of the variable y can be integrated is by using a difference equation approximation to the continuous differential equation:

$$y_{t+1} = y_t + \epsilon f'(y_t) + O(\epsilon^2) \tag{23}$$

with a step size of ϵ, and an accuracy to first order in the Taylor series expansion of $f(y_t)$ (and thus an error term of order ϵ^2). This integration technique is known as the *forward Euler* method, and is commonly used in neural network gradient descent algorithms such as BP. By comparison with (23), the midpoint method takes a "trial" step using the forward Euler method, resulting in an estimate of the next function value (denoted by the *):

$$y^*_{t+1} = y_t + \epsilon f'(y_t) \tag{24}$$

This estimate is then used to compute the actual step, which is the derivative computed at a point halfway between the current y_t and estimated y^*_{t+1} values (see figure 2):

$$y_{t+1} = y_t + \epsilon f'\left(\frac{y_t + y^*_{t+1}}{2}\right) + O(\epsilon^3) \tag{25}$$

In terms of a Taylor series expansion of the function $f(y_t)$ at the point y_t, evaluating the derivative at the midpoint as in (25) cancels out the first-order error term ($O(\epsilon^2)$), resulting in a method with second-order accuracy (Press et al., 1988). Intuitively, the midpoint method is able to "anticipate" the curvature of the gradient, and avoid going off too far in the wrong direction.

There are a number of ways the midpoint method could be applied to error backpropagation. Perhaps the most "correct" way of doing it would be to run an entire batch of training patterns to compute the trial step weight derivative, and then run another batch of patterns with the weights half way between their current and the trial step values to get the actual weight changes to be made. However, this would require roughly twice the number of computations per weight update as standard batch-mode backpropagation, and the two passes of batch-mode learning is not particularly biologically plausible.

The GeneRec version of the midpoint method as given by (18) is an approximation to the "correct" version in two respects:

1. The plus-phase activation value is used as an on-line estimate of the activations that would result from a forward Euler step over the weights. This estimate has the advantages of being available without any additional computation, and it can be used with on-line weight updating, solving both of the major problems with the "correct" version. The relationship between the plus-phase activation and a forward Euler step along the error gradient makes sense given that the plus-phase activation is the "target" state, which is therefore in the direction of reducing the error. Appendix A gives a more formal analysis of this relationship. This analysis shows that the plus phase activation, which does not depend on the learning rate parameter, typically over-estimates the size of a trial Euler step. Thus, the use of the plus-phase activation means that the precise midpoint is not actually being computed. Nevertheless, the anticipatory function of this method is still served when the trial step is exaggerated. Indeed, the simulation results described below indicate that it can actually be advantageous in certain tasks to take a larger trial step.

2. The midpoint method is applied only to the portion of the derivative that distributes the unit's error term amongst its incoming weights, and not to the computation of the error term itself. Thus, the error term $(a_j^+ - a_j^-)$ from (18) is the same as in the standard forward Euler integration method, and only the sending activations are evaluated at the midpoint between the current and the trial step: $\frac{1}{2}(a_i^- + a_i^+)$. This selective application of the midpoint method is particularly efficient for the case of on-line backpropagation because a midpoint value of the unit error term, especially on a single-pattern basis, will typically be smaller than the original error value, since the trial step is in the direction of reducing error. Thus, using the midpoint error value would actually slow learning by reducing the effective learning rate.

To summarize, the advantage of using the approximate midpoint method represented by (18) is that it is so simple to compute, and it appears to reliably speed up learning while still using on-line learning. While other more sophisticated integration techniques have been developed for batch-mode BP (see Battiti, 1992 for a review), they typically require considerable additional computation per step, and are not very biologically plausible.

The GeneRec Approximate Midpoint Method in Backpropagation

In order to validate the idea that CHL is equivalent to GeneRec using the approximation to the midpoint method described above, this approximation can be implemented in a standard backpropagation network and the relative learning speed advantage of this method compared for the two different algorithms. If similar kinds of speedups are found in both GeneRec and backpropagation, this would support the derivation of CHL as given in this paper. Such comparisons are described in the following simulation section.

There are two versions of the GeneRec approximate midpoint method that are relevant to consider. One is a weight-based method which computes the sending unit's trial step activation (h_j^*) based on the trial step weights, and the other is a simpler approximation which uses the unit's error derivative to estimate the trial step activation. In both cases, the resulting trial step activation state h_j^* is averaged with the current activation value h_j to obtain a midpoint activation value which is used as the sending activation state for backpropagation weight updating:

$$\frac{1}{\epsilon}\Delta w_{jk} = \frac{1}{2}(h_j^* + h_j)(t_k - o_k) \tag{26}$$

This corresponds to the GeneRec version of the midpoint method as given in (18). Note that in a three-layer network, only the hidden-to-output weights are affected, since there is no trial step activation value for the input units.

In the BP weight-based midpoint approximation, the trial step activation is computed as if the weights had been updated by the trial step weight error derivatives as follows:

$$\begin{aligned} \eta_j^* &= \sum_i s_i \left(w_{ij} + \epsilon_{ts} \frac{-\partial E}{\partial w_{ij}} \right) \\ h_j^* &= \sigma(\eta_j^*) \end{aligned} \tag{27}$$

where ϵ_{ts} is a learning-rate-like constant that determines the size of the trial step taken. Note that, in keeping with the GeneRec approximation, the actual learning rate ϵ is not included in this equation. Thus, depending on the relative sizes of ϵ_{ts} and ϵ, the estimated trial step activation given by (27) can over-estimate the size of an actual trial step activation. In order to evaluate the effects of this over-estimation, a range of ϵ_{ts} values are explored.

The BP unit-error-based method uses the fact that each weight will be changed in proportion to the derivative of the error with respect to the unit's net input, $\frac{\partial E}{\partial \eta_j}$, to avoid the additional traversal of the weights:

$$\begin{aligned} \eta_j^* &= \eta_j + \epsilon_{ts} F \frac{-\partial E}{\partial \eta_j} \\ h_j^* &= \sigma(\eta_j^*) \end{aligned} \tag{28}$$

where F is the number of receiving weights (fan-in). In this case, the trial step size parameter ϵ_{ts} also reflects the average activity level over the input layer, since each input weight would actually be changed by an amount proportional to the activity of the sending unit. The comments regarding ϵ_{ts} above also apply to this case.

Note that it is the unit-error-based version of the midpoint method that most closely corresponds to the version used in CHL, since both are based on the error derivative with respect to the unit, not the weights into the unit. As the simulations reported below indicate, the midpoint method can speed up on-line BP learning by nearly a factor of two. Further, the unit-error-based version is quite simple and requires little extra computation to implement. Finally, while the unit-error-based version could be applied directly to Almeida-Pineda backpropagation, the same is not true for the weight-based version, which would require an additional activation settling based on the trial step weights. Thus, in order to compare these two ways of implementing the approximate midpoint method, the results presented below are for feedforward backprop networks.

Simulation Experiments

The first set of simulations reported in this section is a comparison of the learning speed between several varieties of GeneRec (including symmetric, midpoint, and CHL) and BP with and without the midpoint integration method. This gives a general sense of the comparative learning properties of the different algorithms, and provides empirical evidence in support of the predicted relationships amongst the algorithms investigated. In the second set of simulations, a detailed comparison of the weight derivatives computed by the Almeida-Pineda version of backpropagation and GeneRec is performed, showing that they both compute the same error derivatives under certain conditions.

Learning Speed Comparisons

While it is notoriously difficult to perform useful comparisons between different learning algorithms, such comparisons could provide some empirical evidence necessary for evaluating the theoretical claims made above in the derivation of the GeneRec algorithm and its relationship to CHL. Note that the intent of this comparison is not to promote the use of one algorithm over another, which would require a much broader sample of commonly-used speedup techniques for backpropagation. The derivation of GeneRec based on AP backpropagation and its relationship with CHL via the approximate midpoint method makes specific predictions about which algorithms will learn faster and more reliably than others, and, to the extent that the following empirical results are consistent with these predictions, this provides support for the above analysis. In particular, it is predicted that GeneRec will be able to solve difficult problems in roughly the same order of epochs as the AP algorithm, and that weight symmetry will play an important role in the ability of GeneRec to solve problems. Further, it is predicted that the midpoint versions of both GeneRec and backprop will learn faster than the standard versions.

Overall, the results are consistent with these predictions. It is apparent that GeneRec networks can learn difficult tasks, and further that the midpoint integration method appears to speed up learning in both GeneRec and backpropagation networks. This is consistent with the idea that CHL is equivalent to GeneRec using this midpoint method. Finally, adding the symmetry preservation constraint to GeneRec generally increases the number of networks that solve the task, except in the case of the 4-2-4 encoder for reasons that are explained below. This is consistent with the idea that symmetry is important for computing the correct error derivatives.

Four different simulation tasks were studied: XOR (with 2 hidden units), a 4-2-4 encoder, the “shifter” task (Galland & Hinton, 1991), and the “family trees” task of Hinton (1986) (with 18 units per hidden and encoding layer). All networks used 0-to-1 valued sigmoidal units. The backpropagation networks used the cross-entropy error function, with an “error tolerance” of .05, so that if the output activation was within .05 of the target, the unit had no error. In the GeneRec networks, activation values were bounded between .05 and .95. In both the GeneRec and AP backpropagation networks, initial activation values were set to 0, and a step size (dt) of .2 was used to update the activations. Settling was stopped when the maximum change in activation (before multiplying by dt) was less than .01. 50 networks with random initial weights (symmetric for the GeneRec networks) were run for XOR and the 4-2-4 encoder, and 10 for the shifter and family trees problems. The training criterion for XOR and the 4-2-4 encoder was .1 total-sum-of-squares error, and the criterion for the shifter and family trees problems was that all units had to be on the right side of .5 for all patterns. Networks were stopped after 5,000 epochs if they had not yet solved the XOR, 4-2-4 encoder and shifter problems, and 1,000 epochs for family trees.

A simple one-dimensional grid search was performed over the learning rate parameter in order to determine the fastest average learning speed for a given algorithm on a given problem. For XOR, the 4-2-4 encoder, and the shifter tasks, the grid was at no less than .05 increments, while a grid of .01 was used for the family trees problem. No momentum or any other modifications to generic backpropagation were used, and weights were updated after every pattern, with patterns presented in a randomly permuted order every epoch. The results presented below are from the fastest networks for which 50% or more reached the training criterion. This is really only important for the XOR problem, since the algorithms did not typically get stuck on the other problems.

The algorithms compared were as follows (see table 3):

BP Standard feedforward error backpropagation using the cross-entropy error function.

Err Method	FF vs. Rec	Euler		Midpoint	
		NonSym	Sym	NonSym	Sym
BP	FF	BP	—	BP Mid	—
	Rec	AP	—	—	—
Act Diff	FF	—	—	—	—
	Rec	GR	GR Sym	GR Mid	CHL

Table 3: Relationship of the algorithms tested with respect to the use of local activations vs. explicit backpropagation (BP, Act Diff) to compute error derivatives, feedforward vs. recurrent (FF, Rec), forward Euler vs. the midpoint method (Euler, Midpoint), and weight symmetrization (NonSym, Sym). GR is GeneRec.

AP Almeida-Pineda backpropagation in a recurrent network using the cross-entropy error function.

BP Mid Wt Feedforward error backpropagation with the weight-based version of the approximate midpoint (27). Several different values of the trial step size parameter ϵ_{ts} were used in order to determine the effects of overestimating the trial step as is the case with GeneRec. The values were: 1, 5, 10, and 25 for XOR and the 4-2-4 encoder, .5, 1, and 2 for the shifter problem, and .05, .1, .2, and .5 for the family trees problem. The large trial step sizes resulted in faster learning in small networks, but progressively smaller step sizes were necessary for the larger problems.

BP Mid Un Feedforward error backpropagation with the unit-error based version of the midpoint integration method (28). The same trial step size parameters as in BP Mid Wt were used.

GR The basic GeneRec algorithm (17).

GR Sym GeneRec with the symmetry preservation constraint (20).

GR Mid GeneRec with the approximate midpoint method (18).

CHL GeneRec with both symmetry and approximate midpoint method, which is equivalent to CHL (7).

XOR and the 4-2-4 Encoder

The results for the XOR problem are shown in table 4, and those for the 4-2-4 encoder are shown in table 5. These results are largely consistent with the predictions made above, with the exception of an apparent interaction between the 4-2-4 encoder problem and the use of weight symmetrization in GeneRec. Thus, it is apparent that the plain GeneRec algorithm is not very successful or fast, and that weight symmetrization is necessary to improve the success rate (in the XOR task) and the learning speed (in the 4-2-4 encoder). As will be shown in more detail below, the symmetrization constraint is essential for computing the correct error derivatives in GeneRec.

However, the symmetry constraint also effectively limits the range of weight space that can be searched by the learning algorithm (only symmetric weight configurations can be learned), which might affect its ability to get out of bad initial weight configurations. This effect may be compounded in an encoder problem, where the input-to-hidden weights also have a tendency to become symmetric with the hidden-to-output weights. Thus, while the symmetry constraint is important for being able to compute the correct error derivatives, it also introduces an additional constraint which can impair learning, sometimes dramatically (as in the case of the 4-2-4 encoder). Note that on larger and more complicated tasks like the shifter and family trees described below, the advantages of computing the correct derivatives begin to outweigh the disadvantages of the additional symmetry constraint.

Algorithm	ϵ	N	Epcs	SEM
BP	1.95	37	305	58
AP	1.40	35	164	23
BP Mid Wt 1	1.85	39	268	59
BP Mid Wt 5	0.25	25	326	79
BP Mid Wt 10	0.25	34	218	25
BP Mid Wt 25	0.35	27	215	40
BP Mid Un 1	1.40	40	222	28
BP Mid Un 5	1.05	34	138	38
BP Mid Un 10	0.40	26	222	10
BP Mid Un 25	0.30	31	178	37
GR	0.20	9†	3795	267
GR Sym	0.60	31	334	7.1
GR Mid	1.75	33	97	4.6
CHL	1.80	28	59	1.8

Table 4: Results for the XOR problem. ϵ is the optimal learning rate, *N* is the number of networks that successfully solved the problem (out of 50, minimum of 25), *Epcs* is the mean number of epochs required to reach criterion, and *SEM* is the standard error of this mean. Algorithms are as described in the text. † Note that this was the best performance for the GR networks.

Algorithm	ϵ	N	Epcs	SEM
BP	2.40	50	60	5.1
AP	2.80	50	54	3.6
BP Mid Wt 1	1.70	50	60	4.3
BP Mid Wt 5	1.65	50	48	2.8
BP Mid Wt 10	2.35	50	45	3.6
BP Mid Wt 25	2.25	50	37	3.0
BP Mid Un 1	2.20	50	54	4.2
BP Mid Un 5	2.10	50	42	2.5
BP Mid Un 10	2.10	50	40	2.9
BP Mid Un 25	1.95	50	34	1.8
GR	0.60	45	418	28
GR Sym	1.40	28	88	2.9
GR Mid	2.40	46	60	3.4
CHL	1.20	28	77	1.8

Table 5: Results for the 4-2-4 encoder problem. ϵ is the optimal learning rate, *N* is the number of networks that successfully solved the problem (out of 50), minimum of 25, *Epcs* is the mean number of epochs required to reach criterion, and *SEM* is the standard error of this mean. Algorithms are as described in the text.

The other main prediction from the analysis is that the approximate midpoint method will result in faster learning, both in BP and GeneRec. This appears to be the case, where the speedup relative to regular back-prop was nearly two-fold for the unit-error based version with a trial step size of 25. The general advantage of the unit-error over the weight based midpoint method in BP is interesting considering that this corresponds to the GeneRec version of the midpoint method. The speedup in GeneRec for both the CHL *vs* GR Sym and GR Mid *vs* GR comparisons was substantial in general. Further, it is interesting that the approximate mid-point method alone (without the symmetrization constraint) can enable the GeneRec algorithm to successfully solve problems. Indeed, on both of these tasks, GR Mid performed better than GR Sym. This might be attributable to the ability of the midpoint method to compute better weight derivatives which are less affected by the inaccuracies introduced by the lack of weight symmetry. However, note that while this seems to hold for all of the three-layer networks studied, it breaks down in the family trees task which requires error derivatives to be passed back through multiple hidden layers. Also, only on the 4-2-4 encoder did GR Mid perform better than CHL, indicating that there is generally an advantage to having the correct error derivatives via weight symmetrization in addition to using the midpoint method.

An additional finding is that there appears to be an advantage for the use of a recurrent network over a feed-forward one, based on a comparison of AP *vs* BP results. This can be explained by the fact that small weight changes in a recurrent network can lead to more dramatic activation state differences than in a feedforward network. In effect, the recurrent network has to do less work to achieve a given set of activation states than does a feedforward network. This advantage for recurrency, which should be present in GeneRec, is probably partially offset by the additional weight symmetry constraint. Further, recurrency appears to become a liability in networks with multiple hidden layers, based on the family trees results presented below.

The Shifter Task

The shifter problem is a larger task than XOR and the 4-2-4 encoder, and thus might provide a more realistic barometer of performance on typical tasks.[4] The version of the shifter problem used here had two 4 bit input patterns, one of which was a shifted version of the other. There were three values of shift, -1, 0, and 1, corresponding to one bit to the left, the same, and one bit to the right (with wrap-around). Of the 16 possible binary patterns on 4 bits, 4 were unsuitable because they result in the same pattern when shifted right or left (1111, 1010, 0101, and 0000). Thus, there were 36 training patterns (the 12 bit patterns shifted in each of 3 directions). The task was to classify the shift direction by activating one of 3 output units. While larger versions of this task (more levels of shift, more bits in the input) were explored, this configuration proved the most difficult (in terms of epochs) for a standard BP network to solve.

The results, shown in table 6, provide clearer support for the predicted relationships than the two previous tasks. In particular, the midpoint-based speedup is comparable between the BP and GeneRec cases, and the role of symmetry in GeneRec is unambiguously important for solving the task, as is evident from the almost complete failure of the non-symmetric version to learn the problem. However, it is interesting that even in this more complicated problem the use of the approximate midpoint method without the additional symmetrizing constraint enables the GeneRec networks to learn the problem. Nevertheless, the combination of the approximate midpoint method and the symmetrizing constraint (i.e., the CHL algorithm) performs better than either alone.

As in the previous tasks, there appears to be an advantage for the use of a recurrent network over a non-recurrent one, as evidenced by the faster learning of AP compared to BP.

The Family Trees Task

As was mentioned in the introduction, the family trees task of Hinton (1986) is of particular interest because Galland (1993) reported that he was unable to train CHL to solve this problem. While I was unable to

[4] Note that other common tasks like digit recognition or other classification tasks were found to be so easily solved by a standard BP network (under 10 epochs), that they did not provide a useful dynamic range to make the desired comparisons.

Algorithm	ϵ	N	Epcs	SEM
BP	1.25	10	76.2	6.4
AP	1.35	10	56.8	4.2
BP Mid Wt .5	0.40	10	63.6	4.8
BP Mid Wt 1	0.45	10	42.5	2.9
BP Mid Wt 2	0.35	10	47.0	3.5
BP Mid Un .5	0.30	10	48.0	1.7
BP Mid Un 1	0.35	10	41.2	3.3
BP Mid Un 2	0.15	10	51.8	3.8
GR	0.10	1	1650	—
GR Sym	0.90	10	105	5.0
GR Mid	0.65	10	84.2	13.4
CHL	0.70	10	42.7	2.2

Table 6: Results for the shifter task. ϵ is the optimal learning rate, *N* is the number of networks that successfully solved the problem (out of 10), *Epcs* is the mean number of epochs required to reach criterion, and *SEM* is the standard error of this mean. Algorithms are as described in the text.

Algorithm	ϵ	N	Epcs	SEM
BP	0.39	10	129	3.0
AP	0.30	10	181	11
BP Mid Wt .05	0.37	10	131	6.4
BP Mid Wt .1	0.38	10	130	5.1
BP Mid Wt .2	0.21	10	136	6.0
BP Mid Un .05	0.24	10	127	6.4
BP Mid Un .1	0.23	10	114	6.7
BP Mid Un .2	0.19	10	123	8.7
GR	—	0	—	—
GR Sym	0.20	10	409	14
GR Mid	—	0	—	—
CHL	0.10	10	328	23

Table 7: Results for the family trees problem. ϵ is the optimal learning rate, *N* is the number of networks that successfully solved the problem (out of 10, minimum of 5), *Epcs* is the mean number of epochs required to reach criterion, and *SEM* is the standard error of this mean. Algorithms are as described in the text.

get a CHL network to learn the problem with the same number of hidden units as was used in the original backprop version of this task (6 "encoding" units per input/output layer, and 12 central hidden units), simply increasing the number of encoding units to 12 was enough to allow CHL to learn the task, although not with 100% reliability. Thus, the learning rate search was performed on networks with 18 encoding and 18 hidden units to ensure that networks were capable of learning.

As can be seen from the results shown in table 7, the CHL networks were able to reliably solve this task within a roughly comparable number of epochs as the AP networks. Note that the recurrent networks (GeneRec and AP) appear to be at a disadvantage relative to feedforward BP[5] on this task, probably due to the difficulty of shaping the appropriate attractors over multiple hidden layers. Also, symmetry preservation appears to be critical for GeneRec learning in deep networks, since GeneRec networks without this were unable to solve this task (even with the midpoint method).

The comparable performance of AP and CHL supports the derivation of CHL via the GeneRec algorithm as essentially a form of backpropagation, and calls into question the analyses of Galland (1993) regarding the limitations of CHL as a deterministic approximation to a Boltzmann machine. It is difficult to determine what is responsible for the failure to learn the family trees problem reported in Galland (1993), since there are several differences in the way those networks were run compared to the ones described above, including the use of an annealing schedule, not using the .05, .95 activation cutoff, using activations with -1 to +1 range, using batch mode instead of on-line weight updating, and using activation-based as opposed to net-input-based settling.

Finally, only the unit-error based midpoint method in backpropagation showed a learning speed advantage in this task. This is consistent with the trend of the previous results. The advantage of the unit-error based midpoint method might be due to the reliance on the derivative of the error with respect to the hidden unit itself, which could be a more reliable indication of the curvature of the derivative than the weight derivatives used in the other method.

The GeneRec Approximation to AP BP

The analysis presented earlier in the paper shows that GeneRec should compute the same error derivatives as the Almeida-Pineda version of error backpropagation in a recurrent network if the following conditions hold:

- The difference of the plus and minus phase activation terms in GeneRec, which are updated in separate iterative activation settling phases, can be used to compute a unit's error term instead of the iterative update of the difference itself, which is what Almeida-Pineda uses.
- The reciprocal weights are symmetric. This enables the activation signals from the output to the hidden units (via the recurrent weights) to reflect the contribution that the hidden units made to the output error (via the forward-going weights).
- The difference of activations in the plus and minus phases is a reasonable approximation to the difference of net inputs times the derivative of the sigmoidal activation function. Note that this only affects the overall magnitude of the weight derivatives, not their direction.

In order to evaluate the extent to which these conditions are violated and the effect that this has on learning in GeneRec, two identical networks were run side-by-side on the same sequence of training patterns, with one

[5] It should be noted that the performance of feedforward BP on this task is much faster than previously reported results. This is most likely due to the use of on-line learning and not using momentum, which enables the network to take advantage of the noise due to the random order of the training patterns to break the symmetry of the error signals generated in the problem and distinguish amongst the different training patterns.

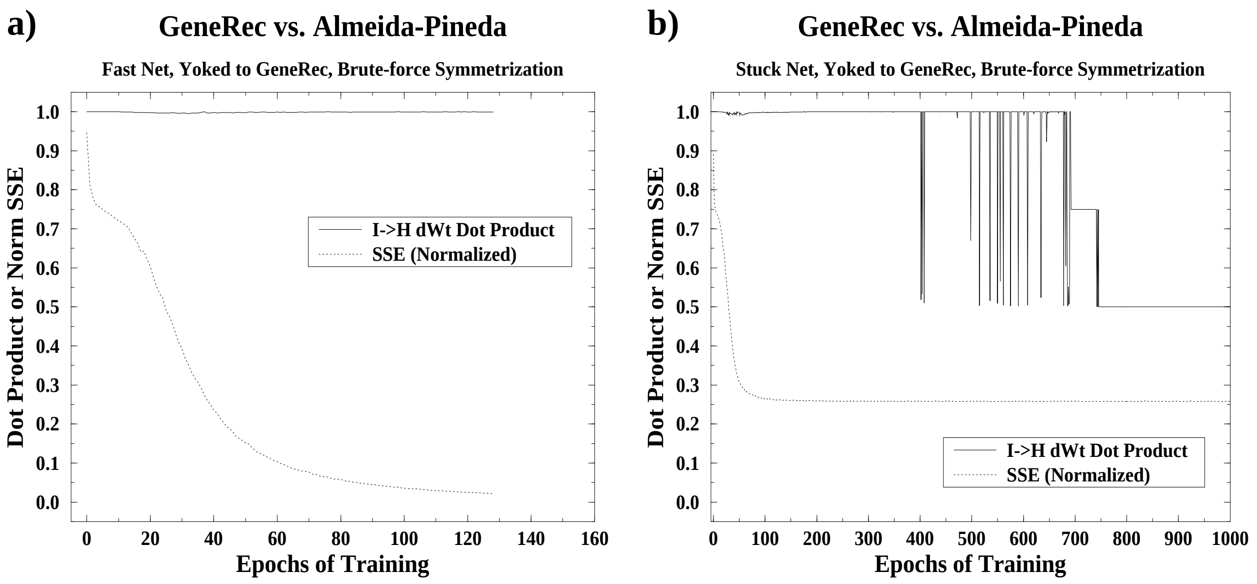

Figure 3: Correspondence (average normalized dot product) of weight derivatives computed by GeneRec and Almeida-Pineda algorithms for two random initial weight configurations. The weights were yoked to those computed by GeneRec, and symmetry was preserved by the brute force method. The correspondence is nearly perfect until late in the training of the stuck network, at which point the network has developed very large weights, which appear to affect the accuracy of the computed weight derivatives.

network using AP (with the cross-entropy error function) to compute the error derivatives, and the other using GeneRec. The standard 4-2-4 encoder problem was used. The extent to which GeneRec error derivatives are the same as those computed by AP was measured by the normalized dot product between the weight derivative vectors computed by the two algorithms. This comparison was made for the input-to-hidden weights ($I \Rightarrow H$) since they reflect the error derivatives computed by the hidden units. Since the hidden-to-output weights are driven by the error signal on the output units, which is given by the environment, these derivatives were always identical between the two networks. In order to control for weight differences that might accumulate over time in the two networks, the weights were copied from the GeneRec network to the AP network after each weight update. Networks were also run without this "yoking" of the weights in order to determine how different the overall learning trajectory was between the two algorithms given the same initial weight values. The weights were always initialized to be symmetric.

As was noted above, the basic GeneRec algorithm does not preserve the symmetry of the weights, which will undoubtedly affect the computation of error gradients. The extent of symmetry was measured by the normalized dot product between the reciprocal hidden and output weights. It is predicted that this symmetry measure will determine in large part the extent to which GeneRec computes the same error derivatives as AP. In order to test this hypothesis, two methods for preserving the symmetry of the weights during learning were also used. One method was to use the symmetry-preserving learning rule shown in (20), and the other was a "brute-force" method where reciprocal weights were set to the average of the two values after they were updated. The advantage of this later method is that, unlike the (20) rule, it does not change the computed weight changes.

The parameters used in the networks were: activation step size (dt) of .2, initial activations set to 0, settling cutoff at .01 maximum change in activation, learning rate of .6, and initial weights uniformly random between $\pm .5$.

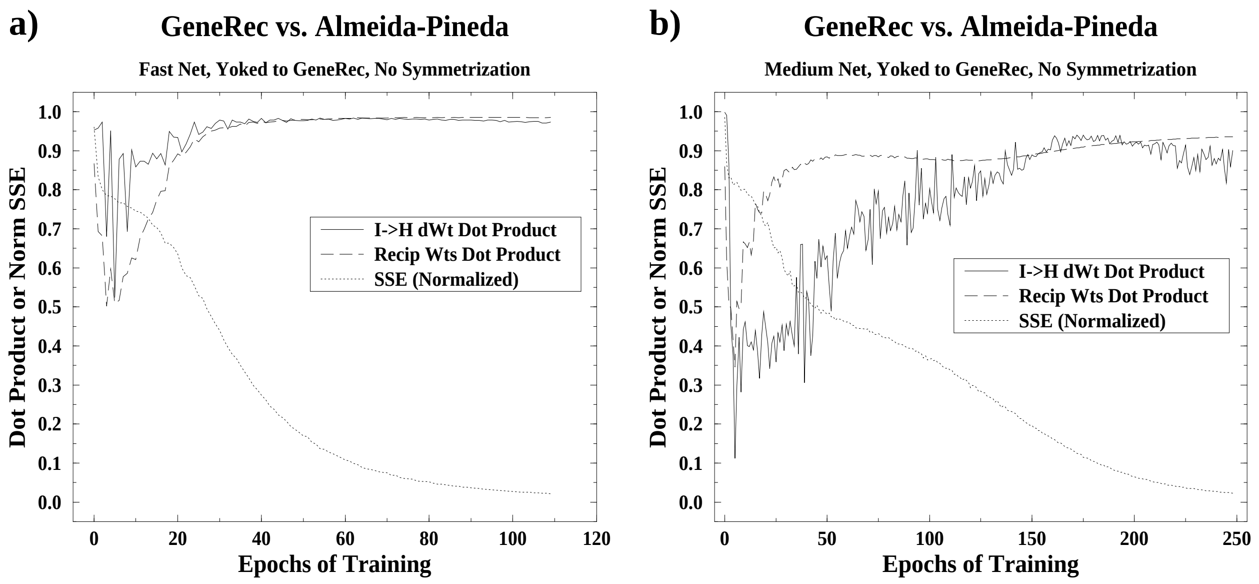

Figure 4: Correspondence (average normalized dot product) of weight derivatives computed by GeneRec and Almeida-Pineda algorithms for two random initial weight configurations. The weights were yoked to those computed by GeneRec. No symmetry was imposed on the weights. The correspondence appears to be roughly correlated with the extent to which the weights are symmetric.

The main result of this analysis is that the GeneRec algorithm typically computes essentially the same error derivatives as AP *except when the weights are not symmetric*. This can be seen in figure 3, which shows two different networks running with weights yoked and using the brute-force symmetrizing method. The weight derivatives computed by GeneRec have a normalized dot product with those computed by AP that is nearly always 1, except when the weights got very large in the network that was stuck in a local minimum. This result shows that GeneRec usually computes the appropriate backpropagation error gradient based on the difference of equilibrium activation states in the plus and minus phases, supporting the approximation given in (16).

In contrast, when no weight symmetrizing is being enforced, the correspondence between the GeneRec and AP weight derivatives appears to be correlated with the extent to which the weights are symmetric, as can be seen in figure 4. Indeed, based on results of many runs (not shown), the ability of the GeneRec network to solve the task appeared to be correlated with the extent to which the weights remain symmetric. Note that even without explicit weight symmetrization or a symmetry preserving learning rule, the weights can become symmetric due to a fortuitous correspondence between weight changes on the reciprocal sets of weights.

Using the symmetry preserving rule (20) resulted in weight changes that were typically different from those computed by AP, even though the above results show that the error derivatives at the hidden unit were correct. This is simply due to the fact that symmetric GeneRec has an additional symmetry preserving term which is not present in AP. Nevertheless, the symmetric GeneRec algorithm resulted in non-yoked learning trajectories which mirrored those of the AP algorithm remarkably closely. A representative example is shown in figure 5. It is difficult to be certain about the source of this correspondence, which did not occur in non-yoked networks using the brute-force symmetry preservation method.

Finally, there is some question as to whether GeneRec will compute the correct error derivatives in a network with multiple hidden layers, where the differences in the way GeneRec and AP compute the error terms might become more apparent due to the greater influence of recurrent setting in both the minus and plus phases. Also, based on the kinds of approximations made in deriving CHL as a deterministic Boltzmann

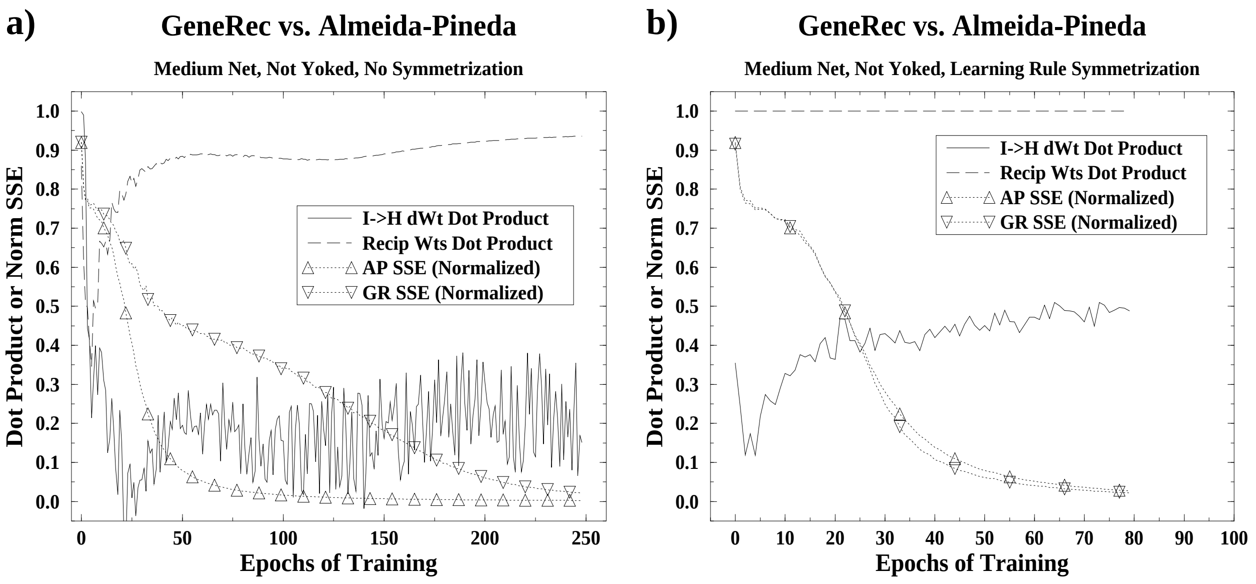

Figure 5: Learning trajectories and error derivative correspondence for non-yoked AP and GeneRec networks with the same initial weights. **a)** Shows standard GeneRec without any weight symmetrization. **b)** shows GeneRec with the symmetry preserving learning rule. Even though this rule does not result in the networks computing the same weight updates, they follow a remarkably similar learning trajectory. This is not the case for regular GeneRec.

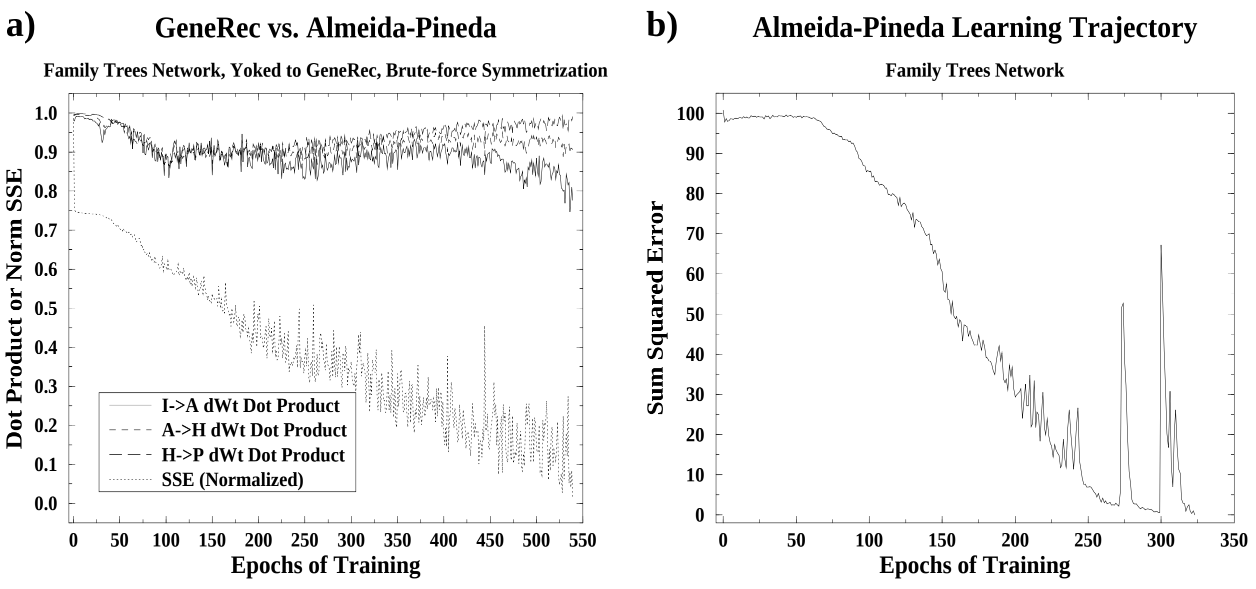

Figure 6: **a)** Correspondence (average normalized dot product) of weight derivatives computed by GeneRec and Almeida-Pineda algorithms for a family trees network. The weights were yoked to those computed by GeneRec, and symmetry was preserved by the brute force method. $I \Rightarrow A$ is the "agent input" to the "agent encoding" hidden layer weights, $A \Rightarrow H$ is the "agent encoding" to the central hidden layer weights, and $H \Rightarrow P$ is the hidden layer to the "patient encoding" layer weights. The correspondence is not as good as a that for the three layer network, but remains largely above .9. **b)** Shows the learning trajectory for just the AP algorithm, which is not smooth like feedforward BP.

machine, and the results of simulations, Galland (1993) concluded that the limitations of CHL become more apparent as the number of hidden layers are increased (i.e., in "deep" networks).

To address the performance of GeneRec in a deep network, the same analysis as described above was performed on the family trees network (Hinton, 1986) with the brute-force symmetrization and weight yoking. This network has three layers of hidden units. Normalized dot-product measurements of the error derivatives computed on the weights from the "agent input" to the "agent encoding" hidden layer $I \Rightarrow A$, "agent encoding" to the central hidden layer $A \Rightarrow H$, and the hidden layer to the "patient encoding" layer $H \Rightarrow P$. These weights are 3, 2, and 1 hidden layers (respectively) removed from the output layer. Figure 6a shows that GeneRec still computes largely the same error derivatives as AP backpropagation even in this case. The normalized dot product measures were usually greater than .9, and never went below .7. The discrepancy between GeneRec and AP tended to increase as training proceeded for the deeper weights ($I \Rightarrow A$). This shows that as the weights got larger, the differences between GeneRec and AP due to the way that the error is computed over recurrent settling became magnified.

One of the primary problems with CHL that was emphasized by Galland (1993) is the jumpy character of the error function over learning, which was argued to not provide much useful guidance in learning. However, figure 6b shows that the AP algorithm also suffers from a bumpy error surface. The frequency of the AP bumps seems to be a bit lower, but the amplitude can be higher. This indicates that the bumpiness is due at least in part to the recurrent nature of the network, where small weight changes can lead to very different activation states, and not to a deficiency in the learning algorithm *per se*.

Possible Biological Implementation of GeneRec Learning

The preceding analysis and simulations show that the GeneRec family of phase-based, error-driven learning rules can approximate error backpropagation using locally available activation variables. The fact that these variables are available locally makes it more plausible that such a learning rule could be employed by real neurons. Also, the use of activation-based signals (as opposed to error or other variables) increases plausibility because it is relatively straightforward to map unit activation onto neural variables such as time-averaged membrane potential or spiking rate. However, there are three main features of the GeneRec algorithm that could potentially be problematic from a biological perspective: 1) weight symmetry; 2) the origin of plus and minus phase activation states; 3) the ability of these activation states to influence synaptic modification according to the learning rule. These issues are addressed below in the context of the CHL version of GeneRec (7), since it is the overall best performer, and has a simpler form than the other GeneRec versions. Since the neocortex is the single most important brain area for the majority of cognitive phenomena, it is the focus of this discussion.

Weight Symmetry in the Cortex

There are two ways in which the biological plausibility of the weight symmetry requirement in GeneRec (which was shown above to be important for computing the correct error gradient) can be addressed. One is to show that exact symmetry is not critical to the proper functioning of the algorithm, so that only a rough form of symmetry would be required of the biology. The other is to show that at least this rough form of symmetry is actually present in the cortex. Data consistent with these arguments are summarized briefly here.

As a first-order point, Hinton (1989b) noted that a symmetry preserving learning algorithm like CHL, when combined with weight decay, will automatically lead to symmetric weights even if they did not start out that way. However, this assumes that all of the units are connected to each other in the first place. This more difficult case of connection asymmetry was investigated in Galland and Hinton (1991) for the CHL learning algorithm. It was found that the algorithm was still effective even when all of the connectivity was asymmetric (i.e., for each pair of non-input units, only one of the two possible connections between them existed). This robustness can be attributed to a redundancy in the ways in which the error signal information can be obtained (i.e., a given hidden unit could obtain the error signal directly from the output units, or indirectly through connections to other hidden units). Also, note that the absence of any connection at all is very different from the presence of connection with a non-symmetric weight value, which is the form of asymmetry that was found to be problematic in the above analysis. In the former case, only a subset of the error gradient information is available, while the latter case can result in specifically *wrong* gradient information due to the influence of the non-symmetric weight. Due to the automatic symmetrization property of CHL, the latter case is unlikely to be a problem.

In terms of biological evidence for symmetric connectivity, there is some indication that the cortex is at least roughly symmetrically connected. At the level of identifiable anatomical subregions of cortex, the vast majority of areas (at least within the visual cortex) are symmetrically connected to each other. That is, if area A projects to area B, area A also receives a projection from area B (Felleman & Van Essen, 1991). At the level of cortical columns or "stripes" within the prefrontal cortex of the monkey, Levitt, Lewis, Yoshioka, and Lund (1993) showed that connectivity was symmetric between interconnected stripes. Thus, if a neuron received projections from neurons in a given stripe, it also projected to neurons in that stripe. The more detailed level of individual neuron symmetric connectivity is difficult to assess empirically, but there is at least no evidence that it does *not* exist. Further, given that there is evidence for at least rough symmetry, detailed symmetry may not be critical since, as demonstrated by Galland and Hinton (1991), CHL can use asymmetrical connectivity as long as there is some way of obtaining reciprocal information through a subset of symmetric connections or indirectly via other neurons in the same area.

Phase-Based Activations in the Cortex

The origin of the phase-based activations that are central to the GeneRec algorithm touches at the heart of perhaps the most controversial aspect of error-driven learning in the cortex: "where does the teaching signal come from?" In GeneRec, the teaching signal is just the plus-phase activation state. Thus, unlike standard backpropagation, GeneRec suggests that the teaching signal is just another state of "experience" in the network. One can interpret this state as that of experiencing the actual *outcome* of some previous conditions. Thus, the minus phase can be thought of as the *expectation* of the outcome given these conditions. For example, after hearing the first three words of a sentence, an expectation will develop of which word is likely to come next. The state of the neurons upon generating this expectation is the minus phase. The experience of hearing or reading the actual word that comes next establishes a subsequent locomotive[6] state of activation, which serves as the plus phase. This idea that the brain is constantly generating expectations about subsequent events, and that the discrepancies between these expectations and subsequent outcomes can be used for error-driven learning, has been suggested by McClelland (1994) as a psychological interpretation of the backpropagation learning procedure. It is particularly attractive for the GeneRec version of backpropagation, which uses only activation states, because it requires no additional mechanisms for providing specific teaching signals other than the effects of experience on neural activation states in a manner that is widely believed to be taking place in the cortex anyway.

Further, there is evidence from ERP recordings of electrical activity over the scalp during behavioral tasks that cortical activation states reflect expectations and are sensitive to differential outcomes. For example, the widely-studied *P300* wave, which is a positive-going wave that occurs around 300 msec after stimulus onset, is considered to measure a violation of subjective expectancy which is determined by preceding experience over both the short and long term (Hillyard & Picton, 1987). In more formal terms, Sutton, Braren, Zubin, and John (1965) showed that the P300 amplitude is determined by the amount of prior uncertainty that is resolved

[6] This is just to demonstrate that such expectations are being generated and it is salient when they are violated.

Plus Phase Variables	Minus Phase Variables	
	$a_i^- a_j^- \approx 0$ ($[Ca^{2+}]_i$ near 0)	$a_i^- a_j^- \approx 1$ ($[Ca^{2+}]_i$ elevated)
$a_i^+ a_j^+ \approx 0$	$\Delta w_{ij} = 0$	$\Delta w_{ij} = -$ (LTD)
$a_i^+ a_j^+ \approx 1$	$\Delta w_{ij} = +$ (LTP)	$\Delta w_{ij} = 0^*$

Table 8: Directions of weight change according to the CHL rule for four qualitative conditions, consisting of the combinations of two qualitative levels of minus and plus phase activation coproduct values. The minus phase activation coproduct is thought to correspond to $[Ca^{2+}]_i$. Increases in synaptic efficacy correspond to long-term potentiation (LTP) and decreases are long-term depression (LTD). * This cell is not consistent with the biological mechanism because both $[Ca^{2+}]_i$ and synaptic activity lead to LTP, not the absence of LTP. See text for a discussion of this point.

by the processing of a given event. Thus, the nature of the P300 is consistent with the idea that it represents a plus phase wave of activation following in a relatively short time-frame the development of minus phase expectations. While the specific properties of the P300 itself might be due to specialized neural mechanisms for monitoring discrepancies between expectations and outcomes, its presence suggests the possibility that neurons in the mammalian neocortex experience two states of activation in relatively rapid succession, one corresponding to expectation and the other corresponding to outcome.

Finally, note that for most of the GeneRec variants, it seems that the neuron needs to have both the plus and minus phase activation signals in reasonably close temporal proximity in order to adjust its synapses based on both values. This is consistent with the relatively rapid expectation-outcome interpretation given above. However, CHL is a special case, since it is simply the difference between the coproduct of same-phase activations, which could potentially be computed by performing simple Hebbian associative learning for the plus phase at any point, and at any other point, performing anti-Hebbian learning on the minus phase activations (Hinton & Sejnowski, 1986). This leaves open the problems of how the brain would know when to change the sign of the weight change, and how this kind of global switch could be implemented. Also, people are capable of learning things relatively quickly (within seconds or at least minutes), so this phase switching is unlikely to be a function of the difference between REM sleep and waking behavior, as has been suggested for phase-based learning algorithms (Hinton & Sejnowski, 1986; Linsker, 1992; Crick & Mitchison, 1983). While it might be possible to come up with answers to these problems, a temporally local mechanism like that suggested above seems more plausible.

Synaptic Modification Mechanisms

Having suggested that the minus and plus phase activations follow each other in rapid succession, it remains to be shown how these two activation states could influence synaptic modification in a manner largely consistent with the CHL version of GeneRec (7). It turns out that the biological mechanism proposed below accounts for only three out of four different qualitative ranges of the sign of the weight change required by CHL (see table 8). Specifically, the proposed mechanism predicts weight increase to occur when both the pre and postsynaptic neurons are active in both the plus and minus phases, whereas CHL predicts that the weight change in this condition should be zero. Thus, the proposed mechanism corresponds to a combination of CHL and a Hebbian-style learning rule, the computational implications of which are the subject of O'Reilly (1996), which shows that the combination of error-driven and associative learning can be generally beneficial for solving many different kinds of tasks.

To briefly summarize the findings of O'Reilly (1996), the Hebbian component can be thought of as imposing additional constraints on learning much in the way that weight decay is used in conjunction with standard error backpropagation. However, the constraints imposed by Hebbian learning are actually capable of producing useful representations on their own (unlike weight decay, which would simply reduce all weights to zero if left to its own devices). Thus, the combination of CHL and Hebbian learning results in networks that, unlike those with weight decay, learn faster (especially in deep networks like the family trees task), and generalize better (due to the effects of the additional constraints) than plain CHL networks. However, for the purposes of the present paper, the crucial aspect of the following mechanism is that it provides the error correction term, which occurs when the synaptic coproduct $a_i a_j$ was larger in the minus phase than in the plus phase. This is the defining aspect of the error-driven learning performed by CHL, since the other qualitative ranges of the CHL learning rule are similar to standard Hebbian learning, as is evident from table 8.

For GeneRec-style learning to occur at a cellular and synaptic level, the neuron needs to be able to retain some trace of the minus phase activation state through the time when the neuron experiences its plus phase activation state. Reasoning from the ERP data described above, this time period might be around 300 milliseconds or so. A likely candidate for the minus phase trace is intracellular Ca^{2+} ($[Ca^{2+}]_i$), which enters the postsynaptic area via NMDA channels if both pre and postsynaptic neurons are active. To implement a GeneRec-style learning rule, this minus phase $[Ca^{2+}]_i$ trace needs to interact with the subsequent plus phase activity to determine if the synapse is potentiated (LTP) or depressed (LTD). In what follows, the term *synaptic activity* will be used to denote the activation coproduct term $a_i a_j$, which is effectively what determines the amount of $[Ca^{2+}]_i$ that enters through the NMDA channel (Collingridge & Bliss, 1987).

There are two basic categories of mechanism which can provide the crucial error-correcting modulation of the sign of synaptic modification required by CHL. One such mechanism involves an interaction between membrane potential or synaptic activity and $[Ca^{2+}]_i$, while another depends only on the level of $[Ca^{2+}]_i$. Further, there are many ways in which these signals and their timing can affect various second-messenger systems in the cell to provide the necessary modulation. In favor of something like the first mechanism, there is evidence that the mere presence of postsynaptic $[Ca^{2+}]_i$ is insufficient to cause LTP (Kullmann, Perkel, Manabe, & Nicoll, 1992; Bashir, Bortolotto, & Davies, 1993), but it is unclear exactly what additional factor is necessary (Bear & Malenka, 1994). One hypothesis is that LTP depends on the activation of metabotropic glutamate receptors, which are activated by presynaptic activity and can trigger various mechanisms in the postsynaptic synaptic compartment (Bashir et al., 1993). On the other hand, a proposed mechanism that depends only on the level of postsynaptic $[Ca^{2+}]_i$ (Lisman, 1989), has received some recent empirical support (reviewed in Lisman, 1994; Bear & Malenka, 1994). This proposal stipulates that increased but moderate concentrations of postsynaptic $[Ca^{2+}]_i$ lead to LTD, while higher concentrations lead to LTP. Artola and Singer (1993) argue that this mechanism is consistent with the *ABS* learning rule (Hancock, Smith, & Phillips, 1991; Artola, Brocher, & Singer, 1990; Bienenstock, Cooper, & Munro, 1982), which stipulates that there are two thresholds for synaptic modification, Θ^+ and Θ^-. A level of $[Ca^{2+}]_i$ which is higher than the high threshold Θ^+ leads to LTP, while a level which is lower than this high threshold but above the lower Θ^- threshold leads to LTD.

Either of the above mechanisms would be capable of producing the pattern of synaptic modification shown in table 8 in the context of a proposed mechanism defined by the following properties:

1. Some minimal level of $[Ca^{2+}]_i$ is necessary for any form of synaptic modification (LTP or LTD).

2. $[Ca^{2+}]_i$ changes relatively slowly, and persists for at least 300+ milliseconds. This allows $[Ca^{2+}]_i$ to represent prior minus phase activity, even if the synapse is not subsequently active in the plus phase.

3. Synaptic modification occurs based on the postsynaptic state after plus phase activity. This can happen locally if synaptic modification occurs after around 300+ milliseconds since the entry of Ca^{2+} into the postsynaptic area (and the plus phase activity states last for at least this long). Alternatively, there could be a relatively global signal corresponding to the plus phase that triggers synaptic modification (e.g., as provided by dopaminergic or cholinergic modulation triggered by systems sensitive to the experience of outcomes following expectations).

4. If $[Ca^{2+}]_i$ was present initially (in the minus phase) due to synaptic activity, but then the synaptic

activity diminished or ceased (in the plus phase), LTD should occur. This would be expected from the mechanisms described above, either because of an explicit interaction between synaptic activity at the time of modification in the plus phase and the trace of $[Ca^{2+}]_i$ from the minus phase, or because the minus phase $[Ca^{2+}]_i$ will have decayed into the LTD range by the time modification occurs in the plus phase.

5. If synaptic activity is taking place in the plus phase state, sufficient $[Ca^{2+}]_i$ is present and LTP occurs. Note that this means that any time the plus-phase activation coproduct $(a_i^+ a_j^+)$ is reasonably large, regardless of whether there was any prior minus phase activity, the weights will be increased. This leads to a combined CHL and Hebbian learning rule as discussed above.

There is direct evidence in support of several aspects of the proposed mechanism, some of which was discussed above, and indirect evidence in support of most of the remainder. Since the empirical literature on LTP and LTD is vast, only a brief summary will be given here (see Artola & Singer, 1993; Bear & Malenka, 1994; Linden, 1994; Malenka & Nicoll, 1993, for recent reviews). It should be noted that most of these findings have been described both in the hippocampus and neocortex, and appear to be quite general (Artola & Singer, 1993; Linden, 1994). Also, note that the NMDA receptor itself is not subject to potentiation, so that the current value of the synaptic weight does not have to be included in learning rules, which is in accordance with GeneRec.

With respect to point 1, the importance of $[Ca^{2+}]_i$ for LTP has been known for a while (e.g., Collingridge & Bliss, 1987), and it is now clear that it is critical for LTD as well (Brocher, Artola, & Singer, 1992; Mulkey & Malenka, 1992; Hirsh & Crepel, 1992). In support of point 2, the time course of $[Ca^{2+}]_i$ concentration has been measured in several studies (e.g., Jaffe, Johnston, Lasser-Ross, Lisman, Miyakawa, & Ross, 1992; Perkel, Petrozzino, Nicoll, & Connor, 1993), and it appears to be relatively long-lasting (on the order of 1 or more seconds), though it is not clear that these results reflect what would happen under less invasive conditions.

As for point 3, Malenka, Lancaster, and Zucker (1992) found that a significant time period (up to 1-2 seconds) of enhanced postsynaptic $[Ca^{2+}]_i$ was necessary for LTP induction. Also, typical LTP and LTD induction regimes involve constant stimulation at a given frequency for time periods longer than a second. However, the precise time course of synaptic potentiation needs to be studied in greater detail to evaluate this issue fully. With respect to the existence of a global learning signal, the ERP data described earlier and the role of neuromodulatory systems like dopamine suggest that such monitoring systems might exist in the brain. For example, Schultz, Apicella, and Ljungberg (1993) describe the important role that dopamine plays in learning and responding to salient environmental stimuli. However, these modulatory effects are probably not of an all-or-nothing nature, and, given that LTP and LTD can be induced by the direct electrical stimulation of individual neurons, it is not likely that learning is completely dependent on a global signal.

To summarize, the proposed synaptic modification mechanism is consistent with several findings, but also requires further mechanisms. As such, the proposal outlined above constitutes a set of predictions regarding additional factors that should determine the sign and magnitude of synaptic modification.

Conclusions

The analysis and simulation results presented in this paper support the idea that the GeneRec family of learning algorithms are performing variations of error backpropagation in a recurrent network using locally available activation variables. However, there is no single GeneRec algorithm which is exactly equivalent to the Almeida-Pineda algorithm for backpropagation in recurrent networks since GeneRec requires symmetric weights yet it is not itself symmetry preserving.

The idea that the CHL algorithm is equivalent to a symmetry preserving version of GeneRec using the midpoint integration method is supported by the pattern of learning speed results for the different versions of GeneRec, and by the learning speed increases obtained when using the approximate midpoint integration method in backpropagation networks. Further, it was shown that CHL (and symmetric GeneRec without the midpoint method) can reliably learn the family trees problem, calling into question the idea that CHL is a fundamentally flawed learning algorithm for deterministic networks, as was argued by Galland (1993). Thus, the weight of the evidence suggests that CHL should be viewed as a variation of recurrent backpropagation, not as a poor approximation to the Boltzmann machine learning algorithm.

However, as a consequence of the differences between GeneRec and AP backpropagation (mainly the symmetry constraint), one can expect GeneRec to have somewhat different characteristics compared to standard backpropagation algorithms, and it may turn out that these differences have implications for psychological or computational models. Thus, the present analysis does not imply that just because there exists a biologically plausible form of backpropagation, all forms of backpropagation are now biologically plausible.

Finally, while CHL gave the best performance of the GeneRec networks in three out of the four tasks studied in this paper, the symmetry preservation constraint ended up being a liability in the 4-2-4 encoder task. Thus, the GeneRec-based derivation of CHL can have practical consequences in the selection of an appropriate algorithm for a given task. Also, this derivation allows one to derive CHL-like algorithms for different activation functions, and other network parameters.

Perhaps the most important contribution of this work is that it provides a unified computational approach to understanding how error-driven learning might occur in the brain. Given that the GeneRec learning rules are quite possibly the most simple and local way of performing a very general and powerful form of learning, it seems plausible that the brain would be using something like them. The specific biological mechanism proposed in this paper, which is consistent with several empirical findings, provides a starting point for exploring this hypothesis.

Appendix A: Plus Phase Approximation to Trial Step Activations

The relationship between the plus-phase activation of a GeneRec hidden unit j (h_j^+) and that which would result if an Euler weight update step were taken to reduce the error (denoted h_j^*) can be more formally established. This is done by simply re-computing the activation of the hidden unit based on the weights after they have been updated from the current error derivatives. Using the basic GeneRec algorithm with the difference of net-input terms instead of activation terms (this makes the computation easier), the trial step (starred) weights would be as follows:

$$\begin{aligned} w_{ij}^* &= w_{ij} + \epsilon s_i (\eta_j^+ - \eta_j^-) \sigma'(\eta_j^-) & (29) \\ w_{kj}^* &\approx w_{kj} + \epsilon o_k^- (\eta_j^+ - \eta_j^-) \sigma'(\eta_j^-) & (30) \end{aligned}$$

Note that the *original* value of o_k^- is used here, whereas in the exact computation of the midpoint method in a recurrent network, the output activation value would change when the weights are changed. However, it is impossible to express in closed form what this value would be since it would result from a settling process in the recurrent network, so the original value is used as an approximation to the actual value. This is the only sense in which the following analysis is approximate.

The trial step weights above can then be used to compute the net-input that the unit would receive after such weight changes (denoted η_j^*) as follows (using the fact that $\eta_j^- = \sum_i s_i w_{ij} + \sum_k o_k^- w_{kj}$):

$$\begin{aligned} \eta_j^* &\approx \sum_i s_i w_{ij}^* + \sum_k o_k^- w_{kj}^* \\ &\approx \sum_i s_i w_{ij} + \sum_k o_k^- w_{kj} + (\eta_j^+ - \eta_j^-)\epsilon\sigma'(\eta_j^-) \left(\sum_i s_i^2 + \sum_k {o_k^-}^2 \right) & (31) \end{aligned}$$

To simplify, let:

$$u = \epsilon\sigma'(\eta_j^-) \left(\sum_i s_i^2 + \sum_k {o_k^-}^2 \right) \qquad (32)$$

Which gives:

$$\begin{aligned} \eta_j^* &\approx \eta_j^+ u + \eta_j^- (1 - u) \\ &= \eta_j^+ \text{ if } u = 1 & (33) \end{aligned}$$

Thus, the plus-phase net-input (and therefore activation) is equivalent to a forward Euler step if the learning rate ϵ is set so as to meet the conditions in (33):

$$\epsilon = \frac{1}{\sigma'(\eta_j^-) \left(\sum_i s_i^2 + \sum_k {o_k^-}^2 \right)} \qquad (34)$$

Using a fixed learning rate which is smaller than that given by (34) would result in a starred (trial step) activation value which is in the same direction as the plus-phase activation value (since the u term is bounded between zero and one) but not quite as different from the minus phase value.

Acknowledgments

I would like to thank the following people for their useful comments on earlier drafts of this manuscript: Peter Dayan, Jay McClelland, Javier Movellan, Yuko Munakata, and Rich Zemel.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147–169.

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In M. Caudil, & C. Butler (Eds.), *Proceedings of the IEEE First International Conference on Neural Networks San Diego, CA* (pp. 609–618).

Artola, A., Brocher, S., & Singer, W. (1990). Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature*, *347*, 69–72.

Artola, A., & Singer, W. (1993). Long-term depression of excitatory synaptic transmission and its relationship to long-term potentiation. *Trends In Neurosciences*, *16*, 480.

Bashir, Z., Bortolotto, Z. A., & Davies, C. H. (1993). Induction of LTP in the hippocampus needs synaptic activation of glutamate metabotropic receptors. *Nature*, *363*, 347–350.

Battiti, T. (1992). First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, *4*(2), 141–166.

Bear, M. F., & Malenka, R. C. (1994). Synaptic plasticity: LTP and LTD. *Current Opinion in Neurobiology*, *4*, 389–399.

Bienenstock, E. L., Cooper, L. N., & Munro, P. W. (1982). Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience*, *2*(2), 32–48.

Brocher, S., Artola, A., & Singer, W. (1992). Intracellular injection of Ca2+ chelators blocks induction of long-term depression in rat visual cortex. *Proceedings of the National Academy of Sciences*, *89*, 123.

Collingridge, G. L., & Bliss, T. V. P. (1987). NMDA receptors - their role in long-term potentiation. *Trends In Neurosciences*, *10*, 288–293.

Crick, F. H. C. (1989). The recent excitement about neural networks. *Nature*, *337*, 129–132.

Crick, F. H. C., & Mitchison, G. (1983). The function of dream sleep. *Nature*, *304*, 111–114.

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, *1*, 1–47.

Galland, C. C. (1993). The limitations of deterministic Boltzmann machine learning. *Network*, *4*, 355–379.

Galland, C. C., & Hinton, G. E. (1990). Discovering high order features with mean field modules. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems, 2*. San Mateo, CA: Morgan Kaufmann.

Galland, C. C., & Hinton, G. E. (1991). Deterministic Boltzmann learning in networks with asymmetric connectivity. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, & G. E. Hinton (Eds.), *Connectionist Models: Proceedings of the 1990 Summer School* (pp. 3–9). San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Hancock, P. J. B., Smith, L. S., & Phillips, W. A. (1991). A biologically supported error-correcting learning rule. *Neural Computation*, *3*, 201–212.

Hillyard, S. A., & Picton, T. W. (1987). Electrophysiology of cognition. In F. Plum (Ed.), *Handbook of physiology, section I: Neurophysiology volume V: Higher functions of the brain* (pp. 519–584). American Physiological Society.

Hinton, G. E. (1986). Learning distributed representations of concepts. *Proceedings of the 8 th Conference of the Cognitive Science Society* (pp. 1–12). Hillsdale, NJ: Lawrence Earlbaum Associates, Inc.

Hinton, G. E. (1989a). Connectionist learning procedures. *Artificial Intelligence*, *40*, 185–234.

Hinton, G. E. (1989b). Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation*, *1*, 143–150.

Hinton, G. E., & McClelland, J. L. (1988). Learning representations by recirculation. In D. Z. Anderson (Ed.), *Neural Information Processing Systems, 1987* (pp. 358–366). New York: American Institute of Physics.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In (Rumelhart, McClelland, & PDP Research Group, 1986b), Chap. 7, pp. 282–317.

Hirsh, J. C., & Crepel, F. (1992). Postsynaptic Ca^{2+} is necessary for the induction of LTP and LTD of monosynaptic EPSPs in prefrontal neurons: An in vitro study in the rat. *Synapse*, *10*, 173–175.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA*, *81*, 3088–3092.

Jaffe, D. B., Johnston, D., Lasser-Ross, N., Lisman, J. E., Miyakawa, H., & Ross, W. N. (1992). The spread of Na^{+} spikes determines the pattern of dendritic Ca^{2+} entry into hippocampal neurons. *Nature*, *357*, 244–246.

Kullmann, D. M., Perkel, D. J., Manabe, T., & Nicoll, R. A. (1992). Ca^{2+} entry via postsynaptic voltage-sensitive Ca^{2+} channels can transiently potentiate excitatory synaptic transmission in the hippocampus. *Neuron*, *9*, 1175–1183.

LeCun, Y., & Denker, J. S. (1991). A new learning rule for recurrent networks. *Proceedings of of the Conference on Neural Networks for Computing (Snowbird, UT)*.

Levitt, J. B., Lewis, D. A., Yoshioka, T., & Lund, J. S. (1993). Topography of pyramidal neuron intrinsic connections in macaque monkey prefrontal cortex (areas 9 & 46). *Journal of Comparative Neurology*, *338*, 360–376.

Linden, D. J. (1994). Long-term synaptic depression in the mammalian brain. *Neuron*, *12*, 457–472.

Linsker, R. (1992). Local synaptic learning rules suffice to maximize mutual information in a linear network. *Neural Computation*, *4*, 691–702.

Lisman, J. (1994). The CaM Kinase II hypothesis for the storage of synaptic memory. *Trends in neurosciences*, *17*, 406.

Lisman, J. E. (1989). A mechanism for the hebb and the anti-hebb processes underlying learning and memory. *Proc. Natl. Acad. Sci. USA*, *86*, 9574–9578.

Malenka, R. C., Lancaster, B., & Zucker, R. S. (1992). Temporal limits on the rise in postsynpatic calcium required for the induction of long-term potentiation. *Neuron*, *9*, 121–128.

Malenka, R. C., & Nicoll, R. A. (1993). NMDA receptor-dependent synaptic plasticity: Multiple forms and mechanisms. *Trends In Neurocience*, *16*, 521–527.

Mazzoni, P., Andersen, R. A., & Jordan, M. I. (1991). A more biologically plausible learning rule for neural networks. *Proc. Natl. Acad. Sci. USA*, *88*, 4433–4437.

McClelland, J. L. (1994). The interaction of nature and nurture in development: A parallel distributed processing perspective. In P. Bertelson, P. Eelen, & G. D'Ydewalle (Eds.), *Current advances in psychological science: Ongoing research* (pp. 57–88). Hillsdale, NJ: Erlbaum.

Movellan, J. R. (1990). Contrastive Hebbian learning in the continuous Hopfield model. In D. S. Touretzky, G. E. Hinton, & T. J. Sejnowski (Eds.), *Proceedings of the 1989 Connectionist Models Summer School* (pp. 10–17). San Mateo, CA: Morgan Kaufmann.

Mulkey, R. M., & Malenka, R. C. (1992). Mechanisims underlying induction of homosynaptic long-term depression in area CA1 of the hippocampus. *Science*, *9*, 967–975.

O'Reilly, R. C. (1996). *The leabra model of neural interactions and learning in the neocortex*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Perkel, D. J., Petrozzino, J. J., Nicoll, R. A., & Connor, J. A. (1993). The role of Ca^{2+} entry via synaptically activated NMDA receptors in the induction of long-term potentiation. *Neuron*, *11*, 817–823.

Peterson, C. (1991). Mean field theory neural networks for feature recognition, content addressable memory, and optimization. *Connection Science*, *3*, 3–33.

Peterson, C., & Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, *1*, 995–1019.

Peterson, C., & Hartman, E. (1989). Explorations of the mean field theory learning algorithm. *Neural Networks*, *2*, 475–494.

Pineda, F. J. (1987a). Generalization of backpropagation to recurrent and higher order neural networks. In D. Z. Anderson (Ed.), *Proceedings of IEEE Conference on Neural Information Processing Systems, Denver, CO* (pp. 602–611). New York: IEEE.

Pineda, F. J. (1987b). Generalization of backpropagation to recurrent neural networks. *Physical Review Letters*, *18*, 2229–2232.

Pineda, F. J. (1988). Dynamics and architecture for neural computation. *Journal of Complexity*, *4*, 216–245.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1988). *Numerical recipies in C: The art of scientific computing*. Cambridge: Cambridge University Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In (Rumelhart et al., 1986b), Chap. 8, pp. 318–362.

Rumelhart, D. E., McClelland, J. L., & PDP Research Group (Eds.). (1986b). *Parallel distributed processing. volume 1: Foundations*. Cambridge, MA: MIT Press.

Schultz, W., Apicella, P., & Ljungberg, T. (1993). Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task. *The Journal of Neuroscience*, *13*, 900–913.

Sutton, S., Braren, M., Zubin, J., & John, E. R. (1965). Evoked-potential correlates of stimulus uncertainty. *Science*, *150*, 1187–1188.

Tesauro, G. (1990). Neural models of classical conditioning: A theoretical viewpoint. In S. J. Hanson, & C. R. Olson (Eds.), *Connectionist modelling and brain function*. Cambridge, MA: MIT Press.

Zipser, D., & Andersen, R. A. (1988). A back propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, *331*, 679–684.

Zipser, D., & Rumelhart, D. E. (1990). Neurobiologcial significance of new learning models. In E. Schwartz (Ed.), *Computational neuroscience* (pp. 192–200). Cambridge, MA: MIT Press.