Equivalence of Backpropagation and Contrastive Hebbian Learning in a Layered Network

Xiaohui Xie
xhx@ai.mit.edu
Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

H. Sebastian Seung
seung@mit.edu
Howard Hughes Medical Institute and Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

Backpropagation and contrastive Hebbian learning are two methods of training networks with hidden neurons. Backpropagation computes an error signal for the output neurons and spreads it over the hidden neurons. Contrastive Hebbian learning involves clamping the output neurons at desired values and letting the effect spread through feedback connections over the entire network. To investigate the relationship between these two forms of learning, we consider a special case in which they are identical: a multilayer perceptron with linear output units, to which weak feedback connections have been added. In this case, the change in network state caused by clamping the output neurons turns out to be the same as the error signal spread by backpropagation, except for a scalar prefactor. This suggests that the functionality of backpropagation can be realized alternatively by a Hebbian-type learning algorithm, which is suitable for implementation in biological networks.

1 Introduction

Backpropagation and contrastive Hebbian learning (CHL) are two supervised learning algorithms for training networks with hidden neurons. They are of interest, because they are generally applicable to wide classes of network architectures. In backpropagation (Rumelhart, Hinton, & Williams, 1986b, 1986a), an error signal for the output neurons is computed and propagated back into the hidden neurons through a separate teacher network. Synaptic weights are updated based on the product between the error signal and network activities. CHL updates the synaptic weights based on the steady states of neurons in two different phases: one with the output neurons clamped to the desired values and the other with the output neurons free (Movellan, 1990; Baldi & Pineda, 1991). Clamping the output neurons causes the hidden neurons to change their activities, and this change constitutes the basis for the CHL update rule.

CHL was originally formulated for the Boltzmann machine (Ackley, Hinton, & Sejnowski, 1985) and was extended later to deterministic networks (Peterson & Anderson, 1987; Hinton, 1989), in which case it can be interpreted as a mean-field approximation of the Boltzmann machine learning algorithm. However, this interpretation is not necessary, and CHL can be formulated purely for deterministic networks (Movellan, 1990; Baldi & Pineda, 1991). Compared to backpropagation, CHL appears to be quite different. Backpropagation is typically implemented in feedforward networks, whereas CHL is implemented in networks with feedback. Backpropagation is an algorithm driven by error, whereas CHL is a Hebbian-type algorithm, with update rules based on the correlation of pre- and postsynaptic activities. CHL has been shown to be equivalent to backpropagation for networks with only a single layer (Movellan, 1990), and there has been some other work to relate CHL to the general framework of backpropagation (Hopfield, 1987; O'Reilly, 1996; Hinton & McClelland, 1988). However, a direct link between them for networks with hidden neurons has been lacking.

To investigate the relationship between these two algorithms, we consider a special network for which CHL and backpropagation are equivalent. This is a multilayer perceptron to which weak feedback connections have been added and with output neurons that are linear. The equivalence holds because in CHL, clamping the output neurons at their desired values causes the hidden neurons to change their activities, and this change turns out to be equal to the error signal spread by backpropagation, except for a scalar factor.

2 The Learning Algorithms

In this section, we describe the backpropagation and CHL algorithms. Backpropagation is in the standard form, implemented in a multilayer perceptron (Rumelhart et al., 1986b). CHL is formulated in a layered network with feedback connections between neighboring layers of neurons. It is an extension of the typical CHL algorithm formulated for recurrent symmetric networks (Movellan, 1990).

2.1 Backpropagation. Consider a multilayer perceptron with $L + 1$ layers of neurons and L layers of synaptic weights (see Figure 1A). The activities of the kth layer of neurons are denoted by the vector x_k, their biases by the vector b_k, and the synaptic connections from layer $k - 1$ to layer k by the matrix W_k. All neurons in the kth layer are assumed to have the same transfer function f_k, but this transfer function may vary from layer to layer. In particular, we will be interested in the case where f_L is linear, though the other transfer functions may be nonlinear. In the basic definition, f_k acts on a scalar and returns a scalar. However, we will generally use f_k to act on a

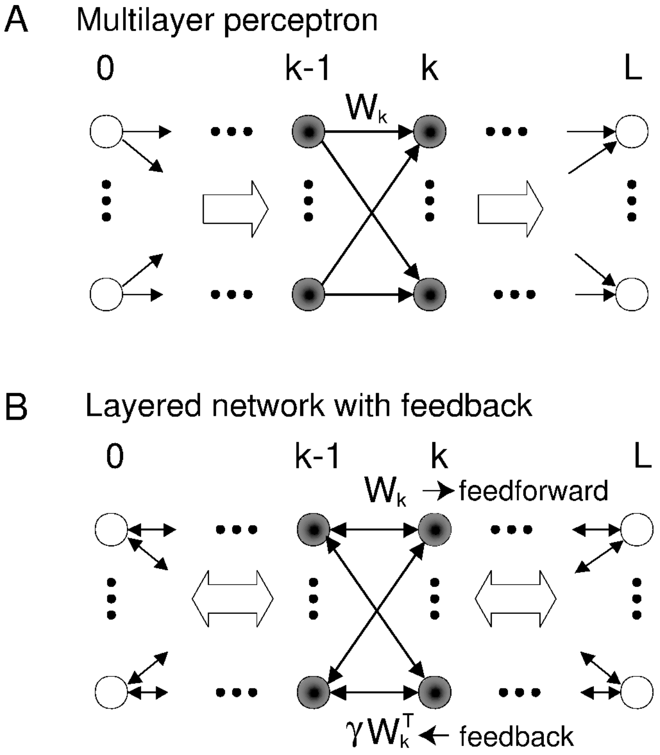

Figure 1: Diagram on the network structures of the (A) multilayer perceptron and the (B) layered network with feedback connections. Layer 0 is the input, layer L is the output, and the others are hidden layers. The forward connections are the same for both networks. In B, there exist feedback connections between neighboring layers of neurons.

vector, in which case it returns a vector, operating component by component by f_k. f_k' is the derivative of f_k with respect to its argument. Similar to f_k, when f_k' acts on a vector, it returns a vector as well, operated by f_k' in each component. We assume that f_k is monotonically increasing.

Backpropagation learning is implemented by repeating the following steps for each example in a training set of input-output pairs:

1. In the forward pass,

$$x_k = f_k(W_k x_{k-1} + b_k) \tag{2.1}$$

is evaluated for $k = 1$ to L, thereby mapping the input x_0 to the output x_L.

2. The desired output d of the network, provided by some teacher, is compared with the actual output x_L to compute an error signal,

$$y_L = D_L(d - x_L). \tag{2.2}$$

The matrix $D_k \equiv \text{diag}\{f'_k(W_k x_{k-1} + b_k)\}$ is defined by placing the components of the vector $f'_k(W_k x_{k-1} + b_k)$ in the diagonal entries of a matrix.

3. The error signal is propagated backward from the output layer by evaluating

$$y_{k-1} = D_{k-1} W_k^T y_k \tag{2.3}$$

for $k = L$ to 2.

4. The weight update

$$\Delta W_k = \eta y_k x_{k-1}^T \tag{2.4}$$

is made for $k = 1$ to L, where $\eta > 0$ is a parameter controlling the learning rate.

2.2 Contrastive Hebbian Learning. To formulate CHL, we consider a modified network in which, in addition to the feedforward connections from layer $k - 1$ to layer k, there are also feedback connections between neighboring layers (see Figure 1B). The feedback connections are assumed to be symmetric with the feedforward connections, except that they are multiplied by a positive factor γ . In other words, the matrix $\gamma\, W_k^T$ contains the feedback connections from layer k back to layer $k - 1$.

CHL is implemented by repeating the following steps for each example of the training set:

1. The input layer x_0 is held fixed, and the dynamical equations

$$\frac{dx_k}{dt} + x_k = f_k(W_k x_{k-1} + \gamma\, W_{k+1}^T x_{k+1} + b_k) \tag{2.5}$$

for $k = 1$ to L are run until convergence to a fixed point. The case $k = L$ is defined by setting $x_{L+1} = 0$ and $W_{L+1} = 0$. Convergence to a fixed point is guaranteed under rather general conditions, to be shown later. This is called the free state of the network and is denoted by $\check{x}_k$ for the kth layer neurons.

2. The anti-Hebbian update,

$$\Delta W_k = -\eta \gamma^{\,k-L} \check{x}_k \check{x}_{k-1}^T, \tag{2.6}$$

is made for $k = 1, \ldots, L$.

3. The output layer x_L is clamped at the desired value d, and the dynamical equation 2.5 for $k = 1$ to $L - 1$ is run until convergence to a fixed point. This is called the clamped state and is denoted by $\hat{x}_k$ for the kth layer neurons.
4. The Hebbian update,

   $$\Delta W_k = \eta\gamma^{k-L}\hat{x}_k\hat{x}_{k-1}^T, \tag{2.7}$$

   is made for $k = 1, \ldots, L$.

Alternatively, the weight updates could be combined and made after both clamped and free states are computed,

$$\Delta W_k = \eta\gamma^{k-L}(\hat{x}_k\hat{x}_{k-1}^T - \check{x}_k\check{x}_{k-1}^T). \tag{2.8}$$

This form is the one used in our analysis.

This version of CHL should look familiar to anyone who knows the conventional version, implemented in symmetric networks. It will be derived in section 4, but first we prove its equivalence to the backpropagation algorithm.

3 Equivalence in the Limit of Weak Feedback

Next, we prove that CHL in equation 2.8 is equivalent to the backpropagation algorithm in equation 2.4, provided that the feedback is sufficiently weak and the output neurons are linear.

In notation, x_k, $\hat{x}_k$, and $\check{x}_k$ represent the kth layer activities of the feedforward network, the clamped state, and the free state, respectively. We consider the case of weak feedback connections, $\gamma \ll 1$, and use $\approx$ to mean that terms of higher order in γ have been neglected and $\sim$ to denote the order.

The proof consists of the following four steps:

1. Show that the difference between the feedforward and free states is of order γ in each component,

   $$\delta\check{x}_k \equiv \check{x}_k - x_k \sim \gamma, \tag{3.1}$$

   for all $k = 1, \ldots, L$.
2. Show that in the limit of weak feedback, the difference between the clamped and free states satisfies the following iterative relationship,

   $$\delta x_k \equiv \hat{x}_k - \check{x}_k = \gamma D_k W_{k+1}^T \delta x_{k+1} + \mathcal{O}(\gamma^{L-k+1}), \tag{3.2}$$

   for $k = 1, \ldots, L - 1$, and $\delta x_L = d - \check{x}_L$.

3. Show that if the output neurons are linear, δx_k is related to the error signal in backpropagation through

$$\delta x_k = \gamma^{L-k} y_k + \mathcal{O}(\gamma^{L-k+1}). \tag{3.3}$$

4. Show that the CHL update can be approximated by

$$\Delta W_k = \eta y_k x_{k-1}^T + \mathcal{O}(\gamma). \tag{3.4}$$

In the CHL algorithm, clamping the output layer causes changes in the output neurons to spread backward to the hidden layers because of the feedback connections. Hence, the new clamped state differs from the free state over the entire network, including the hidden neurons. Equation 3.2 states that δx_k decays exponentially with distance from the output layer of the network. This is because the feedback is weak, so that δx_k is reduced from δx_{k+1} by a factor of γ.

Remarkably, as indicated in equation 3.3, the difference between the clamped and free states is equivalent to the error signal y_k computed in the backward pass of backpropagation, except for a factor of γ^{L-k}, when the output neurons are linear. Moreover, this factor annihilates the factor of γ^{k-L} in the CHL rule of equation 2.8, resulting in the update rule equation 3.4.

3.1 Proof. To prove the first step, we start from the steady-state equation of the free phase,

$$\check{x}_k = f_k(W_k \check{x}_{k-1} + b_k + \gamma W_{k+1}^T \check{x}_{k+1}), \tag{3.5}$$

for $k = 1, \ldots, L-1$. Subtracting this equation from equation 2.1 and performing Taylor expansion, we derive

$$\begin{aligned} \delta\check{x}_k &\equiv \check{x}_k - x_k & (3.6)\\ &= f_k(W_k \check{x}_{k-1} + b_k + \gamma W_{k+1}^T \check{x}_{k+1}) - f_k(W_k x_{k-1} + b_k) & (3.7)\\ &= D_k W_k \delta\check{x}_{k-1} + \gamma D_k W_{k+1}^T \check{x}_{k+1} + \mathcal{O}(\|W_k \delta\check{x}_{k-1} + \gamma W_{k+1}^T \check{x}_{k+1}\|^2) & (3.8) \end{aligned}$$

for all hidden layers, and $\delta\check{x}_L = D_L W_L \delta\check{x}_{L-1} + \mathcal{O}(\|W_L \delta\check{x}_{L-1}\|^2)$ for the output layer. In equation 3.7, the expansion is done around $W_k x_{k-1} + b_k$. Since the zeroth layer is fixed with the input, $\delta\check{x}_0 = 0$, under the above iterative relationships, we must have $\delta\check{x}_k \sim \gamma$ in each component, that is, $\delta\check{x}_k$ is in the order of γ, for all $k = 1, \ldots, L$.

To prove equation 3.2, we compare the fixed-point equations of the clamped and free states,

$$f^{-1}(\check{x}_k) = W_k \check{x}_{k-1} + b_k + \gamma W_{k+1}^T \check{x}_{k+1} \tag{3.9}$$

$$f^{-1}(\hat{x}_k) = W_k \hat{x}_{k-1} + b_k + \gamma W_{k+1}^T \hat{x}_{k+1}, \tag{3.10}$$

for $k = 1, \ldots, L-1$, where f^{-1} is the inverse function of f in each component. Subtract them, and perform Taylor expansion around $\check{x}_k$. Recall the definition of $\delta x_k \equiv \check{x}_k - \hat{x}_k$. We have

$$W_k \delta x_{k-1} + \gamma\, W_{k+1}^T \delta x_{k+1} = f^{-1}(\hat{x}_k) - f^{-1}(\check{x}_k) \tag{3.11}$$

$$= J_k \delta x_k + \mathcal{O}(\|\delta x_k\|^2), \tag{3.12}$$

where the matrix $J_k \equiv \mathrm{diag}\{\partial f^{-1}(\check{x}_k)/\partial \check{x}_k\}$. Since $\check{x}_k - x_k \sim \gamma$, to the leading order in γ , matrix J_k can be approximated by $J_k \approx \mathrm{diag}\{\partial f^{-1}(x_k)/\partial x_k\} = D_k^{-1}$, and $J_k \delta x_k = D_k^{-1} \delta x_k + \mathcal{O}(\gamma\, \|\delta x_k\|)$. Substituting this back to equation 3.12, we get

$$\delta x_k = D_k(W_k \delta x_{k-1} + \gamma\, W_{k+1}^T \delta x_{k+1}) + \mathcal{O}(\gamma\, \|\delta x_k\|) + \mathcal{O}(\|\delta x_k\|^2). \tag{3.13}$$

Let us start from $k = 1$. Since the input is fixed ($\delta x_0 = 0$), we have $\delta x_1 = \gamma\, D_1 W_2^T \delta x_2 + \mathcal{O}(\gamma\, \|\delta x_1\|)$, and therefore, $\delta x_1 = \gamma\, D_1 W_2^T \delta x_2 + \mathcal{O}(\gamma^{\,2} \|\delta x_2\|)$. Next, we check for $k = 2$, in which case $\delta x_2 = D_2 W_2 \delta x_1 + \gamma\, D_2 W_3^T \delta x_3 + \mathcal{O}(\gamma\, \|\delta x_2\|)$. Substituting δx_1, we have $\delta x_2 = \gamma\, D_2 W_3^T \delta x_3 + \mathcal{O}(\gamma\, \|\delta x_2\|)$, and therefore, $\delta x_2 = \gamma\, D_2 W_3^T \delta x_3 + \mathcal{O}(\gamma^{\,2} \|\delta x_3\|)$. Following this iteratively, we have

$$\delta x_k = \gamma\, D_k W_{k+1}^T \delta x_{k+1} + \mathcal{O}(\gamma^{\,2} \|\delta x_{k+1}\|), \tag{3.14}$$

for all $k = 1, \ldots, L-1$. Notice that $\delta x_L = d - \check{x}_L = d - x_L + \mathcal{O}(\gamma\,)$ is of order 1. Hence, $\delta x_{L-1} = \gamma\, D_{L-1} W_L^T \delta x_L + \mathcal{O}(\gamma^{\,2})$, and iteratively, we have

$$\delta x_k = \gamma\, D_k W_{k+1}^T \delta x_{k+1} + \mathcal{O}(\gamma^{\,L-k+1}), \tag{3.15}$$

for $k = 1, \ldots, L-1$. Therefore, equation 3.2 is proved. This equation indicates that $\delta x^k \sim \gamma\, \delta x^{k+1}$ and $\delta x_k \sim \gamma^{\,L-k}$ in each component.

If the output neurons are linear, then $y_L = \delta x_L + \mathcal{O}(\gamma\,)$. Consequently, $\delta x_k = \gamma^{\,L-k} y_k + \mathcal{O}(\gamma^{\,L-k+1})$ for all $k = 1, \ldots, L$.

Finally, the weight update rule of CHL follows:

$$\Delta W_k = \eta\gamma^{\,k-L}(\hat{x}_k \hat{x}_{k-1}^T - \check{x}_k \check{x}_{k-1}^T) \tag{3.16}$$

$$= \eta\gamma^{\,k-L} \delta x_k \check{x}_{k-1}^T + \eta\gamma^{\,k-L} \check{x}_k \delta x_{k-1}^T + \eta\gamma^{\,k-L} \delta x_k \delta x_{k-1}^T \tag{3.17}$$

$$= \eta\gamma^{\,k-L} \delta x_k \check{x}_{k-1}^T + \mathcal{O}(\gamma\,) \tag{3.18}$$

$$= \eta y_k x_{k-1}^T + \mathcal{O}(\gamma\,). \tag{3.19}$$

The last approximation is made because $\check{x}_{k-1} - x_{k-1} \sim \gamma$. This result shows that the CHL algorithm in the layered network with linear output neurons is identical to the backpropagation as $\gamma\, \to 0$.

Remark. For nonlinear output neurons, δx_L of CHL is different from y_L in equation 2.2 computed in backpropagation. However, they are within 90 degrees if viewed as vectors. Moreover, if the activation function for the output neurons is taken to be the sigmoidal function, that is, $f_L(z) = 1/(1+\exp(-z))$, the CHL algorithm is equivalent to backpropagation based on the cost function,

$$-d^T \log(x_L) - (1-d)^T \log(1-x_L), \tag{3.20}$$

since in this case, $y_L = d - x_L$. In the above, function log acts on a vector and returns a vector as well.

4 Contrastive Function

The CHL algorithm stated in section 2.2 can be shown to perform gradient descent on a contrastive function that is defined as the difference of the network's Lyapunov functions between clamped and free states (Movellan, 1990; Baldi & Pineda, 1991).

Suppose $E(x)$ is a Lyapunov function of the dynamics in equation 2.5. Construct the contrastive function $C(W) \equiv E(\hat{x}) - E(\check{x})$, where $\hat{x}$ and $\check{x}$ are steady states of the whole network in the clamped and free phase, respectively, and $W \equiv \{W_1, \ldots, W_L\}$. For simplicity, let us first assume that $E(x)$ has a unique global minimum in the range of x and no local minima. According to the definition of Lyapunov functions, $\check{x}$ is the global minimum of E and so is $\hat{x}$, but under the extra constraints that the output neurons are clamped at d. Therefore, $C(W) = E(\hat{x}) - E(\check{x}) \geq 0$ and achieves zero if and only if $\check{x} = \hat{x}$, that is, when the output neurons reach the desired values. Performing gradient descent on $C(W)$ leads to the CHL algorithm. On the other hand, if $E(x)$ does not have a unique minimum, $\check{x}$ and $\hat{x}$ may be only local minima. However, the above discussion still holds, provided that $\hat{x}$ is in the basin of attraction of $\check{x}$ under the free phase dynamics. This imposes some constraints on how to reset the initial state of the network after each phase. One strategy is to let the clamped phase settle to the steady state first and then run the free phase without resetting hidden neurons. This will guarantee that $C(W)$ is always nonnegative and constitutes a proper cost function.

Next, we introduce a Lyapunov function for the network dynamics in equation 2.5,

$$E(x) = \sum_{k=1}^{L} \gamma^{\,k-L} [1^T \bar{F}_k(x_k) - x_k^T W_k x_{k-1} - b_k^T x_k], \tag{4.1}$$

where function $\bar{F}_k$ is defined so that $\bar{F}_k'(x) = f_k^{-1}(x)$. $x \equiv \{x_1, \ldots, x_L\}$ represents the states of all layers of the network. Equation 4.1 is extended from

Lyapunov functions previously introduced for recurrent networks (Hopfield, 1984; Cohen & Grossberg, 1983).

For $E(x)$ to be a Lyapunov function, it must be nonincreasing under the dynamics equation 2.5. This can be shown by

$$\dot{E} = \sum_{k=1}^{L} \left(\frac{\partial E}{x_k}\right)^T \dot{x}_k \tag{4.2}$$

$$= \sum_{k=1}^{L} \gamma^{\,k-L} [\boldsymbol{f}_k^{-1}(x_k) - W_k x_{k-1} - \gamma\, W_{k+1}^T x_{k+1} - b_k]^T \dot{x}_k \tag{4.3}$$

$$= \sum_{k=1}^{L} -\gamma^{\,k-L} [\boldsymbol{f}_k^{-1}(x_k) - \boldsymbol{f}_k^{-1}(\dot{x}_k + x_k)]^T [x_k - (\dot{x}_k + x_k)] \tag{4.4}$$

$$\leq 0, \tag{4.5}$$

where the last inequality holds because f_k is monotonic as we have assumed. Therefore, $E(x)$ is nonincreasing following the dynamics and stationary if and only if at the fixed points. Furthermore, with appropriately chosen f_k, such as sigmoid functions, $E(x)$ is also bounded below, in which case $E(x)$ is a Lyapunov function.

Given the Lyapunov function, we can form the contrastive function $C(W)$ and derive the gradient-descent algorithm on C accordingly.

The derivative of $E(\hat{x})$ with respect to W_k is

$$\frac{dE(\hat{x})}{dW_k} = \frac{\partial E}{\partial W_k} + \sum_k \frac{\partial E}{\partial \hat{x}_k} \frac{\partial \hat{x}_k}{\partial W_k} \tag{4.6}$$

$$= \frac{\partial E}{\partial W_k} \tag{4.7}$$

$$= -\gamma^{\,k-L} \hat{x}_k \hat{x}_{k-1}^T, \tag{4.8}$$

where the second equality holds because $\partial E / \partial \hat{x}_k = 0$ for all k at the steady states. Similarly, we derive

$$\frac{dE(\check{x})}{dW_k} = -\gamma^{\,k-L} \check{x}_k \check{x}_{k-1}^T. \tag{4.9}$$

Combining equations 4.8 and 4.9, we find the derivative of $C(W)$ with respect to W_k shall read

$$\frac{dC}{dW_k} = \frac{dE(\hat{x})}{dW_k} - \frac{dE(\check{x})}{dW_k} = -\gamma^{\,k-L} (\hat{x}_k \hat{x}_{k-1}^T - \check{x}_k \check{x}_{k-1}^T). \tag{4.10}$$

With a suitable learning rate, gradient descent on $C(W)$ leads to the CHL algorithm in equation 2.8.

5 Equivalence of Cost Functions

In section 3, we proved that the CHL algorithm in the layered network with linear output neurons is equivalent to backpropagation in the weak feedback limit. Since both algorithms perform gradient descent on some cost function, the equivalence in the update rule implies that their cost functions should be equal, up to a multiplicative or an additive constant difference. Next, we demonstrate this directly by comparing the cost functions of these two algorithms.

The backpropagation learning algorithm is gradient descent on the squared difference, $\|d - x_L\|^2/2$, between the desired and actual outputs of the network.

For the CHL algorithm, the cost function is the difference of Lyapunov functions between the clamped and free states, as shown in the previous section. After reordering, it can be written as

$$C = \sum_{k=1}^{L} \gamma^{k-L} [1^T(\bar{F}_k(\hat{x}_k) - \bar{F}_k(\check{x}_k)) - \delta x_k^T (W_k \check{x}_{k-1} + b_k) - \delta x_{k-1}^T W_k^T \hat{x}_k]. \quad (5.1)$$

Recall that $\delta x_k \sim \gamma^{L-k}$. Therefore, the δx_k term above multiplied by the factor γ^{k-L} is of order 1, whereas the δx_{k-1} multiplied by the same factor is of order γ, and thus can be neglected in the leading-order approximation. After this, we get

$$C = \sum_{k=1}^{L} \gamma^{k-L} [1^T(\bar{F}_k(\hat{x}_k) - \bar{F}_k(\check{x}_k)) - \delta x_k^T (W_k \check{x}_{k-1} + b_k)] + \mathcal{O}(\gamma). \quad (5.2)$$

If the output neurons are linear ($f_L(x) = x$), then $\bar{F}_L(x) = x^T x/2$ and $W_L \check{x}_{L-1} + b_L = \check{x}_L$. Substituting them into C and separating terms of the output and hidden layers, we derive

$$\begin{aligned} C &= \frac{1}{2}(\hat{x}_L^T \hat{x}_L - \check{x}_L^T \check{x}_L - \delta x_L^T \check{x}_L) \\ &\quad + \sum_{k=1}^{L-1} \gamma^{k-L} \delta x_k^T [f_k^{-1}(\check{x}_k) - W_k \check{x}_{k-1} - b_k] + \mathcal{O}(\gamma) \\ &= \frac{1}{2}\|d - x_L\|^2 + \mathcal{O}(\gamma), \end{aligned} \quad (5.3)$$

where the second term with the sum vanishes because of the fixed-point equations.

In conclusion, to the leading order in γ, the contrastive function in CHL is equal to the squared error cost function of backpropagation. The demonstration on the equality of the cost functions provides another perspective on the equivalence between these two forms of learning algorithms.

So far, we have always assumed that the output neurons are linear. If this is not true, how different is the cost function of CHL from that of backpropagation? Repeating the above derivation, we get the cost function of CHL for nonlinear output neurons,

$$C = 1^T\bar{F}_L(\hat{x}_L) - 1^T\bar{F}_L(\check{x}_L) - \delta x_L^T \boldsymbol{f}_L^{-1}(\check{x}_L) + \mathcal{O}(\gamma) \tag{5.4}$$

$$= -1^T\bar{F}_L(x_L) - \delta x_L^T \boldsymbol{f}_L^{-1}(x_L) + 1^T\bar{F}_L(d) + \mathcal{O}(\gamma). \tag{5.5}$$

With sigmoidal-type activation function for output neurons, $\bar{F}_L(z) = x\log(z) + (1-z)\log(1-z)$. Substituting this into equation 5.5, we find

$$C = -d^T\log(x_L) - (1-d)^T\log(1-x_L) + 1^T\bar{F}_L(d) + \mathcal{O}(\gamma), \tag{5.6}$$

which is the same as the cost function in equation 3.20 in the small γ limit, except a constant difference.

6 Simulation

In this section, we use the backpropagation and the CHL algorithm to train a 784-10-10 three-layer network to perform handwritten digit recognition. The data we use are abridged from the MNIST database containing 5000 training examples and 1000 testing examples.

We use the sigmoidal function $f_k(x) = 1/(1+\exp(-x))$ for the hidden and output layers. The backpropagation algorithm is based on the cost function in equation 3.20. We simulate the CHL algorithm in the layered network with three different feedback strengths: $\gamma = 0.05$, 0.1, and 0.5.

The label of an input example is determined by the index of the largest output. After each epoch of on-line training, the classification and squared error for both training and testing examples are computed and plotted in Figure 2. The classification error is defined as the percentage of examples classified incorrectly. The squared error is defined as the mean squared difference between the actual and desired output for all training or testing examples. The desired output is 1 on the output neuron whose index is the same as the label and zero on the output neurons otherwise.

The learning curve of CHL algorithm is very similar to those of backpropagation for $\gamma = 0.05$ and 0.1, and it deviates from the learning curve of backpropagation for $\gamma = 0.5$ (see Figure 2). In the simulation, we find the overall simulation time of the CHL algorithm is not significantly longer than that of the backpropagation. This is because the layered network tends to converge to a steady state fast in the case of weak feedback.

7 Discussion

We have shown that backpropagation in multilayer perceptrons can be equivalently implemented by the CHL algorithm if weak feedback is added.

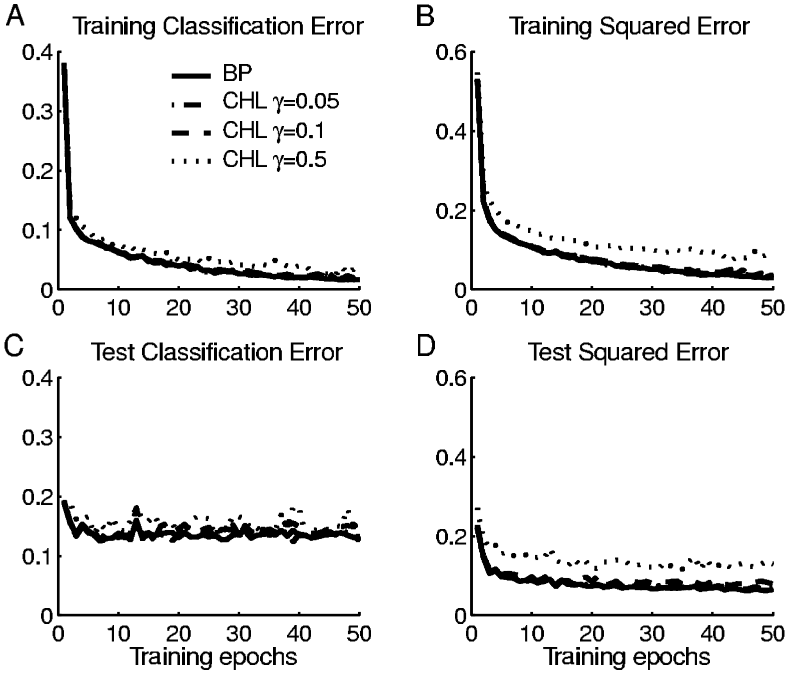

Figure 2: Comparison of performance between the backpropagation and the CHL algorithm. The two algorithms are used to train a 784-10-10 three-layer network to perform handwritten digit recognition. The CHL algorithm is used to train the three-layer network with feedback $\gamma = 0.05, 0.1$, and 0.5 added. (A, B) Classification and squared errors for training examples. (C, D) Test examples.

This is demonstrated from two perspectives: evaluating the two algorithms directly and comparing their cost functions. The essence behind this equivalence is that CHL effectively extracts the error signal of backpropagation from the difference between the clamped and free steady states.

The equivalence between CHL and backpropagation in layered networks holds in the limit of weak feedback, which is true mathematically. This, however, does not imply that in engineering problem solving, we should substitute CHL for backpropagation to train neural networks. This is because in networks with many hidden layers, the difference between the clamped and free states in the first few layers would become very small in the limit of weak feedback, and therefore CHL will not be robust against noise during training. In practice, when CHL is used for training the layered networks with many hidden layers, the feedback strength should not be chosen to be too small, in which case the approximation of CHL to backpropagation algorithm will be inaccurate.

The investigation on the relationship between backpropagation and CHL is motivated by research looking for biologically plausible learning algorithms. It is believed by many that backpropagation is not biologically realistic. However, in an interesting study on coordinate transform in posterior parietal cortex of monkeys, Zipser and Anderson (1988) show that hidden neurons in a network model trained by backpropagation share very similar properties to real neurons recorded from that area. This work prompted the search for a learning algorithm, which has similar functionality as backpropagation (Crick, 1989; Mazzoni, Andersen, & Jordan, 1991) and at the same time is biologically plausible. CHL is a Hebbian-type learning algorithm, relying on only pre- and postsynaptic activities. The implementation of backpropagation equivalently by CHL suggests that CHL could be a possible solution to this problem.

Mazzoni et al. (1991) also proposed a biologically plausible learning rule as an alternative to backpropagation. Their algorithm is a reinforcement-type learning algorithm, which is usually slow, has large variance, and depends on global signals. In contrast, the CHL algorithm is a deterministic algorithm, which could be much faster than reinforcement learning algorithms. However, a disadvantage of CHL is its dependence on special network structures, such as the layered network in our case. Whether either algorithm is used by biological systems is an important question that needs further investigation in experiments and theory.

Acknowledgments

We acknowledge helpful discussions with J. J. Hopfield and S. Roweis. We thank J. Movellan for suggesting the error function for nonlinear output neurons.

References

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

Baldi, P., & Pineda, F. (1991). Contrastive learning and neural oscillator. *Neural Computation, 3*, 526–545.

Cohen, M. A., & Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Trans. on Systems, Man, and Cybernetics, 13*, 815–826.

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

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

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

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.

Hopfield, J. J. (1987). Learning algorithms and probability distributions in feed-forward and feed-back networks. *Proc. Natl. Acad. Sci. USA, 84*, 8429–8433.

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.

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

O'Reilly, R. (1996). Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation, 8*, 895–938.

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

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1: Foundations* (pp. 318–362). Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back–propagating errors. *Nature, 323*, 533–536.

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

Received January 25, 2002; accepted August 1, 2002.