Contrastive Hebbian Learning in the Continuous Hopfield Model

Javier R. Movellan
Department of Psychology
Carnegie Mellon University
Pittsburgh, Pa 15213
email: jm2z+@andrew.cmu.edu

Abstract

This paper shows that contrastive Hebbian, the algorithm used in mean field learning, can be applied to any continuous Hopfield model. This implies that non-logistic activation functions as well as self connections are allowed. Contrary to previous approaches, the learning algorithm is derived without considering it a mean field approximation to Boltzmann machine learning. The paper includes a discussion of the conditions under which the function that contrastive Hebbian minimizes can be considered a proper error function, and an analysis of five different training regimes. An appendix provides complete demonstrations and specific instructions on how to implement contrastive Hebbian learning in interactive activation and competition models (a convenient version of the continuous Hopfield model).

1 INTRODUCTION

In this paper we refer to interactive activation networks as the class of neural network models which have differentiable, bounded, strictly increasing activation functions, symmetric recurrent connections, and for which we are interested in the equilibrium activation states rather than the trajectories to achieve them. This type of network is also known as the continuous Hopfield model [6]. Some of the benefits of interactive activation networks as opposed to feed-forward networks are their completion properties, flexibility in the treatment of units as inputs or outputs, appropriateness for solving soft-constraint satisfaction problems, suitability for modeling cognitive processes [9], and the fact that they have an associated energy function that may be applied in pattern recognition problems [13]. Contrastive Hebbian Learning[7] (CHL), which is a generalization of the Hebbian rule, updates the weights proportionally to the difference in the crossproducts of activations in a clamped and a free running phase. This modification of the Hebbian learning, first applied by Hopfield to improve the storage capacity of discrete content addressable memories without hidden units [5], appears in the Boltzmann learning algorithm [1] and its mean field approximation [11][3]. This paper shows that Hinton's observation that CHL depends on a performance measure [3] can be generalized to any case of the continuous Hopfield model. Contrary to previous approaches, the derivations do not presume the existence of Boltzmann machines approximated with mean field networks. The paper includes a discussion of the conditions under which the function that CHL minimizes can be considered a proper error function, an analysis of undesirable effects that may occur in CHL learning, and a classification of training regimes that minimize these effects.

The paper is divided in two sections and one appendix. Section 1 describes the dynamics of the activations in interactive networks. Section 2 shows how to modify the weights for the stable states of the network to reproduce desired patterns of activations. The original contribution in this paper are:

- To show that CHL works with any continuous Hopfield network and thus that self-connections as well as non-logistic activation functions are allowed.
- To show that the principles involved in the mean field learning algorithm can be derived independently of the Boltzmann Machine.
- To show that except for the case where there are no hidden units, the function that CHL minimizes is not a proper error function but that in practice there are training regimes that make it work as such.

For completeness we present some of the classical proofs provided in Hopfield [6]. The appendix contains mathematical details and specific comments on how to implement Contrastive Hebbian Learning in interactive networks.

2 STABILITY OF ACTIVATIONS

Since interactive networks have recurrent paths, it is in principle possible that their activations never stabilize. Fortunately, if some simple conditions investigated by Hopfield [6] are met, we can guarantee that the activations will settle, and that at equilibrium they will be at a minimum of an *Energy* function.

Let the activation vector $\mathbf{a}^T = [a_1, ...a_n]$, be regulated by bounded, monotonically increasing, differentiable activation functions $f_i(.)$. Let $\mathbf{W} = [\mathbf{w}_1, ...\mathbf{w}_n]$ be the matrix of connections, where the $\mathbf{w}_i^T = [w_{1,i}, ...w_{n,i}]$ are bounded, fan-in weight row vectors. Let $d()/dt$ be derivatives with respect to time, $net_i = \mathbf{a}_i^t\ \mathbf{w}_i$, and $rest = f(0)$. Define a continuous Hopfield Energy function [6]

$$F = E + S \tag{1}$$

where

$$E = -\frac{1}{2}\,\mathbf{a}^t\mathbf{W}\,\mathbf{a} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i w_{ij} a_j \tag{2}$$

and

$$S = \sum_{i=1}^{n}\int_{rest}^{a_i} f_i^{-1}(a)\,da \tag{3}$$

It can be shown that E, which reflects the constraints imposed by the weights in the network, tends to drive activations to extreme values [1]. On the other hand, S is a penalty function that tends to drive the activations to a central value (the resting point). In principle we are interested in activation states that minimize E for they are maximally harmonious with the information encoded in the weights. As we will study later, varying the relative importance of E vs. S as the activations settle may help achieve maximally harmonious (minimum E) states. In the appendix it is shown that if the activation functions are the standard (0-1) logistic, F becomes the Helmholtz free energy function as defined in [3].

Hopfield [6] showed that if the network is governed by the set of differential equations

$$\frac{d\,f_i^{-1}(a_i)}{dt} = \lambda\,(-f_i^{-1}(a_i) + net_i)\ ;\ \ i = 1...n \tag{4}$$

and the weights are symmetric, the activations stabilize in a minimum of F. For completeness, I present in the appendix a version of Hopfield's proof and show that stability in a global minimum can also be achieved with the following equation, typically used in interactive activation networks [8][9] [10]

$$\frac{d\,a_i}{dt} = \lambda\ ((-a_i + f_i(net_i)) \tag{5}$$

Notice that if we apply either equation 4 or 5, on equilibrium (when the derivatives are zero),

$$f_k^{-1}(\breve{a}_i) = \breve{net}_i \tag{6}$$

where ($\breve{}$) represents equilibrium. These properties will be used to derive the learning algorithm in the next section.

3 CONTRASTIVE LEARNING

Learning is viewed as the modification of connections between units so that the stable states of the network reproduce desired patterns of activations. We will see that CHL minimizes a *contrastive function J* [2] and then we will discuss the conditions under which minimization of J guarantees learning.

Define a pattern p as the pair $p = \{\mathbf{a}^{I(+)}, \mathbf{a}^{O(+)}\}$, where I stands for input set, O for output set, and (+) indicates that the activation of these units are fixed by the pattern. In connectionist terms $\mathbf{a}^{O(+)}$ is the teacher vector. We will say that p has been learned if clamping the input units to $\mathbf{a}^{I(+)}$ then

$$\breve{\mathbf{a}}^{O(-)} = \mathbf{a}^{O(+)} \tag{7}$$

where $\breve{\mathbf{a}}^{O(-)}$ denotes the activation of the output set when inputs are clamped and outputs are free. Note that in this form of supervised learning we are only interested in the outputs obtained after equilibrium is achieved and not in the trajectories followed to equilibrium.

Define the *contrastive function* J as

$$J = \breve{F}^{(+)} - \breve{F}^{(-)} \tag{8}$$

where $\breve{F}^{(+)}$ and $\breve{F}^{(-)}$ respectively are the values of the energy functions at equilibrium when the inputs and outputs are clamped (+), and when the inputs are clamped and outputs are free (-). Notice that $\breve{F}^{(-)}$ has the same free parameters that $F^{(+)}$, the activations of the hidden units, plus some additional free parameters, the activations of the output units. Assume that, over the working region of activation states, F has a unique minimum.[3]. Thus, since the minimum is unique $\breve{F}^{(+)} \geq \breve{F}^{(-)}$, and if $J = 0$, then

[1] If self connections are allowed, minima in E may also occur for intermediate activation values.

[2] Based on different arguments, Hinton [3] showed that CHL minimizes the equivalent of J when the activation is logistic.

[3] This assumption, which is shared with Boltzmann learning, Mean Field Learning, and the Almeida-Pineda algorithm is discussed in section 4.

$\breve{\mathbf{a}}^{O(-)} = \mathbf{a}^{O(+)}$. This makes J a potential candidate for learning by gradient descent on weight space. In particular, following the derivations detailed in the appendix, we have

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\breve{a}_i\breve{a}_j - \sum_{k=1}^{n} \breve{net}_k(\frac{\partial \breve{a}_k}{\partial w_{ij}}) \; ; \; i \neq j \quad (9)$$

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\frac{1}{2}\breve{a}_i^2 - \sum_{k=1}^{n} \breve{net}_k(\frac{\partial \breve{a}_k}{\partial w_{ij}}) \; ; \; i = j \quad (10)$$

$$\frac{\partial \breve{S}}{\partial w_{ij}} = \sum_{k=1}^{n} f_k^{-1}(\breve{a}_k)\frac{\partial \breve{a}_k}{\partial w_{ij}} \quad (11)$$

And following equation 6 it is easy to see that

$$\frac{\partial \breve{F}}{\partial w_{ij}} = -\breve{a}_i\breve{a}_j \; ; \; i \neq j \quad (12)$$

$$\frac{\partial \breve{F}}{\partial w_{ij}} = -\frac{1}{2}\breve{a}_i\breve{a}_j \; ; \; i = j \quad (13)$$

$$(14)$$

making

$$\frac{\partial \breve{J}}{\partial w_{ij}} \propto \breve{a}_i^{(-)}\breve{a}_j^{(-)} - \breve{a}_i^{(+)}\breve{a}_j^{(+)} \quad (15)$$

which shows that the contrastive Hebbian learning rule

$$\Delta w_{ij} \propto \breve{a}_i^{(+)}\breve{a}_j^{(+)} - \breve{a}_i^{(-)}\breve{a}_j^{(-)} \quad (16)$$

descends in the J function.

4 DISCUSSION

We have seen that CHL minimizes the contrastive function

$$J = \breve{F}^{(+)} - \breve{F}^{(-)} \quad (17)$$

so that after each learning step, the difference in energy at equilibrium between the clamped and free states becomes smaller. At this point we will study the conditions under which J can be considered a proper error function.

An important property of error functions is that they decrease as the difference between the obtained and the desired states decreases. CHL guarantees that the difference between energies in the clamped and free phases will become smaller but as we will see this does not always guarantee that the difference between the clamped and free state activations will decrease. If both $F^{(+)}$ and $F^{(-)}$ have a unique minimum (e.g. when there are no hidden units) and since in the (-) phase there are more free parameters to minimize F than in the (+) phase it follows that J cannot be smaller than zero and that when $J = 0$ the activations in the free and clamped phase are equal. In this case, J is a proper error function. In the appendix it is shown that if there are no hidden units CHL is in fact equivalent to backpropagation learning.

However, if there are multiple minima in F, we can no longer guarantee that a local minima in the free phase will have lower energy than a local minima in the clamped phase. One way to avoid this problem is to use training regimes that maximize the probability that the activations in both the free-running and the clamped phase equilibrate in the same region of attraction of $F^{(-)}$ space. A way to visualize how CHL works is to imagine the energy surface over activation space as a membrane with several minima. CHL pushes down the minimum corresponding to the clamped case and pulls up the stable states for the free phases. If both stable states are in the same attractor, after several learning trials, both minima converge. What follows is a brief analysis of five different training regimes:

- **Case 1:** *Activations are reset to random numbers after each learning phase.* In this case, the starting points for the clamped and free phase are different and, very likely the stable states will also be apart. Under this condition, CHL learning does not work well.
- **Case 2:** *First settle for the clamped phase, and then, without resetting activations, free the output units and settle again.* This procedure guarantees that $\breve{F}^{(+)} \geq \breve{F}^{(-)}$ and that when $\breve{F}^{(+)} = \breve{F}^{(-)}$ the activations on the free and clamped phases are the same. Gradient descent on J assures that when the minima for the clamped phase is achieved, if the output unit activations are free, they will not change. This form of learning, which can be used for recognition of familiar patterns, is very rapidly achieved with CHL (it just takes about 3 trials to "recognize" the XOR or the 4-3-4 encoder patterns). Unfortunately, if we just clamp the input units without information about the teachers, in general the activations will not converge to the desired minima.
- **Case 3:** *Settle during the free phase and then, without resetting activations, clamp the output units and resettle again.*

  This scheme minimizes the probability that the clamped and free phases end up in different regions of attraction. In general this procedure works well and achieves learning speeds comparable to backpropagation. There are two phenomena though that sometimes occur [3]. Occasionally the network may settle in a different attractor than the one to which it had converged in previous trials. This may result in a sudden change in activations and and a temporary "unlearning" of the weights. In figure 1 it can be seen a typical

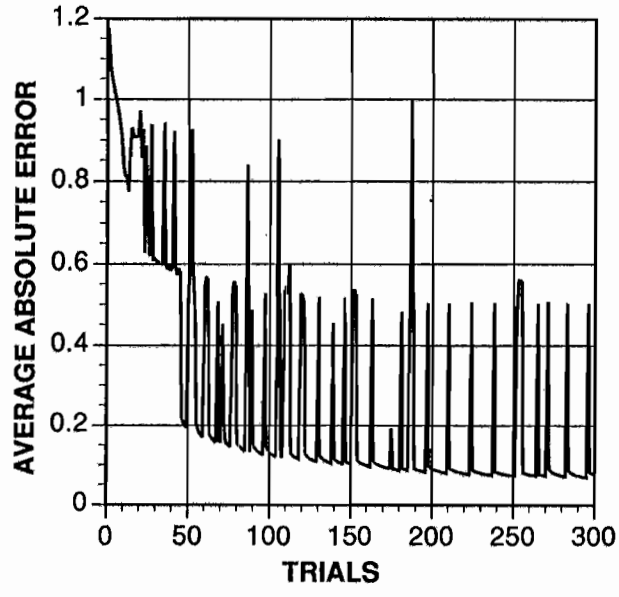

Figure 1: *Typical learning curve for the XOR problem.*

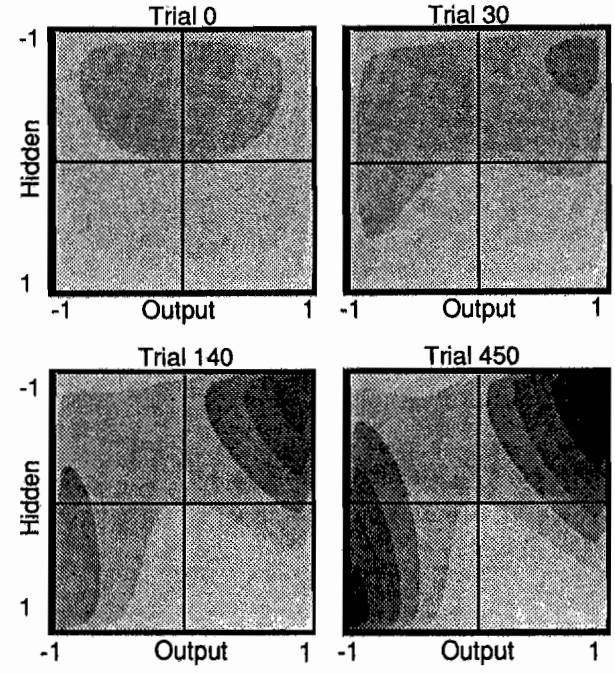

Figure 2: *Contour curves of the free energy function as learning progresses. Note that although initially there is a unique minimum, eventually an spurious local minimum is generated.*

learning curve with these temporary unlearning of the patterns. It is our experience that as learning progresses these sudden jumps to other minima tend to diminish. Another problem is that if the the clamped and free phases stabilize in different regions, the energy of the clamped phase may become lower than the energy in the free phase. If this happens, learning usually deteriorates, making it advisable to start with a different set of weights. Figure 2 shows the energy functions generated after training a simple network, with an input unit, a hidden unit and an output unit, to learn the 1-1-1 encoder problem. It can be seen that as learning progresses a minimum is created at the desired state (output unit activation =1) but that a spurious local minimum is also created. If in some trial the activations equilibrate in that local minimum abrupt distortions in the learning curve and temporary unlearning of the desired activations may occur [4].

- **Case 4:** *Sharpening Schedules* [5].

  As we previously mentioned if there is a unique global minimum of $\breve{F}^{(-)}$ and of $\breve{F}^{(+)}$, and the activations settle into this minimum then the contrastive function J can be considered a proper error measure. One way of decreasing the likelihood of settling into spurious local minima is the use of *sharpening schedules.* Sharpening schedules modify the *gain* or *sharpness* of the activation functions as the settling of the activations proceeds. Usually we start with low gain (flat activation functions) and progressively increase it. The rational for this procedure is as follows: Decreasing the sharpness of the activation functions is equivalent to weighting more heavily the S part of the free energy as defined in equation 1 [6]. Figure 3 shows the effect of sharpening on the error function learned for the 1-1-1 encoder problem mentioned in Case 3. For large decay values (low gain) the spurious local minima dissapear. Ideally, we start settling the activations with appropriately large decay values that get rid of the spurious local minima but allow the global minimum to survive. Then we let the activations settle towards this global minimum and slowly guide them away from rest values by progressively decreasing decay (increasing gain). Peterson and Anderson [11] and Peterson and Hartman[12] call this procedure *annealing* for it is a mean field approximation to annealing schedules of discrete Boltzmann machines. I have decided to use the term *sharpening schedules* as defined in [2] to clarify the fact that sharpening does not have anything to do with increasing the randomness of the network as one would expect of annealing schedules

[4] Since the networks we are considered so far are deterministic, the settling state is determined by the final state. However, slight weight modifications made by the learning algorithm may be sufficient to make the activations settle in completely different attractors

[5] I thank Conrad Galland for showing me the role that sharpening schedules play in contrastive Hebbian learning.

[6] In the interactive activation and competition model [10] this is controlled by a the decay parameter. If logistic activations are used, sharpening is achieved by controlling the gain of the activation functions.

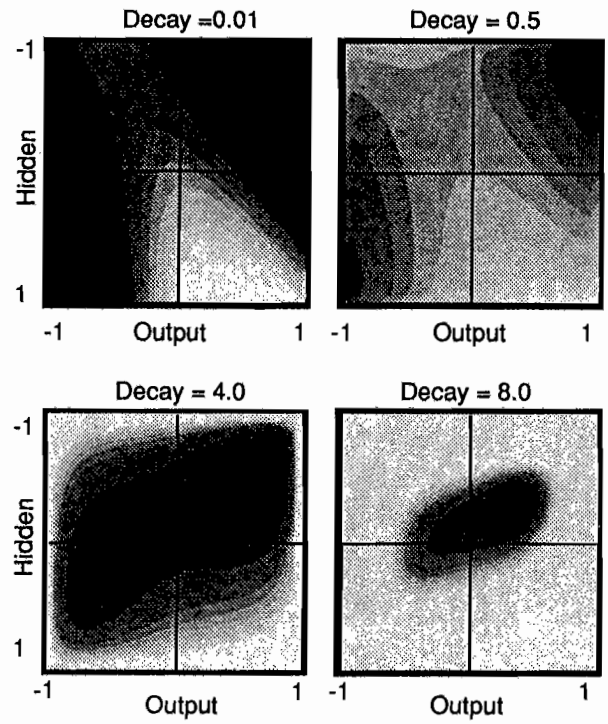

Figure 3: *The energy function for different degrees of sharpening. In this case sharpening is controlled by a decay parameter. Note how for low sharpening levels spurious minima disappear*

[7]. Although sharpening schedules are not strictly necessary for CHL to work, there are reasons to believe that they improve learning performance.

- **Case 5:** *Annealing schedules: In search of the continuous Boltzmann machine.* Annealing schedules are another method for avoiding spurious minima. Annealing schedules progressively decrease the randomness of the activations as settling progresses. This may be achieved in interactive networks by injecting some form of noise (e.g. logistic noise) to the net input of each unit. The standard deviation of the noise distribution plays a similar effect to the temperature parameter in discrete Boltzmann machines. Case 1 in this discussion can be viewed as a particular case of annealing schedule in which the standard deviation of noise is very large for the initial cycle (producing a random starting point) and then goes to zero on the next cycle (making the network deterministic). Using slowly decreasing annealing schedules may improve the likelihood of settling in the best minimum and avoiding spurious ones. It can be shown that this settling method defines a continuous state hidden Markov model (a diffussion process). The activation settling algorithm can be seen as an approximation to gradient descent of the expected value of the Hopfield energy function F. The appropriate learning algorith should use the difference in the equilibrium expected values of the activation crossproducts in the clamped and free running phase.

A nice property shown in the appendix is that if logistic noise is added to the net input, the limiting behavior of interactive networks as the sharpness of the activation functions increases is given by the discrete Boltzmann machine. This property may be used to implement both Boltzmann machines and continuous interactive networks with the same program.

In spite of its problems and the fact that more research is needed before using it for large scale problems, CHL is a promising, simple to implement learning method that works for a wide variety of interactive architectures. In Table 1 appear the results of simulations using CHL with the activation update rule of the Interactive Activation and Competition model. [10].

[7]The term "sharpening" is not without problems either for it conceals the fact that sharpening is a mean field approximation to true annealing. Another possible term is "mean field annealing" but this may be confused with the true annealing method as discussed in case 5.

5 APPENDIX

5.1 S AND THE MEAN FIELD ENTROPY TERM

The entropy term of the energy function proposed by Hinton [3] for the Mean Field Algorithm is

$$-\sum_{i=1}^{n}\left(a_i\, log a_i + (1-a_1)\, log\,(1-a_i)\right). \qquad (18)$$

If we use the standard logistic (0-1) activation function,

$$y = \frac{1}{1+e^{-x/T}} \qquad (19)$$

whose inverse is

$$x = T\, log\left(\frac{y}{1-y}\right) \qquad (20)$$

then

$$\sum_{i=1}^{n}\int_{0.5}^{y} T\, log\left(\frac{y}{1-y}\right)\, dy \qquad (21)$$

$$= T\sum_{i=1}^{n}\left((y\, log\, y + (1-y)\, log\,(1-y)) - log 0.5\right) \qquad (22)$$

5.2 S AND THE INTERACTION ACTIVATION AND COMPETITION MODEL

The activation update rule of the interactive activation and competition model (IAC) as defined in [9] is

$$\Delta a_i = \lambda\left((max - a_i)\, net_i - (a_i - rest)\, decay\right)\ ;\ net \geq 0 \qquad (23)$$

$$\Delta a_i = \lambda\left((a_i - min)\, net_i - (a_i - rest)\, decay\right)\ ;\ net \leq 0 \quad (24)$$

which can be derived from equation 5 applied to the following activation function

$$a_i = \frac{max\ net_i + rest\ decay}{net_i + decay}\ ;\ net \geq 0 \quad (25)$$

$$a_i = \frac{min\ net_i - rest\ decay}{net_i - decay}\ ;\ net \leq 0 \quad (26)$$

where max is the maximum value of the activation, rest the activation when the net input is zero, min the minimum value of the activation, and decay a positive constant. And applying equation 3, it is easy to see that

$$S = \sum_{i=1}^{n} S_i \quad (27)$$

with

$$S_i = decay((max - rest)\ log\left(\frac{max - rest}{max - a_i}\right) \quad (28)$$

$$-(a_i - rest));\ \ a_i \geq rest \quad (29)$$

$$S_i = decay((min - rest)\ log\left(\frac{a_i - min}{rest - min}\right) \quad (30)$$

$$+(a_i - rest));\ \ a_i \leq rest \quad (31)$$

with the decay parameter assuming the same function than gain in the logistic model.

5.3 HOPFIELD'S PROOF OF THE STABILITY OF ACTIVATIONS

Hopfield showed that if the network is regulated by equation 4 it will stabilize. This is done by showing that the Hopfield model has an associated Lyapunov function. Here, a similar argument is used to show that using equation 5 the network also stabilizes. To facilitate the calculation of the derivatives of E, we collapse into Q the part of F that does not depend on a_i,

$$E = -\frac{1}{2}\sum_{k=1}^{n}\sum_{l=1}^{n} a_k w_{kl} a_l \quad (32)$$

$$= -\frac{1}{2}(a_i w_{ii} a_i + a_i(\sum_{\substack{k=1\\k\neq i}}^{n} a_k w_{ik} \quad (33)$$

$$+\sum_{\substack{l=1\\l\neq i}}^{n} a_l w_{li}) + Q) \quad (34)$$

and considering that the weights are symmetric,

$$\frac{\partial E}{\partial a_i} = -\frac{1}{2}\left(2a_i w_{ii} + 2\sum_{\substack{k=1\\k\neq i}}^{n} a_k w_{ik}\right) = -net_i \quad (35)$$

Regarding S, since the derivative of the integral of a function is the function itself

$$\frac{\partial S}{\partial a_i} = f_i^{-1}(a_i) \quad (36)$$

and

$$\frac{\partial F}{\partial a_i} = \frac{\partial E}{\partial a_i} + \frac{\partial S}{\partial a_i} = -net_i + f_i^{-1}(a_i) \quad (37)$$

If we make

$$\frac{da_i}{dt} = \lambda\left(-a_i + f(net_k)\right) \quad (38)$$

then, applying the chain rule,

$$\frac{dF}{dt} = \sum_{k=1}^{n} \frac{\partial F}{\partial a_k}\frac{da_k}{dt} \quad (39)$$

$$= \sum_{k=1}^{n} \lambda\left(-net + f_k^{-1}(a_k)\right)\left(-a_k + f(net_k)\right) \quad (40)$$

but since f_k is monotonic, $\left(-net + f_k^{-1}(a_k)\right)$ has the same sign as $\left(-f(net_k) + a_k\right)$, making

$$\frac{dF}{dt} \leq 0 \quad (41)$$

Since F is bounded and on each time step F decreases then

$$\lim_{t\to\infty} \frac{dF}{dt} = 0 \quad (42)$$

and since equation 39 shows that

$$\frac{dF}{dt} = 0 \Longleftrightarrow \frac{da_k}{dt} = 0 \Longleftrightarrow \frac{\partial F}{\partial a_k} = 0\ \ ;k = 1, ..., n \quad (43)$$

then

$$\lim_{t\to\infty} \frac{da_i}{dt} = 0\ \ ;k = 1, ..., n \quad (44)$$

which tells us that the activations will tend to equilibrium as time progresses and that on equilibrium they are in a minima of F.

5.4 SMOOTHING NET INPUTS VS. ACTIVATIONS

Equation 4 can be discretized as

$$\Delta f_i^{-1}(a_{i(t)}) = \lambda\left(-f_i^{-1}(a_{i(t)}) + net_{i(t)}\right) \quad (45)$$

or

$$f_i^{-1}(a_{i(t+1)}) = (1-\lambda) f_i^{-1}(a_{i(t)}) + \lambda net_{i(t)} \quad (46)$$

equivalently, equation 5 would be discretized as

$$a_{i(t+1)} = (1-\lambda) a_{i(t)} + \lambda f(net_{i(t)}) \quad (47)$$

Equation 46 is an exponential smoothing of the net input. The activation function is then applied to this smoothed net. Another possibility, represented in equation 47, is to apply the activation function to the instantaneous net input and then to exponentially smooth these activations.

5.5 THE DERIVATIVES OF THE EQUILIBRIUM POINTS OF F

First consider the weights that connect different units. Extracting the cross products with a w_{ij} term. We have

$$\breve{E} = -\frac{1}{2}\left(2\breve{a}_i w_{ij}\breve{a}_j + \sum_{k=1}^{n}\sum_{\substack{l=1 \\ k,l\neq i,j;k,l\neq j,i}}^{n} \breve{a}_k w_{kl}\breve{a}_l\right) \quad (48)$$

and considering that w_{ij} is the only weight depending on w_{ij}.

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\frac{1}{2}(2\breve{a}_i\breve{a}_j + 2w_{ij}\breve{a}_i\frac{\partial \breve{a}_j}{\partial w_{ij}} + 2w_{ij}\breve{a}_j\frac{\partial \breve{a}_i}{\partial w_{ij}} \quad (49)$$

$$+\sum_{k=1}^{n}\sum_{\substack{l=1 \\ k,l\neq i,j;k,l\neq j,i}}^{n} w_{kl}(\breve{a}_k\frac{\partial \breve{a}_l}{\partial w_{ij}} + \breve{a}_l\frac{\partial \breve{a}_k}{\partial w_{ij}})) \quad (50)$$

Reorganizing terms considering that the weights are symmetric,

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\frac{1}{2}\left(2\breve{a}_i\breve{a}_j + 2\sum_{k=1}^{n}\frac{\partial \breve{a}_k}{\partial w_{ij}}\sum_{l=1}^{n} w_{kl}\breve{a}_l\right) \quad (51)$$

which easily leads to equation 9. Similar arguments can be applied to derive equation 10. Equation 11 is easy to obtain by applying the chain rule and the fact that the derivative is the inverse of the integral.

5.6 CONTRASTIVE HEBBIAN AND BACKPROPAGATION LEARNING ARE EQUIVALENT WHEN THERE ARE NO HIDDEN UNITS

The backpropagation weight update equation is

$$\Delta w_{ij} \propto I_i f'(a_j)(t_j - a_j) \quad (52)$$

where w_{ij} is the weight connecting input unit i with output unit j, f' the derivative of the activation function, and t_j the teacher for output unit j. The contrastive Hebbian weight update is

$$\Delta w_{ij} \propto \breve{a}_i^{(+)}\breve{a}_j^{(+)} - \breve{a}_i^{(-)}\breve{a}_j^{(-)} \quad (53)$$

Since the input units are clamped in both phases, they are not influenced by the output units and the equilibrium point of the activations would be the same as in backpropagation. Taking this into consideration and reorganizing terms we have

$$\Delta w_{ij} \propto \breve{a}_i^{(+)}(\breve{a}_j^{(+)} - \breve{a}_i^{(-)}) \quad (54)$$

where $\breve{a}_j^{(+)}$ is the same as the teacher, and $\breve{a}_i^{(+)}$ the clamped input. Since the derivative of the activation function is always positive (for strictly increasing activation functions), the cosine of the angle between the update vectors in backpropagation and contrastive Hebbian is positive and therefore they both minimize the same error function. Since there are no hidden units, the error function has a unique minima and thus the final learned solutions will be equivalent in both backpropagation and contrastive Hebbian.

5.7 INTERACTIVE NETWORKS WITH LOGISTIC NOISE APPROXIMATE BOLTZMANN MACHINES AS THE SHARPNESS OF THE ACTIVATION FUNCTIONS INCREASES

As sharpening increases, the activation function converges to a threshold function

$$a_i = max; net_i > 0 \quad (55)$$

$$a_i = min; net_i \leq 0 \quad (56)$$

where max is the upper bound of the activation, min the lower bound, and net_i has an added logistic noise component (σ) with variance $\frac{T^2\pi^2}{2}$.

$$net_i = \mathbf{a}_i^t\ \mathbf{w}_i + \sigma \quad (57)$$

It follows that

$$Prob(a_i = max) = Prob(net_i > 0) \quad (58)$$

$$= 1 - Prob(\sigma \leq -\mathbf{a}_i^T\ \mathbf{w}_i) \quad (59)$$

$$= 1 - \frac{1}{1 + e^{(\mathbf{a}_i^T\ \mathbf{w}_i)/T}} \quad (60)$$

$$= \frac{1}{1 + e^{-(\mathbf{a}_i^T\ \mathbf{w}_i)/T}} \quad (61)$$

which defines a Boltzmann machine.

5.8 SKETCH OF THE MAIN ROUTINES OF A CONTRASTIVE HEBBIAN PROGRAM

1. Get a training pattern.
2. Reset activations to rest and net inputs to zero.
3. Clamp inputs to desired pattern.
4. Settle activations according to equations 23 and 24. The program may provide some facility for sharpening schedules (changing the decay or gain parameter through settling), and annealing schedules (changing the standard deviation of noise added to the net inputs).
5. Collect cross products of activations multiplied by a negative constant.
6. Clamp also the output units to the desired pattern.
7. Settle activations according to equations 23 and 24.
8. Collect cross products of activations multiplied by a positive constant.

Termination of settling may be done after a fixed number of iterations (let us say 30), or after the changes in activations are smaller than a certain criteria (e.g. biggest activation change is smaller than 0.01). Bellow is an example of a settling cycle using the update function of the IAC model [9].

```
for(i=1; i≤ number_of_ units; i++){
   for(j=1; j≤ number_of_units; j++){
      net[i] = net[i] + w[j][i]*activation[j] ;
   }
    if(net[i] ≥ 0) act[i] = acti[i] + lambda*((actimax
-act[i]) *net[i] - decay*(acti[i]-rest));
else acti[i] =acti[i] + lambda*((acti[i] - actimin)*net[i]
- decay*(acti[i]-rest));
if(acti[i] > max) acti[i]=max;
if(acti[i] < min) acti[i]=min;
}
```

Where max is the maximum value of the activations, min the minimum value, rest the activation when net is zero, lambda the stepsize of the activation change (smoothing constant), and decay a positive constant. We have obtained good results with actimax =1.0, actimin = -1.0, rest =0.0, decay =0.1, lambda = 0.2.

This is an example of a weight change routine

```
for(i=1; ≤ number_of_units; i++)
   for(j=1; ≤ number_of_units; j++){
      weight_change[i][j] = weight_change[i][j] + phase
*epsilon* a[i]*a[j] ;
   }
}
```

Where *phase* is (+1) for the clamped case and (-1) for the free case. Epsilon is the stepsize of the weight change. Weights may be updated after each phase, each pattern or after a batch of patterns. Table 1 may be used to standardize new implementations of the algorithm.

	stepsize			
hidden u.	1.0	0.1	0.01	0.001
1	19 (0)	30 (0)	246 (5)	≥300 (10)
2	13 (0)	15 (0)	72 (1)	≥300 (7)
4	55 (3)	20.5 (0)	40.5 (0)	≥300 (6)
8	47 (1)	23 (0)	21 (0)	245 (4)
16	≥300 (10)	98.5 (0)	24.5 (0)	142(1)

Table 1: Results for the XOR problem. Networks were fully connected (including self connections). There were 10 simulations per cell with different starting random weights from a uniform (-0.5 to 0.5) distribution. Outside parenthesis are median number of epochs until maximum absolute error was smaller than 0.2. In parenthesis are number of simulations in which learning took longer than 300 epochs. Learning was on batch mode. Update of activation was done according to the IAC equations with the following parameter values: max=1.0; min=-1.0; rest=0.0; decay=0.1; lambda= 0.2. Update of activations was stopped after the change in all the activations was smaller than 0.0001 or the number of cycles bigger than 100. No annealing or sharpening was used

Acknowledgements

This research was founded by the NSF grants: BNS 88-12048 and BNS 86-09729. This paper would not have been possible without the support and motivating ideas of James McClelland and his research group at Carnegie Mellon University. I also thank Geoffrey Hinton, Conrad Galland and Chris Williams for their advice and helpful comments.

References

[1] Ackley D, Hinton G and Sejnowski T (1985) A learning algorithm for Boltzmann machines. *Cognitive Science*, 9, 147-169.

[2] Akiyama Y, Yamashita A, Kajiura M, Aiso H (1989) Combinatorial Optimization with Gaussian Machines, *Proceedings of the International Joint Conference on Neural Networks* 1, 533-540.

[3] Hinton G E (1989) Deterministic Boltzmann Learning Performs Steepest Descent in Weight-Space, *Neural Computation*, **1**, 143-150.

[4] Hopfield J (1982) Neural Networks and Physical Systems with emergent collective computational abilities. *Proceedings of the National Academy of Science* USA, 79, 2254-2558.

[5] Hopfield J, Feinstein D, Palmer R (1983) Unlearning has a stabilizing effect in collective memories. *Nature* 304, 158-159.

[6] Hopfield J (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences* U.S.A., **81**, 3088-3092.

[7] Galland C, Hinton G (1989) Deterministic Boltzmann Learning in Networks with Asymmetric Connectivity. University of Toronto. Department of Computer Science Technical Report. CRG-TR-89-6.

[8] Grossberg S A (1978) A theory of visual coding, memory, and development. in E Leeuwenberg and H Buffart (Eds.) *Formal Theories of visual perception* New York, Willey.

[9] McClelland J, Rumelhart D (1981) An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An account of Basic Findings. *Psychological Review*, 88, 5.

[10] McClelland J, Rumelhart D (1989) Interactive activation and Competition. Chapter II of *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises* Cambridge, MIT Press.

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

[12] Peterson C, Hartman E (1989) Explorations of the Mean Field Theory Learning Algorithm. *Neural Networks*, 2, 475-494.

[13] Williams C, Hinton G (1990) Mean field networks that learn to discriminate temporally distorted strings. *Pre-Proceedings of the 1990 Connectionist Models Summer School.*