

# Learning fixed points

Sebastian Seung

9.641 Lecture 13: October 30, 2000  
revised Nov. 30

There are many ways to use recurrent networks for computation. For concreteness, let's assume the one used in previous lectures. The inputs are encoded in the  $b_i$  and the outputs are the  $x_i$  after convergence of the network dynamics to a steady state. Suppose now that we would like to train a desired response  $d_i$ , as opposed to the actual response  $x_i$ . Two general methods for this are known, recurrent backpropagation and contrastive Hebbian learning.

## 1 Recurrent backpropagation

As usual, define  $\hat{x} = Wx + b$ . The learning algorithm is

1. Let the network  $\dot{x} + x = f(Wx + b)$  converge to a fixed point, and compute the error  $\epsilon = d - x$ .
2. Solve the equations  $(D - W^T)y = \epsilon$  for  $y$ , where the matrix  $D$  is defined by

$$D_{ij} = \frac{\delta_{ij}}{f'(\hat{x}_j)}$$

3. Make the update  $\Delta W = \eta y x^T$ .

Note that this algorithm works for hidden neurons, since there doesn't have to be a desired value for every neuron.

Alternatively, step two can be performed by letting the "dual network"

$$\dot{y}_j + y_j = \sum_i y_i W_{ij} f'(\hat{x}_j) + f'(\hat{x}_j)(d_j - x_j) \quad (1)$$

converge to a fixed point. The dual network is linear, and has synaptic weights that are the diagonally rescaled transpose of the primal network. It can be shown that the dual network is stable if the steady state of the nonlinear primal network is locally stable.

The above algorithm performs gradient descent with respect to  $W$  on

$$E = \frac{1}{2} \sum_i (d_i - x_i)^2$$

subject to the constraint

$$f^{-1}(x_i) = \sum_j W_{ij}x_j + b_i$$

To prove this, define the Lagrangian

$$L = \frac{1}{2} \sum_i (d_i - x_i)^2 + \sum_i y_i \left[ f^{-1}(x_i) - \sum_j W_{ij}x_j - b_i \right]$$

Note that

$$\frac{\partial L}{\partial y_i} = f^{-1}(x_i) - \sum_j W_{ij}x_j - b_i$$

Therefore, the constraint is satisfied and  $L = E$  at a stationary point of  $L$ . So the optimization can be reformulated as gradient descent on  $L$  evaluated at a stationary point.

Defining  $\hat{x}_i \equiv f^{-1}(x_i)$  and setting the derivative

$$\frac{\partial L}{\partial x_j} = x_j - d_j + \frac{y_j}{f'(x_j)} - \sum_i y_i W_{ij}$$

equal to zero yields

$$(D - W^T)y = d - x \tag{2}$$

If this equation is satisfied, along with the constraint, then we are at a stationary point. Then gradient descent is

$$\Delta W = -\eta \frac{\partial L}{\partial W} = \eta y x^T$$

where we consider only the explicit dependence of  $L$  on  $W$ .

## 2 Contrastive Hebbian learning

Above we ignored the problem that there is no guarantee that the primal network will converge to a steady state. But recall that if the connections  $W_{ij}$  are symmetric, then convergence to a steady state is guaranteed, because of the existence of the Lyapunov function

$$E(x) = \sum_i \bar{F}(x_i) - \sum_i b_i x_i - \frac{1}{2} \sum_{ij} W_{ij} x_i x_j$$

The Lyapunov function can be used to derive the “contrastive Hebbian” learning rule:

$$\Delta W_{ij} = \eta(d_i d_j - x_i x_j)$$

Why should this work? Intuitively, the first term  $\Delta W_{ij} = \eta d_i d_j$  lowers the energy of the desired response, while the second term  $\Delta W_{ij} = -\eta x_i x_j$  raises the energy of the actual response. Just lowering the energy of the desired response is not enough to make it into a minimum; the contrast is needed. Another way of putting it is that the first term

is learning, while the second term is unlearning. The two terms are sometimes called “Hebbian” and “anti-Hebbian.”

More formally, the contrastive update is gradient descent on the cost function

$$C(W) = E_W(d) - E_W(x)$$

which is the difference in energy between the desired and actual responses. Note that  $x$  is actually a function of  $W$ , since the minima of  $E(x)$  shift around if  $W$  changes. So the gradient of the cost function is

$$\frac{\partial C}{\partial W_{ij}} = d_i d_j - x_i x_j - \sum_k \frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial W_{ij}}$$

The last term represents the effect of the shift in the minimum. But it’s weighted by a quantity that is zero, so it vanishes anyway. Note that the cost function is lower bounded by zero, so gradient descent will converge to a minimum of the cost function. Gradient descent tries to achieve a situation where  $d$  and  $x$  have the same energy. This is typically achieved when  $d = x$ , but could be fooled if there are multiple minima.

If there are hidden neurons, then the contrastive Hebbian learning rule still works. Let the neurons with desired values be called visible neurons.

1. Apply the inputs through  $b_i$ , and let the network converge to a fixed point  $x_i$ .
2. Clamp the visible neurons at their desired values  $d_i$ , and let the hidden neurons converge to their steady state values, which are used to complete the desired response vector.
3. Make the contrastive update  $\Delta W_{ij} = \eta(d_i d_j - x_i x_j)$ .

The derivation with hidden neurons is basically the same. The only difference is that there is an extra term in  $\partial C / \partial W_{ij}$  accounting for the shift in the hidden neuron values of  $d$ . But this term vanishes for the same reason the other shift term vanishes.

### 3 CHL as a form of recurrent backpropagation

Suppose that  $\epsilon_i = d_i - x_i$  is small. Then we can approximate the CHL update as

$$\Delta W_{ij} = \eta(\epsilon_i x_j + x_i \epsilon_j) \tag{3}$$

where we have neglected terms of order  $\epsilon^2$ . If we added the constraint of symmetry, then the RBP update would take the symmetric form

$$\Delta W_{ij} = \eta(y_i x_j + x_i y_j)$$

where  $y = (D - W)^{-1} \epsilon$ . So the only difference between the two learning rules is that RBP has an extra factor of  $(D - W)^{-1}$ . What are the consequences of this difference?

We saw that CHL forces the energies of the desired and actual responses to become equal. But does it make the actual response approach the desired response? Recall that

RBP does this: it moves  $x$  closer to  $d$ , where closeness is measured by the squared error cost function  $E = \frac{1}{2} \sum_i \epsilon_i^2$ . But this is not the only cost function we could use to measure closeness. In general, we could use  $E = \frac{1}{2} \sum_{ij} \epsilon_i Q_{ij} \epsilon_j$ , where  $Q_{ij}$  is any positive definite matrix. In particular, we will use  $Q = D - W$ . This is positive definite at any asymptotically stable fixed point of the network dynamics.

We modify the Lagrangian to

$$L = \frac{1}{2} \sum_{ij} \epsilon_i Q_{ij} \epsilon_j + \sum_i y_i \left[ f^{-1}(x_i) - \sum_j W_{ij} x_j - b_i \right]$$

Setting the derivative

$$\frac{\partial L}{\partial x_j} = - \sum_i (d_i - x_i) Q_{ij} + \frac{y_j}{f'(x_j)} - \sum_i y_i W_{ij}$$

equal to zero, we find that

$$(D - W)y = (D - W)\epsilon$$

which implies that  $\delta = \epsilon$ . So the update (3) follows. The implication is that this update moves the actual response closer to the desired response, if we measure closeness using the metric defined by  $D - W$ . This result can be extended to the case where there are hidden neurons.

Note that this argument neglects the dependence of  $L$  on  $W$  through  $Q = D - W$ .

## 4 RBP as a form of contrastive learning

In RBP, the error is computed as the difference between desired and actual responses of the visible neurons, and then is propagated by a dual network to reach the hidden neurons. CHL is interesting because the error propagation is done without a dual network. The visible neurons are clamped at their desired values, and the influence of the clamping is propagated to the hidden neurons. Then the learning update is computed with a difference between clamped and unclamped terms. But CHL is able to dispense with the dual network at the cost of requiring symmetry of the weight matrix.

In fact, gradient descent on the squared error can be performed using a contrastive update, provided that the weight matrix is symmetric. The algorithm is

1. Let the network converge to a fixed point  $x$ .
2. Shift the input by  $\delta b = d - x$ , and observe the new steady state  $x' = x + \delta x$ .
3. Make the update  $\Delta W = \delta x x^T + x \delta x^T$ , which is equivalent to  $\Delta W = x' x'^T - x x^T$  if  $\delta x$  is small.

Suppose that the network is at a steady state  $f^{-1}(x) = Wx + b$ . If we shift  $b \rightarrow b + \delta b$ , then the shift in the steady state  $\delta x$  satisfies

$$D\delta x = W\delta x + \delta b$$

Recall that the RBP update made use of  $(D - W^T)y = d - x$ . Since  $W = W^T$ , we can set  $\delta b = d - x$ , and we will obtain  $\delta x = y$ .