Anti-Hebbian synapses as a linear equation solver

Kechen Zhang*
Giorgio Ganis*
Martin I. Sereno
Department of Cognitive Science
University of California, San Diego
La Jolla, California 92093-0515
{kzhang, ganis, sereno}@cogsci.ucsd.edu

Abstract. It is well-known that Hebbian synapses, with appropriate weight normalization, extract the first principal component of the input patterns (Oja 1982). Anti-Hebb rules have been used in combination with Hebb rules to extract additional principal components or generate sparse codes (e.g., Rubner and Schulten 1990; Földiák 1990). Here we show that the simple anti-Hebbian synapses alone can support an important computational function: solving simultaneous linear equations. During repetitive learning with a simple anti-Hebb rule, the *weights* onto an output unit always converge to the exact solution of the linear equations whose coefficients correspond to the input patterns and whose constant terms correspond to the biases, provided that the solution exists. If there are more equations than unknowns and no solution exists, the weights approach the values obtained by using the Moore-Penrose generalized inverse (pseudoinverse). No explicit matrix inversion is involved and there is no need to normalize weights. Mathematically, the anti-Hebb rule may be regarded as an iterative algorithm for learning a special case of the linear associative mapping (Kohonen 1989; Oja 1979). Since solving systems of linear equations is a very basic computational problem to which many other problems are often reduced, our interpretation suggests a potentially general computational role for the anti-Hebbian synapses and a certain type of long-term depression (LTD).

Suppose we have n input variables $a_1, \ldots, a_n$, and a linear unit whose output y is the weighted sum

$$y = \sum_{i=1}^{n} a_i w_i - b,$$

where variable b is the bias, or an additional input variable with constant weight -1. The weights are modified according to the simple anti-Hebb rule

$$\Delta w_i = -\varepsilon a_i y$$

where the learning rate $\varepsilon > 0$ is a small constant. The weight increments of the anti-Hebb rule presented here become identical to those of the Widrow-Hoff delta rule for supervised learning if the bias is equal to the "desired output". Nonetheless, the actual output of the linear unit has different values in the two cases whenever the bias is nonzero. If the weight for the bias term

*These authors have made equal contributions to this work. KZ's present address: Computational Neurobiology Laboratory, The Salk Institute, La Jolla, California 92037.

0-7803-4122-8/97 $10.00©1997 IEEE

(= −1 in our case) is also modified according to the anti-Hebb rule, the system still can solve linear equations. But in the overdetermined cases, explicit weight normalization is needed to prevent all weights from vanishing altogether, and the result is the minor component (cf. Oja 1992) of the input patterns instead of the one obtained with the pseudoinverse.

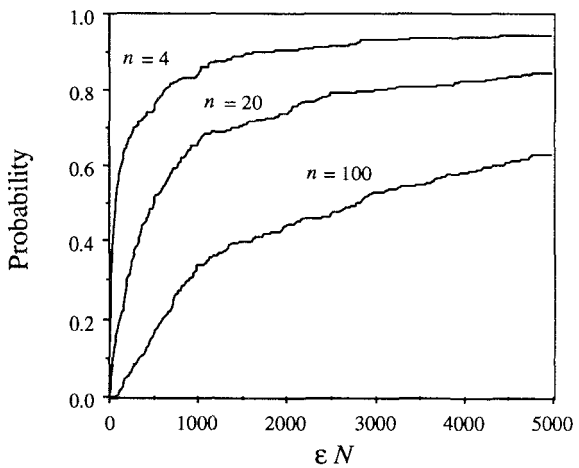

Figure 1: The cumulative probability for the weights to reach the solution *within* iteration number N during anti-Hebbian learning. Here we use εN instead of N because although it takes a larger iteration number N to stop for a smaller learning rate ε, the product εN is basically a constant if ε is small. In the simulation $\varepsilon = 0.01$. All coefficients and constants of the linear equations ($m = n$) are drawn from a uniform distribution between −1 and 1. Each curve ($n = 4$, 20, or 100) is based on the data from 250 repetitive trials, with the initial weights randomly chosen between −0.05 and 0.05. The iteration stops if for all weights $|w_i - x_i| < \left(\sum_{j=1}^{n} |x_j|/n\right) \times 10\%$, where $(x_1, \ldots, x_n)$ is the exact solution. (Numerical simulation performed mainly by G. G.)

Suppose the input variables only have m sets of distinct values: $a_1(j), \ldots, a_n(j), b(j)$, with $j = 1, \ldots, m$. When these values are presented repeatedly to the system, the final weights solve the simultaneous equations

$$\begin{pmatrix} a_1(1) & \cdots & a_n(1) \\ \vdots & & \vdots \\ a_1(m) & \cdots & a_n(m) \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} b(1) \\ \vdots \\ b(m) \end{pmatrix}$$

or

$$\mathbf{A}\mathbf{w} = \mathbf{b}$$

More precisely, when $m = n$, we have $\mathbf{w} \to \mathbf{A}^{-1}\mathbf{b}$, and when $m > n$, we have $\mathbf{w} \to \mathbf{A}^{\dagger}\mathbf{b}$ where $\mathbf{A}^{\dagger} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the pseudoinverse. In the degenerate cases, the weight vector will converge to one of the possible solutions, depending on the initial weights. A noteworthy special case is the learning of invariants (cf. Schraudolph and Sejnowski 1992): when the biases are all equal [$b(j) \equiv$ const.], the system tries to find an invariant linear combination of the inputs, which is similar to what a differential anti-Hebb rule would do (Mitchison 1991). After learning, the output unit becomes a linear filter which, roughly speaking, gives a large response y only when the input pattern is *not* a linear combination of the training patterns, which is similar to Kohonen's novelty detector (Kohonen 1989).

Rewriting the weight update equation in terms of inputs, it can be seen that in the slow learning limit ($\varepsilon \to 0$), the time evolution of the weight vector is governed by the average of all the inputs:

$$\dot{\mathbf{w}} = -\mathbf{A}^T\mathbf{A}\mathbf{w} + \mathbf{A}^T\mathbf{b}$$

This is a gradient descent procedure for minimizing the function $E = \frac{1}{2}\|\mathbf{A}\mathbf{w} - \mathbf{b}\|^2$, because the right-hand side of the equation equals $-\partial E/\partial \mathbf{w}$. When $\dot{\mathbf{w}} = \mathbf{0}$, the final weight vector is the general solution $\mathbf{A}^{\dagger}\mathbf{b}$. In nondegenerate cases, this is identical to $\mathbf{A}^{-1}\mathbf{b}$. The convergence of the discrete iteration process with an adaptive learning rate was studied in general terms by Oja (1979).

For random inputs, the weights converge to the solution at various rates. The convergence is typically slower as the number of variables increases (Fig. 1). The anti-Hebbian algorithm is less efficient than standard linear algebra methods or iterative algorithms that reach the exact solution in a finite number of steps (e.g., Kohonen 1989, Forbes and Mansfield 1990), because it approaches the solution only asymptotically. The importance of this algorithm lies in its great simplicity (as compared with complex "hard-wired" neural network approaches, cf. Cichocki and Unbehauen 1992) and in its possible biological relevance. An effective anti-Hebb rule could arise from Hebbian-type plasticity with the help of inhibitory interneurons; or anti-Hebbian synapses themselves may exist in the brain. In fact, long-term depression (LTD) in the cerebellum (Ito 1989) and in the basal ganglia (Calabresi *et al.* 1992) can be regarded as essentially anti-Hebbian, because in both cases excitatory synapses are *weakened* by simultaneous pre- and post-synaptic activations. Of course, the linear neuron model is only a very crude approximation of the real situation. Yet this simplified model together with a simple anti-Hebb rule is sufficient to solve arbitrary systems of linear equations — a basic task for numerical computation, and perhaps for certain neural subsystems.

Acknowledgments

Supported by McDonnell-Pew Center for Cognitive Neuroscience at San Diego and National Institutes of Health Grant MH47035 to Martin I. Sereno.

References

Calabresi, P., Maj, R., Pisani, A., Mercuri, N. B., and Bernardi, G. 1992. Long-term synaptic depression in the striatum: physiological and pharmacological characterization. *J. Neurosci.* **12**, 4224–4233.

Cichocki, A., and Unbehauen, R. 1992. Neural network for solving systems of linear equations and related problems. *IEEE Trans. Cir. Sys. I* **39**, 124–138.

Földiák, P. 1990. Forming sparse representations by local anti-Hebbian learning. *Biol. Cybern.* **64**, 165–170.

Forbes, A. B., and Mansfield, A. J. 1990. Neural implementation of a method for solving systems of linear equations. *Network* **1**, 217–229.

Ito, M. 1989. Long-term depression. *Annu. Rev. Neurosci.* **12**, 85–102.

Kohonen, T. 1989. *Self-Organization and Associative Memory*, 3rd ed., Springer-Verlag, Berlin.

Mitchison, G. 1991. Removing time variation with the anti-Hebbian differential synapse. *Neural Comp.* **3**, 312–320.

Oja, E. 1979. On the construction of projectors using products of elementary matrices. *IEEE Trans. Comp.* **27**, 65–66.

Oja, E. 1982. A simplified neuron model as a principal component analyzer. *J. Math. Biol.* **15**, 267–273.

Oja, E. 1992. Principal components, minor components, and linear neural networks. *Neural Networks* **5**, 927–935.

Rubner, J., and Schulten, K. 1990. Development of feature detectors by self-organization: A network model. *Biol. Cybern.* **62**: 193–199.

Schraudolph, N. N., and Sejnowski, T. J. 1992. Competitive anti-Hebbian learning of invariants. In *Advances in Neural Information Processing System 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, eds., pp. 1017–1024. Morgan Kaufmann, San Mateo, CA.