Words are wind; learning is nothing but words; ergo, learning is nothing but wind.
—Jonathan Swift

9 Hebbian Learning

Donald Hebb was not a neural network researcher in 1949 when he proposed his model for biological learning. He was a psychologist whose immediate goal was to understand how neurons in the brain change when learning occurs. His model was quickly picked up by neural network researchers, however.

Those seeking a new neural network design often adopt the ideas of biologists or psychologists. The reason, of course, is that it makes sense to copy a system already known to be successful—the brain. Of all the learning concepts appropriated from psychology by neural network researchers, hebbian learning is probably the best known and most used. The original statement of Hebb's law reads as follows: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." *

We will see that this deceptively simple statement is remarkable in its implications. It provides nearly all we need to know about learning to make useful neural networks. Hebb is saying that a neuron, A, that repeatedly happens to stimulate another neuron, B, at the times when B is firing, will have an increased effectiveness in

* D. O. Hebb, *The Organization of Behavior*, (New York: Wiley, 1949), p. 62.

stimulating B to fire in the future. If we translate this into "neurodes" and "weights," we can restate Hebb's law as follows: If neurode A repeatedly stimulates neurode B while B is generating an output signal, the weight of the synapse between A and B will increase in magnitude.

The net effect of this process is that the strength of the interconnect from A to B increases. This implies that neurode B will become more sensitive to neurode A's stimulus after appropriate training has occurred. During training, we sensitize the network's response to signals passing along certain pathways.

There are a number of details that are not addressed by this simple statement of Hebb's law, and it must usually be modified to be useful in actual neural network implementations. In the mid-1950s, when researchers began writing computer simulations of hebbian systems to determine their ability to learn, the inadequacy of the law for computational purposes quickly became apparent. For example, as the law is stated, the weights on the interconnects can rise without an upper bound; they can potentially rise to infinity. This kind of limitless growth, of course, is anathema to computer simulations. As a result, hebbian learning rules often have the additional constraint imposed on them that the weights of each neurode must be normalized; that is, the weight vector is constrained to have a fixed, constant length. This length is usually set to 1.0.

The impact of this constraint goes beyond the simple matter of confining the growth of the weights to manageable bounds. Assume that a particular neurode has several input interconnections and weights and that one of these weights is increased during training. If the neurode is to maintain a fixed-length weight vector while one of the weights increases, then one or more of the other weights must decrease. Because this decrease can come only from an interconnect that is not currently stimulating the neurode, the constraint effectively requires that a synapse that is consistently not stimulated when the neurode fires will gradually decrease in strength during training. Eventually this weight may even wither to a zero value. Here we have the neural network equivalent of "use it or lose it."

This is not the only modification of Hebb's law that is needed to use it as a neural network learning rule. We must provide for both positive and negative interconnect weights—excitatory and inhibitory synapses. With this feature, a positive stimulus from neurode A can have the effect of increasing or decreasing the tendency of neurode B to fire. Another way of saying this is that allowing negative interconnect weights permits inhibition as well as excitation of the stimulated neurode. Of course, this feature is also needed for it to be biologically accurate.

One way of accomplishing this is to alter our normalization procedure slightly. By picking the limits of the weights to be –1.0 and +1.0, and adding any one of several mathematical conditions that force the length of the weight vector to be equal to 1.0, we can provide for both inhibitory and excitatory synapses and for reduction as well as growth of weights during training.

Neohebbian Learning

One of the classical mathematical expressions of hebbian learning was produced by Stephen Grossberg in the mid-1960s. This version is sometimes called neohebbian learning because it expands the original statement of Hebb's law and provides an explicit mathematical model for Hebb's weight change rule.

The neohebbian model accounts for the fact that biological systems not only learn but also forget. This feature is essential if we want to explain the behavior of biological systems. At first glance, forgetting may not seem to be a useful feature in a learning model for artificial systems, but that is often not the case. It can be useful, for instance, to avoid overcrowding memory with seldom used detail or to correct mistakes in previously learned information.

To put it in its normal context, we'll state Grossberg's neohebbian weight change law as a computational rule. Before we actually state the rule, we need to talk a little about the stepwise manner in which such computations, or simulations, are made. Assume that we have presented an input pattern to the neural network and must now change the weights on each neurode so that it can begin learning the input pattern. In computations of this sort, we move the state of the network forward in small time increments; that is, we calculate the activity and value of the output for every neurode and update the weight of every synapse for one instant of time before moving on to the next instant. Because of this iterative procedure, we can state the law for only one neurode pair and apply it in turn to each synapse of each neurode. Let's assume that the synapse to be updated lies on neurode B, which receives output signals from neurode A. Since we are dealing with only one pair of neurodes, the output of neurode A in this case is unambiguously the input to neurode B, and we will use "output of neurode A" and "input to neurode B" synonymously in this discussion.

The content of this law can be summarized by writing it as a word equation. In this form, it is

```
<new weight> =
            <old weight>
            – F <amount forgotten between last and current cycle>
            + L <new learning in this cycle>
```

Here, *F* and *L* are both constants that are in the range of 0 to 1.0. The constant *F* controls how quickly the network forgets, and *L* controls how quickly it learns.

Let's look at this rule for some special values of the forgetting and learning constants. First, if neither forgetting nor learning occurs (both *F* and *L* are zero in other words), the new weight value is the same as the old one; nothing is forgotten, and nothing is learned. If the forgetting constant, *F*, is very large, the hebbian term results in a new weight value, but the old weight is completely forgotten; nothing is retained from cycle to cycle. A very small value for *F* implies that little or no forgetting occurs. If the learning constant, *L*, is zero, the fraction of the old weight determined by the forgetting constant is retained, but no new changes are made. No learning takes place. Finally, the relative values of the forgetting and learning constants pick the relative importance of new and old knowledge.

The third term of this statement, the hebbian learning term, is just a computational statement of Hebb's law. It tells us that if both the incoming stimulus and the output of a neurode are large at the same time, the weight change of the affected synapse will be large and a lot of learning will occur. If, on the other hand, either the stimulus or the output is small or zero, little or no weight change will be made, and little or no learning will occur. Only when an input stimulus from neurode A coincides with an output from neurode B will learning occur.

Neohebbian learning does not resolve all the problems of hebbian learning; it merely provides a mathematical framework for the original concept and introduces the phenomenon of forgetting. It still has no way of dealing with inhibitory stimuli, so it does not model biological systems accurately. Neohebbian learning did, however, act as a precursor to Grossberg's later outstar learning paradigm.

Differential Hebbian Learning

In our discussion of simple hebbian learning, we had to introduce two features found in biological systems that are necessary for proper operation of a neural network: the possibility of both decreasing and increasing weights during learning and the presence of inhibitory as well as excitatory synapses. In differential hebbian learning, we finally have a learning system that provides both of these features in a natural way. In differential hebbian learning, the learning term of the weight change rule is not proportional to the product of the input and output signals of neurode B but instead is proportional to the product of the rates of change in those signals. In mathematical terms, the expression "rate of change" refers to the derivative of a neurode's output with respect to time.

Under this law, if the input signal from neurode A to neurode B increases in strength (a positive change) at the same time that the output signal of neurode B decreases in strength (a negative change), then their product is negative, and the weight itself becomes more negative. If the input signal decreases at the same time that the output signal increases, the product is similarly negative and the weight again becomes more negative. Only when both the input and the output signals increase or decrease at the same time is their product positive, and only under these circumstances will the weight increase and become more positive.

Differential hebbian learning at last provides us with a learning rule that can model a number of aspects of learning in biological systems with considerable accuracy. It is often the basis for neural network simulations today but rarely in this pure form. It has also inspired a variety of modifications to other network paradigms. For example, there is a version of backpropagation, called backpropagation with recirculation, that uses a differential hebbian learning approach.

Let's look at how classical conditioning is applied in artificial neural networks by discussing two network learning paradigms, the outstar and the drive reinforcement learning models. Both use a form of hebbian learning.