8 Plasticity and Learning

8.1 Introduction

Activity-dependent synaptic plasticity is widely believed to be the basic phenomenon underlying learning and memory, and it is also thought to play a crucial role in the development of neural circuits. To understand the functional and behavioral significance of synaptic plasticity, we must study how experience and training modify synapses, and how these modifications change patterns of neuronal firing to affect behavior. Experimental work has revealed ways in which neuronal activity can affect synaptic strength, and experimentally inspired synaptic plasticity rules have been applied to a wide variety of tasks including auto- and heteroassociative memory, pattern recognition, storage and recall of temporal sequences, and function approximation.

Hebb rule

In 1949, Donald Hebb conjectured that if input from neuron A often contributes to the firing of neuron B, then the synapse from A to B should be strengthened. Hebb suggested that such synaptic modification could produce neuronal assemblies that reflect the relationships experienced during training. The Hebb rule forms the basis of much of the research done on the role of synaptic plasticity in learning and memory. For example, consider applying this rule to neurons that fire together during training due to an association between a stimulus and a response. These neurons would develop strong interconnections, and subsequent activation of some of them by the stimulus could produce the synaptic drive needed to activate the remaining neurons and generate the associated response. Hebb's original suggestion concerned increases in synaptic strength, but it has been generalized to include decreases in strength arising from the repeated failure of neuron A to be involved in the activation of neuron B. General forms of the Hebb rule state that synapses change in proportion to the correlation or covariance of the activities of the pre- and postsynaptic neurons.

Experimental work in a number of brain regions, including hippocampus, neocortex, and cerebellum, has revealed activity-dependent processes that can produce changes in the efficacies of synapses that persist for varying amounts of time. Figure 8.1 shows an example in which the data points

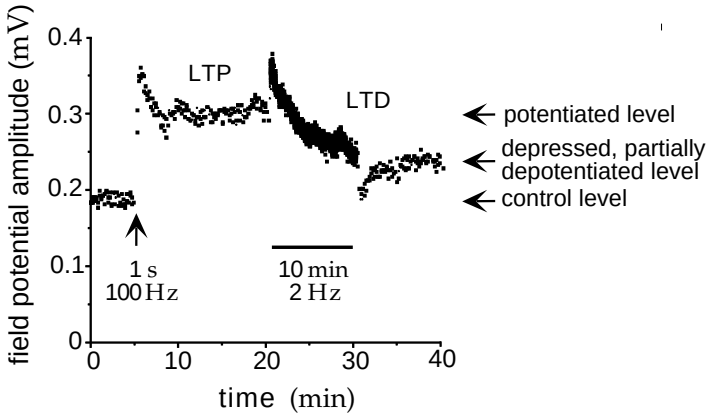

Figure 8.1 LTP and LTD at the Schaffer collateral inputs to the CA1 region of a rat hippocampal slice. The points show the amplitudes of field potentials evoked by constant amplitude stimulation. At the time marked by the arrow (at time 5 minutes), stimulation at 100 Hz for 1 s caused a significant increase in the response amplitude. Some of this increase decayed away following the stimulation, but most of it remained over the following 15 min test period, indicating LTP. Next, stimulation at 2 Hz was applied for 10 min (between times 20 and 30 minutes). This reduced the amplitude of the response. After a transient dip, the response amplitude remained at a reduced level approximately midway between the original and post-LTP values, indicating LTD. The arrows at the right show the levels initially (control), after LTP (potentiated), and after LTD (depressed, partially depotentiated). (Unpublished data of J. Fitzpatrick and J. Lisman.)

indicate amplitudes of field potentials evoked in the CA1 region of a slice of rat hippocampus by stimulation of the Schaffer collateral afferents. In experiments such as this, field potential amplitudes (or more often slopes) are used as a measure of synaptic strength. In figure 8.1, high-frequency stimulation induced synaptic potentiation (an increase in strength), and then long-lasting, low-frequency stimulation resulted in synaptic depression (a decrease in strength) that partially removed the effects of the previous potentiation. This is in accord with a generalized Hebb rule because high-frequency presynaptic stimulation evokes a postsynaptic response, whereas low-frequency stimulation does not. Changes in synaptic strength involve both transient and long-lasting effects, as seen in figure 8.1. Changes that persist for tens of minutes or longer are generally called long-term potentiation (LTP) and long-term depression (LTD). The longest-lasting forms appear to require protein synthesis.

potentiation

depression

LTP and LTD

A wealth of data is available on the underlying cellular basis of activity-dependent synaptic plasticity. For instance, the postsynaptic concentration of calcium ions appears to play a critical role in the induction of both LTP and LTD. However, we will not consider mechanistic models. Rather, we study synaptic plasticity at a functional level, attempting to relate the impact of synaptic plasticity on neurons and networks to the basic rules governing its induction.

Studies of plasticity and learning involve analyzing how synapses are affected by activity over the course of a training period. In this and the following chapters, we consider three types of training procedures. In unsupervised (also called self-supervised) learning, a network responds to a series of inputs during training solely on the basis of its intrinsic connections and dynamics. The network then self-organizes in a manner that depends on the synaptic plasticity rule being applied and on the nature of the inputs presented during training. We consider unsupervised learning in a more general setting called density estimation in chapter 10.

unsupervised learning

In supervised learning, which we consider in the last section of this chapter, a desired set of input-output relationships is imposed on the network by a "teacher" during training. Networks that perform particular tasks can be constructed in this way by letting a modification rule adjust their synapses until the desired computation emerges as a consequence of the training process. This is an alternative to explicitly specifying the synaptic weights, as was done in chapter 7. In this case, finding a biologically plausible teaching mechanism may not be a concern if the question being addressed is whether any weights can be found that allow a network to implement a particular function. In more biologically plausible examples of supervised learning, one network acts as the teacher for another network.

supervised learning

In chapter 9, we discuss a third form of learning, reinforcement learning, that is intermediate between these cases. In reinforcement learning, the network output is not constrained by a teacher, but evaluative feedback about network performance is provided in the form of reward or punishment. This can be used to control the synaptic modification process.

reinforcement learning

In this chapter we largely focus on activity-dependent synaptic plasticity of the Hebbian type, meaning plasticity based on correlations of pre- and postsynaptic firing. To ensure stability and to obtain interesting results, we often must augment Hebbian plasticity with more global forms of synaptic modification that, for example, scale the strengths of all the synapses onto a given neuron. These can have a major impact on the outcome of development or learning. Non-Hebbian forms of synaptic plasticity, such as those that modify synaptic strengths solely on the basis of pre- or postsynaptic firing, are likely to play important roles in homeostatic, developmental, and learning processes. Activity can also modify the intrinsic excitability and response properties of neurons. Models of such intrinsic plasticity show that neurons can be remarkably robust to external perturbations if they adjust their conductances to maintain specified functional characteristics. Intrinsic and synaptic plasticity can interact in interesting ways. For example, shifts in intrinsic excitability can compensate for changes in the level of input to a neuron caused by synaptic plasticity. It is likely that all of these forms of plasticity, and many others, are important elements of both the stability and the adaptability of nervous systems.

non-Hebbian plasticity

In this chapter, we describe and analyze basic correlation- and covariance-based synaptic plasticity rules in the context of unsupervised learning, and

discuss their extension to supervised learning. As an example, we discuss the development of ocular dominance in cells of the primary visual cortex, and the formation of the map of ocular dominance preferences across the cortical surface. In the models we discuss, synaptic strengths are characterized by synaptic weights, defined as in chapter 7.

Stability and Competition

Increasing synaptic strength in response to activity is a positive feedback process. The activity that modifies synapses is reinforced by Hebbian plasticity, which leads to more activity and further modification. Without appropriate adjustments of the synaptic plasticity rules or the imposition of constraints, Hebbian modification tends to produce uncontrolled growth of synaptic strengths.

The easiest way to control synaptic strengthening is to impose an upper limit on the value that a synaptic weight can take. Such an upper limit is supported by LTP experiments. It also makes sense to prevent weights from changing sign, because the plasticity processes we are modeling cannot change an excitatory synapse into an inhibitory synapse or vice versa. We therefore impose the constraint, which we call a saturation constraint, that all excitatory synaptic weights must lie between 0 and a maximum value $w_{\max}$, which is a constant. The simplest implementation of saturation is to set any weight that would cross a saturation bound due to application of a plasticity rule to the limiting value.

synaptic saturation

Uncontrolled growth is not the only problem associated with Hebbian plasticity. Synapses are modified independently under a Hebbian rule, which can have deleterious consequences. For example, all of the synaptic weights may be driven to their maximum allowed values $w_{\max}$, causing the postsynaptic neuron to lose selectivity to different patterns of input. The development of input selectivity typically requires competition between different synapses, so that some are forced to weaken when others become strong. We discuss a variety of synaptic plasticity rules that introduce competition between synapses. In some cases, the same mechanism that leads to competition also stabilizes growth of the synaptic weights. In other cases, it does not, and saturation constraints must also be imposed.

synaptic competition

8.2 Synaptic Plasticity Rules

Rules for synaptic modification take the form of differential equations describing the rate of change of synaptic weights as a function of the pre- and postsynaptic activity and other possible factors. In this section, we give examples of such rules. In later sections, we discuss their computational implications.

In the models of plasticity that we study, the activity of each neuron is described by a continuous variable, not by a spike train. As in chapter 7, we use the letter u to denote the presynaptic level of activity and v to denote the postsynaptic activity. Normally, u and v represent the firing rates of the pre- and postsynaptic neurons, in which case they should be restricted to nonnegative values. Sometimes, to simplify the analysis, we ignore this constraint. An activity variable that takes both positive and negative values can be interpreted as the difference between a firing rate and a fixed background rate, or between the firing rates of two neurons being treated as a single unit. Finally, to avoid extraneous conversion factors in our equations, we take u and v to be dimensionless measures of the corresponding neuronal firing rates or activities. For example, u and v could be the firing rates of the pre- and postsynaptic neurons divided by their maximum or average values.

In the first part of this chapter, we consider unsupervised learning as applied to a single postsynaptic neuron driven by N_u presynaptic inputs with activities represented by u_b for $b = 1, 2, \ldots, N_u$, or collectively by the vector $\mathbf{u}$. In unsupervised learning, the postsynaptic activity v is evoked directly by the presynaptic activity $\mathbf{u}$. We describe v using a linear version of the firing-rate model discussed in chapter 7,

$$\tau_r \frac{dv}{dt} = -v + \mathbf{w} \cdot \mathbf{u} = -v + \sum_{b=1}^{N_u} w_b u_b \,, \tag{8.1}$$

where τ_r is a time constant that controls the firing-rate response dynamics. Recall that w_b is the synaptic weight that describes the strength of the synapse from presynaptic neuron b to the postsynaptic neuron, and $\mathbf{w}$ is the vector formed by all N_u synaptic weights. The individual synaptic weights can be either positive, representing excitation, or negative, representing inhibition. Equation 8.1 does not include any nonlinear dependence of the firing rate on the total synaptic input, not even rectification. Using such a linear firing-rate model considerably simplifies the analysis of synaptic plasticity. The restriction to nonnegative v either will be imposed by hand or, sometimes, will be ignored to simplify the analysis.

weight vector $\mathbf{w}$

The processes of synaptic plasticity are typically much slower than the dynamics characterized by equation 8.1. If, in addition, the stimuli are presented slowly enough to allow the network to attain its steady-state activity during training, we can replace the dynamic equation 8.1 by

$$v = \mathbf{w} \cdot \mathbf{u} \,, \tag{8.2}$$

which instantaneously sets v to the asymptotic steady-state value determined by equation 8.1. This is the equation we primarily use in our analysis of synaptic plasticity in unsupervised learning. Synaptic modification is included in the model by specifying how the vector $\mathbf{w}$ changes as a function of the pre- and postsynaptic levels of activity. The complex time course of plasticity seen in figure 8.1 is simplified by modeling only the longer-lasting changes.

The Basic Hebb Rule

The simplest plasticity rule that follows the spirit of Hebb's conjecture takes the form

$$\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u}\,, \tag{8.3}$$

basic Hebb rule

where τ_w is a time constant that controls the rate at which the weights change. This equation, which we call the basic Hebb rule, implies that simultaneous pre- and postsynaptic activity increases synaptic strength. If the activity variables represent firing rates, the right side of this equation can be interpreted as a measure of the probability that the pre- and postsynaptic neurons both fire spikes during a small time interval.

Synaptic plasticity is generally modeled as a slow process that gradually modifies synaptic weights over a time period during which the components of $\mathbf{u}$ take a variety of different values. Each different set of $\mathbf{u}$ values is called an input pattern. The direct way to compute the weight changes induced by a series of input patterns is to sum the small changes caused by each of them separately. A convenient alternative is to average over all of the different input patterns and compute the weight changes induced by this average. As long as the synaptic weights change slowly enough, the averaging method provides a good approximation of the weight changes produced by the set of input patterns.

In this chapter, we use angle brackets $\langle\ \rangle$ to denote averages over the ensemble of input patterns presented during training (which is a slightly different usage from earlier chapters). The Hebb rule of equation 8.3, when averaged over the inputs used during training, becomes

averaged Hebb rule

$$\tau_w \frac{d\mathbf{w}}{dt} = \langle v\mathbf{u}\rangle\,. \tag{8.4}$$

In unsupervised learning, v is determined by equation 8.2, and if we replace v with $\mathbf{w}\cdot\mathbf{u}$, we can rewrite the averaged plasticity rule (equation 8.4) as

correlation-based rule

$$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{Q}\cdot\mathbf{w} \quad \text{or} \quad \tau_w \frac{dw_b}{dt} = \sum_{b'=1}^{N_u} Q_{bb'} w_{b'}\,, \tag{8.5}$$

input correlation matrix $\mathbf{Q}$

where $\mathbf{Q}$ is the input correlation matrix given by

$$\mathbf{Q} = \langle \mathbf{u}\mathbf{u}\rangle \quad \text{or} \quad Q_{bb'} = \langle u_b u_{b'}\rangle\,. \tag{8.6}$$

Equation 8.5 is called a correlation-based plasticity rule because of the presence of the input correlation matrix.

Whether or not the pre- and postsynaptic activity variables are restricted to nonnegative values, the basic Hebb rule is unstable. To show this, we consider the square of the length of the weight vector, $|\mathbf{w}|^2 = \mathbf{w}\cdot\mathbf{w} = \sum_b w_b^2$. Taking the dot product of equation 8.3 with $\mathbf{w}$, and noting that $d|\mathbf{w}|^2/dt =$

$2\mathbf{w} \cdot d\mathbf{w}/dt$ and that $\mathbf{w} \cdot \mathbf{u} = v$, we find that $\tau_w d|\mathbf{w}|^2/dt = 2v^2$, which is always positive (except in the trivial case $v = 0$). Thus, the length of the weight vector grows continuously when the rule 8.3 is applied. To avoid unbounded growth, we must impose an upper saturation constraint. A lower limit is also required if the activity variables are allowed to be negative. Even with saturation, the basic Hebb rule fails to induce competition between different synapses.

Sometimes, synaptic modification is modeled as a discrete rather than continuous process, particularly if the learning procedure involves the sequential presentation of inputs. In this case, equation 8.5 is replaced by a discrete updating rule

$$\mathbf{w} \to \mathbf{w} + \epsilon \mathbf{Q} \cdot \mathbf{w} \tag{8.7}$$

where ϵ is a parameter, analogous to the learning rate $1/\tau_w$ in the continuous rule, that determines the amount of modification per application of the rule.

The Covariance Rule

If, as in Hebb's original conjecture, u and v are interpreted as representing firing rates (which must be positive), the basic Hebb rule describes only LTP. Experiments, such as the one shown in figure 8.1, indicate that synapses can depress in strength if presynaptic activity is accompanied by a low level of postsynaptic activity. High levels of postsynaptic activity, on the other hand, produce potentiation. These results can be modeled by a synaptic plasticity rule of the form

$$\tau_w \frac{d\mathbf{w}}{dt} = (v - \theta_v)\mathbf{u}\,, \tag{8.8}$$

where θ_v is a threshold that determines the level of postsynaptic activity above which LTD switches to LTP. As an alternative to equation 8.8, we can impose the threshold on the input rather than output activity, and write

postsynaptic threshold θ_v

$$\tau_w \frac{d\mathbf{w}}{dt} = v(\mathbf{u} - \boldsymbol{\theta}_u)\,. \tag{8.9}$$

Here, $\boldsymbol{\theta}_u$ is a vector of thresholds that determines the levels of presynaptic activities above which LTD switches to LTP. It is also possible to combine these two rules by subtracting thresholds from both the $\mathbf{u}$ and v terms, but this has the undesirable feature of predicting LTP when pre- and postsynaptic activity levels are both low.

presynaptic threshold $\boldsymbol{\theta}_u$

A convenient choice for the thresholds is the average value of the corresponding variable over the training period. In other words, we set the threshold in equation 8.8 to the average postsynaptic activity, $\theta_v = \langle v \rangle$, or the threshold vector in equation 8.9 to the average presynaptic activity vector, $\boldsymbol{\theta}_u = \langle \mathbf{u} \rangle$. As we did for equation 8.5, we use the relation $v = \mathbf{w} \cdot \mathbf{u}$ and

average over training inputs to obtain an averaged form of the plasticity rule. When the thresholds are set to their corresponding activity averages, equations 8.8 and 8.9 both produce the same averaged rule,

input covariance matrix **C**

$$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{C} \cdot \mathbf{w}\,, \tag{8.10}$$

where **C** is the input covariance matrix,

$$\mathbf{C} = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)(\mathbf{u} - \langle \mathbf{u} \rangle) \rangle = \langle \mathbf{u}\mathbf{u} \rangle - \langle \mathbf{u} \rangle^2 = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)\mathbf{u} \rangle\ . \tag{8.11}$$

covariance rules

Because of the presence of the covariance matrix in equation 8.10, equations 8.8 and 8.9 are known as covariance rules.

homosynaptic and heterosynaptic depression

Although they both average to give equation 8.10, the rules in equations 8.8 and 8.9 have their differences. Equation 8.8 modifies synapses only if they have nonzero presynaptic activities. When $v < \theta_v$, this produces an effect called homosynaptic depression. In contrast, equation 8.9 reduces the strengths of inactive synapses if $v > 0$. This is called heterosynaptic depression. Note that maintaining $\theta_v = \langle v \rangle$ in equation 8.8 requires changing θ_v as the weights are modified. In contrast, the threshold in equation 8.9 is independent of the weights and does not need to be changed during the training period to keep $\boldsymbol{\theta}_u = \langle \mathbf{u} \rangle$.

Even though covariance rules include LTD and thus allow weights to decrease, they are unstable because of the same positive feedback that makes the basic Hebb rule unstable. For either rule 8.8 with $\theta_v = \langle v \rangle$ or rule 8.9 with $\boldsymbol{\theta}_u = \langle \mathbf{u} \rangle$, $\tau_w d|\mathbf{w}|^2/dt = 2v(v - \langle v \rangle)$. The time average of the right side of this equation is proportional to the variance of the output, $\langle v^2 \rangle - \langle v \rangle^2$, which is positive except in the trivial case when v is constant. Also similar to the case of the Hebb rule is the fact that the covariance rules are noncompetitive, but competition can be introduced by allowing the thresholds to slide, as described below.

The BCM Rule

The covariance-based rule of equation 8.8 does not require any postsynaptic activity to produce LTD, and rule 8.9 can produce LTD without presynaptic activity. Bienenstock, Cooper, and Munro (1982) suggested an alternative plasticity rule, for which there is experimental evidence, that requires both pre- and postsynaptic activity to change a synaptic weight. This rule, which is called the BCM rule, takes the form

BCM rule

$$\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u}\,(v - \theta_v)\ . \tag{8.12}$$

As in equation 8.8, θ_v acts as a threshold on the postsynaptic activity that determines whether synapses are strengthened or weakened.

If the threshold θ_v is held fixed, the BCM rule, like the basic Hebbian rule, is unstable. Synaptic modification can be stabilized against unbounded

growth by allowing the threshold to vary. The critical condition for stability is that θ_v must grow more rapidly than v as the output activity grows large. In one instantiation of the BCM rule with a sliding threshold, θ_v acts as a low-pass filtered version of v^2, as determined by the equation

sliding threshold

$$\tau_\theta \frac{d\theta_v}{dt} = v^2 - \theta_v \, . \tag{8.13}$$

Here τ_θ sets the time scale for modification of the threshold. This is usually slower than the presentation of individual presynaptic patterns, but faster than the rate at which the weights change, which is determined by τ_w. With a sliding threshold, the BCM rule implements competition between synapses because strengthening some synapses increases the postsynaptic firing rate, which raises the threshold and makes it more difficult for other synapses to be strengthened or even to remain at their current strengths.

Synaptic Normalization

The BCM rule stabilizes Hebbian plasticity by means of a sliding threshold that reduces synaptic weights if the postsynaptic neuron becomes too active. This amounts to using the postsynaptic activity as an indicator of the strengths of synaptic weights. A more direct way to stabilize a Hebbian plasticity rule is to add terms that depend explicitly on the weights. This typically leads to some form of weight normalization, which corresponds to the idea that postsynaptic neurons can support only a fixed total synaptic weight, so increases in some weights must be accompanied by decreases in others.

Normalization of synaptic weights involves imposing some sort of global constraint. Two types of constraints are typically used. If the synaptic weights are nonnegative, their growth can be limited by holding the sum of all the weights of the synapses onto a given postsynaptic neuron to a constant value. An alternative, which also works for weights that can be either positive or negative, is to constrain the sum of the squares of the weights instead of their linear sum. In either case, the constraint can be imposed either rigidly, requiring that it be satisfied at all times during the training process, or dynamically, requiring only that it be satisfied asymptotically at the end of training. We discuss one example of each type: a rigid scheme for imposing a constraint on the sum of synaptic weights, and a dynamic scheme for constraining the sum over their squares. Dynamic constraints can be applied in the former case and rigid constraints in the latter, but we restrict our discussion to two widely used schemes. We discuss synaptic normalization in connection with the basic Hebb rule, but the results we present can be applied to covariance rules as well. Weight normalization can drastically alter the outcome of a training procedure, and different normalization methods may lead to different outcomes.

Subtractive Normalization

The sum over synaptic weights that is constrained by subtractive normalization can be written as $\sum w_b = \mathbf{n} \cdot \mathbf{w}$ where $\mathbf{n}$ is an N_u-dimensional vector with all its components equal to 1 (as introduced in chapter 7). This sum can be constrained by replacing equation 8.3 with

Hebb rule with subtractive normalization

$$\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u} - \frac{v(\mathbf{n} \cdot \mathbf{u})\mathbf{n}}{N_u} . \tag{8.14}$$

Note that $\mathbf{n} \cdot \mathbf{u}$ is simply the sum of all the inputs. This rule imposes what is called subtractive normalization because the same quantity is subtracted from the change to each weight, whether that weight is large or small. Subtractive normalization imposes the constraint on the sum of weights rigidly because it does not allow the Hebbian term to change $\mathbf{n} \cdot \mathbf{w}$. To see this, we take the dot product of equation 8.14 with $\mathbf{n}$ to obtain

$$\tau_w \frac{d\mathbf{n} \cdot \mathbf{w}}{dt} = v\mathbf{n} \cdot \mathbf{u} \left(1 - \frac{\mathbf{n} \cdot \mathbf{n}}{N_u}\right) = 0 . \tag{8.15}$$

The last equality follows because $\mathbf{n} \cdot \mathbf{n} = N_u$. Hebbian modification with subtractive normalization is nonlocal in that it requires the sum of all the input activities, $\mathbf{n} \cdot \mathbf{u}$, to be available to the mechanism that modifies each synapse. It is not obvious how such a rule could be implemented biophysically.

Subtractive normalization must be augmented by a saturation constraint that prevents weights from becoming negative. If the rule 8.14 attempts to drive any of the weights below 0, the saturation constraint prevents this change. At this point, the rule is not applied to any saturated weights, and its effect on the other weights is modified. Both modifications can be achieved by setting the components of the vector $\mathbf{n}$ corresponding to any saturated weights to 0 and the factor of N_u in equation 8.14 equal to the sum of the components of this modified $\mathbf{n}$ vector (i.e., the number of unsaturated components). Without any upper saturation limit, this procedure often results in a final outcome in which all the weights but one have been set to 0. To avoid this, an upper saturation limit is also typically imposed. Hebbian plasticity with subtractive normalization is highly competitive because small weights are reduced by a larger proportion of their sizes than are large weights.

Multiplicative Normalization and the Oja Rule

A constraint on the sum of the squares of the synaptic weights can be imposed dynamically by using a modification of the basic Hebb rule known as the Oja rule (Oja, 1982),

Oja rule

$$\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u} - \alpha v^2 \mathbf{w} , \tag{8.16}$$

where α is a positive constant. This rule involves only information that is local to the synapse being modified (namely, the pre- and postsynaptic activities and the local synaptic weight), but its form is based more on theoretical arguments than on experimental data. The normalization it imposes is called multiplicative because the amount of modification induced by the second term in equation 8.16 is proportional to $\mathbf{w}$.

The stability of the Oja rule can be established by taking the dot product of equation 8.16 with the weight vector $\mathbf{w}$ to obtain

$$\tau_w \frac{d|\mathbf{w}|^2}{dt} = 2v^2(1 - \alpha|\mathbf{w}|^2) . \tag{8.17}$$

This indicates that $|\mathbf{w}|^2$ will relax over time to the value $1/\alpha$, which obviously prevents the weights from growing without bound, proving stability. It also induces competition between the different weights, because when one weight increases, the maintenance of a constant length for the weight vector forces other weights to decrease.

Timing-Based Rules

Experiments have shown that the relative timing of pre- and postsynaptic action potentials plays a critical role in determining the sign and amplitude of the changes in synaptic efficacy produced by activity. Figure 8.2 shows examples from an intracellular recording of a pair of cortical pyramidal cells in a slice experiment, and from an in vivo experiment on retinotectal synapses in a *Xenopus* tadpole. Both experiments involve repeated pairing of pre- and postsynaptic action potentials, and both show that the relative timing of these spikes is critical in determining the amount and type of synaptic modification that takes place. Synaptic plasticity occurs only if the difference in the pre- and postsynaptic spike times falls within a window of roughly ± 50 ms. Within this window, the sign of the synaptic modification depends on the order of stimulation. Presynaptic spikes that precede postsynaptic action potentials produce LTP. Presynaptic spikes that follow postsynaptic action potentials produce LTD. This is in accord with Hebb's original conjecture, because a synapse is strengthened only when a presynaptic action potential precedes a postsynaptic action potential and therefore can be interpreted as having contributed to it. When the order is reversed and the presynaptic action potential could not have contributed to the postsynaptic response, the synapse is weakened. The maximum amount of synaptic modification occurs when the paired spikes are separated by only a few milliseconds, and the evoked plasticity decreases to 0 as this separation increases.

Simulating the spike-timing dependence of synaptic plasticity requires a spiking model. However, an approximate model can be constructed on the basis of firing rates. The effect of pre- and postsynaptic timing can be included in a synaptic modification rule by including a temporal difference

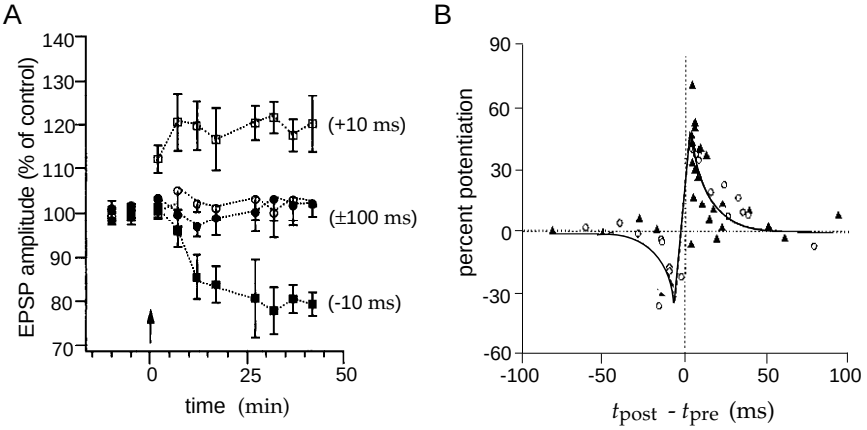

Figure 8.2 LTP and LTD produced by 50 to 75 pairs of pre- and postsynaptic action potential with various timings. (A) The amplitude of the excitatory postsynaptic potential (EPSP) evoked by stimulation of the presynaptic neuron plotted at various times as a percentage of the amplitude prior to paired stimulation. At the time indicated by the arrow, paired stimulations of the presynaptic and postsynaptic neurons were performed. For the traces marked by open symbols, the presynaptic spike occurred either 10 or 100 ms before the postsynaptic neuron fired an action potential. The traces marked by solid symbols denote the reverse ordering in which the presynaptic spike occurred either 10 or 100 ms after the postsynaptic spike. Separations of 100 ms had no long-lasting effect. In contrast, the 10 ms delays produced effects that built up to a maximum over a 5-to-10-minute period and lasted for the duration of the experiment. Pairing a presynaptic action potential with a postsynaptic action potential 10 ms later produced LTP, whereas the reverse ordering generated LTD. (B) LTP and LTD of retinotectal synapses recorded in vivo in *Xenopus* tadpoles. The percent change in synaptic strength produced by multiple pairs of action potentials is plotted as a function of their time difference. The filled symbols correspond to extracellular stimulation of the postsynaptic neuron, and the open symbols, to intracellular stimulation. The H function in equation 8.18 is proportional to the solid curve. (A adapted from Markram et al., 1997; B adapted from Zhang et al., 1998.)

τ between the times when the firing rates of the pre- and postsynaptic neurons are evaluated. A function $H(\tau)$ determines the rate of synaptic modification that occurs due to postsynaptic activity separated in time from presynaptic activity by an interval τ. The total rate of synaptic modification is determined by integrating over all time differences τ. If we assume that the rate of synaptic modification is proportional to the product of the pre- and postsynaptic rates, as it is for a Hebbian rule, the rate of change of the synaptic weights at time t is given by

timing-based rule

$$\tau_w \frac{d\mathbf{w}}{dt} = \int_0^\infty d\tau \; (H(\tau)v(t)\mathbf{u}(t-\tau) + H(-\tau)v(t-\tau)\mathbf{u}(t)) \; . \qquad (8.18)$$

If $H(\tau)$ is positive for positive τ and negative for negative τ, the first term on the right side of this equation represents LTP, and the second, LTD. The solid curve in figure 8.2B is an example of such an H function. The temporal asymmetry of H has a dramatic effect on synaptic weight changes because it causes them to depend on the temporal order of the pre- and

postsynaptic activity during training. Among other things, this allows synaptic weights to store information about temporal sequences.

Rules in which synaptic plasticity is based on the relative timing of pre- and postsynaptic action potentials still require saturation constraints for stability, but, in spiking models, they can generate competition between synapses without further constraints or modifications. This is because different synapses compete to control the timing of postsynaptic spikes. Synapses that are able to evoke postsynaptic spikes rapidly get strengthened. These synapses then exert a more powerful influence on the timing of postsynaptic spikes, and they tend to generate spikes at times that lead to the weakening of other synapses that are less capable of controlling postsynaptic spike timing.

8.3 Unsupervised Learning

We now consider the computational properties of the different synaptic modification rules we have introduced in the context of unsupervised learning. Unsupervised learning provides a model for the effects of activity on developing neural circuits and the effects of experience on mature networks. We separate the discussion of unsupervised learning into cases involving a single postsynaptic neuron and cases in which there are multiple postsynaptic neurons.

A major area of research in unsupervised learning concerns the development of neuronal selectivity and the formation of cortical maps. Chapters 1 and 2 provided examples of the selectivities of neurons in various cortical areas to features of the stimuli to which they respond. In many cases, neuronal selectivities are arranged across the cortical surface in an orderly and regular pattern known as a cortical map. The patterns of connection that give rise to neuronal selectivities and cortical maps are established during development by both activity-independent and activity-dependent processes. A conventional view is that activity-independent mechanisms control the initial targeting of axons, determine the appropriate layer for them to innervate, and establish a coarse order or patterning in their projections. Other activity-independent and activity-dependent mechanisms then refine this order and help to create and preserve neuronal selectivities and cortical maps.

cortical map

Although the relative roles of activity-independent and activity-dependent processes in cortical development are the subject of extensive debate, developmental models based on activity-dependent plasticity have played an important role in suggesting key experiments and successfully predicting their outcomes. A detailed analysis of the more complex pattern-forming models that have been proposed is beyond the scope of this book. Instead, in this and later sections, we give a brief overview of some different approaches and the results that have been obtained. In this section, we consider the case of ocular dominance.

ocular dominance

Ocular dominance refers to the tendency of input to neurons in the adult primary visual cortex of many mammalian species to favor one eye over the other. This is especially true for neurons in layer 4, which receive extensive innervation from the LGN. Neurons dominated by one eye or the other occupy different patches of cortex, and areas with left- or right-eye ocular dominance alternate across the cortex in fairly regular bands known as ocular dominance stripes. In a later section, we discuss how this cortical map can arise from Hebbian plasticity.

ocular dominance stripes

Single Postsynaptic Neuron

Equation 8.5, which shows the consequence of averaging the basic Hebb rule over all the presynaptic training patterns, is a linear equation for $\mathbf{w}$. Provided that we ignore any constraints on $\mathbf{w}$, it can be analyzed using standard techniques for solving differential equations (see chapter 7 and the Mathematical Appendix). In particular, we use the method of matrix diagonalization, which involves expressing $\mathbf{w}$ in terms of the eigenvectors of $\mathbf{Q}$. These are denoted by $\mathbf{e}_\mu$ with $\mu = 1, 2, \cdots, N_u$, and they satisfy $\mathbf{Q} \cdot \mathbf{e}_\mu = \lambda_\mu \mathbf{e}_\mu$. For correlation or covariance matrices, all the eigenvalues, λ_μ for all μ, are real and nonnegative, and for convenience we order them so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N_u}$.

Any N_u-dimensional vector can be represented using the eigenvectors as a basis, so we can write

$$\mathbf{w}(t) = \sum_{\mu=1}^{N_u} c_\mu(t)\mathbf{e}_\mu \,, \tag{8.19}$$

where the coefficients are equal to the dot products of $\mathbf{w}$ with the corresponding eigenvectors. For example, at time 0, $c_\mu(0) = \mathbf{w}(0) \cdot \mathbf{e}_\mu$. Writing $\mathbf{w}$ as a sum of eigenvectors turns matrix multiplication into ordinary multiplication, making calculations easier. Substituting the expansion 8.19 into 8.5 and following the procedure presented in chapter 7 for isolating uncoupled equations for the coefficients, we find that $c_\mu(t) = c_\mu(0)\exp(\lambda_\mu t/\tau_w)$. Going back to equation 8.19, this means that

$$\mathbf{w}(t) = \sum_{\mu=1}^{N_u} \exp\left(\frac{\lambda_\mu t}{\tau_w}\right)\left(\mathbf{w}(0) \cdot \mathbf{e}_\mu\right)\mathbf{e}_\mu \,. \tag{8.20}$$

The exponential factors in 8.20 all grow over time, because the eigenvalues λ_μ are positive for all μ values. For large t, the term with the largest value of λ_μ (assuming it is unique) becomes much larger than any of the other terms and dominates the sum for $\mathbf{w}$. This largest eigenvalue has the label $\mu = 1$, and its corresponding eigenvector $\mathbf{e}_1$ is called the principal eigenvector. Thus, for large t, $\mathbf{w} \propto \mathbf{e}_1$ to a good approximation, provided that $\mathbf{w}(0) \cdot \mathbf{e}_1 \neq 0$. After training, the response to an arbitrary input vector $\mathbf{u}$ is well approximated by

principal eigenvector

$$v \propto \mathbf{e}_1 \cdot \mathbf{u} \,. \tag{8.21}$$

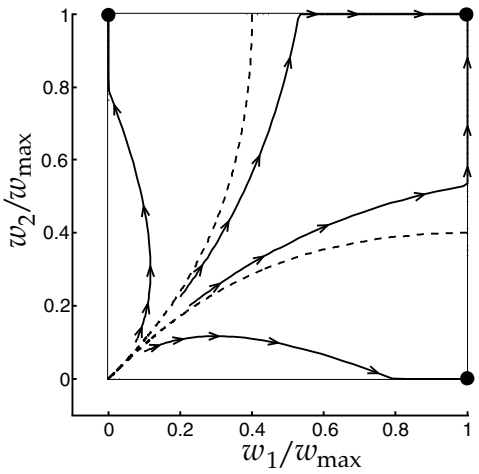

Figure 8.3 Hebbian weight dynamics with saturation. The correlation matrix of the input vectors had diagonal elements equal to 1 and off-diagonal elements of -0.4. The principal eigenvector, $\mathbf{e}_1 = (1, -1)/\sqrt{2}$, dominates the dynamics if the initial values of the weights are small enough (below or to the left of the dashed lines). This makes the weight vector move to the corners $(w_{\max}, 0)$ or $(0, w_{\max})$. However, starting the weights with larger values (between the dashed lines) allows saturation to occur at the corner $(w_{\max}, w_{\max})$. (Adapted from MacKay & Miller, 1990.)

Because the dot product corresponds to a projection of one vector onto another, Hebbian plasticity can be interpreted as producing an output proportional to the projection of the input vector onto the principal eigenvector of the correlation matrix of the inputs used during training. We discuss the significance of this result in the next section.

The proportionality sign in equation 8.21 hides the large multiplicative factor $\exp(\lambda_1 t/\tau_w)$ that is a consequence of the positive feedback inherent in Hebbian plasticity. One way to limit growth of the weight vector in equation 8.5 is to impose a saturation constraint. This can have significant effects on the outcome of Hebbian modification, including, in some cases, preventing the weight vector from ending up proportional to the principal eigenvector. Figure 8.3 shows examples of the Hebbian development of the weights in a case with just two inputs. For the correlation matrix used in this example, the principal eigenvector is $\mathbf{e}_1 = (1, -1)/\sqrt{2}$, so an analysis that ignored saturation would predict that one weight would increase while the other decreased. Which weight moves in which direction is controlled by the initial conditions. Given the constraints, this would suggest that $(w_{\max}, 0)$ and $(0, w_{\max})$ are the most likely final configurations. This analysis gives the correct answer only for the regions in figure 8.3 below or to the left of the dashed lines. Between the dashed lines, the final state is $\mathbf{w} = (w_{\max}, w_{\max})$ because the weights hit the saturation boundary before the exponential growth is large enough to allow the principal eigenvector to dominate.

Another way to eliminate the large exponential factor in the weights is to

use the Oja rule, 8.16, instead of the basic Hebb rule. The weight vector generated by the Oja rule, in the example we have discussed, approaches $\mathbf{w} = \mathbf{e}_1/\alpha^{1/2}$ as $t \to \infty$. In other words, the Oja rule gives a weight vector that is parallel to the principal eigenvector, but normalized to a length of $1/\alpha^{1/2}$ rather than growing without bound.

averaged Hebb rule with subtractive constraint

Finally, we examine the effect of including a subtractive constraint, as in equation 8.14. Averaging equation 8.14 over the training inputs, we find

$$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{Q} \cdot \mathbf{w} - \frac{(\mathbf{w} \cdot \mathbf{Q} \cdot \mathbf{n})\mathbf{n}}{N_u} . \tag{8.22}$$

If we once again express $\mathbf{w}$ as a sum of eigenvectors of $\mathbf{Q}$, we find that the growth of each coefficient in this sum is unaffected by the extra term in equation 8.22, provided that $\mathbf{e}_\mu \cdot \mathbf{n} = 0$. However, if $\mathbf{e}_\mu \cdot \mathbf{n} \neq 0$, the extra term modifies the growth. In our discussion of ocular dominance, we consider a case in which the principal eigenvector of the correlation matrix is proportional to $\mathbf{n}$. In this case, $\mathbf{Q} \cdot \mathbf{e}_1 - (\mathbf{e}_1 \cdot \mathbf{Q} \cdot \mathbf{n})\mathbf{n}/N = 0$, so the projection in the direction of the principal eigenvector is unaffected by the synaptic plasticity rule. Furthermore, $\mathbf{e}_\mu \cdot \mathbf{n} = 0$ for $\mu \geq 2$ because the eigenvectors of the correlation matrix are mutually orthogonal, which implies that the evolution of the remaining eigenvectors is unaffected by the constraint. As a result,

$$\mathbf{w}(t) = (\mathbf{w}(0) \cdot \mathbf{e}_1)\,\mathbf{e}_1 + \sum_{\mu=2}^{N_u} \exp\left(\frac{\lambda_\mu t}{\tau_w}\right)(\mathbf{w}(0) \cdot \mathbf{e}_\mu)\,\mathbf{e}_\mu . \tag{8.23}$$

Thus, ignoring the effects of any possible saturation constraints, the synaptic weight matrix tends to become parallel to the eigenvector with the second largest eigenvalue as $t \to \infty$.

In summary, if weight growth is limited by some form of multiplicative normalization, as in the Oja rule, the configuration of synaptic weights produced by Hebbian modification typically will be proportional to the principal eigenvector of the input correlation matrix. When subtractive normalization is used and the principal eigenvector is proportional to $\mathbf{n}$, the eigenvector with the next largest eigenvalue provides an estimate of the configuration of final weights, again up to a proportionality factor. If, however, saturation constraints are used, as they must be in a subtractive scheme, this can invalidate the results of a simplified analysis based solely on these eigenvectors (as in figure 8.3). Nevertheless, we base our discussion of the Hebbian development of ocular dominance and cortical maps on an analysis of the eigenvectors of the input correlation matrix. We present simulation results to verify that this analysis is not invalidated by the constraints imposed in the full models.

Principal Component Projection

If applied for a long enough time, both the basic Hebb rule (equation 8.3) and the Oja rule (equation 8.16) generate weight vectors that are parallel

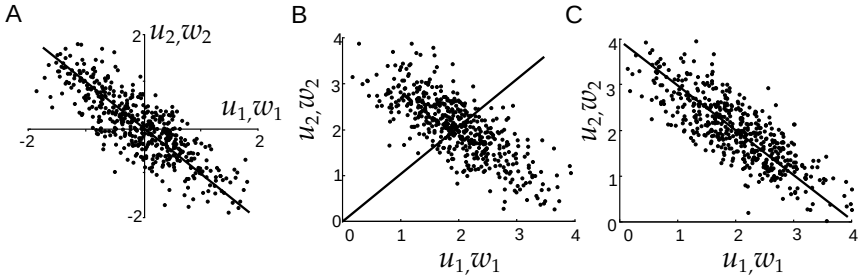

Figure 8.4 Unsupervised Hebbian learning and principal component analysis. The axes in these figures are used to represent the components of both $\mathbf{u}$ and $\mathbf{w}$. (A) The filled circles show the inputs $\mathbf{u} = (u_1, u_2)$ used during a training period while a Hebbian plasticity rule was applied. After training, the vector of synaptic weights was aligned parallel to the solid line. (B) Correlation-based modification with nonzero mean input. Input vectors were generated as in A except that the distribution was shifted to produce an average value $\langle \mathbf{u} \rangle = (2, 2)$. After a training period during which a Hebbian plasticity rule was applied, the synaptic weight vector was aligned parallel to the solid line. (C) Covariance-based modification. Points from the same distribution as in B were used while a covariance-based Hebbian rule was applied. The weight vector becomes aligned parallel to the solid line.

to the principal eigenvector of the correlation matrix of the inputs used during training. Figure 8.4A provides a geometric picture of the significance of this result. In this example, the basic Hebb rule was applied to a unit described by equation 8.2 with two inputs ($N_u = 2$). This model is not constrained to have positive $\mathbf{u}$ and v. The inputs used during the training period were chosen from a two-dimensional Gaussian distribution with unequal variances, resulting in the elliptical distribution of points seen in the figure. The initial weight vector $\mathbf{w}(0)$ was chosen randomly. The two-dimensional weight vector produced by a Hebbian rule is proportional to the principal eigenvector of the input correlation matrix. The line in figure 8.4A indicates the direction along which the final $\mathbf{w}$ lies, with the u_1 and u_2 axes used to represent w_1 and w_2 as well. The weight vector points in the direction along which the cloud of input points has the largest variance, a result with interesting implications.

Any unit that obeys equation 8.2 characterizes the state of its N_u inputs by a single number v, which is equal to the projection of $\mathbf{u}$ onto the weight vector $\mathbf{w}$. Intuition suggests, and a technique known as principal component analysis (PCA) formalizes, that setting the projection direction $\mathbf{w}$ proportional to the principal eigenvector $\mathbf{e}_1$ is often the optimal choice if a set of vectors is to be represented by, and reconstructed from, a set of single numbers through a linear relation. For example, if $\mathbf{e}_1$ is normalized so that $|\mathbf{e}_1| = 1$, the vectors $v\mathbf{w}$ with $v = \mathbf{w} \cdot \mathbf{u}$ and $\mathbf{w} = \mathbf{e}_1$ provide the best estimates that can be generated from single numbers (v) of the set of input vectors $\mathbf{u}$ used to construct $\mathbf{Q}$. Furthermore, the fact that projection along this direction maximizes the variance of the outputs that result can be interpreted using information theory. The entropy of a Gaussian distributed random variable with variance σ^2 grows with increasing variance as $\log_2 \sigma$. For Gaussian input statistics, and output that is corrupted

principal components analysis PCA

by noise that is also Gaussian, maximizing the variance of v by a Hebbian rule maximizes not only the output entropy but also the amount of information v carries about $\mathbf{u}$. In chapter 10, we further consider the computational significance of finding the direction of maximum variance in the input data set, and we discuss the relationship between this and general techniques for extracting structure from input statistics.

Figure 8.4B shows the consequence of applying correlation-based Hebbian plasticity when the average activities of the inputs are not 0, as is inevitable if real firing rates are employed. In this example, correlation-based Hebbian modification aligns the weight vector parallel to a line from the origin to the point $\langle \mathbf{u} \rangle$. This clearly fails to capture the essence of the distribution of inputs. Figure 8.4C shows the result of applying covariance-based Hebbian modification instead. Now the weight vector is aligned with the cloud of data points because this rule finds the direction of the principal eigenvector of the covariance matrix $\mathbf{C}$ of equation 8.11 rather the correlation matrix $\mathbf{Q}$.

Hebbian Development of Ocular Dominance

As an example of a developmental model of ocular dominance at the single neuron level, we consider the highly simplified case of a single layer 4 cell that receives input from just two LGN afferents. One afferent is associated with the right eye and has activity u_R, and the other is from the left eye and has activity u_L. Two synaptic weights $\mathbf{w} = (w_R, w_L)$ describe the strengths of these projections, and the output activity v is determined by equation 8.2,

$$v = w_R u_R + w_L u_L \,. \tag{8.24}$$

The weights in this model are constrained to nonnegative values. Initially, the weights for the right- and left-eye inputs are taken to be approximately equal. Ocular dominance arises when one of the weights is pushed to 0 while the other remains positive.

We can estimate the results of a Hebbian developmental process by considering the input correlation matrix. We assume that the two eyes are statistically equivalent, so the correlation matrix of the right- and left-eye inputs takes the form

$$\mathbf{Q} = \langle \mathbf{u}\mathbf{u} \rangle = \begin{pmatrix} \langle u_R u_R \rangle & \langle u_R u_L \rangle \\ \langle u_L u_R \rangle & \langle u_L u_L \rangle \end{pmatrix} = \begin{pmatrix} q_S & q_D \\ q_D & q_S \end{pmatrix} . \tag{8.25}$$

The subscripts S and D denote same- and different-eye correlations. The eigenvectors are $\mathbf{e}_1 = (1, 1)/\sqrt{2}$ and $\mathbf{e}_2 = (1, -1)/\sqrt{2}$ for this correlation matrix, and their eigenvalues are $\lambda_1 = q_S + q_D$ and $\lambda_2 = q_S - q_D$.

If the right- and left-eye weights evolve according to equation 8.5, it is straightforward to show that the eigenvector combinations $w_+ = w_R + w_L$

and $w_- = w_R - w_L$ obey the uncoupled equations

$$\tau_w \frac{dw_+}{dt} = (q_S + q_D)w_+ \quad \text{and} \quad \tau_w \frac{dw_-}{dt} = (q_S - q_D)w_- . \tag{8.26}$$

Positive correlations between the two eyes are likely to exist ($q_D > 0$) after eye opening has occurred. This means that $q_S + q_D > q_S - q_D$, so, according to equations 8.26, w_+ grows more rapidly than w_-. Equivalently, $\mathbf{e}_1 = (1, 1)/\sqrt{2}$ is the principal eigenvector. The basic Hebbian rule thus predicts a final weight vector proportional to $\mathbf{e}_1$, which implies equal innervation from both eyes. This is not the observed outcome of cortical development.

Figure 8.3 shows, in a different example, that in the presence of constraints certain initial conditions can lead to final configurations that are not proportional to the principal eigenvector. However, in the present case initial conditions with $w_R > 0$ and $w_L > 0$ imply that $w_+ > w_-$, and this coupled with the faster growth of w_+ means that w_+ will always dominate over w_- at saturation. Multiplicative normalization does not change this situation.

To obtain ocular dominance in a Hebbian model, we must impose a constraint that prevents w_+ from dominating the final configuration. One way of doing this is to use equation 8.14, the Hebbian rule with subtractive normalization. This completely eliminates the growth of the weight vector in the direction of $\mathbf{e}_1$ (i.e., the increase of w_+) because, in this case, $\mathbf{e}_1$ is proportional to $\mathbf{n}$. On the other hand, it has no effect on growth in the direction $\mathbf{e}_2$ (i.e., the growth of w_-) because $\mathbf{e}_2 \cdot \mathbf{n} = 0$. Thus, with subtractive normalization, the weight vector grows parallel (or anti-parallel) to the direction $\mathbf{e}_2 = (1, -1)/\sqrt{2}$. The direction of this growth depends on initial conditions through the value of $\mathbf{w}(0) \cdot \mathbf{e}_2 = (w_R(0) - w_L(0))/\sqrt{2}$. If this is positive, w_R increases and w_L decreases; if it is negative, w_L increases and w_R decreases. Eventually, the decreasing weight will hit the saturation limit of 0, and the other weight will stop increasing due to the normalization constraint. At this point, total dominance by one eye or the other has been achieved. This simple model shows that ocular dominance can arise from Hebbian plasticity if there is sufficient competition between the growth of the left- and right-eye weights.

Hebbian Development of Orientation Selectivity

Hebbian models can also account for the development of the orientation selectivity displayed by neurons in primary visual cortex. The model of Hubel and Wiesel for generating an orientation-selective simple cell response by summing linear arrays of alternating ON and OFF LGN inputs was presented in chapter 2. The necessary pattern of LGN inputs can arise from Hebbian plasticity on the basis of correlations between the responses of different LGN cells and competition between ON and OFF units. Such a model can be constructed by considering a simple cell receiving input from ON-center and OFF-center cells of the LGN, and applying Hebbian

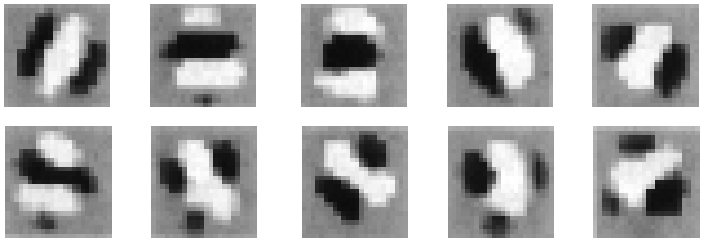

Figure 8.5 Different cortical receptive fields arising from a correlation-based developmental model. White and black regions correspond to areas in the visual field where ON-center (white regions) or OFF-center (black regions) LGN cells excite the cortical neuron being modeled. (Adapted from Miller, 1994.)

plasticity, subject to appropriate constraints, to the feedforward weights of the model.

As in the case of ocular dominance, the development of orientation selectivity can be partially analyzed on the basis of properties of the correlation matrix driving Hebbian development. The critical feature required to produce orientation-selective receptive fields is the growth of components proportional to eigenvectors that are not spatially uniform or rotationally invariant. Application of a Hebbian rule without constraints leads to growth of a uniform component resulting in an unstructured receptive field. Appropriate constraints can eliminate growth of this component, producing spatial structured receptive fields. The development of receptive fields that are not rotationally invariant, and that therefore exhibit orientation selectivity, relies on nonlinear aspects of the model and is therefore difficult to study analytically. For this reason, we simply present simulation results.

arbor function

Neurons in primary visual cortex receive afferents from LGN cells centered over a finite region of the visual field. This anatomical constraint can be included in developmental models by introducing what is called an arbor function. The arbor function, which is often taken to be Gaussian, characterizes the density of innervation from different visual locations to the cell being modeled. As a simplification, this density is not altered during the Hebbian developmental process, but the strengths of synapses within the arbor are modified by the Hebbian rule. The outcome is oriented receptive fields of a limited spatial extent. Figure 8.5 shows the weights resulting from a simulation of receptive-field development using a large array of ON- and OFF-center LGN afferents. This illustrates the variety of oriented receptive field structures that can arise through a Hebbian developmental rule.

Temporal Hebbian Rules and Trace Learning

Unlike correlation- or covariance-based rules, temporal Hebbian rules allow changes of synaptic strength to depend on the temporal sequence of

activity across the synapse. This allows temporal Hebbian rules to exhibit a phenomenon called trace learning, where the term trace refers to the history of synaptic activity.

trace learning

We can approximate the final result of applying a temporal plasticity rule by integrating equation 8.18 from $t = 0$ to a large final time $t = T$, assuming that $w = 0$ initially, and shifting the integration variable to obtain

$$\mathbf{w} = \frac{1}{\tau_w} \int_0^T dt\, v(t) \int_{-\infty}^{\infty} d\tau\, H(\tau)\mathbf{u}(t-\tau)\,. \tag{8.27}$$

The approximation comes from ignoring both small contributions associated with the end points of the integral and the change in $\mathbf{v}$ produced during training by the modification of $\mathbf{w}$. Equation 8.27 shows that temporally dependent Hebbian plasticity depends on the correlation between the postsynaptic activity and the presynaptic activity temporally filtered by the function H.

Equation 8.27 (with a suitably chosen H) can be used to model the development of invariant responses. Neurons in inferotemporal cortex, for example, can respond selectively to particular objects independent of their location within a wide receptive field. The idea underlying the application of equation 8.27 is that objects persist in the visual environment for characteristic lengths of time as their images move across the retina. If the plasticity rule in equation 8.27 filters presynaptic activity over this persistence time, it strengthens synapses from all the presynaptic units that are activated by the image of the object while it persists and moves. As a result, the response of the postsynaptic cell comes to be independent of the position of the object, and position-invariant responses can be generated.

Multiple Postsynaptic Neurons

To study the effect of plasticity on multiple neurons, we introduce the network of figure 8.6, in which N_v output neurons receive input through N_u feedforward connections and from recurrent interconnections. A vector $\mathbf{v}$ represents the activities of the multiple output units, and the feedforward synaptic connections are described by a matrix $\mathbf{W}$, with the element W_{ab} giving the strength and sign of the synapse from input unit b to output unit a. The strength of the recurrent connection from output unit a' to output unit a is described by element $M_{aa'}$ of the recurrent weight matrix $\mathbf{M}$.

feedforward weight matrix $\mathbf{W}$

recurrent weight matrix $\mathbf{M}$

The activity vector for the output units of the network in figure 8.6 is determined by a linear version of the recurrent model of chapter 7,

$$\tau_r \frac{d\mathbf{v}}{dt} = -\mathbf{v} + \mathbf{W}\cdot\mathbf{u} + \mathbf{M}\cdot\mathbf{v}\,. \tag{8.28}$$

Provided that the real parts of the eigenvalues of $\mathbf{M}$ are less than 1, this equation has a stable fixed point with a steady-state output activity vector

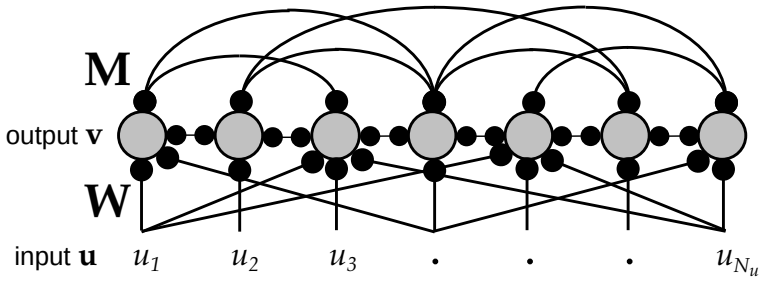

Figure 8.6 A network with multiple output units driven by feedforward synapses with weights **W**, and interconnected by recurrent synapses with weights **M**.

determined by

$$\mathbf{v} = \mathbf{W} \cdot \mathbf{u} + \mathbf{M} \cdot \mathbf{v} \,. \tag{8.29}$$

$\mathbf{K} = (\mathbf{I} - \mathbf{M})^{-1}$

Equation 8.29 can be solved by defining the matrix inverse $\mathbf{K} = (\mathbf{I} - \mathbf{M})^{-1}$, where **I** is the identity matrix, to obtain

$$\mathbf{v} = \mathbf{K} \cdot \mathbf{W} \cdot \mathbf{u} \quad \text{or} \quad v_a = \sum_{a'=1}^{N_v} \sum_{b=1}^{N_u} K_{aa'} W_{a'b} u_b \,. \tag{8.30}$$

It is important for different output neurons in a multi-unit network to be selective for different aspects of the input, or else their responses will be redundant. For the case of a single cell, competition between different synapses could be used to ensure that synapse-specific plasticity rules do not modify all of the synapses onto a postsynaptic neuron in the same way. For multiple output networks, fixed or plastic linear or nonlinear recurrent interactions can be used to ensure that the units do not all develop the same selectivity. In the following sections, we consider three different patterns of plasticity: plastic feedforward and fixed recurrent synapses, plastic feedforward and recurrent synapses, and, finally, fixed feedforward and plastic recurrent synapses.

Hebbian Development of Ocular Dominance Stripes

Ocular dominance stripes can arise in a Hebbian model with plastic feedforward but fixed recurrent synapses. In the single-cell model of ocular dominance considered previously, the ultimate ocular preference of the output cell depends on the initial conditions of its synaptic weights. A multiple-output version of the model without any recurrent connections would therefore generate a random pattern of selectivities across the cortex if it started with random weights. Figure 8.7B shows that ocular dominance is actually organized in a highly structured cortical map. Dominance by the left and right eye alternates across the cortex, forming a striped pattern. Such a map can arise in the context of Hebbian development of feedforward weights if we include a fixed intracortical connection matrix **M**.

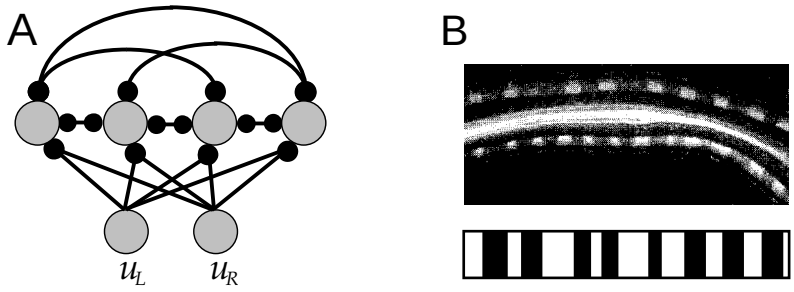

Figure 8.7 The development of ocular dominance in a Hebbian model. (A) The simplified model in which right- and left- eye inputs from a single retinal location drive an array of cortical neurons. (B) Ocular dominance maps. The upper panel shows an area of cat primary visual cortex radioactively labeled to distinguish regions activated by one eye or the other. The light and dark areas along the cortical regions at the top and bottom indicate alternating right- and left-eye innervation. The central region is white matter where fibers are not segregated by ocular dominance. The lower panel shows the pattern of innervation for a 512 unit model after Hebbian development. White and black regions denote units dominated by right- and left-eye projections respectively. (B data of S. LeVay, adapted from Nicholls et al., 1992.)

With fixed recurrent weights $\mathbf{M}$ and plastic feedforward weights $\mathbf{W}$, the effect of averaging Hebbian modifications over the training inputs is

$$\tau_w \frac{d\mathbf{W}}{dt} = \langle \mathbf{v}\mathbf{u} \rangle = \mathbf{K} \cdot \mathbf{W} \cdot \mathbf{Q}\,, \tag{8.31}$$

where $\mathbf{Q} = \langle \mathbf{u}\mathbf{u} \rangle$ is the input autocorrelation matrix. Equation 8.31 has the same form as the single unit equation 8.5, except that both $\mathbf{K}$ and $\mathbf{Q}$ affect the growth of $\mathbf{W}$.

We consider a highly simplified model of the development of ocular dominance maps that considers only a single direction across the cortex and a single point in the visual field. Figure 8.7A shows the simplified model, which has only two input activities, u_R and u_L, that have the correlation matrix of equation 8.25. These are connected to multiple output units through weight vectors $\mathbf{w}_\mathrm{R}$ and $\mathbf{w}_\mathrm{L}$. The output units are connected to each other through weights $\mathbf{M}$, so $\mathbf{v} = \mathbf{w}_\mathrm{R} u_\mathrm{R} + \mathbf{w}_\mathrm{L} u_\mathrm{L} + \mathbf{M} \cdot \mathbf{v}$. The index a denoting the identity of a given output unit also represents the location of that unit on the cortex. This linking of a to locations on the cortical surface allows us to interpret the results of the model in terms of a cortical map.

Writing $\mathbf{w}_+ = \mathbf{w}_\mathrm{R} + \mathbf{w}_\mathrm{L}$ and $\mathbf{w}_- = \mathbf{w}_\mathrm{R} - \mathbf{w}_\mathrm{L}$, the equivalent of equation 8.26 for these sum and difference vectors is

$$\tau_w \frac{d\mathbf{w}_+}{dt} = (q_\mathrm{S} + q_\mathrm{D})\mathbf{K} \cdot \mathbf{w}_+ \qquad \tau_w \frac{d\mathbf{w}_-}{dt} = (q_\mathrm{S} - q_\mathrm{D})\mathbf{K} \cdot \mathbf{w}_-\,. \tag{8.32}$$

As in the single-cell model of ocular dominance, subtractive normalization, which holds the value of $\mathbf{w}_+$ fixed while leaving the growth of $\mathbf{w}_-$ unaffected, eliminates the tendency for the cortical cells to become binocular. Then only the equation for $\mathbf{w}_-$ is relevant, and the growth of $\mathbf{w}_-$ is

dominated by the principal eigenvector of **K**. The sign of the component $[\mathbf{w}_-]_a$ determines whether the neuron in the region of the cortex corresponding to the label a is dominated by the right eye (if it is positive) or the left eye (if it is negative). Oscillations in the signs of the components of $\mathbf{w}_-$ translate into ocular dominance stripes.

We assume that the connections between the output neurons are translation invariant, so that $\mathbf{K}_{aa'} = K(|a - a'|)$ depends only on the separation between the cortical cells a and a'. Note that it is more convenient to consider the form of the function K than to discuss the connections between output neurons (**M**) directly. We use a convenient trick to remove edge effects, which is to impose periodic boundary conditions that require the activities of the units with $a = 0$ and $a = N_v$ to be identical. This provides a reasonable model of a patch of the cortex away from regional boundaries. In some cases, edge effects can impose important constraints on the overall structure of maps, but we do not analyze this here.

In the case of periodic boundary conditions, the eigenvectors of **K** have the form

$$e_a^\mu = \cos\left(\frac{2\pi\mu a}{N_v} - \phi\right) \tag{8.33}$$

for $\mu = 0, 1, 2, \ldots, N_v/2$, and with a phase parameter ϕ that can take any value from 0 to 2π. The eigenvalues are given by the discrete Fourier transform $\tilde{K}(\mu)$ of $K(|a - a'|)$, which is real in the case we consider (see the Mathematical Appendix). The principal eigenvector is the eigenfunction from equation 8.33 with μ value corresponding to the maximum of $\tilde{K}(\mu)$. The functions K and $\tilde{K}$ in figure 8.8 are each the difference of two Gaussian functions. $\tilde{K}$ has been plotted as a function of the spatial frequency $k = 2\pi\mu/(N_v d)$, where d is the cortical distance between locations a and $a + 1$. The value of μ corresponding to the principal eigenvector is determined by the k value corresponding to the maximum of the curve in figure 8.8B.

The oscillations in sign of the principal eigenvector, which is indicated by the dotted line in figure 8.8A, generate an alternating pattern of left- and right-eye innervation resembling the ocular dominance stripes seen in primary visual cortex (upper panel figure 8.7B). The lower panel of figure 8.7B shows the result of a simulation of Hebbian development of an ocular dominance map for a one-dimensional line across cortex consisting of 512 units. In this simulation, constraints that limit the growth of synaptic weights have been included, but these do not dramatically alter the conclusions of our analysis.

Competitive Hebb Rule

In this section and the next, we consider models that deal with Hebbian learning and development on a more abstract level. Although inspired by features of neural circuitry, these models are less directly linked to neurons and synapses. For example, the model discussed in this section considers

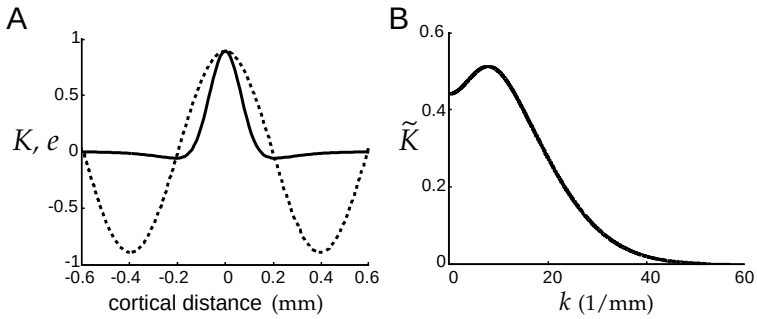

Figure 8.8 Hypothetical K function. (A) The solid line is $K(|a - a'|)$ given by the difference of two Gaussian functions. We have plotted this as a function of the distance between the cortical locations corresponding to the indices a and a', rather than of $|a - a'|$. The dotted line is the principal eigenvector plotted on the same scale. (B) $\tilde{K}$, the Fourier transform of K. This is also given by the difference of two Gaussians. As in A, we use cortical distance units and plot $\tilde{K}$ in terms of the spatial frequency k rather than the integer index μ.

the effects of excitation and inhibition by modeling competitive and cooperative aspects of how activity is generated in, and spread across, the cortex. The advantage of this approach is that it allows us to deal with complex, nonlinear features of Hebbian models in a more controlled and manageable way. As we have seen, competition between neurons is an important element in the development of realistic patterns of selectivity. Linear recurrent connections can produce only a limited amount of differentiation among network neurons, because they induce fairly weak competition between output units. As detailed in chapter 7, recurrent connections can lead to much stronger competition if the interactions are nonlinear. The model discussed in this section allows strongly nonlinear competition to arise in a Hebbian setting.

nonlinear competition

As mentioned in the previous paragraph, the model we discuss represents the effect of cortical processing in two somewhat abstract stages. One stage, modeling the effects of long-range inhibition, involves competition among all the cortical cells for feedforward input in a scheme related to that used in chapter 2 for contrast saturation. The second stage, modeling shorter-range excitation, involves cooperation in which neurons that receive feedforward input excite their neighbors.

In the first stage, the feedforward input for unit a, and that for all the other units, is fed through a nonlinear function to generate a competitive measure of the local excitation generated at location a,

$$z_a = \frac{\left(\sum_b W_{ab} u_b\right)^\delta}{\sum_{a'} \left(\sum_b W_{a'b} u_b\right)^\delta}. \tag{8.34}$$

The activities and weights are all assumed to be positive. The parameter δ controls the degree of competition. For large δ, only the largest feedforward input survives. The case $\delta = 1$ is quite similar to the linear recurrent connections of the previous section.

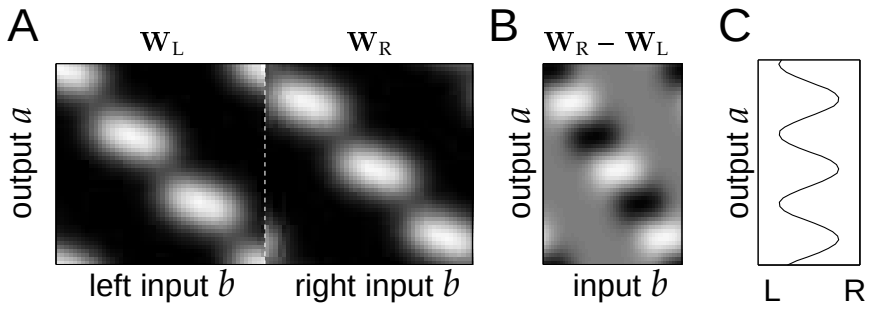

Figure 8.9 Ocular dominance patterns from a competitive Hebb rule. (A) Final stable weights W_{ab} plotted as a function of a and b, showing the relative strengths of the connections from left- and right-eye inputs centered at various retinal locations. White represents a large positive value and black represents 0. (B) The difference in the connections between right- and left-eye inputs. White indicates a positive value and black, a negative value. (C) Difference in the connections summed across all the inputs b to each cortical cell a, showing the net ocularity for each cell. The model used here has 100 input units for each eye and for the output layer, and a coarse initial topography was assumed. Circular (toroidal) boundary conditions were imposed to avoid edge effects. The input activity patterns during training represented single Gaussian illuminations in both eyes centered on a randomly chosen input unit b, with a larger magnitude for one eye (chosen randomly) than for the other. The recurrent weights $\mathbf{M}$ took the form of a Gaussian.

In the cooperative stage, the local excitation of equation 8.34 is distributed across the cortex by recurrent connections, producing a level of activity in unit a given by

$$v_a = \sum_{a'} M_{aa'} z_{a'} \, . \tag{8.35}$$

This ensures that the excitation characterized by z_a is spread across a local neighborhood of the cortex rather than being concentrated entirely at location a. In this scheme, the recurrent connections described by the matrix $\mathbf{M}$ are usually purely excitatory and of fairly short range, because the effect of longer-range inhibition has been modeled by the competition.

competitive Hebbian learning

Using the outputs of equation 8.35 in conjunction with a Hebbian rule for the feedforward weights is called competitive Hebbian learning. The competition between neurons implemented by this scheme does not ensure competition among the synapses onto a given neuron, so some mechanism such as a normalization constraint is still required. For these models, the outcome of training cannot be analyzed simply by considering eigenvectors of the covariance or correlation matrix because the activation process is nonlinear. Rather, higher-order statistics of the input distribution are important. Nonlinear competition can lead to strong differentiation of output units.

An example of the use of competitive Hebbian learning is shown in figure 8.9, in the form of a one-dimensional cortical map of ocular dominance with inputs arising from LGN neurons with receptive fields covering an extended region of the visual field (rather than the single location

of our simpler model). This example uses competitive Hebbian plasticity with nondynamic multiplicative weight normalization. Two weight matrices, $\mathbf{W}_R$ and $\mathbf{W}_L$, corresponding to right- and left-eye inputs, characterize the connectivity of the model. These are shown separately in figure 8.9A, which illustrates that the cortical cells develop retinotopically ordered receptive fields, and they segregate into alternating patches dominated by one eye or the other. The index a indicates the identity and cortical location of the ouput unit, whereas b indicates the retinal location of the center of the corresponding input unit. The ocular dominance pattern is easier to see in figure 8.9B, which shows the difference between the right- and left-eye weights, $\mathbf{W}_R - \mathbf{W}_L$, and figure 8.9C, which shows the net ocularity of the total input to each output neuron of the model ($\sum_b [W_R - W_L]_{ab}$ for each a). It is possible to analyze the structure shown in figure 8.9 and reveal the precise effect of the competition (i.e., the effect of changing the competition parameter δ in equation 8.34). Such an analysis shows, for example, that subtractive normalization of the synaptic weight is not necessary to ensure the robust development of ocular dominance, as it is in the noncompetitive case.

Feature-Based Models

Models of cortical map formation can get extremely complex when multiple neuronal selectivities, such as retinotopic location, ocular dominance, and orientation preference, are considered simultaneously. To deal with this, a class of more abstract models, called competitive feature-based models, has been developed. These use a general approach similar to the competitive Hebbian models discussed in the previous section. Feature-based models are not directly related to the biophysical reality of neuronal firing rates and synaptic strengths, but they provide a compact description of map development.

In a feedforward model, the selectivity of neuron a is determined by the feedforward weights W_{ab} for all values of b, describing how this neuron is connected to the input units of the network. The input units are driven by the stimulus and their responses reflect various stimulus features. Thus, selectivity in these models is determined by how the synaptic weights transfer the selectivities of the input units to the output units. The idea of a feature-based model is to simplify this by directly relating the output unit selectivities to the corresponding features of the stimulus. In feature-based models, the index b is not used to label different input units, but rather to label different features of the stimulus. N_u is thus equal to the number of parameters being used to characterize the stimulus. Also, the input variable u_b is set to the parameter used to characterize feature b of the stimulus. Similarly, W_{ab} does not describe the coupling between input unit b and output unit a, but instead it represents the selectivity of output unit a to stimulus feature b. For example, suppose $b = 1$ represents the location of a visual stimulus, and $b = 2$ represents its ocularity, the difference in strength between the left- and right-eye inputs. Then, u_1 and u_2 would be the location coordinate and the ocularity of the stimulus, and W_{a1} and W_{a2} would be the preferred stimulus location (the center of the

receptive field) and the preferred eye for neuron a. Map development in such a model is studied by noting how the appearance of various stimulus features and neuronal selectivities affects the matrix $\mathbf{W}$. By associating the index a with cortical location, the structure of the final matrix $\mathbf{W}$ that arises from a plasticity rule predicts how selectivities are mapped across the cortex.

The activity of a particular output unit in a feature-based model is determined by how closely the stimulus being presented matches its preferred stimulus. The weights W_{ab} for all b values determine the preferred stimulus features for neuron a, and we assume that the activation of neuron a is high if the components of the input u_b match the components of W_{ab}. A convenient way to achieve this is to express the activation for unit a as $\exp(-\sum_b (u_b - W_{ab})^2/(2\sigma_b^2))$, which has its maximum at $u_b = W_{ab}$ for all b, and falls off as a Gaussian function for less perfect matches of the stimulus to the selectivity of the cell. The parameter σ_b determines how selective the neuron is for characteristic b of the stimulus.

The Gaussian expression for the activation of neuron a is not used directly to determine its level of activity. Rather, as in the case of competitive Hebbian learning, we introduce a competitive activity variable for cortical site a,

$$z_a = \frac{\exp\left(-\sum_b (u_b - W_{ab})^2/(2\sigma_b^2)\right)}{\sum_{a'} \exp\left(-\sum_b (u_b - W_{a'b})^2/(2\sigma_b^2)\right)}. \tag{8.36}$$

In addition, some cooperative mechanism must be included to keep the maps smooth, which means that nearby neurons should, as far as possible, have similar selectivities. The two algorithms we discuss, the self-organizing map and the elastic net, differ in how they introduce this second element.

self-organizing map

The self-organizing map spreads the activity defined by equation 8.36 to nearby cortical sites through equation 8.35, $v_a = \sum_{a'} M_{aa'} z_{a'}$. This gives cortical cells a and a' similar selectivities if they are nearby, because v_a and $v_{a'}$ affect one another through local recurrent excitation. The elastic net sets the activity of unit a to the result of equation 8.36, $v_a = z_a$, which generates competition. Smoothness of the map is ensured not by spreading this activity, as in the self-organizing map, but by including an additional term in the plasticity rule that tends to make nearby selectivities the same (see below).

elastic net

Hebbian development of the selectivities characterized by $\mathbf{W}$ is generated by an activity-dependent rule. In general, Hebbian plasticity adjusts the weights of activated units so that they become more responsive to, and selective for, input patterns that excite them. Feature-based models achieve the same thing by modifying the selectivities W_{ab} so they more closely match the input parameters u_b when output unit a is activated by $\mathbf{u}$. For the case of the self-organized map, this is achieved through the averaged

feature-based learning rule

rule

$$\tau_w \frac{dW_{ab}}{dt} = \langle v_a (u_b - W_{ab}) \rangle \,. \tag{8.37}$$

The elastic net modification rule is similar, but an additional term is included to make the maps smooth, because smoothing is not included in the rule that generates the activity in this case. The elastic net plasticity rule is

elastic net rule

$$\tau_w \frac{dW_{ab}}{dt} = \langle v_a (u_b - W_{ab}) \rangle + \beta \sum_{a' \text{ neighbor of } a} (W_{a'b} - W_{ab}) \,, \tag{8.38}$$

where the sum is over all points a' that are neighbors of a, and β is a parameter that controls the degree of smoothness in the map. The elastic net makes W_{ab} similar to $W_{a'b}$, if a and a' are nearby on the cortex, by reducing $(W_{a'b} - W_{ab})^2$. Both the feature-based and elastic net rules make $W_{ab} \to u_b$ when v_a is positive.

Figure 8.10A shows the results of an optical imaging experiment that reveals how ocularity and orientation selectivity are arranged across a region of the primary visual cortex of a macaque monkey. The dark lines show the boundaries of the ocular dominance stripes. The lighter lines show iso-orientation contours, which are locations where the preferred orientations are roughly the same. They indicate, by the regions they enclose, neighborhoods (called domains) of cells that favor similar orientations. They also show how these neighborhoods are arranged with respect to each other and to the ocular dominance stripes. There are singularities, called pinwheels, in the orientation map, where regions with different orientation preferences meet at a point. These tend to occur near the centers of the ocular dominance stripes. There are also linear zones where the iso-orientation domains are parallel. These tend to occur at, and run perpendicular to, the boundaries of the ocular dominance stripes.

Figure 8.10B shows the result of an elastic net model plotted in the same form as the macaque map of figure 8.10A. The similarity is evident and striking. Here $\mathbf{u} = (x, y, o, e\cos\theta, e\sin\theta)$ includes 5 stimulus features ($N_u = 5$): two (x, y) for retinal location, one (o) for ocularity, and two (θ, e) for the direction and strength of orientation. The self-organizing map can produce almost identical results, and noncompetitive and competitive Hebbian developmental algorithms can also lead to similar structures.

Anti-Hebbian Modification

We previously alluded to the problem of redundancy among multiple output neurons that can arise from feedforward Hebbian modification. The Oja rule of equation 8.16 for multiple output units, which takes the form

$$\tau_w \frac{dW_{ab}}{dt} = v_a u_b - \alpha v_a^2 W_{ab} \,, \tag{8.39}$$

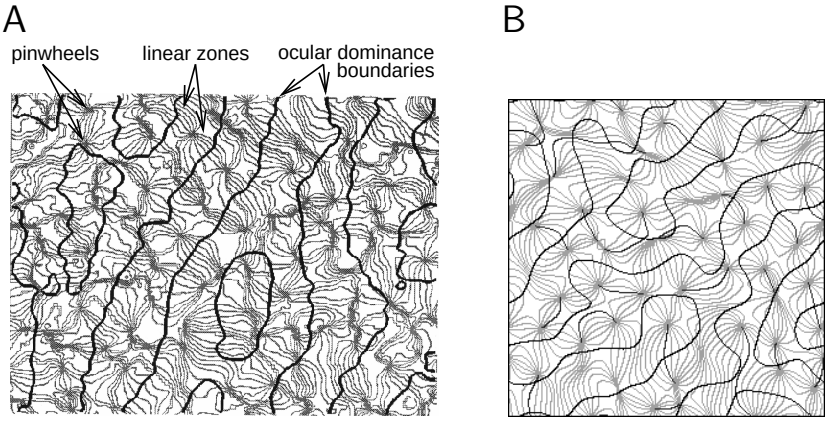

Figure 8.10 Orientation domains and ocular dominance. (A) Contour map showing iso-orientation contours (gray lines) and the boundaries of ocular dominance stripes (black lines) in a 1.7 × 1.7 mm patch of macaque primary visual cortex. Iso-orientation contours are drawn at intervals of 11.25°. Pinwheels are singularities in the orientation map where all the orientations meet, and linear zones are extended patches over which the iso-orientation contours are parallel. (B) Ocular dominance and orientation map produced by the elastic net model. The significance of the lines is the same as in (A), except that the darker gray lines show orientation preferences of 0°. (A adapted from Obermayer & Blasdel, 1993; B from Erwin et al., 1995.)

provides a good illustration of this problem. In the absence of recurrent connections, this rule sets each row of the feedforward weight matrix to the principal eigenvector of the input correlation matrix, making each output unit respond identically.

One way to reduce redundancy in a linear model is to make the linear recurrent interactions of equation 8.29 plastic rather than fixed, using an anti-Hebbian modification rule. As the name implies, anti-Hebbian plasticity causes synapses to decrease (rather than increase) in strength when there is simultaneous pre- and postsynaptic activity. The recurrent interactions arising from an anti-Hebb rule can prevent different output units from representing the same eigenvector. This occurs because the recurrent interactions tend to make output units less correlated by canceling the effects of common feedforward input. Anti-Hebbian modification is believed to be the predominant form of plasticity at synapses from parallel fibers to Purkinje cells in the cerebellum, although this may be a special case because Purkinje cells inhibit rather than excite their targets. A basic anti-Hebb rule for $M_{aa'}$ can be created simply by changing the sign on the right side of equation 8.3. However, just as Hebbian plasticity tends to make weights increase without bound, anti-Hebbian modification tends to make them decrease to 0, and to avoid this it is necessary to use

$$\tau_M \frac{d\mathbf{M}}{dt} = -\mathbf{v}\mathbf{v} + \beta\mathbf{M} \quad \text{or} \quad \tau_M \frac{dM_{aa'}}{dt} = -v_a v_{a'} + \beta M_{aa'} \tag{8.40}$$

to modify the off-diagonal components of $\mathbf{M}$ (the diagonal components are defined to be 0). Here, β is a positive constant. For suitably chosen β and τ_M, the combination of rules 8.39 and 8.40 produces a stable configuration

in which the rows of the weight matrix $\mathbf{W}$ are different eigenvectors of the correlation matrix $\mathbf{Q}$, and all the elements of the recurrent weight matrix $\mathbf{M}$ are ultimately set to 0.

Goodall (1960) proposed an alternative scheme for decorrelating different output units. In his model, the feedforward weights $\mathbf{W}$ are kept constant, whereas the recurrent weights adapt according to the anti-Hebb rule *Goodall rule*

$$\tau_M \frac{d\mathbf{M}}{dt} = -(\mathbf{W} \cdot \mathbf{u})\mathbf{v} + \mathbf{I} - \mathbf{M}. \tag{8.41}$$

The minus sign in the term $-(\mathbf{W} \cdot \mathbf{u})\mathbf{v}$ embodies the anti-Hebbian modification. This term is nonlocal because the change in the weight of a given synapse depends on the total feedforward input to the postsynaptic neuron, not merely on the input at that particular synapse (recall that $\mathbf{v} \neq \mathbf{W} \cdot \mathbf{u}$ in this case because of the recurrent connections). The term $\mathbf{I} - \mathbf{M}$ prevents the weights from going to 0 by pushing them toward the identity matrix $\mathbf{I}$. Unlike 8.40, this rule requires the existence of autapses, synapses that a neuron makes onto itself (i.e., the diagonal elements of $\mathbf{M}$ are not 0).

If the Goodall plasticity rule converges and stops changing $\mathbf{M}$, the right side of equation 8.41 must vanish on average, which requires (using the definition of $\mathbf{K}$)

$$\langle (\mathbf{W} \cdot \mathbf{u})\mathbf{v} \rangle = \mathbf{I} - \mathbf{M} = \mathbf{K}^{-1}. \tag{8.42}$$

Multiplying both sides by $\mathbf{K}$ and using equation 8.30, we find

$$\langle \mathbf{v}\mathbf{v} \rangle = \langle (\mathbf{K} \cdot \mathbf{W} \cdot \mathbf{u})\mathbf{v} \rangle = \mathbf{I}. \tag{8.43}$$

This means that the outputs are decorrelated and also indicates histogram equalization in the sense, discussed in chapter 4, that all the elements of $\mathbf{v}$ have the same variance. Indeed, the Goodall algorithm can be used to implement the decorrelation and whitening discussed in chapter 4. Because the anti-Hebb and Goodall rules are based on linear models, they are capable of removing only second-order redundancy, meaning redundancy characterized by the covariance matrix. In chapter 10, we consider models that are based on eliminating higher orders of redundancy as well.

Timing-Based Plasticity and Prediction

Temporal Hebbian rules have been used in the context of multi-unit networks to store information about temporal sequences. To illustrate this, we consider a network with the architecture of figure 8.6. We study the effect of time-dependent synaptic plasticity, as given by equation 8.18, on the recurrent synapses of the model, leaving the feedforward synapses constant.

Suppose that before training the average response of output unit a to a stimulus characterized by a parameter s is given by the tuning curve $f_a(s)$,

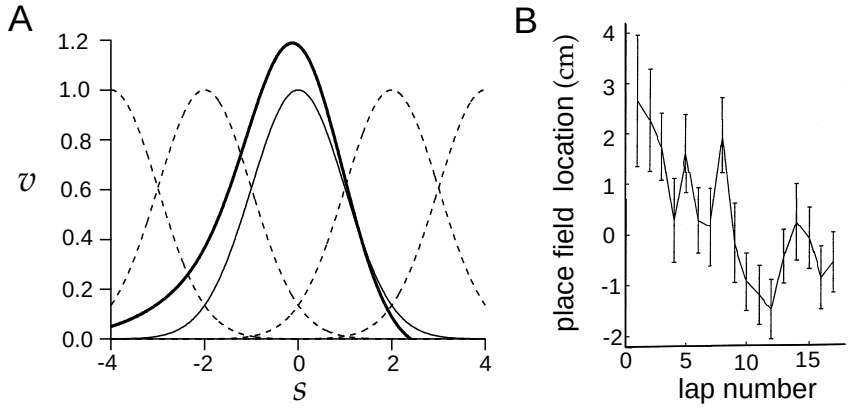

Figure 8.11 Predicted and experimental shifts of place fields. (A) Shift in a neuronal firing-rate tuning curve caused by repeated exposure to a time-dependent stimulus during training. The dashed curves and thin solid curve indicate the initial response tuning curves of a network of interconnected neurons. The thick solid curve is the response tuning curve of the neuron that initially had the thin solid tuning curve after a training period involving a time-dependent stimulus. The tuning curve increases in amplitude, asymmetrically broadens, and shifts as a result of temporally asymmetric Hebbian plasticity. The shift shown corresponds to a training stimulus with a positive rate of change, that is, one that moves rightward on this plot as a function of time. The corresponding shift in the tuning curve is to the left. The shift has been calculated using more neurons and tuning curves than are shown in this plot. (B) Location of place field centers while a rat traversed laps around a closed track (0 is defined as the average center location across the whole experiment). Over sequential laps, the place fields shifted backward relative to the direction the rat moved. (A adapted and modified from Abbott & Blum, 1996; B adapted from Mehta et al., 1997.)

which reaches a maximum value for the optimal stimulus $s = s_a$. Different neurons have different optimal stimulus values, as depicted by the dashed and thin solid curves in figure 8.11A. We now examine what happens when the plasticity rule 8.18 is applied throughout a training period during which the stimulus being presented is an increasing function of time, i.e., moves to the right in figure 8.11A. Such a stimulus excites the different neurons in the network sequentially. For example, the neuron with $s_a = -2$ is active before the neuron with $s_a = 0$, which in turn is active before the neuron with $s_a = 2$. If the stimulus changes rapidly enough, the interval between the firing of the neuron with $s_a = -2$ and that with $s_a = 0$ will fall within the window for LTP depicted in figure 8.2B. This means that a synapse from the neuron with $s_a = -2$ to the $s_a = 0$ neuron will be strengthened. On the other hand, because the neuron with $s_a = 2$ fires after the $s_a = 0$ neuron, falling within the window for LTD, a synapse from it to the $s_a = 0$ neuron will be weakened.

The effect of this type of modification on the tuning curve in the middle of the array (the thin solid curve in figure 8.11A centered at $s = 0$) is shown by the thick solid curve in figure 8.11A. After the training period, the neuron with $s_a = 0$ receives strengthened input from neurons with $s_a < 0$ and

weakened input from neurons with $s_a > 0$. This asymmetrically broadens and shifts the tuning curve of the neuron with $s_a = 0$ to lower stimulus values. The leftward shift seen in figure 8.11A is a result of the temporal character of the plasticity rule and the temporal evolution of the stimulus during training. Note that the shift is in the direction opposite to the motion of the stimulus during training. This backward shift has an interesting interpretation. If the same time-dependent stimulus is presented again after training, the neuron with $s_a = 0$ will respond earlier than it did prior to training. Thus, the training experience causes neurons to develop responses that predict the behavior of the stimulus. Although we chose to discuss the neuron with $s_a = 0$ as a representative example, the responses of other neurons shift in a similar manner.

Asymmetric enlargements and backward shifts of neural response tuning curves similar to those predicted from temporally asymmetric LTP and LTD induction have been seen in recordings of hippocampal place cells in rats (see chapter 1) made by Mehta et al. (1997, 2000). Figure 8.11B shows the average location of place fields (the place-cell analog of receptive fields) recorded while a rat ran repeated laps around a closed track. Over time, the place field shifted backward along the track relative to the direction the rat moved.

8.4 Supervised Learning

In unsupervised learning, inputs are imposed during a training period and the output is determined by the network dynamics, using the current values of the weights. This means that the network and plasticity rule must uncover patterns and regularities in the input data (such as the direction of maximal variance) by themselves. In supervised learning, both a set of inputs and the corresponding desired outputs are imposed during training, so the network is essentially given the answer.

Two basic problems addressed in supervised learning are storage, which means learning the relationship between the input and output patterns provided during training, and generalization, which means being able to provide appropriate outputs for inputs that were not presented during training but are similar to those that were. The main tasks we consider within the context of supervised learning are classification of inputs into two categories and function approximation (or regression), in which the output of a network unit is trained to approximate a specified function of the input. Understanding generalization in such settings has been a major focus of theoretical investigations in statistics and computer science but lies outside the scope of our discussion.

Supervised Hebbian Learning

In supervised learning, paired input and output samples are presented during training. We label these pairs by $\mathbf{u}^m$ and v^m for $m = 1 \ldots N_S$, where

the superscript is a label and does not signify either a component of $\mathbf{u}$ or a power of v. For a feedforward network, an averaged Hebbian plasticity rule for supervised learning can be obtained from equation 8.4 by averaging across all the input-output pairs,

$$\tau_w \frac{d\mathbf{w}}{dt} = \langle v\mathbf{u} \rangle = \frac{1}{N_S} \sum_{m=1}^{N_S} v^m \mathbf{u}^m \,. \tag{8.44}$$

cross-correlation

As in the unsupervised Hebbian learning case, the synaptic modification process depends on the input-output cross-correlation $\langle v\mathbf{u} \rangle$. However, for supervised learning, the output $v = v^m$ is imposed on the network rather than being determined by it.

Unless the cross-correlation is 0, equation 8.44 never stops changing the synaptic weights. The methods introduced to stabilize Hebbian modification in the case of unsupervised learning can be applied to supervised learning as well. However, stabilization is easier in the supervised case, because the right side of equation 8.44 does not depend on $\mathbf{w}$. Therefore, the growth is only linear rather than exponential in time, making a simple multiplicative synaptic weight decay term sufficient for stability. This is introduced by writing the supervised learning rule as

supervised learning with decay

$$\tau_w \frac{d\mathbf{w}}{dt} = \langle v\mathbf{u} \rangle - \alpha \mathbf{w} \,, \tag{8.45}$$

for some positive constant α. Asymptotically, equation 8.45 makes $\mathbf{w} = \langle v\mathbf{u} \rangle / \alpha$, that is, the weights become proportional to the input-output cross-correlation.

We discuss supervised Hebbian learning in the case of a single output unit, but the results can be generalized to multiple outputs.

Classification and the Perceptron

perceptron
binary classifier

The perceptron is a nonlinear map that classifies inputs into one of two categories. It thus acts as a binary classifier. To make the model consistent when units are connected in a network, we also require the inputs to be binary. We can think of the two possible states as representing units that are either active or inactive. As such, we would naturally assign them the values 1 and 0. However, the analysis is simpler (while producing similar results) if, instead, we require the inputs u_a and output v to take the two values $+1$ and -1.

The output of the perceptron is based on a modification of the linear rule of equation 8.2 to

$$v = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{u} - \gamma \geq 0 \\ -1 & \text{if } \mathbf{w} \cdot \mathbf{u} - \gamma < 0 \,. \end{cases} \tag{8.46}$$

The threshold γ determines the dividing line between values of $\mathbf{w} \cdot \mathbf{u}$ that generate $+1$ and -1 outputs. The supervised learning task for the perceptron is to place each of N_S input patterns $\mathbf{u}^m$ into one of two classes designated by the desired binary output v^m. How well the perceptron performs this task depends on the nature of the classification. The weight vector and threshold define a subspace (a hyperplane) of dimension $N_u - 1$ (the subspace of points satisfying $\mathbf{w} \cdot \mathbf{u} = \gamma$, which is perpendicular to $\mathbf{w}$) that cuts the N_u-dimensional space of input vectors into two regions. It is possible for a perceptron to classify inputs perfectly only if a hyperplane exists that divides the input space into one half-space containing all the inputs corresponding to $v = +1$, and another half-space containing all those for $v = -1$. This condition is called linear separability. An instructive case to consider is when each component of each input vector and the associated output values are chosen randomly and independently, with equal probabilities of being $+1$ and -1. For large N_u, the maximum number of random associations that can be described by a perceptron for typical examples of this type is $2N_u$.

linear separability

For linearly separable inputs, a set of weights exists that allows the perceptron to perform perfectly. However, this does not mean that a Hebbian modification rule can construct such weights. A Hebbian rule based on equation 8.45 with $\alpha = N_u/N_S$ constructs the weight vector

$$\mathbf{w} = \frac{1}{N_u} \sum_{m=1}^{N_S} v^m \mathbf{u}^m \,. \tag{8.47}$$

To see how well such weights allow the perceptron to perform, we compute the output generated by one input vector, $\mathbf{u}^n$, chosen from the training set. For this example, we set $\gamma = 0$. Nonzero threshold values are considered later in the chapter.

With $\gamma = 0$, the value of v for input $\mathbf{u}^n$ is determined solely by the sign of $\mathbf{w} \cdot \mathbf{u}^n$. Using the weights of equation 8.47, we find

$$\mathbf{w} \cdot \mathbf{u}^n = \frac{1}{N_u} \left(v^n \mathbf{u}^n \cdot \mathbf{u}^n + \sum_{m \neq n} v^m \mathbf{u}^m \cdot \mathbf{u}^n \right) . \tag{8.48}$$

If we set $\sum_{m \neq n} v^m \mathbf{u}^m \cdot \mathbf{u}^n / N_u = \eta^n$ (where the superscript on η, as on v, is a label, not a power), then $v^n \mathbf{u}^n \cdot \mathbf{u}^n / N_u = v^n$ because $1^2 = (-1)^2 = 1$, so we can write

$$\mathbf{w} \cdot \mathbf{u}^n = v^n + \eta^n \,. \tag{8.49}$$

Substituting this expression into equation 8.46 to determine the output of the perceptron for the input $\mathbf{u}^n$, we see that the term η^n acts as a source of noise, interfering with the ability of the perceptron to generate the correct answer $v = v^n$.

We can think of η^n as a sample from a probability distribution over values of η. Consider the case for which all components of $\mathbf{u}^m$ and v^m for all m

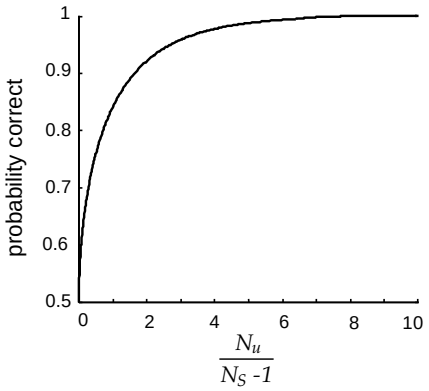

Figure 8.12 Percentage of correct responses for a perceptron with a Hebbian weight vector for a random binary input-output map. As the ratio of the number of inputs, N_u, to one less than the number of input vectors being learned, $N_S - 1$, grows, the percentage of correct responses goes to 1. When this ratio is small, the percentage of correct responses approaches the chance level of $1/2$.

are chosen randomly with equal probabilities of being $+1$ or -1. Including the dot product, the right side of the expression $N_u \eta^n = \sum_{m \neq n} v^m \mathbf{u}^m \cdot \mathbf{u}^n$ defining η^n is the sum of $(N_S - 1)N_u$ terms, each of which is equally likely to be $+1$ or -1. For large N_u and N_S, the central limit theorem (see the Mathematical Appendix) implies that the distribution of η values is Gaussian with mean 0 and variance $(N_S - 1)/N_u$. This suggests that the perceptron with Hebbian weights should work well if the number of input patterns being learned is significantly less than the number of input vector components. We can make this more precise by noting from equations 8.46 with $\gamma = 0$ and equation 8.49 that, for $v^n = +1$, the perceptron will give the correct answer if $-1 < \eta^n < \infty$. Similarly, for $v^n = -1$, the perceptron will give the correct answer if $-\infty < \eta^n < 1$. If v^n has probability $1/2$ of taking either value, the probability of the perceptron giving the correct answer is $1/2$ times the integral of the Gaussian distribution from -1 to ∞ plus $1/2$ times its integral from $-\infty$ to 1. Combining these two terms, we find

$$P[\text{correct}] = \sqrt{\frac{N_u}{2\pi(N_S - 1)}} \int_{-\infty}^{1} d\eta \, \exp\left(-\frac{N_u \eta^2}{2(N_S - 1)}\right). \tag{8.50}$$

This result is plotted in figure 8.12, which shows that the Hebbian perceptron performs quite well if $N_S - 1$ is less than about $0.2N_u$. It is possible for the perceptron to perform considerably better than this if a non-Hebbian weight vector is used. We return to this in a later section.

Function Approximation

In chapter 1, we studied examples in which the firing rate of a neuron was given by a function of a stimulus parameter, namely, the response tuning curve. When such a relationship exists, we can think of the neuronal

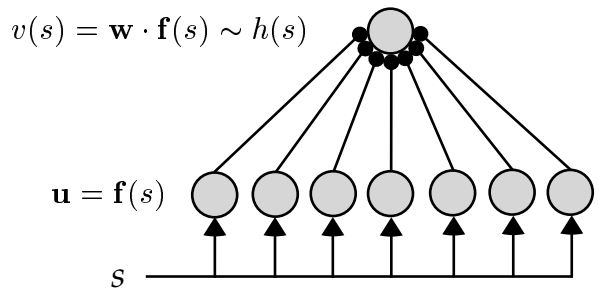

Figure 8.13 A network for representing functions. The value of an input variable s is encoded by the activity of a population of neurons with tuning curves $\mathbf{f}(s)$. This activity drives an output neuron through a vector of weights $\mathbf{w}$ to create an output activity v that approximates the function $h(s)$.

firing rate as representing the function. Populations of neurons (labeled by an index $b = 1, 2, \ldots, N_u$) that respond to a stimulus value s, by firing at average rates $f_b(s)$, can similarly represent an entire set of functions. However, a function $h(s)$ that is not equal to any of the single neuron tuning curves must be represented by combining the responses of a number of units. This can be done using the network shown in figure 8.13. The average steady-state activity level of the output unit in this network, in response to stimulus value s, is given by equation 8.2,

function approximation

$$v(s) = \mathbf{w} \cdot \mathbf{u} = \mathbf{w} \cdot \mathbf{f}(s) = \sum_{b=1}^{N} w_b f_b(s) \,. \tag{8.51}$$

Note that we have replaced $\mathbf{u}$ with $\mathbf{f}(s)$ where $\mathbf{f}(s)$ is the vector with components $f_b(s)$. The network presented in chapter 7 that performs coordinate transformation is an example of this type of function approximator.

In equation 8.51, the input tuning curves $\mathbf{f}(s)$ act as a basis for representing the output function $h(s)$, and for this reason they are called basis functions. Different sets of basis functions can be used to represent a given set of output functions. A set of basis functions that can represent any member of a class of functions using a linear sum, as in equation 8.51, is called complete for this class. For the basis sets typically used in mathematics, such as the sines and cosines in a Fourier series, the weights in equation 8.51 are unique. When neural tuning curves are used to expand a function, the weights tend not to be unique, and the set of input functions is called overcomplete. In this chapter, we assume that the basis functions are held fixed, and only the weights are adjusted to improve output performance. It is also interesting to consider methods for learning the best basis functions for a particular application. One way of doing this is by applying an algorithm called backpropagation, which develops the basis functions guided by the output errors of the network. Other methods, which we consider in chapter 10, involve unsupervised learning.

basis functions

completeness

overcomplete

Suppose that the function-representation network of figure 8.13 is pro-

vided with a sequence of N_S sample stimuli, s^m for $m = 1, 2, \ldots, N_S$, and the corresponding function values $h(s^m)$, during a training period. To make $v(s^m)$ match $h(s^m)$ as closely as possible for all m, we minimize the error

$$E = \frac{1}{2}\sum_{m=1}^{N_S}\left(h(s^m) - v(s^m)\right)^2 = \frac{N_S}{2}\left\langle (h(s) - \mathbf{w}\cdot\mathbf{f}(s))^2\right\rangle . \tag{8.52}$$

We have made the replacement $v(s) = \mathbf{w}\cdot\mathbf{f}(s)$ in this equation and have used the bracket notation for the average over the training samples. Equations for the weights that minimize this error, called the normal equations, are obtained by setting its derivative with respect to the weights to 0, yielding the condition

normal equations

$$\langle \mathbf{f}(s)\mathbf{f}(s)\rangle \cdot \mathbf{w} = \langle \mathbf{f}(s)h(s)\rangle . \tag{8.53}$$

The supervised Hebbian rule of equation 8.45, applied in this case, ultimately sets the weight vector to $\mathbf{w} = \langle \mathbf{f}(s)h(s)\rangle/\alpha$. These weights must satisfy the normal equations 8.53 if they are to optimize function approximation. There are two circumstances under which this occurs. The obvious one is when the input units are orthogonal across the training stimuli, $\langle \mathbf{f}(s)\mathbf{f}(s)\rangle = \mathbf{I}$. In this case, the normal equations are satisfied with $\alpha = 1$. However, this condition is unlikely to hold for most sets of input tuning curves. An alternative possibility is that for all pairs of stimuli s^m and s^n in the training set,

$$\mathbf{f}(s^m)\cdot\mathbf{f}(s^n) = c\delta_{mn} \tag{8.54}$$

for some constant c. This is called a tight frame condition. If it is satisfied, the weights given by supervised Hebbian learning with decay can satisfy the normal equations. To see this, we insert the weights $\mathbf{w} = \langle \mathbf{f}(s)h(s)\rangle/\alpha$ into equation 8.53 and use 8.54 to obtain

tight frame

$$\begin{aligned}\langle \mathbf{f}(s)\mathbf{f}(s)\rangle \cdot \mathbf{w} &= \frac{\langle \mathbf{f}(s)\mathbf{f}(s)\rangle \cdot \langle \mathbf{f}(s)h(s)\rangle}{\alpha} = \frac{1}{\alpha N_S^2}\sum_{mn}\mathbf{f}(s^m)\mathbf{f}(s^m)\cdot\mathbf{f}(s^n)h(s^n)\\ &= \frac{c}{\alpha N_S^2}\sum_m \mathbf{f}(s^m)h(s^m) = \frac{c}{\alpha N_S}\langle \mathbf{f}(s)h(s)\rangle .\end{aligned} \tag{8.55}$$

This shows that the normal equations are satisfied for $\alpha = c/N_S$. Thus, we have shown two ways that supervised Hebbian learning can solve the function approximation problem, but both require special conditions on the basis functions $\mathbf{f}(s)$. A more general scheme, discussed below, involves using an error-correcting rule.

Supervised Error-Correcting Rules

An essential limitation of supervised Hebbian rules is that synaptic modification does not depend on the actual performance of the network. An

alternative learning strategy is to start with an initial guess for the weights, compare the output $v(\mathbf{u}^m)$ in response to input $\mathbf{u}^m$ with the desired output v^m, and change the weights to improve the performance. Two important error-correcting modification rules are the perceptron rule, which applies to binary classification, and the delta rule, which can be applied to function approximation and many other problems.

The Perceptron Learning Rule

Suppose that the perceptron of equation 8.46 (with nonzero γ) incorrectly classifies an input pattern $\mathbf{u}^m$. If the output is $v(\mathbf{u}^m) = -1$ when $v^m = 1$, the weight vector should be modified to make $\mathbf{w} \cdot \mathbf{u}^m - \gamma$ larger. Similarly, if $v(\mathbf{u}^m) = 1$ when $v^m = -1$, $\mathbf{w} \cdot \mathbf{u}^m - \gamma$ should be decreased. A plasticity rule that performs such an adjustment is the perceptron learning rule,

perceptron learning rule

$$\mathbf{w} \to \mathbf{w} + \frac{\epsilon_w}{2}\left(v^m - v(\mathbf{u}^m)\right)\mathbf{u}^m \quad \text{and} \quad \gamma \to \gamma - \frac{\epsilon_w}{2}(v^m - v(\mathbf{u}^m)) \,. \tag{8.56}$$

Here, and in subsequent sections in this chapter, we use discrete updates for the weights (indicated by the $\to$) rather than the differential equations used up to this point. This is due to the discrete nature of the presentation of the training patterns. In equation 8.56, we have assumed that the threshold γ is also plastic. The learning rule for γ is inverted compared with that for the weights, because γ enters equation 8.46 with a minus sign.

To verify that the perceptron learning rule makes appropriate weight adjustments, we note that it implies that

$$\left(\mathbf{w} \cdot \mathbf{u}^m - \gamma\right) \to \left(\mathbf{w} \cdot \mathbf{u}^m - \gamma\right) + \frac{\epsilon_w}{2}(v^m - v(\mathbf{u}^m))\left(|\mathbf{u}^m|^2 + 1\right) \,. \tag{8.57}$$

This result shows that if $v^m = 1$ and $v(\mathbf{u}^m) = -1$, the weight change increases $\mathbf{w} \cdot \mathbf{u}^m - \gamma$. If $v^m = -1$ and $v(\mathbf{u}^m) = 1$, $\mathbf{w} \cdot \mathbf{u}^m - \gamma$ is decreased. This is exactly what is needed to compensate for the error. Note that the perceptron learning rule does not modify the weights if the output is correct.

To learn a set of input pattern classifications, the perceptron learning rule is applied to each one repeatedly, either sequentially or in a random order. For fixed ϵ_w, the perceptron learning rule of equation 8.56 is guaranteed to find a set of weights $\mathbf{w}$ and threshold γ that solve any linearly separable problem. This is proved in the appendix.

The Delta Rule

The function approximation task with the error function E of equation 8.52 can be solved using an error-correcting scheme similar in spirit to the perceptron learning rule, but designed for continuous rather than binary outputs. A simple but extremely useful version of this is the gradient descent procedure, which modifies $\mathbf{w}$ according to

gradient descent

$$\mathbf{w} \to \mathbf{w} - \epsilon_w \nabla_{\mathbf{w}} E \quad \text{or} \quad w_b \to w_b - \epsilon_w \frac{\partial E}{\partial w_b}, \tag{8.58}$$

where $\nabla_{\mathbf{w}} E$ is the vector with components $\partial E / \partial w_b$. This rule is sensible because $-\nabla_{\mathbf{w}} E$ points in the direction (in the space of synaptic weights) along which E decreases most rapidly. This process tends to reduce E because, for small ϵ_w,

$$E(\mathbf{w} - \epsilon_w \nabla_{\mathbf{w}} E) \approx E(\mathbf{w}) - \epsilon_w |\nabla_{\mathbf{w}} E|^2 \leq E(\mathbf{w}). \tag{8.59}$$

If ϵ_w is too large or $\mathbf{w}$ is very near to a point where $\nabla_{\mathbf{w}} E(\mathbf{w}) = \mathbf{0}$, E can increase instead. We assume that ϵ_w is small enough so that E decreases at least until $\mathbf{w}$ is very close to a minimum. If E has many minima, gradient descent will lead to only one of them (a local minimum), and not necessarily the one with the lowest value of E (the global minimum). In the case of function approximation using basis functions as in equation 8.51, gradient descent finds a value of $\mathbf{w}$ that satisfies the normal equations, and therefore constructs an optimal function approximator, because the error function of equation 8.52 has only one minimum.

For function approximation, the error E in equation 8.52 is a sum over the set of examples. As a result, $\nabla_{\mathbf{w}} E$ also involves a sum,

$$\nabla_{\mathbf{w}} E = -\sum_{m=1}^{N_S} (h(s^m) - v(s^m)) \mathbf{f}(s^m), \tag{8.60}$$

where we have used the fact that $\nabla_{\mathbf{w}} v(s^m) = \mathbf{f}(s^m)$. The presence of the sum means that the learning rule of equation 8.58 cannot be applied until all the sample patterns have been presented, because all of them are needed to compute the amount by which the weight vector should be changed. It is much more convenient if updating of the weights takes place continuously while sample inputs are presented. This can be done using a procedure known as stochastic gradient descent. This alternative procedure involves presenting randomly chosen input-output pairs s^m and $h(s^m)$, and change $\mathbf{w}$ according to

stochastic gradient decent

$$\mathbf{w} \to \mathbf{w} + \epsilon_w (h(s^m) - v(s^m)) \mathbf{f}(s^m). \tag{8.61}$$

delta rule

This rule, called the delta rule, allows learning to take place one sample at a time. Use of this rule is based on the fact that summing the changes proportional to $(h(s^m) - v(s^m))\mathbf{f}(s^m)$ over the random choices of m is, on average, equivalent to doing the sum in equation 8.60. The effect of using equation 8.61 instead of equation 8.58 is the introduction of noise that causes the weights to fluctuate about average values that satisfy the normal equations. Replacing a full sum by an appropriately weighted sum of randomly chosen terms is an example of a so-called Monte Carlo method. There are more efficient methods of searching for minima of functions than stochastic gradient descent, but many of them are complicated to implement.

Figure 8.14 shows the result of modifying an initially random set of weights using the delta rule. Ultimately, an array of input neurons with

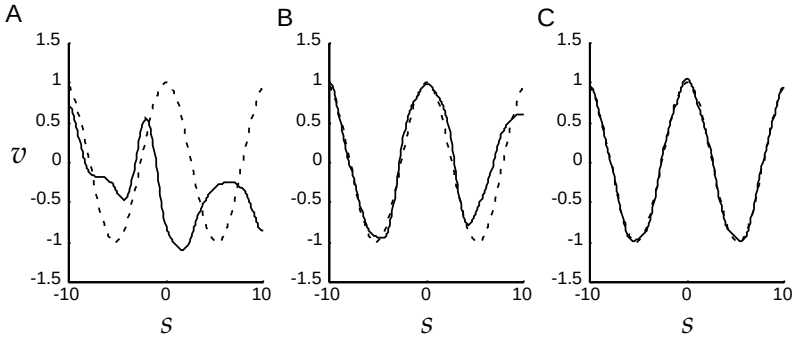

Figure 8.14 Eleven input neurons with Gaussian tuning curves drive an output neuron to approximate a sine function. The input tuning curves are $f_b(s) = \exp[-0.5(s - s_b)^2]$ with $s_b = -10, -8, -6, \ldots, 8, 10$. The delta rule was used to adjust the weights. Sample points were chosen randomly with s in the range between -10 and 10. The firing rate of the output neuron is plotted as a solid curve, and the sinusoidal target function as a dashed curve. (A) The firing rate of the output neuron when random weights in the range between -1 and 1 were used. (B) The output firing rate after weight modification using the delta rule with 20 sample points. (C) The output firing rate after weight modification using the delta rule with 100 sample points.

Gaussian tuning curves drives an output neuron so that it quite accurately represents a sine function. Figures 8.14B and C illustrate the difference between storage and generalization. The output $v(s)$ in figure 8.14B matches the sine function well for values of s that were in the training set, and, in this sense, has stored that information. However, it does not generalize well, in that it does not match the sine function for other values of s not in the training set. The output $v(s)$ in figure 8.14C has good storage and generalization properties, at least within the range of values of s used. The ability of the network to approximate the function $h(s)$ for stimulus values not presented during training depends in a complicated way on the smoothness of the target function, the number and smoothness of the basis functions $\mathbf{f}(s)$, and the size of the training set.

It is not immediately obvious how the delta rule of equation 8.61 could be implemented biophysically, because the network has to compute the difference $h(s^m)\mathbf{f}(s^m) - v(s^m)\mathbf{f}(s^m)$. One possibility is that the two terms $h(s^m)\mathbf{f}(s^m)$ and $v(s^m)\mathbf{f}(s^m)$ are computed in separate phases. First, the output of the network is clamped to the desired value $h(s^m)$ and Hebbian plasticity is applied. Then, the network runs freely to generate $v(s^m)$ and anti-Hebbian modifications are made. In the next section, we discuss a particular example of this in the case of the Boltzmann machine, and we show how learning rules intended for supervised learning can sometimes be used for unsupervised learning as well.

Contrastive Hebbian Learning in Stochastic Networks

In chapter 7, we presented the Boltzmann machine, a stochastic network with binary units. One of the key innovations associated with the Boltzmann machine is a synaptic modification rule that has a sound foundation in probability theory. We start by describing the case of supervised learning, although the underlying theory is similar for both supervised and unsupervised cases with the Boltzmann machine.

Recall from chapter 7 that the Boltzmann machine produces a stochastic output $\mathbf{v}$ from an input $\mathbf{u}$ through a process called Gibbs sampling. This has two main consequences. First, rather than being described by an equation such as 8.2, the input-output relationship of a stochastic network is described by a distribution $P[\mathbf{v}|\mathbf{u}; \mathbf{W}]$, which is the probability that input $\mathbf{u}$ generates output $\mathbf{v}$ when the weight matrix of the network is $\mathbf{W}$. Second, supervised learning in a deterministic network involves the development of an input-output relationship that matches, as closely as possible, a set of samples $(\mathbf{u}^m, \mathbf{v}^m)$ for $m = 1, 2, \ldots, N_S$. Such a task does not make sense for a model that has a stochastic rather than deterministic relationship between its input and output activities. Instead, stochastic networks are appropriate for representing statistical aspects of the relationship between two sets of variables. For example, suppose that we drew sample pairs from a joint probability distribution $P[\mathbf{u}, \mathbf{v}] = P[\mathbf{v}|\mathbf{u}]P[\mathbf{u}]$. In this case, a given value of $\mathbf{u}^m$, chosen from the distribution $P[\mathbf{u}]$, is associated with another vector $\mathbf{v}^m$ stochastically rather than deterministically. The probability that a particular output $\mathbf{v}^m$ is associated with $\mathbf{u}^m$ is given by $P[\mathbf{v}^m|\mathbf{u}^m]$. In other words, $P[\mathbf{u}]$ describes the probability of various $\mathbf{u}$ vectors appearing, and $P[\mathbf{v}|\mathbf{u}]$ is the probability that a particular vector $\mathbf{u}$ is associated with another vector $\mathbf{v}$.

A natural supervised learning task for a stochastic network is to make the input-output distribution of the network, $P[\mathbf{v}|\mathbf{u}; \mathbf{W}]$, match as closely as possible, the probability distribution $P[\mathbf{v}|\mathbf{u}]$ associated with the samples $(\mathbf{u}^m, \mathbf{v}^m)$. This is done by adjusting the feedforward weight matrix $\mathbf{W}$. Note that we are using the argument $\mathbf{W}$ to distinguish between two different distributions, $P[\mathbf{u}|\mathbf{v}]$, which is provided externally and generates the sample data, and $P[\mathbf{u}|\mathbf{v}; \mathbf{W}]$, which is the distribution generated by the Boltzmann machine with weight matrix $\mathbf{W}$. The idea of constructing networks that reproduce probability distributions inferred from sample data is central to the problem of density estimation, which is covered more fully in chapter 10.

density estimation

We first consider a Boltzmann machine with only feedforward weights $\mathbf{W}$ connecting $\mathbf{u}$ to $\mathbf{v}$, and no recurrent weight among the $\mathbf{v}$ units. Given input $\mathbf{u}$, an output $\mathbf{v}$ is computed by setting each component v_a to 1 with probability $F(\sum_b W_{ab} u_b)$ (and 0 otherwise) where $F(I) = 1/(1 + \exp(-I))$. This is the Gibbs sampling procedure discussed in chapter 7 applied to the feedforward Boltzmann machine. Because there are no recurrent connections, the states of the output units are independent, and they can all be sampled simultaneously. Analogous to the discussion in chapter 7, this

procedure gives rise to a conditional probability distribution $P[\mathbf{v}|\mathbf{u};\mathbf{W}]$ for $\mathbf{v}$ given $\mathbf{u}$ that can be written as

$$P[\mathbf{v}|\mathbf{u};\mathbf{W}] = \frac{\exp(-E(\mathbf{u},\mathbf{v}))}{Z(\mathbf{u})} \quad \text{with} \quad Z(\mathbf{u}) = \sum_{\mathbf{v}} \exp(-E(\mathbf{u},\mathbf{v}))\,, \tag{8.62}$$

where $E(\mathbf{u},\mathbf{v}) = -\mathbf{v}\cdot\mathbf{W}\cdot\mathbf{u}$. In this case, the partition function can be written as $Z(\mathbf{u}) = \prod_a (1+\exp(\sum_b W_{ab}u_b))$. However, for the Boltzmann machines with recurrent connections that we consider below, there is no simple closed form expression for the partition function.

The natural measure for determining how well the distribution generated by the network, $P[\mathbf{v}|\mathbf{u};\mathbf{W}]$, matches the sampled distribution, $P[\mathbf{v}|\mathbf{u}]$, for a particular input $\mathbf{u}$ is the Kullback-Leibler divergence,

$$\begin{aligned} D_{\mathrm{KL}}(P[\mathbf{v}|\mathbf{u}],\, P[\mathbf{v}|\mathbf{u};\mathbf{W}]) &= \sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u}] \ln\left(\frac{P[\mathbf{v}|\mathbf{u}]}{P[\mathbf{v}|\mathbf{u};\mathbf{W}]}\right) \\ &= -\sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u}] \ln\left(P[\mathbf{v}|\mathbf{u};\mathbf{W}]\right) + K\,, \end{aligned} \tag{8.63}$$

where K is a term that is proportional to the entropy of the distribution $P[\mathbf{v}|\mathbf{u}]$ (see chapter 4). We do not write out this term explicitly because it does not depend on the feedforward weight matrix, so it does not affect the learning rule used to modify $\mathbf{W}$. As in chapter 7, we have, for convenience, used natural logarithms (rather than base 2 as in chapter 4) in the definition of the Kullback-Leibler divergence.

To estimate, from the samples, how well $P[\mathbf{v}|\mathbf{u};\mathbf{W}]$ matches $P[\mathbf{v}|\mathbf{u}]$ across the different values of $\mathbf{u}$, we average the Kullback-Leibler divergence over all the input samples $\mathbf{u}^m$. Furthermore, the sum over all $\mathbf{v}$ with weighting factor $P[\mathbf{v}|\mathbf{u}]$ in equation 8.63 can be replaced with an average over the sample pairs $(\mathbf{u}^m,\mathbf{v}^m)$ because $\mathbf{v}^m$ is chosen from the probability distribution $P[\mathbf{v}|\mathbf{u}]$. Using brackets to denote the average over samples, this results in the measure

$$\langle D_{\mathrm{KL}}(P[\mathbf{v}|\mathbf{u}],\, P[\mathbf{v}|\mathbf{u};\mathbf{W}])\rangle = -\frac{1}{N_{\mathrm{S}}}\sum_{m=1}^{N_{\mathrm{S}}} \ln\left(P[\mathbf{v}^m|\mathbf{u}^m;\mathbf{W}]\right) + \langle K\rangle \tag{8.64}$$

for comparing $P[\mathbf{v}|\mathbf{u};\mathbf{W}]$ and $P[\mathbf{v}|\mathbf{u}]$. Each logarithmic term in the sum on the right side of this equation is the logarithm of the probability that a sample output $\mathbf{v}^m$ would have been drawn from the distribution $P[\mathbf{v}|\mathbf{u}^m;\mathbf{W}]$, when in fact it was drawn from $P[\mathbf{v}|\mathbf{u}^m]$. This makes the sum in equation 8.64 equal to the logarithm of the likelihood that the sample data could have been produced from $P[\mathbf{v}|\mathbf{u}^m;\mathbf{W}]$. As a result, finding the network distribution $P[\mathbf{v}|\mathbf{u}^m;\mathbf{W}]$ that best matches $P[\mathbf{v}|\mathbf{u}^m]$ (in the sense of minimizing the Kullback-Leibler divergence) is equivalent to maximizing the likelihood that the sample $\mathbf{v}^m$ could have been drawn from $P[\mathbf{v}|\mathbf{u}^m;\mathbf{W}]$.

log likelihood

likelihood maximization

A learning rule that performs gradient ascent of the log likelihood can be derived by changing the weights by an amount proportional to the derivative of the logarithmic term in equation 8.64 with respect to the weights.

As in the discussion of the delta rule, it is more convenient to use a stochastic gradient ascent rule, choosing an index m at random to provide a Monte Carlo sample from the average of equation 8.64, and changing W_{ab} according to the derivative with respect to this sample,

$$\frac{\partial \ln P[\mathbf{v}^m|\mathbf{u}^m;\mathbf{W}]}{\partial W_{ab}} = \frac{\partial}{\partial W_{ab}}\Big(-E(\mathbf{u}^m,\mathbf{v}^m) - \ln Z(\mathbf{u}^m)\Big)$$
$$= v_a^m u_b^m - \sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u}^m;\mathbf{W}] v_a u_b^m \,. \tag{8.65}$$

This derivative has a simple form for the Boltzmann machine because of equation 8.62.

Before we derive the stochastic gradient ascent learning rule, we need to evaluate the sum over $\mathbf{v}$ in the last term of the bottom line of equation 8.65. For Boltzmann machines with recurrent connections, like the ones we discuss below, this average cannot be calculated tractably. However, it can be estimated by stochastic sampling. In other words, we approximate the average over $\mathbf{v}$ by a single instance of a particular output $\mathbf{v}(\mathbf{u}^m)$ generated by the Boltzmann machine in response to the input $\mathbf{u}^m$. Making this replacement and setting the change in the weight matrix proportional to the derivative in equation 8.65, we obtain the learning rule

supervised learning for $\mathbf{W}$

$$W_{ab} \rightarrow W_{ab} + \epsilon_w \left(v_a^m u_b^m - v_a(\mathbf{u}^m) u_b^m\right) \,. \tag{8.66}$$

Equation 8.66 is identical in form to the perceptron learning rule of equation 8.56, except that $\mathbf{v}(\mathbf{u}^m)$ is computed from the input $\mathbf{u}^m$ by Gibbs sampling rather than by a deterministic rule. As discussed at the end of the previous section, equation 8.66 can also be interpreted as the difference of Hebbian and anti-Hebbian terms. The Hebbian term $v_a^m u_b^m$ is based on the sample input $\mathbf{u}^m$ and output $\mathbf{v}^m$. The anti-Hebbian term $-v_a(\mathbf{u}^m)u_b^m$ involves the sample input $\mathbf{u}^m$ and an output $\mathbf{v}(\mathbf{u}^m)$ generated by the Boltzmann machine in response to this input, rather than the sample output $\mathbf{v}^m$. In other words, whereas $\mathbf{v}^m$ is provided externally, $\mathbf{v}(\mathbf{u}^m)$ is obtained by Gibbs sampling, using the input $\mathbf{u}^m$ and the current values of the network weights. The overall learning rule is called a contrastive Hebb rule because it depends on the difference between Hebbian and anti-Hebbian terms. W_{ab} stops changing when the average of $v_a(\mathbf{u}^m)u_b^m$ over the input samples and network outputs equals the average of $v_a^m u_b^m$ over the input and output samples.

contrastive Hebb rule

Supervised learning for the Boltzmann machine is run in two phases, both of which use a sample input $\mathbf{u}^m$. The first phase, sometimes called the wake phase, involves Hebbian plasticity between sample inputs and outputs. The dynamics of the Boltzmann machine play no role during this phase. The second phase, called the sleep phase, consists of the network "dreaming" (i.e., internally generating) $\mathbf{v}(\mathbf{u}^m)$ in response to $\mathbf{u}^m$ based on the current weights $\mathbf{W}$. Then, anti-Hebbian learning based on $\mathbf{u}^m$ and $\mathbf{v}(\mathbf{u}^m)$ is applied to the weight matrix. Gibbs sampling is typically used to generate $\mathbf{v}(\mathbf{u}^m)$ from $\mathbf{u}^m$. It is also possible to use the mean-field method

wake phase

sleep phase

we discussed in chapter 7 to approximate the average over the distribution $P[\mathbf{v}|\mathbf{u}^m; \mathbf{W}]$ in equation 8.65.

Supervised learning can also be implemented in a Boltzmann machine with recurrent connections. When the output units are connected by a symmetric recurrent weight matrix $\mathbf{M}$ (with $M_{aa} = 0$), the energy function is

$$E(\mathbf{u}, \mathbf{v}) = -\mathbf{v} \cdot \mathbf{W} \cdot \mathbf{u} - \frac{1}{2}\mathbf{v} \cdot \mathbf{M} \cdot \mathbf{v} . \tag{8.67}$$

Everything that has been described thus far applies to this case, except that the output $\mathbf{v}(\mathbf{u}^m)$ for the sample input $\mathbf{u}^m$ must now be computed by repeated Gibbs sampling, using $F(\sum_b W_{ab} u_b^m + \sum_{a'} M_{aa'} v_{a'})$ for the probability that $v_a = 1$ (see chapter 7). Repeated sampling is required to assure that the network relaxes to the equilibrium distribution of equation 8.62. Modification of the feedforward weight W_{ab} then proceeds as in equation 8.66. The contrastive Hebb rule for recurrent weight $M_{aa'}$ is similarly given by

supervised learning for $\mathbf{M}$

$$M_{aa'} \rightarrow M_{aa'} + \epsilon_m \left(v_a^m v_{a'}^m - v_a(\mathbf{u}^m) v_{a'}(\mathbf{u}^m) \right) . \tag{8.68}$$

The Boltzmann machine was originally introduced in the context of unsupervised rather than supervised learning. In the supervised case, we try to make the distribution $P[\mathbf{v}|\mathbf{u}; \mathbf{W}]$ match the probability distribution $P[\mathbf{v}|\mathbf{u}]$ that generates the samples pairs $(\mathbf{u}^m, \mathbf{v}^m)$. In the unsupervised case, no output sample $\mathbf{v}^m$ is provided, and instead we try to make the network generate a probability distribution over $\mathbf{u}$ that matches the distribution $P[\mathbf{u}]$ from which the samples $\mathbf{u}^m$ were drawn. As we discuss in chapter 10, a frequent goal of probabilistic unsupervised learning is to generate network distributions that match the distributions of input data.

We consider the unsupervised Boltzmann machine without recurrent connections. In addition to the distribution of equation 8.62 for $\mathbf{v}$, given a specific input $\mathbf{u}$, the energy function of the Boltzmann machine can be used to define a distribution over both $\mathbf{u}$ and $\mathbf{v}$ defined by

$$P[\mathbf{u}, \mathbf{v}; \mathbf{W}] = \frac{\exp(-E(\mathbf{u}, \mathbf{v}))}{Z} \quad \text{with} \quad Z = \sum_{\mathbf{u},\mathbf{v}} \exp(-E(\mathbf{u}, \mathbf{v})) . \tag{8.69}$$

This can be used to construct a distribution for $\mathbf{u}$ alone by summing over the possible values of $\mathbf{v}$,

$$P[\mathbf{u}; \mathbf{W}] = \sum_{\mathbf{v}} P[\mathbf{u}, \mathbf{v}; \mathbf{W}] = \frac{1}{Z} \sum_{\mathbf{v}} \exp(-E(\mathbf{u}, \mathbf{v})) . \tag{8.70}$$

The goal of unsupervised learning for the Boltzmann machine is to make this distribution match, as closely as possible, the distribution of inputs P[$\mathbf{u}$].

The derivation of an unsupervised learning rule for this Boltzmann machine proceeds very much like the derivation we presented for the supervised case. The equivalent of equation 8.65 is

$$\frac{\partial \ln P[\mathbf{u}^m; \mathbf{W}]}{\partial W_{ab}} = \sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u}^m; \mathbf{W}] v_a u_b^m - \sum_{\mathbf{u},\mathbf{v}} P[\mathbf{u}, \mathbf{v}; \mathbf{W}] v_a u_b \, . \qquad (8.71)$$

In this case, both terms must be evaluated by Gibbs sampling. The wake phase Hebbian term requires a stochastic output $\mathbf{v}(\mathbf{u}^m)$, which is calculated from the sample input $\mathbf{u}^m$, just as it was for the anti-Hebbian term in equation 8.66. However, the sleep phase anti-Hebbian term in this case requires both an input $\mathbf{u}$ and an output $\mathbf{v}$ generated by the network. These are computed using a Gibbs sampling procedure in which both input and output states are stochastically generated through repeated Gibbs sampling. A randomly chosen component v_a is set to 1 with probability $F(\sum_b W_{ab} u_b)$ (or 0 otherwise), and a random component u_b is set to 1 with probability $F(\sum_a v_a W_{ab})$ (or 0 otherwise). Note that this corresponds to having the input units drive the output units in a feedforward manner through the weights $\mathbf{W}$, and having the output units drive the input units in a reversed manner using feedback weights with the same values. Once the network has settled to equilibrium through repeated Gibbs sampling of this sort, and the stochastic inputs and outputs have been generated, the full learning rule is

unsupervised learning for $\mathbf{W}$

$$W_{ab} \rightarrow W_{ab} + \epsilon_w \left(v_a(\mathbf{u}^m) u_b^m - v_a u_b \right) \, . \qquad (8.72)$$

The unsupervised learning rule can be extended to include recurrent connections by following the same general procedure.

8.5 Chapter Summary

We presented a variety of forms of Hebbian synaptic plasticity, ranging from the basic Hebb rule to rules that involve multiplicative and subtractive normalization, constant or sliding thresholds, and spike-timing effects. Two important features, stability and competition, were emphasized. We showed how the effects of unsupervised Hebbian learning could be estimated by computing the principal eigenvector of the correlation matrix of the inputs used during training. Unsupervised Hebbian learning can be interpreted as a process that produces weights that project the input vector onto the direction of maximal variance in the training data set. In some cases, this requires an extension from correlation-based to covariance-based rules. We used the principal eigenvector approach to analyze Hebbian models of the development of ocular dominance and its associated map in primary visual cortex. Plasticity rules based on the dependence of synaptic modification on spike timing were shown to implement temporal sequence and trace learning.

Forcing multiple outputs to have different selectivities requires them to be connected, either through fixed weights or by weights that are themselves

plastic. In the latter case, anti-Hebbian plasticity can ensure decorrelation of multiple output units. We also considered the role of competition and cooperation in models of activity-dependent development, and described two examples of feature-based models, the self-organizing map and the elastic net.

Finally, we considered supervised learning applied to binary classification and function approximation, using supervised Hebbian learning, the perceptron learning rule, and gradient descent learning through the delta rule. We also treated contrastive Hebbian learning for the Boltzmann machine, involving Hebbian and anti-Hebbian updates in different phases.

8.6 Appendix

Convergence of the Perceptron Learning Rule

For convenience, we take $\epsilon_w = 1$ and start the perceptron learning rule (equation 8.56) with $\mathbf{w} = \mathbf{0}$ and $\gamma = 0$. Then, under presentation of the sample m, the changes in the weights and threshold are given by

$$\Delta \mathbf{w} = \frac{1}{2}(v^m - v(\mathbf{u}^m))\mathbf{u}^m \quad \text{and} \quad \Delta\gamma = -\frac{1}{2}(v^m - v(\mathbf{u}^m)) \,. \tag{8.73}$$

Given a finite, linearly separable problem, there must be a set of weights $\mathbf{w}^*$ and a threshold γ^* that are normalized ($|\mathbf{w}^*|^2 + (\gamma^*)^2 = 1$) and allow the perceptron to categorize correctly, for which we require the condition $(\mathbf{w}^* \cdot \mathbf{u}^m - \gamma^*)v^m > \delta$ for some $\delta > 0$ and for all m.

Consider the cosine of the angle between the current weights and threshold $\mathbf{w}$, γ and the solution $\mathbf{w}^*$, γ^*

$$\Phi(\mathbf{w}, \gamma) = \frac{\mathbf{w} \cdot \mathbf{w}^* + \gamma\gamma^*}{\sqrt{|\mathbf{w}|^2 + (\gamma)^2}} = \frac{\psi(\mathbf{w}, \gamma)}{|\mathbf{w}, \gamma|} \,, \tag{8.74}$$

to introduce some compact notation. The perceptron convergence theorem proves that the perceptron learning rule solves any solvable categorization problem, because assuming otherwise would imply that Φ would eventually grow larger than 1. This is impossible for a cosine function, which must lie between -1 and 1.

To show this, we consider the change in ψ due to one step of perceptron learning during which $\mathbf{w}$ and γ are modified because the current weights generated the wrong response. When an incorrect response is generated, $v(\mathbf{u}^m) = -v^m$, so $(v^m - v(\mathbf{u}^m))/2 = v^m$, and thus

$$\Delta\psi = (\mathbf{w}^* \cdot \mathbf{u}^m - \gamma^*)v^m > \delta \,. \tag{8.75}$$

The inequality follows from the condition imposed on $\mathbf{w}^*$ and γ^* as providing a solution of the categorization problem. Assuming that ψ is initially positive and iterating this result over n steps in which the weights

change, we find that

$$\psi(\mathbf{w}, \gamma) \geq n\delta . \tag{8.76}$$

Similarly, over one learning step in which some change is made,

$$\Delta|\mathbf{w}, \gamma|^2 = 2(\mathbf{w} \cdot \mathbf{u}^m - \gamma)v^m + |\mathbf{u}^m|^2 + 1 . \tag{8.77}$$

The first term on the right side is always negative when an error is made, and if we define D to be the maximum value of $|\mathbf{u}^m|^2$ over all the training samples, we find

$$\Delta|\mathbf{w}, \gamma|^2 < D + 1 . \tag{8.78}$$

After n nontrivial learning iterations (iterations in which the weights and threshold are modified) starting from $|\mathbf{w}, \gamma|^2 = 0$, we therefore have

$$|\mathbf{w}, \gamma|^2 < n(D + 1) \tag{8.79}$$

Putting together equations 8.76 and 8.79, we find, after n nontrivial training steps,

$$\Phi(\mathbf{w}, \gamma) > \frac{n\delta}{\sqrt{n(D+1)}} . \tag{8.80}$$

To ensure that $\Phi(\mathbf{w}, \gamma) \leq 1$, we must have $n \leq (D + 1)/\delta^2$. Therefore, after a finite number of weight changes, the perceptron learning rule must stop changing the weights, and the perceptron must classify all the patterns correctly (although the weights and threshold that result are not necessarily equal to $\mathbf{w}^*$ and γ^*).

8.7 Annotated Bibliography

Hebb's (1949) original proposal about learning set the stage for many of the subsequent investigations. We followed the treatments of Hebbian, BCM, anti-Hebbian, and trace learning found in Goodall (1960), Sejnowski (1977), Bienenstock et al. (1982), Oja (1982), Földiák (1989; 1991), Leen (1991), Atick & Redlich (1993), and Wallis & Baddeley (1997). Extensive coverage of these topics and related analyses can be found in **Hertz et al. (1991)**. We followed Miller & MacKay (1994) and Miller (1996b) in the analysis of weight constraints and normalization. Jolliffe (1986) treats principal components analysis theoretically (see also chapter 10). Intrator & Cooper (1992) considers the BCM rule from the statistical perspective of projection pursuit (Huber, 1985).

Sejnowski (1999) comments on the relationship between Hebb's suggestions and recent experimental data and theoretical studies on temporal sensitivity in Hebbian plasticity (see Minai & Levy, 1993; Blum & Abbott,

1996; Kempter et al., 1999; Song et al., 2000). Plasticity of intrinsic conductance properties of neurons, as opposed to synaptic plasticity, is considered in LeMasson et al. (1993), Liu et al. (1999), and Stemmler & Koch (1999).

Descriptions of relevant data on the patterns of responsivity across cortical areas and the development of these patterns include Hubener et al. (1997), **Yuste & Sur (1999)**, and Weliky (2000). **Price & Willshaw (2000)** offers a broad-based, theoretically informed review. Recent experimental challenges to plasticity-based models include Crair et al. (1998) and Crowley & Katz (1999). Neural pattern formation mechanisms involving chemical matching, which are likely important at least for establishing coarse maps, are reviewed from a theoretical perspective in **Goodhill & Richards (1999)**. The use of learning algorithms to account for cortical maps is reviewed in **Erwin et al. (1995)**, **Miller (1996a)**, and **Swindale (1996)**. The underlying mathematical basis of some rules is closely related to the reaction diffusion theory of morphogenesis of **Turing (1952)**. Other rules are motivated on the basis of minimizing quantities such as wire length in cortex.

We described Hebbian models for the development of ocular dominance and orientation selectivity due to Linsker (1986), Miller et al. (1989), and Miller (1994); a competitive Hebbian model closely related to that of Goodhill (1993) and Piepenbrock & Obermayer (1999); a self-organizing map model related to that of Obermayer et al. (1992); and the elastic net (Durbin & Willshaw, 1987) model of Durbin & Mitchison (1990), Goodhill & Willshaw (1990), and **Erwin et al. (1995)**. The first feature-based models were called noise models (see **Swindale, 1996**).

The capacity of a perceptron for random associations is computed in Cover (1965) and Venkatesh (1986). The perceptron learning rule is due to Rosenblatt (1958; see also **Minsky & Papert, 1969**). The delta rule was introduced by Widrow & Hoff (1960; see also Widrow & Stearns, 1985) and independently arose in other fields. The widely used backpropagation algorithm (see Chauvin & Rumelhart, 1995) is a form of delta rule learning that works in a larger class of networks. O'Reilly (1996) suggests a more biologically plausible implementation.

Supervised learning for classification and function approximation, and its ties to Bayesian and frequentist statistical theory, are reviewed in **Duda & Hart (1973)**, **Duda at al. (2000)**, **Kearns & Vazirani (1994)**, and **Bishop (1995)**. Poggio and colleagues have explored basis function models of various representational and learning phenomena (see Poggio, 1990). Tight frames are discussed in Daubechies et al. (1986) and are applied to visual receptive fields by Salinas & Abbott (2000).

Contrastive Hebbian learning is due to Hinton & Sejnowski (1986). See Hinton (2000) for discussion of the Boltzmann machine without recurrent connections, and for an alternative learning rule.