Response Modulation

A Mechanism for the Guidance of Learning

A Dissertation Presented to
The Faculty of the Graduate School of Arts and Sciences
Brandeis University, Program in Neuroscience

Laurence F. Abbott, Advisor

In Partial Fulfillment of the Requirements
for the Degree Doctor of Philosophy

by

Christian D. Swinehart

May 2005

Samizdat Drafting Co., Somerville, MA.

Printed in Cambridge, MA.

The paper used in this publication may meet the minimum requirements of the American National Standard for Information Sciences—Permanence of Paper for Printed Library Materials, ANSI Z39.48–1984.

Sections of this work have appeared previously in slightly modified form.

Chapter 2 was published as:

> Swinehart, CD, Bouchard, K, Partensky, P, & Abbott LF (2004). Control of network activity through neuronal response modulation. *Neurocomputing* **58–60**:327–335.

and chapter 3 as:

> Swinehart, CD, & Abbott, LF (2005). Supervised learning through neuronal response modulation. *Neural Computation* **17**:609–631.

Colophon
This dissertation was produced with the help of Donald Knuth's hoary but magnificent typesetting software, TEX. I've also made use of Leslie Lamport's LATEX extensions to the language for proper long document formatting, and am deeply indebted to Peter Wilson for writing the 'memoir' document class which made construction of the stylesheet practically (rather than just theoretically) possible.

The typeface used for body text is Adobe Jenson. Designed by Robert Slimbach, Jenson is modeled after two renaissance typefaces, the roman by Nicolas Jenson, and the italic by Ludovico Arrighi.

Captions and headings are set in DTL Caspari, published by the Dutch Type Library and designed by Gerard Dainëls in 1993.

SIGNATURES

This dissertation, directed and approved by Christian Swinehart's Committee, has been accepted and approved by the Faculty of Brandeis University in partial fulfillment of the requirements for the degree of:

DOCTOR OF PHILOSOPHY

Adam B Jaffe, Dean of Arts and Sciences

DISSERTATION COMMITTEE

Laurence F. Abbott, Ph. D., Department of Biology

Jorge V. José, Ph. D., Northeastern Univ. Department of Physics

Eve Marder, Ph. D., Department of Biology

Xiao Jing Wang, Ph. D., Department of Physics

ACKNOWLEDGMENTS

This work could not have been completed were it not for the support of lots of people. The cast of characters who contributed most directly were those sitting to either side of me in the Abbott lab over the last four years. Many thanks go to:

Patrick Drew, whose combination of curiosity and hectoring forced me to tighten up my arguments and continually reëvaluate my assumptions.

Tim Vogels, whose input on matters of visual communication also put pressure on me to not fall to the level of 'zero information content' in the pursuit of æsthetics.

Peretz Partensky & Kris Bouchard, the undergrad team responsible for much of the initial work on this project while I spent my days on 2D rendering problems.

Cliff Rumsey, for teaching me by example how one gives a painfully eloquent presentation of scientific work.

Kanaka Rajan, for providing a degree of drama which otherwise would not have entered into our staid lab environment.

But beyond the direct contributors, I most certainly would not have survived this experience without the presence in my life of so many people who've walked alongside me throughout these years. Of those in this group I am particularly indebted to Tanya Casimiro, Bob Cudmore, Jenny Mehren, Isaac Kurtzer, Shiva Viswanathan, and Kris Rehm—all of whom have reminded me, in one way or another, how much beauty and value there is in the non-scientific world.

But above all I offer my most heartfelt and unconditional appreciation to Larry Abbott, my advisor and mentor over this period. If he'd limited his guidance strictly to scientific matters I'd already be guilty of receiving more wisdom and insight than any grad student has a right to expect from his P.I. But Larry's impact on my life has gone far beyond that, whether by saving me from a career in computer science, talking me through traumatic summers, or supporting, with compassion and enthusiasm, the craziest plans I could possibly come to him with, he's been there for me in ways that still leave me dumbfounded with gratitude.

I feel lucky beyond words for the privilege of having been his student. And I regularly thank whatever stochastic process led him to welcome me into his lab in the first place.

ABSTRACT

Neural network learning is a topic that has received a substantial amount of attention over the past few decades from two scientific communities, divided in their interests between ends and means. The first, composed of computer scientists searching for new computational techniques has been wholly focused on end results. As a result a large body of optimization techniques has been amassed for modifying networks to perform arbitrary tasks. But this has been done without much thought being given to how such models could correspond to, or be implemented in, biological form. The second community, experimental neuroscience, addresses behavioral end results as well, but due to its bottom-up approach—and the young age of the field—has much to say about the means and mechanisms by which networks change on a micro level, but has only rarely been able to offer explanations that span all levels, from molecular mechanism to organism behavior.

One commonality in the two approaches is that neural network learning is typically treated as the problem of setting synaptic connection strengths to better perform a task. This requires some sort of feedback by which a behavioral-level readout of success can be transmitted back to the network responsible for producing that behavior, and modifications can be made to reduce whatever error is currently being made.

The computer science approach proposes the existence of a Supervisor external to the network which mediates this transfer, reaching in and causing synaptic plasticity to adjust function until some optimal state is reached. However, anatomical data suggest that direct synaptic modification by such a supervisor circuit would be infeasible. We investigate supervision at the level of neurons rather than synapses. By modulating the response properties of cells in the network, this form of supervised learning is able to successfully train a network to perform a function approximation task. We examine the nature of this modulation, consider its implications for supervised learning, and propose a plausible neural basis for this Supervisor circuitry.

Contents

CHAPTER ONE

Preliminary Matters

LIKE MANY OTHER terms used in the sciences, 'learning' has been taken from common usage and invested with a more concrete meaning within the context of neuroscience. Unfortunately in this case it has received not a single definition, but a multitude of them, each specific to a particular level of inquiry, and most in conflict with one another. Thus it is important to establish from the beginning the scope of the investigation discussed in these pages, and accordingly the breadth of the claims made therein.

In the context of the work that follows, learning can best be thought of as a set of functional changes in a neural network coordinated in such a way as to improve its performance on an arbitrary processing task. This is a topic which has received much attention previously using the computational modeling methods that we will employ. Thus, to begin, we will first define the terms of this investigation, review the relevant prior approaches to the problem, then outline our own attempt to answer these questions.

Overture

Defining learning simply as a pattern of coordinated, functional changes manages to capture the spirit of our view of the phenomenon, but clearly glosses over a set of sub-problems which arise when one tries to explain what is happening in an actual living system during such a process. First there is the issue of precisely what form these changes take.

To change the function of a neural network there are two primary locations at which to act: the synapses connecting the neurons, or the neurons themselves. Surely both synaptic plasticity and modifications intrinsic to the neurons occur, but the traditional assumption is that synaptic change is predominant—particularly on the fast time scales associated with adaptive, task-dependent learning. Later we will suggest that intrinsic changes might act as precursors to synaptic change.

Regardless of the actual intrinsic/extrinsic modification balance in biological systems, most of the sorts of behavioral change one wishes to bring about in an artificial neural network can be realized through synaptic plasticity alone. As a result, much prior work—at least from the artificial intelligence (AI) or machine learning (ML) perspective—has focused upon the modulation of synaptic strength to influence network function. However even having simplified the problem by considering only changes extrinsic to the cells, there are still two subproblems which are typically conflated:

1. A strategy for choosing which synapses to strengthen and which to weaken is required.
2. A mechanism must be provided by which to change these synaptic coupling strengths to their desired values.

MODIFICATION STRATEGIES

It is to this question of how best to go about altering synaptic strengths in order to bring a network to some functional end state that the ML/AI project has devoted itself, possibly to the exclusion of other considerations. As a result there is an extensive library of different algorithms which are all optimal in different ways, whether in terms of sheer speed or in catering to one form of 'biological plausibility' or another (Hinton, 1989; Kearns and Vazirani, 1994; Narendra and Thatachar, 1989; Sutton and Barto, 1998; Dayan and Abbott, 2001). Thus, given a network and a desired function, these off-the-shelf strategies will find the proper set of synaptic weights to perform that task.

In addition, there is a wealth of biological data documenting the properties of functioning, tuned neural systems (Dickinson, 1980; Gallistel, 1990). This again allows us to verify that one of these end states is 'realistic' along any number of parameters, ranging from firing rates and mix of conductances in individual cells to the overall balance of excitation and inhibition within the network.

Thus we have a clear picture of what a successful network ought to look like, both in the aggregate and in terms of the relative strengths of each synapse in the network. Presuming there is a way to arbitrarily impose synaptic strengths on a network, the problem is solved. The difficulty is that this presumption is untrue.

PLASTICITY MECHANISMS

It is here that the ML-supplied approaches to 'training' a network begin to break down. By focusing predominantly on finding a proper set of synaptic strengths for a given task, the problem of putting those weights in place has typically been treated as a secondary consideration. Though ultimately this poses as great a problem as determining what those weights ought to be in the first place.

Experimental data has provided a fairly solid picture of how synapses change at the most local levels. But it is less obvious how these single-synapse or single-pair rules function in the larger network to produce coordinated, goal directed tuning of its activity. Instead, most of what has been discovered points to an enhancement of preëxisting, local function rather than the creation new specializations within the network.

However, there is a pair of key results that will be returned to repeatedly in the chapters that follow. First is the well established correlative strengthening effect (i.e., Hebbian Plasticity) in which the simultaneous activity of a pre- and postsynaptic cell will increase the efficacy of the synapse connecting them (Bliss and Collingridge, 1993; Bredt and Nicoll, 2003).

The second effect is more recently characterized, but also seems to be fairly universal among cortical cells. It is commonly observed that cells tend to receive nearly balanced amounts of excitatory and inhibitory current. However the absolute values of these respective currents have not typically been attended to, merely the net current. And while it is true that only net current affects a cell's probability of firing an action potential, whether the balanced currents are large or small can have a significant modulatory effect on the cell's excitability (Shadlen and Newsome, 1994, Troyer and Miller, 1997).

The interplay between these two seemingly unrelated cell- and synapse-level properties may in fact offer a solution to the problem of the missing mechanism for network learning—a solution that is introduced in chapter 3 and elaborated upon in chapter 4. But before examining how the various pieces fit together into a coherent learning scheme, it would be worthwhile to examine each in more detail.

Prior Art

This research sits at the intersection between two bodies of research. On the one side sits the ML/AI approach which is principally concerned with optimal strategies for search within the problem space of synaptic strengths, with a dual focus upon perfecting the performance of the network in its final task-adapted form and upon minimizing the amount of time needed to reach that state. On the other side is the experimental biology approach which instead takes a more bottom-up approach, documenting the biophysical mechanisms which can actually be observed, with the hope of scaffolding up to explanations of progressively more complex neural behaviors.

However, neither of these approaches has entirely delivered upon its promises. The ML/AI field is typically less concerned with biological realism, and the difficulties of monitoring the behaviors and interactions of large numbers of cells places a limit on the phenomenological complexity that neuroscience can address (for the moment). Thus, this work is an attempt to forge some connections between the formal, bloodless insights of machine learning, and the bulletproof, yet myopic, discoveries of the experimentalists. As a result this work is firmly synthetic, drawing a set of building blocks from both sides of this divide and attempting to build something out of the most compatible elements of each. This section is concerned with defining those blocks.

STRATEGIES FROM MACHINE LEARNING

The real accomplishment of the ML/AI program was the development of formal models for network dynamics and the discovery of algorithms that near-optimally modify neural networks to change their function. But the generality of this perspective leads to an agnosticism over how the target network behavior is reached. Thus researchers divided the problem into two categories: one considering cases in which there is a precisely specified goal state, and another where modifications are made as part of a more autonomous, undirected process.

The former describes models referred to as Supervised Learning, in which there is a feedback loop of some sort, offering a network-behavioral readout which can be used to guide changes to the synapses. This form of learning has also received a disproportionate amount of attention in the field since it offers practical uses as an optimization technique. However making such schemes work typically involves the creation of an external 'black box' process which coordinates this feedback pathway—often in strikingly non-biological ways.

The second form, in which functional changes in the network are not strategically directed but are an emergent property of some modification process operating without error feedback, is termed Unsupervised Learning. This naturally appeals to the biologist who is pleased by the fact that the objectionable black box may be removed altogether in favor of a more justifiable network property. And in fact such systems do very well within a certain class of problems—primarily involving self-organization or local signal processing. However, for goal-directed behaviors such approaches are rarely effective.

We will examine variations on both types of learning in the sections that follow, ultimately using the two in combination for the modulatory learning system described in later chapters. But first it is important to examine the network architecture in which either such scheme would function.

Simple Neural Networks · The Perceptron

One of the earliest attempts by computer scientists to build neurally-inspired information processing systems was the feedforward network known as the Perceptron (Rosenblatt, 1958; Minsky and Papert, 1969). This type of network is characterized by a simple architecture in which cells are grouped in layers with each receiving signals from the prior layer and sending its own projections on to the next. Neither descending nor lateral connectivity exists. As a result of this complete absence of recurrence, analysis of the network's behavior is actually a tractable proposition. In addition, the network's final 'response' to an input can be reached in a single pass through the layers rather than needing to wait for it to settle into an attractor state as one typically would with a recurrent network.

On the other hand, the simplicity of the system also raises obvious conflicts in terms of its generalizability to biological neural networks. However even its defining features—layered architecture and feedforward projections—are not entirely unnatural. The brain also possesses a layered structure, and though there are extensive recurrent connections, at least within sensory pathways it can be argued that the majority of processing occurs in a feedforward manner. So while somewhat oversimplified, useful insights might still be drawn from them.

The most basic form of the network only has two layers (see figure 1.1), the lower of which having its firing rates set as a function of an input stimulus pattern. As a result it is only the second layer which is driven by signals produced within the network itself. Accordingly it is the single set of synapses from input layer to output layer that is performing whatever remapping of the stimulus the network is capable of producing. An important implication of this is that the only types of problems that a simple, two-layer perceptron can solve are those in which the mapping to be learned is linearly separable. An observation that leads us to consider the types of problems these networks are capable of solving.

Simple Neural Networks · Tasks Performed

Due to the lack of either recurrence or non-stimulus-driven input, the output pattern of a perceptron (represented by the firing rates of the cells in the output layer) is guaranteed to be some direct transformation of the input signal. Thus all network tasks boil down to one form of association or another. The simplest of these, in cases in which there is an equal number of input- and output-layer neurons is autoassociation: the reproduction of the input pattern in the output layer cells. However, this is more a special case exhibiting signal propagation rather than information processing per se, thus we will concentrate on two other variations of the associative task.

The first is classification. Under this regime the responses of cells in the output layer can be considered as essentially digital readouts. The network is given an input pattern and its objective is to decide which of n arbitrary classes it belongs to. This class membership is then communicated by causing the nth output cell to have the highest firing rate within the layer. Since there is only one

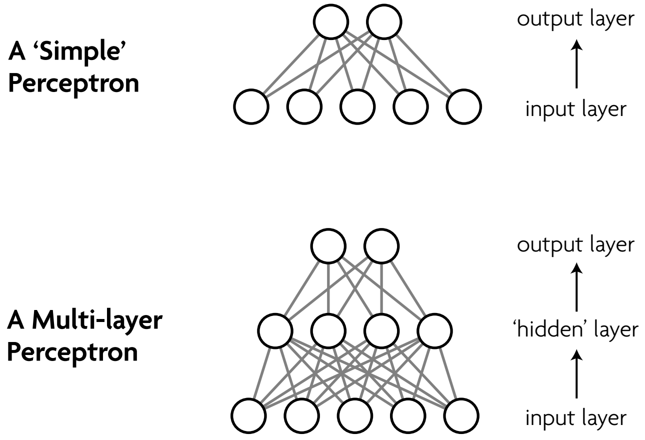

Figure 1.1: The 'perceptron' artificial neural network. In these models all connections are unidirectional with information flowing 'up' from the input layer to the output layer. In simple perceptrons there is only a single set of synapses capable of modification. Multi-layer models add additional cells, but, just as important, a second layer of synapses, allowing such models to solve more complicated learning tasks.

stage of modifiable synapses in the simple perceptron case, the changes made for the proper classification of one input necessarily interfere with the others, but this interference occurs in a systematic way. In fact it is equivalent to selecting a hyperplane through the m-dimensional space (given m input cells) in which all input cases to one side of the plane are given one classification, and all on the other side given a second classification (Amit, 1989; ăHertz, Krogh and Palmer, 1991). The rigidity of this distinction naturally limits the complexity of the judgments such a network can make. However all is not lost since the addition of one or more intermediate, or 'hidden', layers removes this restriction, allowing the network to learn essentially arbitrary remappings—given proper training.

This addition also opens the door to the more interesting second class of processing task. Whereas in the previous case output neurons were treated as glorified LED readouts on a control panel, an alternative exists in which their output firing rates are read as an analog, rate-coded signal rather than the classification-style population code. In this sort of task, the network reconfigures itself to cause the firing rate of the relevant output cell to be a function of the value encoded by the stimulus pattern.

It has been argued that this type of operation—Function Approximation—is a fundamental task performed by the brain (Poggio, 1990). If one subscribes to a pure 'grandmother cell' model in which discrete representations of objects, events, tasks, etc. are stored in dedicated cells or circuits, the result is a combinatorial explosion in which the number of cells required to store this information rapidly outstrips the actual supply of cells and synapses in the brain. The alternative proposed by Poggio is that rather than storing every possible variation on each piece of information, only one or a handful of canonical forms are stored, then the brain uses some additional mechanism to interpolate between them. In such a scenario performing function approximation efficiently becomes of paramount importance. Thus even this simplistic task, operating in our highly generalized network architecture can model operations important to biological systems.

This mechanism is, at its essence, one of composition. This is apparent in the previous example of mapping particular examples back to already-stored templates. However, in addition to creating generalizations it can also be used in the opposite direction—generating a unique output from a combination of stored instances. This scenario has been examined in the context of motor pattern production (Salinas, 2000). Here, one imagines that the network has stored a repertoire of sub-movements composing a basis set from which to build more complex movements. Again, the mechanism is one of interpolating between stored states, but to a rather different end. Thus function approximation allows for both classification—mapping an intermediate representation to the closest canonical copy in storage—as well as diversification—creating entirely new motor movements from a library of components.

Given the broad potential of this task we will return to it throughout this work as a metric on our approach to learning.

Feedback Systems · Supervised Learning

A learning rule is essentially a decision process; it takes in information of some sort, then, based on some transformation of that data, selects a course of action. Thus any network learning algorithm has two common qualities:

1. an input value which is a read-out of the current error on whatever task the network is trying to perform, and
2. an output which instructs the network to modify itself in some way in order to minimize this error.

The variations in the learning rules we will discuss are either differences in these inputs and outputs, or else in the transformation that maps one onto the other.

If one's goal is simply to make the network behave 'correctly' in a minimal amount of time, without worrying about the specifics of how this is accomplished, it is hard to do better than systems using Supervised Learning. In this approach, represented schematically in figure 1.2, the circuit implementing the learning rule is given a huge quantity of resources and, unsurprisingly, does a very good job adapting the network to perform the desired function. This can be seen in all three segments of the diagram. First, the circuit is essentially omniscient, receiving specific error feedback signals for each of the cells in the network it is responsible for. This abundance of connectivity is mirrored on the right side as well with independent modification signals being sent to the individual cells. Directly following from this is the fact that the computations performed (and degree of state which must be maintained) by the supervisor circuit itself are daunting.

This is particularly problematic as the network size scales upward. While it stands to reason that a relatively simple circuit could keep track of a handful of error values and send out a similar number of feedback signals, moving into the range of realistic networks would require it to monitor numbers more on the order of hundreds or thousands. Thus it is unsurprising that most traditional supervised learning implementations treat the circuit itself as a black box, given how much easier it is to offload this computational work on an equation rather than a network of neurons of its own. And in fact, to the extent that the early chapters of this work represent a traditional approach to supervised learning, we too will abstract away the implementation of this Supervisor's decision process—though we will return to the goal of providing a neural implementation in chapter 4.

However, in these early chapters we again borrow techniques from the ML/AI world. Since the initial approach was to merely verify that the Response Modulation mechanism discussed below could provide a neural mechanism for transmitting supervisory signals, we began by leaving the structure as similar to a typical artificial neural network framework as possible, using RM as a drop-in replacement for the least believable portion of the ML/AI supervision scheme. Thus we used a standard algorithm designed for making minimal changes to network connectivity strengths in order to minimize error, but in our case adapted it to optimize modulation states rather than synaptic weights.

This algorithm—the Widrow-Hoff or 'delta' rule—is an approach using stochastic gradient descent to cause network weights to relax to a point that minimizes task-dependent error. In order to do this, one must first define an error measure against which one can look for slight changes that could reduce the error. In more formal terms, by examining the derivative on this error surface for a given synaptic connection, then adjusting that connection strength proportion-

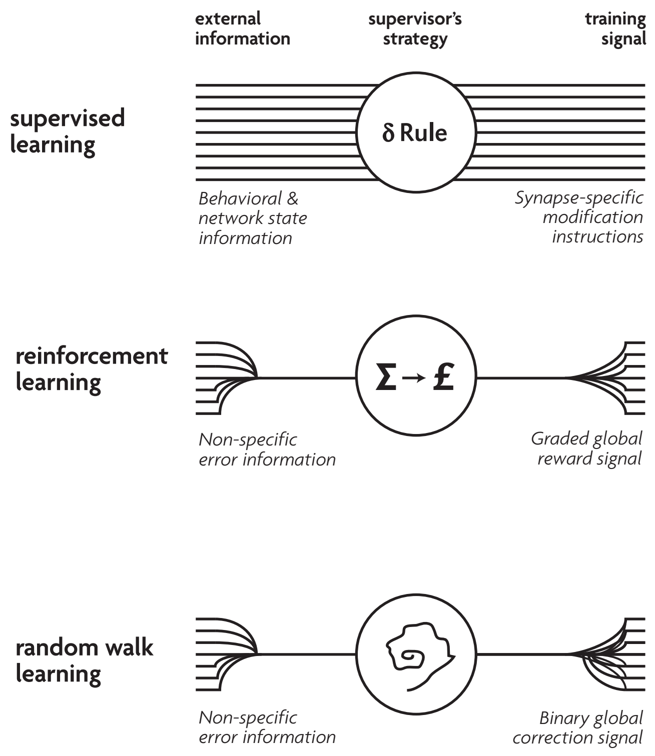
external information
supervisor's strategy
training signal
supervised learning
δ Rule
Behavioral & network state information
Synapse-specific modification instructions
reinforcement learning
Σ → £
Non-specific error information
Graded global reward signal
random walk learning
Non-specific error information
Binary global correction signal

Figure 1.2: Different approaches to network guidance. Three learning algorithms are depicted in terms of the degree of external information the require, the complexity of processing involved in the decision process, and the complexity of the output signal sent to the network being trained. In the top row is supervised learning, a scheme which yields excellent results but has accordingly high costs associated with it. It requires the ability to read out the state of the network at a very low level, with access to all of the firing rates and synaptic strengths. In addition. In terms of processing, it is presumed to have an idea of what the desired end state for the network is, and is capable of comparing the current configuration to this target. In addition, the modification signal it sends to the network is synapse specific, implying a staggering degree of micromanagement. A simpler strategy, reinforcement learning, receives less information from the network and has no direct access to the target state of the network, instead only getting an estimate of how well it is performing, but not specifically why. Accordingly, its output to the network is general as well, providing a global reward signal which is interpreted differentially based on local conditions within the network. Finally, random walk learning represents an additional simplification on reinforcement learning. Rather than sending a modification signal communicating a degree of reward, this approach instead sends a more binary, switch-type signal informing the network that the current strategy is no longer working and that a new random variation ought to be tried.

ally to that error derivative,

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \tag{1.1}$$

the system as a whole will gradually drift down the gradient to a point that is guaranteed to represent at least a local error minimum.

This is a fairly manageable computation in the context of the Simple Perceptrons discussed above given their single layer of synaptic connectivity. In this case if a particular cell is firing too strongly it is clear that the source of the error lies in the synapses between the errant cell and the input cells that project to it. However, in more complicated networks with additional layers this approach will clearly need to be amended—otherwise the only synaptic changes made would be to the final layer of synapses, rendering this more complex network functionally equivalent to the Simple Perceptron (since the additional layers that were added remain unmodified and thus unused).

One way to adapt the delta rule to multi-layer perceptrons is the widely used Backpropagation algorithm (Rumelhart & McClelland, 1986). To compensate for the fact that a given output cell's error may not lie with its own synapses but those which drive another neuron upstream of itself, this algorithm earns its name by propagating the final error back through the layers of the network. Thus, on the positive side, blame for error is shared among all the cells that contributed to it along the feedforward cascade. On the negative side, the informational and computational demands of this algorithm are even higher than for the delta rule. Now the Supervisor needs:

1. a way to read out the synaptic strength of every synapse in the network,
2. a mechanism to specifically modify each of these synapses, and
3. the computational power to perform the blame-assignment operation and to decide on the proper modifications for this huge number of synapses

Any one of these objections represents a serious attack on backpropagation's utility as a realistic model of learning in biological networks—and we address them in turn throughout the remainder of this work—but in the aggregate they are devastating. On the other hand, backprop is amazingly efficient in practice, thus it stands to reason that evolution might have stumbled upon a similarly optimal approach which might share some of its strategic character, if not its implementation. It is for this reason that we begin by compensating for the second objection (corresponding to the right column of figure 1.2) with our Response Modulation regime, and only move onto the other problems later. Luckily, in addition to providing the problem—in the form of the unrealizable backprop algorithm—the

ML/AI literature also provides the tools to remedy the difficulties in the other two columns of the schematic.

Feedback Systems · Reinforcement Learning

The first of these solutions for increasing the biological-ness of Supervised Learning comes by addressing the high information-content expectations of these learning rules. In order to account for this omniscience problem, an alternative scheme has been proposed in the form of Reinforcement Learning in which the Supervisor's ability to specifically read out from and direct the network is greatly reduced (Sutton and Barto, 1998). This in turn requires that more of the problem of learning be solved within the network itself rather than by the under-defined, black box Supervisor. This change can be seen in both the left and right columns of figure 1.2: in terms of input, the supervisor no longer receives a cell-by-cell error signal, but instead only a single error value representing how well the network is performing *in general* over some limited period in the recent past. This has the advantage of falling much more in line with the non-specificity of data that candidate areas in the brain for this sort of function would be able to receive.

However, it also limits the precision of the modification instructions such a Supervisor could offer to the network being trained. And in fact it may be better to think of this form of supervisory circuit as less of a Teacher—as in Supervised Learning—than a Critic. Rather than *showing* the network how to perform properly, it instead merely weighs in on how good of a job it is doing in its current state. All modification decisions are left up to some other mechanism in the network itself. But this 'drawback' actually represents another win for biological plausibility, for now the supervisor needn't send independent modification signals to each cell (or, worse, each synapse), but can instead pass on a global reward signal that is interpreted by all neurons based on their local conditions.

In practice this means that some source of variation must be encapsulated in the network itself. Previously we have relied upon directions from Teacher-style Supervisors to decide on a direction in which to modify synaptic weights, now that job falls back to the cells being modified. One could imagine a sophisticated scheme for using local information in combination with the reward signal to guide these modification, but in fact this is unnecessary, as we show in chapters 3 and 4. Instead, a remarkably basic algorithm inspired by some of nature's simplest creatures can take the hints provided by the Critic's reward signal and translate them into synaptic modifications in a biologically plausible, and surprisingly effective manner.

Feedback Systems · Random Walk Learning

This strategy, known as a guided random walk, represents one of the most computationally cheap methods imaginable for moving down a gradient. And while in this work, 'gradient' typically refers to an imaginary error gradient, in biological systems, the gradient in question is wholly corporeal in nature. The most vivid example of this sort of guidance can be seen in the foraging behavior of bacteria (Koshland, 1980).

Since they lack vision systems, bacteria have no way of spotting a food source from afar and setting a course toward it in order to feed. Instead, possessing only chemical sensors detecting the concentration of the desirable chemical in the current location, the bacterium must monitor changes in the concentration in order to keep traveling up the gradient, and presumably to the food source itself. Thus, the bacterium's task is on the whole quite similar to that of the ideal Critic-style supervisor: it has no external information about what a successful 'solution' would look like, but it *can* detect how well the current configuration satisfies, and with the addition of some sort of integration mechanism would be capable of determining whether the current direction was a positive one or a counter-productive one.

It is precisely this sort of strategy that one sees in bacteria. The general algorithm seems to be to swim in one direction while keeping track of relative changes in concentration. So long as the concentration is increasing, no course correction is necessary and the bacterium continues in this 'successful' direction. However, unless the initial course was headed directly for the food source, it will eventually overshoot and a decision will have to be made on a new direction to swim in.

To make this decision intelligently requires either a fair amount of computational sophistication—which is clearly not available to the bacteria who successfully perform this task to survive—or else a clever 'hack' in terms of behavior which can solve the problem while minimizing the amount of analysis required. As one would expect, the bacterium chooses this latter option, but counterintuitively its response to the need to make a strategic decision on which direction to turn in is instead to reverse its flagellum and go into a tumble, effectively selecting a new course at random.

This is not quite as stupid as it seems. Recall that whenever a decrease in the level of the target chemical is detected this tumble behavior is repeated. Thus whenever one of these decision points is reached, the bacterium will likely make a series of tumbles, the first few in 'bad' directions, each taking it a short distance further away from the food source before it is detected as a bad course and corrected again. This continues until a 'good' direction is finally arrived at randomly. In addition, because these course changes are random, the bacterium is unlikely to move a significant distance during one of these tumbling periods since on average the random steps will counteract one another.

Thus, in its way, the bacteria compute the same derivative of the gradient being optimized as the delta rule does. However while the delta rule requires differentiation, the random walk merely samples points a short distance away from the current position. As a result, the directions chosen are not guaranteed to be optimal courses, but, given enough time, the organism will eventually reach its target.

This approach can be adapted to neural network learning quite easily, particularly in a reinforcement learning context. The bacterium is trying to optimize its position relative to food—i.e., to find an ideal set of x, y, and z coordinates in three dimensional space. The network is similarly trying to find optimal values, albeit in a space of much higher dimensionality: the strengths of the synapses. If we were to map the bacterial strategy onto this problem directly (as we do in chapter 4), the equivalent procedure would be to choose a 'direction' in the space of synaptic weights and update the weight values by a small amount in the direction of this vector at each 'step'. Just as the bacteria would examine the concentration after each of these steps before deciding whether to tumble, the network would receive a reward signal from the Critic-style supervisor. In the case that this reward was lower than the one received prior to this step, the network would 'tumble' by choosing a new random vector in the weight space along which to take its next 'step'.

By borrowing this strategy used by extremely simple organisms to perform highly complicated navigational tasks, we can address the objections that might arise to algorithmic sophistication in the central column of the schematic in figure 1.2. Whereas the switch to reinforcement learning and response modulation reduced the unreasonable expectations on omniscient information availability and independent targeting of each cell or synapse within the network, the integration of a guided random walk strategy for controlling network modification removes the high-computational-power requirement that was keeping our Supervisor in the realm of black boxes. It has now been sufficiently simplified for a neural implementation to be buildable.

However there is a remaining problem with this scheme: it is unlikely that an algorithm that is somewhat pokey when applied to the bacterium's three-dimensional optimization problem will be able to perform well enough to manage a 100- or 100,000-dimensional synaptic weight optimization. That is, unless there is a way to reduce this dimensionality, it is unlikely that guided random walk represents a tenable strategy for network learning. Luckily, it would appear that the high-dimensionality of the weights space is actually something of an illusion. As a result there are mathematical tools which can expose the simpler underlying structure, reducing the problem to one on which even slow, simple random walk is fast enough to suffice.

Dimensional Reduction · Principal Component Analysis

High dimensionality is a problem inherent to neural networks, and is a difficult one to overcome even when not exacerbated by a simplistic guidance scheme such as random walk. This is particularly the case since what one would like to find are scalable learning rules that could function in both small circuits as well as larger network meshes. However, each additional neuron in the net adds at best one dimension to the problem (in the case of neuron-based supervision), and at worst $N-1$ dimensions if the network has all to all connections (in the case of synapse-based supervision).

It is difficult to imagine the brain dealing with this combinatorial explosion through a brute force approach of adding new neurites connecting to each of these locations and increasing the complexity of the control circuitry to hold these thousands of state variables in a single store as it micromanages each and every change. The alternative would be to allow the lower levels of the network to structure themselves in such a way as to allow more general guidance from supervisory circuits to affect them differentially while at the same time simplifying the task for the Supervisor.

Luckily the structure of known neural networks supports this latter approach, particularly in the sorts of sampling array-driven networks like those used for function approximation in this study. The relevant feature making them amenable to this sort of dimensional reduction is the lack of true independence in the responses of individual cells. In part this derives from overlap in the receptive fields of cells in the sensory sampling array (examined in more detail in chapter 4). But regardless of the cause, when cell activities are correlated, it is a waste of resources to devote neural resources to tracking them individually. Instead, there should be a way of supervising them in groups, modulating neighboring cells (whose responses will also be similar) in similar ways.

The trickier proposition is coming up with a way to detect these correlations and devise an innervation scheme that will take advantage of it. The standard tool in applied math for condensing a number of correlated elements into a lower number of dimensions while sacrificing as little information as possible is called Principal Component Analysis or PCA (Jollifee, 1986). The essence of the technique is the examination of the statistics of all the points in a high-dimensional cloud of data, then, by finding the eigenvectors of the correlation matrix, ranking them in terms of the amount of variance in the cloud they account for. Once this is done, many of the lower-valued eigenvectors can be discarded and the data can be represented in terms of this new set of orthogonal axes in the lower-dimensioned subspace they define.

This would be an ideal solution for many neural network learning tasks since it allows even a relatively simplistic search algorithm to function due to the degree of preprocessing. By shrinking the size of the haystack, finding the needle becomes achievable—even while blindfolded. In our case, PCA can be used in combination with a random walk-inspired guidance scheme since it allows for the hundreds or thousands of synaptic weight dimensions to be reduced to a handful, at which point even the slow, irregular drift of this simple algorithm would reach the target point in a reasonable amount of time.

The only problem with this magical solution is that the mathematical processing required to perform is is formidable, requiring a significant amount of linear algebraic manipulation well in excess of what could be easily computed by a single neuron. However this is only true if the solution to the problem is dealt with serially. Luckily it has been demonstrated that through slight modifications to standard Hebbian plasticity rules, this very computation takes place (Oja, 1989; Sanger, 1989). However, rather than the results coming in the form of a set of firing rates they are stored in the set of synaptic strengths connecting to a target neuron. Once this is accomplished, the only remaining complication is extracting these principal component profiles stored in the weights and using them to assemble a feedback pathway which targets cells in the network according to the patterns they dictate. In chapter 4 we explore a new technique which appears to do just this.

Despite the lack of a cohesive, biologically plausible learning model having been put forth by the ML/AI community, this review should illustrate that all the pieces are present and that it is merely a matter of connecting them in a proper fashion. However, while the ML/AI tools have brought us quite close to the goal of a believable learning mechanism, there are two remaining issues that they cannot address and for which we must turn to biological data:

1. There is no known mechanism that could perform the direct synaptic modification demanded by backpropagation and similar algorithms on a sufficiently wide scale.

2. The degree of synapse-level micromanagement foisted upon the high-level supervisory circuitry in these schemes seems biologically unlikely.

The latter issue can be addressed through the random walk guidance (coupled with PCA-based preprocessing) hinted at in the prior section, but just as important is an explanation of how these systematic, error-driven modification signals can be transmitted to the network and put into place synaptically.

To examine what biological mechanisms one would need to use in order to implement these ML/AI strategies in a realistic way—and to try to address the two issues raised above—let us now survey the relevant experimental data. In the end, none of the general strategies have any value in a biological context unless they operate similarly when constrained by the realities of *real* neural networks, and not just artificial ones.

CONSTRAINTS FROM EXPERIMENTAL FINDINGS

While the ML/AI research discussed above provides a compelling—albeit hypothetical—picture of the higher level structure and learning strategies of a neural information processing system, it is decidedly lacking in explanations that reach the biophysical level of individual neurons and synapses. However, this is the level at which experimental neuroscience has had some of its greatest successes in examining questions of learning and adaptation.

Synaptic 'Learning'

The brain is a highly varied system with myriad exotic schemes for governing the strengthening and weakening of synapses. However at the phenomenological level—at least among cortical cells—much of the activity-dependent modification occurring can be described in terms of a process laid out by Donald Hebb over 50 years ago (Hebb, 1949). His hypothesis conservatively proposed that plasticity was necessarily a local computation, with all the information needed to decide on a change in synaptic strength being available directly at that synapse. While hardly a controversial claim, it does create problems of complicated wiring (or worse) for algorithms such as backpropagation.

In its initial form, the archetypal Hebbian Synapse's strength changed to reflect the degree of coincident activity occurring between the presynaptic and postsynaptic cells, with the idea that this correlated activity was evidence of a functional similarity between those two cells that ought to be more permanently stored in the coupling strength between them. The exact mechanism of this change was left undefined, but implicit in the idea is that there would be some form of coincidence detection operating on one or both ends of the synapse, and that the arrival of correlated activity would initiate a synaptic strengthening process.

In more recent years the details of this detection and modification mechanism have been further fleshed out, with intracellular calcium, backpropagating action potentials, and NMDA receptors offering a particularly nice example of coincidence detection. This research is encouraging in that it offers a biologically reasonable defense for the use of Hebbian modification on the more abstract level at which we will employ it.

Reward & Supervision

A common objection by biologists to the scheme of supervised learning discussed in the previous section is that it is merely pushing the problem of learning off to an all-knowing Homunculous who makes all the decisions guiding the process. Given that the eradication of this figure virtually defines the mission of the cognitive neurosciences, it is unsurprising that these ideas encounter resistance.

Yet there is ample evidence of supervisory input in a number of circuits within the brain, so the idea of top-down control will have to be merely amended rather than discarded altogether. However, in practice the form of this top-down supervision falls much more in the mold of the reinforcement learning-style Critic rather than the Teacher of supervised learning. Examples of this can be seen in the vertebrate dopaminergic system in which neurons of the substantia nigra project to the striatum, where they are thought to encode a general reward signal (Montague et al., 1996; Schultz, 2002). In insects, a dopamine precursor, octopamine, serves a similar function, sending a feedback signal to the mushroom bodies (Menzel, 2001).

A common feature of these experimentally observed supervisory pathways is that the signal they deliver is essentially global, and does not individually affect the synapses of the network onto which the projections terminate. This constitutes a near-fatal challenge to the backpropagation hypothesis since it expects the high-level supervisor to tailor its reward/error signal to each synapse in the subordinate network.

In addition backprop requires a mechanism that is clearly not present here by which the supervisory network can strengthen and weaken synapses directly and independently of one another. For this sort of arrangement, one would expect to see a peculiar type of three-element synapse in which the axon of the supervisory cell terminates directly onto synapse being trained. This sort of arrangement has been observed in vertebrate and other nervous systems, however such cases are occur too rarely to explain learning.

In sum, the general characteristics of observed supervisory circuits suggest two constraints upon models that could be considered biologically plausible. First, a reinforcement learning-style approach should characterize the feedback pathway in that the supervisory signal ought to be a general one, not making fine distinctions between its targets within the network being trained. This is consistent with the global quality of dopamine signaling, and is intuitively appealing since it does not demand that the supervisor be both high-level enough to read out task dependent error at the behavioral level and low-level enough to micromanage each synapse within the network responsible for eliciting that behavior.

The second constraint these data put on any effort at modeling these processes is that the actual synaptic modification mechanism must be a locally-driven one, since there is no evidence of top-down control of synaptic strength. Thus the supervisor must be causing dynamic changes to the network that ultimately result in synaptic restructuring rather than plasticity being a single-step process imposed from above.

Dynamic Response Modulation

While a locally-interpreted dopamine signal is one candidate mechanism for communication between supervisory areas and subsidiary networks, the steps leading from dopamine transduction to a change in neural behavior are less well defined. Thus, while there is evidence that it is in fact being used as an error/reward signal, the more interesting question of how that signal is 'obeyed' in order to improve performance is left unaddressed. However, there is a recently discovered mechanism which may fill this gap, providing both a biologically reasonable feedback pathway and a more transparent mapping from command to action.

This mechanism, referred to in these pages as *response modulation* takes advantage of the widely observed phenomenon of balanced synaptic inputs, whereby cells typically receive nearly identical quantities of fast excitatory and inhibitory current. Since these currents are balanced, the net current input is zero and the cell's membrane voltage neither de- nor hyperpolarizes. Thus, at first blush, one would suspect that so long as this balance is conserved, the overall size of the positive and negative currents will have no bearing on the postsynaptic neuron's probability of firing an action potential. But, in fact, just such a relationship exists.

The explanation for this counterintuitive result is that while the net amount of current remains unchanged so long as the excitatory and inhibitory inputs are balanced, the total number of synaptic events *does* change. As a result, even if every glutamate binding event is paired with a hyperpolarizing, GABA binding, the raw number of these events will increase as the magnitude of the balanced currents increases. Most of the time this is an invisible change, since it is not directly reflected in the cell's steady state membrane potential.

Where it does have an effect though is in the fine structure of the voltage trace. Every synaptic current gives this potential an imperceptibly small kick in one direction or the other. The more of these events that occur during a period of time, the more jagged the subthreshold voltage trace will be. Thus, as one increases the balanced currents, the average postsynaptic voltage will remain constant, however the amount of variance around that mean value will increase as a function of this resulting 'synaptic noise'.

Again, this would seem to be of negligible importance in influencing cell behavior since we've already established that balanced currents will not drive the cell to spike. However, this phenomenon can have a profound effect in shaping the cell's response to *other* synaptic inputs. Thus it could represent an important modulatory pathway, affecting a neuron's intrinsic excitability. And, in fact, recent work (Doiron et al, 2001; Chance et al, 2002; Prescott & De Koninck, 2003; Mitchell & Silver, 2003) has demonstrated that this is the case. But it is particularly noticeable under two regimes.

The more important of these is the condition in which the cell's membrane potential is very near to—but just below—its firing threshold. In this case, an increase in the variance of its voltage trace due to synaptic noise has the potential to momentarily knock the voltage above threshold, resulting in an action potential even in the absence of what would otherwise be 'sufficient' excitatory input to elicit a spike.

As represented pictorially in figure 1.3, this noise-driven modulation has the effect of lowering the initial firing threshold of the cell and generally smearing the input/output curve to the left, decreasing its slope in the process. And it is the slope change in this second, midrange regime that marks this form of modulation (which actually influences the gain of the i/o mapping) as functionally distinct from what could be achieved through a simple excitatory bias current. In the bias case the added excitation would also cause the cell to fire in response to smaller than normal amounts of synaptic current, however the slope of the i/o function would remain unchanged. Instead, this type of modulatory input would merely shift the entire curve to the left; its shape otherwise unchanged.

Thus there are in fact two types of modulation that can be achieved merely through changes in ordinary, ionotropic conductances. One, achievable by sending either an excess of excitation or inhibition resulting in a *shifting* of the i/o curve to the left or the right, corresponds to a parallel change in the firing threshold and saturation point. The second results from keeping excitation and inhibition balanced, but scaling them in tandem. Thus by boosting the absolute magnitude of both currents, the *gain* of the i/o function can be be modulated, dropping the firing threshold, flattening the linear phase, and slightly raising the saturation point. In addition, since current magnitude and balance are orthogonal to one another, both types of modulation could be used simultaneously.

Synthesis

Thus far we have focussed on sketching out the various components of the problem, but have shied away from laying out the connections between them that could add up to a believable supervision scheme for realistic neural networks.

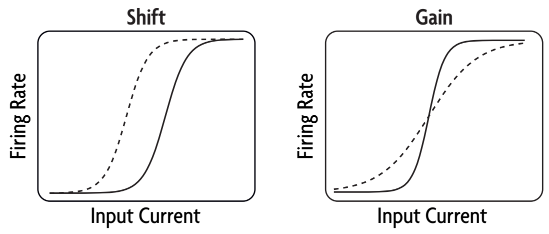

Figure 1.3: Noise-based modulation of neural excitability. Model neurons typically respond to input current in a sigmoidal fashion, with a subthreshold region, in which they don't fire, a linear region, in which increases in current cause increases in firing rate, and a saturation point, beyond which the firing rate will no longer increase, regardless of the input current. The modulation we will discuss in subsequent chapters comes in two forms. The first, depicted in the left panel corresponds to the addition of a bias current. In this case, a constant excitatory input has been added, having the effect of shifting the entire curve to the left, lowering the amount of additional current required to reach threshold, but leaving the slope unchanged. The second type of modulation, seen in the right panel, is the result of adding balanced excitatory and inhibitory currents. This type of modulation also lowers the threshold of the cell, but more importantly affects the slope of the linear region, allowing the response function to be smeared out or sharpened as necessary.

However, all the elements for such a synthesis are present, and the remainder of this thesis will explore how the pieces fit together.

We begin our examination of the use of gain and shift modulation as a mechanism for guiding learning by simply dropping it into a highly artificial, ML/AI-style learning environment. At the outset the goal is merely to show that, when used in an optimal manner, this biologically realistic modulatory mechanism could change a network's behavior in such a way as to better perform a task. This is demonstrated in chapter 2 in which we use standard optimization techniques to arrive at a mixture of modulatory inputs that, if provided by a hypothetical supervisor circuit, would minimize the network's error in an externally-imposed function approximation task.

However even the success of this leaves a pair of open questions. First, there is the problem that network behavior based on this sort of modulatory drive is entirely the result of decisions made by the supervisor. Thus we have put aside for the moment the more difficult question of learning itself by solving it in an algorithmic black box rather than in a biologically believable manner.

The second issue derives directly from this overreliance on the supervisor. Since the learned 'behavior' is dependent upon the modulatory inputs, removing our artificial supervisor's influence would render the network incapable of performing the task. Since it seems unlikely that high-level supervisory circuits

are involved every time a learned behavior must be performed, it would be preferable to be able to somehow make this learning intrinsic to the network itself.

In chapter 3 we attempt to address this transfer of learning issue. Though we are again using an unrealistic, optimizing supervisor, we demonstrate that through the addition of a local, Hebbian rule governing plasticity within the network, the pattern of externally imposed activity can gradually be transferred to the synapses. Thus, over time, the behavior becomes locally produced, allowing the supervisory modulation to relax, thereby freeing the supervisor to attend to other modifications in that (or another) network.

Finally, in chapter 4, we address the issue of the unrealistic supervisor itself. As used in the earlier investigations, it has the dual drawbacks of having essentially omniscient access to network activity state, as well as micromanaging the modulatory input sent to the network on a cell-by-cell basis. Since this places biologically unrealistic expectations in terms of processing power and available information on the supervisor, we instead explore a reinforcement learning-inspired scheme whereby the supervisor only has a general idea of the network's success and cannot locally assign blame. In the place of the implausibly sophisticated gradient descent algorithm for fine tuning neuronal modulation, we instead turn to an easily realizable random walk strategy for exploring the modulation space.

Though these modifications to the supervisor increase its neural believability, they naturally decrease its efficiency, primarily due to the high dimensionality of the search space and the simplicity of the search algorithm. However we also demonstrate that these failings can be mitigated through simple changes to the plasticity rules governing connections between the network and the supervisor. By making use of a modified form of Hebbian learning, the innervation pattern from the supervisor to the network changes to reflect the principal components of the correlations within the network. As a result, the supervisor is able to deal with the handful of most task-relevant dimensions in its random walk rather than being directly dependent upon the number of cells in the network.

In summary, we propose a biophysically plausible mechanism by which a neurally-derived supervisor could influence a subsidiary network in order to perform a task. In addition we demonstrate that through the addition of non-controversial, local plasticity rules, this learning can be made permanent.

CHAPTER TWO

Controlling Network Activity via Response Modulation

Introduction

Neural networks that perform a specific task must be developed through a learning procedure which reconfigures the network in a way that maps patterns of input to a desired output. A standard mechanism for inducing such a correct mapping is the use of supervision. The supervisor can be thought of as a neural circuit that monitors the network's success in performing its task, and, by adjusting the properties of the network, acts to minimize the difference between the desired and actual output.

Traditionally, the supervisor receives an error signal and adjusts the network by directly guiding synaptic plasticity. This approach has proven widely successful in training networks to perform a variety of tasks (Widrow & Stearns, 1985; Chauvin & Rumelhart, 1995; Hertz et al., 1991; Dayan & Abbott, 2001). However, there is little anatomical evidence that such a scheme could work under a biological implementation. In particular, this direct synaptic modification model presumes the existence of a large number of backprojections from the supervisor terminating on the synapses in the network. While massive feedback projections are characteristic of biological nervous systems, the proposed three-element synapses—though they do exists—are too rare for this mechanism to explain network learning on this scale.

Instead, we propose a supervision scheme in which adjustments are made not to synapses, but to the neurons themselves. This connectivity pattern would be consistent with known neuroanatomy, consisting of feedback connections from the supervisor to the neurons within the network. Additionally, these connections could be ordinary excitatory and inhibitory inputs (Chance et al., 2002), allowing for both the speed and input specificity necessary for rapid learning.

The question we address here is whether such a scheme can work. We first survey the range of network functions that can be learned based on this type of supervision. Subsequently, we examine the effects of the supervisor's modulatory input, and discuss the problem of learning in the context of the search space within which the supervisor operates.

Methods

We apply our proposed supervision scheme to the problem of function approximation, a classical neural network task with similarities to those performed by biological circuits (Poggio, 1990). The objective is to elicit a network output that is a specified function of a single stimulus variable, θ. A traditional supervision scheme would bring this about by modifying synaptic weights according to a synaptic learning scheme such as the delta rule (Widrow & Hoff, 1960), in which errors in the output approximation are used to strengthen or weaken connections based on their relative contributions to those errors. Here, a similar approach is used, but rather than adjusting synaptic weights, the supervisor modulates the response properties of individual network neurons.

Our model network consists of a two-layer feedforward architecture containing purely excitatory connections, with N input neurons projecting to a single output neuron (see figure 4.1). The input units are driven by currents that are Gaussian functions of the difference between a stimulus, θ, and a preferred stimulus θ_i. Preferred stimulus values are uniformly distributed across the stimulus space for the different input units.

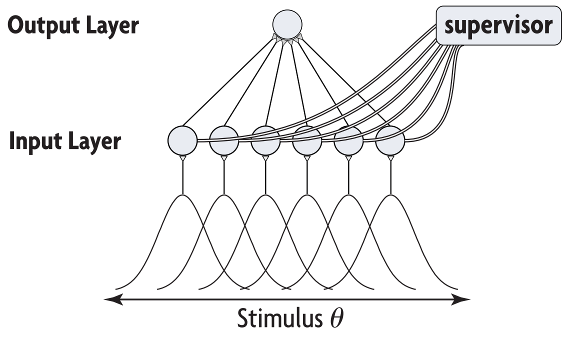

Figure 2.1: Network architecture. 460 input neurons receiving stimulus-tuned input currents project to a single output unit. An external supervisor is connected to all input units and modulates their response properties independently.

Each input unit's firing rate is calculated by passing its stimulus-tuned input

current through a sigmoidal transfer function,

$$r_i(\theta) = \frac{1}{1 + \exp\left(-g_i(I_i(\theta) - s_i)\right)} . \tag{2.1}$$

The parameter s_i, which we call the shift, controls the value of I_i at which r_i reaches half its maximal value. The parameter g_i, which we call the gain, determines the slope of the firing rate versus input curve at this point. These are set to the initial values $s_i = 1.0$ and $g_i = 3.0$ for all units, but are then changed by the supervisor.

The network's output consists of the firing rate of the single unit in the top layer of figure 4.1. This rate is also determined by the sigmoidal function in equation 3.1, but the input factor is instead a sum of the firing rates of the input layer units,

$$R(\theta) = \frac{1}{1 + \exp\left(-g_{out}\left(\sum_{i=1}^{N} r_i(\theta) - s_{out}\right)\right)} . \tag{2.2}$$

The goal of learning for this network is to match the output firing rate, $R(\theta)$, to an arbitrary set of stimulus-dependent target functions, $F(\theta)$, as closely as possible.

NEURONAL RESPONSE MODULATION

Instead of following the classical approach and having supervision occur through changes in the synaptic weights, our form of supervision involves changes in the shift and gain parameters (s_i and g_i) to each of the individual input cells, thereby controlling their response properties. By contrast, all weights are kept fixed at 1.0 throughout the simulation.

Performance error is calculated by comparing the output firing rate to the target function for each stimulus value. The supervisor then uses a stochastic gradient descent algorithm to modify the shifts and gains of the input units in order to reduce the error,

$$E(\theta) = \frac{1}{2}\left(R(\theta) - F(\theta)\right)^2 . \tag{2.3}$$

For each stimulus presentation, a random θ value is chosen in the range from 0 to 2π. The rates of the input and output cells are then computed and shifts and gains for the input cells are modified such that

$$s_i \rightarrow s_i - \epsilon \frac{\partial E(\theta)}{\partial s_i} \quad \text{and} \quad g_i \rightarrow g_i - \epsilon \frac{\partial E(\theta)}{\partial g_i} , \tag{2.4}$$

where ϵ is a small rate factor which constrains the response modulation. In our simulations, ϵ was fixed at 0.2.

Results

RANGE OF PERFORMANCE

We begin our analysis by showing that function approximation can be achieved in the absence of synaptic plasticity solely through changes to the input cells' response properties. Using our supervision regime, the network can approximate a wide variety of target functions, as depicted in figure 2.2. Distribution of shift and gain values for the modulated cells are plotted in the left column, and the corresponding target function and network approximation in the right column. In the center are the modulated firing rates of the input layer cells with respect to the stimulus.

In general, the network can approximate any smooth, continuous target function it is given. However, since the modulated firing rates of the input layer cells represent a basis set for the output approximation, problems arise when these response profiles cannot capture important features of the target function. This limitation can be seen in figure 2.2c where rapid changes in the target function cannot be reproduced by the network due to the Gaussian shape of the input unit response tunings. It is worth noting that in its attempts to deal with the discontinuity, the supervisor has increased the gain of the cells whose response boundaries border the vertical portion of the target function in an attempt to match that contour.

SHIFT VS. GAIN

Following the observation that gain modulation was used to mitigate the approximation failure in the case of the square-wave target function, we examined the relative uses of shift and gain in the learning process. Figure 2.3 illustrates the result of learning by modulating shift alone, gain alone, or both in tandem. Figure 2.3a considers a low-frequency cosine target function, while figure 2.3b considers a high-frequency target.

In the low-frequency case, learning succeeds regardless of which parameters the supervisor was free to modulate. The top row shows the optimal case in which the supervisor relies approximately evenly on shift and gain, with shift contributing to the overall level of activity, and gain controlling the flatness of the response in the peak and trough and its steepness along the sloped portions. This reliance upon shift for setting the overall vertical level can be seen indirectly in the second row of figure 2.3a, corresponding to the gain-alone case. Here the

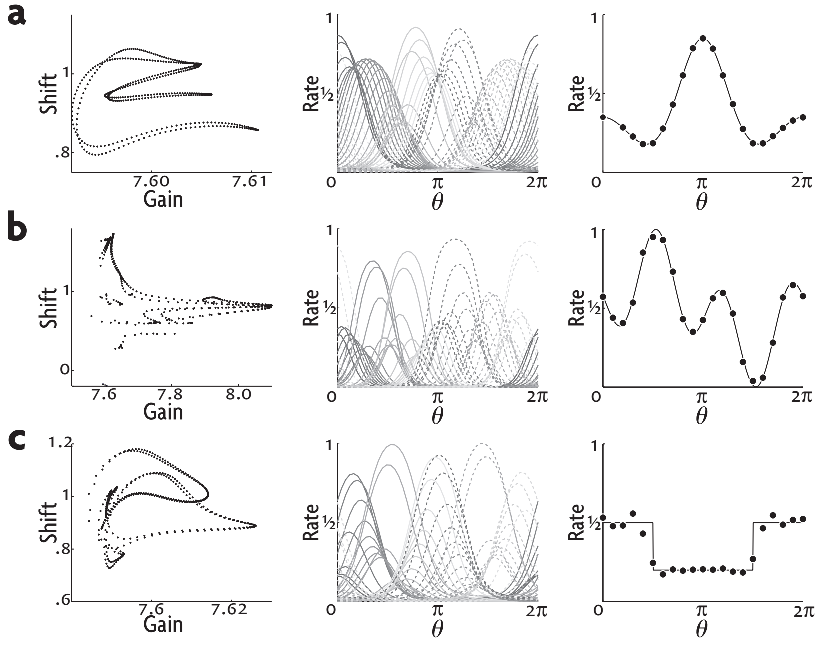

Figure 2.2: Examples of learning via response modulation for a variety of target functions. Plots in the left column depict the modulation state of the network, with each dot corresponding to the final shift and gain values for each input unit. The center column shows the responses of a sampling of input units to a range of stimulus values. The right column plots the network's approximation (dots) of the target function (line) for a selection of stimuli.

shifts remain fixed, but a similar effect can be emulated by flattening the gain profiles of a few inputs which will in turn contribute a stimulus-independent baseline shift.

In comparing the target approximation in the right column of figure 2.3a, the results from learning with gain alone are virtually unchanged as compared to the shift-alone and shift-plus-gain conditions. However, in the high-frequency case of figure 2.3b the differences between shift and gain become stark.

Again the top column of figure 2.3b shows that the function is readily matched by the supervisor when both shift and gain are adjusted. But whereas in the low-frequency case, learning was successful with only shift modulation, here it fails miserably. This failure is due to the breadth of the Gaussian inputs being wider than the half-period of the target function. Thus in order to properly approximate the height of the peaks, inputs must be shifted up, but in the process this

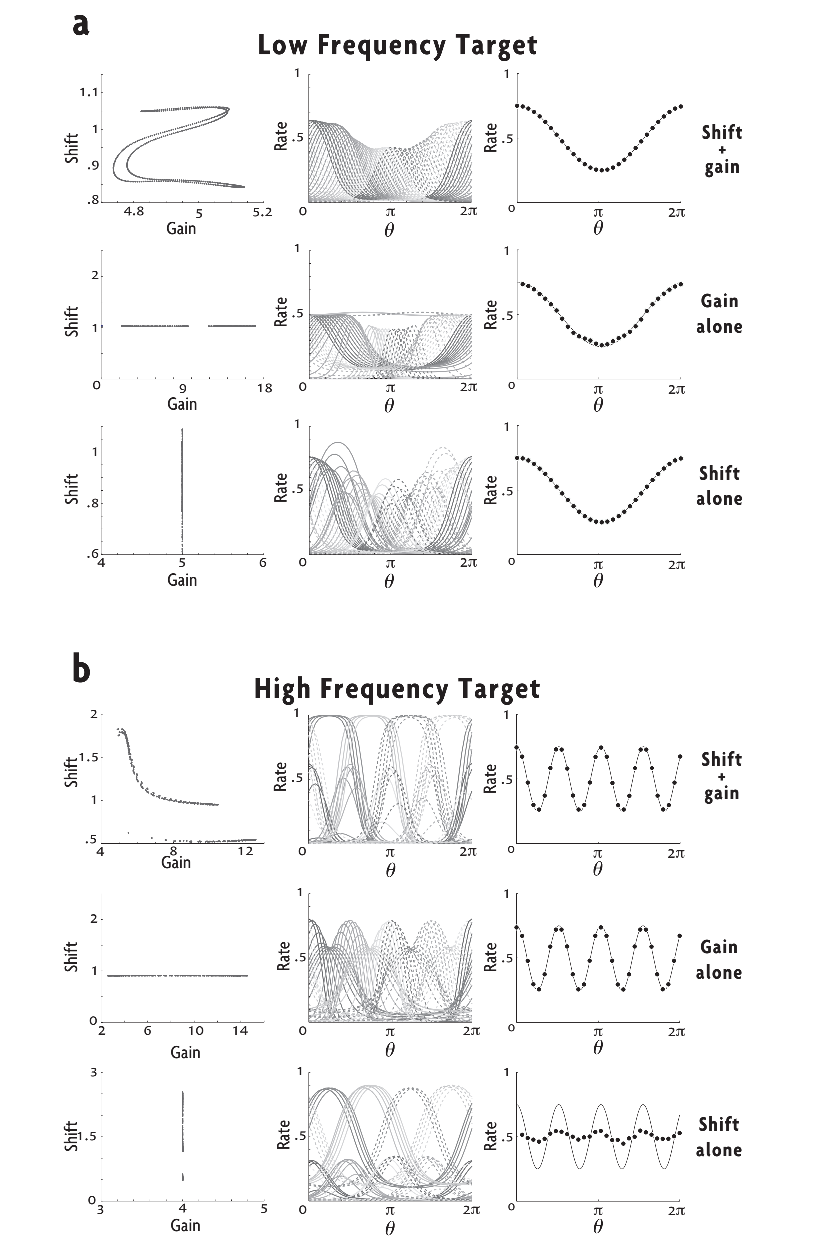

Figure 2.3: Relative contributions to learning of shift- and gain-based modulation under two stimulus regimes. As with figure 2.2, the columns depict (from left to right), the network modulation state, modulated firing rates of input cells, and the network's final approximation of the target function.

causes the depth of the troughs to be overestimated as the Gaussian tails spill off to the sides. As a result, the ultimate approximation splits the difference between peaks and troughs by lying along the target function's mean value, with only small deflections toward the extremes.

It is clear that to successfully approximate this function, the breadth of the response profiles must be narrowed, and this is precisely the result achieved by learning in the gain-alone condition (middle row of figure 2.3b). In this case the network learns successfully even in the absence of shift modulation by boosting the gain of inputs with preferred stimuli near the peaks, and modestly decreasing those preferring the troughs.

UNIQUENESS

Learning is made easier in these networks by the fact that the set of shift and gain variables that leads to successful function approximation is not, in general, unique. An illustration that this is indeed the case can be seen in figure 2.4, in which the final network modulation states of two separate runs are plotted. In both cases the target function was the same, however in the first run the input cells were given initial values of $s_i = 0.91$ and $g_i = 5.0$, and in the second run $s_i = 1.14$ and $g_i = 8.4$. In both cases the supervisor was able to approximate the target perfectly, yet none of the modulation states in the two runs is the same. This suggests that the problem, from the supervisor's perspective, may be simpler than one would guess. Perhaps it is the overall balance of shifts and gains that is significant, and not simply finding a single, globally optimal solution.

INTERPOLATION

It is important to remember that the supervisor's control of the network corresponds to a pattern of excitatory and inhibitory input that it sends to the network. As a result, the same network could conceivably be *switched* (Lukashin et al., 1994) between different target functions simply by the supervisor changing this input pattern. This raises the possibility of the supervisor building up a repertoire of learned states over the course of learning, and then applying those as appropriate to the task at hand. It would be particularly useful if there were some consistent mapping between related modulation states and related target functions.

In fact this relationship seems to exist. Figure 2.5 shows the result of interpolating between learned states to yield an intermediate output. Figures 2.5a and 2.5b show the result of learning two sinusoids phase shifted by $\pi/2$. For figure 2.5c, no learning occurred. Instead, a set of shifts and gains was constructed by taking the vector average of the sets in the previous, learned trials. The resulting

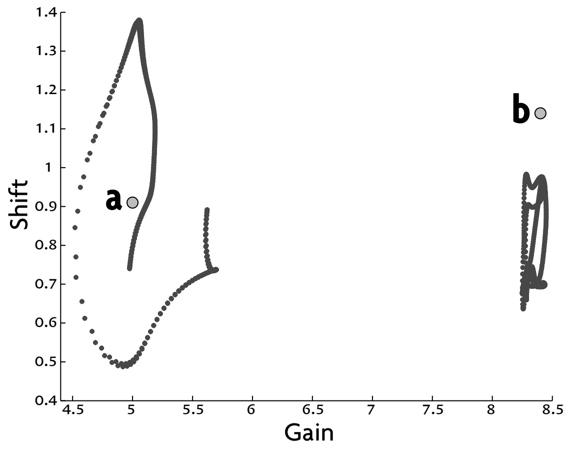

Figure 2.4: Function approximation solutions are not unique. The two clouds depict the final shift and gain states of the input units after learning from two different starting conditions. The cloud on the left corresponds to a starting position of (a), and the cloud on the right to starting position of (b). The target function in both cases was identical.

distribution produces an output quite close to the midpoint (in phase) between the learned functions.

This suggests an additional simplification of the supervisor's task since every new target function need not be learned from scratch. Instead, by consulting a library of previously learned approximations, the supervisor can start with a similar pattern and fine-tune the approximation from there. This result also places a limit on what could otherwise be unsustainable growth in the size of the library over time. Since similar functions yield similar network modulation patterns, it is not necessary to exhaustively store every single pattern experienced. Instead, a pared-down set of characteristic functions can be maintained, each as a basis for an entire group of targets rather than just a single one.

Conclusions

Though we have demonstrated that through its modulatory input the supervisor can put the network into an activity state suitable for the task, it should be noted that this does not result in a permanent change to the network itself. In the absence of modulation the network's response to stimuli will be identical to its response before 'learning' took place. For permanent changes that do not depend on the action of the supervisor, a form of synaptic plasticity is necessary.

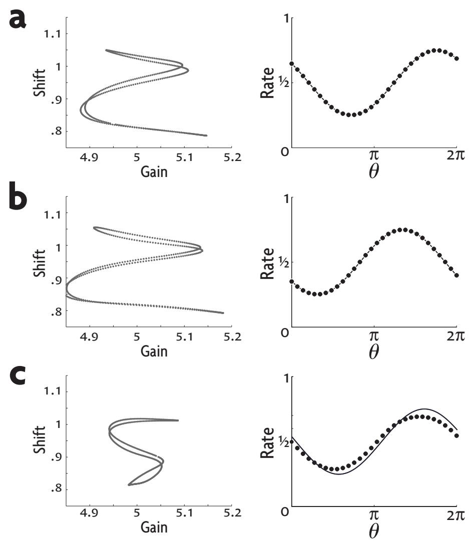

Figure 2.5: Interpolating between learned modulation states yields an interpolated output function. After learning to approximate two sinusoids shifted in phase (a and b), the resulting modulation states were averaged (c, left panel), resulting in an output function corresponding to a sinusoid with an intermediate phase (c, right panel).

In the next chapter we examine similar networks in which an unsupervised, Hebbian learning rule governing plasticity works in concert with supervised response modulation. This results in a pattern of learning similar to that discussed here, with the exception that learning is gradually transferred from the modulation pattern to the synaptic weights. Once this transfer is complete, the supervisory modulation may be removed altogether without affecting the network's performance.

CHAPTER THREE

Guiding Hebbian Learning with Response Modulation

Introduction

Correlation-based, Hebbian mechanisms of synaptic plasticity have been used with considerable success to explain the spontaneous development of selectivity and sensory maps in neural circuits (see Miller, 1996). However, when such plasticity mechanisms are applied to the development of networks that perform specific functions, rather than simply represent input data, a problem arises. To guide correlation-based synaptic plasticity, the activity of a 'naïve' neural circuit must be correlated in a manner similar to that of the final, functioning circuit. But such correlations usually arise only after the synapses of the circuit have been appropriately adjusted. The consequence of this is a chicken-and-egg problem: which comes first, correlations or synaptic modifications?

The traditional answer to this question is that synaptic modifications come first, guided by a supervisor. The supervisor is a hypothetical neural circuit that assesses network performance, computes an error signal, and uses it to direct synaptic plasticity within the network. Such schemes work extremely well for many tasks (Widrow and Stern, 1985; Chauvin and Rumelhart, 1995; Hertz et al., 1991; Dayan and Abbott, 2001) making them attractive models for learning in biological systems. However, for biological applications, it is important to identify the pathways through which the supervisory circuit controls synaptic plasticity. In some cases, such as climbing fiber input to cerebellar Purkinje cells, such a mechanism appears to be in place. In other systems, such as cerebral cortex, an appeal must be made to some form of modulatory (perhaps dopaminergic, see Schultz et al., 1997) control of synaptic plasticity that is largely conjectural. Furthermore, modulatory pathways tend to be slow and nonlocal, making them poorly suited for the rapid, precise control of synaptic plasticity needed during task learning. These considerations lead us to explore the possibility that supervision of synaptic plasticity takes place indirectly rather than directly.

The scheme we study corresponds to correlations coming first. In other words, the supervisor modulates neuronal excitability in order to introduce correlations into network activity. These correlations then generate the synaptic plasticity needed to learn a task through Hebbian synaptic modifications that are not themselves subject to direct supervision. We are interested in supervision through response modulation because it is easy to see how this scheme could be realized in cortical circuitry. The response modulations that we consider can be generated through standard excitatory and inhibitory synaptic input. Therefore, in this scheme, the supervisory circuit can guide learning through the feedback projection pathways that are characteristic of cortical circuitry, and no appeal must be made to as-yet-undiscovered forms of modulation. It is important to realize that we are not proposing this scheme as an algorithmic improvement. Indeed, such indirect supervision of synaptic plasticity has disadvantages, and an important element of our study is to determine how detrimental these are.

In summary, we consider a network in which synaptic plasticity is purely Hebbian, a form typically used in unsupervised learning applications. We ask whether it is possible to implement supervised learning in such a network solely by communicating error signals to the network along conventional excitatory and inhibitory feedback pathways that modulate neuronal responsiveness but do not directly affect synaptic plasticity. Such a scheme is not optimal, so its virtues are not efficiency or elegance. Rather, we take this minimalist approach so that we can determine whether these well-established elements of cortical circuitry provide a sufficient basis for implementing supervised learning.

Response Modulation and Synaptic Plasticity

Neural networks used for supervised learning consist of units with nonlinear response functions connected together through interactions characterized by synaptic weights. The response r_i of network unit i to an input I_i is typically determined by a sigmoidal function,

$$r_i = \frac{1}{1 + \exp\left(-g_i(I_i - s_i)\right)} . \tag{3.1}$$

In biophysical terms, this can be thought of as the normalized firing rate generated by an input current I_i. The parameter s_i, which we call the shift, controls the value of I_i at which r_i reaches ½ its maximal value, while g_i, which we call the gain, determines the slope of the firing rate versus input curve at this point. Input currents are typically computed by multiplying presynaptic responses by synaptic weight factors and summing over all inputs.

The role of the supervisor is to compute an error by comparing actual and desired network output, and to use this error to direct the modification of network parameters such that network performance improves. Conventionally, the major targets of this process are the synaptic weights. For example, weights can be modified to produce a stochastic gradient descent of the error function. We deviate from this procedure by employing a standard Hebbian synaptic modification rule that is not directly affected by the supervisor. At each stimulus presentation, the synaptic weight connecting unit i with response r_i to unit a with response R_a, w_{ai}, is augmented by a term proportional to the product of the pre- and postsynaptic activities,

$$w_{ai} \to w_{ai} + \epsilon_w R_a r_i , \tag{3.2}$$

where the parameter ϵ_w controls the learning rate. In addition, to prevent the runaway excitation that results from this positive-feedback rule, divisive normalization is included. This consisted of dividing all the weights by factors that maintain the sums (for all a)

$$\sum_{i=0}^{N} w_{ai} = \alpha \tag{3.3}$$

at a constant value α. The important point here is that neither of the above rules, 3.2 or 3.3, involves the error function or any other form of supervisory signal.

All the supervision in our network takes place at the level of the gain and shift parameters governing the input-output function of equation 3.1. It is not unusual for supervised learning schemes to modify such parameters, particularly shift parameters. Furthermore, in our scheme, supervision of shift and gain parameters takes place through the same type of stochastic gradient decent procedure used in conventional supervised learning algorithms. The novel element in our approach is that these parameters are the *only* targets of supervised modification. The reason that we restrict supervision to the shift and gain parameters of neuronal response functions is that, unlike the supervision of synaptic plasticity, such supervision can be accomplished by ordinary, fast excitatory and inhibitory synapses from neurons of the supervisory circuit onto neurons of the function-approximation network. Changes in the shift variable s_i correspond to having the supervisor provide either net excitatory or net inhibitory input to network neuron i. It has been shown that balanced, parallel modulations of excitatory and inhibitory input can modify the gain of a postsynaptic neuron (Doiron et al., 2001; Chance et al., 2002; Prescott, and De Koninck, 2003), and on the basis of this result we argue that the supervisor can also control and modify the gain variable g_i.

In summary, the supervisory circuit in our model can modify both the shift and the gain variables for each of the neurons in the network (though in our examples, only the input neurons are modulated) through normal excitatory and inhibitory synaptic pathways. To reiterate what was said in the introduction, our goal is not to introduce a new algorithm, but rather to see if existing algorithms can still operate when supervision is restricted to well-established cortical pathways.

Function Approximation

We apply the proposed mechanism of supervised learning to function approximation, a well-studied task in the artificial neural network literature with obvious applications to biological systems (Poggio, 1990). In this task, network neurons are driven by a stimulus characterized by a single variable θ. The goal of learning is to produce a network output that matches a specified function or set of functions of θ. This is a very easy task for neural network learning that can be accomplished with a single layer of synapses modified, for example, by a delta learning rule (Widrow and Hoff, 1960). We consider this task because it allows us to illustrate clearly the features and limitations of the scheme we are studying.

Specifically, we consider a two-layer feedforward network architecture with purely excitatory connections, as shown in figure 4.1. The network consists of N input units, responding to a stimulus variable θ (which takes values in the range from 0 to 2π), that drive M output units. The input units of the network, indicated by the lower row of circles in figure 4.1, are driven by currents that are Gaussian functions of the difference between θ and a preferred stimulus value, which is different for each input unit. Specifically, the input to unit i, I_i, is given by

$$I_i = G(\theta - \theta_i) + G(\theta - \theta_i - 2\pi) + G(\theta - \theta_i + 2\pi)\,, \tag{3.4}$$

where

$$G(\theta) = 1.5 \exp\left(-\frac{\theta^2}{2}\right) - 0.5\,. \tag{3.5}$$

The three terms appearing in equation 4.2 impose an approximate periodicity on the network, which is convenient (though not essential) because it removes edge effects. The values of the preferred stimulus parameters, θ_i for $i = 1, 2, \ldots, N$, are uniformly distributed over the range from 0 to 2π. The response of input unit i to stimulus θ, $r_i(\theta)$, is given in terms of the input I_i by equation 3.1.

The output of the network consists of the firing rates of the units appearing at the top of figure 4.1. These are determined by the same firing-rate function as in equation 3.1, but their inputs are given by a weighted sum of the firing rates of

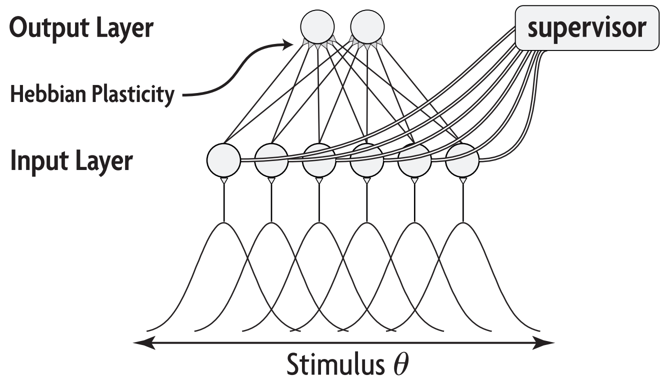

Figure 3.1: The function-approximation network. Input units (lower row of circles) receive input tuned to the value of a stimulus variable θ, as indicated by the Gaussian curves. The input units drive output units, shown at the top of the figure, through synaptic connections that are subject to Hebbian plasticity. A supervisor modifies the response properties of the input units through feedback projections. The task is to induce the firing rates of the output units to match specified functions of the stimulus variable.

the input units. Specifically, using $R_a(\theta)$ to denote the response of output unit a (for $a = 1, 2, \ldots, M$) to stimulus θ,

$$R_a(\theta) = \frac{1}{1 + \exp\left(-g_a\left(\sum_{i=1}^{N} w_{ai} r_i(\theta) - s_a\right)\right)}\,. \tag{3.6}$$

Here, w_{ai} is the weight of the connection from input unit i to output unit a. When we consider networks with a single output unit, we drop the output index and denote the weight from input i simply as w_i. The goal of learning for this network is to match the outputs $R_a(\theta)$, as closely as possible, to a set of stimulus-dependent target functions $F_a(\theta)$.

ACTION OF THE SUPERVISOR

The supervisor in our network model computes an error by comparing the firing rates of the output units to the values of the target functions for each stimulus. It uses a stochastic gradient descent algorithm to adjust the gain and shift values for the input units of the network of figure 4.1 in such a way that the error

$$E(\theta) = \frac{1}{2}\sum_{a=1}^{M}(R_a(\theta) - F_a(\theta))^2 \ , \tag{3.7}$$

is reduced after each stimulus presentation. Here, R_a is the response of output unit a and F_a is the target response for that unit.

As stated above, synaptic weights in the network are subject to Hebbian synaptic plasticity as described by equations 3.2 and 3.3, with $\epsilon_w = 0.03$ and $\alpha = 5.5$. Error-based supervision is used to vary the shift and gain parameters (s_i and g_i) that control neuronal responsiveness. These are set to the initial values $s_i = 1$ and $g_i = 3$ for all units, but are then changed by the supervised learning algorithm. During each run, a stimulus value θ is chosen randomly in the range from 0 to 2π, and the resulting output rates are computed. Then, the shifts and gains for all the input units of the network are updated according to the rules

$$s_i \rightarrow s_i - \epsilon_s \frac{\partial E(\theta)}{\partial s_i} \quad \text{and} \quad g_i \rightarrow g_i - \epsilon_g \frac{\partial E(\theta)}{\partial g_i} \ , \tag{3.8}$$

where ϵ_s and ϵ_g are small parameters that control the rate of response modulation. For our simulations, these took the values $\epsilon_s = \epsilon_g = 0.2/M$. This process is repeated until performance stops improving.

We could also adjust the corresponding parameters s_a and g_a for the output units, but for the examples we give this is unnecessary. Instead, these have been held at their initial values $s_a = 1$ and $g_a = 3$ for all a. The adjustment of output shifts and gains is unnecessary in the examples we present because we have chosen parameters so that the mean of the output response, averaged across all stimuli, is equal to the stimulus-average of the target function. This is not essential, it was done primarily to simplify the presentation.

It is useful to compare and contrast our approach with the conventional use of the delta rule in this situation. In the conventional approach in which synaptic plasticity is supervised, the error in equation 3.7 is differentiated with respect to the synaptic weight w_{ai}. This weight is then updated according to the rule (assuming a gain of one)

$$w_{ai} \rightarrow -\epsilon_w \frac{\partial E}{\partial w_{ai}} = \epsilon_w\,(F_a - R_a)\,R'_a r_i \ , \tag{3.9}$$

where R'_a stands for the derivative of the response of output unit a with respect to its input current. The term $(F_a - R_a)R'_a$ can be thought of as an error signal

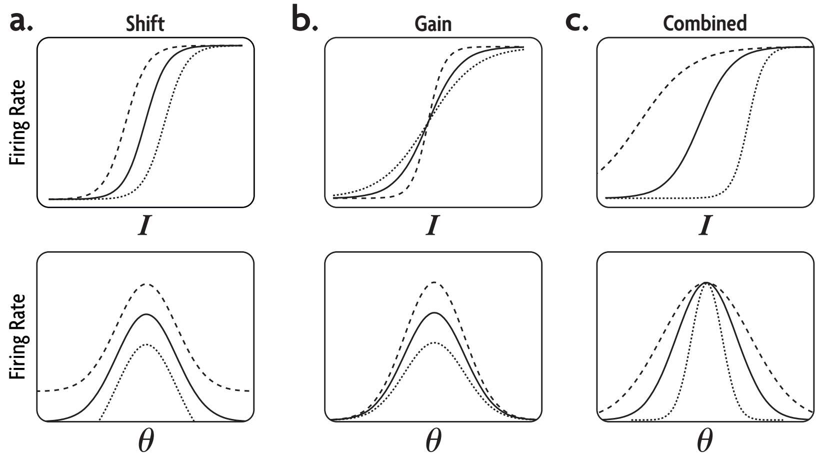

Figure 3.2: Effects of shift and gain modulations on firing rates and response tuning curves. The upper plots show neural responses as a function of the input current I, and the lower plots show them as a function of the stimulus parameter θ. a) Changing the shift variable slides the response-current curve to the left or right and moves the tuned response up and down. b) Changing the gain variable changes the slope of the response-current curve and has a roughly multiplicative effect on the tuned response. c) Changing both variables changes the width of the tuned response.

sent to output unit a that, in conjunction with the presynaptic firing rate r_i, controls modification of the weight w_{ai}.

In contrast, the error signals in our scheme, given by the derivatives in equation 3.8, are "sent" to the input units of the network rather than to the output units. Furthermore, these guide the modification of parameters affecting neuronal responses, not synaptic weights. Although the supervised learning rules in equations 3.8 and 3.9 may look similar in terms of mathematical abstraction, we stress that the modification described by equation 3.8 can be generated by normal, ionotropic synaptic transmission from the supervisory circuit to the targeted neuron, whereas those of equation 3.9 cannot. This is why we are considering such a modified form of delta-rule learning.

The ability to change both the shift and gain variables that determine neuronal excitability provides considerable flexibility in modulated neuronal responsiveness. The different effects of shift and gain modulations on the firing rate of a model neuron, both as a function of its input current and of the stimulus variable, are shown in figure 3.2. Changing, the shift parameter translates the firing-rate curve right and left or, plotted as a function of the stimulus variable, shifts the tuning curve up and down (figure 3.2a). Changing the gain variable modifies the slope of the firing-rate curve and modulates the firing-rate tuning curves in a roughly multiplicative manner (figure 3.2b). Adjusting both variables allows the width of the tuning curve to be changed without an 'iceberg' effect (figure 3.2c).

Results

In studying supervised learning through response modulation, we separately consider networks with a single output unit and networks with multiple output units. Obviously, the case of a single output unit provides less of a challenge than multiple outputs to any learning algorithm. Nevertheless, we consider it here because it provides a clear example of the interaction between supervised response modulation and unsupervised synaptic plasticity. We begin by showing that supervised response modulation, acting by itself without any accompanying synaptic plasticity, leads to a solution of the function approximation problem with a single output unit. This has some implications for network switching that we mention briefly. However, it does not provide a satisfactory long-term solution because the supervisor-induced modulations do not produce any permanent changes in the network. This means that the task can only be performed, even after learning, with continuous input from the supervisor. This problem is resolved by adding Hebbian synaptic plasticity to the learning scheme. This allows the supervisor-induced modulations to be transferred into changes of synaptic strength. Ultimately, this transfer allows the network to function properly even in the absence of supervisory input.

Supervised learning through response modulation is more difficult in networks with multiple output units. In the multi-output case, situations often arise in which response modulation, acting without synaptic plasticity, cannot solve the function approximation task. As an example, consider an input unit that projects to two output units that are supposed to represent two different functions. For one of these functions it might be appropriate to enhance the response of this input unit, while for the other it may be necessary to decrease its responsiveness. Clearly without access to the separate synapses that connect this single input unit to its multiple output targets, both of these criteria cannot be satisfied. In such situations, Hebbian plasticity does not merely act as a way of transferring supervisory modulation into permanent network changes, it must act in concert with response modulation for the task to be learned at all. This is indeed what happens. We find that a combination of supervised response modulation and unsupervised synaptic plasticity allows networks with multiple outputs to compute multiple functions provided that the connection probability between the input and output layers is less than about 95%.

NETWORKS WITH A SINGLE OUTPUT UNIT

Learning and Switching Through Response Modulation

Because response modulation is the pathway through which supervision affects network responses in our studies, it is useful to start off by considering what happens when response modulation acts alone, without the Hebbian synaptic plasticity that will be added later. Therefore, we begin the study of networks with a single output unit by showing that function approximation can be accomplished solely on the basis of response modulation. For figure 3.3, synaptic connection strengths were held fixed while a gradient-decent supervisor varied the shifts and gains of the input units. In other words, we used equation 3.8 but not equation 3.2 during learning. Figure 3.3a shows the initial state of the network in which the output response is independent of the stimulus (upper panel), because all the input units have identical shifts and gains, as revealed by the identically shaped response curves in the lower panel. After the gradient-descent response modulation algorithm has acted, the output response matches the target function (upper panel of figure 3.3b) due to the modulation of responses revealed by the modified response curves seen in the lower panel. To match the cosine-like target response, the input units selective for stimuli near 0 and 2π have been up-regulated by the supervisor, while those selective for stimuli near π have been down-regulated.

Using supervised response modulation, the network can approximate a wide variety of functions (some examples are shown, along with the distributions of shift and gain values that produce them, in figure 3.4). It is important to keep in mind that the distributions of shift and gain variables shown in the left column of this figure, could arise from specific excitatory and inhibitory inputs generated by a supervisor circuit. Thus, each function computed by the network corresponds to a specific pattern of activity within the hypothetical supervisory circuit. If these patterns of activity are remembered and later recreated within the supervisor circuit, this will induce the function-approximation network to compute the target function related to that pattern of activity. Thus, after learning has taken place, the supervisor can act as a controller, rapidly switching the input-output relationship of the function-approximation network between prelearned states. Although we do not consider this form of switching further in this paper, it provides an interesting mechanism by which one neural circuit can control, activate, and switch the function of another (for a related discussion, see Lukashin et al., 1994).

In this network, the input unit responses act as basis functions for representing the output response. Because they do not provide a complete set for arbitrarily high frequencies, there are limits to the types of functions that can

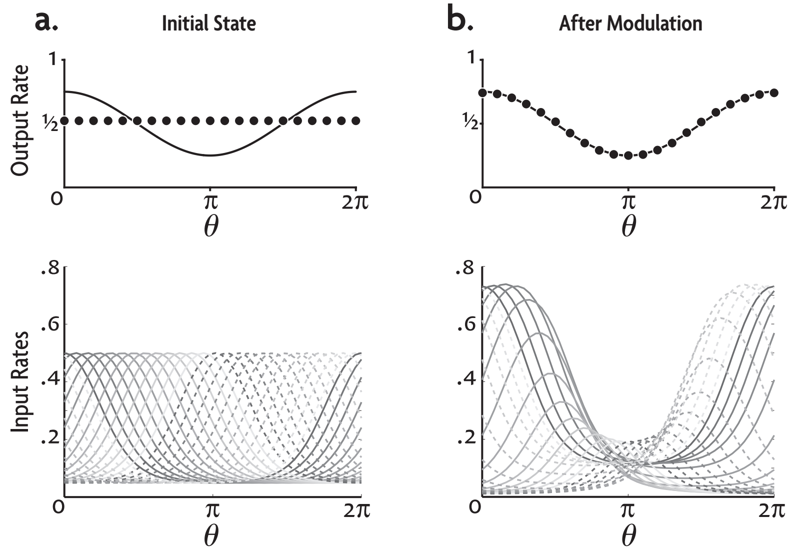

Figure 3.3: Function approximation by supervised response modulation acting alone without synaptic plasticity. Upper curves show the response of the single output unit to various stimulus values (dots) and the target function (line). The lower panels show a sampling of the responses of the 230 input units as a function of the stimulus value. a) State of the network before learning. All input responses have the same shifts and gains, and the output is independent of the stimulus. b) State of the network after learning. The output unit responses match the target function due to modulation of the input unit responses.

be accurately approximated. Limitations arise when the target function varies rapidly, as seen in figure 3.4. Although these limitations exist, they are less severe than they would be in a function approximation network that relied solely on synaptic modification. This is because the tuning curve narrowing seen in figure 3.2c can somewhat ameliorate problems with approximating rapidly varying functions.

In the following examples, we choose to approximate sinusoidally varying functions, and do not present examples with other types of functions. All the networks shown can produce equivalent results with any target functions for which the input responses provide an adequate basis.

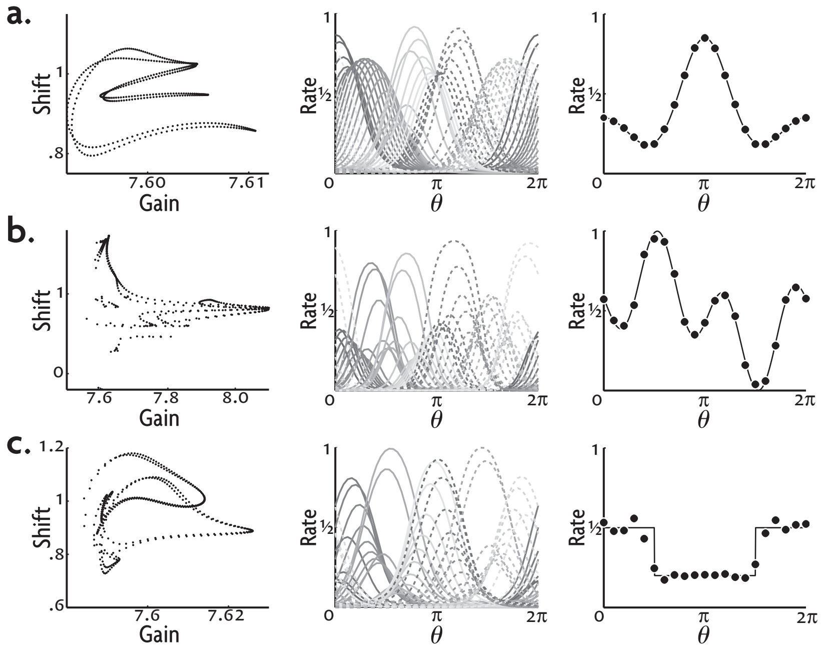

Figure 3.4: Examples of learning through response modulation. Three different functions are approximated by response modulation. The left column shows the shift and gain variables for all 460 of the input units of the network, each dot representing one input unit. Gains and shifts can also be seen in the sampling of input unit response curves seen in the middle column. The right column shows the output unit response (dots) and target function (line) plotted against the stimulus value.

Transfer of Learning to Synapses

In the previous section, we considered supervised response modulation acting alone.

We now add to this a Hebbian plasticity mechanism. In other words, we now use both equation 3.8 and equation 3.2 during learning. The combined effect of supervised response modulation and unsupervised synaptic plasticity is illustrated in figure 3.5. As before, the network is initialized with uniform weights and all shifts and gains set to the same values. The supervisor then modifies response properties to minimize the output error. Early on during the learning process (top row of plots), the performance of the network relies almost entirely on the response modulation of the input units produced by the supervisor (illustrated by the distribution of input responses in the top row, left panel). At this point, the weights have hardly changed from their initial values (as seen in the top row, center panel). However, as the simulation progresses, the weight changes become progressively larger (second row, center panel), and the response modulations become progressively smaller (second row, left panel). Ultimately, the weights take on the cosine shape of the target function (third row, center panel), and the responses are almost uniform for all the input units (third row, left panel), as they were at the beginning of the learning process. Note that a stable equilibrium is reached when Hebbian modification and response modulation act together. Once the response modulation and Hebbian plasticity have equilibrated, the supervisory input can be removed altogether, returning all shifts and gains to their default values, and yet the network can still generate a good approximation of the target function (bottom row of figure 3.5) (although, for stability, this necessitates the deactivation of Hebbian plasticity). Unsupervised synaptic plasticity thus allows the supervisor to contribute progressively less as the burden of representing the target function is taken up by the synapses.

Supervised response modulation plays three critical roles in guiding the Hebbian development of synapses capable of performing the function approximation task. First, because supervised response modulation acting alone can solve the task, the supervisor can act through the input units to effectively clamp the output to the correct response profile while Hebbian plasticity is taking place. Second, by increasing the responsiveness of appropriate input units while clamping the output to the target function, supervised response modulation sets up the appropriate pattern of correlation across the synapses of the network to guide Hebbian modification. For example, input units that are important contributors to the correct output response will be pushed to high levels of responsiveness by the supervisor, enhancing their correlations with the correctly clamped output unit. This causes the synapses connecting such units to the output to grow rapidly. Input units not needed for the task will be made unresponsive by the supervisor, so their synapses to the output unit will not be enhanced by the Hebbian modification rule. Instead these synapses will be weakened due to the synaptic normalization constraint.

Finally, we consider a third role for supervised response modulation on the basis of an analysis of Hebbian modification. The form of synaptic plasticity we are using, Hebbian synaptic modification in conjunction with divisive normalization, ultimately sets synaptic weights in this case equal to

$$w_i = \frac{\alpha\langle Fr_i\rangle}{\sum\limits_j \langle Fr_j\rangle} \tag{3.10}$$

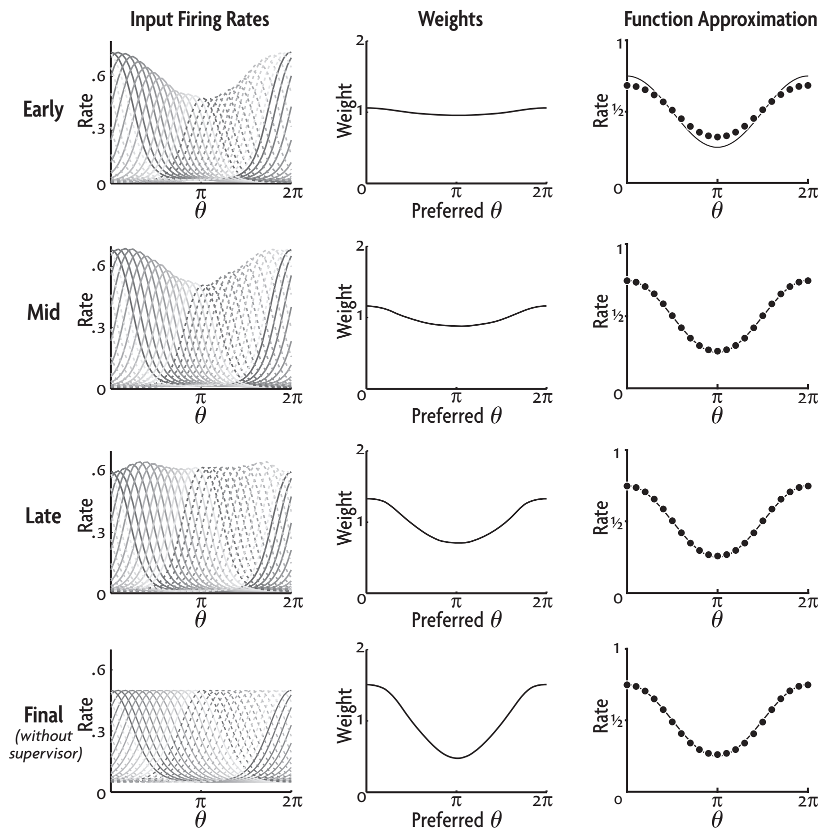

Figure 3.5: Supervised response modulation along with unsupervised synaptic plasticity allows for the transfer of learning to the synapses. The left column shows a sampling of the responses of the 230 input units as a function of the stimulus value. The middle column indicates the synaptic weights of the network plotted as a function of the preferred stimulus value for the presynaptic neuron. The right column shows the responses of the output unit (dots) and the target function (line), plotted against the stimulus value. The top three rows of plots, from top to bottom, show the gradual transfer of learning as the network changes from relying primarily on response modulation (top row of plots) to relying primarily on the pattern of modified synaptic weights (third row of plots). The bottom row illustrates that the network can perform fairly well even when response modulation is totally eliminated, once the appropriate pattern of synaptic weights has been established.

where

$$\langle Fr_i \rangle = \frac{1}{2\pi} \int_0^{2\pi} d\theta \, F(\theta) r_i(\theta) \, . \tag{3.11}$$

To simplify the analysis, we consider a linear approximation for the response function of the output unit, rather than the full sigmoidal form of equation 3.6. In this case and for these weights, the condition that the output response matches the target function,

$$R = \sum_{i=0}^{N} w_{ai} r_i = F \, , \tag{3.12}$$

requires that

$$\alpha \sum_{i=1}^{N} r_i(\theta) r_i(\theta') = \delta(\theta - \theta') \sum_{j=0}^{N} \langle Fr_j \rangle \, . \tag{3.13}$$

The third role of supervised response modulation is to make this equation as near to an equality as possible. The accuracy with which the sum on the left side of this equation can match a δ function profile depends on the narrowness of the tuning curves of the input units. This places a limit on the degree to which rapidly varying target functions can be reproduced but, as mentioned above, modifications in the gain and shift variables can improve this situation by narrowing the input tuning curves. More importantly, supervised response modulation acts to assure that the normalization condition implied by equation 3.13 is met, and this is what ultimately allows Hebbian plasticity to solve the problem (Salinas and Abbott, 2000). Thus, by acting on these multiple levels, supervised response modulation guides Hebbian plasticity to a solution of the function approximation task.

NETWORKS WITH MULTIPLE OUTPUT UNITS

Networks with multiple output units present a greater challenge to the form of supervised learning we are proposing than do single-output networks. In particular, situations frequently arise where the representation of different functions by different output units cannot be achieved by response modulation alone due to shared input. Two cases are simple to analyze. If the connectivity between the input and output units is all-to-all with equal weights, response modulation alone is clearly unable to produce different responses in the output units. With all-to-all coupling, all the output units receive the same total drive, and whatever modulation is done at the input level affects all of the output units in the same way. Unsupervised synaptic plasticity does not help because the input correlation structure seen by the synapses to each output unit is identical, so the synapses will all be modified in an identical manner. Basically, the problem with all-to-all coupling is symmetry; all the output units are equivalent and supervised modulation of input responses is not sufficient to break this symmetry and allow the output units to respond differently to the stimulus. Although we have assumed that the synapses take identical values, setting the initial synaptic weights to different values does not fix this problem.

At the opposite extreme, if the coupling from input to output units is so sparse that each input unit projects to just a single output unit, the situation reduces to multiple copies of the single-output case, and the analysis becomes a trivial extension of what was done in the previous section. In this section, we consider intermediate cases where the input-to-output connectivity is not all-to-all, but there is nevertheless considerable overlap in the input to different output units. We start by considering the case of two-output units and construct networks with various amounts of overlap in the projections they receive from the input units. For an overlap of q, the number of input units that project to both output units is qN, and the number that project to only a single output unit it $(1 - q)N$. We ask whether, in such cases, unsupervised synaptic plasticity can exploit small differences in the drive to each output unit to break the symmetry and allow the output units to represent different functions.

Figure 3.6 illustrates the ability of unsupervised synaptic plasticity to play the role of a symmetry-breaking mechanism. In the first panel, supervised learning acting without synaptic plasticity has set the shifts and gains to their optimal values, but due to the degree of interference caused by shared input, the approximation is quite poor and neither target function has been matched. The network has essentially split the difference between the two functions, with only small disparities between the responses of the two output units. However, when Hebbian plasticity is activated, it is able to exploit and amplify these small differences to improve performance dramatically. This ultimately leads to a match of the two different target functions (center panel of figure 3.6). At this point, the supervisory input is no longer necessary (provided that the Hebbian process is halted). Thus, the combination of supervised response modulation and unsupervised synaptic plasticity allows the network to perform this task at a level that could not be achieved via response modulation alone.

The problem of indirectly supervising the plasticity of NM synapses by modulating only N neurons might, at first, appear to be a crippling limitation of response modulation. The example of figure 3.6 shows at least one case in which this problem is not nearly as severe as might have been imagined. Separation of the two output units could still be achieved when they shared up to 95% of their inputs. However, it is critical to the success of supervision by response modulation that the requirement of a unique component for the input to each output unit scale appropriately as the size of the network and the number of output units increase. Initially, we investigate this issue by varying the amount of shared input in a network with two output units.

The result of varying the proportion of shared inputs in two-output networks of different sizes, when both supervised response modulation and unsupervised synaptic plasticity are active, is shown in figure 3.7. In this figure, and in figure 3.8, performance is quantified by computing the error of equation 3.7, divided by the number of output units. The left panel of this figure shows that, when learning occurs via response modulation alone without synaptic plasticity, errors begin to grow once the proportion of shared inputs exceeds 50% (dashed curves in figure 3.7a). Performance is virtually identical for different input population sizes. With Hebbian plasticity included, the required number of unique inputs decreases dramatically, with little error accumulating until 90–95% of inputs are shared (solid curves in figure 3.7a & b). The point of the transition from small errors to large errors appears to be roughly the same for all the network sizes studied (figure 3.7b). The main effect of increasing the number of input units is to make the transition point, where the function approximation network fails, sharper. This suggests that a discontinuous phase transition occurs at a critical percentage of about 94% shared input in the $N \to \infty$ limit.

We now extend these results to networks with more than two output units. In this case, the proportion of shared inputs (q) is not appropriate for describing all the different possibilities for sharing projections from the input units. Instead, we use the connection probability (p) to characterize the networks we study. To construct these networks, we introduce a connection between any one of the N input units and any one of the M output units with probability p. If such a connection is formed, it is subject to Hebbian plasticity. If no connection forms during this stochastic initial wiring, the connection remains absent for the entire duration of the simulation. The connection probability controls the sparseness of the network in that small values of p correspond to sparse connectivity.

Figure 3.8 shows function-approximation errors for networks with different numbers of output units, as a function of the connection probability p, for two sizes of input unit populations. In this case, both response modulation and synaptic plasticity are activated. As in the two-output case, interference does not become a serious impediment to learning until p reaches the .9–.95 range, indicating that truly unique inputs are not necessary. Rather, the requirement is a certain degree of sparseness. Also noteworthy is the fact that the output population size can approach $^1/_3$ of the total number of inputs before performance begins to suffer from interference (provided p values are not too large).

Our results indicate that connection probability is the dominant factor that controls whether a network with multiple output units, using both supervised response modulation and Hebbian plasticity, can function properly. The required sparseness in the connectivity is not stringent. Furthermore, approximation of

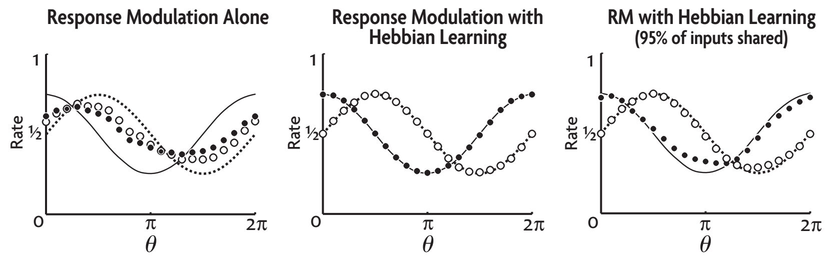

Figure 3.6: Unsupervised synaptic plasticity allows learning to succeed where interference causes supervised response modulation, acting without synaptic plasticity, to fail. In the left panel, a network of 460 input units connected with 88% overlap to two output units was simulated with response modulation acting without synaptic plasticity. The responses of the two output units, denoted by filled and open dots, failed to match the two target functions, indicated by dashed and solid curves, as a function of the stimulus value. When Hebbian plasticity was included, the two output responses accurately matched the target functions (middle panel). The combination of supervised response modulation and Hebbian plasticity continued to produce accurate outputs until the common input to the output units was increased to 95% (right panel).

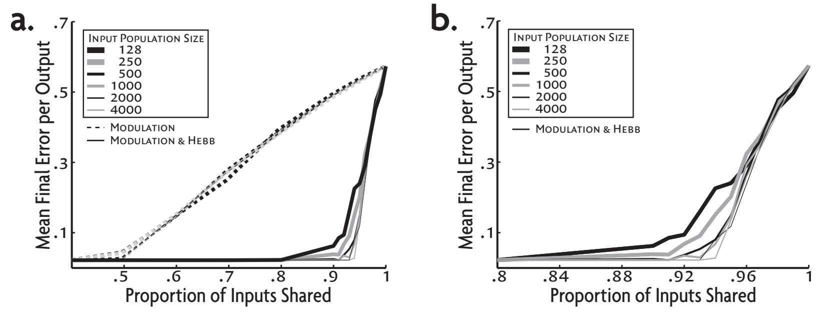

Figure 3.7: Error as a function of the proportion of shared input for a two-output network with different numbers of input units. Dashed lines show the mean error per output unit resulting from supervised response modulation without synaptic plasticity. Solid lines are from runs that also employed Hebbian plasticity. Individual lines correspond to different numbers of input units (N). The functions being approximated are cosine and sine. a) Network performance degrades as the proportion of inputs projecting to both outputs increases. b) Detailed view of the results for supervised response modulation with Hebbian plasticity shown in panel a.

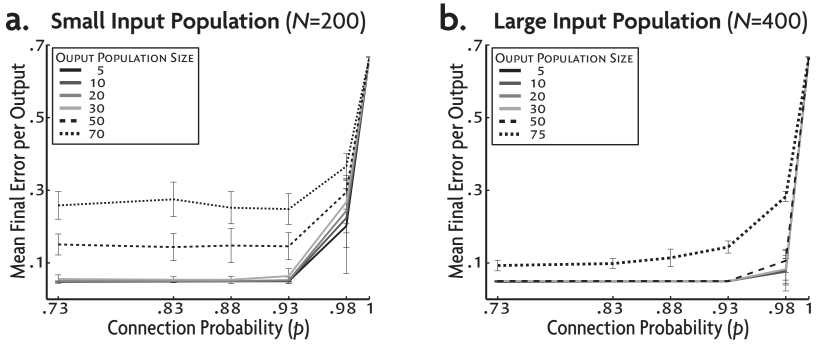

Figure 3.8: Error as a function of network connection probability for different numbers of output units. The functions being approximated are cosines with phases ranging from 0 to 2π, equally spaced between the output units. a) A network with 200 input units. Error increases rapidly for $p > 0.93$ regardless of the number of output units. For 50 and 70 output units, the error is larger for all p values. b) For a population of 400 input units, the results are similar, except that overall performance only degrades when there are 75 output units.

multiple functions is possible even when output population size is a significant fraction of the total input population size.

The analysis of networks with multiple output units is more difficult than in the single-output case, but some of the same basic principles apply. In this case, the supervisor cannot clamp the output units to their target functions, but the existence of even a small number of "symmetry-breaking" synapses is sufficient to break this impasse. These synapses initially get quite strong and drive the output units away from the degenerate state in which they are all the same, which starts off the combined response modulation-Hebbian learning process. Through this process, the bulk of the synapses ultimately come to obey the multi-output generalization of equation 3.10,

$$w_{ai} = \frac{\alpha \langle F_a r_i \rangle}{\sum_j \langle F_a r_j \rangle} . \tag{3.14}$$

Similar to the result in the one-output case, and making the same linear approximation, matching of the target function with these synaptic weights requires that

$$\alpha \sum_{i=1}^{N} r_i(\theta) r_i(\theta') = \delta(\theta - \theta') \sum_{j=0}^{N} \langle F_a r_j \rangle . \tag{3.15}$$

Subject to the same constraints on the approximation of the δ function, these equations represent M constraints that need to be satisfied by appropriate adjustment of the $2N$ shift and gain variables of the input units, which should be possible to satisfy when $M < N$. The key to making the combined response modulation-synaptic plasticity scheme work is that Hebbian modification reduces the problem of setting pNM synaptic weights to the problem of satisfying the M constraints appearing above, and this can be done by the supervision through its control of the $2N$ shift and gain variables.

A STOCHASTIC SUPERVISOR

The supervisor used in the simulations discussed thus far employed a gradient descent algorithm to modify intrinsic response properties on the basis of the error generated by each stimulus. A biological supervisor circuit is more likely to operate under a reinforcement-based scheme. As a first attempt at constructing such a supervisor, we have implemented a model using a stochastic search guided only by a reward signal that reflects network performance. Related ideas have been applied to the supervision of synaptic plasticity (Barto et al., 1983; Mazzoni et al., 1991; Jabri and Flower, 1992; Williams, 1992; Cauwnberghs, 1993; Doya and Sejnowski, 1995; O'Reilly, 1996; Xie and Seung, 2003; Seung, 2003).

For stochastic reward-based supervision, two N-dimensional "modification" vectors, v^s and v^g, of unit length were generated randomly, one for shifts and one for gains. For all i values, the shift and gain of unit i was incremented by an amount proportional to component i of the appropriate modification vector,

$$s_i \to s_i + v_i^s \quad \text{and} \quad g_i \to g_i + v_i^g . \tag{3.16}$$

Simulations were divided into epochs of 20 stimulus presentations and error evaluations. After each epoch, the sum of the 20 errors was compared to the summed error from the previous epoch. If this total error was less than it was previously, the modification vectors were left unchanged. If the summed error increased from the previous epoch, new modification vectors, v^s and v^g, were generated randomly. In either case, the resulting modification vector was then used to further increment the shifts and gains, as described above. In this study of random walk learning through response modulation, we do not include any Hebbian synaptic modification.

This strategy has the effect of steadily, although slowly, reducing the average error. The paths through modulation space of three input units over the course of a run are plotted in figure 3.9a. The improvement in performance can be seen in the reduction in the lengths of the line segments seen in the traces. At the beginning of the run, most of these segments are relatively long as the network

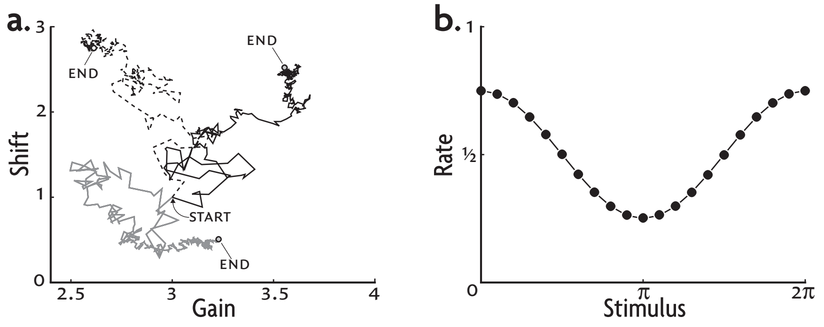

Figure 3.9: Learning under the random-walk supervisor. a) The paths in modulation space of three input units controlled by the supervisor. All three units started with the same shifts and gains (marked 'START'), but these then diverged as the supervisor found values that accomplished the function approximation task ('END'). b) The end result is a good approximation (dots) of the target function (line) by the output unit as a function of stimulus value. In this simulation, 230 input units drove a single output unit.

makes coarse adjustments to approach the target function. Later, more frequent trajectory changes appear as the network approaches a solution and makes fine adjustments.

Figure 3.9b illustrates that this crude strategy is capable of solving the task, given enough time (in this case 600 iterations). Thus, a random-walk supervisor strategy that requires much less information and algorithmic sophistication than gradient descent can, at least in simple cases, provide adequate supervision.

There are clearly severe limitations on the sizes of networks that can be trained by this random-walk algorithm. As the network grows in size, the algorithm gets prohibitively slow. In the Discussion, we propose ways that this problem might be addressed to achieve better scaling of performance with network size.

Discussion

The novel feature of the supervised learning scheme we have proposed is that supervision takes place at the level of neuronal responsiveness rather than synaptic plasticity. Two apparent disadvantages of this scheme, that it does not lead to permanent network modification and that it severely limits the number of elements being supervised, appear to be far less severe than might have been imagined at first. By guiding synaptic plasticity that is otherwise unsupervised, supervised response modulation can lead to permanent changes that allow a network to operate effectively, even in the absence of supervision. Furthermore, Hebbian plasticity can take advantage of small inhomogeneities in randomly coupled networks to allow independent changes in synapses to output units that share presynaptic input.

Given that it works, there are some potential advantages of supervising neuronal excitability rather than synaptic plasticity. First, supervision can occur through ordinary feedback projections that can act rapidly and can target individual neurons independently. Additional advantages concern the nature of the supervisory circuit. We have not attempted to construct a realistic model of this circuit, but we envision it as a network capable of maintaining a continuum of stable, self-sustained patterns of activity (Compte et al. 2000; Seung et al., 2000). Such networks tend to drift, especially if provided with noisy input. Thus, it might be possible to implement the random walk supervisor as a network with self-sustained activity and random drift, with the rate of drift controlled by noisy inputs that are suppressed by reward.

Whatever the form of the supervisory circuit, modulating neuronal responsiveness instead of synaptic plasticity has a number of tactical advantages. We considered two approaches to supervision: gradient descent, which involves more information and mathematical analysis than we would expect from a neural circuit, and a random walk model that uses less. A real circuit should lie somewhere between these extremes. From the point of view of the supervisor, the fact that there are far fewer neurons than synapses to supervise changes from a disadvantage to an advantage. The supervisor must search in the space of the parameters it is modifying for a solution to the problem at hand. By reducing the dimension of this space, supervision of neuronal responses, provided that it works (and we have shown that it does), is far easier than supervision of more numerous synapses.

Another advantage of supervising neuronal responses is that the supervisor can monitor the activities that it is modulating in a way that is impossible with supervised synaptic plasticity. It is almost inevitable that the function approximation network, which receives input from the supervisor circuit, would also send projections to it. Such reciprocal connectivity is a typical feature of neuroanatomy. These projections allow the supervisor to monitor the activity of the units it is supervising and use this information to guide learning. For example, this information could be used to reduce the dimension of the space in which the supervisor must search for solutions of the task being learned. Consider, for example, two input units in the function approximation network that have almost totally overlapping response profiles. It is rather wasteful for the supervisor to vary the response properties of these two neurons independently, and yet this is what was done in the random walk model we studied. A more "intelligent" supervisor would use information about the correlations between the units it is modulating to find strategies that are most likely to produce large changes in the network being supervised, and to avoid wasting time generating modulations that have little effect. Thus, projections from the supervised units to the supervisor could be part of a secondary modulatory process that allows the supervisor to learn about learning.

CHAPTER FOUR

Principal Component-based Strategies for Supervised Learning

Introduction

Learning often takes place solely through the reinforcement of improved performance. Such reinforcement-based learning is challenging because no information is provided to indicate how a task should be done or suggest how performance can be improved. Instead, strategies must be generated internally and evaluated solely on the basis of the reinforcement they generate.

Faced with such a dearth of information, models of learning often rely on a random-search approach in which reinforcement guides an otherwise random walk in the space of parameters controlling task performance (Barto et al., 1983; Mazzoni et al., 1991; Jabri and Flower, 1992; Williams, 1992; Cauwnberghs, 1993; Doya and Sejnowski, 1995; O'Reilly, 1996; Xie and Seung, 2003; Seung, 2003). In a neural network, such a scheme typically involves randomly changing synaptic strengths or neuronal excitabilities and keeping or rejecting those changes on the basis of reward. For example, in the scheme we use, which is based on bacterial chemotaxis, changes are made by moving along a straight line in the space of parameters as long as performance improves and reward is provided. If at some point reward is denied—indicating worsening performance—the system starts moving in a new, randomly chosen direction.

Reward-based learning strategies of this type typically converge to a set of parameters that optimizes task performance if they are applied for a sufficient time. Because this parameter search is random, this time can be very long, and it typically scales badly (i.e., increases dramatically) as the size of the network performing the task increases. In most cases, the space of network parameters is enormous, and the system can easily get lost when reinforcement is the only guide.

Here we explore the idea that the convergence time for reinforcement-based, random-walk learning schemes and its scaling with network size can be improved dramatically by reducing the effective dimension of the parameter space. This is a rather obvious strategy, and we use a standard dimensional reduction method, principal component analysis (PCA) to implement it. The novelty is that we implement both the dimensional reduction scheme and the mechanism by which neuronal excitabilities and synaptic strengths are modified through well-known, local synaptic modification rules acting along biologically plausible pathways.

Methods

THE TASK AND THE NETWORK

To explore the role of dimensional reduction in reinforcement learning, we chose a well-defined task of obvious behavioral and cognitive relevance (Poggio, 1990), function approximation. In the network we consider, N input units respond to inputs that are tuned to the value of a particular stimulus parameter (in our case, an angle θ), and they drive an output unit, making its firing rate follow a specified function of the stimulus value (the network can easily be extended to include more than one output unit, but this is unnecessary for our purposes). The learning task consists of adjusting network parameters so that the firing rate of the output unit matches a specified target function. This is the task to which we apply reinforcement learning because it allows us to illustrate clearly the features and limitations of the scheme we are studying.

The architecture of the network is shown in figure 4.1. Each input unit is characterized by a firing rate, r_i, with $i = 1, 2, \ldots, N$, that is given by a sigmoidal function of the total synaptic current it recieves. The synaptic current is divided into two terms: a stimulus current $I_i(\theta)$ that depends on the stimulus angle θ, and a bias current J_i that is independent of the stimulus and represents synaptic currents arising from the supervisor circuit shown in figure 4.1. Thus,

$$r_i = \frac{1}{1 + \exp\left(-g(I_i(\theta) + J_i - s)\right)} . \tag{4.1}$$

The parameters s and g control the shape of the sigmoid. The shift parameter, $s = 0.9$, determines the location at which the firing rate reaches its half-maximal value, and the gain parameter, $g = 5$, specifies the slope at that half-maximal point. The firing rate is normalized so that its maximum value is 1.

The stimulus current $I_i(\theta)$ that appears in equation 4.1 is constructed from Gaussian functions of the difference between the stimulus variable θ and the preferred stimulus value for unit i, θ_i,

$$I_i(\theta) = G(\theta - \theta_i) + G(\theta - \theta_i - 2\pi) + G(\theta - \theta_i + 2\pi) , \tag{4.2}$$

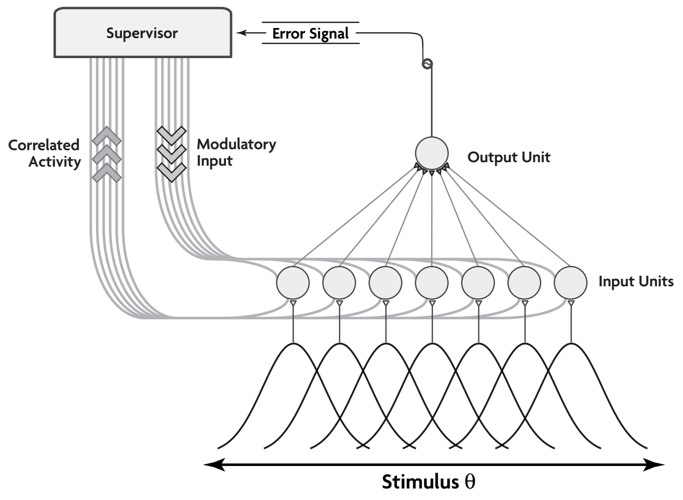

Figure 4.1: Network for function approximation. Input units (lower circles) are driven by input current (tuning curves at bottom) that is a Gaussian function of the difference between a stimulus angle θ and a preferred stimulus value for each unit. The input sends projections to both the output unit (upper circle) and units within the supervisory circuit. Supervisor units also synapse onto each input unit. In addition, The supervisor receives limited information about network performance in the form of reward which it uses to direct its influence on the input units.

where

$$G(\theta) = 1.5 \exp\left(-\frac{\theta^2}{2}\right) - 0.5 . \tag{4.3}$$

The three terms appearing in equation 4.2 impose an approximate periodicity on the network to match the fact that θ is an angle. This stimulus current makes the input units selective for different values of θ, and we can write their firing rates as $r_i(\theta)$. The preferred stimulus values, θ_i for $i = 1, 2, \ldots, N$, are uniformly distributed over the range from 0 to 2π (see figure 4.1), so the input units collectively represent stimulus values over the entire range of angles.

The output unit is driven by the input units through a set of modifiable synapses, so its firing rate R is given by a weighted sum of their rates,

$$R(\theta) = \sum_{i=0}^{N} w_i r_i(\theta) . \tag{4.4}$$

The weight w_i represents the strength of the synapse from input unit i to the output unit. The task for learning in this network is to make $R(\theta)$ match, as closely as possible, a specified target function $F(\theta)$. In our case, this is done by presenting random values of θ to the network on learning trials and providing feedback in the form of a reward or reinforcement signal that indicates improved performance.

The task of making the output unit match the target function, i.e., setting $R(\theta) = F(\theta)$, is assigned to the supervisor circuit shown schematically in figure 4.1. The standard way of doing this is to have the supervisor adjust the synaptic weights between the input and output units of the network using a delta learning rule (Widrow and Hoff, 1960; Widrow and Stearns, 1985). We do not follow this procedure for two reasons. First, the delta rule requires that the supervisor has knowledge of the target function and can determine the error that the network makes on each trial. In reinforcement learning this information is not available. Second, delta-rule learning assumes that the supervisor can control synaptic modification, but there is little evidence for such control of synaptic plasticity in biological circuits. Instead, we assume that the supervisor circuit interacts with the input units solely through conventional excitatory and inhibitory synapses arising from the pathway from the supervisor to the input units depicted in figure 4.1. The input from the supervisor circuit to input unit i is represented by the bias current J_i. Thus, in our biologically realizable scheme, the supervisor modifies the responses of the input units on the basis of a reward signal through ordinary excitatory and inhibitory synapses.

In prior chapters we have examined this form of learning in a supervised rather than reward-based scheme. A pair of results are particularly relevant to the current work:

- It is possible to get the firing rate of the output unit $R(\theta)$ to approximate the target function $F(\theta)$ solely by having inputs from the supervisor adjust the bias currents of the input units to appropriate values. This allows the network to perform the function approximation task as long as the supervisory inputs are maintained at the proper levels.
- If a Hebbian form of synaptic modification is implemented at the synapses between the input and output units, the information about how to perform the function approximation task transfers spontaneously from the bias currents to the synapses. When this transfer is complete, the network can perform the task without any input from the supervisor.

Thus, in this scheme, learning is a two-component process. The excitatory and inhibitory input from the supervisor to the network adjusts the bias currents of the input units to improve network performance. At the same time, the synapses from the input units to the output unit are modified by Hebbian plasticity. This transfers the improvements induced by supervisory input into permanent changes within the network that ultimately allow it to perform the task successfully, even without supervisory input.

A challenge to this two-component scheme is that reinforcement-based supervision and Hebbian plasticity occur together. If reinforcement learning is too slow or spends too long in the wrong parts of parameter space, the Hebbian component will "lock in" modifications that are detrimental to task performance, which can destroy convergence. Thus, it essential for us to find an efficient reinforcement-based scheme that established the correct biases in the input units rapidly enough to keep the unsupervised Hebbian plasticity mechanism on track. For this reason, we begin our study of reinforcement learning by focusing solely on reinforcement-based modification of input unit bias currents by the supervisor circuit, leaving the analysis of the addition Hebbian component until the end.

THE REINFORCEMENT-BASED SUPERVISOR

The supervisor circuit shown in figure 4.1 is responsible for making adjustments to the network on the basis of reward reinforcement to improve performance. To do this, the supervisor must explore the space of network parameters by generating different patterns of excitation and inhibition to the input units.

In a direct application of these ideas to the network of figure 4.1, the supervisor circuit would consist of N units with activities v_i, for $i = 1, 2, \ldots, N$ (we allow v_i to be either positive or negative to represent excitatory or inhibitory inputs provided by the supervisor even though the firing rates of the individual supervisory units would, of course, be positive). The activity of supervisor unit i introduces a bias current into input unit i given by $J_i = v_i$.

We express the activity of the supervisor circuit as the product of a vector of fixed length with components u_i describing the pattern of activity in the circuit, and a scale factor c that represents the level of excitability of the entire circuit, so $v_i = cu_i$. The pattern of activity described by u_i is similar to the sustained activity seen in models of short-term memory (Compte et al. 2000; Seung et al., 2000), and it could easily be generated by such a model (although we will not do this here). The overall factor c could represent the effects of a global modulator acting on the circuit.

The reinforcement-based random walk strategy can now be specific in terms of the effects of reward on the factors c and u_i that determine the supervisory activity $v_i = cu_i$, which in turn determines the bias currents for the input units. As long as reward is obtained, supervisory excitability changes according to $c \to c + \epsilon$, where ϵ is a small parameter that determines the learning rate.

As stated above, this could be due to the effects of a reward-induced modulator. If reward is not obtained, the vector describing the pattern of supervisory activity is changed by choosing a new set of components u_i randomly. This could be realized by having a second modulator or noisy input temporarily disrupt the stability of the self-sustained activity in the supervisory circuit. These two actions completely describe the learning strategy we use although, thus far, we have discussed it and its implementation within the context of a direct approach not the dimensionally reduced scheme we propose.

Chapter 3 demonstrated that the direct scheme can work if N is small enough, but it is slow and gets even slower as N increases. The basic feature we exploit to get around the limitations of the direct approach is the fact that the firing of different input units is correlated. Two units with highly overlapping input tunings tend to fire together at similar rates. If the function being approximated varies slowly on the scale of the separation between preferred stimulus values (i.e., if $F(\theta_i) \approx F(\theta_{i+1})$), it does not make sense to vary the two bias currents J_i and J_{i+1} independently. The best strategy is to vary those combinations of bias currents that have the biggest impact on network output. These combinations can be determined by performing principal component analysis (PCA) on the correlation matrix of the input units. The key result, present in the following section, is that this can be accomplished by applying appropriate forms of synaptic plasticity to the synapses between the input units and the units of the supervisor circuit.

DIMENSIONAL REDUCTION

The learning strategy based on dimensional reduction is similar to the direct reward-guided random-walk strategy discussed above, except that the dimensionality of the space being searched is much smaller. In this case, the modification of the bias current is controlled by only $n << N$ supervisor units through the equation

$$J_i = \sum_{a=1}^{n} w_{ia} v_a = \sum_{a=1}^{n} w_{ia} c u_a \,, \tag{4.5}$$

where v_a for $a = 1, 2, \ldots n$ are the activities of the n supervisor units, and w_{ia} represents the strength of the synapse from supervisor unit a to input unit i. The implementation of reinforcement learning in this dimensionally reduced scheme proceeds exactly as described above for the direct approach. After rewarded trials $c \to c + \epsilon$, and after non-rewarded trials the components u_a are reset to randomly chosen values.

The key to making the modifications described by equation 4.5 as effective as possible is to make sure that the synaptic weight factor w_{ia} is proportional to

the ith component of the ath principal component of the input-input correlation matrix. In other words, we want $w_{ia} \propto \xi_i^a$, where ξ_i^a is the ith component of the eigenvector of the input-input unit correlation matrix with the ath largest eigenvalue. To achieve this, we exploit a virtually universal property of neural circuits, the reciprocal nature of interconnections. The supervisor can obtain information about the correlations between the input units through the projections from the input units to the supervisor depicted in figure 4.1. We refer to these projections as the ascending pathway to the supervisor, and call the projections from the supervisor to the input units the descending pathway. Methods exist for setting the ascending synaptic weights proportional to the principal component eigenvectors of the input-input correlation matrix (Oja, 1989; Sanger, 1989). This is exactly what we want to achieve, but for the wrong set of synapses. We need to set the descending (not the ascending) synaptic weights proportional to these eigenvectors. We now show that this can be achieved by applying ordinary Hebbian-type plasticity to the descending synapses.

We denote the strength of the ascending synapse from input unit i to supervisor unit a by w'_{ai}. Initially, the supervisor units are driven by the input units through these synapses, so the firing rate of supervisor unit a is determined by

$$v_a = \sum_{i=1}^{N} w'_{ai} r_i \,. \tag{4.6}$$

The first step in setting the descending synaptic weights is to apply synaptic plasticity to the ascending synapses that is essentially Hebbian but with the added wrinkle of subtracting out contributions already accounted for by other supervisor units (Sanger, 1989),

$$w'_{ai} \to w'_{ai} + \eta' v_a \left(r_i - \sum_{b=1}^{a} v_b w'_{bi} \right) \tag{4.7}$$

on every time step of the simulation, with η' setting the learning rate, and v_a and r_i the firing rates of supervisor unit a and input unit i, respectively. For the $a = 1$ supervisor unit, this rule is identical to the standard Oja rule (Oja, 1982; see below) which set $w'_{1i} \propto \xi_1^a$, as is well known (Dayan and Abbott, 2001). For the other supervisor units, the subtractive sum in equation 4.7 assures that $w'_{ia} \propto \xi_i^a$ for $a = 2, 3, \ldots n$. The Sanger rule thus sets the connection strengths in such a way that the weight vector from the input units to the $a = 1$ supervisor unit is proportional to the eigenvector with the largest eigenvalue, the $a = 2$ unit to the eigenvector with the next largest eigenvalue, and so on (see figure 4.2).

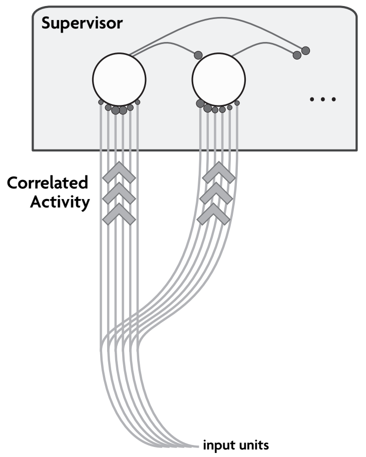

Figure 4.2: Feedforward synapses in the supervisory pathway. Each input unit in the network sends projections to all *n* cells in the supervisor circuit. Plasticity in these synapses is governed by the Sanger rule; a modified Hebb-like scheme that results in the pattern of synaptic weights being proportional to the principal components of the correlation matrix of the input signals.

The second step in setting the descending synapses properly is to transfer the weights from the ascending to the descending pathway. Surprisingly, this can be done by having ordinary, multiplicatively constrained Hebbian plasticity act on the descending synapses. Specifically, we apply the Oja plasticity rule (Oja, 1982) to the descending synapses

$$w_{ai} \rightarrow w_{ai} + \eta v_a \left(r_i - v_a w_{ai}\right) \tag{4.8}$$

on every time step of the simulation, where the parameter η controls the learning rate. The weight decay term $v_a w_{ai}$ in this rule ensures that the weight vector converges toward unit length, preventing the runaway potentiation that would arise form a purely Hebbian rule.

We now consider what happens if, after the ascending weights have been set to the principal component eigenvectors by the Sanger rule (equation 4.7), the Oja rule (equation 4.8) acts on the descending synapses while the supervisor units are driven by the input units according to equation 4.6. The key result is that the descending synaptic weights are set appropriately to $w_{ia} \propto \xi_i^a$ for $i = 1, 2, \ldots, N$ and $a = 1, 2, \ldots n$. To see how this result arises, we note that, when plasticity comes to equilibrum, the Oja rule (equation 4.8) has set the synaptic weights proportional to the pre-postsynaptic correlation matrix (if

η is small enough),

$$w_{ia} \propto \langle r_i v_a \rangle \,, \tag{4.9}$$

where the angle brackets denote an average over time. Combing this result with equation 4.6, we find that

$$w_{ia} \propto \sum_{j=1}^{N} \langle r_i r_j \rangle w'_{aj} \propto w'_{ai} \,. \tag{4.10}$$

The second equality follows from the fact that $w'_{aj} \propto \xi_i^a$ and ξ_i^a is an eigenvector of the correlation matrix,

$$\sum_{j=1}^{N} \langle r_i r_j \rangle \xi_j^a \propto \xi_i^a \,, \tag{4.11}$$

with a constant of proportionality equal to the eigenvalue of the a^{th} eigenvector.

The above derivation shows that, when the ascending synaptic weights are proportional to eigenvectors of the input-input correlation matrix, the Oja rule sets the descending weights proportional to the ascending weights, and thus proportional to those same eigenvectors. That is, $w_{ia} \propto w'_{ai} \propto \xi_i^a$. Thus, the mixture of a Sanger rule on the ascending synapses and a Oja rule on the descending synapses accomplished the goal of setting the descending weights equal to the values needed for the dimensional reduction of reinforcement-based learning. The fact that we only required that equation **??** be satisfied by the learning rule for the descending synapses indicates that any correlation-based plasticity, not only the Oja rule, can be used for this purpose.

Figure 4.3 illustrates how the ascending and descending weights become proportional to the principal eigenvectors through synaptic plasticity. In this example, the input units were driven by randomly chosen angles activating the input currents of equation 4.2 while the Sanger and Oja rules were applied to the ascending and descending synapses, respectively. Because the tuning curves of the input units are placed uniformly over the range of stimulus angle values, assuring translational invariance ($\theta \rightarrow \theta +$ constant is a symmetry), the eigenvectors of the input correlation matrix are sine and cosine functions of the input unit preferred angles θ_i and the principal components are ordered by wavelength. In other words, PCA is equivalent in this case to Fourier analysis. This makes it easy to see that the appropriate synaptic assignments have been made.

The development of the ascending synaptic weights from the input units onto seven supervisor units on the basis of the Sanger rule can be seen in the left panels of figure 4.3 and the progression of the corresponding descending weights

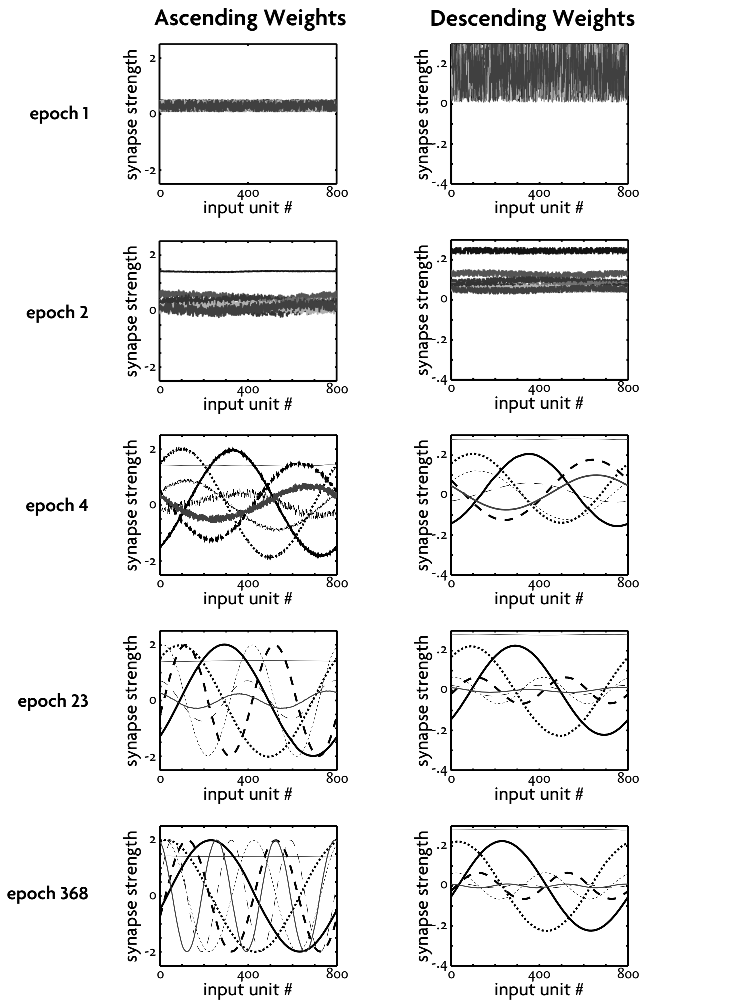

Figure 4.3: Development of ascending and descending synaptic weights. Each line represents the set of synaptic weights to and from one supervisor unit plotted against input unit number. The left column depicts the connections from the input units to the supervisor, with synapses modified over time, from top to bottom, using the Sanger rule. The right column shows the same thing for synapses from the supervisor to the input units, which are controlled by the Oja rule. In the top row, before the network begins receiving random input angles, the weights have random values. Over time (rows below the top show the situations after 256, 1472, and 23,552 trials), the Sanger rule extracts the principal component eigenvectors from the input correlations and sets the ascending weights proportional to them. Simultaneously, the descending weights become proportional to the eigenvector as well.

according to the Oja rule is shown in the right panels. From random initial values (top row), the weights to and from the $a = 1$ supervisor units become equal to the constant Fourier component by 128 trials (the second row of plots in figure 4.3). As the plasticity proceeds (rows 3–5 in figure 4.3 representing the results after 256, 1472, and 23,552 trials, respectively), the additional weights to and from the $a = 2, 3, \ldots 7$ supervisor units take sine and cosine forms. During this process, the development of the descending synapses tends to lag behind that of the ascending synapses because the Sanger rule pulls out the principle components while the Oja rule transfers these components to the descending weights. By the lowest panels, when the system has equilibrated, the weights for synapses to each supervisor unit are paired with weights for synapses from them to the input units proportional to the same principle component eigenvector. The amplitudes of the descending weights are not identical to their ascending siblings, but are instead related to the corresponding eigenvalues. The unevenness of the amplitudes has no effect on the reinforcement learning strategy we study next.

Results

DIMENSIONALLY REDUCED REINFORCEMENT LEARNING

The dimensional reduction process described in the previous section reduces the problem of learning to one of setting appropriate supervisor firing rates v_a for $a = 1, 2, \ldots, n$ to introduce the appropriate biases into the input units to generate the desired responses in the output unit. This is a much simpler problem than trying to set the bias currents independently, and the use of descending weights proportional to the principal component eigenvectors of the input-input correlation matrix assures that a small number ($n = 7$ in the following examples) of supervisor units has the maximum impact on the activity of the output units. The price of lowering the dimensionality of the control process from N, which is typically greater than 800 in our examples, down to $n = 7$ will be addressed after we show how the basic scheme works.

Recall that the dimensionally reduced reinforcement learning process for supervisor unit activities written as $v_a = cu_a$ consists of making the change $c \to c + \epsilon$ to the excitability factor if reward is given and randomly choosing new components u_a for the pattern vector if reward is denied. When the network is in the optimal configuration for task performance, any changes will result in an increased error. To address this type of end-stage thrashing, the learning rate parameter ϵ is decreased as the overall size of the errors made by the network goes down.

Before showing the results of dimensionally reduced reinforcement learning, we need to discuss how reward is delivered to the supervisor to determine whether it increases its overall excitability or choses a new pattern of activity.

The Reward Procedure

Reward criteria are determined by external factors, not by neural circuits. But neural circuits determine in a complex way how reward is interpreted and what results follow from receiving reward. Both of these factors make it impossible to model the reward procedure uniquely. The reinforcement learning scheme we propose will work with any reasonable reward scheme, where reasonable means that reward is based on some cumulative assessment of improvement integrated over a long enough time to assess whether or not a particular set of parameter changes is beneficial. We use a simple scheme to demonstrate that reinforcement learning can work. Undoubtedly, performance could be improved over what we show by using a more elaborate and clever reward schedule, but our purpose is to study the actions of the supervisory circuit, not the reward system.

A trial in our learning scheme consists of the presentation of a randomly chosen stimulus angle (θ), which produces a network output $R(\theta)$ that is supposed to match a target function $F(\theta)$. The reward procedure is based on whether the function approximation error, $(R(\theta) - F(\theta))^2$, averaged over θ values, shows an increasing or decreasing tendency. To assess this fairly, we must computed it over a number of trials with different θ values. To do this, the reward after any trial is based on errors accumulated over the previous 140 trials. These 140 trials are divided into two blocks of 70. The errors for the most recent block of 70 trials are summed, as are the errors for the next most recent block of 70 trials (the 70 trials prior to the most recent block). If the summed error for the most recent block of trials is less than the summed error for the next most recent block, reward is given. If the summed error for the most recent block of trials is larger than that of the next most recent block, reward is denied. This defines our reward procedure.

DEMONSTRATION OF REINFORCEMENT LEARNING

Dimensionally reduced reinforcement learning is illustrated in figure 4.4. The left column in this figure shows the response of the output unit as a function of stimulus angle and the target function that is to be matched. The right column shows the individual contributions to the total bias current provided to each input unit by the 7 supervisory units.

In this simulation, all the supervisor unit firing rates start out at zero (figure 4.4a right panel), which causes all input units to respond identically to their preferred stimuli. This, combined with the fact that synaptic weights between

the input units and the output unit are set equal, in this case results in the output unit response being uniform with respect to the stimulus (figure 4.4a left panel).

After 1600 trials, (second row of figure 4.4), the supervisor units are biasing the input units, resulting in modulation of the flatline response of the initial state. The right panel indicates that this response arises primarily from a baseline shift and a single-cyle cosine modulated bias current that is not well chosen for the target function. The components that ultimately will be responsible for a successful function approximation start to develop at this point as well. After 5632 trials (third row of figure 4.4) a single-cycle sine has become the dominant factor contributing to the approximation. After 11,008 trials (fourth row of figure 4.4) the general shape of the function has been captured by the uniform shift combined with this sine wave, but higher frequency components need to be recruited to account for the detailed shape of the target function. The final state of the bias currents in the bottom row shows that two higher frequency sinusoids now contribute, resulting in a successful approximation. Note that the supervisor units with the highest frequency components were not needed for this particular target function.

Non-Uniform Sampling Distributions

The connectivity patterns we have seen thus far are equivalent to Fourier modes, but the scheme we propose is not limited to this translationally invariant case. To show this, we break the translational invariance by allowing the distribution of preferred stimulus values for the input units to be non-uniform. Two examples of this are shown in figure 4.6.

For the left column of figure 4.6, a bimodal distribution was used in which the peaks had a sampling density triple that of the low-density regions. The ascending weights (middle panel at left) that result when the synaptic weights have equilibrated are decidedly non-Fourier, and are well adapted to the activity correlations among the input units that arise from the bimodal distribution. Rather than being flat, the first component is bimodal with troughs aligned with the centers of the regions where input coverage is less dense. The next two components are also quite non-sinusoidal with dimpled peaks aligned with the central low-density region. Nevertheless, there is a doubling of frequency between the second and third components similar to what is seen in the Fourier-like case. Another frequency doubling can be seen between the fourth and fifth components. The bottom left panel of figure 4.6 shows that the Oja rule sets the descending weights proportional to the ascending weights in this case as well.

Similar results are seen in the right column of figure 4.6 for a unimodal distribution of preferred stimulus angles, with the central region having a density twice that of the surrounding region. Again the baseline shift of the first com-

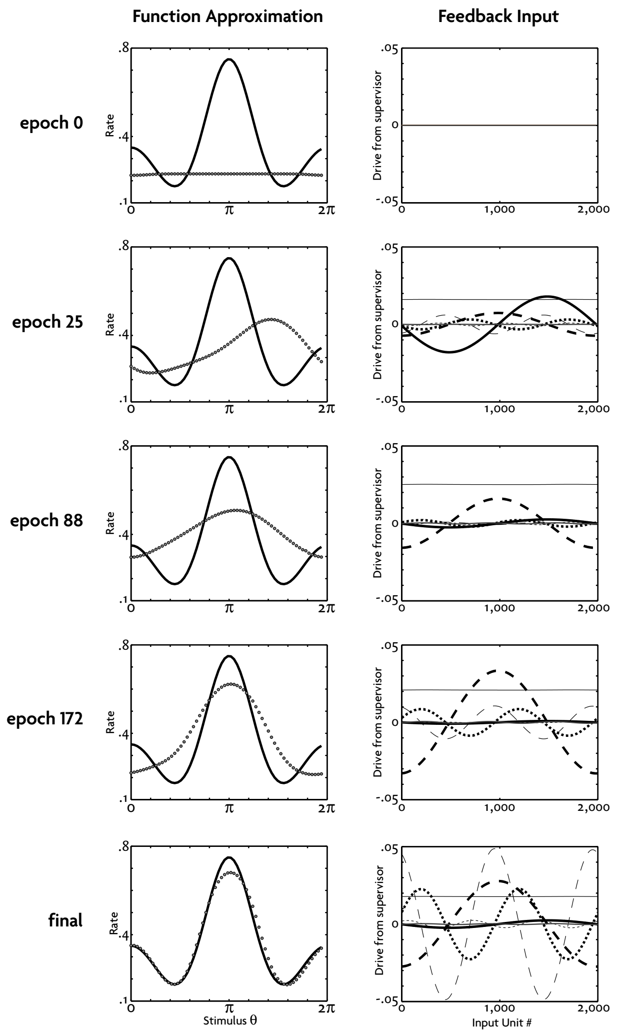

Figure 4.4: Dimensionally reduced reinforcement learning. The left column shows the target function (black line) and the output of the network for various values of the stimulus angle (grey dots). The right column depicts the bias currents sent to the network by the supervisor as a function of the input unit number, with each supervisory unit represented by a separate line. The total bias current received by each input cell can be determined summing the heights of theses curves. Rows show results after 0, 1600, 5632, 11,008, and 30,270 trials. As the the bias currents develop through the reinforcement-based random walk procedure, the initially poor approximation (top row) changes into a fairly accurate approximation of the target function (bottom row).

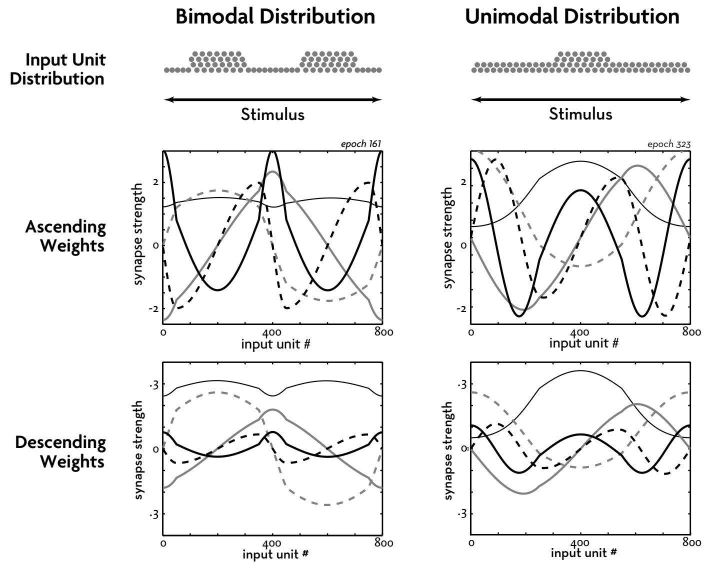

Figure 4.5: Ascending and descending weights for nonuniform distributions of preferred stimulus angles. When the preferred stimulus angles of the input units are not uniform, the components extracted for the ascending units by the Sanger rule (middle panels) and transferred to the descending weights by the Oja rule (bottom panels) are non-Fourier. The top row shows the two alternative distributions used.

ponent is distorted, reflecting the non-uniformity in the input population. The second and fourth components are centered around the high-density peak and are frequency-doubled versions of each other, whereas the third and fifth components are more sine-like but also show frequency-doubling.

Modulation Compensates for Sampling Bias

While the basis functions established by the Sanger rule in these non-uniform cases seem appropriately well suited to the input population activity statistics, it is unclear that they represent a sufficient basis set to modulate the network for the function approximation task. Also, the non-uniformity of the sampling places an additional burden on the network since, due to the equal synaptic strengths between the input units and the output, in its default state the network's output will have shifted from a flatline response to one which mirrors the sampling distribution. Thus if the distribution over-emphasizes regions in which a low firing rate is desired, the supervisor must not only move the net-

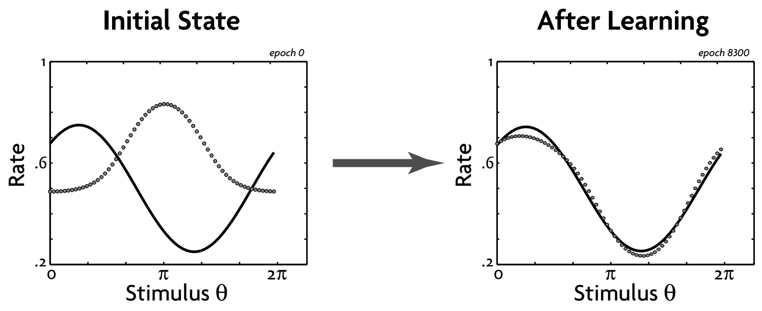

Figure 4.6: Function approximation with a non-uniform sampling distribution. (a) When the sampling density of the input cells is not constant, the initial response of the output unit depends on the stimulus angle. In this instance, a higher density of input units near $\theta = \pi$ causes a larger output response in that region. (b) Nevertheless, a reward-guided random walk procedure, dimensionally reduced with the appropriate principal components, produces a successful approximation or the target function.

work from a neutral state to a successful one, but actually undo an intrinsic bias which impedes the task.

Figure 4.7a shows the initial state for a network with the unimodal sampling distribution seen in figure 4.6. Although the input units all respond to their preferred stimuli identically and the synaptic weights from the input units to the output unit all take the same value as in figure 4.4, the output unit response depends on the stimulus angle. The response is largest for stimuli near the middle of the range because that is where the largest number of input units are respond. As a result of this, the initial approximation is significantly worse than it would be in the uniform-distribution case, and the supervisor must overcome this bias to fit the target function. Nevertheless, as can be seen in Figure 4.7b, once the reward-guided random walk sets activity of the superisor units properly, the network successfully approximates the target function. Thus, the synaptic strength distributions extracted by the Sanger process and communicated to the network through Oja plasticity provide a general solution for constructing an efficient supervisor strategy regardless of the specific input population statistics.

SCALABILITY AND PERFORMANCE IMPROVEMENT

The primary motivation for using the Sanger rule-based PCA approach to supervision has been biological plausibility. By requiring less state and reducing dimensionality to the point that a random walk strategy will yield sufficiently high performance, the circuit has reached a level of simplicity such that it could arguably be implemented with known biological mechanisms. However, if this plausibility were to come at the expense of performing adequately to reproduce the results seen with the more memory- and algorithmically-demanding approach of the traditional, direct-modulation supervisor, the argument becomes less compelling. To address this possibility, we examined the performance of the PCA supervisor in a more rigorous manner, and compared its results to those of the traditional supervisor.

Effects of Input Population Size

Thus far we have emphasized the benefits of dimensional reduction via PCA in terms of simplifying the task of the supervisor and improving the speed with which it can converge upon a solution as compared to a random walk where the shifts and gains are set independently for each of the N input units. A critical advantage of the dimensional reduction implemented by the combination of Sanger and Oja rules is that the number of supervisor units is independent of the number of input units. This means that the poor scaling behavior of the direct reward-based random walk approach as a function of network size is replaced by a scheme in which convergence time is essentially independent of network size.

While at small population sizes the performance difference between the two algorithms is merely one of degree, as the population size increases it quickly becomes the difference between success and failure. That this should be true is fairly intuitive: the traditional supervisor has two dimensions in its search space for each input unit, thus each time the population size doubles, the dimensionality of search is similarly doubled. While in theory this should only slow the learning process down, in practical terms it is equivalent to the supervisor being unable to solve the task for sufficiently large Ns. Unfortunately, as can be seen in figure 4.8, the critical N values are not particularly big.

Error values after different numbers of trials for the direct learning process are plotted as dotted lines in figure 4.8. The direct scheme performs acceptably for moderate input population sizes (e.g., 2,000 units), but once there are 4,000 or more input units, the direct approach fails to converge over the number of trials shown. Although the random walk algorithm ensures that the supervisor will change any one strategy that increases its error, it may make a serious of changes to strategies that continue to be detrimental, resulting in the prolonged

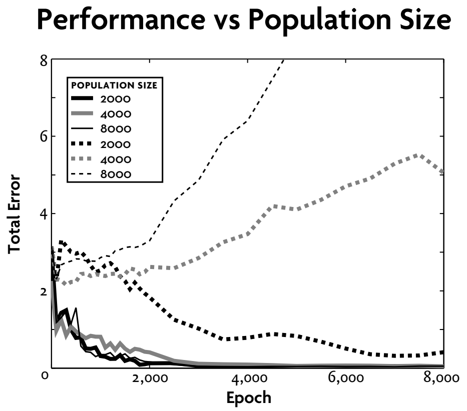

Figure 4.7: Scaling of dimesionally reduced and direct reward-based random walk learning strategies with network size. Total function approximation error is plotted over the course of learning trials for small, medium, and large networks. Dotted lines show the error for the direct random walk strategy, which increases dramatically as the network size increases. Solid lines correspond to the dimensionally reduced scheme and are fairly invariant with respect to population size, with no substantial difference between the different conditions.

increases in error magnitude seen in figure 4.8 for the 4,000- and 8,000-unit cases.

The PCA approach, on the contrary, is effectively invariant with respect to the input population size (solid lines in figure 4.8). In this scheme, the burden of dealing with more input units is handled by the Sanger/Oja learning procedure on the connections between the input and supervisor units prior to any function learning. Though there seems to be slightly more noise in the early stages of learning when the population size is larger, overall the solid traces in figure 4.8 are virtually overlapping. This is not surprising because the dimension of the parameter space being searched is independent of network size in the dimensionally reduced case, and the principal components have the same shape over the input distribution independent of its size. Thus the PCA supervisor is capable of scaling to much larger population sizes than the direct supervisor, and it does so with no noticeable performance degradation during the learning phase.

Number of Components and Diminishing Returns

Since learning by way of the PCA-derived modulation patterns is essentially equivalent to approximating a waveform by scaling Fourier modes, it stands to reason

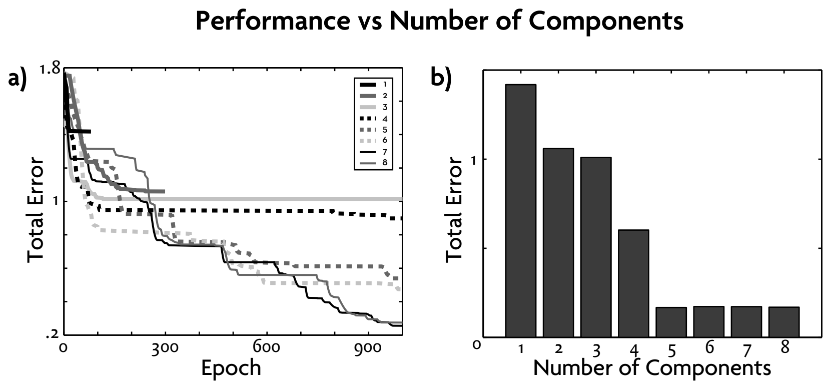

Figure 4.8: Improvements realized by increasing the number of components. Adding additional principal components (and thus additional cells) to the supervisor is an effective means for improving performance. However, the effect saturates rather quickly. The left panel shows error traces corresponding to supervisors with differing numbers of components. Predictably, a larger number of components yields, given time, a lower final error. The right panel depicts the final error reached in each of these conditions when the simulation is allowed to run until improvement stops. In this case it appears that using any more than five components is unnecessary, though using more appears to get the network to its final state more quickly.

that having a greater number of basis functions from which to draw will lead to a more accurate final approximation. With two or three bases a rough correspondence can typically be established between target and approximation, but to capture the character of the target function additional, higher frequency, components may be needed. This pattern can be seen in the traces of figure 4.9a which depicts the error over time for networks learning with a varying number of components represented in the supervisory circuit.

Two noteworthy effects are on display here. The first is that using a smaller number of components leads to a faster convergence toward the optimal solution. For example, in the single-component case, the supervisor finds its optimal (though still quite poor) approximation in under 50 epochs, while the eight-component supervisor requires nearly 20 times that period to reach its more globally optimal solution. Second, increasing the number of components does in fact reduce the error in the ultimate approximation reached through learning as can be seen by the steady decreases in the higher cell-count cases.

However, this effect tends to saturate, as can be seen more clearly in figure 4.9b. Here only the final error values are plotted for each number of components used in the learning. Clearly adding more supervisory units improves performance up to a point, but that point is surprisingly low, in this case showing that only five components are truly necessary to obtain a minimal error rate.

Of course, the number of components used is dependent on the target function being approximated. Were it less complicated and more easily approximated by a pair of low-frequency sinusoids, improvement would stop after the third component. If the target function varies quickly with respect to the stimulus value, more than five components would be needed to capture this higher-frequency information.

Including Hebbian Modification of Network Synapses

The most obvious benefit of reducing the dimensionality of the space that the supervisor must search is that it allows such a simple search algorithm to work. An additional benefit, however, is that it allows a reinforcement-based, random walk strategy to bring the network to a successful state sufficiently quickly to guide Hebbian synaptic plasticity within the network itself. The slower speed with which the traditional supervisor guides learning is actually not a matter of degree but in practical terms is equivalent to failure. The justification for this claim is clearest when one considers the problem of transferring the supervisor's learning to the network's internal synapses. According to our general scheme thus far, the connections between the input units and the output unit (or units) are static throughout learning. However as we have demonstrated previously in chapter 3, by making these synapses plastic and having them obey the Hebbian Oja rule, the learning which initially occurs within the supervisor as it establishes the activity patterns for success through its modulatory input sets the conditions for Hebb to reinforce those activity patterns, transferring the correlations to the synaptic strengths. We now allow connections to be modified by an Oja plasticity rule during the learning process,

$$w_i \to w_i + \eta'' R\,(r_i - R w_i)\ , \qquad (4.12)$$

with η'' determining the learning rate.

These two processes—the supervisor homing in on a proper modulation pattern and the intra-network synaptic weights equilibrating around the imposed activity pattern—may proceed in parallel, however this demands a certain degree of speed and accuracy from the supervisor. Since the Hebb rule is purely local and does not take into consideration how well the task is being performed, the activities leading to a bad approximation will be stored just as readily as a good one. The implications of this can be seen in the comparison of learning between traditional and PCA supervision in figure 4.10.

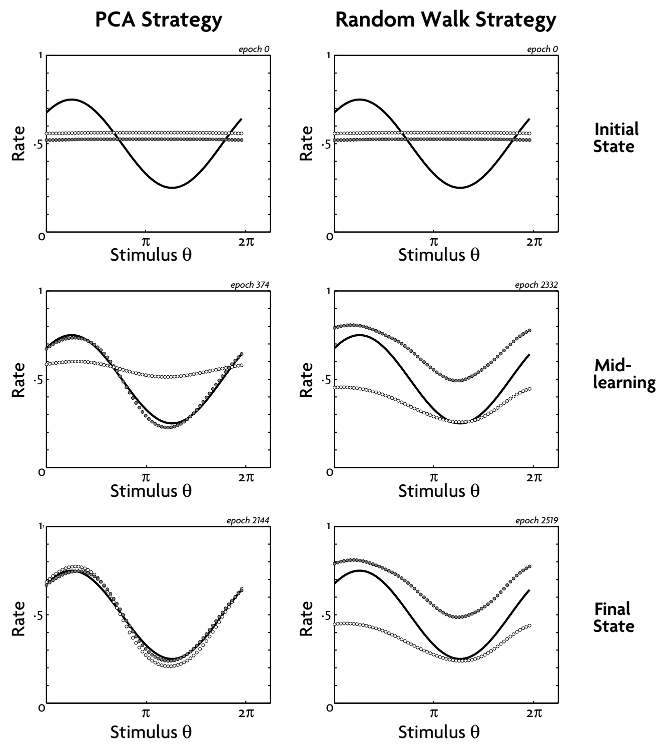

Figure 4.9: Learning with Hebbian plasticity. When the synapses between the input units and output unit are allowed to vary according to the Oja rule, the correlations put in place by the supervisor through modulation can be transferred to the network weights. When this is successful, as in the left column, the supervisor's modulatory input can ultimately be removed and the network will continue to perform appropriately. To visualize this, the network's modulated responses are plotted as grey dots, and the response resulting solely from the network (with no modulation) as white dots. In the PCA case, these two lines grow to overlap over the course of learning, showing that by the end the supervisor's input is no longer necessary to do the task. In the case of the random walk strategy, the supervisor is unable to quickly reach a modulated solution. Thus the synapses begin storing the poor approximation, further complicating the supervisor's task.

In these plots, the modulated response of the network to a given stimulus is plotted in grey, while its response with modulation removed (i.e., the purely intra-network synaptic response) is plotted in white. By the second row the Hebbian effects have begun to result in changes to the network as can be seen by the inflections in the white plot lines. In the case of the PCA supervisor, the network is already succeeding at the task as can be seen by the nice agreement between the target function and the grey plot line. Thus one can assume that the Hebbian learning happening within the network at this point is transferring this successful representation to the synapses.

In the case of the non-dimensionally-reduced random walk supervisor, the story is quite different. Because of its difficulty with populations of this size, it has still not achieved a particularly successful approximation of the target function. Thus the synapses of the network are now learning on a permanent basis how to reproduce the *poor* approximation. As the process continues, there is additional 'drag' on the supervisor as it must not only undo its own errors in deciding on a modulation pattern, but also counteract the network's intrinsic maladaptive behavior brought about by early mistakes already transferred to the network synapses.

Thus by the final row, the PCA supervisor has managed to adapt its own modulation as the synapses have accounted for more and more of the behavior to the point where the 'modulated' and non-modulated states are more or less identical—suggesting that the degree of actual modulation going on at this point is quite minimal. In the case of the traditional supervisor, the process has broken down completely, it seems to be stuck in the same position as before, with the poor-performing modulated activity pattern giving rise to an equally poor synaptic response, with each working to the other's detriment.

DISCUSSION

In this chapter we have laid out a scheme by which an external supervisor circuit could indirectly guide synaptic learning solely through harnessing random drift in a noisy network and sending fast excitatory signals to the learning network. However it is still an open question whether such a process actually occurs in biological networks. As mentioned previously there is good experimental evidence for the types of bump states that could power the search process within the supervisor circuit. But what of the stronger claim that the output of such a supervisory circuit could influence network evolution through the injection of modulatory excitation and inhibition?

In fact a pair of recent papers examining the avian song production system raise suggest that just such a scheme may be involved in rapid learning and fine-tuning of learnt songs. There are two elements to our conjecture which have been observed in these studies:

1. The imposition of random variation in the network which could then be used as grist for a reinforcement learning-style selection process.

2. A greater structure to the degree of noise imposed, gated by the produced behavior's similarity to a desired behavior.

Both studies examine the role in learning of a basal ganglia nucleus called LMAN which is known to innervate the vocal motor pathway. The researchers discovered that when LMAN is silenced, the degree of variability in the song produced by the learning bird is greatly reduced (Ölveczky et al, 2005). In addition there is some evidence that the LMAN signal guides song production in a more directed way: eliciting a specific pattern of LMAN activity could reliably lead to production of an equally specific variation in the song (Kao et al, 2005). These results suggest that LMAN in birds or the basal ganglia in mammals may be the site of the biological correlate of the supervisor circuit we have been discussing.

The one missing piece is the form of modulation used. Projections from LMAN are predominantly excitatory, terminating in NMDA synapses on the motor pathway. This could account for the shift-based modulation discussed in this chapter, but not for the gain modulation via balanced excitation and inhibition postulated previously. However it has recently been suggested that LMAN activity is correlated with a concomitant rise in inhibition within the target nucleus (Kristofer Bouchard, personal communication), raising the possibility that a supervisory circuit posessing only excitatory outputs could be capable of the full range of modulation proposed in the more complicated networks of prior chapters.

Introducing variability is one essential feature of the supervisor we have been studying because it corresponds to exploration of the space of network parameters. Another is modulation of that variability by reward. It will be interesting to see if LMAN is modulated by reward signals, such as dopamine, in a manner consistent with the proposed scheme.

CHAPTER FIVE

Connected Dots & Dangling Ellipses

Over the course of the preceding chapters, we have attempted to lay out a coherent scheme by which a 'controller' circuit (whose biological analog would be some high-level, planning-oriented portion of the brain such as prefrontal cortex) could influence the behavior of a lower-level network on short time scales and in such a way as to lead to permanent changes.

The mechanism by which the controller exerts this influence, as discussed in chapter 2 takes the form of gain- and shift-modulation, which we have demonstrated are capable of eliciting a wide variety of network activity patterns. In addition, experimental data has shown that this form of control can be realized in biological systems, suggesting that its use in the manner we have proposed is at least achievable, if not yet directly observed.

While the existence of this phenomenon is noteworthy, how it could be used in a consistent manner in a realistic network is not immediately apparent. Thus in chapter 3 we examined its utility in a supervised learning context when applied to the classical neural network task of function approximation. The combination of external supervision through modulation with local synaptic changes through Hebbian plasticity addressed a potential concern which arose from the demonstrations of chapter 2: the necessity of continuing supervisory input throughout performance of the task.

The introduction of local plasticity allowed learning to shift away from the constructed, 'black box' supervisor, instead becoming intrinsic to the network itself after a period of training. Thus the role of the supervisor became a transient one, setting up the conditions for a learning that it did not directly control, and from which it could completely remove itself once complete.

All the same, the amount of the task that was solved within the black box was sizable, with the hand-built supervisor keeping track of shift and gain modulation values for the several hundred cells of the network, then using a computationally intensive gradient descent algorithm to fine tune those values as it minimized performance error in the task. Thus, one could fairly argue that the 'hard' problems of learning had simply been pushed back from learning at the synapses (where they are typically handled) to learning within this artificial supervisor.

More concretely, while we had shown how an arbitrary control circuit could affect a network to bring about a desired change in both behavior and connectivity, we provided no explanation for how such a circuit could be constructed, how it would systematically manage its modulatory feedback to the network, or how its search algorithm could be realistically implemented. The work of chapter 4 attempts to address these issues and complete the loop from task to network to control circuit and back again.

Our final supervisor consists of only a handful of cells with the same characteristics as those in the network itself, and the imposing problem of differentially sending modulatory input within that population was solved by decomposing the problem into principal components—each corresponding to an individual cell in the supervisory circuit—and allowing their connections to the network to be set by a variant of the Generalized Hebbian learning rule. We have also shown that by composing the controller circuit in this manner, the sub-problem of optimizing the firing rates of the supervisory cells (thus indirectly optimizing the modulatory input to the network) is simple enough that it can be accomplished even when guided by a random walk strategy tuned to a non-specific reward signal.

REFINING THE SUPERVISION SCHEME

While performance of the current PCA-based supervisor is sufficient as a proof of concept, it could be improved; albeit at the cost of adding complexity to the model. As it stands, the feedback modulation to the network is either purely excitatory or purely inhibitory on a cell-by-cell basis. This is in contrast to the type of modulation used in chapters 2 and 3 in which each network cell received both excitatory and inhibitory input simultaneously. The advantage of this dual-input approach is that it allows for not only the kind of shift modulation used by the PCA supervisor but also the form of gain modulation that can be achieved through balanced input currents.

The fact that the supervision scheme performs adequately using just shift is heartening, but it could surely do better were gain modulation at its disposal as well. This could most trivially be achieved by doubling the number of cells in the supervisory circuit, with one set responsible for excitatory feedback and the other for inhibitory.

Gain modulation would be especially useful for approximating target functions with high spatial-frequencies given the sharpening and broadening of the input tuning curves that is possible through this mechanism. The PCA scheme already has one method of dealing with high-frequency targets by taking advantage of the supervisory cells corresponding to higher-order principal components. However, the ability to modulate neighboring input units differently is diminished if those cells' receptive fields are so broad as to completely overlap. Thus sharpening through gain modulation could make these high-order supervisory cells more useful.

Beyond purely performance-oriented improvements, there is also the question of economy. In the same way that gain modulation is only truly useful for fitting steep variations in the target function (and thus is overkill for functions lacking those rapid changes), the number of principal components needed to successfully perform the task will also be dependent on the structure of the target. Thus it would make sense to recruit the cells of the supervisory circuit as needed rather than simply throwing a fixed number of them at the problem, leading to resources being wasted in some cases and being insufficient in others. As a side benefit, this approach could also lead to faster learning since by minimizing the number of components necessary, it also minimizes the dimensions of the modulatory search space that must be traversed to solve the task.

REALIZING THE BLACK BOX

The supervisory scheme just described is certainly an improvement in terms of parsimony and biological plausibility when compared to the purely black box approach of using an omniscient gradient descent algorithm and directly setting shifts and gains for each cell independently. However, while we have reduced the degree to which the supervisor relies on non-neural implementations, some portions of its behavior are still unaccounted for.

Most prominently, the manner by which the rates of the cells in the supervisory circuit are fine tuned during learning, and the source of the driving input to maintain these rates, has not been identified. This is particularly critical since 'solving' the function approximation problem under this scheme collapses into setting these rates.

Luckily, precisely the qualities that are required for this process—slow drift within some space, gated by an external value (in our case approximation error) and sustained activity—are the definitive characteristics of another circuit which has received much attention from the neural modeling community: prefrontal cortex. Currently the leading candidate as a locus for such behaviors as planning and attending, PFC has a number of characteristics that make it well suited to a the meta-task of supervising the supervisor during task-learning.

First, it has been shown that PFC, like many other parts of the brain, is capable of maintaining stable patterns of activity for extended periods, considered by some to be the analog for a form of working memory or at the least focused attention. This type of phenomenon is precisely what would be be required to maintain the relative drives of the cells in our supervisory circuit over time.

In addition, a mechanism for these sustained activity states has been proposed in recent theoretical work (Wang and Goldman-Rakic, 2004; Compte et al., 2000; Seung et al., 2000). By setting connectivity patterns within a PFC-like network as a sort of topographic map—with 'nearby' nodes having strong excitatory connections, and more 'distant' ones having inhibitory couplings—one can achieve stable representations within the activity pattern of the network. Once a particular location within this ensemble is activated, a 'bump' of excitation will spread to the most proximal cells, but this activity will remain localized due to the inhibition passed to more distal cells.

In order to integrate such a mechanism into our scheme, one could imagine that in addition to being connected to the other cells within this map, the cells in this PFC-like network would also send projections to the cells in our supervisory network with synaptic couplings that would vary based on their position within the map. Thus the location of the bump would correspond to one set of synaptic drives delivered to the supervisory cells.

The second important characteristic of these bump attractor networks is that once a bump has formed, it is subject to a degree of random drift due to noise in the local spread of activation within the excitatory halo. While this is generally considered a defect when used in working memory models (since it undercuts the stability of the representation, allowing it to change despite the lack of external justification for doing so), for our purposes this 'bug' becomes a 'feature'. For the black box in our model was not merely responsible for maintaining drive to the supervisory cells, but also for experimenting with their relative rates in order to minimize error in the function approximation task. Thus, in our case, the drift is not noise to be minimized but in fact provides a mechanism for exploring the modulation space of the network by randomly varying the pattern of activity imposed on the network by the supervisory circuit.

This is why it is significant that, as demonstrated in chapter 4, the task-learning process can proceed successfully when guided by a random walk supervisory strategy. Thus one of these bump networks could potentially function as a drop-in replacement for our more contrived, algorithmic black box. The one missing piece complicating such a replacement is the lack of coupling between task error, as our hand-built supervisor received, and the random drift of the bump network. However this too is fairly easily dealt with, for another quality intrinsic to PFC is reward-processing via dopamine signaling (Schultz, 1998).

If one were to make the speed of drift a function of task-related reward—increasing when reward is low and the network is performing poorly, decreasing to near zero when the network is performing near-optimally—no additional modification would need to be made to this meta-supervisory system for it to

be equivalent to the PCA-based model proposed above. Since the speed of drift is a function of network noise, this could easily be modulated as a function of reward through a simple, global elevation or decrease in input from the circuit managing the reward process.

References

Amit, DJ (1989) *Modeling Brain Function*. New York: Cambridge University Press.

Barlow, HB (1989) Unsupervised learning. *Neural Computation* **1**:295–311.

Barto AB, Sutton RS, Anderson CW (1983) Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Trans on Systems, Man and Cybernetics* **13**:834–846.

Bliss T.V., and Collingridge, G.L. (1993). A synaptic model of memory: long-term potentiation in the hippocampus. *Nature* **361**, 31–39.

Bredt D.S., and Nicoll, R.A. (2003). AMPA receptor trafficking at excitatory synapses. *Neuron* **40**, 361–379.

Cauwnberghs G (1993) A fast stochastic error-descent algorithm for supervised learning and optimization. *Adv Neural Info Proc Sys* **5**:244–251.

Chance FS, Abbott LF, Reyes, AD (2002) Gain modulation from background synaptic input. *Neuron* **35**: 773–782.

Chauvin, Y, Rumelhart, DE, eds. (1995) *Back Propagation: Theory, Architectures, and Applications*. Hillsdale, NJ: Erlbaum.

Compte A, Brunel N, Goldman-Rakic PS, Wang X-J. 2000. Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cereb. Cortex* **10**:910–923.

Dayan, P. and Abbott, L.F. (2001) Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. (MIT Press, Cambridge MA).

Dickinson, A (1980) *Contemporary Animal Learning Theory*. Cambridge: Cambridge University Press.

Doiron B, Longtin A, Berman N, Maler L (2001). Subtractive and divisive inhibition: effect of voltage-dependent inhibitory conductances and noise. *Neural*

Computation **13**:227–248.

Doya K, Sejnowski TJ (1995) A novel reinforcement model of birdsong vocalization learning. *Adv Neural Info Proc Sys* 7:101–108.

Gabriel, M, & Moore, JW, eds. (1990) *Learning and Computational Neuroscience*. Cambridge, MA: MIT Press.

Gallistel, CR (1990) *The Organization of Learning*. Cambridge, MA: MIT Press.

Hebb, DO (1949) *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley.

Hertz, J, Krogh, A, & Palmer, RG (1991) *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley.

Hinton, GE (1989) Connectionist learning procedures. *Artificial Intelligence* **40**:185–234.

Jabri M, Flower B (1992) Weight perturbation - an optimal architecture and learning technique for analog vlsi feedforward and recurrent multilayer networks. *IEEE Trans on Neural Networks* 3:154–157.

Jolliffe, IT (1986) *Principal Component Analysis*. New York: Springer-Verlag.

Kao, MH, Doupe, AJ, & Brainard, MS (2005). Contributions of an avian basal ganglia-forebrain circuit to real-time modulation of song. *Nature* **433**:638–643.

Kearns, MJ, & Vazirani, UV (1994) *An Introduction to Computational Learning Theory*. Cambridge, MA: MIT Press.

Koshland, D (1980) Bacterial Chemotaxis As a Model Behavioral System (Distinguished lecture series of the Society of General Physiologists ; v. 2) Raven Press.

Lukashin AV, Wilcox GL, Georgopoulos AP (1994) Overlapping neural networks for multiple motor engrams. *Proc Natl Acad Sci USA* **9**:8651–8654.

Mazzoni P, Andersen RA, Jordan MI (1991) A more biologically plausible learning rule for neural networks. *Proc Natl Acad Sci USA* **88**:4433–4437.

Miller, KD (1996) Receptive fields and maps in the visual cortex: Models of ocular dominance and orientation columns. In E Domany, JL van Hemmen, & K Schulten, eds., *Models of Neural Networks, Volume 3*, 55–78. New York: Springer-Verlag.

Minsky, M, & Papert, S (1969) Perceptrons. Cambridge, MA: MIT Press.

Mitchell SJ, Silver RA (2003) Shunting inhibition modulates neuronal gain during synaptic excitation. *Neuron* **38**: 433–443.

Montague, PR, Dayan, P, & Sejnowski, TJ (1996). A framework for mesencephalic dopamine systems based on predictive hebbian learning. *J. Neurosci.* **16**:1936–1947

Murphy B, Miller K (2003) Multiplicative gain changes are induced by excitation or inhibition alone. *J Neurosci* (in press).

Narendra, KS, & Thatachar, MAL (1989) *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall.

Oja, E. (1989), Neural networks, principal components, and subspaces, *International Journal of Neural Systems* **1**, 61–68.

Ölveczky, BP, Andalman, AS, & Fee, MS (2005). Vocal experimentation in the juvenile songbird requires a basal ganglia circuit. *PLoS Bio.* 3(5):e153

O'Reilly, RC (1996) Biologically plausible error-driven learning using local activation differences: The generalised recirculation algorithm. *Neural Computation* **8**:895–938.

Poggio T (1990) A theory of how the brain might work. *Cold Spring Harbor Symposium on Quantitative Biology* **55**:899–910.

Prescott SA, De Koninck Y (2003) Gain control of firing rate by shunting inhibition: roles of synaptic noise and dendritic saturation. *Proc. Natl. Acad. Sci. USA* **100**: 2076–2081.

Rosenblatt, F (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* **65**:386–408.

Rumelhart, DE, & McClelland, JL (1986) Parallel Distributed Processing. Cambridge, MA: MIT Press.

Salinas E, Abbott, LF (2000) Do simple cells in primary visual cortex form a tight frame? *Neural Comp* **12**:313–336.

Sanger, T.D. (1989), "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks* **2**, 459–473.

Schultz, W (1998) Predictive reward signal of dopamine neurons. *Journal of Neurophysiology* **80**:1–27.

Schultz W, Dayan P, Montague PR (1997) A neural substrate of prediction and reward. *Science* **275**:1593–1599.

Schultz W. (2002). Getting formal with dopamine and reward. *Neuron* **36**:241–263.

Seung S (2003) The hedonistic synapse. (unpublished).

Seung HS, Lee DD, Reis BY, Tank DW. 2000. Stability of the memory of eye position in a recurrent network of conductance-based model neurons. *Neuron* **26**:259–271.

Shadlen MN, Newsome WT. 1994. Noise, neural codes and cortical organization. *Curr. Opin. Neurobiol.* **4**:569–579.

Sutton, RS, & Barto, AG (1998) *Reinforcement Learning*. Cambridge, MA: MIT Press.

Troyer TW, Miller KD. 1997. Physiological Gain Leads to High ISI Variability in a Simple Model of a Cortical Regular Spiking Cell. *Neural Comput.* **9**:971–983.

Wang X-J, Goldman-Rakic, P, eds. (2004) Special issue: Persistent neural activity: experiments and theory. *Cereb Cortex* **13**:1123–1269.

Widrow, B, & Hoff, ME (1960) Adaptive switching circuits. *WESCON Convention Report* **4**:96–104.

Widrow, B, Stearns, SD (1985) *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8**:229–256.

Xie X, Seung S (2003) Learning in neural networks by reinforcement of irregular spiking. (unpublished).