Neural Network Models for Zebra Finch Song Production and Reinforcement Learning

by

Justin Werfel

A.B. Physics
Princeton University, 1999

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2001

Author .....
Department of Electrical Engineering and Computer Science
August 10, 2001

Certified by .....
H. Sebastian Seung
Robert A. Swanson Career Development Assistant Professor in Life Sciences
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Neural Network Models for Zebra Finch Song Production and Reinforcement Learning

by

Justin Werfel

Submitted to the Department of Electrical Engineering and Computer Science
on August 10, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

The zebra finch is a standard experimental system for studying learning and generation of temporally extended motor patterns. The first part of this project concerned the evaluation of simple models for the operation and structure of the network in the motor nucleus RA. A directed excitatory chain with a global inhibitory network, for which experimental evidence exists, was found to produce waves of activity similar to those observed in RA; this similarity included one particularly important feature of the measured activity, synchrony between the onset of bursting in one neuron and the offset of bursting in another. Other models, which were simpler and more analytically tractable, were also able to exhibit this feature, but not for parameter values quantitatively close to those observed.

Another issue of interest concerns how these networks are initially learned by the bird during song acquisition. The second part of the project concerned the analysis of exemplars of REINFORCE algorithms, a general class of algorithms for reinforcement learning in neural networks, which are on several counts more biologically plausible than standard prescriptions such as backpropagation. The former compared favorably with backpropagation on tasks involving single input-output pairs, though a noise analysis suggested it should not perform so well. On tasks involving trajectory learning, REINFORCE algorithms meet with some success, though the analysis that predicts their success on input-output-pair tasks fails to explain it for trajectories.

Thesis Supervisor: H. Sebastian Seung
Title: Robert A. Swanson Career Development Assistant Professor in Life Sciences

Acknowledgments

Thanks must first and foremost go to my advisor, Sebastian Seung, who introduced me to this area of research, and provided invaluable help and guidance throughout the course of the work.

I would also like to thank the other members of the Seung Lab for interesting discussions and comments, in particular Ben Pearre, who made the computers work when they most needed to.

This research was supported by a National Defense Science and Engineering Graduate Fellowship from the U.S. Department of Defense, with additional support from the David and Lucile Packard Foundation.

Contents

1 Introduction **9**
- 1.1 Neurobiology of the zebra finch . . . 9
- 1.2 Computational learning . . . 10

2 Modeling RA **13**
- 2.1 The model . . . 15
- 2.2 Simulations . . . 15
  - 2.2.1 Forward excitation, reverse inhibition . . . 16
  - 2.2.2 Forward excitation, global inhibition . . . 17
  - 2.2.3 Excitation only . . . 17
- 2.3 Analysis . . . 21
- 2.4 Training the network . . . 23

3 Network reinforcement learning **27**
- 3.1 The model . . . 27
- 3.2 Noise analysis . . . 29
- 3.3 Weight perturbation . . . 30
- 3.4 Simulations . . . 31
  - 3.4.1 Batch vs. online update . . . 31
  - 3.4.2 Comparison of learning curves . . . 34
  - 3.4.3 Noise analysis . . . 36
  - 3.4.4 Autoencoding . . . 39
- 3.5 Trajectory learning . . . 41
- 3.6 Discussion . . . 42

A Derivations **45**
- A.1 The basic REINFORCE result . . . 45
- A.2 Eligibility for continuous-valued units with Gaussian noise . . . 46
- A.3 Weight adjustment in the low noise limit . . . 46
- A.4 Extension to trajectory learning . . . 48
- A.5 Noise analysis . . . 49
- A.6 Weight perturbation . . . 51

List of Figures

1-1 Neuroanatomy of the zebra finch song learning and production system . . . 10

2-1 Recordings from nearby neurons in RA during singing 14
2-2 Model network with forward excitation and reverse inhibition 16
2-3 Model network with forward excitation and global inhibition 18
2-4 Model network with forward excitation only 19
2-5 Forward excitation only, three spikes per burst 20
2-6 Multiple-state spike trains impossible with identical units 23
2-7 The synaptic triad, and its use to implement the Perceptron learning rule . 24

3-1 Sketch of the noise concern in stochastic gradient descent 29
3-2 Learning curves for backpropagation with added noise 33
3-3 Learning curves for training algorithms on digit classification 34
3-4 Learning as progress through error space 35
3-5 SNR as a function of number of noise sources (I) 37
3-6 SNR as a function of number of noise sources (II) 38
3-7 Learning curves for autoencoding 40
3-8 Time course of trajectory learning 42

Chapter 1

Introduction

In the field of motor learning, one issue of particular interest is that of learning to produce a prescribed sequence or trajectory. Active areas of research falling under this heading include legged locomotion, arm movement control, and speech production.

One common experimental system for the investigation of sequence and motor learning is the zebra finch, *Taenopygia guttata*. The songs these birds produce are consistent and reproducible, relatively simple as compared to many other songbirds, but possess a complex acoustic structure that requires the coordination of the vocal musculature and the memorization of extended temporal sequences of neural activity. Birdsong is of particular interest as a system which involves elements also present in human speech, but which is simpler and more experimentally tractable. Song learning and production in the zebra finch has been relatively well characterized both behaviorally and physiologically.

1.1 Neurobiology of the zebra finch

Like other songbirds, finches memorize their song by listening to the vocalizations of tutor birds, during a critical sensory period [3, 7, 8]. Birds isolated during this period will produce only fragmented and simple vocalizations. Later, during a sensorimotor phase, the birds learn to produce the memorized song, beginning with babbling and gradually learning to refine their vocalizations to match the desired result. The sensory and sensorimotor phases overlap for the zebra finch, though for many songbirds they are distinct.

Recordings have been made from several areas of the brain associated with song learning and production, and the behavior of those areas characterized. These include the premotor nucleus HVc, whose activity is correlated with individual syllables of the song; RA, which receives input from HVc and projects to motor areas which in turn project to the syrinx, the sound-producing organ; and several areas in what is collectively termed the anterior forebrain pathway (AFP), which receives input from HVc and projects to RA, and which has been implicated in learning. Figure 1-1 gives a schematic diagram.

The AFP is necessary to young birds for them to learn their song; if it is lesioned in adult birds, however, the previously-learned song remains intact. If an adult bird is deafened, so that it no longer receives auditory feedback about the sounds it produces, its song will gradually degrade over time. However, if the AFP is lesioned in addition to the deafening, the song remains intact. These results suggest that the AFP carries an error signal, representing some comparison between the auditory feedback and the memorized song [3]. Similarly suggestive are recordings showing that neurons in various structures of

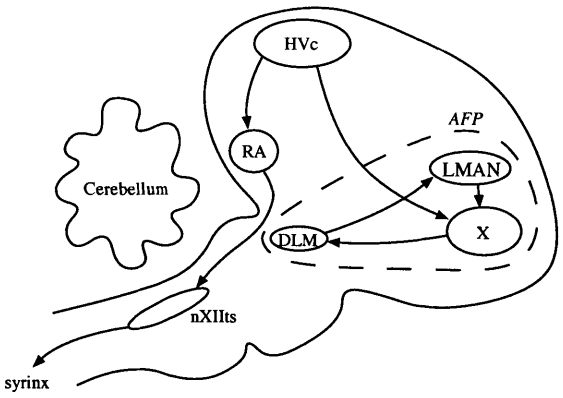

Figure 1-1: Neuroanatomy of the zebra finch song learning and production system. HVc, nucleus hyperstriatalis ventrale pars caudale. RA, nucleus robustus archistriatalis. AFP, anterior forebrain pathway, consisting of areas LMAN, lateral portion of the magnocellular nucleus of the anterior neostriatum; DLM, medial portion of the dorsolateral nucleus of the thalamus; and Area X. nXIIts, tracheosyringeal portion of the hypoglossal nucleus.

the AFP respond very selectively to auditory inputs representing sequences of syllables from the bird's song, and not to individual syllables, syllables in the wrong order, temporally reversed inputs, or other such variations; and the response is stronger for the bird's own song than for those of other individuals of the same species [7, 8].

Zebra finch song is hierarchically structured, with a *song* consisting of *motifs* consisting of *syllables* consisting of *notes*; syllables are separated by brief periods of silence [26]. The activity of HVc neurons is correlated with individual syllables of the bird's song, with firing rates modulated on the order of threefold from one syllable to another. There appear to be two separate HVc populations, intermingled in the same area. One primarily responds to auditory input, and projects to the AFP; the other is associated with voluntary song production, and projects with strictly excitatory connections to RA.

The behavior of neurons in RA is particularly intriguing. When the bird is not singing, neurons fire tonically, with independent rates and spike times. Once song begins, RA neurons go almost completely silent, except for brief periods of intense bursting ($\sim$100 Hz). Each neuron spends about 10% of its time in these bursts, each of which may last from two or three to a dozen or more spikes [9]. Moreover, the onset or offset of one burst very often coincides with those of other bursts, to within an interval on the order of 1 ms. The appearance is that of a sequence of discrete states through which RA passes during song. Neither the mechanism behind nor the purpose of such a sequence of states is yet fully clear.

1.2 Computational learning

On the basis of a memorized tutor song and auditory feedback, the finch learns to produce its own song. At least three major open questions about this process may be asked: How are the neurons within each area connected to produce the observed activity, and just what computation does that pattern of activity accomplish? Under what sort of learning rule are those connections established? And how is each area connected to the others, so that the

high-level behavior and learning we observe may emerge?

Attempts have been made to construct models addressing these issues [5, 20, 21]; but so far these have suffered from problems—being sketchy, excessively high-level, and/or inconsistent with observation—and are ultimately unsatisfying. One serious obstacle is that important experimental information about the system is yet missing, preventing the above questions from being answered definitively at present. However, we can try to propose low-level models that fit with what is currently known and provide possible mechanisms by which the system might plausibly operate.

Standard frameworks for classification, regression, or trajectory learning, such as the Perceptron learning rule and backpropagation, suffer from problems of biological realism. It can be difficult to take a rule that may be simple to write down, and realize it in a network constructed from neurons as we understand them; or the standard prescription may require the network to run both forwards and backwards in time, an ability hard to imagine in living cells. One general framework whose difficulties are less objectionable than these is the class of REINFORCE algorithms described by Williams [24]. The advantages of these algorithms include the following: in adjusting its connections to other units, each unit needs only information that is purely local both spatially and temporally; a single global reward signal is maximized if each unit acts individually to maximize it; in the case of trajectory learning, a single accumulator for each unit, keeping a running total of a given time-varying quantity, suffices to specify the weight update when the reward signal is ultimately delivered.

Such algorithms are a promising model for biological learning in general, and in particular might be applied to account for the establishment of connections in RA. In general, these algorithms are not widely used. One reason for this may be that, when performance is the only consideration, backpropagation provides an efficient and well-characterized way to do gradient descent on an error function. REINFORCE algorithms, by contrast, do noisy gradient descent; and in the paper in which he introduced them, Williams proved only that they follow the gradient on average, without any statement about the extent of the noise in their estimates, or if and how the noise can be sufficiently minimized that the algorithm is useful in practice. The latter part of this document is concerned with investigation of these issues.

Chapter 2

Modeling RA

During singing, neurons in RA spend most of their time silent, interspersed with bursts of rapid firing. The timing of these bursts and the spikes in each burst, with respect to auditory landmarks in the song, is highly reproducible from one instance of singing to the next[1], typically to within about 1 millisecond. Bursts usually consist of three to six spikes, but may range from a single spike to more than a dozen. Figure 2-1 gives an illustration.

Another visually striking feature of the recordings of Figure 2-1 is the extent to which the onsets and offsets of bursts of firing align with one another, often within a few milliseconds. The appearance is reminiscent of that of a synfire chain [1, 13, 11], where pools of cells in turn fire in bursts and then become quiescent as the next pool of cells is activated.

One question that may then be asked is, what kind of neural mechanisms might be responsible for these sorts of observed and suggested firing patterns? Because of the high degree of coordination apparent between cells, we assume that the observed behavior arises from the network structure, and is not strictly a result of purely cellular mechanisms. The question we ask then becomes, what network architecture and synaptic mechanisms are necessary to produce bursts in successive cells or groups of cells, each of which then stops firing as the next cell or group begins?

For instance, assuming something like the synfire chain picture is correct, it is notable that each cell or pool of cells fires several times before the next begins to do so at all; and although the former stops firing at that point, the latter has received enough total excitation that it fires several times despite the lack of ongoing input. This feature suggests the presence of facilitation, a synaptic mechanism whereby each presynaptic spike has an increasingly greater postsynaptic effect. The suggestion that only one one cell or group is active at a time in a local section of the putative chain similarly suggests the operation of a kind of winner-take-all mechanism, which in turn implies global inhibition. Before concluding that such mechanisms are necessarily present, however, we should first consider networks which lack them, in order to see if simpler arrangements are sufficient to produce firing patterns like the ones observed.

[1] Because the tempo of singing can vary from one instance of the song to the next, as well as within a single instance, the multiple instances of the song (and the simultaneous neural recordings) must first be locally time-warped to align auditory landmarks. Once this procedure has been performed, the coincidences described above become apparent.

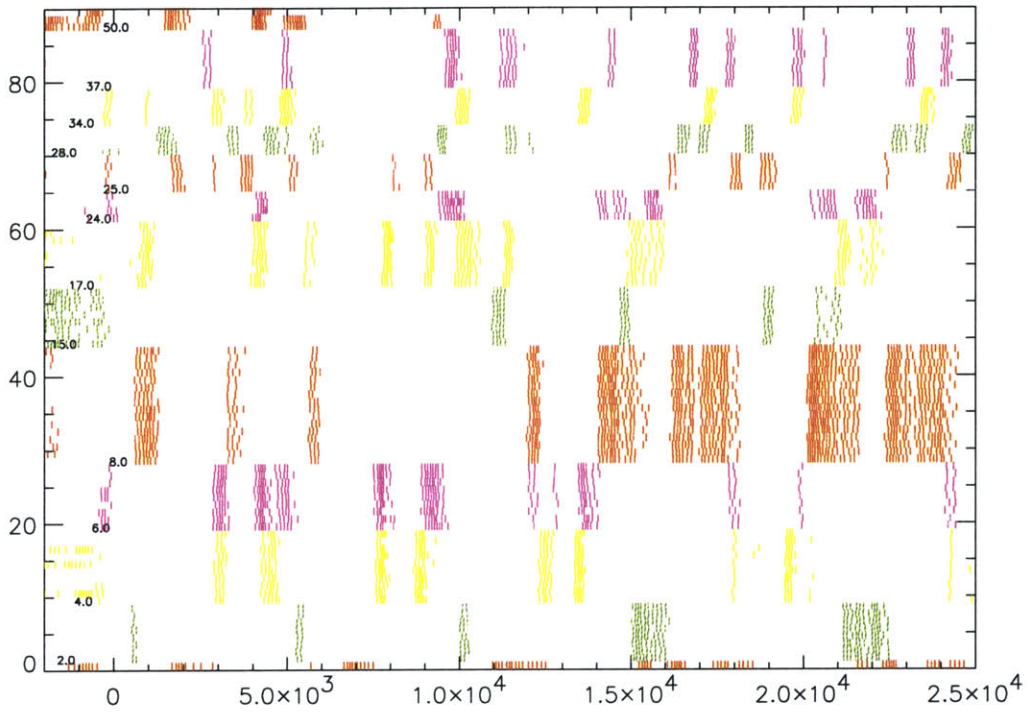

Figure 2-1: Recordings from nearby neurons in RA during singing [9]. Each horizontal line of tick marks represents spike times for a single cell. A block of consecutive lines of the same color represents recordings from the same cell over multiple instances of the song. The x-axis gives time in units of 25 μs. Since the rate of singing can vary from trial to trial as well as within a single bout of singing, recordings have been time-warped to align auditory landmarks in the song; the extent of the time-warping is typically within a few percent. Recordings from 13 separate cells are shown here, with from 1 to 16 instances of song for each.

2.1 The model

Our concern is with individual spike timing rather than simply firing rates, though the detailed dynamics at the level of ion channels are not important; thus we choose an integrate-and-fire model for cells in the network. Units in this model may correspond to individual neurons or pools of cells in RA. Take V to be the membrane potential; its update equation is

$$C\frac{dV}{dt} = g_L(V_L - V) + g_{exc}(V_{exc} - V) + g_{inh}(V_{inh} - V) \tag{2.1}$$

where C is the membrane capacitance, g_L the leak conductance, g_{exc} and g_{inh} the conductances associated with excitatory and inhibitory channels, respectively, and V_L, V_{exc}, and V_{inh} the potentials toward which V is driven by each of those competing conductances. When V reaches a value V_{thresh}, the unit instantaneously fires and V is set to V_{reset}.

When a cell fires, it opens excitatory or inhibitory channels in those cells connected to it. We associate with each unit a variable s, which reflects the degree to which channels in those connected units are open. s jumps discontinuously when a unit fires, and decays otherwise:

$$\tau_{syn}\frac{ds}{dt} = -s + \Delta_s\delta(t - t_{spike}) \tag{2.2}$$

If we consider V, g, and s as vectors, we can write

$$g_{exc} = W_{exc}s, \quad g_{inh} = W_{inh}s \tag{2.3}$$

where W_{exc} and W_{inh} are the weight matrices for both types of connections in the network.[2] With each W a matrix of dimensionless weights, s has units of conductance. For simplicity, we choose the Ws in what follows to be binary matrices, so that each pair of units is either connected with unit strength, or unconnected.

We take as biophysically realistic parameter values the following: $C = 1$ nF, $g_L = 0.025\mu$S, $V_L = -70$ mV, $V_{exc} = 0$ mV, $V_{inh} = -70$ mV, $V_{thresh} = -52$ mV, $V_{reset} = -59$ mV, $\tau_{syn} = 100$ ms, $\Delta_s = 9.5$ nS [25]. We further assume all units and connection strengths are identical, for the purposes of the model; variations in these could help to account for observed features like different numbers of spikes per burst among different cells.

2.2 Simulations

Before we consider any specialized synaptic mechanisms, let us first ask whether the network structure alone can account for these sequences of bursts. We use the simplest possible model synapses and look at different arrangements of excitatory and inhibitory connections in the network.

[2]This arrangement allows violations of the empirical Dale's Law, which states that the outgoing connections for a given neuron are either all excitatory or all inhibitory. For instance, we could choose the Ws such that unit A excites unit B but inhibits C. However, this difficulty can be overcome by inserting auxiliary units; in this example, we disconnect A from C, have A excite a new unit D, and have D in turn inhibit C. We assume that such auxiliary units are implicitly present wherever necessary.

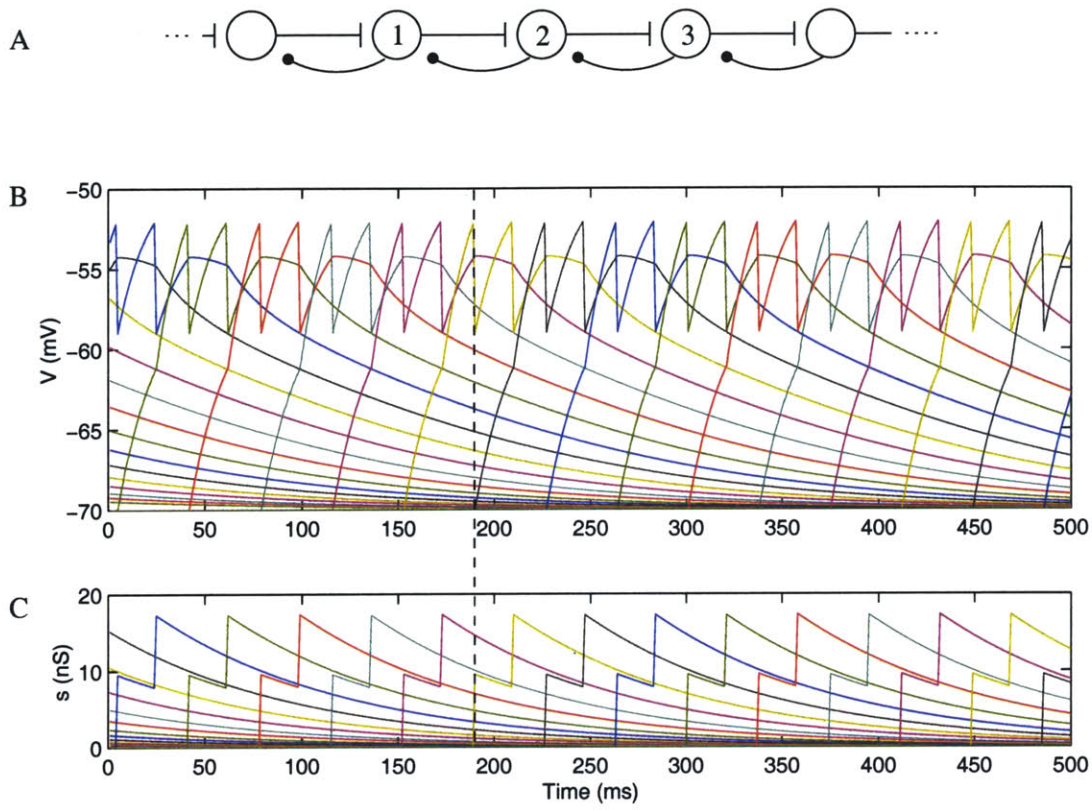

Figure 2-2: First possibility for an architecture wherein each unit bursts in turn. A: Diagram of network with forward excitation and reverse inhibition; a wave of activity travels from left to right. The numeric labels refer to the example given in the text. B, C: Excerpt from simulation of that network. Each successive color represents the successive unit in the chain, with the same color representing the same unit for the upper trace (B, membrane potential V) and the lower one (C, conductivity s). The vertical dashed line just before 200 ms highlights how, when a unit fires, s for that unit increases discontinuously, and the membrane potential of the succeeding unit begins to increase as a result.

2.2.1 Forward excitation, reverse inhibition

The most obvious structure likely to produce an activity pattern where each unit is active alone in turn is a chain wherein each unit excites the one following and inhibits the one previous (Figure 2-2). In such a case, when unit 1 in the figure begins firing, g_{exc} will start increasing for unit 2, driving its potential towards V_{thresh}. At some point, perhaps after several spikes from 1, 2 will reach threshold and begin firing, starting to increase g_{exc} for 3 and simultaneously suppressing 1's activity.

This structure leads to the desired activity pattern very easily. Using the parameter values given above and equal strength for reverse inhibition and forward excitation, a pattern immediately emerges where each unit fires twice and then gives way to the next one, as shown in the figure.

2.2.2 Forward excitation, global inhibition

One disadvantage of the network structure of Figure 2-2 is its complexity. It seems improbable that such a carefully arranged structure, with opposite types of connections running neatly in a linear chain in both directions, would actually be found in RA; nor is it clear how such a structure might be set up by the developing or learning brain in the first place; nor is there direct experimental evidence for such a meticulously arranged architecture.

What there is experimental evidence for is a network consisting mostly of short-range excitatory connections, with longer-range fast inhibition [19]. The recordings shown in Figure 2-1 are taken from the population of excitatory projection neurons, which are much more common than the inhibitory interneurons and easier to record from. Their processes have length on the order of 150 μm radius from the cell body, compared with the roughly 500 μm diameter of RA. There is also a smaller population of inhibitory interneurons, with faster response and with processes of length $\sim$400 μm. It is then plausible to treat this population as providing fast global inhibition.

We thus extend the model by adding a single inhibitory unit, for which τ_{syn} is one-third that for the excitatory units. As before, the latter are connected in a unidirectional excitatory chain; all also connect to the inhibitory unit, which in turn connects back to all the others. In this case, $g_{inh} = \beta s_{inh}$, with s_{inh} subject to the same update equation given for s above, and incremented each time any excitatory unit fires. Figure 2-3 gives a diagram.

Again, this architecture gives rise to a qualitatively correct bursting pattern, as shown in the figure, without need for careful tweaking of parameters. Here each unit's firing provides to the next unit two injections of increased conductivity, which persists long enough for that unit to fire twice itself. The inhibition prevents activity from continuing thereafter, and silences the unit first active, for which conductivity has declined with time, before the one more recently active, with correspondingly higher conductivity.

2.2.3 Excitation only

We can go on to ask, is even this global inhibition necessary? Can a simple feedforward excitatory chain maintain the same kind of firing pattern? Sure enough, using the same parameters named above, the chain produces a burst pattern qualitatively like those described before, here shown in Figure 2-4.

This case is considerably less robust than those previously discussed. If Δ_s is a few percent less, for instance, the wave of activity dies out; if it's a few percent more, activity crescendos, each cell firing more quickly and more often than the one before it. This sensitivity and need for careful parameter tuning suggests that, while a chain without inhibition has the capability of firing in a wave of activity as observed, the actual mechanism in RA likely includes some form of inhibition—particularly since evidence for such inhibition has been observed. However, additional synaptic mechanisms such as facilitation and depression appear to be unnecessary to explain the recordings made of RA; though, again, just because they aren't necessary doesn't mean they might not nevertheless be present. It is also notable that the biophysically motivated choices of parameter values we've been using are within the range that does give a stable traveling wave of activity.

A greater number of spikes per burst is also possible even without inhibition. Figure 2-5(A), for instance, shows a run with three spikes per burst. Note, though, that within each burst, the second interspike interval is 50% longer than the first. Such dramatic slowing of

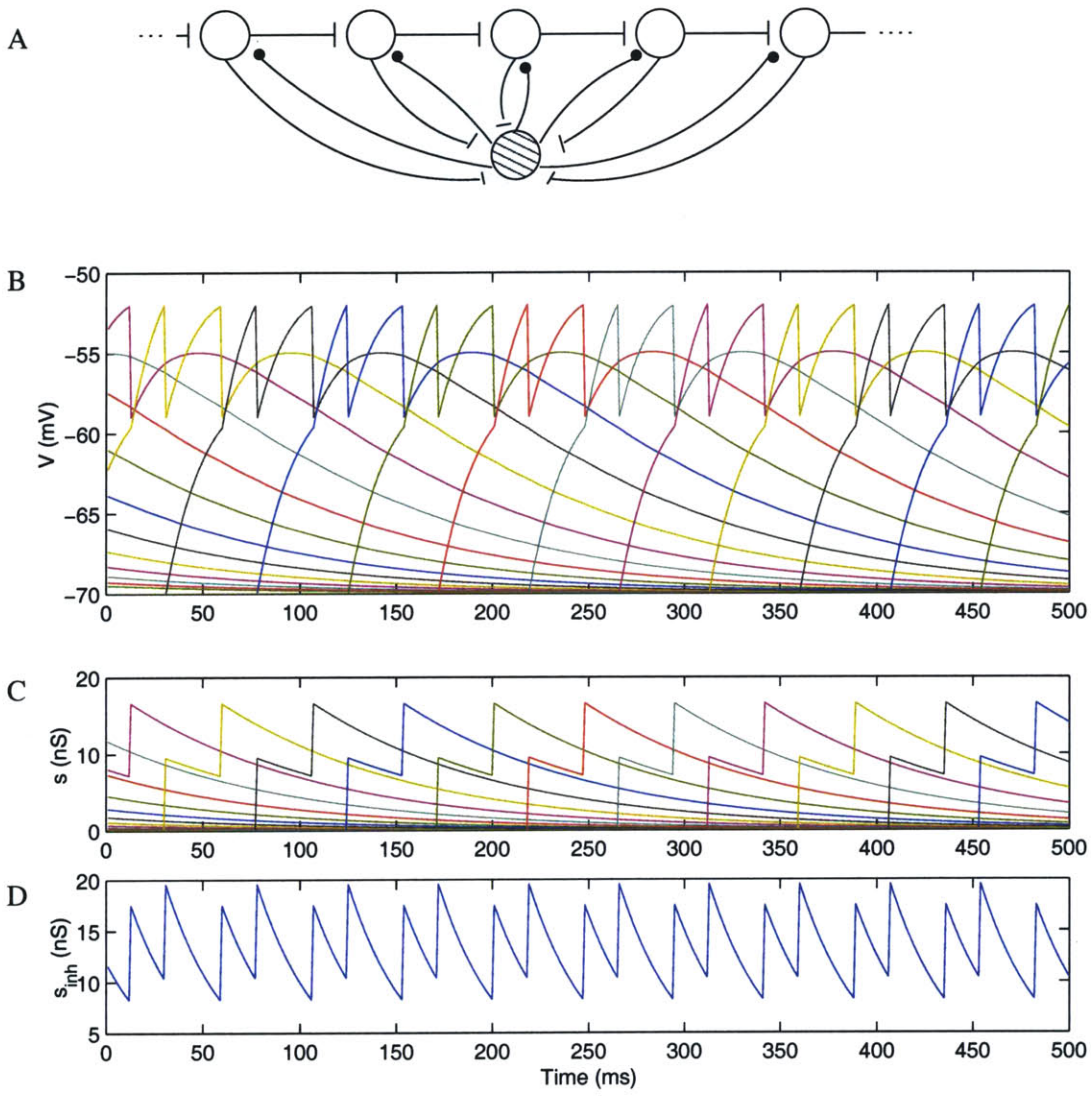

Figure 2-3: Another possibility for an architecture wherein each unit bursts in turn. A: Diagram of network with forward excitation and global inhibition; the empty circles represent excitatory units, the single shaded one inhibitory. B, C, D: Excerpt from simulation of that network. Each successive color represents the successive unit in the chain, with the same color representing the same unit for the trace in B (membrane potential V) and that in C (conductivity s). The trace in D is that of conductance for the global inhibitory unit. $\beta = 0.15$.

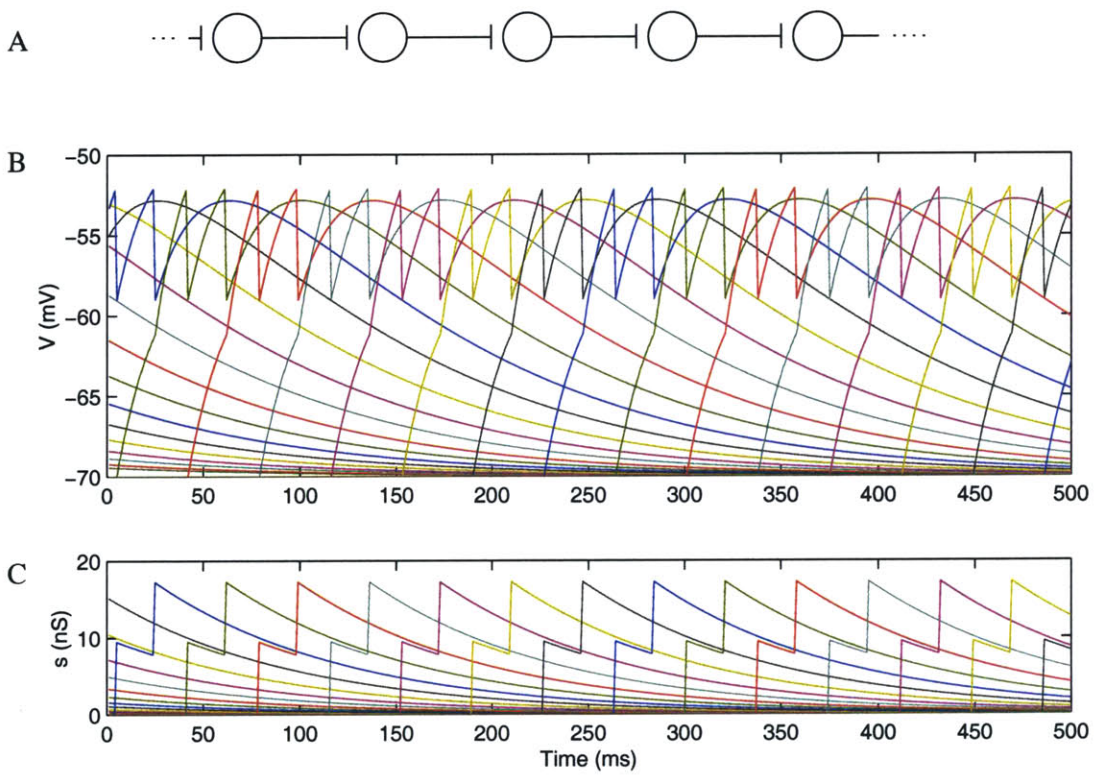

Figure 2-4: A third possibility for an architecture wherein each unit bursts in turn. A: Diagram of network with forward excitation only. B, C: Excerpt from simulation of that network. Each successive color represents the successive unit, with the same color representing the same unit for the trace in B (membrane potential V) and that in C (conductivity s).

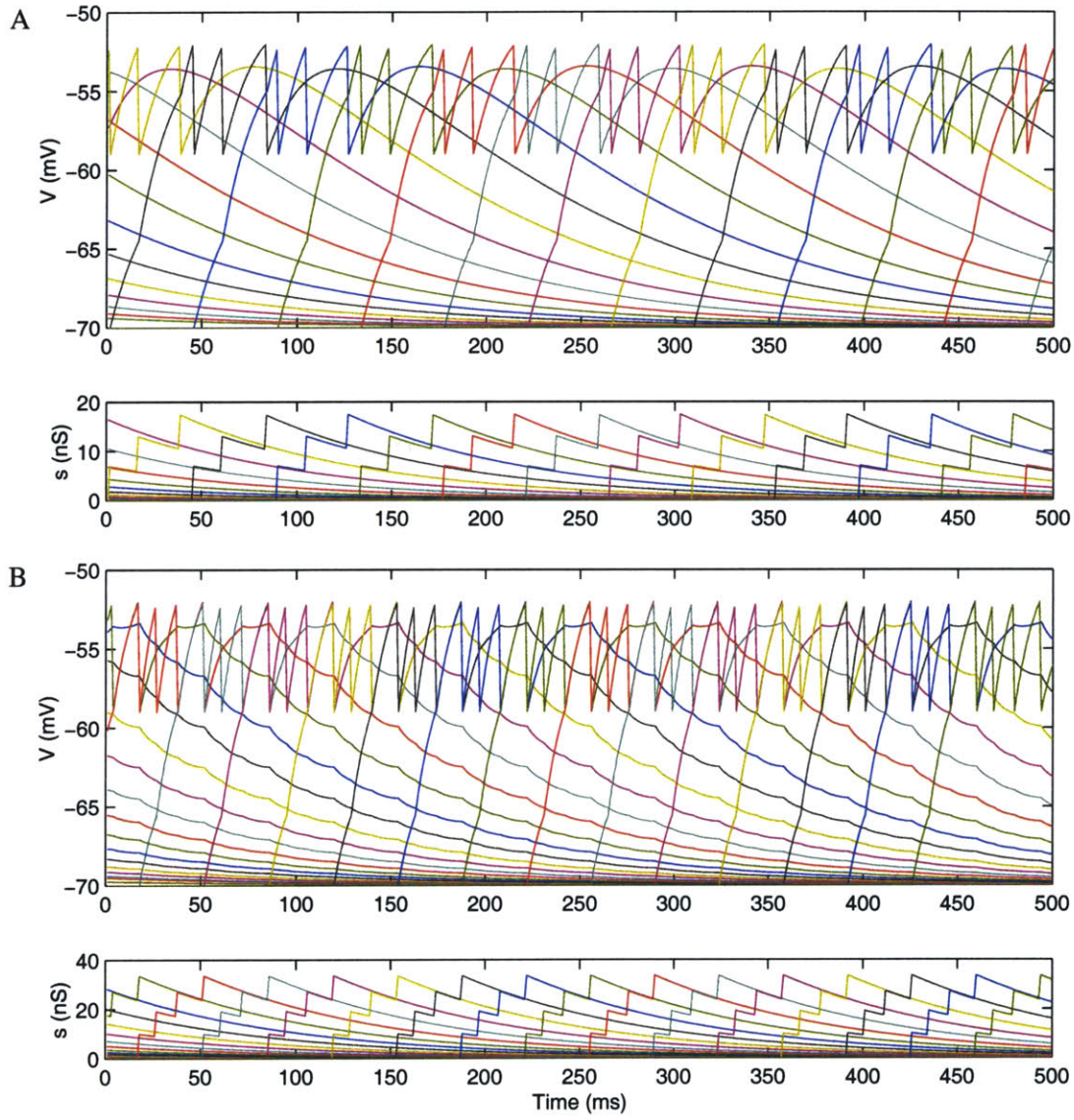

Figure 2-5: A: Same architecture as in Figure 2-4, with forward excitation only, exhibiting an activity pattern with three spikes per burst. $\Delta_s = 7$ nS. B: With global inhibition as in Figure 2-3, and four spikes per burst. $\Delta_s = 8$ nS, $\beta = 0.95$. The interspike interval slows more over the course of a burst in A ($15 \rightarrow 21$ ms) than in B ($8.9 \rightarrow 10.7 \rightarrow 13.2$ ms), despite the greater number of spikes per burst in the latter case.

firing is not observed in the recordings of Figure 2-1, suggesting, again, that the biological system involves inhibition to actively shut off the previously active cell when the next starts firing, rather than each cell's activity just wearing off with time. Figure 2-5(B) shows a run with global inhibition and four spikes per burst, where the interspike interval varies less than 25% from one to the next.

In the simulations above, the interspike intervals are on the order of 20 ms, much longer than the few milliseconds observed in the recordings of Figure 2-1. However, by reducing the synaptic time constant from 100 ms to 5 ms [25] (an appropriate choice for burst neurons), and increasing Δ_s eightfold (which could be achieved, for instance, by linking neurons together into larger coactive pools), the same qualitative firing patterns remain but with interspike intervals of a few milliseconds. This constitutes a prediction of the model for synaptic time constant for cells in RA.

2.3 Analysis

We turn now to a mathematical consideration of the system. For tractability, we simplify the model further: rather than having a spike affect ion channel conductance, we take it to provide a stereotyped current pulse in downstream cells, independent of their membrane potential at the time of firing:

$$\tau_I \frac{dI}{dt} = -I + \Delta_I \delta(t - t_{spike}) \tag{2.4}$$

and, much as before,

$$C\frac{dV}{dt} = g_L(V_L - V) + I \tag{2.5}$$

Between spikes, $I \propto \exp(-t/\tau_I)$, and

$$V = -\frac{1}{g_L}\left[k\mathrm{e}^{-\frac{g_L}{C}t} - \frac{\Delta_I g_L \tau_I}{g_L\tau_I - C}\mathrm{e}^{-t/\tau_I} - g_L V_L\right] \tag{2.6}$$

where k comes from initial conditions on the membrane potential,

$$k = -g_L V(t=0) + \frac{\Delta_I g_L \tau_I}{g_L\tau_I - C} + g_L V_L. \tag{2.7}$$

Suppose we want each unit in turn to fire twice and then stop, in keeping with the simulations above. Let the presynaptic neuron fire at t_0 and t_1, leading the postsynaptic neuron to fire at t_2 ($t_0 < t_1 < t_2$). Then we have $I(t_1) = \Delta_I(1 + \exp(-(t_1 - t_0)/\tau_I))$ and

$$V(t_1) = -\frac{1}{g_L}\left[\frac{\Delta_I g_L \tau_I}{g_L\tau_I - C}\mathrm{e}^{-\frac{g_L}{C}(t_1 - t_0)} - \frac{\Delta_I g_L \tau_I}{g_L\tau_I - C}\mathrm{e}^{-(t_1-t_0)/\tau_I} - g_L V_L\right], \tag{2.8}$$

leading in turn to

$$V(t_2) = V_{thresh} = \\ -\frac{1}{g_L}\left[\left(-g_L V(t_1) + \frac{I(t_1) g_L \tau_I}{g_L\tau_I - C} + g_L V_L\right)\mathrm{e}^{-\frac{g_L}{C}(t_2 - t_1)} - \frac{I(t_1) g_L \tau_I}{g_L\tau_I - C}\mathrm{e}^{-(t_2-t_1)/\tau_I} - g_L V_L\right]. \tag{2.9}$$

If the wave of activity is to propagate unaltered, the postsynaptic neuron must fire for the second time at $t_2 + (t_1 - t_0)$. This gives a second condition that must be satisfied:

$$V(t_2 + t_1 - t_0) = V_{thresh} =$$
$$(V_{reset} - \frac{\Delta_I \tau_I (1 + \exp(-(t_1 - t_0)/\tau_I))}{g_L \tau_I - C} e^{-(t_2 - t_1)/\tau_I} - V_L) e^{-\frac{g_L}{C}(t_1 - t_0)}$$
$$+ \frac{\Delta_I \tau_I (1 + \exp(-(t_1 - t_0)/\tau_I)) \exp(-(t_1 - t_0))/\tau_I}{g_I \tau_I - C} e^{-(t_2 - t_1)/\tau_I} + V_L. \quad (2.10)$$

When parameter values have been chosen, equations (2.9) and (2.10) together specify the intervals $t_1 - t_0$ (interspike interval within a burst) and $t_2 - t_1$ (interval between end of one burst and start of the next). Using the values given above (with either $\tau_I = 100$ ms or $\tau_I = 5$ ms), however, there are no values for those intervals below 250 ms that satisfy both equations ((2.9) is satisfied for intervals on the order of 1 ms, and again for some intervals over 250 ms, but (2.10) is not satisfied for intervals below 250 ms at all). Choices for parameter values that do give valid intervals do exist; but for these values motivated by biological realism, it's not possible in this simplified model to maintain a consistent traveling wave of activity, where each cell fires two spikes separated by the same length of time. The failure of this simplest burst pattern suggests that bursts consisting of more spikes would be unlikely to make for a viable consistent traveling wave either.

Another possibility for such a traveling activity pattern is that each presynaptic spike directly gives rise to a postsynaptic spike, but with a synaptic delay, so that the first postsynaptic spike doesn't occur until after the last presynaptic spike. Such a case would give the appearance of a traveling wave of activity like we've been considering, but instead of N spikes from one unit causing N spikes in the next, the mechanism would be one spike from one unit causing one spike in the next, N times. However, this possibility can be ruled out with the following argument: Suppose the interval between spikes in the presynaptic cell is Δt. When the effect of the first spike reaches the postsynaptic cell, the excitatory current into that cell just beforehand is 0. When the effect of the second spike arrives, some residual current from the first spike will still be present. As a result, the membrane potential will reach V_{thresh} more rapidly than it did the first time, so that the postsynaptic cell's second spike will occur less than Δt after the first. In this way, the interspike interval will be increasingly shorter in each successive cell of the chain, and so a consistent traveling activity pattern is not possible by this mechanism.

Other activity patterns which involve multiple-state spike trains propagated by identical cells can similarly be ruled out. Figure 2-6 shows two examples. In the first case, different states in the pattern have different numbers of spikes. However, if three spikes from one unit give rise to two spikes in the following, identical unit, then those two spikes should hardly be able to give rise to as many as three again in the next unit thereafter; the activity will die out. Conversely, if two spikes from one unit result in three spikes in the next, then those three spikes should be responsible for at least three spikes in the following unit, if not more, and the activity will likely blow up. In the second example, the timing of spikes varies from one cell to the next. But again, if the one spike at the start of the presynaptic train gave rise to the two in quick succession at the start of the postsynaptic train, then the later two together in the presynaptic train should only result in still greater postsynaptic activity, not in only a single spike. Such patterns could be achieved if the model were extended such that not all units in the chain were identical; but that would be an unnecessary complication to this model, under which none of these consistent traveling activity patterns discussed above

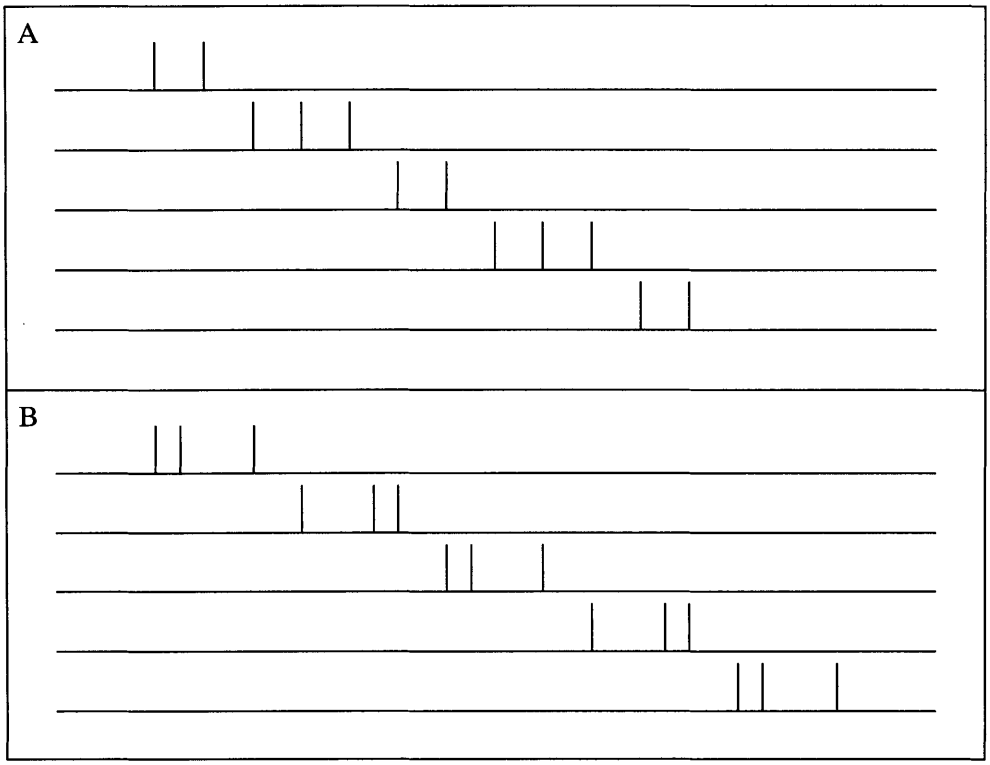

Figure 2-6: Examples of multiple-state activity patterns impossible for identical units to generate stably. A: Different numbers of spikes per burst. B: Different interspike timing.

can be maintained without inhibition.

2.4 Training the network

We now step back from the specific model presented above and turn to the question of how the connections in a network like RA might be learned. At the same time, we now move from spiking to rate-based models, for greater analytical tractability.

One early classic model for learning is the Perceptron [12]. Consider only a single synapse. Let x be the activity of the presynaptic unit, y the activity of the postsynaptic unit, W the strength of the connection between them, and t the desired postsynaptic activity for that input pattern. When the learning rule is formulated as a gradient descent problem, we have

$$\Delta W \propto (t - y)x$$

This learning rule can be viewed as having a mechanism similar to that of contrastive Hebbian learning [16]: reinforce the behavior you want the system to exhibit (tx) and weaken the behavior it's actually exhibiting ($-yx$), with no net effect if the system is behaving as desired. Note also that the weight is modified only if the presynaptic unit is active, which is in keeping with observations of long-term potentiation and long-term depression (the former of which occurs if the postsynaptic unit is also active, the latter if it remains silent).

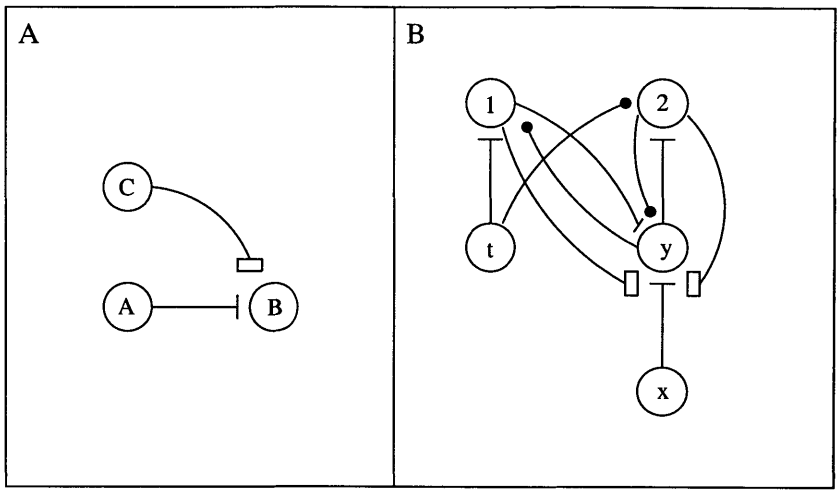

Figure 2-7: Left: The synaptic triad of [6]. The plastic synapse is that from A to B; C is the modulator. Right: An architecture which implements the Perceptron learning rule using synaptic triads.

How such a learning rule could be implemented biologically is less clear. One possible mechanism is via the synaptic triads of Dehaene et al. [6], which are also based on observations of song learning in birds, and can be used to achieve sequence learning. In these structures, one cell acts as a modulator for the plastic synapse between the other two. Figure 2-7 illustrates the operation. Neuron A of the triad excites neuron B, but only if the modulator C has recently been active, so that B acts as a coincidence detector for the other two. The learning rule is such that the strength of the connection from A to B is modified only if A and C are both active: strengthened if B is also active, weakened if B is silent.

The presence of three units in each of these structures suggests that the synaptic triad might serve as a biologically reasonable implementation of the Perceptron learning rule. However, the functional roles of the units are significantly different in the two formulations. In the Perceptron, one unit (x) controls whether the synaptic strength is modified, while the other two, when different, indicate that the synapse should be strengthened or weakened according to which of the two is more active. By contrast, in the synaptic triad, two units (A and C) specify that the weight should be changed if both are active, while the third (B) indicates whether the synapse should be strengthened or weakened. Thus one formulation cannot be mapped directly onto the other; the architecture must be modified. The Perceptron learning rule can be implemented with synaptic triads, but only by complicating the picture with additional units and connections.

If we add two more units (call them 1 and 2) to provide auxiliary logical values, we can assemble a structure out of synaptic triads that implements the Perceptron learning rule, as illustrated in Figure 2-7. Unit 1 effectively gives the value $t\bar{y}$ and unit 2 the value $\bar{t}y$; then the network operates as follows. x drives or fails to drive y, depending on the strength of that connection, which is the one subject to modification. If y and t (which gives the desired state of y) differ, then either 1 or 2 turns on, according to whether y is incorrectly quiescent or active; that auxiliary unit forces y into the correct state, and makes the $x \rightarrow y$ synapse plastic.

Such an approach has numerous problems. First of all, we've dramatically complicated the picture, nearly doubling the number of units, and increasing the number of connections from two to nine. That architecture itself needs to be set up somehow, with all the other synaptic connections having the appropriate locations and strengths; to some extent, we've solved one problem by replacing it with eight others. Dale's Law is again violated, with t and y each having both excitatory and inhibitory projections; fixing that problem as above requires the introduction of still more units. Timing issues have been complicated as well, with x needing to hold its state long enough for a signal to propagate through an extra set of connections, and all units' activity needing to be carefully timed if all are to be in the right states at the appropriate times. This structure can be seen as an improvement in some implementation sense over the original Perceptron learning rule, where the question was how it could be implemented in a biological setting at all; but any improvement is marginal at best.

Such considerations helped motivate the next section of this report, which is concerned with a different approach to training neural networks, based not on supervised learning with labeled input-output pairs but on reinforcement learning. That approach leads networks to produce desired outputs in response to input patterns in an in-place manner, not requiring auxiliary units to mediate the plasticity. Its capacity to handle trajectory learning makes it a potential candidate for a mechanism by which RA might learn to achieve its characteristic firing patterns.

Chapter 3

Network reinforcement learning

With the difficulties associated with the approaches to network training discussed in the previous chapter, we turn elsewhere for models that might reasonably account for learning in RA, and more generally, in recurrent neural networks.

Backpropagation is the well-known standard approach for training recurrent networks, due to its ease of use and well-characterized algorithm, performing explicit gradient descent on an error function [12]. However, it suffers from problems of biological realism and other difficulties. An error signal needs to be sent backwards through the same network that propagates a signal forwards, requiring each synapse to be two-way and multipurpose, and each unit to be able to recognize and transmit two kinds of signals. While the rule is local in space, the adjustment to a weight connecting two units depends on the activity of one unit at one time and the error associated with the other unit at a later time; this temporal nonlocality requires each unit to have information about the structure of the entire network, so that it can associate the earlier activity with the error signal after the appropriate propagation delay. This need for global knowledge is problematic for independent computational units with only local information. Moreover, in the case of trajectory learning, backpropagation requires that the network actually be run backwards in time. Issues such as these motivate investigations into alternative algorithms.

One general framework which suffers less from such difficulties is the class of REINFORCE algorithms described by Williams [24]. The advantages of these algorithms include the following: in adjusting its connections to other units, each unit needs only information that is purely local, both spatially and temporally; a single global reward signal is maximized if each unit acts individually to maximize it; and in the case of trajectory learning, a single accumulator for each unit, keeping a running total of a given quantity, suffices to specify the update when the reward signal is ultimately delivered.

However, one disadvantage is that REINFORCE performs only noisy gradient descent. In [24], Williams proved that these algorithms follow the gradient on average, but made no statement about how great the noise in their estimates is, or if and how the noise can be sufficiently minimized that the algorithm is useful in practice. We investigate these issues in this chapter.

3.1 The model

For simplicity, we consider here only networks of one particular type of unit, and the corresponding REINFORCE algorithm. Once again we consider a rate-based rather than

spiking model. Let the units in question have continuous output $x(t)$, with update equation

$$x(t+1) = f(Wx(t) + \xi) \tag{3.1}$$

where W is the weight matrix; $f(u)$ is the logistic function $f(u) = 1/(1+\exp(-u))$; and ξ is a vector of noise on the inputs to the units, chosen once at the time the input is presented, from a Gaussian distribution with mean 0 and variance σ^2.

The results which follow hold for the class of networks which are given an input by being initialized in some starting state, and thereafter are allowed to evolve until they reach a fixed point, which represents their output. The set of units considered to be part of the input may comprise only a subset of all units in the network, and likewise for the output; the units not part of the input may be arbitrarily initialized, and the final states of those not part of the output ignored. Networks which are strictly feedforward and those with symmetric weight matrices [14] are two subsets of this general class which are guaranteed always to come to a fixed point. Detailed derivations of the REINFORCE result and the first two results below are given in the Appendix.

If d is the desired output for the given input vector, then for stochastic gradient descent on the squared error function $E(x) = |d - x|^2$, the REINFORCE prescription for weight updates is

$$\Delta W = \eta(b - E(x))\xi x^T \tag{3.2}$$

with η a small positive constant, and b some baseline scalar or vector independent of ξ. If b is taken to be the error for the network output in the absence of noise, then it can be shown that in the low noise limit,

$$\Delta W = -\eta\xi\xi^T \frac{\partial\langle E\rangle}{\partial W} \tag{3.3}$$

where the expectation value is over all possible values for the noise ξ. It then follows that

$$\begin{aligned} \Delta W \cdot \frac{\partial\langle E\rangle}{\partial W} &= -\eta \left(\frac{\partial\langle E\rangle}{\partial W}\right)^T \xi\xi^T \frac{\partial\langle E\rangle}{\partial W} && (3.4)\\ &= -\eta \left|\xi^T \frac{\partial\langle E\rangle}{\partial W}\right|^2 && (3.5)\\ &\leq 0 && (3.6) \end{aligned}$$

Thus *every* weight update according to this prescription is within 90 degrees of the negative gradient, and results in a decreased error (or, more strictly, never an increased one) — a much stronger statement than simply that the average weight update follows the gradient downhill.

Notice also that we obtain a similar result if the update is performed only for the weights of those connections leading out of a single unit:

$$\begin{aligned} \sum_i \Delta W_{ij} \frac{\partial\langle E\rangle}{\partial W_{ij}} &= -\eta \sum_{i,k} \xi_i \frac{\partial\langle E\rangle}{\partial W_{ij}} \xi_k \frac{\partial\langle E\rangle}{\partial W_{kj}} && (3.7)\\ &= -\eta \left(\sum_i \xi_i \frac{\partial\langle E\rangle}{\partial W_{ij}}\right)^2 && (3.8)\\ &\leq 0 && (3.9) \end{aligned}$$

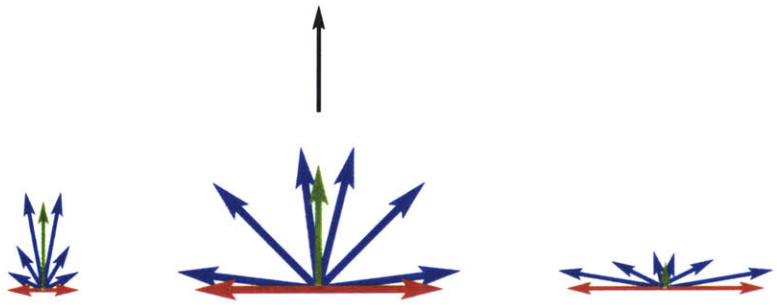

Figure 3-1: Sketch of the potential noise concern in stochastic gradient descent. In this picture, the true gradient (black) points upwards; individual weight adjustments (blue) given by Eq. (3.2) for different values of ξ are always within 90 degrees of the gradient, and their average over all values of noise is equal to the gradient. Depending on whether they are typically close to aligned with the gradient as at left, or close to orthogonal as at right, the noise represented by the orthogonal component (red) can mask the signal represented by the parallel component (green).

Thus the error also always decreases when this weight update rule is applied only to the weights fanning out of any subset of units, from a single unit to all N units in the network.

3.2 Noise analysis

A remaining concern is that the weight update might typically be closer to 90 degrees from the true gradient than 0 degrees; that is, that the noise represented by adjustments in directions orthogonal to the gradient will swamp the signal represented by the component of the update in the direction of the gradient. Figure 3-1 gives a sketch of the potential problem. Indeed, the following straightforward analysis suggests that the update will in fact be dominated by this noise.[1]

Consider only outputs from a single unit, and write $y \equiv -\eta \frac{\partial \langle E \rangle}{\partial W}$ for brevity. We can apply an arbitrary rotation to Eq. (3.3), multiplying on the left by a rotation matrix R:

$$\Delta W = \xi \xi^T y \tag{3.10}$$

$$R\Delta W = R\xi\xi^T R^{-1} R y \tag{3.11}$$

$$= (R\xi)(R\xi)^T R y \qquad (\text{since } R^T = R^{-1}) \tag{3.12}$$

This is the same as the original equation in the rotated coordinate system. ξ is drawn from an isotropic distribution, with any direction equally likely, and its magnitude is effectively scaled by the arbitrary learning rate η. Thus we can choose $y = [1, 0, 0, 0, \cdots]$ without loss of generality, giving $\Delta W = \xi_1[\xi_1, \xi_2, \cdots]$. We can define

$$u \equiv \xi_1^2, v_1 \equiv \xi_1 \xi_2, \cdots, v_{N-1} \equiv \xi_1 \xi_N \tag{3.13}$$

[1]Note that "noise" in this context represents an entirely different concept than the Gaussian variable ξ we add to the inputs of the units in the network.

so that u is the magnitude of the component of the weight update in the direction of the gradient, and v_i are the orthogonal components. We can interpret u^2, the squared magnitude of the weight update in the desired direction, as a signal, and $\sum_i v_i^2$, the squared magnitude of the rest of the weight update, as a noise term masking that signal. Because ξ_i are N independent Gaussian variables, we have $\langle u^2 \rangle = \sigma^4, \langle \sum_i v_i^2 \rangle = (N-1)\sigma^4$; this gives, as a sort of average signal-to-noise ratio, $\langle u^2 \rangle / \langle \sum_i v_i^2 \rangle = 1/N$. A more rigorous analysis, omitted here for space considerations, can be found in section A.5 of the Appendix.

On this basis alone, we would expect the algorithm to perform badly at training networks of any moderate size, and to scale up poorly. However, because of the correlations in the noise, with the same factor ξ_1 appearing in u and every v_i, we might hope that another quantity other than $\langle u^2 \rangle / \langle \sum_i v_i^2 \rangle$ is a more relevant measure of effective signal-to-noise ratio, and hence that the algorithm's scaling-up behavior is better than the above analysis suggests. We investigate this issue empirically in section 3.4 below.

3.3 Weight perturbation

The operation of the learning rule described above can be thought of as the following. Given an input, the network makes a small change to the fixed point which represents its output, and compares the resulting error E to the error E_0 it would have incurred in the absence of that change. If the error is decreased, it adjusts its weights in such a way that its output is shifted toward that new value; if the error would be increased, then the network adjusts itself in the opposite direction in weight space, which (because the change in behavior is linear in the limit of small adjustments) again leads to a decrease in the error.

Such a procedure immediately suggests the following, simpler algorithm. Make a small random change to all the weights. If the error decreases as a result, keep the change; if the error would increase, make the opposite weight change, which in the limit of small adjustments should again lead to a decrease in the error. This approach is in fact the basis of a learning rule already used in some applications, termed "weight perturbation"; it is used especially in training networks implemented in analog VLSI, where the resources required by backpropagation in particular are excessive, and the simplicity of this approach (computationally and in terms of space and other circuit resources) makes it advantageous [4, 10].

We formalize the algorithm by giving the units the update equation

$$x(t+1) = f((W + \xi)x(t)) \tag{3.14}$$

where again the noise ξ is chosen once at the time the input is presented, and held fixed until the output is produced; the weight update is

$$\Delta W = \eta(E_0 - E(x))\xi. \tag{3.15}$$

This algorithm too, with the noise entering as a perturbation to the weights, gives an update on average in the direction of the gradient:

$$E(x) - E_0 \quad = \sum_{i,j} \frac{\partial \langle E \rangle}{\partial W_{ij}} dW_{ij} \quad = \sum_{i,j} \frac{\partial \langle E \rangle}{\partial W_{ij}} \xi_{ij} \tag{3.16}$$

$$\langle \Delta W_{hk} \rangle \quad = -\eta \sum_{i,j} \frac{\partial \langle E \rangle}{\partial W_{ij}} \langle \xi_{ij} \xi_{hk} \rangle \quad = -\eta \frac{\partial \langle E \rangle}{\partial W_{hk}} \sigma^2 \tag{3.17}$$

Moreover, as expected, this update is always within 90 degrees of the true gradient:

$$\Delta W \cdot \frac{\partial \langle E \rangle}{\partial W} = -\eta \sum_{i,j} \left(\sum_{h,k} \frac{\partial \langle E \rangle}{\partial W_{hk}} \xi_{hk} \xi_{ij} \right) \frac{\partial \langle E \rangle}{\partial W_{ij}} \tag{3.18}$$

$$= -\eta \left(\sum_{i,j} \frac{\partial \langle E \rangle}{\partial W_{ij}} \xi_{ij} \right)^2 \tag{3.19}$$

$$\leq 0 \tag{3.20}$$

though the stronger statement we have for REINFORCE, that the error also strictly decreases when the update is performed only for weights leading out of a subset of units, is not necessarily true in this case.

As one might expect, this simple algorithm, so similar in concept to the REINFORCE class, can in fact be shown to be a member of that class; this result is shown in section A.6 of the Appendix. The advantages of weight perturbation over the standard REINFORCE algorithm include the fact that it is marginally simpler to implement and its operation is more intuitive. Its disadvantages arise from the fact that it requires the generation of more noise terms; in particular, since the signal-to-noise ratio scales as $1/n$ with the number of noise terms n, as discussed in section 3.2, the performance of weight perturbation can suffer as a result as network size increases.

3.4 Simulations

We performed benchmark tests to compare the performance of backpropagation with that of the two variants of REINFORCE. Our first test case was handwritten digit classification, using an abridged version of the modified NIST database of labeled handwritten digit images, with 5000 training examples and 1000 test examples. We used a three-layer feedforward network, with 784 input units (representing the 28×28 image), 49 hidden units, and 10 output units. The desired output d was a 10-element binary vector of which exactly one component was 1, corresponding to the digit represented by that input. We took $\sigma = 10^{-2}$ and $\eta = 2 \cdot 10^{-3}$. Weights were initialized to random values drawn from a Gaussian distribution centered at 0 with standard deviation 0.05. Weight updates were scaled in the REINFORCE runs to be the same magnitude on average as for backpropagation, using the fact that for standard REINFORCE

$$\langle \Delta W_{ij} \rangle = -\eta \sum_k \langle \xi_i \xi_k \rangle \frac{\partial E}{\partial W_{kj}} = -\eta \sigma^2 \frac{\partial E}{\partial W_{ij}} \tag{3.21}$$

and likewise for weight perturbation

$$\langle \Delta W_{ij} \rangle = -\eta \sigma^2 \frac{\partial \langle E \rangle}{\partial W_{ij}} \qquad \text{(from Eq. (3.17))}. \tag{3.22}$$

3.4.1 Batch vs. online update

Two schemes for performing weight changes are online update, in which a weight adjustment is made after consideration of each individual example in the data set; and batch update, in which the adjustments based on every example in the data set are calculated and added

together before any update is actually made. At a given point in weight space, because the quickest way to reduce error on a single example is rarely the quickest way to reduce error on the data set as a whole, the individual adjustments of online update are scattered about the single adjustment of batch update. Thus the use of online update introduces an additional source of noise into the experiments, one which furthermore can vary according to the order in which training examples are chosen; and so batch update is more appropriate for noise comparisons between the different approaches.

However, this sort of network can have a tendency to get stuck in a plateau during training with backpropagation, with very little improvement for many training epochs. In a statistical mechanics analysis of such nonlinear two-layer networks, the problem can be shown to arise from a symmetry in the functional roles of the hidden units, when all are doing effectively the same thing; in this regime the gradient can be nearly flat. When the network has been provided with enough training examples, there is ultimately a phase transition; the permutation symmetry is broken and the hidden units take on different roles, and error starts to decrease quickly again [17]. We speculate that such a process is responsible for the observations associated with the constant-error plateaus here.

The length of time it takes a network trained with batch update to break out of that plateau can be both long and variable. Online update can advance beyond that plateau more quickly than batch update, presumably because the spread of its adjustments around the single value available to batch update allows the network to reach areas of weight space where the gradient is steeper; and it does so after a more consistent and reproducible interval. Thus online update was used for the experiments first described in this section, comparing the learning curves of the different approaches.

It is interesting to consider a network trained with batch update, with noise added to the weight adjustments; if the intrinsic variation of online update speeds the network's learning time, then simply adding Gaussian noise to batch update could conceivably improve its performance. Figure 3-2 shows several backpropagation runs with multiplicative noise of various magnitudes added to the weight updates; the results with additive noise are qualitatively the same (data not shown). The three error measures are squared error, defined as $\sum_i (d_i - x_i)^2$, where the sum is over all output units; and training and test error, defined as the percentage of examples in the respective data sets which were misclassified by the network, where the classification was determined according to which of the output units had the largest response.

With batch update, adding multiplicative Gaussian noise with standard deviation up to 50% of the magnitude of the noiseless weight update does give a marginal improvement in how quickly the network advances beyond the constant-error plateau to the next stage of learning. These runs with low or moderate noise closely resemble the runs with online update, supporting the idea that that the variations in gradient encountered in online update are what allow it to progress beyond that plateau more rapidly than batch update. When the noise is made very large (here, 200% of the noiseless update), however, there is an effect; the network fails to get caught on the constant-error plateau, and improvements in all three error measures are faster in the early stages of learning. However, at the point where the presumed symmetry is broken in noiseless or low-noise training and error begins to decrease rapidly, the high-noise training fails to benefit from the same rapid improvement; instead its error continues to decrease at about the same low rate as before, and its performance falls behind that of the lower-noise versions. It is not clear if it would eventually attain the same low error levels at all, but it fails to do so in a comparable time. With online update, even noise at the 200% level has no qualitative effect on the learning curves.

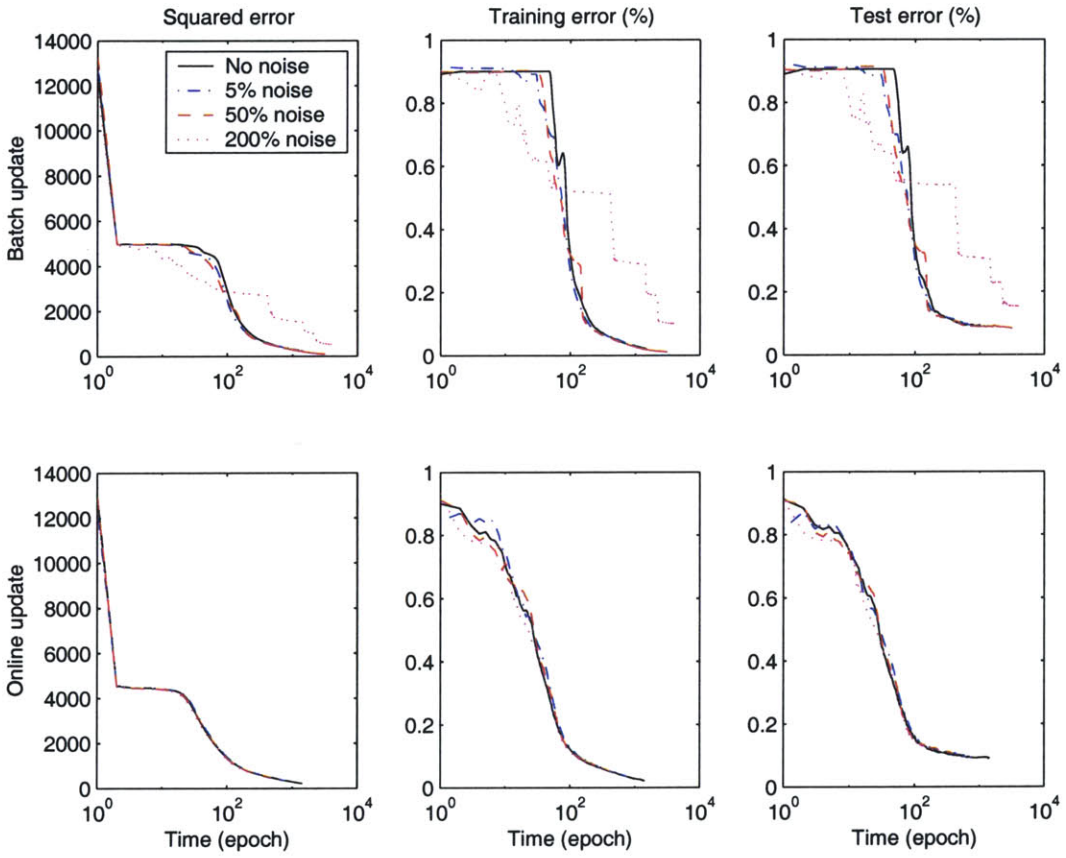

Figure 3-2: Learning curves for networks trained with backpropagation, with multiplicative noise on the weight updates. The matrix of noise terms was of the same dimensions as the weight update, with entries chosen from a Gaussian distribution with standard deviation equal to the percentage shown of the corresponding entry of the weight update. Top row, batch update; bottom row, online update; left column, squared error; center column, training error; right column, test error. Note that the network stays in the constant-error plateau longer for batch than for online update; and with the latter, the training and test errors hardly show a plateau at all. These effects are more pronounced in networks with more hidden units (not shown).

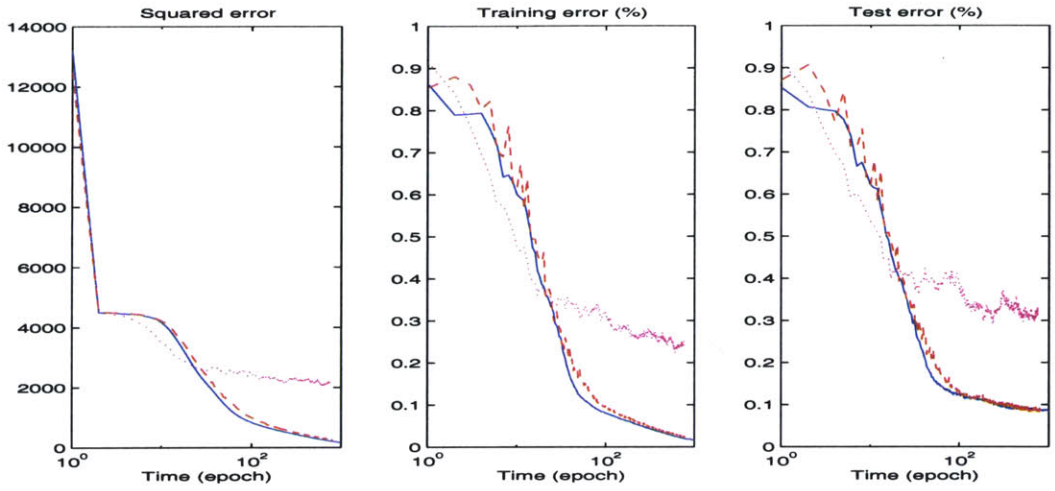

Figure 3-3: Comparison of learning curves for backpropagation (blue, solid), standard REINFORCE (red, dashed), and weight perturbation (pink, dotted) for learning digit classification on the abridged MNIST data set, for three error measures: squared (left), training (center), test (right).

3.4.2 Comparison of learning curves

Figure 3-3 shows the squared error, training error, and test error as a function of time for all three training algorithms on this data set, using online update.

The REINFORCE algorithm performs nearly as well as does backpropagation, attaining the same levels on all three error measures, at most a few epochs behind. Weight perturbation appears to learn faster at first, all three error measures decreasing more quickly than for either of the other two training methods, and it avoids the constant-error plateau. However, just as with the highest-noise runs with backpropagation and batch update above, weight propagation fails to show the rapid error decrease of the other methods a few epochs later when, for the others, the symmetry is broken and the gradient becomes steeper; and none of its error measures fall to levels comparable to those of the other methods, even when run for thousands of epochs (not shown).

If we leave out an explicit consideration of time, and examine the course of learning in terms of how the error measures change in relation to each other, we see in Figure 3-4 that all three algorithms follow essentially the same course through error space. That is, for a given value of, e.g., training error, all three training algorithms will have nearly the same values as one another for squared and test error. The most obvious exception is in the first few data points, where the networks have been randomly initialized to different locations in error space but quickly converge to the same path. While the path is the same, however, not all three methods progress along it to the same extent; in particular, weight perturbation slows or halts at moderate error levels, failing to reach the same low error levels as the other two in reasonable time.

Weight perturbation compared to the other two methods is thus reminiscent of the noisy backpropagation run with the highest noise level compared to the lower-noise runs, in section 3.4.1 above. The suggestion is that a similar mechanism is at work; a moderate amount of noise, as found with REINFORCE, has very little effect on the learning curves, while

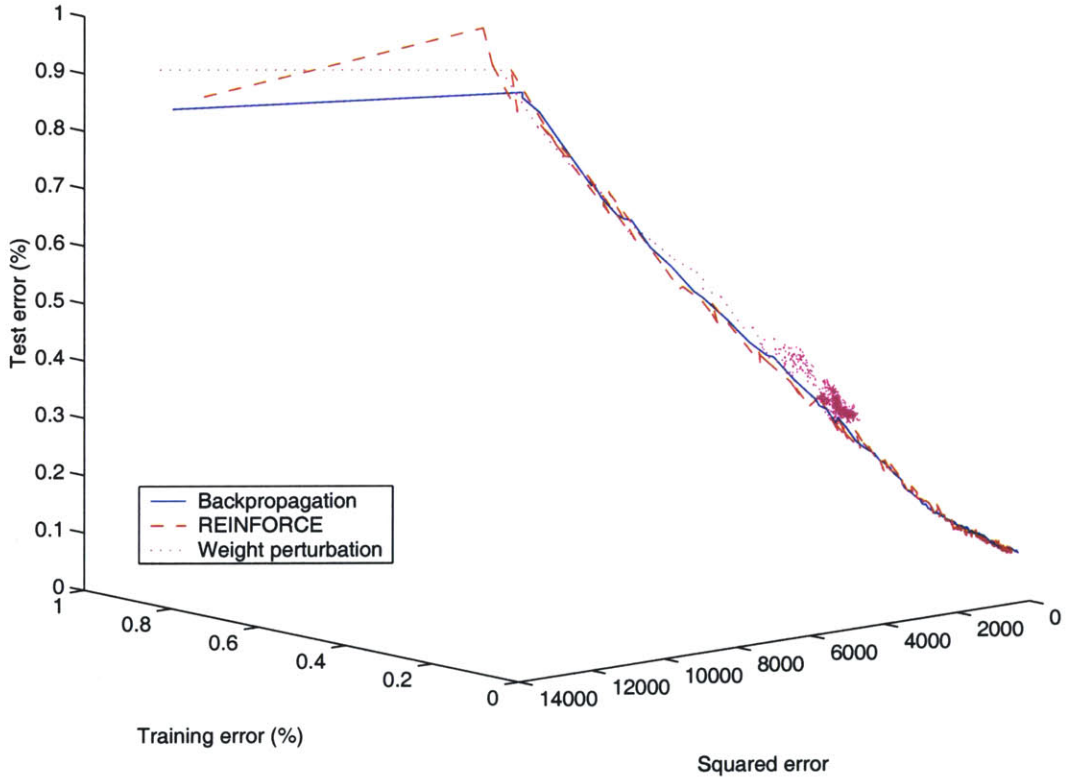

Figure 3-4: Learning as progress through error space, discarding explicit consideration of time, for backpropagation (blue, solid), standard REINFORCE (red, dashed), and weight perturbation (pink, dotted). The three lines are as close or closer than shown here when viewed from other angles.

a much larger noise level allows the network to bypass the plateau at first but ultimately prevents it from reaching the same speed and extent of improvement.

But does REINFORCE really have only a moderate level of noise? And is weight perturbation not so noisy that we might be surprised it works at all? In the case of REINFORCE, there is noise on all units in the hidden and output layers, 59 units in total; hence by the argument given in section 3.2 above, we expect the SNR to be low enough (on the order of 5%) that such favorable performance compared to REINFORCE is quite surprising. The performance of weight perturbation is equally surprising; with one noise source per weight, there are a total of 38965 noise terms, giving an expected SNR on the order of 10^{-5}. The fact that the algorithm is as effective as it is suggests that the noise analysis above may be flawed or incomplete.

3.4.3 Noise analysis

To investigate the noise issue further, we trained networks with different numbers of hidden units, using both REINFORCE and weight perturbation, and recorded the components of the weight update parallel and orthogonal to the gradient (u^2 and $\sum_i v_i^2$, respectively, in the terminology of section 3.2 above). For a fairer noise comparison, we used batch update, in order to eliminate the extra source of noise due to online update.

The procedure in more detail was as follows: Once every 25 epochs, one training example was chosen at random. The actual gradient for that example was calculated, and then sample weight updates were calculated according to the learning rule, for 150 independently chosen sets of values for ξ. Each weight update was decomposed into parallel and orthogonal components with respect to the gradient, and the magnitude and squared magnitude of both components stored, as well as the ratios of the magnitudes and squared magnitudes of the two components. The 150 values for each of these quantities were averaged together, giving values for mean and standard deviation for that example; and averaging over multiple examples, over thousands of epochs, gave overall means that we hope to consider "typical", along with standard deviations on those means.

As shown in Figure 3-5, the average signal-to-noise ratio, as defined by $\langle u^2 \rangle / \langle \sum_i v_i^2 \rangle$, does fall with the number of noise sources n as SNR $\sim n^{-1}$, supporting the analysis of section 3.2; linear regression gives a best-fit line with equation $\log \text{SNR} = -0.98 \log n + 0.89$. Based on this result, it remains surprising that REINFORCE should perform as well as it does, with SNR by this definition so low.

There are multiple quantities that might legitimately be considered to represent signal-to-noise ratio. We chose $\langle u^2 \rangle / \langle \sum_i v_i^2 \rangle$ in section 3.2 because of its analytical tractability; but a measure arguably one step better is $\langle u^2 / \sum_i v_i^2 \rangle$. Using this formula means that we average the ratio of the signal and noise quantities over many trials (and that ratio is what we mean by SNR for a given trial), rather than averaging the two quantities themselves and only then taking the ratio of those means. Figure 3-6 shows the empirical result using this latter measure. However, while the variance of the SNR defined this way is significantly lower, the overall result is the same; the best-fit line has equation $\log \text{SNR} = -1.06 \log n + 0.54$.

We thus have an explanation for why weight perturbation does more poorly than standard REINFORCE at training networks of the same size; and we have reason to expect that these algorithms may encounter serious problems in trying to train much larger networks. However, we still lack a compelling explanation for why REINFORCE should do nearly as well as backpropagation on networks of the size tested, when analysis and experiment suggest that the weight update should be dominated by the component orthogonal to the

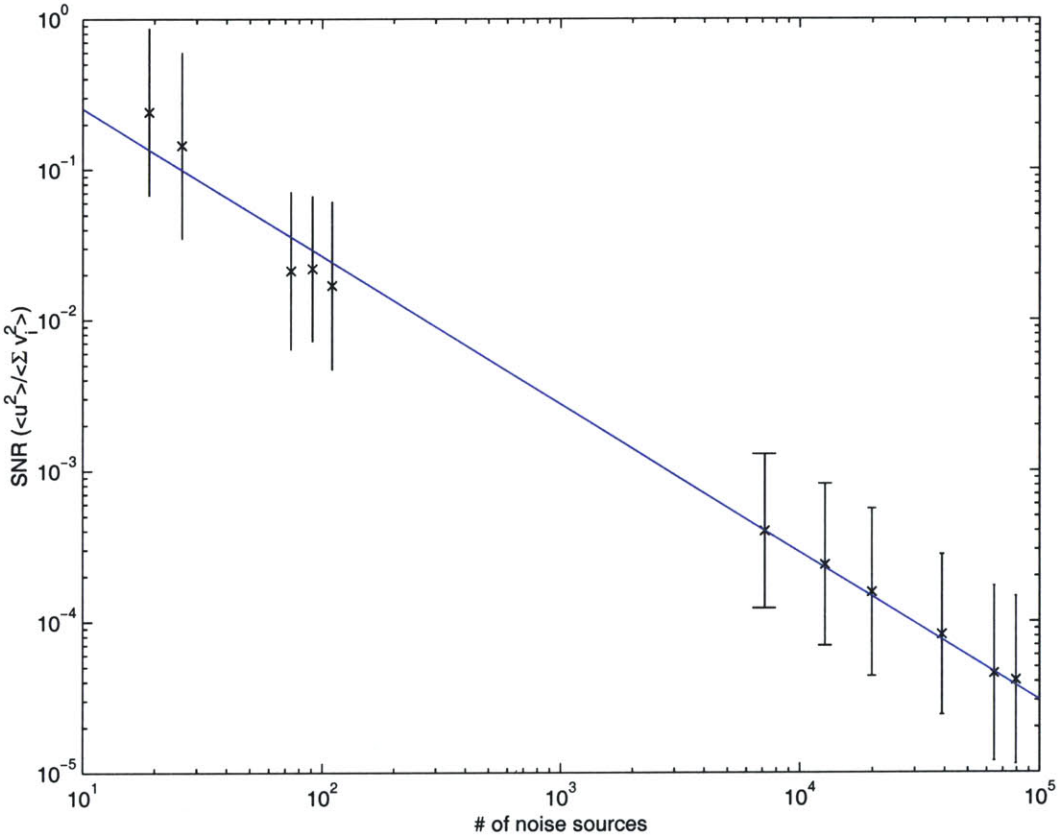

Figure 3-5: Average empirical SNR ($\langle u^2 \rangle / \langle \sum_i v_i^2 \rangle$) versus number of noise sources. The left five data points are due to REINFORCE runs, the right six data points to weight perturbation. The least-squares best-fit line (blue) has slope -0.98 on these logarithmic axes and y-intercept $10^{0.388}$.

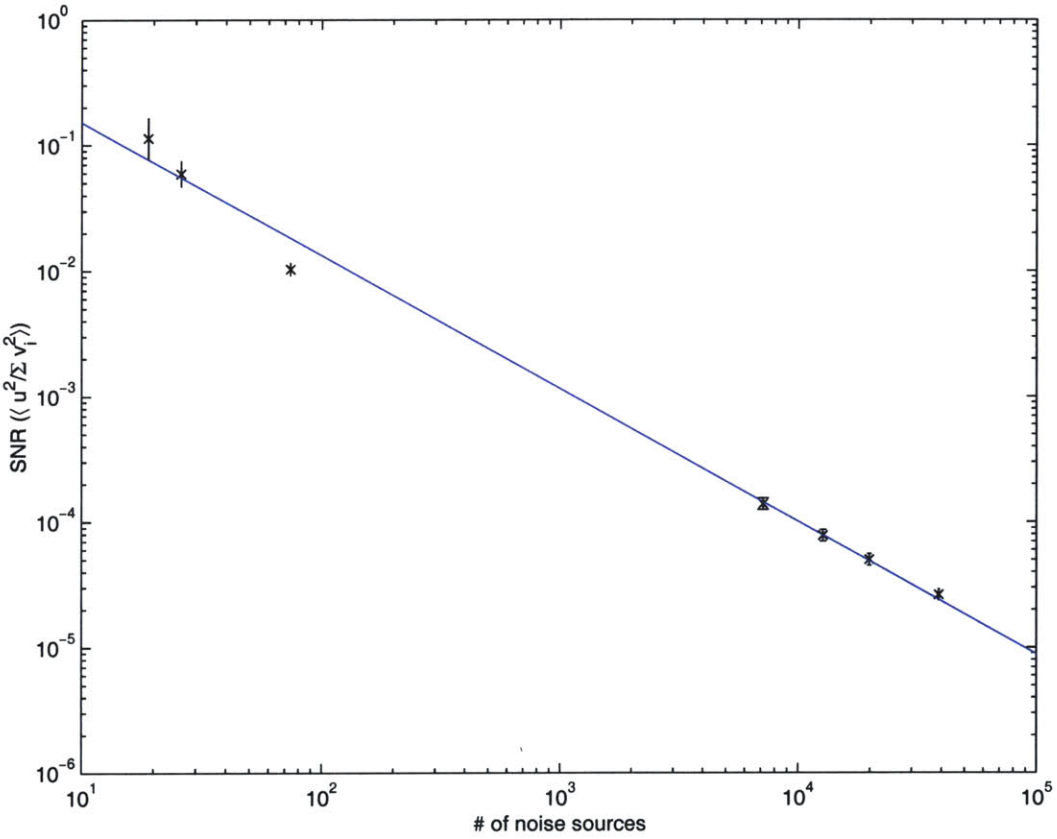

Figure 3-6: Empirical average SNR ($\langle u^2 / \sum_i v_i^2 \rangle$) versus number of noise sources. The left three data points are due to REINFORCE runs, the right four to weight perturbation. The least-squares best-fit line (blue) has slope -1.06 and y-intercept $10^{0.236}$.

true gradient. Even if the error landscape is smooth enough at the scale of the updates that steps orthogonal to the gradient have no effect on the error, the decreasing relative size of the parallel component should decrease the effective step size in the direction of the gradient, and thereby slow the learning curve. One possibility is that the shape of the error landscape is such that in the regime we consider, even steps orthogonal to the gradient reduce the error (as is the case, for instance, for descending on the outside of a cone).

A more likely explanation is that $u^2/\sum_i v_i^2$ is not the most appropriate measure of signal-to-noise ratio, regardless of how the average is taken. The correlations in u and the v_is, due to the same term ξ_1 appearing in each, gave the first indication that another quantity might be more relevant. The much smaller variations in the measure of Figure 3-6 as compared to those in that of Figure 3-5, as reflected in the size of the error bars, also point to the fact that while the parallel and orthogonal components of the weight update vary considerably from trial to trial, they vary together. Thus we have both theoretical and experimental reasons to believe that another quantity is the relevant measure of signal-to-noise ratio that determines the learning performance of the variants of REINFORCE. We do not pursue this issue further here; but it represents the most immediate direction for future work.

3.4.4 Autoencoding

As a second test case, we can consider a similar comparison of the three algorithms on training three-layer feedforward networks on another task, autoencoding. Here the task is to reconstruct the original input image at an output layer of the same size as the input layer, after passing through a hidden layer of many fewer units; the network is forced to encode the most relevant parts of the image at that bottleneck. Using the same data set as before, we then have 784 units in each of the input and output layers, and choose a hidden layer of 100 units. Rather than classification, our task is reconstruction, and so the two relevant error measures are squared error on the training and test sets. Online update is again used. All other parameters are as above.

One approach, clearly, is to do principal components analysis—to have the h hidden units represent the eigenvectors of the data set with the h largest eigenvalues—and this is in fact the approach learned by a network of linear units trained with backpropagation [15]. But as shown in Figure 3-7, networks trained with REINFORCE seem unable to learn to perform this task at all well, at least when given anything like a comparable amount of training time. One possibility is that this failure is due to the much larger number of units with noise added to their inputs in this case—equal to the number of pixels in the image in addition to the number of hidden units, here a total of 884—and on this task, that represents an amount of noise too large for the network to make any significant progress.

Weight perturbation, with its 157684 noise sources in this case, does much worse still, at first; but interestingly, its error goes on to decrease relatively quickly and its performance soon surpasses that of REINFORCE, though the error is still huge compared to that of backpropagation. The reasons behind this improvement are not clear.

It is also conceivable that the error landscape for this network and task is such that improvement lies primarily along narrow channels in weight space, and little gradient information is available elsewhere, so that a noisy learning rule that ranged more widely through weight space would have little success. If this were the case, a reduction in the amount of noise ξ might improve performance. Or the noise level might simply be too high; the results proved above about the error strictly decreasing hold only in the low noise limit,

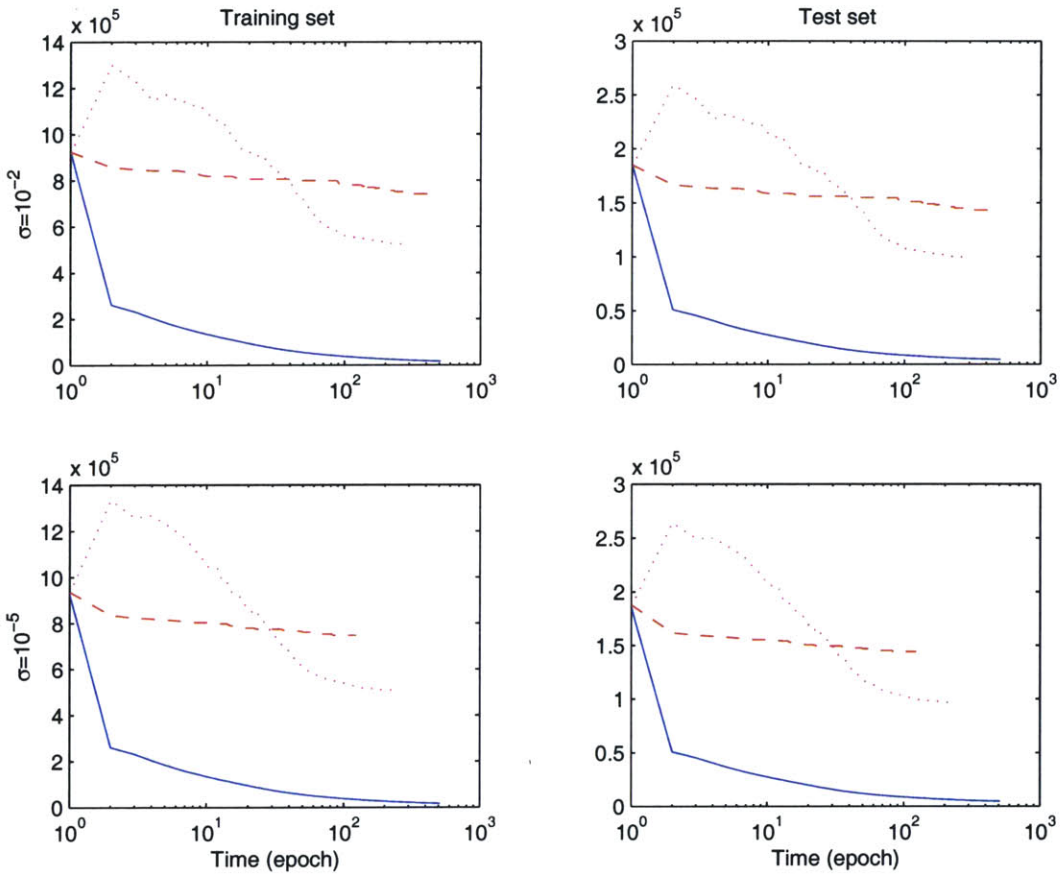

Figure 3-7: Reconstruction (squared) error for autoencoding of the digit images of the abridged mNIST data set, for backpropagation (blue, solid), REINFORCE (red, dashed), and weight perturbation (pink, dotted). Left column, training set; right column, test set; top row, noise with variance $\sigma = 10^{-2}$; bottom row, noise with variance $\sigma = 10^{-5}$.

so that lowering the noise level might be expected to improve performance in any case if REINFORCE is having trouble. However, as shown in Figure 3-7, reducing the standard deviation of the noise from 10^{-2} to 10^{-5} has no qualitative effect at all on the learning curves. Thus other mechanisms may be at work here.

Despite the failure of the two REINFORCE variants on this task, the remarkable performance of standard REINFORCE as compared to backpropagation on the moderate-sized networks considered here, the moderate training success of weight perturbation despite putative signal-to-noise ratios that should be crippling, and the correlations in the noise analysis mentioned in section 3.2, all suggest that continued examination of this algorithm and the mechanisms behind its successes may well prove worthwhile. It is also worth noting that the experiments described above demonstrate the ability of REINFORCE to train networks considerably larger than those typically trained by REINFORCE or weight perturbation in the literature, where networks tend to have on the order of a dozen units at most [2, 4, 10].

3.5 Trajectory learning

Of course, if we would like to be able to use REINFORCE to help explain how, in the zebra finch, RA is able to learn its extended temporal firing patterns, we need to consider the case where the desired output is an entire sequence, not just a single input-output pair. The REINFORCE algorithm can easily be extended to the trajectory-learning case; and the basic result still holds, that the weight update is on average in the direction of the gradient [24]. Unfortunately, the same analysis that gave the results of Equations (3.4)–(3.6) predicts no such result when the desired output is extended in time, as demonstrated in Appendix A.

Although we are not necessarily guaranteed the stronger statement, REINFORCE still has some success in practice at training small networks to learn trajectories. We demonstrate this fact by using REINFORCE to train a network of two neurons to learn a circular trajectory.

The update equation is

$$x(t) = f(Wx(t-1) + \xi(t-1)) \tag{3.23}$$

where ξ is now allowed to vary with time. The REINFORCE prescription is

$$\Delta W = \eta(E_0 - E)\sum_t \xi(t)(x(t))^T \tag{3.24}$$

where $E = \sum_t \sum_i (x_i(t) - d_i(t))^2$ is the squared error for the entire trajectory, delivered to the network only at the end of the trajectory, and E_0 is the squared error for the trajectory in the absence of noise. The desired trajectory was chosen to be 80 points lying around the circumference of a circle of radius 0.5 centered at the origin, completing a single loop. The sigmoidal function was taken to be $f(u) = \tanh(u)$, since hyperbolic tangent, unlike the logistic function, can take on negative values. W was initialized to be the identity matrix. At the beginning of each trial, the trajectory was initialized at the desired starting point. We chose $\sigma = 5 \cdot 10^{-3}$ and $\eta = 1$.

For comparison and proof of feasibility, we also trained the network on the same task

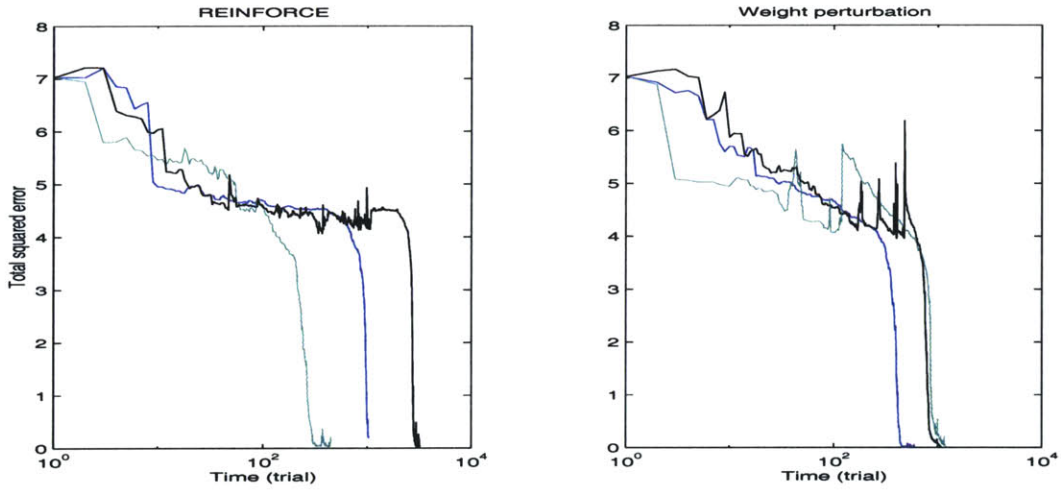

Figure 3-8: Time course of trajectory learning, with squared error learning curves shown for REINFORCE (left) and weight perturbation (right). Different colors represent different runs.

using weight perturbation. Here the update equation was

$$x(t) = f((W + \xi(t))x(t-1)) \tag{3.25}$$

and the weight adjustment was

$$\Delta W = \eta(E_0 - E)\sum_t \xi(t). \tag{3.26}$$

The results of both experiments are shown in Figure 3-8. This demonstration with this toy problem serves as a proof of concept only; we have no strong theoretical result showing that the error should always decrease after each weight adjustment, as we do for the non-trajectory case. Note in fact that the error frequently does increase, and that the time to convergence can be highly variable, for both standard REINFORCE and weight perturbation (on a network this small, we would expect their performance to be comparable).

3.6 Discussion

REINFORCE has a number of attractive features as a model for learning in a system like the brain, or as a training method for a network without a predetermined architecture. It can easily be implemented by a physical network of independent processors, be they neurons or artificial units. Each weight update depends only on purely local quantities, both temporally and spatially, plus a global reward signal; and there is extensive physiological evidence for just such a nonspecific reward signal in the brain, in the form of networks of dopaminergic neurons projecting from the basal ganglia [18]. In the case of trajectory learning, a single accumulator suffices for each weight (which could be realized as, e.g., a trace of calcium concentration local to each synapse, reflecting the course of recent activity), to keep a running total of the quantity equivalent to $\sum_t \xi^t (x^t)^T$ and supply the necessary value for

the weight update when the reward is ultimately delivered.

Reinforcement learning in general is a broad class of training algorithms in which the network is rewarded or punished according to its behavior in response to some input; on the basis of that reward alone, the network adjusts itself so as to maximize the expected reward over time. This approach contrasts with supervised learning, in which training inputs are presented along with the corresponding desired outputs. The latter approach explicitly tells the network what is expected of it; the former requires it to vary its strategy, to explore the available behavior space. In REINFORCE, the noise (ξ, in the case discussed here) is directly responsible for that exploration.

The weight-perturbation variant of REINFORCE suffers compared to the version first introduced because it has more noise sources than in the latter case—generically, N^2 rather than N for a fully recurrent network of N units—and as shown above, the algorithm's performance degrades as the number of noise terms increases. With weight perturbation, variations are explicitly made to each weight, while in the standard version of REINFORCE, a smaller number of variations are used in a coordinated way to construct the weight update. The power of the standard algorithm is in the way that the network's wanderings through weight space are usefully constrained to a much smaller subspace at each step, allowing more rapid and closer convergence to error minima.

It is notable that in the formulation given here, the standard REINFORCE algorithm closely resembles the usual Hebbian prescription for synaptic updates, with one significant substitution. While the Hebbian rule $\Delta W_{ij} \propto x_i x_j$ associates presynaptic activity with postsynaptic activity, the REINFORCE rule $\Delta W_{ij} \propto \xi_i x_j$ associates presynaptic activity with postsynaptic *noise*. This represents a formulation for learning not commonly considered, but one which is successful despite its oddness at first glance. The rule becomes more familiar if one thinks of the postsynaptic noise as a sort of offset activity; if the noise is interpreted as representing something like the postsynaptic activity compared to its baseline level, then the rule becomes Hebbian in appearance.

A clearer intuition about the operation of the REINFORCE algorithm is as follows. If we explicitly include biases in the update equation, writing $x(t+1) = f(Wx(t)+b(t)+\xi(t))$, then we can interpret ξ as noise on the biases b of the units in the network. Each unit i has access to the global offset reward signal $\delta R \equiv R - R_0$, and has a record of the noise ξ_i on its bias that led to that δR. Suppose first that δR and ξ_i are both positive. Then in that trial, the unit acted more excitable than usual, its threshold effectively lower than the noiseless value; as far as it can tell, based on the limited information it has available, it should be more excitable in response to future inputs in order to maximize reward in general. As a result, it strengthens connections to its presynaptic neighbors ($\Delta W_{ij} = \xi_i x_j$, where x_j is always nonnegative), to increase its mean activity in the future. Those neighbors that were most active in the last trial, and hence would likely contribute most strongly to increased activity of unit i for similar inputs in the future, have their connections strengthened the most. Analogous arguments apply when one or both of ξ_i and δR are negative. The interesting thing about the REINFORCE algorithm is that when every neuron behaves in this locally greedy manner, the global reward on average provably increases.

Another interesting point to consider is the fact that neurons are, to some extent, intrinsically noisy units; at the subcellular scale, events are discrete and stochastic. It is possible that evolution never found a way to make cells perfectly reliable, and that life has simply always had to exist with that limitation. But another possibility is that life found a way of exploiting that fact—as tends to happen with evolutionary processes—and made that randomness functional. REINFORCE suggests just such a function. The

approach depends critically on the presence of noise, to provide the exploration required by reinforcement learning; and the noise appears explicitly in the weight update. Thus if the brain is using something like a REINFORCE approach—and recall that there is evidence at least for a global reward signal, for the presence of noise, and of course for strictly local connections—then that suggests that the unreliability of cells in the brain may be a feature crucial to the operation and success of the learning algorithm, rather than an obstacle that needs to be overcome.

For these and other reasons, REINFORCE stands out as an algorithm of interest, worthy of further investigation. Future work remaining to be done includes continued analysis of how REINFORCE manages to avoid being swamped by noise to achieve its success on training moderate-sized networks; additional investigation of why weight perturbation can surpass standard REINFORCE on some tasks, such as autoencoding; and better characterization of why REINFORCE succeeds to the degree it does in the trajectory learning case.

Appendix A

Derivations

A.1 The basic REINFORCE result

Williams showed in [24] that the update rule

$$\Delta W_{ij} = \eta(R - b_{ij})e_{ij} \tag{A.1}$$

did gradient ascent on average on the expected value of the reward function R,

$$\langle \Delta W \rangle = \eta \frac{\partial \langle R \rangle}{\partial W}, \tag{A.2}$$

where the expectation value is over all possible values for the noise ξ. e_{ij}, the characteristic eligibility, is defined as $\partial(\log \Pr_W(s))/\partial W_{ij}$, where $\Pr_W(s)$ is the probability that the network will follow a certain trajectory s for a given input and weight matrix W. The argument ran along the following lines:

$$\langle R(s) \rangle = \sum_s \Pr_W(s) R(s) \tag{A.3}$$

by definition. Since $R(s)$ is taken to be independent of the weight matrix (which affects the probability of a given trajectory s occurring, but not the reward that trajectory receives if it does occur), we have

$$\begin{aligned}
\frac{\partial}{\partial W_{ij}} \langle R \rangle &= \sum_s R(s) \frac{\partial}{\partial W_{ij}} \Pr_W(s) && \text{(A.4)} \\
&= \sum_s R(s) \Pr_W(s) \frac{\partial}{\partial W_{ij}} \log \Pr_W(s) && \text{(A.5)} \\
&= \langle R(s) \frac{\partial}{\partial W_{ij}} \log \Pr_W(s) \rangle && \text{(A.6)} \\
&\equiv \langle R(s) e_{ij} \rangle. && \text{(A.7)}
\end{aligned}$$

If we choose $R(s) = 1$, we obtain

$$\frac{\partial}{\partial W_{ij}} \langle 1 \rangle = 0 = \langle e_{ij} \rangle \tag{A.8}$$

from which we see that $\langle b_{ij}e_{ij}\rangle = 0$ so long as b_{ij} and e_{ij} are uncorrelated. The result

$$\langle \Delta W_{ij}\rangle = \langle \eta(R - b_{ij})e_{ij}\rangle = \eta\frac{\partial\langle R\rangle}{\partial W_{ij}} \tag{A.9}$$

immediately follows.

Note that maximizing a reward is equivalent to minimizing an error; the error E in the main body of the text is identical to the negative of the reward R here.

A.2 Eligibility for continuous-valued units with Gaussian noise

For the specific case treated in Chapter 3, we have

$$\mathrm{Pr}_W(s) = \mathrm{Pr}(x(t+1)|x(t)) = \mathrm{Pr}'(\xi) = K\exp\left(-\frac{\xi^2}{2\sigma^2}\right) \tag{A.10}$$

since ξ is a Gaussian-distributed random variable with variance σ^2; K is a normalization constant. Then

$$\begin{aligned}
\log \mathrm{Pr}_W(s) &= \log K - \xi^2/2\sigma^2 && \text{(A.11)}\\
e_{ij} &\equiv \frac{\partial}{\partial W_{ij}}\log \mathrm{Pr}_W(s) && \text{(A.12)}\\
&= -\frac{\xi_i}{\sigma^2}\frac{\partial}{\partial W_{ij}}\xi_i && \text{(A.13)}\\
&= -\frac{\xi_i}{\sigma^2}\frac{\partial}{\partial W_{ij}}\left[f^{-1}(x_i(t+1)) - \sum_j W_{ij}x_j(t)\right] && \text{(A.14)}\\
&= \frac{1}{\sigma^2}\xi_i x_j(t) && \text{(A.15)}
\end{aligned}$$

The variance σ^2 is absorbed into the learning rate η.

A.3 Weight adjustment in the low noise limit

In the absence of noise, the network is presented with an input and thereafter reaches a fixed point $x = f(Wx)$. When the noise is added, the network comes to a new fixed point, $x' = f(Wx' + \xi)$. $\delta x \equiv x' - x$ is the linear response of the network to the small perturbation ξ. Expanding to first order and using the operator $\times$ to refer to multiplication of corresponding elements in a vector, we have

$$\begin{aligned}
x' &= f(W(x+\delta x) + \xi) && \text{(A.16)}\\
&= f(Wx) + f'(Wx)\times(W\delta x + \xi) && \text{(A.17)}\\
\Rightarrow \delta x &= f'(Wx)\times(W\delta x + \xi). && \text{(A.18)}
\end{aligned}$$

Taking the derivative of both sides of the fixed-point equation $x = f(Wx)$ with respect to x gives

$$1 = f'(Wx)\frac{\partial(Wx)}{\partial x} \tag{A.19}$$

$$\Rightarrow \frac{1}{f'(Wx)} = \frac{\partial}{\partial x} f^{-1}(x), \tag{A.20}$$

so that defining

$$D \equiv \mathrm{diag}(f^{-1\prime}(x)), \tag{A.21}$$

we have

$$(D - W)\delta x = \xi \tag{A.22}$$

which gives the linear response of the network to the perturbation ξ in terms of its fixed point:

$$\delta x = (D - W)^{-1}\xi. \tag{A.23}$$

Next, in order to maximize the reward subject to the fixed-point constraint $x = f(Wx)$, we write the Lagrangian

$$\mathcal{L}(W, x(W), y(W)) = R(x) + y^T(Wx - f^{-1}(x)) \tag{A.24}$$

where y is the vector of Lagrange multipliers, one for each unit in the network. We can write for independent variations in x and y

$$\delta\mathcal{L} = \left(\frac{\partial R}{\partial x}\right)^T \delta x + (\delta y)^T(Wx - f^{-1}(x)) + y^T(W\delta x - f^{-1\prime}(x) \times \delta x) \tag{A.25}$$

$$= \left(\frac{\partial R}{\partial x}\right)^T \delta x + (\delta y)^T(Wx - f^{-1}(x)) + y^T((W - D)\delta x). \tag{A.26}$$

Setting the derivative of $\mathcal{L}$ with respect to y to 0,

$$\frac{\partial \mathcal{L}}{\partial y} = Wx - f^{-1}(x) = 0, \tag{A.27}$$

enforces the fixed-point constraint.

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial R}{\partial x} + (W^T - D^T)y; \tag{A.28}$$

setting

$$y = (D^T - W^T)^{-1}\partial R/\partial x \tag{A.29}$$

enforces $\partial\mathcal{L}/\partial x = 0$.

$$\frac{d\mathcal{L}}{dW} = \frac{\partial \mathcal{L}}{\partial W} + \frac{\partial \mathcal{L}}{\partial x}\frac{\partial x}{\partial W} + \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial W} \tag{A.30}$$

$$= \frac{\partial \mathcal{L}}{\partial W} \tag{A.31}$$

$$= \frac{\partial R}{\partial W} \tag{A.32}$$

since at the fixed point of the dynamics, $\mathcal{L} = R$.

Written in terms of components,

$$\mathcal{L} = R + \sum_i y_i \left(f(\sum_j W_{ij}x_j) - x_i \right), \tag{A.33}$$

so that

$$\frac{\partial R}{\partial W_{ij}} = y_i x_j, \quad \text{or} \tag{A.34}$$

$$\frac{\partial R}{\partial W} = y x^T. \tag{A.35}$$

Next choose b_{ij} to be R_0, the reward the network would receive for coming to the noiseless fixed point x, for all i and j. Then the weight update rule is

$$\begin{aligned}
\Delta W_{ij} = & \quad \eta(R - R_0)\xi_i x_j & & (A.36)\\
\approx & \quad \eta(\textstyle\sum_k (R'(x_k))\delta x_k)\xi_i x_j & & (A.37)\\
= & \quad \eta \textstyle\sum_k (\partial R/\partial x_k) \left[\sum_h (D_{kh} - W_{kh})^{-1}\xi_h\right] \xi_i x_j & \text{(using (A.23))} & (A.38)\\
= & \quad \eta \textstyle\sum_h y_h \xi_h \xi_i x_j & \text{(using (A.29))} & (A.39)\\
= & \quad \eta \xi_i \textstyle\sum_h \xi_h \frac{\partial R}{\partial W_{hj}} & \text{(using (A.34)).} & (A.40)
\end{aligned}$$

This is the result given in Equation (3.3).

The main body of the text uses this result to show that ΔW is always within 90 degrees of the true gradient $\partial R/\partial W$. Another way to recognize this fact from Equation (A.40) is to note that $\hat{e}\hat{e}^T\vec{x}$ is the projection of a vector $\vec{x}$ onto a unit vector $\hat{e}$. Thus, each column of the matrix ΔW_{ij} is the projection of the corresponding column of the gradient matrix (all connections originating at presynaptic unit j) onto the random noise vector ξ, multiplied by $\eta|\xi|^2$, and therefore necessarily within 90 degrees of that column of the gradient matrix. Since each column of ΔW then has a positive dot product with the corresponding column of $\partial R/\partial W$, it follows that the dot product of the two complete matrices must likewise be positive.

A.4 Extension to trajectory learning

The extension to the trajectory case is straightforward. Suppose the reward is a function of every state the system goes through from time 1 to time t, and let a superscript represent the time index. ξ may vary as a function of time. We also include an explicit (time-varying) bias term in the update equation for each unit, $x^{s+1} = f(Wx^s + b)$, for reasons that will become clear shortly. Then we can write

$$\begin{aligned}
\mathcal{L} = & \quad R(x^1, x^2, \cdots, x^t) + (y^1)^T(Wx^1 + b^1 + \xi^1 - f^{-1}(x^2)) + \cdots & (A.41)\\
& + (y^{t-1})^T(Ws^{t-1} + b^{t-1} - f^{-1}(x^t)) & (A.42)
\end{aligned}$$

In this way, ξ can be seen as noise on the biases of the units. Again, $\partial\mathcal{L}/\partial y = 0$ enforces the update equation. We have

$$\frac{\partial \mathcal{L}}{\partial b^s} = y^s \tag{A.43}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial R}{\partial W} = \sum_{s=1}^{t-1} y^s (x^s)^T \tag{A.44}$$

$$\delta R = \sum_s \left(\frac{\partial R}{\partial b^s}\right)^T \xi^s = \sum_s (y^s)^T \xi^s \tag{A.45}$$

The REINFORCE prescription for trajectory learning is [24]

$$\Delta W_{ij} = \eta(R - b_{ij}) \sum_s e_{ij}^s \tag{A.46}$$

so that in the case treated here, we have

$$\Delta W = \eta \delta R \sum_s \xi^s (x^s)^T \tag{A.47}$$

$$= \eta \sum_{r,s} (y^r)^T \xi^r \xi^s (x^s)^T \tag{A.48}$$

Since $(y^r)^T \xi^r$ is just a scalar, we can write

$$\Delta W_{ij} = \eta \sum_{r,s} \xi_i^s [(\xi^r)^T y^r] x_j^s \tag{A.49}$$

$$\frac{\partial R}{\partial W_{ij}} = \sum_m y_i^m x_j^m \tag{A.50}$$

$$\Delta W \cdot \frac{\partial R}{\partial W_{ij}} = \sum_{i,j} \Delta W_{ij} \frac{\partial R}{\partial W_{ij}} = \sum_{i,j,k,m,r,s} \xi_i^m \xi_k^r y_i^s y_k^r x_j^m x_j^s \tag{A.51}$$

Unfortunately, in this form we cannot make the same statement as we did in Chapter 3 about this quantity being a sum of squares. While this weight update still does gradient ascent on the expected reward on average,

$$\langle \Delta W_{ij} \rangle = \eta \sum_{r,s} \sum_k \langle \xi_i^s \xi_k^r \rangle y_k^r x_j^s \tag{A.52}$$

$$= \eta \sum_{r,s} \sum_k \delta_{ik} \delta_{rs} y_k^r x_j^s \tag{A.53}$$

$$= \eta \sum_s y_i^s x_j^s \tag{A.54}$$

$$= \eta \frac{\partial R}{\partial W_{ij}}, \tag{A.55}$$

it is not clear whether every update is within 90 degrees of the true gradient, as it was in the non-trajectory case.

A.5 Noise analysis

For cleaner notation, consider again only outputs fanning out from a single unit, so that the second subscript is implied for matrices; and write $y \equiv \eta \frac{\partial R}{\partial W}$. The REINFORCE prescription for the weight update, derived in (A.40) above, can then be written as

$$\Delta W_i = \xi_i \sum_h \xi_h y_h. \tag{A.56}$$

Define $U_{||}$ to be the component of the weight update in the direction of the actual reward gradient y, and $U_{\perp}$ to be the orthogonal component:

$$U_{||} = \left(\Delta W \cdot \frac{y}{|y|}\right)\frac{y}{|y|} \tag{A.57}$$

$$U_{\perp} = \Delta W - U_{||} \tag{A.58}$$

We then have

$$|U_{||}|^2 = \frac{(\sum_i \xi_i y_i \sum_h \xi_h y_h)^2}{\sum_i y_i^2} \tag{A.59}$$

$$|U_{\perp}|^2 = \sum_i \left(\xi_i \sum_h \xi_h y_h - (\sum_h \xi_h y_h)^2 \frac{y_i}{|y|^2}\right) \tag{A.60}$$

$$= \sum_i \xi_i^2 (\sum_h \xi_h y_h)^2 - 2(\sum_i \xi_i y_i)^4 \frac{1}{|y|^2} + (\sum_h \xi_h y_h)^4 \sum_i y_i^2 \frac{1}{|y|^4} \tag{A.61}$$

$$= \sum_i \xi_i^2 (\sum_h \xi_h y_h)^2 - \frac{(\sum_i \xi_i y_i)^4}{\sum_i y_i^2} \tag{A.62}$$

It is easy to show that the sum of a set of Gaussian-distributed random variables is itself a Gaussian-distributed random variable, with mean equal to the sum of the means of its elements, and variance equal to the sum of the variances. In particular, we can define $z \equiv \sum_i \xi_i y_i$, which is thus Gaussian-distributed with mean 0 and variance $\sigma^2 \sum_i y_i^2$.

When the expectation value over all values of noise is taken, all terms with ξ to an odd power vanish. We thus obtain

$$\langle |U_{||}|^2 \rangle = \frac{\langle z^4 \rangle}{\sum_i y_i^2} \tag{A.63}$$

$$= 3 \sum_i y_i^2 \tag{A.64}$$

$$\langle |U_{\perp}|^2 \rangle = \sum_i \langle \xi_i^4 \rangle y_i^2 + \sum_i \sum_{j \neq i} \langle \xi_i^2 \xi_j^2 \rangle y_j^2 - \frac{\langle z^4 \rangle}{\sum_i y_i^2} \tag{A.65}$$

$$= 3 \sum_i y_i^2 + (n-1) \sum_i y_i^2 - 3 \sum_i y_i^2 \tag{A.66}$$

$$= (n-1) \sum_i y_i^2 \tag{A.67}$$

We can consider $\langle |U_{||}|^2 \rangle$ to be an average signal and $\langle |U_{\perp}|^2 \rangle$ to be an average noise; then $\langle |U_{||}|^2 \rangle / \langle |U_{\perp}|^2 \rangle$, one possible measure of a signal-to-noise ratio, scales up as $1/n$ with increasing network size. Note that the quantity $\langle |U_{||}|^2 / |U_{\perp}|^2 \rangle$ would be a better indicator of average SNR, but simplifying that expression into an illuminating form is not straightforward. Section 3.4 looks at both quantities empirically, and finds that both scale as $1/n$ with network size, though the variance of the latter across trials is much lower.

Because $U_{\perp}$ has mean 0 and $U_{||}$ has mean y, we might subtract the mean from each quantity before using it, $U'_{||} = U_{||} - \langle U_{||} \rangle$, $U'_{\perp} = U_{\perp} - \langle U_{\perp} \rangle$, to consider fluctuations about the mean. The same approach as above then gives the same result, $\langle |U'_{||}|^2 \rangle / \langle |U'_{\perp}|^2 \rangle \sim 1/n$.

n is the number of *noise sources* in the network. Here that quantity is equal to the number of cells; in the weight perturbation variant of the REINFORCE algorithm described

elsewhere, it is equal to the number of weights. For a network of N cells, the latter quantity scales up as N^2, helping to account for the significantly better performance of the standard REINFORCE algorithm as compared to weight perturbation.

A.6 Weight perturbation

In this section we show that the weight perturbation algorithm described in the text actually does belong to the class of REINFORCE algorithms, rather than just sharing certain surface characteristics.

Define

$$u_{ij}(t) \equiv (W_{ij} + \xi_{ij}(t))x_j(t) \tag{A.68}$$

Then the update equation can be written

$$x_i(t+1) = f(\sum_j u_{ij}(t)) \tag{A.69}$$

The above equation shows that we can write the network trajectory entirely in terms of u; thus we can write the reward, which is a function only of trajectory, as $R(u)$. The derivation of section A.1 now applies again, with s replaced everywhere by u; the network evolution is still Markovian and the components of u at a given time are conditionally independent, so that $\log \Pr_W(u) = \sum_t \sum_{i,j} \log \Pr_W(u_{ij}(t+1)|u(t))$. In this case,

$$\begin{aligned}
e_{ij} &\equiv \frac{\partial}{\partial W_{ij}} \log \Pr_W(u) && \text{(A.70)}\\
&= \frac{\partial}{\partial W_{ij}} \sum_t \sum_{h,k} \log \Pr_W(u_{hk}(t+1)|u(t)) && \text{(A.71)}\\
&= \sum_t \frac{\partial}{\partial W_{ij}} \log \left(\frac{u_{ij}(t)}{x_j(t)} - W_{ij} \right) && \text{(A.72)}\\
&= \sum_t \frac{\partial}{\partial W_{ij}} \left(-\frac{\xi_{ij}^2(t)}{2\sigma^2} \right) && \text{(A.73)}\\
&= \sum_t \frac{1}{\sigma^2} \xi_{ij}(t), && \text{(A.74)}
\end{aligned}$$

giving the update equation for weight perturbation. In the non-trajectory case, where ξ is fixed with respect to time and $R(u)$ only depends on the final state, the sum over time is omitted.

Bibliography

[1] Abeles, M. *Corticonics: Neuronal Circuits of the Cerebral Cortex.* Cambridge University Press, 1991.

[2] Bartlett, P., and J. Baxter. Hebbian synaptic modifications in spiking neurons that learn. Technical report, November 27, 1999.

[3] Brainard, M., and A. Doupe. Interruption of a basal ganglia-forebrain circuit prevents plasticity of learned vocalizations. *Nature* **404**:762–766, 2000.

[4] Cauwenberghs, G. An analog VLSI recurrent neural network learning a continuous-time trajectory. *IEEE Transactions on Neural Networks* **7**(2):346–361, 1996.

[5] Dave, A. and D. Margoliash. Song replay during sleep and computational rules for sensorimotor vocal learning. *Science* **290**:812–816, 2000.

[6] Dehaene, S., J.-P. Changeux, and J.-P. Nadal. Neural networks that learn temporal sequences by selection. *PNAS USA* **84**:2727–2731, 1987.

[7] Doupe, A. Song- and order-selective neurons in the songbird anterior forebrain and their emergence during vocal development. *J. Neurosci.* **17**(3):1147–1167, 1997.

[8] Doupe, A. and M. Solis. Song- and order-selective neurons develop in the songbird anterior forebrain during vocal learning. *J. Neurobiol.* **33**(5):694–709, 1997.

[9] Fee, M. Personal communications.

[10] Flower, B. and Jabri, M. Weight perturbation: an optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks. *IEEE Transactions on Neural Networks* **3**(1):154–157, 1992.

[11] Hermann, M., J. Hertz, and A. Prugel-Bennett. Analysis of synfire chains. *Network* **6**(3):403–414, 1995.

[12] Hertz, J., A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[13] Hertz, J. and A. Prügel-Bennett. Learning short synfire chains by self-organization. *Network: Computation in Neural Systems* **7**:357–363, 1996.

[14] Hopfield, J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, **79**(8):2554-8, 1982.

[15] Levine, J., and H. S. Seung. Unpublished result.

[16] Movellan, J. R. Contrastive Hebbian learning in the continuous Hopfield model. In D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton: *Connectionist Models: Proceedings of the 1990 Summer School.* Morgan Kaufmann, 1991.

[17] Oh, J.-H., K. Kang, C. Kwon, and Y. Park. Generalization in two-layer neural networks. In J.-H. Oh, C. Kwon, and S. Cho: *Neural Networks: The Statistical Mechanics Perspective.* World Scientific, 1995.

[18] Schultz, W. Dopamine neurons and their role in reward mechanisms. *Curr. Op. Neurobiol.* **7**:191–197, 1997.

[19] Spiro, J., M. B. Dalva, and R. Mooney. Long-range inhibition within the zebra finch song nucleus RA can coordinate the firing of multiple projection neurons. *J. Neurophys.* **81**(6):3007–3020, 1999.

[20] Troyer, T. and A. Doupe. An associational model of birdsong sensorimotor learning I. Efference copy and the learning of song syllables. *J. Neurophys.* **84**(3):1204–1223, 2000.

[21] Troyer, T. and A. Doupe. An associational model of birdsong sensorimotor learning II. Temporal hierarchies and the learning of song sequence. *J. Neurophys.* **84**(3): 1224–1239, 2000.

[22] Vicario, D. Contributions of syringeal muscles to respiration and vocalization in the zebra finch. *J. Neurobiol.* **22**(1):63–73, 1991.

[23] Warner, R. The anatomy of the syrinx in passerine birds. *J. Zool., Lond.* **168**:381–393, 1972.

[24] Williams, R. Simple statistical gradient-following algorithnms for connectionist reinforcement learning. *Machine Learning* **8**:229–256, 1992.

[25] Xie, X. and H. S. Seung. Spike-based learning rules and stabilization of persistent neural activity. *Advances in Neural Information Processing Systems* **12**, 2000.

[26] Yu, A. C. and D. Margoliash. Temporal hierarchical control of singing in birds. *Science* **273**:1871–1875, 1996.