The Role of Sensitivity Derivatives in Sensorimotor Learning

by

Mohamed Nabeel Abdelghani

A thesis submitted in conformity with the requirements

for the degree of Doctor of Philosophy

Graduate Department of Physiology

University of Toronto

© Copyright by Mohamed Nabeel Abdelghani 2011

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-97694-4

Our file Notre référence
ISBN: 978-0-494-97694-4

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distrbute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protege cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

The Role of Sensitivity Derivatives in Sensorimotor Learning

Mohamed Nabeel Abdelghani

Doctor of Philosophy

Graduate Department of Physiology
University of Toronto

2011

Abstract

To learn effectively, an adaptive controller needs to know its sensitivity derivatives — the variables that quantify how system performance depends on the commands from the controller. In the case of biological sensorimotor control, no one has explained how those derivatives themselves might be learned, and some authors suggest they aren't learned at all but are known innately. Here I show that this knowledge can't be *solely* innate, given the adaptive flexibility of neural systems. And I show how it could be learned, using forms of information transport that are available in the brain, by a mechanism I call implicit supervision.

I show that implicit supervision explains a wide range of otherwise-puzzling facts about motor learning. It explains how we can cope with conditions that reverse the signs of sensitivity derivatives, e.g. nerve or muscle transpositions, reversing goggles, or tasks like drilling teeth seen in a mirror. It also explains why it is harder to recover from reversals than from other alterations such as magnifying, minifying or displacing goggles.

A further prediction of the theory of implicit supervision, in its simplest form, is that each control system — say for gaze stabilization, or saccades, or reaching — has one single, all-purpose estimate of its sensitivity derivatives for all parts of the motion. When that estimate is revised, it should affect all stages of the task. For instance, when you learn to move to mirror-reversed targets then your adapted estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ should reverse not only your initial aiming but also your online course adjustments: when the target jumps in mid-movement, your path adjustment should be appropriately reversed. Here I put subjects through many trials with jumping targets, and show that, given enough practice, they do learn to reverse their course adjustments, and therefore both initial aiming and later adjustments are governed by revisable estimates of sensitivity derivatives. And I argue that all the available data, from my own experiments and earlier ones, are compatible with a single, adaptable, all-purpose estimate of these derivatives, as in the simplest form of implicit supervision.

Acknowledgments

I am deeply grateful and will surely be indebted all my life to my mentor Dr. Douglas Tweed for his dedication to my training and general well-being. He has instilled in me his love for research, insight in neuroscience, coherence of ideas, focus and attention to detail. It has been a great pleasure and a challenging experience to work with Doug on this thesis.

I thank my committee members, Dr. Dianne Broussard and Dr. William MacKay, for their suggestions, which improved my work. And, for their comments I am grateful to Dr. Hon Kwan, Dr. Manfredi Maggiore, Dr. Ken Norwich, Dr. Francis Skinner, and Dr. Martin Wojtowicz.

A special thanks to my colleagues Dr. Lak Chinta, Kristen Fortney, and Timothy Lillicrap. I would also like to thank our lab's research assistant Danitza Goche for making sure that I got paid and for helping with my experiments.

I am grateful to my father Nabeel Abdelghani and my mother Soad Anastasiou for their love, support and encouragement. I would also like to thank my sister Lulu and my brother Mohanned for their love. I dedicate this work to my grandmother Fatima.

I would also like to acknowledge the support of the Canadian Institutes of Health Research for this work.

Table of Contents

Table of Symbols

$\|\|\ \|\|$	Size of a vector, e.g. $\|\|\boldsymbol{e}\|\|$ is the size of the error vector $\boldsymbol{e}$
$\times$	Cartesian product of spaces
$\langle\ \rangle$	Estimate, e.g. $\langle \boldsymbol{x} \rangle$ is an estimate of $\boldsymbol{x}$
AE	Adjustment error
AL	Adjustment latency
D	Domain where the sensitivity derivatives are defined
$d\boldsymbol{e}/d\boldsymbol{z}$	Jacobian of sensorimotor system
$\partial\boldsymbol{e}/\partial\boldsymbol{u}$	Matrix of sensitivity derivatives, also known as the control jacobian
$\partial L/\partial\boldsymbol{u}$	Gradient of loss with respect to motor commands
$\partial L/\partial\boldsymbol{w}$	Gradient of loss with respect to the controller's adjustable parameters
$\boldsymbol{e}$	Vector of performance errors, e.g. $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}^*$
$\bar{\boldsymbol{e}}$	Plant model error, $\bar{\boldsymbol{e}} = \langle \dot{\boldsymbol{e}} \rangle - \dot{\boldsymbol{e}}$
η	Learning rate parameter
$\boldsymbol{f}$	Function of a controlled object or plant
$\boldsymbol{g}$	Performance error function
γ	Controller function

$\dot{\boldsymbol{h}}$	Head velocity
k	Mechanical stiffness
L	Loss, i.e. the quantity to be minimized, defined to be $\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}/2$
$\bar{L}$	Model loss
L_p	Perturbed loss
LE	Launch error
LIP	Linear in the parameters (a class of learning algorithms)
LL	Launch latency
LMS	The least mean square online learning algorithm
LTD	Long-term depression
LTP	Long-term potentiation
l	Layer in a neural network
NLMS	Normalized least mean square online learning algorithm
$\boldsymbol{\varphi}(\boldsymbol{z})$	Vector of all the features $\varphi_i(\boldsymbol{z})$, called the feature vector
ρ	Viscosity
$^{\mathsf{T}}$	Transpose, e.g. $\boldsymbol{A}^{\mathsf{T}}$ is the transpose of the matrix $\boldsymbol{A}$, obtained by flipping the matrix so the rows of $\boldsymbol{A}$ are the columns of $\boldsymbol{A}^{\mathsf{T}}$ and vice versa

$\boldsymbol{T}_{new}$	New target location on the computer screen
$\boldsymbol{T}_{prev}$	Previous target location on the screen
tanh	Hyperbolic tangent function
U	Space of motor commands
$\boldsymbol{u}$	Motor command vector
$\boldsymbol{u}_p$	Perturbed command
VOR	Vestibulo-ocular reflex
$\boldsymbol{v}$	Context vector
$\boldsymbol{W}$	Synaptic weight matrix in implicit supervisor network
$\boldsymbol{w}$	Vector of internal controller parameters
X	State space of the plant
X^*	Space of target states
$\boldsymbol{x}$	Plant state vector
$\boldsymbol{x}^*$	Desired state vector, also called the goal or reference
y	Output signal of a neural network
$\boldsymbol{z}$	Vector of all inputs to the plant, i.e. $\boldsymbol{z} = (\boldsymbol{v}, \boldsymbol{u})$

List of Figures

List of Appendices

Chapter 1

Scope of the thesis

Learning shapes most if not all the brain's structures and functions, including motor behavior. Most motor skills are not inborn but develop over the first few years of life. And many continue to improve with practice, especially in the case of complex tasks such as writing or playing the piano. Also, after injury, adaptive mechanisms in the brain can often restore function.

The aim of this thesis is to discover computational principles that underlie sensorimotor learning. Advances here will help unify many diverse branches of brain theory, and may lead to improved rehabilitation after damage to the brain, sensors or muscles, and to improved motor skill training in children and athletes, and eventually to more versatile robots.

Because the problem is complex, I will try to make it tractable by focusing on learning in simple sensorimotor systems such as the vestibulo-ocular reflex (VOR) and reaching, in the hope that what I discover here will generalize to other brain functions.

I am concerned with computational principles of sensorimotor learning. To this end I make heavy use of concepts from control theory, because sensorimotor skills are essentially control tasks. For example in reaching, motor commands issued by the brain activate muscles to move the arm to the target. And in the VOR, extraocular muscles are activated by the brain to rotate the eyeballs to stabilize the visual image on the retina as we move in the world. From this point of view, the brain is a *controller* and the body and external world are its controlled objects or

plants. But the brain is not just any controller; the brain's neural circuits that steer our motor behavior are a*daptive controllers*, in the terminology of control theory, because the brain can adapt to new and uncertain environments. Moreover, the brain's capacity for adaptation far exceeds that of any artificial control system. My aim in this thesis is to apply ideas from adaptive control theory to try to understand sensorimotor control and learning in the brain.

But adaptive control methods that are popular in engineering must usually be modified before they can provide viable theories of adaptive control in the brain. Engineering methods rely on forms of information transfer that are not possible in the brain. Also, most industrial control systems don't need to be as versatile and autonomous as animals are, so industrial control methods often ignore information necessary for flexible adaptation. Therefore, in trying to understand sensorimotor learning, I need to take into account biological constraints, devising novel adaptive control algorithms that are biologically plausible.

In this thesis I will describe two projects. The first is theoretical, addressing the following two fundamental questions:

1. To learn effectively, an adaptive controller needs to know its sensitivity derivatives — the variables that quantify how system performance depends on the commands from the controller. This is true also for biological sensorimotor control. But is the brain's knowledge of sensitivity derivatives innate, or is it learned?

2. If knowledge of sensitivity derivatives is acquired rather than solely innate then how can the sensitivity derivatives be learned by neural controllers, given the computational complexity of computing these derivatives and the biological constraints on information flow in the brain?

In this first project I show that, in at least some sensorimotor systems, sensitivity derivatives are learned, and that this fact explains, among other things, how people can learn to reach for targets when the relation between vision and motion is reversed, as with mirrors or reversing glasses. I also suggest a mechanism, called implicit supervision, for how the brain could learn sensitivity derivatives.

My second project is an experiment aimed at testing a further prediction of the theory of implicit supervision. That theory suggests that adjustable estimates of sensitivity derivatives should govern all aspects of a task. For instance, when you learn to move to mirror-reversed targets, your adapted estimate of those derivatives should reverse both your initial aiming and your online course adjustments: when the target jumps in mid-movement, your path adjustment should be appropriately reversed.

Here I describe a series of experiments that test whether subjects, if given ample practice tracking mirror-reversed targets that jump in mid-movement, and ample feedback on their performance, can learn to reverse their online corrections, as predicted by my theory of implicit supervision from Project 1. And I consider whether the data imply separate estimates of sensitivity derivatives for different stages of the task such as launch and adjustment, or a single all-purpose estimate governing all stages.

In both these projects, two concepts are pivotal. The first is that of sensitivity derivatives, also known in the control literature as the control jacobian. I will define this notion in the next section and discuss it in detail in Chapter 3. The other concept is that of biological constraints on algorithms, an issue that arises when we try to determine whether mathematical ideas apply to neuronal tissue. I will discuss this issue in Section 1.2 and later in Chapter 2.

1.1 Sensitivity derivatives

Sensitivity derivatives quantify how motor commands $\boldsymbol{u}$ affect performance errors $\boldsymbol{e}$. For example, suppose that your vestibulo-ocular reflex, or VOR, is malfunctioning, so that you experience horizontal retinal slip e_h in the positive direction, where I define "positive" to mean that the visual field is moving rightward from the subject's point of view (of course that means the image is actually moving to the left across the retinal surface, owing to optical inversion by the ocular lens, but I will keep to the convention of describing the motion in terms of the subject's impression). At this point the adaptive mechanisms in your VOR will step in, adjusting your synapses so as to increase (eventually) the activation of your rightward-pulling muscles, the medial rectus of the left eye and the lateral rectus of the right, and so reduce the slip. Giving the name u_h to the motor commands to these muscles, we can say that the brain increases the command u_h to reduce the error e_h. But this decision, to increase rather than decrease u_h, shows that the brain knows an increase in u_h will drive e_h downward, toward 0. In mathematical terms, the brain knows that the derivative $\partial e_h/\partial u_h$ is negative. Derivatives like this one, relating error and commands, are called *sensitivity derivatives* in the control literature (Astrom and Wittenmark 1995).

Normally the error and command are not scalars but vectors, $\boldsymbol{e}$ and $\boldsymbol{u}$, consisting of many components; e.g. retinal slip has three dimensions (horizontal, vertical, and torsional) or six dimensions if we consider both eyes, and the motor commands to the two eyes are vectors of at least 12 components, corresponding to the 12 extraocular muscles. Therefore the derivative $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is not a single number but a matrix, called the *control jacobian* (Callier and Desoer 1991), and its elements are the many sensitivity derivatives of the system. A flexible adaptive system

has to know all these derivatives, at least roughly. And adding to the complexity, $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is not a constant matrix but varies with the state of the plant, the current u, and other context.

1.2 Biological constraints on algorithms

Biological constraints are criteria meant to screen out algorithms that work well when implemented in computers but aren't feasible for use in the brain. A major problem, though, is that no one is sure what is really feasible in the brain.

There are many approaches to this issue, but they can be usefully arranged along a spectrum. At one extreme, we can take a severely conservative approach, rejecting any computational step that hasn't yet been verified physiologically. At the other extreme we can ignore biological constraints entirely, on the grounds that they are so uncertain, and because the learning and control problems solved by the brain are so difficult, and so far beyond anything in our current technology, that it makes sense to give ourselves as much freedom as possible in the beginning and only later, once we have devised one or two candidate algorithms, start bringing in constraints.

In my work I have tried to chart a middle road. If I discover that a novel mechanism would be useful for some computational task facing the brain then I won't reject it on the grounds that it hasn't been observed, but I will reject it if I find positive evidence that it doesn't exist in the brain. By far the most important example of this kind is the constraint on information transmission in the brain.

1.2.1 Nontransmissible variables

Whereas computers represent all their data in a uniform way, as voltages stored in latches, the brain represents information in (at least) two different forms: in patterns of action potentials traveling along axons and in synaptic weights. Action potentials are short-lived, and they require a whole neuron for each signal, so they are metabolically expensive, but they have the advantage that the data can be transmitted rapidly along axons to remote sites. Synaptic storage is compact, durable, and metabolically inexpensive, but not transmissible (i.e. there is no synaptic *weight transport*), so they influence remote computations only indirectly, via their effects on the firing of their own postsynaptic cells (Grossberg 1987; Mazzoni, Andersen et al. 1991; Levine 2000; Rolls & Deco 2002). This lack of *weight transport* constrains the algorithms that can run in biological networks, as I will describe in more detail in Chapter 2.

1.3 Outline of the thesis

After laying out general background concepts in Chapter 2, I devote Chapter 3 to my first project, a theoretical analysis of the role played by sensitivity derivatives in motor learning, and of possible mechanisms by which these derivatives might be learned in the brain. In Chapter 4 I describe my second project, testing further predictions of my theory of sensitivity derivatives. And finally in Chapter 5 I sum up my main conclusions and consider possible extensions and open questions.

Chapter 2

General background

This chapter describes the background knowledge needed to explain my work. First I introduce adaptive control theory. Then in Section 2 I explain some elements of neural computation and how they relate to adaptive control. After that I discuss in more detail some possible biological constraints on neural communication. And finally, I will present adaptive control schemes from the sensorimotor literature and identify the gaps in current theory that prompted my own research.

2.1 Elements of control

The brain's job is control. It controls voluntary and reflex movements of the body by contracting and relaxing skeletal muscles. It also controls autonomic variables like heart rate, blood pressure, and hormone levels via smooth muscle and glands. In this thesis I will focus on the control of simple eye and limb motions, but many of the principles apply to all types of neural control.

The nervous system is in many ways a special kind of controller. For one thing, is it *adaptive*: based on information from the senses it alters its own processing to improve its future performance. It is also *distributed*: it holds information at different locations in a network, and

needs time and energy to move data from place to place. And it is "z*ippable*", in the sense that the instructions for building an entire brain have to fit in a single egg cell.

To begin with I will introduce the basic elements of any control system: the plant and controller.

2.1.1 The plant

Fundamental to control theory is the concept of the *controlled process* or *plant*: the system we wish to steer to some goal. In driving, for example, the plant is the car. The word comes from "plant" in the sense of machinery or a mechanized installation, as in "power plant". Some sensorimotor examples: in reaching, the plant is an arm; in looking around, it might be the eyeballs and the head. In many tasks the plant includes objects outside the body, e.g. when you make a bed, the plant includes many body parts and also the bed, blankets, pillows and so on. So the concept is very general.

Usually the plant has some sort of intrinsic dynamics. For instance a reaching arm must obey the physical laws of motion. Normally these intrinsic dynamics are represented by a differential equation called the *plant equation*. A general form of plant equation is $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, t)$ where $\boldsymbol{x}$ is the state of the plant (e.g. if the plant is an arm then $\boldsymbol{x}$ might be a vector consisting of the positions and velocities of all the arm joints), the vector $\boldsymbol{u}$ is the *control signal* or *command* (e.g. the firing rates of motoneurons), and t is time. So the plant equation describes how the state evolves through time under the influence of the commands it is receiving.

For a simple example, consider the vestibulo-ocular reflex, or VOR, which counterrotates the eyes when the head turns, so as to keep the visual images stationary on the retinas. In this system

the state $\boldsymbol{x}$ is eye position — the angle of rotation of the eyeball in its socket — and the plant equation is

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{u} - k\boldsymbol{x}}{\rho} \tag{2.1}$$

where k and ρ are constants related to the mechanical stiffness and viscosity of the tissues in the eye socket.

2.1.2 The controller

The controller sends commands to the plant to try to achieve some desired outcome. For instance, an airplane autopilot is a controller: it sends commands to the throttles and ailerons to achieve a desired heading. In reaching, the controller is a part of the nervous system that sends commands to the arm, steering it to some desired position. Parts of the plant that transduce commands are called *actuators*, e.g. for the autopilot these include the motors that move the ailerons; for the arm controller they might be the arm muscles.

A general form of the controller equation is $\boldsymbol{u} = \boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{w}, t)$, where $\boldsymbol{x}$ is the plant state, $\boldsymbol{x}^*$ is the desired state (also called the goal or *reference*), $\boldsymbol{w}$ is a vector of internal controller parameters, and t is time.

If $\boldsymbol{u}$ is a function of time alone, with no information about the state of the plant $\boldsymbol{x}$ — $\boldsymbol{u} = \boldsymbol{\gamma}(t)$ — then the controller is said to be *open-loop*. Open-loop controllers are usually designed for some particular, narrow task, e.g. reaching between two specified arm positions. If $\boldsymbol{u}$ is a function of the state or the state and target — e.g. $\boldsymbol{u} = \boldsymbol{\gamma}(\boldsymbol{x})$ or $\boldsymbol{u} = \boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{x}^*)$ — then we have a *closed–loop* or

feedback controller. A feedback controller is a general rule which says what command is appropriate for any given state $\boldsymbol{x}$ and $\boldsymbol{x}^*$, e.g. to produce efficient hand movements between *any* two positions. Feedback controllers are more versatile than open-loop ones, and they don't fail when the plant is perturbed away from its planned course, so feedback control is what most brain systems need.

An *adaptive* controller has the form $\boldsymbol{u} = \boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{w})$ or $\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{w}, t)$ where $\boldsymbol{w}$ is a vector (or often a matrix) of adjustable parameters. This $\boldsymbol{w}$ evolves, and its rate of change is a function of the plant state and goal. In a good adaptive controller, the differential equation relating $d\boldsymbol{w}/dt$ with $\boldsymbol{x}$ and $\boldsymbol{x}^*$ causes $\boldsymbol{w}$ to evolve in a way that improves the controller. In the brain, $\boldsymbol{w}$ might be synapses or inter-neurons within a controller. This point of view will be made clearer when I discuss neural computation in later sections. In this thesis I will be concerned mainly with adaptive feedback controllers.

2.1.3 An example of sensorimotor control: the vestibulo-ocular reflex

As I have mentioned above, the vestibulo-ocular reflex, or VOR, counterrotates the eyes when the head turns. So the VOR acts like a Steadicam for the eyes, stabilizing the retinal images when the head moves. For example when the head turns right, the VOR rotates the eyes left in the head to keep them on target; and vice versa when the head turns left. The VOR's plant is the eyeballs, eye muscles, and other tissues within the eye socket, and its plant equation is (2.1).

The VOR is driven by sensors called semicircular canals in the inner ear, as shown in Figure 2.1. These canals sense rotary head velocity, i.e. afferent nerve fibers from any one canal fire at a rate proportional to the component of rotary velocity in the plane of that canal. The canals are called

vestibular organs because they lie partly within a chamber called the vestibulum in the inner ear. And the vestibulo-ocular reflex is so called because it uses information from vestibular sensors to control the eyes. The VOR's controller involves several brain structures, but its core is the set of brainstem neurons shown in Figure 2.1. As illustrated there, signals from the canals travel via the primary vestibular neurons in the vestibular ganglion to the secondary vestibular neurons in the vestibular nucleus and then to motoneurons in cranial-nerve nuclei III, IV and VI and finally to the eye muscles.

Figure 2.2 shows the essentials of the VOR control system in the form of a flow diagram. The desired outcome for the VOR controller is to make eye velocity = –head velocity, e.g. if the head turns at 50°/s rightward then the eyes should turn at 50°/s leftward in their orbits. Writing $\dot{x}$ for eye velocity and $\dot{h}$ for head velocity, the aim is to have $\dot{x} = -\dot{h}$, so we can define an *error* e for the system in this way:

$$e = \dot{x} + \dot{h} \tag{2.2}$$

The definition is sensible because this quantity will equal zero if and only if the reflex achieves its aim of making $\dot{x} = -\dot{h}$.

What is the controller that will achieve that aim? We can derive it by solving the equation $(u - kx)/\rho = -\dot{h}$, where I have replaced the $\dot{x}$ in Equation (2.1) with its desired value, $-\dot{h}$. Rearranging terms, we get

$$u = kx - \rho\dot{h} \tag{2.3}$$

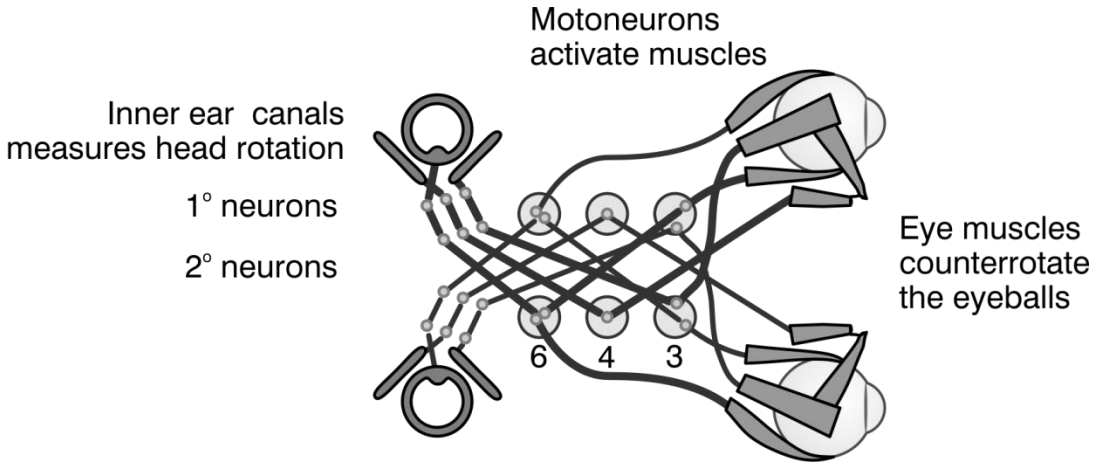

Figure 2.1: Neural control system for the vestibulo-ocular reflex, or VOR. The VOR is driven by sensors called semicircular canals in the inner ear. These canals sense rotary head velocity: afferent nerve fibers from any one canal fire at a rate proportional to the component of rotary velocity in the plane of that canal. The canals are called vestibular organs because they lie partly within a chamber called the vestibulum in the inner ear. The VOR's controller involves several brain structures, but its core is these brainstem neurons: From the canals, signals travel via primary vestibular neurons in the vestibular ganglion, to secondary vestibular neurons in the vestibular nucleus, to motoneurons in cranial-nerve nuclei 3, 4 and 6, to the eye muscles.

As this equation shows, the VOR controller needs information about eye position $\boldsymbol{x}$ (as well as of course $\dot{\boldsymbol{h}}$). But sensory feedback about eye position is slow, whether from vision or proprioception, so the brain instead uses an internal estimate of $\boldsymbol{x}$ from a plant model. I will return to this concept of internal feedback in Chapter 3.

Equation (2.3) shows also that the VOR controller needs information about the mechanical parameters of the eye plant, k and ρ. This point is general: any effective controller has to take into account its plant's dynamics, including mechanical properties like elasticity, viscosity and inertia. But what if the plant properties change? In that case an adaptive controller can revise its estimates of those properties to maintain functionality.

2.1.4 Examples of adaptive control in the brain

Many neural controllers are adaptive, adjusting their own processing to improve their performance. For example, most sensorimotor skills, from piano playing to tennis, improve with practice. And basic skills like reaching for a visual target are not inborn but develop over the first few months of life.

A relatively simple example of neural adaptive control is the vestibulo-ocular reflex, which adjusts itself to restore retinal-image stability when vision is altered by lenses. Magnifying lenses increase retinal-image motion; e.g. 100^{o}/s of head rotation might now cause 200^{o}/s of image slip. So with these lenses, a normal-strength VOR is too weak to stabilize the retinal image. But if you wear the lenses for 30 minutes then your VOR will grow stronger, improving your retinal-image stability (Gauthier & Robinson 1975). Similarly, if you wear minifying lenses or see a head-fixed

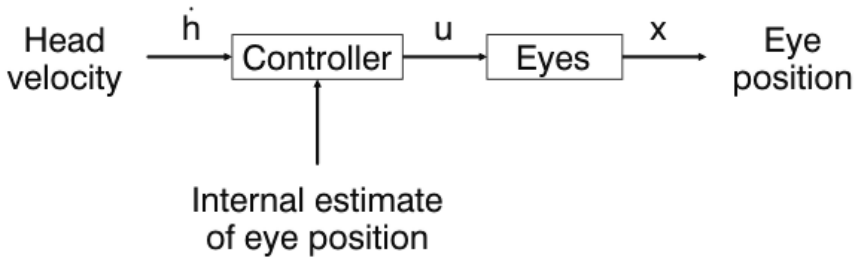

Figure 2.2: A flow diagram of the VOR.

visual field then your VOR will grow weaker, again improving stability (Miles & Eighmy 1980). Altered vision can also change the direction of the VOR. If you wear right-left reversing goggles for many hours, your VOR will weaken and then reverse direction, generating rightward eye motion in response to rightward head rotation (Gonshor & Jones 1976). Or if you view a computer-generated scene which moves vertically when your head turns horizontally, your VOR will learn to rotate your eyes vertically during horizontal head rotation (Schultheis & Robinson 1981; Harrison et al. 1986).

Another example of an adaptive controller is the system that generates *saccades*, the rapid eye movements that shift the gaze point from one object to another. If one of your eye muscles is damaged, your saccades will miss their targets and then drift; but if the damage isn't too severe, neural adaptation will restore accuracy and eliminate drift, even if the muscle remains abnormal (Kommerell et al. 1976; Abel et al. 1978; Optican & Robinson 1980; Optican et al. 1985).

Arm control also is adaptive. If you throw a ball while wearing prisms that shift your visual world 10^{o} right, you will miss your target by about 10^{o} right, but soon you will regain your accuracy (Martin et al. 1996, 2002). Then when you remove the prisms, for a while you will miss on the left side until your arm controller recorrects itself (Martin et al. 1996, 2002).

In all these cases, how do the control networks adjust themselves to improve performance? The best clues come from the engineering field of adaptive control, which is concerned with the study and design of controllers that are able to modify their own parameters, and thereby their control law, to cope with changes in an unknown or uncertain environment. For example, a flight controller in an airplane must adapt to the changing mass of the aircraft, which decreases as fuel is consumed during the journey. Adaptive control provides a general framework with which we

can study brain function and in particular learning in a systematic and unified way. Next I will discuss some basic concepts of this theory.

2.1.5 Error and context

The aim of an adaptive controller is to adjust its parameters $\boldsymbol{w}$ so as to zero some sort of error $\boldsymbol{e}$, which is usually a function of $\boldsymbol{x}$, and possibly also of other variables. For instance we have seen that in the VOR, the aim of the reflex is to make eye velocity $\dot{\boldsymbol{x}}$ equal and opposite head velocity $\dot{\boldsymbol{h}}$, so the error is

$$\boldsymbol{e} = \dot{\boldsymbol{x}} + \dot{\boldsymbol{h}} = \frac{\boldsymbol{u} - k\boldsymbol{x}}{\rho} + \dot{\boldsymbol{h}} \tag{2.4}$$

where the second equality follows from the VOR plant equation (2.1).

It is convenient to define a vector $\boldsymbol{v}$ which contains all those variables, apart from $\boldsymbol{u}$, that affect the error. So for example in the VOR, the error equation (2.4) shows that $\boldsymbol{e}$ depends on $\boldsymbol{u}$, $\boldsymbol{x}$, and $\dot{\boldsymbol{h}}$, so in this case $\boldsymbol{v} = (\boldsymbol{x}, \dot{\boldsymbol{h}})$. I will call $\boldsymbol{v}$ the *context* vector.

This same $\boldsymbol{v}$ is also an appropriate input to the controller. It contains everything, besides the command $\boldsymbol{u}$, that influences the error $\boldsymbol{e}$, so it is just what the controller needs to know to compute a command $\boldsymbol{u}$ that will zero that error. From now on, in my flow diagrams and equations I will use $\boldsymbol{v}$ to represent all the controller's inputs, apart from the vector of adjustable parameters $\boldsymbol{w}$ and the error signal $\boldsymbol{e}$ that is used to adjust $\boldsymbol{w}$.

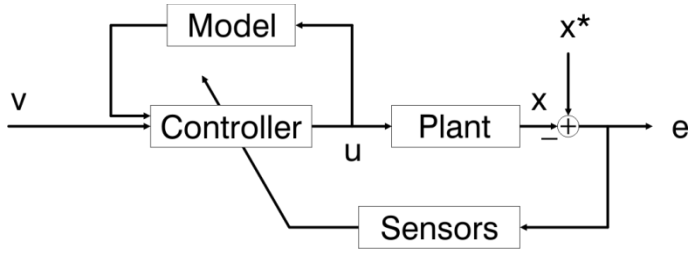

Figure 2.3: Error-driven learning. The plant is the system to be controlled, e.g. the eyeball or arm. The model is an internal representation of the plant established through learning. The controller is a network of neurons. The diagonal arrow means that the error signal e adjusts the controller's synapses to improve its commands so that in future e is smaller.

2.1.6 Loss and gradient descent

How does an adaptive controller adjust its parameters? There are many possible algorithms, but all viable approaches share certain properties: sensors of some sort measure the error $\boldsymbol{e}$ and convey it to the controller; based on this feedback, the controller adjusts its own synapses to improve its commands so that future $\boldsymbol{e}$ values are smaller.

A control system wants to have its error $\boldsymbol{e}$ near $\mathbf{0}$. Equivalently, it wants to minimize its *loss*, L, which we define as $L = \boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}/2$, where $\boldsymbol{e}^{\mathsf{T}}$ is the transpose of the vector $\boldsymbol{e}$. Most or all biologically plausible schemes for motor learning are based on the idea of computing, or approximating, the derivative, or *gradient*, of the loss with respect to the controller's adjustable parameters, $\partial L/\partial \boldsymbol{w}$, and then altering the $\boldsymbol{w}$ vector in the direction opposite the gradient:

$$\dot{\boldsymbol{w}} = -\eta \frac{\partial L}{\partial \boldsymbol{w}} \tag{2.5}$$

where η is a rate parameter, which may be constant or may vary for more efficient learning. This general approach is called *gradient descent*. Its rationale is that the gradient is the direction of steepest upward slope of the graph relating L and $\boldsymbol{w}$, so by moving $\boldsymbol{w}$ opposite that gradient, as in Equation (2.5), we are moving down the slope, reducing the loss. In Section 2.3 I will discuss how a neural controller might compute this gradient, but first I need to establish some basic facts about information processing in neurons.

2.2 Neural computation and communication

2.2.1 Neurons

I am concerned with how controllers and plant models are learned in networks of nerve cells. Real neurons are complex and still not well understood in all of their details. To make our analysis tractable, we have to model these cells as simply as possible while retaining their essential properties.

What are those essentials? Certain items clearly belong on the list. Neurons communicate through connections called synapses. Each neuron sends voltage pulses, called spikes or action potentials, down a branching filament, called its axon, which synapses with many other neurons. Whenever a voltage pulse reaches a synapse it releases molecules that increase or decrease the rate of voltage pulses generated by the downstream cell. Synapses come in many grades of strength: the stronger the synapse, the greater the effect of each incoming pulse on the pulse rate of the downstream cell. A single neuron can receive signals from other cells through as many as 100,000 synapses.

The neuron model I use incorporates these features in the simplest possible way. In this model the cell's *output signal y* (i.e. its rate of action potentials, or in other words its *spike rate* or *firing rate*) is a weighted sum of its inputs put through a function φ which may be linear or nonlinear (Figure 2.4). That is, the neuron's output is

$$y = \varphi\left(\sum_i w_i z_i\right) \tag{2.6}$$

where z_i are the inputs to the neuron and w_i are the strengths, or *weights*, of its synapses. The most common functions φ used in computational neuroscience are the identity function and the *hyperbolic tangent*, or *tanh*. The tanh function mirrors the property of real neurons that their outputs are confined to a finite range, while the identity, being linear, reflects that fact that many neurons have a linear range and may operate in that range under physiological conditions. But whatever the specific function, this class of simple model neuron incorporates the properties of real nerve cells that are likely essentially to understanding adaptive control: a large number of inputs, a single output, and synapses of unequal weight.

2.2.2 Universal approximation

It has been proven that networks of hyperbolic-tangent neurons of this type are universal approximators (Hornik et al. 1989), which means essentially that, given enough neurons, they can compute any function that might arise in control theory. More precisely, for any piecewise-continuous function over a closed and bounded domain, and any desired degree of accuracy, there is a network which approximates the function at least that well. Hyperbolic-tangent cells aren't the only kind that allows universal approximation — many other nonlinearities would work as well — but I will mainly use the tanh function because it is somewhat biologically realistic and it is easy to compute.

Because networks of hyperbolic-tangent neurons are universal approximators, they can act as flexible plant models and controllers. All we need is some algorithm to adjust those networks' weight vectors $\boldsymbol{w}$ down the gradient $\partial L/\partial \boldsymbol{w}$. But the algorithm must be consistent with the kinds of information flow that are available in the brain.

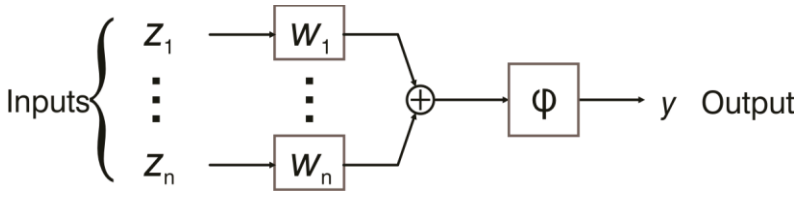

Figure 2.4: Simple model of a nerve cell. This diagram shows one neuron receiving a large number, *n*, of input signals z_i (firing rates of upstream neurons). The neuron multiplies each of its inputs by a synaptic weight w_i and sends the sum of these products through a function φ to yield an output, *y*.

2.2.3 Communication

The control literature describes many different schemes for adaptive systems, but these algorithms were designed to be implemented in digital computers, not in brains. It is not surprising, then, that in their present forms they aren't suited for biological neural networks, because there are important differences between brains and computers. Most notably, any variable represented in a computer can be transported to the central processing unit to participate in any computation. In a computer it is never the case, for instance, that we would like to add two variables *a* and *b* which are both present in memory but we can't manage it because *a* and *b* can't be brought together.

But in the brain, information is represented in quite different physical forms. Some data are coded in patterns of action potentials. Variables coded in this way are *transmissible* in the sense that they can be conveyed along axons to interact with other variables anywhere in the brain. But other data are stored in the strengths, or weights, of synapses. This form of storage is believed to underlie long-term memory. And most current theories explain learning as an adjustment of synaptic weights. In fact this form of information storage probably has a far higher capacity than do patterns of action potentials, because any one cell can generate only one action potential at a time, but a single cell can have up to about 100,000 synapses. Synapses are small, enduring, and numerous, outnumbering neurons by a factor of more than 5000 (Shepherd & Koch 1990), so they provide an enormous, high-density storage medium.

Synapses are also a metabolically cheaper form of long-term data storage than are action potentials. Though synapses consume a great deal of energy it is nevertheless the case that if we had a piece of information to store for, say, an hour, it would be cheaper to store it as a synaptic weight than in the rate or pattern of firing of action potentials, because we would need just one

synapse to store that piece of data as a weight, whereas we would need an entire neuron, generating a steady stream of spikes, to store the same data as action potentials. So the advantages of synaptic storage probably include both compactness and energy efficiency.

But synaptic weights are not transmissible. They are complex properties of pre- and postsynaptic structures, and they don't travel along axons or any other path, so they can't affect remote computations except indirectly, via their effects on the firing of their own postsynaptic cells (Grossberg 1987; Mazzoni et al. 1991; Levine 2000; Rolls & Deco 2002). For this reason we say that the brain has no *weight transport*.

The biological implausibility of weight transport has been pointed out many times in the neuroscience literature, usually in discussions of the learning algorithm known as error-backpropagation, or backprop (Zipser & Rumelhart 1988; Hinton 1989; Bengio et al. 1991; Crick 1989). When this algorithm was published in the 1980s (Rumelhart & McClelland 1986; Rumelhart et al. 1987), neuroscientists soon realized that it wasn't suited for the brain because it requires some mechanism by which one neuron could tell other neurons the strengths, or weights, of its synapses (Grossberg 1987), i.e. backprop requires weight transport (Stork 1989; Kolen 1994; Van Ooyen 2003).

It is important to ask whether weight transport is really so implausible. After all, there is at least one way that synaptic weights might, in principle, be transmitted from place to place — by riding on action potentials. That is, if a neuron N has an input synapse with weight w, then N's firing carries information about w, which could be sent to other cells. The problem is that this information is mixed up with many other variables, such as the weights of all the other synapses onto N, and the values of all of N's inputs. It would be possible in principle to observe N's inputs and firing patterns and deduce individual weights from them (Levine 2000), but the deduction

would be computationally demanding and would therefore call for a large number of cells. So this approach would lose the central advantage of storing information as weights in the first place: the fact that synapses are smaller and metabolically cheaper than whole neurons. For these reasons, the term "weight transport" is usually taken to mean some *other* mechanism, besides action potentials on the postsynaptic neuron, for transmitting synaptic weights to other cells. In this sense it is widely agreed that there is no weight transport in the brain (Grossberg 1987; Stork 1989; Kolen & Pollack1994; Van-Ooyen & Roelfsema 2003). In the next section I will consider in greater detail the biological mechanisms that might be considered a basis for weight transport and I will explain why they don't work.

2.2.4 Modes of communication other than action potentials

Neurons contain several mechanisms for *intracellular transport*, i.e. for moving materials, and therefore potentially information, around the inside of the cell. All known mechanisms of this type use the cytoskeleton as a kind of road or track, and all depend on motile protein molecules such as kinesins (Hirokawa et al. 1989; Yildiz, Tomishige et al. 2004; Asbury 2005) and dyneins (Bloom 1992).

Transport from the cell body toward the axon terminals is called *anterograde.* There are different kinds of anterograde transport which work at different rates: fast, intermediate, and slow. Fast transport seems to move mainly organelles, large molecules such as proteins, and some smaller ones including amino acids and sugars (Halperin & Lavail 1975; Knull & Wells 1975; Neale & Barker 1975; Iqbal & Ochs 1978). It moves its cargo at rates of up to about 5 μm/s (Oztas 2003). Intermediate transport manages only 0.25 μm/s, and slow just 0.15 μm/s. But these slower modes

may be the higher-capacity systems, transporting larger amounts of protein than does fast transport (Kristensson 1977) .

These same cargoes also move the opposite way, backwards up axons and dendrites, by *retrograde* transport, but even more slowly — probably less than half as fast as anterograde (Frizell & Sjostrand 1974; Stoeckel et al. 1975; Brimijon & Helland 1976; Price & Griffin 1977; Brimijon & Wiermaa 1978) .

Might the brain use intracellular transport to carry chemical messages about synaptic weights through neurons, and then use transmitters to carry these messages to other cells, i.e. might these mechanisms provide weight transport?

2.2.5 Implications for neural adaptive control

All known mechanisms of intracellular transport, even so-called fast transport, are too slow to support motor learning, though this statement requires some explanation. At first glance it might seem that even very slow transport might suffice if the learning itself were very slow. Here I will explain why anything but very rapid weight transport wouldn't suffice for sensorimotor learning, even if that learning were also slow.

The main problem is that in any feasible learning scheme, the information carried by weight transport would have to be brought together rather precisely with certain other signals, and the slower the transport, the harder it is to achieve this precision. For concreteness, consider the equation below. It describes weight updates in a specific situation — a layered network learning by backprop (Rumelhart et al. 1987) — but it illustrates a general point.

$$\dot{w}[l,j,i] = -\eta \sum_k \left\{ \frac{\partial L}{\partial y[l+1,k,t]} \; 1 - y^2[l+1,k,t] \; w[l+1,k,j] \right\} \; 1 - y^2[l,j,t] \; y[l-1,i,t] \quad (2.7)$$

Here $w[l, j, i]$ is the synaptic weight from cell i in layer $l - 1$ to cell j in layer l. The y's are neural signals, and L is the network's loss function. The equation says that the signal $y[l - 1, i, t]$ — the input to synapse $w[l, j, i]$ at time step t — is multiplied by downstream y's and $\partial L/\partial y$'s for the *same* t. In continuous time, this means $y[l - 1, i, t]$ is multiplied by all the downstream signals which it affects, all the way to the network's output layer. This is easy to arrange in a computer-simulated network, but in the brain it takes time for $y[l - 1, i, t]$ to affect the output layer. And it takes more time to transport information from the output layer back to the synapse $w[l, j, i]$. So the synapse has to remember its input signal $y[l - 1, i, t]$ during this whole round trip.

How long do real synapses remember information about individual input spikes? Experiments on spike-timing-dependent plasticity suggest this memory may persist as long as 250 ms (because synaptic change varies with the time interval between the spikes on the pre- and postsynaptic cells when that interval is shorter than 250 ms (Chen & Thompson 1995; Song et al. 2000; Yamamoto et al. 2002) ; of course LTP may last much longer, but it doesn't store information about individual spikes). Even this upper limit of 250 ms is too brief to permit learning by weight transport. Given the fastest known mechanisms of intracellular transport (~4 μm/s anterograde, ~2 μm/s retrograde), and assuming that plant model and controller are as close together as neighbouring cells in human cerebral grey matter (~35 μm), it would take ~30 s to transport $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ through the model and back to the controller's input synapses. That is, the known storage time of ~250 ms is ~120 times shorter than the shortest delay achievable by known mechanisms of intracellular transport.

Of course it may turn out that synapses are capable of longer storage, and so may be able to cope with long transport delays. But the longer the delay, the more information has to be stored in the synapse. If the round-trip transport time is T seconds then each adjustable synapse has to store its inputs, $y_{(l-1)it}$, that long. If its input signal changes significantly C times per second, then the synapse has to store C inputs for each second; i.e. it has to store CT inputs in all. So even if T was just 30 and C was just 10, each synapse would have to store 300 past inputs at all times.

And it is not enough that each synapse store all its inputs from the last T seconds. It would also have to multiply each input by the specific downstream y's and $\partial L/\partial y$'s to which it gave rise. Therefore the mechanisms that transported the y's and $\partial L/\partial y$'s up the axons would have to be temporally very consistent, or they would have to attach some sort of time marker to each y and $\partial L/\partial y$. In other words, as the transport delay increases beyond ~100 ms, the complexity and implausibility of the mechanism increase steadily.

The weights might be transported not intracellularly but via other, "weight-transmitter" cells, e.g. by action potentials in other neurons or by glia. These cells would "read off" synaptic weights and transmit the information by some rapid means. But there is no evidence that any such cells exist, and if they did, they would likely need huge numbers of read-off mechanisms: one for each synapse whose weight has to be transported. And it is unclear how a cell could "read off" synaptic weights; e.g. how it might measure the strength of a synapse between two other cells.

Finally, one could try using a mix of intracellular transport and action potentials. Referring again to the backprop equation above, suppose we use cytoplasmic transport to carry only the information about weights — the $w[l+1, k, j]$ — and let the other information, such as $y[(l+1), k, t]$ be carried by action potentials, which travel quickly. This way, only the weight information

is slow, and that may be acceptable if weights don't change very quickly; i.e. synapse $w[l, j, i]$ will know the value that $w[(l + 1), k, j]$ had a few minutes ago rather than its value now, but that doesn't matter if $w[(l + 1), k, j]$ has hardly changed in the interim. Assuming that the weights really do change slowly enough, does this idea make learning by weight transport biologically feasible? Not likely, because any one neuron projects to many downstream cells. Looking once more at the backprop equation, we see that to compute the adjustment to synapse $w[l, j, i]$ on cell j in layer l, we need to summate, over all k (i.e. over all neurons to which cell j projects) the product of $w[(l + 1), k, j]$ with a term involving $y^2[(l + 1), k, t]$. If the $w[(l + 1), k, j]$ and the $y[(l + 1), k, t]$ travel by separate routes (y on action potentials and w via the cytoplasm) then how can cell j pair them up again? How can it know which w goes with which y?

To make weight transport plausible, then, we would probably need to discover a new, fast form of intracellular transport. And we would have to find transportable molecules that represented the strengths of all downstream synapses, multiplied by their appropriate downstream factors. But at present there is just one known mechanism for moving information rapidly over more than a few μm, and that is action potentials.

In this thesis, therefore, I have avoided weight transport and insisted that all variables that have to be transported rapidly must be coded in neural firing. In Chapter 3 I will devise an algorithm called implicit supervision, which works without weight transport by a novel arrangement of cells whose basic mechanisms are well established and uncontroversial.

2.2.6 Linear-in-the-parameters learning

We need learning mechanisms that work without weight transport. There is a class of algorithms called reinforcement learning that doesn't need weight transport, but these algorithms are considerably slower than supervised schemes like backprop, especially on complex problems (O'Reilly 1996).

A much more powerful alternative is known as *linear-in-the-parameters* learning (Farrell & Polycarpou 2006) (Figure 2.5). Here the key idea is that learning is restricted to a single layer of synapses — those onto the output neurons of the network — and all the upstream weights are frozen: they do not change with learning, and therefore of course there is no need to transport any data to them regarding the weights of downstream synapses. In most implementations, the output neurons are made linear to simplify the learning rule; i.e. y is no longer a nonlinear function of the neuron's inputs, but is simply

$$y = \sum_i w_i \varphi_i(\boldsymbol{z}) \tag{2.8}$$

Here $\boldsymbol{z}$ is the vector of signals carried by the input-layer neurons, e.g. if there are 3 input neurons, as in Figure 2.5, then $\boldsymbol{z}$ is a vector of 3 components. The quantities $\varphi_i(\boldsymbol{z})$ are the signals carried by the neurons in the second-last layer of the network; the notation reflects the fact that these signals are functions — normally nonlinear functions — of the input-layer signals $\boldsymbol{z}_i$. These $\varphi_i(\boldsymbol{z})$ are called *features*. They are nonlinear functions of the network's inputs, but they are combined linearly to yield the network's output $\boldsymbol{y}$. The vector $\boldsymbol{\varphi}(\boldsymbol{z})$ composed of all the features $\varphi_i(\boldsymbol{z})$ is called the *feature vector*.

In this setting, with learning only in the output-layer weights, we need only compute $\partial L/\partial \boldsymbol{W}$ for the weights in that layer, not for all the weights in the network as is done in backprop (and it is convenient to represent these weights as a matrix $\boldsymbol{W}$ rather than a vector $\boldsymbol{w}$ — e.g. if the network has 6 features and 4 output neurons then the feature vector $\boldsymbol{\varphi}$ is 6-dimensional, the output vector $\boldsymbol{y}$ is 4-dimensional, and $\boldsymbol{W}$ is a 4-by-6 matrix — but the principles are the same as in my earlier discussion of the gradient $\partial L/\partial \boldsymbol{w}$). Loss is $L = \boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}/2 = (\boldsymbol{y} - \boldsymbol{y}^*)^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{y}^*)/2$, and therefore

$$\frac{\partial L}{\partial w_{ij}} = \frac{dL}{d\boldsymbol{e}}\frac{\partial \boldsymbol{e}}{\partial w_{ij}} = \boldsymbol{e}^{\mathsf{T}}\frac{\partial \boldsymbol{e}}{\partial w_{ij}} = \boldsymbol{e}^{\mathsf{T}}\frac{\partial \boldsymbol{y}}{\partial w_{ij}} \tag{2.9}$$

Now $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{\varphi}$, so $\partial \boldsymbol{y}_i/\partial w_{ij} = \varphi_j$, and $\partial L/\partial w_{ij} = e_i\varphi_j$. Hence the learning rule is simply

$$\dot{w}_{ij} = -\eta e_i \varphi_j \quad \text{or} \quad \dot{\boldsymbol{W}} = -\eta \boldsymbol{e}\boldsymbol{\varphi}^{\mathsf{T}} \tag{2.10}$$

where η is positive. With this rule there is no need for weight transport and no long transport delays because $\boldsymbol{e}$ can be conveyed quickly to the synapses by sensory axons — e.g. for the VOR, the error is retinal slip, which is conveyed by visual neurons — so linear-in-the parameters learning overcomes the main objection to backprop (Kawato 1990; Porrill et al. 2004).

Some brain regions do produce huge numbers of features turning a vector of inputs into a larger number of features — this is called *expansion recoding* or feature production. There is an example of this in the cerebellum, where 70 million mossy fibers project to cerebellar cortex via 70 billion granule cells. The reason may be to provide a 1000-fold increase in features for motor learning.

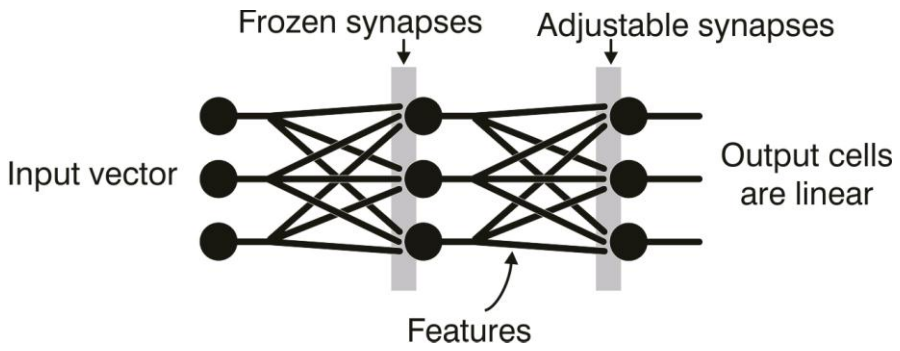

Figure 2.5: Linear-in-the-parameters learning. Learning takes place only in the synapses onto the output layer of cells. Weights of all the upstream synapses are frozen, meaning they can't change through learning, and therefore they don't need any information about the weights of downstream synapses. In other words this learning network doesn't need weight transport. This approach is called linear-in-the-parameters because the neurons in the output layer are linear and hence the output error is a linear function of the adjustable weights.

2.2.7 The curse of dimensionality

The key to LIP learning is having lots of features; and with any type of learning, a network with more cells and synapses is more flexible than one with fewer. So how many synapses, cells or features are enough to approximate a given set of functions? The number depends on the dimension of the functions' input space, or domain, because, roughly speaking, when the input dimension is higher there are more possible functions that have to be distinguished. Specifically, the number of adjustable parameters we need for good approximation increases exponentially with the dimension of the input domain — this is the "curse of dimensionality" (Bellman 1961).

The curse of dimensionality hits LIP learning harder than backprop. In backprop, the adjustable parameters are all of the synaptic weights, so we need a large number of *synapses*. But in LIP learning, for each scalar output of the network there is just one adjustable parameter per feature — i.e. per neuron in the pre-output layer — so we need large numbers of *neurons*: a harder condition because neurons are less plentiful than synapses.

A possible mechanism for coping with the curse of dimensionality is to use prior knowledge. If you can guess what sort of functions your network will be called upon to approximate then just give it features suitable for those functions. Natural selection may do that for us by building into our brains certain hard-wired features that are useful for life on Earth; i.e. natural selection may use information about useful features collected over the generations.

Another coping strategy is to settle for simple functions when the domain is high-dimensional. For example if the domain is 1-D we can specify any 5th-order function using just 6 parameters (p_0, p_1, p_2, p_3, p_4, p_5), i.e. we can write $f(z) = p_0 + p_1 z + p_2 z^2 + p_3 z^3 + p_4 z^4 + p_5 z^5$. If the domain is 2-D, we need 21 parameters for 5th-order functions, but just 6 for 2nd-order, i.e. $f(z_1, z_2) = p_0 +$

$p_1z_1 + p_2z_2 + p_3z_1^2 + p_4z_1z_2 + p_5z_2^2$ — that is, 6 parameters will suffice if we don't need any functions of order higher than 2. And so on: if the domain is 3-D we need 56 parameters to cover all the 5th-order functions but just 20 for the 3rd-orders, 10 for 2nd-order, or 4 for 1st-order. Maybe the brain gets by with simple high-dimensional functions.

Another indication that LIP may be biologically plausible, despite the curse of dimensionality, is that Nature is clearly not averse to attacking problems by sheer force of numbers. For example organisms make millions of seeds, spores, sperm cells, to get a few offspring. Similarly, our immune systems make millions of antibodies, trying by brute force to anticipate every possible antigen. And again there is the example of the 70 billion granule cells in cerebellar cortex, whose computational role is unknown but which may be performing expansion recoding.

Further, it helps that in LIP learning, features can be *shared* by many systems, so if you need a million features for some task, you can easily use those same features for a thousand other tasks as well.

And finally, there are speculative but not implausible ways that the brain could have more than one feature per neuron, and might be able to preserve useful features while replacing useless ones.

2.3 Sensitivity derivatives in adaptive control

I have said that most or all approaches to sensorimotor adaptive control are based on gradient descent: we estimate $\partial L/\partial \boldsymbol{w}$ and then adjust $\boldsymbol{w}$ by a learning rule $\dot{\boldsymbol{w}} = -\eta \partial L/\partial \boldsymbol{w}$. But how can a neural controller estimate the gradient $\partial L/\partial \boldsymbol{w}$? Clearly $\partial L/\partial \boldsymbol{w}$ depends on a long chain of

influence, because the controller weights $\boldsymbol{w}$ influence L only indirectly: we can assume that these weights $\boldsymbol{w}$ have a fairly direct influence on the command $\boldsymbol{u}$ issuing from the controller; this command then acts on the plant, resulting in some behavior and ultimately in an error signal $\boldsymbol{e}$ in a sensor. By the chain rule, $\partial L/\partial \boldsymbol{w}$ is the product of these 3 influences: $\boldsymbol{w}$ on $\boldsymbol{u}$, $\boldsymbol{u}$ on $\boldsymbol{e}$, and $\boldsymbol{e}$ on L:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{dL}{d\boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{w}} \tag{2.11}$$

Of course the first of these three factors is easy to compute: we have $L = (\frac{1}{2})\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}$, so $dL/d\boldsymbol{e} = \boldsymbol{e}^{\mathsf{T}}$. The formula simplifies to this

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{e}^{\mathsf{T}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{w}} \tag{2.12}$$

It is reasonable to assume that two of these three factors are readily available. The first is simply the error signal, which could be delivered to the controller by sensory axons. The third term, $\partial \boldsymbol{u}/\partial \boldsymbol{w}$, may be somewhat complicated, but it depends entirely on processes within the controller itself, so there is in principle no mystery as to how the controller might be able to obtain the information to compute $\partial \boldsymbol{u}/\partial \boldsymbol{w}$. But the middle term, $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, which is the matrix of sensitivity derivatives I discussed in Chapter 1, is harder. It depends on processes in the physical plant — muscles, tendons, and perhaps other objects outside the body — that determine how errors relate to motor commands. The brain receives information about the plant via sensory feedback, but how can it use that information to compute $\partial \boldsymbol{e}/\partial \boldsymbol{u}$? This is a central puzzle in sensorimotor control, and to my knowledge just one laboratory has proposed a solution.

2.3.1 Distal teacher theory

Jordan and Rumelhart proposed a mechanism by which a neural network called a plant model could learn to estimate $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ and send the information to a controller. I will discuss their distal teacher theory in detail below, in Section 3.3.3 and Figure 3.3, but for now I will foreshadow the key problem in the scheme.

This problem has to do with how information from the model gets back to the controller. In the distal teacher scheme, the information about $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is contained in the plant model, though not in its output — i.e. not in its neural firing rates — but in its synaptic weights. Or more precisely the information is distributed among the firing rates and synapses of the model network. But in any case transmitting this information from the model to the controller means transmitting data about synaptic weights, i.e. it means weight transport, which is biologically implausible, as I argued above.

Jordan and Rumelhart suggested that the information might travel by backprop. They showed that the controller and model can be serial parts of a single layered network, and that the backprop algorithm could then carry the necessary information from the downstream model to the upstream controller. Theirs was a slightly modified form of backprop, where two different error signals drove learning in separate sections of the network, and different parts of the network could learn at unequal rates or times, the idea being that the plant model should complete its learning first because otherwise it would send back inaccurate sensitivity derivatives and mislead the controller. But the scheme didn't avoid the crucial problem with the backprop algorithm, namely its dependence on weight transport.

This central implausibility is probably the reason that the distal teacher has not caught on, and that other theories of adaptive control have either ignored the problem of computing sensitivity derivatives, or have assumed that knowledge of these derivatives is innate, and so doesn't have to be learned at all.

2.3.2 Theories based on innate knowledge of sensitivity derivatives

Some of the most influential theories of motor learning are those developed over the years by Kawato and colleagues. These theories don't explain how sensitivity derivatives could be learned, nor do they emphasize the issue of sensitivity derivatives at all, but they do imply fairly explicitly that knowledge of those derivatives is inborn. For example, one of the Kawato group's key ideas has been the notion of parallel control paths: in any one motor system there are two controllers, one of them adaptive, the other innate and fixed, programmed by the genome (Figure 2.9). The innate controller sends its commands to the plant but also to the adaptive controller, which uses those commands to guide its own learning.

A closer look at this scheme shows that it assumes knowledge of sensitivity derivatives in the innate controller. To illustrate this point as clearly as possible, I will consider an extremely simple system in which the plant equation is $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \alpha \boldsymbol{u}$, where α is a scalar constant, and the error is given by $\boldsymbol{e} = \dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*$, where $\boldsymbol{x}^*$ is a reference signal — a moving target state, as in the VOR. For this system, it is clear that $\partial \boldsymbol{e} / \partial \boldsymbol{u} = \alpha$. Let us suppose that the adaptive controller is initially null — i.e. its command $\boldsymbol{u}_A$ is always zero — whereas the innate controller is given by $\boldsymbol{u}_I = -\beta \boldsymbol{e}$, where β is a positive scalar constant. Then we have $\dot{\boldsymbol{x}} = -\alpha\beta \boldsymbol{e} = \alpha\beta(\dot{\boldsymbol{x}}^* - \dot{\boldsymbol{x}})$. Rearranging, we find that $\dot{\boldsymbol{x}} = [\alpha\beta/(1+\alpha\beta)]\dot{\boldsymbol{x}}^*$. So this innate controller isn't very good: what we

want is $\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}^*$, but that would imply that $\alpha\beta = 1 + \alpha\beta$, which is impossible — it is roughly true when $\alpha\beta$ is very large, but in that case control would be unstable.

But a key virtue of the parallel control paths is that even a barely decent innate controller can train a good adaptive controller. To achieve this barely decent performance, the theory need only assume that the innate controller's β be chosen so that the product $\alpha\beta$ is positive, i.e. $\text{sign}(\beta) = \text{sign}(\alpha) = \text{sign}(\partial \boldsymbol{e}/\partial \boldsymbol{u})$, so in effect the theory assumes that the innate controller conveys information about the signs of sensitivity derivatives to the controller. If the derivatives ever changed, so that the innate estimate had the wrong sign, then the controller would fail permanently, as will any control system that cannot learn $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. I will expand on this theme in the next chapter.

In summary, Jordan and Rumelhart asked how the brain might learn $\partial \boldsymbol{e}/\partial u$, and suggested an answer that unfortunately relies on biologically implausible forms of communication. Other theorists have ignored the issue or assumed that at least a crude estimate of $\partial \boldsymbol{e}/\partial u$ is genetically hard-wired into the brain. In what follows I will show that these hard-wiring theories are incompatible with the adaptive flexibility of real sensorimotor controllers, and I will suggest a mechanism for learning sensitivity derivatives that uses only well-established forms of neural communication.

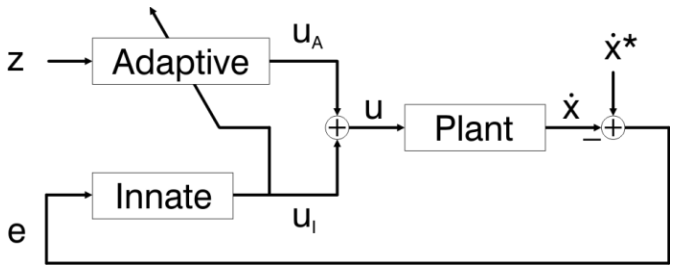

Figure 2.6: Kawato's parallel control paths. Two controllers work in parallel to steer the plant. One controller is programmed by the genome, active from birth, and incapable of learning. It sends a copy of its commands to the adaptive controller top guide the latter's learning.

Chapter 3

Sensitivity derivatives for flexible sensorimotor learning

Contents of this chapter have been published in Neural Computation: **Abdelghani et al.** Sensitivity derivatives for flexible sensorimotor learning. *Neural Computation*, 2008 Aug; 20(8): 2085-2111. A link to the published paper can be found at

http://www.mitpressjournals.org/toc/neco/20/8

As I have described above, variables called sensitivity derivatives quantify how a system's performance depends on the commands from its controller (Åström & Wittenmark, 1995). Knowledge of these derivatives is a prerequisite for adaptive control, including sensorimotor learning in the brain, but it is unclear how that knowledge could be acquired by neural controllers.

The pieces of the puzzle are these: any controller sends its commands $\boldsymbol{u}$ to its controlled object (or *plant*), with the aim of reducing some error $\boldsymbol{e}$. For example the vestibulo-ocular reflex, or VOR, measures head rotation and counterrotates the eyes to keep the retinal images stable, so its error $\boldsymbol{e}$ might be retinal-image slip. In reaching, $\boldsymbol{e}$ might be the vector from target to hand. And similarly for any learned behavior (see *Mathematical setting* and Appendix A). If the controller is adaptive, it can improve based on feedback about $\boldsymbol{e}$, but to do this it needs to know the relation between $\boldsymbol{e}$ and $\boldsymbol{u}$: it needs the matrix $\partial\boldsymbol{e}/\partial\boldsymbol{u}$, the *control jacobian* (Callier & Desoer, 1991) also

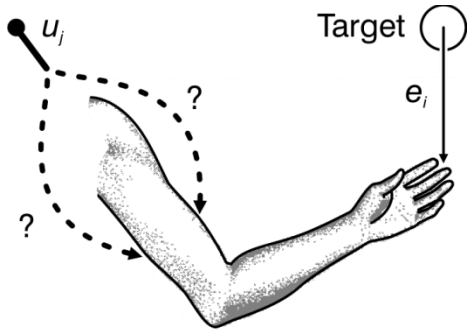

Figure 3.1: Error and sensitivity. Suppose an arm controller knows its vertical reaching error e_i (downward in this case) and the state of the arm. Still it can't tell how to improve the component u_j of its motor command, or even whether to increase u_j or decrease it, unless it knows whether u_j's effect is to raise or to lower the hand; i.e. it needs the relation between e_i and u_j, which is $\partial e_i/\partial u_j$, the sensitivity derivative. The same holds for any adaptive controller.

known as the matrix of *sensitivity derivatives* (Åström & Wittenmark, 1995). This matrix is the crucial information that tells an adaptive controller how to improve based on error feedback.

For example, suppose you design an adaptive controller for the arm, and suppose that when the vertical component e_i of reaching error is negative (downward), the controller adjusts its weights so as to increase some component u_j of its motor command (Figure 3.1). This decision, to increase rather than decrease u_j, means the controller believes an increase in u_j will drive e_i in the positive direction, toward 0, *i.e.* it believes that $\partial e_i/\partial u_j > 0$. In a similar way, any adaptive adjustment of any controller reflects an assumption about sensitivity derivatives.

How can a controller get information about these derivatives? The fundamental question is whether they are learned or known innately. Recent theories propose that the knowledge is innate (Porrill, Dean & Stone 2002 & 2004; Kawato & Gomi 1992), or they don't discuss where it comes from (*e.g.* Todorov & Jordan 2002; Yamamoto et al. 2002). But in this chapter I will show that theories where this knowledge is *solely* innate can't explain the versatility of real sensorimotor learning — its capacity to recover from major lesions and to cope with complex tasks and tools. So sensitivity derivatives — the prerequisites for sensorimotor learning — must themselves be learned. No theory has explained how they *could* be learned, given the known forms of information flow in the brain. I will consider possible mechanisms and argue that the best option is a process I call implicit supervision.

3.1 Mathematical setting

Consider a control system with plant equation

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \tag{3.1}$$

where $\boldsymbol{x}$ is the plant state, $\boldsymbol{u}$ is the command from the controller, and $\dot{\boldsymbol{x}}$ is the time derivative, or rate of change, of $\boldsymbol{x}$. We will suppose that the aim of the controller is to zero an error vector $\boldsymbol{e}$ which is a function of $\boldsymbol{u}$ and a *context vector* $\boldsymbol{v}$,

$$\boldsymbol{e} = \boldsymbol{g}(\boldsymbol{v}, \boldsymbol{u}). \tag{3.2}$$

More precisely, the aim is to minimize the *loss*, often defined as $L = \frac{1}{2}\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}$. To achieve this goal, the controller reads in context and generates commands according to a function called its *control law*,

$$\boldsymbol{u} = \boldsymbol{\gamma}(\boldsymbol{v}). \tag{3.3}$$

3.1.1 Example

In the horizontal vestibulo-ocular reflex, $\boldsymbol{x}$ is eye position relative to the head, $\boldsymbol{u}$ is the net motoneuron signal to the horizontal eye muscles, and the plant equation (in a simplified form) is

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{u} - \kappa\boldsymbol{x}}{\rho}. \tag{3.4}$$

where κ and ρ are constants. $\boldsymbol{e}$ is retinal-image slip velocity, which is the sum of eye and head velocity,

$$\boldsymbol{e} = \dot{\boldsymbol{x}} + \dot{\boldsymbol{h}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \dot{\boldsymbol{h}}. \tag{3.5}$$

And the context $\boldsymbol{v}$ is, by definition, everything besides $\boldsymbol{u}$ that affects $\boldsymbol{e}$, so in this case $\boldsymbol{v}$ is a vector consisting of eye position $\boldsymbol{x}$ and head velocity $\dot{\boldsymbol{h}}$. Then $\boldsymbol{e}$ is a function of $\boldsymbol{u}$ and $\boldsymbol{v}$, as required by Equation 3.2. The optimal control law is

$$\boldsymbol{u} = \kappa \boldsymbol{x} - \rho \dot{\boldsymbol{h}} = (\kappa, -\rho) \cdot \boldsymbol{v}, \tag{3.6}$$

because this $\boldsymbol{u}$, plugged into Equation 3.4, makes $\dot{\boldsymbol{x}} = -\dot{\boldsymbol{h}}$ and so zeroes the error $\boldsymbol{e}$ defined in Equation 3.5. As required by Equation 3.3, $\boldsymbol{u}$ is a function of $\boldsymbol{v}$. And biologically, there is no difficulty making $\boldsymbol{v}$ available to the VOR controller: $\dot{\boldsymbol{h}}$ signals can be delivered from the inner ear and eye-position feedback $\boldsymbol{x}$ from spindles or efference copy.

This example illustrates that the horizontal VOR fits the framework defined in Equations 3.1–3.3. That framework is quite general, encompassing a wide range of sensorimotor control systems, though to fit the scheme they must be expressed in a form in which a quantity called relative degree is zero. This issue is discussed in Appendix A.

3.1.2 Sensitivity derivatives

In the horizontal VOR, with the plant described by Equation 3.4 and the error by Equation 3.5, the sensitivity derivative is a single number,

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} = \frac{\partial}{\partial \boldsymbol{u}} \left(\dot{\boldsymbol{x}} + \dot{\boldsymbol{h}} \right) = \frac{\partial}{\partial \boldsymbol{u}} \left(\frac{\boldsymbol{u} - \kappa \boldsymbol{x}}{\rho} + \dot{\boldsymbol{h}} \right) = \frac{1}{\rho}. \tag{3.7}$$

Most sensorimotor systems are more complex, and for them the sensitivity derivative is not a scalar constant but a matrix of functions. For some control tasks involving a planar 2-link arm, for instance, the matrix is

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} = \left[\delta(\alpha + \beta \cos x_2) - \left(\delta + \frac{\beta \cos x_2}{2} \right)^2 \right]^{-1} \begin{bmatrix} \delta & -\delta - \dfrac{\beta \cos x_2}{2} \\ -\delta - \dfrac{\beta \cos x_2}{2} & \alpha + \beta \cos x_2 \end{bmatrix} \tag{3.8}$$

where α, β, and δ are constants and x_2 is elbow angle. For a real arm, with 7 degrees of freedom, the matrix is more complex. And for tasks involving many body parts, or tools or other props, or interactions with other agents, it will be more complex again.

3.2 Sensorimotor learning

To learn is to adjust your control law $\boldsymbol{u} = \boldsymbol{\gamma}(\boldsymbol{v})$ so as to reduce $\boldsymbol{e}$. Here I show, with a simple example, how the proper adjustments depend on the sensitivity derivatives. The principle holds for any sensorimotor task fitting Equations 3.1–3.3 and for any learning algorithm, but for illustration purposes I will choose the VOR and the Widrow-Hoff learning rule, also known as online-gradient or least-mean-square (LMS) learning.

Suppose the VOR controller has the form of the ideal controller (Equation 3.6),

$$\boldsymbol{u} = \langle \kappa \rangle \boldsymbol{x} - \langle \rho \rangle \dot{\boldsymbol{h}} \tag{3.9}$$

where the small corner brackets ‹› indicate neural estimates, so ‹κ› and ‹ρ› are parameters which are shaped, by learning, to equal the κ and ρ in the plant equation, Equation 3.4. In this setting

the LMS learning rule is described by the following equations, where η is the learning-rate constant:

$$\frac{d\langle\kappa\rangle}{dt} = -\eta\frac{\partial L}{\partial\langle\kappa\rangle} = -\eta\frac{dL}{d\boldsymbol{u}}\frac{\partial\boldsymbol{u}}{\partial\langle\kappa\rangle} = -\eta\frac{dL}{d\boldsymbol{e}}\frac{\partial\boldsymbol{e}}{\partial\boldsymbol{u}}\frac{\partial\boldsymbol{u}}{\partial\langle\kappa\rangle} = -\eta\boldsymbol{e}^{\mathrm{T}}\frac{\partial\boldsymbol{e}}{\partial\boldsymbol{u}}\boldsymbol{x}, \tag{3.10}$$

$$\frac{d\langle\rho\rangle}{dt} = -\eta\frac{\partial L}{\partial\langle\rho\rangle} = -\eta\frac{dL}{d\boldsymbol{u}}\frac{\partial\boldsymbol{u}}{\partial\langle\rho\rangle} = -\eta\frac{dL}{d\boldsymbol{e}}\frac{\partial\boldsymbol{e}}{\partial\boldsymbol{u}}\frac{\partial\boldsymbol{u}}{\partial\langle\rho\rangle} = \eta\boldsymbol{e}^{\mathrm{T}}\frac{\partial\boldsymbol{e}}{\partial\boldsymbol{u}}\dot{\boldsymbol{h}}. \tag{3.11}$$

Clearly this learning rule requires knowledge of the sensitivity derivative $\partial\boldsymbol{e}/\partial\boldsymbol{u}$. (In the VOR, as we have seen, $\partial\boldsymbol{e}/\partial\boldsymbol{u} = 1/\rho$, so an estimate of $\partial\boldsymbol{e}/\partial\boldsymbol{u}$ is at the same time an estimate of $1/\rho$, but notice that the estimate $\langle\partial\boldsymbol{e}/\partial\boldsymbol{u}\rangle$ is physically a different thing than the controller parameter $\langle\rho\rangle$, so $\langle\partial\boldsymbol{e}/\partial\boldsymbol{u}\rangle$ is not necessarily equal to $1/\langle\rho\rangle$.)

Now suppose the controller parameters $\langle\kappa\rangle$ and $\langle\rho\rangle$ are incorrect, *i.e.* they don't equal κ and ρ, as in the early part of the time plot in Figure 3.2A. Then eye velocity $\dot{\boldsymbol{x}}$ is incorrect and $\boldsymbol{e}$ is nonzero. But the learning rule in Equations 3.10 and 3.11 repairs the problem: as long as the estimate $\langle\partial\boldsymbol{e}/\partial\boldsymbol{u}\rangle$ is reasonably accurate and the model inputs are sufficiently varied, $\langle\kappa\rangle$ and $\langle\rho\rangle$ converge to their correct values and $\dot{\boldsymbol{x}}$ comes to match $-\dot{\boldsymbol{h}}$.

What is a "reasonably accurate" estimate of the sensitivity derivative? Usually if $\langle\partial\boldsymbol{e}/\partial\boldsymbol{u}\rangle$ is, say, 10 times larger or smaller than the true $\partial\boldsymbol{e}/\partial\boldsymbol{u}$, then learning will still proceed, though 10 times faster or slower than usual — as if the learning-rate constant η were scaled up or down. But large misestimates of this type can cause problems — instability or uselessly slow learning.

But errors in the *sign* of $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u} \rangle$ will be disastrous. If the sign of $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u} \rangle$ is opposite the sign of the true $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u} \rangle$ then the learning rule will drive $\langle \kappa \rangle$ and $\langle \rho \rangle$ in the wrong directions, making the error worse and worse, as in Figure 3.2B. Equations 3.10 and 3.11 reveal the central problem: a reversal in $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ causes a reversal in $\partial L/\partial \boldsymbol{u}$, which means the controller's estimates of $\partial L/\partial \langle \kappa \rangle$ and $\partial L/\partial \langle \rho \rangle$ have the wrong signs, so $\langle \kappa \rangle$ and $\langle \rho \rangle$ are driven up rather than down the gradient of the loss function.

The horizontal VOR is one-dimensional, but in multidimensional problems the principle is the same except that the signs of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ and $\partial L/\partial \boldsymbol{u}$ needn't correspond one-to-one. If $\boldsymbol{e}$ is a vector of more than one component, then $\partial L/\partial \boldsymbol{u}$ is the matrix product $\boldsymbol{e}^{\mathsf{T}}(\partial \boldsymbol{e}/\partial \boldsymbol{u})$, so it is possible that several components of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ may reverse without reversing $\partial L/\partial \boldsymbol{u}$, and conversely $\partial L/\partial \boldsymbol{u}$ may reverse without any component of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ doing so. In any adaptive-control task of any dimension, the crucial thing is that no element of $\langle \partial L/\partial \boldsymbol{u} \rangle$ should have the wrong sign.

This principle is general: a sign error in any component of $\langle \partial L/\partial \boldsymbol{u} \rangle$ will reverse learning in any adaptive controller. And with fast learning algorithms, even scaling errors may seriously slow or disrupt learning. So adaptive controllers in the brain need to know their sensitivity derivatives. How do they do it?

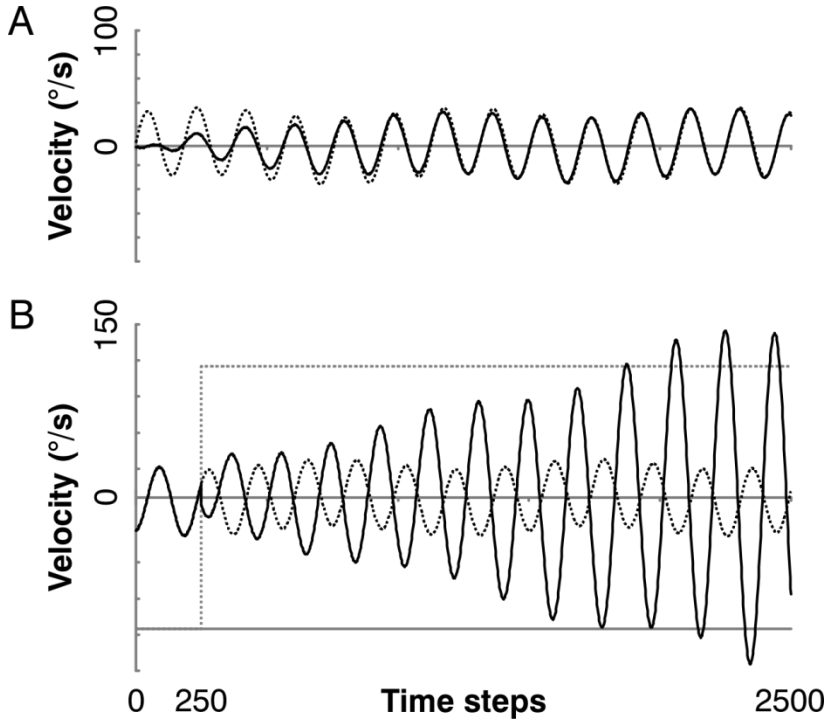

Figure 3.2: When sensitivity derivatives aren't learned, control is inflexible. (A) In the VOR, eye velocity (solid black line) should equal –1 × head velocity (dotted black). Here a controller with correct innate knowledge of sensitivity learns the task. (B) Here the trained controller is working well until, at time step 250, sensitivity (dotted gray line) switches sign (this could be achieved by transposing eye muscles). The switch makes the controller's innate estimate ‹$\partial \mathbf{e}/\partial \mathbf{u}$› (solid gray line) incorrect, so ‹$\partial L/\partial \mathbf{u}$› is also reversed and drives learning in the wrong direction. The system never recovers. (Ordinate scale refers to velocities, not sensitivity derivatives.)

3.3 Are sensitivity derivatives innately known or learned?

Theoretically, innateness and learning each have their own pros and cons. If sensitivity derivatives are not learned then the system can get by with a simpler learning algorithm, and it will often still adapt, eventually, to changes in the plant that don't reverse any component of $\partial L/\partial \boldsymbol{u}$; but if any element of $\partial L/\partial \boldsymbol{u}$ ever *does* change sign, the system will be helpless, as all its attempts at adaptation will just make things worse. On the other hand, if sensitivity derivatives are learned then the learning algorithm will have to be more sophisticated, but the system will adapt to a far wider range of conditions, and it will have other advantages, which I will discuss later.

3.3.1 Current theories

What do the leading theories of sensorimotor learning say about sensitivity derivatives? Most don't mention them: in their simulations they assume the derivatives are known, without discussing how. But a few papers do posit fairly explicitly that knowledge of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is innate (Kawato & Gomi 1992; Porrill, Dean & Stone 2002 & 2004).

There are many theories of plasticity in the VOR, but none of them explain how sensitivity derivatives could be learned, so they predict that the VOR will never recover when $\partial L/\partial \boldsymbol{u}$ reverses sign, as in Figure 2B. Many of these theories do explain our adaptation to reversing *spectacles*, but spectacles don't reverse the sign of $\partial L/\partial \boldsymbol{u}$ for eye control: they alter the relation between retinal image slip and head motion, but not the relation between slip and motor commands. For example if, while wearing reversing glasses, you see your visual world slipping rightward then

the correct response is still the usual one: rotate your eyes faster to the right. To reverse the relation between slip and command, an experimenter could transpose the eye muscles, or put in reversing contact lenses, or use a virtual-reality setup where one measures eye movement and programs the visual scene to rotate, as a function of eye motion, *as if* the subject were wearing reversing contacts. So far as I am aware, no one has carried out this experiment and no previous theory predicts that the VOR would recover under these conditions. (On the other hand, reversing prisms *do* reverse $\partial L/\partial \boldsymbol{u}$ in tasks involving skeletal movements like reaching, as I discuss in Section 3.3.2 below.)

In the general learning theory of Kawato and colleagues, information about sensitivity derivatives is built into an innate controller and delivered by it to an adaptive controller. This point is clear in their early papers (Kawato & Gomi 1992) — they don't mention $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ by name but they call for an innate controller that works at least half-decently, which means it responds to errors in an appropriate way, given the $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ of its plant. So the innate controller contains information about $\partial \boldsymbol{e}/\partial \boldsymbol{u}$.

Later papers from this same lab focus on other issues, but they work similarly as regards $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. For example in Yamamoto *et al.* (2002), synaptic change depends on unlearned constants that carry information about $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. Kawato and colleagues have a specific theory of VOR adaptation, developed in the 1990s but still in use in recent papers such as Shibata *et al.* (2005). Simulations of that theory, applied to the same tasks as those in Figure 3.2, are indistinguishable from the plots in that figure — it fails to recover from a reversal in $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. On the other hand, if this theory were supplemented with a mechanism for learning sensitivity derivatives then it would recover; in other words, this sort of simulation doesn't challenge any aspect of Kawato's theory except its reliance on innate estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$.

A more recent general theory of motor learning, developed by Porrill, Dean and Stone (2004) similarly relies on innate estimates. Applied to the VOR, it too reproduces the behavior in Figure 3.2. And as with Kawato's theory, this one could be made flexible by adding a mechanism that learns sensitivity derivatives.

Another example is the papers by Todorov and colleagues on the LQG framework for motor control (*e.g.* Todorov & Jordan 2002). Their algorithms for shaping motor control sequences aren't presented as biologically feasible, but still they illustrate that knowledge of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is needed to shape a controller. In these papers a central role is given to a matrix $\boldsymbol{B}$ (or $\boldsymbol{B}_k$), which is not itself learned but is assumed to be known. $\boldsymbol{B}$ is defined to be $\partial \boldsymbol{x}/\partial \boldsymbol{u}$, where $\boldsymbol{x}$ is the plant state, so if we give the name $\boldsymbol{x}^*$ to the optimal state (at some moment in the trajectory) and let $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}^*$, then $\boldsymbol{B} = \partial \boldsymbol{e}/\partial \boldsymbol{u}$.

3.3.2 Experimental evidence

If knowledge of sensitivity derivatives were solely innate then any major change in the plant — any change that caused the controller's estimate of $\partial L/\partial \boldsymbol{u}$ to have the wrong sign — would cause learning to become counterproductive, strengthening those commands that should be weakened and vice versa. But a long series of studies, going back to G. M. Stratton in the 1890s, show that real learning, in at least some sensorimotor systems, doesn't become counterproductive when a component of $\partial L/\partial \boldsymbol{u}$ changes sign.

For instance, people can learn to mirror-draw, or live with reversing goggles, even though the mirror and goggles reverse the relation between visual error and motor commands (Stratton

1897; Ewert 1930; Sugita 1996). Similarly, when antagonist muscles or nerves are transposed, then the relation between motor commands and motion is reversed, but animals can regain their coordination (Sperry 1945; Missiuro & Kozlowski 1963; Vera et al. 1975; Leffert & Meister 1976; Yumiya, Larsen & Asanuma 1979; Brinkman, Porter & Norman 1983).

Another example, involving not a reversal but a drastic qualitative change in $\partial L/\partial \boldsymbol{u}$, is facial palsy treated by hypoglossal nerve transposition. In these cases, the facial nerve is damaged, so surgeons cut it and attach to its stump a branch of the nerve to the tongue. In this way, errors in facial motion become associated with motor commands in the hypoglossal nerve which formerly had no effect at all, *i.e.* where $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ was formerly zero. Right after the operation, patients move their faces whenever they try to move their tongues, but in time they learn to control face and tongue independently. So this is another case where the brain copes well with an extreme change in $\partial L/\partial \boldsymbol{u}$.

Not *all* controllers recover when $\partial L/\partial \boldsymbol{u}$ reverses, and in particular small brains don't always cope well. For example in 1943 Sperry rotated the eyes of some newts upside-down, so food sighted in their lower visual fields could now be reached only by moving up. The newts never learned the new arrangement, even over 4 months. Success on these tasks appears to vary from species to species. On the tasks that have been tested, it seems that primates adapt consistently, cats and rats show mixed results, and newts never recover (Ewert 1930; Sperry 1943 & 1945; Vera Lewin, Kasa & Calderon 1975; Leffert & Meister 1976; Yumiya, Larsen & Asanuma 1979; Brinkman, Porter & Norman 1983; Forssberg & Svatengren 1983; Sugita 1996). So maybe simple organisms do rely solely on innate estimates of sensitivity. And it is possible that simple or primitive subsystems of large brains do the same. In this regard, it would be interesting to study the human response to reversed $\partial L/\partial \boldsymbol{u}$ in a simple system like the VOR.

But it is clear that in at least some sensorimotor systems, learning is flexible in the sense that it can deal with reversals in $\partial L/\partial \boldsymbol{u}$. We have looked at simple examples involving limb and tongue movements, because in them the commands and loss functions are fairly easy to identify, but flexibility becomes even more vital in more-complex control systems, because with more components in $\partial L/\partial \boldsymbol{u}$ there is a greater risk that some component will change sign — and again, reversal in even one component will confound forever an inflexible learner. For example, if $\boldsymbol{u}$ is 10-dimensional then an inflexible learner can adapt to only 2^{-10}, or ~0.1%, of all possible values of the partial-derivative of the loss with respect to $\boldsymbol{u}$. This calculation, and the experimental data, both indicate that in a complex, changing world, survival depends on learning sensitivity derivatives.

3.3.3 Learning sensitivity

Given this fact, we have to address what has long been the major objection to learned sensitivity: the lack of a feasible learning mechanism (Kawato & Gomi 1992). Jordan and Rumelhart (1992) have suggested an approach called the *distal teacher*, which I show in Figure 3.3. In this method, one introduces a device called a plant model, which receives the same inputs as the plant and learns to simulate its behavior. The output of the model is compared with the output of the real plant, and difference is called the *model error*, $\bar{\boldsymbol{e}}$, because it measures how well the model is doing its job of mimicking the plant. The model error is fed back to the model where it alters the synapses, improving performance. Once the model has been trained (once it has become a faithful simulation of the plant) it contains a lot of information about the plant, including the

sensitivity derivatives, which it sends to the controller. And with $\partial \boldsymbol{e} / \partial \boldsymbol{u}$ in hand, the controller can learn its job.

But distal teachers haven't caught on, because in Jordan and Rumelhart's implementation the information about sensitivity was distributed among the firing patterns and synaptic weights of the model. To get that information to the controller, then, one needed fast weight transport — rapid transmission of information about synapses to other, remote synapses (Figure 3.3). This kind of transport is biologically implausible (Mazzoni, Andersen & Jordan 1991; Kawato & Gomi 1992; Rolls & Deco 2002), which is why recent theories have instead suggested that knowledge of sensitivity derivatives is innate. I will show that there are ways sensitivity derivatives could be learned using only biologically realistic forms of information transport. I start in the next section with the mechanism I consider most promising — essentially a version of the distal teacher without weight transport — and then consider other approaches.

3.4 Implicit supervision

3.4.1 The basic idea

What we want is a plant model that represents sensitivity derivatives not in its synaptic weights but in its neural firing, so they are available to be transmitted wherever they are needed, such as to adaptive controllers. The problem is that there is no supervisor to train such a model — no signal to tell it the true values of the sensitivity derivatives. But no supervisor is needed if we relate the unknown sensitivity derivatives to known variables.

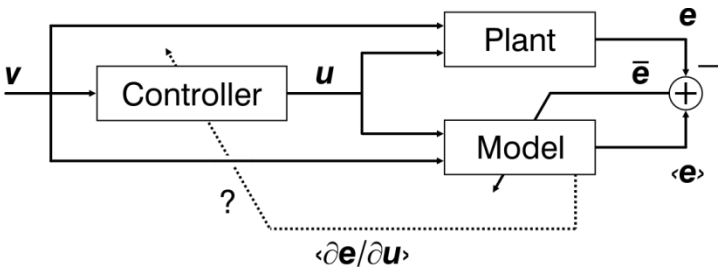

Figure 3.3: How could controllers learn sensitivity derivatives? As Jordan and Rumelhart (1992) showed, a plant model contains information about those derivatives. But normally the information is distributed among the synaptic weights of the model. Unless that information is represented in neural firing, how can it be transmitted to the controller? (To save space in this diagram, the box labelled “plant” incorporates not just the plant equation 3.1 but also the processes that determine the error, so it actually corresponds to the function e = g(v, u) in Equation 3.2.)

If we define $\boldsymbol{z} = (\boldsymbol{v}, \boldsymbol{u})$ then we can write

$$\boldsymbol{e} = \boldsymbol{g}(\boldsymbol{v}, \boldsymbol{u}) = \boldsymbol{g}(\boldsymbol{z}). \tag{3.12}$$

The point is to provide a single vector, $\boldsymbol{z}$, which determines $\boldsymbol{e}$. As defined in Example 3.1.1, the context vector $\boldsymbol{v}$ is everything besides $\boldsymbol{u}$ that is needed to determine $\boldsymbol{e}$, so by definition $\boldsymbol{e}$ is a function of $\boldsymbol{z}$, the combination of $\boldsymbol{v}$ and $\boldsymbol{u}$. In the VOR, for example, $\boldsymbol{z}$ contains information about the motor command, eye position, and head velocity. In reaching, $\boldsymbol{z}$ includes motor commands, target locations, and the angles and velocities of the joints.

By the chain rule,

$$\dot{\boldsymbol{e}} = \frac{d\boldsymbol{e}}{d\boldsymbol{z}}\,\dot{\boldsymbol{z}}. \tag{3.13}$$

We can reasonably assume that the rates of change $\dot{\boldsymbol{e}}$ and $\dot{\boldsymbol{z}}$ are known to the plant model, because they can be computed from $\boldsymbol{e}$ and $\boldsymbol{z}$. The model's aim, then, is to deduce the unknown derivative matrix $d\boldsymbol{e}/d\boldsymbol{z}$, which contains as a submatrix the sensitivity derivatives $\partial\boldsymbol{e}/\partial\boldsymbol{u}$.

So we want a network that learns to compute $d\boldsymbol{e}/d\boldsymbol{z}$. By Equation 3.12, $\boldsymbol{e}$ is determined by $\boldsymbol{z}$, so $d\boldsymbol{e}/d\boldsymbol{z}$ is computable from $\boldsymbol{z}$ as well. Usually the function relating $\boldsymbol{z}$ and $d\boldsymbol{e}/d\boldsymbol{z}$ is nonlinear, so the simplest way to get a useful input is by "expansion-recoding" $\boldsymbol{z}$ (Rolls & Deco 2002), sending it through an array of nonlinear functions φ_i to yield a *feature vector* $\boldsymbol{\varphi}(\boldsymbol{z})$, or $\boldsymbol{\varphi}$ for short. Then we approximate each element of $d\boldsymbol{e}/d\boldsymbol{z}$ by taking the inner product of $\boldsymbol{\varphi}$ with a weight vector $\boldsymbol{w}$,

$$\frac{\partial e_i}{\partial z_j} \approx \boldsymbol{w}_{ij}{}^{\mathsf{T}}\boldsymbol{\varphi} = \sum_{k=1}^{n_\varphi} w_{ijk}\varphi_k \tag{3.14}$$

as in Figure 3.4.

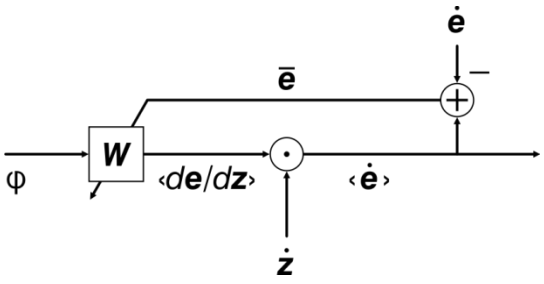

Figure 3.4: Implicit supervision creates a network that represents sensitivity derivatives in transmissible form. Here ***z*** is a vector containing information about the context ***v*** and command ***u***, and φ is a function of ***z***. This circuit learns to code the total derivative $d\mathbf{e}/d\mathbf{z}$ in neural firing. Because $d\mathbf{e}/d\mathbf{z}$ contains the sensitivity $\partial\mathbf{e}/\partial\mathbf{u}$ as a submatrix, all components of the estimate $\langle\partial\mathbf{e}/\partial\mathbf{u}\rangle$ are coded in a subset of the axons carrying $\langle d\mathbf{e}/d\mathbf{z}\rangle$. So this scheme can deduce sensitivity derivatives and transmit them to a controller.

Expansion recoding and feature vectors appear also in many other theories that strive for biological realism (*e.g.* Rolls & Deco 2002; Kawato & Gomi 1992; Yamamoto et al. 2002), and there are many ways to choose the features, φ_i. If you know beforehand that certain features are well-suited for your task then it makes sense to choose them, *e.g.* if you know that the sensitivity derivatives are quadratic functions of $\boldsymbol{z}$, choose quadratic features. If you don't have much prior knowledge then more-generic features also work, so long as you have a large and varied set of them. In this paper I take both approaches to feature selection — handpicking for the VOR, generic for my simulations of a 2-joint arm. For the VOR, the sensitivity derivative is a single, constant scalar, so we choose a single, constant feature — $\varphi = 1$. For the arm model, we make no attempt to select apt features but simply put $\boldsymbol{z}$ through an array of up to 25 hyperbolic tangent neurons, with random, fixed (non-learning) synaptic weights w_f, to yield a $\boldsymbol{\varphi}$ vector with components

$$\varphi_i = tanh\left(\sum_{j=1}^{n_z} w_{fij} z_j \right) \tag{3.15}$$

One concern with all expansion-recoding schemes is that complex tasks call for many features. No one knows how many, chosen from this or that generic set, are needed for realistic sensorimotor control, but if the number turns out to be implausibly large, we can invoke several mechanisms to create smaller sets, *e.g.* features could be shared between sensorimotor systems, useful features could be built into the brain from birth by natural selection, or they could be shaped by learning upstream from $\boldsymbol{W}$. And similarly, upstream learning could also provide an efficient, low-dimensional $\boldsymbol{z}$.

Given a feature vector, learning becomes a matter of finding the weights w_{ijk} that yield the best approximations to $\partial e_i/\partial z_j$. Again, there is no supervisor coding the true values of $\partial e_i/\partial z_j$, but from

Equation 3.13 we know that if we adjust the w_{ijk} so that the product $\langle d\boldsymbol{e}/d\boldsymbol{z}\rangle\,\dot{\boldsymbol{z}}$ is close to $\dot{\boldsymbol{e}}$ for a wide range of $\dot{\boldsymbol{z}}$ vectors then we will have $\langle d\boldsymbol{e}/d\boldsymbol{z}\rangle \approx d\boldsymbol{e}/d\boldsymbol{z}$.

What is the learning rule that will achieve this goal? We define a model error

$$\bar{\boldsymbol{e}} = \langle\dot{\boldsymbol{e}}\rangle - \dot{\boldsymbol{e}} = \langle d\boldsymbol{e}/d\boldsymbol{z}\rangle\dot{\boldsymbol{z}} - \dot{\boldsymbol{e}} \tag{3.16}$$

and model loss

$$\bar{L} = \frac{\bar{\boldsymbol{e}}^{\mathsf{T}}\bar{\boldsymbol{e}}}{2}. \tag{3.17}$$

To minimize this loss, we want to adjust each w_{ijk} down the gradient

$$\begin{aligned}
\frac{\partial \bar{L}}{\partial w_{ijk}} &= \frac{\partial \bar{L}}{\partial \bar{\boldsymbol{e}}}\frac{\partial \bar{\boldsymbol{e}}}{\partial w_{ijk}} = \bar{\boldsymbol{e}}^{\mathsf{T}}\frac{\partial \bar{\boldsymbol{e}}}{\partial w_{ijk}} = \sum_{\alpha=1}^{n_e}\bar{e}_\alpha\frac{\partial \bar{e}_\alpha}{\partial w_{ijk}} \\
&= \sum_{\alpha=1}^{n_e}\bar{e}_\alpha\frac{\partial}{\partial w_{ijk}}\left(\left[\langle d\boldsymbol{e}/d\boldsymbol{z}\rangle\dot{\boldsymbol{z}} - \dot{\boldsymbol{e}}\right]_\alpha\right) \quad \text{(by equation 3.16)} \\
&= \sum_{\alpha=1}^{n_e}\bar{e}_\alpha\frac{\partial}{\partial w_{ijk}}\left(\sum_{\beta=1}^{n_z}\langle de_\alpha/dz_\beta\rangle\dot{z}_\beta\right) \quad \text{(omit } -\dot{\boldsymbol{e}}\text{, as it doesn't depend on } w_{ijk}\text{)} \\
&= \bar{e}_i\frac{\partial}{\partial w_{ijk}}\left(\langle de_i/dz_j\rangle\dot{z}_j\right) \quad (\langle de_\alpha/dz_\beta\rangle \text{ depends on } w_{ijk} \text{ only if } \alpha = i,\ \beta = j) \\
&= \bar{e}_i\frac{\partial}{\partial w_{ijk}}\left(\sum_{\gamma=1}^{n_\varphi}w_{ij\gamma}\varphi_\gamma\dot{z}_j\right) \quad \text{(by equation 3.14)} \\
&= \bar{e}_i\frac{\partial}{\partial w_{ijk}}\left(w_{ijk}\varphi_k\dot{z}_j\right) \\
&= \bar{e}_i\dot{z}_j\varphi_k
\end{aligned} \tag{3.18}$$

where the notation $[\ ]_\alpha$ means component number α of the vector in the square brackets. So a simple learning rule would be

$$\dot{w}_{ijk} = -\eta \bar{e}_i \dot{z}_j \varphi_k . \tag{3.19}$$

where η is a positive number called the learning rate parameter. This rule requires no weight transport, as all the variables are either present in the synapse automatically (w_{ijk}, η and φ_k) or rapidly transmissible there because they are coded in neural firing ($\bar{e}_i$ and $\dot{z}_j$) — Figure 3.4 shows $\bar{\boldsymbol{e}}$ arriving at the weight array $\boldsymbol{W}$; to avoid clutter here and in Figure 3.5, I don't show $\dot{\boldsymbol{z}}$ being sent to $\boldsymbol{W}$, but $\dot{\boldsymbol{z}}$ is present and coded in action potentials, so it can be delivered to $\boldsymbol{W}$ by axon collaterals.

Equation 3.19 is a variant of the LMS learning rule (Haykin, 2002), though it differs from most applications of LMS in that it uses a supervisor to train its output $\langle \dot{\boldsymbol{e}} \rangle$, but only as a means of driving certain of its internal signals to equal a different, supervisorless variable, namely $d\boldsymbol{e}/d\boldsymbol{z}$. I call this mechanism *implicit supervision*, because $\dot{\boldsymbol{e}}$ acts as a kind of indirect supervisor to train the network to compute $d\boldsymbol{e}/d\boldsymbol{z}$.

Many other learning rules besides LMS can be used for implicit supervision. In my simulations of implicit supervision in this paper, I use a rule called *normalized least-mean-square*, or NLMS (Nagumo & Noda 1967):

$$\Delta w_{ijk} = -\eta \frac{\bar{e}_i \dot{z}_j \varphi_k}{\left(\dot{\boldsymbol{z}}^\mathsf{T} \dot{\boldsymbol{z}}\right)\left(\boldsymbol{\varphi}^\mathsf{T} \boldsymbol{\varphi}\right)} \tag{3.20}$$

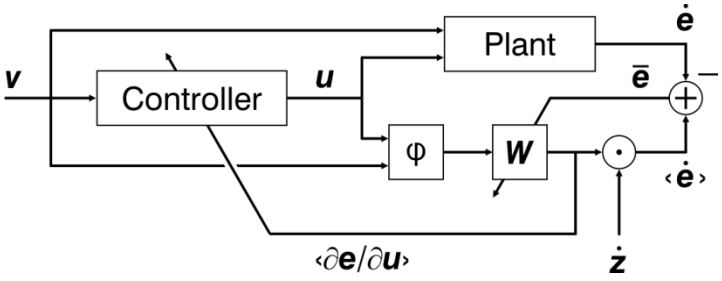

Figure 3.5: Control by implicit supervision. The circuit from Figure 3.4, incorporated at lower right in this flow diagram, serves as a plant model which learns to represent ∂***e***/∂***u*** in neural firing and sends the information to the controller.

The simulations run in discrete time, so here Δw_{ijk} is the change in weight w_{ijk} in the current time step. An advantage of NLMS over LMS is that it converges as fast but is less fussy about input statistics. That is, to get good convergence with LMS, you have to choose a suitable learning rate constant η, and the optimal η depends on the variances of the inputs to the network. But with NLMS, the optimal η is 1, regardless of input variance. (In Figures 3.2 and 3.6A I deliberately chose a suboptimal rate constant, setting η equal to 0.01 to slow down the learning and make the different stages more visible. In all other the figures showing implicit supervision, I used the optimal η.) Another possible learning rule, which I haven't used in this thesis but which may be viable in the brain, is the recursive least-squares algorithm, or RLS, which is faster than LMS and NLMS, though also more complex (Haykin 2002).

However it is implemented, the idea behind implicit supervision is quite general: to compute a variable for which there is no supervisor, relate it to another variable that does have a supervisor, and build a circuit that reflects the known relation between the two. Then as one signal converges to the supervisor, another converges to the variable you want.

Applied to sensorimotor learning, the idea looks like Figure 3.5. The lower part of the circuit serves as a plant model, and because it codes $\boldsymbol{de}/\boldsymbol{dz}$ and therefore $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u} \rangle$ in its firing, not in its weights, it can rapidly transmit that information to the controller.

The flexibility of this scheme is illustrated by simulations in Figure 3.6: unlike those controllers whose knowledge of sensitivity is solely innate, those trained by implicit supervision can recover when $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u} \rangle$ changes so as to reverse the sign of some component of $\partial L/\partial \boldsymbol{u}$.

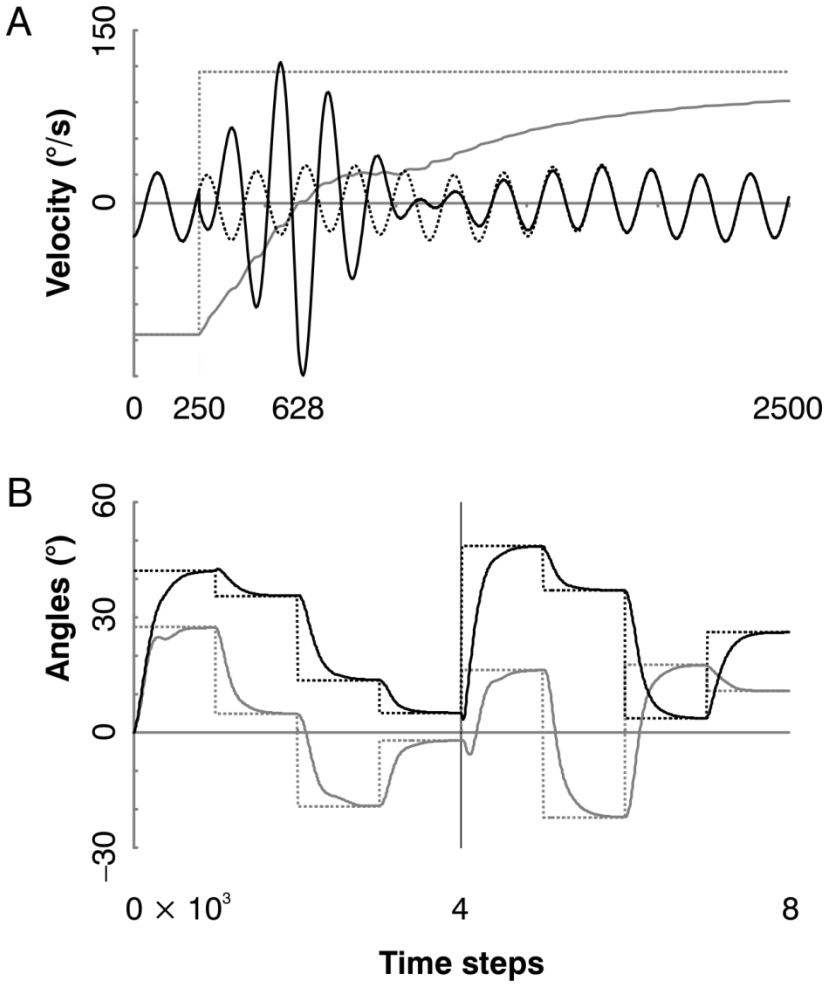
A
150
0
Velocity (°/s)
0
250
628
2500
B
60
30
0
−30
Angles (°)
0 × 10³
4
8
Time steps

Figure 3.6: Implicit supervision provides flexible adaptive control. (A) As in Figure 3.2B, the sensitivity derivative switches sign at time step 250, but here the controller learns in the wrong direction only until time step 628, when the learned estimate ‹$\partial \mathbf{e}/\partial \boldsymbol{u}$› (the rising, solid gray line) crosses zero, regaining the correct sign, and thereafter control recovers. (B) The same method works for a more complex task: we see shoulder and elbow angles (gray and black solid lines) and their targets (dotted) as a 2-joint arm learns to reach. At time step 4000, all components of sensitivity change sign, as if antagonist muscles were transposed at both joints, but control recovers. For details of the arm simulation, see Appendix B.

Figure 3.7 illustrates two further points regarding implicit supervision. The first is that the plant model computes not a single, fixed $d\boldsymbol{e}/d\boldsymbol{z}$ matrix but a function that takes $\boldsymbol{u}$ and $\boldsymbol{v}$ to $d\boldsymbol{e}/d\boldsymbol{z}$, so it still works when $d\boldsymbol{e}/d\boldsymbol{z}$ varies as a function of $\boldsymbol{u}$ and $\boldsymbol{v}$. This point follows from my equations in Section 3.4, and Figures 3.7A, B provide an example: the sensitivity derivatives change as the arm moves about its workspace, but the plant model's estimates continues to track them. The second point is that an adaptive controller doesn't need the exact values of the sensitivity derivatives, but only the signs of all components of $\partial L/\partial \boldsymbol{u}$, as I discussed in Section 3.2. It follows, then, that noise on the model's estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ will not prevent learning so long as the estimated sign of $\partial L/\partial \boldsymbol{u}$ is correct. Figure 3.7C provides an example: the model reports only the signs of both components of $\partial L/\partial \boldsymbol{u}$, but even with this limited information, the controller gradually learns its job.

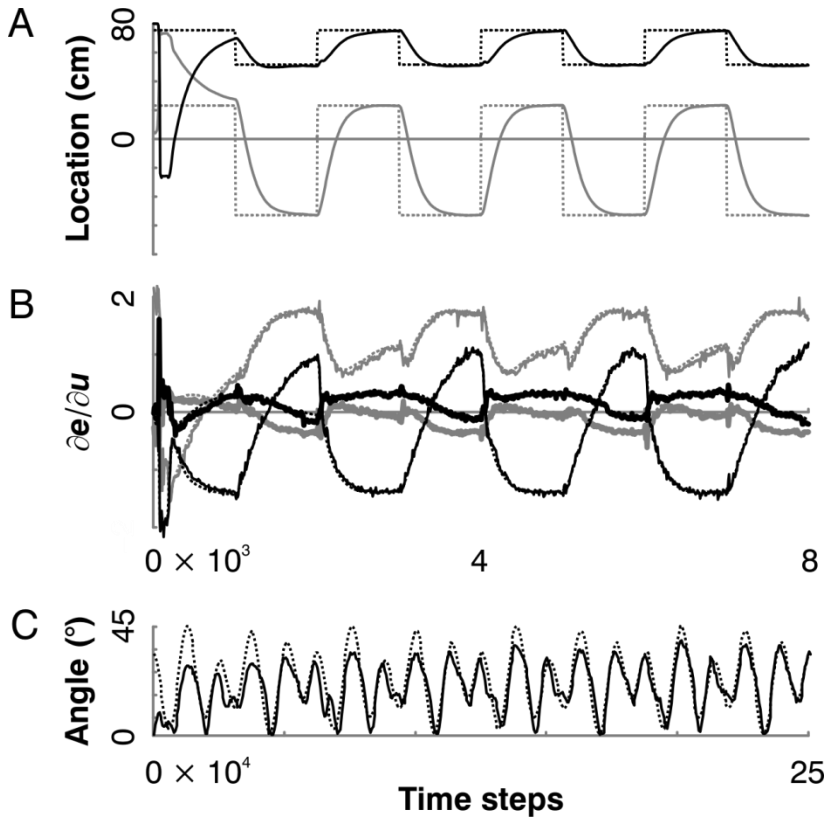

Figure 3.7: Properties of implicit supervision. (A) A controller learns to move the horizontal and vertical components of hand location (black and gray solid lines) to their targets (dotted), even though the sensitivity derivatives change through the workspace. (B) The plant model's estimates (solid lines) of all four sensitivity derivatives track the true values (dotted). Specifically, if e_1 is the horizontal component of hand location error and e_2 is the vertical, and u_1 and u_2 are the motor commands to shoulder and elbow, then $\partial e_1/\partial u_1$ is the thick black line, $\partial e_1/\partial u_2$ the thin black, $\partial e_2/\partial u_1$ the thick gray, and $\partial e_2/\partial u_2$ the thin gray. (C) Control eventually becomes accurate even when the controller receives only the signs of the elements of $\partial L/\partial \boldsymbol{u}$.

3.4.2 Redundancy and constraints

Figure 3.8 shows that implicit supervision can deal with kinematic redundancy, *i.e.* with situations where the plant has more degrees of freedom than it needs for its task. In Figure 3.8A, the task is to control a 2-joint arm so that the angle of the forearm in space (which is the sum of the shoulder and elbow joint angles) matches some target value. The system is redundant because it uses two joints to control a single number, the orientation of the forearm. As shown in Figure 3.8A, the controller learns to drive the forearm (dashed line) to its target (dotted line). The angles of the individual joints — the elbow in black and the shoulder in gray — wander about, even when their sum, the forearm angle, is being held steady, as one would expect, because the controller is concerned only with the forearm, and cares not at all about the individual joints. Such complete indifference would be unlikely in a real sensorimotor system, because it permits the individual joints to wander without bound, or to knock against their mechanical stops, impairing performance. A more plausible controller might constrain the redundant variables, as in Figure 3.8B, where the controller learns to orient the forearm, as before, but now prefers configurations where shoulder and elbow angles are equal. Figure 3.8C shows another version, which prefers to hold the elbow near 15° of flexion. This last case is closely analogous to the control of saccadic eye movements, where each eyeball has three degrees of freedom — horizontal, vertical, and torsional — but needs only two, and so Listing's law of the eye holds ocular torsion near 0° (Tweed, Misslisch & Fetter 1994). Interestingly, Figure 8C shows small fluctuations of the elbow angle when the target moves, like the "blips" seen in ocular torsion during saccades (Tweed, Misslisch & Fetter 1994). These blips may arise for the same reason in Figure 8C and in real saccades — the constraint has been imperfectly learned.

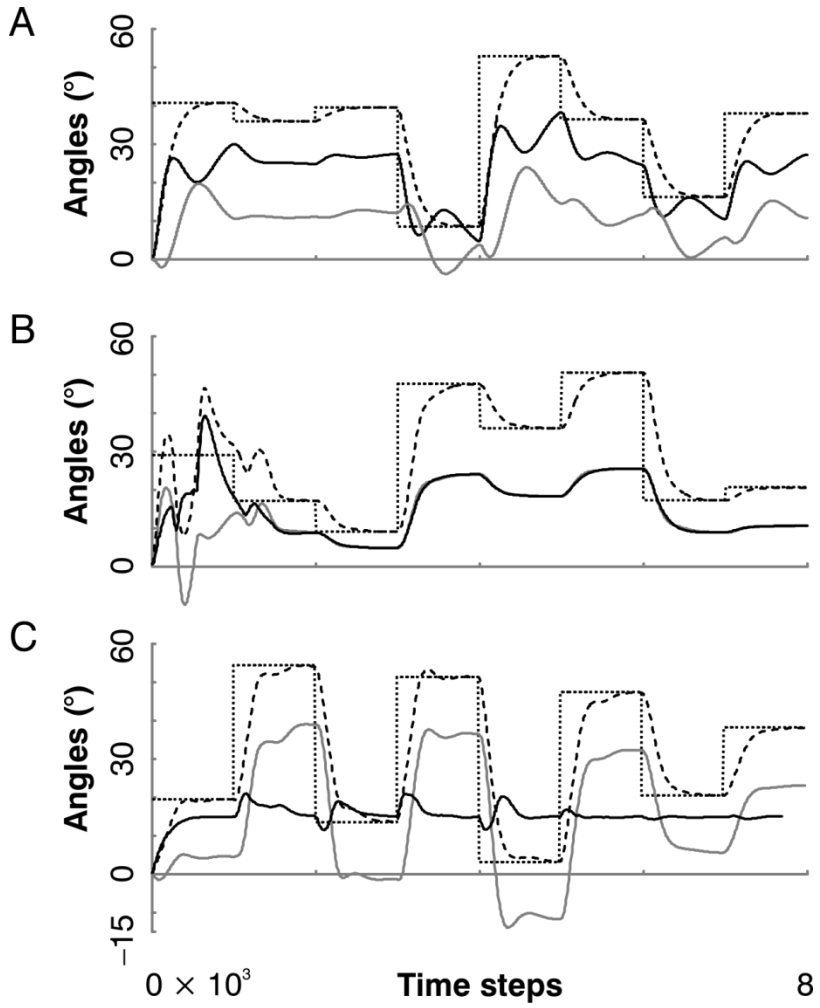

Figure 3.8: Implicit supervision can cope with kinematic redundancy. (A) The task is to control shoulder and elbow so as to bring the forearm's angle relative to space (dashed line) to a target value (dotted line). The controller learns the task, but allows the angles of the individual joints — the elbow in black and the shoulder in gray — to wander. (B) A controller learns to orient the forearm, as before, but with a more-complex error signal which prefers arm configurations where shoulder and elbow angles are equal. (C) Another version prefers to hold the elbow near 15° of flexion.

3.4.3 Convergence, speed and neurons

It is straightforward to show, by Lyapunov's second method, that model error $\bar{\boldsymbol{e}}$ converges to zero (so long as zero error is attainable, given the feature vector $\boldsymbol{\varphi}$). The Lyapunov proof doesn't say how *quickly* the error declines, because the rate depends on the sequence of input vectors $\boldsymbol{z}$. Using methods from Gardner (1984) and Werfel, Xie & Seung (2005) we can prove that learning time is roughly proportional to the product $n_z n_\varphi$., or in other words to the number of adjustable weights divided by the number of elements of $\boldsymbol{e}$. But this method of proof relies on assumptions about the statistics of the network's input signals which are unrealistic in a sensorimotor setting. We can do better if we revise our learning rule: instead of deriving it from LMS or NLMS, as in Equations 3.19 or 3.20, we can base it on recursive least squares, or RLS (Haykin 2002). The RLS version of implicit supervision will normally converge over a time interval that is proportional to $n_z n_\varphi$, given realistic signal statistics. But if we stick to our simpler, NLMS-based learning rule from Equation 3.20, we can still get an impression of learning speed, by using simulations, as in Figures 3.6B, 3.7A and 3.7B, and 3.8. In these figures, learning runs much faster than in real life, but the speed is a welcome feature, because learning would slow down if the simulations were made more complex, with realistic numbers of neurons and degrees of freedom. We need fast convergence to explain how real sensorimotor systems learn as quickly as they do, given their complexity.

As for neurons, the method in its most straightforward implementation calls for n_φ cells to compute the feature vector $\boldsymbol{\varphi}$ and $n_e n_z$ more to carry the elements of $\langle d\boldsymbol{e}/d\boldsymbol{z}\rangle$, plus smaller numbers of other neurons for other tasks.

3.5 Other approaches

There are other mechanisms besides implicit supervision that could learn sensitivity derivatives for sensorimotor control, though the literature at present offers only a few alternatives, because it contains very few algorithms that address my question, *i.e.* that are flexible in the sense that they recover from reversed sensitivity derivatives, that deal with nonlinear plants and with states and commands that are real-valued vectors rather than elements of finite sets, and that transport information in a way that is feasible for biological neural networks. So far as I know, the only algorithms that fulfill these conditions belong to just two classes: Boltzmann-like mechanisms and the reinforcement-learning algorithms called perturbation methods.

In the case of Boltzmann machines, for my purposes they would appear to need two "clamped" states rather than their usual one, adding complexity (Ackley, Hinton & Sejnowski 1985; Rolls & Deco 2002). Further, any clamping would interrupt the controller, and this seems unlikely for sensorimotor systems, which continue working while they learn. But some newer variant of the Boltzmann machine (*e.g.* Hinton, Osindero & Teh 2006) may avoid these problems.

3.5.1 Perturbation

The most promising alternative to implicit supervision may be the reinforcement-learning method known as node perturbation (Werfel, Xie & Seung 2005), which works by taking two stabs at the same problem and seeing which did better. In a control setting it would look like this: a controller computes a command $\boldsymbol{u}$, *e.g.*

$$\boldsymbol{u} = \boldsymbol{w} \cdot \varphi(\boldsymbol{v}), \tag{3.21}$$

where φ is a feature vector derived from the context $\boldsymbol{v}$. This $\boldsymbol{u}$ is sent to the plant where, together with $\boldsymbol{v}$, it determines the loss, L. Immediately thereafter, before the context has had a chance to change, the controller emits a randomly perturbed command $\boldsymbol{u}_p$, resulting in a slightly different loss, L_p. Then the controller adjusts its weights according to the learning rule,

$$\dot{w}_i = -\eta(L_p - L)(\boldsymbol{u}_p - \boldsymbol{u})\varphi_i. \tag{3.22}$$

Here the quantity $(L_p - L)(\boldsymbol{u}_p - \boldsymbol{u})$ — an estimate of $\partial L/\partial \boldsymbol{u}$, or in other words of $\boldsymbol{e}^\mathsf{T}\partial \boldsymbol{e}/\partial \boldsymbol{u}$ — carries information about sensitivity derivatives to the controller. The method may require fewer neurons than implicit supervision — essentially just the n_φ cells that compute the controller's φ.

But there is a problem: in a control setting, it is impossible to take two stabs at exactly the same problem, because the context is always in flux. It is impossible to send out a second command $\boldsymbol{u}_p$ before $\boldsymbol{v}$ has evolved, if only because the first command, $\boldsymbol{u}$, itself alters $\boldsymbol{v}$. So if, for instance, L_p is smaller than L, it need not mean that $\boldsymbol{u}_p$ is a better command than $\boldsymbol{u}$; it may be a worse command delivered in a more favorable context. As a result, $(L_p - L)(\boldsymbol{u}_p - \boldsymbol{u})$ is a poor estimate of $\partial L/\partial \boldsymbol{u}$, and the controller learns far more slowly than with implicit supervision.

We can improve the method by sending $\boldsymbol{u}$ alone to the real plant but both $\boldsymbol{u}$ and $\boldsymbol{u}_p$, one after the other, to a plant model. That way, the simulated context vector $\boldsymbol{v}$ driving the model really can be made identical for both commands. The controller learns faster this way, but still far slower than with implicit supervision, as you can see by comparing Figure 3.9 to Figure 3.6B. So this method seems unsuited for complex sensorimotor tasks, though it may be viable in settings where learning quickly is less important than getting by with few neurons.

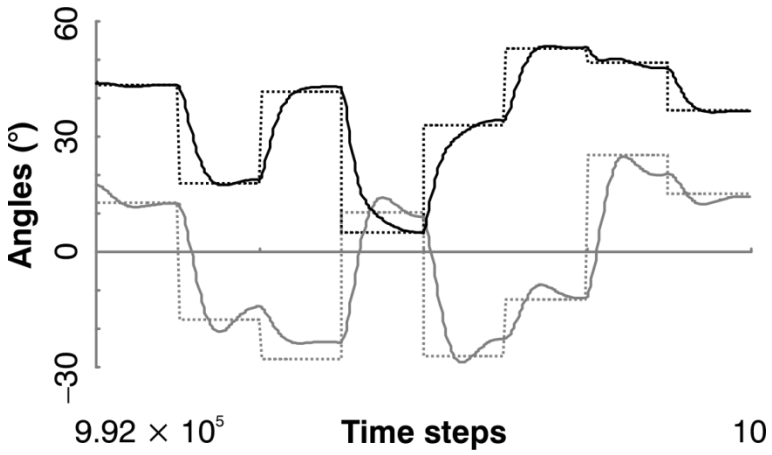

Figure 3.9: A controller learns by command perturbation to reach for targets, but the learning is slow compared to implicit supervision in Figure 3.6A. This plot shows the final 8000 time steps of a million-step run, when elbow control (black) is fairly good but the shoulder (gray) is still ragged.

3.5.2 Loss-based implicit supervision

In Equations 3.13–3.20 I described implicit supervision as a method of estimating $d\boldsymbol{e}/d\boldsymbol{z}$, but we can instead use the same ideas to estimate the loss-derivative $dL/d\boldsymbol{z}$. As the loss-derivative vector has fewer elements than the error-derivative matrix, we need fewer neurons to carry the estimates — just n_z rather than $n_e n_z$. But the components of $dL/d\boldsymbol{z}$ tend to be more-complex functions of $\boldsymbol{z}$ than are the components of $d\boldsymbol{e}/d\boldsymbol{z}$, so we need more features (more components in the vector $\boldsymbol{\varphi}$), and more neurons to compute them. To judge from simulations, loss-based implicit supervision appears less efficient than the error-based version.

3.5.3 Distal teacher

Jordan and Rumelhart's distal-teacher method is roughly as fast and general as implicit supervision. Its only flaw is that, in its one detailed formulation (Jordan & Rumelhart 1992), it relies on rapid weight transport, which is biologically implausible. But implicit supervision can be viewed as a distal-teacher method that works without weight transport. Figures 3.3 and 3.5 display the similarity between the two approaches: the chain of operators comprising $\boldsymbol{\varphi}$, $\boldsymbol{W}$ and the multiplier in Figure 3.5 forms a plant model, and plays the same role as the plant model in Figure 3.3 (Jordan and Rumelhart's model represents plant state $\boldsymbol{x}$ rather than error $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}^*$, but this is a minor difference of formulation).

3.6 Other advantages of learned sensitivity

Whatever the algorithm that underlies it, an ability to learn sensitivity derivatives brings some functional advantages besides flexibility.

3.6.1 Fast-learning controllers

In Figure 3.7 we saw that even very rough estimates of sensitivity derivatives can be used to train a controller. But on the other hand, the more exactly $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is known, the faster the controller can learn — compare the rapid improvement in Figure 3.6A with the slow progress in Figure 3.7C. So for this reason also, natural selection may have favored mechanisms for deducing $\partial \boldsymbol{e}/\partial \boldsymbol{u}$.

3.6.2 Learning complex plants

I have cited evidence for learned sensitivity from lesion studies, but learned sensitivity is useful even in the absence of lesions altering the sensitivity derivatives, because those derivatives will change naturally depending on context and commands. Consider a human arm, with 7 degrees of freedom (3 in the shoulder, 2 in the elbow, 2 in the wrist). If e_3 is the vertical component of reaching error — the vector from target to fingertip — and u_2 is a command for wrist flexion, then $\partial e_3/\partial u_2$ is positive when the elbow is supinated and negative when it is pronated — that is, wrist flexion moves your fingertips upward in the one case and downward in the other, so the relation between the flexion commands and visual error reverses, depending on the state of the elbow.

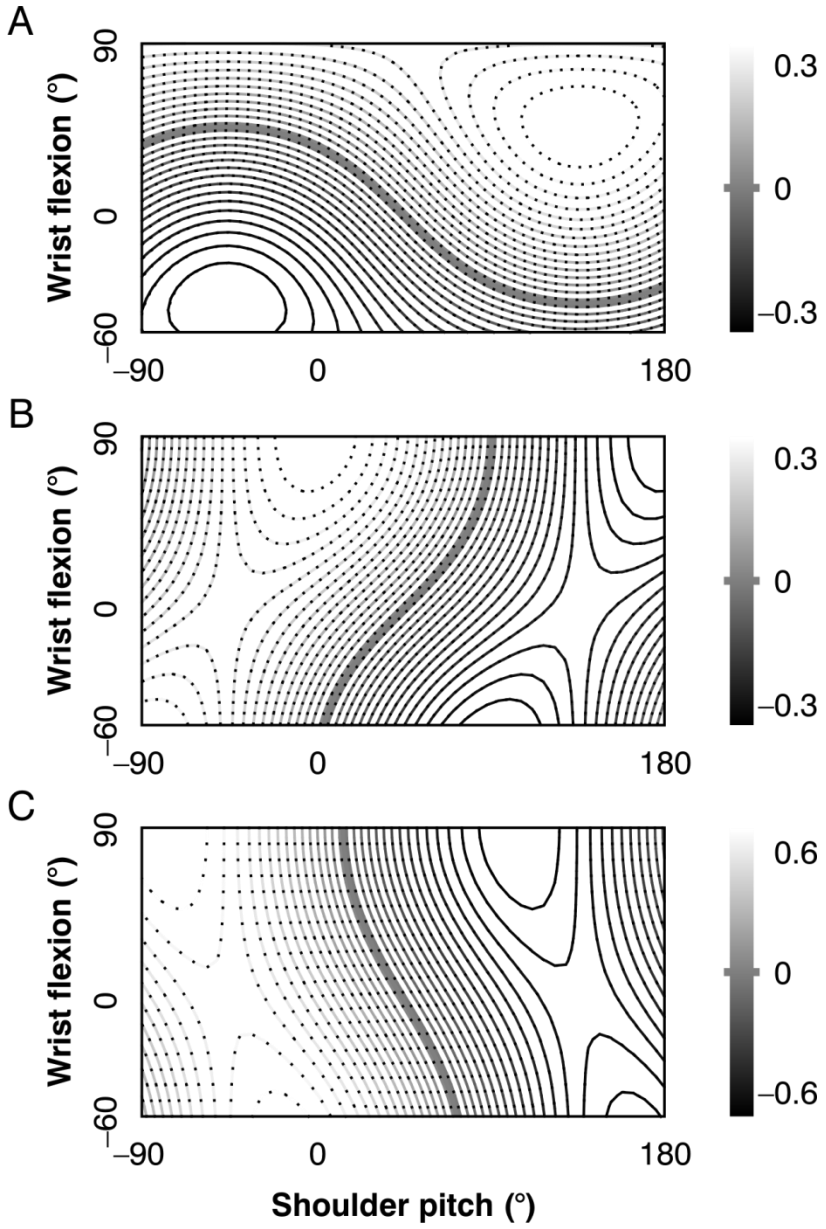
A
B
C
Wrist flexion (°)
90
0
−60
−90
0
180
0.3
0
−0.3
0.6
−0.6
Shoulder pitch (°)

Figure 3.10: Sensitivity derivatives vary depending on context. (A) A map of $\partial e_3/\partial u_2$ — the derivative of vertical reaching error with respect to wrist-flexion commands — versus shoulder pitch and wrist flexion, for a 7-DOF stepper-motor arm. The derivative is represented by the gray level, or lightness, of the contours (not of the dots, which are all black). The thick gray line marks zero. (B) A different derivative, $\partial e_3/\partial u_5$ (vertical error with respect to shoulder roll) plotted relative to the same joint angles, shoulder pitch and wrist flexion. (C) Effect of tools: $\partial e_3/\partial u_5$ again, plotted relative to the same joint angles, but now e_3 is the vertical error from the target to the tip of a hand-held stick.

Figure 3.10 gives an impression of how complex these dependencies can be. It is based on a simplified "stepper-motor" arm, where $\boldsymbol{u}$ is a vector of 7 commands that directly determine joint angles (in a real arm, $\boldsymbol{u}$ would have many more components and would affect the angles indirectly through the nonlinear dynamics of the muscles, so the relations would be even more complex). With the stepper arm, $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is a 3-by-7 matrix. Figure 3.10A shows a map of one component, $\partial e_3/\partial u_2$ (represented by contour lines), versus 2 joint angles. The derivative varies in a complicated way depending on arm position.

To be able to learn in all contexts, the controller has to know this map of derivatives versus context. Might the map be known innately, from the genome? Not likely, because it is complex: even with this simplified, stepper arm, the state space is 7-D. Through each point in the space there are 21 orthogonal 2-D slices, and over each slice, each of the 21 components of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ varies in a different way — compare Figures 3.10A and 3.10B. With a real, non-stepper arm the maps would be more complex, and the principle holds for sensorimotor tasks generally: $\boldsymbol{u}$ and $\boldsymbol{v}$ are usually high-dimensional, and the sensitivity derivatives vary over most of these dimensions.

Moreover, real sensorimotor plants involve elements outside the body. If you are making a bed then the pillows and blankets are part of the plant. If you are speaking to a group of people then they are part of the plant. Given this complexity, there may be many factors that could alter and reverse components of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. For instance, Figure 3.10C shows how one derivative map for reaching error transforms when you reach with a hand-held stick rather than a fingertip. So if a controller is to learn to use a tool, it has to know the derivative map for that tool. Does that mean we need thousands of maps — one for every size and shape of stick or hammer or screwdriver we pick up? No, tools could be represented by elements of the context vector $\boldsymbol{v}$, so we don't need multiple maps, just one map that is higher-dimensional than the one for a tool-free arm. And

implicit supervision makes it possible to learn the map rather than rely on estimates from the genome.

3.7 Conclusion

This theory boils down to three points, with different degrees of experimental support. First, I claim that at least some sensorimotor systems must deduce the sensitivity derivative matrix relating $\boldsymbol{e}$ and $\boldsymbol{u}$. The main evidence is that some systems recover when that relation changes so as to reverse $\partial L/\partial \boldsymbol{u}$ (Stratton 1897; Ewert 1930; Sperry 1945; Missiuro & Kozlowski 1963; Vera et al. 1975; Leffert & Meister 1976; Yumiya, Larsen & Asanuma 1979; Brinkman, Porter & Norman 1983; Sugita 1996). Further support comes from the arguments in Sections 3.6.1 and 3.6.2, that fast motor learning and tool use suggest detailed knowledge of sensitivity derivatives.

Second, I claim that sensitivity is transmitted by action potentials (together, of course, with synaptic transmission), and here the evidence is that studies of neural data flow have revealed no alternative. If it turned out that rapid weight transport were available after all, then the motivation for this claim would vanish, but all known mechanisms are far too slow (Rolls & Deco 2002; Oztas 2003).

Third, I have proposed three mechanisms for creating the sensitivity signal — command perturbation and two variants of implicit supervision — and argued for the error-based version of implicit supervision. Here the evidence shows that the mechanisms are plausible. For instance, all the operations in Figure 3.5, including multiplication and differentiation, can be done with neurons (Koch 1999; Tripp & Eliasmith 2006), and the mechanism is consistent with many

neural circuits; *e.g.* $\boldsymbol{\varphi}$ could be carried on parallel fibers, $\boldsymbol{W}$ could be synapses onto Purkinje cells, and $\dot{\boldsymbol{z}}$ could be inputs to deep cerebellar nuclei or brainstem. As for neural activity, each method is compatible with many different patterns: any given system can be controlled using many different $\boldsymbol{e}$'s, $\boldsymbol{z}$'s and $\boldsymbol{\varphi}$'s, and all these signals can be distributed across multiple neurons.

The theory makes further predictions, besides the ones discussed earlier. It implies that learning should be slower when a change in sensitivity reverses the sign of some component of $\partial L/\partial \boldsymbol{u}$ than when sensitivity changes by the same amount without causing a sign-change. It predicts that when a component of $\partial L/\partial \boldsymbol{u}$ changes sign the system should show an initial phase where control worsens or at least fails to improve, as in Figure 3.6A. In subjects who are thoroughly adapted to reversing goggles or transpositions, the theory predicts that their rapid, reflexive error corrections should also be appropriately reversed. And the theory says that if learning is based on Equation 3.22 then it should be blocked when sensory feedback is altered so as to hide or distort information about $\dot{\boldsymbol{e}}$ while still accurately reporting $\boldsymbol{e}$.

In short, I have identified two principles of sensorimotor learning: that it deduces the relation between neural commands and performance — the sensitivity derivative matrix — and that it represents this quantity in transmissible form, in neural firing rather than in synaptic weights. To accomplish these things I have described a mechanism, called implicit supervision, which is biologically plausible, fast, robust, and general: it applies to linear and nonlinear systems of any dimension or order, so long as they fulfill Equations 3.1–3.3, and it could be applied whenever the brain needs to learn a computation for which it has no supervisor.

Chapter 4

Learning course adjustments during arm movements with reversed sensitivity derivatives

Contents of this chapter have been published as **Abdelghani and Tweed.** Learning course adjustments during arm movements with reversed sensitivity derivatives. *BMC Neuroscience* 2010: 11:150. http://www.biomedcentral.com/1471-2202/11/150.

In Chapter 3 we learned that for a sensorimotor system to learn effectively, it needs to know how its error ***e*** (for instance, the vector from target to hand in a reach) depends on the vector of neural commands ***u*** sent to the muscles (e.g. the signals to biceps, triceps, brachioradialis etc.). Mathematically, what the system needs is the matrix $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, called the control jacobian (Callier and Desoer 1991) or the matrix of *sensitivity derivatives* (Astrom and Wittenmark 1995). But is $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ itself learned, or is it known innately?

Also in Chapter 3 I pointed out the importance, for this question, of the *signs* of the elements of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$: if your brain knew $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ innately, then over time its innate estimates would of course become inaccurate (owing to your growth, aging, injuries, and healing), but so long as the signs of the estimates were correct, you could usually maintain good performance. But if the signs of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ reversed (e.g. if you put on reversing goggles and tried to reach for things) then your

innate estimates would make you "learn" the wrong way, strengthening those components of $\boldsymbol{u}$ that should be weakened and vice versa. Given this kind of reversal, recovery is possible only for systems that can revise their estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. I argued that neural controllers can learn this kind of task, and I proposed a mechanism, called *implicit supervision*, by which the brain might deduce $\partial \boldsymbol{e}/\partial \boldsymbol{u}$.

There is empirical support for implicit supervision. It is the only theory that explains how neural controllers can deal with sign changes in $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, as happen with reversed vision or nerve transposition (Stratton 1897; Ewert 1930; Sperry 1943; Sperry 1945; Sperry 1947; Leffert and Meister 1976; Yumiya, Larsen et al. 1979; Brinkman, Porter et al. 1983; Scaramella 1996; Sugita 1996; Tate and Tollefson 2006). And it explains why it is harder to adapt to reversals than to other changes (von Helmholtz 1962; Harris 1965; Abdelghani, Lillicrap et al. 2008).

The next question is whether adjustable estimates of sensitivity derivatives govern all aspects of a task. For instance, when you learn to move to mirror-reversed targets, does your adapted estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ reverse both your initial aiming and your online course adjustments: when the target jumps in mid-movement, is your path adjustment appropriately reversed?

Data relevant to this issue have come from a novel experiment by Gritsenko and Kalaska (Gritsenko and Kalaska 2010). They trained people to reach to stationary (i.e. non-jumping) right-left-reversed targets. After the training was complete, they tested the subjects' responses when the mirror-reversed target jumped suddenly in mid-reach, and they found that in many cases the subjects' earliest course adjustments were *not* appropriately reversed, as they should have been if $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ had been learned. What are the implications of this fact for the theory of implicit supervision? Does it mean that the reach controller in the brain has multiple estimates of

$\partial \boldsymbol{e}/\partial \boldsymbol{u}$ — one perhaps concerned with launching a reach toward its target, and a different one concerned with course adjustments? Is this latter estimate incapable of adapting, or might it adapt given a different training regimen — the point of Gritsenko and Kalaska's study was to train on stationary targets and then test generalization to jumps, but what if subjects were trained on jumping targets? And finally, might Gritsenko and Kalaska's findings be compatible after all with a single, all-purpose estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ rather than separate ones for launch and adjustment?

Here I put subjects through many trials with jumping targets, and show that they can learn to reverse their rapid, online course adjustments; i.e. I show that these adjustments are governed by an adjustable estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. And I argue that all the available data are compatible with a single, adaptable, all-purpose estimate.

4.1 Method

This study complies with the Helsinki Declaration and was approved by the Ethics Review Office of the University of Toronto, reference number 16210. All subjects gave their informed consent.

4.1.1 Experimental task

Subjects bent a joystick to move a cursor toward a jumping target on a computer screen. They sat facing the screen at a distance of 80 cm, and used their dominant arm to manipulate an Impulse Stick — a USB force feedback joystick made by Immersion Inc. (San Jose, CA, USA) — through its full range of ±40º, or about ±6 cm. The joystick was placed to the subject's right or left side, its x-axis parallel with the screen.

On the screen were the cursor and target — the cursor was an X, sized 0.5 cm by 0.5 cm, and the target was two concentric circles, 1 and 0.3 cm across. Cursor location was related to joystick angle by a linear mapping, with no velocity dependence. Subjects were instructed to bring the cursor to the target. At the start of each trial the target appeared at a random location within an *initial target range*, 13 cm by 13 cm, centered within a larger *movement range*, 13.75 cm by 13.75 cm, which was the range of motion of the cursor (the initial target range was smaller so that the target could always jump away from its initial point of appearance in an unpredictable direction and still remain within the movement range) (Figure 4.1A). The target's new location was computed from its previous one, at the end of the preceding trial, by the formula $\boldsymbol{T}_{new} = \boldsymbol{T}_{prev} \pm$ random(5.5, 13) where random(5.5, 13) was a random vector, its two components chosen independently at the start of each trial from uniform distributions between 5.5 and 13; this random vector was added to $\boldsymbol{T}_{prev}$ on 50% of trials, and subtracted on the rest. Addition or subtraction was chosen randomly, except that if addition would have placed the target outside the initial target range then subtraction was used instead, and vice versa. In this way I kept the initial target inside its range but with plenty of variability and distance between successive appearances.

During each trial the target jumped once. Jump time was determined randomly, though based on cursor motion to help ensure that it occurred during the arm movement: the target jumped when $||\boldsymbol{e}||$, the size of the error vector $\boldsymbol{e}$ from target to cursor, first fell below a threshold value of random(0.25, 0.75)$||\boldsymbol{e}_0||$, where $\boldsymbol{e}_0$ was the initial error vector when the target appeared at the start of the trial, and random(0.25, 0.75) was a number chosen at the start of each trial from a uniform distribution between 0.25 and 0.75. The *size* of the target jump was 60% as large as the error at jump time, i.e. $0.6||\boldsymbol{e}||$, except when a jump of that magnitude would have carried the target outside the movement range, in which case the *x*- and/or *y*-components of the jump were truncated to stay in range. The jump was orthogonal to the vector from target to cursor at the

time of the jump (Figure 4.1B), except, again, when *x*- or *y*-components were truncated to stay in range. The *direction* of the jump, along this orthogonal line, was random: half the time in one direction, the rest in the other.

After the jump, subjects adjusted their motion to try to reach the target. If they managed to get the center of the cursor within 0.3 cm of the center of the target and hold it there for at least 100 ms within 2 s of the target's initial appearance then they were rewarded with a beep and a flash, i.e. the target changed momentarily from a pair of concentric blue circles to a filled-in red disk (Figure 4.1C). If they scored a beep, the next trial began immediately. If not, the next trial began 2 s after the initial appearance of the target. The initial cursor location, at the start of each new trial, was simply wherever the cursor happened to be at the end of the previous trial. Subjects saw the cursor and the target at all times throughout each trial, so they got plenty of feedback about their performance.

Subjects performed multiple blocks of 50 reaches. They had the option to rest as long as they liked between blocks. In control blocks, pushing the joystick forward moved the cursor up, and pushing right moved it rightward (Figure 4.1D). In test blocks, the relation between joystick and cursor was altered, in different ways in the two experiments described below, changing $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. On Day 1 subjects performed 20 blocks of 50 control reaches each, for 1000 reaches in all. On each of three or four subsequent days they did 20 blocks of reversed reaches, for a total of 3000 or 4000 reversed reaches. Finally, they did another 20 blocks of control trials. Through all these trials I sampled joystick position at 10-ms intervals.

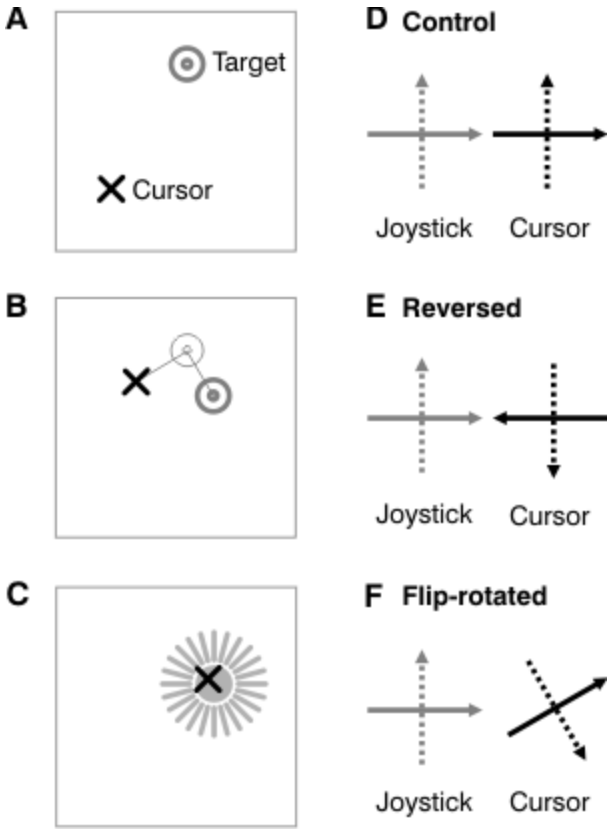

Figure 4.1: Subjects bent a joystick to move a cursor to a target. (A) In each trial, the target appeared at a random location. (B) The subject then moved the cursor, though often they didn't manage to move it exactly along a straight line to the target. During the cursor motion, the target jumped, at an unpredictable time, at right angles to the line from cursor to target. Jump size and direction were random. (C) If subjects got to the target and stayed there for 100 ms then they were rewarded with a flash and beep. (D) In control trials, pushing the joystick forward moved the cursor up, and pushing right moved it right (E) In Experiment 1, pushing the joystick forward moved the cursor down, and pushing right moved it left, i.e. cursor motion was reversed in both dimensions, flipping the signs of all components of $\partial \mathbf{e}/\partial \mathbf{u}$. (F) In Experiment 2, the relation between joystick and cursor was more complex: reflected vertically through the midline and rotated 30 degrees counterclockwise.

Experiment 1. Course adjustment with reversed sensitivity derivatives

In test blocks, both dimensions of cursor motion were reversed from control, flipping the signs of all components of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ (Figure 4.1E). Five subjects took part — one female, four males, all healthy, aged 21–48. Three of them knew the experiment involved a reversed relation between joystick and cursor. One of these three had experience with joystick experiments, and one with joystick computer games. All my single-person data plots (Figures 4.2, 4.3, and 4.4) are of subjects who were unfamiliar both with joysticks and with the idea of motor adaptation to reversals, but the key findings were the same for all subjects, as shown in Figure 4.5.

Experiment 2. Reversal and rotation

Here the relation between joystick and cursor was more complex: reflected vertically through the midline and rotated 30 degrees counterclockwise (Figure 4.1F). Five subjects took part — one female, four males, all healthy, aged 21–48. None of them knew the joystick-cursor relation beforehand. All found it bewildering, and none was able to state it afterwards based on their experience. Four of the subjects were veterans of Experiment 1, and therefore had more joystick experience in this second part, but that fact is irrelevant here because my hypothesis and analysis involved no comparisons of the two experiments. The single-person data plot (Figure 4.6) is of the new subject, without joystick experience, but the key results were the same for all, as shown in Figure 4.7.

4.2 Results

4.2.1 Control trials

In control trials, subjects' course adjustments usually go the right way. Figure 4.2A shows a typical movement: the cursor moves off in the direction of the initial target; after the target jumps, the cursor adjusts in the direction of the jump. The same pattern is seen in a plot of cursor velocity for the same movement, Figure 4.2B. In this figure and all other velocity-versus-time plots in the paper, I show different components of the cursor-velocity vector before and after the jump (marked by the vertical dashed gray line): before the jump I plot the component of cursor velocity in the direction from initial cursor location to initial target location; after the jump I plot the component in the direction of the jump. Graphed this way, positive velocity in the first stage of the plot indicates that the cursor has launched appropriately, in the direction of the target, and positive velocity in the second stage means the cursor has adjusted appropriately to the jump. In Figure 4.2B, both components are mainly positive.

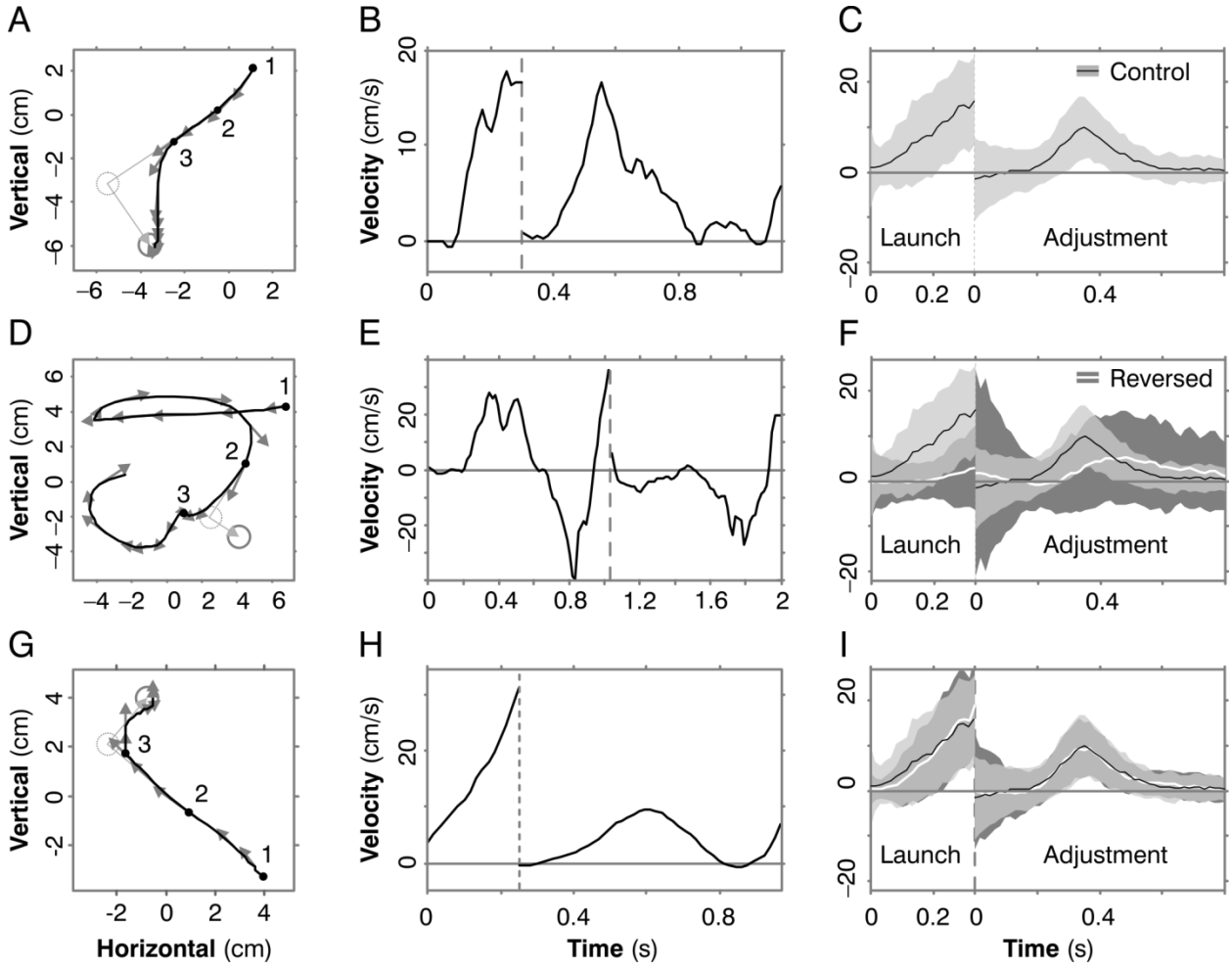
A
B
C
D
E
F
G
H
I
Vertical (cm)
Velocity (cm/s)
Horizontal (cm)
Time (s)
Control
Reversed
Launch
Adjustment
1
2
3

Figure 4.2: Sample trajectories for one subject. (A) A typical trajectory under control conditions (without reversed sensitivity derivatives). The cursor is at position 1 when the target appears (dotted gray circle). The cursor moves (black line) towards the target. Gray arrows are cursor-velocity vectors. When the cursor is at 2 the target jumps to its new location (solid gray circle). Near 3 the cursor adjusts course appropriately. (B) The same pattern is seen when I plot components of cursor velocity. Before the target jump (vertical dashed line), I plot cursor velocity in the direction from initial cursor location to initial target location; appropriately, this velocity is positive. After the jump, I plot cursor velocity in the direction of the jump, again appropriately positive. (C) The same pattern is seen in averaged velocity traces. Here we see cursor velocity — mean (black line) and standard deviation (light gray band) over 200 randomly chosen control trials. Velocity is mainly positive, as it should be, during both launch and adjustment. (D) In early reversed trials, launch and course adjustments go the wrong way. (E) These errors are revealed also in velocity traces. (F) The white line and dark gray band are the mean and SD of cursor velocity over the first 200 reversed trials. Launch and adjustment are impaired, i.e. they are not consistently positive, and those portions that are positive occur later than in the control trials. (G, H) In *late* reversed trials, after the subject has learned, launch and adjustment are both correct. (I) Velocities averaged over 200 randomly chosen late reversed trials resemble controls.

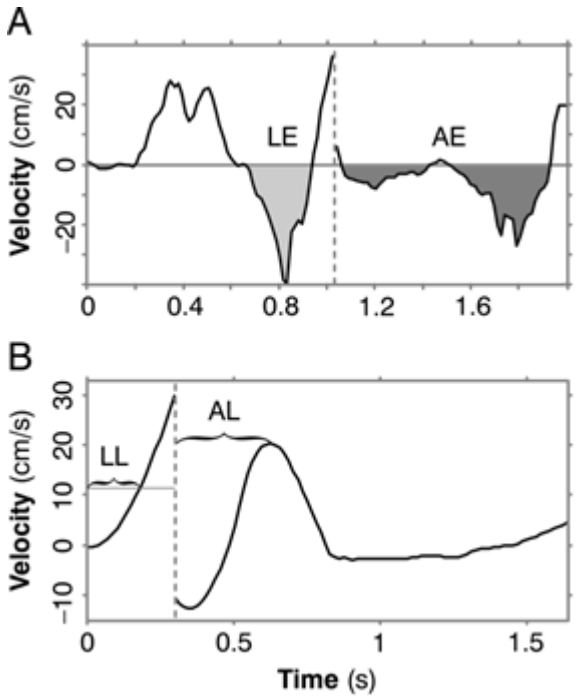

Figure 4.3: Performance measures. (A) Launch error (LE) is the area (colored light gray) above the negative parts of the cursor-velocity curve before the target jump. Adjustment error (AE) is the area (dark gray) above the negative parts of the curve after the jump. (B) Launch latency (LL) is the time from target appearance till cursor velocity exceeds the mean plus 3 times the standard deviations of initial velocity. Adjustment latency (AL) is the time from the jump till peak velocity.

Experiment 1: Course adjustments with reversed sensitivity derivatives

In *early* reversed trials, both launch and course adjustment go the wrong way, as shown in Figure 4.2D. The errors are revealed also in the velocity trace for the movement, Figure 4.2E: before the target jump, cursor velocity is not consistently positive, i.e. not in the direction of the target; after the jump, cursor velocity is mostly negative, i.e. opposite the jump. These plots also show that, even in early reversed trials, subjects don't move relentlessly in the wrong direction, but rather their trajectories are confused, with a tendency to go the wrong way and then try to correct. But after the subject has performed 3000 trials under reversed conditions, launch and adjustment are both appropriate, as shown in Figure 4.2G and 4.2H. This is the key result of my study: movement traces like these show that the subject, after training, could make online course adjustments with no wrong-way response, as predicted by the theory of implicit supervision.

To show that this behavior was consistent, I averaged cursor velocities over many trials. Figures 4.2F and 4.2I show mean velocity and its standard deviation for one subject, with control data superimposed on the data for trials with reversed sensitivity derivatives. In control traces (Figure 4.2C), cursor velocity is appropriately positive in both the launch and adjustment stages of the movement. In early reversed trials (Figure 4.2F), velocities are not consistently positive during launch or adjustment. In *late* reversed trials (Figure 4.2I), velocities are again appropriate, and resemble controls as regards direction, size, timing, and variance. This same result was seen in all five subjects.

To chart these improvements through time, I plotted four performance measures. I quantified wrong-way responses by integrating negative velocity, as shown in Figure 4.3A. I defined the *launch error* (LE), for any one trial, to be the area above the negative parts of the cursor-velocity curve before the target jump (light gray region), i.e. launch error is the distance traveled away

from the target over this period. *Adjustment error* (AE) is the area above the negative parts of the velocity curve *after* the jump (dark gray region), i.e. it is the distance traveled opposite the jump. To quantify timing, I defined *launch latency* (LL) to be the interval from the target's initial appearance until positive cursor velocity (the component in the direction from initial cursor position to initial target) exceeded a threshold value v_{th}, where v_{th} was equal to the mean plus three times the standard deviation of the subject's cursor velocity at target appearance, averaged across all control trials for that subject. *Adjustment latency* (AL) is the time it takes from the target jump until cursor velocity (the component in the jump direction) reaches its first maximum at least 150 ms after the jump (Figure 4.3B). So adjustment latency indicates how quickly a given subject on a given trial produced an appropriate course adjustment. It overestimates reaction time, because (for robustness) it is based on the peak, rather than the onset, of the right-way velocity. But it provides a consistent measure of the time frame of my subjects' course adjustments.

Figure 4.4 shows moving-window plots for all four measures — launch error, adjustment error, launch latency, and adjustment latency — for one subject. Each of the curves shows one performance measure, averaged over a 50-trial moving window, through one session of 1000 trials. In each panel, the jagged gray line shows the error or latency improving over the first 1000 reversed trials; the light gray band shows the mean and one standard deviation for the same error or latency over the 1000 initial control trials; and the jagged black line shows performance in the final 1000 reversed trials. In all four panels, performance over the last 1000 trials hovers near the control range. Again, the plots were very similar for all five subjects.

Figure 4.5 summarizes the results for these four measures for all subjects. Means and standard errors are plotted for the 1000 control trials on the ordinates and for reversed trials on the abscissas, with early and late trials in different colors: the five gray symbols — one for each subject — mark the means and standard errors over the first 200 reversed trials; black symbols show the data for the final 1000 reversed trials. Slanted gray lines indicate where values for control and reversed performance are equal. In all four plots, all five subjects improved: the black symbols lie to the left of the gray ones, and significantly at $p < 0.05$ by t-test and sign test (a non-parametric test for paired samples, (Mendenhall, Wackerly et al. 2008)).

Further, in all four plots there was *no* significant difference, at the same p level, between control and late reversed trials, i.e. subjects returned to something like control performance. This finding is interesting but peripheral to my purposes, because the theory of implicit supervision doesn't imply anything about whether post-adaptation performance will be identical with controls. The point of Figure 4.5 is that all subjects improved, driving down their adjustment errors without appreciably slowing their responses.

Three subjects knew the experiment involved a reversed relation between joystick and cursor. The other two subjects never recognized the relation, i.e. they couldn't state it in words when questioned after the experiment was over. None of the five subjects felt, introspectively, that it helped to try to work out the relation of cursor to joystick, or to imagine the target in some reversed location on the computer screen, or to reverse their hand motion deliberately. What worked was simply to chase the target with the cursor, giving no thought to hand motion, improving gradually and automatically.

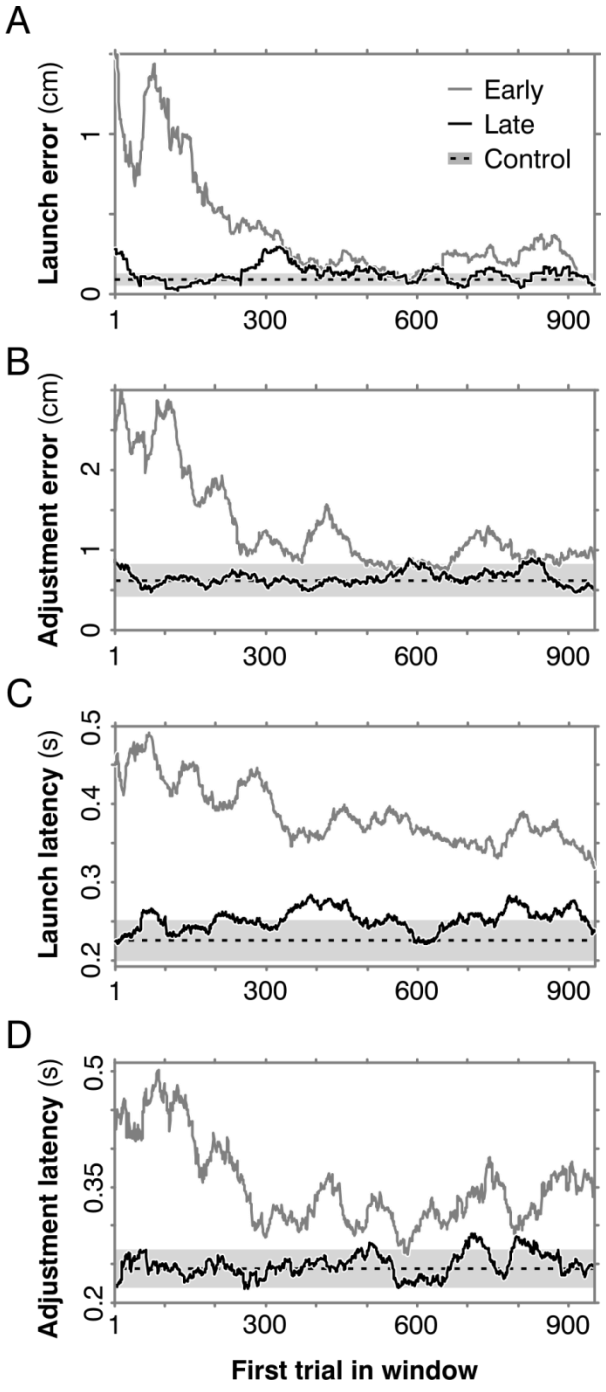
A
Launch error (cm)
0
1
Early
Late
Control
1
300
600
900
B
Adjustment error (cm)
0
1
2
1
300
600
900
C
Launch latency (s)
0.2
0.3
0.4
0.5
1
300
600
900
D
Adjustment latency (s)
0.2
0.35
0.5
1
300
600
900
First trial in window

Figure 4.4: Moving-window averages. (A) To study changes through time I plot averages over a moving window of 50 trials, e.g. in this panel, the height of the curve above the 300 tick mark is the average launch error for trials 300–349; the rightmost point in the curve is the average for trials 951–1000. The plot shows that as the subject learns, launch error falls to control levels. The gray curve shows launch errors improving over this subject's first 1000 reversed trials. Black shows the final 1000 reversed trials. The light gray band is the mean ± 1 SD of the averaged launch error across all 950 windows in the initial control session. (B) Adjustment error also falls to control levels, as do (C) launch latency and (D) adjustment latency.

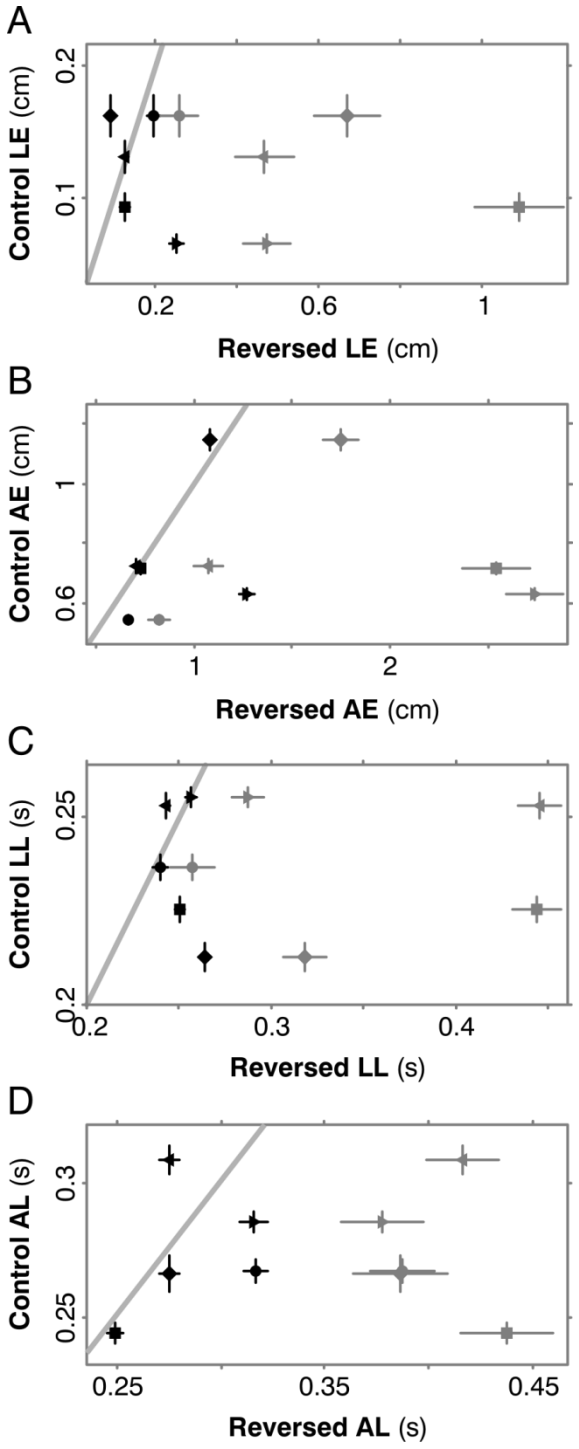
A
Control LE (cm)
0.2
0.1
0.2
0.6
1
Reversed LE (cm)
B
Control AE (cm)
1
0.6
1
2
Reversed AE (cm)
C
Control LL (s)
0.25
0.2
0.2
0.3
0.4
Reversed LL (s)
D
Control AL (s)
0.3
0.25
0.25
0.35
0.45
Reversed AL (s)

Figure 4.5: All 5 subjects' learning. (A) On the ordinate are means and standard errors of launch error (LE) for the 1000 initial control trials. On the abscissa are the same measures for first 200 reversed trials (5 gray symbols for the 5 subjects) and for the last 1000 reversed trials (black symbols). Across all subjects, launch error improved significantly by t-test ($p = 0.025$) and in late reversed trials was not significantly different from control ($p = 0.35$). (B) Adjustment (AE) error also improved ($p = 0.025$) to control levels ($p = 0.47$). (C, D) Similarly for launch latency (LL) ($p = 0.035$ and 0.26) and adjustment latency (AL) ($p = 0.004$ and 0.43).

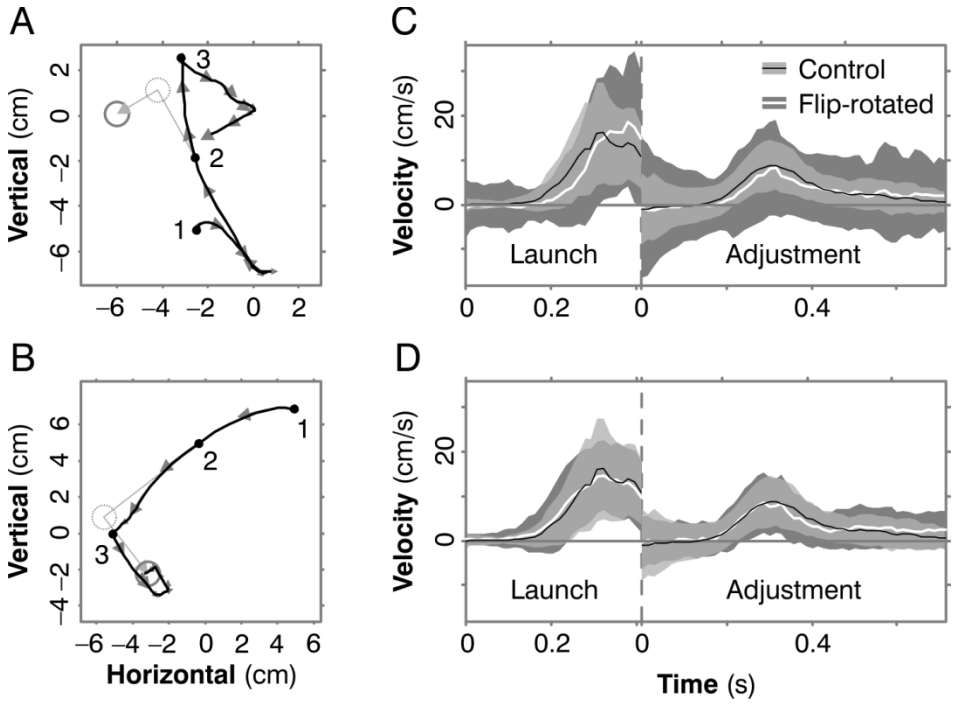

Figure 4.6: Trajectories for one subject in Experiment 2. Symbols are as in Figure 4.2. (A) In early flip-rotated trials, launch and adjustment go the wrong way. (B) In late flip-rotated trials, launch and adjustment are correct. (C) Averaged velocities in early flip-rotated trials are less consistently positive than in controls. (D) In late flip-rotated trials, they resemble controls.

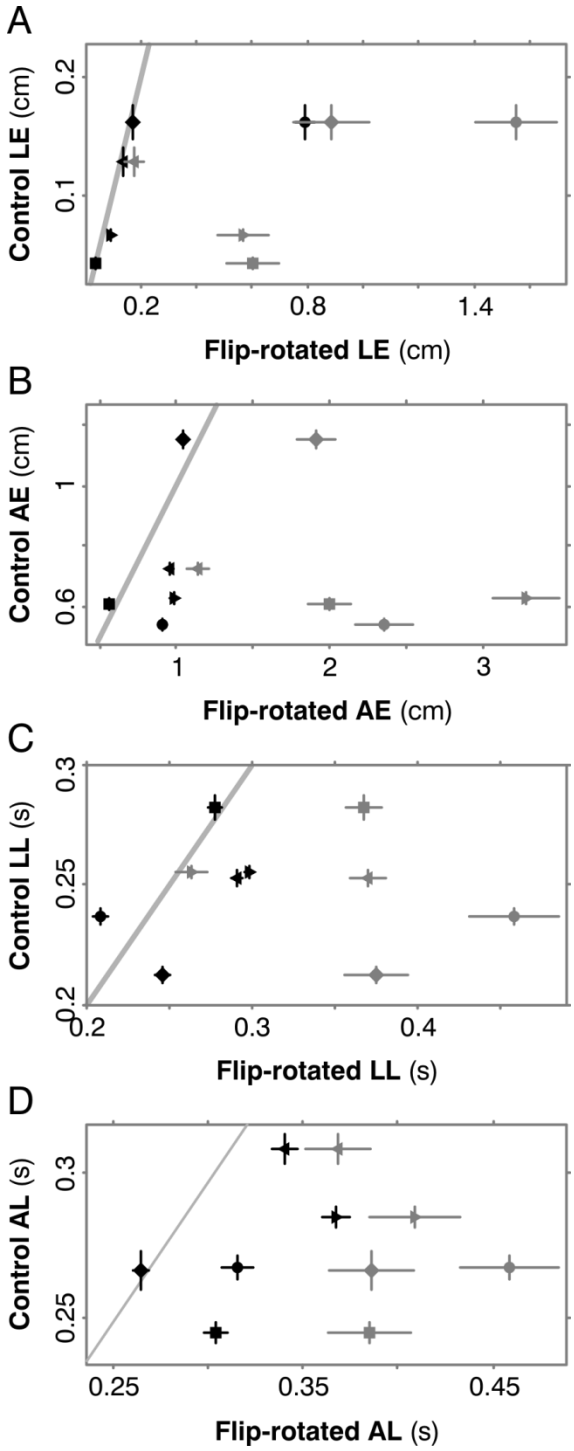
A
Control LE (cm)
0.1
0.2
0.2
0.8
1.4
Flip-rotated LE (cm)
B
Control AE (cm)
0.6
1
1
2
3
Flip-rotated AE (cm)
C
Control LL (s)
0.2
0.25
0.3
0.2
0.3
0.4
Flip-rotated LL (s)
D
Control AL (s)
0.25
0.3
0.25
0.35
0.45
Flip-rotated AL (s)

Figure 4.7: Results for all subjects in Experiment 2. (A) Launch error improved by t-test ($p = 0.008$) to levels not significantly different from control ($p = 0.339$). (B) Adjustment error improved ($p = 0.012$) to control levels ($p = 0.195$). (C) Launch latency improved ($p = 0.004$) to control levels ($p = 0.431$). (D) Adjustment latency improved ($p = 0.01$), not quite to control levels ($p = 0.036$) but to within 80 ms of control.

Experiment 2: Reversal and rotation

In this harder task, the results were like those in Experiment 1 except that learning was slower. Plots of individual movements showed that in early flip-rotated trials, launch and adjustment were both inappropriate (Figure 4.6A), but in late trials both had adapted (Figure 4.6B). Averaged velocity traces showed that this pattern was consistent (Figure 4.6 C, D). Across subjects, all four performance measures improved significantly with training, to near-control levels (Figure 4.7). None of the subjects, even after learning the task, was able to state in words the relation between joystick and cursor.

4.3 Discussion

The theory of implicit supervision holds that the brain's estimates of sensitivity derivatives, $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, can be revised based on sensory feedback (Stratton 1897; Ewert 1930; Sugita 1996). This theory explains how neural controllers can deal with sign changes in $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. For instance, humans and monkeys can learn to handle objects and navigate while wearing reversing prisms (von Helmholtz 1962; Harris 1965; Abdelghani, Lillicrap et al. 2008). People can learn to mirror-draw, and dentists can drill teeth seen in a mirror. When antagonist muscles or nerves are transposed, animals can sometimes regain their coordination (Sperry 1943; Sperry 1945; Sperry 1947; Missiuro and Kozlowski 1963; Leffert and Meister 1976; Yumiya, Larsen et al. 1979; Brinkman, Porter et al. 1983). And facial-palsy patients treated by hypoglossal nerve transposition learn to control face and tongue independently (Vera, Lewin et al. 1975; Scaramella 1996; Tate and Tollefson 2006). The theory also explains why it is harder to adapt to reversals than to other changes: displacing, magnifying, and minifying goggles don't flip the

signs of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, so we can adapt to them without revising our estimates of the sensitivity derivatives; reversing prisms, on the other hand, do flip the signs, so we can't adapt without re-estimating $\partial \boldsymbol{e}/\partial \boldsymbol{u}$.

Here I have confirmed another prediction of the theory: in arm-movement tasks with reversed and rotated sensitivity derivatives, my subjects learned to make appropriate course adjustments when the target jumped. After training, individual movements often showed no wrong-way response (naturally some movements did show mistakes, during launch or adjustment, but similar mistakes were seen also in control trials). Averaged velocity traces in the reversed task after training resembled control traces as regards direction, size, timing, and variance. For all subjects, wrong-way responses shrank (as quantified by adjustment errors), to near control levels. So the neural estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ that is used for course adjustments is clearly revisable.

How do my results fit with those of Gritsenko and Kalaska (Gritsenko and Kalaska 2010)? For my purpose — testing implicit supervision — what is important about that study is the discrepancy between launch and adjustment: some of the subjects who learned to launch toward the target still made course adjustments in the wrong direction. That finding raised a question for my theory: if the subjects improved their launches by re-estimating their sensitivity derivatives then why didn't the revised estimate correct their adjustments?

One possible explanation is that there are two (or more) separate estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ for different aspects of a task, e.g. for launch and adjustment. In this view, a subject might reverse their launch-related estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ but not their adjustment-related estimates, maybe because the latter change more slowly, or because the training included no practice adjusting to target jumps. (A different issue is whether launch and adjustment involve separate controllers, e.g. one using

feedback and the other not. This is a separate question because even entirely disjoint launchers and adjusters, whether feedback-guided or not, could still be governed by a single estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. My concern here is with $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, not with other possible contrasts between launching and adjusting.)

But a simpler explanation is that there is one all-purpose estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, and Gritsenko and Kalaska's subjects revised it over only part of its domain. The key point is that $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is not a constant matrix but varies over a domain D. For instance we might have $D = X \times X^* \times U$, where X is the state space of the plant (e.g. the space of all possible combinations of arm joint angles and velocities), X^* is the space of target states, U is the space of motor commands, and $\times$ is the Cartesian product. When a target jumps during an arm movement, it suddenly transports the subject to a new region of D. (In Gritsenko and Kalaska's experiment, subjects may not have been transported very *far* through D, as the target jumped only 10º, measured from the starting point of the movement. But by the time they reacted, their angular errors would be larger than 10º. And even if the new region of D were close to the old, the appropriate motor command might be quite different there, as the subjects would need lateral acceleration in situations where the target had jumped.) Gritsenko and Kalaska's study was designed to train people with no jumps and then test their generalization to jumping targets, so their subjects had little experience with the post-jump regions of D, and therefore, I suggest, didn't completely revise their estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ there; some learning may have generalized from nearby regions, but not enough to abolish their inappropriate, unreversed responses. My study was designed to give subjects plenty of experience with jumps during their training, and so they learned $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ over the relevant parts of D.

This idea doesn't imply that there are "boundaries" within D, or that different regions of it are linked with different learning mechanisms or controllers. The point is simply that a learner trained in one domain usually does poorly in others, e.g. a neural network trained to approximate the function x^2 over the domain [0, 0.1] does poorly when tested over a different region, say [0.1, 0.2]. And the failure is worse, the more the target function differs between the two regions. Similarly, implicit supervision trained exclusively on one subset of D — the subset inhabited by reaches to fixed targets — yields poor estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ elsewhere.

The four types of learning curves in Figure 4.4 — launch error, adjustment error, launch latency and adjustment latency — decline with roughly similar time courses. Unfortunately their shapes offer no clues as to how many estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ are being adapted. The similarity between the four curves need not imply a single estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ underlying them all; it is also compatible with multiple estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ if those estimates learn in similar ways. And conversely, even markedly dissimilar curves would be compatible with a single estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ because the four curves reflect different aspects of the task, occurring in different regions of the domain D. They are expected to differ, even if they all depend on the same estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. In simulations, the correlations and other similarities between these curves vary enormously depending on assumptions about learning algorithms, neural coding, and noise throughout the control system, i.e. both single and multiple estimates are compatible with a wide variety of curves.

In both my experiments, subjects' responses were often delayed, e.g. in Figures 4.5 and 4.7, LL and AL were always greater in early reversed trails than in control trials, and often stayed greater for thousands of trials, though eventually they improved to roughly control values. Evidently subjects slowed some aspects of their movements in unfamiliar conditions, maybe to permit more voluntary control.

Voluntary reversals have been studied by Day and Lyon (Day and Lyon 2000). Their subjects reached straight ahead for a target which jumped right or left in mid-reach. The subjects were told to react to the jump by moving in the opposite direction, but even after several hundred trials, their first reaction was still in the jump direction, followed by a reversed response. What does this mean for implicit supervision? There are many possibilities, e.g. 1) Day and Lyon's results may have nothing to do with changes in $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. Their study involved no sensory reversal, so there was no change in the relation between any sensory error signal $\boldsymbol{e}$ and motor commands; rather there was a verbal instruction to reverse. Subjects may simply have tried to aim for an imaginary target opposite the real one, in keeping with their instructions. 2) Subjects may have created a new, mental error signal $\boldsymbol{e}'$ equal to –1 times the visual error $\boldsymbol{e}$, and then learned $\partial \boldsymbol{e}'/\partial \boldsymbol{u}$. They may have had two separate representations of $\partial \boldsymbol{e}'/\partial \boldsymbol{u}$ for early and late responses to jumps. Or their early and late responses may have been guided by $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ and $\partial \boldsymbol{e}'/\partial \boldsymbol{u}$ respectively. 3) Subjects may have had one representation of $\partial \boldsymbol{e}'/\partial \boldsymbol{u}$ for reflexive control generally and another for higher-level control, i.e. separate representations for different levels of control rather than for different stages of a movement.

There may be hints of multilevel control in my results as well, e.g. in Figure 4.1D, an early reversed trial, the subject launches in an inappropriate direction but then later, something makes them reverse course with a tight U-turn (though the new direction is also inappropriate). If there is a high-level controller that steps in here, it may have a separate estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, better than the reflexive controller's, but this scheme would be inefficient: learning $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is computationally costly, so there are good reasons to do it just once. Another possibility is that the high-level controller has no good estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, but adopts some simple, exploratory strategy, e.g. it thinks "my estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is clearly inaccurate, and my most recent action was

counterproductive, so I'll try undoing it or doing something else different". Or maybe high-level controllers can rapidly estimate the current value of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, i.e. they don't learn the *function* $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ but just estimate its *value at the current spot* in its domain D, which is easier. This approach would bring advantages if used to supplement (not replace) learning the function $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ — see Fortney and Tweed (Fortney and Tweed 2006).

Where in the brain might $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ be represented? One possibility is the cerebellum, which is involved in sensorimotor learning and internal models (Kawato 1999). These models are neural circuits that mimic aspects of the system to be controlled, such as the mechanical properties of an eyeball or limb, and especially the relation between neural commands and motor performance. In particular, so-called forward models mimic the response of the controlled system to neural commands (Kawato 1999). Therefore an estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ is a kind of forward model, representing the relation between performance error $\boldsymbol{e}$ and command $\boldsymbol{u}$.

In summary, I have shown that people can learn to reverse their online course adjustments, implying that these adjustments are based on revisable estimates of sensitivity derivatives, as predicted by the theory of implicit supervision. And I have argued that the available data are consistent with the simplest version of the theory, that a single, contextual estimate of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ guides motor learning for all stages of a task, including launch and adjustment.

Chapter 5

Overall conclusions

In this thesis I have pointed out the crucial role played in motor learning by the variables known as sensitivity derivatives, which express how motor performance depends on commands from neural controllers. And I have emphasized in particular the importance of the *signs* of these derivatives: an adaptive control system that can't learn sensitivity derivatives but instead relies on innate estimates of them can still recover from changes to the plant so long as those changes don't reverse the signs of the derivatives. If the signs are reversed then recovery is possible only for systems that can relearn $\partial \boldsymbol{e}/\partial \boldsymbol{u}$.

I have pointed out that previous theories of motor control, while they didn't mention sensitivity derivatives by name, did assume implicitly that they (or at least their signs) are known by the brain. But with one exception, they didn't explain how that knowledge might be acquired. The only theory that offered a mechanism was Jordan and Rumelhart's, with its "distal teacher", but it relied on a form of communication known as weight transport which is not available in the brain. With no biologically feasible learning mechanism, all other theories assumed implicitly that the derivatives aren't learned at all but are known innately.

In Chapter 3 I showed that this knowledge can't be *solely* innate. At least some sensorimotor systems must learn their sensitivity derivatives, the main evidence being that they can recover when those derivatives change sign. Further support for this idea came from the theoretical observations that fast motor learning and tool use benefit from detailed knowledge of sensitivity

derivatives. I proposed three mechanisms by which sensitivity derivatives could be learned using only forms of communication that are known to exist in the brain, and I argued particularly for the algorithm I have called error-based implicit supervision.

Implicit supervision explains, for the first time, how animals can adapt to conditions that reverse the signs of sensitivity derivatives, and it explains why it is harder to recover from reversals than from other alterations such as magnifying, minifying or displacing goggles.

As I described in Chapter 3, the theory of implicit supervision makes a number of predictions, and in Chapter 4 I tested one of them: that updated estimates of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ should improve reflexive, online course adjustments as well as initial aiming. I had healthy human subjects manipulate a joystick to move a cursor to a target on a computer screen, and I made the target jump once in the middle of the cursor movement so subjects would have to alter course. As predicted by the theory of implicit supervision, subjects did learn to make appropriate, reversed course adjustments when the target jumped.

An open question is whether there are alternatives to, or variants of, implicit supervision that might have advantages. One specific motivation is that implicit supervision computes time-derivatives of model error and the vector $\boldsymbol{z}$, containing information about context and commands. This reliance on derivatives can impair the performance of the algorithm in the presence of noise because noise is amplified by differentiators. To cope with his problem I have devised a variant of implicit supervision that replaces the error-differentiator with a highpass filter in a feedback loop, and it performs very well in the presence of noise. But probably there are other options using other methods of robust differentiation, such as prefiltering, filtered error or regressor, Savitsky-Golay and Kalman filtering (Savitzky & Golay 1964; Bryson & Ho 1975; Åström & Wittenmark 1989; Farrell & Polycarpou 2006; Tripp & Eliasmith 2006).

Appendices

Appendix A: Generality of implicit supervision

The framework defined in Equations 2.1–2.3 is very general, though some sensorimotor control systems fit in only after they are recast into a form in which relative degree is zero. I can explain the idea of relative degree using the example of saccadic eye movements. Here, as in the VOR, the plant equation is

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{u} - \kappa \boldsymbol{x}}{\rho}, \tag{A.1}$$

where $\boldsymbol{x}$ is eye position. The aim is to bring the eye to some target angle $\boldsymbol{x}^*$. So a reasonable definition of the error might be

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}^*. \tag{A.2}$$

However, $\boldsymbol{e}$ so defined is not a function of $\boldsymbol{u}$, because neither $\boldsymbol{x}$ nor $\boldsymbol{x}^*$ is a function of $\boldsymbol{u}$. But $\dot{\boldsymbol{x}}$ is, as Equation A.1 shows. And $\boldsymbol{x}$ is the time-integral of $\dot{\boldsymbol{x}}$, so in this sense there is one integration standing between $\boldsymbol{u}$ and $\boldsymbol{x}$, and therefore between $\boldsymbol{u}$ and $\boldsymbol{e}$. We say this control system is of *relative degree* 1. If instead it were the acceleration $\ddot{\boldsymbol{e}}$ that was a function of $\boldsymbol{u}$, then the system would be of relative degree 2, and so on. By the same reasoning, the VOR is of relative degree zero. Only systems of relative degree zero make $\boldsymbol{e}$ a function of $\boldsymbol{u}$, and therefore only such systems fulfill Equation 2.2. But most sensorimotor control tasks are convertible to a form where relative degree is zero.

To see how this conversion can work, and can be useful, notice that an error signal like the one in Equation A.2 wouldn't give an adaptive controller much moment-to-moment guidance — it wouldn't tell it how to adjust its control law, precisely because the equation for ***e*** doesn't involve ***u***. For saccade adaptation, a more useful error might be something like

$$\boldsymbol{e} = \dot{\boldsymbol{x}} + (\boldsymbol{x} - \boldsymbol{x}^*) = \frac{\boldsymbol{u} - \kappa \boldsymbol{x}}{\rho} + (\boldsymbol{x} - \boldsymbol{x}^*) \tag{A.3}$$

With this ***e***, the system is now of relative degree zero. And if the controller adjusts its commands so as to zero ***e*** then saccades will obey the equation

$$\dot{\boldsymbol{x}} = \boldsymbol{x}^* - \boldsymbol{x} \tag{A.4}$$

which will carry the eye smoothly to its target angle. In reality, the equation relating saccadic eye velocity to eye-position error $\boldsymbol{x}^* - \boldsymbol{x}$ is nonlinear, so a more realistic error might be something like $\boldsymbol{e} = \dot{\boldsymbol{x}} + \varphi(\boldsymbol{x} - \boldsymbol{x}^*)$, where φ is a sigmoid nonlinearity, but these details are outside the scope of this thesis. My point here is simply that a higher-degree control problem can be converted to relative degree zero.

Appendix B: Two-joint arm

In Figures 3.6B through 3.9 I use a 2-joint arm to show that implicit supervision still works when the plant equation is nonlinear, the state and command are vectors rather than scalars, the order of the dynamics exceeds 1, and (in Figure 3.8) the system is kinematically redundant. This arm "has all the nonlinear effects common to general robotic manipulators" (Lewis, Jagannathan & Yeşildirek 1999). It is simpler than a real arm, most notably in that its motor command ***u*** is just

2-dimensional and determines joint torques directly, rather than via the nonlinear dynamics of muscles, but its mechanics are accurate, reflecting the fact that the torque at each joint affects the motion also at the other joint. Its plant equation is

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{5}{3} + c_2 & \frac{1}{3} + \frac{c_2}{2} \\ \frac{1}{3} + \frac{c_2}{2} & \frac{1}{3} \end{bmatrix}^{-1} \left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} (1 - s_2 \dot{x}_2) & -s_2 (\dot{x}_1 + \dot{x}_2) \\ s_2 \dot{x}_1 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \qquad \text{(B.1)}$$

where x_1 and x_2 are shoulder and elbow angles and c_2 and s_2 are the cosine and sine of the elbow angle. To simulate muscle transposition in Figure 3.6B, I reverse the polarities of both joint torques, *i.e.* in the plant equation I replace $\boldsymbol{u}$ by $-\boldsymbol{u}$. Performance error is $\boldsymbol{e} = \ddot{\boldsymbol{p}} + 2\dot{\boldsymbol{p}} + \boldsymbol{p}$ where $\boldsymbol{p}$ is the position error, $\boldsymbol{x} - \boldsymbol{x}^*$, and $\boldsymbol{x}^*$ is the vector of target joint angles. The context and $\boldsymbol{z}$ vectors are $\boldsymbol{v} = (\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{x}^*)$ and $\boldsymbol{z} = (\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{x}^*, \boldsymbol{u})$, and $\varphi(\boldsymbol{z})$ is computed as described above Equation 3.14, with fixed weights w_f drawn from a uniform distribution of mean 0 and standard deviation 0.125 (1 divided by the number of elements in $\boldsymbol{z}$). This plant model can work together with several kinds of controllers. The one in Figures 3.6B through 3.9 computes its command as a linear function of its own feature vector $\varphi^{\boldsymbol{u}}$ and adjusts its weight matrix $\boldsymbol{W}^{\boldsymbol{u}}$ by NLMS,

$$\Delta w^{\boldsymbol{u}}{}_{ij} = -\eta \frac{L_{\boldsymbol{u}i} \varphi^{\boldsymbol{u}}{}_j L}{\left\| L_{\boldsymbol{u}} \right\|^2 \left\| \varphi^{\boldsymbol{u}} \right\|^2} \qquad \text{(B.2)}$$

where $L_{\boldsymbol{u}i}$ is the i'th component of an estimate of $\partial L / \partial \boldsymbol{u}$ computed from the plant model's $\langle \partial \boldsymbol{e} / \partial \boldsymbol{u} \rangle$.

References

Abdelghani MN, Lillicrap TP, Tweed DB (2008) Sensitivity derivatives for flexible sensori-motor learning. Neural Comput. 20:2085-2111

Abel LA, Schmidt D, Dell'Osso LF, Daroff RB (1978) Saccadic system plasticity in humans. Ann. Neurol. 4:313-318

Ackley D, Hinton GE, Sejnowski T (1985) A learning algorithm for Boltzmann machines. Cog. Sci. 9:147-169

Asbury CL (2005) Kinesins: world's tiniest biped. Curr. Opin. Cell Biol. 17(1):89-97

Åström KJ, Wittenmark B (1995) Adaptive Control. Addison-Wesley, USA

Bellman RE (1961) Adaptive control processes: a guided tour. Princeton University Press

Bengio Y, Bengio S, Cloutier J (1991) Learning a synaptic learning rule. International Joint Conference on Neural Networks, Seattle, USA

Bloom GS (1992). Motor proteins for cytoplasmic microtubules. Curr. Opin. Cell Biol. 4:66-73

Brimijon S, Helland L (1976) Rapid retrograde transport of dopamine-beta-hydroxylase as examined by the stopflow technique. Brain Res. 102:217-228

Brimijon S, Wiermaa MJ (1978) Rapid orthograde and retrograde axonal transport of acetylcholinesterase as characterized by the stop-flow technique. J. Physiol. 285:129-142

Brinkman C, Porter R, Norman J (1983) Plasticity of motor behaviour in monkeys with crossed forelimb nerves. Science 220:438-440

Bryson AE, Ho YC (1975) Applied Optimal Control: Optimization, Estimation, and Control. Taylor & Francis

Callier FM, Desoer CA (1991) Linear System Theory. Springer, New York, USA

Chen C, Thompson RF (1995) Temporal specificity of long-term depression in parallel fiber-Purkinje synapses in rat cerebellar slice. Learn. Mem. 2:185-198

Crick FHC (1989) Recent excitement about neural networks. Nature 337:129-132

Day BL, Lyon IN (2000) Voluntary modification of automatic arm movements evoked by motion of a visual target. Experimental Brain Research 130(2):159-168

Dean P, Porrill J, Stone JV (2002) Decorrelation control by the cerebellum achieves oculomotor plant compensation in simulated vestibulo-ocular reflex. Proc. R. Soc. B 269:1895-1904

Ewert P (1930) A study of the effect of inverted retinal stimulation upon spatially coordinated behaviour. Genet. Psychol. Monogr. 7:177-363

Farrell JA, Polycarpou MM (2006). Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches. Wiley-Interscience

Forssberg H, Svatengren G (1983) Hardwired locomotor network in cat revealed by a retained motor pattern for gastrocnemius after muscle transposition. Neurosci. Lett. 41:283-288

Fortney KP, Tweed DB (2006) Learning without synaptic change: a mechanism for sensorimotor control. Society for Neuroscience, Atlanta

Frizell M, Sjostrand J (1974) Retrograde axonal transport of rapidly migrating proteins in the vagus and hypoglossal nerves of the rabbit. J. Neurothem. 23:651-658

Gardner WA (1984) Learning characteristics of stochastic-gradient-descent algorithms: a general study, analysis, and critique. Signal Processing 6:113-133

Gauthier GM, Robinson DA (1975) Adaptation of the human vestibuloocular reflex to magnifying lenses. Brain Research 92:331-335

Gonshor A, Jones GM (1976) Extreme vestibulo-ocular adaptation induced by prolonged optical reversal of vision. Journal of Physiology 256:381-414

Gritsenko V, Kalaska J (2010) Rapid online correction is selectively suppressed during movement with a visuomotor transformation. Journal of Neurophysiology

Grossberg S (1987) Competitive learning: From interactive activation to adaptive resonance. Cognitive Science 11:23-63

Halperin JJ, Lavail JH (1975) A study of the dynamics of retrograde transport and accumulation of horseradish peroxidase in injured neurons. Brain Res. 100:253-269

Harris CS (1965) Perceptual adaptation to inverted, reversed, and displaced vision. Psychological Review 72:419-444

Harrison RE, Baker JF, Isu N, Wickland CR, Peterson BW (1986) Dynamics of adaptive changes in vestibulo-ocular reflex direction. I. Rotations in the horizontal plane. Brain Research 371:162-165

Haykin S (2002) Adaptive Filter Theory. Prentice Hall, NJ, USA

Hinton GE (1989) Connectionist learning procedures. Artificial Intelligence 40:185-234

Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput. 18:1527-1554

Hirokawa N, Pfister KK, et al. (1989) Submolecular domains of bovine brain kinesin identified by electron microscopy and monoclonal antibody decoration. Cell 56:867-878

Hornik K, White H, et al. (1989) Multilayer feedforward netrworks are universal approximators. Neural Networks 2:59-66

Iqbal Z, Ochs S (1978) Fast axoplasmic transport of a calcium-binding protein in mammalian nerve. J. Neurochem. 31:409-418

Jordan MI, DE Rumelhart (1992) Forward models: Supervised learning with a distal teacher. Cognitive Science 16:307-354

Kawato M (1990) Feedback-error learning neural network for supervised motor learning. Elsevier Science Publishers, B.V. North-Holland

Kawato M (1999) Internal models for motor control and trajectory planning. Current Opinion in Neurobiology 9:718-727

Kawato M, Furukawa K, et al. (1987) A Hierarchial Neural-Network Model for Control and Learning of Volunary Movement. Biol. Cybern. 57:169-185

Kawato M, Gomi H (1992) The cerebellum and VOR/OKR learning models. TINS 15:445-453

Knull HR , Wells WW (1975) Axonal transport of cations in the chick optic system. Brain Res. 100:21-124

Koch C (1999) Biophysics of Computation. Oxford University Press, New York, USA

Kolen JF, Pollack JB (1994) Back-propagation without weight transport. Neural Networks, IEEE World Conference on Computational Intelligence

Kommerell G, Olivier D, Theopold H (1976) Adaptive programming of phasic and tonic components in saccadic eye movements. Investigations in 20 patients with abducens palsy. Invest. Ophthalmol. 15:657-660

Kreutzberg GW, Toth L (1974) Dendritic secretion: a way for the neuron to communicate with the vasculature. Naturwissenschaften 61:37

Kristensson K (1977) Retrograde axonal transport of horseradish peroxidase. Uptake at mouse neuromuscular junctions following systemic injection. Acta Neuropathol. 38:143-147

Leffert RD, Meister M (1976) Patterns of neuromuscular activity following tendon transfer in the upper limb: a preliminary study. J. Hand Surg. 1:181-189

Levine DS (2000) Introduction to neural and cognitive modeling. Lawrence Erlbaum Associates

Lewis FL, Jagannathan S, Yeşildirek A (1999) Neural Network Control of Robot Manipulators and Nonlinear Systems. Taylor and Francis, London, UK

Lisberger SG, Sejnowski TJ (1992) Motor learning in a recurrent network model based on the vestibulo-ocular reflex. Nature 360:159-161

Martin TA, Keating JG, Goodkin HP, Bastian AJ, and Thach WT (1996) Throwing while looking through prisms. I. Focal olivocerebellar lesions impair adaptation. Brain 119:1183-1198

Martin TA, Keating JG, Goodkin HP, Bastian AJ, and Thach WT (1996) Throwing while looking through prisms. II. Specificity and storage of multiple gaze-throw calibrations. Brain 119: 1199-1211

Martin TA, Norris SA, Greger BE, Thach WT (2002) Dynamic coordination of body parts during prism adaptation. J Neurophysiol 88:1685-1694

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

Mazzoni P, Krakauer JW (2006) An implicit plan overrides an explicit strategy during visuo-motor adaptation. J. Neurosci. 26:3642-3645

Mendenhall W, Wackerly D et al. (2008) Mathematical Statistics with Applications. Belmont, CA, USA, Thomson Brooks/Cole

Miles FA, Eighmy BB (1980) Long-term adaptive changes in primate vestibulo-ocular reflex. I. Behavioral observations. Journal of Neurophysiology 43:1406-1425

Missiuro W, Kozlowski S (1963) Investigation on adaptive changes in reciprocal innervation of muscles. Arch. Phys. Med. Rehabil. 44:37-41

Nagumo J, Noda A (1967) A learning method for system identification. IEEE Trans. Automat. Contr. AC-12:283–287

Neale JH, Barker JH (1975) Bidirectional axonal transport of Ca: studies in isolated frog sensory, motor and sympathetic neurons, Aplysia cerebral ganglion and the goldfish visual system. Brain Res. 129:45-59

Optican LM, Robinson DA (1980) Cerebellar dependent adaptive control of primate saccadic system. J. Neurophysiol. 44:1058-1076

Optican LM, Zee DS, Chu FC (1985) Adaptive response to ocular muscle weakness in human pursuit and saccadic eye movements. J. Neurophysiol. 54:110-122

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

Oztas E (2003) Neuronal tracing. Neuroanatomy 2:2-5

Porrill J, Dean P, Stone JV (2004) Recurrent cerebellar architecture solves the motor-error problem. Proc. R. Soc. B 271:789–796

Price DL, Griffin JW (1977) Tetanus toxin: retrograde axonal transport of systemically administered toxin. Nerosci. Lett. 4:61-65

Rolls ET, Deco G (2002) Computational Neuroscience of Vision. Oxford University Press, New York, USA

Rumelhart DE, Hinton GE, et al. (1987) Learning internal representations by back-propagating errors. Nature 323:533-536

Rumelhart DE, McClelland JL (1986) Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge, MA, USA

Savitzky A, Golay MJE (1964) Smoothing and differentiation of data by simplified least squares procedures. Analytical Chemistry 36: 1627-1639

Scaramella LF (1996) Cross-face facial nerve anastomosis: historical notes. Ear Nose Throat J. 75:343-54

Schultheis LW, Robinson DA (1981) Directional plasticity of the vestibuloocular reflex in the cat. Ann NY Acad Sci 374:504-512

Shepherd GM, Koch C (1990) Introduction to synaptic circuits. The Synaptic Organization of the Brain. Oxford University Press, New York, USA

Shibata T, Tabata H, Schaal S, Kawato M (2005) A model of smooth pursuit in primates based on learning the target dynamics. Neural Networks 18:213-224

Song S, Miller KD, et al. (2000) Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. Nat. Neurosci. 3:919-926

Sperry RW (1943) Effect of 180 degree rotation of the retinal field on visuomotor coordination. J. Exp. Zool. 92:263-279

Sperry RW (1945) The problem of central nervous reorganization after nerve regeneration and muscle transposition. Q. Rev. Biol. 20:311-369

Sperry RW (1947) Effect of crossing nerves to antagonistic limb muscles in the monkey. Arch. Neurol. Psychiatr. 58:452-473

Stoeckel K, Schwab M, et al. (1975) Comparison between the retrograde axonal transport of nerve growth factor and tetanus toxin in motor, sensory and adrenergic neurons. Brain Res. 99:1-16

Stork DG (1989) Is backpropagation biologically plausible? International Joint Conference on Neural Networks

Stratton GM (1897) Vision without inversion of the retinal image. Psychol. Rev. 4:341-360, 463-481

Sugita Y (1996) Global plasticity in adult visual cortex following reversal of visual input. Nature 380:523-526

Tate JR, Tollefson TT (2006) Advances in facial reanimation. Curr Opin Otolaryngol Head Neck Surg. 14:242-248

Todorov E, Jordan MI (2002) Optimal feedback control as a theory of motor coordination. Nat. Neurosci. 5:1226-1235

Tripp BY, Eliasmith C (2006) Comparison of neural circuits that estimate temporal derivatives. COSYNE 2006 abstracts, http://cosyne.org/program06/cosyne06_abstractbook_fin-al.pdf

Tweed D (1997) Three-dimensional model of the human eye-head saccadic system. J Neurophysiol 77:654-666

Tweed D, Haslwanter T, et al. (1998) Optimizing gaze control in three dimensions. Science 281: 1363-1366

Tweed D, Misslisch H, Fetter M (1994) Testing models of the oculomotor velocity-to-position transformation. J. Neurophysiol. 72:1425-1429

Van-Ooyen A, Roelfsema PR (2003) A biologically plausible implementation of error-backpropagation for classification tasks. Supplementary Proceedings of the International Conference on Artificial Neural Networks

Vera CL, Lewin MG, Kasa JC, Calderon MTD (1975) Central functional changes after facial-spinal-accessory anastomosis in man and facial-hypoglossal anastomosis in the cat. J. Neurosurg. 43:181-191

Von Helmholtz HEF (1962) Treatise on Physiological Optics. Dover, Phoenix, USA

Werfel J, Xie X, Seung HS (2005) Learning curves for stochastic gradient descent in linear feedforward networks. Neural Comput. 17:2699-2718

Widrow B, Hoff ME (1960) Adaptive switching circuits. 1960 IRE WESCON Convention Rec. 96-104

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

Yamamoto K, Kobayashi Y, Takemura A, Kawano K, Kawato M (2002) Computational studies on acquisition and adaptation of ocular following responses based on cerebellar synaptic plasticity. J. Neurophysiol. 87:1554-1571

Yildiz A, Tomishige M, et al. (2004) Kinesin walks hand-over-hand. Science 303:676-678

Yumiya H, Larsen KD, Asanuma H (1979) Motor readjustment and input-output relationship of motor cortex following crossconnection of forearm muscles in cats. Brain Res. 177:566-570

Zipser D, Rumelhart DE (1988) A backpropagation programmed network that stimulates response properties of a subset of posterior parietal neurons. Nature 331:679-684