LETTER **Communicated by Paul Cisek**

Sensitivity Derivatives for Flexible Sensorimotor Learning

M. N. Abdelghani
mohamed.abdelghani@utoronto.ca
Department of Physiology, University of Toronto, Toronto, Ontario, Canada

T. P. Lillicrap
tim@biomed.queensu.ca
Centre for Neuroscience Studies, Queen's University, Kingston, Ontario K7L 3N6, Canada

D. B. Tweed
douglas.tweed@utoronto.ca
Departments of Physiology and Medicine, University of Toronto, Toronto, Ontario M5S 1A8, Canada, and Centre for Vision Research, York University, Toronto, Ontario M3J 1P3, Canada

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 are not learned at all but are known innately. Here we show that this knowledge cannot be solely innate, given the adaptive flexibility of neural systems. And we show how it could be learned using forms of information transport that are available in the brain. The mechanism, which we call *implicit supervision*, helps explain the flexibility and speed of sensorimotor learning and our ability to cope with high-dimensional work spaces and tools.

1 Introduction

In control theory, 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 (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.

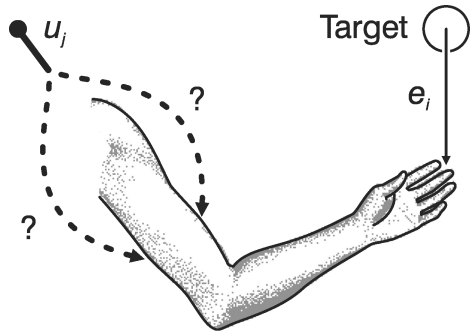

Figure 1: Error and sensitivity. Suppose an arm controller knows its vertical reaching error e_i (downward in this case) and the state of the arm. Still it cannot tell how to improve the component u_j of its motor command, or even whether to increase u_j or decrease it, unless it knows whether u_j's 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.

And similarly for any learned behavior (see section 2 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 $\partial \boldsymbol{e} / \partial \boldsymbol{u}$, called the control jacobian (Callier & Desoer, 1991), or 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 (see Figure 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, 2004; Dean, Porrill, & Stone, 2002; Kawato & Gomi, 1992), or they do not discuss where it comes from (Todorov & Jordan, 2002; Yamamoto, Kobayashi, Takemura, Kawano, & Kawato, 2002). But we will show that theories that hold that this knowledge is solely innate cannot explain the versatility of real sensorimotor learning—its capacity to recover from major lesions and 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. We consider possible mechanisms and argue that the best option is a process we call implicit supervision.

2 Mathematical Setting

We consider a control system with a plant equation,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \tag{2.1}$$

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 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 $\boldsymbol{v}$,

$$\boldsymbol{e} = \boldsymbol{g}(\boldsymbol{v}, \boldsymbol{u}). \tag{2.2}$$

More precisely, the aim is to minimize the risk, or expected loss, often defined as $\mathrm{E}(L) = \mathrm{E}(\boldsymbol{e}^T\boldsymbol{e}/2)$. To achieve this goal, the controller reads in context and generates commands according to a function called its control law,

$$\boldsymbol{u} = \boldsymbol{\gamma}(\boldsymbol{v}). \tag{2.3}$$

2.1 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

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{u} - \kappa\boldsymbol{x}}{\rho}, \tag{2.4}$$

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

$$\boldsymbol{e} = \dot{\boldsymbol{x}} + \dot{\boldsymbol{h}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \dot{\boldsymbol{h}}. \tag{2.5}$$

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 2.2. The optimal control law is

$$\boldsymbol{u} = \kappa\boldsymbol{x} - \rho\dot{\boldsymbol{h}} = (\kappa, -\rho) \cdot \boldsymbol{v}, \tag{2.6}$$

because this $\boldsymbol{u}$, plugged into equation 2.4, makes $\dot{\boldsymbol{x}} = -\dot{\boldsymbol{h}}$, and so zeroes the error $\boldsymbol{e}$ defined in equation 2.5. As required by equation 2.3, $\boldsymbol{u}$ 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 2.1 to 2.3. 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.

2.2 Sensitivity Derivatives. In the horizontal VOR, with the plant described by equation 2.4 and the error by equation 2.5, the sensitivity derivative is a single number:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} = \frac{\partial}{\partial \boldsymbol{u}}(\dot{\boldsymbol{x}} + \dot{\boldsymbol{h}}) = \frac{\partial}{\partial \boldsymbol{u}}\left(\frac{\boldsymbol{u} - \kappa \boldsymbol{x}}{\rho} + \dot{\boldsymbol{h}}\right) = \frac{1}{\rho}. \tag{2.7}$$

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 two-link arm, for instance, the matrix is

$$\begin{aligned}\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} = &\left[\delta(\alpha + \beta \cos x_2) - \left(\delta + \frac{\beta \cos x_2}{2}\right)^2\right]^{-1} \\ &\times \begin{bmatrix} \delta & -\delta - \dfrac{\beta \cos x_2}{2} \\ -\delta - \dfrac{\beta \cos x_2}{2} & \alpha + \beta \cos x_2 \end{bmatrix},\end{aligned} \tag{2.8}$$

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 Sensorimotor Learning

To learn is to adjust your control law $\boldsymbol{u} = \gamma(\boldsymbol{v})$ so as to reduce $\boldsymbol{e}$. Here we show, with a simple example, how the proper adjustments depend on the sensitivity derivatives. The principle holds for any sensorimotor task fitting equations 2.1 to 2.3 and for any learning algorithm, but for illustration purposes, we will choose the VOR and the Widrow-Hoff learning rule, also known as online-gradient or least-mean-square (LMS) learning.

We will suppose the VOR controller has the form of the ideal controller (see equation 2.6),

$$\boldsymbol{u} = \langle\kappa\rangle \boldsymbol{x} - \langle\rho\rangle \dot{\boldsymbol{h}}, \tag{3.1}$$

where the small corner brackets $\langle\rangle$ indicate neural estimates, so $\langle\kappa\rangle$ and $\langle\rho\rangle$ are parameters that are shaped, by learning, to equal the κ and ρ in the plant equation, 2.4. In this setting the LMS learning rule is described by the following equations, where η is the learning rate constant:

$$\frac{d\langle\kappa\rangle}{dt} = -\eta \frac{\partial L}{\partial \langle\kappa\rangle} = -\eta \frac{dL}{d\boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \langle\kappa\rangle} = -\eta \frac{dL}{d\boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \langle\kappa\rangle} = -\eta \boldsymbol{e}^{\mathrm{T}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} \boldsymbol{x} \tag{3.2}$$

$$\frac{d\langle\rho\rangle}{dt} = -\eta \frac{\partial L}{\partial \langle\rho\rangle} = -\eta \frac{dL}{d\boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \langle\rho\rangle} = -\eta \frac{dL}{d\boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \langle\rho\rangle} = \eta \boldsymbol{e}^{\mathrm{T}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} \dot{\boldsymbol{h}}. \tag{3.3}$$

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 from 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 do not equal κ and ρ), as in the early part of the time plot in Figure 2A. Then eye velocity $\dot{\boldsymbol{x}}$ is incorrect, and $\boldsymbol{e}$ is nonzero. But the learning rule in equations 3.2 and 3.3 repairs the problem: as long as the estimate $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u}\rangle$ is reasonably accurate, $\langle\kappa\rangle$ 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.

And 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 $\partial \boldsymbol{e}/\partial \boldsymbol{u}$, 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 2B. Equations 3.2 and 3.3 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 $\partial L/\partial \boldsymbol{u}$ need not 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 $\boldsymbol{e}^T(\partial \boldsymbol{e}/\partial \boldsymbol{u})$, so it is possible that several components of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ may reverse without reversing $\partial L/\partial \boldsymbol{u}$, and

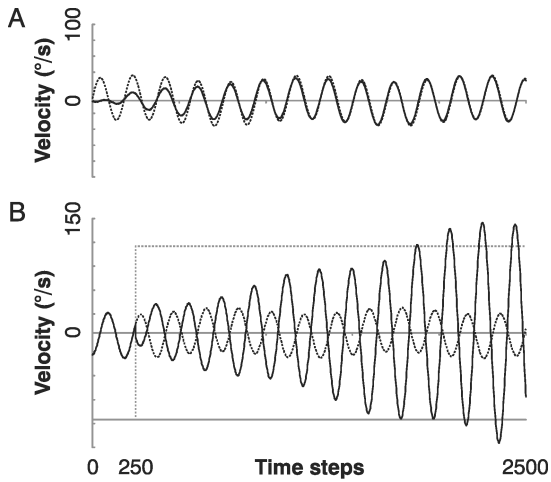

Figure 2: When sensitivity derivatives are not learned, control is inflexible. (A) In the VOR, eye velocity (solid black line) should equal –1 × head velocity (dotted black). Here a controller with correct innate knowledge of sensitivity learns the task. (B) Here the trained controller is working well 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 $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u} \rangle$ (solid gray line) incorrect, so $\langle \partial L/\partial \boldsymbol{u} \rangle$ is also reversed and drives learning in the wrong direction. The system never recovers. (Ordinate scale refers to velocities, not sensitivity derivatives.)

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?

4 Are Sensitivity Derivatives Innately Known or Learned?

Theoretically innateness and learning both 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 do not 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. 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 we discuss later.

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

There are many theories of plasticity in the VOR, but none explains 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 do not reverse the sign of $\partial L/\partial \boldsymbol{u}$: 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 we are 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 we discuss in section 4.2.)

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 work (Kawato & Gomi, 1992): they do not 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 articles from this same lab focus on other issues, but they work similarly regarding $\partial \boldsymbol{e}/\partial \boldsymbol{u}$. 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 work such as Shibata, Tabata, Schaal, and Kawato (2005). Simulations of that theory, applied to the same tasks as those in Figure 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 does not 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 et al. (2004), similarly relies on innate estimates. Applied to the VOR, it too reproduces the behavior in Figure 2. And as with Kawato's theory, this one could be made flexible by adding a mechanism that learns sensitivity derivatives.

Another example is work by Todorov and colleagues on the LQG framework for motor control (e.g., Todorov & Jordan, 2002). Their algorithms for shaping motor control sequences are not 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 $\boldsymbol{B} = \partial \boldsymbol{e}/\partial \boldsymbol{u}$.

4.2 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, shows that real learning, in at least some sensorimotor systems, does not 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, Lewin, Kasa, & Calderon, 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; that is, $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ 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 do not 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

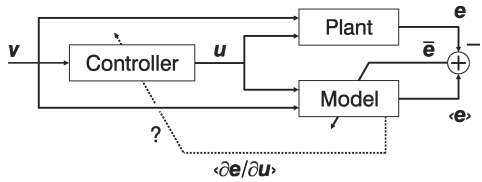

Figure 3: How could controllers learn sensitivity derivatives? As Jordan and Rumelhart (1992) showed, a plant model contains information about sensitivity derivatives, but normally the information is distributed among the synaptic weights of the model. Unless that information is represented in neural firing, how can it be transmitted to the controller?

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 et al., 1975; Leffert & Meister, 1976; Yumiya et al., 1979; Brinkman et al., 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 $\sim$0.1%, of all possible values of the partial derivative of the loss with respect to $\boldsymbol{u}$. This calculation, and the experimental data, indicate that in a complex, changing world, survival depends on learning sensitivity derivatives.

4.3 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 we show in Figure 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 the 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 have not 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 (see Figure 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. We will show that there are ways sensitivity derivatives could be learned using only biologically realistic forms of information transport. We start in the next section with the mechanism we consider most promising—essentially a version of the distal teacher without weight transport—and then consider other approaches.

5 Implicit Supervision

5.1 The Basic Idea. We want 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. If we define $\boldsymbol{z} = (\boldsymbol{v}, \boldsymbol{u})$, then we can write

$$\boldsymbol{e} = \boldsymbol{g}(\boldsymbol{v}, \boldsymbol{u}) = \boldsymbol{g}(\boldsymbol{z}). \tag{5.1}$$

The point is to provide a single vector, $\boldsymbol{z}$, which determines $\boldsymbol{e}$. As defined in example 2.1, 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,

$$\dot{\boldsymbol{e}} = \frac{d\boldsymbol{e}}{d\boldsymbol{z}}\dot{\boldsymbol{z}}. \tag{5.2}$$

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 $\partial \boldsymbol{e}/\partial \boldsymbol{u}$.

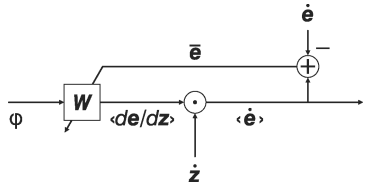

Figure 4: Implicit supervision creates a network that represents sensitivity derivatives in transmissible form. Here $\boldsymbol{z}$ is a vector containing information about the context $\boldsymbol{v}$ and command $\boldsymbol{u}$, and $\boldsymbol{\varphi}$ is a function of $\boldsymbol{z}$. This circuit learns to code the total derivative $d\boldsymbol{e}/d\boldsymbol{z}$ in neural firing. Because $d\boldsymbol{e}/d\boldsymbol{z}$ contains the sensitivity $\partial\boldsymbol{e}/\partial\boldsymbol{u}$ as a submatrix, all components of the estimate $\langle\partial\boldsymbol{e}/\partial\boldsymbol{u}\rangle$ are coded in a subset of the axons carrying $\langle d\boldsymbol{e}/d\boldsymbol{z}\rangle$. So this scheme can deduce sensitivity derivatives and transmit them to a controller.

We want a network that learns to compute $d\boldsymbol{e}/d\boldsymbol{z}$. What sort of input does it need? By equation 5.1, we know $\boldsymbol{e}$ is determined by $\boldsymbol{z}$, so $d\boldsymbol{e}/d\boldsymbol{z}$ is computable from $\boldsymbol{z}$ as well. In most systems, 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 $\boldsymbol{\varphi}(\boldsymbol{z})$—$\boldsymbol{\varphi}$, for short. Then we approximate each element of the matrix $d\boldsymbol{e}/d\boldsymbol{z}$ by taking the inner product of $\boldsymbol{\varphi}$ with a weight vector $\boldsymbol{w}$,

$$\frac{\partial e_i}{\partial z_j} \approx \boldsymbol{w}_{ij}^{\mathsf{T}}\,\boldsymbol{\varphi} = \sum_{k=1}^{n_\varphi} w_{ijk}\varphi_k, \tag{5.3}$$

as in Figure 4.

Expansion recoding and feature vectors appear also in many other theories that strive for biological realism (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; for example, if you know that the sensitivity derivatives are quadratic functions of $\boldsymbol{z}$, choose quadratic features. If you do not have much prior knowledge, then more generic features also work, so long as you have a large and varied set of them. In this letter we take both approaches to feature selection: handpicking for the VOR, generic for our simulations of a two-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 (nonlearning) synaptic weights w_f, to yield a $\boldsymbol{\varphi}$ vector

with components

$$\varphi_i = \tanh\left(\sum_{j=1}^{n_z} w_{fij} z_j\right). \tag{5.4}$$

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; for example, 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 5.2, 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,

$$\bar{\boldsymbol{e}} = \langle \dot{\boldsymbol{e}}\rangle - \dot{\boldsymbol{e}} = \langle d\boldsymbol{e}/d\boldsymbol{z}\rangle \dot{\boldsymbol{z}} - \dot{\boldsymbol{e}}, \tag{5.5}$$

and model loss,

$$\bar{L} = \frac{\bar{\boldsymbol{e}}^{\mathsf{T}}\bar{\boldsymbol{e}}}{2}. \tag{5.6}$$

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

$$\begin{aligned}
\frac{\partial \bar{L}}{\partial w_{ijk}} &= \frac{\partial \bar{L}}{\partial \bar{\boldsymbol{e}}}\frac{\partial \bar{\boldsymbol{e}}}{\partial w_{ijk}} = \bar{\boldsymbol{e}}^{\mathsf{T}}\frac{\partial \bar{\boldsymbol{e}}}{\partial w_{ijk}} = \sum_{\alpha=1}^{n_e} \bar{e}_\alpha \frac{\partial \bar{e}_\alpha}{\partial w_{ijk}} \\
&= \sum_{\alpha=1}^{n_e} \bar{e}_\alpha \frac{\partial}{\partial w_{ijk}} \left([\langle d\boldsymbol{e}/d\boldsymbol{z}\rangle \dot{\boldsymbol{z}} - \dot{\boldsymbol{e}}]_\alpha\right) \text{(by equation 5.5)} \\
&= \sum_{\alpha=1}^{n_e} \bar{e}_\alpha \frac{\partial}{\partial w_{ijk}} \left(\sum_{\beta=1}^{n_z} \langle de_\alpha/dz_\beta\rangle \dot{z}_\beta\right) \\
&\qquad\qquad \text{(omit } -\dot{\boldsymbol{e}}\text{, as it doesn't depend on } w_{ijk}\text{)}
\end{aligned}$$

$$
\begin{aligned}
&= \bar{e}_i \frac{\partial}{\partial w_{ijk}} (\langle de_i/dz_j \rangle \dot{z}_j) \\
&\qquad\qquad (\langle de_\alpha/dz_\beta \rangle \text{ depends on } w_{ijk} \text{ only if } \alpha = i, \beta = j) \\
&= \bar{e}_i \frac{\partial}{\partial w_{ijk}} \left(\sum_{\gamma=1}^{n_\varphi} w_{ij\gamma} \varphi_\gamma \dot{z}_j \right) \text{(by equation 5.3)} \\
&= \bar{e}_i \frac{\partial}{\partial w_{ijk}} \left(w_{ijk} \varphi_k \dot{z}_j \right) \\
&= \bar{e}_i \dot{z}_j \varphi_k, \qquad (5.7)
\end{aligned}
$$

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

$$\dot{w}_{ijk} = -\eta \bar{e}_i \dot{z}_j \varphi_k. \qquad (5.8)$$

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 4 shows $\bar{\boldsymbol{e}}$ arriving at the weight array $\boldsymbol{W}$. To avoid clutter here and in Figure 5, we do not 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 5.8 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, $d\boldsymbol{e}/d\boldsymbol{z}$. We 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 our simulations of implicit supervision in this letter we use a rule called *normalized least mean square* (NLMS) (Nagumo & Noda, 1967):

$$\Delta w_{ijk} = -\eta \frac{\bar{e}_i \dot{z}_j \varphi_k}{(\dot{\boldsymbol{z}}^\top \dot{\boldsymbol{z}})(\boldsymbol{\varphi}^\top \boldsymbol{\varphi})}. \qquad (5.9)$$

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 2B and 6A of this letter, we 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 the other figures showing implicit supervision,

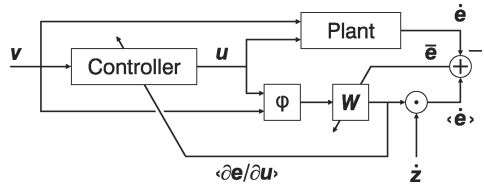

Figure 5: Control by implicit supervision. The circuit from Figure 4, incorporated at the lower right in this flow diagram, serves as a plant model that learns to represent $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ in neural firing and sends the information to the controller.

we used the optimal η.) Another possible learning rule, which we did not use in this letter but may be viable in the brain, is the recursive least-squares algorithm (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 5. The lower part of the circuit serves as a plant model, and because it codes $d\boldsymbol{e}/d\boldsymbol{z}$ 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 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 $\partial L/\partial \boldsymbol{u}$.

Figure 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 our equations in section 5, and Figures 7A and 7B provide an example: the sensitivity derivatives change as the arm moves about its work space, but the plant model's estimates continues to track them. The second point is that an adaptive controller does not need the exact values of the sensitivity derivatives, but only the signs of all components of $\partial L/\partial \boldsymbol{u}$, as we discussed in section 3. 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 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.

5.2 Redundancy and Constraints. Figure 8 shows that implicit supervision can deal with kinematic redundancy, that is, situations where the

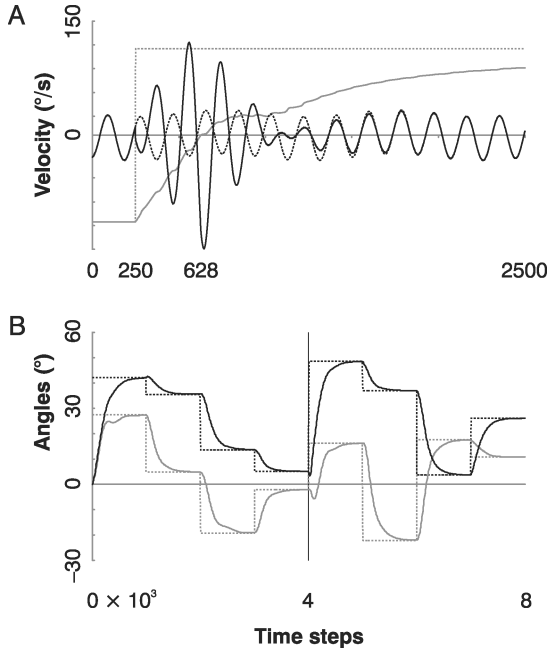

Figure 6: Implicit supervision provides flexible adaptive control. (A) As in Figure 2B, the sensitivity derivative switches sign at time step 250, but here the controller learns in the wrong direction only until time step 628, when the learned estimate $\langle \partial \boldsymbol{e}/\partial \boldsymbol{u} \rangle$ (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 two-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.

plant has more degrees of freedom than it needs for its task. In Figure 8A, the task is to control a two-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 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

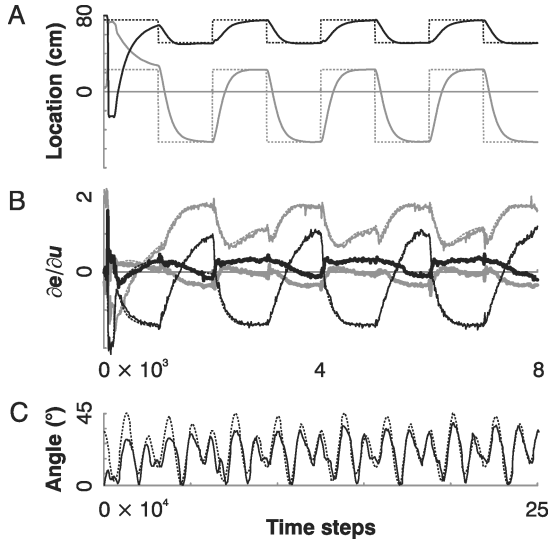

Figure 7: Properties of implicit supervision. (A) A controller learns to move the horizontal and vertical components of hand location (black and gray solid lines) to their targets (dotted), even though the sensitivity derivatives change through the work space. (B) The plant model's estimates (solid lines) 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 $\partial L/\partial \boldsymbol{u}$.

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 8B, where the controller learns to orient the forearm, as before, but now prefers configurations where shoulder and elbow angles are equal. Figure 8C shows another version, which prefers to hold the elbow near 15 degrees of flexion. This last case is closely analogous to the control of saccadic eye movements, where each eyeball has 3 degrees of freedom—horizontal, vertical, and torsional—but needs only 2, and so Listing's law of the eye holds ocular torsion near 0 degrees (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 et al., 1994). These blips may arise for the same reason in Figure 8C and in real saccades—the constraint has been imperfectly learned.

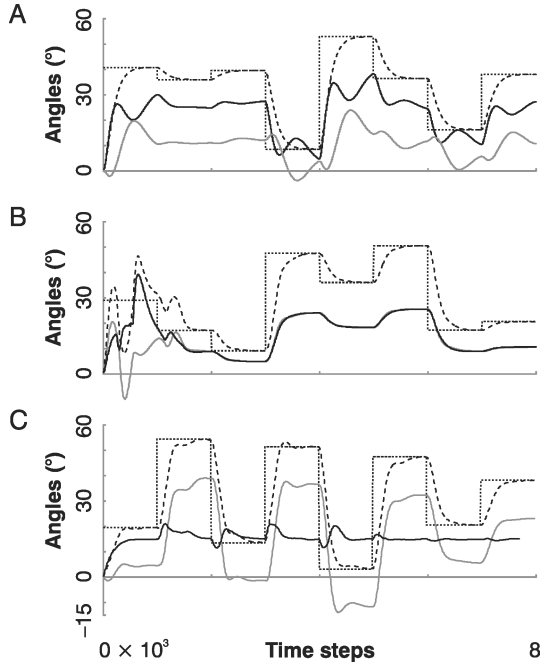

Figure 8: Implicit supervision can cope with kinematic redundancy. (A) The task is to control shoulder and elbow so as to bring the forearm's angle relative to space (dashed line) to a target value (dotted line). The controller learns the task but allows the angles of the individual joints—the elbow 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 that prefers arm configurations where shoulder and elbow angles are equal. (C) Another version prefers to hold the elbow near 15 degrees of flexion.

5.3 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 does not 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, and 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 5.8 or 5.9, we can base it on recursive least squares (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 5.9, we can still get an impression of learning speed by using simulations as in Figures 6B, 7A, 7B, 8A, 8B, and 8C. 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 φ 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.

6 Other Approaches

There are other mechanisms besides implicit supervision that could learn sensitivity derivatives for sensorimotor control, though the literature offers only a few alternatives, because it contains very few algorithms that address our 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 we know, the only algorithms that fulfill these conditions belong to just two classes: Boltzmann-like mechanisms and some forms of reinforcement learning (Sutton & Barto, 1998) called perturbation methods.

In the case of Boltzmann machines, for our 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 variants of the Boltzmann machine (e.g., Hinton, Osindero, & Teh, 2006) may avoid these problems.

6.1 Perturbation. The most promising alternative to implicit supervision may be the reinforcement-learning method known as node perturbation (Werfel et al., 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 $\boldsymbol{u}$, for example,

$$\boldsymbol{u} = \boldsymbol{w} \cdot \boldsymbol{\varphi}(\boldsymbol{v}), \tag{6.1}$$

where $\boldsymbol{\varphi}$ 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

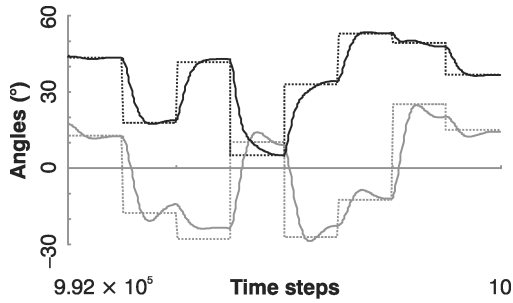

Figure 9: A controller learns by command perturbation to reach for targets, but the learning is slow compared to implicit supervision in Figure 6A. This plot shows the final 8000 time steps of a million-step run, when elbow control (black) is fairly good but the shoulder (gray) is still ragged.

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,

$$\dot{w}_i = -\eta(L_p - L)(\boldsymbol{u}_p - \boldsymbol{u})\varphi_i. \tag{6.2}$$

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 feature vector $\boldsymbol{\varphi}$.

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. Then $(L_p - L)(\boldsymbol{u}_p - \boldsymbol{u})$ is a better estimate of $\partial L/\partial \boldsymbol{u}$, and the controller learns faster. But it is still far slower than with implicit supervision, as you can see by comparing Figure 9 to Figure 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.

6.2 Loss-Based Implicit Supervision. In equations 5.2 to 5.9 we 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.

6.3 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 and 5 display the similarity between the two approaches: the chain of operators comprising $\boldsymbol{\varphi}$, $\boldsymbol{W}$, and the multiplier in Figure 5 forms a plant model, and plays the same role as the plant model in Figure 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).

7 Other Advantages of Learned Sensitivity

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

7.1 Fast-Learning Controllers. In Figure 7 we saw that even very rough estimates of sensitivity derivatives can be used to train a controller. But, the more exactly $\partial\boldsymbol{e}/\partial\boldsymbol{u}$ is known, the faster the controller can learn—compare the rapid improvement in Figure 6B with the slow progress in Figure 7C. So for this reason also, natural selection may have favored mechanisms for deducing $\partial\boldsymbol{e}/\partial\boldsymbol{u}$.

7.2 Learning Complex Plants. We 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. 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. Figure 10 gives an impression of how complex these

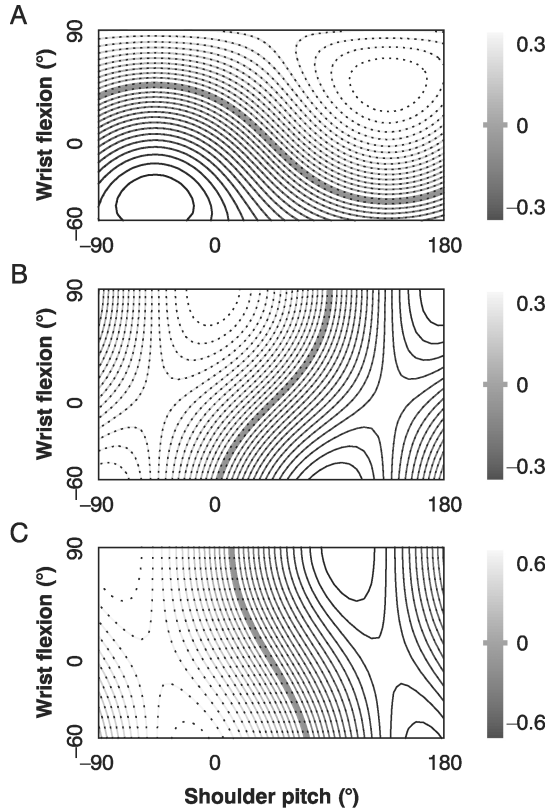

Figure 10: Sensitivity derivatives vary depending on context. (A) A map of $\partial e_3/\partial u_2$—the derivative of vertical reaching error with respect to wrist-flexion commands—versus shoulder pitch and wrist flexion, for a 7-DOF stepper-motor arm. The derivative is represented by the gray level, 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 handheld stick.

dependencies can be. It is based on a simplified "stepper-motor" arm, where $\boldsymbol{u}$ is a vector of seven 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 10A shows a map of one component, $\partial e_3/\partial u_2$ (represented by contour lines), versus two 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 7D. Through each point in the space, there are 21 orthogonal 2D slices, and over each slice, each of the 21 components of $\partial \boldsymbol{e}/\partial \boldsymbol{u}$ varies in a different way (compare Figures 10A and 10B). With a real, nonstepper 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 10C shows how one derivative map for reaching error transforms when you reach with a handheld 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 do not 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.

8 Conclusion

This theory boils down to three points, with different degrees of experimental support. First, we 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 et al., 1979; Brinkman et al., 1983; Sugita, 1996). Further support comes from the arguments in sections 7.1 and 7.2 that fast motor learning and tool use suggest detailed knowledge of sensitivity derivatives.

Second, we claim that sensitivity is transmitted by action potentials, 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, we 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 5, including multiplication and differentiation, can be done with neurons (Koch, 1999; Tripp & Eliasmith, 2006), and the mechanism is

consistent with many neural circuits; for example, $\boldsymbol{\varphi}$ 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 of 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 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 5.8, 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, we 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, we have described a mechanism, called implicit supervision, that 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 2.1 to 2.3, and it could be applied whenever the brain needs to learn a computation for which it has no supervisor.

Appendix A: Generality

The framework defined in equations 2.1 to 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. We 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 would not give an adaptive controller much moment-to-moment guidance; it would not tell it how to adjust its control law, precisely because the equation for $\boldsymbol{e}$ does not involve $\boldsymbol{u}$. 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 $\boldsymbol{e}$, the system is now of relative degree zero. And if the controller adjusts its commands so as to zero $\boldsymbol{e}$, then saccades will obey

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

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 letter. Our point here is simply that a higher-degree control problem can be converted to relative degree zero.

Appendix B: Two-Joint Arm

In Figures 6B through 9, we use a two-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 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 $\boldsymbol{u}$ is just two-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

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

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 6B, we reverse the polarities of both joint torques; in the plant equation, we 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 $\boldsymbol{\varphi}(\boldsymbol{z})$ is computed as in equation 5.4, 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 6B through 9 computes its command as a linear function of its own feature vector $\boldsymbol{\varphi}^{\boldsymbol{u}}$ and adjusts its weight matrix $\boldsymbol{W}^{\boldsymbol{u}}$ by NLMS,

$$\Delta w_{ij}^{\boldsymbol{u}} = -\eta \frac{L_{ui}\varphi_j^{\boldsymbol{u}} L}{\|L_{\boldsymbol{u}}\|^2 \|\boldsymbol{\varphi}^{\boldsymbol{u}}\|^2}, \tag{B.2}$$

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

Acknowledgments

For their comments we thank K. Beykirch, D. Broussard, J. Butler, L. Chinta Venkataswararao, A. Coderre, K. Fortney, D. Henriques, I. Kurtzer, M. Maggiore, K. Norwich, J. Peters, A. Pruszynski, S. Scott, D. Tomlinson, and M. Wojtowicz. This work was funded by the Canadian Institutes of Health Research.

References

Ackley, D., Hinton, G. E., & Sejnowski, T. (1985). A learning algorithm for Boltzmann machines. *Cog. Sci.*, *9*, 147–169.

Åström, K. J., & Wittenmark, B. (1995). *Adaptive control*. Reading, MA: Addison-Wesley.

Brinkman, C., Porter, R., & Norman, J. (1983). Plasticity of motor behavior in monkeys with crossed forelimb nerves. *Science*, *220*, 438–440.

Callier, F. M., & Desoer, C. A. (1991) *Linear system theory*. New York: Springer.

Dean, P., Porrill, J., & Stone, J. V. (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 behavior. *Genet. Psychol. Monogr.*, *7*, 177–363.

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.

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

Haykin, S. (2002). *Adaptive filter theory*. Upper Saddle River, NJ: Prentice Hall.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, *18*, 1527–1554.

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cog. Sci.*, *16*, 307–354.

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

Koch, C. (1999). *Biophysics of computation*. New York: Oxford University Press.

Leffert, R. D., & Meister, M. (1976). Patterns of neuromuscular activity following tendon transfer in the upper limb: A preliminary study. *J. Hand Surg.*, *1*, 181–189.

Lewis, F. L., Jagannathan, S., & Yeşildirek, A. (1999). *Neural network control of robot manipulators and nonlinear systems*. London: Taylor and Francis.

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

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.

Oztas, E. (2003). Neuronal tracing. *Neuroanatomy*, *2*, 2–5.

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

Rolls, E. T., & Deco, G. (2002). *Computational neuroscience of vision*. New York: Oxford University Press.

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*(3), 213–224.

Sperry, R. W. (1943). Effect of 180 degree rotation of the retinal field on visuomotor coordination. *J. Exp. Zool.*, *92*, 263–279.

Sperry, R. W. (1945). The problem of central nervous reorganization after nerve regeneration and muscle transposition. *Q. Rev. Biol.*, *20*(4), 311–369.

Stratton, G. M. (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.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning*. Cambridge, MA: MIT Press.

Todorov, E., & Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nat. Neurosci.*, *5*, 1226–1235.

Tripp, B. Y., & Eliasmith, C. (2006). Comparison of neural circuits that estimate temporal derivatives. *COSYNE 2006 abstracts*. Available online at http://cosyne.org/program06/cosyne06_abstractbook_final.pdf.

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

Vera, C. L., Lewin, M. G., Kasa, J. C., & Calderon, M. T. D. (1975). Central functional changes after facial-spinal-accessory anastomosis in man and facial-hypoglossal anastomosis in the cat. *J. Neurosurg.*, *43*, 181–191.

Werfel, J., Xie, X., & Seung, H. S. (2005). Learning curves for stochastic gradient descent in linear feedforward networks. *Neural Comput.*, *17*, 2699–2718.

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.

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

Received April 2, 2007; accepted December 13, 2007.