LETTER **Communicated by Jean-Philippe Thivierge**

A Reservoir Computing Model of Reward-Modulated Motor Learning and Automaticity

Ryan Pyle
rpyle1@nd.edu
Department of Applied and Computational Mathematics and Statistics,
University of Notre Dame, Notre Dame, IN 46556, U.S.A.

Robert Rosenbaum
Robert.Rosenbaum@nd.edu
Department of Applied and Computational Mathematics and Statistics
and Interdisciplinary Center for Network Science and Applications,
University of Notre Dame, Notre Dame, IN 46556, U.S.A.

Reservoir computing is a biologically inspired class of learning algorithms in which the intrinsic dynamics of a recurrent neural network are mined to produce target time series. Most existing reservoir computing algorithms rely on fully supervised learning rules, which require access to an exact copy of the target response, greatly reducing the utility of the system. Reinforcement learning rules have been developed for reservoir computing, but we find that they fail to converge on complex motor tasks. Current theories of biological motor learning pose that early learning is controlled by dopamine-modulated plasticity in the basal ganglia that trains parallel cortical pathways through unsupervised plasticity as a motor task becomes well learned. We developed a novel learning algorithm for reservoir computing that models the interaction between reinforcement and unsupervised learning observed in experiments. This novel learning algorithm converges on simulated motor tasks on which previous reservoir computing algorithms fail and reproduces experimental findings that relate Parkinson's disease and its treatments to motor learning. Hence, incorporating biological theories of motor learning improves the effectiveness and biological relevance of reservoir computing models.

1 Introduction

Even simple motor tasks require intricate, dynamical patterns of muscle activations. Understanding how the brain generates this intricate motor output is a central problem in neuroscience that can inform the development of brain-machine interfaces, treatments for motor diseases, and control algorithms for robotics. Recent work is largely divided into addressing two

Neural Computation **31, 1–32** (2019)
doi:10.1162/neco_a_01198

distinct questions: How are motor responses encoded, and how are they learned?

From the coding perspective, it has been shown that the firing rates of cortical neurons exhibit intricate dynamics that do not always code for specific stimulus or movement parameters (Churchland et al., 2012; Russo et al., 2018). A prevailing theory poses that these firing rate patterns are part of an underlying dynamical system that serves as a high-dimensional "reservoir" of dynamics from which motor output signals are distilled (Shenoy, Sahani, & Churchland, 2013; Sussillo, 2014). This notion can be formalized by reservoir computing models, in which a chaotic or near-chaotic recurrent neural network serves as a reservoir of firing rate dynamics and synaptic readout weights are trained to produce target time series (Maass, Natschläger, & Markram, 2002; Jaeger & Haas, 2004; Sussillo & Abbott, 2009; Lukoševičius, Jaeger, & Schrauwen, 2012; Sussillo, 2014).

Reservoir computing models can learn to generate intricate dynamical responses and naturally produce firing rate dynamics that are strikingly similar to those of cortical neurons (Sussillo, Churchland, Kaufman, & Shenoy, 2013; Mante, Sussillo, Shenoy, & Newsome, 2013; Laje & Buonomano, 2013; Hennequin, Vogels, & Gerstner, 2014). However, most reservoir computing models rely on biologically unrealistic, fully supervised learning rules. Specifically, they must learn from a teacher signal that can already generate the target output. Many motor tasks are not learned in an environment in which such a teacher signal is available. Instead, motor learning is at least partly realized through reward-modulated, reinforcement learning rules (Izawa & Shadmehr, 2011).

A large body of studies are committed to understanding how reinforcement learning is implemented in the motor systems of mammals and songbirds (Brainard & Doupe, 2002; Olveczky, Andalman, & Fee, 2005; Kao, Doupe, & Brainard, 2005; Ashby, Turner, & Horvitz, 2010; Izawa & Shadmehr, 2011; Fee, 2014). The basal ganglia and their homologue in songbirds play a critical role in reinforcement learning of motor tasks through dopamine-modulated plasticity at corticostriatal synapses. This notion inspired the development of a reward-modulated learning rule for reservoir computing (Hoerzer, Legenstein, & Maass, 2014). However, we found that this learning rule fails to converge on many simulated motor tasks.

We propose that the shortcomings of previous reservoir computing models can be resolved by a closer inspection of the literature on biological motor learning. A large body of evidence across multiple species supports a theory of learning in which dopamine-modulated plasticity in the basal ganglia or its homologues is responsible for early learning, and this pathway gradually trains a parallel cortical pathway that takes over as tasks become well learned or "automatized" (Bottjer, Miesner, & Arnold, 1984; Carelli, Wolske, & West, 1997; Brainard & Doupe, 2000; Pasupathy & Miller, 2005; Ashby, Ennis, & Spiering, 2007; Obeso et al., 2009; Andalman & Fee, 2009; Ashby et al., 2010; Turner & Desmurget, 2010; Fee & Goldberg, 2011;

Ölveczky, Otchy, Goldberg, Aronov, & Fee, 2011), although the biology is not settled (Kawai et al., 2015). This model of motor learning has been tested computationally only in discrete choice tasks that do not capture the intricate, dynamical nature of motor responses (Ashby et al., 2007).

Inspired by this theory of automaticity from parallel pathways, we derived a new architecture and learning rule for reservoir computing. In this model, a reward-modulated pathway is responsible for early learning and serves as a teacher signal for a parallel pathway that takes over the production of motor output as the task becomes well learned. This algorithm is applicable to a large class of motor learning tasks to which fully supervised learning models cannot be applied, and it outperforms previous reward-modulated models. We also show that our model naturally produces experimental and clinical findings that relate Parkinson's disease and its treatment to motor learning (Ashby et al., 2007, 2010; Turner & Desmurget, 2010).

2 Results

We first review two previous learning rules for reservoir computing and then introduce a new, biologically inspired learning rule that combines their strengths.

2.1 FORCE Learning. One of the most powerful and widely used reservoir computing algorithms is first-order reduced and controlled error (FORCE; Sussillo & Abbott, 2009), which is able to rapidly and accurately learn to generate complex, dynamical outputs. The standard architecture for FORCE is schematized in Figure 1A (FORCE variants exist, although the underlying principle is the same). The reservoir is composed of a recurrently connected population of "rate-model" neurons. The output of the reservoir is trained to produce a target time series by modifying a set of readout weights, and the output affects the reservoir through a feedback loop.

The reservoir dynamics are defined by

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + J\mathbf{r} + Q\mathbf{z}. \tag{2.1}$$

Here,

$$\mathbf{r} = \tanh(\mathbf{x}) + \boldsymbol{\epsilon}$$

is a time-dependent vector representing the activity of units within the reservoir, τ is a time constant, $\boldsymbol{\epsilon}$ is a small noise term,

$$\mathbf{z} = W\mathbf{r}$$

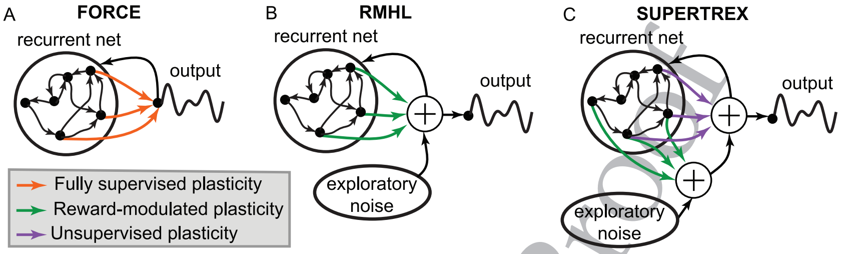

Figure 1: Network diagrams for three reservoir computing algorithms. (A) In FORCE learning, readout weights are trained to match a target using a fully supervised error signal. Output is fed back to the reservoir. (B) RMHL is similar to FORCE, but learning is driven by a reward-modulated plasticity and exploratory noise. (C) SUPERTREX combines elements of FORCE and RMHL. The exploratory pathway (green) is driven by noise and acts similar to RMHL. The mastery pathway (purple) is analogous to FORCE, but uses the output of the exploratory pathway in place of the fully supervised signal that FORCE uses. The sum of both pathways provides the total output.

is the reservoir output, J is the recurrent connectivity matrix, Q the feedback weights, and W the readout weights. A feedforward input term is also commonly included (Sussillo & Abbott, 2009), but we do not use one here. When the spectral radius of J is sufficiently large, the intrinsic dynamics of $\mathbf{r}(t)$ become rich and chaotic (Sompolinsky, Crisanti, & Sommers, 1988). The goal of FORCE is to utilize these rich dynamics by modifying readout weights, W, in such a way that the output, $\mathbf{z}(t)$, matches a desired target function, $\mathbf{f}(t)$. A powerful and widely used learning algorithm for FORCE, recursive least squares (RLS), is defined by (Sussillo & Abbott, 2009)

$$\tau_w \frac{dW}{dt} = -\mathbf{e}\mathbf{r}^T P, \tag{2.2}$$

where

$$\mathbf{e}(t) = \mathbf{z}(t) - \mathbf{f}(t) \tag{2.3}$$

is the error vector and τ_w is the learning timescale. The matrix, P, is a running estimate of the inverse of the correlation matrix of rates, $\mathbf{r}$ (see section 4).

FORCE excels at generating a target time series by harvesting reservoir dynamics, but it is incomplete as a model of motor learning. As a fully supervised learning rule, FORCE must have access to the correct output to determine its error (see the presence of $\mathbf{f}$ in equation 2.3). Since the correct output must already be generated to compute the error, FORCE can learn only target functions that are already known explicitly and can be generated. Many motor learning tasks require the generation of an unknown target using a lower-dimensional error signal (Izawa & Shadmehr, 2011). We consider examples of such tasks below. A potential solution for these issues is provided by appealing to biological motor learning, which is controlled at least in part by dopamine-modulated reinforcement learning in the basal ganglia (Turner & Desmurget, 2010; Ashby et al., 2010; Izawa & Shadmehr, 2011).

2.2 Reward-Modulated Hebbian Learning. Reward-modulated Hebbian learning (RMHL) (Hoerzer et al., 2014) is a reinforcement learning rule for reservoir computing in which reward is indicated by a one-dimensional error signal using a plasticity rule inspired by dopamine-dependent Hebbian plasticity observed in the basal ganglia. RMHL uses the same reservoir dynamics (see equation 2.1) and the same basic architecture as FORCE (see Figure 1B), but the learning rule is fundamentally different.

The original RMHL algorithm (Hoerzer et al., 2014) used a binary error signal, despite potentially poorer performance from other options, to demonstrate that the algorithm could learn with minimal information. We implement a modified version of RMHL with an error signal,

$$e(\mathbf{z}(t), t),$$

that can be any time-dependent, nonnegative function of the output, $\mathbf{z}$, which the algorithm will seek to minimize. Equivalently, e is proportional to the negative of reward.

In contrast to the fully supervised error signal, $\mathbf{e}$, used in FORCE learning, e is scalar (one-dimensional) even when it is a higher-dimensional vector. Moreover, e can quantify any notion of "error" or "cost" associated with the output, $\mathbf{z}$, and does not assume that a target output is known or even that there exists a unique target output. This allows RMHL to be applied to a large class of learning tasks to which FORCE cannot be applied, as we demonstrate below.

To decrease error, RMHL makes random perturbations to the reservoir output as a form of exploration. Specifically, the output, $\mathbf{z}$, is given by

$$\mathbf{z} = W\mathbf{r} + \Psi(e)\boldsymbol{\eta},$$

where $\boldsymbol{\eta}(t)$ is exploratory noise and Ψ is a sublinear function that serves to damp runaway oscillations during learning. The learning rule is then given by

$$\tau_w \frac{dW}{dt} = \Phi(\hat{e})\hat{\mathbf{z}}\mathbf{r}^T, \tag{2.4}$$

where $\hat{\mathbf{x}}$ denotes a high-pass filtered version of $\mathbf{x}$, which represents recent changes in $\mathbf{x}$ and Φ is a sublinear function that controls when to update the weights. We assume that Ψ is an increasing function and Φ an odd function with $\Psi(0) = \Phi(0) = 0$. This ensures that exploration and learning are effectively quenched when the error is consistently near zero. Intuitively, the learning rule can be understood as follows: if a random perturbation from $\boldsymbol{\eta}$ has recently decreased e, this perturbation is then incorporated into W.

2.3 SUPERTREX: A New Learning Algorithm for Reservoir Computing.

Unfortunately, on many tasks, the weights trained by RMHL fail to converge to an accurate solution, as we show below. RMHL models dopamine-modulated learning in the basal ganglia but does not account for experimental evidence for the eventual independence of well-learned tasks on the activity of the basal ganglia. It has been proposed that the basal ganglia are responsible for early learning but train a parallel cortical pathway that gradually takes over the generation of output as tasks become well learned and "automatized" (Pasupathy & Miller, 2005; Ashby et al., 2007, 2010; Turner & Desmurget, 2010; Hélie, Paul, & Ashby, 2012). This could explain why some neurons in the basal ganglia are active during early learning and exploration but inactive as the task becomes well learned (Carelli et al., 1997; Miyachi, Hikosaka, & Lu, 2002; Pasupathy & Miller, 2005; Poldrack et al., 2005; Ashby et al., 2007, 2010; Tang et al., 2009; Hélie et al., 2012). It could also explain why patients or animals with basal ganglia lesions can perform previously learned tasks well but suffer impairments at learning new tasks (Miyachi, Hikosaka, Miyashita, Kárádi, & Rand, 1997; Obeso et al., 2009; Turner & Desmurget, 2010). This idea is also consistent with many findings suggesting that the basal ganglia homologue in songbirds is responsible for early learning and exploration of novel song production, but not for the vocalization of well-learned songs (Brainard, 2004; Kao et al., 2005; Aronov, Andalman, & Fee, 2008; Andalman & Fee, 2009; Fee & Goldberg, 2011).

The FORCE and RMHL algorithms could be seen as analogous to the individual pathways in this theory of motor learning: RMHL learns through reward-modulated exploration analogous to the basal ganglia, while FORCE models cortical pathways that learn from the output produced by the basal ganglia pathway. Inspired by this analogy, we introduce a new algorithm, supervised learning trained by rewarded exploration (SUPERTREX), that combines the strengths of RMHL and FORCE to overcome the limitations of each.

The architecture of SUPERTREX (see Figure 1C) is different from the architectures of FORCE and RMHL: There are now two distinct sets of weights from the reservoir to the outputs, and each is trained with a separate learning rule. The exploratory pathway learns via an RMHL-like, reinforcement learning algorithm, requiring only a one-dimensional metric of performance rather than an explicit error signal. The exploratory pathway is roughly based on the biological basal ganglia pathway. The mastery pathway learns through a FORCE-like algorithm. The key idea is that the activity of the exploratory pathway can act as a target for the mastery pathway to learn, replacing the supervised error signal required by FORCE. Hence, SUPERTREX does not need the explicit supervisory error signal that FORCE does. The mastery pathway is roughly based on the biological cortical pathway.

Importantly, the convergence issues we have found with RMHL are not problematic for SUPERTREX because weights in the RMHL-like exploratory pathway do not need to converge to a correct solution; weights in the mastery pathway converge instead.

SUPERTREX uses the same reservoir dynamics (see equation 2.1), but the outputs are determined by

$$\begin{aligned}\mathbf{z}_1 &= W_1\mathbf{r} + \Psi(e)\boldsymbol{\eta},\\ \mathbf{z}_2 &= W_2\mathbf{r},\\ \mathbf{z} &= \mathbf{z}_1 + \mathbf{z}_2.\end{aligned}$$

Here, $\mathbf{z}_1$ is the output from the exploratory pathway, $\mathbf{z}_2$ from the mastery pathway, and $\mathbf{z}$ is the total output. The learning rules are defined by

$$\begin{aligned}\tau_{w1}\frac{dW_1}{dt} &= \Phi(\hat{e})\hat{\mathbf{z}}\,\mathbf{r}^T,\\ \tau_{w2}\frac{dW_2}{dt} &= (\mathbf{z} - \mathbf{z}_2)\mathbf{r}^T P.\end{aligned} \tag{2.5}$$

Intuitively, the first learning rule works like RMHL to quickly minimize the total error, as it uses the error of the total output, $\mathbf{z}$. However, it controls only the $\mathbf{z}_1$ component of $\mathbf{z}$, resulting in $\mathbf{z}_1 + \mathbf{z}_2 \approx \mathbf{f}$. The error between the $\mathbf{z}_2$ component and $\mathbf{f}$ is therefore just $\mathbf{z}_1$, which replaces the error in the second learning rule since $\mathbf{z} - \mathbf{z}_2 = \mathbf{z}_1$. As $\mathbf{z}_2$ approaches $\mathbf{f}$, learning in the exploratory pathway causes $\mathbf{z}_1$ to approach $\mathbf{0}$ in order to keep the total $\mathbf{z}$ correct.

Also, we added one extra component to the SUPERTREX algorithm. Learning transfer from the exploratory pathway is soft thresholded based on total error: if the error grows above this point, the transfer rate is gradually reduced to 0. This means that transfer can occur only if the total combined output of both pathways is correct. In practice, this is true for the entire learning period except for a small initial period while the exploratory pathway is finding a solution. Performance without this addition was similar overall but slightly slower. The exact thresholding rule used was to multiply weight updates to both P and W_2 by $(-0.5 \times \tanh(5 \times 10^5 \times (\bar{e} - (1.5 \times 10^{-3}))) + 0.5)$, which will apply a learning rate factor that decays from near 1 with no error to 0 with errors as they exceed 1.5×10^{-3}.

Note that the learning rule for W_1 is local in the sense that it involves only values of the presynaptic and postsynaptic variables in addition to the error signal, e. The learning rule for W_2 would be local were it not for the computation of P, which is biologically unrealistic. However, P can be replaced by the identity matrix to make the learning rules for SUPERTREX purely local. This slows learning, but the network can still learn to produce target outputs from a one-dimensional error signal (see Hoerzer et al., 2014, and the disrupted learning example below).

In summary, RMHL-like learning in the exploratory pathway uses a one-dimensional error signal, e, to track the target, while FORCE-like learning in the mastery pathway uses the exploratory pathway as a teacher signal until it learns the output and takes over. This models current theories of biological motor learning in which early learning is dominated by dopamine-dependent plasticity in the basal ganglia, which gradually trains parallel cortical pathways as the task becomes well learned.

We next test SUPERTREX on three increasingly difficult motor tasks, comparing its performance to those of FORCE and RMHL.

2.3.1 Task 1: Generating a Known Target Output. We first consider a task in which the goal is to draw a parameterized curve of a butterfly by directly controlling the coordinates of a pen (see Figure 2A). Specifically, the target is given by $\mathbf{f}(t) = (x(t), y(t))$, where $x(t)$ and $y(t)$ parameterize the x- and y-coordinates of a pen that successfully traces out the butterfly. The reservoir output, $\mathbf{z}(t)$, controls the coordinates of the pen, so the goal is to train the weights so that $\mathbf{z}(t)$ closely matches the target, $\mathbf{f}(t)$.

The learning algorithms are first allowed to learn for 10 repetitions of the task. As a diagnostic, the error signals are not computed, and the weights are frozen for a further five repetitions. This provides a way to check the accuracy of the final solution, demonstrating whether the algorithm has converged to an accurate solution. During this testing phase, feedback to the system comes from the true solution (Sussillo & Abbott, 2009). Specifically, $Q\mathbf{z}$ is replaced by $Q\mathbf{f}$ in equation 2.1. This avoids a drift in the phase of the solution that otherwise occurs when weights are frozen. In addition, for SUPERTREX, the exploratory pathway was shut off during these last five repetitions ($\mathbf{z}_1$ set to zero) to test how well the mastery pathway converged.

This simple task is well suited to FORCE, which requires a known target, $\mathbf{f}$, in order to compute the fully supervised error signal,

$$\mathbf{e} = \mathbf{z} - \mathbf{f}.$$

FORCE was able to quickly find the correct solution to the task and maintained the correct result even after weights were frozen (see Figures 2B and 2C). Another measure of convergence is the activity of the weight matrix, W, which quickly converged and then stabilized (see Figure 2Cii, bottom).

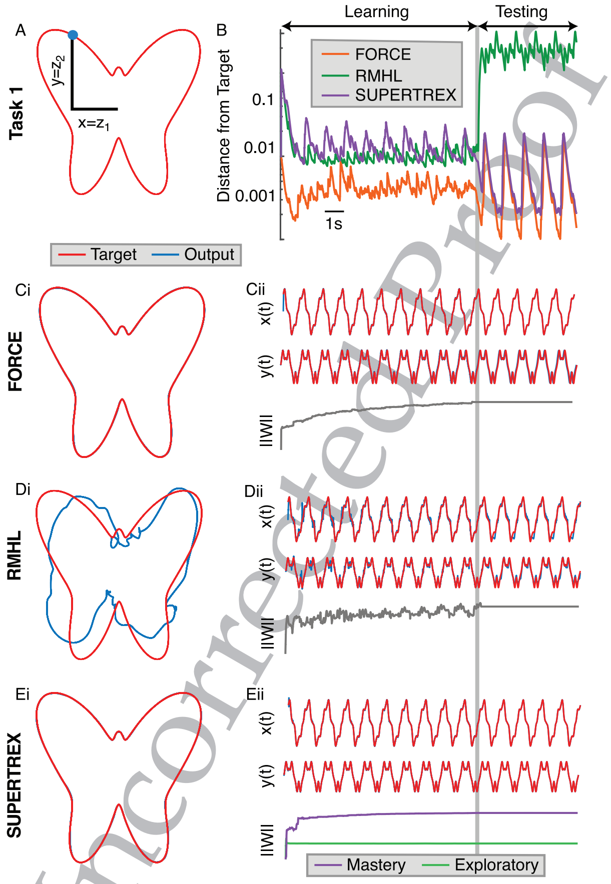

Figure 2: Performance of three learning algorithms on task 1. (A) Task 1 is to draw a butterfly curve by directly controlling the x- and y-coordinates of a pen. Specifically, the output of the reservoir is $\mathbf{z}(t) = (x(t), y(t))$ where $x(t)$ and $y(t)$ are the Cartesian coordinates of the pen. (B) Euclidean distance of the pen from the target for FORCE (orange), RMHL (green), and SUPERTREX (purple). Learning was halted by freezing weights and exploration after 10 periods, so the remaining 5 periods represent a testing phase. (Ci) Target butterfly curve (red) versus the butterfly drawn by FORCE (blue) during the testing phase. (Cii) Target (red) and actual (blue) outputs, $x(t)$ and $y(t)$, and the norm of the weight matrix, $\|(WW^T)^{\frac{1}{2}}\|_2$, produced by FORCE. (Di–ii) Same as panel C, but for RMHL. (Di–ii) Same as panel C, but for SUPERTREX and the norm of each matrix, W_1 (exploratory; green) and W_2 (mastery; purple) are plotted separately. The vertical gray bar indicates the time at which weights were frozen. Note that exploratory weights do change, but primarily at the start, and they are hard to see over the full trial timescale. See Figure 3 for more details.

Error also remained low after learning was disabled (see Figure 2B). In summary, as expected, FORCE learned this task quickly and accurately.

To apply RMHL and SUPERTREX to this task, we set

$$e = \|\mathbf{z} - \mathbf{f}\|^2,$$

where $\| \cdot \|$ is the Euclidean norm—the distance of the pen from its target. This error contains strictly less information than the fully supervised error that FORCE used. RMHL performed well during learning, but the performance after weights were frozen (see Figures 2B and 2D), along with the cyclical changes in $\|W\|$ during learning (see Figure 2Dii) demonstrate that the RMHL algorithm never actually converged. Instead, RMHL relied on rapid changes in W to mimic the correct output at each time point without truly learning it. Even when the number of learning trials was dramatically increased (to 100, not shown), RMHL's W continually oscillated rather than converged.

SUPERTREX performed well on this task. During learning, it performed slightly worse than FORCE and similar to RMHL (see Figures 2B and 2E). Unlike RMHL, though, SUPERTREX continued to track the target after learning was disabled and performed similar to FORCE during that phase (see Figure 2B). This, combined with the apparent convergence of $\|W\|$ during learning (see Figure 2Eii, purple and green curves converge), indicates that the SUPERTREX algorithm did converge, albeit more slowly than FORCE.

Interestingly, SUPERTREX produced less error during the testing phase than during learning (see Figure 2B). This is because exploration introduces random errors during learning, but exploration was turned off during testing so that output was produced only by the well-trained mastery pathway. This is comparable to findings in songbirds in which natural or artificial suppression of neural activity in brain areas homologous to the basal ganglia reduces exploratory song variability and vocal errors (Kao et al., 2005).

From Figure 2, it can be hard to tell whether the exploratory pathway is active, since the weights do not seem to change. This is due to the large timescale of the trial compared to the exploratory-dominated phase, which occurs only as the algorithm is first adjusting to the task. An interesting illustration of the exploration/mastery hand-off in SUPERTREX is provided by suddenly changing the target from a butterfly to a circle during learning (see Figure 3). The relative contributions from the exploratory pathway and the mastery pathway show that the exploratory pathway initially tracks the new target (see Figure 3B). Since the exploratory pathway is equivalent to RMHL, we know that the pathway is only mimicking the output through rapid weight changes. Over time, the mastery pathway learns from the activity of the exploratory pathway and begins taking over the generation of

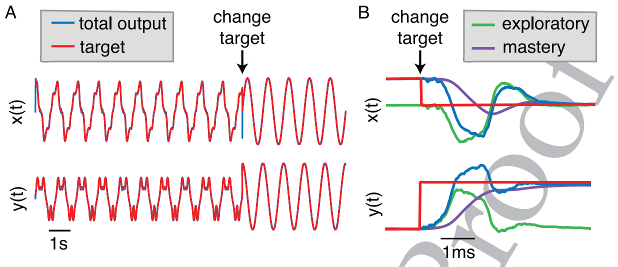

Figure 3: The dynamics of SUPERTREX with an abruptly changed target. (A) Target (red) and actual (blue) output. Same as task 1, but the target was changed from a butterfly to a circle after 10 periods. (B) Detail around the time of change. Same colors as panel A (red for target and blue for total output), with the addition of exploratory (green) and mastery (purple) components of the total output. Note that exploratory + mastery = total output.

the output. This handoff from the exploratory to the mastery pathway produces a damped oscillation around the target (see Figure 3B).

2.3.2 Task 2: Generating an Unknown Target from a Scalar Error Signal. Task 1 is a simple introductory task to compare the three learning algorithms, but it is also unrealistic in some ways that play toward FORCE's strengths. Specifically, the task involves producing an output, $\mathbf{z}$, to match a known target, $\mathbf{f}$, and error is computed in terms of the difference between $\mathbf{z}$ and $\mathbf{f}$. In many tasks, the motor output has indirect effects on the environment and the target and error are given in terms of these indirect effects. For example, consider a human or robot performing a drawing task. Motor output does not control the position of the pen directly, but instead controls the angles of the arm joints, which are nonlinearly related to pen position. On the other hand, error might be evaluated in terms of the distance of the pen from its target. Task 2 models this scenario.

The goal in task 2 is to draw the same butterfly from task 1, parameterized by the same target coordinates $\mathbf{f}(t) = (x(t), y(t))$. However, the reservoir output controls the angles of two arm joints (see Figure 4A),

$$\mathbf{z}(t) = (\theta_1(t), \theta_2(t)).$$

We assume that the subject does not have access to the target angles that draw the butterfly. Instead, they have access only to the target pen coordinates, $\mathbf{f}$, and its distance to the actual pen coordinates, which are related to the angles through a nonlinear function:

$$h(\theta_1, \theta_2) = (x, y).$$

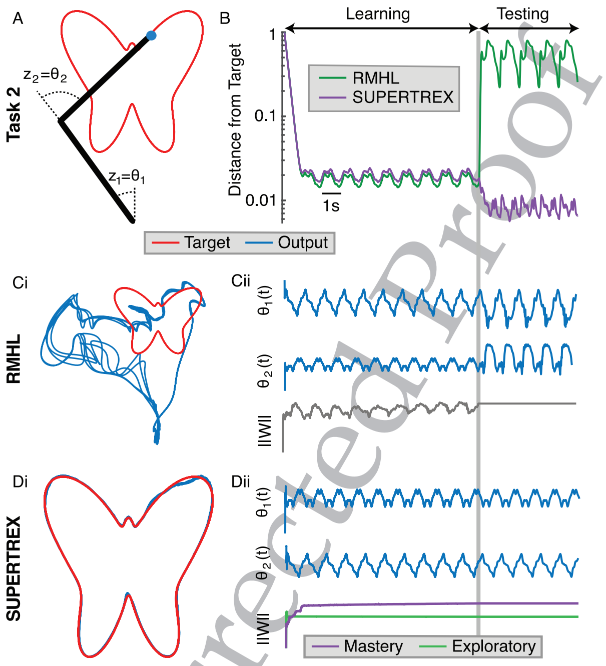

Figure 4: Performance of RMHL and SUPERTREX on task 2. (A) Task 2 is to draw the same butterfly (red) as in task 1, but the reservoir output now controls the arm joint angles, $z_1(t) = \theta_1(t)$ and $z_2(t) = \theta_2(t)$. Error is still computed in terms of pen coordinates. FORCE is not applicable to this task. (B) Euclidean distance of pen from target for RMHL (green) and SUPERTREX (purple). Learning was halted by freezing weights after 10 periods, so the remaining 5 periods represent a testing phase. (C–D) Same as Figures 2D and 2E except that angles, $\theta_1(t)$ and $\theta_2(t)$, are plotted in place of pen coordinates.

FORCE cannot be applied directly to this task since the fully supervised error required for FORCE would need to be computed in terms of target angles instead of target pen position.

RMHL and SUPERTREX can be applied to this task since they require only a signal that provides enough information to determine whether error recently increased or decreased. In particular, this is accomplished by setting

$$e = \|h(\mathbf{z}) - \mathbf{f}\|^2,$$

where $h(\mathbf{z}) = (x, y)$ is the pen position.

Once again, the task is divided into 10 learning cycles and 5 test cycles, with learning algorithms and the exploratory pathway of SUPERTREX disabled during the test cycles. Since the target angles, $\mathbf{z}(t)$, are unknown, feedback during testing cannot be replaced by the target, as was done for task 1. Instead, it is provided by the output from five previous periods.

RMHL performed poorly on this task. It eventually mimicked the target (see Figure 4B), but once again failed to converge (see Figures 2B and 2C). SUPERTREX was able to track the target and continue to produce it even after weight changes ceased (see Figures 4B and 4D). Hence, the combination of FORCE-like learning and RMHL-like learning implemented by SUPERTREX is able to learn a task that neither FORCE nor RMHL can learn on its own.

2.3.3 Task 3: Learning and Optimizing a Task with Multiple Candidate Solutions. While FORCE cannot be applied to task 2 as it is currently defined, it could be applied if the inverse of h were explicitly computed offline to provide the target angles, $(\theta_1, \theta_2) = h^{-1}(\mathbf{f})$, from which to compute a fully supervised error signal. This approach assumes that the subject knows the inverse of h and therefore does not easily extend to learning tasks in which h is difficult or impossible to invert. We now consider a task in which the error is not an invertible function of the motor output.

Specifically, we consider an arm with three joints (see Figure 5A) and a cost function, $C(\theta_1', \theta_2', \theta_3')$, that penalizes the movement of some joints more than others. Here, θ_j' is the time derivative of θ_j. SUPERTREX can work with any penalty structure, making the choice arbitrary. Given that, we decided to loosely model our arm on a real human arm, with the joints corresponding to shoulder, elbow, and wrist. The penalties are larger for the angles controlling larger arm lengths, so the cost is lowest for the wrist joint, θ_3, and largest for the shoulder joint, θ_1, based on the intuition that you are more likely to move your wrist than your entire shoulder and arm for a small reaching task. This can also be seen as an energy conservation principle, with larger costs associated with the more costly shoulder joint compared to the wrist joint.

A recent study (Weiler, Gribble, & Pruszynski, 2015) and its follow-up (Weiler, Saravanamuttu, Gribble, & Pruszynski, 2016) support our intuition. In those studies, human subjects performed a reaching task while subjected to shoulder, elbow, or arm perturbations. One major finding was that perturbing any joint led to other joints compensating for the movement, with overall compensation correlation between different joints, supporting the idea that a single objective is optimized throughout the entire motion rather than independently moving joints. I addition, after a detrimental perturbation, the wrist responded significantly faster than the elbow, which in turn responded faster than the shoulder. In the follow-up, they also find that elbow perturbations lead to nearly no shoulder correction, but a significant wrist and elbow correction, where the wrist connection is larger than the elbow correction.

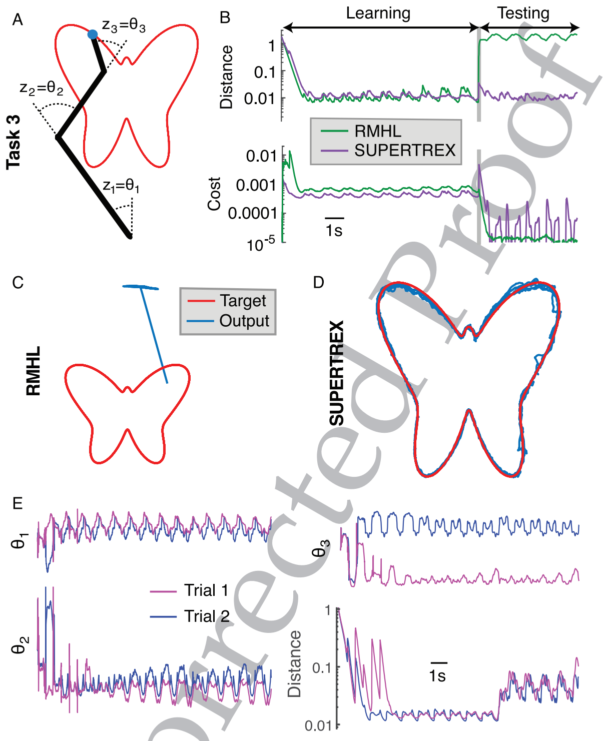

Figure 5: Performance of RMHL and SUPERTREX on task 3. (A) Task 3 is to draw the same butterfly (red) as in tasks 1 and 2, but the reservoir output now controls three arm joint angles, $z_1(t) = \theta_1(t)$, $z_2(t) = \theta_2(t)$, and $z_3(t) = \theta_3(t)$, with a different cost function associated with moving each joint. Error is computed in terms of pen coordinates and the cost of moving joints. FORCE is not applicable to this task. (B) Euclidean distance of pen from target (top) and cost (bottom; $C(\theta_1', \theta_2', \theta_3')$) for RMHL (green) and SUPERTREX (purple). (C,D) Same as Di and Ei in Figure 2. (E) Angular outputs and distance from target across two different SUPERTREX trials. The overall solution found was similar, with a mirrored rotation in one joint angle.

For this task, there are infinitely many candidate solutions that successfully draw the butterfly, differing by the cost of joint movement. This turns our learning task into an optimization problem.

Using FORCE for this task does not make sense, as it would require prior, explicit knowledge of the desired time series of joint angles. Essentially, it would require that the optimization problem had already been solved offline. RMHL and SUPERTREX can be applied to this problem by setting

$$e = \|h(\mathbf{z}) - \mathbf{f}\|^2 + C(\theta_1', \theta_2', \theta_3').$$

In this context, RMHL and SUPERTREX work as greedy search algorithms that make local changes to the angular output to reduce error and cost. Note, however, that the solution they find may not be globally optimal.

We applied RMHL and SUPERTREX to this task using the same protocol for the learning and testing phases that we used for task 2. RMHL performed poorly on this task (see Figures 5B and 5C), which is not surprising considering its poor performance on task 2. SUPERTREX performed much better than RMHL. It was able to track the target, and continue to produce it even after weights were frozen (see Figures 5B and 5D). Over multiple runs SUPERTREX will find different solutions, as seen in Figure 5E. The solution found will primarily depend on the initial condition, but the randomness in searching will also play a role. In this task, SUPERTREX tended to find similar solutions, except for random mirroring of certain angles. In summary, SUPERTREX can solve motor learning tasks in which there are multiple "correct" solutions with different costs.

2.4 Disrupted Learning as a Model of Parkinson's Disease. The design of SUPERTREX was motivated in part by observations about the role of the basal ganglia in motor learning and Parkinson's disease (PD). PD is caused by the death of dopamine-producing neurons in the basal ganglia, resulting in motor impairment. A common treatment for PD is a lesion of basal ganglia output afferents. Such lesions alleviate PD symptoms and impair performance on new learning tasks more than well-learned tasks (Obeso et al., 2009; Turner & Desmurget, 2010). These and other findings have inspired a theory of motor learning in which the basal ganglia are responsible for early learning but not the performance of well-learned tasks and associations (Turner & Desmurget, 2010; Hélie et al., 2012). SUPERTREX is consistent with this theory if the exploratory pathway is interpreted as a basal ganglia pathway and the mastery pathway the cortical pathway. To test this model, we next performed an experiment in SUPERTREX that mimics the effects of PD and its treatment with basal ganglia lesion.

The hand-off of learning from the exploratory to the mastery pathway occurs extremely quickly in SUPERTREX due to the powerful but biologically unrealistic RLS learning rule used in the mastery pathway (see Figure 3). To make SUPERTREX more biologically plausible for this experiment, we replaced the RLS learning rule with a least-mean-squares (LMS) rule by replacing P in equation 2.2 with the identity matrix (Hoerzer et al., 2014). This modified rule is more realistic because it avoids the complicated computation of the matrix, P; it makes the learning rules local; and it causes the mastery pathway to learn more slowly, which slows the hand-off from the exploratory pathway.

We applied this modified SUPERTREX algorithm to task 1. For 100 trials, learning proceeded normally. SUPERTREX learned the target more slowly than in Figure 2 and with a slight degradation in performance due to the use of LMS instead of RLS learning in the mastery pathway (see Figure 6,

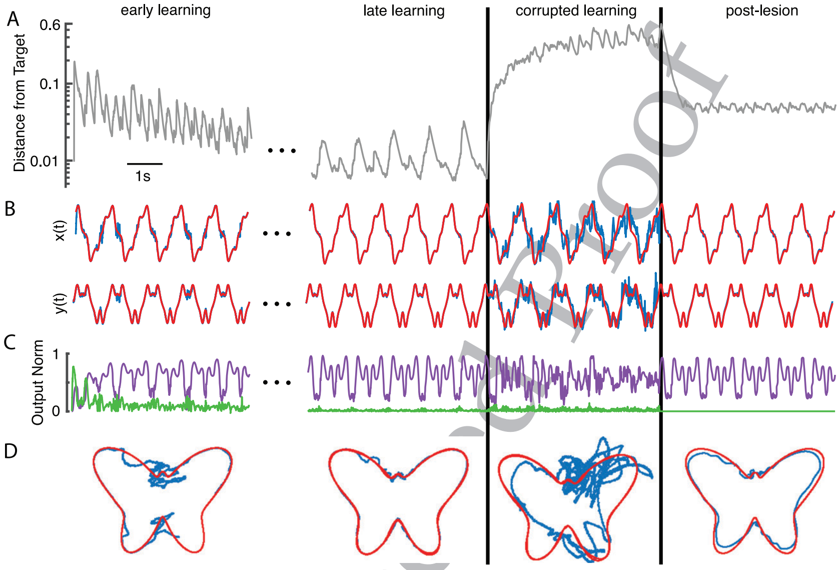

Figure 6: SUPERTREX with a corrupted error signal models Parkinson's disease and its treatment. (A) Euclidean distance of pen from its target for a modified version of SUPERTREX on task 1. Learning proceeded normally for 100 trials. The first 5 trials (early learning) and last 5 trials (late learning) are shown. The error signal was corrupted over the following 5 trials (corrupted learning), and the exploratory pathway was lesioned for the last 5 trials (postlesion). (B) Target (red) and actual (blue) outputs. (C) Normed outputs from mastery pathway (purple) and exploratory pathway (green). (D) Target butterfly (red) versus drawn curve during each of the plotted groups of 5 trials.

early and late learning). This phase models normal learning before the onset of PD. By the end of this phase (see Figure 6, late learning), the task has become "well learned" in the sense that output is generated by the mastery pathway instead of the exploratory pathway. The system output depending primarily on the mastery rather than exploratory pathway can be seen in Figure 6C.

For the next five trials, we corrupted the error signal to model the effects of PD. Since e models the error or cost of motor output, it is negatively related to dopamine release. Specifically, $\hat{e} \propto D_{\max} - D$ where D quantifies dopamine release and $D_{\max}$ is the maximum possible value of D. Hence, PD-induced dopamine depletion is modeled by artificially increasing $\hat{e}$, which we achieve by setting

$$e = \|\mathbf{f} - \mathbf{z}\|^2,$$

$$\hat{e} = \hat{e} + p$$

where $p(t)$ increases over time. Here, $p = 0$ corresponds to a healthy subject, and as p increases, SUPERTREX falsely evaluates more of its actions as being in error or costly. For our Parkinsonian task, we chose a $p(t)$ that linearly increased to .1 over the duration of the corrupted learning phase.

Although the mastery pathway had taken over motor output before the error signal of the exploratory pathway was corrupted, the perceived increase in error caused the exploratory pathway to take over during the corrupted learning phase because the contribution of the exploratory pathway increases with error. Although the actual disruption may seem small (see Figure 6C, where the exploratory activity is similar to that of early learning), the mismatch between actual error and perceived error during the corrupted learning phase results in highly inaccurate motor output (see Figure 6, corrupted learning phase) as activity leaves the learned manifold and is unable to recover. These results model the motor impairments associated with PD. Indeed, PD symptoms are believed to be caused at least in part by aberrant learning in the basal ganglia (Turner & Desmurget, 2010; Ashby et al., 2010).

In the last five trials, we disabled the exploratory pathway, modeling basal ganglia lesion, and the feedback term, $Q\mathbf{z}$, was replaced by $Q\mathbf{f}$ in equation 2.1 (see below and section 3), SUPERTREX recovered nearly correct output during this last stage (see Figure 6, postlesion phase) because the output had been stored in the mastery pathway before learning in the exploratory pathway was corrupted.

As shown in Figure 6, immediately before corruption began, the mastery pathway was essentially solely responsible for generating the correct output. After the Parkinsonian effect, the final output is given solely by the mastery pathway as the malfunctioning exploratory pathway is lesioned. Thus, any degradation in the drawn butterfly is due to harmful changes made to the mastery pathway during the Parkinsonian effect. There are two main reasons that these harmful changes should be small. One is that the exploratory pathway changes are kept only if they result in a decrease in error even after taking into account the additional Parkinsonian error, or that the Parkinsonian error term makes changes due to exploration less likely to be accepted. In addition, for sufficiently large errors, the SUPERTREX component that controls transfer from exploration to mastery pathways shuts down, limiting the degree to which harmful perturbations can be assimilated. Thus, postlesion performance will depend on the specific $p(t)$ Parkinsonian effect used, along with the overall duration of the Parkinsonian effect.

2.5 State Information Promotes Stability of Learned Output. During our previous examples comparing FORCE, RMHL, and SUPERTREX, the comparison was made by allowing 10 trials of training the algorithm and then with 5 trials of the learning algorithm shut off (weights frozen) to see if the method had converged. During this testing phase, feedback was modified. In task 1, it was replaced with the correct output (the target), and for tasks 2 and 3, it was replaced with the output from previous trials during learning, which nearly matched the target due to the learning algorithm being active. This allowed us to check whether an algorithm had converged, in the sense that there would be no further feedback and weight changes. However, providing the correct answer as feedback, also known as teacher forcing, could be considered cheating here. Teacher forcing essentially ignores the stability of the solution and instead checks only whether the system can correctly produce the next time step of the solution given a perfect fit to the current time step. In order to address this, we repeated task 1 but without teacher forcing.

FORCE has previously been shown to perform well in the absence of teacher forcing (Sussillo & Abbott, 2009; Abbott, Depasquale, & Memmesheimer, 2016), but it failed in our simulations (see Figure 7A, solid orange). We suspected that this was due to the extra additive noise, $\boldsymbol{\epsilon}$, during learning. Noise is not typically included in applications of FORCE, but reservoir learning is known to be sensitive to noise and other perturbations (Vincent-Lamarre, Lajoie, & Thivierge, 2016; Sussillo, 2014; Miconi, 2017), which are ubiquitous in biological neuronal networks. Indeed, FORCE performed better when this noise was removed (see Figure 7A, dashed orange). Noise is an inherent part of RMHL and SUPERTREX, so they cannot be tested without it. Unsurprisingly, RMHL and SUPERTREX also perform poorly without teacher forcing (see Figures 7A to 7C). In summary, learning a noisy version of the target prevents all three algorithms from reproducing the target postlearning in the absence of teacher forcing.

We resolve this issue by augmenting the feedback to include full information about the state of the system, allowing the system to self-correct. Specifically, we concatenated the x- and y-coordinates of the target pen position onto the feedback signal, replacing the $Q\mathbf{z}$ term in equation 2.1 with $Q[\mathbf{z}\;\;\mathbf{f}]$, during both training and testing. Under this modified framework, we again tested all three algorithms on task 1 and tested RMHL and SUPERTREX on task 2. For task 1, this change is analogous to teacher forcing (since the target coordinates are the same as the target reservoir output). For task 2, it is distinct from teacher forcing because the feedback is in terms of the Cartesian coordinates of the target, whereas the output must be in terms of arms' angles. Hence, for task 2, the system must learn to self correct. If $\mathbf{z}$ and $\mathbf{f}$ differ, then the networks need to learn how to generate the correct θ_1 and θ_2 to correct the error. This change greatly improved the accuracy of FORCE and SUPERTREX, but not RMHL (see Figures 7D to 7F).

Note that this change is not the same as just providing the correct answer as teacher forcing does. Teacher forcing essentially resets the system to be correct after every time step by replacing $Q\mathbf{z}$ with $Q\mathbf{f}$, preventing drift. The augmented feedback instead provides sufficient information for the system to be autonomously self-correcting and the feedback is provided as is, with no context. In task 2, the algorithm does not have access to the solution it

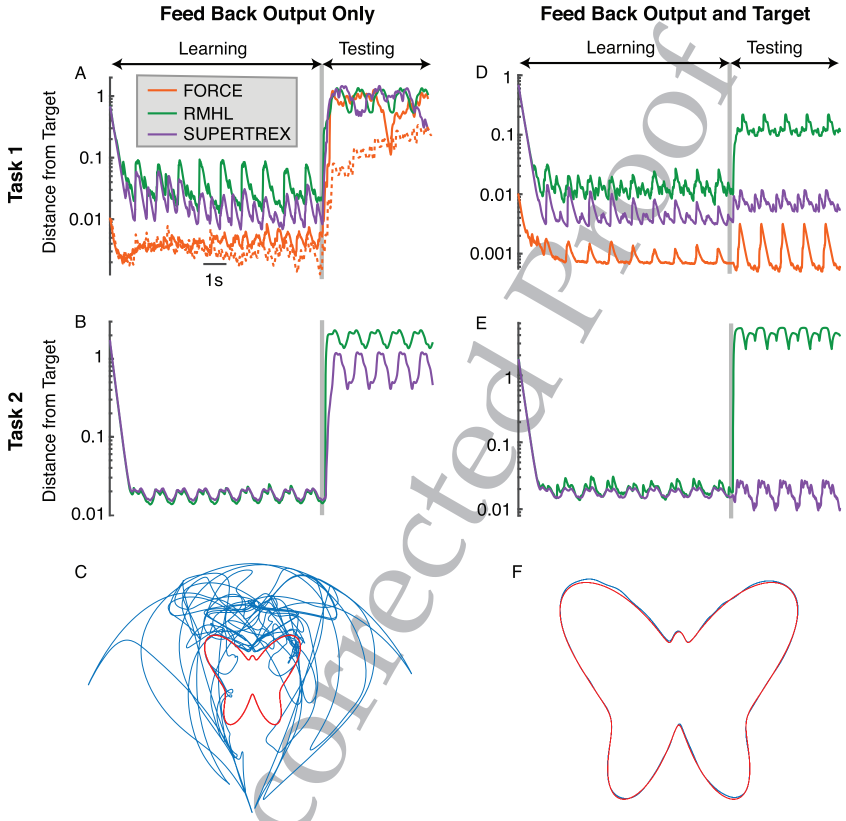

Figure 7: Including target information in feedback promotes stability without teacher forcing. (A) Euclidean distance of the pen from its target for FORCE (orange), RHML (green), and SUPERTREX (purple) on task 1. Same as Figure 2B except feedback during the testing phase was not replaced with the true solution (teacher forcing) but is instead given by $Q\mathbf{z}$ exactly. (B) Same as panel A, but for task 2 and without FORCE (since it cannot be applied to task 2). (C) Butterfly drawn by SUPERTREX from the simulation in panel B. (D,F) Same as panels A to C except feedback was augmented by the target, $Q[x\ y\ \mathbf{f}]$.

must produce (in terms of arm angles); it has access only to the target pen coordinates, which are nonlinearly related to arm angles. This is akin to including a sensory feedback term, where the algorithms have sensory information about the actual and target positions, but do not have explicit information on necessary joint movements to make them overlap. Note that simply replacing the feedback from position to target will not result in convergence; for example, replacing $Q\mathbf{z}$ with $Q\mathbf{f}$ throughout training and testing does not work. Both pieces of information together are required to build a stable system.

The extra feedback term can be simplified further by changing **f** into a simple phase variable, which gives similar results as those shown in Figures 7D to 7F (data not shown). Similar approaches have been proposed previously (Vincent-Lamarre et al., 2016). These approaches can model the presence of time-keeping neural populations. For example, in songbirds, motor learning is believed to be supported by a timekeeping signal from HVC, which is extensively used in models of songbird learning (Doya & Sejnowski, 1995; Fiete, Fee, & Seung, 2007; Fee & Goldberg, 2011).

2.6 Reward-Modulated Learning with Velocity Control. In all examples considered so far, the output of the reservoir controlled the position of a pen or the angle of arm joints. In control problems, motor output controls velocity or acceleration (e.g., applied force) of limbs or joints. From a naive perspective, SUPERTREX should still be able to complete such a task; random perturbations still change error, and SUPERTREX can learn to produce perturbations associated with lower error.

However, a more careful consideration reveals that SUPERTREX and RMHL applied directly to control velocity would not be effective. To understand why, we first review and schematicize how SUPERTREX and RMHL successfully learn task 1, where the output controls the position of the pen, then consider why they would not work when the reservoir output controls the velocity of the pen.

In task 1, suppose the pen is displaced from its target (see Figure 8, top left), and an exploratory perturbation is made to the reservoir output that successfully moves the pen closer to its target (see Figure 8, bottom left). In this case, the change in error is negative ($\Delta e \approx \hat{e} < 0$), so the perturbation is correctly rewarded (see equations 2.4 and 2.5).

Now consider task 1 except that the reservoir output controls the velocity of the pen instead of the position. Again, suppose the pen is displaced from its target, and also suppose that it is moving away from the target (see Figure 8, top middle). A beneficial exploratory perturbation changes the velocity of the pen in the direction of the target (see Figure 8, bottom middle). However, if the perturbation was not strong enough to change the direction of the pen, then the error (which is measured as the distance of the pen from its target) will still have increased after the perturbation (as in Figure 8, bottom middle), so that $\Delta e \approx \hat{e} > 0$, and this perturbation will be penalized instead of rewarded (as again indicated by equations 2.4 and 2.5).

This problem is overcome by taking the derivative of the error, specifically defining e to be the derivative of the distance between the pen and its target. When this change is made, a reservoir controlling pen velocity will be correctly rewarded for beneficial perturbations (see Figure 8, right) and penalized for harmful perturbations.

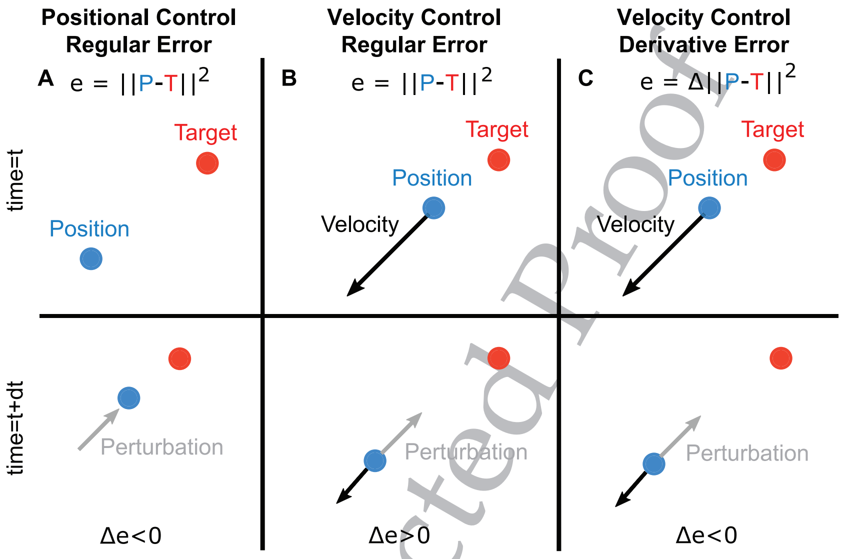

Figure 8: Velocity control in SUPERTREX. (A) When the position of a pen is controlled and error is the distance of the pen from its target, a beneficial perturbation correctly results in a decreased error. (B) When the velocity of the pen is controlled and error is computed in the same way, in some situations, a beneficial perturbation results in increased error. (C) With velocity control, replacing the error by the derivative of the distance causes beneficial perturbations to correctly produce decreased error.

To test these conclusions, we repeated task 1 with SUPERTREX, except with output now corresponding to velocity rather than position,

$$\frac{d[x, y]}{dt} = z_1 + z_2,$$

and we set $[x(0)\ y(0)] = \mathbf{f}(0)$. During the course of training this model, we discovered two other adjustments were required. As our goal was to track a signal rather than reach a target, adding a penalty term based on velocity was helpful in order to prevent oscillations around our target—for example,

$$e = \Delta(\|\mathbf{f} - \mathbf{z}\|^2 + \gamma|dtz|).$$

Unfortunately, we did not find a systematic way to determine γ. Instead, γ is chosen via iteration in order to prevent over- or underdamped oscillatory behavior.

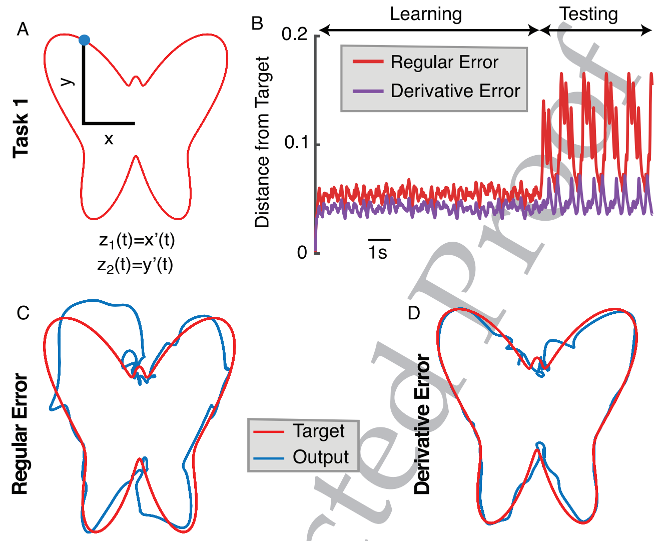

Figure 9: Velocity control in SUPERTREX. (A) A schematic of the velocity control task, which is identical to task 1 except that the velocity of the pen is controlled by the reservoir instead of its position. (B) Using the regular error (distance of pen from target) produces large errors, but using the derivative of the distance produces smaller errors, especially during testing. (C) Butterfly drawn during the testing phase using regular error and (D) derivative error.

Additionally, standard feedback $Q\mathbf{z}$ clearly does not provide enough information. If we do not explicitly know our starting position, knowing only velocity does not help. Instead, we provided full-state information $Q[x\ y\ \mathbf{f}]$ since we care more about our position than our velocity in terms of feedback. This is also more realistic. It makes sense to modify $\mathbf{z}$ based on our position rather than velocity, and position is more likely to be available as sensory feedback. Making these changes, we can compare SUPERTREX with error computed as the distance between the pen and its target (see Figure 9, regular error) and with error computed as the derivative of the distance between the pen and its target (see Figure 9, derivative error). As predicted, SUPERTREX with velocity control performs better when using the derivative of the distance as the error signal (see Figure 9; compare red to purple in panel B, and compare panel C to D).

3 Discussion

We presented a novel, reward-modulated method of reservoir computing, SUPERTREX, that performs nearly as well as fully supervised methods. This is desirable as there is a broad class of problems where traditional supervised methods are not applicable, such as our tasks 2 and 3. Moreover, humans can learn motor tasks from reinforcement signals alone (Izawa & Shadmehr, 2011). In place of a supervised error signal, SUPERTREX bootstraps from a dopamine-like, scalar error signal to a full error signal using rewarded exploration. This serves as an approximate target solution, which is then transferred to a more traditional reservoir learning algorithm. This transfer of learned behavior to a mastery pathway, along with continued rewarded exploration, automatically creates a balanced system where the total output is correct, but the composition shifts over time from exploration to mastery. SUPERTREX performed similar to FORCE on tasks where both were applicable, but also worked well on tasks where FORCE was not applicable. SUPERTREX also outperformed RMHL, a previously developed reward-modulated algorithm, on all tasks we considered.

Unlike RMHL and other reinforcement learning models, SUPERTREX models the complementary roles of cortical and basal ganglia pathways in motor learning. Under this interpretation, dopamine concentrations play the role of the reward signal, and the basal ganglia is the site of the RMHL-like, exploratory learning. Direct intracortical connections would then learn from Hebbian plasticity in the mastery pathway. Consistent with this interpretation, SUPERTREX produces inaccurate motor output when the reward signal is corrupted, modeling dopamine depletion in PD, but recovers the generation of well-learned output when the exploratory pathway is removed, modeling basal ganglia lesions used to treat PD. Hence, SUPERTREX provides a model for understanding the role of motor learning in PD and its treatments.

As models of motor learning, reward-modulated algorithms like SUPERTREX and RMHL assume no knowledge of the relationship between motor output and error. In contrast, fully supervised algorithms like FORCE require perfect knowledge of this relationship. In reality, we learn through some combination of supervisory and reward-modulated error signals (Izawa & Shadmehr, 2011). To account for this, SUPERTREX could potentially be extended to incorporate both one-dimensional reward and higher-dimensional sensory feedback.

The FORCE-like learning algorithm used for the mastery pathway of SUPERTREX is biologistically unrealistic in some ways. The presence of the matrix, P, causes the rule to be nonlocal. However, we showed that SUPERTREX still works when P is removed to implement a local LMS learning rule (see Figure 6). Indeed, one can replace the mastery pathway with any supervised learning rule. This could open the way for an implementation of SUPERTREX with spiking neural networks using existing supervised learning rules (Maass et al., 2002; Bourdoukan & Deneve, 2015; Abbott et al., 2016; Pyle & Rosenbaum, 2017). In order to have a fully spiking-based version of SUPERTREX, this would also require a spike-based reinforcement learning rule, most likely an eligibility-trace based rule (Seung, 2003; Xie & Seung, 2004; Fiete & Seung, 2006; Miconi, 2017).

As with most other reservoir computing algorithms, SUPERTREX implements online learning in which a local error signal is provided and used at every time step. This is partly by design; SUPERTREX learns extremely (even unrealistically) quickly as weights are updated at a high frequency. This learning is slowed to some extent by switching to the more realistic LMS learning rule (as in Figure 6). For some biological learning tasks, however, error signals are temporally sparse or reflect temporally nonlocal information. Trial-based learning rules for reservoir computing (Fiete & Seung, 2006; Miconi, 2017) are applicable in the presence of sparse or nonlocal rewards. At least one of these algorithms learns very slowly, requiring thousands of trials (Miconi, 2017), which may be an inevitable consequence of learning from sparse rewards. In reality, biological motor learning likely makes use of both online and sparse feedback. An extension of SUPERTREX that accounts for both types of feedback could be more versatile and realistic.

SUPERTREX is conceptually an extension of SPEED (Ashby et al., 2007), which has a similar framework for categorization and other discrete tasks. SPEED learns to map arbitrary discrete inputs to discrete outputs, such as in categorization tasks. While the architecture and learning rule are similar to SUPERTREX, SPEED cannot produce continuous, dynamical output and requires a separate pathway for each possible input-output pairing.

SUPERTREX could also be compared to a class of RNN algorithms that use a teacher network to train the final output network. However, many of these networks use the activity of the teacher network as a way to train the recurrence *J* of the output network; in SUPERTREX, there is only one recurrent network (used for both outputs). These methods are often even more biologically implausible; for example, the recent FULL-FORCE extension of FORCE (DePasquale et al., 2018) feeds the target signal information into the first, chaotic reservoir, and then uses the activities of each reservoir unit in the teacher network as a target for training the second network, drastically increasing the amount of supervision required.

SUPERTREX loses accuracy when learning is halted when feedback consists solely of the system's output (see Figures 7A to 7C) due to the fact that it learns from a noisy estimate of the target. This shortcoming can be overcome by augmenting the feedback with the target, allowing the system to learn to self-correct noise-induced errors (see Figures 7D to 7F). FORCE is susceptible to the same instabilities as SUPERTREX under the biologically realistic assumption of noise during learning (see Figure 7A), but SUPERTREX can solve tasks that FORCE cannot (see Figures 4 and 5). RMHL is also susceptible to the same instabilities and is applicable to the same tasks as SUPERTREX, but the instabilities in RMHL are not resolved by including target information in the feedback as they are for SUPERTREX (see Figures 7C and 7D). Hence, SUPERTREX is the only one of the three algorithms that can be applied to reward-modulated learning tasks and achieves stability with target information in the feedback.

Stability in reward-modulated reservoir computing without target information in the feedback term remains an open problem. This problem could potentially be solved by providing external input in-phase with the target output. This could help the reservoir "keep time" by realigning the reservoirs' state on each trial, allowing the system to self-correct its phase. A similar approach was shown to improve robustness of FORCE to perturbations in previous work (Vincent-Lamarre et al., 2016).

Interestingly, biology may have already solved this problem. Research by Toledo-Suarez, Duarte, and Morrison (2014) has found that the striatum may act as a reservoir computer that processes state information. Rather than rely on raw inputs, the motor learning system instead has access to preprocessed state information that is both simpler and more relevant. In SUPERTREX, this could correspond to replacing our simple feedback of raw state information Qz or $Q[z\,f]$ with Qs, where s is a preprocessed state information vector. s could even come from another reservoir, designed to ensure s contains maximally relevant information to the task at hand. This would be an interesting extension to SUPERTREX.

In summary, SUPERTREX is a new biologically inspired framework for reservoir computing that is more realistic and more effective than its predecessors. Using a general error signal allows it to be used in places where a more powerful algorithm like FORCE cannot. The hand-off from exploration to mastery allows SUPERTREX to perform nearly as well as FORCE with the generality of reward-modulated algorithms. Moreover, SUPERTREX offers a computational formalization of widely supported theories of motor learning and reproduces several experimental and clinical findings. Hence, this new framework opens the way for truly two-way communication between biological and computational theories of motor learning.

4 Materials and Methods

4.1 Simulation and Reservoir Parameters. All simulations were performed using a forward Euler method, with $dt = 0.2$ ms. Each task period or "trial" was 10^4 ms long, and all simulations except those in Figure 6 had 15 trials. Figure 6 had 110 trials.

The reservoir equation used in all algorithms was

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + J\mathbf{r} + Q\mathbf{z},$$

where $\mathbf{r} = \tanh(\mathbf{x}) + \alpha\boldsymbol{\eta}$, $\boldsymbol{\eta}$ was uniformly drawn from $[-1, 1]$ on every time step, τ =10, and $\alpha = 2.5 \times 10^{-2}$ during training and $\alpha = 0$ during testing. Reservoir size was set to $N = 1000$ neurons, with connection probability $p = 0.1$. Connection strengths in J were normally distributed with mean 0 and variance $\lambda^2/(pN)$ with $\lambda = 1.5$. Feedback Q was dense, with weights uniformly between −1 and 1. Initial readout weights for RMHL and SUPERTREX exploratory pathways, as well as weights for FORCE and the SUPERTREX mastery pathways, were initialized at 0. Initial voltages were set uniformly between −0.5 and 0.5, while initial rates were the hyperbolic tangent of initial voltages. Displayed outputs and errors were low-pass-filtered according to

$$\tau_{MSE}\frac{d\overline{MSE}(t)}{dt} = -\overline{MSE}(t) + MSE(t)$$

$$\tau_{bar}\frac{d\overline{\mathbf{z}}(t)}{dt} = -\overline{\mathbf{z}}(t) + \mathbf{z}(t),$$

where $\tau_{MSE} = 1000$, $\tau_{bar} = 10$, and $\overline{x}$ represents a low-pass-filtered version of the variable x. The plotted distance from target was computed as $\sqrt{\overline{MSE}}$, where $MSE(t)$ is the squared distance of the pen from its target.

4.2 FORCE.

Reservoir output was $\mathbf{z} = W\mathbf{r}$, and the learning rule is

$$\tau_w\frac{dW}{dt} = -[\mathbf{z} - \mathbf{f}]\mathbf{r}^T P,$$

with $\tau_w = 0.02$. The matrix P is a running estimate of the inverse of the correlation matrix of rates $\mathbf{r}$, initialized to

$$P(0) = \frac{1}{\gamma}I$$

and updated according to

$$\tau_p\frac{dP}{dt} = -\frac{P\mathbf{r}\,\mathbf{r}^T P}{1 + \mathbf{r}^T P\mathbf{r}},$$

where $\tau_P = dt$, $\gamma = 10$ is a constant and I is the identity matrix. The matrix P is updated only every 10 time steps in order to save on computing time.

4.3 RMHL.

For RMHL, outputs were given by

$$\mathbf{z} = W\mathbf{r} + \Psi(e)\boldsymbol{\eta},$$

and the learning rule was

$$\tau_w\frac{dW}{dt} = \Phi(\hat{e})(\hat{\mathbf{z}})\mathbf{r}^T,$$

where $\tau_w = 0.02$, η is uniformly distributed noise between $[-1, 1]$, and the high-pass-filtered version, $\hat{x}$, of variable x was computed as

$$\begin{aligned} \tau \frac{d\overline{x}}{dt} &= -\overline{x} + x, \\ \hat{x} &= x - \overline{x}, \end{aligned}$$

with $\tau = 1$ used for all tasks and trials.

4.4 SUPERTREX. Updates to P were identical to the method used in FORCE. Relevant other changes are

$$\begin{aligned} \mathbf{z}_1 &= W_1\mathbf{r} + \Psi(e)\boldsymbol{\eta}, \\ \mathbf{z}_2 &= W_2\mathbf{r}, \\ \mathbf{z} &= \mathbf{z}_1 + \mathbf{z}_2, \end{aligned}$$

for $\boldsymbol{\eta}$ uniformly drawn from $[-1, 1]$. For the learning algorithm,

$$\begin{aligned} \tau_w \frac{dW_1}{dt} &= \Phi(\hat{e})\hat{\mathbf{z}}\mathbf{r}^T, \\ \tau_w \frac{dW_2}{dt} &= -k\mathbf{z_1}\mathbf{r}^T P, \end{aligned}$$

with τ_w remaining 0.02 and constant learning rate k which varies by task. Finally, an extra condition was imposed on updates to P, W_2. Both updates were multiplied by $(-0.5 \times \tanh(5 \times 10^5 \times (\overline{e} - (1.5 \times 10^{-3}))) + 0.5)$, which acts as a soft threshold around $e = 1.5 \times 10^{-3}$. Effectively, for errors larger than this, the mastery pathway would not activate. Performance was similar, but slightly slower, without this thresholding.

4.5 Tasks. In all tasks, the target was to draw a butterfly, given by a polar curve $x(t) = r(t)\cos(t)$ and $y(t) = r(t)\sin(t)$ where

$$\begin{aligned} r(t) = c[&9 - \sin(qt) + 2\sin(3qt) \\ &+ 2\sin(5qt) - \sin(7qt) + 3\cos(2qt) - 2\cos(4qt)] \end{aligned}$$

and $c = 1/\max_t[r(t)]$ is a normalizing constant. For a single repetition, t went from 0 to 10^4 ms, and $q = \frac{2\pi}{10^4}$ scales the system such that qt goes from 0 to 2π over the duration.

In task 2, the task is instead to draw a butterfly by controlling two angles, representing radians from the y-axis and radians from the first joint. The arm is positioned at $(0, -2)$, and each arm segment has fixed length of 1.8. $h(\mathbf{z})$ is therefore

$$h(\mathbf{z}) = \begin{bmatrix} 1.8\sin(z_1\pi) + 1.8\sin((z_1+z_2)\pi) \\ -2 + 1.8\cos(z_1\pi) + 1.8\cos((z_1+z_2)\pi) \end{bmatrix}.$$

In task 3, there are now three angles to control. The arm is positioned at $(0, -2)$, and each arm segment has fixed length. The first segment has length 1.8, the next 1.2, and the final .6. $h(\mathbf{z})$ is therefore

$$h(\mathbf{z}) = \begin{bmatrix} 1.8\sin(z_1\pi) + 1.2\sin((z_1+z_2)\pi) + .6\sin((z_1+z_2+z_3)\pi) \\ -2 + 1.8\cos(z_1\pi) + 1.2\cos((z_1+z_2)\pi) + .6\cos((z_1+z_2+z_3)\pi) \end{bmatrix}.$$

In the first task, $\Psi(x) = 0.025 \times \sqrt[4]{10x}$ and $\Phi(x) = -5\sqrt[4]{x}$ for both RMHL and SUPERTREX. When testing for swapping targets, $\Psi(x) = 0.1 \times (-5x)^{0.3}$ and $\Phi(x) = 2.5 \times \sqrt[4]{x}$. For SUPERTREX, learning rate k was 0.5. For the second task, SUPERTREX learning rate k was still 0.5, $\Psi(x) = 0.01 \times \sqrt[5]{10x}$, and $\Phi(x) = 5 \times \sqrt[4]{x}$. For the third task, SUPERTREX learning rate was $k = 0.9$, $\Psi(x) = .025 \times \sqrt[4]{10x}$, and $\Phi(x) = 5 \times \sqrt[4]{x}$. The error metric was changed slightly, to

$$e = ||h(\mathbf{z}) - \mathbf{f}||_2 + \alpha|\hat{\mathbf{z}}_1| + \beta|\hat{\mathbf{z}}_2| + \gamma|\hat{\mathbf{z}}_3|,$$

for $\alpha = 0.1$, $\beta = 0.05$, $\gamma = 0$. This implemented an additional cost for moving joints—highest for the longest arm segment and 0 for the smallest arm segment.

For the corrupted learning example, LMS learning was used, which is obtained by setting $P = I$. The learning rate was changed to $k = 0.003$. Note that LMS learning rather than RMS learning generally requires a much lower learning rate. Other parameter values were the same as in the first task. The perturbation, $p(t)$, increased linearly from 0 to 0.1 over the corrupted learning time frame.

For the velocity-controlled example in Figure 8, more significant changes were needed. As detailed,

$$[x, y](0) = f(0)$$

and

$$\frac{d[x, y]}{dt} = \mathbf{z_1} + \mathbf{z_2},$$

as well as using full state feedback $Q[x\ y\ \mathbf{f}]$. Learning rate k was .025, as smaller velocities were needed relative to direct control of output. Velocity penalty $\gamma = .3$. Ψ and Φ were the same as in task 1. Finally, we changed how we calculated $\hat{e}$. Rather than use a high-pass filter as a crude derivative estimator, we instead used a finite difference approximation $\hat{e} = e(t) - e(t - dt)$. Note that, as described above, $e(t)$ now refers to $\Delta d(t) = d(t) - d(t - dt)$ where $d(t) = \|\mathbf{f} - \mathbf{z}\|^2 + \gamma |dtz|$—for example, the squared Euclidean distance between the position and target plus a penalty term. Thus, our total update metric $\hat{e} = d(t) - 2d(t - dt) + d(t - 2dt)$, or the finite difference approximation to the second derivative of our error metric, which is Euclidean distance plus penalty.

Acknowledgments

This work was supported by National Science Foundation grants DMS-1517828, DMS-1654268, and DBI-1707400. We thank Jonathan Rubin, Robert Turner, and Robert Mendez for helpful comments.

References

Abbott, L. F., Depasquale, B., & Memmesheimer, R.-m. (2016). Building functional networks of spiking model neurons. *Nat. Neurosci.*, *19*(3), 1–16.

Andalman, A. S., & Fee, M. S. (2009). A basal ganglia–forebrain circuit in the songbird biases motor output to avoid vocal errors. *Proc. Natl. Acad. Sci. U.S.A., 106*(30), 12518–12523.

Aronov, D., Andalman, A., & Fee, M. (2008). A specialized forebrain circuit for vocal babbling in the juvenile songbird. *Science*, *320*, 630–635.

Ashby, F. G., Ennis, J. M., & Spiering, B. J. (2007). A neurobiological theory of automaticity in perceptual categorization. *Psychol. Rev., 114*(3), 632–656.

Ashby, F. G., Turner, B. O., & Horvitz, J. C. (2010). Cortical and basal ganglia contributions to habit learning and automaticity. *Trends Cog. Sci.*, *14*(5), 208–215.

Bottjer, S. W., Miesner, E. A., & Arnold, A. P. (1984). Forebrain lesions disrupt development but not maintenance of song in passerine birds. *Science, 224*(4651), 901–903.

Bourdoukan, R., & Deneve, S. (2015). Enforcing balance allows local supervised learning in spiking recurrent networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, 28* (pp. 982–990). Red Hook, NY: Curran.

Brainard, M. S. (2004). Contributions of the anterior forebrain pathway to vocal plasticity. *Ann. NY Acad. Sci.*, *1016*(1), 377–394.

Brainard, M. S., & Doupe, A. (2000). Interruption of a basal ganglia–forebrain circuit prevents plasticity of learned vocalizations. *Nature*, *404*, 762–766.

Brainard, M. S., & Doupe, A. (2002). What songbirds teach us about learning. *Nature*, *417*, 351–358.

Carelli, R. M., Wolske, M., & West, M. O. (1997). Loss of lever press–related firing of rat striatal forelimb neurons after repeated sessions in a lever pressing task. *J. Neurosci.*, *17*(5), 1804–1814.

Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Foster, J. D., Nuyujukian, P., Ryu, S. I., & Shenoy, K. V. (2012). Neural population dynamics during reaching. *Nature, 487*(7405), 51–56.

DePasquale, B., Cueva, C. J., Rajan, K., Escola, G. S., & Abbott, L. (2018). Full-force: A target-based method for training recurrent networks. *PloS One*, *13*(2), e0191527.

Doya, K., & Sejnowski, T. J. (1995). A novel reinforcement model of birdsong vocalization learning. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 101–108). Cambridge, MA: MIT Press.

Fee, M. S. (2014). The role of efference copy in striatal learning. *Curr. Opin. Neurobiol., 25*, 194–200.

Fee, M. S., & Goldberg, J. H. (2011). A hypothesis for basal ganglia–dependent reinforcement learning in the songbird. *Neuroscience*, *198*, 152–170.

Fiete, I. R., Fee, M. S., & Seung, H. S. (2007). Model of birdsong learning based on gradient estimation by dynamic perturbation of neural conductances. *J. Neuropysiol.*, *98*(4), 2038–2057.

Fiete, I. R., & Seung, H. S. (2006). Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Phys. Rev. Lett.*, *97*(4), 048104.

Hélie, S., Paul, E. J., & Ashby, F. G. (2012). A neurocomputational account of cognitive deficits in Parkinson's disease. *Neuropsychologia, 50*(9), 2290–2302.

Hennequin, G., Vogels, T. P., & Gerstner, W. (2014). Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron, 82*(6), 1394–13406.

Hoerzer, G. M., Legenstein, R., & Maass, W. (2014). Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cereb. Cort.*, *24*(3), 677–690.

Izawa, J., & Shadmehr, R. (2011). Learning from sensory and reward prediction errors during motor adaptation. *PLoS Comput. Biol.*, *7*(3), 1–11.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science, 304*(5667), 78–80.

Kao, M. H., Doupe, A. J., & Brainard, M. S. (2005). Contributions of an avian basal ganglia–forebrain circuit to real-time modulation of song. *Nature*, *433*(7026), 638–643.

Kawai, R., Markman, T., Poddar, R., Ko, R., Fantana, A. L., Dhawale, A. K., . . . Ölveczky, B. P. (2015). Motor cortex is required for learning but not for executing a motor skill. *Neuron*, *86*(3), 800–812.

Laje, R., & Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nat. Neurosci., 16*(7), 925–933.

Lukoševičius, M., Jaeger, H., & Schrauwen, B. (2012). Reservoir computing trends. *Künstliche Intelligenz*, *26*(4), 365–371.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput., 14*(11), 2531–2560.

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature, 503*(7474), 78–84.

Miconi, T. (2017). Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *E-Life, 6*.

Miyachi, S., Hikosaka, O., & Lu, X. (2002). Differential activation of monkey striatal neurons in the early and late stages of procedural learning. *Exp. Brain Res.*, *146*(1), 122–126.

Miyachi, S., Hikosaka, O., Miyashita, K., Kárádi, Z., & Rand, M. K. (1997). Differential roles of monkey striatum in learning of sequential hand movement. *Exp. Brain Res.*, *115*(1), 1–5.

Obeso, J. A., Jahanshahi, M., Alvarez, L., Macias, R., Pedroso, I., Wilkinson, L., . . . Rothwell, J. C. (2009). What can man do without basal ganglia motor output? The effect of combined unilateral subthalamotomy and pallidotomy in a patient with Parkinson's disease. *Exp. Neurol.*, *220*(2), 283–292.

Olveczky, B. P., Andalman, A. S., & Fee, M. S. (2005). Vocal experimentation in the juvenile songbird requires a basal ganglia circuit. *PLoS Biol.*, *3*(5), e153.

Ölveczky, B. P., Otchy, T. M., Goldberg, J. H., Aronov, D., & Fee, M. S. (2011). Changes in the neural control of a complex motor sequence during learning. *J. Neurophysiol.*, *106*(1), 386–397.

Pasupathy, A., & Miller, E. K. (2005). Different time courses of learning-related activity in the prefrontal cortex and striatum. *Nature*, *433*(7028), 873–876.

Poldrack, R. A., Sabb, F. W., Foerde, K., Tom, S. M., Asarnow, R. F., Bookheimer, S. Y., & Knowlton, B. J. (2005). The neural correlates of motor skill automaticity. *J. Neurosci.*, *25*(22), 5356–5364.

Pyle, R., & Rosenbaum, R. (2017). Spatiotemporal dynamics and reliable computations in recurrent spiking neural networks. *Phys. Rev. Lett.*, *118*(1), 018103.

Russo, A. A., Bittner, S. R., Perkins, S. M., Seely, J. S., London, B. M., Lara, A. H., . . . Churchland, M. M. (2018). Motor cortex embeds muscle-like commands in an untangled population response. *Neuron*, *97*(4), 953–966.

Seung, H. S. (2003). Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, *40*(6), 1063–1073.

Shenoy, K. V., Sahani, M., & Churchland, M. M. (2013). Cortical control of arm movements: A dynamical systems perspective. *Annu. Rev. Neurosci.*, *36*, 337–359.

Sompolinsky, H., Crisanti, A., & Sommers, H. J. (1988). Chaos in random neural networks. *Phys. Rev. Lett.*, *61*(3), 259–262.

Sussillo, D. (2014). Neural circuits as computational dynamical systems. *Curr. Opin. Neurobiol.*, *25*, 156–163.

Sussillo, D., & Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, *63*(4), 544–557.

Sussillo, D., Churchland, M. M., Kaufman, M. T., & Shenoy, K. V. (2013). A neural network that finds naturalistic solutions for the production of muscle activity. *Nat. Neurosci.*, *18*(7), 1025–1033.

Tang, C. C., Root, D. H., Duke, D. C., Zhu, Y., Teixeria, K., Ma, S., . . . West, M. O. (2009). Decreased firing of striatal neurons related to licking during acquisition and overtraining of a licking task. *J. Neurosci.*, *29*(44), 13952–13961.

Toledo-Suárez, C., Duarte, R., & Morrison, A. (2014). Liquid computing on and off the edge of chaos with a striatal microcircuit. *Frontiers in Computational Neuroscience*, *8*, 130.

Turner, R. S., & Desmurget, M. (2010). Basal ganglia contributions to motor control: A vigorous tutor. *Curr. Opin. Neurobiol.*, *20*(6), 704–716.

Vincent-Lamarre, P., Lajoie, G., & Thivierge, J.-P. (2016). Driving reservoir models with oscillations: A solution to the extreme structural sensitivity of chaotic networks. *J. Comput. Neurosci.*, *41*(3), 305–322.

Weiler, J., Gribble, P. L., & Pruszynski, J. A. (2015). Goal-dependent modulation of the long-latency stretch response at the shoulder, elbow and wrist. *American Journal of Physiology—Heart and Circulatory Physiology*, *114*(6), 3242–3254.

Weiler, J., Saravanamuttu, J., Gribble, P. L., & Pruszynski, J. A. (2016). Coordinating long-latency stretch responses across the shoulder, elbow and wrist during goal-directed reaching. *American Journal of Physiology—Heart and Circulatory Physiology*, *116*(5), 2236–2249.

Xie, X., & Seung, H. S. (2004). Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, *69*(4), 041909.

Received December 7, 2018; accepted February 24, 2019.