Flexible decision-making in recurrent neural networks trained with a biologically plausible rule

Thomas Miconi
The Neurosciences Institute
La Jolla, CA 92037, USA
miconi@nsi.edu

Abstract

Neural activity during cognitive tasks exhibits complex dynamics that flexibly encode task-relevant variables. Recurrent neural networks operating in the near-chaotic regime, which spontaneously generate rich dynamics, have been proposed as a model of cortical computation during cognitive tasks. However, existing methods for training these networks are either biologically implausible, and/or require a continuous, real-time error signal to guide the learning process. The lack of a biological learning method currently restricts the plausibility of recurrent networks as models of cortical computation. Here we show that a biologically plausible learning rule can train such recurrent networks, guided solely by delayed, phasic rewards at the end of each trial, for nontrivial tasks. We use this method to learn various tasks from the experimental literature, showing that this learning rule can successfully implement flexible associations, memory maintenance, nonlinear mixed selectivities, and coordination among multiple outputs. We show that the resulting networks exhibit complex dynamics previously observed in animal cortex, such as dynamic encoding and maintenance of task features, switching from stimulus-specific to response-specific representations, and selective integration of relevant input streams. We conclude that recurrent neural networks offer a plausible model of cortical dynamics during both learning and performance of flexible behavior.

Introduction

Recent evidence suggests that neural representations are highly dynamic, encoding multiple aspects of tasks, stimuli, and commands in the joint fluctuating activity of interconnected populations of neurons, rather than in the stable activation of specific neurons (Meyers et al., 2008; Barak et al., 2010; Churchland et al., 2012; Stokes et al., 2013; Raposo et al., 2014). Models based on recurrent neural networks (RNN), operating in the near-chaotic regime, seem well-suited to capture similar dynamics (Jaeger, 2001; Maass et al., 2002; Buonomano and Maass, 2009; Sussillo and Abbott, 2009). For this reason, such models have been used to investigate the mechanisms by which neural populations solve various computational problems, including working memory (Barak et al., 2013; Rajan et al., 2016), motor control (Laje and Buonomano, 2013; Hennequin et al., 2014; Sussillo et al., 2015), and selective evidence accumulation (Mante et al., 2013).

However, the methods commonly used to train these recurrent models are generally not biologically plausible. The most common training methods are based on supervised learning, in which a non-biological algorithm (usually a form of backpropagation or regression) minimizes the difference between the network's output and a target output signal (Pearlmutter, 1995; Jaeger, 2001; Sussillo and Abbott, 2009; Rajan et al., 2016; Song et al., 2016). Besides the non-biological nature of these algorithms, the requirement for a constant supervisory signal is in stark contrast with most behavioral tasks, in which the only source of information about performance are time-sparse rewards that are usually delayed with regard to the actions that caused them.

A more biologically plausible form of learning is reward-modulated Hebbian learning: during ongoing activity, each synapse accumulates a *potential* weight change according to classical Hebbian learning, by multiplying pre- and post-synaptic activities at any time and accumulating this product over time. These potential weight changes are then multiplied by a global reward

signal, which determines the *actual* weight changes. This method, inspired by the effect of dopamine on synaptic plasticity, has been successfully demonstrated and analyzed in feedforward or weakly connected spiking (Florian, 2007; Izhikevich, 2007; Frémaux et al., 2010) and firing-rate (Soltoggio and Steil, 2013) networks. However, simple reward-modulated Hebbian learning does not work for strongly-connected recurrent networks that can generate complex trajectories of the type discussed here (Fiete et al., 2007) (see also Appendix).

Recently, Hoerzer, Legenstein and colleagues (Legenstein et al., 2010; Hoerzer et al., 2014) proposed a form of reward-modulated Hebbian learning that can successfully train recurrent neural networks. This method builds upon the so-called *node-perturbation* method (Fiete and Seung, 2006; Fiete et al., 2007). Node-perturbation consists in applying small perturbations to neural activity, then calculating weight changes by multiplying the "normal" (non-perturbation) inputs by the perturbations, and then multiplying the result by a reward signal (interestingly, this method is largely similar to the REINFORCE algorithm, which is widely used in reinforcement learning (Williams, 1992; Peters and Schaal, 2008; Kober et al., 2013; Mnih et al., 2014)). A problem with this method is that it is non-Hebbian (since it multiplies two types of inputs, rather than pre- and post-synaptic activities) and requires information that is not local to the synapse (namely, the perturbatory inputs, which must somehow be kept separate from the "normal" inputs). Legenstein and colleagues showed that, under certain conditions, this method could be made more biologically plausible and Hebbian by leveraging moment-to-moment fluctuations in post-synaptic activity: by keeping a running average of recent activity and subtracting it from the current instantaneous response at any time, we obtain a "high-pass" filtered trace of post-synaptic activity, which can be used as a proxy for the exploratory perturbations of post-synaptic activity. If this signal is then multiplied by the pre-synaptic inputs, and the final accumulated product is then modulated by a reward signal, the node-perturbation method has been recreated in a more biologically plausible, Hebbian manner (i.e. as a product of pre-synaptic and post-synaptic activities rather than between two input sources) (Legenstein et al., 2010). Hoerzer and colleagues showed that this method could be successfully applied to train chaotic recurrent neural networks (Hoerzer et al., 2014).

However, this method critically requires an instantaneous, real-time continuous reward signal to be provided at each point in time. The continuous, real-time reward signal is necessary to allow the subtraction method to extract task-relevant information and counter the effect of spurious deviations introduced by the running-average-subtraction process (see Appendix). Obtaining such an instantaneous reward signal is difficult for most tasks. Furthermore, this requirement for a continuous, real-time reward signal negates the major advantage of reinforcement learning -- the ability to learn from sparse, delayed rewards.

In summary, to our knowledge, there is currently no biologically plausible learning algorithm that can successfully train chaotic recurrent neural networks with realistic reward regimes. This limitation may restrict the potential plausibility of recurrent neural networks as operational models of actual cortical networks.

Here we introduce a novel reward-modulated Hebbian learning rule that can be used to train recurrent networks for flexible behaviors, with reward occurring in a delayed, one-time fashion after each trial, as in most animal training paradigms. This method is Hebbian and uses only synapse-local information, without requiring instantaneous reward signals (see Methods). We apply our method to several tasks that require flexible associations, memory maintenance, and coordination among multiple outputs. By investigating the network's representation of task-relevant aspects over time, we find that trained networks exhibit dynamic coding, as observed in recordings of animal frontal cortices (Meyers et al., 2008; Jun et al., 2010; Stokes et al., 2013). Furthermore, the system shows a shift from stimulus-dominated to decision-dominated encodings, flexibly "routing" stimulus representations to the adequate decision state, in accordance with physiological observations (Stokes et al., 2013). The networks can also learn to flexibly select one of two incoming input streams, reproducing the behavior described in (Mante et al., 2013), but with a biologically plausible plasticity rule rather than Hessian-free supervised training. Finally, we show that the model can learn to control a musculoskeletal model of the human arm for input-guided reaching movements, a task that

requires coordinating 16 muscle outputs. We conclude that reward-modulated Hebbian learning offers a plausible model of cortical learning, capable of building networks that dynamically represent and analyze stimuli and produce flexible responses in a way that is compatible with observed evidence in behaving animals.

Results

Task 1: Delayed nonmatch-to-sample task

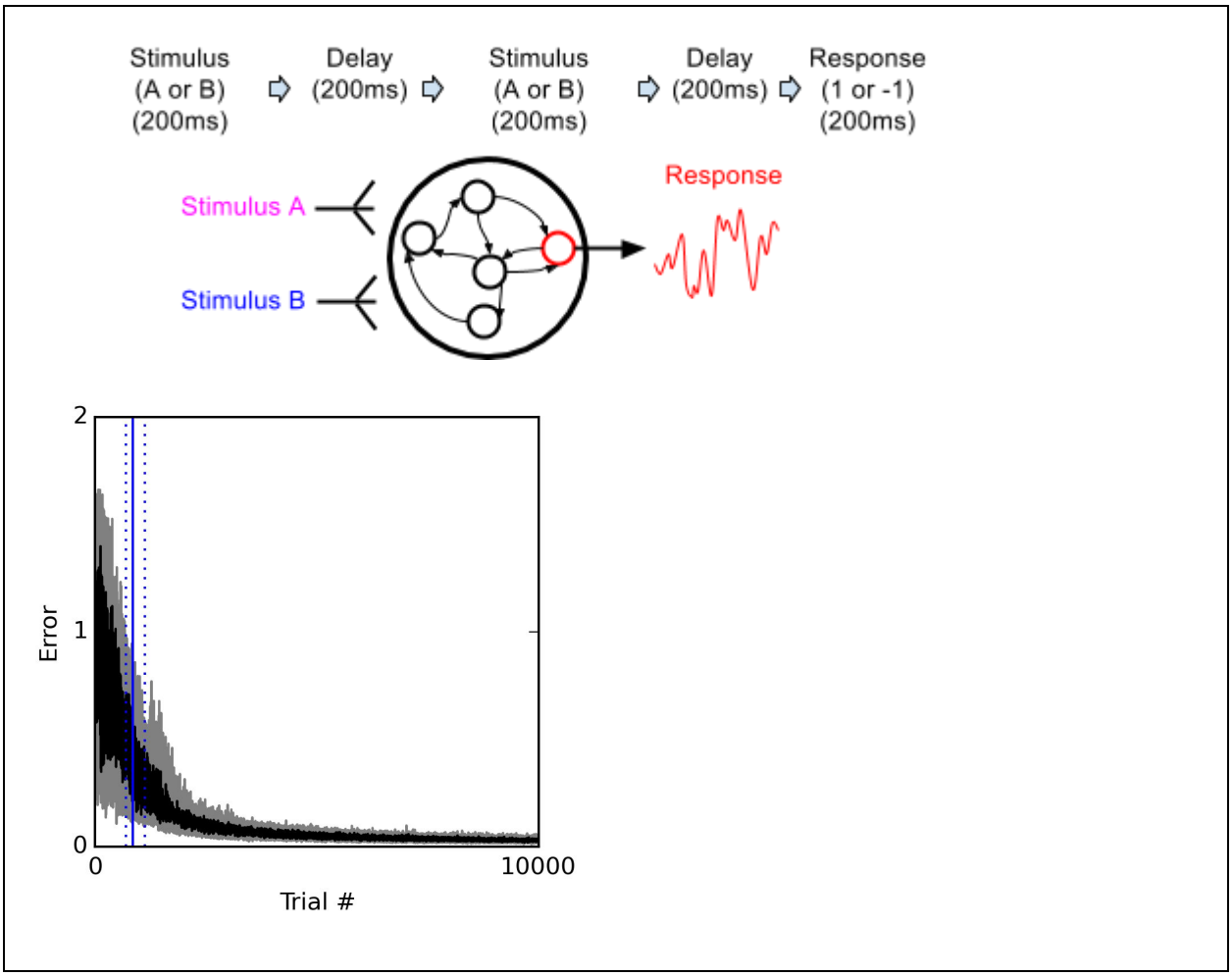

Figure 1. Delayed nonmatch-to-sample task. Top: task description. The network is exposed to two successive stimuli, with an intervening delay. The task is to produce output -1 if the two stimuli were identical (AA or BB), or 1 if they were different (AB or BA); the output of the network is simply the activity of one arbitrarily chosen "output" neuron, averaged over the last 200 ms of the trial. Bottom: time course of response error for each successive trial, as a function of the trial number (dark curve: median over 20 runs; gray area: inter-quartile range). The solid vertical line indicate the median number of trials needed to reach the criterion of 95% correct responses over 100 successive trials (843 trials); dotted vertical lines indicate the inter-quartile range (692-1125 trials). Performance (i.e., magnitude of the response error) continues to improve after reaching criterion and reaches a low, stable residual asymptote.

The first task considered here is a simple delayed nonmatch-to-sample problem (Figure 1). In every trial, we present two brief successive inputs to the network, with an intervening delay. Each input can take either of two values, labelled A and B respectively. The task is to determine whether the two successive inputs are identical (AA or BB), in which case the network should output -1; or different (AB or BA), in which case the network should output 1. We specify the input stimuli by using two different input channels u1 and u2; the identity of the input stimulus is determined by which channel is activated (i.e., for stimulus A, u1=1 and u2=0; for stimulus B, u1=0 and u2=1; remember that each input channel u_k is transmitted to the network by its own independent set of weights - see Methods). In every trial, the first stimulus is presented for 200 ms, then after a 200 ms delay the second stimulus is presented for 200 ms. Outside of input presentation periods, both input channels are set to 0. The trial goes on for an additional 400ms, thus each trial is 1000ms long. The network's overall response is determined by the activity of the arbitrarily chosen output neuron over the last 200 ms of the trial (the so-called "response" period). The overall error for this trial is the average *absolute* difference between the network's output (that is, the activity of the output neuron) and the target response (1 or -1 depending on presented stimuli), over these last 200ms.

This simple task exhibits several interesting features. First, it is arguably the simplest possible flexible decision task: on sensing the second stimulus, the network must flexibly produce a different response depending on the identity of the first stimulus. Second, because the intervening delay is much longer than the neural time constant, the network must maintain some memory of the first stimulus before the second stimulus arises. Third, to solve this task, some neurons in the network must necessarily possess some form of nonlinear mixed selectivity (note that the problem is in essence a delayed exclusive-or problem), a hallmark of neural activities in prefrontal cortices (Rigotti et al., 2013).

The networks consistently learn to perform the task with high accuracy. Figure 1 shows the time course of the median error over 20 training runs, each starting with a different randomly initialized network. Shaded area indicates 1st and 3rd quartile over the 20 runs. To define a measure of successful performance, we set a criterion of 95% "correct" responses (i.e., correct sign of the output cell activity) over 100 successive trials ($p<10^{-20}$ under random choice, binomial test). The median time to criterion across the 20 runs is 912 trials (inter-quartile range: 658-1393). Response error reliably converges towards a very low residual value.

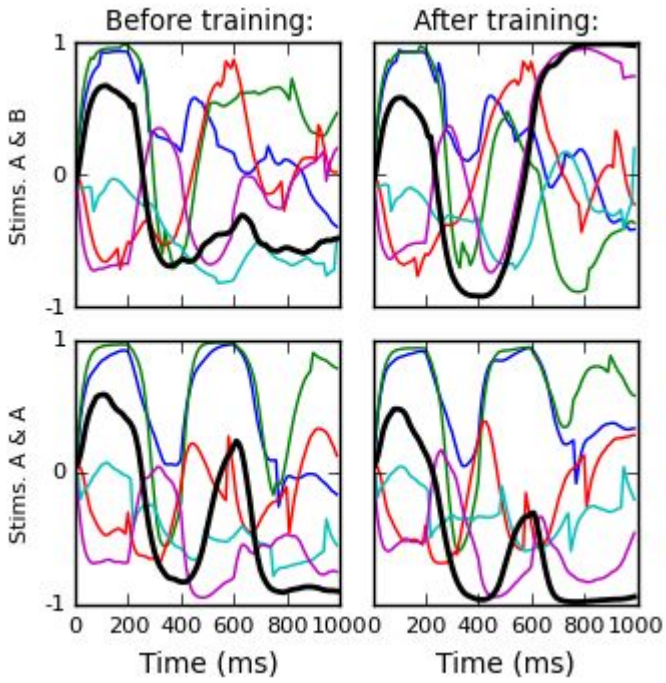

Figure 2. Activities of 6 different neurons, including the output neuron (thick black line), for two stimulus combinations, before training (left) and after training (right). Note that neural traces remain highly dynamical even after training.

How does the network represent and maintain traces of incoming stimuli? One possibility is that certain neurons encode stimulus identity by maintaining a stable "register" value over time, such that the firing rate of certain cells directly specify stimulus identity in a relatively time-independent manner. By contrast, physiological studies suggest that neural coding during working memory task is highly dynamic, with stimulus identity being represented by widely fluctuating cell activations, in such a way that the tuning of individual neurons significantly changes over the course of a trial (Meyers et al., 2008; Barak et al., 2010; Stokes et al., 2013).

To analyze the encoding and maintenance of stimulus identity over time in the network, we used a cross-temporal classification approach (Meyers et al., 2008; Stokes et al., 2013; Stanislas Dehaene, 2016). We trained a maximum-correlation classifier to decode various task-relevant

features (identity of 1st and 2nd stimulus, final response), based on whole-population activity at any given time. If we train one such classifier at time *t* in the trial, and then use it to decode population activity at the same time *t*, then decoding accuracy measures how strongly the network encodes the feature at that time point *t* (note that we always use separate subsets of the data for training and decoding). However, when the decoder is trained on data at time t_{learn} and then applied to population activity data at time t_{decode}, the resulting accuracy measures the stability in the network's "neural code" for this feature across both time points, i.e., how similarly the decoded feature is represented by the network across these time points. If representations are similar across both time points (that is, if the network use similar patterns to represent each possible value of the feature across both time points), then classifiers successfully trained with population activities at time t_{learn} should also produce accurate decoding of population activities at time t_{decode}. By contrast, if the network uses different representations/encoding of task features at these two time points, cross-temporal accuracy should be poor; this should be represented as "bottlenecks" of high accuracy on the cross-temporal decoding plots, whereby information is high along the diagonal (i.e. the feature is indeed encoded by the network at that given time), but away-from-diagonal (cross-temporal) decoding accuracy is low.

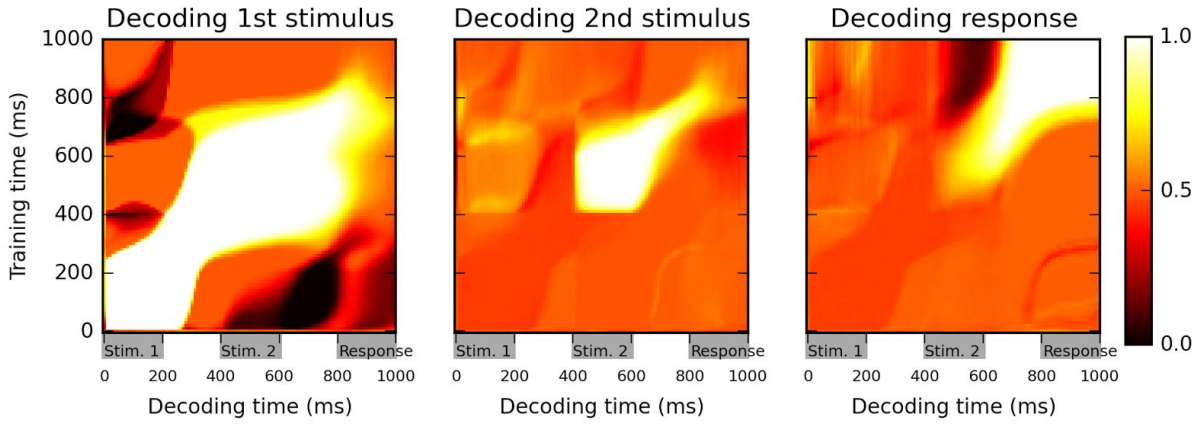

Figure 3. Cross-temporal classification performance reveals dynamic coding. Cross-temporal classification of 1st stimulus identity (left panel), 2nd stimulus identity (middle panel) and

network response (right panel). Row i and column j of each matrix indicates the accuracy of a classifier, trained on population activity data at time i, in guessing a specific task feature using population activity data at time j (training and decoding data are always separate). While the network reliably encodes information about stimulus identity right until the onset of the response period (as shown by high accuracy values along the diagonal in left and middle panel), this information is stored with a highly dynamic encoding (as shown by low cross-classification accuracy across successive periods, i.e., "bottlenecks" with high accuracy on the diagonal but low accuracy away from the diagonal). Note that in both left and middle panel, stimulus identity information decreases greatly at the onset of the response period, reflecting a shift to a from stimulus-specific to response-specific encoding (see also Figure 4).

The results in Figure 3 suggest highly dynamic representation of stimuli by the network. For example, the identity of the first stimulus can be successfully decoded during both first and second stimulus presentation, as well as during the intervening delay, as shown by high classification accuracy values on the diagonal during this entire period (Figure 3, left panel). However, the cross-temporal classification performance between these two periods (for example, in the areas at 0-200ms on one axis and 400-600 on the other, corresponding to training the classifier based on data from one stimulus presentation and testing it on data from the other stimulus presentation) is essentially at chance level (accuracy ~ .5), or even below chance (dark patches). This suggests that while the network reliably encodes information about 1st-stimulus identity across the first 800ms of the trial, the way in which this identity is represented changes widely between successive periods within the trial. Similarly, 2nd-stimulus identity is maintained from its onset until the beginning of the response period, but in a dynamical manner (lower cross-classification accuracy between the 400-600ms period and the 600-800ms period, in comparison to the high diagonal accuracy over the entire 400-800ms period).

Another feature of these plots is that the accuracy of stimulus identity decoding strongly decreases over the course of the "response" period (the 800-1000ms period, over which the

output cell is evaluated for the error signal). In this period, even along the diagonal, decoding accuracy for the identity of either first or second stimulus is initially high, but diminishes greatly by ~800ms. This suggests that the network largely stops maintaining information about the specific identity of previous stimuli, and instead encodes solely the actual response, as shown by the very strong classification accuracy in the third plot. While the network starts encoding its eventual response during the presentation of the 2nd stimulus, this representation is also dynamic initially (low cross-temporal classification performance between the 400-600 and the 800-1000 period on the right panel of Figure 3). However, at the onset of the response period, the networks implements a highly stable representation of the response, as seen from the square zone of high cross-classification accuracy in the top-right corner of the 3rd plot. Importantly, this does not imply that all or most neurons enter a stable, 'frozen' activity state: Figure 2 shows that individual neural activities remain dynamic during the response period. Rather, it means that enough neurons have sufficiently stable firing rates over that time that network response can be accurately decoded by using the same comparisons over the entire response period.

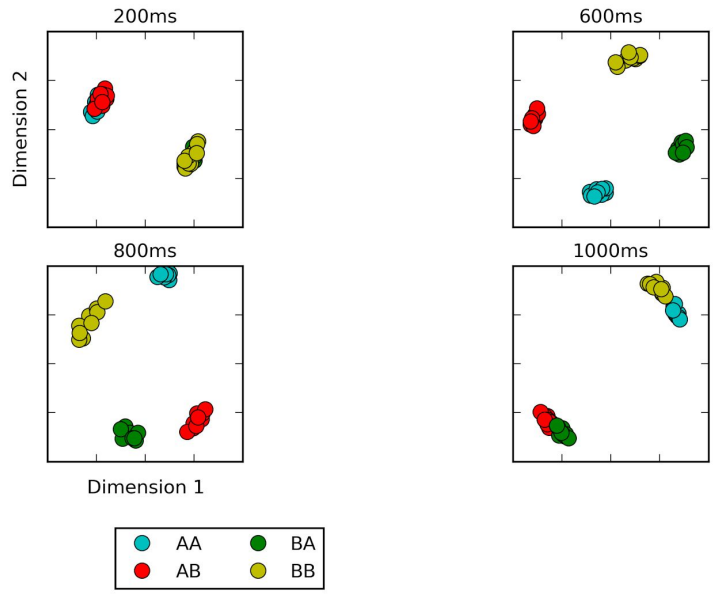

Figure 4. Multi-dimensional scaling plots of population activity reflect shifting encodings of task-relevant information. Population response vectors at various points in time (color-coded

by stimulus combination) are projected in two dimensions while preserving distances between data points as much as possible, using multi-dimensional scaling. At the end of the first stimulus presentation (200ms), population states are firmly separated by first stimulus identity, as expected. After second stimulus presentation (600ms), all four possible stimulus combinations lead to clearly separated population activity states. However, population states corresponding to different responses start to cluster together at the onset of the response period (800ms). Late in the response period (1000ms), population trajectories corresponding to the same response (AA and BB, or BA and AB) have largely merged together, reflecting a shift from stimulus-specific to response-specific representation and a successful "routing" of individual stimulus-specific states to the adequate response-specific state.

The fact that the network mostly "forgets" specific stimulus identity during the response period suggests that the population moves from a so-called "stimulus-specific" encoding to a "response-specific" encoding: the stimulus-specific response is flexibly routed to the appropriate, context-dependent response state, as previously observed in cortical activity during a flexible association task (Stokes et al., 2013). To test this interpretation, following (Stokes et al., 2013), we produce Multi-dimensional scaling (MDS) plots of population activity at different time points and for different stimulus conditions (Figure 4). MDS attempts to find a 2D projection such that the distance between any two data points is as similar as possible to their actual distance in the full-dimensional space: nearby (distant) population states should thus produce nearby (distant) points on the MDS plot.

During the first stimulus presentation (200ms), population trajectories are grouped according to 1st stimulus identity, as expected, since this is the only information available to the network at that time. After the second stimulus presentation (600ms), population trajectories have split again, in such a way that all possible stimulus identity combinations generate different, consistent trajectories. This four-way distinction begins to erode at the onset of the response period (800ms), in which the population states for "different-identity" stimulus combinations

(AB and BA) begin to come closer to each other. By the late response period (1000ms), the trajectories have essentially merged into two clusters, corresponding to the network response ("same" or "different") and largely erasing any distinction based on specific identity of either first or second stimulus. Thus, during the response period, the network flexibly moves from a stimulus-specific representation to a response-specific representation, consistent with physiological observations.

Task 2: Flexible selective integration of sensory inputs

An important aspect of cognitive control is the ability to attend selectively to specific portions of the sensory input, while ignoring the rest, in a flexible manner. Recently Mante, Sussillo and colleagues have studied the neural basis of this ability in macaque monkeys prefrontal cortex (Mante et al., 2013). In this study, monkeys looked at randomly-moving colored dots, in which both the value and coherence of motion direction and dot color varied from trial to trial. Monkeys had to report the dominant motion direction, or the dominant color, according to current task conditions; thus, the same stimulus could entail different appropriate responses depending on current context. Furthermore, due to the noisy stimulus, the task required temporal integration of sensory input. Importantly, the authors showed that prefrontal neurons registered inputs from both the relevant and the irrelevant modality; however, inputs from the irrelevant modality had no long-term impact on neural activities, while inputs from the relevant modality were selectively integrated over time. Thus, only information from the relevant modality contributed to the final decision.

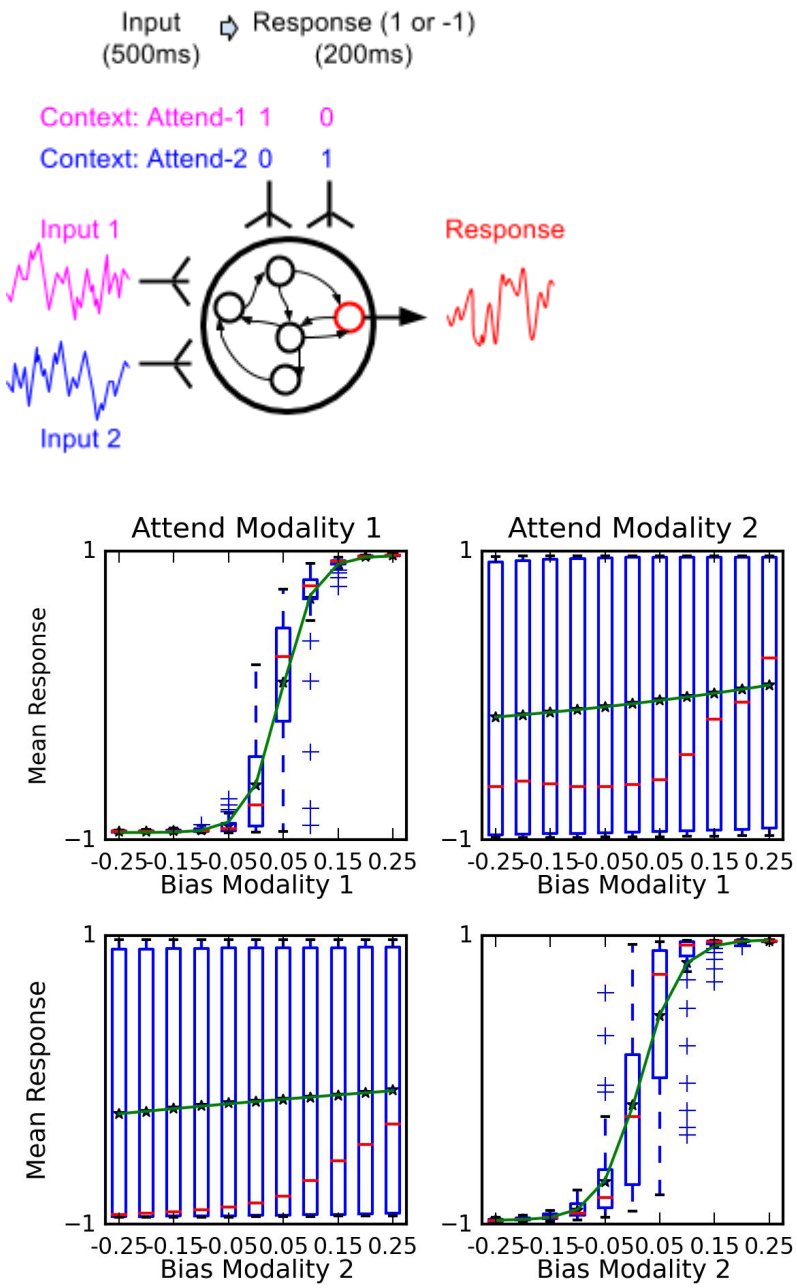

Figure 5. Selective integration task. Top: task description. The network receives two noisy inputs, simulating sensory information coming from two different modalities, as well as two "context" inputs to indicate which of the two sensory inputs must be attended. The task is to produce output 1 if the cued input has a positive bias, and -1 if the cued input has negative bias; this task requires both selective attention and temporal integration of the attended input. Bottom: Psychometric curves of network responses. Top-left panel: average response when

context requires attending to modality 1, sorted by the bias of modality 1 inputs. The network response correctly tracks the overall bias of modality 1 inputs. Bottom-left panel: same data, but sorted by modality 2 bias. Network response is mostly unaffected by modality 2 bias, as expected since the network is required to attend to modality 1 only. Right panels: network responses when context requires attending to modality 2. Again, the network correctly identifies the direction of the relevant modality while mostly ignoring the irrelevant modality.

In addition to neural recordings, Mante and Sussillo also trained a recurrent neural network to perform the same task, using supervised learning based on Hessian-free optimization. By analyzing the trained network, they identified mechanisms for selective integration of flexibly-specified inputs in a single network (Mante et al., 2013). This task was also used as an example application by Song and colleagues for their recurrent network training framework (Song et al., 2016).

We trained a network to perform the same task, using our proposed plasticity rule (see Figure 5). Our settings are deliberately similar to those described by Mante, Sussillo and colleagues. The network has two "sensory" inputs (representing the two stimulus modalities of motion and color) and two "context" inputs (which specify which of the two modalities must be attended to). The sensory inputs are noisy time-series, centered on a specific mean which indicates the "value" of this input for this trial. More precisely, each of the two sensory inputs is sampled at each time step from a Gaussian variable with variance 1 and a mean, or bias, randomly chosen between -0.25 and 0.25 for each trial. The mean of the Gaussian represents the "direction" or "value" of the corresponding sensory input. The context inputs are set to 1 and 0, or 0 and 1, to indicate the relevant modality for this trial; the goal of the network is to determine whether the sensory input in the relevant modality has positive or negative mean. Sensory inputs are presented for the first 500ms of the trial, followed by a 200ms response period during which all sensory inputs are set to 0. The expected response of the network is 1 if the relevant sensory input has positive mean, and -1 otherwise; thus the same sensory input can entail different appropriate responses depending on the context. As for the previous task, the network's response for a trial is the firing

rate of the arbitrarily chosen output cell, and the error for a trial is the average absolute difference between the firing rate of this output cell and the appropriate response for this trial (either -1 or 1) over the 200ms response period.

Figure 5 shows the psychometric curves of a fully-trained network, that is, the mean response as a function of stimulus value. For either modality, we show separate psychometric curves for when this modality was the relevant one and when it was irrelevant. When trials are sorted according to the value of the relevant modality, responses form a steep sigmoid curve with a relatively sharp transition between -1 and +1 centered roughly at 0. By contrast, when trials are sorted according to the value of the irrelevant modality, responses are evenly distributed across the entire range. Thus, the network accurately responds to the relevant signal, while largely ignoring the irrelevant one in each context (Compare to Figure Extended Data 2 in (Mante et al., 2013)). This indicates that the network has learned not only to perform temporal integration of an ambiguous, stochastic input, but also to flexibly "attend" to different input streams depending on context.

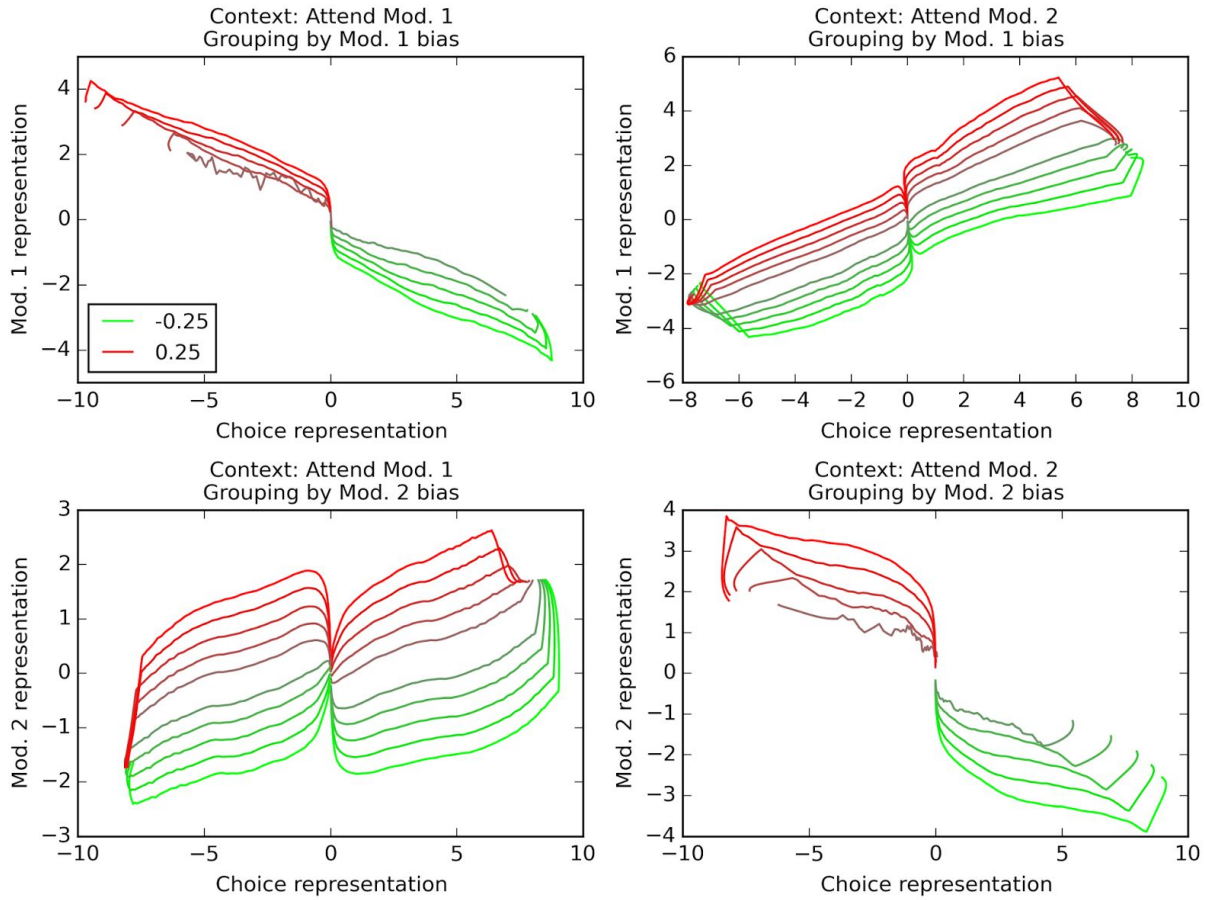

Figure 6. Orthogonal decoding of population activities. Population responses of trials having the same context, the same final choice and the same modality 1 bias (from -0.25 to 0.25) are averaged at each point in time, resulting in different trajectories (one per modality bias value), and projected over dimensions indicating how much the network encodes modality 1 value, modality 2 value, and final choice (the latter always on the x axis). Only correct trials are used (thus top-left and bottom-right panels only have 10 trajectories). The trajectories reveal that the network encodes both the relevant and the irrelevant modality, though only the relevant one is linked to final choice. See text for details.

How is information represented in the network over time? We use Mante & Sussillo's orthogonal decoding procedure, which seeks to extract independent measures of how various task features (stimulus values, context, decision) are encoded in the network (see Methods). Briefly, this

method consists in using multiple linear regression to associate each neuron with a "weight" describing the correlation of its activity with each task feature at successive points in time, then using these set of weights as vectors in population activity space (after orthogonalization), and projecting population activity on these vectors to measure how much each feature is being *independently* represented by the network at any time. This method was also employed by Song and colleagues to analyze their network's behavior (Song et al., 2016). The results are shown in Figure 6 (compare to Figures 2, 5 in (Mante et al., 2013)). These trajectories plot the evolution of network information over time, for various combinations of context and task feature. Each trajectory represents the average population activity of (correct) trials grouped according to the value of a given modality, as well as by the final choice for this trial; that is, for each trajectory, we group together all trials that share the same value of the specified modality, the same final choice, and the same context (relevant modality); then we average the responses of each neuron over all trials in each group at each point in time, giving rise to one population trajectory per group. We then project these averaged population trajectories along the orthogonal feature dimensions extracted by orthogonal decoding, and plot the resulting trajectories in feature dimension subspaces ('final choice' dimension is always used as the horizontal axis, while the vertical axis may be either of the two sensory modality dimensions).

While the trajectories are more complex than those produced by existing models (Mante et al., 2013; Song et al., 2016) (unsurprisingly, since we did not seek to faithfully reproduce cortical data and thus did not use the various regularizations and constraints included in these works), they share the same essential features. In particular, we see that *both* the relevant and the irrelevant modality are actually represented in the network: the trajectories for varying value of either modality form an ordered progression in the corresponding "modality" dimension (y-axis), even when that modality is irrelevant (bottom-left and top-right panels in Figure 6); however, only the relevant modality correlates with representation of final choice (compare panels with congruent vs. incongruent context and grouping modality), in accordance with physiological observations (Mante et al., 2013).

Task 3: Controlling a musculoskeletal model of the human arm

In both of the previous tasks, the network output was a single response channel. However, flexible behavior often requires coordinating multiple outputs, especially during movement. To test whether our plasticity rule can produce coordinated multiplexed responses, we trained a network to control a biomechanical model of the human arm. The model is a custom modification of the one described in (Saul et al., 2015) (itself an extension of (Holzbaur et al., 2005)) and uses the Thelen muscle model (Thelen, 2003). The model implements the human upper skeleton, with 4 degrees of freedom (3 at the shoulder, 1 at the elbow), actuated by 16 muscles attached to the shoulder, chest, and upper and lower arm bones. Each of the 16 muscles is controlled by a specific network output cell. The task is to reach towards one of two spherical targets, located in front of the body on either side of the sagittal plane. The appropriate target ball for each trial is indicated by two input channels, set either to 1 and 0 (left-side target) or to 0 and 1 (right-side target) respectively for the entire duration of the trial (700 ms). No other inputs are provided to the system. The error at the end of each trial is measured by the absolute distance between the tip of the hand and the center of the target ball, plus a small penalty for total muscle activation over the entire trial. Note that while the target balls are symmetrically arranged with regard to the body, they are not symmetrical with regard to the right arm (which is the one we model): the right-side ball is closer than the left-side one, and thus reaching either target requires qualitatively different movements.

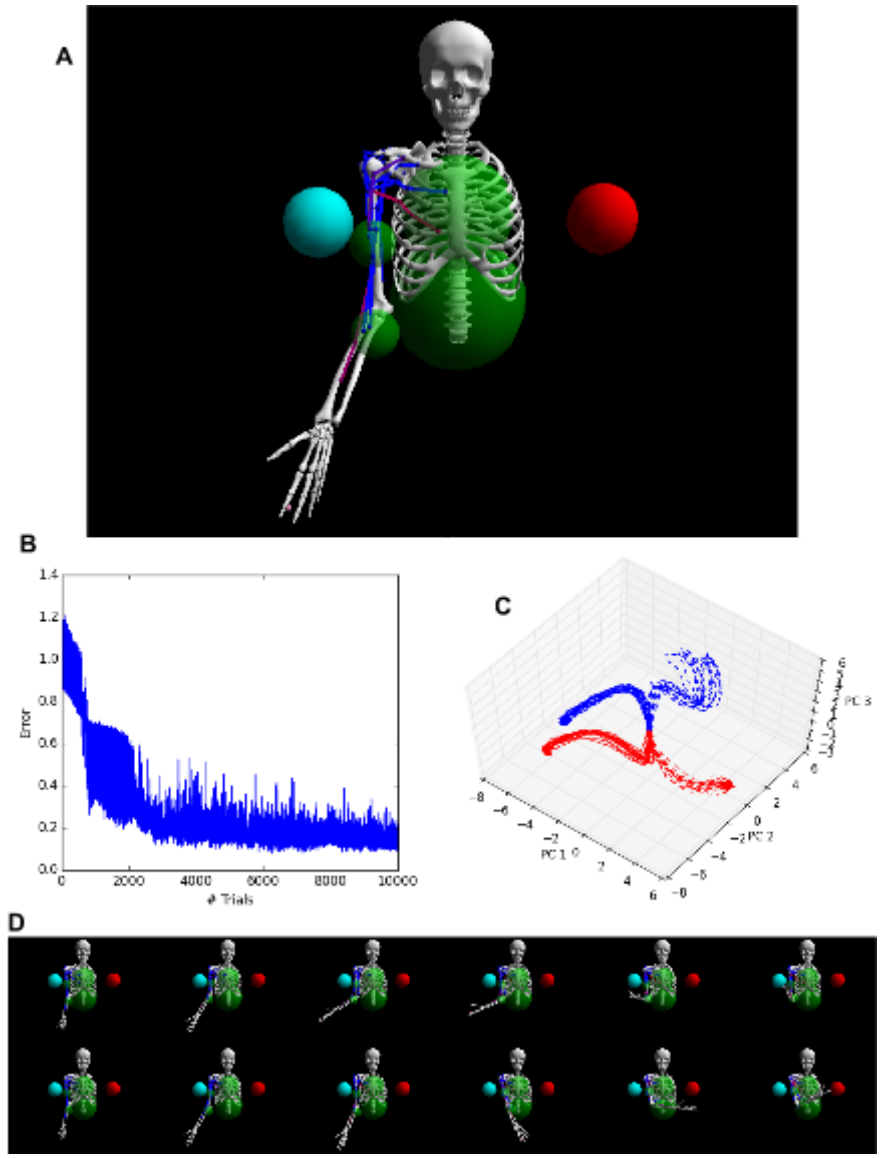

Figure 7. Controlling a biophysical model of the human arm. A: a model of the human upper skeleton with 4 degrees of freedom (shoulder and elbow joints), actuated by 16 muscles at the shoulder, chest and arm (colored strings: blue indicates low activation, red indicates high activation). The task is to reach either of two target balls (blue or red), depending on a context input. B: During training, the error (measured by the distance between tip of hand and target ball at the end of each trial) improves immediately and reaches a low residual plateau after about 3000 trials. C: Projection of network responses along the first three principal

components of network activity, both before and after training (blue: left-target, red: right-target, dotted lines: untrained network, solid lines: trained network). D: frame-by-frame illustrations of a right-target trial (top row) and a left-target trial (bottom row), after training.

Results are shown in Figure 7. Initially, as expected, the untrained network performs random, aimless movements, resulting in high initial error (Figure 7B). Performance improves almost from the start of the training process, reaching a low residual error after about 3000 trials. To visualize the impact of training on the dynamics of population activity, we project the population activity over its first three principal components at successive points in time over the course of each trial, using 16 trials for either target context, both before and after training (i.e., 64 trials in total). The projection shows that training considerably alters network trajectories (Figure 7C). The fully trained network correctly reaches the adequate target according to context (Figure 7D).

Discussion

This paper makes three contributions:

1- We introduce a biologically plausible learning algorithm that can train a recurrent neural network to learn flexible (input-specified) tasks, using only time-sparse, delayed rewards to guide learning, with information local to the synapse.

2- We show that this rule can train network for relatively complex tasks, requiring memory maintenance, selective attention, and coordination of multiple outputs.

3- We show that the trained networks exhibit features of neural activity observed in primate higher cortex during similar tasks. In particular, we demonstrate highly dynamic population-wide encoding of task-relevant information, as observed in neural recordings (Meyers et al., 2008; Barak et al., 2010; Stokes et al., 2013); and we show that selective integration of sensory inputs occurs as described in both observational and modelling studies of primate prefrontal cortex during a similar selective attention task (Mante et al. 2013). In our view, the fact that these features of cortical activity arise spontaneously in networks trained with a biologically plausible

rule (as opposed to training the network to directly reproduce observed neural activity traces) increases the plausibility of recurrent neural networks as a model of cortical computation, during both performance and learning of cognitive tasks.

Our proposed plasticity rule implements reward-modulated Hebbian learning between inputs, outputs, and rewards, with the crucial introduction of a supralinear function S applied to the Hebbian plasticity increments (see Methods). In other words, we suggest that plasticity is dominated by large correlations (or anti-correlations) of inputs and outputs, while smaller ones are relatively ignored. While the particular choice of supralinear function is not critical, simply setting S to identity fails to produce adequate learning. This hypothesis of non-linear effects in Hebbian plasticity allows our rule to support robust learning in highly dynamic networks, without requiring non-Hebbian plasticity between segregated driving and perturbatory inputs (Fiete et al., 2007), or a continuous, real-time reward signal (Legenstein et al., 2010; Hoerzer et al., 2014) (see Appendix). We note that this suggestion is similar to the independent proposal of so-called thresholded Hebbian rules (Soltoggio and Steil, 2013), in which plasticity is only triggered if the Hebbian product reaches a certain threshold.

The flexible, dynamic coding observed in prefrontal activity has led to suggestions that cortex implements "silent" memory traces by using short-term synaptic plasticity (Barak et al., 2010; Stokes, 2015). Short-term synaptic plasticity clearly plays an important role in neural responses, and may well play an important role in maintaining a "hidden internal state" of the network (Buonomano and Maass, 2009). However, our network does not implement short-term synaptic plasticity; no weight modification occurs during the course of a trial (all learning occurs between trials), and all the decoding results reported above were obtained with frozen synaptic weights. Our results suggest that the highly dynamic activities spontaneously produced by near-chaotic recurrent networks can be harnessed to produce the dynamic encodings observed in experiments, using only sparse, delayed rewards and biologically plausible plasticity rules. Thus, while short-term synaptic plasticity clearly affects neural responses, it may not be required to explain the highly dynamic nature of working-memory encodings.

It is unlikely that cortical connectivity should be drastically and finely remodeled through a long training process for any new task. For example, while monkeys require extensive training to perform decision tasks, human subjects can quickly perform new tasks simply by verbal instruction. Rather, it is more likely that the long process of slow, reward-modulated synaptic modification in cortical circuitry depicted here reflects the learning of functional networks capable of implementing a certain *type* of task (or cognitive ability), which must then be activated and adapted to the particular parameters required by current needs. Then, to perform a specific task, it would be sufficient to learn to provide adequate inputs to these areas in order to activate the proper generic networks, using much faster, ephemeral plasticity in feedforward connections between areas. The latter process of flexible task specification through fast input plasticity is likely to involve not just other cortical areas, but also the basal ganglia, under the guidance of moment-to-moment dopamine delivery. In the present paper, we simplified the complex architecture of animal decision-making by using only one single network to represent what is likely to involve many different loci in the brain, in order to better illustrate the proposed learning rule. While these multiple components were unnecessary for the simple, single-task settings described in this paper, elucidating the interactions between cortical, limbic, and dopaminergic structures is an important future task for the study of flexible behavior and its neural implementation.

Methods

Model description

Our model is a fully-connected continuous-time recurrent neural network of N neurons, governed by the following classical RNN equations (Sompolinsky et al., 1988; Jaeger, 2001; Maass et al., 2002; Sussillo and Abbott, 2009):

$$\tau \frac{dx_i}{dt} = -x_i(t) + \sum_{j=1}^{N} J_{j,i} r_j(t) + \sum_{k=1}^{M} B_{k,i} u_k(t) \qquad \text{[Equation 1]}$$

$$r_i(t) = tanh(x_i(t)) \qquad \text{[Equation 2]}$$

where x_i is the excitation (or "potential") of neuron i, r_i is its response (or "firing rate" / activity), $J_{j,i}$ is the connection weight from neuron j to neuron i, $u_k(t)$ is the current value of each of the M external inputs to the network, and $B_{k,i}$ is the connection weight from external input k to neuron i (τ is the relaxation time constant of neuron activation). In addition, four arbitrarily chosen neurons have a constant activation x=1 and thus provide a bias input to other neurons. There is no separate feedback or output network. Instead, one or more neurons in the network are arbitrarily designated as the "output" neurons, and their responses at any given time are used as the network's response (these neurons are otherwise identical to all others). J is initialized with weights taken from a normal distribution with mean 0 and variance g^2/N, while the input weights $B_{k,i}$ are fixed and taken from a uniform distribution over the [-1,1] interval. Activations x_i are initialized at the start of every trial with uniform noise in the [-0.1, 0.1] range. For the simulations reported here, N=200 (400 for the motor control task), τ=30ms, and g=1.5. Note that the latter value places the networks in the early chaotic regime, where the long-term behavior generally remains non-periodic (Sompolinsky et al., 1988).

Synapses between neurons are modified according to a novel form of reward-modulated Hebbian learning, as described below. To produce exploratory variation in network responses across trials, each neuron in the network occasionally receives a random perturbation $\Delta_i(t)$ to its activation; these perturbations are part of the inputs, and are not segregated from "normal" inputs (in contrast to (Fiete and Seung, 2006; Fiete et al., 2007)). Note that $\Delta_i(t)$ might also represent random noise, or a "teaching" signal from a different area. In this paper, $\Delta_i(t)$ is taken from a uniform distribution within the [-.5, .5] range, occurring randomly and independently for each neuron with a mean rate of 3Hz.

During a trial, at every time step, every synapse from neuron *i* to neuron *j* accumulates a *potential* Hebbian weight change (also called eligibility trace (Izhikevich, 2007)) according to the following equation:

$$e_{i,j}(t) = e_{i,j}(t-1) + S(r_i(t-1) * (x_j(t) - \bar{x}_j)) \qquad \text{[Equation 3]}$$

Remember that r_i represents the output of neuron *i*, and thus the current input at this synapse. x_j represents the activation of neuron *j* and $\underline{x}_j$ represents a short-term running average of x_j, and thus $x(t)$ - $\underline{x}$ tracks the fast fluctuations of neuron output. Thus this rule is essentially Hebbian, based on the product of inputs and output (fluctuations). Crucially, *S* is a monotonic, supralinear function of its inputs. The particular choice of *S* is not critical, as long as it is supralinear. In this paper we simply used the cubic function $S(x) = x^3$. Sign-preserving squaring $S(x) = x|x|$ also gives satisfactory results; however, simply using the identity function fails to produce learning. The supralinear function allows our algorithm to successfully learn from instantaneous deviations of activity, using only sparse, delayed rewards, and without requiring a continuous, real-time reward signal (Legenstein et al., 2010; Hoerzer et al., 2014); see Discussion and Supplementary Material.

Note that the eligibility trace for any synapse is accumulated over the course of a trial, with each new timestep adding a small component to the synapse's eligibility trace / potential weight change.

At the end of each trial, a certain reward R is issued to the network, based on the network's performance for this trial as determined by the specific task. From this reward, the system computes a *reward prediction error* signal, as observed in physiological experiments, by subtracting the expected reward for this trial $\underline{R}$ (see below for computation of $\underline{R}$) from the actually received reward R. This reward-prediction signal is used to modulate the eligibility trace into an actual weight change:

$$\Delta J_{i,j} = \eta\, e_{i,j}(R - \overline{R}) \quad \text{[Equation 4]}$$

where η is a learning rate constant, set to 0.5 for all simulations described here. Intuitively, reward-modulated Hebbian learning as implemented here will change synaptic weights in such a way that the network's future responses will be more (respectively less) similar to the perturbed responses over this trial, if the reward was higher (respectively lower) than expected. In other words, reward-modulated learning 'incorporates' random perturbations into the network, if the perturbation leads to a positive outcome (see Discussion and Appendix).

To compute the reward prediction error signal (R-$\underline{R}$), we need to estimate the expected reward $\underline{R}$. Following (Frémaux et al., 2010), we simply maintain a running average of recent rewards for trials of the same type (where trial type is determined by the combination of inputs). As (Frémaux et al., 2010) pointed out, it is important that separate traces should be maintained for each trial type, so as to provide an accurate estimation of the expected reward $\underline{R}$ for each trial. Thus, after the *n*-th trial of a given type, $\underline{R}$ is updated as follows:

$$\overline{R}(n) = \alpha_{trace}\, \overline{R}(n-1) + (1 - \alpha_{trace})\, R(n) \quad \text{[Equation 5]}$$

Where R(n) is the reward for this trial, and $\underline{R}$(n-1) was the expected reward after the previous trial of the same type. In all simulations, $\alpha_{trace} = 0.33$.

To stabilize learning, we clip the weight modifications for each trial to have a maximum absolute value of 10^{-4} (across experiments, roughly 10% of all potential weight modifications exceed this value and are clipped).

Decoding of network information in a delayed nonmatch-to-sample task

In the delayed nonmatch-to-sample task (Figure 1), we used a cross-temporal classification analysis (Meyers et al., 2008; Stokes et al., 2013) to investigate how fully trained networks encode information over time (see Figure 3). We follow the maximal-correlation classifier

approach described in (Meyers et al., 2008) as closely as possible. Briefly, we want to measure how well a certain task-relevant feature (identity of 1st presented stimulus, or identity of 2nd presented stimulus, or final response) can be predicted by observing network activity at time t_1, using a classifier trained on network activity at time t_2. First, we sample the activation of each neuron, every 10 ms, for each trial. This data is stored in a matrix of 100 rows and 200 columns, indicating the activities (firing rates) of all 200 neurons at each of the 100 sampling times. We first generate 80 trials (20 per possible condition, where "condition" is defined as one of the 4 possible stimulus combination: AA, AB, BA or BB) with a trained network. The time course of neural activity will differ somewhat between successive trials, even for identical conditions, due to noise. Then we iterate the following procedure. For each of all 4 possible conditions, we randomly choose half the trials as "training" trials, and the other half as "testing" or "decoding" trials. The training trials corresponding to the same category that we are trying to decode (for example, all stimuli having the same 1st presented stimulus) are averaged together, pointwise, for each neuron and each time point, giving a "prototype" matrix of activation for each neuron at each timepoint under this category. This training data allows us to decode the category of each testing trial, at each point in time, using maximum-correlation classification, in the following way. We compute the Pearson correlation between each row of each "testing" trial and each row of each "prototype" trial. Each such correlation between row *i* of a testing trial and row *j* of a training category-average tells us how much the population activity at time *i* in the testing trial resembles the average population activity at time *j* for this particular category. We can then select the category for which this correlation is maximal, at each training/testing timepoint pair, as the "decoded" category for each testing trial. For each testing trial, this provides a 100x100 matrix of decoded categories (one for each pair of training and testing timepoints). Of course, each testing trial belongs to only one category, so only one possible answer is correct, and thus we can compute another 100x100 matrix of binary values, indicating whether the decoded category at a given point in the decoding matrix (i.e., for any given pair of training and testing timepoints) is correct. The average of these "correctness matrices", over all testing trials, provides the accuracy in cross-temporal decoding of this category for every training/testing pair of timepoints. We iterate this whole procedure 100 times and average together the resulting

"correctness" matrices. The resulting 100x100 matrix indicates at each row *i* and column *j* the proportion of times that the decoded category for population activity at timepoint *j* was correct, using training data from timepoint *i*. This is the matrix shown in each of the panels in Figure 3 (one for each of the three categories to be decoded).

Orthogonal decoding of network information during a selective integration task

For the selective integration task, we used the analysis method introduced by Mante, Sussillo and colleagues (Mante et al., 2013), and also used by Song and colleagues (Song et al., 2016) (see Figure 6). Intuitively, the purpose of this method is to estimate how much information the network encodes about different task feature (input value, context, final choice, etc.) *independently* from each other.

After generating multiple trials under various conditions (context -- that is, relevant modality -- and bias for each modality) with a fully trained network, we regress the activity of each neuron over the values of features of interest (context, value of each modality, and final choice) for each trial. This gives us a set of weights for each neuron, one for each feature, representing how much each feature influences the neuron's firing rate. We then "switch views" by grouping together all such weights for any given feature (200 weights - one per neuron). This in turn produces vectors in neuron population space, along which the feature is in a sense maximally represented (notice that this is quite different from, and not equivalent to, the simpler idea of simply regressing each feature over the firing rates of the neurons across trials). We then orthogonalize these vectors using QR decomposition, to ensure that these representations are as independent from each other as possible. Projecting population activity at a given time over the resulting vectors approximates the network's current estimate of the corresponding feature at that time. For successive time slices, we average network activity vectors corresponding to the same value of bias in a certain modality, a certain attended modality, and a certain final choice. We refer the reader to (Mante et al., 2013) for a complete description of the method.

We project population activity, averaged within various groups of trials, at each point in time, over these decoding axes. The trials are grouped according to final choice, value of one modality (either modality 1 or modality 2), and current context (i.e., relevant modality), and the population activity at each point in time is averaged across all trials within each group. When the resulting averages are projected over the orthogonal feature vectors, they produce trajectories, indicating the network's encoded value for each feature, at each point in time, for trials of this group. Only correct trials are used, and thus certain combinations are impossible (for example, positive value of modality 1 bias, while attending modality 1, with a final choice of -1); this is reflected in the top-left and bottom-right panels of Figure 7, which contain half as many trajectories as the top-right and bottom-left panels.

Appendix

Here we attempt to provide a more intuitive explanation of various learning algorithms for strongly connected RNN, including the node-perturbation method (Fiete and Seung, 2006), the Hoerzer-Legenstein-Maas method (Hoerzer et al., 2014), and the new algorithm introduced in the present paper.

Why does node-perturbation work?

Node-perturbation consists in applying perturbations to the outputs of neurons, then computing an eligibility trace equal to the product of the (non-perturbative) inputs by the perturbations, accumulating this product over a period of time, and finally multiplying this eligibility trace by a baseline-subtracted reward signal to obtain the actual synaptic modifications. Fiete and Seung (Fiete and Seung, 2006; Fiete et al., 2007) showed formally that this method descends the gradient of error over the weights. Interestingly, this method is highly similar to the classical

REINFORCE algorithm (see Equation 11 in (Williams, 1992), and discussion in (Fiete and Seung, 2006)), which can also be shown to descent the gradient of error over the weights with a very different approach (Peters and Schaal, 2008; Kober et al., 2013). As such, it is essentially a form of policy gradient search, to use the terminology of reinforcement learning (Peters and Schaal, 2008; Kober et al., 2013).

Our previous experiments showed that the node-perturbation method can successfully learn complex tasks in strongly-connected, chaotic RNN (see [Reference removed for double-blinding]). The plasticity rule described in this paper, much like the method proposed by Legenstein, Hoerzer and Maass (Legenstein et al., 2010; Hoerzer et al., 2014), is largely an implementation of this method with more biologically plausible mechanisms.

Why is node-perturbation learning so efficient in learning correct trajectories? While the mathematical derivation of node-perturbation and REINFORCE are well established (Fiete and Seung, 2006; Peters and Schaal, 2008), here we suggest a more intuitive explanation of how this method works.

Briefly, node-perturbation learning imposes a phasic perturbation on the network, then "incorporates" the effect of this perturbation into the network weights if the resulting trajectory turns out to produce a higher reward than expected - or conversely, incorporates the "opposite" of this perturbation if the resulting trajectory turned out to produce lower reward than expected. The effect of the weight modification is to ensure that, next time the same input is presented, the neurons will respond in a way that will be a bit more similar to what their rewarded perturbed response was. Conversely, if the trajectory turned out to lead to a lower reward, the weight modification will be "anti-incorporated" into the weights, making sure that the next presentation of the same inputs will be nudged away from the direction of the perturbation.

Consider what happens when a perturbation z(t) is applied to neuron j at time t. Under node-perturbation, the weight modification dW at a given incoming synapse of neuron j is equal

to x(t) * z(t) * DR, where x(t) is the current pre-synaptic input at the synapse, z(t) the perturbation received by the neuron, and DR the net reward received at the end of the trial (centered to mean zero over many trials). Suppose that DR is positive (the effect of the perturbation was "good") and z was positive (the neuron received a positive perturbation). The effect of this dW will then be to add a positive multiple of x(t) to the weight Wj,i. As a result, the input weight vector of neuron j will become more correlated with the current population response vector X(t); this will result in a higher response of j next time the input X(t) is presented. Since the perturbation z(t) was positive, this is exactly what is needed to replicate the effect of the perturbation, that is, increase the firing of j. Conversely, if z(t) was negative, the effect would be to subtract a multiple of X(t) from the input weight vector of neuron j, which would reduce the correlation between X(t) and its weight vector (possibly making it more negative if it already was), and therefore reduce the response of j next time input X(t) is presented - again, mimicking the effect of the (negative) perturbation.

The converse applies when DR is negative (that is, when the resulting trajectory was "bad"): the resulting modification will lead to incorporating the opposite of the received perturbation into the incoming weight vector.

Why does simple reward-modulated Hebbian learning not work (for strongly-connected recurrent networks)?

Node perturbation is not strictly Hebbian. In reward-modulated Hebbian learning (RMHL), weight modifications are proportional to the product of pre- and post-synaptic activities (later multiplied by a reward signal), whereas node-perturbation computes the product of pre-synaptic activity by *perturbations*, rather than the full post-synaptic activity. As explained above, this allows the synaptic change to "incorporate" the effect of the perturbation if this perturbation led to a higher reward, or incorporate its opposite if the perturbation led to a lower reward.

At first sight it might seem that the two are easily reconciled. Let us write the total reward-modulated Hebbian synaptic weight change after a trial as x * (y + z) * DR, where x is pre-synaptic activity, y is post-synaptic activity, z is the perturbation and DR is the zero-mean reward signal. In the limit of slow weight modifications, for successive presentations of a same input, the total weight change will be $\sum$ x * (y + z) * DR = $\sum$ (xyDR + xzDR) (where summation is taken across successive trials). Because xy is roughly constant across successive presentations of the same input (for slow weight modification) and DR is centered to have zero mean, the first term vanishes, leaving only the second term x*z*DR. But this remaining term is exactly equal to the equation for node-perturbation learning, as described above. Thus, intuitively, the bulk of the post-synaptic response should "cancel out" due to the zero-mean centering of the reward signal, leaving only the perturbation-proportional term: the only change between RMHL and node-perturbation should thus be a change in variance, rather than in the asymptotic weight value.

However, while the foregoing is valid when inputs and outputs are constant for a given trial, it does not hold when the input (and thus the outputs) are highly dynamic - especially when the outputs at time t influence the inputs at times t'>t, as is the case in a strongly recurrent network. In this case, xy is not constant over successive presentations of the same trial; indeed, a small perturbation may lead to a drastic, unpredictable change in future activity y within the trial. Therefore, the x*y*DR term does not vanish from the summation.

Under node-perturbation learning, these potentially large trial-to-trial fluctuations in intrinsic activity are ignored, because only the perturbation itself is used in the plasticity rule, which allows its incorporation in the weights. But in reward-modulated Hebbian learning, these fluctuations between successive presentations of the same input will typically dwarf the perturbations and thus dominate the synaptic changes, creating unpredictable weight changes that are unrelated to the direction of the perturbation (a related point is made by Legenstein and colleagues (Legenstein et al., 2010)).

Why does supralinear amplification of the weight increments restore the viability of reward-modulated Hebbian learning?

In the present article, we first isolate the perturbations from overall post-synaptic activity by subtracting a very short-term running average (trace) of ongoing activity. This idea was introduced by Legenstein and colleagues (Legenstein et al., 2010) and applied to recurrent neural networks by Hoerzer and colleagues (Hoerzer et al., 2014). However, their method required a continuous, real-time reward signal. This eliminates a major advantage of reinforcement learning, namely, the ability to learn from sparse, delayed rewards. By comparison, our method also requires an additional ingredient: we apply a supralinear function to the weight increments, amplifying the larger ones and suppressing small ones. This allows our rule to learn from delayed, time-sparse rewards.

The reason why such requirements (real-time reward signal or supralinear amplification) are needed is that simply subtracting a running average from ongoing activity does not perfectly isolate external perturbations, due to spurious relaxation effects. For example, after a positive perturbation is applied, the running average is now elevated, and the difference between ongoing activity is now negative (until the average decays down to the ongoing value); these negative relaxation terms will be accumulated into the Hebbian product and cancel the positive, perturbation-related initial term.

To take a simple example, suppose we apply a positive perturbation to a flat signal. The perturbation itself cause a sharp positive deviation from the previous activity. Thus, as expected, subtracting ongoing activity from recent average correctly isolates the received perturbation. But after the perturbation has occurred, the running average (which includes the recent perturbation) is now elevated, and thus the next few time steps of (unperturbed) activity will be lower than the trace, creating spurious negative deviations. If the signal remains flat, the sum of negative deviations will in fact equal and fully cancel out the positive deviation caused by the perturbation

itself, causing spurious weight changes in the opposite direction of the ones created by the perturbation (assuming slowly changing inputs in comparison to the perturbations).

In the Legenstein / Hoerzer method, this problem is addressed by the assumption of a continuous, real-time reward signal, which also undergoes subtraction of a short-term running average. Because the same effect will occur in both the activity trace and the continuous reward trace, the negative "spurious" deviations in both traces, multiplied with each other, will produce a positive addition to the synaptic changes, reinforcing the one created by the perturbation rather than cancelling it. But when the reward signal is sparse and delayed (and fixed for a given trial), the positive perturbation-related deviations are canceled out by the subsequent relaxation-related negative deviations. If we increase the time constant of the running average, the post-perturbation negative deviations become smaller, but they also begin to include an impact from the varying ongoing activity, becoming essentially random and unpredictable.

A simple way to counter this effect is to impose a supralinear function upon the deviations themselves. By imposing a supralinear amplification of the deviations, we would magnify the large deviation from the running average caused by the perturbation itself, and suppress the smaller countering deviations that accumulate as the running average falls back towards the underlying signal value. As a result, the perturbation itself would be successfully isolated from ongoing activity, without requiring either an explicit separation between "normal" and "perturbative" inputs or a continuous, real-time reward signal. The product of perturbations by incoming pre-synaptic activities could then be accumulated over the course of a trial, and multiplied by a single overall reward value at the end of each trial, reproducing the effect of node-perturbation learning.

However, it is not easy to see how a biological plasticity mechanism could selectively apply a nonlinear transformation to the output perturbations. It seems less onerous to assume that the plasticity increments are computed in a standard Hebbian manner, and then amplified nonlinearly - in the sense that overall plasticity would be dominated by larger Hebbian products

and minimize small ones. Note that this is very similar to recently-proposed thresholded Hebbian rules, whereby plasticity is only triggered by events in which the Hebbian product reaches a certain threshold (Soltoggio and Steil, 2013). A supralinear amplification offers a smoother amplification of larger Hebbian events, by comparison to the all-or-nothing effect of a threshold, but the overall effect is similar: ignore small, possibly incidental correlations of input and output, but retain the larger ones, which are more likely to be informative.

Acknowledgements

We thank W. Einar Gall for useful comments and suggestions. We thank Vishwa Goudar for helpful discussions. We thank H. Francis Song for important insight regarding the computation of state-space trajectories in Figure 6. This work was supported by Neurosciences Research Foundation through funding from The G. Harold and Leila Y. Mathers Charitable Foundation and the William and Jane Walsh Charitable Remainder Unitrust, for which we are grateful.

References

Barak O, Sussillo D, Romo R, Tsodyks M, Abbott LF (2013) From fixed points to chaos: three models of delayed discrimination. Prog Neurobiol 103:214–222.

Barak O, Tsodyks M, Romo R (2010) Neuronal population coding of parametric working memory. J Neurosci 30:9424–9430.

Buonomano DV, Maass W (2009) State-dependent computations: spatiotemporal processing in cortical networks. Nat Rev Neurosci 10:113–125.

Churchland MM, Cunningham JP, Kaufman MT, Foster JD, Nuyujukian P, Ryu SI, Shenoy KV (2012) Neural population dynamics during reaching. Nature 487:51–56.

Fiete IR, Fee MS, Seung HS (2007) Model of birdsong learning based on gradient estimation by dynamic perturbation of neural conductances. J Neurophysiol 98:2038–2057.

Fiete IR, Seung HS (2006) Gradient learning in spiking neural networks by dynamic perturbation

of conductances. Phys Rev Lett 97:048104.

Florian RV (2007) Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. Neural Comput 19:1468–1502.

Frémaux N, Sprekeler H, Gerstner W (2010) Functional requirements for reward-modulated spike-timing-dependent plasticity. J Neurosci 30:13326–13337.

Hennequin G, Vogels TP, Gerstner W (2014) Optimal control of transient dynamics in balanced networks supports generation of complex movements. Neuron 82:1394–1406.

Hoerzer GM, Legenstein R, Maass W (2014) Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. Cereb Cortex 24:677–690.

Holzbaur KRS, Murray WM, Delp SL (2005) A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. Ann Biomed Eng 33:829–840.

Izhikevich EM (2007) Solving the distal reward problem through linkage of STDP and dopamine signaling. Cereb Cortex 17:2443–2452.

Jaeger H (2001) The "echo state" approach to analysing and training recurrent neural networks – with an Erratum note1. German National Research Center for Information Technology. Available at: http://web.info.uvt.ro/~dzaharie/cne2013/proiecte/tehnici/ReservoirComputing/EchoStatesTechRep.pdf.

Jun JK, Miller P, Hernández A, Zainos A, Lemus L, Brody CD, Romo R (2010) Heterogenous population coding of a short-term memory and decision task. J Neurosci 30:916–929.

Kober J, Bagnell JA, Peters J (2013) Reinforcement Learning in Robotics: A Survey. Int J Rob Res Available at: http://ijr.sagepub.com/content/early/2013/08/22/0278364913495721.abstract.

Laje R, Buonomano DV (2013) Robust timing and motor patterns by taming chaos in recurrent neural networks. Nat Neurosci 16:925–933.

Legenstein R, Chase SM, Schwartz AB, Maass W (2010) A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. J Neurosci 30:8400–8410.

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

Mante V, Sussillo D, Shenoy KV, Newsome WT (2013) Context-dependent computation by

recurrent dynamics in prefrontal cortex. Nature 503:78–84.

Meyers EM, Freedman DJ, Kreiman G, Miller EK, Poggio T (2008) Dynamic population coding of category information in inferior temporal and prefrontal cortex. J Neurophysiol 100:1407–1419.

Mnih V, Heess N, Graves A, Kavukcuoglu K (2014) Recurrent Models of Visual Attention. In: Advances in Neural Information Processing Systems 27 (Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, eds), pp 2204–2212. Curran Associates, Inc.

Pearlmutter BA (1995) Gradient calculations for dynamic recurrent neural networks: a survey. IEEE Trans Neural Netw 6:1212–1228.

Peters J, Schaal S (2008) Reinforcement learning of motor skills with policy gradients. Neural Netw 21:682–697.

Rajan K, Harvey CD, Tank DW (2016) Recurrent Network Models of Sequence Generation and Memory. Neuron 90:128–142.

Raposo D, Kaufman MT, Churchland AK (2014) A category-free neural population supports evolving demands during decision-making. Nat Neurosci 17:1784–1792.

Rigotti M, Barak O, Warden MR, Wang X-J, Daw ND, Miller EK, Fusi S (2013) The importance of mixed selectivity in complex cognitive tasks. Nature 497:585–590.

Saul KR, Hu X, Goehler CM, Vidt ME, Daly M, Velisar A, Murray WM (2015) Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. Comput Methods Biomech Biomed Engin 18:1445–1458.

Soltoggio A, Steil JJ (2013) Solving the distal reward problem with rare correlations. Neural Comput 25:940–978.

Sompolinsky H, Crisanti A, Sommers HJ (1988) Chaos in random neural networks. Phys Rev Lett 61:259.

Song HF, Yang GR, Wang X-J (2016) Training Excitatory-Inhibitory Recurrent Neural Networks for Cognitive Tasks: A Simple and Flexible Framework. PLoS Comput Biol 12:e1004792.

Stanislas Dehaene J-RK (2016) Decoding the Dynamics of Conscious Perception: The Temporal Generalization Method - Springer. In: Micro-, Meso- and Macro-Dynamics of the Brain (G. Buzsaki YC, ed). Springer.

Stokes MG (2015) “Activity-silent” working memory in prefrontal cortex: a dynamic coding framework. Trends Cogn Sci 19:394–405.

Stokes MG, Kusunoki M, Sigala N, Nili H, Gaffan D, Duncan J (2013) Dynamic coding for

cognitive control in prefrontal cortex. Neuron 78:364–375.

Sussillo D, Abbott LF (2009) Generating coherent patterns of activity from chaotic neural networks. Neuron 63:544–557.

Sussillo D, Churchland MM, Kaufman MT, Shenoy KV (2015) A neural network that finds a naturalistic solution for the production of muscle activity. Nat Neurosci 18:1025–1033.

Thelen DG (2003) Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. J Biomech Eng 125:70–77.

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