One step back, two steps forward: interference and learning in recurrent neural networks

Chen Beer\textsuperscript{1,2}, Omri Barak\textsuperscript{2,3*},

1 Viterby Faculty of Electrical Engineering, Technion Israel Institute of Technology, Haifa, Israel
2 Network Biology Research Laboratories, Technion Israel Institute of Technology, Haifa, Israel
3 Rappaport Faculty of Medicine, Technion Israel Institute of Technology, Haifa, Israel

* omri.barak@gmail.com

Abstract

Artificial neural networks, trained to perform cognitive tasks, have recently been used as models for neural recordings from animals performing these tasks. While some progress has been made in performing such comparisons, the evolution of network dynamics throughout learning remains unexplored. This is paralleled by an experimental focus on recording from trained animals, with few studies following neural activity throughout training.

In this work, we address this gap in the realm of artificial networks by analyzing networks that are trained to perform memory and pattern generation tasks. The functional aspect of these tasks corresponds to dynamical objects in the fully trained network – a line attractor or a set of limit cycles for the two respective tasks. We use these dynamical objects as anchors to study the effect of learning on their emergence. We find that the sequential nature of learning – one trial at a time – has major consequences for the learning trajectory and its final outcome. Specifically, we show that Least Mean Squares (LMS), a simple gradient descent suggested as a biologically plausible version of the FORCE algorithm, is constantly obstructed by forgetting, which is manifested as the destruction of dynamical objects from previous trials. The degree of interference is determined by the correlation between different trials. We show which specific ingredients of FORCE avoid this phenomenon. Overall, this difference results in convergence that is orders of magnitude slower for LMS.

Learning implies accumulating information across multiple trials to form the overall concept of the task. Our results show that interference between trials can greatly affect learning, in a learning rule dependent manner. These insights can help design experimental protocols that minimize such interference, and possibly infer underlying learning rules by observing behavior and neural activity throughout learning.

Author summary

Understanding how neural network dynamics give rise to behavior is a challenging endeavor that has been advanced in recent years. How these dynamics evolve when behavior is acquired, however, remains a mystery. Inspecting the emergence of new building blocks throughout the training of a new task could elucidate what kinds of tasks can be learned, why in some cases learning fails and how previous knowledge is
affected by newly acquired information. Moreover, given a pre-determined set of tasks to learn, such insights could identify which learning sequences result in the best performance in minimal time. In this work, we analyze the process of learning in recurrent neural networks that were trained to perform simple memory tasks. We follow the formation of dynamical structures throughout training to study how it is affected by learning. Our results show that interference between consecutive trials can greatly affect learning, in a learning-rule dependent manner. We inspect two different learning rules and show the specific latent causes for such interference in one, and the exact mechanisms that avoid it in the other. Our framework facilitates the inference of the underlying mechanisms of learning and can thus guide the design of better training protocols.

Introduction

In recent years, trained recurrent neural networks were used as a model for several experimental tasks in neuroscience [1-7]. In particular, recurrent neural networks were both designed [8-11] and trained [2,12-14,15] to memorize analog values and periodical patterns. In the resulting networks, memory is implemented by attractors such as fixed points and limit cycles. The stability properties of these attractors were studied to understand where they arise from, how they can be improved [16], and how they link to experimental observations [3,5,17]. While some progress was made on understanding networks dynamics following training [1,4,18,19], the formation of dynamics throughout learning remains a puzzle.

Learning memory tasks entails two interacting dynamical processes. Within the learning of a single trial, the system is expected to create a stable attractor, representing the memorized target. On a longer timescale, learning the complete task requires integrating sequentially presented trials into a coherent behavior. Sequential learning from many trials was mostly studied in the context of learning multiple tasks. In this case, the danger of forgetting was stressed, and several suggestions were made on how to alleviate it [20-22]. However, forgetting within a single task – from one trial to another – was not explored, and neither the reasons for such cases.

We address these gaps – formation of dynamics and forgetting within a single task – in a simple setting. We train a recurrent neural network on two tasks: a simple memory task and a memory task combined with pattern generation. We use both FORCE learning [23] and its simple, biologically plausible version – LMS. We use reverse engineering to follow the formation of dynamical structures throughout training. We show that, depending on the learning rule used, the sequential nature of learning can cause major interference within a single task. By studying these dynamics, we reveal the underlying causes for training success and failure. Specifically, we show that LMS learning suffers heavily from interference between trials, causing the previous dynamical objects to vanish. We show that this phenomenon results from high correlations between trials, and expose the explicit components of the FORCE algorithm that prevent such destruction. We show that this fundamental difference between the two algorithms, contrary to previous beliefs [23,24], results in orders of magnitude slower convergence of LMS compared to FORCE. Finally, we show how careful ordering of trials can decrease this interference and promote efficient sequential learning.
Results

We use a rate based recurrent neural network, defined by the equations (Fig 1A):

\[
\tau \dot{x}_i = -x_i + \sum_{j=1}^{N} J_{ij} r_j + B_i u + w_i^{FB} z,
\]

(1)

where \(x_i\) is the input to neuron \(i = 1, \ldots, N\), \(J\) is a random connectivity matrix, \(r_i = \phi(x_i)\) is the firing rate with \(\phi(x) = \tanh(x)\), and the external inputs \(u\) are fed through weights \(B_i\). The current state of the network is defined by \(x \in \mathbb{R}^N\) and the output of the network \(z = w_{out}^T r\) is fed back through weights \(w_i^{FB}\).

The network is trained on two memory tasks as follows: The first task is a simple memory task – we present transient stimuli to the network, and train it to output their value for a long time after stimulus removal (Fig 1B, top). The stimulus amplitudes are uniformly sampled from \([1, 5]\), and are presented for 500 ms, followed by a delay period uniformly sampled from \([0.5, 6]\) sec.

The second task is a memory task combined with pattern generation – the network is required to generate a sine wave after receiving a specific frequency value as an input for a limited period of time. The network receives the required frequency as a DC input for 2 cycles and produces the corresponding sine wave for 10 more cycles (Fig 1B, bottom). For this task, we use a two dimensional output – demanding both a sine and a cosine, thereby generating a rank two modification to connectivity.

Fig 1. Network architecture and tasks illustration. (A) Network architecture. Only the outputs weights are modified during training (see methods). (B) Top: Simple memory task. At each trial a stimulus with amplitude in the range \([1, 5]\) is presented for 500 ms (red). The delay period between different stimuli is uniformly distributed between 0.5 and 6 seconds. The desired behavior is to output this value for the entire trial duration (blue). Bottom: Memory with pattern generation task. At each trial a DC stimulus is presented for 2 cycles (red). The desired behavior is to output sine and cosine waves matching to this frequency for 10 cycles (blue, only sine shown).

We trained networks on the simple memory task for 300 trials using either FORCE or LMS, modifying only output weights in both cases (Fig 1A, Methods). Fig 2A shows that while FORCE training resulted in almost perfect performance, this was not the case for LMS. Closer inspection of the LMS output during testing indicates that the network converged to a fixed point corresponding to the last trained stimulus value.
Since in [23, 24] LMS was shown to require roughly 10 times more trials to converge than FORCE, we continued training for a further $10^4$ trials, but to no avail.

Fig 2 shows test results for the memory and pattern generation task – FORCE training resulted in a network which was able to output all different frequencies perfectly, while LMS converged to the last training stimulus, as in the previous task.

**Fig 2. Learning the complete task.** Actual network output (red) using FORCE (top) and LMS (bottom) training. During learning, both algorithms lead to the desired output (grey). Testing, however, reveals that the LMS-trained network converges to the output value of the last training stimulus. The dashed line denotes the beginning of testing. The rightmost plot in (A) shows a test session following an additional $10^4$ training trials. (B) Arrows indicate target switch; testing only 3 cycles.

**Emergence of dynamics**

To gain insight into the underlying reasons for this behavior, we reverse engineer the network to uncover its dynamics. We will now focus on the simple memory task.

Following [25], we define a scalar function $q(x) = \frac{1}{2}|\dot{x}|^2$ which is zero for fixed points. In a successfully trained network, we expect to find an approximate line attractor – a one dimensional manifold of very low $q$-values. Each point along this line represents a memorized value.

As we only train the output weights, the location of this line attractor is pre-determined in the following manner: For a given output value $z(t) \equiv A$, the open loop system (in which the target output $A$ is fed back to the network) converges to a unique stable state $\bar{x}$ which is given by $\bar{x} = J\phi(\bar{x}) + w^{FB}A$. This happens for $A$s that are large enough to suppress chaos [26], which is true for the values used in this work. Fig 3A shows $\bar{x}$ for the entire output range $[1, 5]$ in PCA space.

Training the output weights can determine whether these $\bar{x}$ are indeed fixed points ($w^T_{out}\bar{r} = A$ where $\bar{r} = \phi(\bar{x})$), and whether they are stable. Fig 3B shows the value of $q(\bar{x})$ for different values of $z$. The resulting behavior of the FORCE and LMS trained networks is captured by these graphs. The $q$-values of FORCE are very low for the entire $z$ range, indicating an approximate line attractor. Conversely, LMS leads to a single fixed point around $z = 4.2$, matching the behavior seen in Fig 2A. When inspecting these curves during learning the difference between the two algorithms is
clearly demonstrated – following each trial, a new fixed point emerges in both FORCE and LMS but it remains to exist after the consequent trial only in FORCE (Fig 3C).

Fig 3. The line attractor. (A) The locations $\bar{x}$ for the entire range $z \in [1, 5]$ in PCA space. (B) $q$-values along the expected line attractor following 300 trials of training in either FORCE (yellow) or LMS (green). (C) $q$-values along the expected line attractor during the first 10 trials for both FORCE and LMS. The red stars denote the specific stimulus at each trial.

What enables FORCE-trained networks to have many fixed points, while LMS only leads to a single one? Our results suggest that both algorithms will lead to a fixed point after the first trial, but only FORCE will maintain this fixed point after the second trial. We investigate this hypothesis in a slightly reduced setup.

Reduced setup

In order to disambiguate the pure formation and maintenance of fixed points from input pulse effects and transitions between fixed points, we investigate a reduced setup of the task. Instead of using external inputs to shift between the various output levels, we use the fact that the internal state $\bar{x}$ is precisely defined by output values. We thus use these states as initial conditions for each trial, and eliminate external input (Methods).

Training the network on the first trial leads to a fixed point using either algorithm. But the FORCE algorithm, through its use of recursive least squares, maintains a matrix $P$ that implicitly contains information on past trials. $P$ is a running estimate for the inverse correlation matrix of the network rates $r$ plus a regularization term:

$$P = \left( \int r(t)r^T(t) dt + \alpha I \right)^{-1}. \tag{2}$$
The eigenvectors of $P$ are thus approximately the principal components of the activity, but since $P$ retains the inverse of the activity history, the eigenvalues of these principle components will be very small. The update of synaptic weights in FORCE is proportional to $P$ (Eq 3), and thus protects the directions most visited (e.g. fixed points) from further change.

Now that we understand the information stored in $P$, we can examine what happens during the second trial (Fig 4). We denote the targets of the two trials $A_1$ and $A_2$ in the simple memory task (with the corresponding putative fixed points $\bar{r}_1$ and $\bar{r}_2$), or $f_1$ and $f_2$ in the pattern generation task. In the simple memory task (Fig 4A), training with FORCE results in two stable fixed points, indicating that $r_1$ was preserved. Training with LMS causes the network to forget the first value, and as a result, only one fixed point corresponding to the last value $A_2$ remains. Correspondingly, in the pattern generation task (Fig 4B), training with FORCE results in two stable limit cycles, while in LMS after the second trial there is a single limit cycle matching $f_2$. When resetting the $P$ matrix after the first trial to be $(\alpha I)^{-1}$, the resulting dynamics are the same as with LMS training.

Fig 4. The first two trials. (A) Simple memory task: The second trial is learned via FORCE, leading to two stable fixed points or via LMS, leading to a single fixed point. Top row: Testing after two training trials – the output during testing the two targets separately. Second row: $q$-values along the expected final line attractor after the first (grey) and second (black) trials. (B) Memory with pattern generation: Learning with FORCE leads to two stable limit cycles while learning with LMS results in only one.

These results demonstrate how information regarding the existence of fixed points is stored in $P$ during the learning process, and is used to predict these points.

Interference in LMS

The results shown so far indicate a qualitative difference between FORCE and LMS learning mechanisms due to second order information used by FORCE [27,28]. The Hessian, which is the second order derivative of the loss, is given by:

$$\frac{\partial^2}{\partial w^2} \frac{1}{2}(w^T r - f)^2 = r^T r,$$

and is stored in $P$ (Eq 2). This information enables the maintenance of previous knowledge. In our setting, the rate vectors corresponding to different points along the line attractor are highly correlated. Therefore, when learning these rate vectors sequentially, first order algorithms like LMS suffer from constant interference, leading to what seems to be a complete forgetting of the past. Inspecting this phenomenon thoroughly uncovers the slow underlying processes.
We consider the same example of memorizing only two target values, this time for two cases that differ in the correlation between \( \bar{r}_1 \) and \( \bar{r}_2 \). When the correlation is high, viewing the validation error throughout learning might imply that no learning is taking place (Fig 5A, left, yellow line). On the other hand, observing the projections of \( w_{out} \) onto the plane spanned by the corresponding fixed points \( \bar{r}_1, \bar{r}_2 \) reveals the slow underlying process. The black lines in Fig 5A show zero error for either target, and their intersection (Red dot) is the optimal solution. The training trajectory (white) reveals that learning is constantly interfered (Fig 5A, bottom), but the independent components of \( \bar{r}_1 \) and \( \bar{r}_2 \) survive. The network converges eventually when training one target no longer affects the readout from the other and the following equalities hold: \( w_{out}^T \bar{r}_1 = A_1 \), \( w_{out}^T \bar{r}_2 = A_2 \). When \( \bar{r}_1 \) and \( \bar{r}_2 \) are less correlated, the amount of interference decreases, leading to much faster convergence (Fig 5A, right).

Fig 5. Interference in LMS. (A) Simple memory task for only two values \( A_1, A_2 \) repeatedly presented while learning using LMS. Top: average \( q \)-values measured at the corresponding fixed points \( \bar{r}_1, \bar{r}_2 \) (green) and the validation error during learning (yellow); left: \( A_1 = 2, A_2 = 2.5 \), right: \( A_1 = 2, A_2 = 5 \). Bottom: the projections of \( w_{out} \) onto \( v_1 = \bar{r}_1 \) and \( v_2 = \bar{r}_2 - \langle \bar{r}_1, \bar{r}_2 \rangle \frac{\bar{r}_1}{\langle \bar{r}_1, \bar{r}_1 \rangle} \) (white); background colors represent the average error, the red dot denotes the optimal solution, and the black lines are the local minima for each target value. The inset on the left highlights the destructive interference at each step in the case where \( \bar{r}_1 \) and \( \bar{r}_2 \) are more correlated. (B) Number of trials per target required to achieve 1% error as a function of the number of targets for LMS (blue) and FORCE (green). Target values are sampled from \([1, 5] \); training continues as long as the error is higher than 1%. The solid line denotes the mean of 5 simulations. Light blue shade denotes the standard deviation across simulations.

The complete task of memorizing a wide range of values requires the formation of a line attractor – a continuous set of highly correlated fixed points. Fig 5B shows the number of trials required for LMS and FORCE as the number of target values increases. The detrimental effects of interference rapidly add up and lead to diverging convergence times.

Minimizing interference

In both the training of animals (shaping, [29]) and of artificial networks (curriculum learning, [30]), it has been observed that the order of training can have an effect on the final performance. Our results indicate an interplay between the correlation of
sequentially learned fixed points and the process of learning. We hypothesize that this feature can be used to design optimal training sequences.

In the two-target case above, the correlation between consecutive stimuli was given by the choice of targets. When learning a full line attractor, however, the precise sequence matters. As a proof of concept, we compare learning when the target values are presented in an ordered fashion (Fig 6, purple) or in a random ordering (green). The ordered sequence has very high correlations between neighboring targets. The random sequence is expected to be more balanced, with lower correlation values. For each ordering, we can calculate the sum of correlation coefficients along the sequence. Indeed, considering the maximal and minimal sums attainable, the former is obtained by the ordered sequence, whereas random sequences result in intermediate values. Based on our previous results (Fig 5), we expect these sums to affect the level of interference during learning. When using LMS, training with a random sequence proceeds much faster compared to the ordered sequence (though the error still remains high after many trials). In this case, each new fixed point is sufficiently different from the previous, and therefore, the interference during learning is smaller. Moreover, when comparing the error decay of a random sequence training to the one obtained by the minimal sum sequence – the results are very similar (Methods). As demonstrated in previous sections, FORCE is able to protect previously visited fixed points, and therefore, the difference between random and ordered training sequences is not significant.

**Fig 6. Order of trials affects performance.** Average error along the learning process for two learning sequences – random (green) and ordered (purple) for both FORCE and LMS. The solid lines denote the mean of 10 simulations; light shades denote the standard deviation across simulations.

**Discussion**

In this work we followed the formation of dynamical objects through the online training of a recurrent neural network. To enable a feasible analysis we designed two simple memory tasks, each one requires a different dynamical object in the resulting dynamics. We showed that FORCE, an RLS based algorithm, avoids forgetting by implicitly avoiding updates to previously visited directions. In contrast, we showed that LMS learning suffers from constant interference, leading to recurring elimination of what was learned in earlier stages. By dissecting the contributions of the various parts of a trial to the implicitly acquired memory, we could understand how the formation and maintenance properties of these dynamical objects lead to learning.

We focused on the FORCE algorithm due to its simplicity and usage within neuroscience inspired tasks [13,31]. This algorithm, however, is not biologically plausible. Both LMS and a reward based rule [24] were suggested as slower but effective plausible versions of FORCE. Our results indicate that this might not be the case in
more complex tasks. We demonstrate the qualitative differences between FORCE and 
LMS learning mechanisms, leading to orders of magnitude difference in convergence 
rate, which rapidly increases with task complexity.

Aiming to understand the learning process of cognitive tasks, we addressed the case 
of online learning, in which learning takes place continuously while the trials are 
presented sequentially. This differs from the typical machine learning setting of batch 
learning, in which several stimuli are presented in parallel. In our setting, online 
learning is manifested by the formation of a new dynamical object after each trial either 
in FORCE or LMS. When using very small learning rates in LMS, such dynamics are 
not formed after a single trial, and therefore it is equivalent to presenting multiple trials 
at the same time. In this case, interference between trials is not exhibited, and the 
network can eventually converge to the desired dynamics. However, this scenario 
describes a case of batch learning (as the learning rate approaches zero), and therefore it 
is not addressed here.

Although we stress the sequential nature of learning, the fact that this is a single 
task leads to important distinctions from the sequential learning of uncorrelated tasks. 
In the simple memory task, the rate vectors corresponding to different points along the 
line attractor are highly correlated. Our results show that these correlations are the 
main cause for the extensive interference during LMS learning, seemingly leading to 
catastrophic forgetting within a single task.

Shaping, or choosing an optimal sequence of training examples, is a topic of 
importance in both animal training and machine learning research [29,30]. Here, we 
show that correlations between sequential trials could indicate difficulty, and suggest to 
decrease interference within trials and accelerate learning by minimizing these 
correlations. In the future, it might be possible to probe the system to understand its 
neural correlates properties, and use this information to guide the ordering of the 
following stimuli to reduce interference during training. Our results indicate no 
significant difference between random and optimal ordering, therefore, these online 
improvements might be simple to achieve. Such an online choice of the next training 
target is similar in spirit to adaptive choice of stimuli that was suggested as an efficient 
method to estimate response properties of neurons [32].

Understanding the dynamics of trained recurrent neural networks is an important 
and difficult challenge [6,7,33]. Understanding the formation of these dynamics is an 
even more formidable task. Here, we take a first step towards this goal and show that it 
sheds light on the nature of learning rules, and can provide clues on how to improve 
learning.

Methods

Learning rules. We inspect the learning process using two different training 
methods: FORCE algorithm [23], based on recursive least squares (RLS), with the 
following update rule:

\[
\begin{align*}
w(t) &= w(t - \Delta t) - e^-(t) P(t) r(t), \\
e^-(t) &= w^T(t - \Delta t) r(t) - f(t),
\end{align*}
\]

where \( P(t) \) is a running estimate of the inverse correlation matrix of the network rates \( r \) 
plus a regularization term, and \( f(t) \) is the target output.

Although it is widely used, FORCE is not biologically plausible since the 
modification of a given synapse depends on information from the entire neural 
population. These locality considerations led the authors of [23] to suggest a local 
learning rule: Least mean squares (LMS), in which the modification rule for the output
weights is

\[ w(t) = w(t - \Delta t) - \eta(t) e^{-t} r(t), \tag{5} \]

where \( e^{-t} \) is defined as in FORCE, and \( \eta(t) \) is a time-varying learning rate:

\[ \frac{d\eta}{dt} = \eta(-\eta + |e^{-t}|). \]

A further step towards biological realism was made by the introduction of a reward based learning rule \[24\]. This rule was described as leading to convergence on the same set of tasks as FORCE, albeit with a convergence rate that is slower than FORCE and similar to LMS.

We consider the case where only the output weights are modified during training (Fig 1A, red), which facilitates our analysis. Because of the feedback connection, training still modifies network dynamics, and thus allows successful training of the task. We also verified that the main effects are qualitatively similar when training the internal connections as well.

**Simulation parameters.** In all simulations \( J \) elements were independently sampled from \( \mathcal{N}(0, g^2/N) \) with \( N = 500 \) and \( g = 1.2 \). The input weights \( B \) and the feedback weights \( w^{FB} \) were drawn from a uniform distribution between \(-1\) and \(1\). \( w_{out} \) were initially set to zero. \( \tau = 100 \) ms. In FORCE: \( \alpha = 10 \). For LMS, we used \( \gamma \) values of \( 1, 2, 5 \), and found that the results were qualitatively similar to a constant \( \eta \). Therefore, to reduce the number of parameters, we used constant \( \eta \) for all figures.

**Reduced setup.** Instead of using external inputs \( u \) to shift between output levels, we use the fact that the internal state \( \bar{x} \) is precisely defined by the target values. We thus use these states as initial conditions for each trial, and eliminate external input. Therefore, the network dynamics is defined by the equations:

\[ \tau \dot{x}_i = -x_i + \sum_{j=1}^{N} J_{ij} r_j + w^{FB}_i z, \tag{6} \]

and at the beginning of each trial the network state is initialized to the fixed point corresponding to target value \( A \):

\[ \bar{x} = J \phi(\bar{x}) + w^{FB} A. \tag{7} \]

**Figure 5B.** Networks were trained using both LMS and FORCE to perform the simple memory task for 1,2,3,4 and 5 target values. Training stopped when reaching 1\% error difference between \( w^T_{out} r_i \) and the desired output \( A_i \). FORCE training converges after a single trial for each value, while the number of trials required for LMS grows exponentially with the number of targets. Results shown for LMS were obtained using the fastest learning rate that converged to the desired solution: For a single target training \( \eta = 10^{-2} \) and for 2,3,4 and 5 targets \( \eta \in [10^{-3}, 10^{-5}] \). Target values were \([1, \ldots, 9] \) and \([1, \ldots, 5] \).

**Figure 6** Networks were trained using both LMS and FORCE to perform the simple memory task for 9 target values in the range \([1, 5]\). Two training sequences were addressed – ordered sequence, from 1 to 5 and back to 1 repeatedly, and random sequence. In both cases, each target was presented at least 10 times.

In addition to these two sequences, we tested the minimal-sum sequence: For given 9 target values, we can find the corresponding 9 fixed points and calculate the correlation coefficients between them. For a single training round, in which each trial is presented exactly once, we can find the sequence which results in the minimal correlation sum from one trial to the other. When presenting this sequence 10 times repeatedly, the validation error throughout training is very similar to the one obtained using a random sequence.
Acknowledgments

We thank Friedrich Schuessler and Lee Susman for comments on the manuscript.

References

1. Mante V, Sussillo D, Shenoy KV, Newsome WT. Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature. 2013;503(7474):78–84. doi:10.1038/nature12742.

2. Miconi T. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. eLife. 2017;6. doi:10.7554/eLife.20899.

3. Russo AA, Bittner SR, Perkins SM, Seely JS, Brian M, Lara AH, et al. Motor cortex embeds muscle-like commands in an untangled population response Abigail. Neuron. 2018;97(4):953–966. doi:10.1016/j.neuron.2018.01.004.Motor.

4. Remington ED, Narain D, Hosseini EA, Jazayeri M. Flexible Sensorimotor Computations through Rapid Reconfiguration of Cortical Dynamics. Neuron. 2018;98(5):1005–1019.e5. doi:10.1016/j.neuron.2018.05.020.

5. Wang J, Narain D, Hosseini EA, Jazayeri M. Flexible timing by temporal scaling of cortical responses. Nature Neuroscience. 2018;21(1):102–110. doi:10.1038/s41593-017-0028-6.

6. Sussillo D. Neural circuits as computational dynamical systems; 2014. Available from: https://www.sciencedirect.com/science/article/pii/S0959438814000166?via%3Dihub.

7. Barak O. Recurrent neural networks as versatile tools of neuroscience research. Current Opinion in Neurobiology. 2017;46:1–6. doi:10.1016/J.CONB.2017.06.003.

8. Machens CK, Romo R, Brody CD. Flexible control of mutual inhibition: A neural model of two-interval discrimination. Science. 2005;307(5712):1121–1124. doi:10.1126/science.1104171.

9. Miller P, Brody CD, Romo R, Wang XJ. A Recurrent Network Model of Somatosensory Parametric Working Memory in the Prefrontal Cortex. Cerebral Cortex. 2003;13(11):1208–1218. doi:10.1093/cercor/bhg101.

10. Ben-Yishai R, Bar-Or RL, Sompolinsky H. Theory of orientation tuning in visual cortex. Proceedings of the National Academy of Sciences. 1995;92(9):3844–3848. doi:10.1073/pnas.92.9.3844.

11. Singh R. Higher-Dimensional Neurons Explain the Tuning and Dynamics of Working Memory Cells. Journal of Neuroscience. 2006;26(14):3667–3678. doi:10.1523/JNEUROSCI.4864-05.2006.

12. Emin Orhan A, Ma WJ. Efficient probabilistic inference in generic neural networks trained with non-probabilistic feedback. Nature Communications. 2017;8(1). doi:10.1038/s41467-017-00181-8.

13. Barak O, Sussillo D, Romo R, Tsodyks M, Abbott LF. From fixed points to chaos: Three models of delayed discrimination: 2013. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3622800/pdf/nihms448582.pdf.
14. Song HF, Yang GR, Wang XJ. Reward-based training of recurrent neural networks for cognitive and value-based tasks. eLife. 2017;6:e21492. doi:10.7554/eLife.21492.

15. Song HF, Yang GR, Wang XJ. Training Excitatory-Inhibitory Recurrent Neural Networks for Cognitive Tasks: A Simple and Flexible Framework. PLoS Computational Biology. 2016;12(2):e1004792. doi:10.1371/journal.pcbi.1004792.

16. Itskov V, Hansel D, Tsodyks M. Short-Term Facilitation may Stabilize Parametric Working Memory Trace. Frontiers in Computational Neuroscience. 2011;5:40. doi:10.3389/fncom.2011.00040.

17. Wimmer K, Nykamp DQ, Constantinidis C, Compte A. Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory. Nature Neuroscience. 2014;17(3):431–439. doi:10.1038/nn.3645.

18. Sussillo D, Churchland MM, Kaufman MT, Shenoy KV. A neural network that finds a naturalistic solution for the production of muscle activity. Nature Neuroscience. 2015;18(7):1025–1033. doi:10.1038/nn.4042.

19. Yang GR, Joglekar MR, Song HF, Newsome WT, Wang XJ. Task representations in neural networks trained to perform many cognitive tasks. Nature Neuroscience. 2019;22(2):297–306. doi:10.1038/s41593-018-0310-2.

20. Kemker R, McClure M, Abitino A, Hayes T, Kanan C. Measuring Catastrophic Forgetting in Neural Networks. Proceedings of the National Academy of Sciences of the United States of America. 2017;114(13):3521–3526. doi:10.1073/pnas.1611835114.

21. Zenke F, Poole B, Ganguli S. Continual Learning Through Synaptic Intelligence. 2017;doi:10.1073/pnas.1611835114.Martens.

22. Zeno C, Golan I, Hoffer E, Soudry D. Bayesian Gradient Descent: Online Variational Bayes Learning with Increased Robustness to Catastrophic Forgetting and Weight Pruning. 2018;doi:10.1016/S0040-4039(00)97456-2.

23. Sussillo D, Abbott LF. Generating coherent patterns of activity from chaotic neural networks. Neuron. 2009;63(4):544–57. doi:10.1016/j.neuron.2009.07.018.

24. Hoerzer GM, Legenstein R, Maass W. Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning. Cerebral Cortex. 2014;24(3):677–690. doi:10.1093/cercor/bhs348.

25. Sussillo D, Barak O. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. Neural Computation. 2013;25(3):626–649. doi:10.1162/NECO_a_00409.

26. Rajan K, Abbott LF, Sompolinsky H. Stimulus-dependent suppression of chaos in recurrent neural networks. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics. 2010;82(1):1–5. doi:10.1103/PhysRevE.82.011903.

27. Vaits N, Moroshko E, Crammer K. Second-Order Non-Stationary Online Learning for Regression:. Available from: https://arxiv.org/pdf/1303.0140.pdf

28. Gruber MHJ, Hayes MH. Statistical Digital Signal Processing and Modeling. Technometrics. 1997;39(3):335. doi:10.2307/1271141.
Supporting information

S1 Appendix. Existence and stability. In our simple memory task (even in the reduced setup), each trial begins with a short transient until the network converges into the fixed point corresponding to the current target value. Assuming that the time spent at the fixed point $\bar{r}$ is significantly longer than the transient period, we can write the following approximation for $P$ following the first trial:

$$\hat{P} = (T\bar{r}\bar{r}^T + \alpha I)^{-1}, \quad (8)$$

where $T$ is trial duration. The eigenvalues of $P$ and $\hat{P}$ after a single trial are shown in Fig 7A. The lowest eigenvalue in both matrices is $(\alpha + T\bar{r}\bar{r}^T)^{-1}$, which corresponds to the eigenvector $\bar{r}$. The transient history is reflected in the eigenvalues of the nominal $P$, and not in $\hat{P}$.

Performing the two trials training using $\hat{P}$, which contains information on the fixed point $\bar{r}_1$, but not the transients leading to it, results in an intermediate behavior – the resulting network contains two fixed points, but only the one corresponding to the last value is stable.
Fig 7. Partial inverse correlation matrix. (A) The last 10 eigenvalues of $P$ and $\tilde{P}$ following FORCE training of the first trial. The remaining $N - 10$ equal $1/\alpha$ in both cases. Network and training parameters: $N = 1000$, $g = 1.2$, $\alpha = 1$ and target value $A_1 = 1$. (B) FORCE training using $\tilde{P}$ leading to one stable and one unstable fixed point. (C) The spectrum of the network linearized around $\bar{r}_1$ after the second trial.

In our case, the rate vectors corresponding to different points along the line attractor are highly correlated. This is the reason it is possible to achieve stability even when removing the transient contributions to $P$. 