4-26-2016

Memory and Information Processing in Recurrent Neural Networks

Alireza Goudarzi
University of New Mexico, Albuquerque, NM

Sarah Marzen
University of California Berkeley

Peter Banda
University of Luxembourg, tremor11@gmail.com

Guy Feldman
Purdue University

Matthew R. Lakin
University of New Mexico, Albuquerque, NM

See next page for additional authors

Citation Details

Goudarzi, A., Marzen, S., Banda, P., Feldman, G., Teuscher, C., & Stefanovic, D. (2016). Memory and Information Processing in Recurrent Neural Networks. Neural and Evolutionary Computing. https://arxiv.org/abs/1604.06929

Memory and Information Processing in Recurrent Neural Networks

Alireza Goudarzi[1], Sarah Marzen[2], Peter Banda[3], Guy Feldman[4],
Matthew R. Lakin[1], Christof Teuscher[5] and Darko Stefanovic[1]
[1] *University of New Mexico, Albuquerque, NM 87131*
[2] *University of California Berkeley, Berkeley, CA 94720*
[3] *University of Luxembourg, Belvaux, Luxembourg L-4367*
[4] *Purdue University, West Lafayette, IN 47907*
[5] *Portland State University, Portland, OR 97206*
(Dated: April 26, 2016)

Recurrent neural networks (RNN) are simple dynamical systems whose computational power has been attributed to their short-term memory. Short-term memory of RNNs has been previously studied analytically only for the case of orthogonal networks, and only under annealed approximation, and uncorrelated input. Here for the first time, we present an exact solution to the memory capacity and the task-solving performance as a function of the structure of a given network instance, enabling direct determination of the function–structure relation in RNNs. We calculate the memory capacity for arbitrary networks with exponentially correlated input and further related it to the performance of the system on signal processing tasks in a supervised learning setup. We compute the expected error and the worst-case error bound as a function of the spectra of the network and the correlation structure of its inputs and outputs. Our results give an explanation for learning and generalization of task solving using short-term memory, which is crucial for building alternative computer architectures using physical phenomena based on the short-term memory principle.

Excitable dynamical systems, or reservoirs, store a short-term memory of a driving input signal in their instantaneous state [1]. This memory can produce a desired output in a linear readout layer, which can be trained efficiently using ordinary linear regression or gradient descent. This paradigm, called *reservoir computing* (RC), was originally proposed as a simplified model of information processing in the prefrontal cortex [2]. It was later generalized to explain computation in cortical microcircuits [3] and to facilitate training in recurrent neural networks [4]. A central feature of RC is the lack of fine tuning of the underlying dynamical system: any random structure that guarantees a stable dynamics gives rise to short-term memory [3, 4]. Analogous behavior has also been observed in selective response in random neural populations [6]. Furthermore, fixed underlying structure in RC makes it suitable for implementing computation using spatially distributed physical phenomena[13–15, 17–21]. Such approaches can give us a way to store and process information more efficiently than with von Neumann architecture [7].

Short-term memory in neural networks has been studied for uncorrelated input $u(t)$ under annealed approximation, i.e., connectivity is resampled independently at each time step [8]. That study considered only linear orthogonal networks, where the columns of the connectivity matrix are pairwise orthogonal and the node transfer functions are linear. A memory function $m(\tau)$ was defined to measure the ability of the system to reconstruct input from τ time steps ago, i.e., $u(t-\tau)$, from the present system state $x(t)$. It was shown that the total memory capacity cannot exceed the N degrees of freedom in the system. For networks with saturating nonlinearity, the memory scales with $\sqrt{N}$ [1]; however, by fine-tuning the nonlinearity one can achieve near-linear scaling of memory capacity [9]. In nonlinear networks, it is very difficult to analyze the complete memory function and even harder to relate it to the performance on computational tasks, as is evident from many works in this area with hard-to-reconcile conclusions (see Ref. [5]).

Model— Consider a discrete-time network of N nodes. The network weight matrix $\mathbf{\Omega}$ is $N \times N$ with spectral radius $\lambda < 1$. A time-dependent scalar input signal u_t is fed to the network using the input weight vector $\boldsymbol{\omega}$. The evolution of the network state x_t and the output y_t is governed by

$$x_{t+1} = \mathbf{\Omega} x_t + \boldsymbol{\omega} u_t, \text{ and} \tag{1}$$

$$y_{t+1} = \mathbf{\Psi} x_{t+1}, \tag{2}$$

where $\mathbf{\Psi} = (\mathbf{X}\mathbf{X}^\intercal)^{-1}\mathbf{X}\widehat{\mathbf{Y}}^\intercal$ is an N-dimensional column vector calculated for a desired output $\widehat{y}_t$. Here, each column of $\mathbf{X}$ is the state of the network at time x_t and each column of $\widehat{\mathbf{Y}}^\intercal$ is the corresponding desired output at each time step. In practice it is sometimes necessary to use Tikhonov regularization to calculate the readout weights, i.e., $\mathbf{\Psi} = \left(\mathbf{X}\mathbf{X}^\intercal + \gamma^2\mathbf{I}\right)^{-1}\mathbf{X}\widehat{\mathbf{Y}}^\intercal$, where γ is a regularization factor that needs to be adjusted depending on $\mathbf{\Omega}, \boldsymbol{\omega}$, and u_t [5].

Calculating $\mathbf{\Psi}$ for a given problem requires the following input-output-dependent evaluations (Appendix A):

$$\mathbf{X}\mathbf{X}^\intercal = \sum_{i,j=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega}\boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^j R_{uu}(i-j), \text{ and} \tag{3}$$

$$\mathbf{X}\mathbf{Y}^\intercal = \sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{u\widehat{y}}(i), \tag{4}$$

where $R_{uu}(i-j) = \langle u_t u_{t-(i-j)} \rangle$ and $R_{u\widehat{y}}(i-j) = \langle u_t \widehat{y}_{t-(i-j)} \rangle$ are the autocorrelation of the input and the

cross-correlation of the input and target output. This may also be expressed more generally in terms of the power spectrum of the input and the target:

$$\mathbf{X}\mathbf{X}^\intercal = \frac{1}{2T}\int_{-T}^{T} \mathbf{\Omega}_+^{-1}\boldsymbol{\omega} S_{uu}(f)\boldsymbol{\omega}^\intercal \mathbf{\Omega}_-^{-1} df, \tag{5}$$

$$\mathbf{X}\mathbf{Y}^\intercal = \frac{1}{2T}\int_{-T}^{T} \mathbf{\Omega}_+^{-1}\boldsymbol{\omega} S_{u\widehat{y}}(f) e^{if\tau} df. \tag{6}$$

where $\mathbf{\Omega}_+ = \left(\mathbf{I} - e^{if}\mathbf{\Omega}\right)$ and $\mathbf{\Omega}_- = \left(\mathbf{I} - e^{-if}\mathbf{\Omega}\right)$, $S_{uu}(f)$ is the power spectral density of the input, and $S_{u\widehat{y}}(f)$ is the cross-spectral density of the input and the target output.

The performance can be evaluated by the mean-squared-error (MSE) as follows:

$$\langle E^2 \rangle = \left\langle \left(\widehat{y}(t) - y(t)\right)^2 \right\rangle = \widehat{\mathbf{Y}}\widehat{\mathbf{Y}}^\intercal - \widehat{\mathbf{Y}}\mathbf{Y} \tag{7}$$

$$= \widehat{\mathbf{Y}}\widehat{\mathbf{Y}}^\intercal - \widehat{\mathbf{Y}}\mathbf{X}^\intercal \left(\mathbf{X}\mathbf{X}^\intercal\right)^{-1}\mathbf{X}\widehat{\mathbf{Y}}^\intercal. \tag{8}$$

The MSE gives us a distribution-independent upper bound on the instantaneous-squared-error through the application of Markov inequality:

$$P\left[\left(\widehat{y}(t) - y(t)\right)^2 \geq a\right] \leq \frac{\langle E^2 \rangle}{a}. \tag{9}$$

The existence of a worst-case bound is important for engineering applications of RC.

Memory and task solving— The memory function of the system is defined as [8]

$$m(\tau) = (\mathbf{Y}\mathbf{X}^\intercal)(\mathbf{X}\mathbf{X}^\intercal)^{-1}(\mathbf{X}\mathbf{Y}^\intercal), \tag{10}$$

where $\mathbf{Y}$ is the input with lag τ, $u_{t-\tau}$.

The exact evaluation of this function has previously proved elusive for arbitrary $\mathbf{\Omega}$ and $\boldsymbol{\omega}$. Here we provide a solution using eigendecomposition of $\mathbf{\Omega}$ and the power spectral density of the input signal. The solution may also be described directly in terms of the autocorrelation of the input, as in Appendix B.

Let $\mathbf{\Omega} = \mathbf{U}\mathrm{diag}(\mathbf{d})\mathbf{U}^{-1}$ and $\bar{\boldsymbol{\omega}} = \mathbf{U}^{-1}\boldsymbol{\omega}$ so that

$$\left(\mathbf{I} - e^{if}\mathbf{\Omega}\right)^{-1} = \mathbf{U}\mathrm{diag}\left(\frac{1}{1 - e^{if}\mathbf{d}}\right)\mathbf{U}^{-1}. \tag{11}$$

The memory function is reduced to

$$m(\tau) = \frac{1}{2T}\bar{\boldsymbol{\omega}}^\top \left(A \circ C^{-1}\right)\bar{\boldsymbol{\omega}}, \tag{12}$$

where the matrix C is given by

$$C = \int_{-T}^{T} \left[\frac{\bar{\boldsymbol{\omega}}}{1 - e^{if}\mathbf{d}}\right] \otimes \left[\frac{\bar{\boldsymbol{\omega}}}{1 - e^{-if}\mathbf{d}}\right] S_{uu}(f) df, \tag{13}$$

and the matrix A is given by

$$A = \left[\int_{-T}^{T} \frac{S_{u\widehat{y}}(f)e^{if\tau}}{1 - e^{if}\mathbf{d}} df\right] \otimes \left[\int_{-T}^{T} \frac{S_{u\widehat{y}}(f)e^{if\tau}}{1 - e^{if}\mathbf{d}} df\right]. \tag{14}$$

The total memory is then given by

$$\sum_{\tau=0}^{\infty} m(\tau) = \frac{1}{2T}\bar{\boldsymbol{\omega}}^\top \left(B \circ C^{-1}\right)\bar{\boldsymbol{\omega}} \tag{15}$$

where

$$B = \int_{-T}^{T}\int_{-T}^{T} \frac{df\,df'}{1 - e^{i(f+f')}} \left[\frac{S_{u\widehat{y}}(f)}{1 - e^{if}\mathbf{d}}\right] \otimes \left[\frac{S_{u\widehat{y}}(f')}{1 - e^{if'}\mathbf{d}}\right]. \tag{16}$$

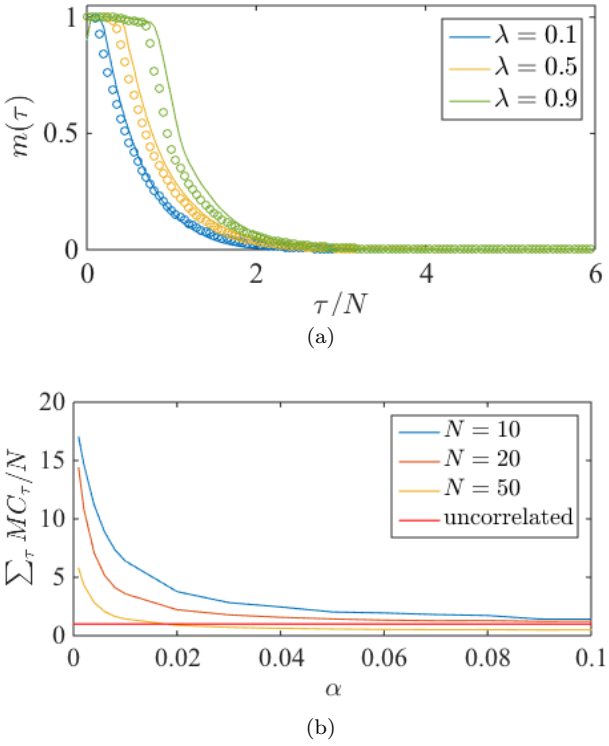

FIG. 1. (a) Agreement of analytical and empirical memory function for different λ. (b) Scaling of memory capacity with increasing structure in the input.

We validate our formula by comparing analytical and empirical evaluation of the memory curve. The input is assumed to be a sequence of length T with autocorrelation function $R(\tau) = e^{-\alpha\tau}$ (Appendix C). FIG 1(a) shows the result of the single-instance calculation of the analytical and the empirical memory curves for different λ (Appendix C). As expected, the analytical and empirical results agree.

We also study the memory capacity for different levels of structure in the input signal. Here, we use simple ring topologies with $\lambda = 0.9$ and vary the decay exponent α, FIG 1(b). For a fixed system size, decreasing α exponentially increases the correlation in the input, which increases the memory capacity.

Next, we use our method to calculate the optimal output layer, the expected average error, and bounds on

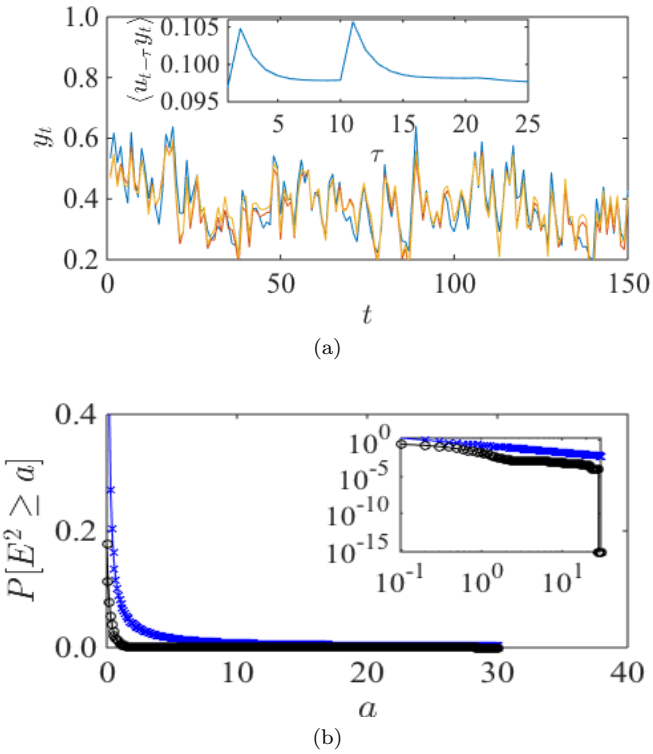

FIG. 2. Target output and system output generated with analytical weights and trained weights for the NARMA10 task (a), the worst-case bounds using the Markov inequality, with the same plot on a log-log scale in the inset (b).

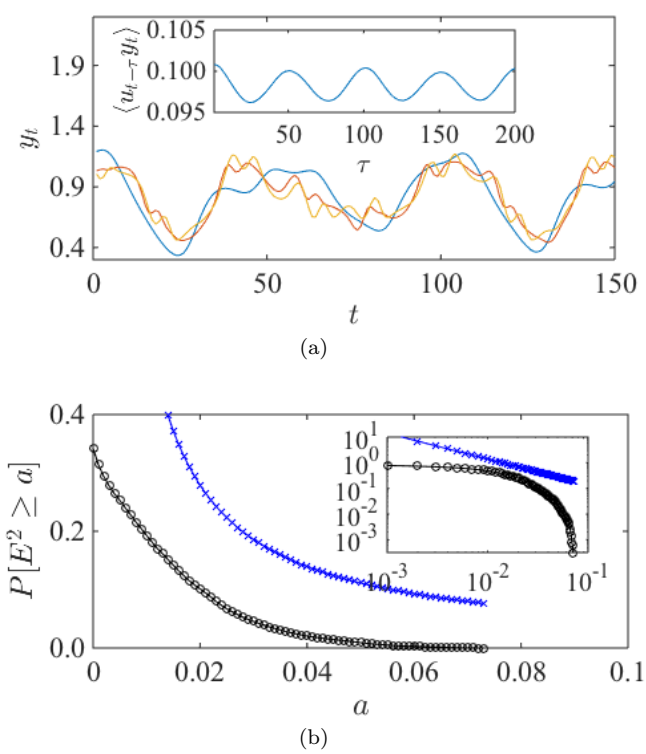

FIG. 3. Target output and system output generated with analytical weights and trained weights for the Mackey-Glass 10 step ahead prediction task (c), the worst-case bounds using the Markov inequality, with the same plot on a log-log scale in the inset (d).

worst-case error for a common NARMA10 benchmark task:

$$y_t = \alpha y_{t-1} + \beta y_{t-1} \sum_{i=1}^{n} y_{t-i} + \gamma u_{t-n} u_{t-1} + \delta, \qquad (17)$$

where $n = 10$, $\alpha = 0.3, \beta = 0.05, \gamma = 1.5, \delta = 0.1$. The input u_t is drawn from a uniform distribution over the interval $[0, 0.5]$. Here the evaluation of $\mathbf{XX}^\intercal$ follows the same calculation as for the memory capacity for the uniform distribution. For $\mathbf{XY}^\intercal$ we must estimate the cross-correlation of y_t and u_t and substitute it into Equation 4. FIG 2(a) shows the output of a network of $N = 20$ nodes and $\lambda = 0.9$ with analytically calculated optimal output layer. The output agrees with the correct output of the NARMA10 system. The cross-correlation of the system used for the calculation is shown in the inset. FIG 2(b) shows the worst-case error bound for this system and the empirical errors generated from the system output, showing that the bound we derived is tight.

Finally, we show how the framework can be used in a prediction scenario, namely the prediction of the Mackey-Glass system. The Mackey-Glass system [22] was first proposed as a model for feedback systems that may show different dynamical regimes. The system is a one-dimensional delayed feedback differential equation and manifests a wide range of dynamics, from fixed points to strange attractors with varying divergence rates (Lyapunov exponent). This system has been used as a benchmark task for chaotic signal prediction and generation [4]. It is defined as:

$$\frac{dx(t)}{dt} = \beta \frac{x(t-\tau)}{1 + x(t-\tau)^n} - \gamma x(t), \qquad (18)$$

where $\beta = 2, \gamma = 1, n = 9.7451, \tau = 2$ ensure the chaoticity of the dynamics [4].

FIG 2(c) shows the prediction result for 10 time steps ahead and the inset shows the autocorrelation at different lags. The autocorrelation is characterized by a long correlation length evident from non-zero correlation values for large τ. This long memory is a hallmark of chaotic systems. We use this information to evaluate Equation 3 and Equation 4, where for predicting τ steps ahead we have $\mathbf{XY}^\intercal = \sum_{i=\tau}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{u\tilde{y}}(i)$.

The effect of network structure— The effect of randomness and sparsity of reservoir connectivity has been a subject of debate [5]. To study the effect of network structure on memory and performance, we systematically explore the range between sparse deterministic uniform networks and random graphs. We start from a simple ring topology with identical weights and induce noise to ℓ random links by sampling the normal distribution $\mathcal{N}(0, 1)$. We then re-evaluate the memory and task

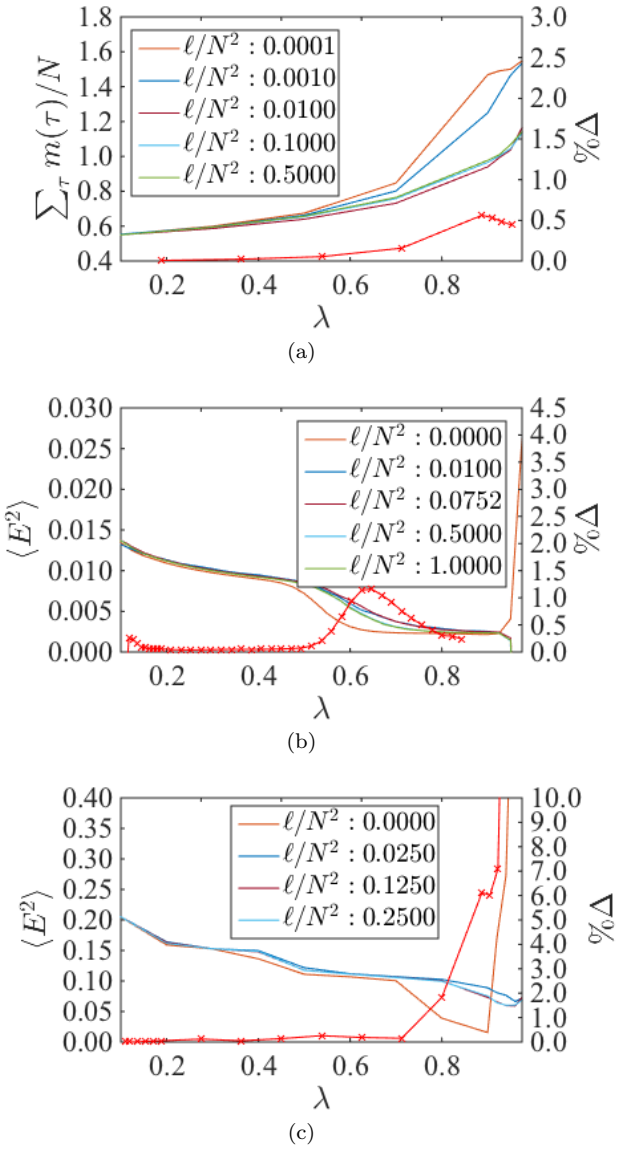

FIG. 4. Total memory capacity (a), NARMA10 error (b), and Mackey-Glass prediction error (c) as a function of λ and increasing network irregularity. The best results in all cases are for completely regular networks.

solving performance keeping the weight matrix fixed. We evaluate system performance on a memory task with exponentially correlated inputs, the nonlinear autoregressive NARMA10 task, and the Mackey-Glass chaotic time series prediction. We systematically explore the effects of λ, N, and ℓ on the performance of the system.

FIG 4(a) shows the resulting total memory capacity normalized by N as a function of increasing randomness $\frac{\ell}{N^2}$ for different spectral radii λ. The expected theoretical total memory capacity for an uncorrelated signal is $\sum_\tau m(\tau)/N = 1$. Here the system exploits the structure of the input signal to store longer input sequences, i.e., $\sum_\tau m(\tau)/N > 1$. This effect has been studied previously under annealed approximation and in a compressive sensing setup [12]. However, here we see that even without the sparse input assumption and L_1 optimization in the output (a computationally expensive optimization used in compressive sensing) the network can achieve capacity greater than its degrees of freedom N. FIG 4(b) and (c) show the error in the NARMA10 and the Mackey-Glass prediction tasks. Here, best performance is achieved for a regular architecture. A slight randomness significantly increases error at first, but additional irregularity will decrease it. This can be observed for the NARMA10 task at $\lambda = 0.6$ and for the Mackey-Glass prediction task at $\lambda = 0.9$.

Discussion— Although memory capacity of RNNs has been studied before, it is learning and generalization ability in a task solving setup is not discussed. Our derivation allows us to relate the memory capacity to task solving performance for arbitrary RNNs and reason about their generalization. In empirical experiments with systems presented here, the training and testing are done with finite input sequences that are sampled independently for each experiment, so the statistics of the training and testing inputs vary according to a Gaussian distribution around their true values and one expects these estimates to approach their true values with increasing sample size. Hence, the mean-squared-error $\langle E^2 \rangle$, which is linear in the input and output statistics, is also distributed as a Gaussian for repeated experiments. By the law of large numbers, the difference between testing and training mean-squared-error tends to zero in the limit. This explains the ability of the system to generalize its computation from training to test samples.

Conclusion— The computational power of reservoir computing networks has been attributed to their memory capacity. While their memory properties have been studied under annealed approximation, thus far no direct mathematical connection to their signal processing performance had been made. We developed a mathematical framework to exactly calculate the memory capacity of RC systems and extended the framework to study their expected and worst-case errors on a given task in a supervised learning setup. Our result confirms previous studies that the upper bound for memory capacity for uncorrelated inputs is N. We further show that the memory capacity monotonically increases with correlation in the input. Intuitively, the output exploits the redundant structure of the inputs to retrieve longer sequences. Moreover, we generalize our derivation to task solving performance. Our derivation help us reason about the memory and performance of arbitrary systems directly in terms of their structure. We showed that networks with regular structure have a higher memory capacity but are very sensitive to slight changes in the structure, while irregular networks are robust to variation in their structure.

Acknowledgments. This work was partly supported by NSF grants #1028238 and #1028120, #1518861, and #1525553.

[1] S. Ganguli, D. Huh, H. Sompolinsky, Proc. Natl. Acad. Sci. USA **105**, 18970–18975 (2008)
[2] P. Dominey, M. Arbib, and J. P. Joseph, J. Cogn. Neurosci. **7**, 311–336 (1995)
[3] W. Maass, T. Natschläger, and H. Markram, Neural Comput. **14**, 2531–60, (2002)
[4] H. Jaeger and H. Haas, Science **304**, 148102 (2004)
[5] Lukoševičius, H. Jaeger, and B. Schrauwen, Künstliche Intelligenz **26**, 365–371 (2012)
[6] D. Hansel and C. van Vreeswijk, J. Neurosci. **32**, 4049-4064 (2012)
[7] J. P. Crutchfield, W. L. Ditto, and S. Sinha, Chaos **20**, 037101 (2010)
[8] O. L. White, D. D. Lee, and H. Sompolinsky, Phys. Rev. Lett. **92**, 148102 (2004)
[9] T. Toyoizumi, Neural Comput. **24**, 2678–99, (2012)
[10] L. Büsing, B. Schrauwen, and R. Legenstein, Neural Comput. **22**, 1272–1311 (2010)
[11] A. Rodan and P. Tiňo, Neural Networks, IEEE Transactions on **22**, 131-144 (2011)
[12] S. Ganguli and H. Sompolinsky, Advances in Neural Information Processing Systems, **23**, 667–675, (2010)
[13] A. Goudarzi, C. Teuscher, N. Gulbahce, and T. Rohlf, Phys. Rev. Lett. **108**, 128702 (2012)
[14] D. Snyder, A. Goudarzi, and C. Teuscher, Phys. Rev. E **87**, 042808 (2013)
[15] H. O. Sillin, R. Aguilera, H. Shieh, A. V. Avizienis, M Aono, A. Z. Stieg, and J. K. Gimzewski, Nanotechnology, **24**, 384004, (2013)
[16] A. Goudarzi and D. Stefanovic, Procedia Computer Science **41**, 176–181, (2014)
[17] N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, Phys. Rev. E **91**, 020801, (2015)
[18] K. Nakajima, T. Li, H. Hauser, and R. Pfiefer, J. R. Soc. Interface **11**, 20140437, (2014)
[19] J. Bürger, A. Goudarzi, D. Stefanovic, and C. Teuscher, AIMS Materials Science **2**, 530–545, (2015)
[20] Y. Katayama, T. Yamane, D. Nakano, R. Nakane, and G. Tanaka, Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH '15), IEEE, p. 23-24, (2015)
[21] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, Nat. Commun. **5**, 3541, (2014)
[22] M. C. Mackey and L. Glass, Science **197**, 287–289, (1977)

Appendix A: Computing the optimal readout weights using autocorrelation

The recurrent neural network (RNN) model that we study in this paper is an echo state network (ESN) with linear activation function. This system consist of an input driven recurrent network of size N, and a linear readout layer trained to calculate a desired function of the input. Let $u(t)$ and $\boldsymbol{\omega}$ indicate a one-dimensional input at time t and an input weight vector respectively. Let $\boldsymbol{\Omega}$ be a $N \times N$ recurrent weight matrix, $\mathbf{x}(t)$ be an N-dimensional network state at time t, and $\boldsymbol{\Psi}$ be the readout weight vector. The dynamics of the network and output is described by:

$$\mathbf{x}(t+1) = \boldsymbol{\Omega}\mathbf{x}(t) + \boldsymbol{\omega}u(t) \tag{A1}$$
$$y(t+1) = \boldsymbol{\Psi}\mathbf{x}(t+1), \tag{A2}$$

where the readout weights are given by [8]:

$$\boldsymbol{\Psi} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{Y}^\top. \tag{A3}$$

The value of the optimal readout weights depend on the covariance and cross-covariance components $(\mathbf{X}\mathbf{X}^\top)$ and $\mathbf{X}\mathbf{Y}^\top$. Here we show that these can be computed exactly for any arbitrary system given by $\boldsymbol{\Omega}$ and $\boldsymbol{\omega}$ and autocorrelation of the input $R_{uu}(\tau)$ and cross-correlation of input and output $R_{u\widehat{y}}$.

We begin by noting that the explicit expression for the system state is given by:

$$\mathbf{x}(t+1) = \sum_{k=0}^{t} \boldsymbol{\Omega}^{t-k}\boldsymbol{\omega}u_k. \tag{A4}$$

Calculating Ψ for a given problem requires the following input-output-dependent evaluations:

$$\begin{aligned}\mathbf{X}\mathbf{X}^\intercal &= \left\langle \sum_{i,j=0}^{\infty} \boldsymbol{\Omega}^i\boldsymbol{\omega}u_{t-i}u_{t-j}^\intercal\boldsymbol{\omega}^\intercal{\boldsymbol{\Omega}^\intercal}^j \right\rangle \\ &= \sum_{i,j=0}^{\infty} \boldsymbol{\Omega}^i\boldsymbol{\omega}R_{uu}(i-j)\boldsymbol{\omega}^\intercal(\boldsymbol{\Omega}^\intercal)^j, \text{ and} \end{aligned} \tag{A5}$$

$$\begin{aligned}\mathbf{X}\mathbf{Y}^\intercal &= \left\langle \sum_{i=0}^{\infty} \boldsymbol{\Omega}^i\boldsymbol{\omega}u_{t-i}\widehat{y}_t^\intercal \right\rangle \\ &= \sum_{i=0}^{\infty} \boldsymbol{\Omega}^i\boldsymbol{\omega}R_{u\widehat{y}}(i). \end{aligned} \tag{A6}$$

Appendix B: Computing the total memory using autocorrelation

Here we compute the memory function and the total memory of the recurrent neural network described in Appendix A for exponentially correlated input where $R_{uu}(\tau) = e^{-\alpha\tau}$. The total memory of the system is given by the following summation over the memory function [8]:

$$\sum_{\tau} m(\tau) = \mathrm{Tr}((\mathbf{X}\mathbf{X}^\intercal)^{-1}\sum_{\tau=0}^{\infty}(\mathbf{X}\mathbf{Y}^\intercal)_\tau(\mathbf{Y}\mathbf{X}^\intercal)_\tau). \tag{B1}$$

where $\mathbf{Y}$ is the input with lag τ, $u_{t-\tau}$.
Computing $\mathbf{X}\mathbf{X}^\intercal$ requires the evaluation of:

$$\mathbf{X}\mathbf{X}^\intercal = \sum_{i,j=0}^{\infty} \boldsymbol{\Omega}^i\boldsymbol{\omega}R_{uu}(i-j)\boldsymbol{\omega}^\intercal(\boldsymbol{\Omega}^\intercal)^j. \tag{B2}$$

This assumes an even correlation function, i.e., $R_{uu}(i-j) = R_{uu}(j-i)$. For numerical computation it is more convenient to perform the calculation as follows:

$$\begin{aligned}\mathbf{X}\mathbf{X}^\intercal &= \mathbf{X}\mathbf{X}^\intercal_{i\geq j} + \mathbf{X}\mathbf{X}^\intercal_{i\leq j} - \mathbf{X}\mathbf{X}^\intercal_{i=j} \\ &= \mathbf{X}\mathbf{X}^\intercal_{i\geq j} + \left(\mathbf{X}\mathbf{X}^\intercal_{i\geq j}\right)^\intercal - \mathbf{X}\mathbf{X}^\intercal_{i=j}, \end{aligned} \tag{B3}$$

where $\mathbf{X}\mathbf{X}^\intercal_{i\geq j}$ is a partial sum of $\mathbf{X}\mathbf{X}^\intercal$ satisfying $i \geq j$, $\mathbf{X}\mathbf{X}^\intercal_{i\leq j} = \left(\mathbf{X}\mathbf{X}^\intercal_{i\geq j}\right)^\intercal$ is a partial sum of $\mathbf{X}\mathbf{X}^\intercal$ satisfying $i \leq j$, and $\mathbf{X}\mathbf{X}^\intercal_{i=j}$ is a partial sum of $\mathbf{X}\mathbf{X}^\intercal$ satisfying $i = j$, which is double counted and must be subtracted. We can substitute $\tau = |i-j|$ and evaluate $\mathbf{X}\mathbf{X}^\intercal_{i\geq j}$ and $\mathbf{X}\mathbf{X}^\intercal_{i=j}$ as follows:

$$\mathbf{X}\mathbf{X}^\intercal_{i\geq j} = \sum_{i,\tau=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{uu}(\tau) \boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i+\tau}, \text{ and} \tag{B4}$$

$$\mathbf{X}\mathbf{X}^\intercal_{i=j} = \sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{uu}(0) \boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i}. \tag{B5}$$

$$\mathbf{X}\mathbf{X}^\intercal_{i\geq j} = \sum_{i,\tau=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{uu}(\tau) \boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i+\tau} \tag{B6}$$

$$= \sum_{i,\tau=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} e^{-\alpha\tau} \boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i+\tau} \tag{B7}$$

$$= \sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega}\boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i} \sum_{\tau=0}^{\infty} (e^{-\alpha}\mathbf{\Omega}^\intercal)^{\tau} \tag{B8}$$

$$= \mathbf{U}\mathbf{\Lambda} \circ (\mathbf{I}^\circ - \mathbf{d}\mathbf{d}^\intercal)^{-1^\circ} \mathbf{U}^\intercal (\mathbf{I} - e^{-\alpha}\mathbf{\Omega}^\intercal)^{-1}, \tag{B9}$$

$$\mathbf{X}\mathbf{X}^\intercal_{i=j} = \sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{uu}(0) \boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i} \tag{B10}$$

$$= \sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega}\boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i} \tag{B11}$$

$$= \mathbf{U}\mathbf{\Lambda} \circ (\mathbf{I}^\circ - \mathbf{d}\mathbf{d}^\intercal)^{-1^\circ} \mathbf{U}^\intercal. \tag{B12}$$

Here $\mathbf{I}^\circ$ is the identity of the Hadamard product denoted by $\circ$, and $^{-1^\circ}$ is a matrix inverse with respect to the Hadamard product. Here the trick is that $\bar{\boldsymbol{\omega}} = \mathbf{U}^{-1}\boldsymbol{\omega}$ takes the input to the basis of the connection matrix $\mathbf{\Omega}$ allowing the dynamics to be described by the powers of the eigenvalues of $\mathbf{\Omega}$, i.e., $\mathbf{D}$. Since $\mathbf{D}$ is symmetric we can use the matrix identity $\mathbf{D}\mathbf{\Lambda}\mathbf{D} = \mathbf{\Lambda} \circ \mathbf{d}\mathbf{d}^\intercal$, where $\mathbf{d}$ is the main diagonal of $\mathbf{D}$. Summing over the powers of $\mathbf{D}$ gives us $\sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega}\boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{i} = \mathbf{U}\mathbf{\Lambda} \circ (\mathbf{I}^\circ - \mathbf{d}\mathbf{d}^\intercal)^{-1^\circ} \mathbf{U}^\intercal$.

The covariance of the network states and the expected output is given by:

$$\mathbf{X}\mathbf{Y}^\intercal_\tau = \sum_i \mathbf{\Omega}^i \boldsymbol{\omega} R(|i-\tau|) = \sum_i \mathbf{\Omega}^i \boldsymbol{\omega} e^{-\alpha|i-\tau|}. \tag{B13}$$

For $\alpha \to \infty$, the signal becomes i.i.d. and the calculations simplify as follows [16]:

$$\mathbf{X}\mathbf{X}^\intercal = \langle u^2 \rangle \mathbf{U}\mathbf{\Lambda} \circ (\mathbf{I}^\circ - \mathbf{d}\mathbf{d}^\intercal)^{-1^\circ} \mathbf{U}^\intercal, \tag{B14}$$

$$\mathbf{X}\mathbf{Y}^\intercal = \mathbf{\Omega}^\tau \boldsymbol{\omega} \langle u^2 \rangle. \tag{B15}$$

The total memory capacity can be calculated by summing over $m(k)$:

$$\sum_\tau m(\tau) = \mathrm{Tr}((\mathbf{X}\mathbf{X}^\intercal)^{-1} \sum_{\tau=0}^{\infty} (\mathbf{X}\mathbf{Y}^\intercal)_\tau (\mathbf{Y}\mathbf{X}^\intercal)_\tau). \tag{B16}$$

Appendix C: Experimental Setup for Memory Task

For our experiment with memory capacity of network under exponentially correlated input we used the following setup. We generated $T = 2,000,000$ long sample inputs with autocorrelation function $R_{uu}(\tau) = e^{-\alpha\tau}$. To generate exponentially correlated input we draw T samples u_i from a uniform distribution over the interval $[0, 1]$. The samples are passed through a low-pass filter with a smoothing factor α. We normalize and center u_t so that $\langle u(t) \rangle_t = 0$ and $\langle u(t)^2 \rangle_t = 1$. The resulting normalized samples $u(t)$ have exponential autocorrelation with decay exponent α, i.e., $R_{uu}(\tau) = e^{-\alpha\tau}$. To validate our calculations, we use a network of $N = 20$ nodes in a ring topology and identical weights. The spectral radius $\lambda = 0.9$. The input weights $\boldsymbol{\omega}$ are created by sampling the binomial distribution and multiplying with 0.1. The scale of the input weights does not affect the memory and the performance in linear systems and therefore we adopt this convention for generating $\boldsymbol{\omega}$ throughout the paper. We also assumed $\alpha = 0.05$, the number of samples $T = 30,000$, washout period of $5,000$ steps, and regularization factor $\gamma^2 = 10^{-9}$.

Appendix D: Experimental Setup for Topological Study

A long standing question in recurrent neural network is how its structure effect its memory and task solving performance. Our derivation lets us compute optimal readout layer for arbitrary network. Here we describe the calculations we performed to examine the effect of structure of the network on its memory and task solving performance. To this end, we use networks of size $N = 100$, $\alpha = 0.01$, and $\gamma = 10^{-13}$ and we systematically study the randomness and spectral radius. We start from a uniform weight ring topology and incrementally add randomness from $\ell = 0$ to $\ell = \frac{N^2}{2}$. The results for each value of ℓ and λ are averaged over 50 instances. This averaging is necessary even for $\ell = 0$ because the input weights are randomly generated and although their scaling does not affect the result their exact values do [1].

Appendix E: Computing the optimal readout weights using power spectrum

The calculations in Appendix A for optimal layer of a recurrent network may be described in a more generally in terms of power spectrum of the input signal. Here we assume the setup in Appendix A and derive an expressions for optimal readout layer using its the power spectrum of the input and output.

We start by the standard calculation of $\mathbf{X}\mathbf{X}^\intercal$ and $\mathbf{X}\mathbf{Y}^\intercal$:

$$\begin{aligned}\mathbf{X}\mathbf{X}^\intercal &= \left\langle \sum_{i,j=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} u_{t-i} u_{t-j}^\intercal \boldsymbol{\omega}^\intercal {\mathbf{\Omega}^\intercal}^j \right\rangle \\ &= \sum_{i,j=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{uu}(i-j) \boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^j, \text{ and} \qquad \text{(E1)} \\ \mathbf{X}\mathbf{Y}^\intercal &= \left\langle \sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} u_{t-i} \widehat{y}_t^\intercal \right\rangle \\ &= \sum_{i=0}^{\infty} \mathbf{\Omega}^i \boldsymbol{\omega} R_{u\widehat{y}}(i). \qquad \text{(E2)}\end{aligned}$$

We replace

$$R_{uu}(t) = \frac{1}{2T}\int_{-T}^{T} S_{uu}(f) e^{ift} df \qquad \text{(E3)}$$

and

$$R_{u\widehat{y}}(t) = \frac{1}{2T}\int_{-T}^{T} S_{u\widehat{y}}(f) e^{ift} df \qquad \text{(E4)}$$

which gives

$$\begin{aligned}\mathbf{X}\mathbf{X}^\intercal &= \frac{1}{2T}\int_{-T}^{T} \sum_{t,t'=0}^{\infty} \mathbf{\Omega}^t \boldsymbol{\omega} S_{uu}(f) e^{if(t-t')} \boldsymbol{\omega}^\intercal (\mathbf{\Omega}^\intercal)^{t'} \qquad \text{(E5)} \\ &= \frac{1}{2T}\int_{-T}^{T} \left(\sum_{t=0}^{\infty} (e^{if}\mathbf{\Omega})^t \right) \boldsymbol{\omega} S_{uu}(f) \boldsymbol{\omega}^\intercal \left(\sum_{t'=0}^{\infty} (e^{-if}\mathbf{\Omega}^\intercal)^{t'} \right) df \qquad \text{(E6)} \\ &= \frac{1}{2T}\int_{-T}^{T} \left(I - e^{if}\mathbf{\Omega}\right)^{-1} \boldsymbol{\omega} S_{uu}(f) \boldsymbol{\omega}^\intercal \left(I - e^{-if}\mathbf{\Omega}^\intercal\right)^{-1} df \qquad \text{(E7)}\end{aligned}$$

and

$$\begin{aligned}(\mathbf{X}\mathbf{Y}^\intercal)_\tau &= \frac{1}{2T}\int_{-T}^{T} \sum_{t=0}^{\infty} \mathbf{\Omega}^t \boldsymbol{\omega} S_{u\widehat{y}}(f) e^{if(t+\tau)} df \qquad \text{(E8)} \\ &= \frac{1}{2T}\int_{-T}^{T} \left(\sum_{t=0}^{\infty} (e^{if}\mathbf{\Omega})^t \right) \boldsymbol{\omega} S_{u\widehat{y}}(\omega) e^{if\tau} df \qquad \text{(E9)} \\ &= \frac{1}{2T}\int_{-T}^{T} \left(I - e^{if}\mathbf{\Omega}\right)^{-1} \boldsymbol{\omega} S_{u\widehat{y}}(f) e^{if\tau} df. \qquad \text{(E10)}\end{aligned}$$

Appendix F: Memory capacity expressed in terms of power spectrum

Here we use the derivation in Appendix E and compute the memory function and the total memory of the system.

Let $\mathbf{\Omega} = \boldsymbol{U}\text{diag}(\mathbf{d})\boldsymbol{U}^{-1}$ and $\bar{\boldsymbol{\omega}} = \mathbf{U}^{-1}\boldsymbol{\omega}$ so that

$$\left(I - e^{if}\mathbf{\Omega}\right)^{-1} = \boldsymbol{U}\text{diag}\left(\frac{1}{1-e^{if}\mathbf{d}}\right)\boldsymbol{U}^{-1}. \qquad \text{(F1)}$$

We find that

$$m(\tau) = \frac{1}{2T}\bar{\boldsymbol{\omega}}^\top \left(A \circ C^{-1}\right).\bar{\boldsymbol{\omega}} \qquad \text{(F2)}$$

The matrix C is given by

$$C = \int_{-T}^{T} \left[\frac{\bar{\boldsymbol{\omega}}}{1-e^{if}\mathbf{d}}\right] \otimes \left[\frac{\bar{\boldsymbol{\omega}}}{1-e^{-if}\mathbf{d}}\right] S_{uu}(f) df, \qquad \text{(F3)}$$

and the matrix A is given by:

$$A = \left[\int_{-T}^{T} \frac{S_{u\widehat{y}}(f)e^{if\tau}}{1-e^{if}\mathbf{d}} df\right] \otimes \left[\int_{-T}^{T} \frac{S_{u\widehat{y}}(f)e^{if\tau}}{1-e^{if}\mathbf{d}} df\right]. \qquad \text{(F4)}$$

The total memory is then given by:

$$\frac{1}{N}\sum_{\tau=0}^{\infty} m(\tau) = \frac{1}{2TN}\bar{\boldsymbol{\omega}}^\top \left(B \circ C^{-1}\right)\bar{\boldsymbol{\omega}} \qquad \text{(F5)}$$

where

$$B = \int_{-T}^{T} \int_{-T}^{T} \frac{df df'}{1 - e^{i(f+f')}} [\frac{S_{u\widehat{y}}(f)}{1 - e^{if}\mathbf{d}}] \otimes [\frac{S_{u\widehat{y}}(f')}{1 - e^{if'}\mathbf{d}}]. \tag{F6}$$