Difference between memory and prediction in linear recurrent networks

Sarah Marzen[*]
Department of Physics, Physics of Living Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

(Received 29 June 2017; revised manuscript received 14 August 2017; published 11 September 2017;
corrected 22 October 2018)

Recurrent networks are trained to memorize their input better, often in the hopes that such training will increase the ability of the network to predict. We show that networks designed to memorize input can be arbitrarily bad at prediction. We also find, for several types of inputs, that one-node networks optimized for prediction are nearly at upper bounds on predictive capacity given by Wiener filters and are roughly equivalent in performance to randomly generated five-node networks. Our results suggest that maximizing memory capacity leads to very different networks than maximizing predictive capacity and that optimizing recurrent weights can decrease reservoir size by half an order of magnitude.

DOI: 10.1103/PhysRevE.96.032308

Often, we remember for the sake of prediction. Such is the case, it seems, in the field of echo state networks (ESNs) [1,2]. ESNs are large input-dependent recurrent networks in which a "readout layer" is trained to match a desired output signal from the present network state. Sometimes, the desired output signal is the past or future of the input to the network.

If the recurrent networks are large enough, they should have enough information about the past of the input signal to reproduce a past input or predict a future input well, and only the readout layer need be trained. Still, the weights and structure of the recurrent network can greatly affect the predictive capabilities of the recurrent network, and so, many researchers are now interested in optimizing the network itself to maximize task performance [3].

Much of the theory surrounding echo state networks centers on memorizing white noise, an input for which memory is essentially useless for prediction [4]. This leads to a rather practical question: How much of the theory surrounding optimal reservoirs, based on maximizing memory capacity (MC) [5–9], is misleading if the ultimate goal is to maximize predictive power?

We study the difference between optimizing for memory and optimizing for prediction in linear recurrent networks subject to scalar temporally correlated input generated by countable hidden Markov models. Reference [10] gave closed-form expressions for the memory function of continuous-time linear recurrent networks in terms of the autocorrelation function of the input and closely studied the case of an exponential autocorrelation function. Reference [11] gave similar expressions for discrete-time linear recurrent networks. Reference [12] gave closed-form expressions for the Fisher memory curve of discrete-time linear recurrent networks, which measure how many changes in the input signal perturb the network state; for linear recurrent networks, this curve is independent of the particular input signal.

We differ from these previous efforts mostly in that we study both memory capacity and newly defined "predictive capacity (PC)". We derive an upper bound for predictive capacity via Wiener filters in terms of the autocorrelation function of the input. Two surprising findings result. First, predictive capacity is not typically maximized at the "edge of criticality", unlike memory capacity [5,7,9]. Instead, maximizing memory capacity can lead to minimization of predictive capacity. Second, optimized one-node networks tend to achieve more than 99% of the possible predictive capacity, whereas (unoptimized) linear random networks need at least five nodes to reliably achieve similar memory and predictive capacities, and ten-node nonlinear random networks cannot match the optimized one-node linear network. The latter result suggests that optimizing reservoir weights can lead to at least half an order-of-magnitude reduction in the size of the reservoir with no loss in task performance.

I. MODEL

Let $s(n)$ denote the input signal at time n, and let $x(n)$ denote the network state at time n. The network state updates as

$$x(n+1) = Wx(n) + s(n)v, \tag{1}$$

where W,v are two reservoir properties that we wish to optimize. We restrict our attention to the case that W is diagonalizable,

$$W = P\,\mathrm{diag}(\vec{d})P^{-1}, \tag{2}$$

where P is the matrix of the eigenvectors of W and $\vec{d}$'s are the corresponding eigenvalues. For reasons that will become clear later, we define a vector,

$$\omega = P^{-1}v. \tag{3}$$

We further assume that the input $s(t)$ has been generated by a countable hidden Markov model so that its autocorrelation function can be expressed as

$$R_{ss}(t) = \sum_{\lambda\in\Lambda} A(\lambda)\lambda^{|t|}, \tag{4}$$

where Λ is a set of numbers with a magnitude less than 1. See Ref. [13] or Appendix A. To avoid normalization factors and to ensure that this autocorrelation function represents that from an HMM, we assert that

$$R_{ss}(0) = \sum_{\lambda\in\Lambda} A(\lambda) = 1. \tag{5}$$

[*] semarzen@mit.edu

The power spectral density of this input process with

$$R_{ss}(t)=\frac{1}{2\pi}\int_{-\pi}^{\pi}S(f)e^{ift}df \tag{6}$$

is

$$S(f)=\sum_{k=\infty}^{\infty}R_{ss}(k)e^{-ifk} \tag{7}$$

$$=\sum_{\lambda\in\Lambda}A(\lambda)\sum_{k=-\infty}^{\infty}\lambda^{|k|}e^{-ifk} \tag{8}$$

$$=\sum_{\lambda\in\Lambda}A(\lambda)\frac{1-\lambda^2}{(1-\lambda e^{-if})(1-\lambda e^{if})}, \tag{9}$$

by the Wiener-Khinchin theorem.

II. RESULTS

The memory function is classically defined by [5]

$$m(k):=p_k^{\top}C^{-1}p_k, \tag{10}$$

where

$$p_k=\langle s(n-k)x(n)\rangle_n, \tag{11}$$

and

$$C=\langle x(n)x(n)^{\top}\rangle_n. \tag{12}$$

Due to Eq. (5), we need not divide $p_k^{\top}C^{-1}p_k$ by the variance of the input. This memory function is also the squared correlation coefficient between our optimal linear estimate of input $s(n-k)$ from network state $x(n)$ and the true input $s(n-k)$.

Memory capacity usually is defined as $\sum_{k=0}m(k)$, but since Eq. (1) updates $x(n)$ with $s(n-1)$ instead of $s(n)$, we have

$$\mathrm{MC}=\sum_{k=1}^{\infty}m(k), \tag{13}$$

and we define the *predictive capacity* as

$$\mathrm{PC}:=\sum_{k=0}^{\infty}m(-k). \tag{14}$$

Intuitively, MC is higher when the present network state is better able to remember inputs, whereas PC is higher when the present network state is better able to forecast inputs based on what it remembers of past inputs.

We have made an effort here to find the most useful expressions for MC and PC so that one might consider using the expressions here to calculate MC, PC instead of simulating the input and recurrent network. As shown in Appendix B,

$$\mathrm{PC}=2\pi\omega^{\top}(D_{\mathrm{PC}}\odot B^{-1})\omega, \tag{15}$$

where

$$B:=\int_{-\pi}^{\pi}S(f)\left(\frac{\omega}{e^{-if}-\vec{d}}\right)\left(\frac{\omega}{e^{if}-\vec{d}}\right)^{\top}df, \tag{16}$$

which is related to $2\pi C$ by a similarity transform, and where

$$D_{\mathrm{PC}}:=\sum_{\lambda,\lambda'\in\Lambda}\frac{A(\lambda)A(\lambda')}{1-\lambda\lambda'}\left(\frac{1}{\lambda^{-1}-\vec{d}}\right)\left(\frac{1}{(\lambda')^{-1}-\vec{d}}\right)^{\top}. \tag{17}$$

The expression for memory capacity is more involved

$$\mathrm{MC}=2\pi\omega^{\top}\big(D_{\mathrm{MC}}\odot B^{-1}\big)\omega, \tag{18}$$

where the matrix D_{MC} has entries,

$$\begin{aligned}(D_{\mathrm{MC}})_{ij}=\sum_{\lambda,\lambda'\in\Lambda}A(\lambda)A(\lambda')\bigg(&\frac{1+d_id_j\lambda(\lambda')^3+d_id_j\lambda'\lambda^3+d_id_j(\lambda\lambda')^2-d_id_j\lambda\lambda'-d_id_j(\lambda')^2-d_id_j\lambda^2-d_i^2d_j^2}{(1-\lambda\lambda')(1-d_i\lambda)(1-d_i\lambda')(1-d_j\lambda)(1-d_j\lambda')(1-d_id_j)}\\&-\frac{d_i(1-d_id_j)\lambda^2\lambda'+d_j(1-d_id_j)\lambda(\lambda')^2}{(1-\lambda\lambda')(1-d_i\lambda)(1-d_i\lambda')(1-d_j\lambda)(1-d_j\lambda')(1-d_id_j)}\bigg),\end{aligned} \tag{19}$$

as shown in Appendix B. Together, these expressions explain why simple linear ESNs [14] can perform just as well as nonsimple linear ESNs on the maximization of MC; from Eq. (18), the memory capacity of a linear ESN is the same as the memory capacity of a simple linear ESN with $v=\omega$ and $W=\mathrm{diag}(\vec{d})$.

It is unsurprising but not often mentioned that the reservoirs which maximize memory capacity are different than the reservoirs that maximize predictive capacity. To illustrate how different the two reservoirs might be, we consider the capacity of a one-node network subject to two types of input.

The first type of input considered has autocorrelation $R_{ss}(t)=e^{-\alpha|t|}$. Some algebra reveals that $\mathrm{MC}=\frac{e^{4\alpha}-2e^{\alpha}W+2e^{3\alpha}W-W^2}{(e^{2\alpha}-1)(e^{2\alpha}-W^2)}$ and $\mathrm{PC}=\frac{e^{2\alpha}(1-W^2)}{(e^{2\alpha}-1)(e^{2\alpha}-W^2)}$. Inspection of these formulas or inspection of the plots of these formulas in Fig. 1 (blue lines) for $\alpha=0.1$ shows that MC is maximized at the edge of criticality $W\to1$ at which point $x(n)$ is an average of observed $s(n)$—i.e., $x(n)=\langle s(k)\rangle_{k\leqslant n}$. Interestingly, at that point, PC is minimized, i.e., $\mathrm{PC}=0$. Instead, for this particular input, PC is maximized at $W=0$ at which point $x(n)=s(n-1)$—i.e., $x(n)$ is the last observed input symbol.

Both memory and predictive capacities can increase without bound by increasing the length of temporal correlations in this input: $\lim_{W\to1}\mathrm{MC}=\coth(e^{\alpha/2})$ and $\lim_{W\to0}\mathrm{PC}=\frac{1}{2}(\coth\alpha-1)$. These results mirror what was found in Ref. [10] for continuous-time networks: $\lim_{W\to1}\mathrm{MC}=\frac{2}{\alpha}$ plus corrections of $O(\alpha)$ and $\lim_{W\to0}\mathrm{PC}=\frac{1}{2\alpha}$ plus corrections of $O(1)$.

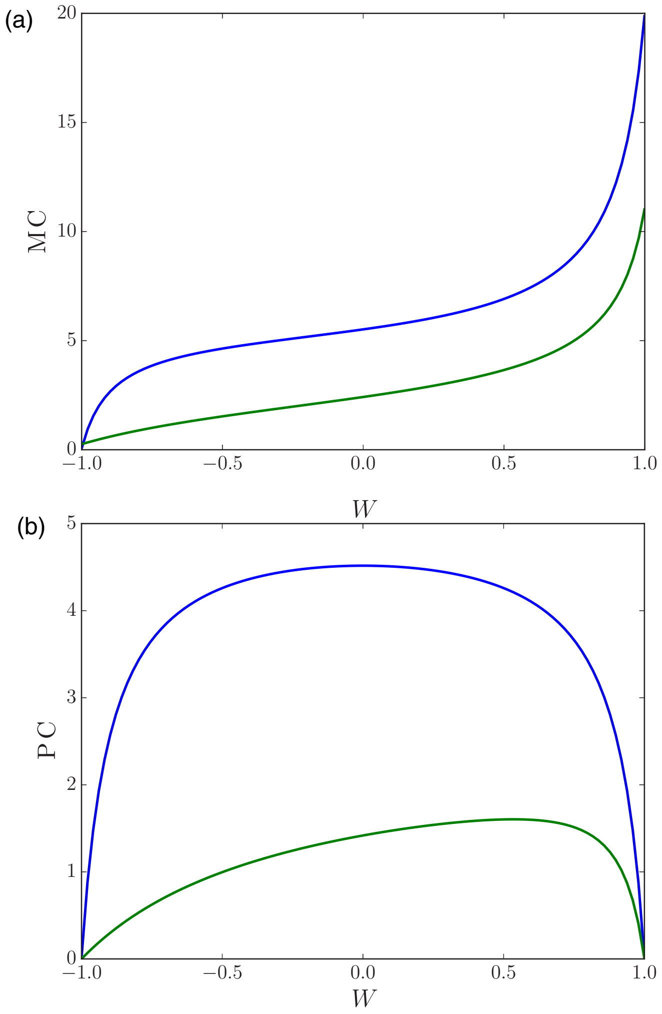

FIG. 1. [Top (a)] MC and [bottom (b)] PC as a function of W for $R_{ss}(t)=e^{-0.1|t|}$ (blue lines) and $R_{ss}(t)=\frac{1}{2}e^{-0.1|t|}+\frac{1}{2}e^{-|t|}$ (green lines), computed using Eqs. (15) and (18) in the main text. Whereas PC is maximized for some intermediate W that depends on the input signal, MC is maximized in the limit $W\to 1$. When $|W|\geqslant 1$, the network no longer satisfies the echo state property, and so we only calculate PC, MC for $|W|<1$.

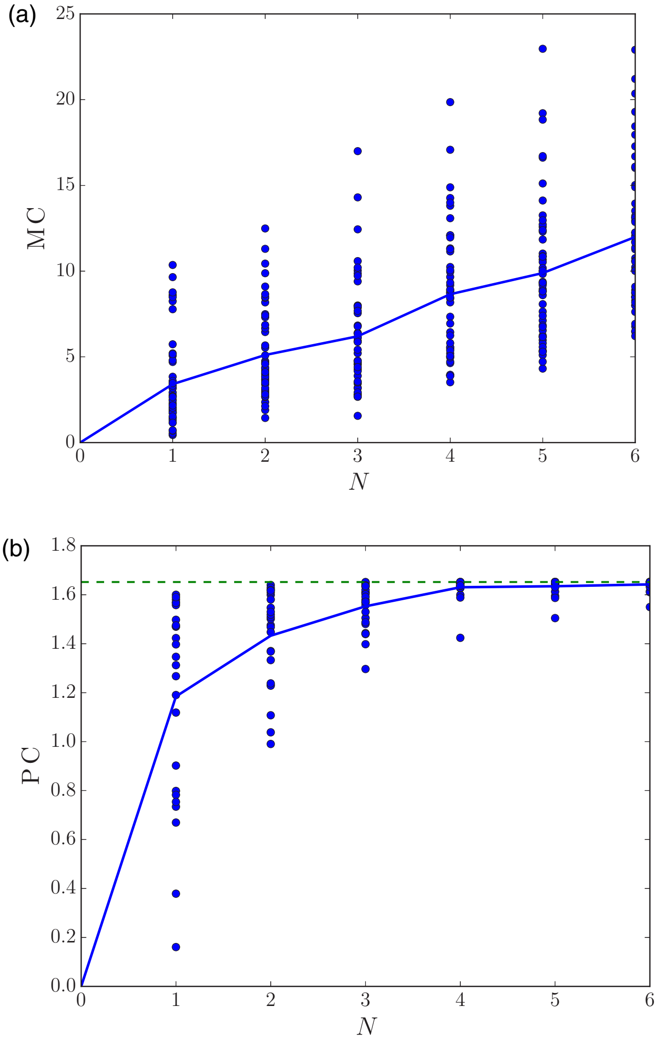

FIG. 2. [Top (a)] MC and [bottom (b)] PC as a function of N for $R_{ss}(t)=\frac{1}{2}e^{-0.1|t|}+\frac{1}{2}e^{-|t|}$ and $\omega,\vec{d}$ drawn randomly: $\omega_i\sim\mathcal{U}[0,1]$, $d_i\sim\mathcal{U}[-1,1]$. The signal-limited maximal PC is shown in green, whereas MC is network limited. Both MC, PC were computed using Eqs. (15) and (18) in the main text.

It is a little strange to say that $W=0$ can maximize the predictive capacity of a reservoir as $W=0$ implies that there essentially is no reservoir. But such $\arg\max_W$ PC is unusual. Consider input with $R_{ss}(t)=\frac{1}{2}e^{-0.1|t|}+\frac{1}{2}e^{-|t|}$ to a one-node network. Memory capacity still is maximized as $W\to 1$, but predictive capacity is now maximized at $W\approx 0.8$. See Fig. 1 (green lines). Interestingly, we still minimize any error in memorization of previous inputs by storing their average value.

The scaling of capacity with the network size is also very different for memory and prediction. MC for linear recurrent networks famously scales linearly with the number of nodes for linear recurrent networks [5]. Unlike memory, PC is bounded by the signal itself. The Wiener filter $k_\tau(n)$ minimizes the mean-squared error $\langle[s(n+\tau)-\hat{s}(n+\tau)]^2\rangle_n$ of future input $s(n+\tau)$ and a forecast of future input from past input $\hat{s}(n+\tau):=\sum_{m=0}^{\infty}k_\tau(m)s(n-m)$. Recall that minimizing the mean-squared error is equivalent to maximizing the correlation coefficient between a future input and an optimal linear estimate of this future input. Hence, we can place an upper bound on PC in terms of Wiener filters, which, after some straightforward simplification shown in Appendix C, takes the form

$$\mathrm{PC}\leqslant\sum_{\tau=0}^{\infty}\vec{r}_\tau^{\top}R^{-1}\vec{r}_\tau, \tag{20}$$

where $(\vec{r}_\tau)_i=R_{ss}(\tau+i)$ and $R_{ij}=R_{ss}(i-j)$.

As PC is at most finite, the scaling of PC with the number of nodes of the network N must eventually be $o(1)$. See Fig. 2 (bottom). For instance, for $R_{ss}(t)=\frac{1}{2}e^{-0.1|t|}+\frac{1}{2}e^{-|t|}$, Eq. (20) gives PC $\leqslant 1.652$, which nearly is attained by the optimal one-node network, for which $\max_W$ PC is ≈ 1.65. And this is not a special property for a cherry-picked input signal; similar results hold for other different randomly chosen Λ,A_λ combinations not shown here.

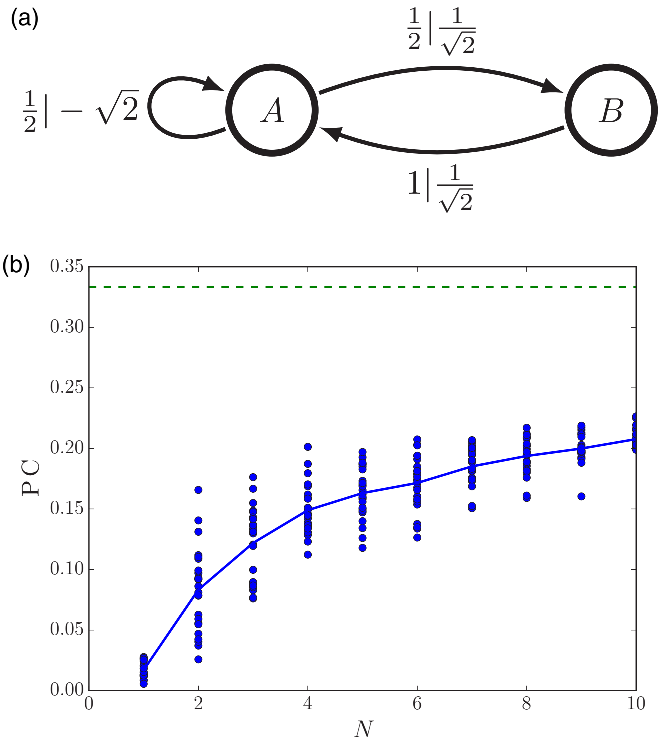

FIG. 3. At the top (a), the hidden Markov model generating input to the nonlinear recurrent network. Edges are labeled $p(x|g)|x$, where x is the emitted symbol and $p(x|g)$ is the probability of emitting that symbol when in hidden state g and the arrows indicate which hidden state one goes to after emitting a particular symbol from the previous hidden state. This hidden Markov model generates a zero-mean unit-variance even process, which has the autocorrelation function $R_{ss}(t) = \begin{cases} (-\frac{1}{2})^{|t|+1} & |t| \geqslant 1 \\ 1 & t = 0 \end{cases}$. At the bottom (b), the predictive capacity of random nonlinear recurrent networks whose evolution is given by Eq. (21) with $f(x) = \tanh(x)$ and entries of W and v drawn randomly: $W_{ij}, v_i \sim \mathcal{U}[0,1]$, where W then is scaled so that its largest magnitude eigenvalue has an absolute value of $1/1.1$. Twenty-five random networks are surveyed at each N, and the blue line tracks the mean. The green line shows both the predictive capacity of the optimized one-node linear network and the upper bound from Eq. (20).

The surprisingly good performance of optimized one-node networks leads us then to ask how big random (unoptimized) networks need be in order to achieve similar results. Unoptimized random networks need ≈5 nodes to reliably achieve similar results for the optimized one-node network for both memory and predictive capacities. See Fig. 2 in comparison to Fig. 1.

Finally, we ask whether any of the lessons learned here for linear recurrent networks extend to nonlinear recurrent networks in which

$$x(n+1) = f[Wx(n) + s(n)v] \tag{21}$$

for some nonlinear function f. From Eq. (B1), we see that linear recurrent networks forecast input via a linear combination of past input; therefore, as noted previously, their performance is bounded from above by the performance of Wiener filters. The performance of nonlinear recurrent networks is bounded above by a quantity that depends on the nonlinearity, which in principle might surpass the bound on predictive capacity given by Eq. (20).

However, optimizing the weights of nonlinear recurrent networks is far more difficult than for linear recurrent networks. This is illustrated by Fig. 3 (bottom), which shows the estimated PC of random nonlinear networks. We estimate the predictive capacity from simulations via $\sum_{k=0}^{M} \hat{p}_k^\top \hat{C}^{-1} \hat{p}_k$, where $\hat{p}_k^\top$ is the sample covariance of $s(n+\tau)$, $x(n)$,$\hat{C}$ is the sample variance of $x(n)$, and M is taken to be 100 as the correlation coefficient dies off relatively quickly. The reservoir properties W and v are chosen randomly in that both matrix elements W_{ij} and vector elements v_i are drawn randomly at uniform from the unit interval and the matrix W is rescaled so that the eigenvalue of maximum magnitude has a magnitude of $1/1.1$ and the nonlinearity is set to $f(x) = \tanh x$. The input to the network is generated by the hidden Markov model shown in Fig. 3 (top). For comparison, the green line shows the upper bound on predictive capacity for linear recurrent networks given by Eq. (20), which is achieved by one-node linear networks with $W = 0$. These numerical results are qualitatively similar to results attained when comparing the memory capacity of linear and nonlinear recurrent networks in that linear networks tend to outperform nonlinear networks [12,15].

III. DISCUSSION

The famous Wiener filter is a linear combination of the past input signal that minimizes the mean-squared error between the said linear combination and a future input. Linear recurrent networks are, in some sense, an attempt to approximate the Wiener filter under constraints on the kernel that come from the structure of the recurrent network. Here, the linear filter is not allowed access to all the past of the signal but is only allowed access to the echoes of the signal past provided by the present state of the nodes. The advantage of such an approximation is that one only need store the present network state as opposed to storing the entire past of the input signal. In other words, the present network state provides a nearly sufficient echo of the input signal's past for input prediction.

We have studied the resource savings that can come from optimizing the recurrent network and readout weights as opposed to just optimizing the readout weights. Surprisingly, we find that a network designed to maximize memory capacity has arbitrarily low predictive capacity; see Fig. 1. More encouragingly, we find that an optimized single-node linear recurrent network is essentially equivalent in terms of both memory and predictive capacities to a five-node random linear recurrent network and near maximal predictive capacity. Finally, numerical results suggest that nonlinear recurrent networks have more difficulty achieving high predictive capacity relative to the Wiener filter-placed upper bound on linear recurrent networks, even though these nonlinear networks might in principle surpass such an upper bound.

It is unclear whether or not the factor of 5 will generalize to nonlinear recurrent networks or for inputs generated by uncountable hidden Markov models, e.g., the output of chaotic dynamical systems. Perhaps more importantly, predictive capacity is not necessarily the quantity that we would most like to maximize [16]. Hopefully, the differences between memory and predictive capacities presented here will stimulate the search for more task-appropriate objective functions and for more reservoir optimization recipes.

ACKNOWLEDGMENTS

We owe substantial intellectual debts to A. Goudarzi, J. P. Crutchfield, S. Still, and A. Bell and thank I. Nemenman, N. Ay, C. Hillar, S. Dedeo, and W. Bialek for very useful conversations. S.M. was funded by a MIT Physics of Living Systems Fellowship.

APPENDIX A: AUTOCORRELATION FUNCTION OF HIDDEN MARKOV MODELS

This is a simple version of the argument in Ref. [13] that assumes diagonalizability of the transition matrix. Let $T^{(x)}$ be the labeled transition matrices of the hidden Markov model, let

$$T = \sum_x T^{(x)} \tag{A1}$$

be the transition matrix, and let $\vec{p}_{\text{eq}} = \text{eig}_1(T)$ be the stationary distribution over the hidden states. Assuming zero-mean input, we have

$$\begin{aligned} R(t) &= \langle x(t-1)x(0)\rangle && \text{(A2)}\\ &= \sum_{x,x'} x x' \Pr(X_{t-1}=x,\ X_0=x') && \text{(A3)}\\ &= \sum_{x,x'} x x' \vec{1}^\top T^{(x)} T^{t-1} T^{(x')} \vec{p}_{\text{eq}} && \text{(A4)}\\ &= \sum_{x,x'} \vec{1}^\top (xT^{(x)}) T^{t-1} (x'T^{(x')}) \vec{p}_{\text{eq}} && \text{(A5)}\\ &= \vec{1}^\top \left(\sum_x xT^{(x)}\right) T^{t-1} \left(\sum_x xT^{(x)}\right) \vec{p}_{\text{eq}}. && \text{(A6)} \end{aligned}$$

If T is diagonalizable (and it typically is), then $T = P\,\text{diag}(\vec{\lambda})P^{-1}$ leads to

$$R(t) = \vec{1}^\top \left(\sum_x xT^{(x)}\right) P\,\text{diag}(\vec{\lambda}^{t-1}) P^{-1} \left(\sum_x xT^{(x)}\right) \vec{p}_{\text{eq}}, \tag{A7}$$

and so $R(t)$ is a linear combination of λ_i^t.

APPENDIX B: DERIVATION OF CLOSED-FORM EXPRESSIONS FOR PC, MC

From Eq. (1), we have

$$x(n) = \left(\sum_{k=1}^{\infty} W^{k-1} s(n-k)\right) v, \tag{B1}$$

assuming the echo state property. Thus,

$$\begin{aligned} p_k &= \langle s(n-k)x(n)\rangle_n && \text{(B2)}\\ &= \sum_{m=1}^{\infty} W^{m-1} R_{ss}(k-m) v, && \text{(B3)} \end{aligned}$$

and

$$\begin{aligned} C &= \langle x(n)x(n)^\top\rangle_n && \text{(B4)}\\ &= \sum_{m,m'=1}^{\infty} W^{m-1} vv^\top (W^\top)^{m'-1} R_{ss}(m-m'). && \text{(B5)} \end{aligned}$$

Substituting Eq. (6) into the above equation gives

$$\begin{aligned} C &= \frac{1}{2\pi} \sum_{m,m'=1}^{\infty} W^{m-1} vv^\top (W^\top)^{m'-1} \int_{-\pi}^{\pi} S(f) e^{if(m-m')} df && \text{(B6)}\\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(f) \left(\sum_{m=1}^{\infty} e^{ifm} W^{m-1}\right) vv^\top \left(\sum_{m'=1}^{\infty} (W^\top)^{m'-1} e^{-ifm'}\right) df && \text{(B7)}\\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(f) \left(\sum_{m=0}^{\infty} e^{ifm} W^{m}\right) vv^\top \left(\sum_{m'=0}^{\infty} (W^\top)^{m'} e^{-ifm'}\right) df && \text{(B8)}\\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(f) (I - e^{if} W)^{-1} vv^\top (1 - e^{-if} W^\top)^{-1} df, && \text{(B9)} \end{aligned}$$

and using Eq. (2),

$$C = \frac{1}{2\pi} P \left[\int_{-\pi}^{\pi} S(f) \left(\frac{\omega}{1 - e^{if}\vec{d}}\right) \left(\frac{\omega}{1 - e^{-if}\vec{d}}\right)^\top df \right] P^{-1}. \tag{B10}$$

Returning to Eq. (B3) and using Eq. (4), we have

$$\begin{aligned} p_k &= \sum_{m=1}^{\infty} W^{m-1} \left(\sum_{\lambda\in\Lambda} A(\lambda)\lambda^{|k-m|}\right) v && \text{(B11)}\\ &= \sum_{\lambda\in\Lambda} A(\lambda) \sum_{m=1}^{\infty} W^{m-1} \lambda^{|k-m|} v && \text{(B12)} \end{aligned}$$

$$= \begin{cases} \sum_{\lambda\in\Lambda} A(\lambda) \sum_{m=1}^{\infty} W^{m-1}\lambda^{m-k} v, & k < 1, \\ \sum_{\lambda\in\Lambda} A(\lambda)\Big(\sum_{m=1}^{k} W^{m-1}\lambda^{k-m} + \sum_{m=k+1}^{\infty} W^{m-1}\lambda^{m-k}\Big) v, & k \geqslant 1 \end{cases} \tag{B13}$$

$$= \begin{cases} \sum_{\lambda\in\Lambda} A(\lambda)\lambda^{-k} W^{-1}\big(\sum_{m=1}^{\infty} W^{m}\lambda^{m}\big) v, & k < 1, \\ \sum_{\lambda\in\Lambda} A(\lambda)\Big(\lambda^{k} W^{-1}\sum_{m=1}^{k} W^{m}\lambda^{-m} + W^{-1}\lambda^{-k}\sum_{m=k+1}^{\infty} W^{m}\lambda^{m}\Big) v, & k \geqslant 1 \end{cases} \tag{B14}$$

$$= \begin{cases} \sum_{\lambda\in\Lambda} A(\lambda)\lambda^{-k}(\lambda^{-1} - W)^{-1} v, & k < 1, \\ \sum_{\lambda\in\Lambda} A(\lambda)[(W^{k} - \lambda^{k})(W - \lambda)^{-1} + W^{k}(\lambda^{-1} - W)^{-1}] v, & k \geqslant 1. \end{cases} \tag{B15}$$

Using Eq. (2),

$$p_k = P \begin{cases} \displaystyle\sum_{\lambda\in\Lambda} A(\lambda)\lambda^{-k}\left(\frac{\omega}{\lambda^{-1} - \vec{d}}\right), & k < 1, \\ \displaystyle\sum_{\lambda\in\Lambda} A(\lambda)\mathrm{diag}\left(\frac{\vec{d}^{k} - \lambda^{k}}{\vec{d} - \lambda} + \frac{\vec{d}^{k}}{\lambda^{-1} - \vec{d}}\right)\omega, & k \geqslant 1. \end{cases} \tag{B16}$$

Thus we have

$$\mathrm{PC} = \sum_{k=0}^{\infty} p_{-k}^{\top} C^{-1} p_{-k} \tag{B17}$$

$$= 2\pi \sum_{k=0}^{\infty} \left[\sum_{\lambda\in\Lambda} A(\lambda)\lambda^{k}\left(\frac{\omega}{\lambda^{-1} - \vec{d}}\right)\right]^{\top} B^{-1} \left[\sum_{\lambda\in\Lambda} A(\lambda)\lambda^{k}\left(\frac{\omega}{\lambda^{-1} - \vec{d}}\right)\right] \tag{B18}$$

$$= 2\pi \sum_{\lambda\in\Lambda} \frac{A(\lambda)A(\lambda')}{1 - \lambda\lambda'}\left(\frac{\omega}{\lambda^{-1} - \vec{d}}\right)^{\top} B^{-1}\left(\frac{\omega}{(\lambda')^{-1} - \vec{d}}\right) \tag{B19}$$

$$= 2\pi \sum_{i,j}\sum_{\lambda\in\Lambda} \frac{A(\lambda)A(\lambda')}{1 - \lambda\lambda'}\left(\frac{\omega_i}{\lambda^{-1} - d_i}\right)(B^{-1})_{ij}\left(\frac{\omega_j}{(\lambda')^{-1} - d_j}\right) \tag{B20}$$

$$= 2\pi \sum_{i,j} \omega_i \left(\sum_{\lambda\in\Lambda} \frac{A(\lambda)A(\lambda')}{1 - \lambda\lambda'} \frac{1}{\lambda^{-1} - d_i} \frac{1}{(\lambda')^{-1} - d_j}\right)\left(B^{-1}\right)_{ij}\omega_j, \tag{B21}$$

which gives the formula in the main text. Similar manipulations with the help of *Mathematica* give the more involved formula for MC.

APPENDIX C: DERIVATION OF THE UPPER BOUND FOR PC

Recall that

$$\mathrm{PC}_{\tau} = \frac{\langle s(t+\tau)\hat{s}(t+\tau)\rangle_t^2}{\langle \hat{s}(t)^2\rangle_t}, \tag{C1}$$

and

$$\mathrm{PC} = \sum_{\tau=0}^{\infty} \mathrm{PC}_{\tau}. \tag{C2}$$

As our problem setup naturally restricts us to causal linear filters, PC_{τ} is maximized with $\hat{s}(t+\tau) = \sum_{n=1}^{\infty} s(t-n)k_{\tau}(n)$ with $k_{\tau}(n)$ as a Wiener filter. In particular, suppose that $k_{\tau}(n)$ satisfies the Wiener-Hopf equation,

$$R_{ss}(\tau + t) = \sum_{m=1}^{\infty} R_{ss}(t-m)k_{\tau}(m). \tag{C3}$$

In matrix form, this reads

$$\begin{pmatrix} R_{ss}(\tau+1) \\ R_{ss}(\tau+2) \\ \vdots \end{pmatrix} = \begin{pmatrix} R_{ss}(0) & R_{ss}(-1) & R_{ss}(-2) & \dots \\ R_{ss}(1) & R_{ss}(0) & R_{ss}(1) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} k_{\tau}(1) \\ k_{\tau}(2) \\ \vdots \end{pmatrix}, \tag{C4}$$

and so

$$\begin{pmatrix} k_\tau(1) \\ k_\tau(2) \\ \vdots \end{pmatrix} = \begin{pmatrix} R_{ss}(0) & R_{ss}(-1) & R_{ss}(-2) & \dots \\ R_{ss}(1) & R_{ss}(0) & R_{ss}(1) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}^{-1} \begin{pmatrix} R_{ss}(\tau+1) \\ R_{ss}(\tau+2) \\ \vdots \end{pmatrix}. \tag{C5}$$

For ease of notation, we define R as

$$R := \begin{pmatrix} R_{ss}(0) & R_{ss}(-1) & R_{ss}(-2) & \dots \\ R_{ss}(1) & R_{ss}(0) & R_{ss}(1) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{C6}$$

and

$$\vec{r}_\tau := \begin{pmatrix} R_{ss}(\tau+1) \\ R_{ss}(\tau+2) \\ \vdots \end{pmatrix}, \tag{C7}$$

so in short, $\vec{k}_\tau = R^{-1}\vec{r}_\tau$. Then, $\langle s(t+\tau)\hat{s}(t+\tau)\rangle_t = \langle \hat{s}(t)^2\rangle_t$, and so then

$$\mathrm{PC}_\tau = \langle s(t+\tau)\hat{s}(t+\tau)\rangle_t = \sum_{n=1}^{\infty} R_{ss}(\tau+n)k_\tau(n) = \vec{r}_\tau^\top \vec{k}_\tau = \vec{r}_\tau^\top R^{-1}\vec{r}_\tau. \tag{C8}$$

As these $\vec{k}_\tau$'s are the causal linear filters that maximize the correlation coefficient between $s(t+\tau)$ and $\hat{s}(t+\tau)$, we have

$$\mathrm{PC} \leqslant \sum_{\tau=0}^{\infty} \vec{r}_\tau^\top R^{-1}\vec{r}_\tau \tag{C9}$$

for any linear recurrent network.

[1] H. Jaeger, The "echo state" approach to analyzing and training recurrent neural networks-with an erratum note, Bonn, Germany: German National Research Center for Information Technology GMD Technical Report No. 148 2001 (unpublished).

[2] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[3] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comput. Sci. Rev. **3**, 127 (2009).

[4] Memory can be used to estimate the bias of the coin, but nothing else about the past provides a guide to the future input.

[5] H. Jaeger, *Short term memory in echo state networks*, German National Research Center for Information Technology, Technical report GMD-Forschungszentrum Informationstechnik 2001 (unpublished).

[6] O. L. White, D. D. Lee, and H. Sompolinsky, Short-Term Memory in Orthogonal Neural Networks, Phys. Rev. Lett. **92**, 148102 (2004).

[7] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, Information processing in echo state networks at the edge of chaos, Theory Biosci. **131**, 205 (2012).

[8] I. Farkaš, R. Bosák, and P. Gergel', Computational analysis of memory capacity in echo state networks, Neural Networks **83**, 109 (2016).

[9] P. Barančok and I. Farkaš, Memory capacity of input-driven echo state networks at the edge of chaos, in *International Conference on Artificial Neural Networks* (Springer, Cham, 2014), pp. 41–48.

[10] M. Hermans and B. Schrauwen, Memory in linear recurrent neural networks in continuous time, Neural Networks **23**, 341 (2010).

[11] A. Goudarzi, S. Marzen, P. Banda, G. Feldman, C. Teuscher, and D. Stefanovic, Memory and information processing in recurrent neural networks, arXiv:1604.06929.

[12] S. Ganguli, D. Huh, and H. Sompolinsky, Memory traces in dynamical systems, Proc. Natl. Acad. Sci. USA **105**, 18970 (2008).

[13] P. M. Riechers, D. P. Varn, and J. P. Crutchfield, Pairwise correlations in layered close-packed structures, Acta Crystallogr. Sect. A: Found. Adv. **71**, 423 (2015).

[14] G. Fette and J. Eggert, Short term memory and pattern matching with simple echo state networks, in *Artificial Neural Networks: Biological Inspirations—ICANN 2005, Warsaw, 2005*, edited by W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny (Springer, Berlin/Heidelberg, 2005), pp. 13–18.

[15] T. Toyoizumi, Nearly extensive sequential memory lifetime achieved by coupled nonlinear neurons, Neural Comput. **24**, 2678 (2012).

[16] J. Collins, J. Sohl-Dickstein, and D. Sussillo, *ICLR 2017: 5th International Conference on Learning Representations, Toulon, France, 2017* (ICLR, 2017).