Difference between memory and prediction in linear recurrent networks

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th>Citation</th>
<th>Marzen, Sarah et al. &quot;Difference between memory and prediction in linear recurrent networks.&quot; Physical Review E 96, 3 (September 2017): 032308 © 2017 American Physical Society</th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1103/PhysRevE.96.032308">http://dx.doi.org/10.1103/PhysRevE.96.032308</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>American Physical Society</td>
</tr>
<tr>
<td>Version</td>
<td>Final published version</td>
</tr>
<tr>
<td>Accessed</td>
<td>Mon Apr 15 08:32:53 EDT 2019</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/114553">http://hdl.handle.net/1721.1/114553</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td></td>
</tr>
</tbody>
</table>

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
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)

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 (un-optimized) 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

\[
x(n + 1) = Wx(n) + s(n)v,
\]

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

\[
W = P \text{ diag}(\tilde{d}) P^{-1},
\]

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

\[
\omega = P^{-1} v.
\]

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

\[
R_{ss}(t) = \sum_{\lambda \in \Lambda} A(\lambda) \delta^{(t)},
\]

where \( \Lambda \) is a set of numbers with a magnitude less than 1. See Ref. [13] or Appendix A. To avoid normalization factors, we assert that

\[
R_{ss}(0) = \sum_{\lambda \in \Lambda} A(\lambda) = 1.
\]
The power spectral density of this input process with

\[ R_{xx}(t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(f) e^{ift} df \]

is

\[ S(f) = \sum_{k=\infty}^\infty R_{xx}(k)e^{-ifk} \]

\[ = \sum_{\lambda \in \Lambda} A(\lambda) \sum_{k=-\infty}^\infty \lambda^{-k} e^{-ifk} \]

\[ = \sum_{\lambda \in \Lambda} A(\lambda) \frac{1 - \lambda^2}{(1 - \lambda e^{-if})(1 - \lambda e^{if})}, \]

by the Wiener-Khinchin theorem.

II. RESULTS

The memory function is classically defined by [5]

\[ m(k) := p_k C^{-1} p_k, \]

where

\[ p_k = \langle s(n - k) x(n) \rangle_n, \]

and

\[ C := \langle x(n) x(n)^\top \rangle_n. \]

The expression for memory capacity is more involved and we define the predictive capacity as

\[ PC := \sum_{k=0}^\infty m(-k). \]

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 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 \( W \rightarrow 1 \) at which point \( x(n) \) is an average of observed \( s(n) \)—i.e., \( x(n) = \langle s(k) \rangle_{k \leq n} \). Interestingly, at that point, PC is minimized, i.e., \( PC = 0 \). 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: \( \lim_{W \to 1} MC = \coth(e^{\alpha/2}) \) and \( \lim_{W \to 0} PC = \frac{1}{\alpha} \) plus corrections of \( O(\alpha) \) and \( \lim_{W \to 0} PC = \frac{1}{\alpha} \) plus corrections of \( O(1) \).
calculate PC, MC for network no longer satisfies the echo state property, and so we only PC is maximized for some intermediate lines), computed using Eqs. (15) and (18) in the main text. Whereas mean-squared error is equivalent to maximizing the correlation for essentially is no reservoir. But such arg max predictive capacity of a reservoir as different for memory and prediction. MC for linear recurrent networks famously scales linearly with the number of nodes memorization of previous inputs by storing their average value. (green lines). Interestingly, we still minimize any error in (a)

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| \( \geq 1 \), the network no longer satisfies the echo state property, and so we only calculate PC, MC for |W| < 1.

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 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 memorialization 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 \( k_t(n) \) minimizes the mean-squared error \( \langle [s(n + \tau) - \delta(n + \tau)]^2 \rangle_n \) of future input \( s(n + \tau) \) and a forecast of future input from past input \( \delta(n + \tau) := \sum_{\nu=0}^{\infty} k_\nu(m)s(n - m) \). 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

\[
PC \leq \sum_{\tau=0}^{\infty} \tilde{r}_\tau R^{-1} \tilde{r}_{\tau}, \tag{20}
\]

where \( \tilde{r}_\tau = R_{ss}(\tau + i) \) 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 \( \leq 1.652 \), which nearly is attained by the optimal one-node network, for which max\( w \) PC is \( \approx 1.65 \). And this is not a special property for a cherry-picked input signal; similar results hold for other different randomly chosen \( \Lambda, A_\lambda \) 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 \( R_{ss}(t) = (-\frac{1}{2})^{|t|} \). 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: \( W_{ij},v_i \sim U[0,1] \), where \( W \) then is scaled so that its largest magnitude eigenvalue has an absolute value 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. Dedeco, 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

\[
T = \sum_x T^{(x)}
\]

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

\[
R(t) = \langle x(t - 1)x(0) \rangle = \sum_{x,x'} xx' \Pr(X_{t-1} = x, X_0 = x') \\sum_{x,x'} \tilde{X}_t \tilde{X}_t' \tilde{p}_{eq}
\]

\[
= \sum_{x,x'} \tilde{X}_t T^{(x)} T^{(x')} \tilde{p}_{eq}
\]

\[
= \sum_{x} \sum_{x'} \tilde{X}_t T^{(x)} T^{(x')} \tilde{p}_{eq}
\]

\[
= \sum_{x} \tilde{X}_t (x T^{(x)}) \tilde{X}_t' (x T^{(x)}) \tilde{p}_{eq}
\]

\[
= \tilde{X}_t \left( \sum_x x T^{(x)} \right) \tilde{X}_t \left( \sum_x x T^{(x)} \right) \tilde{p}_{eq}
\]

\[
= \tilde{X}_t (\sum_x x T^{(x)}) \tilde{p}_{eq}
\]

\[
= \tilde{X}_t (\sum_x x T^{(x)}) \tilde{p}_{eq}
\]

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

\[
\tilde{R}(t) = \tilde{X}_t \left( \sum_x x T^{(x)} \right) P \text{diag}(\tilde{\lambda}) P^{-1} \left( \sum_x x T^{(x)} \right) \tilde{p}_{eq}
\]

and so \( \tilde{R}(t) \) is a linear combination of \( \tilde{\lambda}_i^1 \).

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

From Eq. (1), we have

\[
\tilde{X}_t = \left( \sum_{k=1}^{\infty} W^{k-1} s(n-k) \right) v,
\]

assuming the echo state property. Thus,

\[
\tilde{p}_{eq} = (s(n-k)x(n))_{n}
\]

\[
= \sum_{m=1}^{\infty} W^{m-1} R_{ss}(k - m)v,
\]

and

\[
C = \langle x(x(n))_n \rangle
\]

\[
= \sum_{m,m'=1}^{\infty} W^{m-1} W^{m'-1} R_{ss}(m - m').
\]

Substituting Eq. (6) into the above equation gives

\[
C = \frac{1}{2\pi} \sum_{m,m'=1}^{\infty} W^{m-1} W^{m'-1} \int_{-\pi}^{\pi} S(f) e^{i f (m-m')} df
\]

\[
= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(f) \left( \sum_{m=1}^{\infty} e^{i m W} W^{-1} \right) \left( \sum_{m'=1}^{\infty} e^{-i m' W} W^{-1} \right) df
\]

\[
= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(f) \left( \sum_{m=0}^{\infty} e^{i m W} W^{-1} \right) \left( \sum_{m'=0}^{\infty} e^{-i m' W} W^{-1} \right) df
\]

\[
= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(f) (I - e^{i f W})^{-1} e^{-i f W} (I - e^{-i f W})^{-1} df,
\]

and using Eq. (2),

\[
C = \frac{1}{2\pi} P \left[ \int_{-\pi}^{\pi} S(f) \left( \frac{\omega}{1 - e^{i f d}} \right) \left( \frac{\omega}{1 - e^{-i f d}} \right)^{\top} df \right] p^{-1}.
\]

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

\[
p_k = \sum_{m=1}^{\infty} W^{m-1} \sum_{\lambda \in \Lambda} A(\lambda) \lambda^{k-m} v
\]

\[
= \sum_{\lambda \in \Lambda} A(\lambda) \sum_{m=1}^{\infty} W^{m-1} \lambda^{k-m} v
\]

\[
= \sum_{\lambda \in \Lambda} A(\lambda) \sum_{m=1}^{\infty} W^{m-1} \lambda^{k-m} v
\]
In matrix form, this reads

\[ \sum_{\lambda \in \Lambda} A(\lambda) \sum_{m=1}^{\infty} W_{m-1}^{\lambda} \chi_{m-k} v, \quad k < 1, \]

\[ \sum_{\lambda \in \Lambda} A(\lambda) \left( \sum_{m=1}^k W_{m-1}^{\lambda} \chi_{m-k} + \sum_{m=k+1}^{\infty} W_{m-1}^{\lambda} \chi_{m-k} \right) v, \quad k \geq 1 \]

Thus we have

\[ k\tau = \sum_{\lambda \in \Lambda} A(\lambda) \chi_{\lambda-k} W^{-1} \left( \sum_{m=1}^{\infty} W_{m} \lambda_{m} \right) v, \quad k < 1, \]

\[ \sum_{\lambda \in \Lambda} A(\lambda) \chi_{\lambda-k} (\lambda_1 - W)^{-1} v, \quad k < 1, \]

\[ \sum_{\lambda \in \Lambda} A(\lambda) [(W_k - \lambda^k)(W - \lambda)^{-1} + W^k (\lambda^{-1} - W)^{-1}] v, \quad k \geq 1. \]

Using Eq. (2),

\[ p_k = P \left[ \sum_{\lambda \in \Lambda} A(\lambda) \lambda^{-k} \left( \frac{\omega}{\lambda^{-1} - d} \right) \right], \quad k < 1, \]

\[ \sum_{\lambda \in \Lambda} A(\lambda) \text{diag} \left( \frac{d^k - \lambda^k}{d - \lambda} + \frac{\tilde{d}^k}{\lambda^{-1} - d} \right) \omega, \quad k \geq 1. \]

Thus we have

\[ \text{PC} = \sum_{k=0}^{\infty} p_{-k}^T C^{-1} p_{-k} \]

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

\[ = 2\pi \sum_{\lambda \in \Lambda} A(\lambda) A(\lambda') \left( \frac{\omega}{1 - \lambda \lambda'} \right)^T B^{-1} \left( \frac{\omega}{(\lambda')^{-1} - d} \right) \]

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

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

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

\[ \text{PC}_r = \frac{\langle \hat{s}(t + \tau) \hat{s}(t + \tau) \rangle_t^2}{\langle \hat{s}(t)^2 \rangle_t}, \]

and

\[ \text{PC} = \sum_{\tau=0}^{\infty} \text{PC}_r. \]

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

\[ R_{ss}(\tau + t) = \sum_{m=1}^{\infty} R_{ss}(t - m) k_r(m). \]

In matrix form, this reads

\[
\begin{pmatrix}
R_{ss}(\tau + 1) \\
R_{ss}(\tau + 2)
\end{pmatrix} =
\begin{pmatrix}
R_{ss}(0) & R_{ss}(-1) & R_{ss}(-2) & \cdots & k_r(1) \\
R_{ss}(1) & R_{ss}(0) & R_{ss}(-1) & \cdots & k_r(2) \\
\vdots & \vdots & \vdots & \ddots & \vdots
\end{pmatrix}
\]

\[ \text{PC}_r = \sum_{\tau=0}^{\infty} \text{PC}_r. \]
and so
\[
\begin{pmatrix}
(k_1(1) \\
k_2(2)
\end{pmatrix} = \begin{pmatrix}
R_{ss}(0) & R_{ss}(-1) & R_{ss}(-2) & \cdots \\
R_{ss}(1) & R_{ss}(0) & R_{ss}(-1) & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{pmatrix}^{-1} \begin{pmatrix}
R_{ss}(\tau + 1) \\
R_{ss}(\tau + 2) \\
\vdots
\end{pmatrix}.
\]
(C5)

For ease of notation, we define \( R \) as
\[
R := \begin{pmatrix}
R_{ss}(0) & R_{ss}(-1) & R_{ss}(-2) & \cdots \\
R_{ss}(1) & R_{ss}(0) & R_{ss}(-1) & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{pmatrix}.
\]
(C6)

and
\[
\bar{r}_\tau := \begin{pmatrix}
R_{ss}(\tau + 1) \\
R_{ss}(\tau + 2) \\
\vdots
\end{pmatrix},
\]
(C7)

so in short, \( \tilde{k}_\tau = R^{-1} \bar{r}_\tau \). Then, \( \langle s(t + \tau) \hat{s}(t + \tau) \rangle_t = \langle \hat{s}(t)^2 \rangle_t \), and so then
\[
\text{PC}_\tau = \langle s(t + \tau) \hat{s}(t + \tau) \rangle_t = \sum_{n=1}^{\infty} R_{ss}(\tau + n)k_\tau(n) = \bar{r}_\tau^T \tilde{k}_\tau = \bar{r}_\tau^T R^{-1} \bar{r}_\tau.
\]
(C8)

As these \( \tilde{k}_\tau \)'s are the causal linear filters that maximize the correlation coefficient between \( s(t + \tau) \) and \( \hat{s}(t + \tau) \), we have
\[
\text{PC} \leq \sum_{\tau=0}^{\infty} \bar{r}_\tau^T R^{-1} \bar{r}_\tau
\]
(C9)

for any linear recurrent network.

[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.