ARTICLE **Communicated by John Hertz**

Randomly Connected Networks Have Short Temporal Memory

Edward Wallace
ewjwallace@gmail.com
Department of Biochemistry and Molecular Biophysics, University of Chicago, Chicago, IL 60637, U.S.A., and FAS Center for Systems Biology, Harvard University, Cambridge, MA 02138, U.S.A.

Hamid Reza Maei
maei@stanford.edu
Electrical Engineering Department, Stanford University, Stanford, CA, U.S.A.

Peter E. Latham
pel@gatsby.ucl.ac.uk
Gatsby Computational Neuroscience Unit, University College, London, London WC1N 3AR, U.K.

The brain is easily able to process and categorize complex time-varying signals. For example, the two sentences, "It is cold in London this time of year" and "It is hot in London this time of year," have different meanings, even though the words *hot* and *cold* appear several seconds before the ends of the two sentences. Any network that can tell these sentences apart must therefore have a long temporal memory. In other words, the current state of the network must depend on events that happened several seconds ago. This is a difficult task, as neurons are dominated by relatively short time constants—tens to hundreds of milliseconds. Nevertheless, it was recently proposed that randomly connected networks could exhibit the long memories necessary for complex temporal processing. This is an attractive idea, both for its simplicity and because little tuning of recurrent synaptic weights is required. However, we show that when connectivity is high, as it is in the mammalian brain, randomly connected networks cannot exhibit temporal memory much longer than the time constants of their constituent neurons.

1 Introduction

To function well in the world, it is absolutely necessary to have a memory; without one, we would be reduced to a set of reflex arcs, able to carry out only the simplest tasks. On long timescales, hours or more, it is thought that memories are stored in synaptic weights. On shorter timescales, a few seconds, it is likely that information about the past is stored in patterns of activity. For example, at the ends of the two sentences, “It is cold in London this time of year” and “It is hot in London this time of year,” patterns of activity in at least some circuits in the brain must be sufficiently different to drive different behavior (packing a jacket for one’s trip to London in the first case, not packing one in the second).

Building a network whose state depends on input that occurred seconds to tens of seconds ago would seem difficult, as neurons have short memories—typically not more than a few hundred ms. However, a network’s time constant can be considerably longer than its single-neuron time constants; all that is required is sufficient positive feedback to make the network almost self-sustaining. As with the well-studied neural integrator, such feedback can boost single-neuron time constants by large factors—one to two orders of magnitude (Seung, 1996; Koulakov, Raghavachari, Kepecs, & Lisman, 2002; Brody, Romo, & Kepecs, 2003).

Jaeger (2001; Jaeger & Haas, 2004) used just this method—positive feedback—to achieve long time constants in randomly connected networks. That analysis, however, focused on analog neurons, and it was not immediately clear whether it would apply to networks of spiking neurons. This was partially remedied by Maass, Natschläger, and Markram (2002), who showed numerically that randomly connected networks of spiking neurons could exhibit at least some memory of past input (for a review, see Buonomano & Maass, 2009).

In spite of these encouraging results, whether realistic networks can exhibit long temporal memory is still an open question, in large part because networks of spiking neurons are so hard to analyze. Bertschinger and Natschläger (2004) took a step toward addressing this issue by considering a network in which such analysis was possible. What they found was that networks were able to exhibit long memories if they operated near the edge of chaos. This idea—that temporal memory is longest near the edge of chaos—was confirmed in subsequent studies (Maass, Legenstein, & Bertschinger, 2005; Legenstein & Maass, 2007; Büsing, Schrauwen, & Legenstein, 2010).

While Bertschinger and Natschläger’s analysis provided a great deal of insight into the mechanism by which networks process time-varying input, they considered networks with unrealistically low connectivity, as did subsequent studies of operation near the edge of chaos (Maass et al., 2005; Legenstein & Maass, 2007). Here we consider the more realistic case of high connectivity, in which the average number of connections per neuron is proportional to the number of neurons, and both are large. When we do that, we find a number of differences, one of which is that networks, or at least networks in the class Bertschinger and Natschläger considered, are guaranteed to be chaotic. Chaos is not an especially surprising feature of spiking networks (van Vreeswijk & Sompolinsky, 1996, 1998; Banerjee, 2006; Izhikevich & Edelman, 2008; Monteforte & Wolf, 2010; London, Roth, Beeren, Häusser, & Latham, 2010). Here, however, we go beyond just showing that networks are chaotic; our main result is that chaos implies a short temporal memory—not much longer than the time constants of the constituent neurons (a result partly anticipated by Ganguli, Huh, & Sompolinsky, 2008, although in analog rather than spiking networks). This suggests that randomly connected networks are not able to exhibit memory much longer than the time constants of their constituent neurons.

2 Temporal Memory in Recurrent Networks: Basic Principles

For a network to exhibit temporal memory, its state must depend on past input, not merely on current input. Probably the simplest way to think about this is as follows. A network receives one of two inputs before time $t = 0$, and it receives just one input after time $t = 0$ (see Figure 1a). If the input is strong and has been applied for a long time, as we assume here, the state of the network at $t = 0$ is very different under the two inputs. Consequently, at $t = 0$, an observer of the network would know which of the two inputs had been applied. After time $t = 0$, however, the network no longer receives any information about which of the two inputs was presented, and so it becomes harder and harder to distinguish them. What we want to know is how long before it is essentially impossible to distinguish them.

To develop the machinery to answer this question, we take a geometric approach in which network dynamics is described in terms of trajectories in state-space and statements about those trajectories are translated into statements about temporal memory. Specifically, suppose that each neuron in a network is described by one variable, so for N neurons, the state space is N-dimensional. The state of the network corresponds to a single point in this space, and trajectories correspond to paths. (To simplify the discussion, here we treat the state-space as continuous, although the model we analyze is discrete.)

It turns out that the temporal memory depends strongly on whether trajectories converge, diverge, or are neutrally stable, with both converging and diverging trajectories leading to short temporal memory and neutrally stable trajectories leading to long memory. This observation allows us to replace a hard question, "Can a network have long temporal memory?" with a much easier one, "What happens to nearby trajectories under the network dynamics?"

The three types of trajectories are illustrated in Figures 1b to 1d, with the input given in Figure 1a. We assume that before time $t = 0$, the two inputs are sufficiently different that the trajectories are easily distinguishable at time $t = 0$. After $t = 0$, the behavior—and the distance between trajectories—depends strongly on the dynamical regime.

Consider first the converging regime, illustrated in Figure 1b. In this regime, there is a general contraction of volume in state-space, so the

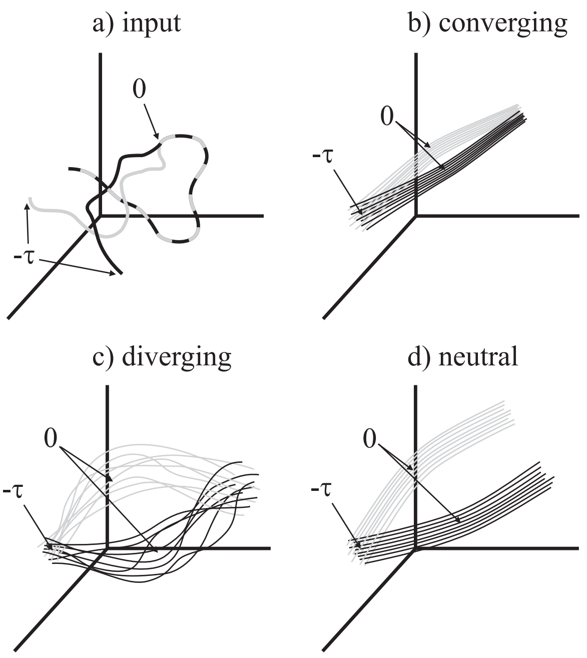

Figure 1: Networks evolve under one of three possible dynamical regimes. Although drawn as a three-dimensional space, the actual dimensionality is much higher: it is proportional to N, the number of neurons in the network. (a) The input to the network, starting at time $t = -\tau$. Before time $t = 0$, there are two possible inputs, shown as black and gray; after time $t = 0$, there is only one. The job of the network is to tell us which input was present after time $t = 0$. (b) Converging dynamics. After time $t = 0$, when trajectories receive the same input, they are pushed together—typically exponentially fast. In this regime, past input is quickly forgotten. (c) Diverging dynamics. Even when receiving the same input ($t > 0$), nearby trajectories are pushed apart. However, because activity space is bounded, all trajectories eventually converge to a chaotic attractor. If there is only one such attractor, as we assume here, then after convergence onto it, the network provides little information about which of the two inputs it received. (d) Neutrally stable dynamics. After time $t = 0$, trajectories are nearly parallel, and so the distance between them stays approximately constant. Because of nonlinearities, trajectories are not exactly parallel, so eventually the input becomes indistinguishable, but that takes a long time.

distance between trajectories decreases with time—typically exponentially. Thus, the trajectories very quickly merge, making it effectively impossible to distinguish the two inputs.

Diverging trajectories, illustrated in Figure 1c, would seem to offer much longer temporal memory: nearby trajectories are pushed apart, which over short timescales makes them easy to distinguish. However, if there is a divergence of trajectories in a bounded state-space, then trajectories that were initially far apart must eventually become very close. This implies mixing, and the trajectories quickly become hopelessly entangled, again making it essentially impossible to distinguish the two inputs. In the language of dynamical systems, diverging trajectories lead to chaotic dynamics, and trajectories approach a chaotic attractor. Once near the attractor, the input in the distant past, before time $t = 0$, becomes effectively unknowable.

Between diverging and converging lies a regime in which trajectories are neutrally stable, so they stay approximately the same distance apart (see Figure 1d) and thus are distinguishable for long periods. In this regime, sometimes referred to as the the edge of chaos (Langton, 1990; Kauffman & Johnsen, 1991), networks can exhibit long temporal memory.

This suggests a natural way to build networks that can distinguish among different inputs: tune them so that they are close to neutral stability. And indeed, both numerical and theoretical studies have shown that temporal memory is much longer in the neutrally stable regime than in the converging or diverging regimes (Bertschinger & Natschläger, 2004; Maass et al., 2005; Legenstein & Maass, 2007; Büsing et al., 2010). However, in all of these studies the connectivity was low—no more than about 30 connections per neuron. What happens when the connectivity is high, as it is in the brain? The answer is that it is not possible to access the neutrally stable regime. We show here that high-connectivity networks, at least the high-connectivity networks in the class considered by Bertschinger and Natschläger, have only diverging trajectories; that is, they are always chaotic. This rules out operation on the edge of chaos, and so rules out long temporal memory.

3 Model

The model we investigate is a sparse, randomly connected network of N synchronously updating McCullough-Pitts neurons, with N taken to be large. We work in discrete time; the length of the time-step is intended to be roughly the membrane time constant, about 10 ms. The update rule is

$$x_i(t+1) = \text{sgn}(h_i(t) + u_i(t)) \tag{3.1a}$$

$$h_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t), \tag{3.1b}$$

where $x_i(t)$ is the state of the ith neuron at time t, w_{ij} is the connection strength from neuron j to neuron i, and $u_i(t)$ is the input at time t. Because of the sign function, $x_i(t)$ is either -1 or $+1$, with -1 indicating silence and $+1$ indicating spiking. Following Bertschinger and Natschläger (2004), we let the connection strength, w_{ij}, be independent and identically distributed (i.i.d.) with zero mean. We set the variance, the main statistical property we will need, to σ_w^2/N. In addition, we use sparse connectivity, meaning the neurons make an average of K connections out of a possible N, with $K < N$. Thus, the weights are drawn i.i.d. from the distribution $P_w(w)$, which is given by

$$P_w(w) = (1 - K/N)\delta(w) + (K/N)P_w^c(w), \tag{3.2}$$

where $\delta(\cdot)$ is the Dirac delta function and $P_w^c(w)$, which is the distribution of nonzero weights (the superscript c stands for "connected"), has zero mean and a variance of σ_w^2/K, the latter chosen to ensure that the total variance of the weights is σ_w^2/N.

Finally, we need to choose a form for the input, $u_i(t)$. For ease of calculation, we assume that it is i.i.d. and gaussian with mean $\overline{u}$ and variance σ_u^2,

$$P(u) = \frac{e^{-(u-\overline{u})^2/2\sigma_u^2}}{(2\pi\sigma_u^2)^{1/2}}. \tag{3.3}$$

We also assume that the u_i are drawn independently on each time step, so $u_i(t)$ and $u_j(t')$ are uncorrelated if $t \neq t'$, even when $i = j$.

Although our network does not obey Dale's law, it retains what is probably the most important feature of real neuronal networks, at least when they are operating in the asynchronous regime: neurons are driven by fluctuations in synaptic drive. Indeed, for balanced networks that obey Dale's law, once balance is enforced, spikes are produced by a term much like the synaptic drive, $h_i(t)$, that appears in equation 3.1b (van Vreeswijk & Sompolinsky, 1998). Thus, although we are using a simple network, we expect our conclusions, especially the relationship between chaos and temporal memory, to apply to more realistic networks.

3.1 Population Firing Rates. When the network-averaged firing rate (the fraction of active spikes per time step) is small, equation 3.1 describes dynamics that is reminiscent of what real neurons do: the total synaptic drive, $h_i(t) + u_i(t)$, is typically below zero (and thus below threshold for firing); only occasionally does it cross zero. This is illustrated in Figure 2a where we plot the total synaptic drive for two representative neurons, with spikes shown whenever the drive crosses zero. We plot spike rasters in Figure 2b. These are similar to the rasters seen in any network operating in the asynchronous regime.

The apparent randomness of the synaptic drive plotted in Figure 2a suggests that we could analyze the network by treating it probabilistically. We do this by approximating the total synaptic drive, $h_i(t) + u_i(t)$, as a gaussian random variable with respect to the index i (van Vreeswijk & Sompolinsky,

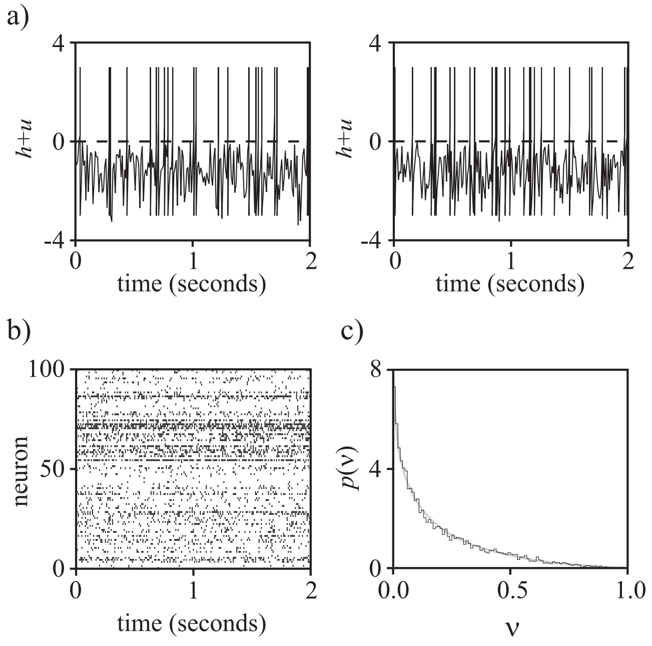

Figure 2: Network simulations show irregular spike trains. (a) Total synaptic drive, $h_i(t) + u_i(t)$, for the bottom two neurons in panel b. Spikes (sharp upward deflections) were added artificially whenever the voltage crossed the threshold at 0. To obtain the timescale on the x-axis, we assume each time step corresponds to 10 ms. (b) Spike rasters for the first 100 neurons. (c) Firing rate distribution, computed empirically (black) and from equation 3.11 (gray). Parameters were $N = 8{,}192$, $\sigma = 1$, $\sigma_u = 0.5$, $\overline{u} = -0.941$, the last chosen to ensure a firing rate of 0.2 (corresponding to 20 Hz under our assumption of a 10 ms time step), and $K/N = 0.2$.

1998). It is not entirely obvious that this approximation is valid, especially given that the network equations are deterministic. Nevertheless, it gives very accurate results; we take that accuracy as an indication that the approximation is a good one.

As discussed above, the quantity of interest is the rate at which trajectories diverge. Before computing that (which we do in the next section), we compute the firing rate, as we will need it later, it is relatively simple to compute, and the techniques will be useful for subsequent analysis.

In the probabilistic spirit, the population-averaged firing rate, denoted $\overline{\nu}$, is given by

$$\overline{\nu} = \frac{1}{N}\sum_i \frac{\langle x_i(t) + 1\rangle_t}{2} = \frac{1}{N}\sum_i \langle \Theta[h_i(t) + u_i(t)]\rangle_t \tag{3.4}$$

where Θ is the Heaviside step function: $\Theta(y) = 1$ if $y > 0$ and 0 otherwise. Essentially $\overline{\nu}$ is the probability that x_i is 1, averaged over both index i and time t.

To compute the right-hand side of equation 3.4, we treat both $u_i(t)$ and $h_i(t)$ as random variables with respect to index, i. This allows us to turn sums over index into integrals over distributions,

$$\frac{1}{N}\sum_i \langle \Theta[h_i(t) + u_i(t)]\rangle_t \rightarrow \int du\, dh\, P(u,h)\Theta(u+h). \tag{3.5}$$

Because $h_i(t)$ depends on $u_i(t-1)$ (see equation 3.1) and $u_i(t)$ and $u_i(t-1)$ are independent, it follows that $h_i(t)$ are $u_i(t)$ are independent. Consequently, $p(u, h) = p(u)p(h)$. The distribution of u is given in equation 3.3, so all we need is the distribution of h. Based on the fact that h is the sum of a large number of variables (see equation 3.1b), we treat it as a gaussian random variable. This is not exactly right, as the variables in the sum are correlated, but we assume, without proof, that the correlations are weak enough to be ignored. We take the good agreement between theoretical predictions and simulations (see Figures 5 and 7) as evidence that this assumption is valid.

With the gaussian assumption, we need only the mean and variance of h. Its mean is zero (because w_{ij} has zero mean) and its variance is given by, via equation 3.1b,

$$\mathrm{Var}[h] = \frac{1}{N}\sum_i \left[\sum_j w_{ij}x_j\right]^2 = \frac{1}{N}\sum_{ijj'} w_{ij}w_{ij'}x_j x_{j'}. \tag{3.6}$$

The weights are i.i.d. with zero mean and variance σ_w^2/N, so in the large N limit, the average over i (which involves only the weights) is $\delta_{jj'}\sigma_w^2/N$ where $\delta_{jj'}$ is the Kronecker delta. Consequently,

$$\mathrm{Var}[h] = \frac{\sigma_w^2}{N}\sum_j x_j^2 = \sigma_w^2. \tag{3.7}$$

The second equality follows from the fact that x_j is ± 1.

Using equations 3.3 and 3.7, and noting that $h_i(t)$ and $u_i(t)$ are independent, we see that $h_i + u_i$ is gaussian with mean $\overline{u}$ and variance $\sigma_u^2 + \sigma_w^2$. Thus,

$$P(h+u) = \frac{e^{-(h+u-\overline{u})^2/2\sigma_{\mathrm{tot}}^2}}{(2\pi\sigma_{\mathrm{tot}}^2)^{1/2}}, \tag{3.8}$$

where

$$\sigma_{\text{tot}}^2 \equiv \sigma_w^2 + \sigma_u^2. \tag{3.9}$$

Equation 3.4 tells us that $\overline{\nu}$ is the probability that $h + u > 0$. Since $h+u$ is gaussian (see equation 3.8), we see that

$$\overline{\nu} = \Phi(\overline{u}/\sigma_{\text{tot}}). \tag{3.10}$$

Here Φ is the standard cumulative normal function, defined explicitly in equation A.7. Assuming, as above, that one time step corresponds to 10 ms, the actual firing rate, in Hz, is $100\overline{\nu}$. For most of our simulations, we use $\overline{\nu} = 0.2$, which corresponds to 20 Hz.

Using similar analysis, it is relatively straightforward to compute the distribution of firing rates, something that is useful for verifying the validity of our approximations. This analysis is performed in the appendix (see section A.1), where we show that

$$p(\nu) = \frac{\beta}{\alpha}\, e^{(\overline{u}+\alpha\eta(\nu))^2/2\beta^2 - \eta(\nu)^2/2}, \tag{3.11}$$

where $\eta(\nu)$ must be found by inverting the equation $\nu(\eta) = \Phi((\overline{u} + \alpha\eta)/\beta)$ (see equation A.8) and α and β are parameters that must be determined from a set of nonlinear mean field equations (see equations A.3, A.5, and A.9). We compare this distribution to simulations in Figure 2c and see that agreement is very good.

3.2 Evolution of Nearby Trajectories. The temporal memory of a network depends on the behavior of nearby trajectories, with short temporal memory if trajectories either diverge or converge and long memory at the boundary between the two. To determine the behavior of nearby trajectories, we compute the time evolution of the distance between two trajectories that differ only in their initial conditions. We use superscripts 1 and 2 to denote the activity of two trajectories, and for the measure of distance, we use the natural one for our model, the normalized Hamming distance. For this measure, the distance at time t, denoted $d(t)$, is given by

$$d(t) \equiv \frac{1}{N}\sum_i \frac{1}{2}\left|x_i^{(1)}(t) - x_i^{(2)}(t)\right|.$$

Our goal is to compute the distance at time $t+1$ as a function of the distance at time t. To do that, it is helpful to work with synaptic drive rather than activity, so we use equation 3.1a to write

$$d(t+1) = \frac{1}{N}\sum_i \frac{1}{2}|\text{sgn}(h_i^{(1)}(t)+u_i(t)) - \text{sgn}(h_i^{(2)}(t)+u_i(t))|_d,$$

where the synaptic input, $h_i^{(k)}(t), k = 1, 2$, is given by equation 3.1b but with $x_i(t)$ in that equation replaced by $x_i^{(k)}(t)$, and the subscript d indicates that the trajectories associated with $h_i^{(1)}(t)$ and $h_i^{(2)}(t)$ are a distance d apart. It is convenient to define

$$d(t+1) \equiv f(d(t)), \tag{3.12}$$

as this allows us to drop the time dependence in equation 3.12, and write

$$f(d) = \frac{1}{N}\sum_i \frac{1}{2}|\text{sgn}(h_i^{(1)}+u_i) - \text{sgn}(h_i^{(2)}+u_i)|_d. \tag{3.13}$$

Note that we are assuming that the right-hand side depends only on the distance between trajectories. In principle, the right-hand side could depend on dynamical variables (e.g., population firing rate) as well as d. However, as we will see below, it does not.

The right-hand side of equation 3.13 can be computed using the same techniques we used to compute the mean firing rate; that computation is carried out in the appendix (see section A.2). However, we are interested primarily in the small d limit, and for that we can use qualitative arguments. Those arguments start with two observations: (1) the only nonzero terms in equation 3.13 are those for which $h_i^{(1)}+u_i$ and $h_i^{(2)}+u_i$ have opposite signs, and (2) when d is small, $h_i^{(1)}$ and $h_i^{(2)}$ are close to each other. Together, these imply that for $h_i^{(1)}+u_i$ and $h_i^{(2)}+u_i$ to have opposite signs, both must be close to zero. Specifically, both must be within about Δh of zero, where Δh is the typical size of $h_i^{(1)}-h_i^{(2)}$. (For example, if $h_i^{(1)}+u_i$ is $+10\Delta h$, it is highly unlikely that $h_i^{(2)}+u_i$ will be negative; that would represent a 10-standard-deviation outlier.)

This is illustrated in Figure 3, which shows the distribution of $h_i^{(1)}+u_i$ with respect to index, i (see Figure 3a), and the distribution of $h_i^{(1)}-h_i^{(2)}$, also with respect to index, i (see Figure 3b). The synaptic drives that contribute to the sum are shown as a gray region around $h_i^{(1)}+u_i=0$; the width of this region is the same as the width of the bottom plot. If Δh is small (which we expect it to be if d is small), the area of the gray region scales as

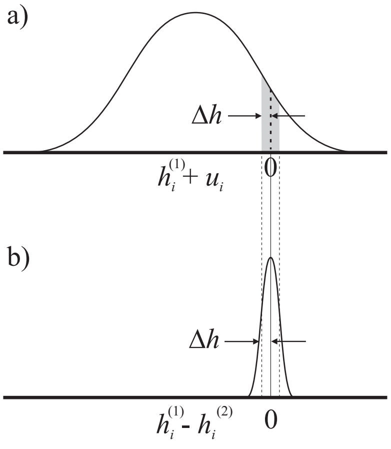

Figure 3: For small distances, the evolution of distance is determined by near-threshold neurons. (a) Distribution of $h_i^{(1)} + u_i$ with respect to index, i. The gray region corresponds to synaptic drives for which $h_i^{(1)} + u_i$ and $h_i^{(2)} + u_i$ are reasonably likely to have different signs, and thus contribute to $f(d)$ in equation 3.13. (b) Distribution of $h_i^{(1)} - h_i^{(2)}$ with respect to index, i. The width, Δh, is the standard deviation of $h_i^{(1)} - h_i^{(2)}$.

$P(h + u = 0) \times \Delta h$, which implies that

$$f(d) \propto P(h + u = 0)\, \Delta h(d), \tag{3.14}$$

where we have now explicitly included the fact that Δh depends on distance, d.

To compute $\Delta h(d)$, we use equation 3.1b to write

$$h_i^{(1)} - h_i^{(2)} = \sum_j w_{ij} (x_j^{(1)} - x_j^{(2)}) \tag{3.15}$$

and then compute the variance of the right-hand side. Using the fact that $d = P(x_j^{(1)} \neq x_j^{(2)})$ and applying our standard assumption that the x_i are uncorrelated, we see that the sum on the right-hand side consists of approximately Kd uncorrelated, zero mean random variables of size $\pm 2w_{ij}$. If Kd is large, this sum is approximately gaussian, so all we need are its mean and variance. The mean is zero (because w_{ij} is zero mean), and the variance is, by the central limit theorem, $Kd \times \mathrm{Var}[2w] = 4Kd\sigma_w^2/K$. Consequently, $\Delta h(d) = 2\sigma_w d^{1/2}$, and equation 3.14 becomes

$$f(d) \propto 2\sigma_w P(h + u = 0)\, d^{1/2}. \tag{3.16}$$

The key result is that $f(d)$ is proportional to $d^{1/2}$, something that was found by van Vreeswijk and Sompolinsky (1998) for a similar model but with excitatory and inhibitory neurons. This square root scaling with d is important because it means nearby trajectories diverge no matter what the network parameters; that is, no matter what the scaling factor in front of $d^{1/2}$. In fact, it is not hard to show, simply by iterating the equation $d(t+1) = f_0 d(t)^{1/2}$, that for t not too large and $d(0)$ very small,

$$d(t) = f_0^2 \left[d(0)/f_0^2\right]^{1/2^t}. \tag{3.17}$$

The exponent $1/2^t$ indicates very rapid divergence. For example, if $t = 10$, it says that we should take the 1024th root of $d(0)/f_0^2$. When $d(0) = 1/N$, which is as small as it can possibly be, $d(t = 10) \approx f_0^2[1 - (\log N f_0^2)/1024]$, which is close to f_0^2 for any reasonable value of N.

The full expression for $f(d)$, which we derive in the appendix (see section A.2, especially equation A.17), is not very illuminating. However, the small d limit of the full expression is somewhat more interesting:

$$f(d) \approx (2/\pi)^{1/2}\, 2\sigma_w \, \frac{e^{-\bar{u}^2/2\sigma_{\text{tot}}^2}}{(2\pi\sigma_{\text{tot}}^2)^{1/2}}\, d^{1/2}. \tag{3.18}$$

Except for the leading factor of $(2/\pi)^{1/2}$, this is exactly the approximate expression we derived above (see equation 3.16 for $f(d)$ and 3.8 for $P(h+u)$). A plot of the full expression for $f(d)$, via equation A.17, is shown in Figure 4, along with the small d limit given in equation 3.18; they are almost indistinguishable. Moreover, when we simulate the network and compute the distance between trajectories, we find that it evolves almost exactly as predicted by equation 3.12 (see Figure 5a).

What are the implications of the shape of $f(d)$ for the behavior of nearby trajectories? As shown in Figure 5a, when we iterate the update equation, equation 3.12, with $f(d)$ given in equation A.17, the distance increases rapidly with each iteration. In addition, at long times, the distance converges to the intersection of $f(d)$ with the 45 degree line—the point where $d = f(d)$. That intersection, which we denote d^*, corresponds to the equilibrium distance between trajectories.

Note that this analysis is valid only if $Kd \gg 1$, a condition that is necessary for us to treat the sum in equation 3.15 as gaussian. This is a reasonable condition, as we are assuming that K is large. However, the minimum value of d is $1/N$, and so Kd can be as small as K/N. Below, we consider the small Kd limit. Although somewhat different analysis is required, our main conclusions are substantially the same.

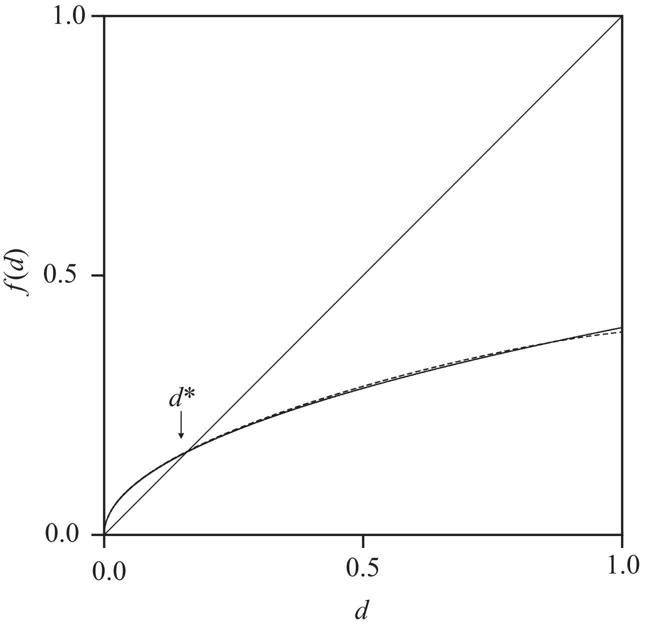

Figure 4: The distance evolution function for the same network as in Figure 2. Dashed line: Exact expression, equation A.17. Solid line: Approximate expression, equation 3.18. The intersection of $f(d)$ with the line at 45 degrees is the equilibrium distance, d^*; for these parameters, $d^* = 0.162$.

4 Memory Lifetime

The above analysis tells us that nearby trajectories diverge. Can it also provide us with a quantitative estimate of memory lifetime? The answer is yes, although we need to investigate the behavior of the network when d is near the equilibrium distance, d^*, rather than when it is near 0. In the next three sections, we look at memory lifetime in several ways. First we provide a naive, but essentially correct, analysis; then we address memory lifetime using an optimal linear classifier; and finally we examine the regime in which Kd is small.

4.1 Memory Lifetime I: Naive Analysis. To compute memory lifetime, we consider our usual scenario in which a network receives one of two inputs, either $\mathbf{u}^{(1)}(t)$ or $\mathbf{u}^{(2)}(t)$, up to time $t = 0$; after that, it receives just one, $\mathbf{u}(t)$. Before time $t = 0$, when the network is receiving different inputs, the distance between trajectories is larger than the equilibrium distance, d^* (see Figure 5b; $t \leq 0$). After $t = 0$, the distance decreases and eventually

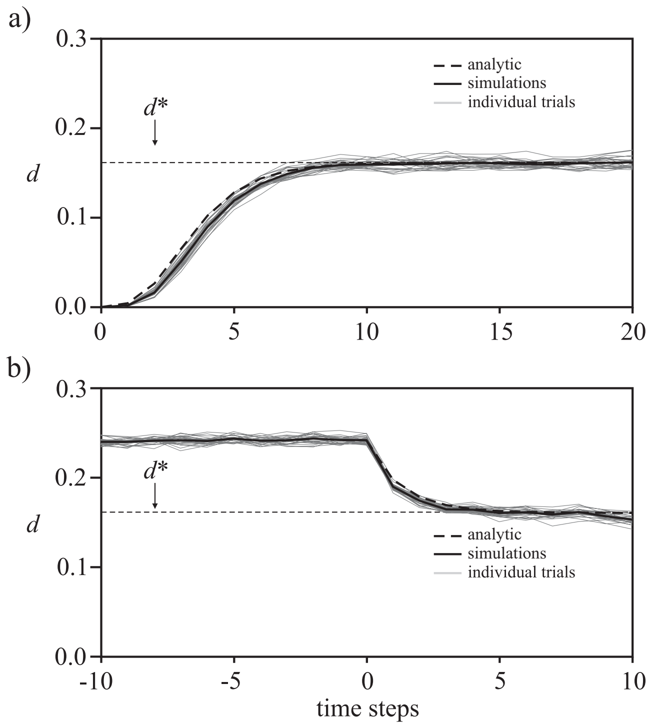

Figure 5: The evolution of distance between trajectories, using the same network as in Figures 2 and 4. (a) Distance, $d(t)$, versus (discrete) time for a network receiving the same input on different trials. The initial distance, $d(0)$, was $1/N$ (=1/8192). Gray lines: simulations; 25 different realizations of the network. Solid line: average of the gray lines. Dashed line: analytic prediction, determined by iterating the equation $d(t+1) = f(d(t))$ with $f(d)$ given by equation A.17. The slight disagreement is because when $d = 1/N$, $Kd = K/N$; this violates our assumption that $Kd \gg 1$. The resulting error propagates to later times. (b) Distance, $d(t)$, versus (discrete) time for a network receiving different input before $t = 0$ and the same input starting at $t = 0$. Gray lines: simulations; 25 different initial conditions. Solid line: average of the gray lines. Dashed line: analytic prediction, determined by iterating the equation $d(t+1) = f(d(t))$ with $f(d)$ given by equation A.17.

approaches d^* (see Figure 5b; $t > 0$). Consequently, information about what happened before $t = 0$ should be lost. How fast it is lost depends on how fast the distance, $d(t)$, approached d^* relative to the fluctuations in $d(t)$.

To determine the rate at which $d(t)$ approaches d^*, we use the distance update equation, equation 3.12. Because we are interested in large times, for which $d(t)$ is near d^*, we can linearize that equation around d^*. This yields

$$d(t+1) - d^* \approx f'(d^*)(d(t) - d^*).$$

Solving for $d(t)$ yields, in the large t limit,

$$d(t) - d^* \propto e^{-\lambda t}\,, \tag{4.1}$$

where λ, the rate of convergence to the fixed point, is given by

$$\lambda = -\log f'(d^*).$$

The other quantity we need is the size of the fluctuations. Assuming, as usual, that the neurons are independent, the fluctuations should scale as $1/N^{1/2}$. Consequently, information about the past starts to degrade when $d(t) - d^*$ is on the order of $1/N^{1/2}$. Using t^*_{naive} to denote the (naively computed) memory lifetime and using equation 4.1 for $d(t) - d^*$, we have $e^{-\lambda t^*_{naive}} \propto 1/N^{1/2}$. Solving for t^*_{naive} yields

$$t^*_{naive} \sim \frac{1}{2\lambda} \log N, \tag{4.2}$$

where $\sim$ indicates equality up to an additive constant. The key result here is the scaling: memory lifetime increases only with the log of the network size.

4.2 Memory Lifetime II: Optimal Linear Classifier. While the above analysis provides qualitatively correct scaling, it does not give a prescription for using network activity to provide information about past input (nor, as we will see below, does it give us the correct prefactor). To remedy that, we compute the optimal linear classifier and calculate how fast its performance degrades with time. Using an optimal linear classifier to provide long temporal memory is the hope of the liquid state machine of Maass et al. (2002) and Bertschinger and Natschläger (2004) and of the echo state network of Jaeger and colleagues (Jaeger, 2001; Jaeger & Haas, 2004).

A linear classifier is a vector, $\mathbf{J}(t)$, that is used to distinguish between the two trajectories, $\mathbf{x}^{(1)}(t)$ and $\mathbf{x}^{(2)}(t)$. Here our goal is to choose $\mathbf{J}(t)$ so that if input 1 is shown, $\mathbf{J}(t) \cdot \mathbf{x}(t)$ is above some threshold, and if input 2 is shown, $\mathbf{J}(t) \cdot \mathbf{x}(t)$ is below that threshold (here "$\cdot$" denotes the standard dot product). In addition, we demand that $\mathbf{J}(t)$ is optimal, in the sense that it maximizes the mean difference between $\mathbf{J}(t) \cdot \mathbf{x}^{(1)}(t)$ and $\mathbf{J}(t) \cdot \mathbf{x}^{(2)}(t)$

relative to its standard deviation. Consequently, $\mathbf{J}(t)$ is given by

$$\mathbf{J}(t) = \arg\max_{\mathbf{J}'} \frac{[\mathbf{J}' \cdot \langle \mathbf{x}^{(1)}(t) - \mathbf{x}^{(2)}(t) \rangle]^2}{\mathrm{Var}[\mathbf{J}' \cdot (\mathbf{x}^{(1)}(t) - \mathbf{x}^{(2)}(t))]}, \tag{4.3}$$

where angle brackets without a subscript t denote an average over trials.

Note that $\mathbf{J}$ depends on t. This makes readout difficult in practice, since a linear decoder would have to be constructed for each time point, which produces issues of multiple comparisons. We will ignore those issues here, so the performance we derive is an upper bound on the performance of a practical classifier.

Assuming, as usual, that the x_i and x_j are uncorrelated if $i \neq j$, we can find the optimal weights by differentiating the right-hand side of equation 4.3 with respect to $\mathbf{J}'$ and setting the resulting expression to zero. This leads to

$$J_i(t) = \frac{\langle x_i^{(1)}(t) - x_i^{(2)}(t) \rangle}{N\sigma_i^2(t)}, \tag{4.4}$$

where $\sigma_i^2(t)$ is the variance of $x_i^{(1)}(t) - x_i^{(2)}(t)$. Because $x_i^{(1)}$ and $x_i^{(2)}$ are independent and they take on only the values ± 1, it is given by

$$\sigma_i^2(t) = 1 - \langle x_i^{(1)}(t) \rangle^2 + 1 - \langle x_i^{(2)}(t) \rangle^2.$$

The expression for $J_i(t)$ is entirely reasonable: it is large for neurons whose mean difference in activity is large and whose variance is small.

Figure 6 shows the distribution, across trials, of $\mathbf{J}(t) \cdot \mathbf{x}(t)$ at several time points, both when stimulus 1 is shown (left distribution) and when stimulus 2 is shown (right distribution). As is clear from this figure, the classification error is the probability of the distribution leaking across the midline, denoted θ (see the dashed vertical line in Figure 6); that is, it is the probability that $\mathbf{J}(t) \cdot \mathbf{x}(t) < \theta$ when stimulus 1 is shown or, equivalently, the probability that $\mathbf{J}(t) \cdot \mathbf{x}(t) > \theta$ when stimulus 2 is shown, where θ is given by

$$\theta \equiv \frac{\mathbf{J}(t) \cdot \left\langle \mathbf{x}^{(1)}(t) + \mathbf{x}^{(2)}(t) \right\rangle}{2}.$$

Assuming that trial-to-trial fluctuations in both $\mathbf{J}(t) \cdot \mathbf{x}^{(1)}(t)$ and $\mathbf{J}(t) \cdot \mathbf{x}^{(2)}(t)$ are gaussian with the same variance (as is consistent with Figure 6), to compute the probability of an error, all we need is the ratio of the mean of $\mathbf{J}(t) \cdot \mathbf{x}(t) - \theta$ to its variance. This quantity, denoted $Z(t)$, is given by

$$Z(t) = \frac{\mathbf{J}(t) \cdot \langle \mathbf{x}^{(1)}(t) - \mathbf{x}^{(2)}(t) \rangle / 2}{\mathrm{Var}[\mathbf{J}(t) \cdot (\mathbf{x}^{(1)}(t) - \mathbf{x}^{(2)}(t))/2]^{1/2}}, \tag{4.5}$$

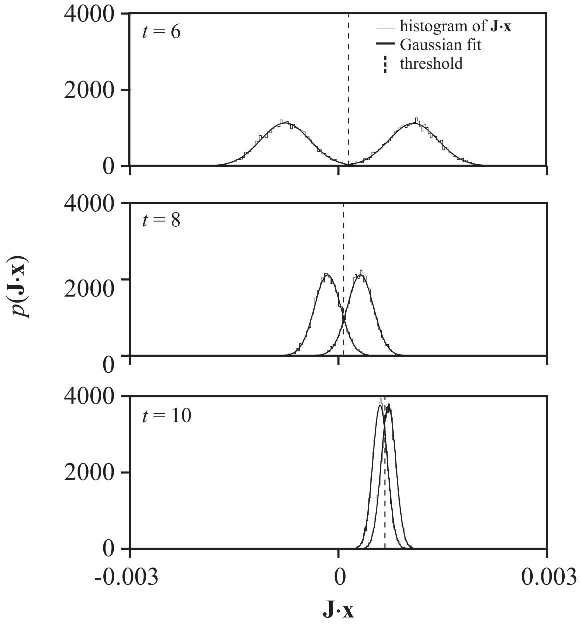

Figure 6: A linear classifier distinguishes past inputs for a limited time. Distribution of $\mathbf{J}(t) \cdot \mathbf{x}(t)$, the output of the linear classifier, at times $t = 6$, 8, and 10. Each distribution (thin lines) came from 10,000 trials using the same network as in Figures 2, 4, and 5, except with 16,384 rather than 8,192 neurons. For the left distribution in each panel, the network was receiving input 1; for the right distribution, it was receiving input 2. The thick curves are gaussian distributions with mean and variance computed from the histograms. The dashed vertical lines are the thresholds, θ. See the right-most curve in Figure 7a for the fraction correct at each of these time points.

and the fraction correct, denoted $\mathcal{P}_c(t)$, is given by

$$\mathcal{P}_c(t) = \Phi(Z(t)), \tag{4.6}$$

where, recall, Φ is the cumulative normal function (defined in equation A.7).

It is difficult to compute $Z(t)$ exactly, but we can determine how it scales with the distance between the two trajectories. The analysis, which is relatively straightforward, is carried out in the appendix (see section A.3, in particular, equation A.24). There we find that the numerator scales as $d(t) - d^*$, and the square of the denominator scales as $[d(t) - d^* + \mathcal{O}(1/R)]/N$, where R is the number of trials used for training and testing (assumed to be the same). Consequently, using the fact that in the large t limit, $d(t) - d^* \sim e^{-\lambda t}$ (see equation 4.1), we see that

$$Z(t) = c_1 \left[\frac{N e^{-\lambda t}}{1 + c_0 e^{\lambda t}/R} \right]^{1/2}, \tag{4.7}$$

where c_0 and c_1 are $\mathcal{O}(1)$ quantities that depend on network parameters. Note that equality holds only when N, R, and t are large. The memory lifetime, denoted t^*, is found by setting $Z(t)$ to a constant in equation 4.7. Using Z_0 for the constant, we find that

$$t^* = \frac{1}{\lambda} \log \left[\frac{\left[1 + 4(c_0 c_1^2/Z_0^2)N/R\right]^{1/2} - 1}{2c_0/R} \right]. \tag{4.8}$$

Although the particular form for t^* isn't especially intuitive, it is reasonable that the quality of the decoder, and thus the memory lifetime, should depend on the number of trials, R, used to train the linear classifier. Fortunately, the expression for the memory lifetime greatly simplifies in the limit of a large and small number of trials, where "large" and "small" are relative to the size of the network, N.

In the limit of a large number of trials ($R \gg N$), equation 4.8 becomes

$$t^*_{R \gg N} \sim \frac{1}{\lambda} \log N,$$

where, as above, $\sim$ indicates equality up to an additive constant. Note that this scaling is a factor of two better than that of the naive analysis given in equation 4.2. The increase in lifetime arises because the variance of the optimal estimator, and not just the mean, decreases as $d(t)$ approaches d^* (see Figure 6), something we did not take into account in our naive analysis. However, it is not an increase that is very helpful, as memory lifetime still scales with the log of the network size. Moreover, for sufficiently large N, we always enter the opposite limit, $R \ll N$. There we recover the naive scaling,

$$t^*_{R \ll N} \sim \frac{1}{2\lambda} \log N.$$

To verify that this analysis is correct, we computed the fraction correct from simulations based on the the optimal linear classifier given in equation 4.4, and compared that to the predicted fraction correct, $\mathcal{P}_c(t)$, given in equation 4.6. The results are shown in Figure 7 for two networks—our standard one (see Figures 7a and 7c) and one with slightly higher firing rate, 0.3 instead of 0.2 (see Figures 7b and 7d, and the figure caption for network details). In Figures 7a and 7b, we plot fraction correct versus time

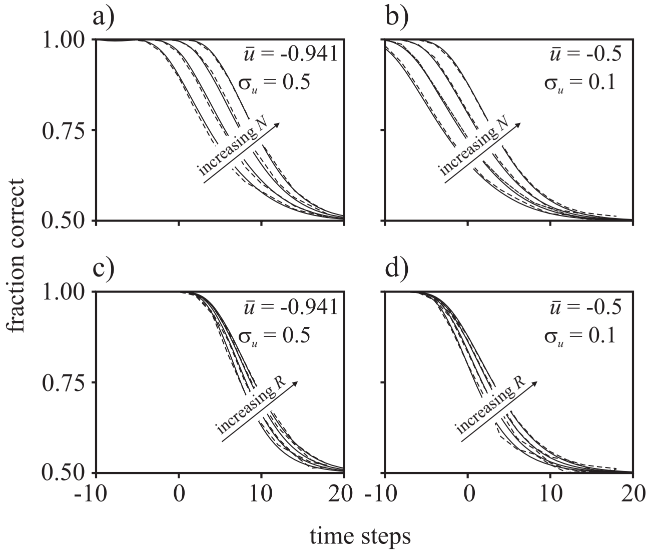

Figure 7: Accuracy of the optimal linear classifier increases with network size and number of trials. In all panels, we plot fraction correct from simulations (dashed lines) and the analytic prediction, $\mathcal{P}_c(t)$, given in equation 4.6 (solid lines). For panels a and c, we used the same network as in Figures 2 and 4–6; in panels b and d, we used $\bar{u} = -0.5$ instead of -0.941 and $\sigma_u = 0.1$ instead of 0.5, which yielded a slightly higher average firing rate, 0.3 spikes/neuron on each time step rather than 0.2. We adjusted the two free parameters in equation 4.7, c_0 and c_1, to match the fraction correct with $N = 16{,}384$ (note that this yielded different values of c_0 and c_1 for panels a and c versus b and d). (a, b) $R = 10{,}000$ training and testing examples; $N = 2{,}048, 4{,}096, 8{,}192$, and 16,384 neurons, with N increasing from left to right. (c, d) $N = 16{,}384$; $R = 1{,}250$, 2,500, 5,000 and 10,000 training and testing examples, with R increasing from left to right.

for different network sizes, N; for these panels, we used 10,000 trials to train and test the linear classifier. The right-most curve corresponds to 16,384 neurons; for the other curves, we used a factor of 2, 4, and 8 fewer neurons. The approximately equal spacing between the curves verifies the $\log N$ scaling. In panels Figures 7c and 7d, we fixed the network size at 16,384 neurons and varied R, the number of training and testing examples. The right-most curve corresponds to $R = 10{,}000$; for the other curves, we used factor of 2, 4, and 8 fewer examples. The decrease in memory lifetime is consistent with the prediction given in equation 4.8.

4.3 Memory Lifetime III: The Edge of Chaos. The results so far would seem bad for temporal memory: the optimal linear classifier has a memory

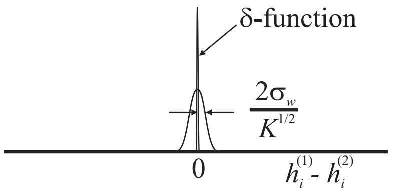

Figure 8: Distribution of $h_i^{(1)} - h_i^{(2)}$ when $Kd \ll 1$ is a mixture of a δ-function and a gaussian. The central peak is meant to represent a δ-function at zero with area $1 - Kd$; it corresponds to the synaptic drives for which $h_i^{(1)} = h_i^{(2)}$. The broad gaussian portion corresponds to the distribution associated with synaptic drives that differ in one presynaptic neuron; its total weight is Kd, and its width is $2\sigma_w/K^{1/2}$, the standard deviation of the nonzero weights.

lifetime that scales as the log of the network size, and the square root singularity in $f(d)$ indicates faster than exponential divergence of nearby trajectories, no matter what the network parameters. The latter observation in particular precludes operation on the edge of chaos. How, then, were model networks in other studies able to achieve long temporal memory (Bertschinger & Natschläger, 2004; Maass et al., 2005; Legenstein & Maass, 2007; Büsing et al., 2010)? The answer has to do with connectivity: in all studies that operated on the edge of chaos, the average number of connections per neuron, K, was small—never more than about 30. By contrast, in our networks, so far we have considered the large K regime (we explicitly considered the the large N regime, but in all of our analysis, K is proportional to N).

To understand why the size of K makes such a difference, we need to take a close look at the assumptions that led to the square root singularity in $f(d)$. The main, somewhat hidden, one was that the distribution of $h_i^{(1)} - h_i^{(2)}$ is gaussian and so yielded the distribution shown in Figure 3b. Gaussianity is in fact guaranteed if the number of nonzero terms in equation 3.15 is large, which in turn is guaranteed in the limit $K \to \infty$. However, for finite K and sufficiently small d, Kd is small rather than large. In this regime, the sum over j in equation 3.15 is typically zero, but with probability Kd, one of the terms is nonzero. If there is a nonzero term, its typical size is $2\sigma_w/K^{1/2}$, the standard deviation of the connection strength if neurons are connected (see equation 3.2). This leads to a very different distribution of $h_i^{(1)} - h_i^{(2)}$ than the one shown in Figure 3b: there is a δ-function peak at zero with area $1 - Kd$, corresponding to neurons for which $h_i^{(1)} = h_i^{(2)}$; added to that is a gaussian distribution with total weight Kd and width $2\sigma_w/K^{1/2}$. This distribution is illustrated in Figure 8.

Although the distribution of $h_i^{(1)} - h_i^{(2)}$ is nongaussian when $Kd \ll 1$, it still has a gaussian component. And, in fact, it is only the gaussian component that contributes to $f(d)$; that is because the δ-function peak corresponds to neurons whose synaptic drives are equal on two trials. Thus, we can use essentially the same arguments as for the large Kd regime; the only difference is that the fraction of neurons that contribute to $f(d)$ is reduced by a factor of Kd. This means $f(d)$ is proportional to $KdP(h+u=0)\Delta h(d)$, where now $\Delta h(d)$ is is the typical difference between $h_i^{(1)}$ and $h_i^{(2)}$ given that $h_i^{(1)} \neq h_i^{(2)}$ (and, of course, trajectories are a distance d apart). As just discussed and as can be seen in Figure 8, this distance is $2\sigma_w/K^{1/2}$. Putting this all together and using equation 3.8 for $P(h+u)$, we see that in the small Kd limit,

$$f(d) \propto 2K^{1/2}\sigma_w \frac{e^{-\overline{u}^2/2\sigma_{\text{tot}}^2}}{(2\pi\sigma_{\text{tot}}^2)^{1/2}}\, d. \tag{4.9}$$

See section A.2 of the appendix (especially equation A.18), for a more detailed derivation.

What this analysis tells us is that when $Kd \ll 1$, $f(d)$ is proportional to d, not $d^{1/2}$. Thus, if we tune the parameters so that $f(d) = d$, the network is at the edge of chaos, and nearby points in phase-space stay roughly the same distance apart as the network evolves. That such a regime exists has been known for some time (Bertschinger & Natschläger, 2004; Maass et al., 2005; Legenstein & Maass, 2007; Büsing et al., 2010); the reason we missed it above (see in particular sections 4.1 and 4.2) is that we assumed $K \gg 1$ and d small but $\mathcal{O}(1)$, for which we can never have $Kd \ll 1$.

While the edge of chaos can be accessed when K is finite, it is not robust to a ubiquitous feature in cortex: synaptic failures (Del Castillo & Katz, 1954; Branco & Staras, 2009; Borst, 2010). That is because failures automatically induce a distance between trajectories. Even if the pattern of activity on time step t is exactly the same on two trials, on time step $t+1$, the synaptic drive to each neuron will be different, thus inducing an effective distance. To determine the effective distance, note that the fraction of active presynaptic neurons is the mean firing rate, $\overline{\nu}$, and the probability that there will be release from a single active presynaptic neuron on one trial and not another is $2p_f(1-p_f)$ (there is a factor of 2 because there could be a failure on either the first or second trial). Thus, the average effective distance is at least $2\overline{\nu}p_f(1-p_f)$. Using this distance, which we denote d_{eff}, the condition $Kd \ll 1$ translates to

$$Kd_{eff} = 2K\overline{\nu}p_f(1-p_f) \ll 1.$$

For this to be satisfied, we must have $\overline{\nu} \ll 1/(2Kp_f(1-p_f))$. Given that the average failure rate in the brain is on the order of 0.5 (Branco & Staras, 2009; Borst, 2010), and using $K = 5000$ (Braitenberg & Schüz, 1991), this implies that we must have $\overline{\nu} \ll 0.0004$. With a time step of 10 ms, this corresponds to a mean firing rate less than 0.04 Hz, much smaller than is observed in the brain. Thus, synaptic failures preclude operation on the edge of chaos.

5 Discussion

Using a simple network model consisting of randomly connected McCullough-Pitts neurons, we established a link between chaotic dynamics and temporal memory: in the chaotic regime, temporal memory scales as the log of the network size times the characteristic timescales in the network. Because of the $\log N$ scaling, increasing the size of the network has very little effect on memory lifetime. For example, increasing the network from 10^5 to 10^{11} neurons, about the increase one sees in humans over fruit flies, results in an improvement in memory lifetime by a factor of only about two.

Although we used a simple model (chosen primarily for its analytical tractability), our conclusions are consistent with theoretical studies. For example, in simulations of integrate-and-fire neurons, temporal memory is on the same order as the time constants of the synapses (Maass et al., 2002); both are about 1 second. It is also likely that our conclusions apply to cortical networks, for which connectivity is high. In essence, the link between chaos and short temporal memory relies on the fact that trajectories approach chaotic attractors exponentially rapidly. Exponential approach to attractors is typical (Ashwin, Buescu, & Stewart, 1996), so the only question is: Is chaos typical? There is now strong theoretical (van Vreeswijk & Sompolinsky, 1996, 1998; Banerjee, 2006; Izhikevich & Edelman, 2008; Monteforte & Wolf, 2010) and experimental (London et al., 2010) evidence that it is for the kinds of high-connectivity networks found in the brain. Our results effectively rule out the possibility that randomly connected cortical networks can exhibit long temporal memory at the level of spike times.

There are, of course, several caveats to this study. The first is that nonrandom connectivity could lead to long temporal memory at the level of spike times, as in synfire chains or polychronization (Diesmann, Gewaltig, & Aertsen, 1999; Izhikevich, 2006). Although at this time there is little experimental evidence for either of these in cortex, their existence cannot be ruled out. The second is that it may be possible to operate at the edge of chaos at the level of firing rates rather than spike times (Sussillo & Abbott, 2009; Rajan, Abbott, & Sompolinsky, 2010), thus leading to long temporal memory using a rate-based neural code. Importantly, a recent study showed that spiking networks can exhibit complex, if not necessarily chaotic, behavior at the level of firing rates (Litwin-Kumar & Doiron, 2012), exactly what is needed for liquid state machines. Third, our results should not be interpreted to mean that the liquid state machine operating at the level of spike times, as proposed originally by Maass and colleagues (Maass et al., 2002) is not useful; it could still perform complex computations, just not with long temporal memory.

Appendix: Rate Distribution, Distance Evolution Function, and Linear Classifier

Here we compute various mean field quantities used in the main text. Our primary approximation in all of these computations is that sums over index (e.g., the sum over j on the right-hand side of equation 3.1b) can be treated as gaussian random variables. This has two important ramifications. The first is that for such sums, all we need is the mean and variance, which greatly simplifies our calculations. The second is that sums over indices turn into averages over gaussian distributions. It is not entirely obvious that this is a good approximation, especially since we are considering deterministic equations. However, there are two sources of randomness. One is initial conditions; the other is chaotic dynamics, which amplifies small differences in initial conditions. We assume, without proof, that these two sources of randomness are sufficient to justify this assumption. Comparison of theoretical predictions with simulations will be our litmus test.

A.1 Firing Rate Distribution. Primarily to provide a stringent test of our analysis, here we compute the distribution of firing rates. Our starting point is an expression for the firing rate of the ith neuron, denoted ν_i. Using the fact that ν_i is the probability, over time, that $h_i(t) + u_i(t)$ is positive, we have

$$\nu_i = \langle \Theta(h_i(t) + u_i(t)) \rangle_t \tag{A.1}$$

where $h_i(t)$ is given by equation 3.1b, the distribution of $u_i(t)$ is given by equation 3.3, the subscript t on the angle bracket represents an average over time, and, recall, Θ is the Heaviside step function.

The term inside the Heaviside step function, $h_i(t) + u_i(t)$, can be broken into time-independent and time-dependent pieces,

$$h_i(t) + u_i(t) = \overline{u} + \alpha\eta_i + \beta\xi_i(t),$$

where

$$\alpha\eta_i \equiv \sum_j w_{ij} \langle x_j(t) \rangle_t, \tag{A.2a}$$

$$\beta\xi_i(t) \equiv \sum_j w_{ij}(x_j(t) - \langle x_j(t) \rangle_t) + u_i(t) - \overline{u}. \tag{A.2b}$$

As just discussed, we assume that sums over indices can be treated as gaussian random variables. Using also the fact that $u_i(t)$ is gaussian (see equation 3.3), it follows that both η_i and $\xi_i(t)$ are gaussian random variables. Since they are zero mean by construction, we can, without loss of generality, let them have unit variance. Consequently, the variances on the right-hand sides of equations A.2a and A.2b are α^2 and β^2, respectively. Once we know α and β, computing the distribution of firing rates is, as we show below (see in particular equations A.6 to A.9), straightforward.

We first compute the variance of the expression on the right-hand side of equation A.2a. Using exactly the same techniques we used to derive an expression for the mean firing rate (see in particular equations 3.6 and 3.7), we find, after a small amount of algebra, that

$$\alpha^2 = \sigma_w^2 \sigma_x^2, \tag{A.3}$$

where

$$\sigma_x^2 \equiv \frac{1}{N} \sum_j \langle x_j(t) \rangle_t^2. \tag{A.4}$$

We next turn to equation A.2b, which is slightly more complicated. Assuming the terms in the sum over j are independent and using the fact that $x_j^2 = 1$, we have

$$\text{Var}\left[\sum_j w_{ij} (x_j(t) - \langle x_j(t) \rangle_t) \right] = \sum_j w_{ij}^2 (1 - \langle x_j(t) \rangle_t^2).$$

All terms on the right-hand side are nonnegative, so the sum self-averages, meaning it is independent of i. We can therefore average it over i. This yields a factor of σ_w^2/N in place of w_{ij}^2 (see equation 3.2), and we are left with an average over $1 - \langle x_j(t) \rangle_t^2$. This average is, via equation A.4, just $(1 - \sigma_x^2)$. Combining this with equation 3.3, which tells us that the variance of $u_i(t) - \overline{u}$ is σ_u^2, and using the fact that $\xi_i(t)$ has unit variance, we have

$$\beta^2 = \sigma_w^2 (1 - \sigma_x^2) + \sigma_u^2. \tag{A.5}$$

We can now compute ν_i by turning the time average in equation A.1 into an average over the gaussian random variables, $\xi_i(t)$. This gives us

$$\nu_i = \int \frac{d\xi e^{-\xi^2/2}}{(2\pi)^{1/2}} \Theta(\overline{u} + \alpha \eta_i + \beta \xi) = \Phi((\overline{u} + \alpha \eta_i)/\beta), \tag{A.6}$$

where

$$\Phi(z) \equiv \int_{-\infty}^{z} dy \, \frac{\exp(-y^2/2)}{(2\pi)^{1/2}} \tag{A.7}$$

is the cumulative normal function.

Because we are interested only in the probability distribution of the firing rate, we do not need to keep track of which neuron has which firing rate. We may therefore interpret equation A.6 to mean

$$\nu(\eta) = \Phi((\overline{u} + \alpha\eta)/\beta), \tag{A.8}$$

where, recall, η is a zero mean, unit variance gaussian random variable. To find $p(\nu)$, we use the usual relationship for transforming probability distributions, $p(\nu) = p(\eta)\,|d\eta/d\nu|$. Since the derivative of the cumulative normal function is just a gaussian, we arrive at the expression given in equation 3.11.

Our remaining task is to compute σ_x^2, which must be done self-consistently. Using equation A.4 for the definition of σ_x^2, replacing $\langle x_i(t)\rangle$ with $2\nu_i - 1$, and turning averages over index into averages with respect to $p(\nu)$, we see that

$$\sigma_x^2 = (2\overline{\nu} - 1)^2 + 4\mathrm{Var}[\nu],$$

where $\overline{\nu}$ is given by equation 3.10 and

$$\mathrm{Var}[\nu] = \int d\nu\, p(\nu)\big(\nu - \overline{\nu}\big)^2.$$

Finally, using the fact that $p(\nu)d\nu = p(\eta)d\eta$ to turn the integral over ν into an integral over η, and using equation A.8 for $\nu(\eta)$, we arrive at

$$\sigma_x^2 = (2\overline{\nu} - 1)^2 + 4\int \frac{d\eta\, e^{-\eta^2/2}}{(2\pi)^{1/2}} (\Phi((\overline{u} + \alpha\eta)/\beta) - \overline{\nu})^2. \tag{A.9}$$

Because α and β, which appear on the right-hand side, depend on σ_x^2 (via equations A.3 and A.5, respectively), equation A.9 must be solved self-consistently. This can be done efficiently by iterating that equation.

A.2 The Distance Evolution Function. Here we compute the distance evolution function, $f(d)$, which tells us the time evolution of the distance between trajectories that have different initial conditions but receive the same input. This function is given by the right-hand side of equation 3.13. Using our approximation that sums over indices can be turned into averages over distributions, this equation becomes

$$f(d) = \frac{1}{2}\int du\, P(u) \int dh^{(1)} dh^{(2)}\, P(h^{(1)}, h^{(2)}) \times |\mathrm{sgn}(h^{(1)} + u) - \mathrm{sgn}(h^{(2)} + u)|_d, \tag{A.10}$$

where, as in the main text, the subscript d indicates that the two trajectories are a distance d apart. It is convenient to make the change of variables

$$a = \frac{h^{(1)} + h^{(2)}}{2} + u, \tag{A.11a}$$

$$b = \frac{h^{(1)} - h^{(2)}}{2}. \tag{A.11b}$$

It is easy to see that a and b are uncorrelated, which follows because $\langle ab \rangle \propto \mathrm{Var}[h^{(1)}] - \mathrm{Var}[h^{(2)}] = 0$ (the second equality follows from the symmetry between $h^{(1)}$ and $h^{(2)}$). We assume also that they are independent. With this assumption, we find, after a small amount of algebra to determine the limits of integration in the new variables, that equation A.10 becomes

$$f(d) = \int_{-\infty}^{\infty} du\, P(u) \int_{-\infty}^{\infty} db\, P(b) \int_{-|b|}^{|b|} da\, P(a). \tag{A.12}$$

Because a consists of the sum of a large number of random variables, $P(a)$ is gaussian. Its variance is given by

$$\mathrm{Var}[a] = \frac{\mathrm{Var}[h^{(1)}] + 2\mathrm{Covar}[h^{(1)}, h^{(2)}|d] + \mathrm{Var}[h^{(1)}]}{4}. \tag{A.13}$$

The variances we computed in equation 3.7; they are equal to σ_w^2. The covariance can be computed in the same way,

$$\mathrm{Covar}[h^{(1)}, h^{(2)}] = \frac{1}{N} \sum_i \sum_j w_{ij} x_j^{(1)} \sum_{j'} w_{ij'} x_{j'}^{(2)} = \frac{1}{N} \sum_{ijj'} w_{ij} w_{ij'} x_j^{(1)} x_{j'}^{(2)}.$$

As usual, because the weights are i.i.d. with mean zero and variance σ_w^2/N, in the large N limit the average over i (which involves only the weights) is $\delta_{jj'}\sigma_w^2/N$. Consequently,

$$\mathrm{Covar}[h^{(1)}, h^{(2)}|d] = \frac{\sigma_w^2}{N} \sum_j x_j^{(1)} x_j^{(2)} = 1 \times (1-d) + (-1) \times d = \sigma_w^2(1-2d).$$

Inserting this into equation A.13 yields $\mathrm{Var}[a] = \sigma_w^2(1-d)$. The mean of a is u, and so we can write

$$f(d) = \int_{-\infty}^{\infty} db\, P(b) \int_{-\infty}^{\infty} du\, \frac{e^{-(u-\overline{u})^2/2\sigma_u^2}}{(2\pi\sigma_u^2)^{1/2}} \int_{-|b|}^{|b|} da\, \frac{e^{-(a-u)^2/2\sigma_w^2(1-d)}}{(2\pi\sigma_w^2(1-d))^{1/2}},$$

where we used equation 3.3 for $P(u)$ and exchanged the order of the u and b integrations.

The integrals over u and a are straightforward, and we have

$$f(d) = \int_{-\infty}^{\infty} db\, P(b) \left[\Phi\left(\frac{\overline{u} + |b|}{\sigma_d} \right) - \Phi\left(\frac{\overline{u} - |b|}{\sigma_d} \right) \right], \tag{A.14}$$

where

$$\sigma_d^2 \equiv \sigma_w^2 (1-d) + \sigma_u^2 = \sigma_{\text{tot}}^2 - d\sigma_w^2. \tag{A.15}$$

The second equality in equation A.15 follows from equation 3.9.

To compute the right-hand side of equation A.14, we need an explicit expression for $P(b)$. For that, we restore indices on b and write

$$b_i = \frac{1}{2} \sum_j w_{ij} \big(x_j^{(1)} - x_j^{(2)}\big). \tag{A.16}$$

The probability distribution over b, $P(b)$, is a probability distribution with respect to index, i. Normally when we have a sum over indices, we assume that it has a gaussian distribution. For that, however, we need a large number of terms in the sum to be nonzero. Here this is not necessarily the case, since the number of nonzero terms on the right-hand side is approximately Kd. (That follows because the probability that $x_j^{(1)} - x_j^{(2)} \neq 0$ is d, the probability that w_{ij} is nonzero is K/N, and there are N terms in the sum; multiplying these together yields Kd.) We generally assume that $Kd \gg 1$. That is typically true when K is large, but we note that $d = 1/N$, the smallest possible value; then $Kd = K/N$, which need not be large. We thus consider two regimes: $Kd \gg 1$ and $Kd \ll 1$. We start with the former.

The mean of b is clearly zero (independent of the size of Kd). And since we are assuming $Kd \gg 1$, b is gaussian, and we can use our usual approach to finding the variance,

$$\begin{aligned}\text{Var}[b] &= \frac{1}{4N} \sum_{ijj'} w_{ij} w_{ij'} \big(x_j^{(1)} - x_j^{(2)}\big)\big(x_{j'}^{(1)} - x_{j'}^{(2)}\big) \\ &= \frac{\sigma_w^2}{4N} \sum_j \big(x_j^{(1)} - x_j^{(2)}\big)^2 = \sigma_w^2 d.\end{aligned}$$

Thus, when $Kd \gg 1$,

$$P(b) = \frac{e^{-b^2/2\sigma_w^2 d}}{(2\pi\sigma_w^2 d)^{1/2}}.$$

Inserting this expression for $P(b)$ into equation A.14 and noting that both $P(b)$ and the term in brackets in equation A.14 are symmetric around $b = 0$, we arrive at

$$f(d) = 2\int_0^\infty db\, \frac{\exp[-b^2/2\sigma_w^2 d]}{(2\pi\sigma_w^2 d)^{1/2}} \left[\Phi\left(\frac{\overline{u}+b}{\sigma_d}\right) - \Phi\left(\frac{\overline{u}-b}{\sigma_d}\right)\right]. \quad \text{(A.17)}$$

In the small d limit, b is small; we can Taylor-expand the term in brackets around $b = 0$. Working to first order in b and using that fact that in the small d limit $\sigma_d^2 \approx \sigma_{\text{tot}}^2$ (see equation A.15) equation A.17 reduces to equation 3.18.

We now consider the second limit, $Kd \ll 1$. In this limit, with probability $1 - Kd$ (plus $\mathcal{O}((Kd)^2)$ corrections), all the terms on the right-hand side of equation A.16 are zero. Thus, the dominant component of $P(b)$ is $(1 - Kd)\delta(b)$. The next most likely case is exactly one term nonzero, for which $x^{(1)} - x^{(2)} = \pm 2$. This happens with probability Kd (again with $\mathcal{O}((Kd)^2)$ corrections). Because of the factor of $1/2$ in equation A.16, this means b_i is $+w_{ij}$ with probability $Kd/2$ and $-w_{ij}$ with the same probability. Combining these two components and using equation 3.2 for the distribution of nonzero w's, we have, in the small Kd limit,

$$p(b) = (1 - Kd)\delta(b) + Kd\, \frac{P_w^c(b) + P_w^c(-b)}{2}.$$

Inserting this into equation A.14, Taylor-expanding the term in brackets around $b = 0$ (which is valid since $P_w^c(w)$ has a standard deviation of $\sigma_w/K^{1/2}$ and $K \gg 1$), and replacing σ_d^2 with σ_{tot}^2, we find that

$$f(d) = \left[\frac{2}{\sigma_w/K^{1/2}} \int_{-\infty}^{\infty} db\, P_w^c(b)\, |b|\right] K^{1/2}\sigma_w \frac{e^{-\overline{u}^2/2\sigma_{\text{tot}}^2}}{(2\pi\sigma_{\text{tot}}^2)^{1/2}}\, d \quad \text{(A.18)}$$

with, of course, $\mathcal{O}(Kd)$ corrections. The divisive term, $\sigma_w/K^{1/2}$, inside the square brackets ensures that the term in brackets is $\mathcal{O}(1)$. Consequently, we recover equation 4.9, but with the constant of proportionality computed explicitly.

A.3 Scaling of the Linear Classifier with Distance, *d*. Here we determine how $Z(t)$, equation 4.5, scales with the distance between the two trajectories. Our starting point is to determine the scaling of $\mathbf{J}(t)$ and $\langle \mathbf{x}^{(1)}(t) - \mathbf{x}^{(2)}(t)\rangle$. That will tell us how the numerator scales and, as it turns out, will provide us enough information to write down the scaling of the denominator.

We assume, for simplicity, that we use R trials for both training and testing (which is what we do in our simulations). Consequently, for the numerator of equation 4.5, we have

$$\mathbf{J}(t) \cdot \langle \mathbf{x}^{(1)}(t) - \mathbf{x}^{(2)}(t) \rangle = \frac{1}{R^2} \sum_{r=1}^{R} \sum_{s=1}^{R} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \big(x_i^{(1)r}(t) - x_i^{(2)r}(t) \big) \big(x_i^{(1)s}(t) - x_i^{(2)s}(t) \big), \tag{A.19}$$

where we used equation 4.4 for $\mathbf{J}(t)$, and $x_i^{(k)r}$ is the value of x_i on the rth trial of trajectory k. We assume that σ_i^2 is, on average, independent of the product of the x_i's, which allows us to write

$$\frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \big(x_i^{(1)r}(t) - x_i^{(2)r}(t) \big) \big(x_i^{(1)s}(t) - x_i^{(2)s}(t) \big) \approx \frac{\overline{1/\sigma^2}}{N} \sum_{i=1}^{N} \big(x_i^{(1)r}(t) - x_i^{(2)r}(t) \big) \big(x_i^{(1)s}(t) - x_i^{(2)s}(t) \big), \tag{A.20}$$

where $\overline{1/\sigma^2} \equiv N^{-1} \sum_i 1/\sigma_i^2$ is the population average of $1/\sigma_i^2$. Explicitly multiplying the terms in parentheses yields sums over products of x_i's (e.g., $\sum_i x_i^{(1)r} x_i^{(1)s}$). Using the fact that by definition, $d^*(t) = P\big(x_i^{(1)r}(t) \neq x_i^{(1)s}(t)\big)$ and $d(t) = P\big(x_i^{(1)r}(t) \neq x_i^{(2)s}(t)\big)$, we see that those sums depend on only the distance between trajectories,

$$\frac{1}{N} \sum_{i=1}^{N} x_i^{(k)r}(t) x_i^{(k)s}(t) = 1 - 2d^*(t), \tag{A.21a}$$

$$\frac{1}{N} \sum_{i=1}^{N} x_i^{(k)r}(t) x_i^{(\bar{k})s}(t) = 1 - 2d(t), \tag{A.21b}$$

where $\bar{k}$ means "not k": $\bar{k} = 1$ if $k = 2$ and $\bar{k} = 2$ if $k = 1$. Inserting these expressions into equation A.20, we find that the right-hand side is equal to $4(d(t) - d^*(t))$ on average. Because this is independent of r and s, the sums over r and s in equation A.19 are trivial, and we arrive at

$$\mathbf{J}(t) \cdot \big\langle \mathbf{x}^{(1)}(t) - \mathbf{x}^{(2)}(t) \big\rangle \approx \overline{1/\sigma^2} \times 4(d(t) - d^*(t)). \tag{A.22}$$

This is as expected: the closer $d(t)$ is to $d^*(t)$, the harder it is to distinguish the two trajectories.

For the denominator in equation 4.5, we start with the explicit expression

$$\begin{aligned}&\mathrm{Var}\big[\mathbf{J}(t)\cdot\big(\mathbf{x}^{(1)}(t)-\mathbf{x}^{(2)}(t)\big)\big]\\&\quad=\mathbf{J}(t)\cdot\big\langle\big(\mathbf{x}^{(1)}(t)-\mathbf{x}^{(2)}(t)\big)\big(\mathbf{x}^{(1)}(t)-\mathbf{x}^{(2)}(t)\big)\big\rangle\cdot\mathbf{J}(t).\end{aligned}$$

Making the usual approximation that the neurons are independent, the term in angle brackets is a diagonal matrix with the ith diagonal element given by σ_i^2. Thus, using equation 4.4 for $\mathbf{J}(t)$, we have

$$\mathrm{Var}\big[\mathbf{J}(t)\cdot\big(\mathbf{x}^{(1)}(t)-\mathbf{x}^{(2)}(t)\big)\big]=\frac{1}{N^2}\sum_i\frac{\big\langle\big(x_i^{(1)}-x_i^{(2)}\big)^2\big\rangle}{\sigma_i^2}. \tag{A.23}$$

Again using the fact that angle brackets correspond to an average over trials, this expression is almost the same as equation A.20. However, there are two differences. One is that there is an extra factor of $1/N$. The other is that the x_i are evaluated on the same set of trials. So when $r=s$, $x_i^{(1)r}$ is identical to $x_i^{(1)s}$, and similarly for $x_i^{(2)}$. We thus need to consider $r=s$ and $r\neq s$ separately. For $r\neq s$, we get the same answer as in equation A.22 (except, of course, for the extra factor of $1/N$). Since there are $R(R-1)$ such terms, equation A.23 becomes

$$\begin{aligned}&\mathrm{Var}\big[\mathbf{J}(t)\cdot\big(\mathbf{x}^{(1)}(t)-\mathbf{x}^{(2)}(t)\big)\big]\\&\quad\approx\frac{\overline{1/\sigma^2}}{N}\left[4(1-1/R)(d(t)-d^*(t))+\frac{1}{R^2}\sum_r\frac{1}{N}\sum_i\big(x_i^{(1)r}-x_i^{(2)r}\big)^2\right].\end{aligned}$$

The second term inside the brackets is just $4d(t)/R$, and so

$$\mathrm{Var}\big[\mathbf{J}(t)\cdot\big(\mathbf{x}^{(1)}(t)-\mathbf{x}^{(2)}(t)\big)\big]=\frac{\overline{4/\sigma^2}}{N}[d(t)-d^*(t)+d^*(t)/R].$$

We can now compute $Z(t)$: combining equation A.22 with equation A.23 and using the definition of $Z(t)$, equation 4.5, we have

$$Z(t)=\left[\frac{\overline{2/\sigma^2}\,N(d(t)-d^*(t))}{1+d^*(t)/[R(d(t)-d^*(t))]}\right]^{1/2}. \tag{A.24}$$

Replacing $d(t)-d^*(t)$ by $(d^*/c_0)e^{-\lambda t}$, and defining $c_1\equiv\big(\overline{2/\sigma^2}\,d^*/c_0\big)^{1/2}$, where c_0 and c_1 are $\mathcal{O}(1)$ quantities that depends on network parameters, we arrive at equation 4.7.

Acknowledgments

We thank Lars Büsing for insightful discussions and comments on the manuscript. This work was supported by the Gatsby Charitable Foundation.

References

Ashwin, P., Buescu, J., & Stewart, I. (1996). From attractor to chaotic saddle: A tale of transverse instability. *Nonlinearity*, *9*, 703–737.

Banerjee, A. (2006). On the sensitive dependence on initial conditions of the dynamics of networks of spiking neurons. *J. Comput. Neurosci.*, *20*, 321–348.

Bertschinger, N., & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Comput.*, *16*, 1413–1436.

Borst, J. (2010). The low synaptic release probability in vivo. *Trends Neurosci.*, *33*, 259–266.

Braitenberg, V., & Schüz, A. (1991). Anatomy of the cortex. Berlin: Springer-Verlag.

Branco, T., & Staras, K. (2009). The probability of neurotransmitter release: Variability and feedback control at single synapses. *Nat. Rev. Neurosci.*, *10*, 373–383.

Brody, C., Romo, R., & Kepecs, A. (2003). Basic mechanisms for graded persistent activity: Discrete attractors, continuous attractors, and dynamic representations. *Curr. Opin. Neurobiol.*, *13*, 204–211.

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

Büsing, L., Schrauwen, B., & Legenstein, R. (2010). Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural Comput.*, *22*, 1272–1311.

Del Castillo, J., & Katz, B. (1954). Quantal components of the end-plate potential. *J. Physiol. (London)*, *124*, 560–573.

Diesmann, M., Gewaltig, M., & Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature*, *402*, 529–533.

Ganguli, S., Huh, D., & Sompolinsky, H. (2008). Memory traces in dynamical systems. *Proc. Natl. Acad. Sci. USA*, *105*, 18970–18975.

Izhikevich, E. (2006). Polychronization: Computation with spikes. *Neural Comput*, *18*, 245–282.

Izhikevich, E., & Edelman, G. (2008). Large-scale model of mammalian thalamocortical systems. *Proc. Natl. Acad. Sci. USA*, *105*, 3593–3598.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. *Biol. Cybern.*, *81*, 211–225.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, *304*, 78–80.

Kauffman, S., & Johnsen, S. (1991). Co-evolution to the edge of chaos: Coupled fitness landscapes, poised states, and co-evolutionary avalanches. *J. Theor. Biol.*, *149*, 467–505.

Koulakov, A., Raghavachari, S., Kepecs, A., & Lisman, J. (2002). Model for a robust neural integrator. *Nature Neurosci.*, *5*, 775–782.

Langton, C. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, *42*, 12–37.

Legenstein, R., & Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, *20*, 323–334.

Litwin-Kumar, A., & Doiron, B. (2012). Slow dynamics and high variability in balanced cortical networks with clustered connections. *Nat. Neurosci.*, *15*(11), 1498–1505.

London, M., Roth, A., Beeren, L., Häusser, M., & Latham, P. E. (2010). Sensitivity to perturbations in vivo implies high noise and suggests rate coding in cortex. *Nature*, *466*, 123–127.

Maass, W., Legenstein, R., & Bertschinger, N. (2005). Methods for estimating the computational power and generalization capability of neural microcircuits. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems, 17* (pp. 865–872). Cambridge, MA: MIT Press.

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

Monteforte, M., & Wolf, F. (2010). Dynamical entropy production in spiking neuron networks in the balanced state. *Phys. Rev. Lett.*, *105*, 268104.

Rajan, K., Abbott, L., & Sompolinsky, H. (2010). Stimulus-dependent suppression of chaos in recurrent neural networks. *Phys Rev. E Stat Nonlin. Soft Matter Phys.*, *82*, 011903.

Seung, H. (1996). How the brain keeps the eyes still. *Proc. Natl. Acad. Sci. USA*, *93*, 13339–13344.

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

van Vreeswijk, C., & Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, *274*, 1724–1726.

van Vreeswijk, C., & Sompolinsky, H. (1998). Chaotic balanced state in a model of cortical circuits. *Neural Comput.*, *10*, 1321–1371.

Received November 3, 2012; accepted December 16, 2012.