Memory in reservoirs for high dimensional input

Michiel Hermans and Benjamin Schrauwen

***Abstract*— Reservoir Computing (RC) is a recently introduced scheme to employ recurrent neural networks while circumventing the difficulties that typically appear when training the recurrent weights. The 'reservoir' is a fixed randomly initiated recurrent network which receives input via a random mapping. Only an instantaneous linear mapping from the network to the output is trained which can be done with linear regression. In this paper we study dynamical properties of reservoirs receiving a high number of inputs. More specifically, we investigate how the internal state of the network retains fading memory of its input signal. Memory properties for random recurrent networks have been thoroughly examined in past research, but only for one-dimensional input. Here we take into account statistics which will typically occur in high dimensional signals. We find useful empirical data which expresses how memory in recurrent networks is distributed over the individual principal components of the input.**

I. Introduction

A. Memory in recurrent networks

A significant body of research in machine learning focuses on recurrent neural networks. These networks have for instance been studied for their ability to store patterns (the so-called Hopfield networks [1]). More recently however, recurrent networks are being used to process temporal information; due to internal feedback, these networks have an intrinsic ability to retain information about past input for a certain time, which allows for the processing of signals that are explicitly coded in time. This property is often called 'fading memory', 'short term memory' or the 'echo state property'. Though training algorithms for recurrent neural networks exist which can be quite successful (most notably *backpropagation through time* [2]), some problems like slow convergence and limited applicability due to high computational costs remain [3].

An alternative approach is generally known as *Reservoir Computing* (RC), discovered independently by Jaeger [4] for analog hyperbolic tangent neurons, and by Maass [5] for spiking neurons, where the networks are called 'Echo State Networks' and 'Liquid State Machines' respectively. In these systems, the "reservoir" is a randomly initiated recurrent neural network with fixed interconnection weights, excited by the input which needs to be processed. The readout mechanism consists of a single layer of linear nodes observing the reservoir nodes, which are usually trained using linear regression. This approach is very fast and easy to implement, and does not suffer from common problems found in other training algorithms such as fading error gradients or slow convergence.

ELIS at Ghent University, Sint Pietersnieuwstraat 41, 9000 Ghent, Belgium (phone: +32 9 264 3368; email: michiel.hermans@ugent.be, benjamin.schrauwen@ugent.be).

As a rule of thumb, good performance requires the number of network nodes to be significantly higher than the number of input channels. Intuitively this is due to the fact that the input signal (which can be considered to exist in feature space) needs to have space in which to be nonlinearly expanded to form a representation (which can be considered to exist in 'reservoir space') which retains both temporal and spatial information on the input signal. For many tasks however, this will quickly prove difficult due to the high-dimensional nature of input data, e.g. sound (preprocessed speech [6], echolocation [7]) and image processing (handwriting recognition [8], automated surveillance cameras etc.) and one commonly resorts to dimensionality reduction techniques.

In this paper we will research memory properties for these kinds of input signals. Past research has focused on one-dimensional signals to quite some detail [9], [10], [11], [12]. The main conclusions found in this area of study are the following.

- Memory capacity (which will be formally defined in Section I-C) of a neural network cannot exceed the number of neurons in the network [9], [11].
- A saturating nonlinearity has necessarily a detrimental effect on memory [9].
- Noise robustness depends on the eigenvalue spectrum of the linearized system [10], [12].
- Noise robustness is best when the spectral radius of the linearized system is equal to or close to one [11], [12].

Before elaborating on the main bulk of our research we shall discuss the network model, then we introduce the measures commonly used to quantify memory in RC and the way we use them.

B. Network model

We shall work with standard discrete time hyperbolic tangent networks which evolve according to

$$\mathbf{a}(k+1) = \tanh\left(\mathbf{W}\mathbf{a}(k) + \mathbf{V}\mathbf{s}(k)\right), \qquad (1)$$

where $\mathbf{a}(k)$ and $\mathbf{s}(k)$ are respectively the network state and the input signal at the k-th time step, and $\mathbf{W}$ and $\mathbf{V}$ are the connection matrices for respectively the internal connections and the connections from the input to the reservoir.

Input weights and internal connection weights are initially drawn from a normal distribution with unit standard deviation and zero mean. $\mathbf{W}$ is then scaled such that its spectral radius equals ρ.

Usually, $\mathbf{V}$ is scaled with an input scaling factor which does not depend on the input signal dimensionality or energy. We would however want a useful scaling factor for these connections like the spectral radius for $\mathbf{W}$. More specifically, we wish that this scaling factor expresses the degree of

nonlinearity in the reservoir. A measure that we use to express this is the mean standard deviation of the reservoir states, as this quantifies how strongly the nonlinearity acts upon them. It is not easy to come up with a general estimator for the mean standard deviation of the reservoir states since this will depend on many factors such as the spatiotemporal correlation of the input signal and the exact connection matrix. However, it is nevertheless possible to find a relatively simple scaling factor that approximately reaches this goal. We derive and empirically test this measure in Appendix IV-B. We shall denote this scaling factor ϕ and it has the physical meaning that it roughly expresses the mean standard deviation for linearized networks.

C. Measures of memory

In order to study the temporal processing ability of recurrent networks, one needs to know how much information on past input is linearly coded in the immediate spatial state of the system. This property is represented by the *memory function* (MF) [11], which measures the ability to linearly reconstruct the input signal from a past time τ from the immediate state of the system. In the one-dimensional case, the input consist of a signal $s(i)$, $i \in \mathbb{N}$ that takes on a random value from a normal distribution at each time step. The MF is then defined as

$$m(k) = \frac{\langle s(i-k)u_k(i)\rangle_i^2}{\sigma^2(s)\sigma^2(u_k)}, \tag{2}$$

in which $\langle\cdots\rangle_i$ is the mean over all values i, and $u_k(i) = \mathbf{y}(k)^\mathsf{T}\mathbf{a}(i)$, the output trained to reconstruct the signal from k time steps ago, with $\mathbf{y}(k)$ optimized output weights and $\mathbf{a}(i)$ the reservoir state at the i-th time step, and σ denotes standard deviation. The MF is strictly between zero and one. Typically it is close to one for $k < N$ (corresponding to good memory) and drops to zero when $k > N$, with N the number of neurons in the network. Memory capacity (MC), denoted by M, is then simply the area under the memory function: $M = \sum_{k=0}^{\infty} m(k)$.

Both definitions can be readily expanded to a multi-dimensional input signal, simply by applying it on the individual input channels. The total memory capacity is then the sum of the memory capacities of the individual signals. To avoid an artificial overestimation of the total memory capacity we will need to make sure that the input channels on which we define the memory function are uncorrelated. One could for instance define two, highly correlated input signals and measure their respective memory capacities. As the two signals are almost identical, both memory capacities will be close to N, giving a total memory capacity exceeding the number of neurons in the network. For this reason, we define the MF and MC on the principal components of the input signal, which are uncorrelated by definition. If $\mathbf{s}(i)$ represents the vector with the input at time step i, we denote the principal components as $\tilde{\mathbf{s}}(i) = \mathbf{Q}\mathbf{s}(i)$, where $\mathbf{Q}$ is the matrix with the eigenvectors of the covariance matrix of $\mathbf{s}$ on its rows. The principal components also have so called 'energies' (their individual variances), which are equal to the eigenvalues of the covariance matrix of $\mathbf{s}$. We shall denote these by γ_n. Finally, we can define the MF and MC for the n-th principal component:

$$m_n(k) = \frac{\langle \tilde{s}_n(i-k)u_{k,n}(i)\rangle_i^2}{\gamma_n\sigma^2(u_{k,n})} \tag{3}$$

$$M_n = \sum_{k=0}^{\infty} m_n(k) \tag{4}$$

$$M = \sum_{n=1}^{N_{in}} M_n, \tag{5}$$

where $u_{k,n}$ is the optimal reconstruction of $\tilde{s}_n$. If one calculates the expression of the optimal readout weights, one can rewrite[1] equation 3 to a more useful expression:

$$m_n(k) = \frac{\mathbf{x}_n^\mathsf{T}(k)\mathbf{A}^{-1}\mathbf{x}_n(k)}{\gamma_n}, \tag{6}$$

where $\mathbf{x}_n(k) = \langle \mathbf{a}(i)\tilde{s}_n(i-k)\rangle_i$ and $\mathbf{A} = \left\langle \mathbf{a}(i)\mathbf{a}^\mathsf{T}(i)\right\rangle_i$. This allows us to calculate the MF without having to explicitly train readout weights.

Measuring the MC has one small but important difficulty. Due to the finite number of samples and the fact that the memory function is strictly positive, there will exist a positive bias in the measured MF. This will generally be small but it is nevertheless important when one measures M since this error is then multiplied with the width of the measured MF and the number of input channels. In the appendix we derive that the bias for the MF is approximately equal to $\frac{N}{T}$, with T the number of samples. We shall subtract this bias when calculating the memory capacities of the individual channels and the total MC.

In this paper, we shall always use 10^5 samples for measuring the MF. Results shown are always averaged over 10 reservoirs, but apparently only small differences between individual networks exist for the MF and MC.

II. Uncorrelated input

A. Total memory capacity

The first situation we consider is when all the principal components of the input channels have the same energy. We look at the total memory capacity M as a function of the spectral radius and the number of input channels. As input signals, we used white noise with unit standard deviation and zero mean and we used reservoirs with 100 nodes. Figure 1 shows the results for three different input scaling factors. In the quasilinear regime (left window), M is mostly equal to the number of reservoir nodes, virtually independent of the spectral radius or number of input channels. Only when both N_{in} and the spectral radius are small, M drops significantly below N. This is due to the fact that the spectral radius of a reservoir will determine the speed at which its transient dynamics fade. If the number of input channels is low, the MFs of the principal components can extend far to the past. However, if the transients are quenched too quickly,

[1] see for instance [11]

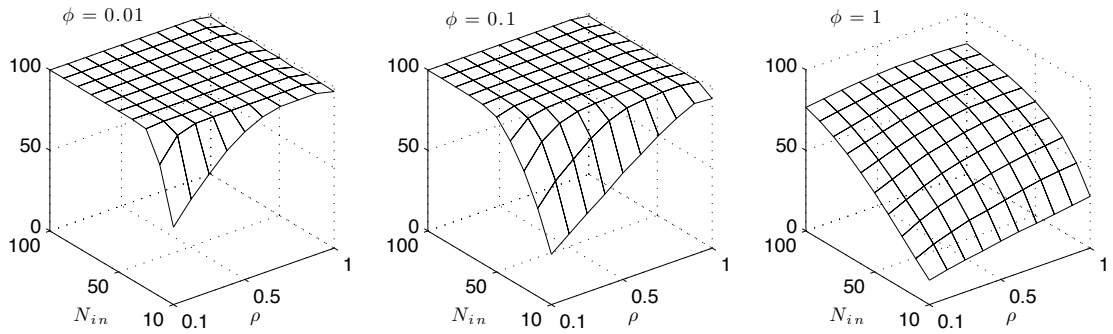

Fig. 1. Total MC M for different input scaling factors. On the left the reservoir is in the quasilinear regime, in the middle moderately nonlinear and on the right highly nonlinear.

recovering the past input from the current states becomes more difficult, resulting in lower M. When the number of input channels increases, this effect becomes less severe since the memory of the principal components will extend less far in the past (see next paragraph).
Increasing the input scaling factor aggravates the memory deterioration described above. This is due to the fact that the reservoir states are pushed into the saturating parts of their activation function, which decreases the 'effective' spectral radius (the spectral radius of the Jacobian [13], [14]).

B. Shape of the memory function

To get a good idea of exactly what the reservoir remembers of the input, it is useful to take a look at the MF itself. Figure 2 shows the memory functions (averaged over all input channels) in different situations. One obvious fact is clearly that - as N_{in} increases - the amount of memory capacity available for each individual input channel becomes smaller. The next interesting fact is that the spectral radius will greatly determine the shape of the MF. If ρ is small, the reservoir will have a good memory of only a few steps back in the past, and if it is close to one, the reservoir memory extends further but is less precise. This means that tasks with high input dimensionality which need short but precise memory may in fact benefit from choosing a small spectral radius.

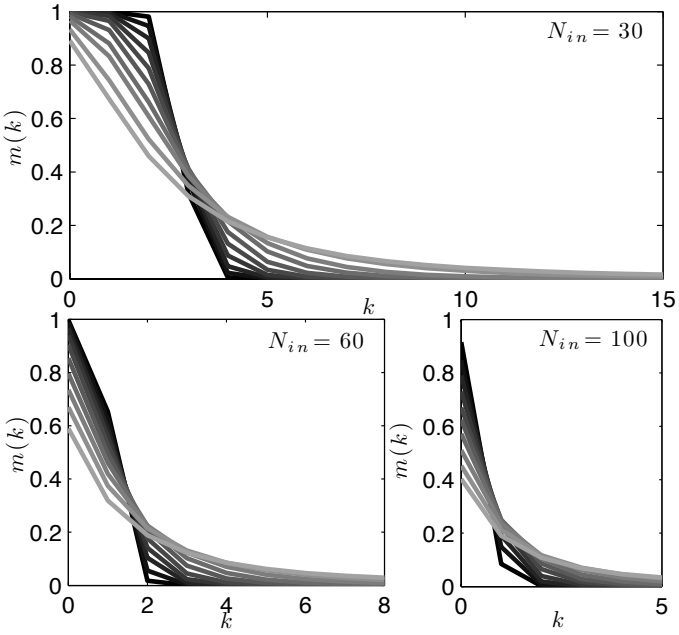

Fig. 2. Average memory functions for different N_{in}. The spectral radius varies from 0.1 (black line) to 1 (light gray line). Horizontal axis have the same scale in each window. The input scaling ϕ is equal to 0.01. Results are averaged over 10 runs.

III. Generalized signals

Typically, high dimensional input data will consist of principal components with widely varying energies. Usually, only a few principal components contain the bulk of the variance and a large fraction of principal components have low energies and contain less meaningful features or noise. In this section we will consider the impact of the energy contained in each component on its memory capacity. We use a reservoir with $N = 100$, $N_{in} = 50$, and $\phi = 0.1$. The input signal consists of 50 white noise channels with variance $\gamma_i = x^i$, where we choose $x = 0.8$. This allows us to measure memory capacity in function of signal energy for about 5 orders of magnitude. Figure 3 shows the results measured for three reservoirs with different spectral radii. When ρ is close to zero, all principal components are given roughly the same amount of memory by the reservoir. This makes sense if one considers the extreme case where $\rho = 0$, and thus $\mathbf{W} = \mathbf{0}$. Then, the 'reservoir' just acts as a random linear mapping, followed by a static nonlinear transformation. If we disregard the hyperbolic tangent, each separate input channel then can be recovered by an inverse linear mapping.
If the spectral radius increases, more memory is given to the channels with higher energy. When $\rho = 1$, M_i is roughly proportional to $\sqrt{\gamma_i}$. This means that 'energy of principal components' might not be a useful way to describe the way a reservoir deals with high dimensional input. Consider for instance a signal which consists of 10 useful channels with the same energy, and 90 noise channels, where the sum of the energies of the noise equals that of one useful channel,

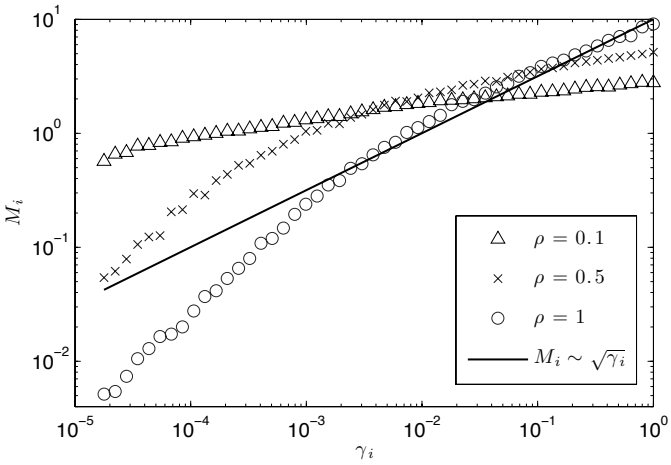

Fig. 3. Individual memory capacities per channel versus the corresponding energy (shown on logarithmic scale). Results are averaged over 10 runs.

i.e., only one eleventh of the total signal energy is devoted to noise. If $M_i \sim \sqrt{\gamma_i}$ we can calculate that the memory for the useful channels is equal to about $N/20$, i.e only half of the optimal value $N/10$. Indeed, we empirically verified this result to be correct with $N = N_{in} = 100$ where we found that the memory capacity for the useful channels is roughly equal to 5.

This result suggests that RC might benefit greatly from only presenting the first few principal components to the reservoir, rather than the full signal. Another potential method would be to multiply each principal component of the input signal with its energy in order to get $M_i \sim \gamma_i$ which might give a more 'fair' distribution of reservoir memory to the different signal components. Indeed, we repeated the experiment mentioned above after rescaling the input signal components and got a significant increase in memory for the useful channels (equal to about 9). It is also interesting to remark that PCA also can be linked to Hebbian learning [15], and that signal compression and feature selection plays a major role in the human brain.

IV. Conclusions

In this paper we investigated the memory properties for high dimensional input signals in random recurrent neural networks (reservoirs). We explored the impact of correlations in the input signal and how these affect the networks ability to store temporal information. We defined measures to quantify memory for high dimensional signals, which act on its principal components and are variations on the typical measure found in literature called the memory function and the memory capacity. We considered network setups with different degrees of nonlinearity and different spectral radii. We found that the total memory capacity (signifying the sum of the memory capacities of the different principal components) in the best case equals the number of neurons in the network, which was already a well established fact for one dimensional input signals. The available memory is distributed over the principal components of the input signal. The nature of the memory depends on the spectral radius of the network. If it is small (close to zero), the network will have a very good memory of the input but only for a few steps back in time. If the spectral radius is close to one, the memory stretches further back in the past but is significantly less precise.

Typically, the principal components of a high dimensional signal have widely varying energies, where the first few components carry the bulk of useful information, and the smaller components carry noise or less meaningful features. For this reason we tested how signal energy relates to its corresponding memory capacity. We found that a network with a spectral radius equal to one, spends an amount of memory capacity to each principal component roughly proportional to the square root of the corresponding energy (i.e. the standard deviation of the principal component). If the spectral radius drops, memory capacities become less dependent on energy. This result implies that reservoirs will lose a large part of memory to principal components which might carry only little useful information. For this reason it seems that PCA - which is known to be a valuable form of preprocessing for high dimensional signals - can also be used in Reservoir Computing. Other, more advanced techniques for variable selection could also be readily applied on the input mapping to only excite the reservoir with useful data features, such as not to congest the reservoirs memory.

It is a known fact that RC has difficulty to get good results when input dimensionality is close to that of the number of neurons, which can partially be explained by the results found in this paper. In future research it might be interesting to investigate ways to decrease signal redundancy by e.g. PCA based dimensionality reduction, or to find unsupervised or supervised techniques to adapt the random input mapping to improve the spatiotemporal representation of the input signal in the reservoir. Also it would be desirable to find an underlying mathematical framework which can confirm the empirically found results from this paper.

Appendix

A. Bias for the memory function

Here we will explain that the measured MF will approximately have a positive offset of $\frac{N}{T}$. We start by considering the principal components of the reservoir activation a. If we write the eigendecomposition of the covariance matrix $\mathbf{A} = \mathbf{U}\mathbf{\Psi}\mathbf{U}^\mathsf{T}$, the principal components are given by $\tilde{\mathbf{a}} = \mathbf{U}^\mathsf{T}\mathbf{a}$. Using this expression and the fact that $\mathbf{A}^{-1} = \mathbf{U}\mathbf{\Psi}^{-1}\mathbf{U}^\mathsf{T}$, we can rewrite equation 6 as

$$m_n(k) = \frac{\left\langle s_n(i-k)\tilde{\mathbf{a}}^\mathsf{T}(i)\right\rangle_i \mathbf{\Psi}^{-1} \left\langle s_n(i-k)\tilde{\mathbf{a}}(i)\right\rangle_i}{\gamma_n} \quad (7)$$

$$= \frac{1}{\gamma_n}\sum_{j=1}^{N} \psi_j^{-1} \left\langle s_n(i-k)\tilde{a}_j(i)\right\rangle_i^2, \quad (8)$$

with ψ_j the variance of the j-th principal component of the reservoir state. If the number of samples by which the covariances $\left\langle s_n(i-k)\tilde{a}_j(i)\right\rangle_i$ are estimated are finite,

there will always be a positive overestimation of the memory function; We start by splitting the principal components of the reservoir states: $\tilde{a}_j(l) = \bar{a}_j(l) + \hat{a}_j(l)$, where $\bar{a}_j(l) = \langle s_n(i-k)\tilde{a}_j(i)\rangle_i s_n(l-k)$, i.e. $\bar{a}_j(l)$ is the part of $\tilde{a}_j(l)$ that is fully correlated with the delayed signal, and $\hat{a}_j(l)$ is completely uncorrelated with the signal. Generally $\langle s_n(i-k)\tilde{a}_j(i)\rangle_i$ will be very small for each individual delay and principal component, especially if N_{in} is high. Therefore we approximate that $\sigma^2(\hat{a}_j(l)) \approx \sigma^2(\tilde{a}_j(l))$. Equation 8 then reduces to

$$\begin{aligned} m_n(k) &= \frac{1}{\gamma_n}\sum_{j=1}^{N}\psi_j^{-1}(\langle s_n(i-k)\bar{a}_j(i)\rangle_i^2 \\ &+ \langle s_n(i-k)\hat{a}_j(i)\rangle_i^2 + 2\langle s_n(i-k)\bar{a}_j(i)\hat{a}_j(i)\rangle_i). \end{aligned}$$

The third term in this equation will be very small and therefore we shall neglect it (this is especially valid since its expectance value is equal to zero). The first term is the actual measurement of the memory function, the second term is always positive and leads to the systematic overestimation, since it will always be positive with a finite sample size. The central limit theorem states that the expected value of the second term will be $\frac{1}{T}\sigma^2(s_n(i-k)\hat{a}_j(i))$, with T the number of samples. When we assume that $s_n(i-k)$ and $\hat{a}_j(i)$ are statistically independent, the variance of their product is equal to the product of their variances. This, and the assumption that $\sigma^2(\hat{a}_j(l)) \approx \sigma^2(\tilde{a}_j(l))$ allows us to work out the second term to be approximately equal to $\frac{N}{T}$.

B. Approximating the standard deviation for linear neural networks

To quantify the amount of nonlinearity in a reservoir, we will form an estimation of the standard deviation of a linear reservoir. It is possible to exactly calculate the standard deviations of the states for any given linear network, but the resulting formula is highly convoluted and does not give insight into its dependence on general parameters such as spectral radius and number of input channels. Therefore we shall have to make some broad assumptions to simplify this formula.

In a linear reservoir, the states have an analytical expression in function of the input signal [12]:

$$\mathbf{a}(i) = \sum_{k=0}^{\infty}\mathbf{W}^k\mathbf{V}\mathbf{s}(i-1-k). \tag{9}$$

Using this, we can calculate the average variance of the reservoir states:

$$\langle\mathbf{a}^\mathsf{T}(i)\mathbf{a}(i)\rangle_i = \left\langle\sum_{k=0}^{\infty}\sum_{l=0}^{\infty}\mathbf{s}^\mathsf{T}(i-1-k)\mathbf{P}^{kl}\mathbf{s}(i-1-l)\right\rangle_i,$$

with

$$\mathbf{P}^{kl} = \mathbf{V}^\mathsf{T}(\mathbf{W}^\mathsf{T})^k\mathbf{W}^l\mathbf{V}.$$

Writing this out as an explicit sum this becomes

$$\langle\mathbf{a}^\mathsf{T}(i)\mathbf{a}(i)\rangle_i = \sum_{k,l=0}^{\infty}\sum_{m,n=1}^{N_{in}}P_{mn}^{kl}\langle s_n(i-1-k)s_m(i-1-l)\rangle_i.$$

If we assume that $\langle s_n(i-1-k)s_m(i-1-l)\rangle_i = \gamma_m\delta_{nm}\delta_{lk}$, i.e. no spatial or temporal correlation exists in the input signal, this expression simplifies to

$$\langle\mathbf{a}^\mathsf{T}(i)\mathbf{a}(i)\rangle_i = \sum_{m=1}^{N_{in}}\gamma_m\sum_{k=0}^{\infty}P_{mm}^{kk}.$$

The term $\sum_{k=0}^{\infty}P_{mm}^{kk}$ can be calculated analytically for individual reservoirs, but it doesn't give a general idea of how it relates to N_{in} and the spectral radius of the reservoir. For this reason we make the following strong simplification: we only consider orthogonal connection matrices, such that $\mathbf{W}\mathbf{W}^\mathsf{T} = \rho^2\mathbf{1}$. With these assumptions we can then calculate that

$$\begin{aligned}\sum_{k=0}^{\infty}P_{mm}^{kk} &= \sum_{k=0}^{\infty}\rho^{2k}\left[\mathbf{V}^\mathsf{T}\mathbf{V}\right]_{mm} \\ &= \frac{1}{1-\rho^2}\left|\mathbf{V}_m\right|^2,\end{aligned}$$

where $\mathbf{V}_m$ signifies the m-th column of $\mathbf{V}$. To come to a useful final result, we can make further assumptions. If the elements of $\mathbf{V}_m$ are chosen from a normal distribution with unit standard deviation and zero mean, then its square norm has a Chi-square distribution and its mean is equal to the number of elements (i.e. N). Using this, we can finally write down the mean variance of the reservoir states:

$$\frac{1}{N}\langle\mathbf{a}^\mathsf{T}(i)\mathbf{a}(i)\rangle_i = \frac{\sum_{m=1}^{N_{in}}\gamma_m}{1-\rho^2}. \tag{10}$$

We now have a formula that gives an estimation of the variance of the network states with a very clear relation to the signal energy and ρ: the state variance is proportional to the total signal energy, and increases when ρ increases. In the special case that all energies γ_m are equal to one (whitened data), this gives us

$$\frac{1}{N}\langle\mathbf{a}^\mathsf{T}(i)\mathbf{a}(i)\rangle_i = \frac{N_{in}}{1-\rho^2}. \tag{11}$$

We will check whether the approximations we made still can be generalized for regular (i.e. non-orthogonal $\mathbf{W}$). Therefore, we simulate linear networks with the conditions described above (i.e the elements of $\mathbf{V}$ have unit standard deviation and zero mean). We study this for two types of input signals: one with $\gamma_i = 1$ (using Equation 11), and one where $\gamma_i = 10^{-5\frac{i}{N_{in}}}$ (using Equation 10), which is comparable to the signal used in Section III. We measure the average standard deviation in relation to ρ and N_{in} and compare this with our estimation. We used 10000 input samples and averaged over 10 reservoir initializations with 100 neurons. Figure 4a and 4b shows the comparison between our estimation and the measured value of the mean standard deviation of the network activations. Clearly there is a good correspondence between our formula and the actual result. When ρ gets closer to one, it appears we systematically overestimate the standard deviation. This is likely due to the fact that the eigenvalues of orthogonal matrices all have the same absolute value, which corresponds to the same

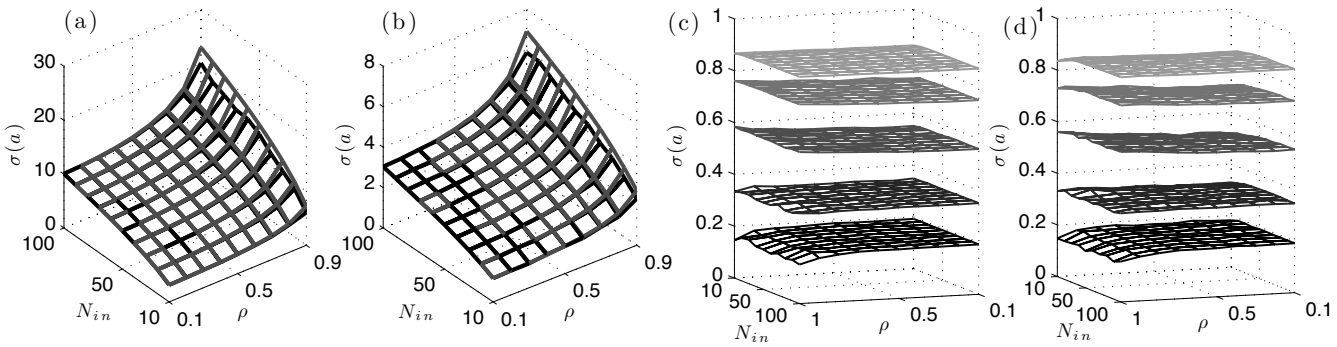

Fig. 4. (a,b): Comparison between the mean standard deviation of reservoir states measured empirically (black) and the estimated value (grey) for principal components with the same energy (a) and an energy spectrum as described in the text (b). (c,d): Empirically measured standard deviations in hyperbolic tangent reservoirs for different corrected input scaling factors: from light grey to black, $\phi = \{0.2,\ 0.4,\ 0.8,\ 1.6,\ 3.2\}$, for principal components with the same energy (c) and an energy spectrum as described in the text (d)

amplification for all its oscillatory modes. Random matrices have their eigenvalues more or less uniformly spread over the area of the disk with radius ρ, such that many of its oscillatory modes will decay more quickly.

It is now easy to define an input scaling ϕ which signifies the amount of nonlinearity of the network states. We choose the initial input weights from a normal distribution with unit variance and zero mean, and next we multiply this with

$$\phi \sqrt{\frac{1-\rho^2}{\sum_{m=1}^{N_{in}} \gamma_m}}, \tag{12}$$

where ϕ is the desired standard deviation of the reservoir states. One major drawback of the above formula is the fact that it goes to zero when ρ approaches one (and becomes even imaginary when $\rho > 1$). Obviously there is no real issue using a reservoir with spectral radius equal to or slightly higher than one, exactly because the effective spectral radius will always drop under one when the states are pushed far enough in the nonlinear part of the hyperbolic tangent. We will replace ρ with an estimate of the spectral radius of the Jacobian $\mathbf{J}$ of the system, which is defined as

$$J_{ij}(k) = \frac{\partial a_i(k)}{\partial a_j(k-1)} = (1 - a_i^2(k-1))W_{ij}.$$

Disregarding individual differences between the reservoir states, we estimate the average spectral radius of $\mathbf{J}$ as

$$\rho_J = (1 - \frac{1}{N} \left\langle \mathbf{a}(k)^\mathsf{T} \mathbf{a}(k) \right\rangle_k)\rho.$$

We now have to estimate the variance of the states for a nonlinear reservoir, which is very hard indeed. However, since our goal is to give a rough input scaling factor which also works for $\rho \approx 1$, we found an ad hoc solution which works quite well. Since we 'impose' the standard deviation ϕ upon the linearized reservoirs, we expect that the standard deviation in the nonlinear reservoir will be roughly equal to the hyperbolic tangent of ϕ. This gives us the above correction for equation 12:

$$\phi \sqrt{\frac{1-\rho^2(1-\tanh(\phi)^2)^2}{\sum_{m=1}^{N_{in}} \gamma_m}}. \tag{13}$$

We tested these assertions by measuring the standard deviation of reservoir states in hyperbolic tangent reservoirs, and see how much the actual mean standard deviation of the network states still depends on ρ and N_{in} after rescaling. Figure 4 shows the result for mildly to very nonlinear reservoirs (i.e. with mean standard deviation close to one). It appears that our result generally holds quite well, and the mean standard deviation only depends little on ρ and N_{in} in the tested ranges.

ACKNOWLEDGMENTS

The research leading to the results presented here has received funding from the European Community's Seventh Framework Programme (EU FP7) under grant agreement n. 231267 " Self-organized recurrent neural learning for language processing (ORGANIC)" and the Photonics@be Interuniversity Attraction Poles program (IAP 6/10), initiated by the Belgian state, Prime Minister's Service, Science Policy Office. This work was partially funded by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

REFERENCES

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proc Natl Acad Sci U S A*, vol. 79, no. 8, pp. 2554–2558, Apr 1982.

[2] D. Rumelhart, G. Hinton, and R. Williams, *Learning internal representations by error propagation*. MIT Press, Cambridge, MA, 1986.

[3] B. Hammer and J. J. Steil, "Perspectives on learning with recurrent neural networks," in *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2002.

[4] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 148, 2001.

[5] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[6] D. Verstraeten, B. Schrauwen, and J. Van Campenhout, "Recognition of isolated digits using a liquid state machine," in *Proceedings of SPS-DARTS 2005*, April 2005, pp. 135–138.

[7] B. Fontaine, H. Peremans, and B. Schrauwen, "Bat echolocation modelling using spike kernels with Support Vector Regression," in *Proceedings of the 15th European Symposium on Artificial Neural Networks*, 2007, pp. 367–372.

[8] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2009, pre–publication.

[9] S. Ganguli, D. Huh, and H. Sompolinsky, "Memory traces in dynamical systems." *Proceedings of the National Academy of Sciences of the United States of America*, November 2008.

[10] M. Hermans and B. Schrauwen, "Memory in linear recurrent neural networks in continuous time," *Neural Networks, (doi:10.1016/j.neunet.2009.08.008)*, 2009.

[11] H. Jaeger, "Short term memory in echo state networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 152, 2001.

[12] O. L. White, D. D. Lee, and H. Sompolinsky, "Short-term memory in orthogonal neural networks," *Physical Review Letters*, vol. 92, p. 148102, 2004.

[13] M. C. Ozturk, D. Xu, and J. C. Principe, "Analysis and design of echo state networks," *Neural Computation*, vol. 19, pp. 111–138, 2006.

[14] D. Verstraeten and B. Schrauwen, "On the quantification of dynamics in reservoir computing," in *Proceedings of the International Conference on Analog Neural Networks (ICANN)*, 2009, pp. 985–994, (submitted).

[15] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, 1982.