Reservoir computing

Course: Advanced machine learning using neural networks.
Contact: Ludvig Storm, ludvig.storm@physics.gu.se

---

1 Background[1]

Recurrent neural networks can be trained in different ways. A common method is backpropagation through time [1]. An alternative to backpropagation through time is *reservoir computing* [1, 2], the topic of this assignment. Reservoir computing has been used with tremendous success to predict chaotic dynamics [3, 4], and rare transitions in stochastic systems [5].

Consider input data in the form of a time series $\boldsymbol{x}(0), \dots \boldsymbol{x}(T-1)$ of N-dimensional vectors $\boldsymbol{x}(t)$, and a corresponding series of M-dimensional targets $\boldsymbol{y}(t)$. The goal is to train the recurrent network so that its outputs $\boldsymbol{O}(t)$ approximate the targets as precisely as possible, by minimising the energy function $H = \frac{1}{2}\sum_{t=\tau}^{T-1}\sum_{i=1}^{N}[y_i(t) - O_i(t)]^2$, where τ represents an initial transient that is disregarded.

Figure 1 shows the layout for this task. There are N input terminals. They are connected with weights $w_{jk}^{(\mathrm{in})}$ to a reservoir of hidden neurons with state variables $r_j(t)$. The reservoir is linked to M linear output units $O_i(t)$ with weights $w_{ij}^{(\mathrm{out})}$. The reservoir itself is a recurrent network by weights w_{ij}. There are many different versions of the update rules that differ in detail [6]. One possibility is [5]

$$r_i(t+1) = g\Big(\sum_j w_{ij} r_j(t) + \sum_{k=1}^{N} w_{ik}^{(\mathrm{in})} x_k(t)\Big)\,, \tag{1a}$$

$$O_i(t+1) = \sum_{j=1}^{M} w_{ij}^{(\mathrm{out})} r_j(t+1)\,, \tag{1b}$$

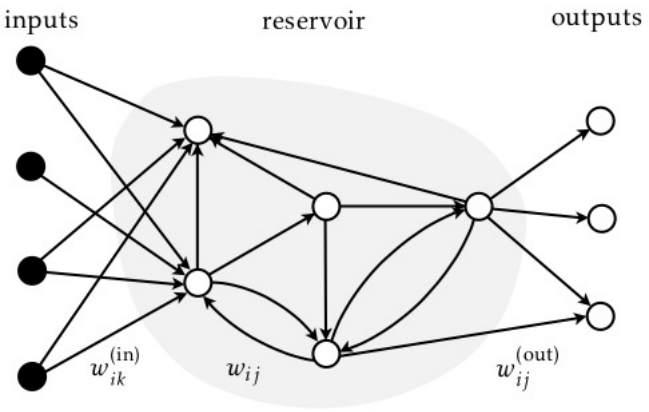
for $t = 0, \dots, T-1$, with initial conditions $r_j(0) = 0$.

The main difference to standard training with backpropagation [1] is that the input weights $w_{jk}^{(\mathrm{in})}$ and the reservoir weights w_{jk} are randomly initialised and kept constant. Only the output weights $w_{jk}^{(\mathrm{out})}$ are trained. The idea is that the dynamics of a sufficiently large reservoir finds nonlinear, high-dimensional representations of the input data [2], not unlike sparse representations of binary classification problems embedded in a high-dimensional space that become linearly separable in this way [1]. In addition, the reservoir serves as a dynamical memory.

This requires that the reservoir states faithfully represent the input sequence: similar input sequences should yield similar reservoir activations, provided one iterates the reservoir dynamics long enough. However, for random weights the recurrent reservoir dynamics can be chaotic [7]. In this case, the state of the reservoir after many iterations bears no relation to the input sequence. To avoid this, one requires that the reservoir dynamics is linearly stable. Linearising the reservoir dynamics (1a) gives

$$\delta\boldsymbol{r}(t+1) = \mathbb{D}(t+1)\mathbb{W}\delta\boldsymbol{r}(t)\,, \tag{2}$$

[1] This text is adapted from Ref. [1].



Figur 1: Reservoir computing (schematic). Not all connections are drawn. There can be connections from all inputs to all neurons in the reservoir (gray), and from all reservoir neurons to all output neurons.

where $\mathbb{D}(t+1)$ is a diagonal matrix with entries $D_{ii}(t+1) = g'[b_i(t+1)]$. Whether or not the norm of $\delta\boldsymbol{r}$ grows is then determined by the eigenvalues of matrices of the form $\mathbb{J}_n^\mathsf{T}\mathbb{J}_n$, where $\mathbb{J}_n$ are matrix products $\mathbb{D}(n)\mathbb{W}\mathbb{D}(n-1)\mathbb{W}\cdots\mathbb{D}(1)\mathbb{W}$, see Ref. [1]. The square roots of these eigenvalues are the singular values of the matrix $\mathbb{J}_n$, denoted by $\Lambda_1(n) \geq \Lambda_2(n) \geq \cdots$. At large times, the stability condition reads:

$$\lim_{n\to\infty} n^{-1} \log \Lambda_1(n) < 0\,. \tag{3}$$

Sometimes the stability criterion is quoted in terms of the maximal eigenvalue of $\mathbb{W}$. If one uses tanh activation-functions and ensures that the local fields $b_i(t)$ remain small, then the diagonal elements of $\mathbb{D}(t)$ remain close to unity. In this case the stability condition for the reservoir dynamics is given by the weight matrix $\mathbb{W}$ alone. But in general this is not the case. Note also that in general the singular values of $\mathbb{W}$ are different from its eigenvalues.

There are many different recipes for setting up a reservoir. Usually one draws the elements of $\mathbb{W}$ independently from a given distribution and rescales them in a suitable way. One may take a uniform distribution within given thresholds, or a Gaussian one, or one may assume that $w_{ij} = \pm 1$ with equal probability. Usually one assumes that the reservoir is sparse, with only a small fraction of weights non-zero. The weight matrix $\mathbb{W}^{(\mathrm{in})}$ is usually taken to be a full matrix, and its elements are drawn from the same distribution as those of the reservoir. Lukosevicius [6] gives a practical overview over different schemes for setting up reservoir computers. Tanaka *et al.* [9] describe different physical implementations of reservoir computers, based on electronic RC-circuits, optical cavities or resonators, spin-torque oscillators, or mechanical devices.

2 Assignment

For an introduction to non-linear time-series prediction, refer to Ref. [10]. To predict time series with a reservoir computer, one trains it on an input series $\boldsymbol{x}(0), \ldots, \boldsymbol{x}(T-1)$ with targets $\boldsymbol{y}(t) = \boldsymbol{x}(t)$. After training, one continues to iterate the network dynamics with inputs $\boldsymbol{x}(T+k) = \boldsymbol{O}(T+k)$ to predict $\boldsymbol{x}(T+k+1)$, for $k = 0, 1, 2, \ldots$. In order

to represent complex spatio-temporal patterns, Pathak *et al.* [3] found it necessary to use multiple reservoirs. Lim *et al.* [5] use a multi-layer version of the reservoir dynamics, replacing Equation (1a) by a set of nested update rules.

The Lorenz system

$$\frac{\mathrm{d}}{\mathrm{d}t}x_1 = -\sigma x_1 + \sigma x_2\,, \tag{4a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}x_2 = -x_1 x_3 + r x_1 - x_2\,, \tag{4b}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}x_3 = x_1 y - b x_3\,, \tag{4c}$$

is a model for atmospheric convection [11]. Time series generated by Equation (4) are chaotic and therefore difficult to predict.

1. Generate a training set of time series $x_2(t)$ by by numerical solution of Equation (4), for $\sigma = 10$, $r = 28$, and $b = 8/3$. See Exercise 3.2 in *Nonlinear time series analysis* by Kantz and Schreiber [10] (**2**pt).

2. Set up a reservoir as described in Equation (1) and train the output weights so that the energy function is minimised when predicting the Lorenz system (**2**pt).

3. Train two reservoirs, one on the entire Lorenz system and one on a single variable of the Lorenz system, say x_2. When training on a single variable, you only need to predict that variable. Evaluate their prediction performance over a range of different values for the maximal singular value of the reservoir weight matrix $\mathbb{W}$. Approximate the upper and lower bounds of singular values that allow the reservoir to predict the time series in both cases. Discuss the bounds and what differences you observe in the two cases, and why the differences appear. *Hint: Plot the prediction performance versus the maximal singular value using a log-scale for the x-axis* (**4**pt).

4. Evaluate how well the reservoir computer predicts the time series $x_2(t)$ and compare with standard methods [10]. To this end, determine the maximal Lyapunov exponent λ_1 by numerical simulation of Eq. (4) [7] of the Lorenz system and check for how many Lyapunov times the time-series prediction works. Plot the prediction error as a function of $t\lambda_1$ (**2**pt).

3 Examination

For the individual oral examination you should present your own codes you used to answer the above questions. Make sure you are well prepared to present your results as the time is short.

- Non-linear time-series analysis. Lyapunov exponent.
- Reservoir computers. Reservoir dynamics. Criteria for initialisation of reservoir.
- Training a reservoir computer on chaotic Lorenz time series.
- How well does the reservoir computer predict the Lorenz time series? How can the performance be improved?

Referenser

[1] B. Mehlig. *Machine learning with neural networks*. Cambridge University Press, Cambridie, 2021.

[2] M. Lukosevicius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3:127, 2009.

[3] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, 120:024102, 2018.

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

[5] S. H. Lim, L. T. T. Giorgini, W. Moon, and J. S. Wettlaufer. Predicting critical transitions in multiscale dynamical systems using reservoir computing. arxiv.org/abs/1908.03771, 2019. [Last accessed 5-December-2020].

[6] M. Lukosevicius. A practical guide to applying echo state networks. In G. Montavon, G.B. Orr, and K.R. Müller, editors, *Neural Networks: Tricks of the Trade*, Berlin,Heidelberg, 2012. Springer.

[7] S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

[8] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos*, 27:121102.

[9] G. Tanaka, T. Yamane, J. B. Héroux, et al. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100 – 123, 2019.

[10] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, Cambridge, 2004.

[11] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.