Research | March 01, 2018

Reservoir Computing: Harnessing a Universal Dynamical System

By Daniel J. Gauthier

There is great current interest in developing artificial intelligence algorithms for processing massive data sets, often for classification tasks such as recognizing a face in a photograph. But what if our goal is to learn a deterministic dynamical system? Relevant applications include forecasting the weather, controlling complex dynamical systems, and fingerprinting radio-frequency transmitters to secure the internet of things.

Training a “universal” dynamical system to predict the dynamics of a desired system is one approach to this problem that is well-suited for a reservoir computer (RC): a recurrent artificial neural network for processing time-dependent information (see Figure 1). It can operate in many modes, including prediction mode, the task described above. While researchers have studied RCs for well over 20 years [1] and applied them successfully to a variety of tasks [2], there are still many open questions that the dynamical systems community may find interesting and be able to address.

An RC is distinguished from traditional feed-forward neural networks by the following qualities:

- The network nodes each have distinct dynamical behavior
- Time delays of signals may occur along the network links
- The network’s hidden part has recurrent connections
- The input and internal weights are fixed and chosen randomly
- Only the output weights are adjusted during training.

The last point drastically speeds up the training process.

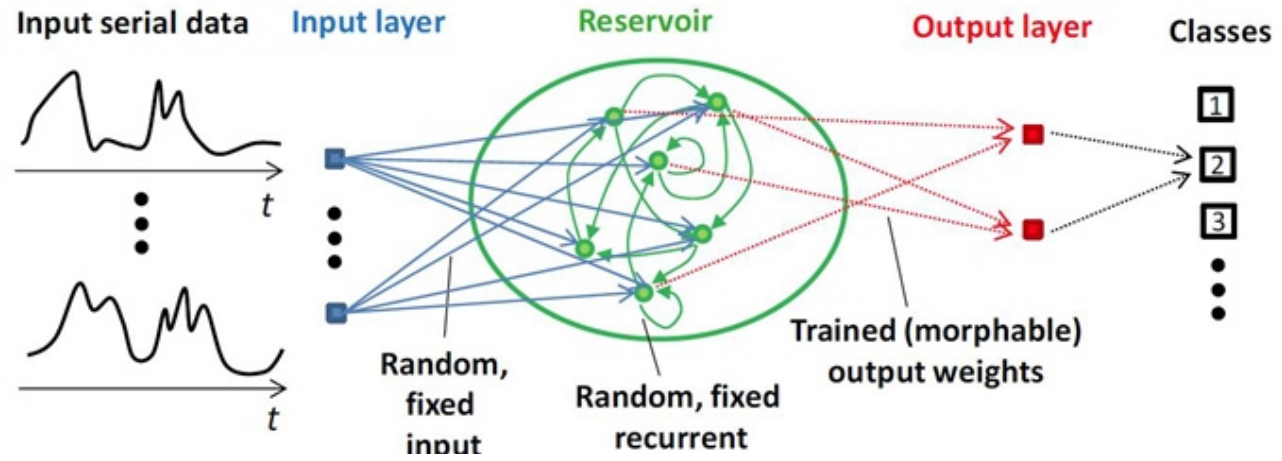

Figure 1.. Illustration of the reservoir computer architecture. Figure credit: Daniel J. Gauthier.

Mathematically, an RC is described by the set of autonomous, time-delay differential equations given by

$$\frac{dx_i}{dt} = -\gamma_i x_i + \gamma_i f_i\Big[\sum_{j=1}^{j} W_{i,j}^{in} u_j(t) + \sum_{n=1}^{N} W_{i,n}^{res} x_n(t-\tau_{i,n}) + b_i\Big], \qquad (1)$$

$$y_k(t) = \sum_{m=1}^{N} W_{k,m}^{out} \mathcal{X}_m, \quad i = 1,\ldots,N \quad k = 1,\ldots,K,$$

with J inputs u_j, N reservoir nodes x_i, and K outputs with values y_k. Here, γ_i are decay constants, $W_{i,j}^{in}(W_{i,n}^{res})$ are fixed input (internal) weights, $\tau_{i,n}$ are link time delays, b_i are biases, and $W_{k,m}^{out}$ are the output weights whose values are optimized for a particular task. The nonlinear function f is typically sigmoidal, which we can take to the limit of an on-off thresholding (Boolean) function, as is done in traditional Hopfield networks. The reservoir maps the J input data streams to a higher dimensional phase space — dimension expansion.

For the prediction task, we adjust $W_{k,m}^{out}$ using a finite-duration “training” data sample so that the resulting output represents the input data in a least-square sense. After training, the input signals are disconnected and the outputs are wired to inputs to start the prediction phase.

In greater detail, W^{out} is determined by injecting an input training data set U over a time T_{train} and observing the network dynamics X over this interval. Based on these observations, we modify the weights to minimize the error of the output Y to the desired output Y^{des}, resulting in

$$\mathrm{W}^{out} = \mathrm{Y}^{des}\mathrm{X}^{T}(\mathrm{X}\mathrm{X}^{T} + \alpha^2\mathrm{I})^{-1}, \qquad (2)$$

where α is a regularization parameter, I is the identity matrix, and T indicates the transpose.

We can solve (2) in a least-square sense using pseudo-inverse matrix routines that are often included in a variety of computer languages, some of which can take advantage of the matrices’ sparseness. A nonzero value of ensures that the norm of W^{out} does not become large, which improves the generalizability of the system to different inputs and increases noise tolerance. We can also find a solution to (2) using gradient descent methods, which are helpful when the matrix dimensions are large, and leverage toolkits from the deep learning community that take advantage of graphical processing units. Use of recursive least-squares is another approach.

An RC can work very well in the prediction task. For example, it is possible to learn the full attractor of a dynamical system when the reservoir dynamics is projected to a lower-dimensional

phase space before training [3]. We can also learn the attractor with standard training approaches and accurately find Lyapunov exponents from the time series produced by the RC, even for spatial-temporal dynamics [7]. Furthermore, we can utilize the predicted time series as an observer in a control system [4] or for data assimilation of large spatiotemporal systems without use of an underlying model [6]. These results suggest that an RC is a powerful tool for characterizing complex dynamical systems.

While these conclusions are compelling, designing an RC for a particular task is largely a trial-and-error undertaking, and authors tend to present results that work without dwelling on those that fail. The following is an open question: how can we optimize the parameters in (1) and (2) to obtain the most accurate prediction in either the prediction or classification tasks, while simultaneously allowing the RC to function well on data that is similar to the training data set? Early studies focused on the so-called echo state property of the network—where the output should eventually forget the input—and the consistency property, where outputs from identical trials should be similar over some period. These conditions were initially assumed to be guaranteed when the spectral radius of $\mathbf{W}^{res}$ is less than one (for the case when $b_i = 0$.

However, this scenario ignores the input dynamics and is mostly a statement of the stability of $\mathbf{X} = 0$. Recent work is beginning to address this shortcoming for the case of a single input channel, demonstrating that there must be a single entire output solution given the input [5].

While a base of past research exists, many questions that demand quantitative and rigorous answers remain. For example, how large must N be to achieve a desired error rate? How should we adjust γ_i relative to the timescales of the original dynamical system? Why do sparsely-connected reservoirs often perform best?

At the 2017 SIAM Conference on Applications of Dynamical Systems, held in Snowbird, Utah, last May, Edward Ott and I organized a minisymposium on RCs to discuss these and other problems. Ott showed that RCs can learn the “climate” of a dynamical system and accurately forecast spatiotemporal chaos in a scalable manner. Roger Brockett indicated that dense network connections might give rise to partial or full synchronization of the reservoir nodes, thus diminishing the diversity of waveforms that an RC can learn. Brian Hunt suggested that an RC must synchronize to the input data in a generalized sense when used for the prediction task. Finally, I discussed a hardware-based RC capable of predicting at a rate exceeding tens of MHz.

In summary, RCs can serve as a universal dynamical system capable of learning the dynamics of other systems. This may prove advantageous when obtaining data for the learned dynamical system is expensive or difficult, for example. While the field is progressing rapidly, there are still substantial openings for others to join the effort.

Acknowledgments: I gratefully acknowledge discussions of this work with Roger

Brockett, Daniel Canaday, Ingo Fischer, Michelle Girvan, Aaron Griffith, Alexander Hartemink, Nicholas Haynes, Brian Hunt, Zhixin Lu, Edward Ott, and Jaideep Pathak, and the financial support of the U.S. Army Research Office Grant No. W911NF12-1-0099.

References

[1] Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science, 304*(5667), 78-80.

[2] Larger, L., Baylón-Fuentes, A., Martinenghi, R., Udaltsov, V.S., Chembo, Y.K., & Jacquot, M. (2017). High-speed photonic reservoir computing using time-delay-based architecture: Million words per second classification. *Phys. Rev. X, 7*, 011015.

[3] Løkse, S., Bianchi, F.M., & Jessen, R. (2017). Training echo state networks with regularization through dimensionality reduction. *Cogn. Comput., 9*, 364.

[4] Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., & Ott, E. (2017). Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos, 27*, 041102.

[5] Manjunath, G., & Jaeger, H. (2013). Echo State Property Linked to an Input: Exploring a Fundamental Characteristic of Recurrent Neural Networks. *Neur. Comp., 25*, 671.

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

[7] Pathak, J., Lu, Z., Hunt, B.R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. Preprint, *arXiv:1710.07313*.

Daniel J. Gauthier is a professor of physics at Ohio State University and is interested in a wide range of topics in dynamical systems, quantum optics, and quantum information science.