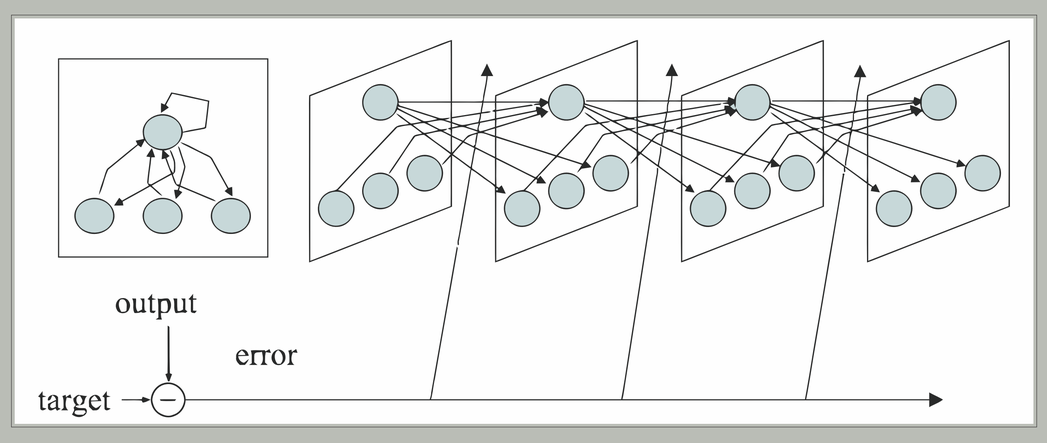
Reservoir Computing for Recurrent Neural Networks

Felix Grezes {fgrezes@gc.cuny.edu}
advised by Pr. Andrew Rosenberg and Pr. Noemie Elhadad
The Speech Lab at Queens College CUNY

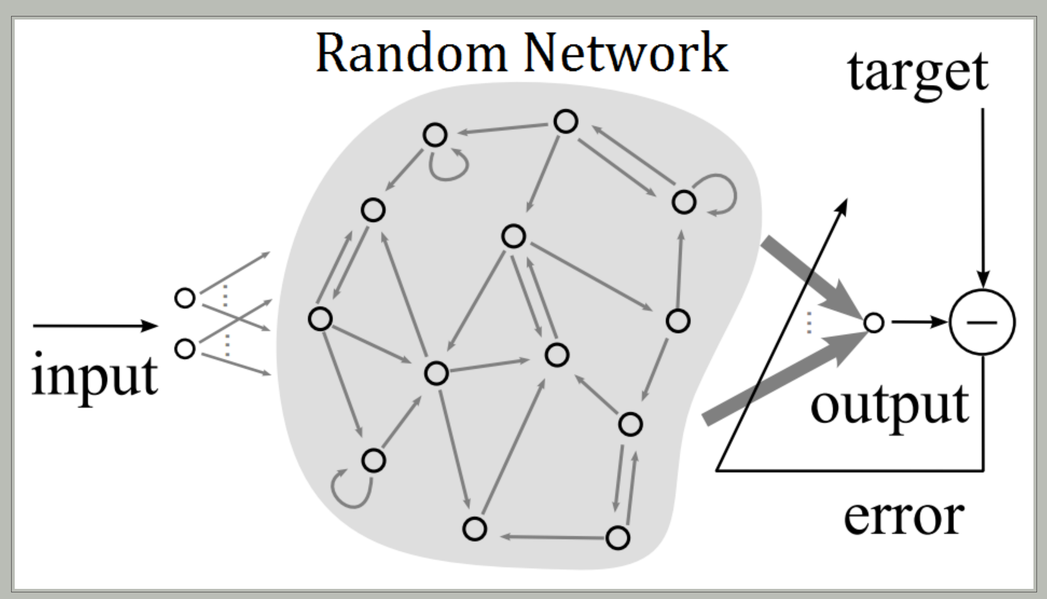
Abstract

Historically, research has focused on *Recurrent Neural Networks* (RNNs) because of their capacity to model dynamical system and/or their biological plausibility. However traditional neural network approaches, such has gradient descent, are either impossible or too slow to be applied successfully to RNNs.
The *Reservoir Computing* paradigm, by separating the optimization of the network layer from the output layer, attempts to solve these issues. Since 2001, it has proven itself to be fast and effective in a variety of applications.

The Reservoir Computing Paradigm



In traditional RNN approaches, the input-output error is propagated throughout all the weights of the networks. Unfortunately RNNs are difficult to train by gradient descent based methods, which aim at iteratively reducing the error. The gradual changes lead to bifurcations, and at such points, the gradient information degenerates and may become ill-defined. As a consequence, convergence cannot be guaranteed. Additionally gradient information dissolves exponentially fast, making it is intrinsically hard to learn long-range memory dependencies.



In a reservoir network, only the weights in the output layer are trained by the errors. The network weights and connections are ***randomly*** set at the beginning and never updated. Like conventional RNNs, a reservoir network possesses a *dynamical memory* and is able to process temporal context information.

Output Layer Training

Training the output layer is a common, non-temporal task of mapping the input to the target output. This is a well investigated domain in machine learning, and a large choice of methods are available. Some of the most popular are Linear Regression and SVMs.

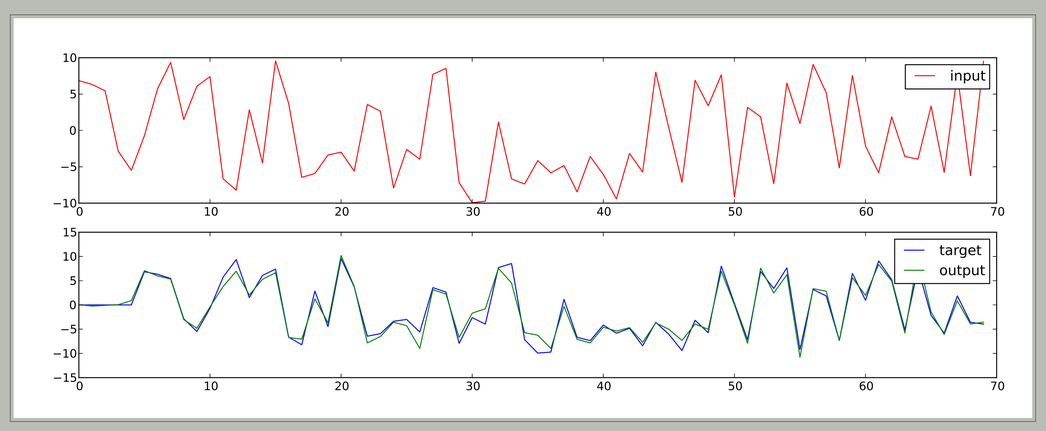
Reservoir Creation

However, not all reservoir networks with random architectures have desirable behavior. Most importantly, the cycles in the network can produce amplification effects, eventually saturating the nodes.
Good networks possess the following two useful, but conflicting, properties:
- Echo State Property: the effect of a previous state and input should vanish over time.
- Separability: different inputs should stabilize on different outputs.

A common practice to balance these two properties is to make the weight matrix *big, sparse and random*; which makes the activation signals *numerous, decoupled and varied*. The weight values are then downscaled to prevent any amplification.

A Toy Problem

We define a simple task to display the temporal memory of reservoir networks. The goal is to reproduce a random input signal with a K-steps delay.



As shown above, a network consisting of just 50 neurons can reproduce the input signal (in red) with a 5 time-step delay almost perfectly. The desired target signal is in blue and the produced output signal in green.

∗ Note: The above results are artificially worsened for visualization purposes.

Current Research and Applications

Since it's inception in the early 2000s, the Reservoir Computing paradigm has been successfully been applied to a variety of scientific fields:
- NLP: *Phoneme recognition with large hierarchical reservoirs*, Triefenbach et al., NIPS 2011.
- Computational Neuroscience: *A reservoir of time constants for memory traces in cortical neurons*, Bernacchia et al., Nature Neuroscience 2011.
- Physics: *Constructing optimized binary masks for reservoir computing with delay systems*, Appeltant et al., Nature 2014

My own research is centering on two topics: applying the Reservoir Computing paradigm to speech NLP tasks within Pr. Rosenberg's lab; and studying how reservoir can be used for multi-task learning (as described by Collobert et al.), applying the paradigm to medical data under the supervision of Pr. Elhadad.