FORCE learning

We studied the FORCE (First Order Reduced Controlled Error) learning approach. This approach uses a recurrent neural network and its chaotic neural activity to reproduce an arbitrary function. In contrast to more traditional approaches, the goal is to reduce the amount of modification needed to keep errors small (rather than reducing initially large errors).

From a machine learning point of view, FORCE learning can generate complex and controllable patterns of activity in the absence of or in response to input. From a biological perspective, it can be model training-induced modification.

FORCE learning works in the following way: A weighted sum of the output of the neurons approximates an arbitrary target function. The weights are fitted to the target function using the recursive least-squares method. The approximation is fed back into the network. This is a difference to echo-state networks, were the target function is used as a feedback. Using the approximation turns out to make the learning more robust, because it includes fluctuations.

This way, three problems characteristic to training recurrent neural networks are solved:

1. Feeding erroneous output back into the network during training can cause the learning to diverge and fail.
2. Figuring out which neurons and synapses are most responsible for output errors (and therefore most in need of modification) can be very challenging.
3. Chaotic spontaneous activity can be problematic.

During the group work, we studied a paper about FORCE learning by Sussillo and Abbot. They describe the approach and apply it to various problems. We decided to reproduce their results for the simplest test: Fitting a triangular wave and a square wave function. Although we were able to get reasonable results, we had to use different parameters than in the paper, otherwise the learning would not converge. This is probably because our implementation of FORCE learning is not equivalent to theirs.

Result:

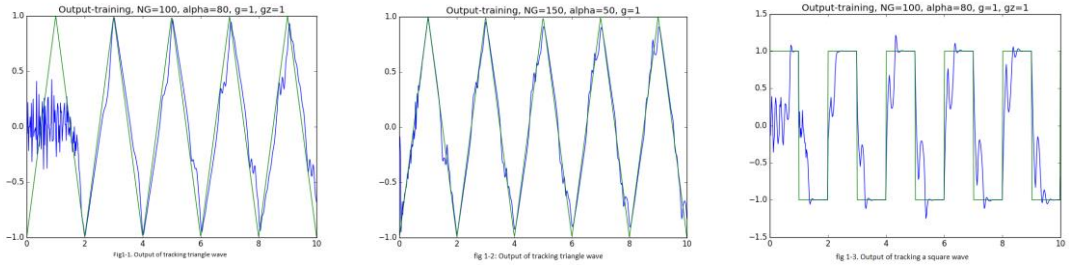

Figure1: Output of tracking arbitrary input

During the training, we train the output weight using force learning method and it can track arbitrary inputs, here a triangle wave and square wave.

```
steps = 500
t = np.linspace(0, 10, steps)
inputsig = signal.square(2 * np.pi * 0.5 * t)
inputsig = signal.sawtooth(2 * np.pi * 0.5 * t, width = 0.5)
```

As shown in the plot, at the beginning of the training, the output is far away from the desired wave. But after training for around 100 steps, the output of the network can track the input with reasonable error. And the increasing the network size in a certain range can improve the performance.

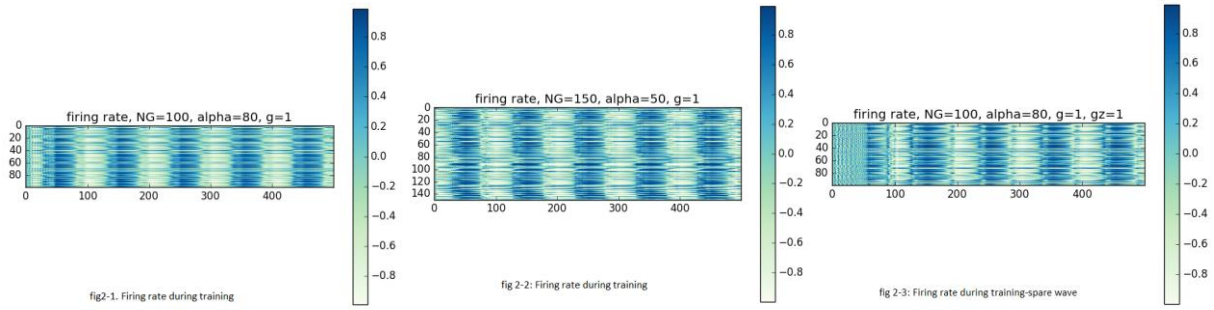

Figure 2: Firing rate during training.

As shown in Figure 2, the firing rate of the network is chaotic at the beginning of the training, and later fall into nice pattern due to the periodic feature of the input. And a closer look of individual neurons firing rate, it also displays a nice pattern despite the large high frequency of the fluctuation at the beginning.

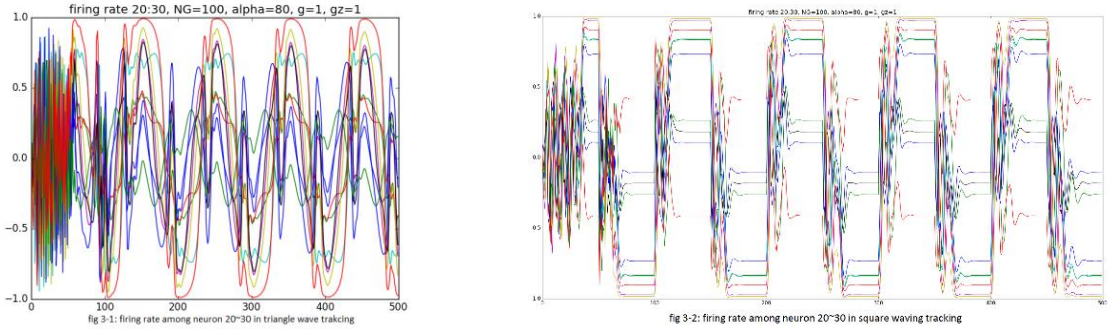

Figure3: selected neurons firing rate

In Figure 3, they are randomly selected neurons' firing rate. As it is shown, after a short time of chaotic adjustment, they all follow nice patterns reflecting they are following the periodic inputs.

If you are interested in the original paper, here is the link.
https://www.ncbi.nlm.nih.gov/pubmed/19709635