Paper Implementation: Reproduce some basic results of the article “Generating Coherent Patterns of Activity from Chaotic Neural Networks”

Hsieh Yu-Guan

June 17, 2017

Summary

In this report we'll be working on the article “Generating Coherent Patterns of Activity from Chaotic Neural Networks” published by David Sussillo and L.F. Abbott in 2009. In the work of D. Sussillo and L.F. Abbott, they were interested in neural networks that are capable of generating complex activity patterns either spontaneously or in response to different stimuli.

The network is meant to be chaotic before training, i.e. its activity should be highly sensitive to initial conditions. This is acheieved by the usage of strong recurrent synaptic connections inside it. The so-called FORCE learning algorithm is then employed to modify synaptic weights either external to or within the network. After the training phase, the network will be able to produce a wide variety of complex output patterns and we can switch between different outputs by controlling the input signal.

Now let's come back to this report itself. I only worked on a relatively small part of the article: the generation of periodic patterns in the absence inputs by using FORCE learning. Moreover, three distinct network structures are considered in the article (this will be detailed later on), but only the first one, which is probably also the simpest, is implemented here.

Introduction

When we're performing some motor action, what is the source of the nerual activity that initiates and carries out this behavior? In the paper of D. Sussillo and L.F. Abbott, they examined the hypothesis that such actions may come from the reorgnaization of spontaneous neural activity, which then leads us to another question: how can a such reorgnaization happen?

The objective is thus to find a possible synaptic weight modification rule that allows the network activity to be reorgnaized into coherent patterns required to produce complex motor actions after training. However, comparing with feedfoward architectures, the training of a recurrent network is shown to be much more difficult due to several reasons. First, feeding erroneous output back into a network during training may prevent its activity from converging. Secondly, in a recurrent network the credit assignement for output errors becomes quite a hard problem because of the presence of recurrent weights. Finally, while using a network that display chaotic activity prior to training can be beneficial, chaos must be avoided during the learning phase.

FORCE learning solved these problems by keeping the errors always small and forcing the synaptic modifications to be strong and rapid during traing. This is quite different from classic learning algorithms which usually reduce the errors little by little. Therefore, the goal of FORCE learning is not significant error reduction, but rather reducing the amount of modification needed to keep the errors small.

According to the article, the contribution of FORCE learning can be viewed from two different angles. From a biological point of view, it can be regarded as a model for learning-induced moficifation of biological models. Or more simply, it can just be seen as a powerful algorithm that is able to construct recurrent networks being able to generate complex and controlable output patterns.

Network Structures

The network that was studied in the paper and will be study here is composed of individual neurons that are characterized by their firing rates and sparsely interconnected through excitatory and inhibiroty synapses of various strengths (when it comes to the biological reality, we may need to seperate group of excitatory and inhibitory neurons, but when we're modeling neural circuits, such details become less importants). Initially, the connectivity and synaptic strengths of the network are chosen randomly.

Now the problem is the definition of the output of the networks. The technical choice is to define it as a linear combination of neuronal activities. Formally, if at time t, we assemble the firing rates of all the neurons into a column vector $\mathbf{r}(t)$ and the weights connecting these neurons to the output is described by the column vector $\mathbf{w}(t)$, then the network output is

$$z(t) = \mathbf{w}^\top \mathbf{r}(t).$$

Schematically, this output will be stored in a readout unit and its value may be fed back into the network (it depends on the concrete structure). Though thus far the network output is only a scalar, by using several differents decoding vectors $\mathbf{w}$ we can easily define a multidimensional readout. This is exactly what D. Sussillo and L.F. Abbott did in their paper when they later asked the network to generate multiple outputs; however this point will not covered by the report.

Now that we have defined z as the output of the network, the goal of the algorithm is to set $z = f$ for somme target function $f : t \mapsto f(t)$. Either z is generated in the abscence of any input or it may depend on inputs of the network in a specific way. In the scope of this report, only the first case will be discussed.

Finally, as shown in Figure 1, three different network structures are employed in the article.