Modeling Motor Neuron Timing Using a Recurrent Neural Network

Emily Whyms

July 25, 2025

Abstract

Introduction

Like circuits, your brain passes information through electrical signal. To pass these signals, you brain contains millions if not billions of neural networks of neurons passing information to one another. These networks can then send signals through your nervous system to your muscles to perform daily tasks such as walking, breathing, and writing. The network I have focused in on is arm movement and the time it takes for the network to signal the muscle. [1]

Methods

The Recurrent Neural Network(RNN) is trained on a motor timing dataset which is then given a set pulse to produce a timed interval once the output reaches the target. The model was run on PyTorch using Matplotlib, pandas, Neurogym, and Numpy. It was run with the basic RNN, LSTM, and GRU as parameters; however, the GRU is known to be a simpler version of an LSTM. The dimention was set to 5 and the hidden layers were set to 81(this has to be able to be square rooted since the eigenvalues can only be taken from the tensors if the matrices are squares).

A RNN is much like a feedforward network in the sense that it takes an input, runs it through a few hidden layers with weights, and spits out an output influenced by those layers. Where it differs is how the hidden layers are used. RNNs will sum up their data in every layer since they like sharing their information and have a common weight. This weight can change as the RNN is fed more data based on previous input data. The RNNs back propagation is what allows it to be a better learner since it does some stuff... yah! *(will get back to this) Starting with the basic RNN, it tends to do well with "short-term memory" like getting the next word in a sentence. Where it starts to struggle is finding a word that was hinted at 3 or 4 sentences ago. That's why long-short-term memory(LSTM)s and gated recurrent unit(GRU)s came in. They use states to store long-term memory and can call back to that data. GRUs are known to be a simpler version of an LSTM, having 2 gates instead of 3 and requiring fewer parameters to run.

Results

When running the RNN as a basic RNN, its loss function was very different from the LSTM and GRU plots because of its sudden drop in losses within the first 50 epochs. It seemed to reach its average loss at 350 epochs much like the GRU. In comparison, the LSTM's loss had more of an exponential decay curve within the first 100 epochs and became steady at around 600 epochs. All three parameters were able to average 1.0 accuracy, since their losses stayed below 0.1 after the first few epochs.

whatever order I introduced them should be the order I talk about their loss graphs?

weights and how it changes? because rnn weight is different from linear weight. Quick explanation for future me: rnn weight is the input, hidden layers, and hidden layer dimensions. Linear weight is going to be the output with the hidden layers and dimensions. yes they are very different, the dictionary made the linear weight 2 matrices and the rnn weight a stupidly giant stand alone matrix.

Discussion

RNNs are still usable and quite useful even though there are other learning models such as GPT. In this case a simple RNN had a steeper learning curve when it came to being trained on the dataset. *(umm i forgot what this means so will do this *later*)

The RNN is modeling the time it takes for a neuron to respond since the physical reaction was too slow to (gather brain fast data) *(stupid way of saying, so find a better option) The original research was testing monkey's movement and reaching behaviors? *(fact check that)

I don't think there was much in the way of predicting what was going to happen, it was more of a learning from what it produced and how it worked based on its dictionaries.

There were a lot of limitations, one of the main ones is time, we can never have enough of that. The other thing that I was hoping on doing was to use a real dataset from DANDI. However, the real datasets come as nwb files which needs to be preprocessed in a way I am not capable of yet. One of the biggest hold-backs that made me take so long to do some basic calculations and making a RNN to train was not knowing the math and only grasping at the basic concepts that I should have know like the back of my hand. I also regret not doing more research on different datasets outside of motor networks since there might have been a better chance of getting real data working and it being a smoother journey since there was a lot of trying to find papers and datasets and not as much reading as I had hoped.

The next step is getting more than just the weights, such as graphing the biases. Looking at the rest of the dataset[1] and training a model on the non-motor sides of the data. To work out real data and learning how to preprocess more difficult datasets would allow a better range of data to choose from in training the next model.

References

References

[1] Jing Wang et al. "Flexible timing by temporal scaling of cortical responses". en. In: *Nat. Neurosci.* 21.1 (Jan. 2018), pp. 102–110.