

Investigation into STDP Networks

Izhikevich's proposition that a massively connected neural network implementing a synaptic weight changing algorithm dependent on the timing of the signal was highly interesting in terms of what such a network could be possible of doing. He proposes that Spike Timing Dependent Plasticity (STDP) could create synfire chains which can be used for storage of information.

I attempted to further investigate the details of the network, and how it could be used to learn. Through modification of Izhikevich's neural network, several findings surfaced that may aid future research in this area.

The goal in this investigation was to demonstrate that the STDP network can use the synfire chains as a result of the STDP synaptic rule to learn in an unsupervised environment. Since Izhikevich's proposition was that the consistent firings of certain neurons in such a network would stimulate a chain of neurons so that the activity of the particular chain, instead of a particular neuron, would indicate differences in the stimulus, our test here would be to show that when presented with a stimulus, the synfire chains activated are different from those of a similar stimulus.

Trial Conditions

In order to test the hypothesis, several methods for network stimulation were tried. In the original code, Izhikevich sets the input of a random neuron every millisecond to 20, which by itself is not enough to trigger the neuron to fire. This serves to simulate thalamic input. In these experiments, that random input every millisecond serves as “noise”, so as to simulate how the network might function in presence of many other unrelated things going on, such as the background noise in a party while attending to a conversation.

The network itself consists of 800 excitatory neurons and 200 inhibitory neurons. The neurons start with synapses from 1 ms in length to 20 ms in length. The distribution of these synaptic connections are entirely even. As the STDP rule applies to the excitatory neurons (inhibitory neurons are not affected by the STDP rules in Izhikevich's code), these distributions change.

Because it is unknown how long it takes before the distributions reach stability, only guesses based on observations of firing rates are available. But from my observations, the first 100 seconds sees the general activation of the network start high (~7%), then drop quickly to very low (1%), and slowly climb. This climbing peaks at around 200 seconds at around 11%, then gradually falls back to a range of 8-9%. This seems to remain for the duration of the 24 hour simulation that Izhikevich originally had the network undergo. However, to cut down on computation time, I chose the 10 minute mark as the time at which the network has mostly stabilized. It could well be that further simulation times would change the strengths of synfire chains, but it is unlikely to cause a dramatic shift in the nature of the network itself.

Also not known was whether or not the stimuli had to be present at the time the synaptic distributions in the network were stabilizing in order for the network to learn, or if the network could learn patterns after the network has already stabilized. For sake of guaranteeing that the network would learn, I chose to present the stimuli at all times since simulation start. It would be simple to find out the alternative,

once the chosen condition had been established.

Then in order to determine how to present the stimulus, I took a number of trials. In all trials, the stimuli were based on “words”, which are 10 randomly selected excitatory neurons with inputs set to 30 (solely sufficient to fire the neurons, given they were at resting potential) at the same time, and “songs”, which are sequences of different words with time delays in between word presentations. Seconds separate iterations in the simulation, and were used as separations for training instances.

In the first trials, a single word was presented constantly (every millisecond) with no noise. The resulting network activations were just about bare. Only the 10 neurons in the word itself fired, and no further activity took place for the majority of the time. Occasionally, neural dynamics would lead to a random neuron firing; even less often was when one of those coincided with enough other neurons firing to create a large volume of firings, which quickly died out with only the word neurons active afterward.

That the network for the most part did not respond at all was indicative that constant presentation is not the right way to do things. But to be sure, I also tried constant presentation of two words: word A for first 500ms, word B for next 500ms. The results were not much different. While the network did show some activity when the words changed, there were no regular sustaining activation patterns.

I had also tried the constant presentation in presence of noise, but the result was that the network pretty much behaved as if the word stimuli were not there at all. At this point, I had no way of telling if there were consistent activation patterns.

At this point, I reasoned that first, constantly presenting a stimulus every millisecond makes little sense in the framework of STDP networks. If a neuron fires constantly, there would be no need for timing differences later on in the chain to see if the spikes from that neuron came before or after the activation. Therefore, a different scheme of stimulation was swapped in. Instead of constantly presenting a word, a “song” is created by stringing different words together, to be activated only once and at a specific time. This song is then presented in the presence of noise, since I suspected that noise was vital to the establishment of the synfire chains.

This was tried with both inter-word intervals below 20ms and above 20ms. However, because noise was now a part of the activation patterns, spotting consistent patterns that would indicate learning became humanly impossible. A better method was needed to find out if any neurons were consistently activating with the stimuli.

Test for Patterns

Because the song was presented at every second, the timings for neuron firings are relative to the beginning of the second. Hence, to say that there is a pattern for a particular song's presentation, the neuron in question must fire and fire at the right time. In order to test if a particular neuron fired consistently at a particular time, a method was needed to keep track of which neurons fired over the course of the training. Then a test is needed to see if this neuron is significantly firing more than others. Finally, the neuron must be a part of the synfire chain of the chain that results from the song being presented, not the stimuli themselves nor any other chains that might occur from noise.

Essentially, we needed to find three things. First, $P(n,t)$ where P is whether or not (n,t) pair fired significantly during training such that

$$P(n, t) = \sum_i a_i(n, t) > \gamma$$

where n is the neuron, t is the time since beginning of second when that neuron fired, a_i is whether or not that neuron n fired at time t at second i , and γ is some threshold for significance.

Second, we need to find $Q(n, t)$ where Q is whether or not neuron n fired at time t when presented with the song without noise.

Lastly, we need to find $R(n, t)$ where R is whether or not neuron n firing at time t is because of activation of stimuli.

So then the significance of neuron n firing at time t would be

$$S(n, t) = P(n, t) \wedge Q(n, t) \wedge \neg R(n, t)$$

for all (n, t) pairs. Those pairs for which $S(n, t)$ is true I called Predictor Neurons.

The threshold γ in this case was determined graphically. The reason that I used a predetermined threshold instead of a statistical procedure here is that because of the inclusion of the input neurons, the mean calculation would be extremely skewed. While removal of those values from $P(n, t)$ would give me a better data set to work with, because of the timing issues, those values are often spread over 3 or 4 pairs. Other more sophisticated measures were considered, but implementation time and run time increases were deemed not worth it for this investigation.

With the Predictor Neurons, I could theoretically tell how much the network considered the input to be similar to what it was trained on simply by how many of those (n, t) pairs fired out of those predicted, giving me a simple fast estimation on how much like the training the input was.

Results

This was the test applied to four conditions: presentation of the first word from a song and presentation of the entire song with or without noise (inter-word intervals were kept below 20ms). The results for the most part confirmed the hypothesis. In the majority of cases, the presentation of the entire song scored higher than presentation of a single word (15 of 20 fired versus 10 of 20 without noise, 5 of 22 fired vs 0 of 22 fired with noise). However, one would expect the non-noisy full song presentation to score a perfect 22 out of 22 predictor neurons. This was never the case in the simulations I ran. I suspect this is because of the nature of the code's mechanism to determine if the neuron should fire. Because of the internal dynamics of the neuron, a neuron with enough stimulus to fire may not fire. This was most clearly seen in the data for $P(n, t)$. Since $P(n, t)$ included the neurons that would fire just from direct input, one would expect there to be only one (n, t) pair for each of the 10 neurons at time t when the word was presented. However, the data shows $P(n, t)$ for the 10 neurons for $t, t+1, t+2, t+3$, and sometimes $t+4$. This would indicate that the mechanism to fire in Izhikevich's network not to be as precise as our prediction model would like.

By graphically examining the firing pairs vs Predictor Neurons, I found that in the noisy condition, many more than were reported to have fired fired within 1ms of predicted time t . Hence to improve the robustness of the Predictor Neuron test, some sort of temporal window should be allowed. One idea is to repeat the test $Q(n, t)$ and take multiple trials of presentation of song without noise as a single element. Hence:

$$Q(n, t) = Q_1(n, t) \vee Q_2(n, t) \vee \dots \vee Q_k(n, t)$$

or,

$$Q(n, t) = q(n, t-k) \vee q(n, t-k+1) \vee \dots \vee q(n, t+k)$$

where q is not Q to avoid recursion.

Thus even with a noisy environment, the STDP network can pick out stimuli that are similar (identical in this instance) to that on which it had been trained without supervision. There are still several factors that can be manipulated to possibly increase the strength of the network and the strength of the Predictor Neurons test. Allowing the network to run for longer time could possibly strengthen the synfire chains existent in the network. Relaxing the timing for a fired neuron in the test could increase the power of the tests, but also allow the possibility for false positives.

At this point, it would be interesting to try if the network is capable of learning a song after the initial period of stabilization had completed. If so, how long does it take for the network to respond consistently? Another possibility is to vary the onset of the stimuli in the testing conditions so that the training is done with words presented at a certain time during the second, but the testing is done at a completely different time within the second. In order to use the Predictor Neuron test, one would have to use (n, t) pairs such that t is relative to the onset of the presentation, instead of the beginning of the second. Lastly, one could modify the song such that the grammar of the song changes, such that A-B-C and A-C-B are both resented to the network, then observing whether or not the network is capable of accurately differentiating the two by employing different Predictor Neurons.