Unsupervised Neuromorphic Learning Using Spike Timing Dependent Plasticity

Bianca Bologa
bbologa@unm.edu
University of New Mexico, Albuquerque, New Mexico
Boise State University, Boise, Idaho

Memristors are resistive circuit elements that allow changes in resistance using voltage pulses. The neuromorphic computing group at Boise State University has been building memristors and using them in neural networks. Learning in the network is done in an unsupervised manner using spike timing dependent plasticity (STDP). Unsupervised learning occurs without explicit signals controlling the neuron's learning and without any feedback from the system. Through STDP, the neuron becomes sensitive to repeating spike patterns in a spike train. The synaptic weights are increased on early firing afferents and the postsynaptic latencies are decreased. We are currently evaluating how the learning algorithm works. Simulations in Matlab focusing on unsupervised learning using STDP will provide a good application for memristors. Since STDP might be the mechanism through which the synapses in the brain operate, modeling it using memristors is a good choice for neural networks.

Introduction

The neuromorphic computing group at Boise State University has built memristors for implementation into neural networks. In these neural networks, memristors will be used to represent the weight in the neurons' synapses. The weights will be changed and adjusted using STDP, which works by increasing the weights through long-term potentiation (LTP), and decreasing them through long-term depression (LTD). LTP occurs when the presynaptic neuron fires shortly before the postsynaptic one. LTD occurs when the opposite is true – when the postsynaptic neuron fires first. Memristors are a good choice for synaptic weights because their resistance is changed using voltage pulses, which will in our case be the signals between afferents (our inputs) and the neurons.

It has been shown that a leaky integrate-and-fire (LIF) neuron using STDP can detect a spike pattern hidden in a spike train. Through STDP, it becomes sensitive to the successive coincidences of the pattern. STDP concentrates weights on afferents that consistently fire early; it also reduces postsynaptic latencies. It is successful even in the presence of degradations, such as lowered pattern frequency, reduced initial weights and a lowered proportion of afferents in patterns as well as the presence of jitter and spike deletion, and it is therefore a robust learning mechanism. The reason why it works is that connections from afferents that helped the neuron reach its firing threshold are strengthened every time. Each time the neuron fires to the pattern, it makes it more likely that it will fire again, earlier, the next time the pattern occurs. This way, the neuron becomes selective to the pattern.

The learning is, however, not limited to a single neuron learning a single pattern. It is possible to have multiple neurons trying to detect multiple spike patterns in the spike train, which is indeed a more realistic scenario[2]. Here, several neurons are listening to a number of afferents. They have initial weights between 0 and 1. As soon as a neuron fires, lateral inhibition occurs; the neuron sends an inhibitory postsynaptic potential (IPSP) to the other neurons, making them less likely to fire. This is known as a one-winner-take-all mechanism. Therefore, the other neurons can either learn a different pattern or a smaller, later part of the same pattern as the first one.

Furthermore, the learning will be entirely unsupervised. There is no signal for the first afferent in the pattern to which the neuron must become sensitive. It just learns the pattern by being exposed to it multiple times; it becomes more and more sensitive to it, reducing the time necessary for it to fire to the pattern. There is also no explicit feedback from the system.

Results

If the neuron learns successfully, there are three stages to the learning. In the first stage, the neuron is not selective; it tends to fire both to the pattern and outside of it. However, due to STDP, the neurons start becoming selective to the pattern. Here, the false alarms (firing outside the pattern) are reduced in number, usually disappearing entirely. Finally, the learning is completed, with the latencies settling to a small, fairly stable value. The threshold required for a firing event is reached quickly. There will be no more false alarms. Figure 1 illustrates these three stages.

It is, however, also possible that the neurons do not learn. They fire outside the pattern, reducing the weights until the threshold cannot be reached any more. Basically, the neuron fails and dies. It is a convention that the latency is zero if the neuron generates a false alarm; it does not mean that the neuron learned really well. It is practically impossible that the neuron learns so well that this is the reason the latency is zero.

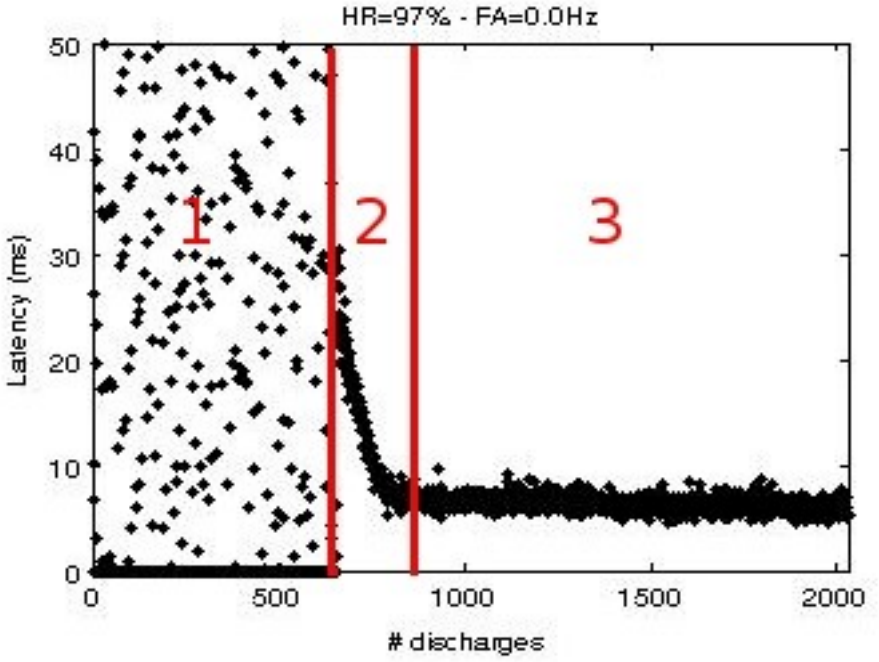

Figure 1. Stages of learning for one successful neuron. The postsynaptic latency is shown. For stage 1, during the first 650 discharges, the neuron is not selective, firing both inside and outside the pattern without much discrimination. For stage 2 over the following 100 discharges selectivity starts. For the remaining discharges, forming stage 3, the latency has stabilized at a low value, here at less than 10 milliseconds.

As previously stated, lateral inhibition between neurons occurs when there is more than one neuron. First, one neuron fires at the start of a pattern after finding it. If a second neuron tries to fire for the same pattern, it cannot do so at its start; the first neuron will have sent an inhibition signal, prohibiting the second neuron from reaching the threshold for some time. This duration is determined by the strength of the inhibition, being longer for a higher IPSP amplitude. Therefore, the second neuron will have a higher postsynaptic latency. It can fire in response to a different pattern, if one is available, or fire to later parts of the first one. Since that would be a shorter pattern and therefore harder to learn, it is easier for the neuron to learn a new pattern. Figure 2 shows three neurons trying to learn the same pattern with all three succeeding. Figure 3 shows a scenario with three neurons attempting to learn three different patterns, with varying success.
Both figures show hit rates and false alarms at the top. These are used to determine the success of the neuron in learning. In my simulations, the criteria for successful learning in a neuron were set as follows: the neuron had to have both a hit rate above 90 percent and a false alarm value below 1 hertz. Changes in these values can be investigated.

From the simulations run for this project it resulted that the number of afferents drastically impacted the results of the learning process. In Figure 3, 2000 afferents are being used. It is obvious that both the second and third neuron learned. However, Figure 4 shows a scenario typical for a lower number of afferents. The learning is inferior to the one with 2000 afferents.

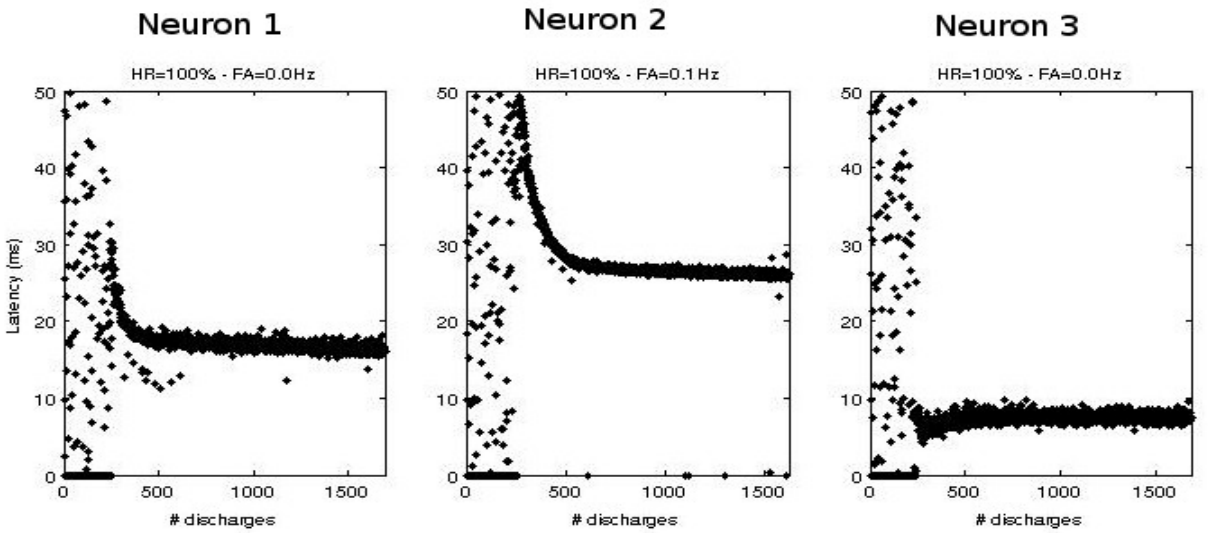

Figure 2. Latencies of three neurons successfully learning the same pattern. The third neuron learns first, inhibiting the first and second neuron. Since the second neuron settles in the highest latency, it is likely that it started learning last, being inhibited by both other neurons.

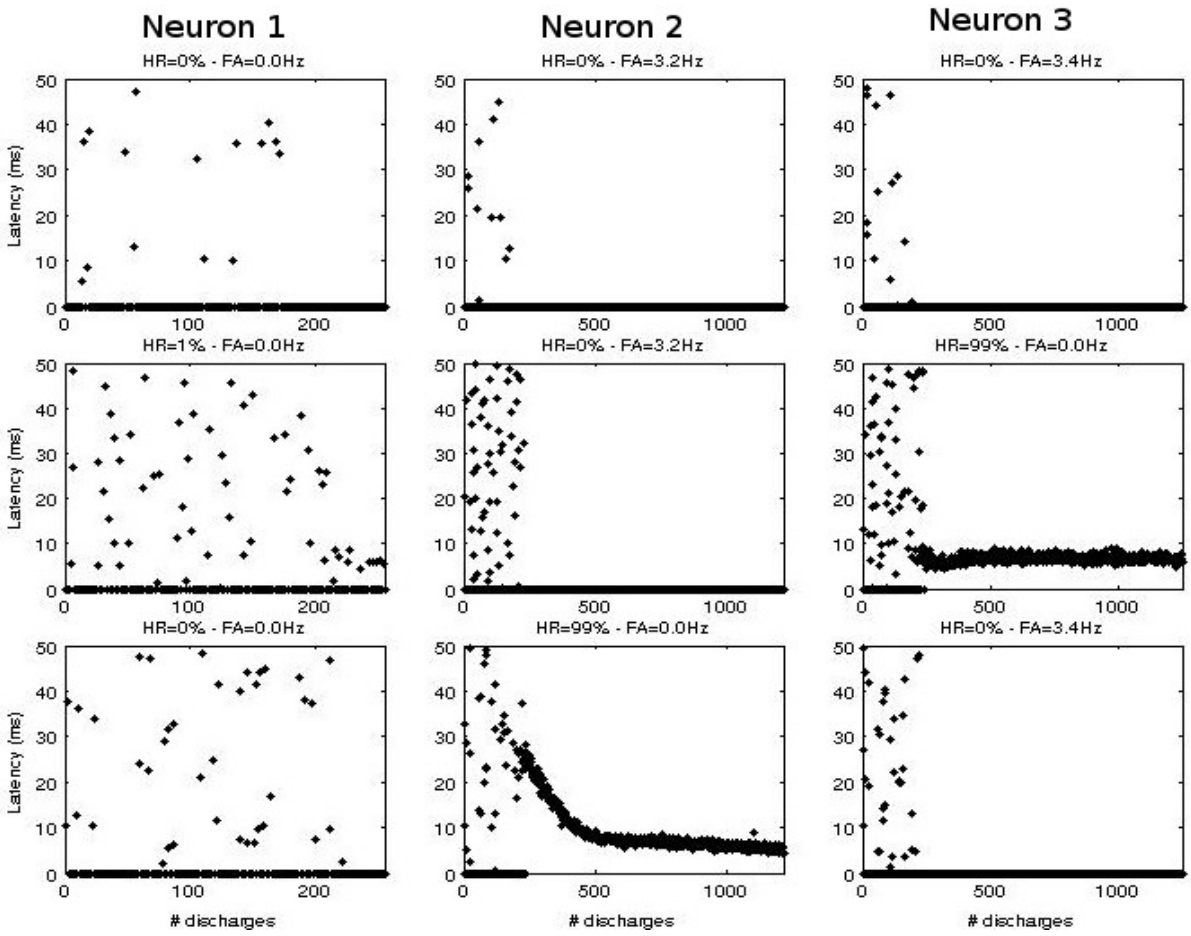

Figure 3. Three neurons attempting to learn the same pattern. The first neuron fires outside the pattern until it can no longer reach the threshold for firing. The plot shows how most of its latencies are zero. Both the second and third neurons learn; the latter learns faster, but both settle to a latency below 10 milliseconds.

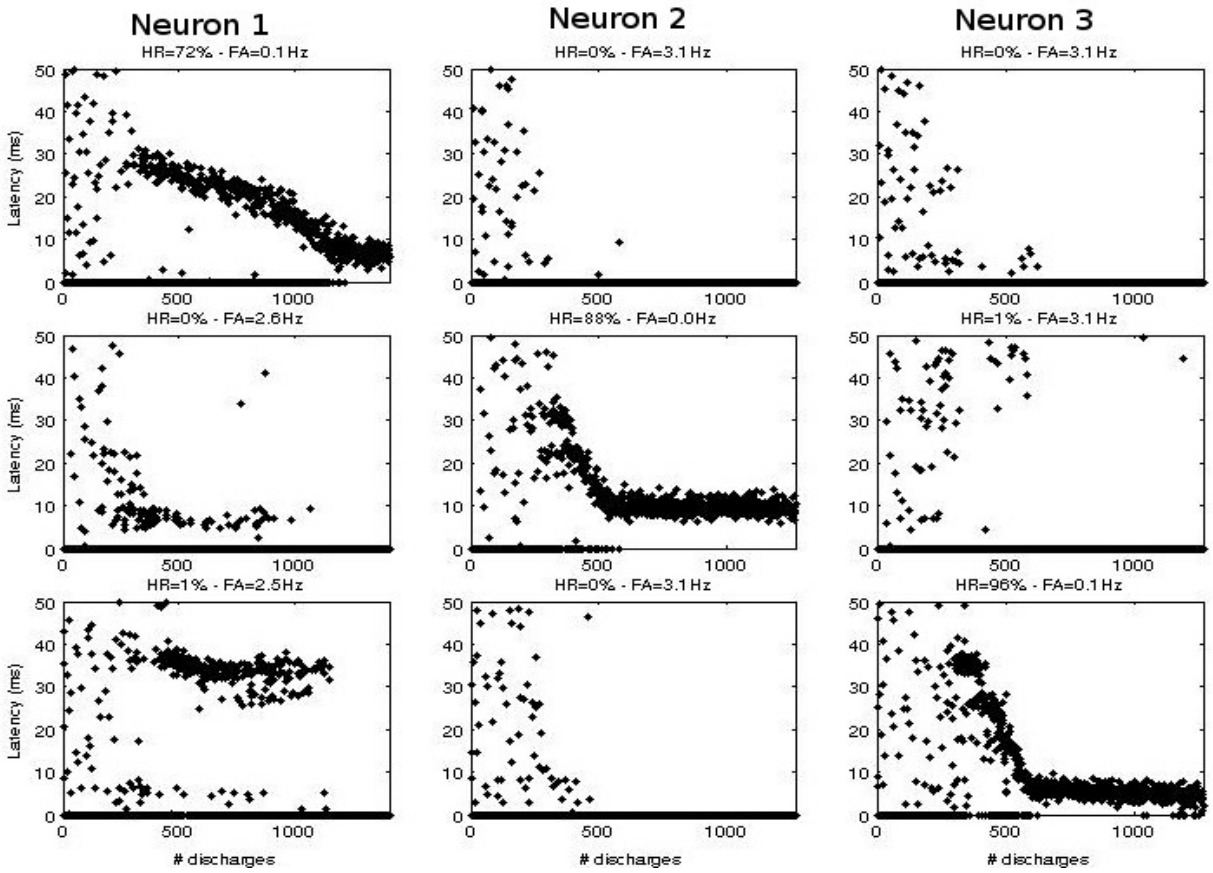

Figure 4. Three neurons attempting to learn three different patterns with only 400 afferents. Learning, as determined by the criteria mentioned above, only occurs in the third neuron. The first neuron tries to learn two different patterns. This is almost never successful; it only happens when the two patterns are very similar.

Discussion

Since the purpose of the memristors is to represent the weights in the neural network, part of the goal of the simulations was to analyze the change in the weights and their distribution. The weights were recorded and plotted at the end of the simulation. Figure 5 shows histograms of the weights of two neurons: one that has learned and one that has failed to do so. It is interesting to note the difference in their distribution: if the neuron is successful, most of its weights are low (between 0 and 0.1), but there are many between 0.9 and 1. There are very few between 0.1 and 0.9. This makes the final distribution bimodal. The reason might, however, be the fact that in these simulations the weights are purely additive. When the neuron fails, the weights follow a distribution similar to those in Figure 5 on the right panel. Figure 6 also shows how weight values are scattered over all values between 0 and 1 for neurons failing to learn, even though a significant part of them is still low, between 0 and 0.1.

I also previously mentioned that, with fewer afferents, success in learning is reduced as well. Figure 4 shows results with as few as 400 afferents: there is not enough learning, with neurons frequently firing outside the pattern. However, the goal of the simulations was to modify the code to allow good results with very few afferents, no more than 20. Therefore three of the degradations researched in previous work were eliminated in an attempt to improve the learning: pattern frequency was increased, while jitter and spike deletion were eliminated. The results improved, as expected. Over 20 runs of the simulation, on average, 0.5 neurons learned with degradations; otherwise, without degradations, the average was 1.9. This was with 500 afferents, which is relatively low, but still far from low enough. It does, however, suggest that degradation should be given some thought during the implementation of the neural network with memristors.

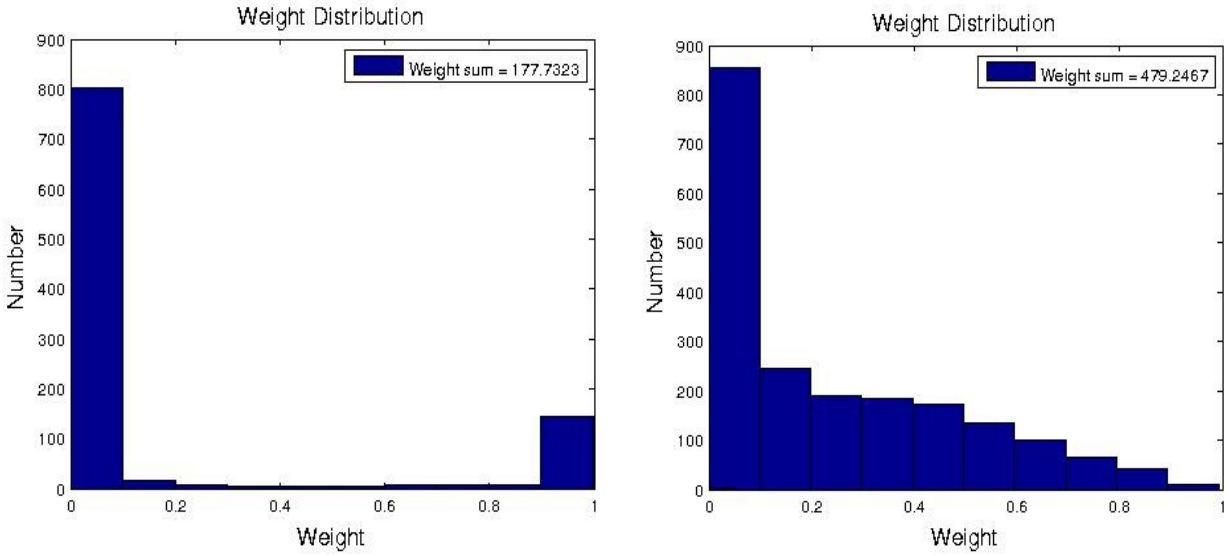

Figure 5. The left panel shows a histogram of the final weight distribution for a neuron that has learned a pattern successfully. The distribution is bimodal. The right panel shows the result for a neuron that fails to learn.

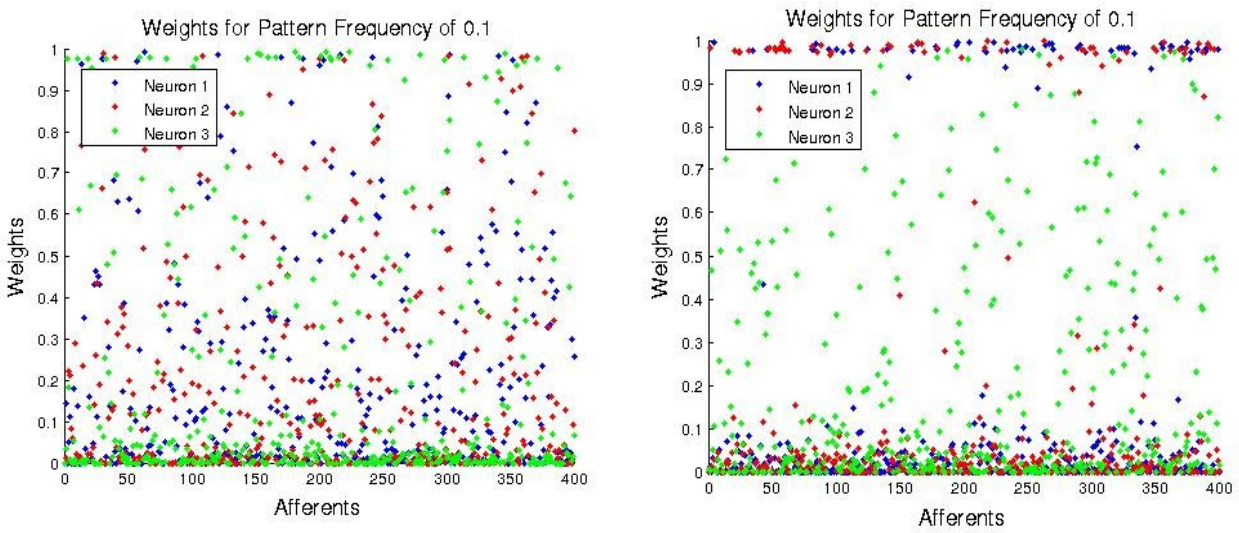

Figure 6. The left panel shows weights for 400 afferents, with no neurons learning. Weights are very scattered for all three neurons. The right panel represents the weights for two neurons learning highly. Only the third neuron, whose weights are shown as green dots, did not learn and therefore has scattered weights.

The criteria for successful learning were set at above 90 percent hit rate and false alarms below 1 hertz. However, when running simulations with 10 afferents, interesting results were found; both hit rates and false alarms were unrealistically high, with hit rates above 350 percent and false alarms of around 70 hertz. This was quite puzzling for a while, until I found that the reason was that there were many more discharges. There were between 30 and 50 times more, compared to the simulations with 1500 afferents. An example of the results with 10 afferents is shown in Figure 7. There are two different possibilities to improve these results: the firings could be inhibited more, or the algorithm to find the results could be modified to adapt to this high number of discharges and yield more realistic results. More research needs to be done to gain better insight into this problem.

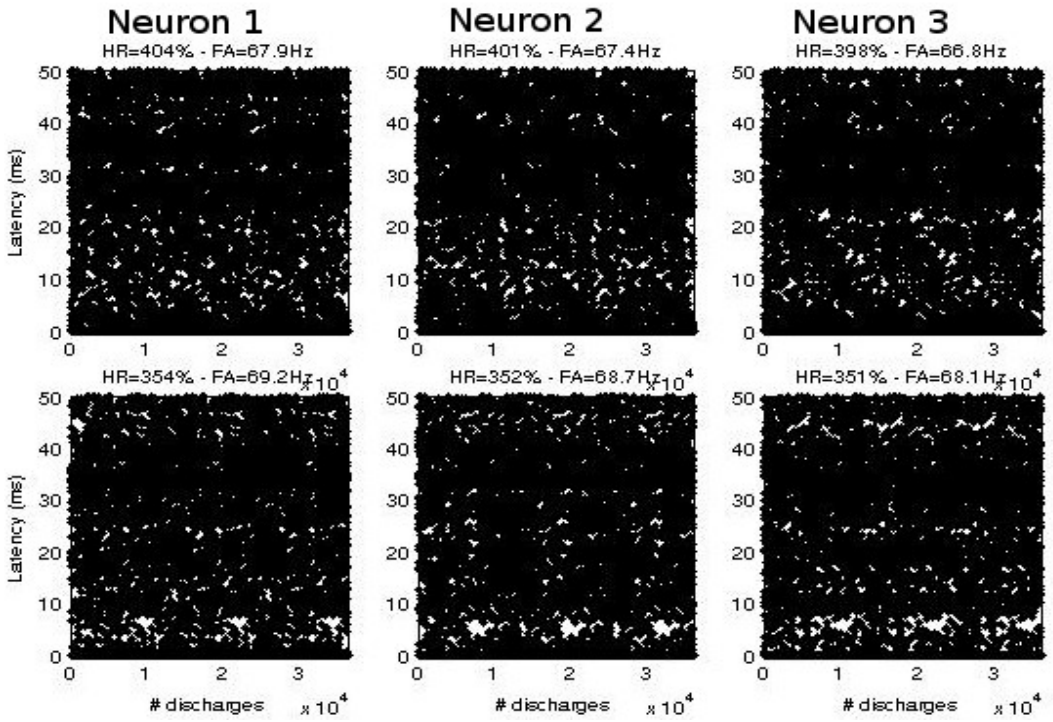

Figure 7. Example of result for a simulation using ten afferents. There is a very high number of discharges. Both hit rates and false alarms are significantly too high, with the highest hit rate of 404% and the highest false alarm value of 69.2 Hz.

Conclusion

STDP is a good mechanism for neural learning. Using STDP, the neurons learn a repeating pattern in an unsupervised fashion. STDP leads to high performance and efficiency and definitely deserves more attention since it might be the way physical neurons learn in the brain.

In the future, more work needs to be done to adapt the current code for successful learning with very few afferents. The problem of the high hit rates and false alarms needs to be investigated. The code will have to be optimized for efficiency to allow many simulation runs to be completed fairly quickly.

Acknowledgements

I would like to thank Dr. Elisa Barney Smith for supporting my research and for making comments and helping me edit this report. I would also like to thank the DREU program for giving me the opportunity to participate in this research.

References

[1] Masquelier T, Thorpe SJ (2007) Unsupervised learning of visual features through spike timing dependent plasticity. PLoS Comput Biol 3(2): e31. doi:10.1371/journal.
pcbi.0030031

[2] Masquelier T, Guyonneau R, Thorpe SJ (2008) Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains. PLoS ONE 3(1): e1377. doi:10.1371/journal.pone.0001377

[3] Masquelier T, Guyonneau R, Thorpe SJ (2009) Competitive STDP-Based Spike Learning. Neural Computation 21,1259-1276