Assignment 2: Spike time dependent plasticity

Neural Computation 2008-2009. Mark van Rossum

27th March 2009

Practical info

You will find that some questions are quite open-ended. A particularly well-researched answer can receive additional points, but core-dumping (just writing down all you can think of) does not. Ideally you substantiate your explanations, for instance by additional simulations. Plots should include axes labels and units (either on the plot, or mentioned in the text), see my web page link. There might be a to be determined normalization factor between the number of points scored and the resulting percentage mark.

It should not be necessary to consult scientific literature. If you do use additional literature, cite it.

Copying results is absolutely not allowed and can lead to severe punishment. It's OK to ask for help from your friends. However, this help must not extend to copying code, results, or written text that your friend has written, or that you and your friend have written together. I assess you on the basis of what you are able to do by yourself. It's OK to help a friend. However, this help must not extend to providing your friend with code or written text. If you are found to have done so, a penalty will be assessed against you as well.

Deadline will be announced via email and the website. Hand in a paper copy to ITO, but if you are out of town an email to me is ok (pdf or postscript). Late policies are stated at `http://www.inf.ed.ac.uk/teaching/years/msc/courseguide08.html#exam`.

Framework of simulation

In this exercise we study spike timing dependent plasticity (STDP) in a population of integrate and fire neurons. We will model a single layer of neurons with $n = 5$ neurons. Each neuron receives input from other neurons in the network through lateral connections, and in addition each neuron receives its own external input. We use the term 'spikes' when refering to the output spikes of the neurons in the network; we use the term 'events' for the spikes of the inputs (although these could in principle be spikes from another set of neurons).

Make the inputs periodically active, so that input to neuron 1 has an event at 10 ms, input to neuron 2 at 20 ms, input to neuron 3 at 30ms, etc. This is followed by silence. The pattern repeats with a period of 100 ms. So input 1 is active again at 110 ms. You can use the 'mod' function to implement this:

`relativetime = mod(time, period)` returns the time relative to the period. And `if(mod(relativetime, 10)==0)` can be used to check if the time is an exact multiple of 10 ms. Check your code by plotting the inputs as a function of time.

The neurons are modelled as leaky integrate-and-fire neurons (without adaptation). The input events and the spikes from other neurons give rise to synaptic input currents, which for simplicity are modelled as currents (not as conductances). The synaptic currents rise instantaneously and decay as single exponentials with a synaptic time constant $\tau_{syn} = 5ms$. Other parameters: $\tau_m = 20ms$, simulation timestep 0.1 ms, $V_{thr} = -50mV$, $V_{reset} = V_{rest} = -70mV$. Take $R_m = 1\Omega$ (this is far from realistic but it is easier and scales out anyway). This means that the currents are measured in mA.

Question 1 (5 points): First we need to find a good initial value for the synaptic weights. Calculate analytically the EPSC amplitude required for a single input to fire the cell from rest and check in simulation.

Use an amplitude for the inputs of 150mA, these input synapses are not plastic.

Connect the neurons with the lateral weight matrix. Initialise all the weights in the matrix to 20mA, but always exclude self-connections.

For the rest of this practical we need raster plots. For instance, one can plot on the x-axis the time since pattern onset, and on the y-axis the pattern repetition number. The spikes are indicated by crosses. One can plot spikes from multiple neurons in one plot using code like,

```
for icell=1:n
sptimes = find(spikes(icell,:));
relsptimes=mod(sptimes, period/dt)*dt
spy = icell + 0.9*sptimes/ndt;
plot(relsptimes, spy, 'x')
hold on
end
```

Question 2 (5 points): Simulate for 1 second. Plot the spikes in a raster plot. Explain the various aspects of the result.

Plasticity

Next, the connections between the neurons are made plastic. With w_{ij} we denote the synapse from neuron i to neuron j. Here we implement the plasticity as follows: every time neuron j spikes, the synapses onto this neuron are increased with an amount

$$\Delta w_{ij} = \frac{\alpha w_{max}}{\tau_{plast}} \exp[-|t_i - t_{now}|/\tau_{plast}]$$

where t_i is the last time that neuron i was active, and t_{now} is the current time. Parameters: $\alpha = 3$ms, $\tau_{plast} = 20ms$. Similarly, depression in the reciprocal synapse occurs according to

$$\Delta w_{ji} = -\frac{\alpha w_{max}}{\tau_{plast}} \exp[-|t_i - t_{now}|/\tau_{plast}]$$

These rules can be compactly written in Matlab by the following scheme: 1) track the last time each input was active in an array with n elements. 2) When you update the membrane potential, use the 'find' command to find which neurons reached threshold. Whenever one or more neurons spiked, potentiate and depress the weights. When you keep the neurons' membrane potential in an array, you can use that statements such as `(vmem>vthr)` to return an array of 0s and 1s.

Finally, after changing the weights, put hard bounds on the synaptic weights so that $0 \leq w_{ij} \leq w_{max}$, $w_{max} = 150mA$.

Question 3 (10 points): Simulate the network until the weights reach a stable value. Plot the spikes and explain the results.
Also plot or describe the resulting lateral weight matrix and interpret it.

Next, we study if this network can be used to store and recall the input train.

Question 4 (5 points): Train the network as in Question 3 and then just give the first neuron an input. Plot and describe the resulting activity in the network.

Question 5 (10 points): Explore ways to store the input pattern more precisely in terms of number of spikes and timing.