Modelling Neurons

Due at 4:00pm on Monday 28 January 2019

What you need to get

- `YOU_a1.ipynb`: a Python notebook (hereafter called "the notebook")
- `a1_solutions.pyc`: compiled Python 3 module that contains hidden implementations of the `LIFNeuron` class and `SpikingNetwork` class.

What you need to know

The notebook includes some helper functions that you may use. These include: `PlotSpikeRaster`, and `GenerateSpikeTrain`. See each function's documentation for more details on how to use it.

What to do

1. **LIF Neuron Model** [10 marks total]

   In this exercise, you will finish the implementation of a leaky integrate-and-fire (LIF) neuron. The notebook includes an incomplete version of the class `LIFNeuron`. The class models the dynamics of the LIF neuron, including its membrane potential (`v`), post-synaptic (input) current (`s`), and spike times (`spikes`). The class includes the functions `Slope` and `Step` that, together, update the state of the neuron using a single step of Euler's method.

   The class also has a function called `SpikesBetween`, which counts the number of times the neuron spiked between a specified start and end time.

   (a) [1 mark] Complete the function `ReceiveSpikes`. It accumulates the weighted sum of incoming spikes during a time step. These incoming spikes influence the neuron's input current in the `Step` function.

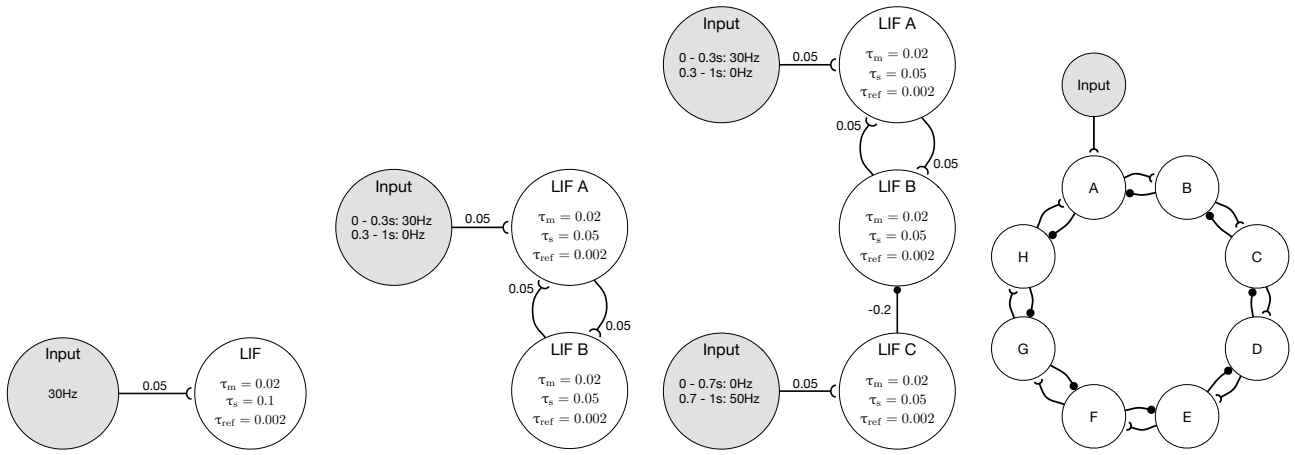
   (b) [2 marks] Complete the function `Slope` as specified by its documentation. It evaluates the right-hand side of the differential equations that govern the dynamics of the LIF neuron, and saves those slopes in its member variables `dvdt` and `dsdt`.

   (c) [7 marks] Complete the function `Step`, which advances the neuron's state one time step. You should use Euler's method, along with the previously-calculated slopes, to update the membrane potential and the input current. Remember that when you detect a spike, you should record the spike time and reset the membrane potential to zero[1]. Don't forget about the refractory period, during which the membrane potential is fixed at zero.

   While running Euler's method, a spike occurs if the membrane potential reaches the threshold value of 1. Most likely, the step will produce a `v`-value greater than 1. You should use linear interpolation to estimate the time at which `v` was exactly 1, and use that interpolated time as the spike time.

2. **Spiking Network Model** [5 marks total]

[1] Or you can set the membrane potential to 1, since it will be set to zero in the next time step.



(a) One LIF (b) Two LIF (c) Three LIF: Inhibition (d) Ring Oscillator

Figure 1: Network Models

Now that you've implemented the LIF neuron model, you can build a network out of them. In this task, you will complete the class `SpikingNetwork`.

Complete the function `Simulate`. As its documentation explains, this function simulates the operation of the network, including all of the neurons in the network, for a prescribed amount of time. The supplied code includes a loop over time steps. For each time step, each neuron has to be updated, and spikes have to be tabulated for the next time step.

3. **Experiments with Spiking Networks** [25 marks total]

   For these exercises, you will build and simulate various neural networks. To help you, I have supplied you with another type of neuron class called `InputNeuron`. On the outside, it looks like a LIF neuron. However, it's not a real neuron; its only job is to deliver spikes, which is useful to activate the the other (real) neurons in the network.

   For all these simulations, you should use a time step of 1 ms (0.001 s).

   (a) **One LIF Neuron**

   The notebook includes some code that builds a very simple network consisting of one `InputNeuron` feeding into one `LIFNeuron`. The input neuron is set to deliver spikes at around 30 Hz. The input neuron is connected to the LIF neuron with a connection weight of 0.05. Finally, the code simulates the network for 1 second.

   i. [2 marks] Plot the sub-threshold membrane potential of the LIF neuron as a function of time. Note that the `SpikingNetwork` class stores the simulation time steps in a list called `t_history`. Your `LIFNeuron` class should also keep a history of its membrane potential, stored in a list called `v_history`. Make sure you label the axes of your plot.

   ii. [1 mark] Produce a spike-raster plot for the two neurons in the network. You can use the supplied function `PlotSpikeRaster` for this.

   (b) **LIF Firing Rate Curve**

   i. [3 marks] Copy the network from 3a. However, alter the input neuron so that it incrementally ramps up its firing rate, starting with firing at 5 Hz for 2 s, then 10 Hz for the next 2 s, then 15 Hz for the next 2 s, etc. Continue until you reach a firing rate of 100 Hz. You will

find `GenerateSpikeTrain` very useful for this. Connect this input neuron to the LIF neuron with a weight of 0.03. Simulate this network through the entire sequence of input firing rates. If you've set it up properly, you can do this with a single call to `Simulate`.

ii. [3 marks] For each 2-second interval, compute how many spike occurred. You can use `LIFNeuron.SpikesBetween` to help you with this.

iii. [1 mark] Make a plot of input firing rate (on the horizontal axis) vs. LIF firing rate (vertical axis). Just plot one dot per input firing rate (so your plot should have 18 dots on it). Don't forget about labelling the axes.

(c) **Two LIF Neurons**

i. [4 marks] Build a network with **two** `LIFNeuron`s, each projecting its input to the other. That is, neuron A is connected to neuron B with a weight of 0.05, and B is connected to A with a weight of 0.05.

Add an input neuron and connect it to one of the two LIF neurons with a weight of 0.05. Have the input neuron fire at 30 Hz for 0.3 seconds at the start, and then go silent. You can create that input neuron using

```
InputNeuron( GenerateSpikeTrain([30], [0.3]) )
```

This simple network is depicted in Fig. 1(b).

ii. [1 mark] Simulate the network for at least 1 second, and produce a spike-raster plot for the three neurons in the network.

(d) **Three LIF Neurons: Inhibition**

i. [3 marks] Copy your network from question 3c (which included neurons A and B). Now add a third LIF neuron (C), and connect it to one of the original LIF neurons with a connection weight of -0.2; the negative weight makes this an *inhibitory* connection. Also add another input neuron and connect it to C with a weight of 0.05. This input neuron should be dormant for the first 0.7 s, and then fire at 50 Hz from 0.7 s until 1 s. You can create that input neuron using

```
InputNeuron( GenerateSpikeTrain([0, 50], [0.7, 1.]) )
```

This network is shown in Fig 1c.

ii. Simulate the network for at least 1.5 seconds, and produce a spike-raster plot of all 5 neurons in the network.

iii. [2 mark] Comparing your results to those in question 3c, what effect does the addition of neuron C and its negative connection have on the activity of neurons A and B?

(e) **Ring Oscillator**

i. [2 marks] Create a network with 8 LIF neurons; for the sake of this question, let's label them A through H. For all 8 neurons, use a `tau_m` value of 50 ms, and a `tau_s` value of 100 ms. Connect A to B with a weight of 0.2. Likewise, connect B to C with a weight of 0.2, etc. until you finally close the loop by connecting H to A with a weight of 0.2. This is the excitatory ring.

ii. [1 mark] Now, add inhibitory connections between the same pairs of neurons, but running in the opposite direction around the ring. That is, connect B to A with a weight of -0.4, and A to H with a weight of -0.4, etc.

iii. [1 mark] Add an input neuron to stimulate any one of the ring neurons (with a weight of 0.2) for 0.3 seconds with a firing rate of 25 Hz. This full network is shown in Fig. 1(d).

iv. [1 mark] Simulate the network for at least 4 seconds, and produce a spike-raster plot of all 9 neurons in the network. If all is working, you should see each neuron in the ring exhibit a "burst" of spikes in sequence. This excitation travels like a wave around the ring. If this is not what you are seeing, play with `tau_m`, `tau_s`, and the connection weights until you see this wave of activation travel around the ring.

4. **Neural Activation Functions** [14 marks total]

   For the function plots in this question, you may use functions from Python libraries, or you can implement them yourself.

   (a) [2 marks] Calculate the derivative of $ReLU(z)$ with respect to z.

   (b) [6 marks] For each of the functions below, create a plot of the function over the domain $z \in [-6, 6]$. State the values of the domain for which the derivative approaches or equals zero, and the values of the domain where the gradient is maximized.

      i. [2 marks] Standard logistic function, $\sigma(z)$.

      ii. [2 marks] $\tanh(z)$.

      iii. [2 marks] $ReLU(z)$.

      Note: You may use Numpy's `linspace` function to generate regularly-spaced samples in the domain.

   (c) [6 marks] Consider the expression $f(wx + b)$, where w, x and b are all real numbers, and $f : \mathbb{R} \to \mathbb{R}$ is some differentiable activation function.

      i. [2 marks] Calculate $\frac{d}{dx} f(wx + b)$.

      ii. [2 marks] Calculate $\frac{d}{dw} f(wx + b)$.

      iii. [2 marks] Calculate $\frac{d}{db} f(wx + b)$.

What to submit

Your assignment submission should be a single jupyter notebook file, named (`<WatIAM>_a1.ipynb`), where `<WatIAM>` is your UW WatIAM login ID (not your student number). The notebook must include solutions to **all** the questions. Submit this file to Desire2Learn. You do not need to submit any of the modules supplied for the assignment.