9.641 Neural Networks
Problem Set 1

(Due Sep. 21, Thursday before class)

The **integrate-and-fire neuron** is a simple model of spiking behavior that sacrifices biophysical realism for mathematical simplicity.

1. Single neuron model

   First, let's consider an isolated neuron into which we inject a current I_{app}. Below threshold, the membrane potential V obeys the differential equation

   $$C\frac{dV}{dt} = -g_L(V - V_L) + I_{app} \tag{1}$$

   If V reaches a threshold V_θ, then the neuron is said to spike, and V is instantaneously reset to a value of V_0, where $V_0 < V_\theta$.

   (a) Analytically determine the threshold current I_θ (or rheobase) below which the neuron is inactive, and above which the neuron fires repetitively. The sign of I_θ should depend on whether V_θ is above or below V_L.

   (b) Experimentally determine I_θ and compare it to the value you found analytically. In MATLAB, a system $\frac{dy}{dt} = f(y)$ can be simulated by choosing the initial conditions $y(1)$ and then repeatedly performing the Euler integration step $y(t+1) = y(t) + dt\frac{dy}{dt}(t)$.
   Use the following values for your simulations: $V_L = -74mV$, $g_L = 25nS$, $V_\theta = -54mV$, $V_0 = -60mV$, $C = 500pF$. Plot a trace of the membrane potential V, one for I right below and one for I right above I_θ.

   (c) If I_{app} is held constant in time above threshold, the neuron fires action potentials repetitively, as you should have observed in your simulations. Find the relationship between the frequency of firing f and I_{app}.

   (d) Show that f behaves roughly linearly for large I_{app} and can be approximated by

   $$f \approx \frac{\left[I_{app} - g_L(V_{1/2} - V_L)\right]^+}{C(V_\theta - V_0)} \tag{2}$$

   with $V_{1/2} = (V_\theta + V_0)/2$. Explain in words the reason for this linearity. [*Hint:* Use the Taylor series expansion $[log(1+z)]^{-1} \approx 1/z + 1/2$.]
   Plot your results from (c) and (d) together and compare them.

2. Modeling synapses

   A synapse is modeled by a variable conductance g in the postsynaptic neuron. A spike in the presynaptic neuron causes an increase of the conductance according to $g := g + \frac{\alpha}{\tau}$. Between spikes, g decays exponentially: $\frac{dg}{dt} = -\frac{g}{\tau}$. So a synapse is a leaky integrator, counting spikes but forgetting them over time periods longer than τ. The area under the exponential caused by a single spike is given by the parameter α.

   Under certain conditions this can be approximated by $\tau\frac{dx}{dt} + x \approx f$, where x is proportional to g and f is the frequency of incoming spikes.

   Simulate the time course of the conductance of a synapse for $f = 25$Hz for different τ. For what values of τ is this approximation valid? Illustrate your answer with two plots.

3. From synapses to current

   In practice, neurons are a part of networks and receive input currents through synapses instead of an electrode. For a neuron i receiving inputs from neurons j, this can be written as:

$$C_i \frac{dV_i}{dt} = -g_{Li}(V_i - V_L) - \sum_j g_{ij}(V_i - V_{ij}) \tag{3}$$

   Show that equation (3) can be simplified to the form of equation (1), describing a neuron with leak conductance g_L receiving an external current I_{app} if the synaptic conductances g_{ij} are changing slowly (meaning they are constant for a small interval dt). Determine I_{app} and g_L analytically in terms of g_{ij}, V_{ij}, V_L and g_{Li}.

4. From spikes to rates

   We are now ready to derive a nonspiking model of a neuron. To do that, we will assume that all neurons have the same membrane capacitance C, the same time constant τ and that conductances are changing slowly (meaning they are constant for a small interval dt).

   Using the results of 1, 2 and 3, show that equation (1) can be approximated by

$$\tau \frac{dx_i}{dt} + x_i \approx f\left(b_i + \sum_j W_{ij} x_j\right) \tag{4}$$

   Starting with the approximation in (2), plug in the approximated f-I relationship from 1(d). Then, substitute I_{app} and g_L with the expressions you found in (3). Assuming all time constants are the same, all synapses emanating from a single neuron have the same temporal behavior, because they are driven by the same spike train, and decay at the same rate. This yields $x_j = \frac{g_{ij}}{\alpha_{ij}}$. Finally, identify b_i and W_{ij} in terms of α_{ij}, g_{Li}, V_L, $V_{1/2}$ and V_{ij}.

5. About linear perceptrons

   (a) We first consider the toy problems from Fig. 1 for which we want to separate filled from empty circles. Draw by hand (and submit the resulting figure) a possible separation surface for each of the examples and name their corresponding logical operation.

   Hint: Recall that in 2D, the decision boundary for a simple perceptron is a line.

   For each problem, propose a perceptron architecture (*i.e.* a network diagram and its associated weights). Although there are many possible solutions we ask for a single one.

   (b) Show that a simple perceptron cannot solve the XOR problem from Fig. 1.

   Hint: The general equation of a perceptron with two inputs is given by:

$$w_0 + w_1 X_1 + w_2 X_2 = 0 \tag{5}$$

   Assume that there exists a solution and find a system of equations that the perceptron coefficients must satisfy. Show that this leads to a contradiction.

   (c) Show that a two-layer perceptron can solve the XOR problem. Draw a possible solution to the XOR problem on Fig. 1. Submit a figure and a network diagram (with associated weights).

   Hint: You can solve the problem by combining the output of two simple perceptrons with an AND function.

   (d) The previous example illustrates how adding more layers to a network allows for more complex decision boundaries. We are now moving toward more interesting problems. What is the minimum number of layers you need to solve the final two problems in Fig. 1? Your solution should contain combinations of simple perceptrons (some of them computing logical functions).

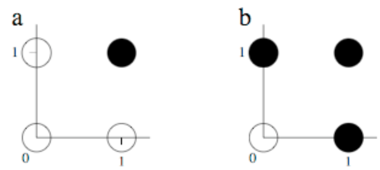

Figure 1: Simple binary cases

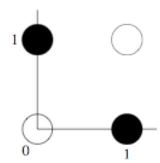

Figure 2: XOR

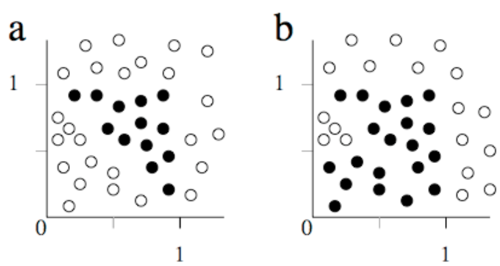

Figure 3: More complex problems

Figure 1: Simple binary cases