

Problem Set 9, CNS 185 1999-2000
The final problem set of the pseudo-millennium

Handed out: 30 Nov 99
Due: 7 Dec 99
Points: 11 points out of 100 for the term

9.1 Computing with binary numbers using McCulloch & Pitts neurons (5 points)

One of the first milestones in the history of neural networks was Warren McCulloch and Walter Pitts' model of a neuron as a binary linear threshold unit. While the biological relevance of this model is dubious, its ability to solve problems by using a network of distributed units was a revolutionary finding. In fact, in their 1943 paper, McCulloch and Pitts showed that any finite logical expression could be constructed using these units.

In this problem set, we will use McCulloch-Pitts units to build simple computational devices. For a more detailed discussion of linear threshold units, see Marvin Minsky's *Computation: Finite and Infinite Machines*. If these topics interest you, you might consider taking CNS188a and b.

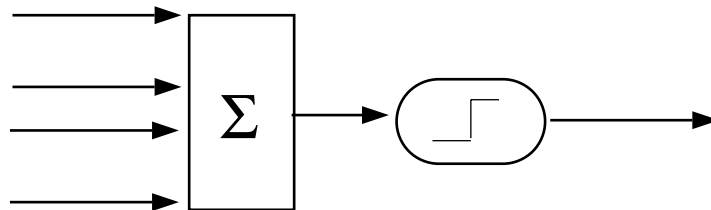


Figure 1: The workings of a McCulloch-Pitts unit

Every McCulloch-Pitts unit outputs a binary value (1 or 0, corresponding to 'active' or 'quiescent') based on the activity of its inputs. As shown in Figure 1, a McCulloch-Pitts unit performs two basic tasks in every time step. It first sums the activity of the input units, and then compares that sum to the unit's threshold value. When the number of active inputs is greater than or equal to the unit's threshold, the unit will be active in the next time step. Otherwise it will become inactive. The threshold is an intrinsic parameter of each unit, although it can vary among units.

Figure 2 shows a few simple McCulloch-Pitts units with their threshold specified in the center of the unit. The first two examples show how to build two of the standard logical gates with a M-P unit. An OR gate is simply a unit with a threshold of 1. Activity in either (or both) of the two inputs will cause the unit to output a 1. The AND gate looks about the same, but it has a threshold of 2, ensuring that only activity in both of the inputs will result in the unit's activation. Note that M-P units can have more than two inputs as well. To build an n -input OR or an n -input AND, we would need to put n input connections on a unit and set its threshold to 1 (for OR) or n (for AND).

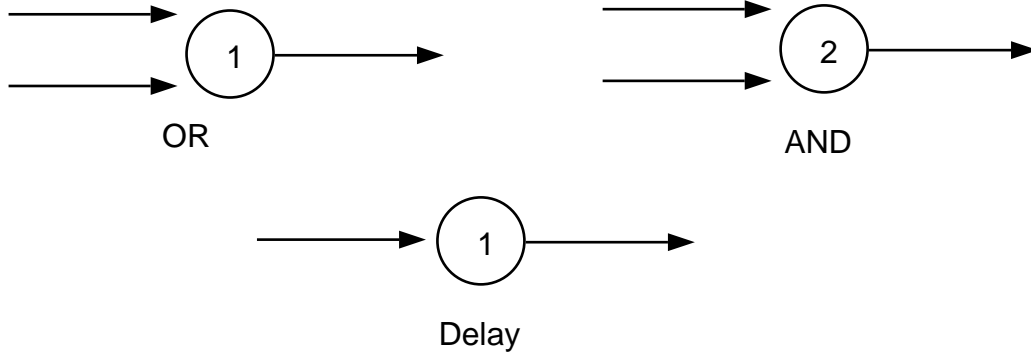


Figure 2: Some example McCulloch-Pitts units

The third example shows a delay circuit, that stores the value of its input during time t , and returns that same value, unchanged, at time $t+1$. Remember that the output of these units is based on the inputs from the previous time step. Therefore it takes one time cycle for information to propagate through any of these units. A simple delay could be useful if different bits of the input to a M-P unit are expected at different times. The bits that arrive earlier could be sent through a delay line, until the other inputs have arrived.

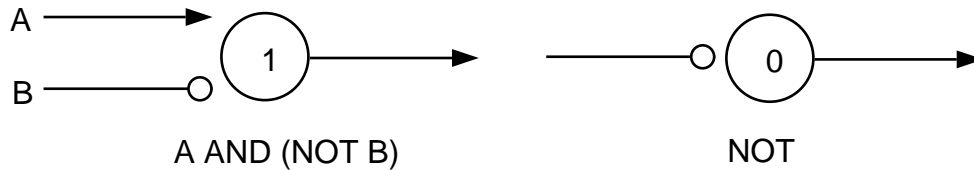


Figure 3: Inhibitory inputs

The M-P units, as we have described them so far, are rather limited because there are some very basic operations that they cannot perform (e.g., NOT, XOR). Like real neurons, however, M-P units are not only restricted to excitatory connections. M-P units can have inhibitory connections (as shown in Figure 3). Some paradigms use absolute inhibition (a unit will output 0 any time one or more of its inhibitory inputs are active), and some use relative inhibition (an active inhibitory inputs sums as a -1; alternatively it can be thought of as raising the threshold by 1).

1. A neural network can easily implement addition of two arbitrarily long numbers, assuming that the numbers are presented to the network starting with the lowest order digit first (from least to most-significant bit). Design a network that can perform such an addition using McCulloch & Pitts neurons with absolute inhibition (that is, a single active inhibitory input prevents the neuron from firing). Your network should accept two input bit streams, representing the two number to be added. It should output a single bit stream, corresponding to the sum of the two inputs. The output can be delayed several time steps with respect to the input. Try to keep the network size small—you should be able to do this using 6 units.

Hand in a diagram of your circuit, making sure you specify the threshold value of each unit. Show how your circuit will handle the following inputs: $A = 11101$ $B = 01110$ (at $t=1$, input A is 1 and input B is 0, at $t=2$, A is 0 and B is 1,...)

2. Can you design another network with fewer units using relative inhibition (a unit fires if net excitation minus net inhibition equals or exceeds the threshold t)? Hand in a diagram of your new network, and work through the same example as before.

- Now we will look at the multiplication of bit streams. Wire up a network of M-P neurons that can multiply two 2-bit long numbers? For this problem you can assume that in addition to the two input lines (A and B), your network has access to a clock signal that will have value 1 at $t=1$ (i.e., the first timestep of the input signal) and will be 0 at any other time. You can also assume that the input streams will have value 0 after the first two time steps. Hand in a diagram of your network and a description of how it works (you can either go through examples, or just verbally describe how it works).
- Can you describe a similar finite network of McCulloch & Pitts neurons that performs multiplication of two arbitrarily long numbers? Justify your answer if you can or can't. Assume now that you know the two numbers are n bits long. What is the minimal number of internal states the network will need to be able to store to compute their product? Why?

9.2 Population Coding (6 points)

One central question in neural coding is, how are vectors represented and transformed in the brain? In other words how is vectorial information encoded and decoded in terms of neuronal activities? Population coding is a very popular candidate to solve this problem, where the idea is to represent a vector in terms of the activity of a population of directionally-tuned neurons. Salient examples include neurons representing wind direction in the cricket, arm movement direction in the monkey and body position in the rat hippocampus. Decoding or read-out can take place simply by forming a *population vector* estimate from the neuronal activities or using more sophisticated decoding methods.¹ In this question we shall explore the properties of the population vector and quantify the accuracy of the estimate in terms of the characteristics of the neuron's tuning curve.

9.2.1 The Population vector

Consider a population of N identical neurons generating Poisson-distributed spike trains in response to a directional stimulus $\theta \in [0; 2\pi]$. The neurons themselves are directionally tuned *i.e.*, they fire preferentially more for some directions and less for others. The firing rate as a function of stimulus angle θ is called the neuron's *tuning curve*. Let the mean firing λ_i of neuron i , $i = 1, \dots, N$, be given by

$$\lambda_i(\theta) = \lambda(\theta - \theta_i),$$

where θ_i is the preferred orientation of neuron i , and the functional form of $\lambda(\theta)$ is

$$\begin{aligned} \lambda(\theta) &= \lambda_{sp} + (\lambda_{max} - \lambda_{sp}) \cos\left(\frac{\pi\theta}{2w}\right) & \text{for } |\theta| < w; \\ \lambda(\theta) &= \lambda_{sp} & \text{for } |\theta| \geq w. \end{aligned}$$

w is the half-width of the tuning curve, λ_{sp} is the spontaneous firing rate, and λ_{max} is the maximal firing rate.

The population vector $\mathbf{P}(\theta)$ in response to a stimulus with direction θ , is obtained by taking the weighted average of the preferred orientations of all the neurons in the population using the the activities of the neurons as their respective weights. $\mathbf{P}(\theta)$ can be written as a vector in the complex plane as,

$$\mathbf{P}(\theta) = \frac{1}{N} \sum_{k=1}^N \lambda_k e^{i\theta_k}, \quad (1)$$

¹For a comprehensive review, see E. Salinas and L.F. Abbott, "Vector Reconstruction from Firing Rates," *J. Comp. Neuroscience*, 89-107, 1994; or the more recent Pouget *et. al* "Statistically efficient estimation using population coding," *Neural Computation*, 10:(2) 373-401, Feb 15 1998. Both are available from your TAs.

where λ_k is the mean firing rate of neuron k . Since the decoder does not have access to the actual firing rates and can only estimate them from the spike counts, λ_k in the equation above have to be replaced by their estimate $\tilde{\lambda}_k = N_k/T$ where N_k is the number of spikes in the time interval T . Note that since $\tilde{\lambda}_k$ is a random variable, $\mathbf{P}(\theta)$ is a random vector. The angle between $\mathbf{P}(\theta)$ and the real-axis is an estimate of the stimulus direction θ . Let us denote the population estimate by θ_{est} .

A popular criterion used to assess the quality of the estimate is the average mean-square error between the stimulus direction θ and the population estimate θ_{est} , $\mathcal{E} = \langle (\theta - \theta_{est})^2 \rangle$. The preferred directions of the neurons in the population can be assumed to either be spaced regularly or distributed uniformly over a given range. A regular distribution of preferred directions, although found in biology (the cricket cercal system and the interneurons in Leech are prominent examples), is more commonly associated with a small network of neurons. For large neuronal populations, a reasonable assumption is that the preferred directions are distributed uniformly, particularly when all stimulus directions $\theta \in [0, 2\pi]$ are equally likely to occur. This homogeneity assumption allows one to derive a closed-form expression for \mathcal{E} in the limit of large N . However, since we are interested in small-sample effects, we shall have to resort to numerical calculations.

1. Use the function `population_activity.m` to identify the dependence of \mathcal{E} on N , T , and λ_{sp} for fixed λ_{max} and w (In other words hand-in plots of \mathcal{E} vs. N , T and λ_{sp}). Compare performance for the case when the preferred directions are uniformly distributed over $[0, 2\pi]$ to when they are regularly spaced.

In order to obtain an accurate estimate of \mathcal{E} , average over several random values of θ chosen uniformly on a given range (choose $[0, 2\pi]$ for the most part, but if you are curious see what happens for smaller ranges). Is there a simple explanation for these relationships in terms of the number of neurons and the variance of a Poisson process? Suggested parameter values: $\lambda_{max} = 50$ Hz, $w = 20$ deg, (remember to change it to radians). Vary N from 10 to 400, T from 0.25 sec. to 4 sec. and λ_{sp} from 0 to 15 Hz.

2. How does the optimal tuning curve width w depend on the spontaneous activity rate λ_{sp} of the cells for a fixed peak firing rate λ_{max} ? Does the conclusion depend on the distribution of preferred directions (uniform vs. regular) ?
3. If the peak rate λ_{max} varies widely across the population (choose λ_{max} uniformly distributed between 50 and 150 Hz for the largest N above), is there a simple way to improve the estimate of $\mathbf{P}(\theta)$? Was this borne out in your simulations?