Problem set 1 Part 1
Perceptrons

Note: if you get stuck, you can find sample code on the course website.

1. Implement Perceptron learning for binary patterns. An input pattern ξ^μ generates an output O^μ that we would like following learning to be equal to σ^μ. Recall that the perceptron learning rule for changing the weight between the j-th input neuron and the output neuron is:

$$\Delta w_j = \begin{cases} 0 & \text{if } \sigma^\mu = O^\mu \\ 2\eta\xi_j^\mu\sigma^\mu & \text{if } \sigma^\mu \neq O^\mu \end{cases} \tag{1}$$

   Or in vector form, meaning an equation for all synapses at once:

$$\Delta\overrightarrow{w} = \begin{cases} 0 & \text{if } \sigma^\mu = O^\mu \\ 2\eta\overrightarrow{\xi^\mu}\sigma^\mu & \text{if } \sigma^\mu \neq O^\mu \end{cases} \tag{2}$$

   (a) Write down a word-level description of what code implementing perceptron learning needs to do (this is often called pseudo-code).

   (b) Turn this description into actual code implementing perceptron learning.

   (c) Generate the set of input patterns and desired outputs for the Boolean AND function. Show that it can be learned. Don't forget to include a non-zero threshold.

   (d) Generate the set of input patterns for the Boolean XOR function. Show that it can't be learned.

   (e) **Bonus:** implement perceptron learning for multi-layer perceptrons to show that the XOR function can be classified once a hidden layer is added. You can look up how the perceptron rule is related to back-prop in Hertz, Krogh, Palmer or online.

2. How many patterns can a perceptron learn? Implement a 10 input neuron Perceptron. Generate datasets with different number of randomly generated binary patterns. Train the Perceptron. See how large a dataset can still successfully train.