Coursework 2

This coursework relates to the perceptron. In a perceptron there is an input vector of size d:

$$\boldsymbol{x} = \{x_1, x_2, \ldots, x_d\}^\top \tag{1}$$

These are not restricted to discrete values.

There is also a vector of synaptic weights

$$\boldsymbol{w} = \{w_1, w_2, \ldots, w_d\}^\top \tag{2}$$

The perceptron's output is obtained by taking the inner (dot) product between the weights $\boldsymbol{w}$ and the inputs $\boldsymbol{x}$, and then applying a nonlinearity (in this case a threshold). The result is a predicted output $\hat{y}$:

$$\hat{y} = \begin{cases} 1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} \geq \theta \\ -1 & \text{otherwise,} \end{cases} \tag{3}$$

where $\boldsymbol{w}^\top \boldsymbol{x} \geq= \sum_i w_i x_i$ is the dot product.

Here, we will learn the perceptron weights $\boldsymbol{w}$ using a supervised training process. We will use a training data set consisting of input–output pairs $(\boldsymbol{x}_n, y^*)$. $\boldsymbol{x}_n$ are inputs, and $y^* \in \{-1, 1\}$ is a *target* output. We want to use this training data so that the perceptron's predicted output $\hat{y}$ matches y^*.

$$\begin{aligned} \mathrm{X} &= \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \\ \mathrm{Y}^* &= \{y_1^*, y_2^*, \ldots, y_n^*\} \end{aligned} \tag{4}$$

In this example the label -1 corresponds to **blue** and the label 1 corresponds to **red**.

The goal is to use the perceptron to map:

$$\mathbf{x} \mapsto y^* \tag{5}$$

so we want to adjust the weights $\boldsymbol{w}$ so that the predicted $\hat{y}$ match the supervised targets y^* as much as possible. The perceptron rule is that we can do this by the update:

$$\begin{aligned} w_i &\leftarrow w_i + \eta x_i (y^* - \hat{y}) \\ \theta &\leftarrow \theta + \eta (y^* - \hat{y}) \end{aligned} \tag{6}$$

We can also write the weight update Equation (6) using vector notation as

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta(y^* - \hat{y})\boldsymbol{x} \tag{7}$$

In the `Coursework2` folder you will find a list of labelled points called `points.txt`. The goal is to programme a perceptron capable of classifying these points. Pick a modest η such as $\eta = 0.01$ and it should learn the task in tens of trials. I have supplied small programmmes, `load.py` and `load.jl`, to help load the data in python or julia.

If you have this working you can experiment with what happens if you change the MP neuron, or the perceptron rule, or mislabel some points.