Exercise 1

Perceptrons

The activation function for each perceptron will be the step function

$$\varphi(x) = \begin{cases} 1 & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

The weights and biases will be the following:

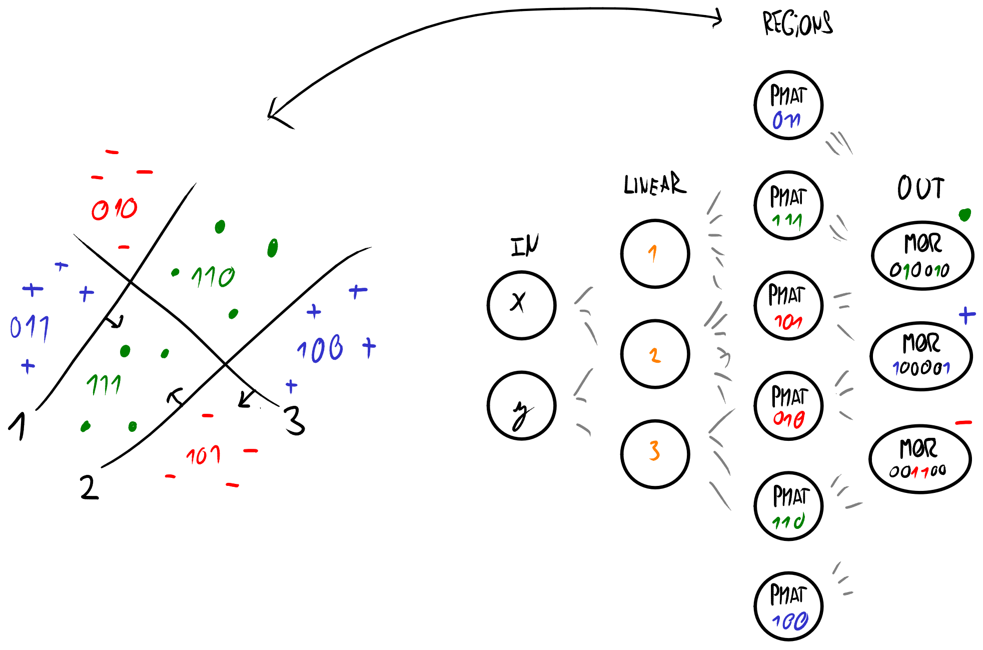
1) OR
   - weights: $\{1\}^D$
   - bias: 0
2) MOR
   - weights: c
   - bias: 0
3) PMAT
   - weights: c with 0s replaced by -1
   - bias: $-\sum c + 1$

The idea for PMAT is that if it perfectly corresponds, βX will sum to $\sum c$ and so it will activate, and if it doesn't it will always be lower due to either $+0$ if it doesn't correspond or -1 .

The ideas are the same. The main difference is that our step function is 0 at $x = 0$, whereas that of the sample solution is 1 at that point, which alters the choice for the bias.

Network

The network, along with the linear decision planes, is described in the following diagram:



This classifier (especially our version) will likely be very hard to train, since the derivative for the step function will be 0 essentially everywhere.

Again, the ideas are the same, although our explanation of the different layers could have been more elaborate. We did not mention how to generalize our method, but it should be pretty obvious. Our reasoning for why the classifier would fail in practice was perhaps a bit misguided.

Exercise 2

We want to prove that we can merge two layers into one by multiplying out the activation:

$$\begin{aligned} y &= \varphi(\varphi(X\beta_1 + b_1)\beta_2 + b_2) \\ &= (X\beta_1 + b_1)\beta_2 + b_2 && \text{identities} \\ &= X\beta_1\beta_2 + b_1\beta_2 + b_2 && \text{multiply out} \\ &= X\underbrace{(\beta_1\beta_2)}_{\beta} + \underbrace{(b_1\beta_2 + b_2)}_{b} \end{aligned}$$

As before, our idea is essentially right, but our explanation could be more verbose.

Exercise 3

In its own HTML file.