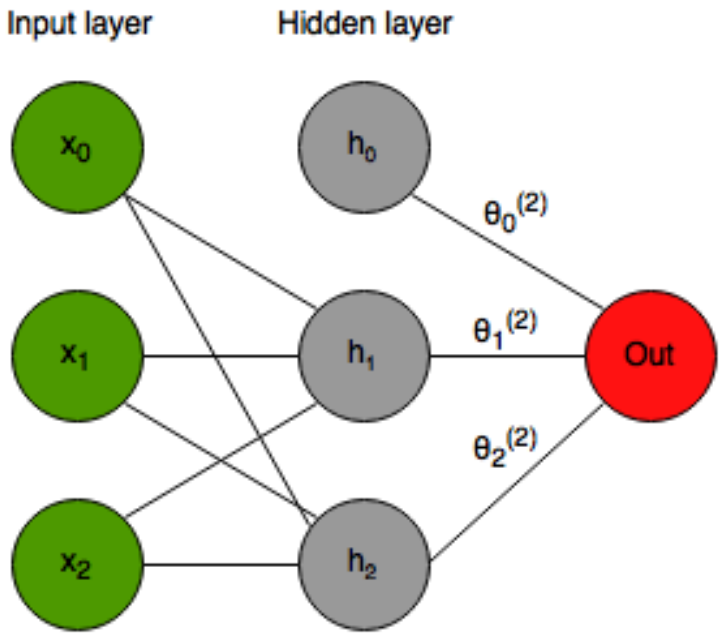
AUC Course: Machine Learning

Solutions to Assignment 3

Vincent Roest, 10904816
vincentr@live.nl

I Worked together with Bas Straathof in my quest to correct solutions

Problem 3)
a)



The network looks like that one depicted in the figure above. Let us define $h_i^{(j)}$ as the activation unit i in layer j. Let us also define $\theta^{(j)}$ as the matrix of weights that control the function mapping from layer j to layer j+1. We have as activation functions:

$$h_1^{(2)} = g(\theta_{10}^{(1)} \cdot x_0 + \theta_{11}^{(1)} \cdot x_1 + \theta_{12}^{(1)} \cdot x_2$$

$$h_2^{(2)} = g(\theta_{20}^{(1)} \cdot x_0 + \theta_{21}^{(1)} \cdot x_1 + \theta_{22}^{(1)} \cdot x_2$$

$$h_\theta(x) = h_3^{(1)} = g(\theta_{10}^{(2)} \cdot h_0^{(2)} + \theta_{11}^{(2)} \cdot h_1^{(2)} + \theta_{12}^{(2)} \cdot h_2^{(2)})$$

Here g is of course the sigmoid function defined as: $g(x) = \frac{1}{1+e^{-x}}$. We are given the vectors $v^{(1)} = [0.5, 0.1, 0.5, 0.7]$ and $v^{(2)} = [1, 2]$. These values correspond to the weights (the θ values).

We set the value of the bias nodes to 1. Then we get the following:

$$h_1^{(2)} = g(0.2 \cdot 1 + 0.5 \cdot 0.5 + 0.5 \cdot 0.9 = g(0.9) = 0.711$$

$$h_2^{(2)} = g(0.2 \cdot 1 + 0.1 \cdot 0.5 + 0.7 \cdot 0.9 = g(0.88) = 0.707$$

$$h_\theta(x) = h_3^{(1)} = g(0.2 \cdot 1 + 1 \cdot h_1^{(2)} + 2 \cdot h_2^{(2)}) g(0.2 \cdot 1 + 1 \cdot 0.711 + 2 \cdot 0.707 = g(2.325) = 0.91093$$

b) We calculate the errors for all nodes and updates of the weights by using back propagation. Let us define $\delta_j^{(i)}$ as the error of node j in layer i. We have three layers in our network, so we start from the last node and work our way backwards. Of course, we do not calculate the input layer's errors $\delta_1^{(1)}$ and $\delta_2^{(1)}$.
At the last layer we have, letting y be the actual value and a again the activation value:

$$\delta_1^{(3)} = y^{(i)} - h_3^{(1)} = 1 - 0.91 = 0.09$$

So now we can work our way backwards for the nodes in layer 2:

$$\delta_1^{(2)} = \theta_{11}^{(2)} \cdot \delta_1^{(3)} \cdot h_1^{(2)} \cdot (1 - h_1^{(2)}) = 1 \cdot 0.09 \cdot 0.711 \cdot (1 - 0.711) = 0.01876$$

$$\delta_2^{(2)} = \theta_{12}^{(2)} \cdot \delta_1^{(3)} \cdot h_1^{(2)} \cdot (1 - h_1^{(2)}) = 2 \cdot 0.09 \cdot 0.707 \cdot (1 - 0.707) = 0.03729$$

This is then the one iteration we were looking for. Then we can "correct" the weights by using the update formula defined by: $weight_{ij}^{(l)} = weight_{ij}^{(l)} + a_j^{(l)} \cdot \delta_i^{(l+1)}$. With the definition of the updated edges the activation $h_3^{(1)}$ becomes:

$$h_3^{(1)} = g(\theta_{10}^{(2)} \cdot h_0^{(2)} + (\theta_{11}^{(2)} + \delta_1^{(2)} \cdot h_1^{(2)}) \cdot h_1^{(2)} + (\theta_{12}^{(2)} + \delta_2^{(2)} \cdot h_2^{(2)}) \cdot h_2^{(2)})$$

$$h_3^{(1)} = g(0.2 + (1 + 0.01876 \cdot 0.711) \cdot 0.711) + (2 + 0.03729 \cdot 0.707) \cdot 0.707) = g(2.35312) = 0.91318$$

Now, this is not a very significant improvement compared to the value we had (0.91093), but this is of course only one iteration. Repeating this various times would definitely decrease the error from the output.

Problem 4)
4.1) For figure b, this classifier will be useless, but should we still want to define it, it's output would be the following for values of x:

$$out(x_1, \ldots, x_n) = \begin{cases} 1 & w_0 + w_1 \cdot x_1 + w_n \cdot x_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

We know two points of the boundary: A = (-1,0) and B = (0,2). Now we can easily construct a boundary line:

$$\frac{x_1 - x_1 A}{x_1 b - x_1 A} = \frac{x_2 - x_2 A}{x_2 b - x_2 A} \rightarrow \frac{x_1 - (-1)}{0 - (-1)} = \frac{x_2 - 0}{2 - 0} \rightarrow 2 + 2x_1 - x_2 = 0$$

So now we have as possible weights $w_0 = 2$, $w_1 = 2$, $w_0 = -1$. Since figure b is not linearly separable, there is no real way to determine whether these weights are right. We can, however, determine if they are right for figure a. Suppose we want to classify the origin (0,0). Then the output for this point should be negative if we look at the graph. With these weights, however, it is positive. Therefore, we negate all the weights and get the following weights: $w_0 = -2$, $w_1 = -2$, $w_0 = 1$

4.2 a) Let us start with a truth table for A AND (NOT B):

A	B	(A	&	∼	B)
1	1	1	−1	−1	1
1	−1	1	1	1	−1
−1	1	−1	−1	−1	1
−1	−1	−1	−1	1	−1

The Boolean function A AND (NOT B) has inputs A and B (both can be 1 for true or -1 for false). Let us define A as x_1 and B as x_2 For A AND (NOT B), we have to find a line that separates the positive outputs from the negative values. If we were to plot the inputs according to the truth table, we would have the first, second and third quadrant (counterclockwise starting at the positive x positive quadrant) to be -1 and the fourth to be 1. Any line that separates these points is a correct boundary for our input. A good line by observation from the truth table and the graph as we defined it is: $w_0 = -1$ (crossing the y axis in -1), $w_1 = 1$ and $w_2 = -1$. Then the output of the perceptron is given by:

$$out(x_1, \ldots, x_2) = \begin{cases} 1 & -1 + 1 \cdot x_1 + -1 \cdot x_2 > 0 \\ -1 & \text{otherwise} \end{cases}$$

We can check the output for which we know we have 1 as a correct classification, namely A (x_1) = 1 and B (x_2) = -1. Then we get: $-1 + 1 \cdot 1 + -1 \cdot -1 > 0$). This is true, so we classify this as 1, which is correct. Another example might be A (x_1) = -1 and B (x_2) = -1. Then we have: $-1 + 1 \cdot -1 + -1 \cdot -1 < 0$), so we correctly output -1. The same procedure works for all possible inputs for A and B.

4.2 b) Let us start with a truth table for A XOR B:

A	B	((A	&	∼	B)	∨	(∼	A	&	B))
1	1	1	−1	−1	1	−1	−1	1	−1	1
1	−1	1	1	1	−1	1	−1	1	−1	−1
−1	1	−1	−1	−1	1	1	1	−1	1	1
−1	−1	−1	−1	1	−1	−1	1	−1	−1	−1

We have to design a neural network with one input layer, a hidden layer and an output layer. If we hold on to the graph imagery, then we would now classify the points in the second and fourth quadrant as positive: A and negative B and negative A and positive B (From the truth table). We can build two "hidden" nodes. One node determines whether we have negative A and B or that we don't, whereas the other node takes care of the case that A (x_1) and B (x_2)

are positive. Then we have these two nodes:

$$h_1(x_1, x_2) = \begin{cases} 1 & \text{for } x_1 + x_2 + 0.5 < 0 \\ -1 & \text{otherwise} \end{cases}$$

$$h_2(x_1, x_2) = \begin{cases} 1 & \text{for } x_1 + x_2 - 0.5 < 0 \\ -1 & \text{otherwise} \end{cases}$$

Then as output we get:

$$h_3(x_1, x_2) = \begin{cases} 1 & \text{for } x_1 + x_2 - 0.5 < 0 \\ -1 & \text{otherwise} \end{cases}$$