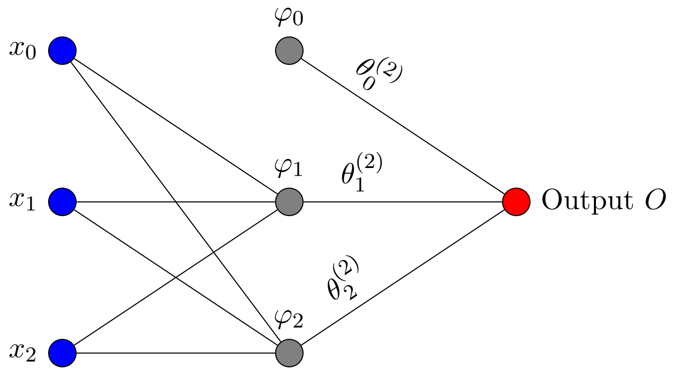
AUC Course: Machine Learning

Written Assignment 3

Simon Fang
simon.fang@student.auc.nl
10898492

Exercise 3:
We have the following Artificial Neural Network (ANN), where the first layer - denoted by the blue nodes - is the input layer, the second layer - denoted by the grey nodes - is the hidden layer and the last layer - denoted by the singel red node is the output layer. The bias nodes of the input and the hidden layer are given as x_0 and φ_0 respectively. For the ANN, see the figure below.



We are given the following θ-values:

$$\theta^{(1)} = \begin{bmatrix} 0.5 \\ 0.1 \\ 0.5 \\ 0.7 \end{bmatrix} \qquad \theta^{(2)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{1}$$

where $\theta^{(1)}$ are the weights from x_i to φ_i for $i = 1, \dots, n$ and the first two elements in $\theta^{(1)}$ correspond to the edge from x_1 to φ_1 and φ_2 respectively, while the last two elements in $\theta^{(1)}$ correspond to the edge from x_2 to φ_1 and φ_2.

We are given the following two values of the activation nodes in the input layer: $x_1 = 0.5$ and $x_2 = 0.9$. Moreover, we are given that the values of θ

for the bias nodes are 0.2 and it is given that the value of every bias node is equal to 1 as usual. Using the information above, we will answer the questions.

1. Calculate the activation of all nodes in the ANN by hand:

$$\begin{aligned}\varphi_1 &= g(0.2 + 0.5x_1 + 0.5x_2)\\ &= g(0.2 + 0.5 \cdot 0.5 + 0.5 \cdot 0.9)\\ &= g(0.9) \approx 0.7109\end{aligned} \tag{2}$$

$$\begin{aligned}\varphi_1 &= g(0.2 + 0.1x_1 + 0.7x_2)\\ &= g(0.2 + 0.1 \cdot 0.5 + 0.7 \cdot 0.9)\\ &= g(0.88) \approx 0.7068\end{aligned} \tag{3}$$

$$\begin{aligned}O_1 &= g(0.2 + \varphi_1 + 2\varphi_2)\\ &= g(0.2 + 0.7109 + 2 \cdot 0.7068)\\ &= g(2.3247) \approx 0.9109\end{aligned} \tag{4}$$

where $g(z) = \frac{1}{1+\exp(-z)}$ and the prediction function is $h_\theta(x) = g(\theta^T x)$ for logistic regression

2. Suppose that the correct output is equal to 1. Calculate the errors for all nodes and the updates of the weights. Note that the cost function $J(\theta)$ is defined as

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{5}$$

where each $h_\theta(x^{(i)}$ is the predicted value calculated in the first section and each $y^{(i)}$ is the actual output. In our case, all the nodes in the hidden layer have to be equal to 1 in order to obtain an output node that is equal to 1. We can calculate the cost function or the mean squared error (MSE) for each node using the information above.

The MSE for node O_1 is

$$\begin{aligned}J(\theta^{(2)}) &= \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2\\ &= \frac{1}{2}(0.9109 - 1)^2 = 0.0040\end{aligned} \tag{6}$$

The MSE for node φ_1 is

$$\begin{aligned} J(\theta^{(1)}) &= \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2}(0.7109 - 1)^2 = 0.0418 \end{aligned} \tag{7}$$

The MSE for node φ_2 is

$$\begin{aligned} J(\theta^{(1)}) &= \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2}(0.7068 - 1)^2 = 0.0430 \end{aligned} \tag{8}$$

The weights can be calculated with the following formula:

$$\theta_i = \theta_i - \alpha\frac{\partial J}{\partial \theta_i} \tag{9}$$

where $\frac{\partial J}{\partial \theta_i} = (h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$. For the update, we choose $\alpha = 1$, which does not necessarily give us the global minimum, but it suffices for one iteration. We use equation (9) to calculate the new weights for each layer.

$$\begin{aligned} \theta_1^{(2)} &= \theta_1^{(2)} - \alpha\frac{\partial J}{\partial \theta_1} \\ &= \theta_1^{(2)} - (h(\varphi_{(1)}) - y_{(1)})\varphi_{(1)} \\ &= 1 - (0.7109 - 1)0.7109 \\ &= 1.21 \end{aligned} \tag{10}$$

$$\begin{aligned} \theta_2^{(2)} &= \theta_2^{(2)} - \alpha\frac{\partial J}{\partial \theta_2} \\ &= \theta_2^{(2)} - (h(\varphi_{(2)}) - y_{(2)})\varphi_{(2)} \\ &= 2 - (0.7068 - 1)0.7068 \\ &= 2.2 \end{aligned} \tag{11}$$

$$\begin{aligned} \theta_1^{(1)} &= \theta_1^{(1)} - \alpha\frac{\partial J}{\partial \theta_1} \\ &= \theta_1^{(1)} - (h(x_{(1)}) - y_{(1)})x_{(1)} \\ &= 0.5 - (0.622 - 1)0.5 \\ &= 0.69 \end{aligned} \tag{12}$$

$$\begin{aligned}\theta_2^{(1)} &= \theta_1^{(1)} - \alpha\frac{\partial J}{\partial \theta_1} \\ &= \theta_2^{(1)} - (h(x_{(1)}) - y_{(1)})x_{(1)} \\ &= 0.1 - (0.52 - 1)0.5 \\ &= 0.34\end{aligned} \tag{13}$$

$$\begin{aligned}\theta_3^{(1)} &= \theta_3^{(1)} - \alpha\frac{\partial J}{\partial \theta_3} \\ &= \theta_3^{(2)} - (h(x_{(3)}) - y_{(3)})x_{(3)} \\ &= 0.5 - (0.622 - 1)0.5 \\ &= 0.69\end{aligned} \tag{14}$$

$$\begin{aligned}\theta_4^{(1)} &= \theta_4^{(1)} - \alpha\frac{\partial J}{\partial \theta_4} \\ &= \theta_4^{(1)} - (h(x_{(4)}) - y_{(4)})x_{(4)} \\ &= 0.7 - (0.668 - 1)0.7 \\ &= 0.93\end{aligned} \tag{15}$$

Exercise 4.1: In this case, the decision surface is a line that crosses both axes. The general equation of a line is given as

$$w_0 + w_1x_1 + w_2x_2 = 0 \tag{16}$$

where $w_i \in \mathbb{R}$ for $i = 0, \ldots, 2$. Our aim is to find the values of the weights w_i and we know that it passes throught the points $m = (-1, 0)$ and $n = (0, 2)$. If we fix $w_1 = 1$, then we can re-arrange the equation such that we obtain the same form as $y = ax + b$, where $a = \frac{\Delta y}{\Delta x}$. Using this, the value of w_2 is given by

$$w_2 = \frac{\Delta x_2}{\Delta x_1} = \frac{-2}{-1} = 2 \tag{17}$$

and we can plug in this value into the equation, so that we can obtain w_0

$$x_1 = 2x_2 + w_0 \implies 0 = -2 + w_0 \implies w_0 = 2 \tag{18}$$

But the perceptron outputs positive values for inputs above the surface and, therefore, we have to multiply our answer with -1, i.e.,

$$
\begin{aligned}
w_0 &= -2 \\
w_1 &= -2 \\
w_2 &= \ 1
\end{aligned}
$$

Exercise 4.2:

1. Design a two-input perceptron that implements: A AND (NOT B). Let A and B correspond to x_1 and x_2 respectively. We will get the following perceptron:

$$O(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 - x_2 + 0.5 < 0 \\ -1 & \text{otherwise} \end{cases} \tag{19}$$

2. In order to design the XOR function, we have to design an ANN with an input layer, a hidden layer and an output layer. The idea is that if we plot A and B in a two-dimensional graph, then we would classify the points in the second and fourth orthant as positive output, i.e., these are points near positive A and negative B or negative A and positive B. We can build two hidden layer nodes, such that one classifies whether it is negative A and B or not and the other classifies the case when both A and B are positive at the same time:

$$h_1(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 + x_2 + 0.5 < 0 \\ -1 & \text{otherwise} \end{cases} \tag{20}$$

$$h_2(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 + x_2 - 0.5 < 0 \\ -1 & \text{otherwise} \end{cases} \tag{21}$$

And the output will be

$$h_2(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 + x_2 - 0.5 < 0 \\ -1 & \text{otherwise} \end{cases} \tag{22}$$

In all the above, x_1 and x_2 correspond to A and B respectively.