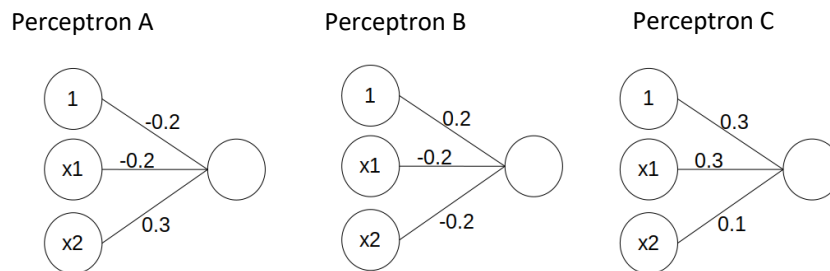


School of Computing and Information Systems
The University of Melbourne
COMP90049 Introduction to Machine Learning (Semester 2, 2020)
Sample solutions: Week 7

1. Consider a multiclass classification task using the three perceptrons below:



Recall the training procedure for the multi-class perceptron: we predict \hat{y} as the class whose parameters lead to the largest activation. During training, if a wrong class was predicted, we update the parameters activated for the wrong class (decreasing their activation), and the parameters activated for the correct class (increasing their activation).

Following this scheme, do the following:

- i. *Training:* For training example $x_{\text{train}} = (0, 1)$, with class **B**, calculate the updated weights of each perceptron after processing x . Use learning rate $\eta = 0.1$.

To train our perceptrons, we first compute the activation for each class given x_{train} .

$$\begin{aligned} \text{Perceptron A:} & \quad -0.2 - 0.2 \times 0 + 0.3 \times 1 = 0.1 \\ \text{Perceptron B:} & \quad 0.2 - 0.2 \times 0 - 0.2 \times 1 = 0 \\ \text{Perceptron C:} & \quad 0.3 + 0.3 \times 0 + 0.1 \times 1 = 0.4 \end{aligned}$$

In multiclass perceptron models, the model prediction is calculated as follow:

$$\hat{y} = \underset{c \in \{A, B, C\}}{\operatorname{argmax}} \left(\sum_i (\theta_i^c x_i) \right)$$

Since Perceptron **C** returns the maximum activation value (0.4), the **prediction** of our model for our training instance ($x_{\text{train}} = (0, 1)$) is label **C**. This does not match the **actual** label for our training instance (**B**).

Therefore, we need to update the parameters activated for the **wrong class** (Perceptron **C**) and the parameters activated for the **correct class** (Perceptron **B**).

We update the parameters for Perceptron **B** and **C** as follow:

Perceptron B:

For the actual (correct) class Perceptron, we move **toward** predicting y^c for x^i ($c = B$ in this case and $x^i = (0, 1)$).

$$\theta_j^{(t)} \leftarrow \theta_j^{(t-1)} + \eta x^i$$

Update θ :

$$\begin{aligned} \theta_0^{(1)} &= 0.2 + 0.1 \times 1 = 0.3 \\ \theta_1^{(1)} &= -0.2 + 0.1 \times 0 = -0.2 \\ \theta_2^{(1)} &= -0.2 + 0.1 \times 1 = -0.1 \end{aligned}$$

Perceptron C:

For the incorrect class Perceptron, we move **away from** predicting y^c for x^i ($c = C$ in this case and $x^i = (0, 1)$).

$$\theta_j^{(t)} \leftarrow \theta_j^{(t-1)} - \eta x^i$$

Update θ :

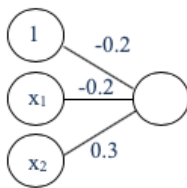
$$\theta_0^{(1)} = 0.3 - 0.1 \times 1 = 0.2$$

$$\theta_1^{(1)} = 0.3 - 0.1 \times 0 = 0.3$$

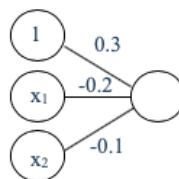
$$\theta_2^{(1)} = 0.1 - 0.1 \times 1 = 0$$

After these updates our three perceptrons will be as follows:

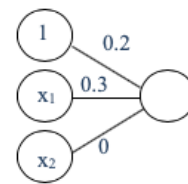
Perceptron A



Perceptron B



Perceptron C



- ii. *Testing:* Using the new weights from part (a), given test example $x_{\text{test}} = (1, 1)$ with true class **C**, give the class prediction of the group of three perceptrons. Is the prediction correct? Should we change the weights?

Perceptron A: $-0.2 - 0.2 \times 1 + 0.3 \times 1 = -0.1$

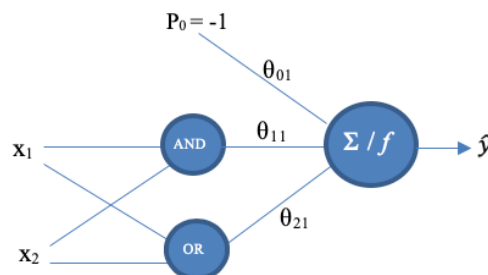
Perceptron B: $0.3 - 0.2 \times 1 - 0.1 \times 1 = 0$

Perceptron C: $0.2 + 0.3 \times 1 + 0 \times 1 = 0.5$

Based on these results our prediction for our x_{test} is **C**, which match our actual label. So our prediction is **correct**. 😊

Remember, even if our prediction was incorrect, we cannot update the weights of our perceptrons because we are in the testing phase.

2. Consider the two levels deep network illustrated below. It is composed of three perceptrons. The two perceptrons of the first level implement the AND and OR function, respectively.



Determine the weights θ_1 , θ_2 and bias θ_0 such that the network implements the XOR function. The initial weights are set to zero, i.e. $\theta_{01} = \theta_{11} = \theta_{21} = 0$, and the learning rate η (eta) is set to 0.1.

Notes:

- The input function for the perceptron on level 2 is the weighted sum (Σ) of its input.
- The activation function f for the perceptron on level 2 is a *step function*:

$$f = \begin{cases} 1 & \text{if } \Sigma > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Assume that the weights for the perceptrons of the first level are given.

Learning Algorithm for XOR:

For each training example x
 $p \leftarrow (p_0, f_{AND}(x), f_{OR}(x))$
 $\hat{y} \leftarrow f(\sum_{i=0}^n \theta_i p_i)$
 $y \leftarrow \text{target of } x$
For $i = 1 : n$
 $\Delta\theta_i \leftarrow \eta(y - \hat{y}) p_i$
 $\theta_i \leftarrow \theta_i + \Delta\theta_i$

To calculate the *output* of the two level 1 perceptrons, we can use the following table (Remember since the weights for these perceptrons are given we don't need to do the iterative learning):

Train instance #	x_1	x_2	p_1 $f_{AND}(x)$	p_2 $f_{OR}(x)$	y $x_1 \text{ XOR } x_2$
1	1	0	0	1	1
2	0	1	0	1	1
3	1	1	1	1	0
4	0	0	0	0	0

Based on the results from above table, our input signals (training instances) for level 2 perceptron (XOR) are $P_i = \langle p_0, p_1, p_2 \rangle = \langle -1, p_1, p_2 \rangle$ and the parameters are $\theta = \langle \theta_{01}, \theta_{11}, \theta_{21} \rangle = \langle 0, 0, 0 \rangle$.

So, for the first epoch we have:

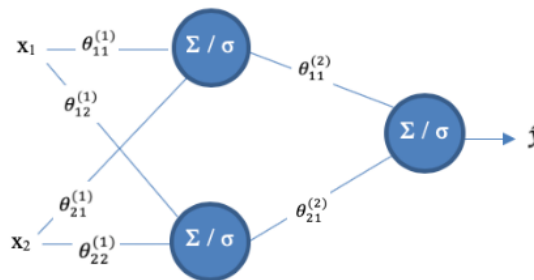
ep	P $\langle p_0, p_1, p_2 \rangle$	$z = \theta \cdot P$ $\theta_0 \times p_0 + \theta_1 \times p_1 + \theta_2 \times p_2$	\hat{y} $f(z)$	y	$\Delta\theta_{1i}$ $\eta (y - \hat{y}) p_i$
1	$\langle -1, 0, 1 \rangle$	$0 \times (-1) + 0 \times 0 + 0 \times 1 = 0$ Update θ to $\langle 0, 0, 0 \rangle + \langle -0.1, 0, 0.1 \rangle = \langle -0.1, 0, 0.1 \rangle$	$f(0)=0$	1	$0.1 \times (1-0) \times \langle -1, 0, 1 \rangle$ $= \langle -0.1, 0, 0.1 \rangle$
	$\langle -1, 0, 1 \rangle$	$(-0.1) \times -1 + 0 \times 0 + (0.1) \times 1 = 0.2$	$f(0.2)=1$	1	No change required
	$\langle -1, 1, 1 \rangle$	$(-0.1) \times -1 + 0 \times 1 + (0.1) \times 1 = 0.2$ Update θ to $\langle -0.1, 0, 0.1 \rangle + \langle 0.1, -0.1, -0.1 \rangle = \langle 0, -0.1, 0 \rangle$	$f(0.2)=1$	0	$0.1 \times (0-1) \times \langle -1, 1, 1 \rangle$ $= \langle 0.1, -0.1, -0.1 \rangle$
	$\langle -1, 0, 0 \rangle$	$0 \times -1 + (-0.1) \times 0 + 0 \times 0 = 0$	$f(0)=0$	0	No change required
2	$\langle -1, 0, 1 \rangle$	$0 \times -1 + (-0.1) \times 0 + 0 \times 1 = 0$ Update θ to $\langle 0, -0.1, 0 \rangle + \langle -0.1, 0, 0.1 \rangle = \langle -0.1, -0.1, 0.1 \rangle$	$f(0)=0$	1	$0.1 \times (1-0) \times \langle -1, 0, 1 \rangle$ $= \langle -0.1, 0, 0.1 \rangle$
	$\langle -1, 0, 1 \rangle$	$(-0.1) \times -1 + (-0.1) \times 0 + (0.1) \times 1 = 0.2$	$f(0.2)=1$	1	No change required
	$\langle -1, 1, 1 \rangle$	$(-0.1) \times -1 + (-0.1) \times 1 + (0.1) \times 1 = 0.1$ Update θ to $\langle -0.1, -0.1, 0.1 \rangle + \langle 0.1, -0.1, -0.1 \rangle = \langle 0, -0.2, 0 \rangle$	$f(0.1)=1$	0	$0.1 \times (0-1) \times \langle -1, 1, 1 \rangle$ $= \langle 0.1, -0.1, -0.1 \rangle$
	$\langle -1, 0, 0 \rangle$	$0 \times -1 + (-0.2) \times 0 + 0 \times 0 = 0$	$f(0)=0$	0	No change required

3	<-1,0,1>	$0 \times -1 + (-0.2) \times 0 + 0 \times 1 = 0$ Update θ to $\langle 0, -0.2, 0 \rangle + \langle -0.1, 0, 0.1 \rangle = \langle -0.1, -0.2, 0.1 \rangle$	$f(0)=0$	1	$0.1 \times (1-0) \times \langle -1, 0, 1 \rangle$ $= \langle -0.1, 0, 0.1 \rangle$
	<-1,0,1>	$(-0.1) \times -1 + (-0.2) \times 0 + (0.1) \times 1 = 0.2$	$f(0.2)=1$	1	No change required
	<-1,1,1>	$(-0.1) \times -1 + (-0.2) \times 1 + (0.1) \times 1 = 0$	$f(0)=0$	0	No change required
	<-1,0,0>	$(-0.1) \times -1 + (-0.2) \times 0 + (0.1) \times 0 = 0.1$ Update θ to $\langle -0.1, -0.2, 0.1 \rangle + \langle 0.1, 0, 0 \rangle = \langle 0, -0.2, 0.1 \rangle$	$f(0.1)=1$	0	$0.1 \times (0-1) \times \langle -1, 0, 0 \rangle$ $= \langle 0.1, 0, 0 \rangle$
4	<-1,0,1>	$0 \times -1 + (-0.2) \times 0 + (0.1) \times 1 = 0.1$	$f(0.1)=1$	1	No change required
	<-1,0,1>	$0 \times -1 + (-0.2) \times 0 + (0.1) \times 1 = 0.1$	$f(0.1)=1$	1	No change required
	<-1,1,1>	$0 \times -1 + (-0.2) \times 1 + (0.1) \times 1 = -0.1$	$f(-0.1)=0$	0	No change required
	<-1,0,0>	$0 \times -1 + (-0.2) \times 0 + (0.1) \times 0 = 0$	$f(0)=0$	0	No change required

Since in the last epoch we didn't have any updated for our weights (θ_1), the algorithm converges here.

So for our network to perform XOR function, the final θ_1 would be $\theta_1 = \langle 0, -0.2, 0.1 \rangle$. In other words $\theta_{10} = 0$, $\theta_{11} = -0.2$ and $\theta_{12} = 0.1$

3. Consider the following multilayer perceptron.



The network should implement the XOR function. Perform **one** epoch of backpropagation as introduced in the lecture on multilayer perceptrons.

Notes:

- The activation function f for a perceptron is the *sigmoid function*:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- The bias nodes are set to -1.** They are not shown in the network
- Use the following initial parameter values:

$$\begin{array}{lll} \theta_{01}^{(1)} = 2 & \theta_{02}^{(1)} = -1 & \theta_{01}^{(2)} = -2 \\ \theta_{11}^{(1)} = 6 & \theta_{12}^{(1)} = 8 & \theta_{11}^{(2)} = 6 \\ \theta_{21}^{(1)} = -6 & \theta_{22}^{(1)} = -8 & \theta_{21}^{(2)} = -6 \end{array}$$

- The learning rate is set to $\eta = 0.7$

i. Compute the activations of the hidden and output neurons

Since our activation function here is the sigmoid (σ) function, in each node (neuron) we can calculate the output by applying the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$) on the weighted sum (Σ) of its input (that we usually represent by $z_b^{(a)}$ where a shows the level of the neuron (e.g. level 1, 2, 3) and b is the index).

So for the neuron i in level 1, we will have:

$$a_i^{(1)} = \sigma(z_i^{(1)}) = \sigma(\theta_i^{(1)} \cdot X) = \sigma(\theta_{0i}^{(1)} \times x_0 + \theta_{1i}^{(1)} \times x_1 + \theta_{2i}^{(1)} \times x_2)$$

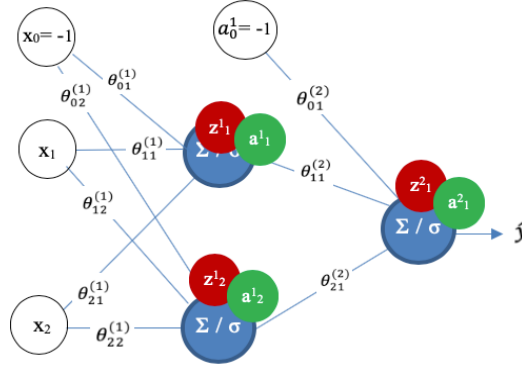
So we will have:

$$\begin{aligned} a_1^{(1)} &= \sigma(\theta_{01}^{(1)} \times x_0 + \theta_{11}^{(1)} \times x_1 + \theta_{21}^{(1)} \times x_2) \\ &= \sigma(2 \times (-1) + 6x_1 - 6x_2) \\ a_2^{(1)} &= \sigma(\theta_{02}^{(1)} \times x_0 + \theta_{12}^{(1)} \times x_1 + \theta_{22}^{(1)} \times x_2) \\ &= \sigma((-1) \times (-1) + 8x_1 - 8x_2) \end{aligned}$$

Now the output of our network is simply applying the same rule on the inputs of our last neuron:

$$\begin{aligned} \hat{y} = a_1^{(2)} &= \sigma(z_1^{(2)}) = \sigma(\theta^{(2)} \cdot A^{(1)}) = \sigma(\theta_{01}^{(2)} \times a_0^{(1)} + \theta_{11}^{(2)} \times a_1^{(1)} + \theta_{21}^{(2)} \times a_2^{(1)}) \\ &= \sigma((-2) \times (-1) + 6a_1^{(1)} - 6a_2^{(1)}) \end{aligned}$$

You can find the schematic representation of these calculations in the following network.



ii. Compute the error of the network;

$$E = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - a_1^{(2)})^2$$

iii. *Backpropagate* the error to determine $\Delta\theta_{ij}$ for all weights θ_{ij} and updates the weight θ_{ij} .

In neural networks with backpropagation, we want to minimize the error of our network by finding the optimum weights (θ_{ij}) for our network. To do so we want to find the relation (dependency) between the error and the weights in each layer. Therefore, we use the derivatives of our error function.

$$\theta_{jk}^{(l)} \leftarrow \theta_{jk}^{(l)} + \Delta\theta_{jk}^{(l)}$$

$$\text{where: } \Delta\theta_{jk}^{(l)} = -\eta \frac{\partial E}{\partial \theta_{jk}^{(l)}} = \eta \delta_k^{(l)} a_j^{(l-1)}$$

$$\text{and } \delta_k^{(l)} = \underbrace{(1 - \sigma(z_k^{(l)})) \sigma(z_k^{(l)})}_{\sigma'(z_k^{(l)})} (y - a_k^{(l)}) \quad \text{for the last layer}$$

$$\text{or } \delta_k^{(l)} = \sigma(z_k^{(l)}) (1 - \sigma(z_k^{(l)})) \theta_{k1}^{(l+1)} \delta_1^{(l+1)} \quad \text{for the layer before}$$

Now we use our first training data (1, 0) to train the model:

$$a_1^{(1)} = \sigma(z_1^{(1)}) = \sigma(6x_1 - 6x_2 - 2) = \sigma(6 \times 1 - 6 \times 0 - 2) = \sigma(4) \cong 0.982$$

$$a_2^{(1)} = \sigma(z_2^{(1)}) = \sigma(8x_1 - 8x_2 + 1) = \sigma(8 \times 1 - 8 \times 0 + 1) = \sigma(9) \cong 0.999$$

$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma(2 + 6a_1^{(1)} - 6a_2^{(1)}) = \sigma(2 + 6 \times 0.982 - 6 \times 0.999) = \sigma(1.898) \\ \cong 0.8691$$

$$E = \frac{1}{2} (1 - 0.8691)^2 = 0.0086$$

x_1	x_2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	$E(\theta) = \frac{1}{2} (y - \hat{y})^2$
1	0	$\sigma(4) = 0.982$	$\sigma(9) = 0.999$	0.8691	1	0.0086

Now we calculate the backpropagation error. Starting from the last layer, we will have:

$$\delta_1^{(2)} = \sigma(z_1^{(2)}) (1 - \sigma(z_1^{(2)})) (y - a_1^{(2)}) \\ = \sigma(1.898) (1 - \sigma(1.898)) (1 - 0.8691) = 0.8691 (1 - 0.8691) (0.13) = 0.0149$$

$$\delta_1^{(1)} = \sigma(z_1^{(1)}) (1 - \sigma(z_1^{(1)})) \theta_{11}^{(2)} \delta_1^{(2)} \\ = \sigma(4) (1 - \sigma(4)) \times 6 \times 0.0149 = 0.982 (1 - 0.982) 0.0882 = 0.0016$$

$$\delta_2^{(1)} = \sigma(z_2^{(1)}) (1 - \sigma(z_2^{(1)})) \theta_{21}^{(2)} \delta_1^{(2)} \\ = \sigma(9) (1 - \sigma(9)) \times (-6) \times 0.0149 = 0.999 (1 - 0.999) (-0.0882) = -0.000001103$$

Using the learning rate of $\eta = 0.7$ we can now calculate $\Delta\theta_{jk}^{(l)}$:

$$\Delta\theta_{01}^{(2)} = \eta \delta_1^{(2)} a_0^{(1)} = 0.7 \times 0.0149 \times (-1) = -0.0104$$

$$\Delta\theta_{11}^{(2)} = \eta \delta_1^{(2)} a_1^{(1)} = 0.7 \times 0.0149 \times 0.982 = 0.0102$$

$$\Delta\theta_{21}^{(2)} = \eta \delta_1^{(2)} a_2^{(1)} = 0.7 \times 0.0149 \times 0.999 = 0.0104$$

$$\Delta\theta_{01}^{(1)} = \eta \delta_1^{(1)} x_0 = 0.7 \times 0.0016 \times (-1) = -0.0011$$

$$\Delta\theta_{11}^{(1)} = \eta \delta_1^{(1)} x_1 = 0.7 \times 0.0016 \times 1 = 0.0011$$

$$\Delta\theta_{21}^{(1)} = \eta \delta_1^{(1)} x_2 = 0.7 \times 0.0016 \times 0 = 0$$

$$\Delta\theta_{02}^{(1)} = \eta \delta_2^{(1)} x_0 = 0.7 \times 0.000001103 \times (-1) \cong 0$$

$$\Delta\theta_{12}^{(1)} = \eta \delta_2^{(1)} x_1 = 0.7 \times 0.000001103 \times 1 \cong 0$$

$$\Delta\theta_{22}^{(1)} = \eta \delta_2^{(1)} x_2 = 0.7 \times 0.000001103 \times 0 = 0$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2 + (-0.0104) = -2.0104$$

$$\theta_{11}^{(2)} = \theta_{11}^{(2)} + \Delta\theta_{11}^{(2)} = 6 + 0.0102 = 6.0102$$

$$\theta_{21}^{(2)} = \theta_{21}^{(2)} + \Delta\theta_{21}^{(2)} = -6 + 0.0104 = -5.9896$$

$$\theta_{01}^{(1)} = \theta_{01}^{(1)} + \Delta\theta_{01}^{(1)} = 2 + (-0.0011) = 1.9989$$

$$\theta_{11}^{(1)} = \theta_{11}^{(1)} + \Delta\theta_{11}^{(1)} = 6 + 0.0011 = 6.0011$$

The rest of the weights do not change i.e.

$$\theta_{02}^{(1)} = -1, \theta_{21}^{(1)} = -6, \theta_{22}^{(1)} = -8, \Delta\theta_{12}^{(1)} = 8$$

Now we use our next training instance (0,1):

$$\begin{aligned} a_1^{(1)} &= \sigma(z_1^{(1)}) = \sigma(1.9989 + 6.00112x_1 - 6x_2) = \sigma(6.00112 \times 0 - 6 \times 1 + 1.9989) \\ &= \sigma(-7.9989) \cong 3.387e - 4 \end{aligned}$$

$$a_2^{(1)} = \sigma(z_2^{(1)}) = \sigma(-1 + 8x_1 - 8x_2) = \sigma(8 \times 0 - 8 \times 1 + 1) = \sigma(-7) \cong 9.11e - 4$$

$$\begin{aligned} a_1^{(2)} &= \sigma(z_1^{(2)}) = \sigma(-2.01029 + 6.0101 \times \sigma(-7.9989) - 5.98972 \times \sigma(-7)) = \sigma(2.007) \\ &= 0.8815 \end{aligned}$$

$$E = \frac{1}{2} (1 - 0.8815)^2 = 0.007$$

x_1	x_2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	$E(\theta) = \frac{1}{2} (y - \hat{y})^2$
1	0	$\sigma(4)$	$\sigma(9)$	0.8691	1	0.0086
0	1	$\sigma(-7.9989)$	$\sigma(-7)$	0.8815	1	0.007

To calculate the backpropagation error, we will have¹:

$$\begin{aligned} \delta_1^{(2)} &= \sigma(z_1^{(2)}) (1 - \sigma(z_1^{(2)})) (y - a_1^{(2)}) \\ &= \sigma(2.007) (1 - \sigma(2.007)) (1 - 0.8815) = 0.0124 \\ \delta_1^{(1)} &= \sigma(z_1^{(1)}) (1 - \sigma(z_1^{(1)})) \theta_{11}^{(2)} \delta_1^{(2)} \\ &= \sigma(-7.9989) (1 - \sigma(-7.9989)) \times 6.0102 \times 0.0124 \cong 0 \\ \delta_2^{(1)} &= \sigma(z_2^{(1)}) (1 - \sigma(z_2^{(1)})) \theta_{21}^{(2)} \delta_1^{(2)} \\ &= \sigma(-7) (1 - \sigma(-7)) \times (-5.9896) \times 0.0124 \cong 0 \end{aligned}$$

$$\Delta\theta_{01}^{(2)} = \eta \delta_1^{(2)} a_0^{(1)} = 0.7 \times 0.0124 \times (-1) = -0.0087$$

$$\Delta\theta_{11}^{(2)} = \eta \delta_1^{(2)} a_1^{(1)} = 0.7 \times 0.0124 \times 3.387e - 4 \cong 0$$

$$\Delta\theta_{21}^{(2)} = \eta \delta_1^{(2)} a_2^{(1)} = 0.7 \times 0.0124 \times 9.11e - 4 \cong 0$$

$$\Delta\theta_{01}^{(1)} = \eta \delta_1^{(1)} x_0 = 0.7 \times (2.496e - 5) \times (-1) \cong 0$$

$$\Delta\theta_{11}^{(1)} = \eta \delta_1^{(1)} x_1 = 0.7 \times (2.496e - 5) \times 0 = 0$$

$$\Delta\theta_{21}^{(1)} = \eta \delta_1^{(1)} x_2 = 0.7 \times (2.496e - 5) \times 1 \cong 0$$

$$\Delta\theta_{02}^{(1)} = \eta \delta_2^{(1)} x_0 = 0.7 \times (-6.7454e - 5) \times (-1) \cong 0$$

$$\Delta\theta_{12}^{(1)} = \eta \delta_2^{(1)} x_1 = 0.7 \times (-6.7454e - 5) \times 0 = 0$$

$$\Delta\theta_{22}^{(1)} = \eta \delta_2^{(1)} x_2 = 0.7 \times (-6.7454e - 5) \times 1 \cong 0$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(1)} + \Delta\theta_{01}^{(2)} = -2.01029 + (-0.0087) \cong -2.0191$$

¹ In this assignment, we make the simplifying assumption that any update $< 1e-4 \cong 0$ to keep the computations feasible.

Since the other values are very small, the rest of the weights do not change.

Now we use our next training instance (1, 1):

$$\begin{aligned}
 a_1^{(1)} &= \sigma(z_1^{(1)}) = \sigma(6.0102x_1 - 5.9896x_2 + 2.0191) \\
 &= \sigma(6.0102 \times 1 - 5.9896 \times 1 + 2.0191) = \sigma(-1.9978) \\
 a_2^{(1)} &= \sigma(z_2^{(1)}) = \sigma(8x_1 - 8x_2 + 0.9999) = \sigma(8 \times 1 - 8 \times 1 + 0.9999) = \sigma(0.9999) \\
 a_1^{(2)} &= \sigma(z_1^{(2)}) = \sigma(2.0191 + 6.0102 \times \sigma(-1.9978) - 5.9896 \times \sigma(0.9999)) \\
 &= \sigma(-1.6416) = 0.1622 \\
 E &= \frac{1}{2} (0 - 0.1622)^2 = 0.0132
 \end{aligned}$$

x_1	x_2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	$E(\theta) = \frac{1}{2} (y - \hat{y})^2$
1	0	$\sigma(4)$	$\sigma(9)$	0.8691	1	0.0086
0	1	$\sigma(-7.9989)$	$\sigma(-7)$	0.8815	1	0.007
1	1	$\sigma(-1.9978)$	$\sigma(0.9999)$	0.1622	0	0.0132

To calculate the backpropagation error, we will have:

$$\begin{aligned}
 \delta_1^{(2)} &= \sigma(z_1^{(2)}) (1 - \sigma(z_1^{(2)})) (y - a_1^{(2)}) = -0.0221 \\
 \delta_1^{(1)} &= \sigma(z_1^{(1)}) (1 - \sigma(z_1^{(1)})) \theta_{11}^{(2)} \delta_1^{(2)} = -0.0139 \\
 \delta_2^{(1)} &= \sigma(z_2^{(1)}) (1 - \sigma(z_2^{(1)})) \theta_{21}^{(2)} \delta_1^{(2)} = 0.0260
 \end{aligned}$$

$$\begin{aligned}
 \Delta\theta_{01}^{(2)} &= \eta \delta_1^{(2)} a_0^{(1)} = 0.0154 \\
 \Delta\theta_{11}^{(2)} &= \eta \delta_1^{(2)} a_1^{(1)} = -0.0018 \\
 \Delta\theta_{21}^{(2)} &= \eta \delta_1^{(2)} a_2^{(1)} = -0.0113 \\
 \Delta\theta_{01}^{(1)} &= \eta \delta_1^{(1)} x_0 = -0.0098 \\
 \Delta\theta_{11}^{(1)} &= \eta \delta_1^{(1)} x_1 = -0.0098 \\
 \Delta\theta_{21}^{(1)} &= \eta \delta_1^{(1)} x_2 = 0.0182 \\
 \Delta\theta_{02}^{(1)} &= \eta \delta_2^{(1)} x_0 = 0.0182 \\
 \Delta\theta_{12}^{(1)} &= \eta \delta_2^{(1)} x_1 = -0.0098 \\
 \Delta\theta_{22}^{(1)} &= \eta \delta_2^{(1)} x_2 = 0.0182
 \end{aligned}$$

Based on these results we can update the network weights:

$$\begin{aligned}
 \theta_{01}^{(2)} &= \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2.0037 \\
 \theta_{11}^{(2)} &= \theta_{11}^{(2)} + \Delta\theta_{11}^{(2)} = 6.0084 \\
 \theta_{21}^{(2)} &= \theta_{21}^{(2)} + \Delta\theta_{21}^{(2)} = -6.0009 \\
 \theta_{01}^{(1)} &= \theta_{01}^{(1)} + \Delta\theta_{01}^{(1)} = 2.0086 \\
 \theta_{11}^{(1)} &= \theta_{11}^{(1)} + \Delta\theta_{11}^{(1)} = 5.9913 \\
 \theta_{21}^{(1)} &= \theta_{21}^{(1)} + \Delta\theta_{21}^{(1)} = -6.0097
 \end{aligned}$$

$$\theta_{02}^{(1)} = \theta_{02}^{(1)} + \Delta\theta_{02}^{(1)} = -1.0181$$

$$\theta_{12}^{(1)} = \theta_{12}^{(1)} + \Delta\theta_{12}^{(1)} = 8.0182$$

$$\theta_{22}^{(1)} = \theta_{22}^{(1)} + \Delta\theta_{22}^{(1)} = -7.9819$$

Now we use our next training instance (0, 0):

$$\begin{aligned} a_1^{(1)} &= \sigma(z_1^{(1)}) = \sigma(5.9913x_1 - 6.00097x_2 + 2.0037) \\ &= \sigma(5.9913 \times 0 - 6.00097 \times 0 + 2.0037) = \sigma(2.0037) \end{aligned}$$

$$a_2^{(1)} = \sigma(z_2^{(1)}) = \sigma(8.0182x_1 - 7.9819x_2 + 1.0181) = \sigma(1.0181)$$

$$\begin{aligned} a_1^{(2)} &= \sigma(z_1^{(2)}) = \sigma(2.0037 + 6.0084 \times \sigma(2.0037) - 6.0009 \times \sigma(1.0181)) \\ &= \sigma(-1.6938) = 0.1553 \end{aligned}$$

$$E = \frac{1}{2} (0.1553)^2 = 0.0121$$

x_1	x_2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	$E(\theta) = \frac{1}{2} (y - \hat{y})^2$
1	0	$\sigma(4)$	$\sigma(9)$	0.8691	1	0.0086
0	1	$\sigma(-7.9989)$	$\sigma(-7)$	0.8815	1	0.007
1	1	$\sigma(-1.9978)$	$\sigma(0.9999)$	0.1622	0	0.0132
0	0	$\sigma(-2.0086)$	$\sigma(1.0181)$	0.1553	0	0.0121

To calculate the backpropagation error, we will have:

$$\delta_1^{(2)} = \sigma(z_1^{(2)}) (1 - \sigma(z_1^{(2)})) (y - a_1^{(2)}) = -0.0204$$

$$\delta_1^{(1)} = \sigma(z_1^{(1)}) (1 - \sigma(z_1^{(1)})) \theta_{11}^{(2)} \delta_1^{(2)} = -0.0128$$

$$\delta_2^{(1)} = \sigma(z_2^{(1)}) (1 - \sigma(z_2^{(1)})) \theta_{21}^{(2)} \delta_1^{(2)} = 0.0238$$

$$\Delta\theta_{01}^{(2)} = \eta \delta_1^{(2)} a_0^{(1)} = 0.0143$$

$$\Delta\theta_{11}^{(2)} = \eta \delta_1^{(2)} a_1^{(1)} = -0.0017$$

$$\Delta\theta_{21}^{(2)} = \eta \delta_1^{(2)} a_2^{(1)} = -0.0105$$

$$\Delta\theta_{01}^{(1)} = \eta \delta_1^{(1)} x_0 = -0.0098$$

$$\Delta\theta_{11}^{(1)} = \eta \delta_1^{(1)} x_1 = 0$$

$$\Delta\theta_{21}^{(1)} = \eta \delta_1^{(1)} x_2 = 0$$

$$\Delta\theta_{02}^{(1)} = \eta \delta_2^{(1)} x_0 = 0.0167$$

$$\Delta\theta_{12}^{(1)} = \eta \delta_2^{(1)} x_1 = 0$$

$$\Delta\theta_{22}^{(1)} = \eta \delta_2^{(1)} x_2 = 0$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2.0176$$

$$\theta_{11}^{(2)} = \theta_{11}^{(2)} + \Delta\theta_{11}^{(2)} = 6.0067$$

$$\theta_{21}^{(2)} = \theta_{21}^{(2)} + \Delta\theta_{21}^{(2)} = -6.0113$$

$$\theta_{01}^{(1)} = \theta_{01}^{(1)} + \Delta\theta_{01}^{(1)} = 2.0176$$

$$\theta_{11}^{(1)} = \theta_{11}^{(1)} + \Delta\theta_{11}^{(1)} = 5.9913$$

$$\theta_{21}^{(1)} = \theta_{21}^{(1)} + \Delta\theta_{21}^{(1)} = -6.0097$$

$$\theta_{02}^{(1)} = \theta_{02}^{(1)} + \Delta\theta_{02}^{(1)} = -1.0348$$

$$\theta_{12}^{(1)} = \theta_{12}^{(1)} + \Delta\theta_{12}^{(1)} = 8.0182$$

$$\theta_{22}^{(1)} = \theta_{22}^{(1)} + \Delta\theta_{22}^{(1)} = -7.9819$$

And this would be the end of epoch one! 😊