School of Computing and Information Systems The University of Melbourne
COMP30027 Machine Learning (Semester 1, 2021)

Sample solutions: Week 11

1. Why is a perceptron (which uses a **sigmoid** activation function) equivalent to logistic regression?

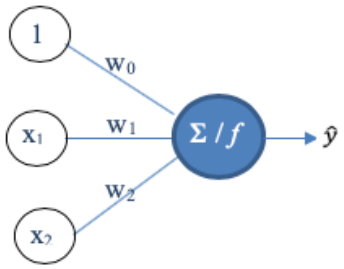
   A perceptron has a weight associated with each input (attribute); the output is acquired by applying the activation function. This activation function can be the **step function** (as shown in the lectures) or any other (appropriate) function. If we use the **sigmoid** activation function ($\sigma(x) = f(x) = \frac{1}{1+e^{-x}}$) to the linear combination of inputs ($w_0 + w_1x_1 + w_2x_2 + \cdots$), which simplifies to $f(w^Tx) = \sigma(w^Tx)$— this is the same as the logistic regression function.

2. Consider the following training set:

(x_1, x_2)	y
(0,0)	0
(0,1)	1
(1,1)	1

Consider the initial weight function as $w = \{w_0, w_1, w_2\}$ = {0.2, -0.4, 0.1}, the bias of 1 and the activation function of the perceptron as the step function of $f = \begin{cases} 1 & if\ \Sigma > 0 \\ 0 & otherwise \end{cases}$.

a) Draw the perceptron graph and calculate the accuracy of the perceptron on the training data before training?



To calculate the accuracy of the system we first need to calculate the output (prediction) of our perceptron and then compare it the actual labels of the class.

(x_1, x_2)	$\Sigma = w_0 + w_1x_1 + w_2x_2$	$\hat{y} = f(\Sigma)$	y
(0,0)	0.2 – 0.4 x 0 + 0.1 x 0 = 0.2	f(0.2) = 1	0
(0,1)	0.2 – 0.4 x 0 + 0.1 x 1 = 0.3	f(0.3) = 1	1
(1,1)	0.2 – 0.4 x 1 + 0.1 x 1 = -0.1	f(-0.1) = 0	1

As you can see one of the predictions (outputs) of our perceptron match the actual label and therefore the accuracy of our perceptron is $\frac{1}{3}$ at this stage.

b) Using the perceptron *learning rule* and the learning rate of $\lambda = 0.2$. Train the perceptron **for one epoch**. What are the weights after the training?

Remember the perceptron weights learning rule in iteration *t* and for train instance *i* is as follows:

$$w_j \leftarrow w_j + \lambda(y^i - \hat{y}^i)\, x_j^i$$

For epoch 1 we will have:

(x_1, x_2)	$\Sigma = w_0 + w_1 x_1 + w_2 x_2$	$\hat{y} = f(\Sigma)$	y
(0,0)	0.2 - 0.4 x 0 + 0.1 x 0 = 0.2 Update w: $w_0 = w_0 + \lambda(y^1 - \hat{y}^1)\, x_0^1 = 0.2 + 0.2\,(0-1)\,1 = 0$ $w_1 = w_1 + \lambda(y^1 - \hat{y}^1)\, x_1^1 = -0.4 + 0.2\,(0-1)\,0 = -0.4$ $w_2 = w_2 + \lambda(y^1 - \hat{y}^1)\, x_2^1 = 0.1 + 0.2\,(0-1)\,0 = 0.1$	1	0
(0,1)	0 – 0.4 x 0 + 0.1 x 1 = 0.1 Correct prediction → no update	1	1
(1,1)	0 – 0.4 x 1 + 0.1 x 1 = – 0.3 Update w: $w_0 = w_0 + \lambda(y^3 - \hat{y}^3)\, x_0^3 = 0 + 0.2\,(1-0)\,1 = 0.2$ $w_1 = w_1 + \lambda\,(y^3 - \hat{y}^3)\, x_1^3 = -0.4 + 0.2\,(1-0)\,1 = -0.2$ $w_2 = w_2 + \lambda(y^3 - \hat{y}^3)\, x_2^3 = 0.1 + 0.2\,(1-0)\,1 = 0.3$	0	1

c) What is the accuracy of the perceptron on the training data after training for one epoch? Did the accuracy improve?

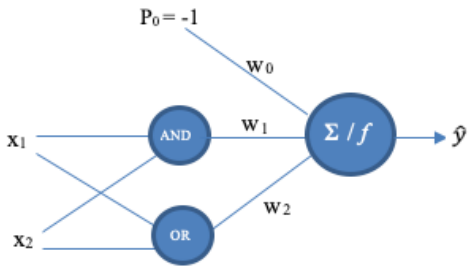
With the new weights we get

- for instance (0,0) with y=0: 0.2 – 0.2 x 0 + 0.3 x 0 = 0.2; f(0.2) = 1 ; **incorrect**
- for instance (0,1) with y=1: 0.2 – 0.2 x 0 + 0.3 x 1 = 0.5; f(0.5) = 1 ; **correct**
- for instance (1,1) with y=1: 0.2 – 0.2 x 1 + 0.3 x 1 = 0.1; f(0.3) = 1 ; **correct**

The accuracy of our perceptron is now $\frac{2}{3}$. So the accuracy of the system has been improved ☺

3. Consider the two layers deep network illustrated below. It is composed of three perceptrons. The two perceptrons of the first layer implement the AND and OR function, respectively.

   Determine the weights w_1, w_2 and bias w_0 such that the network implements the XOR function. The initial weights are set to zero, i.e., $w_0 = w_1 = w_2 = 0$, and the learning rate λ (lambda) is set to 0.1.



Notes:

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

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

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

Learning Algorithm for XOR:

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

To calculate the *output* of the two layer 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 $x_1\ XOR\ x_2$
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 layer 2 perceptron (XOR) are $P_i =< p_0, p_1, p_2 > = < -1 , p_1 , p_2 >$ and the parameters are w =<w_0, w_1 , w_2> = <0, 0, 0>. For the first epoch we have:

ep	*P* $<p_0, p_1, p_2>$	z = w . *P* $w_0 \times p_0 + w_1 \times p_1 + w_2 \times p_2$	$\hat{y}$ *f(z)*	*y*	Δw_{1i} $\lambda\,(y - \hat{y})\, p_i$
1	<-1,0,1>	0 x (-1) + 0 x 0 + 0 x 1=0 Update w to <0, 0, 0> + <-0.1,0 ,0.1>=<-0.1,0 ,0.1>	f(0)=0	1	0.1 x (1-0) x <-1, 0,1> = <-0.1, 0, 0.1>
	<-1,0,1>	(-0.1) x -1 + 0 x 0 + (0.1) x 1 = 0.2	f(0.2)=1	1	No change required
	<-1,1,1>	(-0.1) x -1 + 0 x 1 + (0.1) x 1 = 0.2 Update w to <-0.1,0 ,0.1> + <0.1,-0.1 ,-0.1> = < 0, -0.1 , 0>	f(0.2)=1	0	0.1 x (0-1) x <-1, 1,1> = <0.1,-0.1 ,-0.1>
	<-1,0,0>	0 x -1 + (-0.1) x 0 + 0 x 0 = 0	f(0)=0	0	No change required

We continue the iteration as follow:

ep	*P* $<p_0, p_1, p_2>$	z = w . *P* $w_0 \times p_0 + w_1 \times p_1 + w_2 \times p_2$	$\hat{y}$ *f(z)*	*y*	Δw_{1i} $\lambda\,(y - \hat{y})\, p_i$
2	<-1,0,1>	0 x -1 + (-0.1) x 0+ 0 x 1 = 0 Update w to < 0, -0.1 , 0> + <-0.1, 0, 0.1> = < -0.1, -0.1 , 0.1>	f(0)=0	1	0.1 x (1-0) x <-1, 0,1> = <-0.1, 0, 0.1>
	<-1,0,1>	(-0.1) x -1 + (-0.1) x 0 + (0.1) x 1 = 0.2	f(0.2)=1	1	No change required
	<-1,1,1>	(-0.1) x -1 + (-0.1) x 1 + (0.1) x 1 = 0.1 Update w to < -0.1, -0.1 , 0.1> + <0.1,-0.1 ,-0.1> = <0, -0.2, 0>	f(0.1)=1	0	0.1 x (0-1) x <-1, 1,1> = <0.1,-0.1 ,-0.1>
	<-1,0,0>	0 x -1 + (-0.2) x 0 + 0 x 0 = 0	f(0)=0	0	No change required

ep	*P* $<p_0, p_1, p_2>$	z = w . *P* $w_0 \times p_0 + w_1 \times p_1 + w_2 \times p_2$	$\hat{y}$ *f(z)*	*y*	Δw_{1i} $\lambda\,(y - \hat{y})\,p_i$
	$<p_0, p_1, p_2>$	$w_0 \times p_0 + w_1 \times p_1 + w_2 \times p_2$	*f(z)*		$\lambda\,(y - \hat{y})\,p_i$
	<-1,0,1>	(-0.1) x -1 + (-0.2) x 0 + (0.1) x 1 = 0.2	f(0.2)=1	1	No change required
	<-1,1,1>	(-0.1) x -1 + (-0.2) x 1 + (0.1) x 1 = 0	f(0)=0	0	No change required
	<-1,0,0>	(-0.1) x -1 + (-0.2) x 0 + (0.1) x 0 = 0.1 Update w to < -0.1, -0.2 , 0.1> + < 0.1 , 0, 0> = < 0, -0.2 , 0.1>	f(0.1)=1	0	0.1 x (0-1) x <-1, 0, 0> = < 0.1 , 0, 0>

ep	*P* $<p_0, p_1, p_2>$	z = w . *P* $w_0 \times p_0 + w_1 \times p_1 + w_2 \times p_2$	$\hat{y}$ *f(z)*	*y*	Δw_{1i} $\lambda\,(y - \hat{y})\,p_i$
4	<-1,0,1>	0 x -1 + (-0.2) x 0 + (0.1) x 1 = 0.1	f(0.1)=1	1	No change required
	<-1,0,1>	0 x -1 + (-0.2) x 0 + (0.1) x 1 = 0.1	f(0.1)=1	1	No change required
	<-1,1,1>	0 x -1 + (-0.2) x 1 + (0.1) x 1 = -0.1	f(-0.1)=0	0	No change required
	<-1,0,0>	0 x -1 + (-0.2) x 0 + (0.1) x 0 = 0	f(0)=0	0	No change required

Since in the last epoch we didn't have any updated for our weights (w_1), the algorithm converges here.

So for our network to perform XOR function, the final w_1 would be w_1= < 0, -0.2 , 0.1> . In other words $w_0 = 0$, $w_1 = -0.2$ and $w_2 = 0.1$

4. Why is a neural network suitable for deep learning? What is significant about the representation that we attempt to learn?

Hypothetically, the weights across the network describe some useful properties of the input attributes. In effect, we hope to auto–magically engineer the necessary features to solve our problem, based only on the simplest inputs (where we hopefully haven't already introduced a bias).

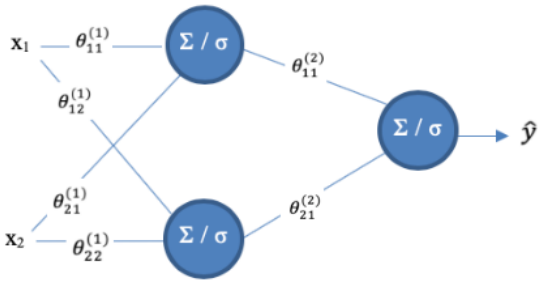
The most interesting about the representation is that it is simultaneously useful for solving the problem and un-interpretable by humans. Although the claims of the automaton "learning" are probably over–blown, it can indeed discover properties of the data that were previously unknown to be useful (given enough data).

The subsequent "embedding" (weights of the final hidden layer of neurons) can also be used as a dense instance representation, which is sometimes helpful in unexpected problems.

5. Describe the mathematical formula of a multilayer perceptron with 1 hidden layer. Assume the input size is 1000, the hidden layer size is 100, and the output size is 20. Identify the parameters of the model, and their size.

The mathematical formula for a MLP with one hidden layer is: g(x) = $f_2(f_1(x\,.\,w_1 + b_1)\,.\,w_2 + b_2)$, where x ∈ (1, 1000), w_1 ∈ (1000, 100), b_1 ∈ (1, 100), w_2 ∈ (100, 20), b_2 ∈ (1, 20), and f_1 is an activation function such as ReLU or tanh, and f_2 is softmax (why?).

6. [OPTIONAL] Consider the following multilayer perceptron.



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

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:

$$\theta_{01}^{(1)} = 2 \qquad \theta_{02}^{(1)} = -1 \qquad \theta_{01}^{(2)} = -2$$
$$\theta_{11}^{(1)} = 6 \qquad \theta_{12}^{(1)} = 8 \qquad \theta_{11}^{(2)} = 6$$
$$\theta_{21}^{(1)} = -6 \qquad \theta_{22}^{(1)} = -8 \qquad \theta_{21}^{(2)} = -6$$

- 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:

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

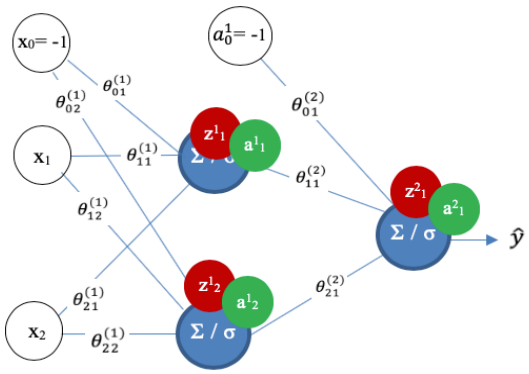
Therefore, we will have:

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

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

$$\hat{y} = a_1^{(2)} = \sigma\left(z_1^{(2)}\right) = \sigma\left(\theta^{(2)}.A^{(1)}\right) = \sigma\left(\theta_{01}^{(2)} \times a_0^{(1)} + \theta_{11}^{(2)} \times a_1^{(1)} + \theta_{21}^{(2)} \times a_2^{(1)}\right)$$
$$= \sigma((-2) \times (-1) + 6a_1^{(1)} - 6a_2^{(1)})$$

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.

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

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

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

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

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:

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

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

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

Using the learning rate of η = 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.

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

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

$$a_1^{(1)} = \sigma\left(z_1^{(1)}\right) = \sigma(1.9989 + 6.00112x_1 - 6x_2) = \sigma(6.00112 \times 0 - 6 \times 1 + 1.9989)$$
$$= \sigma(-7.9989) \cong 3.387e-4$$
$$a_2^{(1)} = \sigma\left(z_2^{(1)}\right) = \sigma(-1 + 8x_1 - 8x_2) = \sigma(8 \times 0 - 8 \times 1 + 1) = \sigma(-7) \cong 9.11e-4$$
$$a_1^{(2)} = \sigma\left(z_1^{(2)}\right) = \sigma\left(-2.01029 + 6.0101 \times \sigma(-7.9989) - 5.98972 \times \sigma(-7)\right) = \sigma(2.007)$$
$$= 0.8815$$
$$E = \frac{1}{2}(1 - 0.8815)^2 = 0.007$$

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 have[1]:

$$\delta_1^{(2)} = \sigma\left(z_1^{(2)}\right)\left(1 - \sigma\left(z_1^{(2)}\right)\right)\left(y - a_1^{(2)}\right)$$
$$= \sigma(2.007)\left(1 - \sigma(2.007)\right)(1 - 0.8815) = 0.0124$$
$$\delta_1^{(1)} = \sigma\left(z_1^{(1)}\right)\left(1 - \sigma\left(z_1^{(1)}\right)\right)\theta_{11}^{(2)}\delta_1^{(2)}$$

[1] In this assignment, we make the simplifying assumption that any update < 1e-4 ≅ 0 to keep the computations feasible.

$$= \sigma(-7.9989)\big(1-\sigma(-7.9989)\big) \times 6.0102 \times 0.0124 \cong 0$$

$$\delta_2^{(1)} = \sigma\big(z_2^{(1)}\big)\Big(1-\sigma\big(z_2^{(1)}\big)\Big)\theta_{21}^{(2)}\delta_1^{(2)}$$
$$= \sigma(-7)\big(1-\sigma(-7)\big) \times (-5.9896) \times 0.0124 \cong 0$$

$$\Delta\theta_{01}^{(2)} = \eta\delta_1^{(2)}a_0^{(1)} = 0.7 \times 0.0124 \times (-1) = -0.0087$$
$$\Delta\theta_{11}^{(2)} = \eta\delta_1^{(2)}a_1^{(1)} = 0.7 \times 0.0124 \times 3.387e-4 \cong 0$$
$$\Delta\theta_{21}^{(2)} = \eta\delta_1^{(2)}a_2^{(1)} = 0.7 \times 0.0124 \times 9.11e-4 \cong 0$$
$$\Delta\theta_{01}^{(1)} = \eta\delta_1^{(1)}x_0 = 0.7 \times (2.496\,e-5) \times (-1) \cong 0$$
$$\Delta\theta_{11}^{(1)} = \eta\delta_1^{(1)}x_1 = 0.7 \times (2.496\,e-5) \times 0 = 0$$
$$\Delta\theta_{21}^{(1)} = \eta\delta_1^{(1)}x_2 = 0.7 \times (2.496\,e-5) \times 1 \cong 0$$

$$\Delta\theta_{02}^{(1)} = \eta\delta_2^{(1)}x_0 = 0.7 \times (-6.7454\,e-5) \times (-1) \cong 0$$
$$\Delta\theta_{12}^{(1)} = \eta\delta_2^{(1)}x_1 = 0.7 \times (-6.7454\,e-5) \times 0 = 0$$
$$\Delta\theta_{22}^{(1)} = \eta\delta_2^{(1)}x_2 = 0.7 \times (-6.7454\,e-5) \times 1 \cong 0$$

Based on these results we can update the network weights:

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

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

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

$$a_1^{(1)} = \sigma\big(z_1^{(1)}\big) = \sigma(6.0102x_1 - 5.9896x_2 + 2.0191)$$
$$= \sigma(6.0102 \times 1 - 5.9896 \times 1 + 2.0191) = \sigma(-1.9978)$$

$$a_2^{(1)} = \sigma\big(z_2^{(1)}\big) = \sigma(8x_1 - 8x_2 + 0.9999) = \sigma(8 \times 1 - 8 \times 1 + 0.9999) = \sigma(0.9999)$$

$$a_1^{(2)} = \sigma\big(z_1^{(2)}\big) = \sigma\big(2.0191 + 6.0102 \times \sigma(-1.9978) - 5.9896 \times \sigma(0.9999)\big)$$
$$= \sigma(-1.6416) = 0.1622$$

$$E = \frac{1}{2}(0 - 0.1622)^2 = 0.0132$$

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:

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

$$\Delta\theta_{01}^{(2)} = \eta\delta_1^{(2)}a_0^{(1)} = 0.0154$$
$$\Delta\theta_{11}^{(2)} = \eta\delta_1^{(2)}a_1^{(1)} = -0.0018$$
$$\Delta\theta_{21}^{(2)} = \eta\delta_1^{(2)}a_2^{(1)} = -0.0113$$

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

$$\Delta\theta_{11}^{(1)} = \eta\delta_1^{(1)}x_1 = -0.0098$$
$$\Delta\theta_{21}^{(1)} = \eta\delta_1^{(1)}x_2 = 0.0182$$

$$\Delta\theta_{02}^{(1)} = \eta\delta_2^{(1)}x_0 = 0.0182$$
$$\Delta\theta_{12}^{(1)} = \eta\delta_2^{(1)}x_1 = -0.0098$$
$$\Delta\theta_{22}^{(1)} = \eta\delta_2^{(1)}x_2 = 0.0182$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2.0037$$
$$\theta_{11}^{(2)} = \theta_{11}^{(2)} + \Delta\theta_{11}^{(2)} = 6.0084$$
$$\theta_{21}^{(2)} = \theta_{21}^{(2)} + \Delta\theta_{21}^{(2)} = -6.0009$$

$$\theta_{01}^{(1)} = \theta_{01}^{(1)} + \Delta\theta_{01}^{(1)} = 2.0086$$
$$\theta_{11}^{(1)} = \theta_{11}^{(1)} + \Delta\theta_{11}^{(1)} = 5.9913$$
$$\theta_{21}^{(1)} = \theta_{21}^{(1)} + \Delta\theta_{21}^{(1)} = -6.0097$$

$$\theta_{02}^{(1)} = \theta_{02}^{(1)} + \Delta\theta_{02}^{(1)} = -1.0181$$
$$\theta_{12}^{(1)} = \theta_{12}^{(1)} + \Delta\theta_{12}^{(1)} = 8.0182$$
$$\theta_{22}^{(1)} = \theta_{22}^{(1)} + \Delta\theta_{22}^{(1)} = -7.9819$$

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

$$a_1^{(1)} = \sigma\left(z_1^{(1)}\right) = \sigma(5.9913x_1 - 6.00097x_2 + 2.0037)$$
$$= \sigma(5.9913 \times 0 - 6.00097 \times 0 + 2.0086) = \sigma(2.0086)$$
$$a_2^{(1)} = \sigma\left(z_2^{(1)}\right) = \sigma(8.0182x_1 - 7.9819x_2 + 1.0181) = \sigma(1.0181)$$
$$a_1^{(2)} = \sigma\left(z_1^{(2)}\right) = \sigma\left(2.0037 + 6.0084 \times \sigma(2.0086) - 6.0009 \times \sigma(1.0181)\right)$$
$$= \sigma(-1.6938) = 0.1553$$
$$E = \frac{1}{2}(00.1553)^2 = 0.0121$$

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:

$$\delta_1^{(2)} = \sigma\left(z_1^{(2)}\right)\left(1 - \sigma\left(z_1^{(2)}\right)\right)\left(y - a_1^{(2)}\right) = -0.0204$$
$$\delta_1^{(1)} = \sigma\left(z_1^{(1)}\right)\left(1 - \sigma\left(z_1^{(1)}\right)\right)\theta_{11}^{(2)}\delta_1^{(2)} = -0.0128$$
$$\delta_2^{(1)} = \sigma\left(z_2^{(1)}\right)\left(1 - \sigma\left(z_2^{(1)}\right)\right)\theta_{21}^{(2)}\delta_1^{(2)} = 0.0238$$

$$\Delta\theta_{01}^{(2)} = \eta\delta_1^{(2)}a_0^{(1)} = 0.0143$$
$$\Delta\theta_{11}^{(2)} = \eta\delta_1^{(2)}a_1^{(1)} = -0.0017$$
$$\Delta\theta_{21}^{(2)} = \eta\delta_1^{(2)}a_2^{(1)} = -0.0105$$

$$\Delta\theta_{01}^{(1)} = \eta\delta_1^{(1)}x_0 = -0.0098$$
$$\Delta\theta_{11}^{(1)} = \eta\delta_1^{(1)}x_1 = 0$$
$$\Delta\theta_{21}^{(1)} = \eta\delta_1^{(1)}x_2 = 0$$

$$\Delta\theta_{02}^{(1)} = \eta\delta_2^{(1)}x_0 = 0.0167$$
$$\Delta\theta_{12}^{(1)} = \eta\delta_2^{(1)}x_1 = 0$$
$$\Delta\theta_{22}^{(1)} = \eta\delta_2^{(1)}x_2 = 0$$

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! ☺