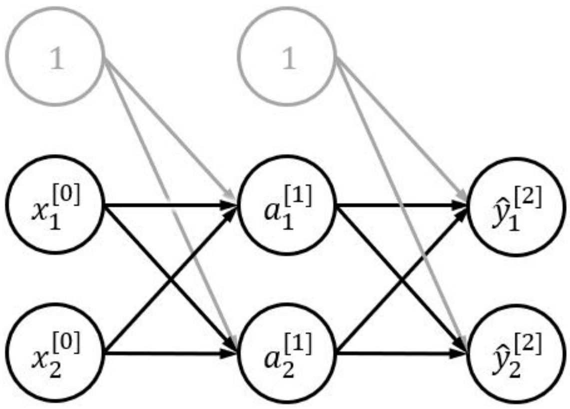
National University of Singapore
School of Computing
CS3244: Machine Learning
Solution to Tutorial 07

Perceptrons and Neural Networks

Colab Notebook Solutions : Perceptrons and Neural Networks

1. **Backpropagation algorithm.** In this question, we're going to use a neural network with **a 2-d input, one hidden layer with two neurons and two output neurons**. Additionally, the hidden neurons and the input will **include a bias**. We use **ReLU function** as the nonlinear activation function.

   Here's the basic structure:



   (a) Suppose there is a data input $\mathbf{x} = (2, 3)^\top$ and the actual output label is $\mathbf{y} = (0.1, 0.9)^\top$. The weights for the network are

$$\boldsymbol{W}^{[1]} = \begin{bmatrix} 0.1 & 0.1 \\ -0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix}, \boldsymbol{W}^{[2]} = \begin{bmatrix} 0.1 & 0.1 \\ 0.5 & -0.6 \\ 0.7 & -0.8 \end{bmatrix},$$

   Calculate the following values after forward propagation:
   $\mathbf{a}^{[1]}$, $\hat{\mathbf{y}}^{[2]}$ and $L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})$.

$$\mathbf{a}^{[1]} = ReLU((\mathbf{W}^{[1]})^\top \mathbf{X})$$

$$\begin{aligned} a_1^{[1]} &= ReLU(x_0 \times W_{10}^{[1]} + x_1 \times W_{11}^{[1]} + x_2 \times W_{12}^{[1]}) \\ &= ReLU(0.1 + 2 \times (-0.1) + 3 \times 0.3) \\ &= ReLU(0.8) \\ &= 0.8 \end{aligned}$$

$$\begin{aligned}
a_2^{[1]} &= ReLU(x_0 \times W_{20}^{[1]} + x_1 \times W_{21}^{[1]} + x_2 \times W_{22}^{[1]}) \\
&= ReLU(0.1 + 2 \times 0.2 + 3 \times (-0.4)) \\
&= ReLU(-0.7) \\
&= 0
\end{aligned}$$

$$\hat{\mathbf{y}}^{[2]} = ReLU((\mathbf{W}^{[2]})^\top \mathbf{a}^{[1]})$$

$$\begin{aligned}
\hat{y}_1^{[2]} &= ReLU(a_0^{[1]} \times W_{10}^{[2]} + a_1^{[1]} \times W_{11}^{[2]} + a_2^{[1]} \times W_{12}^{[2]}) \\
&= ReLU(0.1 + 0.8 \times 0.5 + 0 \times 0.7) \\
&= ReLU(0.5) \\
&= 0.5
\end{aligned}$$

$$\begin{aligned}
\hat{y}_2^{[2]} &= ReLU(a_1^{[0]} \times W_{20}^{[2]} + a_1^{[1]} \times W_{21}^{[2]} + a_2^{[1]} \times W_{22}^{[2]}) \\
&= ReLU(0.1 + 0.8 \times (-0.6) + 0 \times (-0.8)) \\
&= ReLU(-0.38) \\
&= 0
\end{aligned}$$

$$\begin{aligned}
L(\hat{\mathbf{y}}^{[2]}, \mathbf{y}) &= \frac{1}{2} \times ((\hat{y}_1^{[2]} - y_1)^2 + (\hat{y}_2^{[2]} - y_2)^2) \\
&= \frac{1}{2} \times ((0.5 - 0.1)^2 + (0 - 0.9)^2) \\
&= \frac{1}{2} \times (0.16 + 0.81) \\
&= 0.485
\end{aligned}$$

(b) Suppose we already know that $\frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial y_1^{[2]}} = 0.5, \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial y_2^{[2]}} = 0.3, a_1^{[1]} = 0.5, a_2^{[1]} = 0.4,$ $\hat{y}_1^{[2]} > 0, \hat{y}_2^{[2]} > 0$ Calculate the following gradient (partial derivative): $L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})$ with respect to $W_{21}^{[2]}$ and $L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})$ with respect to $W_{12}^{[2]}$.

$$\begin{aligned}
\frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial W_{21}^{[2]}} &= \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_1^{[2]}} \times \frac{\partial \hat{y}_1^{[2]}}{\partial W_{21}^{[2]}} + \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_2^{[2]}} \times \frac{\partial \hat{y}_2^{[2]}}{\partial W_{21}^{[2]}} \\
&= \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_1^{[2]}} \times 0 + \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_2^{[2]}} \times a_1^{[1]} \\
&= 0.3 \times 0.5 \\
&= 0.15
\end{aligned} \tag{1}$$

$$
\begin{aligned}
\frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial W_{12}^{[2]}} &= \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_1^{[2]}} \times \frac{\partial \hat{y}_1^{[2]}}{\partial W_{12}^{[2]}} + \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_2^{[2]}} \times \frac{\partial \hat{y}_2^{[2]}}{\partial W_{12}^{[2]}} \\
&= \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_1^{[2]}} \times a_2^{[1]} + \frac{\partial L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})}{\partial \hat{y}_2^{[2]}} \times 0 \\
&= 0.5 \times 0.4 \\
&= 0.20
\end{aligned} \tag{2}
$$

2. **Perceptrons**

   (a) Model AND, OR, and NOT logic functions using a perceptron. Assume AND, and OR functions take 2 inputs where while the NOT functions takes a single output. Additionally, is it possible to model XOR function using a single Perceptron? Comment on your answer.
   $\mathbf{w}_1 = (-1.5, 1, 1)^\top$, $\mathbf{w}_2 = (-0.5, 1, 1)^\top$, $\mathbf{w}_3 = (0.5, -1)^\top$. *AND, OR, and NOT functions can be modelled using perceptrons with weights* $\mathbf{w}_1$*,* $\mathbf{w}_2$ *and* $\mathbf{w}_3$ *respectively.*
   No, a single perceptron cannot model XOR because it is not linearly separable.

   (b) Model XOR function(takes 2 inputs) using a number of perceptrons which implement AND, OR, and NOT functions. Show the diagram of the final Perceptron network. Clearly specify the weights of your network.
   $XOR(x_1, x_2) = AND(NOT(AND(x_1, x_2)), OR(x_1, x_2))$. *Figure 1 shows the XOR function.*

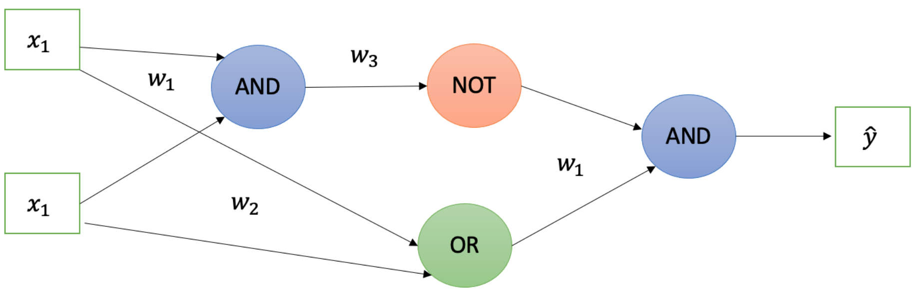


Figure 1: XOR

   (c) Can the following function in Figure 2 be expressed with a 3-layer perceptron?

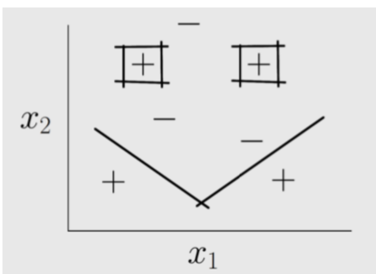


Figure 2: Function for 2(e).

Yes, as each ± 1 *region is clearly defined by the hyperplanes given in the figure.*