Machine Learning

Neural Networks

Machine Learning

Dr. Shahid Mahmood Awan
Assistant Professor
School of Systems and Technology, University of Management and Technology
shahid.awan@umt.edu.pk

Umer Saeed(MS Data Science, BSc Telecommunication Engineering)
Sr. RF Optimization & Planning Engineer
f2017313017@umt.edu.pk

Neural Networks

Non-Linear Hypotheses

Question

- Suppose you are learning to recognize cars from 100×100 pixel images (grayscale, not RGB). Let the features be pixel intensity values. If you train logistic regression including all the quadratic terms (x_i, x_j) as features, about how many features will you have?

- (a) 5,000
- (b) 100,000
- **(c) 5 X 10^7** **Answer**
- (d) 5 X 10^9

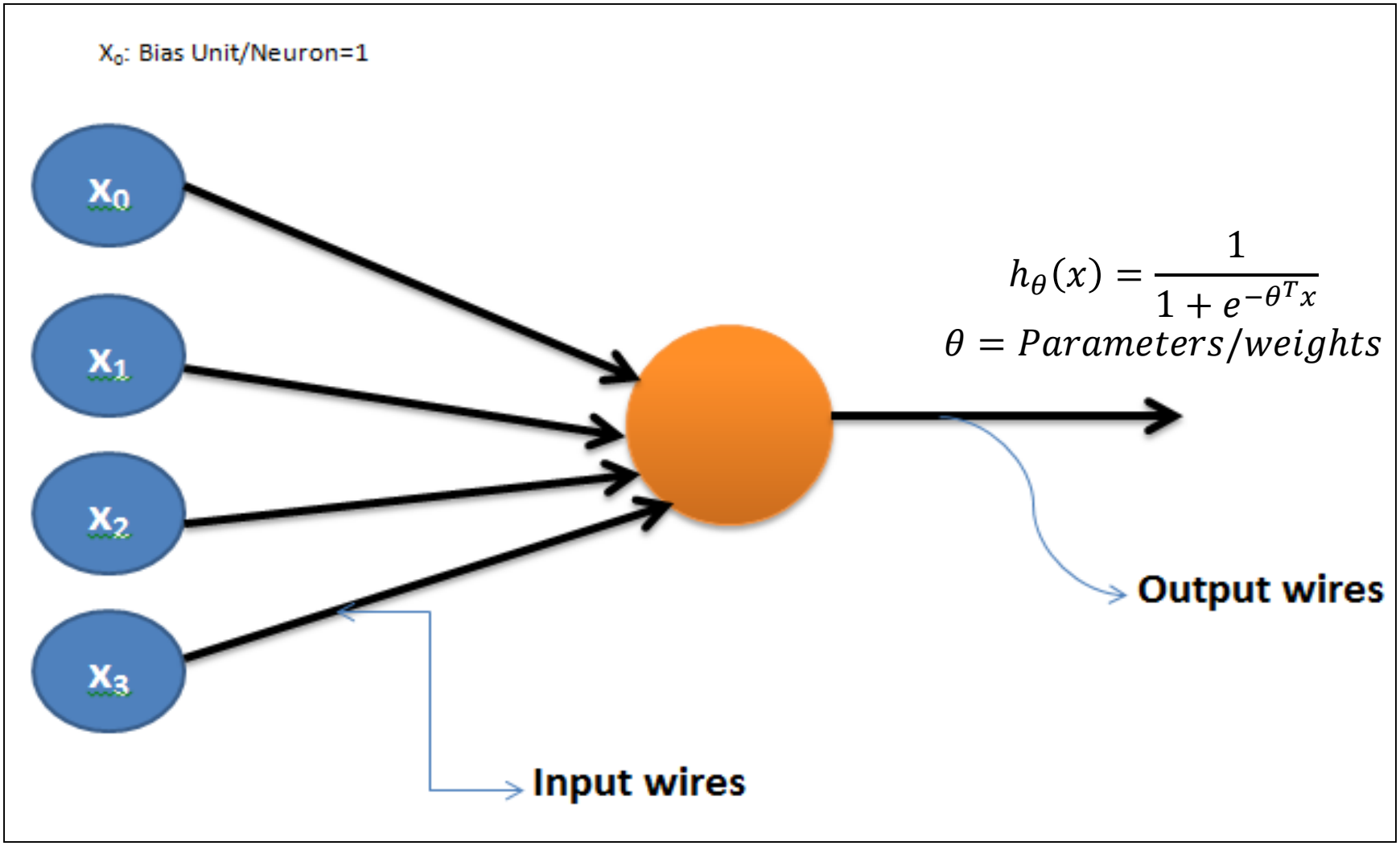
Neural Networks

- **Origins:** Algorithms that try to mimic the brain.
- Was very widely used in 80s and early 90s; popularity diminished in late 90s.
- **Recent resurgence:** state-of-the are technique for many applications.

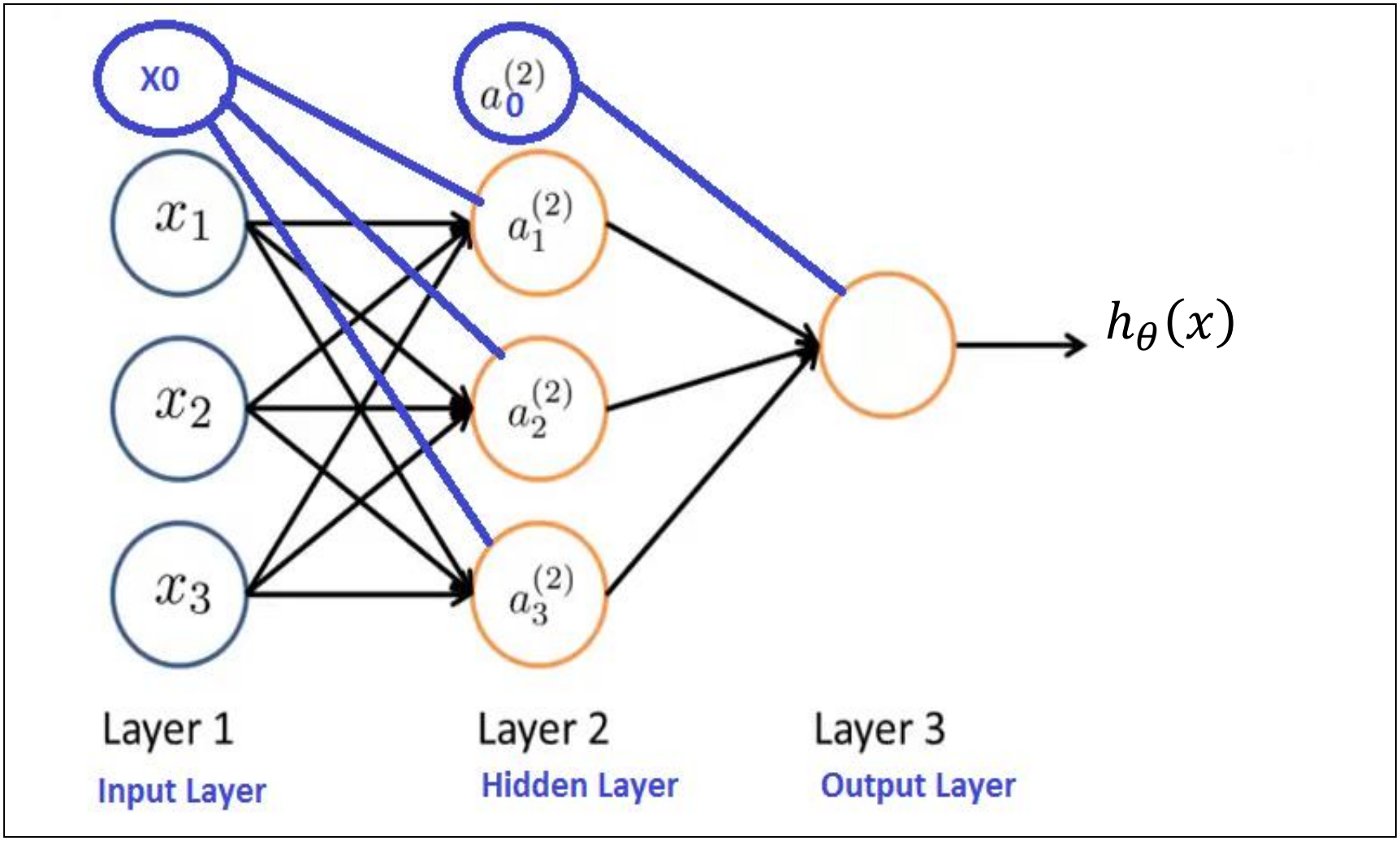
Neural Networks

Model representation

Neural Networks



Neural Networks



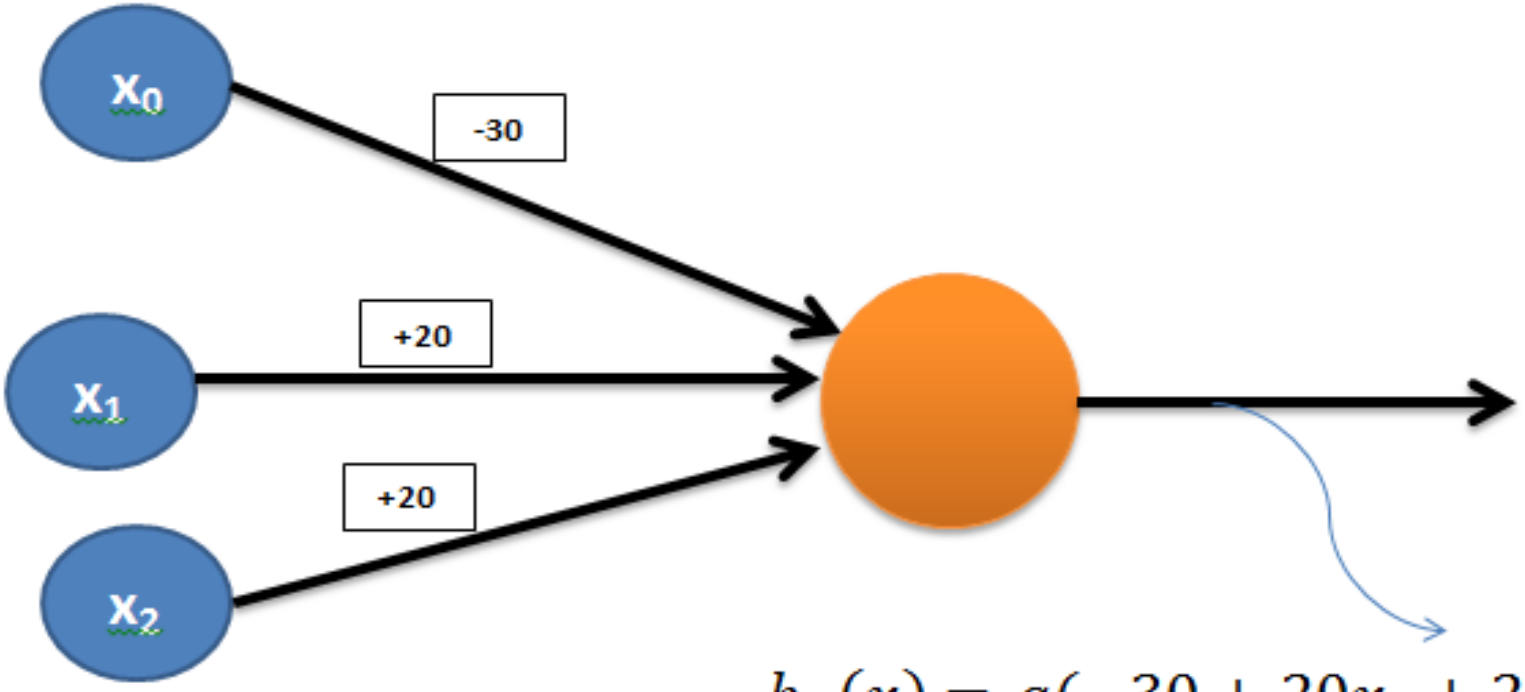
Neural Networks

Applications

Example:: AND Function

$$x_1, x_2 \in \{0,1\}$$

$$y = x_1 AND\ x_2$$



$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$

AND Example :: AND Function

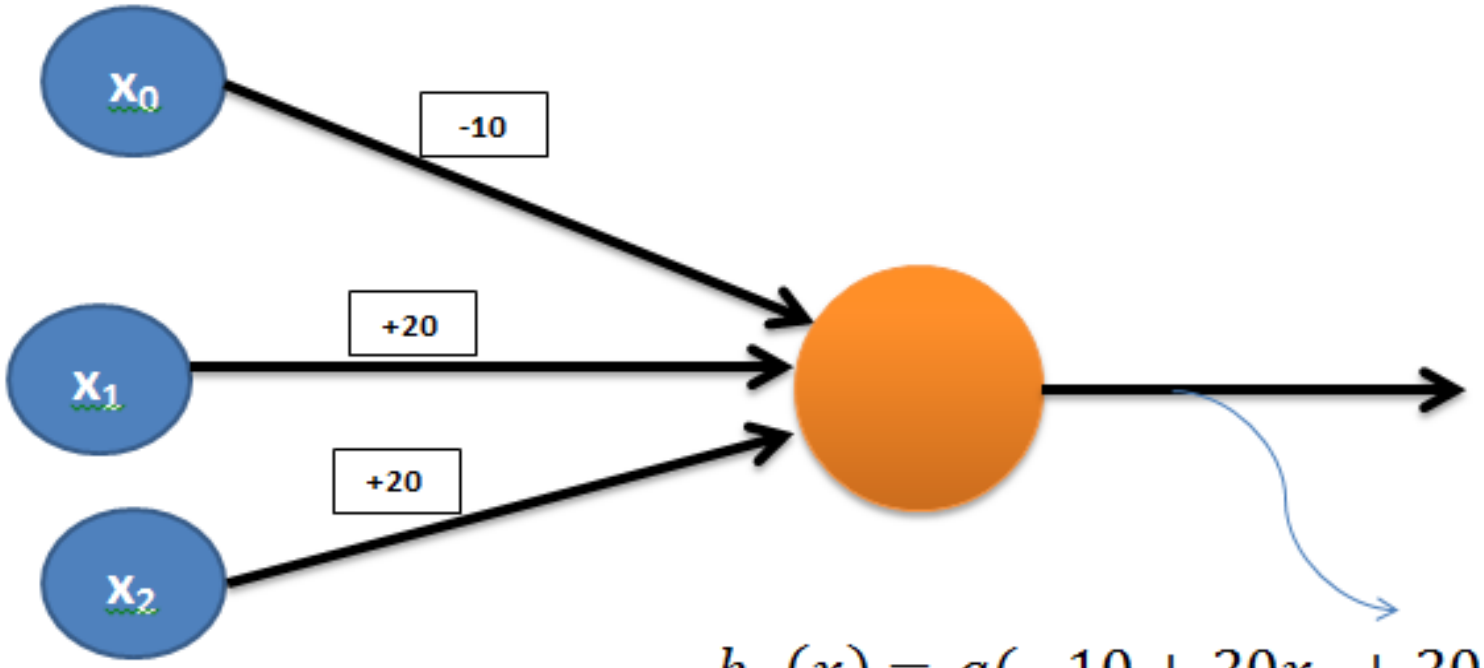
x_1	x_2	$h_\theta(x)$	$g(z)$
0	0	$g(-30)$	≈ 0
0	1	$g(-10)$	≈ 0
1	0	$g(-10)$	≈ 0
1	1	$g(+10)$	≈ 1

$$h_\theta(x) \approx x_1 \; AND \; x_2$$

Example: :: OR Function

$$x_1, x_2 \in \{0,1\}$$

$$y = x_1 OR\ x_2$$



$$h_\theta(x) = g(-10 + 20x_1 + 20x_2)$$

AND Example :: OR Function

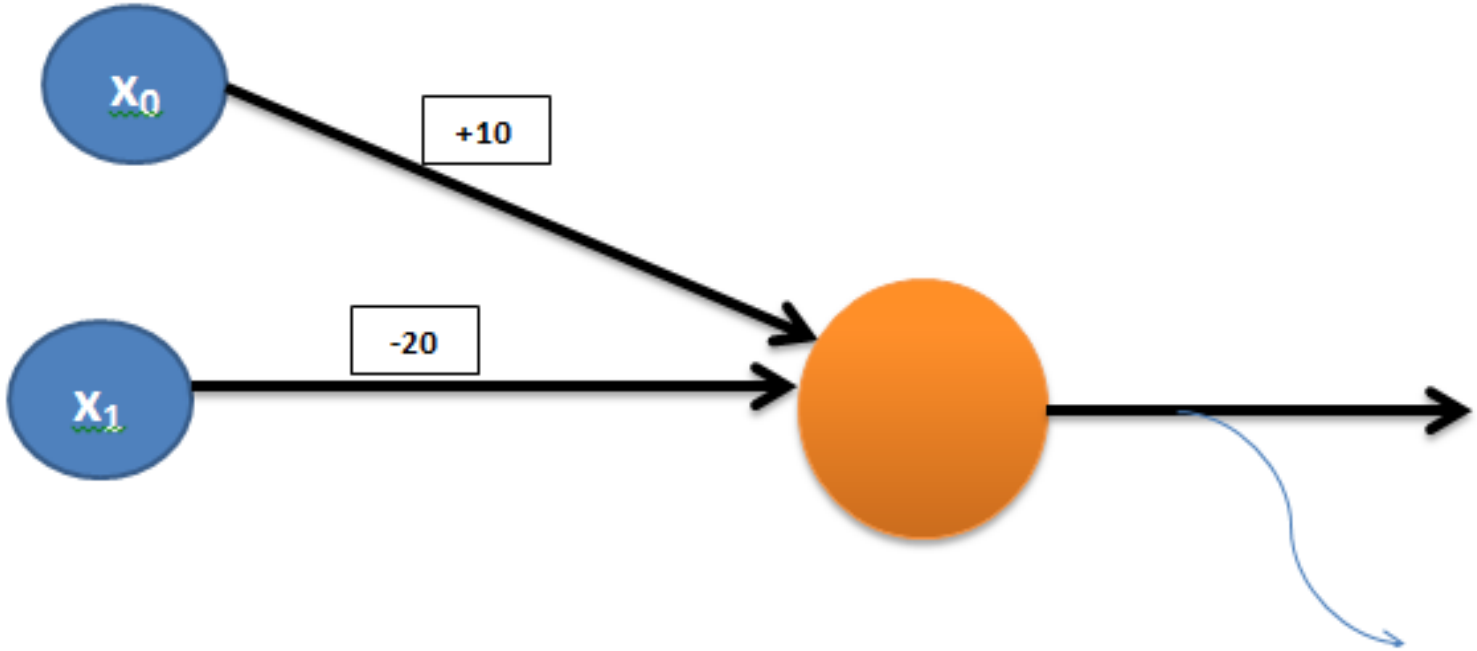
x_1	x_2	$h_\theta(x)$	$g(z)$
0	0	$g(-10)$	≈ 0
0	1	$g(+10)$	≈ 1
1	0	$g(+10)$	≈ 1
1	1	$g(+30)$	≈ 1

$$h_\theta(x) \approx x_1 \; OR \; x_2$$

Example: :: NOT Function

$$x_1, x_2 \in \{0,1\}$$

$$y = NOT\ x_1$$



$$h_\theta(x) = g(+10 - 20x_1)$$

Example :: NOT Function

x_1	$h_\theta(x)$	$g(z)$
0	$g(+10)$	≈ 1
1	$g(-10)$	≈ 0

$$h_\theta(x) \approx NOT\ x_1$$

NOTE

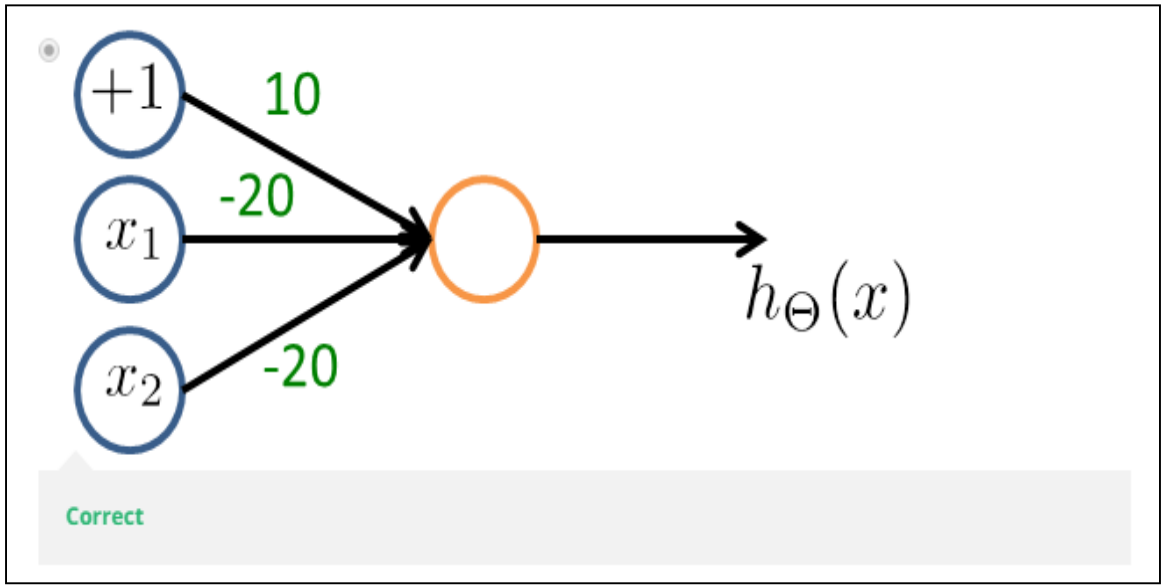
$$(NOT\ x_1) AND\ (NOT\ x_2) = 1$$

$$If\ and\ only\ if\ x_1 = x_2 = 0$$

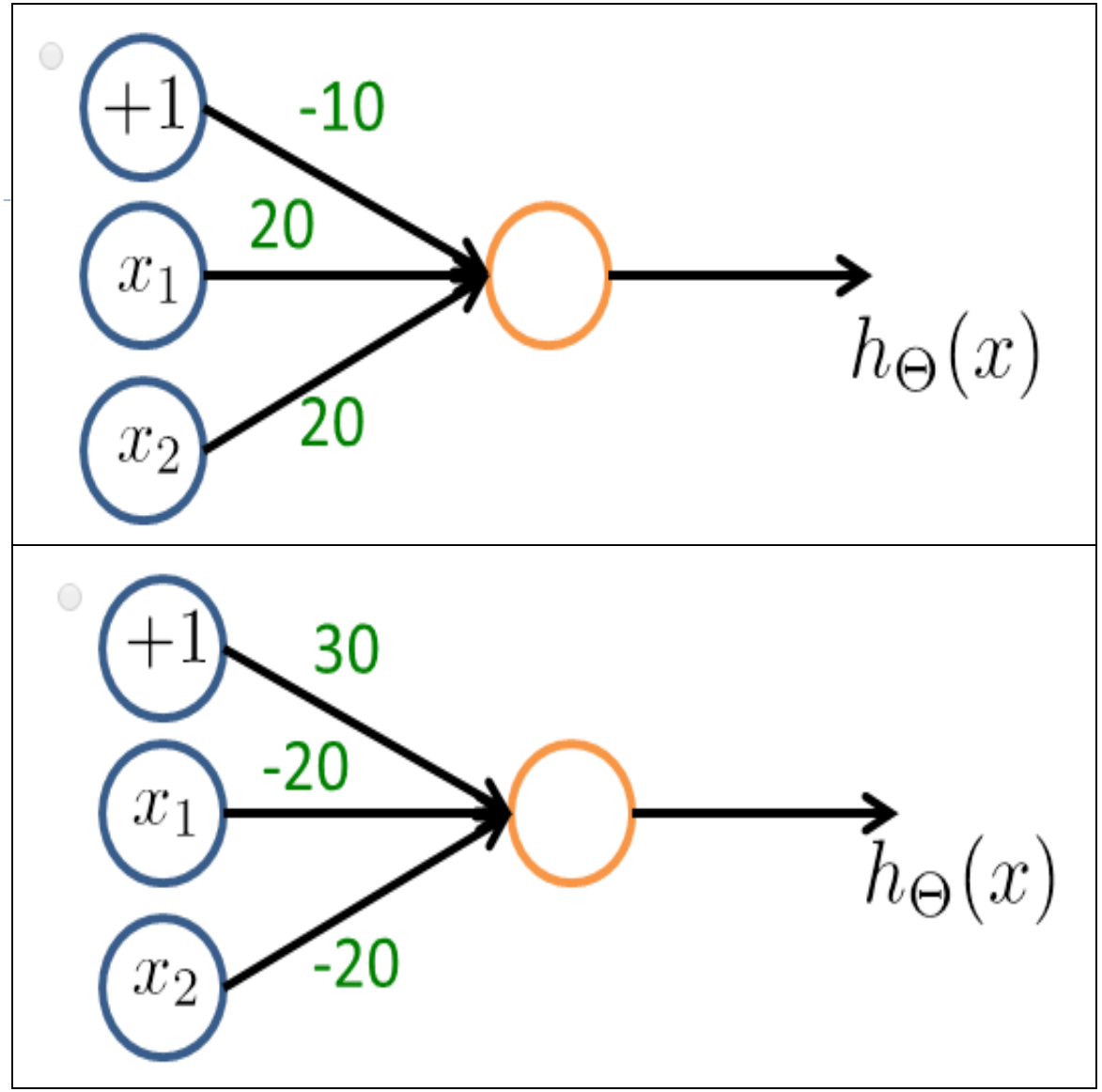
x_1	x_1	$NOT\ x_1$	$NOT\ x_1$	$(NOT\ x_1) AND (NOT x_2)$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Question

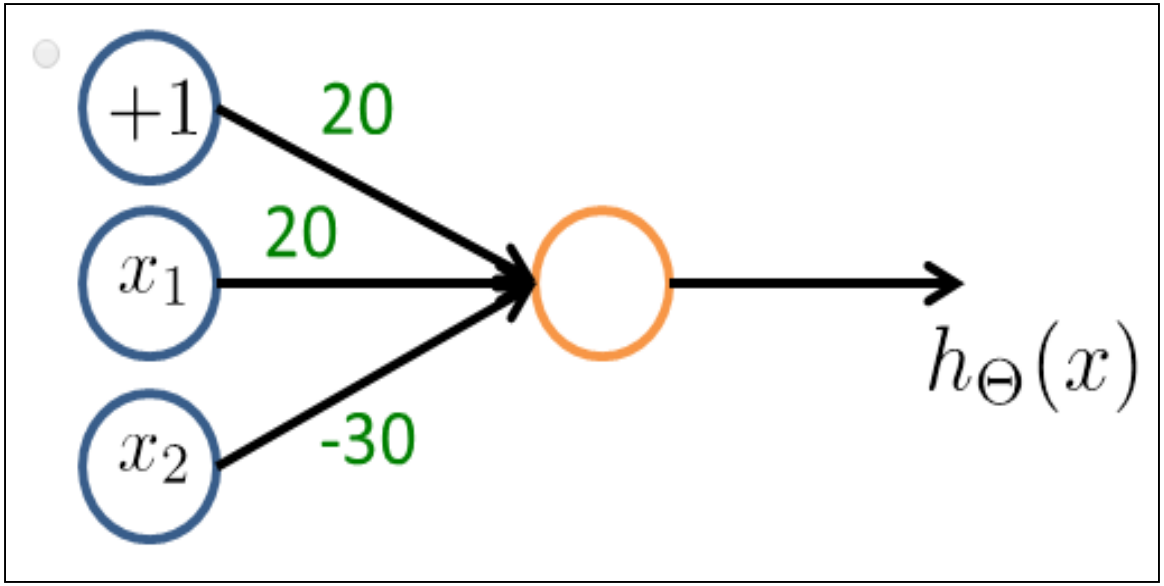
- Suppose that $x1$ and $x2$ are binary valued (0 or 1). Which of the following networks (approximately) computes the boolean function (NOT $x1$) AND (NOT $x2$)?



Question



Question



Solution

From Fig (a) $h_\theta(x) = g(10 - 20x_1 - 20x_2)$

From Fig (b) $h_\theta(x) = g(-10 + 20x_1 + 20x_2)$

From Fig (c) $h_\theta(x) = g(30 - 20x_1 - 20x_2)$

From Fig (d) $h_\theta(x) = g(20 + 20x_1 - 30x_2)$

- **Calculation For Fig (a):**

x_1	x_2	$h_\theta(x)$	$g(z)$
0	0	$g(+10)$	≈ 1
0	1	$g(-10)$	≈ 0
1	0	$g(-10)$	≈ 0
1	1	$g(-30)$	≈ 0

Network (from fig a) (approximately) computes the boolean function $(NOT\ x_1) AND\ (NOT\ x_2)$

Solution

- **Calculation For Fig (b):**

x_1	x_2	$h_\theta(x)$	$g(z)$
0	0	$g(-10)$	≈ 0
0	1	$g(+10)$	≈ 1
1	0	$g(+10)$	≈ 1
1	1	$g(+30)$	≈ 1

- **Calculation For Fig (c):**

x_1	x_2	$h_\theta(x)$	$g(z)$
0	0	$g(+30)$	≈ 1
0	1	$g(+10)$	≈ 1
1	0	$g(+10)$	≈ 1
1	1	$g(-10)$	≈ 0

Solution

- **Calculation For Fig (d):**

x_1	x_2	$h_\theta(x)$	$g(z)$
0	0	$g(+20)$	≈ 1
0	1	$g(+40)$	≈ 1
1	0	$g(-10)$	≈ 0
1	1	$g(+10)$	≈ 1

Neural Networks

Perceptrons

Perceptrons

- The perceptrons which was first proposed by Rosenbaltt (1958), is a simple neuron that is used to classify its inputs into one of two categories.
- The perceptron can have any number of input, which are sometimes arranged into a grid.
- This grid can be used to represent an image, or a field of vision and so perceptrons can be used to carry out simple image classification or recognition taks.
- A perceptron use a step function that returns +1 if the weighted sum of the inputs, X, is greater that a threshold, *t*, and -1 if X is less than or equal to *t*:

Perceptrons

$$X = \sum_{i=1}^{n} w_i \, x_i$$

$$Y = \begin{cases} +1 & for\ X > t \\ 0 & for\ X \leq t \end{cases}$$

This function is often written as Step(X);

$$Step(X) = \begin{cases} +1 & for\ X > t \\ 0 & for\ X \leq t \end{cases}$$

In which case, the activation function for a perceptron can be written as;

$$Y = Step\left[\sum_{i=1}^{n} w_i \, x_i\right]$$

Perceptrons

- Note that here we have allowed *i* to run from 0 instead of from 1. This means that we have introduced two new variables: *w*0 and *x*0.We define *x*0 as 1, and *w*0 as *t*.

- A single perceptron can be used to learn a classification task, where it receives an input and classifies it into one of two categories: 1 or 0.We can consider these to represent *true* and *false*, in which case the perceptron can learn to represent a Boolean operator, such as AND or OR.

- **The learning process for a perceptron is as follows:**

- First, random weights are assigned to the inputs. Typically, these weights will be chosen between 0.5 and +0.5.

Perceptrons

- Next, an item of training data is presented to the perceptron, and its output classification observed. If the output is incorrect, the weights are adjusted to try to more closely classify this input. In other words, if the perceptron incorrectly classifies a positive piece of training data as negative, then the weights need to be modified to increase the output for that set of inputs. This can be done by adding a positive value to the weight of an input that had a negative input value, and vice versa.

- The formula for this modification, as proposed by Rosenblatt (Rosenblatt 1960) is as follows:

$$wi \leftarrow wi + (a * x_i * e)$$

- where e is the error that was produced, and a is the **learning rate**,where $0 < a < 1$; e is defined as 0 if the output is correct, and otherwise it is positive if the output is too low and negative if the output is too high. In this way, if the output is too high, a decrease in weight is caused for an input that received a positive value. This rule is known as the **perceptron training rule**.

Perceptrons

- Once this modification to the weights has taken place, the next piece of training data is used in the same way. Once all the training data have been applied, the process starts again, until all the weights are correct and all errors are zero. Each iteration of this process is known as an **epoch**.

Example

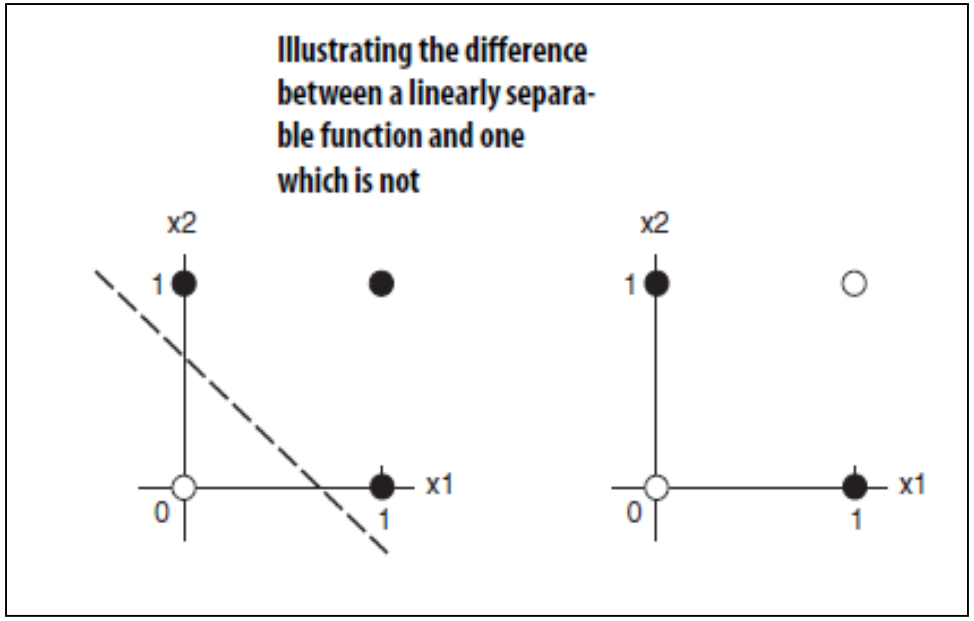
- t=0 Learning Rate=0.2

Epoch	X1	X2	w1	w2	x=x1*w1+x2*w2	step(x)	Expected value	error	Comments	Updated w1=w1+(a*x1*e)	Updated w2=w2+(a*x2*e)
1	0	0	-0.2	0.4	0	0	0	0	error=0, no need to change w		
1	0	1	-0.2	0.4	0.4	1	1	0	error=0, no need to change w		
1	1	0	-0.2	0.4	-0.2	0	1	1	error=1, no need to change w	0	0.4
1	1	1	0	0.4	0.4	1	1	0	error=0, no need to change w		
2	0	0	0	0.4	0	0	0	0	error=0, no need to change w		
2	0	1	0	0.4	0.4	1	1	0	error=0, no need to change w		
2	1	0	0	0.4	0	0	1	1	error=1, no need to change w	0.2	0.4
2	1	1	0.2	0.4	0.6	1	1	0	error=0, no need to change w		
3	0	0	0.2	0.4	0	0	0	0	error=0, no need to change w		
3	0	1	0.2	0.4	0.4	1	1	0	error=0, no need to change w		
3	1	0	0.2	0.4	0.2	1	1	0	error=0, no need to change w		
3	1	1	0.2	0.4	0.6	1	1	0			

- After just three epochs, the perceptron learns to correctly model the logical- OR function.
- In the same way, a perceptron can be trained to model other logical functions such as AND, but there are some functions that cannot be modeled using a perceptron, such as exclusive OR.

Example

- The reason for this is that perceptrons can only learn to model functions that are **linearly separable**. A linearly separable function is one that can be drawn in a two-dimensional graph, and a single straight line can be drawn between the values so that inputs that are classified into one classification are on one side of the line, and inputs that are classified into the other are on the other side of the line.



Reference

- https://www.coursera.org/