Latakia University
Information Engineering Faculty
Artificial Intelligence Department
Fourth Year

Artificial Neural Networks

Lecture 5 : Neural Networks-Representation &Regularization

Dr. Wisam Ibrahim

2025-2026 Semester 1

WHAT IS NEURAL NETWORK ?

Neural networks were developed as a way to simulate networks of neurons.

What does a neuron look like?

Three things to notice:

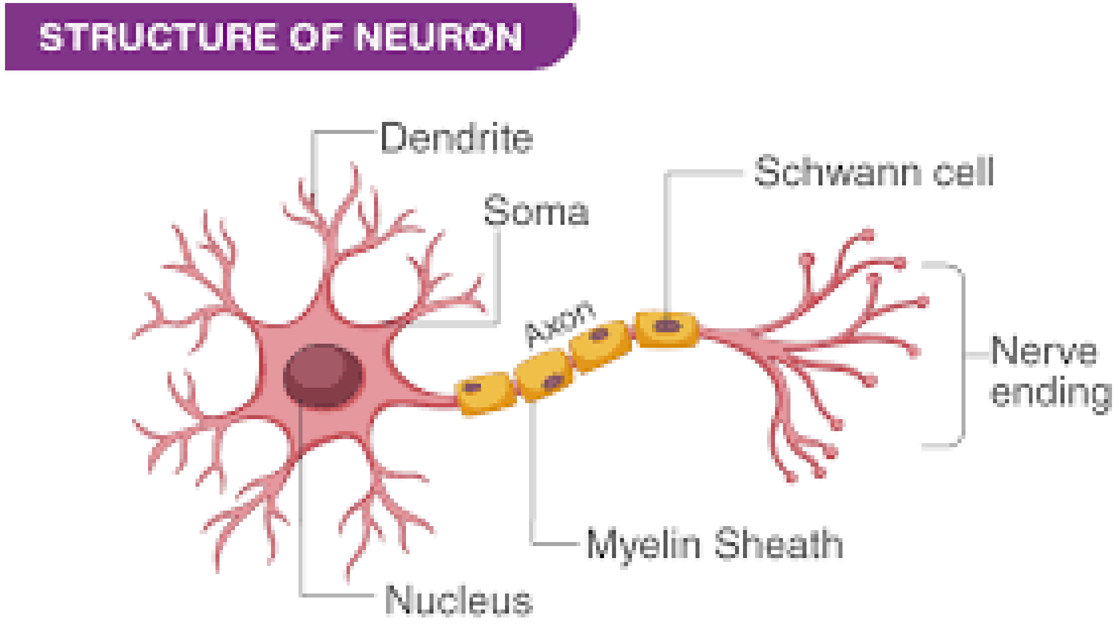
- Cell body
- Number of input wires (dendrites)
- Output wire (axon)

How does it work?

- Neuron gets one or more inputs through dendrites.
- Does processing.
- Sends output down axon.

Neurons communicate through electric spikes

- Pulse of electricity via axon to another neuron



NEURAL NETWORK - REPRESENTATION

Artificial neural network - representation of a neuron:

In an artificial neural network, a neuron is a logistic unit

- Feed input via input wires *x1*, *x2* ,*x3*, ...
- Logistic unit does computation (hypothesis function h(x)).
- Sends output down output wires

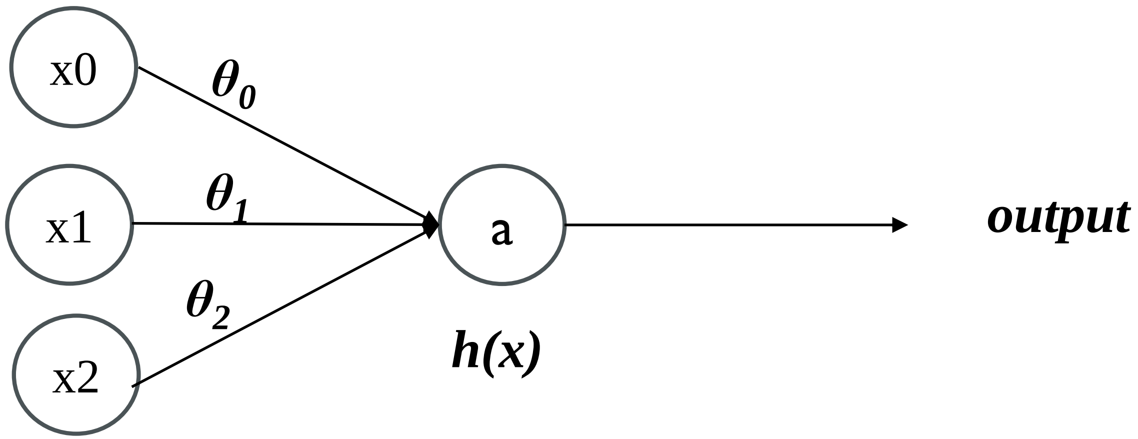
This logistic computation is just like our previous logistic regression hypothesis calculation

NEURAL NETWORK - REPRESENTATION

θ or w is called weight, first layer is the input layer. There is one or more hidden layers containing neurons called ***activation units*** a^j_i (i layer no. j unit no.)

Neuron applies the hypothesis function on input values.

the following graph represents a simple neural network:



(x_0, x_1, x_2) is input layer. $(\theta_0, \theta_1, \theta_2)$ weights of first layer, where θ_0 is the bias value b.

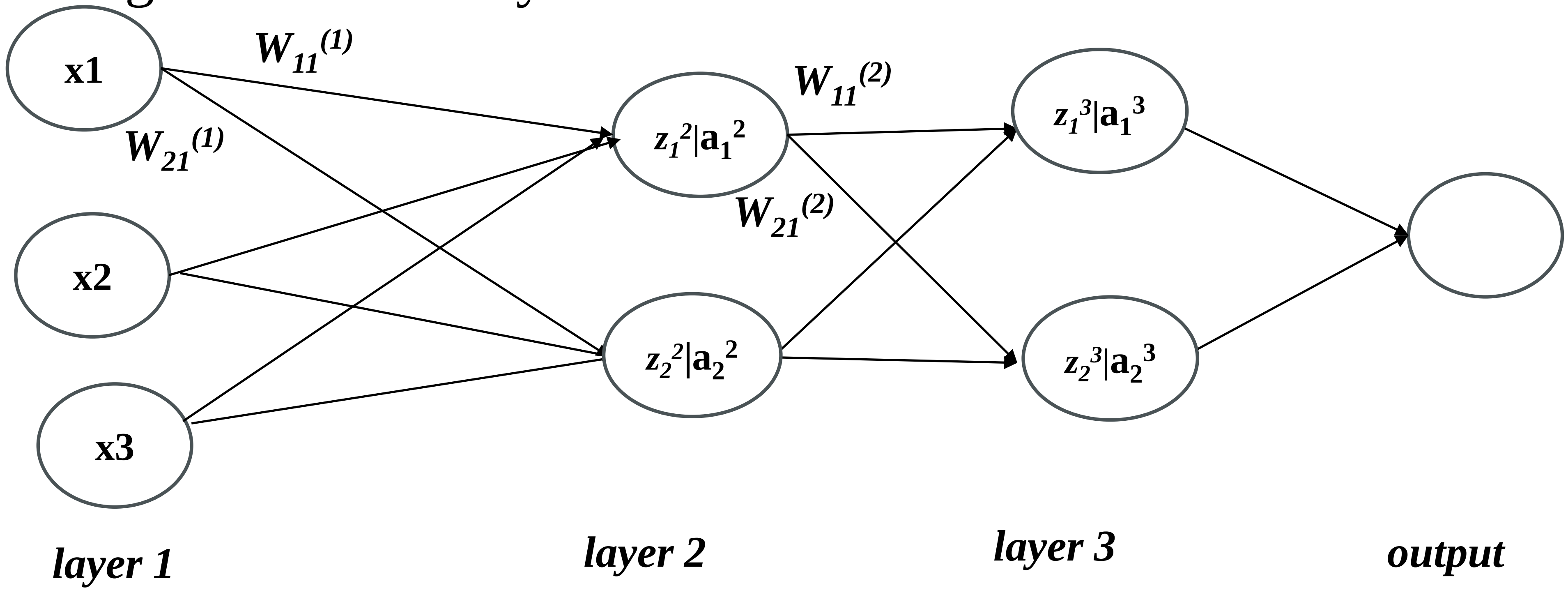
h(x)= $g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2)$: g is sigmoid function

NEURAL NETWORK - REPRESENTATION

The following graph is a NN with three layers two hidden layers and one input layer.

$W_{11}^{(1)}$: is weight in the first layer which is sent from unit 1 to unit 1 in second layer.

$W_{21}^{(1)}$: is weight in the first layer which is sent from unit 1 to unit 2 in second layer.



$$a_1^2 = g(z_1^2) = g(\, x_0\, w_{10}^1 + x_1\, w_{11}^1 + x_2\, w_{12}^1 + x_3\, w_{13}^1 \,)$$

$$a_2^2 = g(z_2^2), \quad a_1^3 = g(z_1^3)\,, \qquad a_2^3 = g(z_2^3)$$

NEURAL NETWORK - REPRESENTATION

The weight matrix:

If the network has $\boldsymbol{S_j}$ units in the layer j and has $\boldsymbol{S_{j+1}}$ units in layer $j+1$, the dimensions of weight matrix $\boldsymbol{W^{(j)}} : \boldsymbol{S_{j+1} \times S_j}$

For example for the previous neural networks: $\boldsymbol{W^{(1)}}_{\boldsymbol{2\times 3}}$, $\boldsymbol{W^{(2)}}_{\boldsymbol{2\times 2}}$, $\boldsymbol{W^{(3)}}_{\boldsymbol{1\times 2}}$

$$W^{(3)}_{1\times 2} = [W_{11}^{(3)}, W_{12}^{(3)}]$$

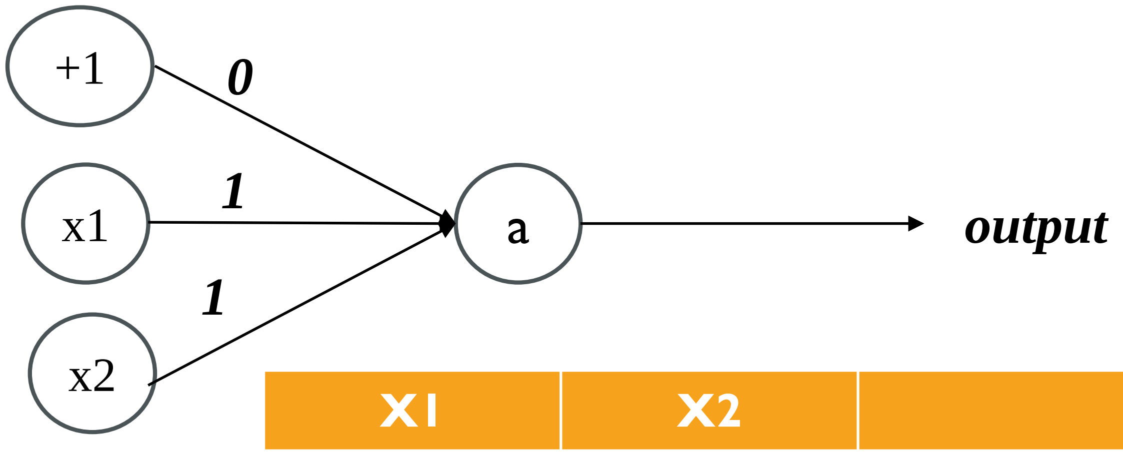
$$X = \begin{bmatrix} x0 \\ x1 \\ x2 \\ x3 \end{bmatrix} \qquad Z^2 = \begin{bmatrix} z_1{}^2 \\ z_2{}^2 \end{bmatrix}$$

$$Z^2 = \theta^1 X \;\; => \;\; a^2 = g(z^2),$$

NEURAL NETWORK - **AND** NETWORK

Example 1:

we have the following graph which represents the AND network . *x1*, *x2* belong to {0,1}. We search about the weights that make the activation function is suitable to give the correct outputs.



Activation function is sigmoid function:

$h(x)= g(w_0 + w_1 x_1 + w_2 x_2)$

If the weights: $w_0=0$, $w_1=1$, $w_2 =1$

Output of the function will be as following table:

X1	X2	
0	0	g(0)=0.5
0	1	g(1)≈1
1	0	g(1) ≈1
1	1	g(1) ≈1

NEURAL NETWORK - AND NETWORK

We update the weights to be as following:

$w_0= -30$, $w_1=20$, $w_2 =20$

The output of the neural network will be correct as following:

X1	X2	
0	0	g(-30) ≈0
0	1	g(-10)≈0
1	0	g(-10) ≈0
1	1	g(+10) ≈1

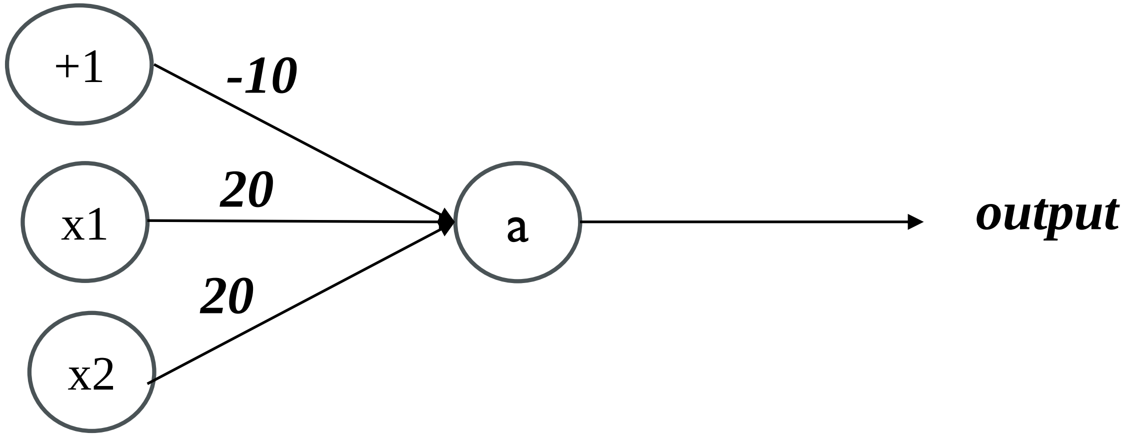
NEURAL NETWORK - **OR** NETWORK

We have the following graph which represents the OR network . *x1*, *x2* belong to {0,1}. we search about the weights that make the activation function is suitable to give the correct outputs.

Activation function is sigmoid function:

$h(x)= g(w_0 + w_1 x_1 + w_2 x_2)$

If the weights: $w_0 = -10$, $w_1 = 20$, $w_2 = 20$



Outputs of the function will be as following table:

X1	X2	
0	0	g(-10) ≈0
0	1	g(10) ≈1
1	0	g(10) ≈1
1	1	g(30) ≈1

NEURAL NETWORK - **NOT** NETWORK

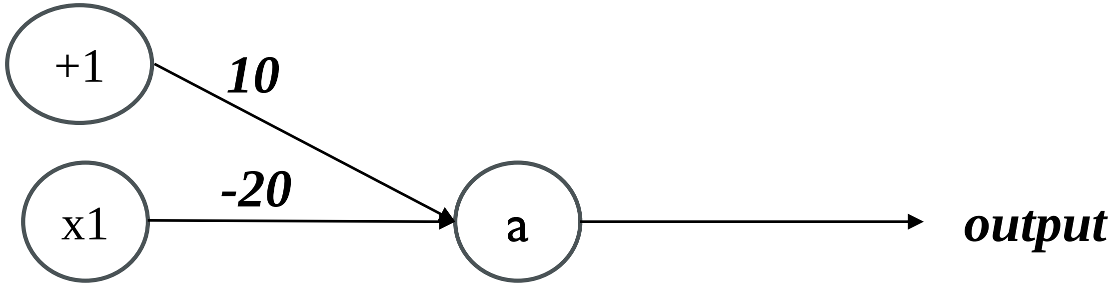
We have the following graph which represents the NOT network .

x1 belong to {0,1}. we search the weights that make the activation function is suitable to give the correct outputs.

Activation function is sigmoid function:

$h(x)= g(w_0 + w_1 x_1)$

If the weights: $w_0=10$, $w_1= -20$



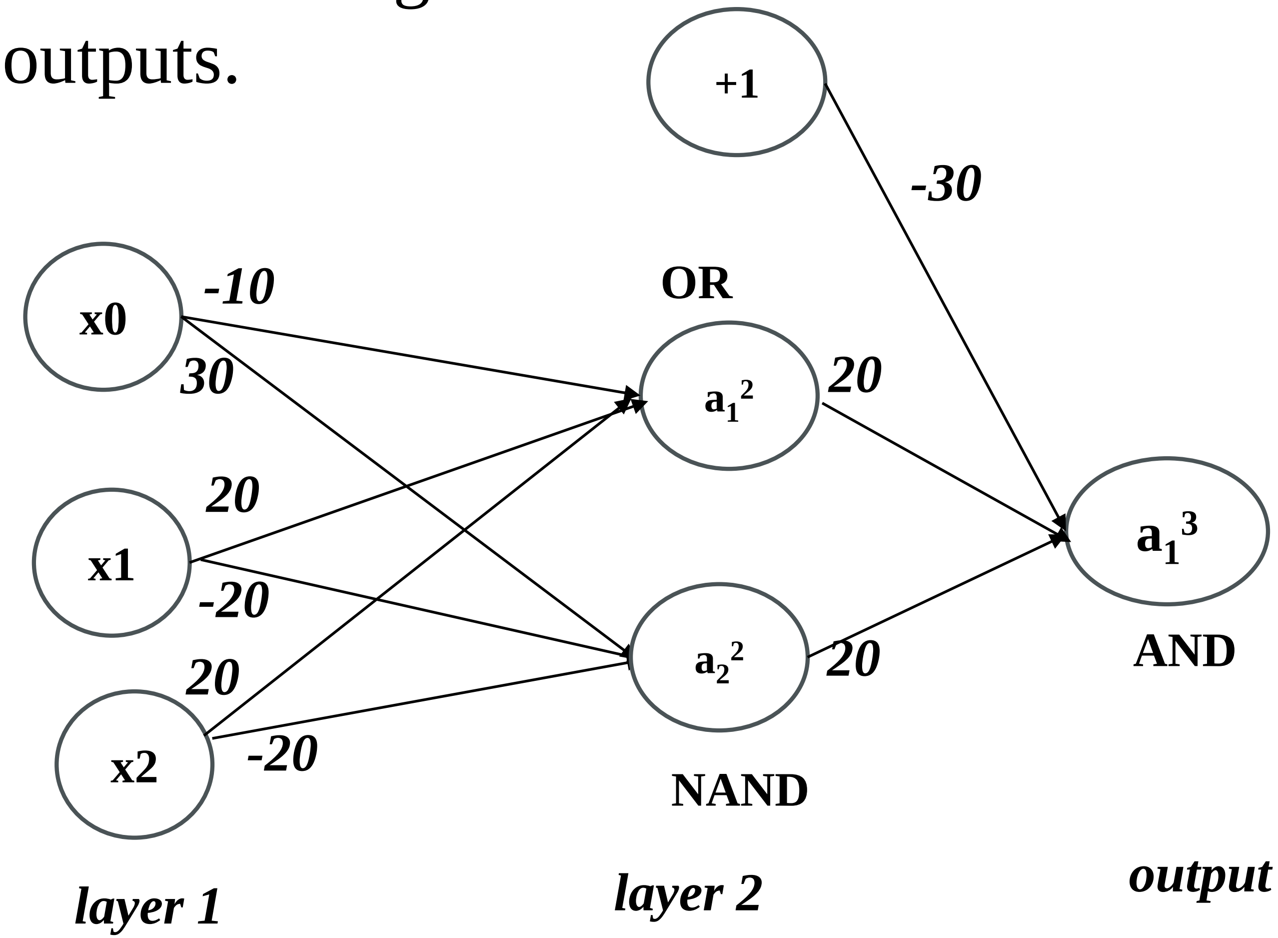
NEURAL NETWORK - **XOR** NETWORK

We have the following graph which represents the XOR network .

x1, *x2* belong to {0,1}. We search about the weights that make the activation function is suitable to give the correct outputs.

XOR(x1, x2) = (x1∨ x2) ∧ ¬ (x1 ∧ x2)

where: NAND (x1, x2) = ¬ (x1 ∧ x2)



NEURAL NETWORK - XOR NETWORK

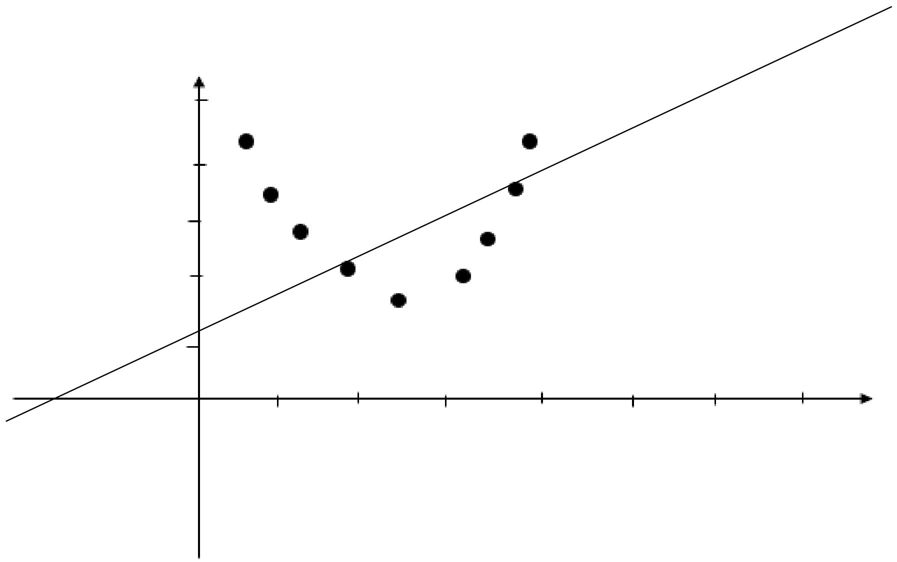
The output of the network is:

X1	X2	a_1^2	a_2^2	a_1^3
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

REGULARIZATION

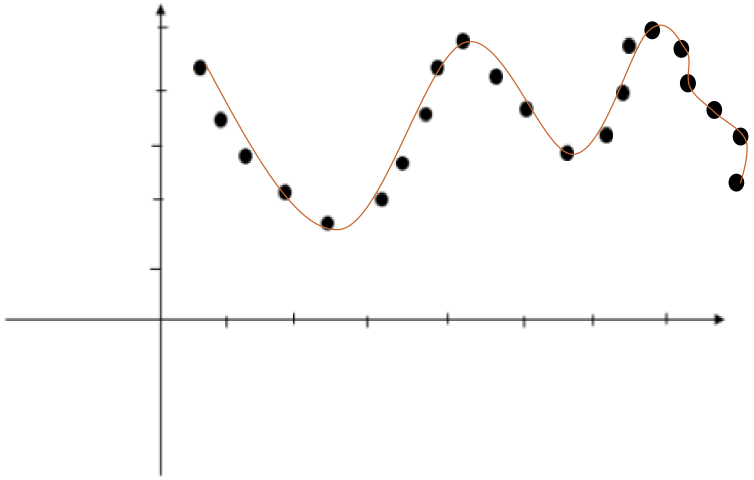
- Under fitting: means that the function (model) accuracy in the training phase (training data) is so low. Thus, the accuracy in the test phase (new data) will be low too. also called **High bias**
- as follows:

we have used a line function : h(x)= $\boldsymbol{\theta_0 x_0 + \theta_1 x_1}$



REGULARIZATION

- Over Fitting: means that the function's (model) accuracy in the training phase (training data) is very High. However, the accuracy in the test phase (new data) will be low. also called **High variance**
- We have used a high order polynomial function :
- h(x)= $\boldsymbol{\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_1 + \theta_3 x_3 + \theta_4 x_4}$

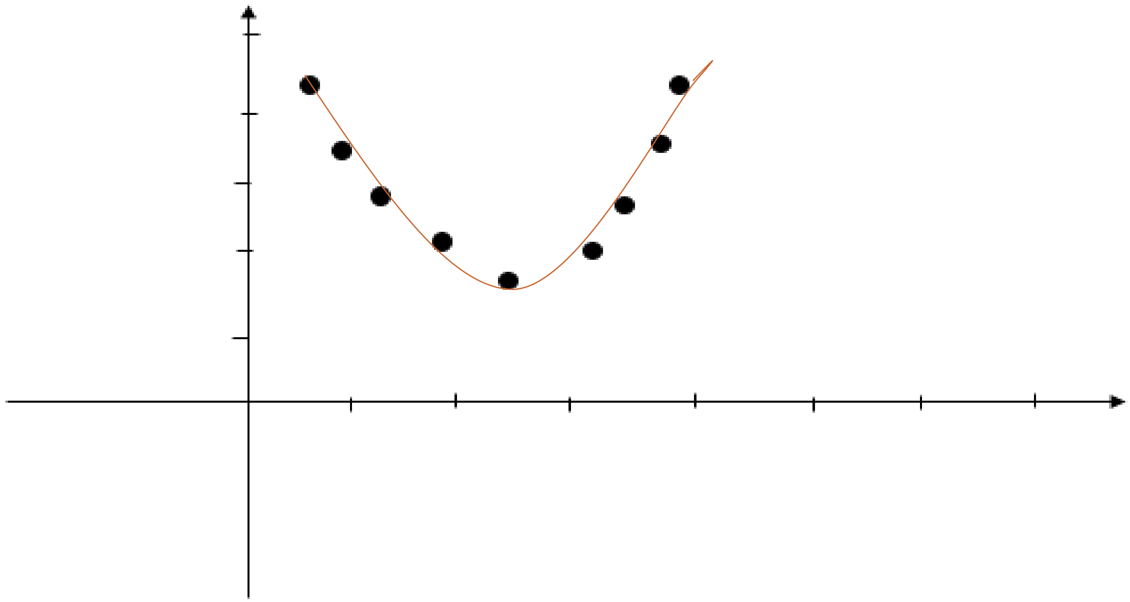


REGULARIZATION

Best fit : is the situation that can make balance between the OF and UF for example .

In the following graph we have used a quadratic function (neither line or complicated polynomial)

$$h(x)= \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_1^2$$



REGULARIZATION

We notice that to solve UF problem we have to increase the complexity of the function by adding more features and using more samples for obtaining more accuracy.

In addition, to solve OF we have to decrease the dimensionality by removing some features (unimportant features). But we have to be careful with removing features such as *Area* or *Bedroom* in house pricing example.

REGULARIZATION

Regularization

- Keep all features, but reduce magnitude of parameters θ
- Works well when we have a lot of features, each of which contributes a bit to predicting y.

To obtain the best fit, a new parameter is used called λ as follows:

- In the cost function:

$$J(\theta) = \frac{1}{2m} \left(\sum_i (h_\theta (x^i) - y^i)^2 + \lambda \sum_j \theta^2_j\right) \; : i=1,2,\ldots, m \quad j=1,2, \ldots, n$$

- In the gradient Descent:

$$\theta_0 = \theta_0 - \alpha \left(\frac{1}{m} \sum_i (h_\theta (x^i) - y^i)\right)$$

$$\theta_j = \theta_j - \alpha \left(\frac{1}{m} \sum_i (h_\theta (x^i) - y^i) x^i \right) + \frac{\lambda}{m} \theta_j)$$

$$\theta_j = \theta_j \left(1- \alpha \frac{\lambda}{m}\right) - \alpha \left(\frac{1}{m} \sum_i (h_\theta (x^i) - y^i) x^i \right)$$

REGULARIZATION

- The term $(1- \alpha \frac{\lambda}{m})$ is going to be a number less than 1 usually.
- Usually learning rate is small and *m* is large.
- So this typically evaluates to *(1 - a small number)*
- So the term is often around 0.99 to 0.95
- This in effect means θ_j gets multiplied by 0.99.
- Means θ_j a little smaller
- The second term is exactly the same as the original gradient descent.

REGULARIZATION

- If λ is very large we end up penalizing all the parameters *(θ_3 , θ_4 etc.)*, so all the parameters end up being close to zero.
- If this happens, it's like we got rid of all the terms in the hypothesis. This result here is then under fitting.
- So, λ should be chosen carefully - *not too big*.
- In order to find the best λ, we can start with $\lambda = 0$ and measure the cross-validation error at each iteration, increasing the λ with a fixed value .