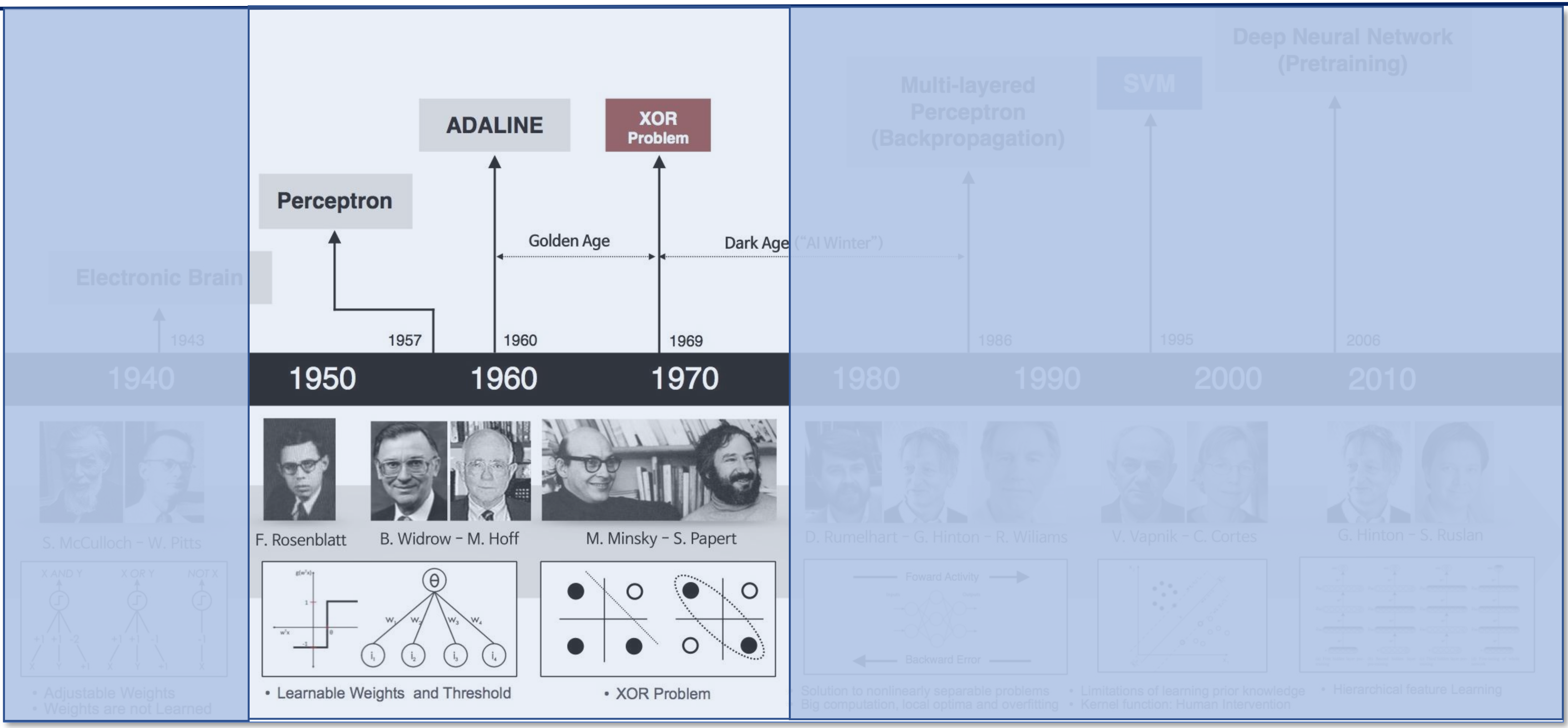
CS 103 -09
Perceptron Learning and ADALINE

Jimmy Liu 刘江

2020-11-12

Group Project Update

AI algorithm Developments - A Closer Look



Perceptron

3 4

1 Perceptron

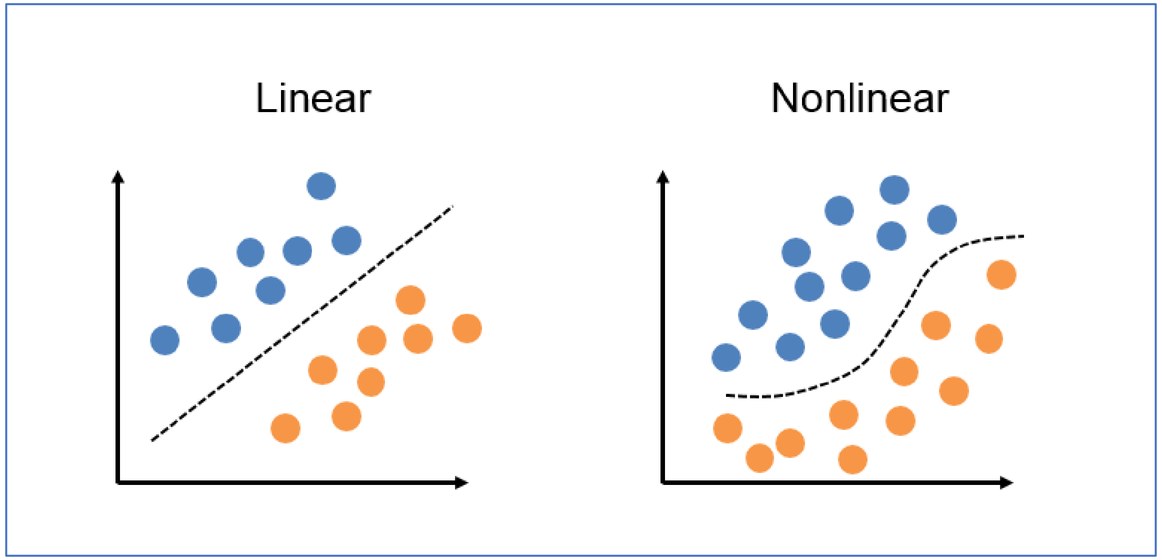
2 Perceptron Learning

3 ADALINE

4 Limitation of Perceptron

Q1: What Does “A Function is Linearly Separable ”Mean?

Linearly Separable



A function is said to be linearly separable when its outputs can be discriminated by a function which is a linear combination of features, that is we can discriminate its outputs by a line or a hyperplane.

Traditional Perceptron Decision Surface

A threshold perceptron returns 1 iff the weighted sum of its inputs (including the bias) is positive, i.e.,:

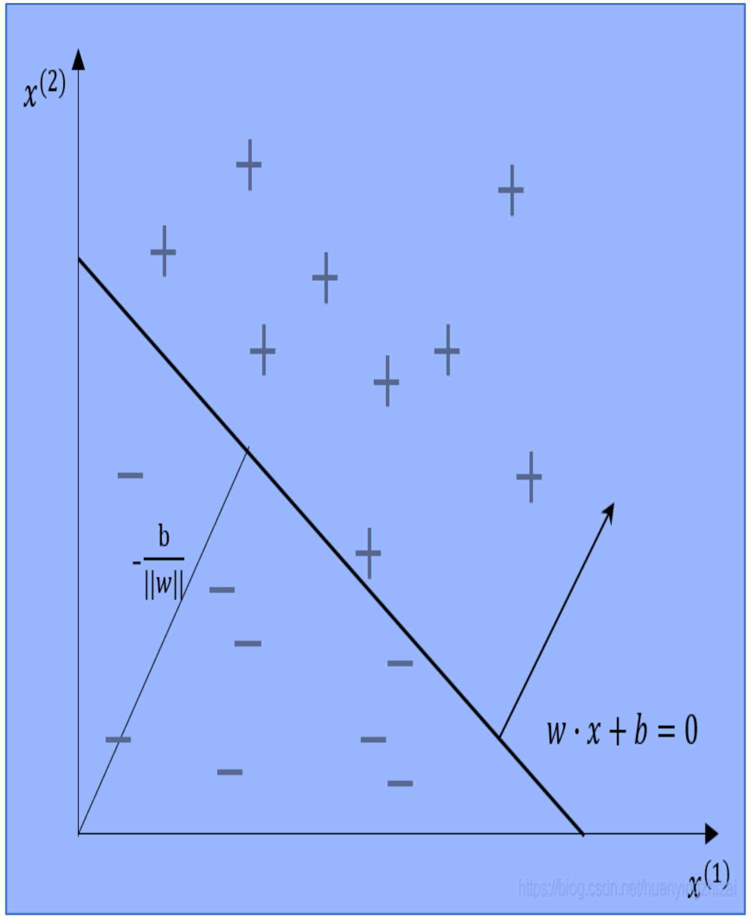
$$\sum_{j=0}^{n} W_j x_j > 0 \quad \text{or} \quad \mathbf{W} \cdot \mathbf{x} > 0$$

I.e., iff the input is on one side of the hyperplane it defines.

Perceptron → Linear Separator

Linear discriminant function or linear decision surface.

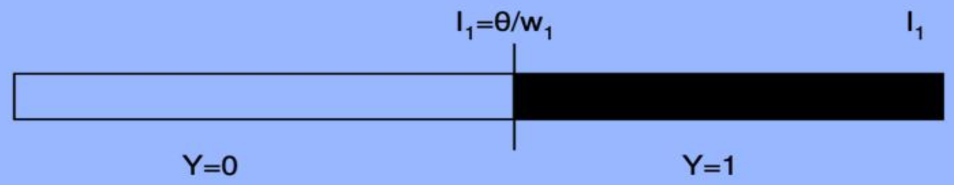
Weights determine slope and bias determines offset.



Decision Surface

Decision surface is the surface at which the output of the unit is precisely equal to the threshold, i.e. $\sum_{i=1....n} w_i I_i = \theta$

In **1-D** the surface is just a point:



In **2-D**, the surface is

$$I_1 \cdot w_1 + I_2 \cdot w_2 - \theta = 0$$

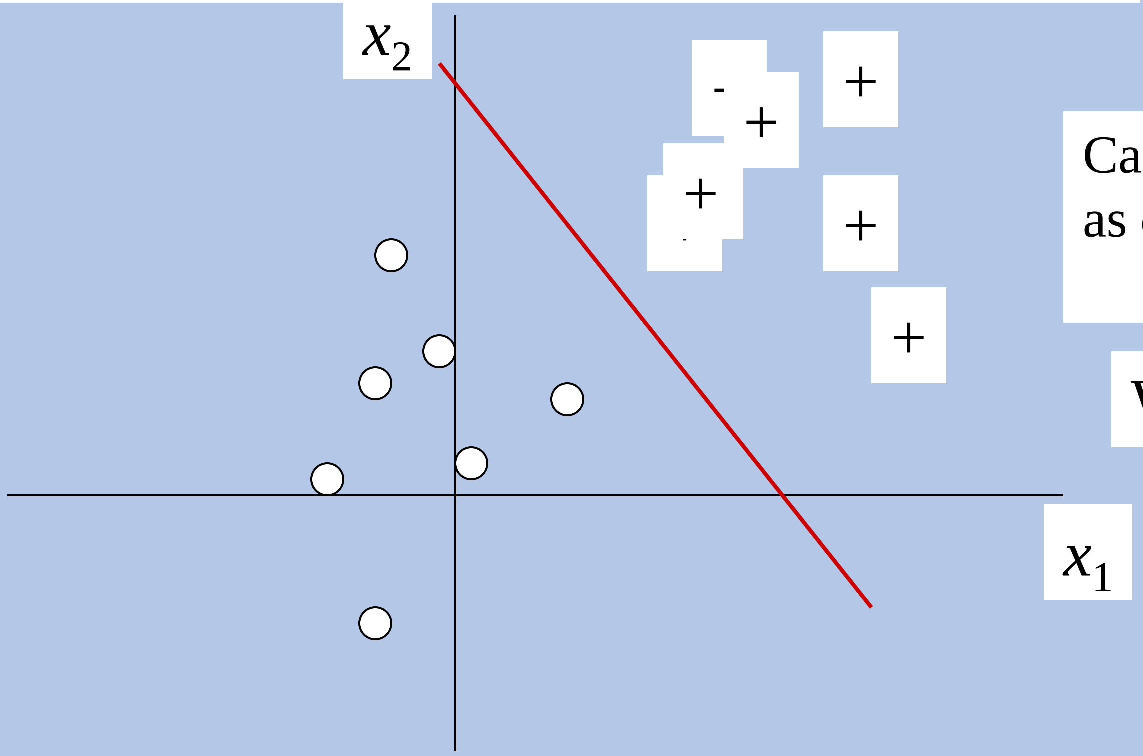
which we can re-write as

$$I_2 = \frac{\theta}{w_2} - \frac{w_1}{w_2} I_1$$

So, in 2-D the decision boundaries are **<u>always</u>** straight lines.

Exercise: Separation Line

Consider example with two inputs, x1, x2:



Can view trained network as defining a "separation line".

What is its equation?

$$-w_0 + w_1 x_1 + w_2 x_2 = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{w_0}{w_2}$$

Percepton used for classification

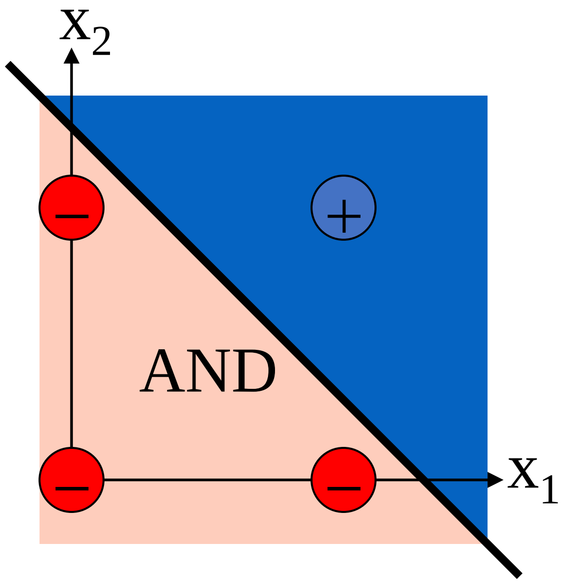
Exercise: Plot the Separation Lines for “AND”

AND

$W1 = 1$

$W2 = 1$

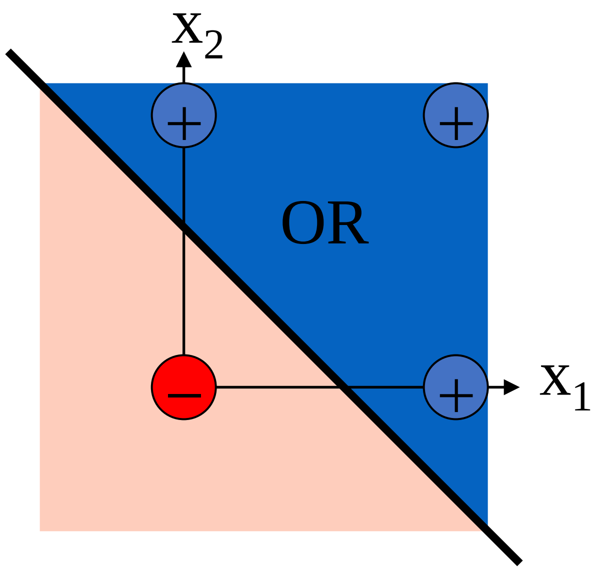
$W0/\theta = 1.5$



Exercise: Plot the Separation Lines for “OR”

OR

W1 = 1
W2 = 1
W0 / θ = 0.5



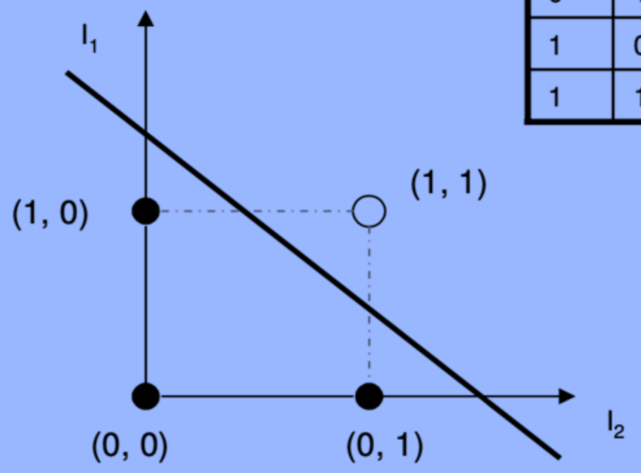
Decision Surfaces/Boundaries for AND and OR

We can now plot the decision boundaries of our logic gates

AND

w1=1, w2=1, θ=1.5

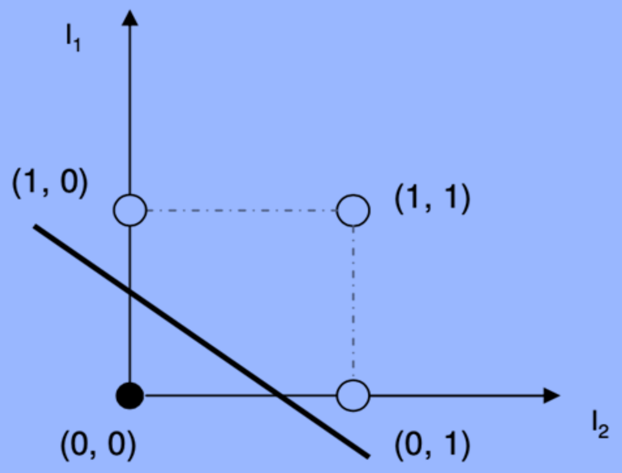
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



OR

w1=1, w2=1, θ=0.5

OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



How to Learn "OR Perceptron"? Training Input Samples

Consider learning the logical OR function.

Our examples are:

Sample	x0	x1	x2	label
1	1	0	0	0
2	1	0	1	1
3	1	1	0	1
4	1	1	1	1

Activation Function

$$S = \sum_{k=0}^{k=n} w_k x_k \quad S > 0 \; then \; O = 1 \quad else \quad O = 0$$

Recall: Typical Perceptron Weight Updates

- Weights modified for each example
- Update Rule:

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

learning rate | target value | perceptron output | input value

OR Perceptron
Weight Update Rule

$$S = \sum_{k=0}^{k=n} w_k x_k \quad S > 0 \; then \; O = 1 \quad else \quad O = 0$$

Weight Update (We set learning rate =1)

If perceptron output is 0 while it should be 1,

add the input vector to the weight vector (if input = 1, you add 1)

(if input =0, you can assume that you add 0)

If perceptron output is 1 while it should be 0,

subtract the input vector to the weight vector if input x is 1

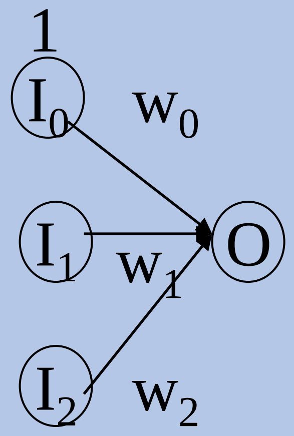
(if input =0, you substract 0)

Otherwise do nothing.

OR Perceptron
Learn First 2 Examples in Epoch 1

We'll use a single perceptron with three inputs.

We'll start with all weights 0 W= <0,0,0>



Example 1 I= < 1 0 0> label=0 W= <0,0,0>

Perceptron (1×0+ 0×0+ 0×0 =0, S=0) output → 0

→it classifies it as 0, so correct, do nothing

Example 2 I=<1 0 1> label=1 W= <0,0,0>

Perceptron (1×0+ 0×0+ 1×0 = 0) output →0

→it classifies it as 0, while it should be 1, so add input to weights

W = <0,0,0> + <1,0,1>= <1,0,1>

OR Perceptron
Learn Next 2 Samples in Epoch 1

Example 3 I=<1 1 0> label=1 W= <1,0,1>

Perceptron ($1\times1+ 1\times0+ 0\times1 > 0$) output = 1

→it classifies it as 1, while it should be 1, so do nothing

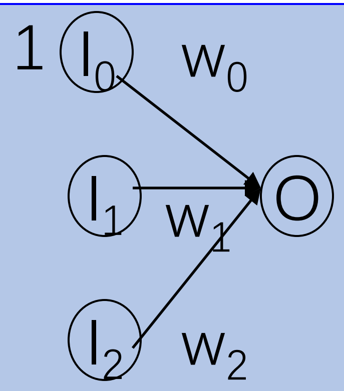
Example 4 I=<1 1 1> label=1 W= <1,0,1>

Perceptron ($1\times1+ 1\times0+ 1\times1 > 0$) output = 1

→it classifies it as 1, correct, do nothing

W = <1,0,1>

OR Perceptron Training and Learning Example
Learn First 2 Samples in Epoch 2



Epoch 2, through the examples, W = <1,0,1> .

Example 1 I = <1,0,0> label=0 W = <1,0,1>

Perceptron (1×1+ 0×0+ 0×1 >0) output → 1

→it classifies it as 1, while it should be 0,

so subtract input from weights

W = <1,0,1> - <1,0,0> = <0, 0, 1>

Example 2 I=<1 0 1> label=1 W= <0,0,1>

Perceptron (1×0+ 0×0+ 1×1 > 0) output →1

→it classifies it as 1, so correct, do nothing

OR Perceptron Training and Learning Example Learn Next 2 Samples in Epoch 2

Example 3 I=<1 1 0> label=1 W= <0,0,1>

Perceptron $(1\times0+ 1\times0+ 0\times1 > 0)$ output = 0

→it classifies it as 0, while it should be 1, so

add input to weights

W = <0,0,1> + W = <1,1,0> = <1, 1, 1>

Example 4 I=<1 1 1> label=1 W= <1,1,1>

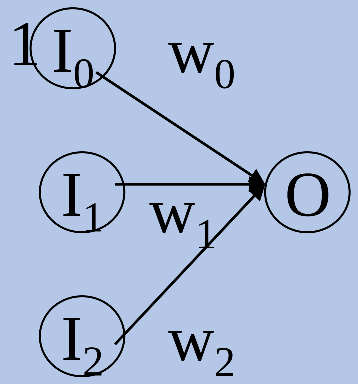
Perceptron $(1\times1+ 1\times1+ 1\times1 > 0)$ output = 1

→it classifies it as 1, correct, do nothing

W = <1,1,1>

OR Perceptron Training and Learning Example
Learn First 2 Samples in Epoch 3

Epoch 3, through the examples, W = <1,1,1> .



Example 1 I=<1,0,0> label=0 W = <1,1,1>

Perceptron (1×1+ 0×1+ 0×1 >0) output → 1

→it classifies it as 1, while it should be 0, so

subtract input from weights

W = <1,1,1> - W = <1,0,0> = <0, 1, 1>

Example 2 I=<1 0 1> label=1 W= <0, 1, 1>

Perceptron (1×0+ 0×1+ 1×1 > 0) output →1

→it classifies it as 1, so correct, do nothing

OR Perceptron Training and Learning Example
Learn Next 2 Samples in Epoch 3

Example 3 I=<1 1 0> label=1 W= <0, 1, 1>

Perceptron ($1\times0+ 1\times1+ 0\times1 > 0$) output = 1

→it classifies it as 1, correct, do nothing

Example 4 I=<1 1 1> label=1 W= <0, 1, 1>

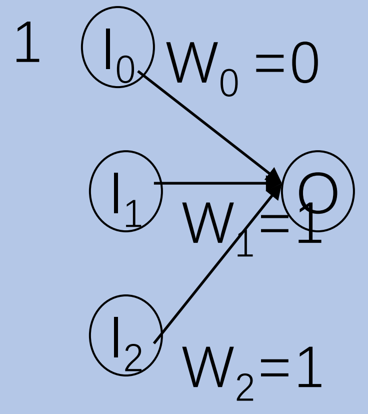
Perceptron ($1\times0+ 1\times1+ 1\times1 > 0$) output = 1

→it classifies it as 1, correct, do nothing

W = <1,1,1>

OR Perceptron Training and Learning Example
Learn First Samples in Epoch 4

Epoch 4, through the examples, W= <0, 1, 1>.



OR

Example 1 I= <1,0,0> label=0 W = <0,1,1>

Perceptron ($1 \times 0 + 0 \times 1 + 0 \times 1 = 0$) output → 0

→it classifies it as 0, so correct, do nothing

So the final weight vector W= <0, 1, 1> classifies all examples correctly, and the perceptron has learned the function!

In more realistic cases the bias (W0) will not be 0.
Also, in general, many more inputs (100 to 1000….)

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1
example 3	1	1	0	1	0	0	1	0	1	1	1	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1
example 3	1	1	0	1	0	0	1	0	1	1	1	1
example 4	1	1	1	1	1	1	1	1	0	1	1	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1
example 3	1	1	0	1	0	0	1	0	1	1	1	1
example 4	1	1	1	1	1	1	1	1	0	1	1	1
3 example 1	1	0	0	0	1	1	1	1	-1	0	1	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1
example 3	1	1	0	1	0	0	1	0	1	1	1	1
example 4	1	1	1	1	1	1	1	1	0	1	1	1
3 example 1	1	0	0	0	1	1	1	1	-1	0	1	1
example 2	1	0	1	1	0	1	1	1	0	0	1	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1
example 3	1	1	0	1	0	0	1	0	1	1	1	1
example 4	1	1	1	1	1	1	1	1	0	1	1	1
3 example 1	1	0	0	0	1	1	1	1	-1	0	1	1
example 2	1	0	1	1	0	1	1	1	0	0	1	1.
example 3	1	1	0	1	0	1	1	1	0	0	1	1

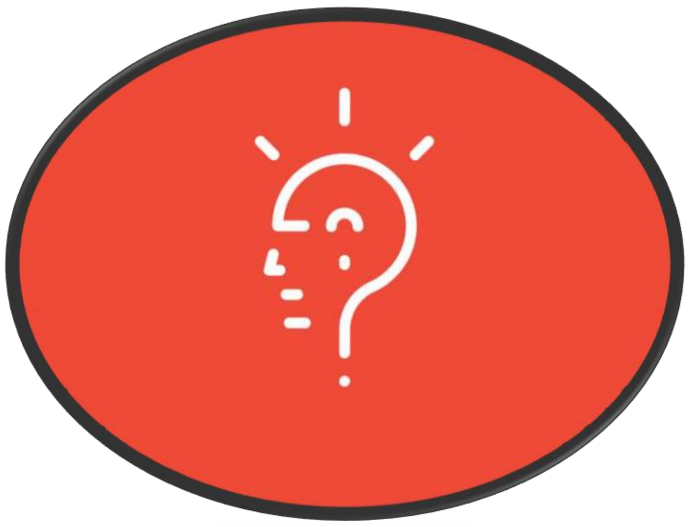
OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1
example 3	1	1	0	1	0	0	1	0	1	1	1	1
example 4	1	1	1	1	1	1	1	1	0	1	1	1
3 example 1	1	0	0	0	1	1	1	1	-1	0	1	1
example 2	1	0	1	1	0	1	1	1	0	0	1	1.
example 3	1	1	0	1	0	1	1	1	0	0	1	1
example 4	1	1	1	1	0	1	1	1	0	0	1	1

OR Perceptron Learning Summary

Epoch	x0	x1	x2	Desired Target	w0	w1	w2	Output	Error	New w0	New w1	New w2
1 example 1	1	0	0	0	0	0	0	0	0	0	0	0
example 2	1	0	1	1	0	0	0	0	1	1	0	1
example 3	1	1	0	1	1	0	1	1	0	1	0	1
example 4	1	1	1	1	1	0	1	1	0	1	0	1
2 example 1	1	0	0	0	1	0	1	1	-1	0	0	1
example 2	1	0	1	1	0	0	1	1	0	0	0	1
example 3	1	1	0	1	0	0	1	0	1	1	1	1
example 4	1	1	1	1	1	1	1	1	0	1	1	1
3 example 1	1	0	0	0	1	1	1	1	-1	0	1	1
example 2	1	0	1	1	0	1	1	1	0	0	1	1
example 3	1	1	0	1	0	1	1	1	0	0	1	1
example 4	1	1	1	1	0	1	1	1	0	0	1	1
4 example 1	1	0	0	0	0	1	1	0	0	0	1	1

Any Question?



Perceptron

3 4

1 Perceptron

2 Perceptron Learning

3 ADALINE

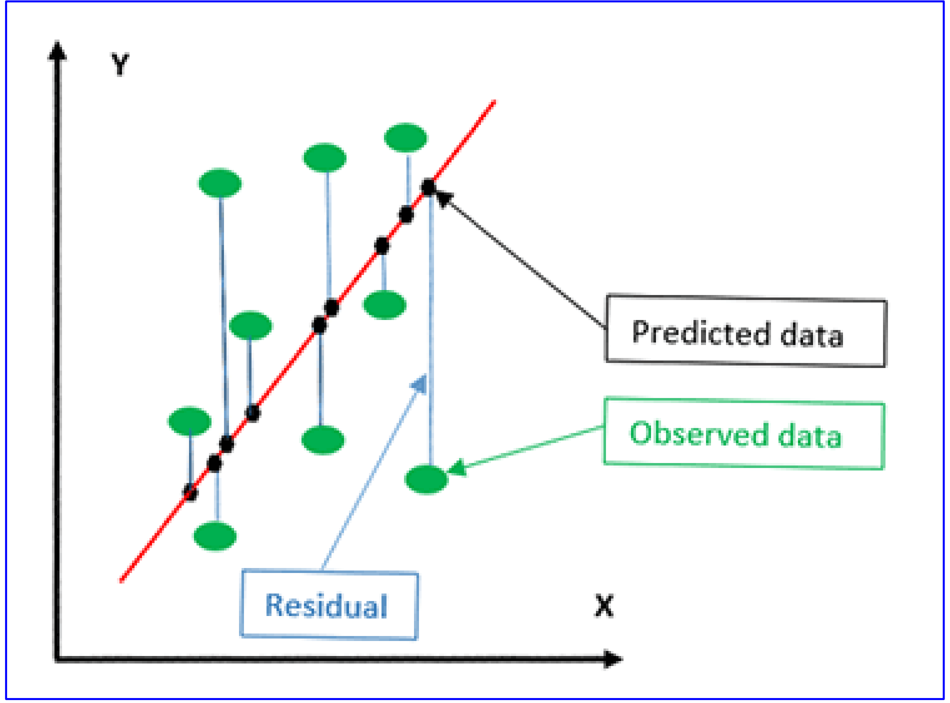
4 Limitation of Perceptron

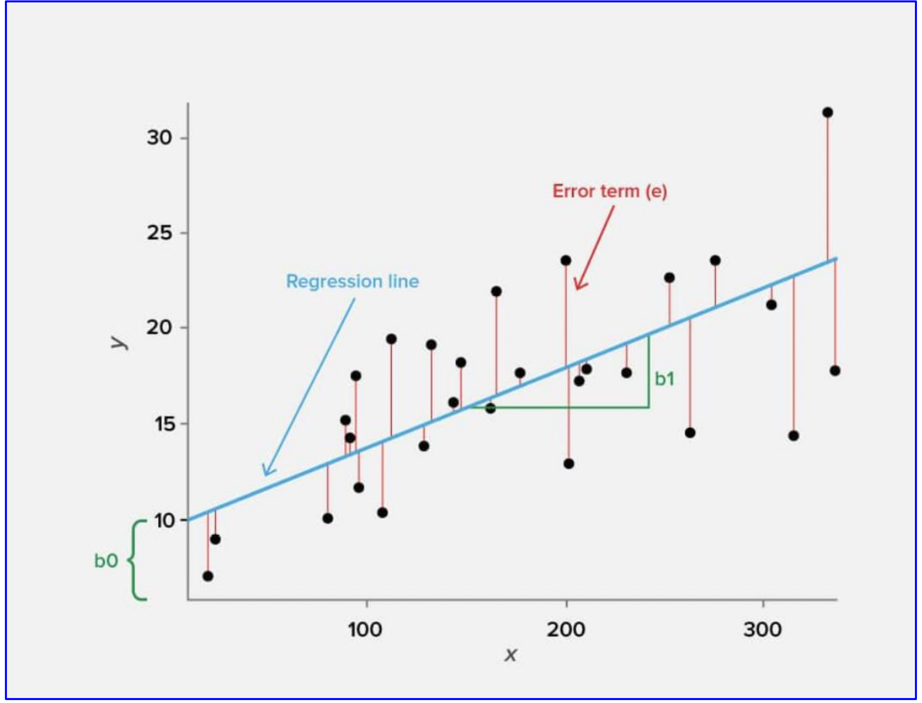
Q2: What Does “Linear Regression”Mean?

Linear Regression

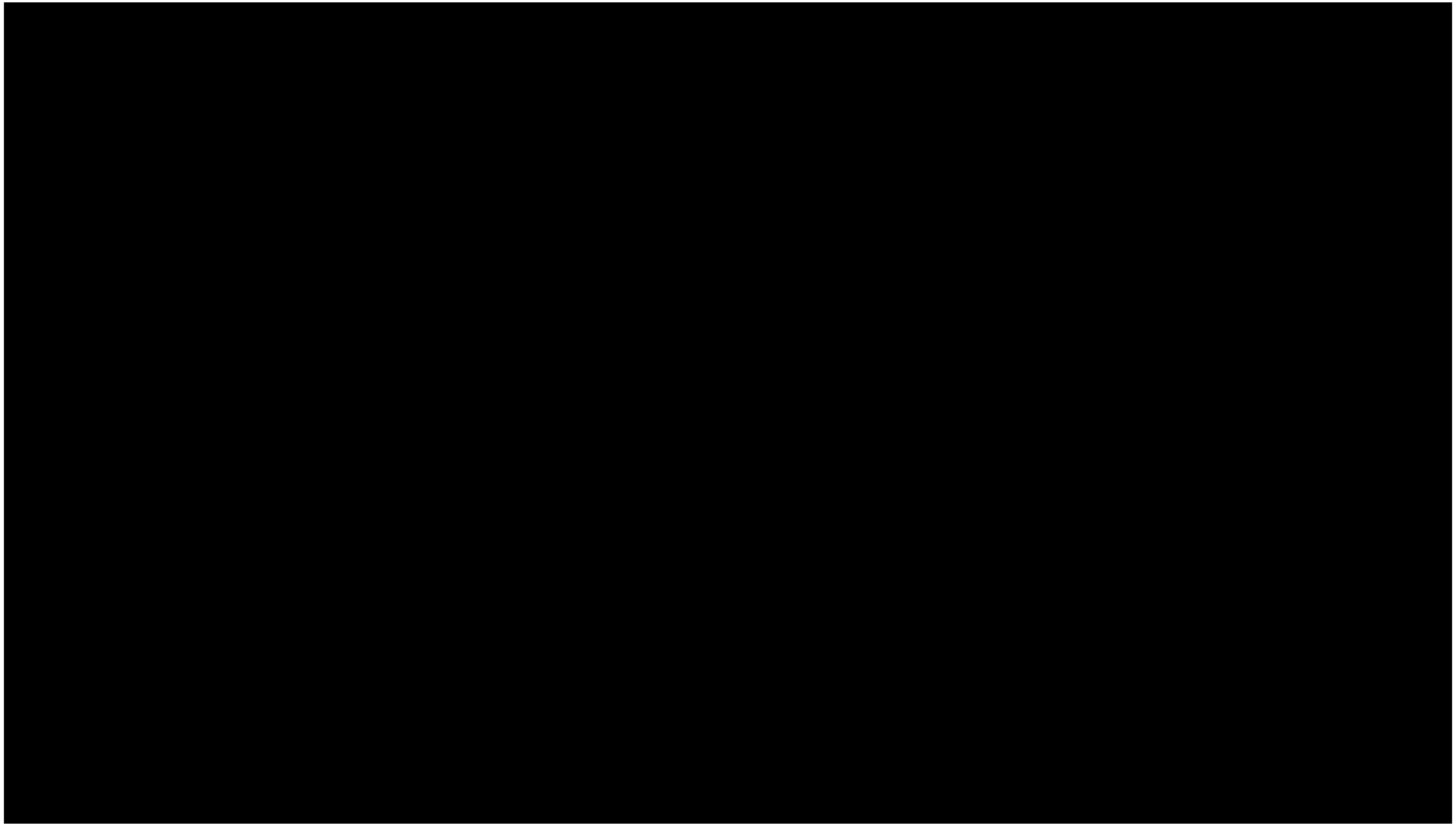
Linear Regression

- Linear Regression is a statistical procedure that determines the equation for the straight line that best fits a specific set of data.





Classification and Regression

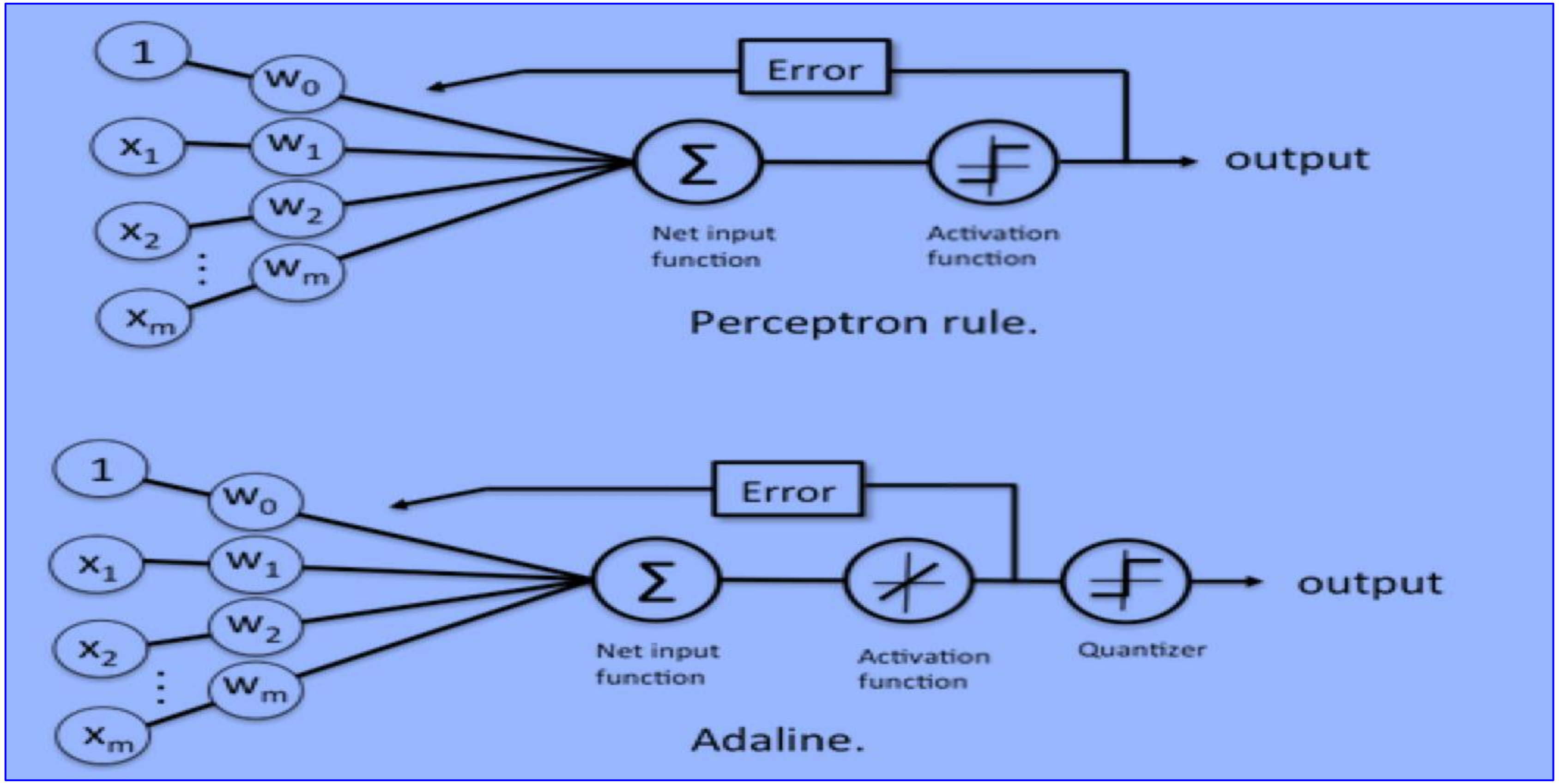


ADALINE (Adaptive Linear Neuron)

ADALINE is an early single-layer artificial neural network based on Least Mean Squares (LMS) algorithms. 最小均方算法

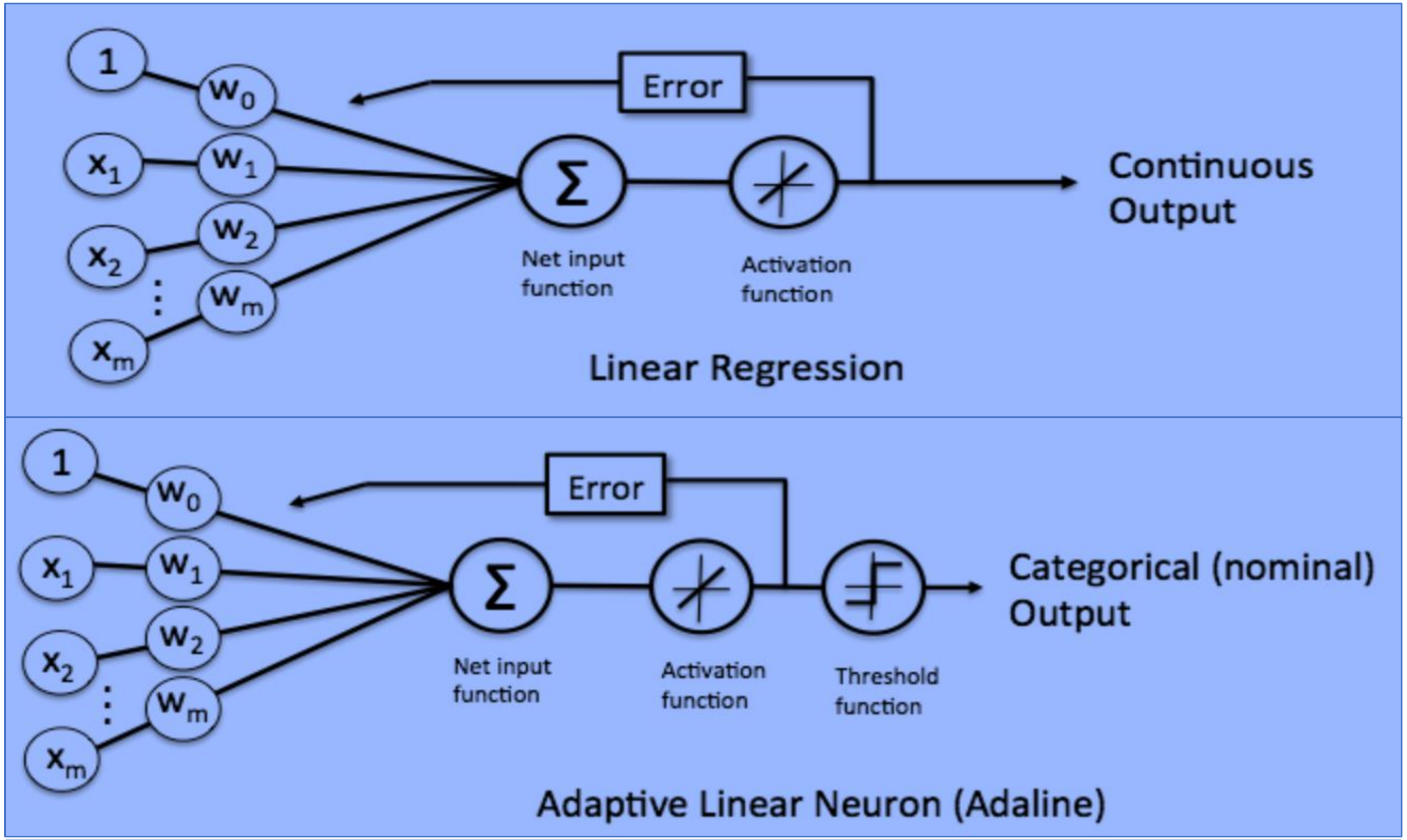
It was invented in 1960 by Stanford University science professor Bernard Widrow and his first Ph.D. student, Ted Hoff.

Perceptron and ADALINE



In the perceptron, we use the predicted class labels to update the weights, and in ADALINE, we use output to update,
it tells us by “how much” we were right or wrong

Linear Regression and ADALINE



Adaline algorithm is identical to linear regression except for a threshold function that converts the continuous output into a categorical class label

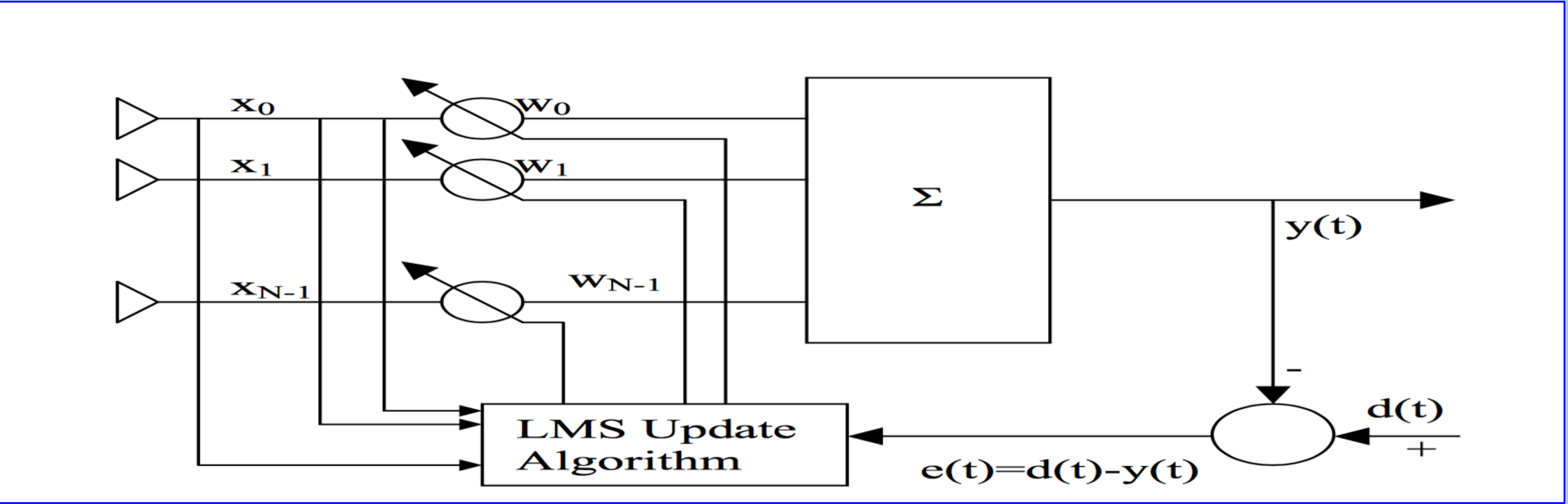
Widrow Hoff Learning Algorithm

- Also known as **Delta Rule**. It follows gradient descent rule for linear regression. It updates the connection weights with the difference between the target and the output value. It is the least mean square learning algorithm falling under the category of the supervised learning algorithm.

- This rule is followed by ADALINE (ADAptive LInear NEuron or NEural Networks) and **MADALINE**. Unlike Perceptron, the iterations of Adaline networks do not always stop, but it converges by reducing the least mean square error. MADALINE is a network of more than one ADALINE.

ADALINE (Adaptive Linear Neuron)

LMS algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean square of the error signal. It is a stochastic gradient descent method, which does not require gradient to be know and it is estimated at every iteration.



* B.Widrow and M.E.Hoff, "Adaptive switching circuits," Proc. Of WESCON Conv. Rec., part 4, pp.96-140, 1960

Delta Learning Rule

- The motive of the delta learning rule is to minimize the error between the output and the target vector. The weights in ADALINE networks are updated by:

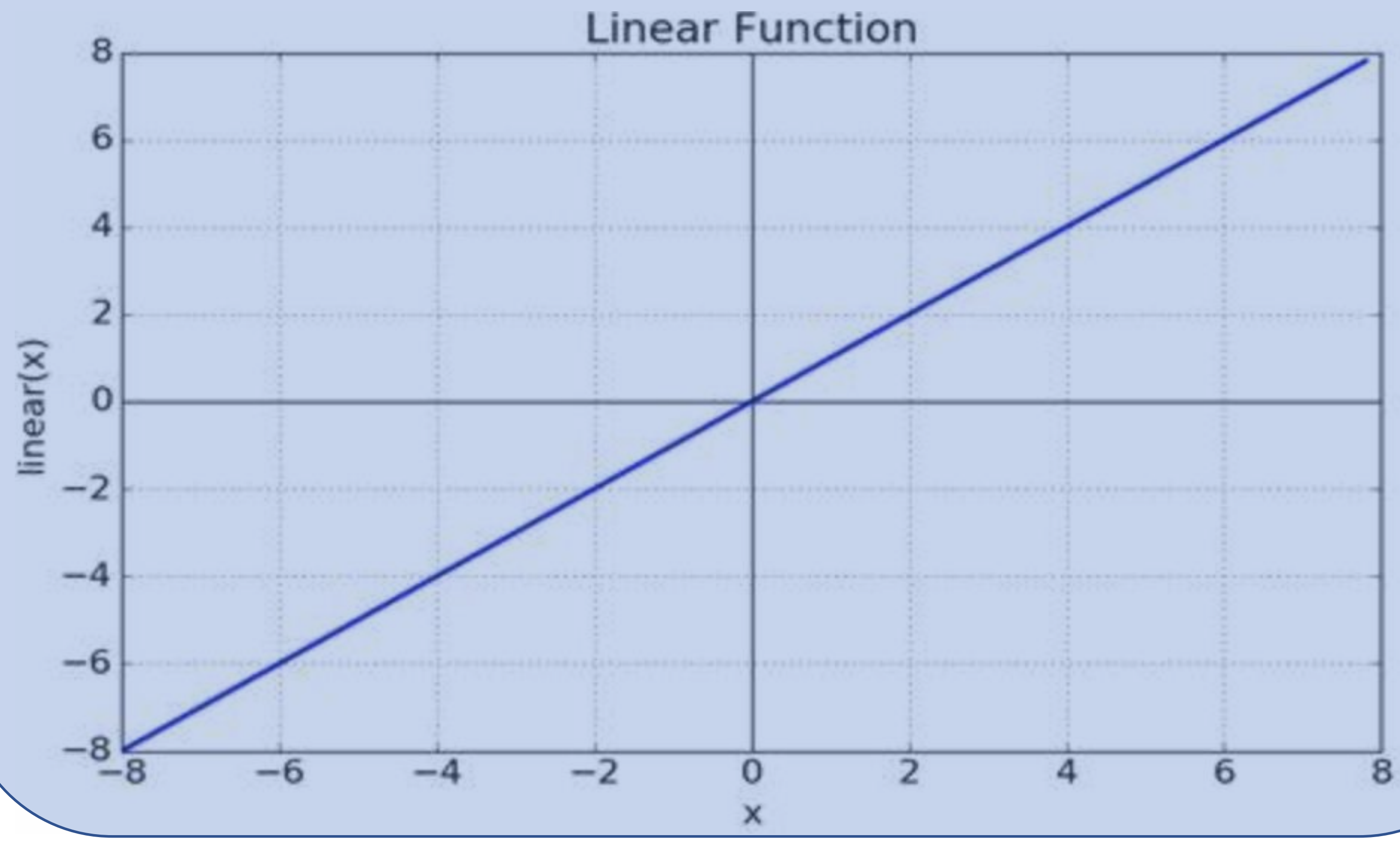
  Least Mean Square error (LMS) = $(t-O)^2$,

  ADALINE converges when the least mean square error is reached.

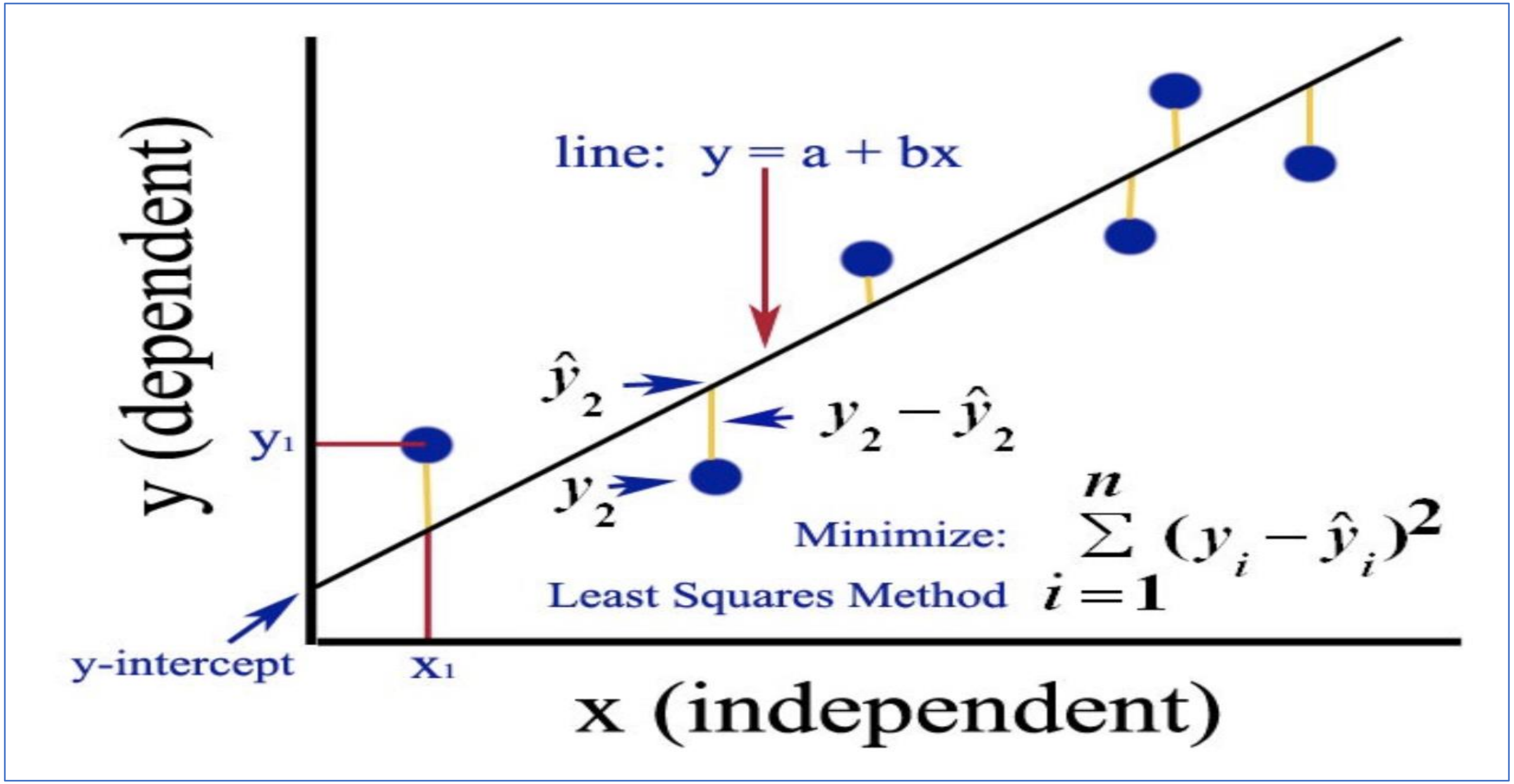
- Learning is an optimization search problem in weight space

$$\Delta w_i = \eta(t-o)x_i$$

"Artificial" Neuron
Linear Transfer (Activation) Function



Least Sum of Squared Errors (SSE) for MADALINE

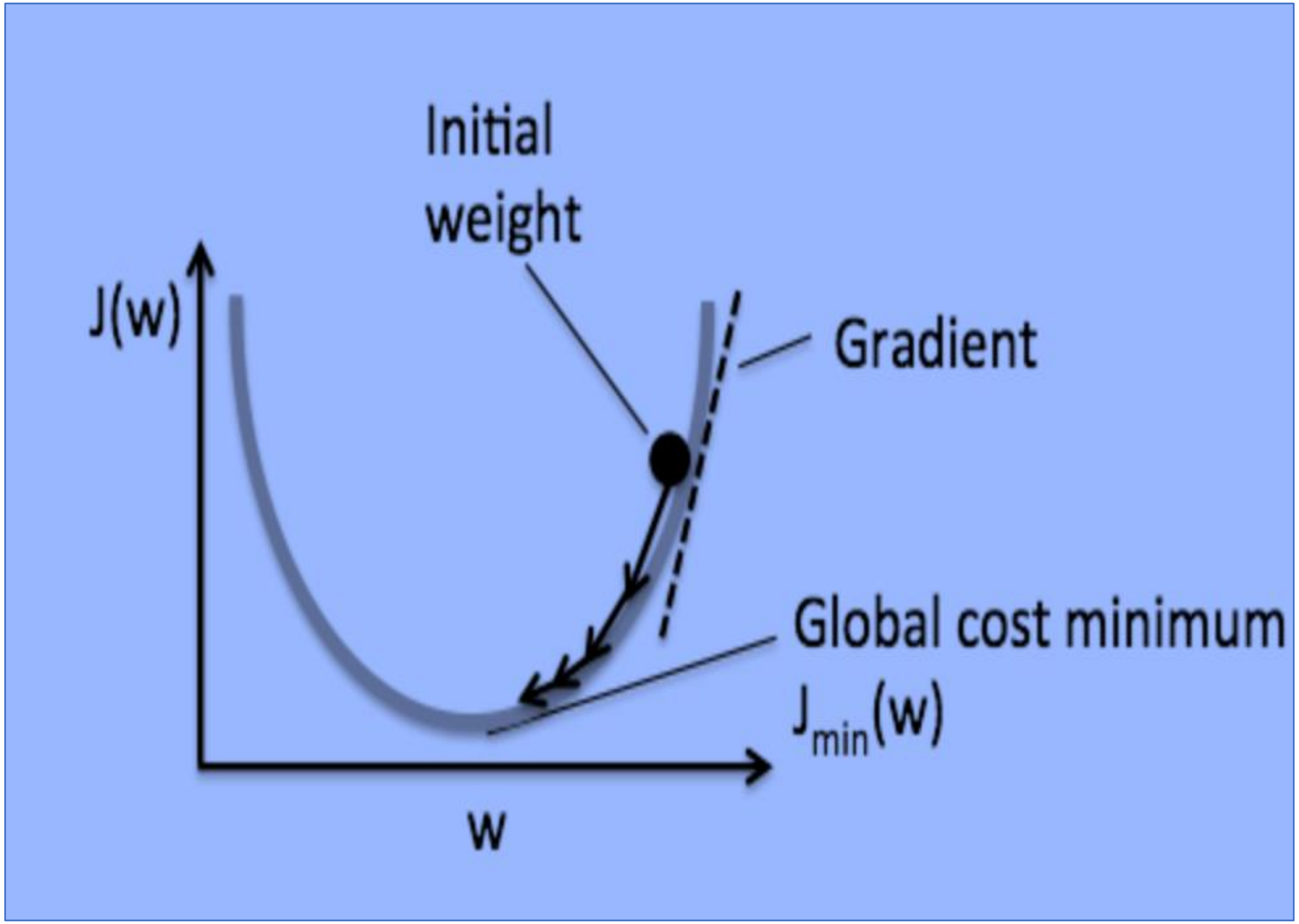


LMS Gradient Descent

Gradient Descent

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. But if we instead take steps proportional to the positive of the gradient, we approach a local maximum of that function; the procedure is then known as gradient ascent. Gradient descent is generally attributed to Cauchy, who first suggested it in 1847, but its convergence properties for non-linear optimization problems were first studied by Haskell Curry in 1944.

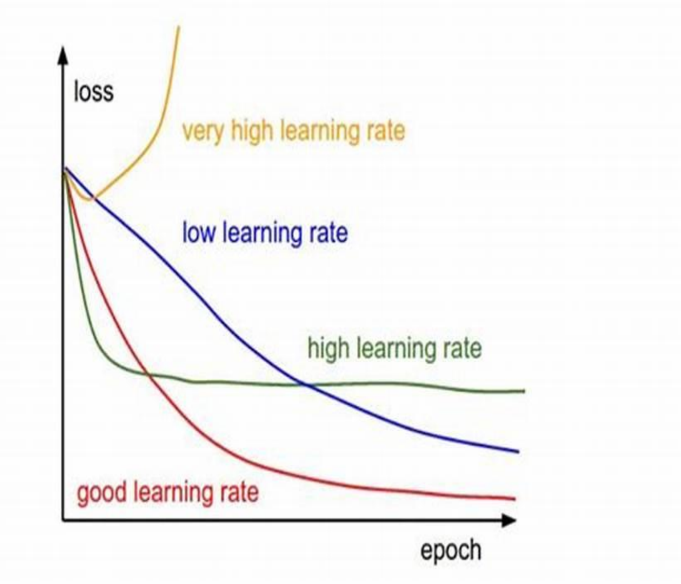
Wikipedia



LMS Gradient Calculation

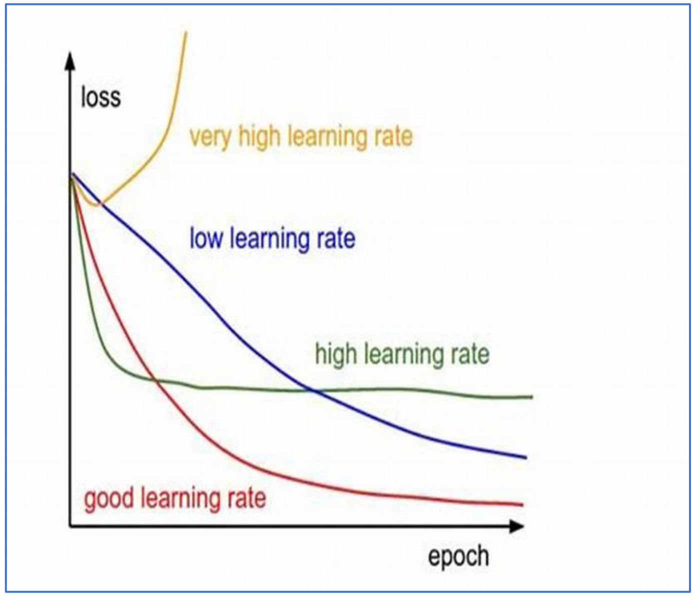
$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

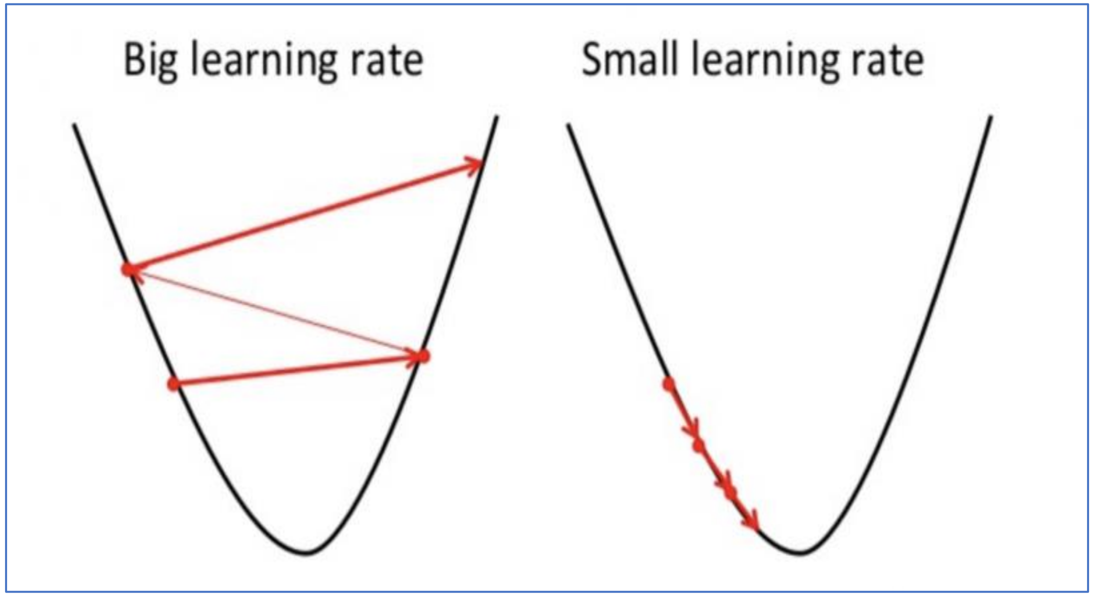
$$\begin{aligned}
&\frac{\partial J}{\partial w_j} \\
&= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2 \\
&= \frac{1}{2} \frac{\partial}{\partial w_j} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2 \\
&= \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \phi(z)_A^{(i)}\right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \sum_i \left(w_j^{(i)} x_j^{(i)}\right)\right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)\left(-x_j^{(i)}\right) \\
&= -\sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) x_j^{(i)}
\end{aligned}$$



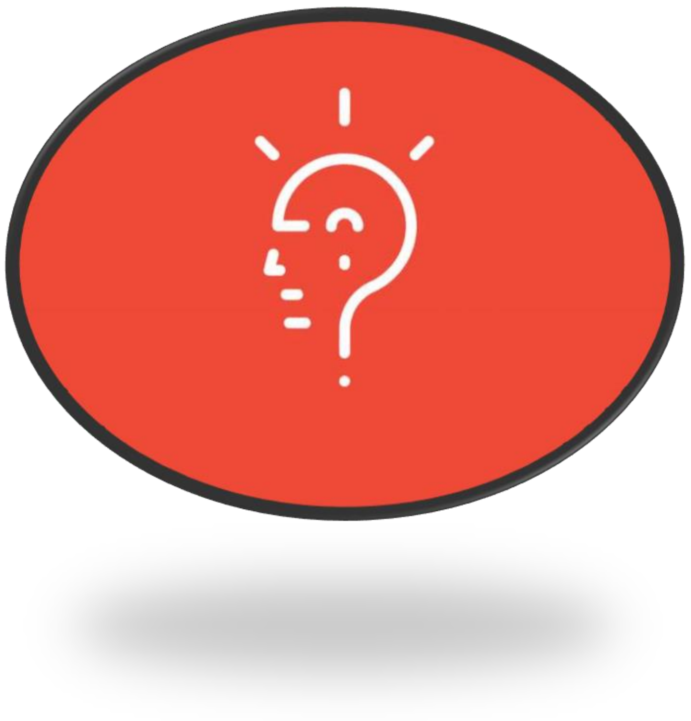
LMS Gradient Calculation

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$



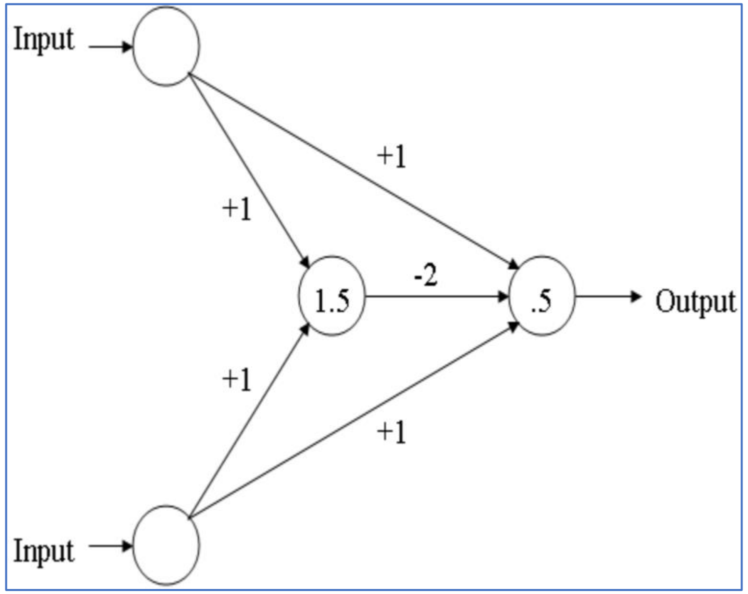


Any Question?



Homework 08

1 Prove the network is an XOR network



Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

2 Update Your Project Progress in 2 Minutes in Morning Class and 3 Minutes in Afternoon Class Next Lecture

CS 103 -09
Perceptron Learning and ADALINE

Jimmy Liu 刘江

2021-11-12