AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

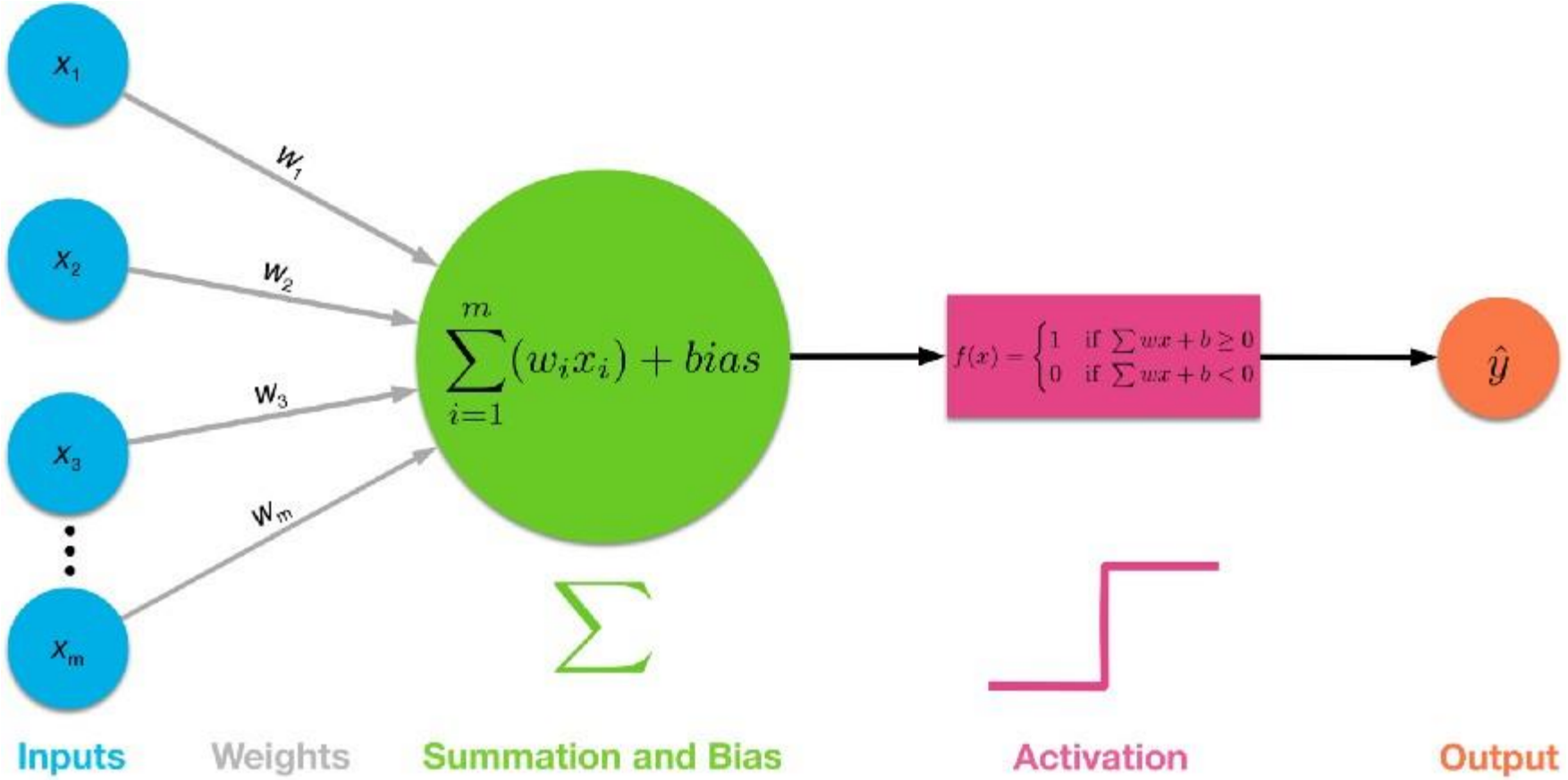
श्रद्धावान् लभते ज्ञानम्

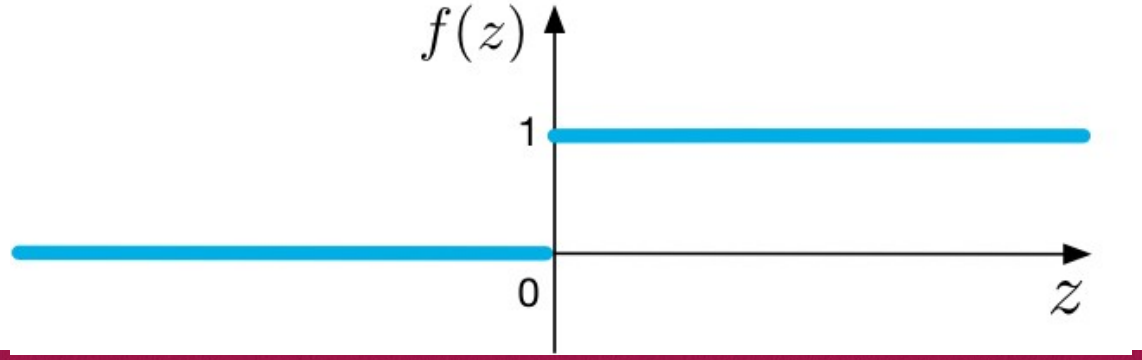
19CSE437
DEEP LEARNING FOR COMPUTER VISION
L-T-P-C: 2-0-3-3

Sigmoid Neuron

*Citation Note: The content, of this presentation were inspired by the the materials offered by Prof. **Mitesh M. Khapra** on NPTEL's Deep Learning course, fast.ai,Andrew NG Coursera , internet sources*

Procedures of a Single-layer Perceptron Network



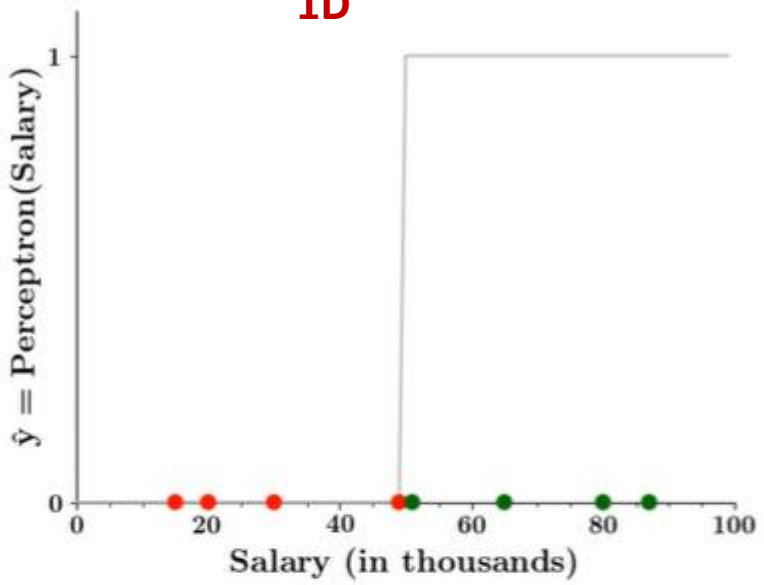


Step -Activation Function

Courtesy: towardsdatascience.com

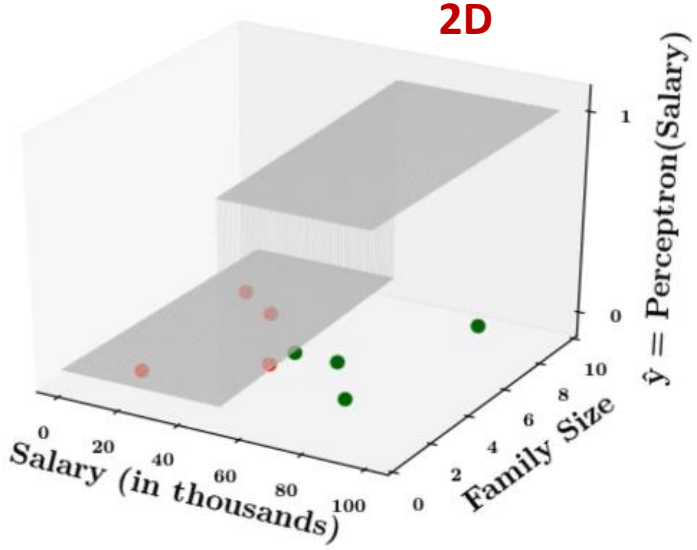
Limitations of Perceptron

1D



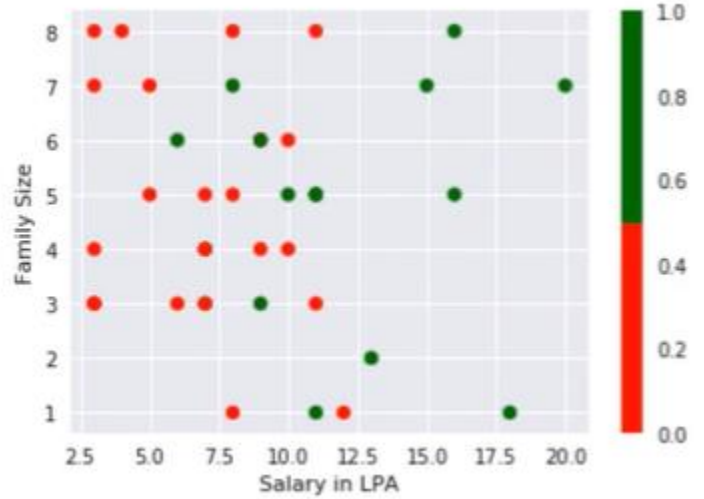
Salary (in thousands)	Can buy a car?
80	1
20	0
65	1
15	0
30	0
49	0
51	1
87	1

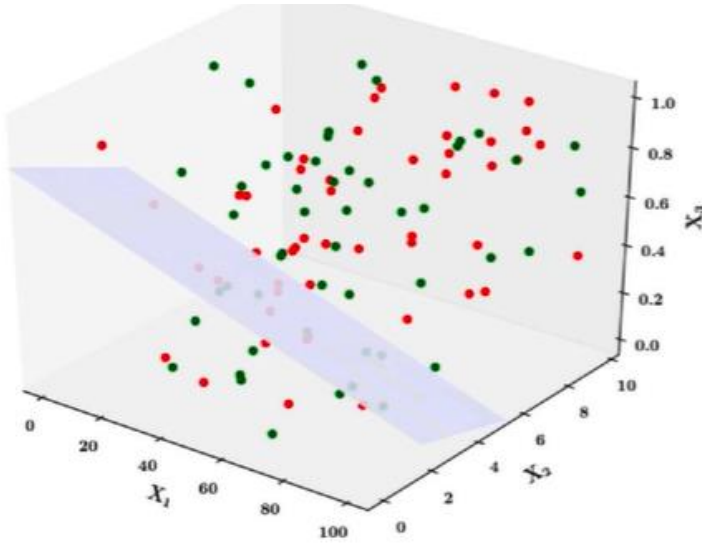
2D



Salary (in thousands)	Family size	Can buy a car?
80	2	1
20	1	0
65	4	1
15	7	0
30	6	0
49	3	0
51	4	1
87	8	1

How about these? Non linearly separable data?

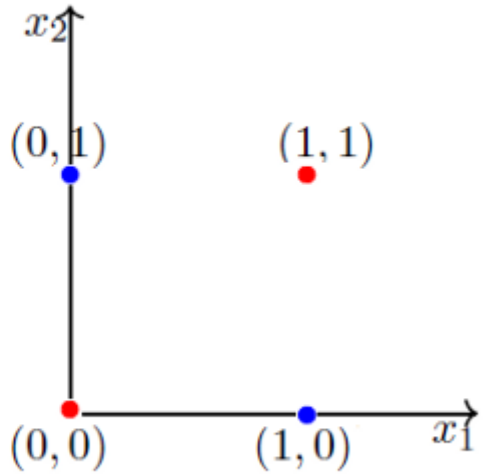




XOR Function — Can't Do!- Non Linearly separated data

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 < -w_0$$



Notice that the fourth equation contradicts the second and the third equation. Point is, there are no *perceptron* solutions for non-linearly separated data. So the key take away is that a **single** *perceptron* cannot learn to separate the data that are non-linear in nature.

Sigmoid Neuron-

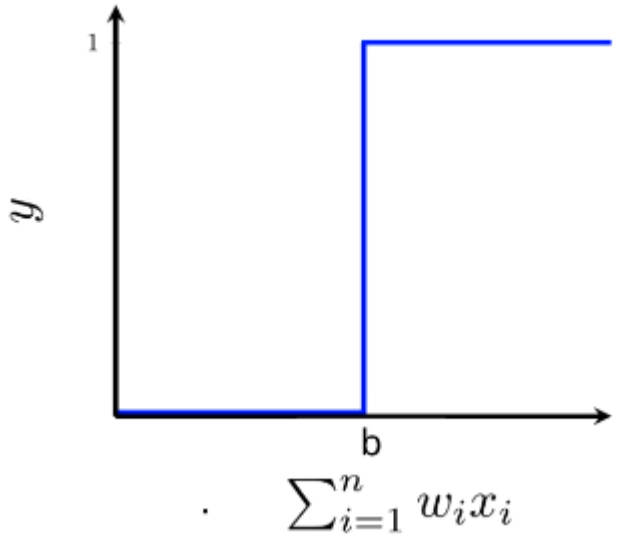
A smoother activation function

The Sigmoid Function curve looks like a S-shape.

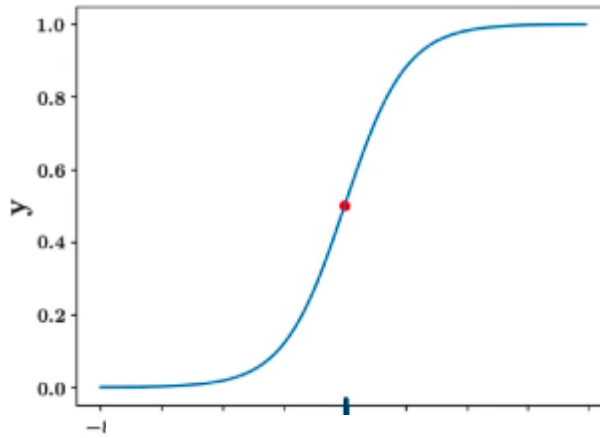
It exists between **(0 to 1).** especially used for models where we have to predict the probability as an output.

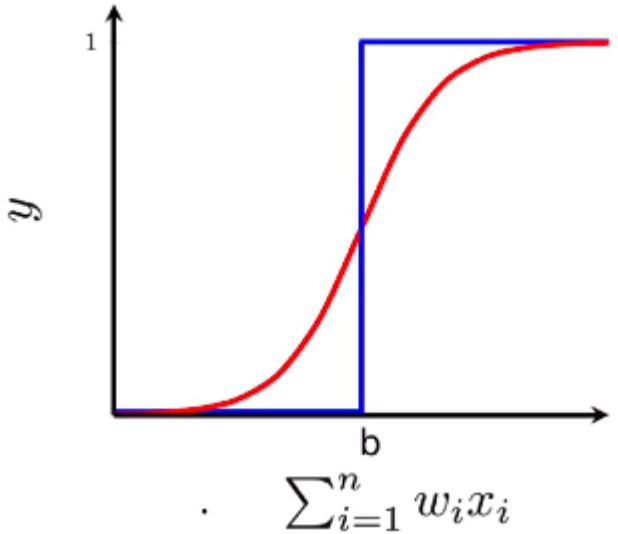
The function is **differentiable**.That means, we can find the slope of the sigmoid curve at any two points. It is **monotonic**

Step function



Sigmoid function





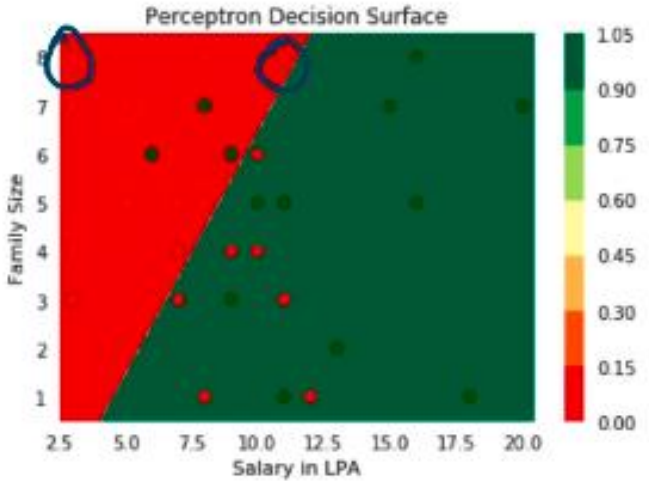
$$y = \frac{1}{1 + e^{-(wx+b)}}$$

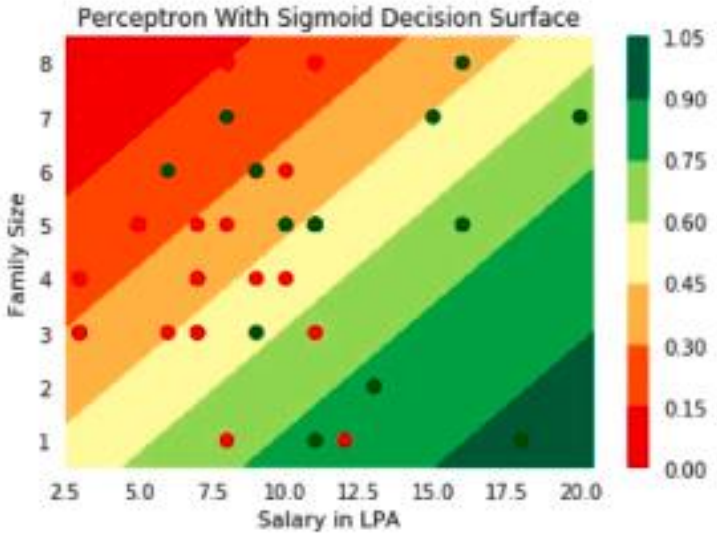
So the output is σ(w·x+b), where σ is called the Sigmoid Function and is def by:- $\sigma(z) \equiv \frac{1}{1+e^{-z}}$.

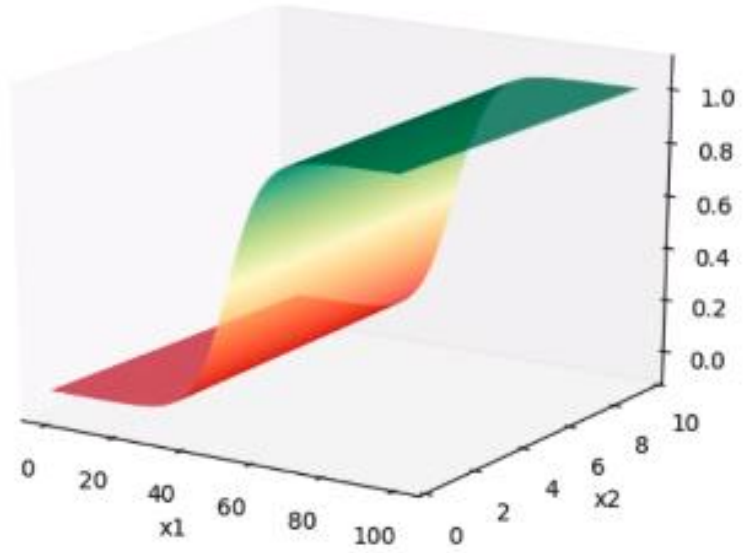
A monotonic function is a **function which is either entirely nonincreasing or nondecreasing.**

Sigmoid -2D

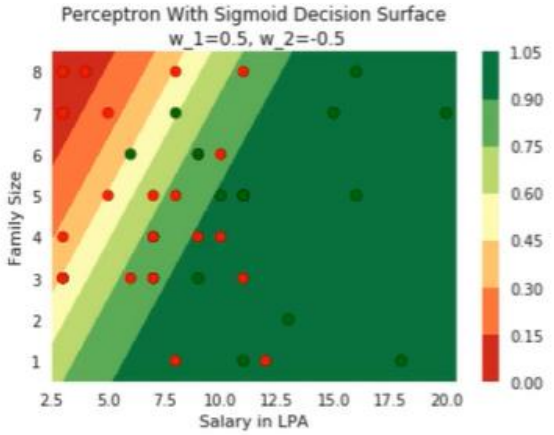
$$y = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}}$$

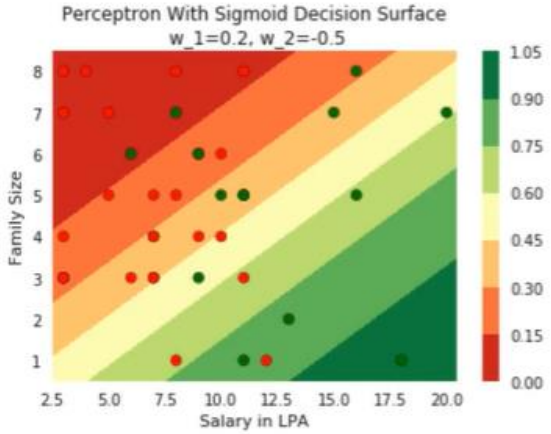


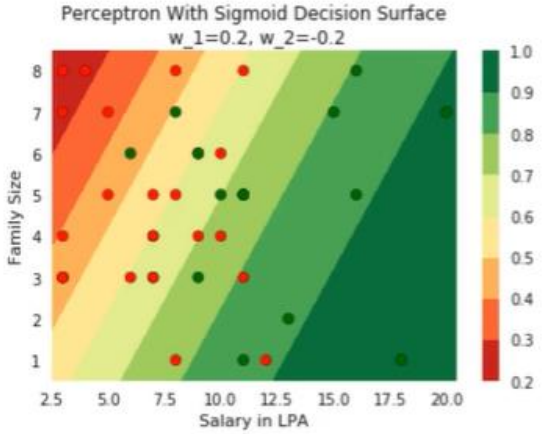




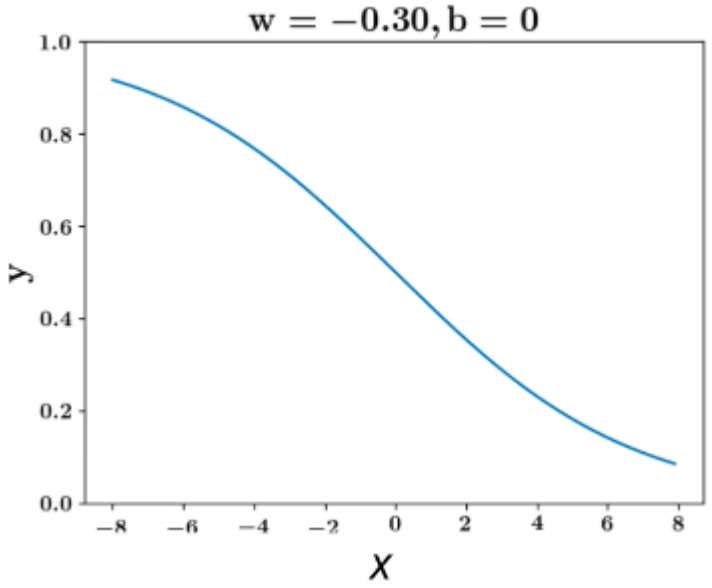
	Salary in LPA	Family Size	Buys Car?
0	11	8	1
1	20	7	1
2	4	8	0
3	8	7	0
4	11	5	1







Loss Function-Sigmoid

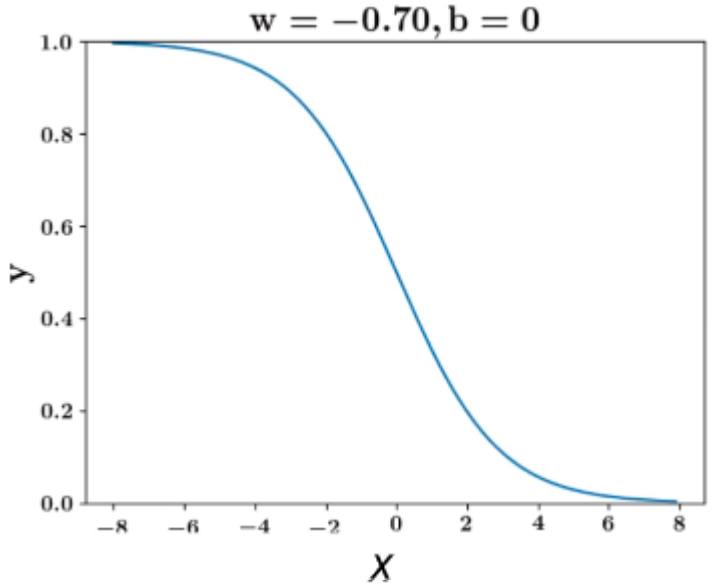


$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

Loss Function-Sigmoid

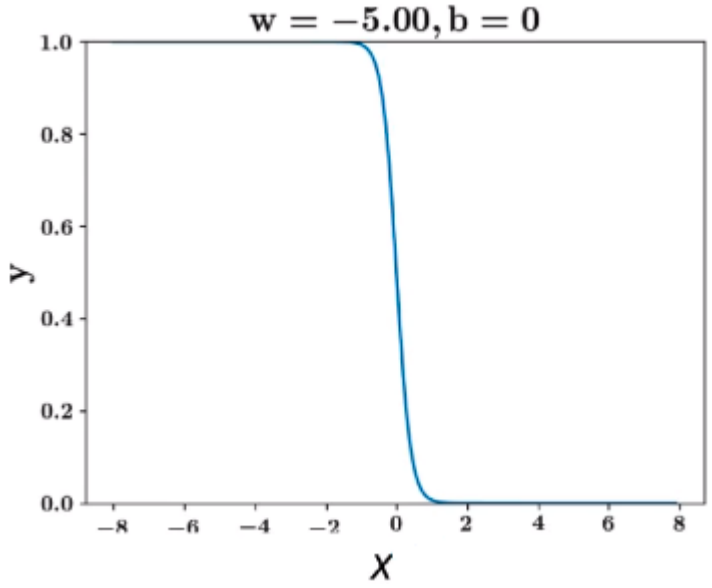


$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

Loss Function-Sigmoid

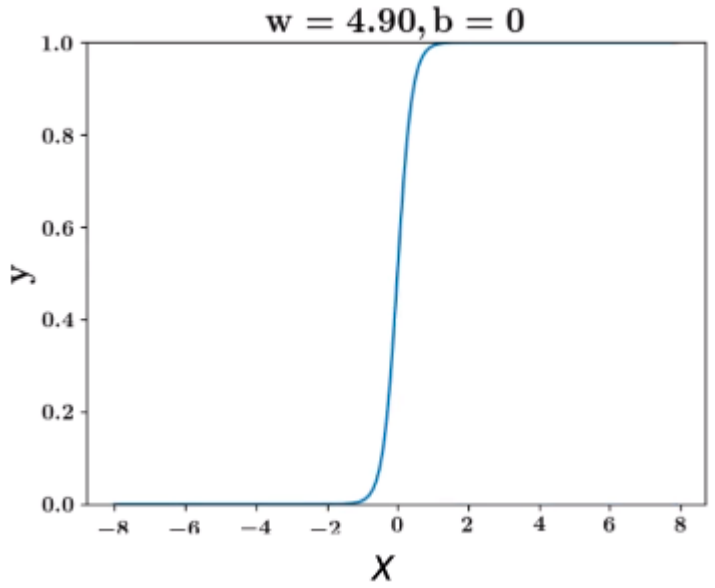


$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

Loss Function-Sigmoid

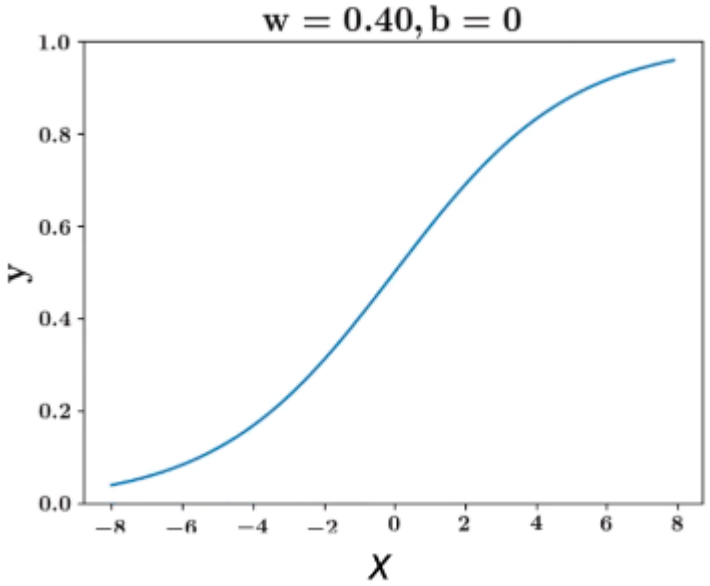


$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

Loss Function-Sigmoid

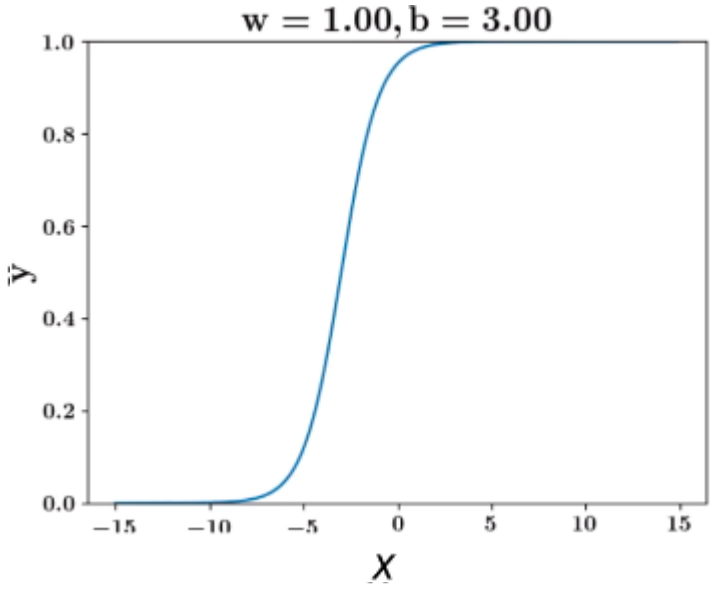


$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

Loss Function-Sigmoid

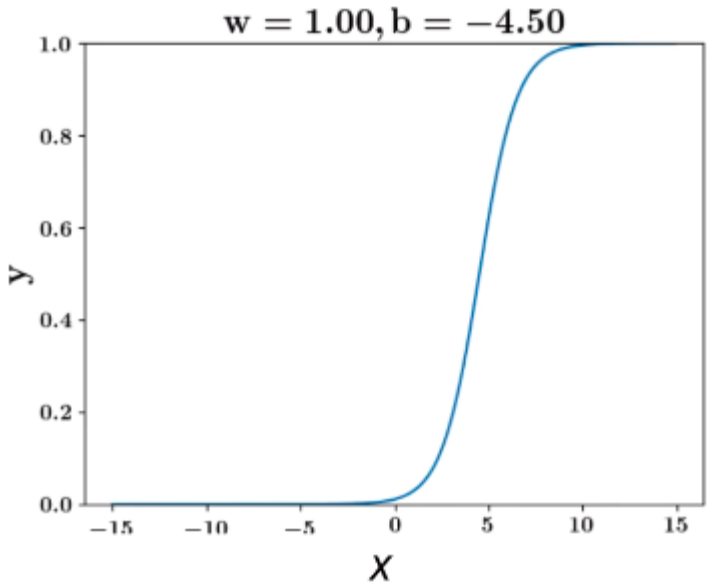


$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

Loss Function-Sigmoid

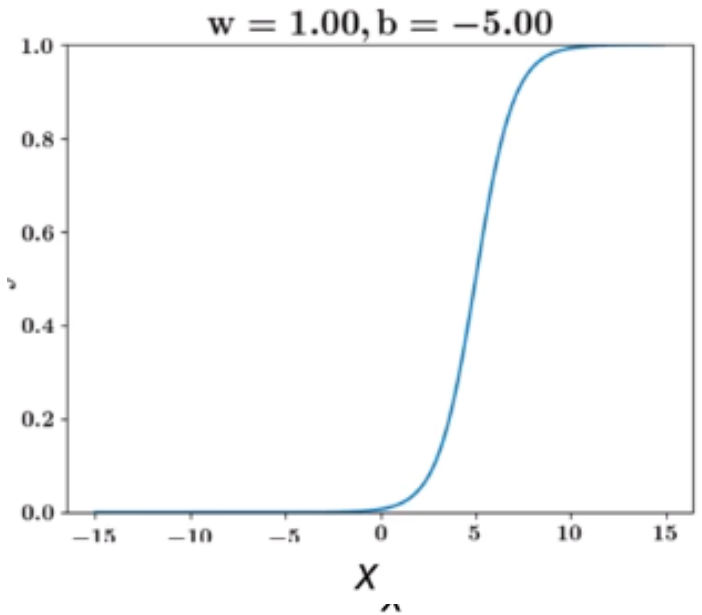


$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

Loss Function-Sigmoid



$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$Loss = \sum_{i=1}^{4} (y - \hat{y})^2 = 0.18$$

x_1	x_2	y	$\hat{y}$
1	1	0.5	0.6
2	1	0.8	0.7
1	2	0.2	0.2
2	2	0.9	0.5

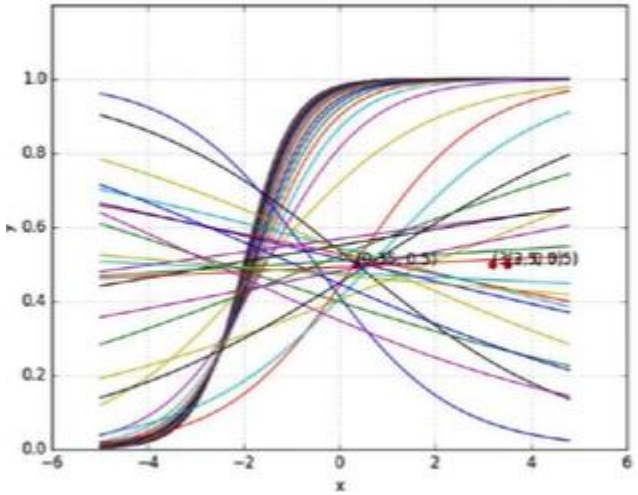
Learning Algorithm

Initialise

Iterate over data:

Update w and b

till satisfied



I/P	O/P
2	0.047
3	0.268
4	0.73
5	0.952
8	0.999

$$h = \frac{1}{1+e^{-(w*x+b)}}$$

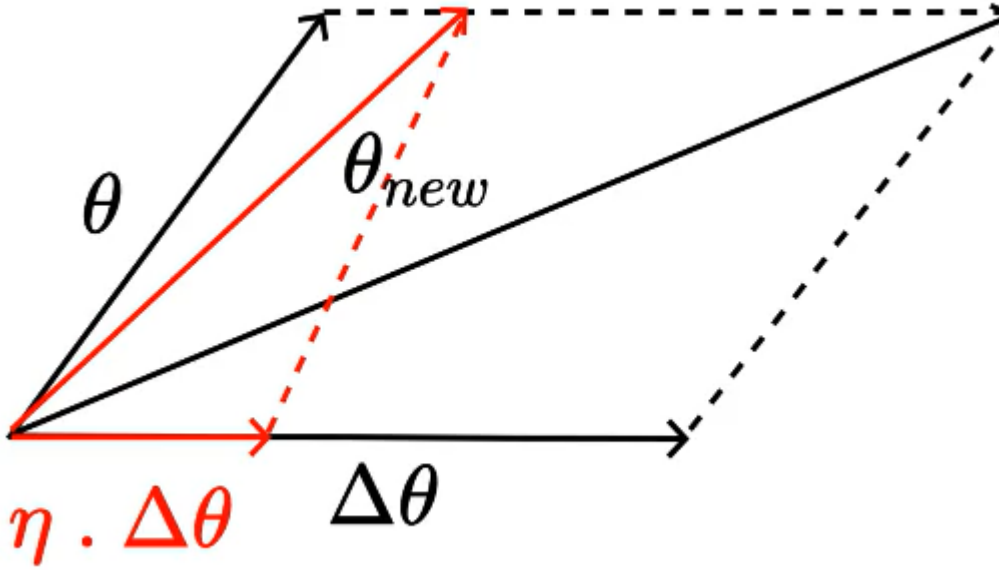
- Calculate $\hat{y}$ with sigmoid
- Try to minimize y- $\hat{y}$ with loss function
- How to update is the next question
- $\min_{w,b} \; Loss\mathscr{L}(w,b)$

Learning Algorithm

$$\theta = [w, b]$$

$$\Delta\theta = [\Delta w, \Delta b]$$

$$\theta_{new} = \theta + \eta \,.\, \Delta\theta$$

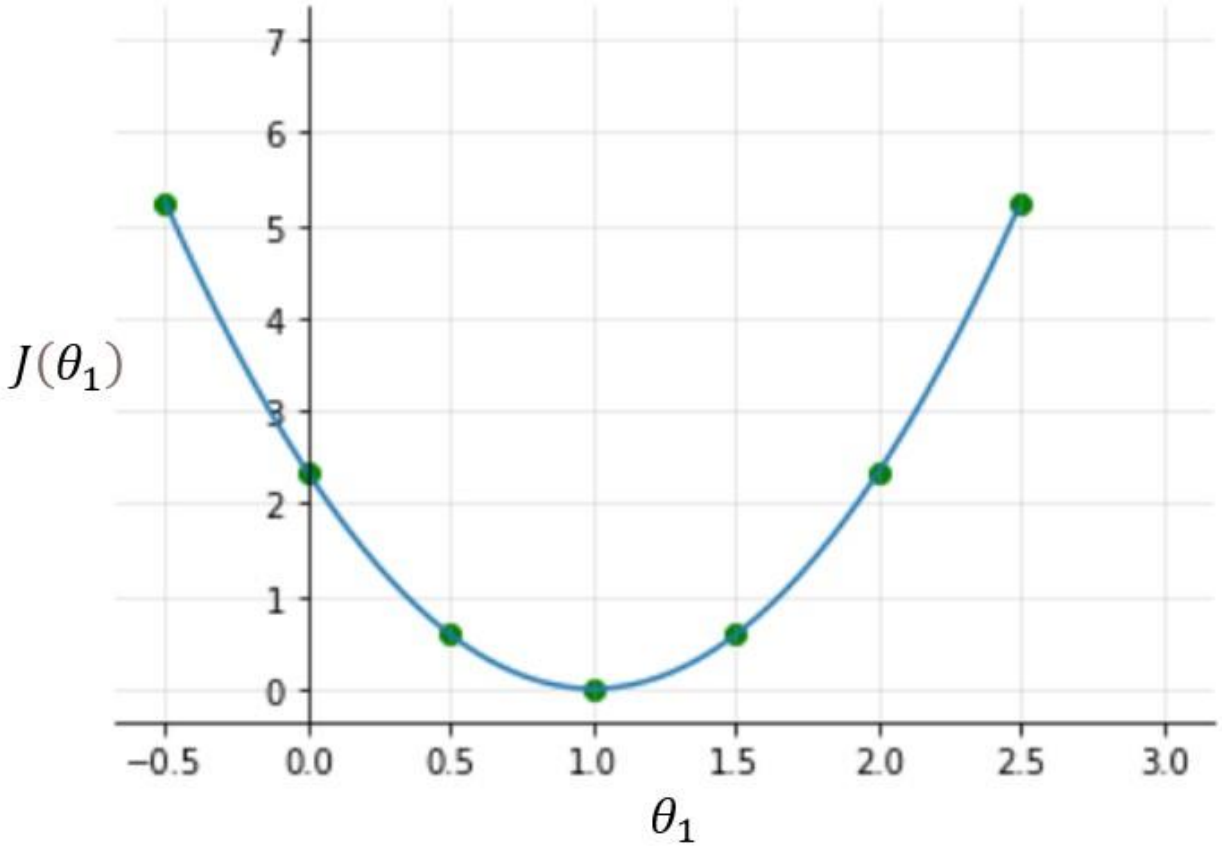



```
while !convergence do
    Pick random x ∈ P ∪ N;
    if x ∈ P and Σ_{i=0}^{n} w_i * x_i < 0 then
        w = w + x;
    end
    if x ∈ N and Σ_{i=0}^{n} w_i * x_i ≥ 0 then
        w = w − x;
    end
end
```

Instead of guessing Δw andΔ b, we need a principled way of changing w and b based on the loss function

Gradient descent optimization

Gradient Descent is the process of minimizing a function by following the **gradients** of the cost function. This involves knowing the form of the cost as well as the derivative so that from a given point you know the **gradient** and can move in that direction, e.g. downhill towards the minimum value.



Once you plot these all dots, the cost function will look like a bowl-shaped curve as shown in the figure below.

it is clear that the cost function is minimum **at theta1=1** or at the bottom of the bowl-shaped curve. The purpose of all this hard work is not to calculate the minimum value of cost function,
Value of the parameters, for the minimum cost function.

In machine learning, a gradient is a derivative of a function that has more than one input variable.

Gradient Descent

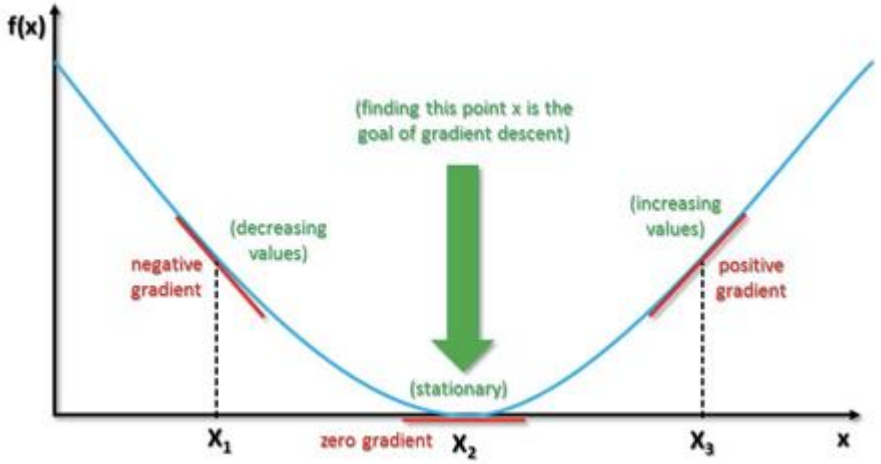
To find the local minimum of a function using gradient descent, we must take steps proportional to the negative of the gradient (move away from the gradient) of the function at the current point

A gradient measures how much the output of a function changes if you change the inputs a little bit.

Parameter Update rule

$$w_{t+1} = w_t - \eta \Delta w_t$$
$$b_{t+1} = b_t - \eta \Delta b_t$$

$$where\ \Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}\ at\ w{=}w_t, b{=}b_t, \Delta b_t = \frac{\partial \mathscr{L}(w,b)}{\partial b}\ at w{=}w_t, b{=}b_t$$



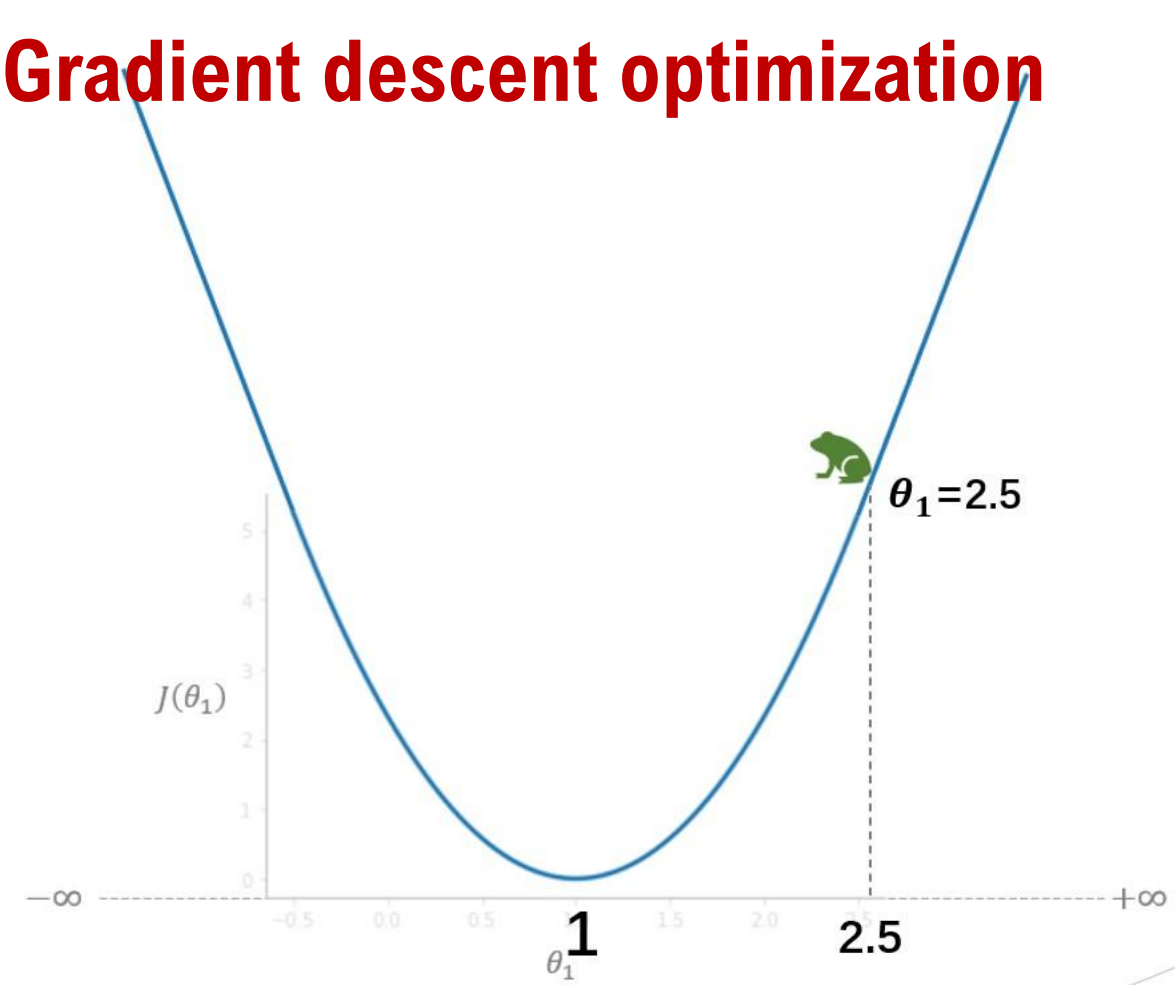
$$\min_{w,b}\ Loss\mathscr{L}(w,b)$$

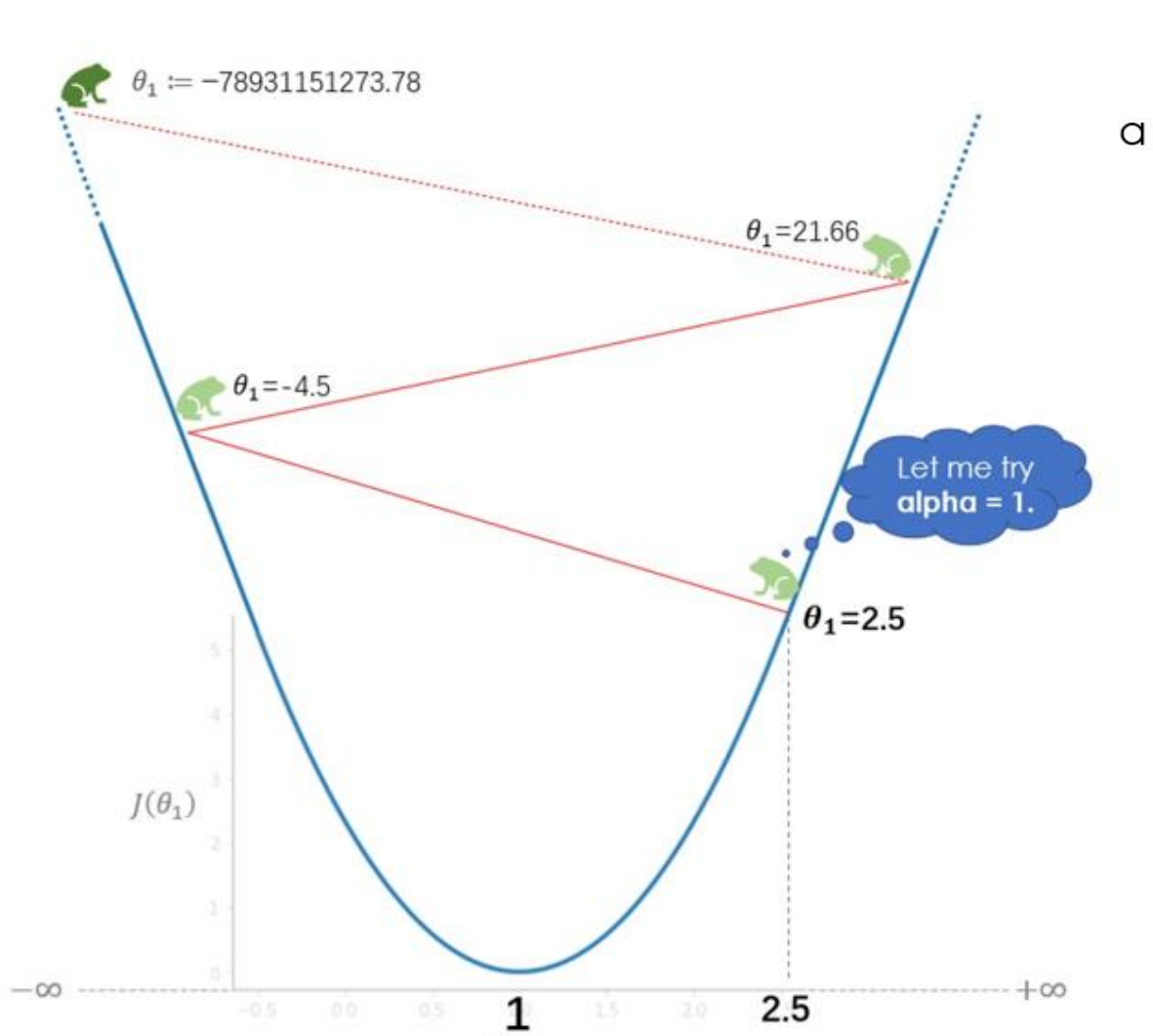
$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

Δw_t **Is the partial derivative of the Loss function w.r.t w.**

It can be proved that with this updation in weight and b parameter, the new Loss will be less than the old loss

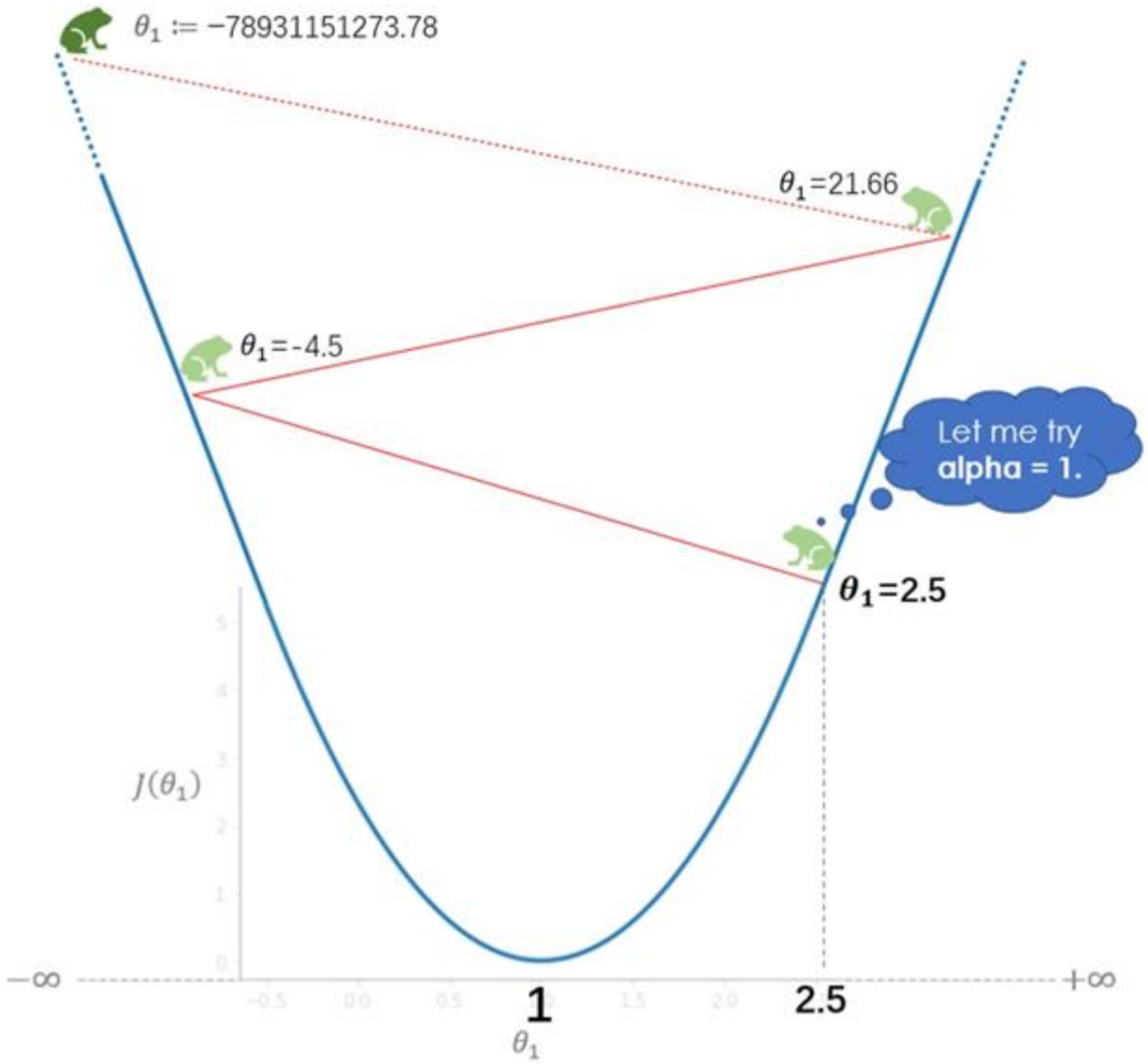
Gradient descent optimization

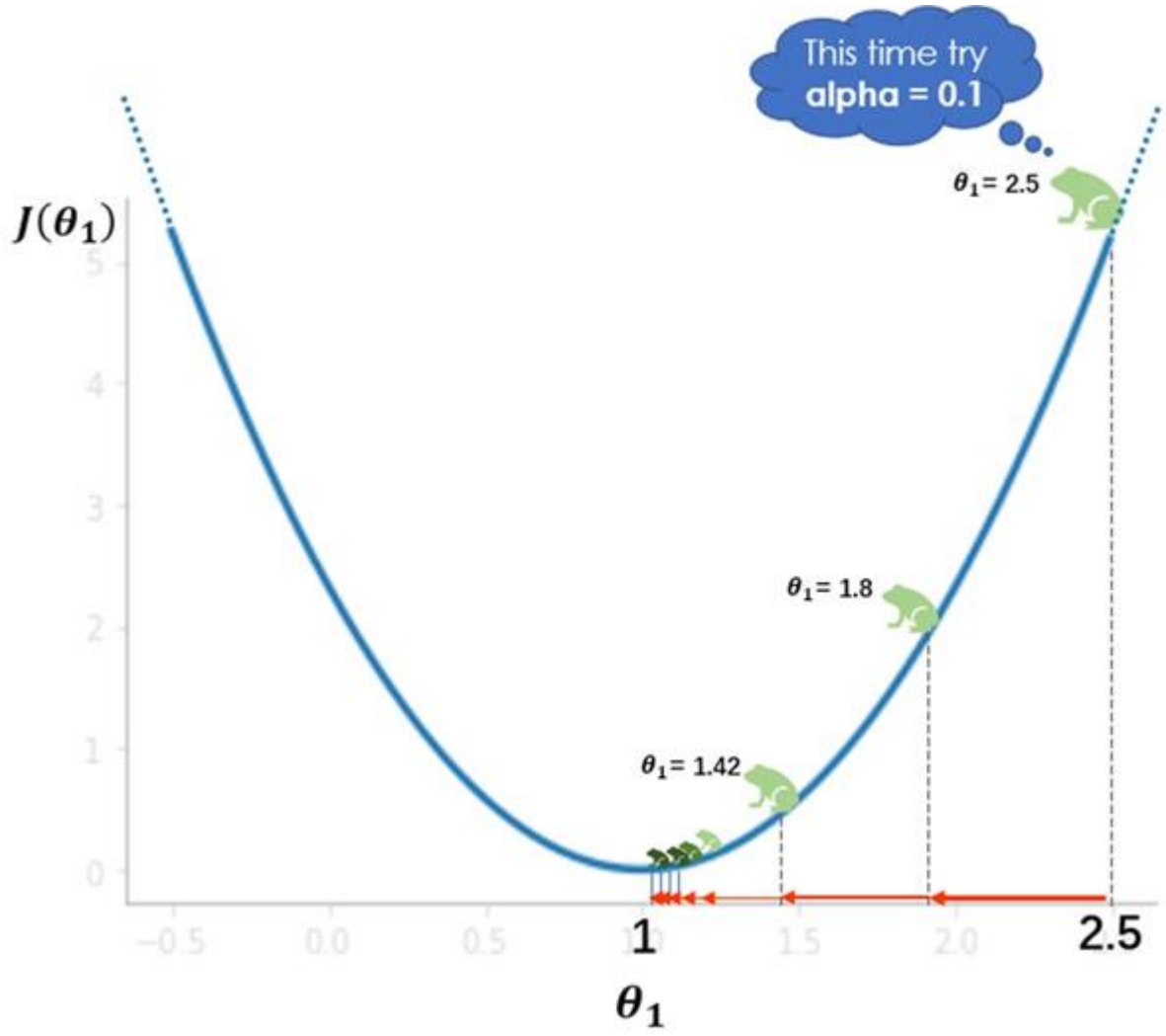




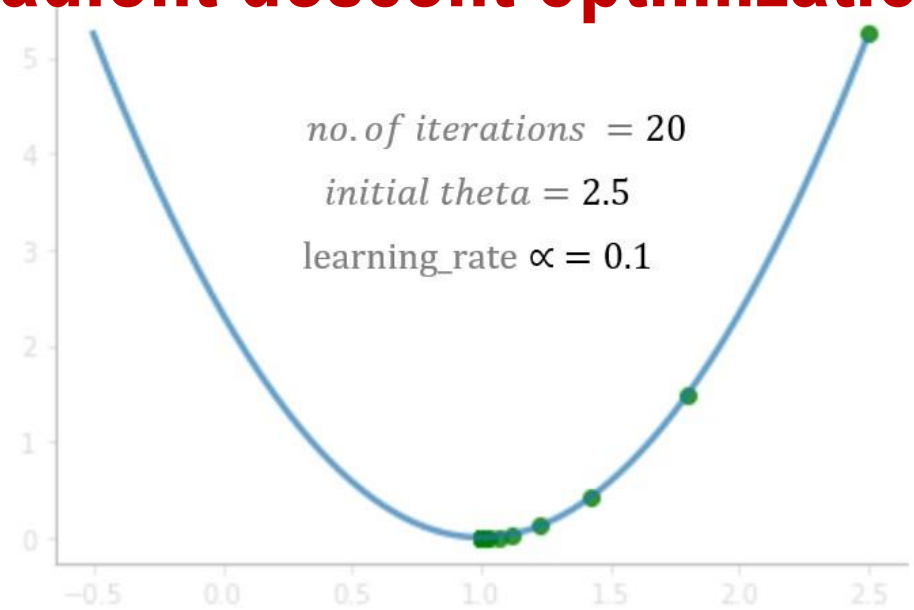
Alpha learning rate

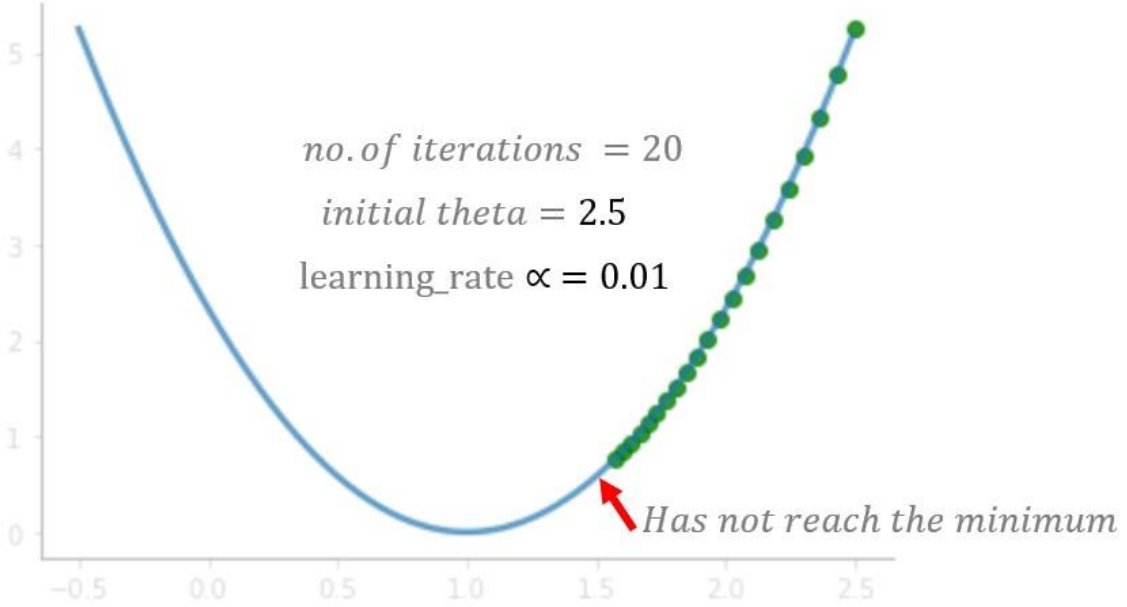
Gradient descent optimization

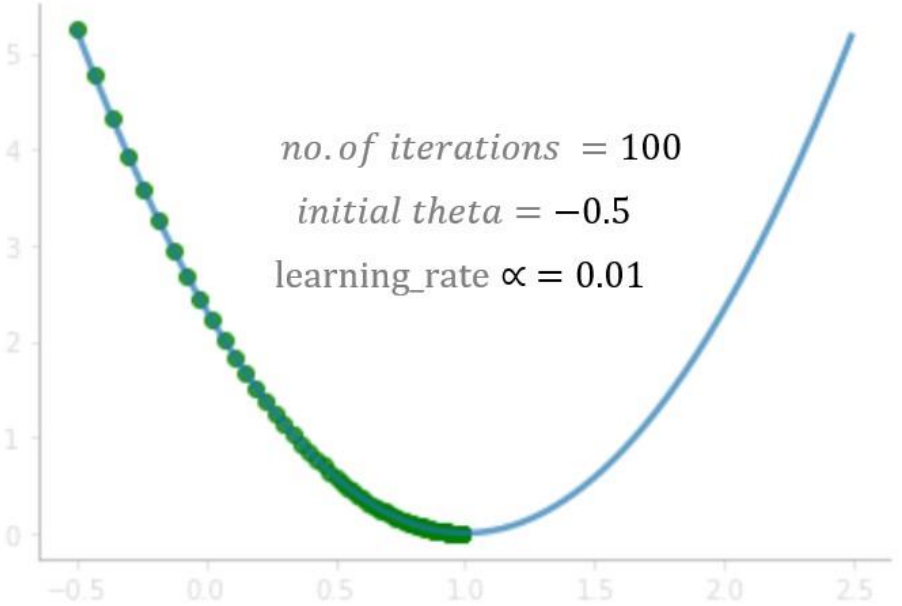


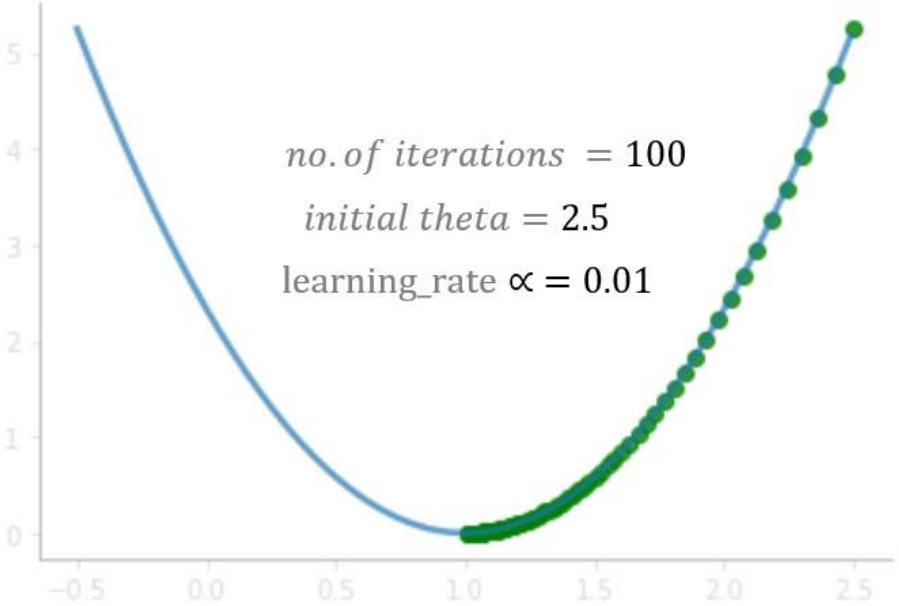


Gradient descent optimization –









Initialise w, b

Iterate over data:

$compute\ \hat{y}$

$compute\ \mathscr{L}(w, b)$

$$w_{t+1} = w_t - \eta \Delta w_t$$

$$b_{t+1} = b_t - \eta \Delta b_t$$

till satisfied

I/P	O/P
3	0.268
4	0.73
5	0.952
6	0.994
8	0.999

$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

$$Loss \mathscr{L}(w, b) = \sum_{i=1}^{5} (y_i - \hat{y}_i)^2$$

$$\mathscr{L} = \frac{1}{5} \sum_{i=1}^{i=5} (f(x_i) - y_i)^2$$

$$\frac{\partial \mathscr{L}}{\partial w} = \frac{\partial}{\partial w} [\frac{1}{5} \sum_{i=1}^{i=5} (f(x_i) - y_i)]^2$$

$$\Delta w = \frac{\partial \mathscr{L}}{\partial w} = \frac{1}{5} \sum_{i=1}^{i=5} \frac{\partial}{\partial w} (f(x_i) - y_i)^2$$

f(x) is the sigmoid function – the output

Calculation of Δw

$$\nabla w = \frac{\partial}{\partial w}\left[\frac{1}{2} * (f(x) - y)^2\right]$$

$$= \frac{1}{2} * \left[2 * (f(x) - y) * \frac{\partial}{\partial w}(f(x) - y)\right]$$

$$= (f(x) - y) * \frac{\partial}{\partial w}(f(x))$$

$$= (f(x) - y) * \frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$= (f(x) - y) * f(x) * (1 - f(x)) * x$$

$$\frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$= \frac{-1}{(1+e^{-(wx+b)})^2}\frac{\partial}{\partial w}(e^{-(wx+b)}))$$

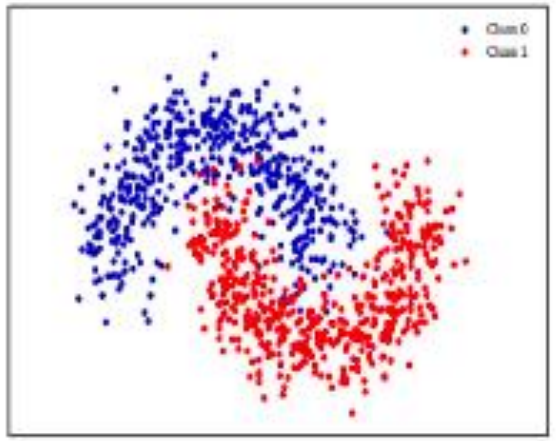
$$= \frac{-1}{(1+e^{-(wx+b)})^2} * (e^{-(wx+b)})\frac{\partial}{\partial w}(-(wx+b)))$$

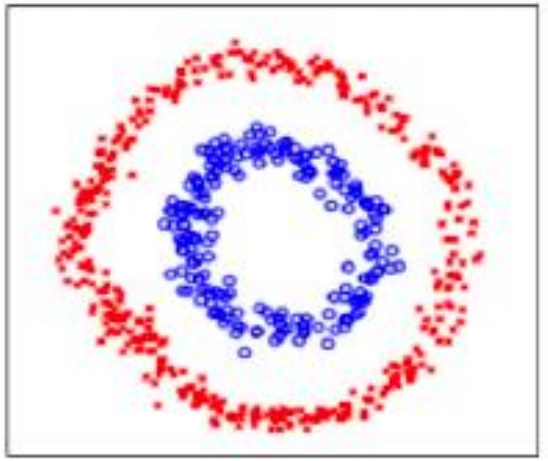
$$= \frac{-1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (-x)$$

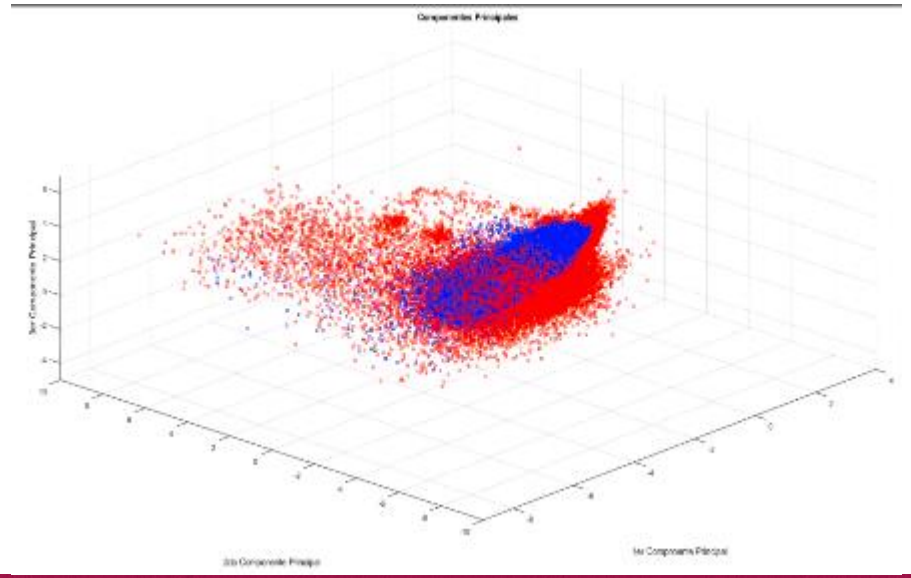
$$= \frac{1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (x)$$

$$= f(x) * (1 - f(x)) * x$$

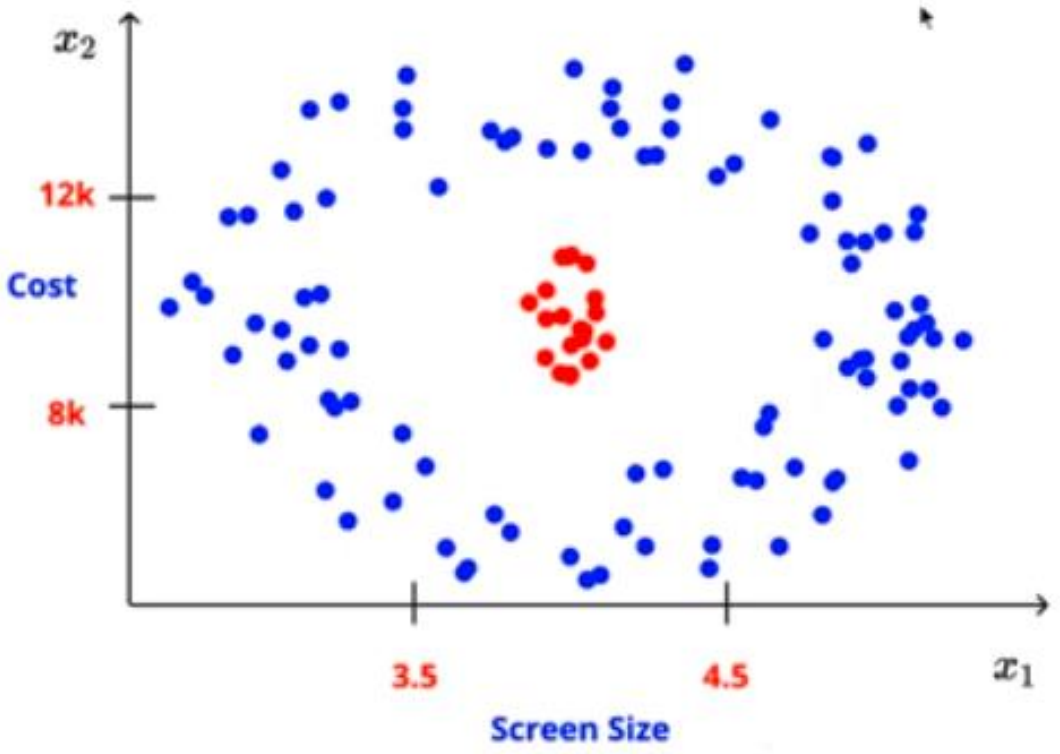
Non linearly separated data

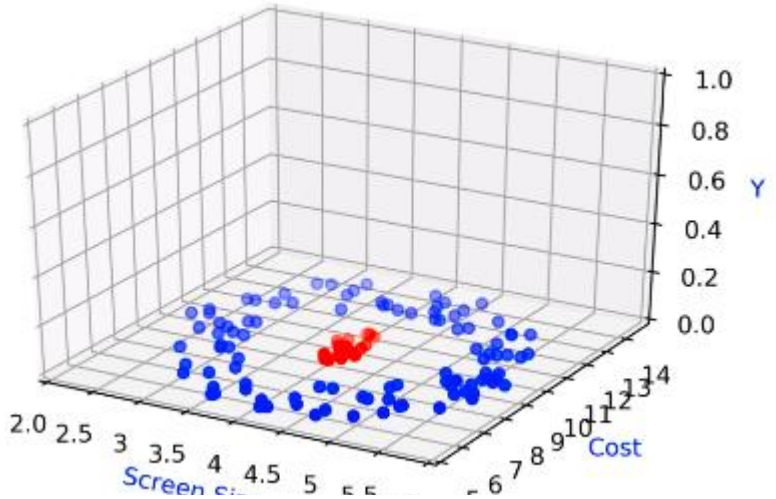


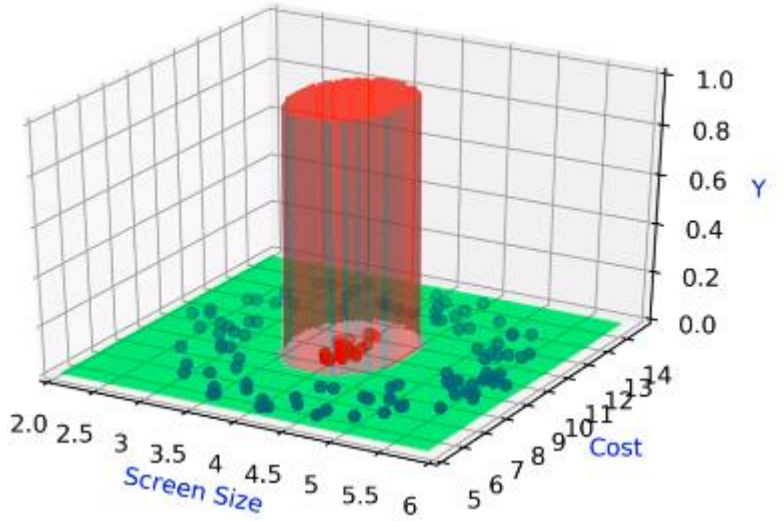




Non linearly separated data

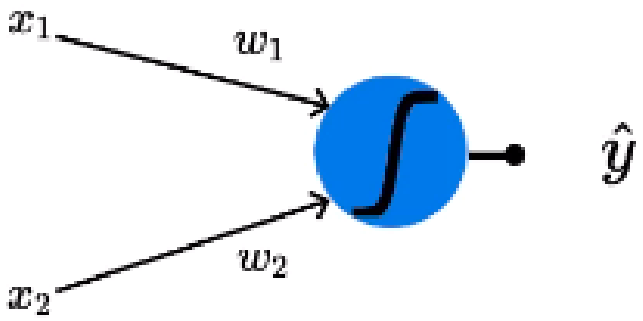


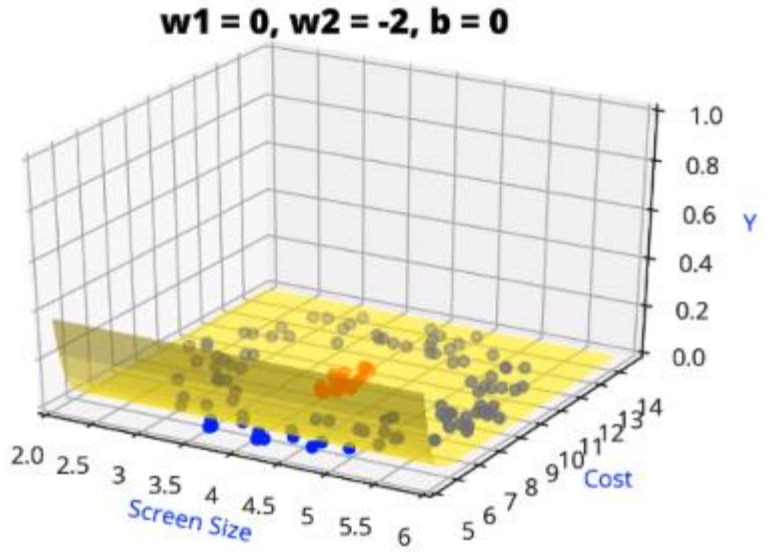




Will Sigmoid work ??

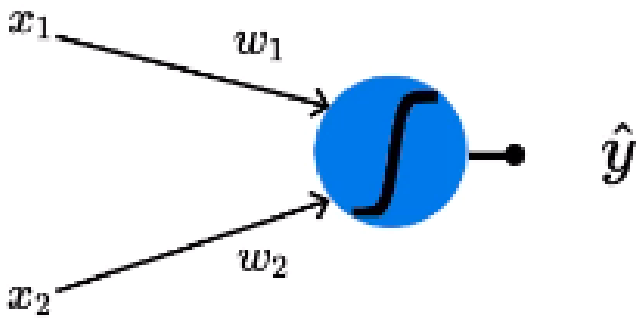
$$\hat{y} = \frac{1}{1+e^{-(w_1 * x_1 + w_2 * x_2 + b)}}$$

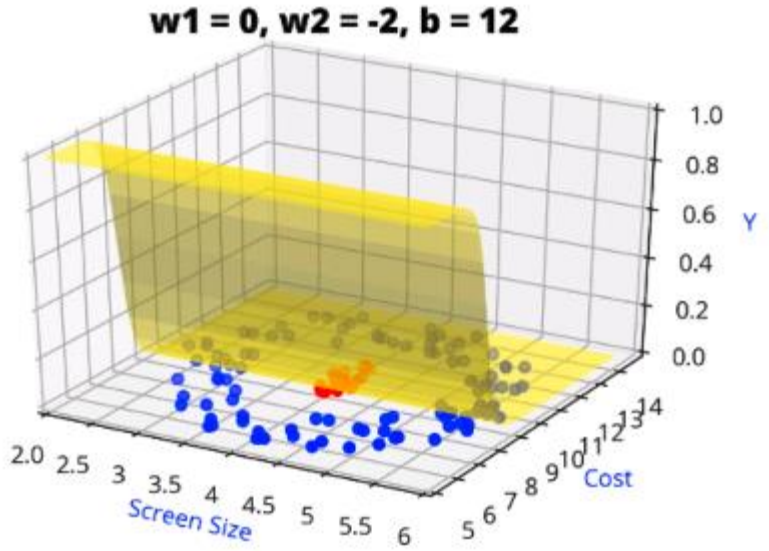




Will Sigmoid work ??

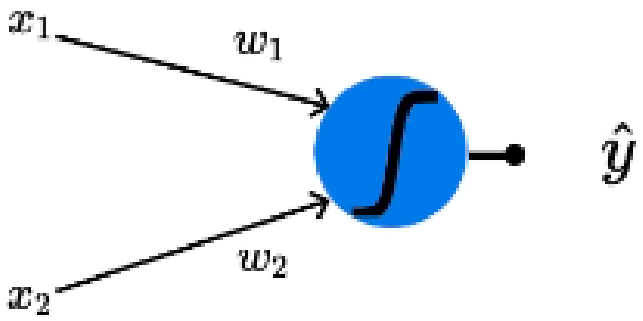
$$\hat{y} = \frac{1}{1+e^{-(w_1 * x_1 + w_2 * x_2 + b)}}$$

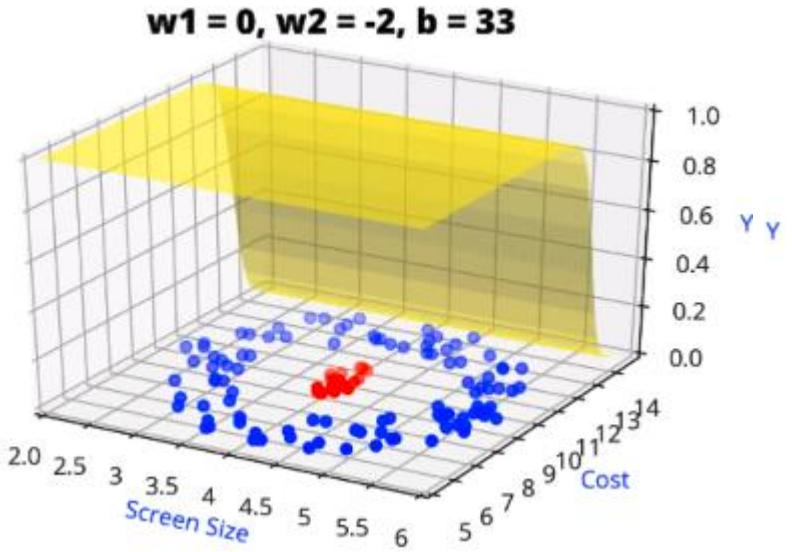




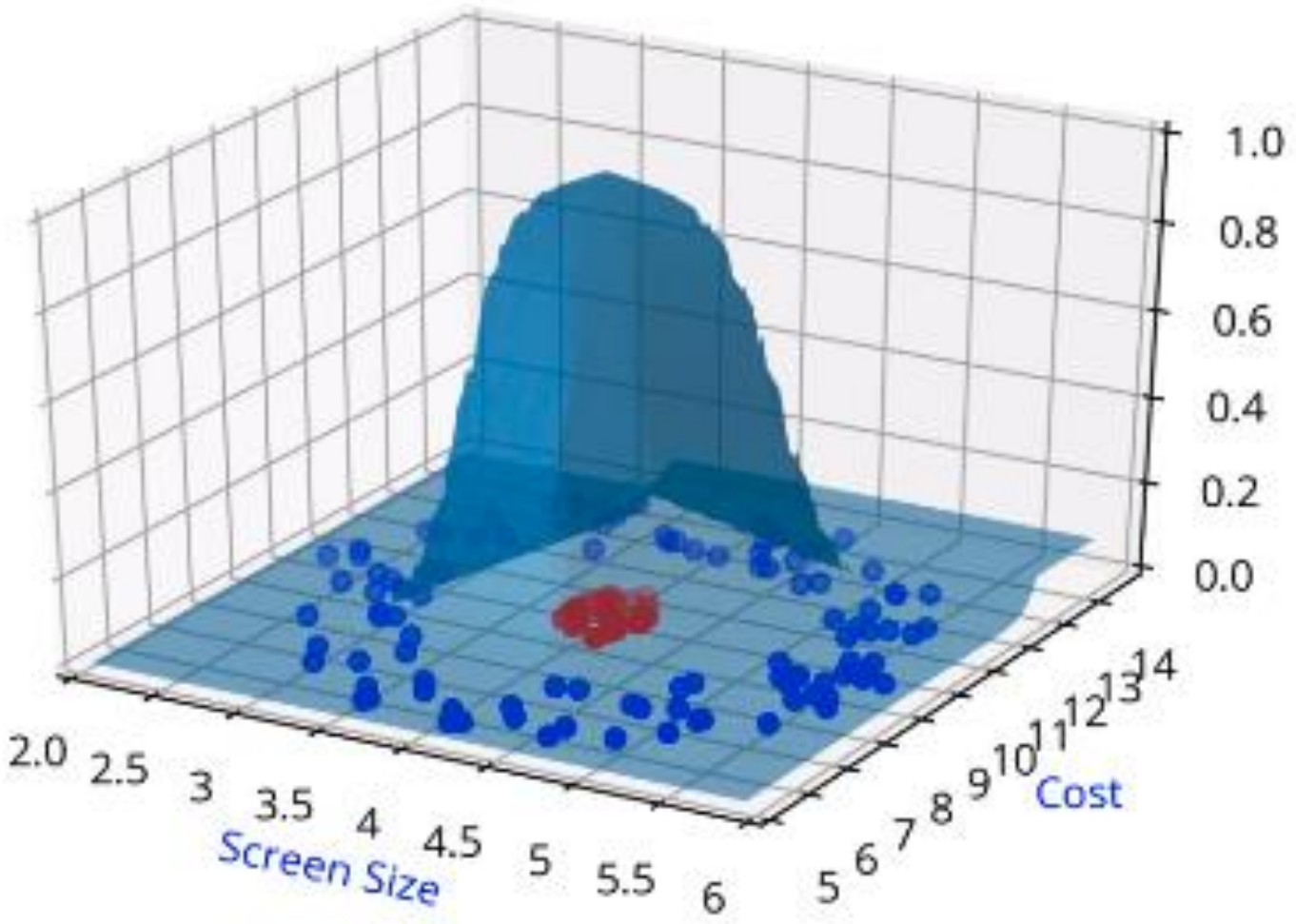
Will Sigmoid work ??

$$\hat{y} = \frac{1}{1+e^{-(w_1 * x_1 + w_2 * x_2 + b)}}$$

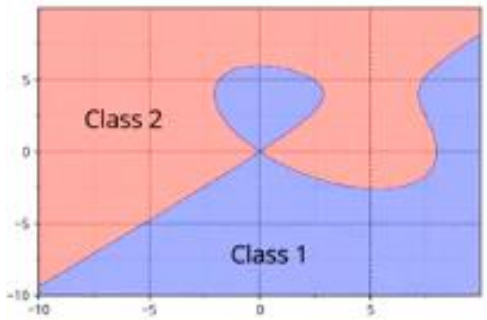


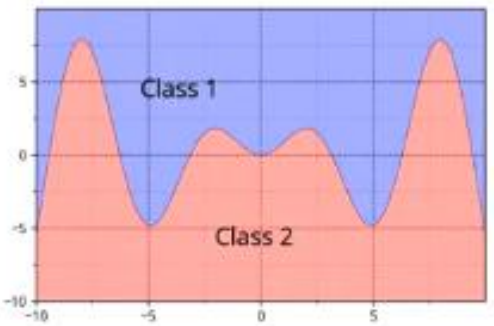


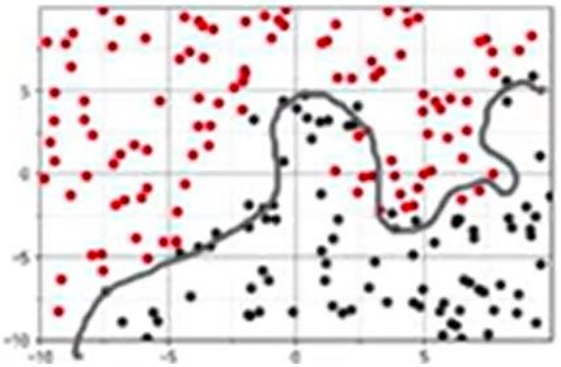
A complex function like this works



Need of DEEP ML models- complex real world functions







In real life, we deal with complex functions where the relationship between input and output might be complex, unlike the simple sigmoid neuron or perceptron.

It's hard to come up with such functions.
We need a simple approach!

Namah Shivaya