

Introduction to Machine Learning (CS601)

Credits: 4 (Lect: 3 & Lab: 2)

By:

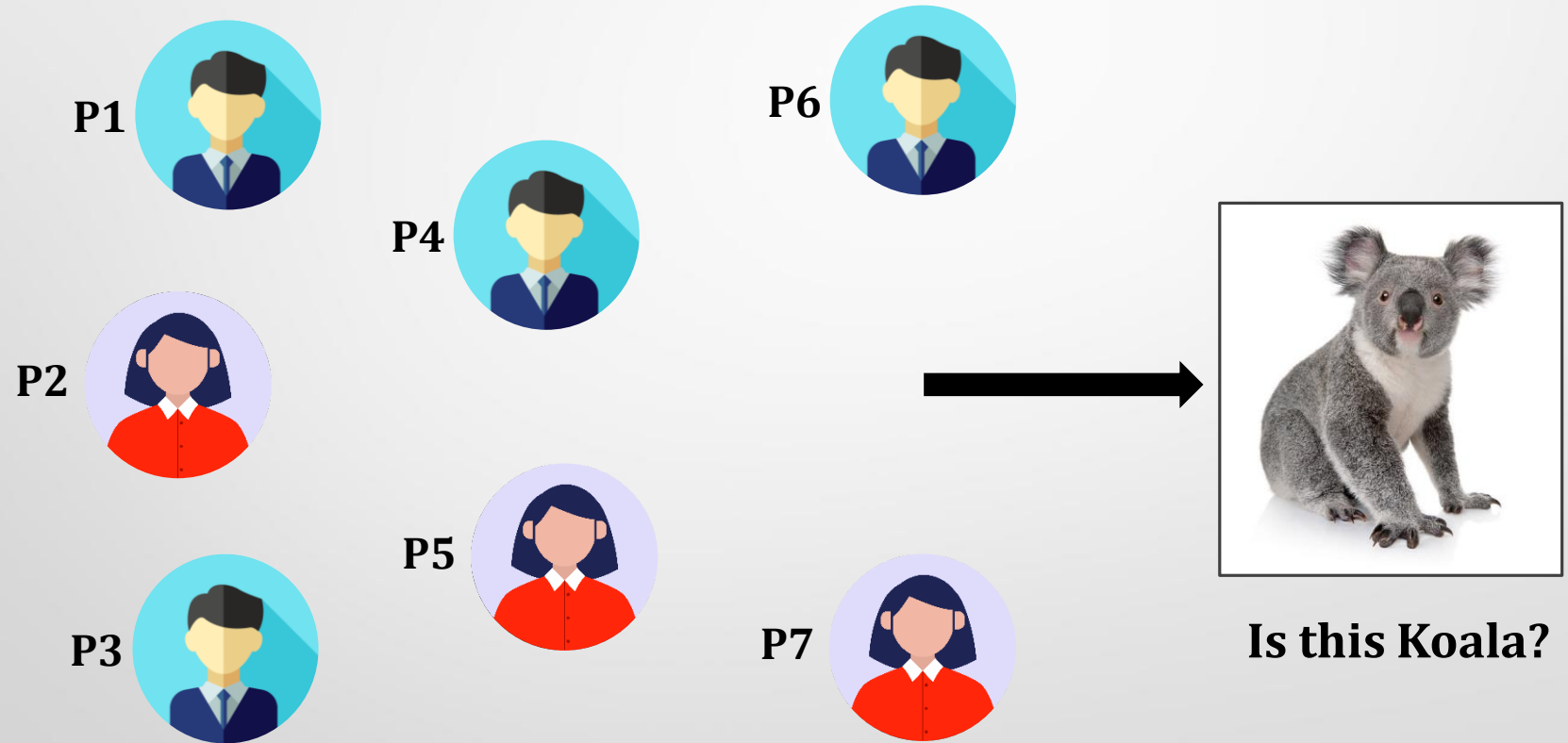
Rajesh K. Ahir

Department of Computer Science & Engineering
Indian Institute of Information Technology
Surat

विद्या विनयेन शोभते

Introduction of ANN

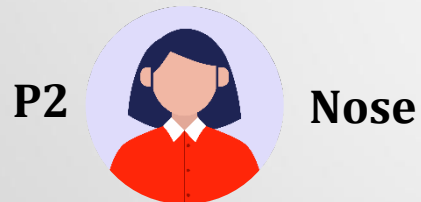
- Understanding ANN without any mathematics



Job: To train these people to tell whether this animal is Koala or not?

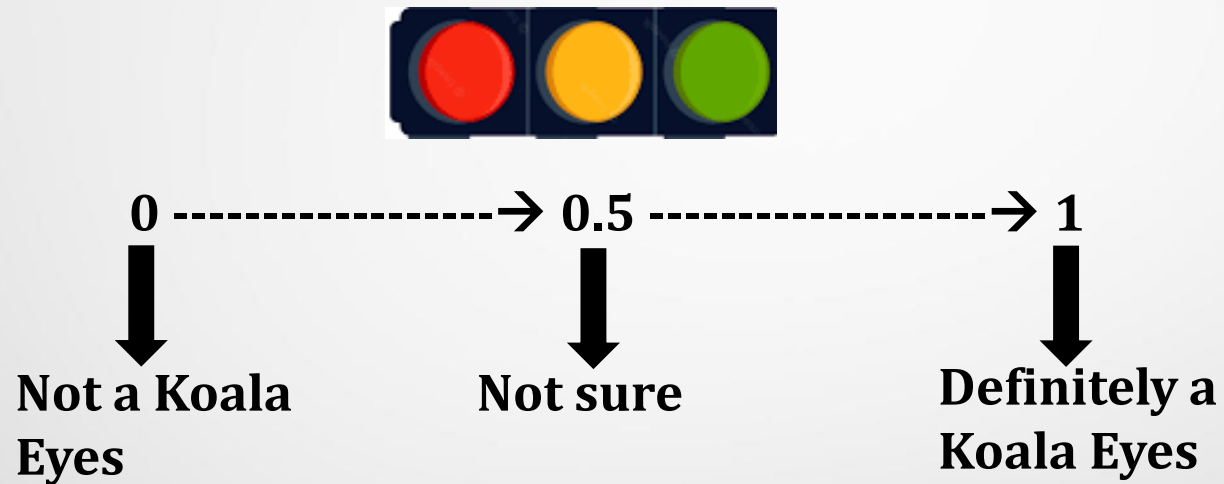
Introduction of ANN

- Give individual responsibility to each of these



Introduction of ANN

- How they responds (give decision)?



Introduction of ANN

- Now these people are trained

P1  **Expert in
detecting
Eyes**

P2  **Expert in
detection
Nose**

P3  **Expert in
Detecting
Ears**



P4  **Expert in
Detecting
Legs**

P5  **Expert in
Detecting
Tail**

Introduction of ANN

- Testing



Introduction of ANN

- Testing (give some score)

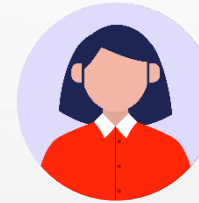


P1

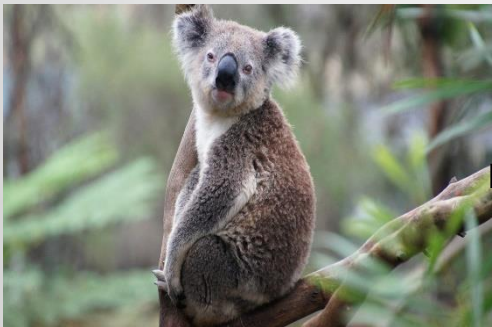


Eyes = 0.1

P2



Nose = 0.02



P1



Eyes = 0.7

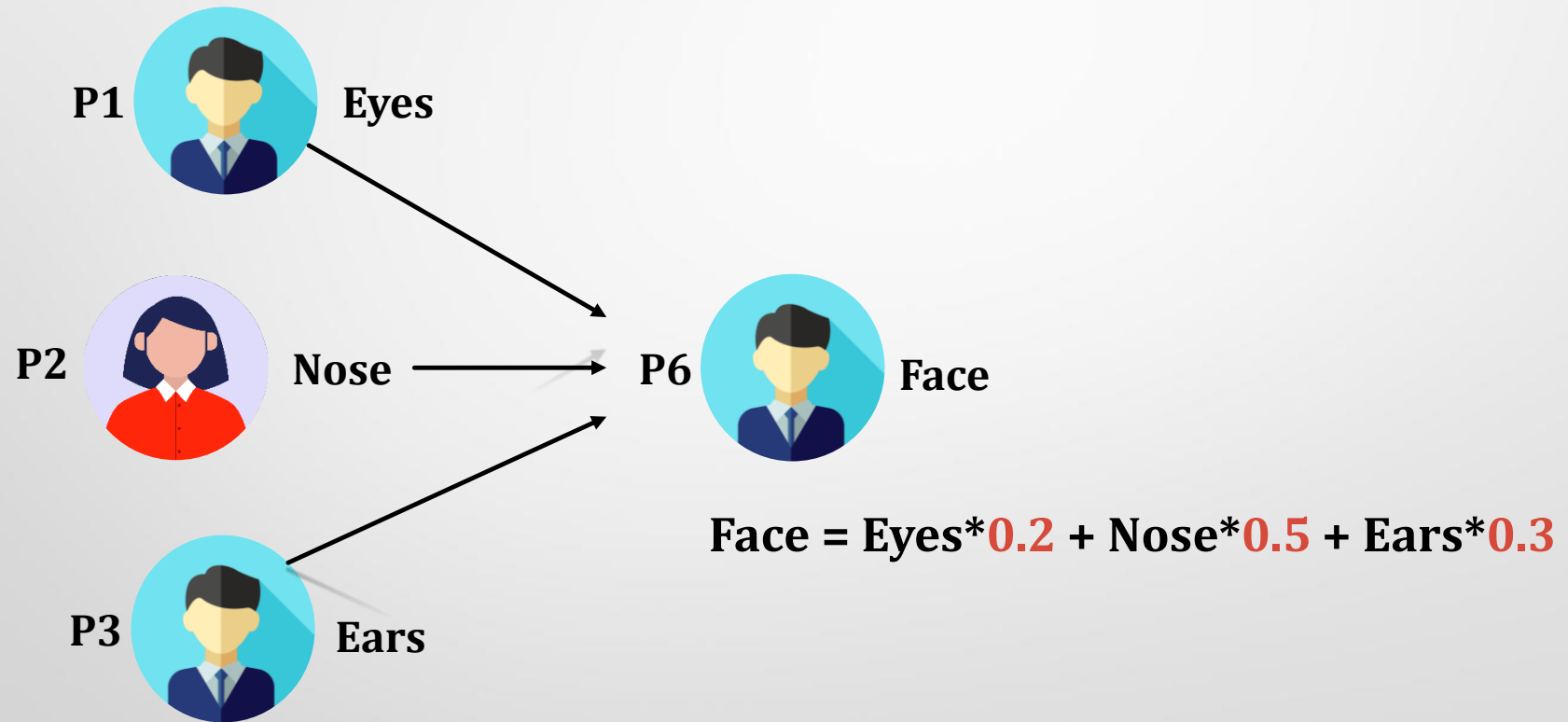
P2



Nose = 0.8

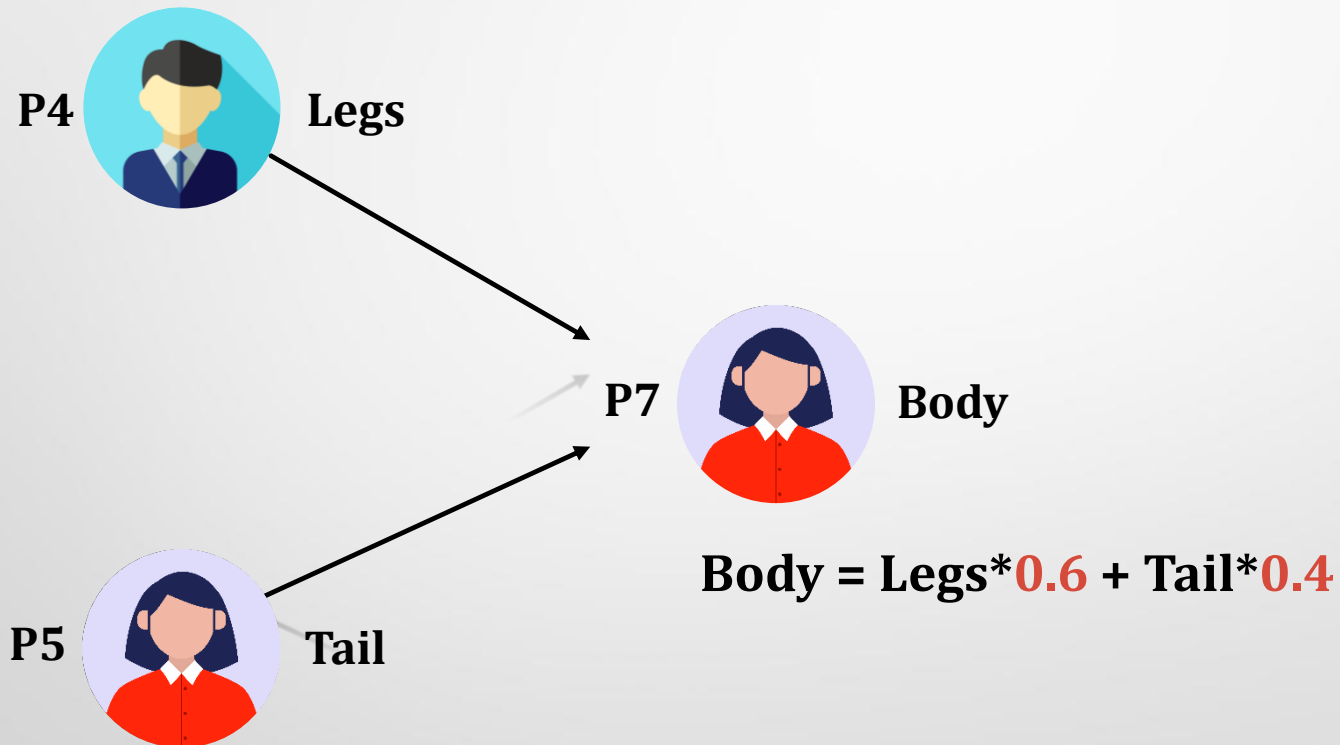
Introduction of ANN

- Face detection



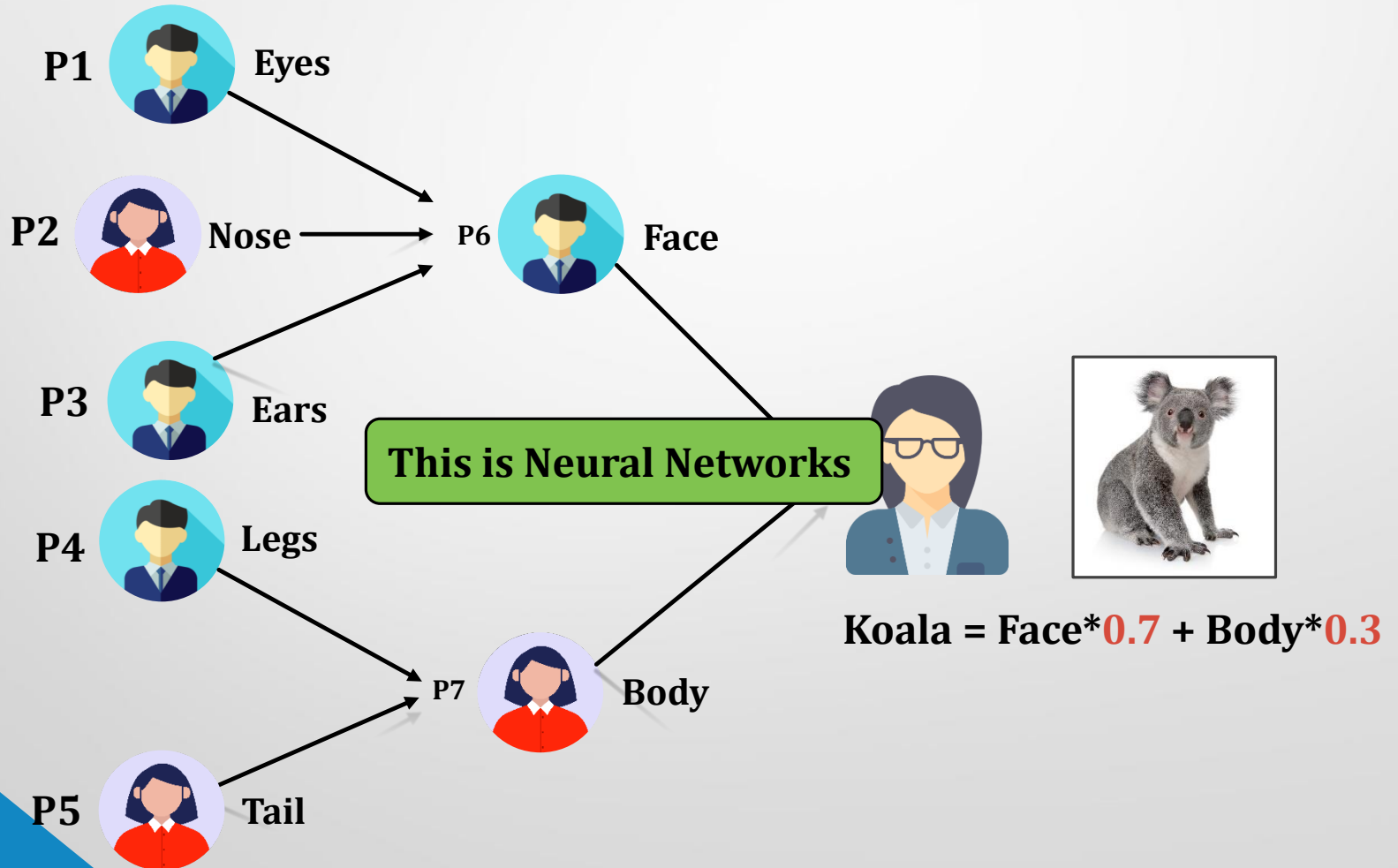
Introduction of ANN

- Body detection



Introduction of ANN

- Koala detection



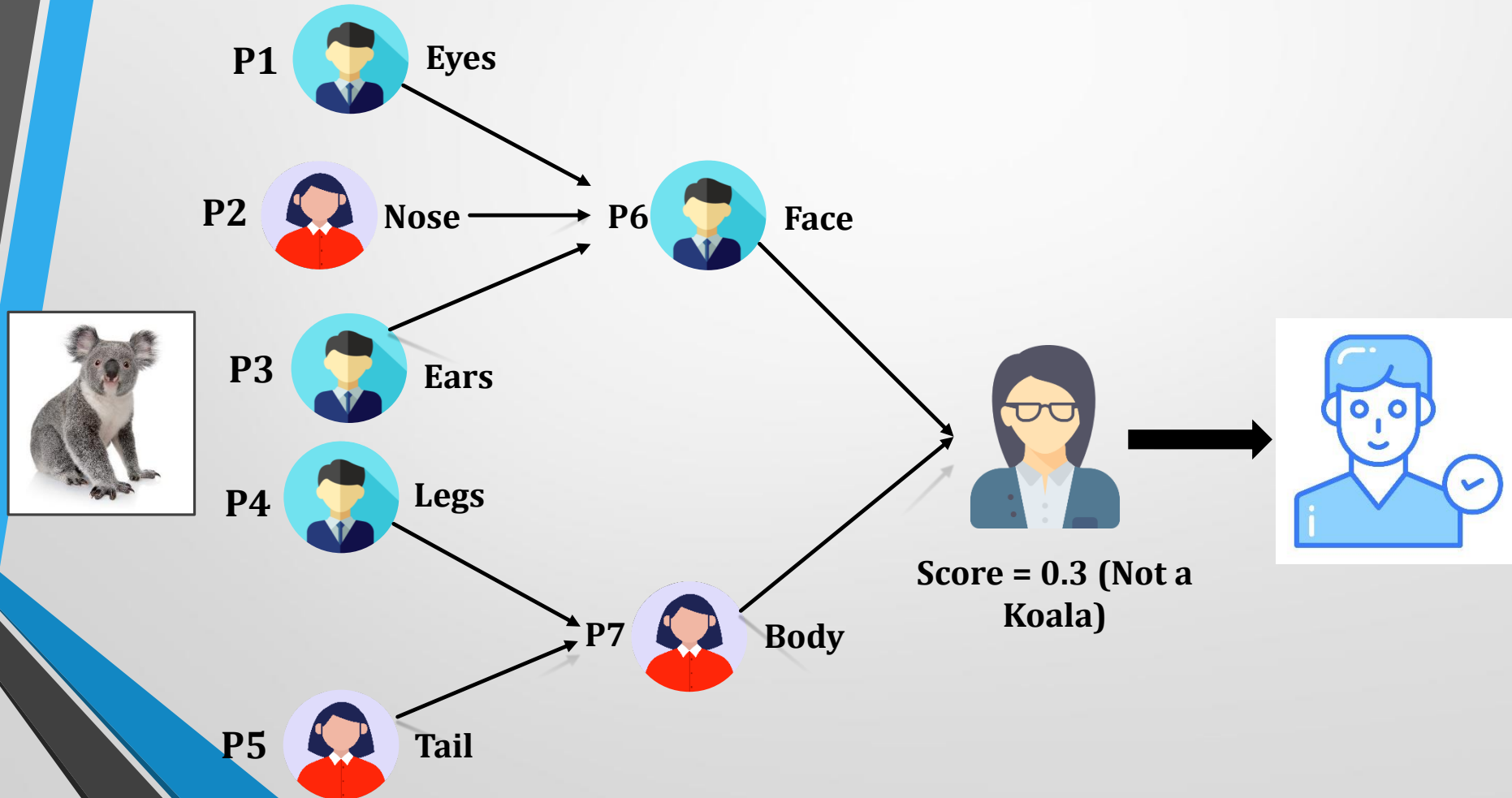
Introduction of ANN

- Each individual person are individual neuron
 - Working on specific sub-tasks
 - They pass results of sub-task to the next level
 - Output
- It's a trained neural network

How do you train the neural network?

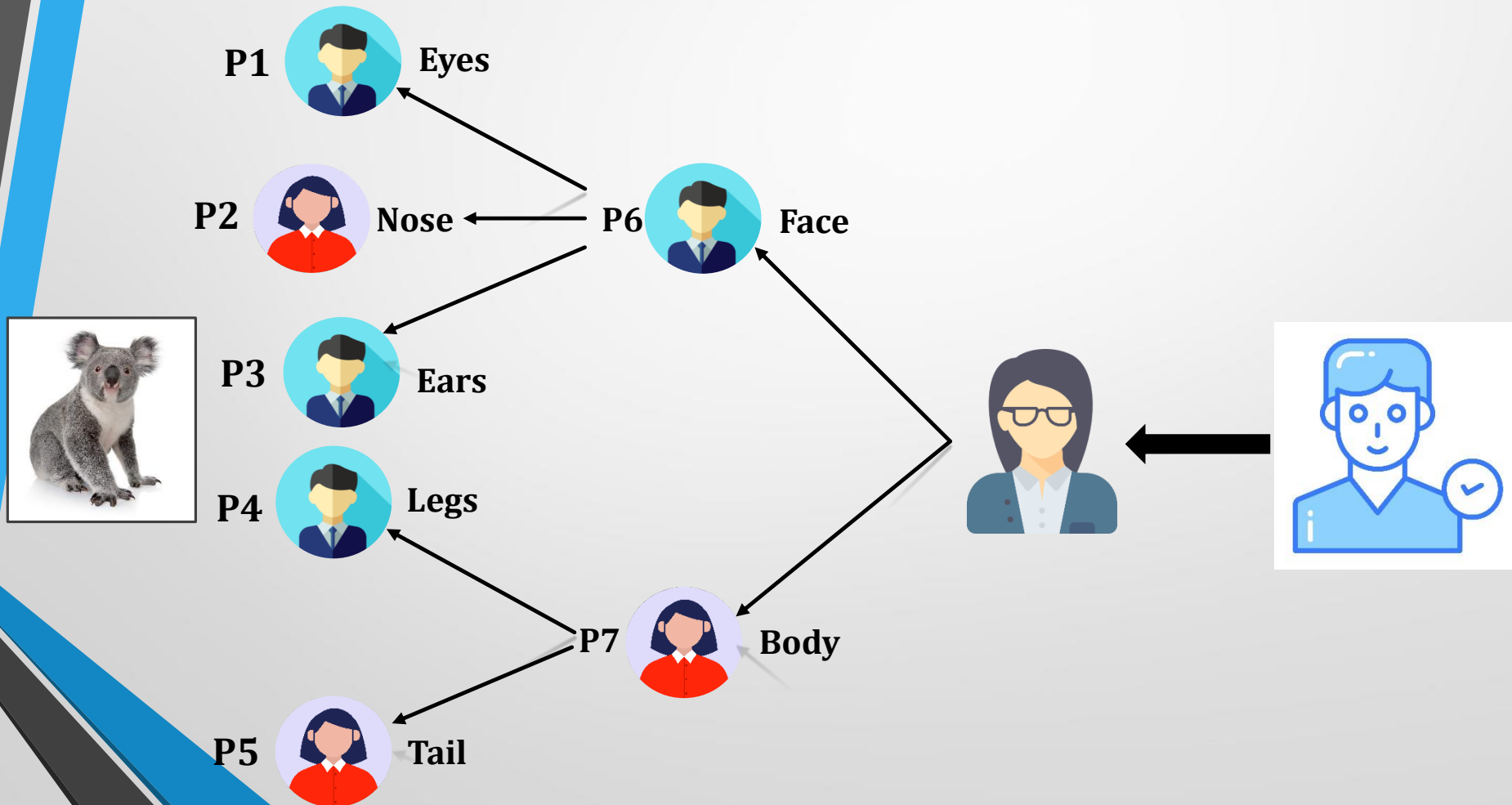
Koala detection: Training

- Make a random guess



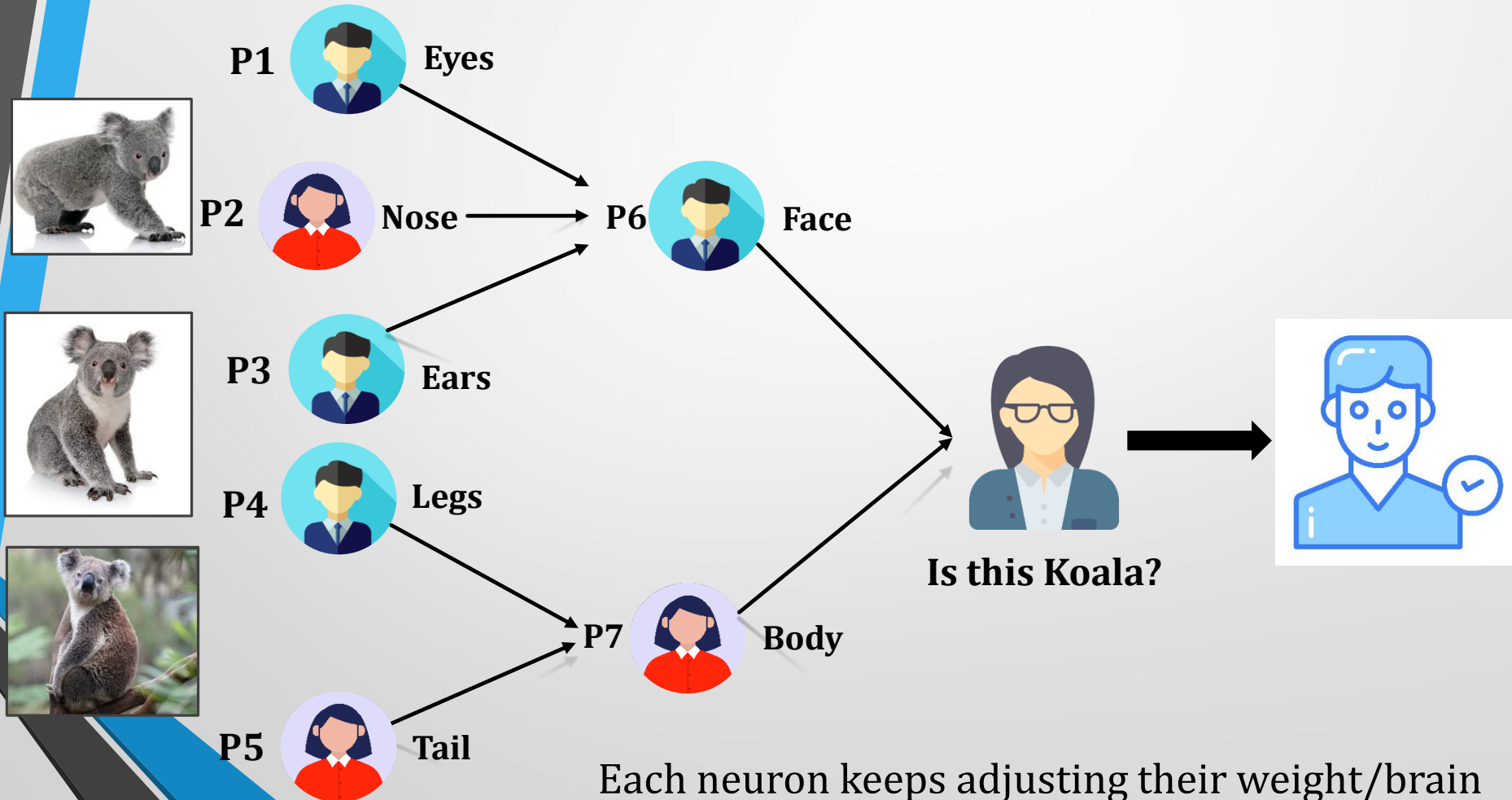
Training

- Backward Error Propagation



Training

- Give various images of Koala to train the NN



Artificial Neural Networks

- Computational models **inspired by the human brain**:
 - Massively parallel, distributed system, made up of simple processing units (neurons)
 - Synaptic connection strengths among neurons are used to store the acquired knowledge.
 - Knowledge is acquired by the network from its environment through a learning process

ANN > Properties

- Learning from examples
 - labeled or unlabeled
- Adaptivity
 - changing the connection strengths to learn things
- Non-linearity
 - the non-linear activation functions are essential
- Fault tolerance
 - if one of the neurons or connections is damaged, the whole network still works quite well

ANN: Basic terminologies

- Neuron
- Weights
- Bias
- Summation Function
- Activation function
- Learning rate

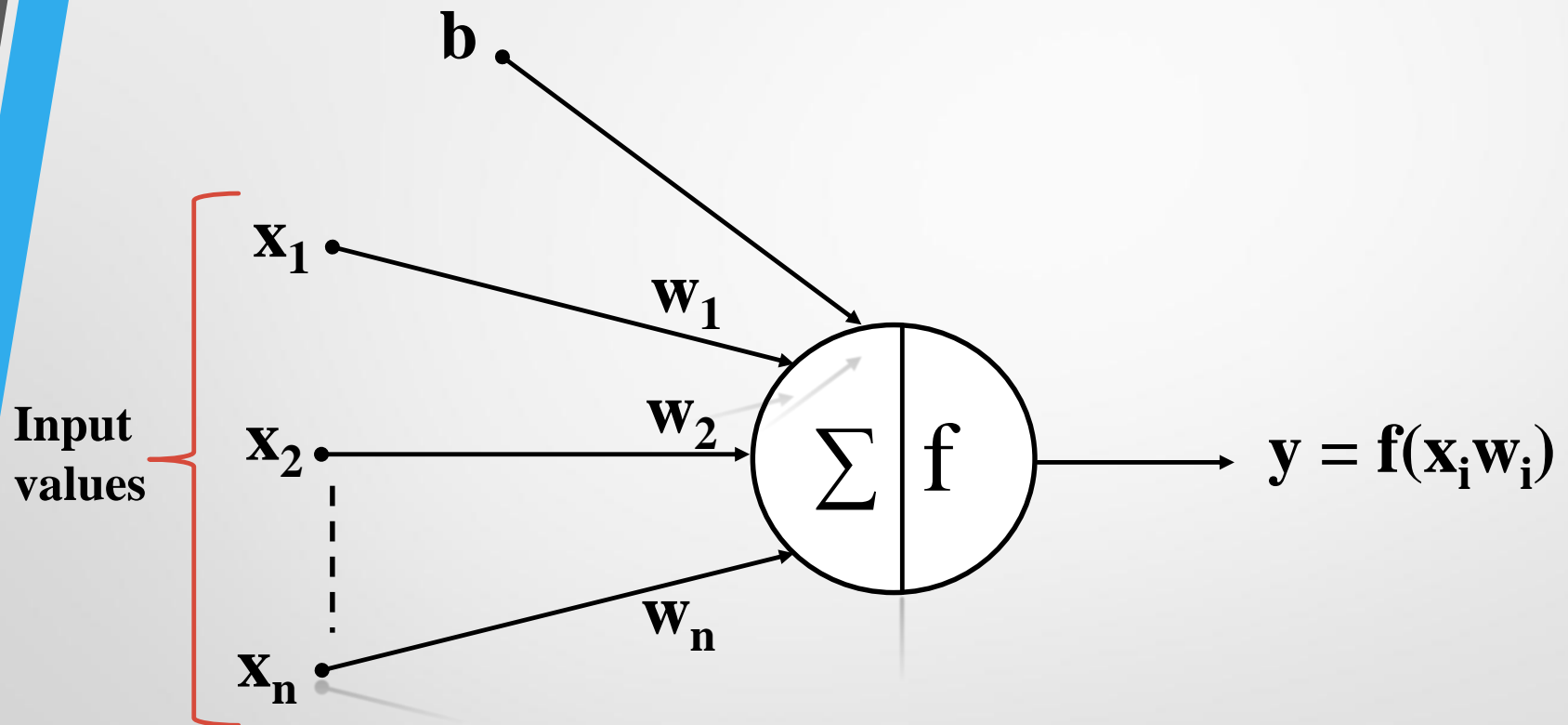
ANN: Basic terminologies

- **Neuron:** It is a basic information processing unit of a NN. It consists of:
 - **Weights:** Weights are the real values that are attached with each input/feature and they convey the importance of that corresponding feature in predicting the final output.
 - **Bias:** Bias is used for shifting the activation function towards left or right, you can compare this to y-intercept in the line equation.

ANN: Basic terminologies

- **Neuron:** It is a basic information processing unit of a NN. It consists of:
 - **Summation Function:** The work of the summation function is to bind the weights and inputs together and calculate their sum.
 - **Activation Function:** It is used to introduce non-linearity in the model. In other words, it is used for limiting the amplitude of the neuron output.

Neuron



$$\Sigma = x_1 w_1 + x_2 w_2 + \dots + x_n w_n$$

Single Neuron Example

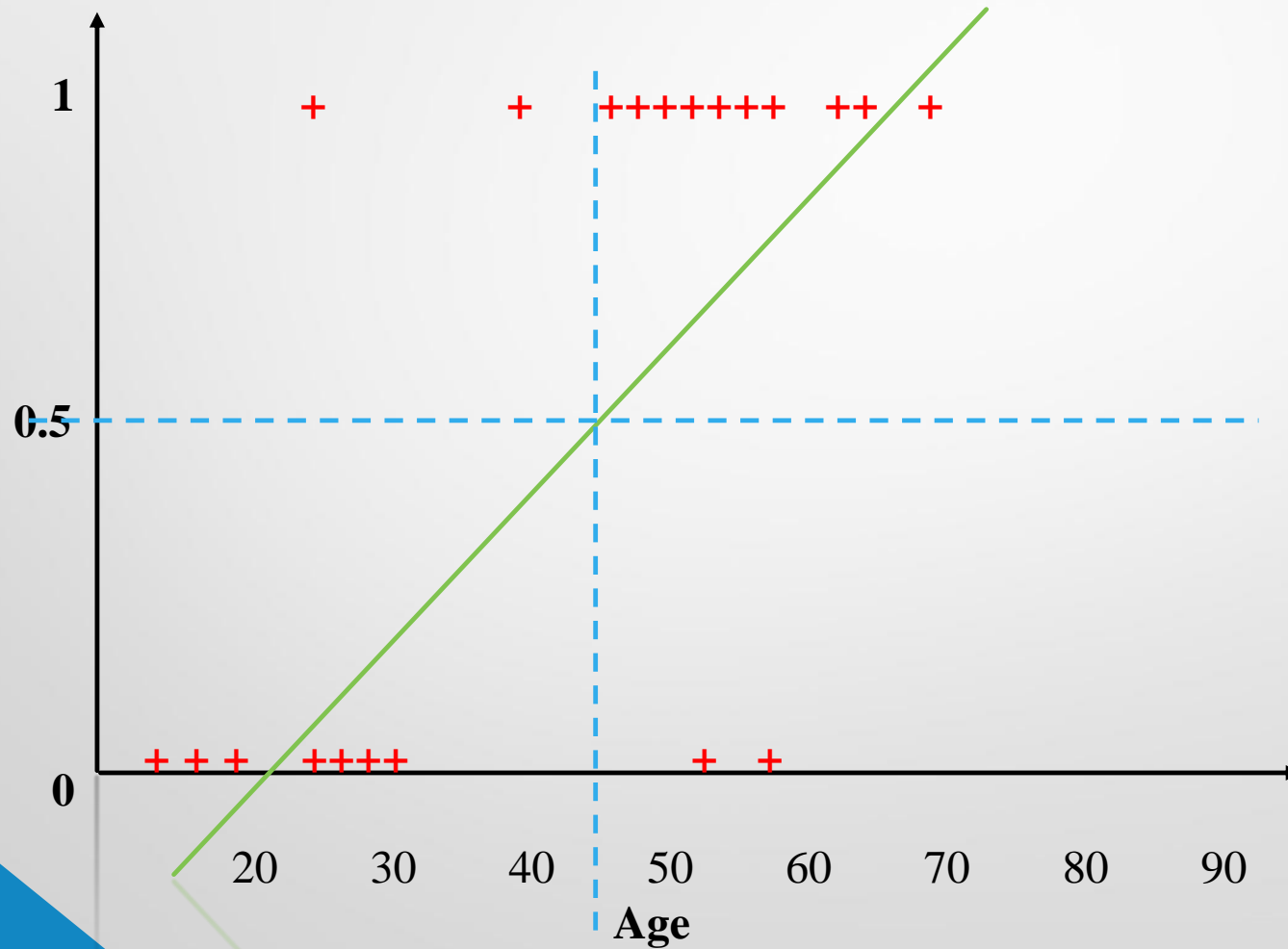
Age	Have insurance
22	0
25	0
47	1
52	0
46	1
56	1
55	0
60	1
62	1
61	1
18	0
28	0

Binary Classification Problem

Given an age of a person, come up with a model/function that can predict/classify if person will buy insurance or not.

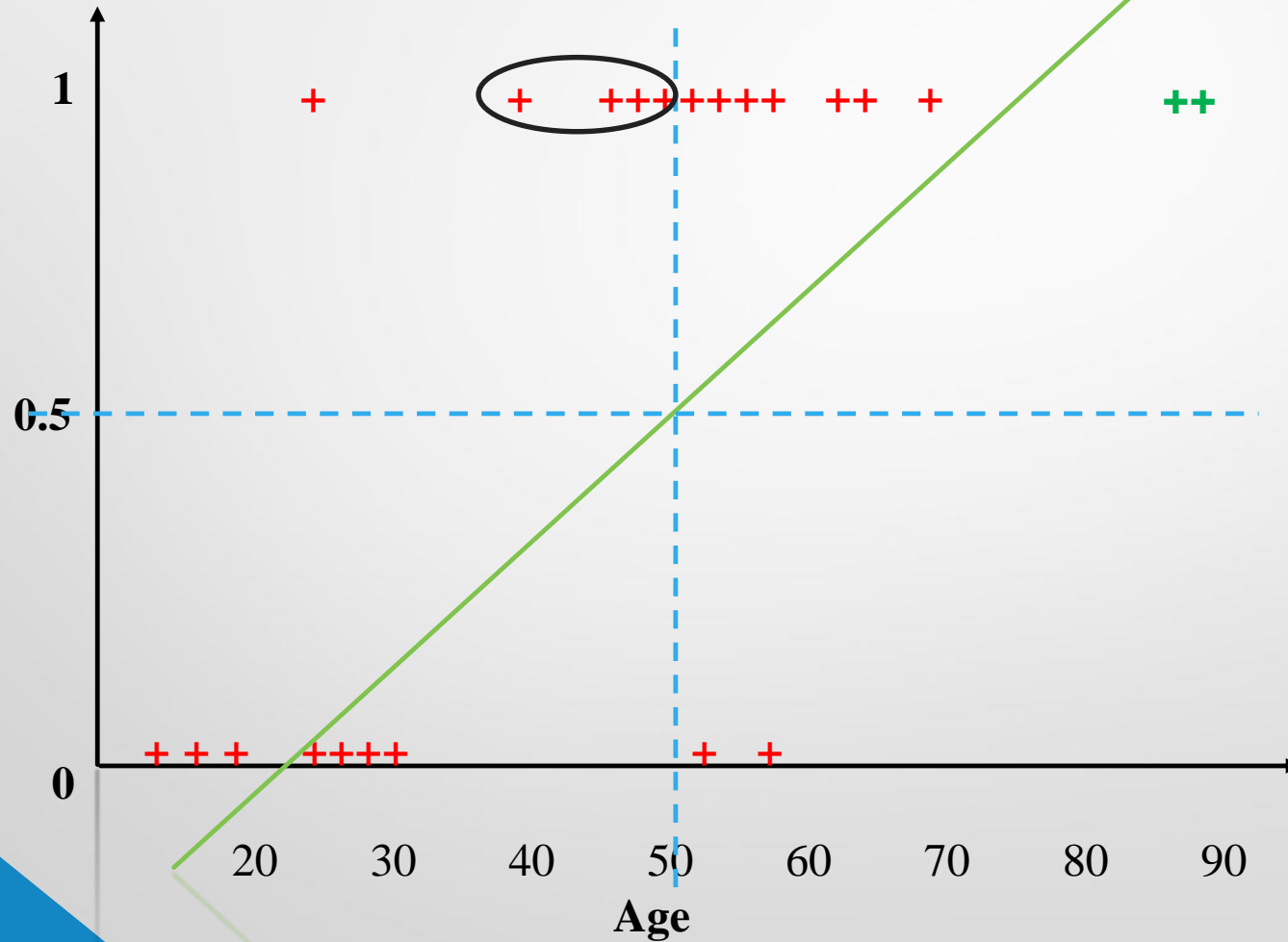
Single Neuron Example

- Plot the data



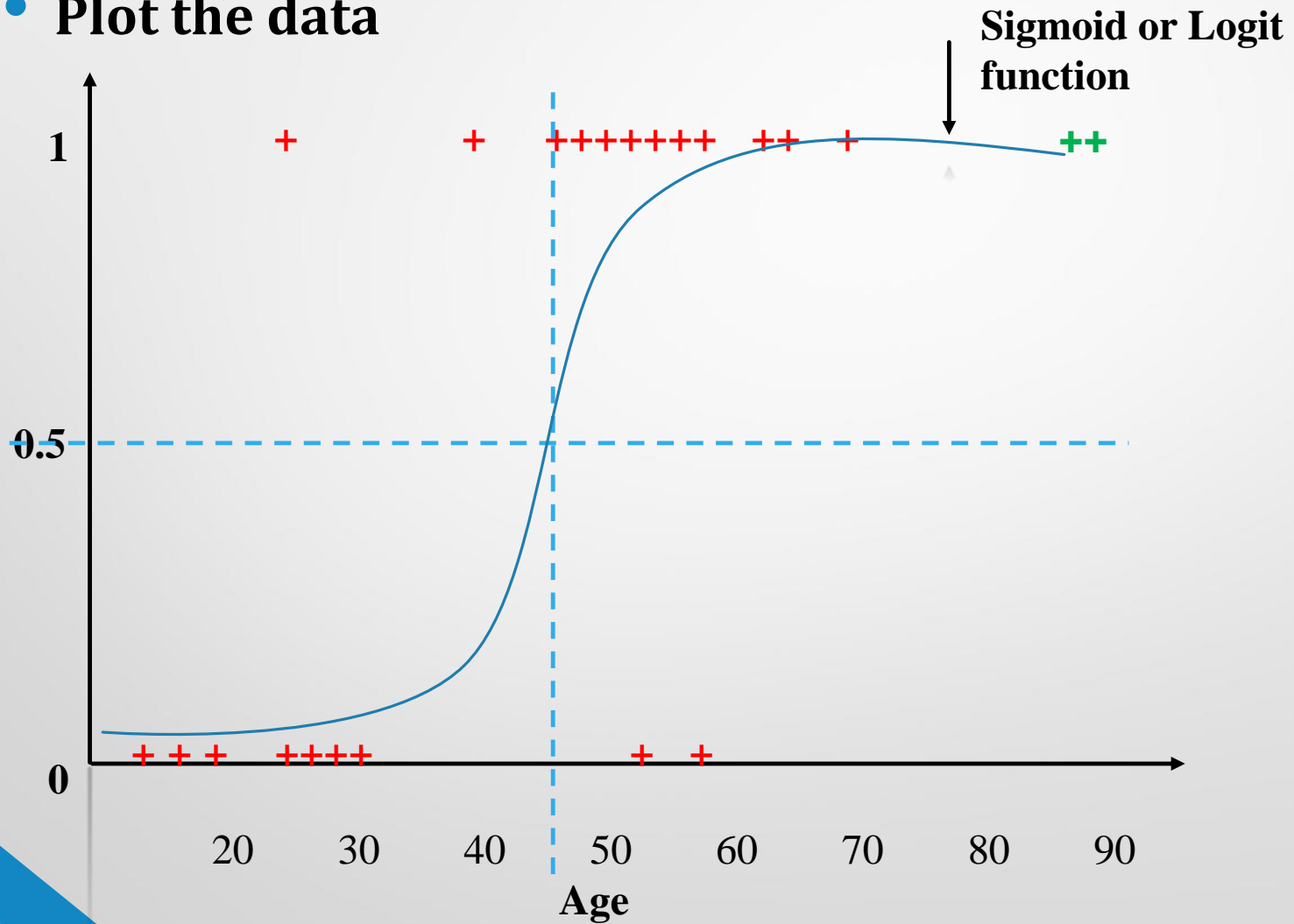
Single Neuron Example

- Plot the data



Single Neuron Example

- Plot the data



Single Neuron Example

$$\textit{Sigmoid}(z) = \frac{1}{1 + e^{-z}} \text{ where } e = \text{Euler's number} \sim 2.71828$$

$$\text{Sigmoid}(200) = 1 / (1 + 2.71^{-200}) = \text{close to } 1$$

$$\text{Sigmoid}(-200) = 1 / (1 + 2.71^{200}) = \text{close to } 0$$



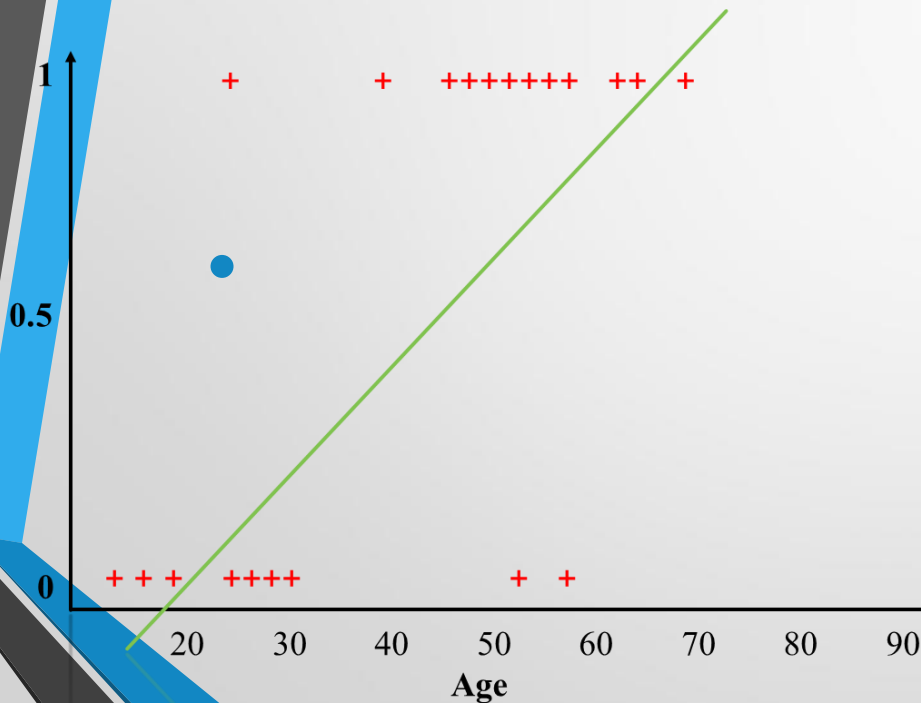
So, this function converts input into range [0, 1]

Single Neuron Example

Step 1:

$y = mx + b$

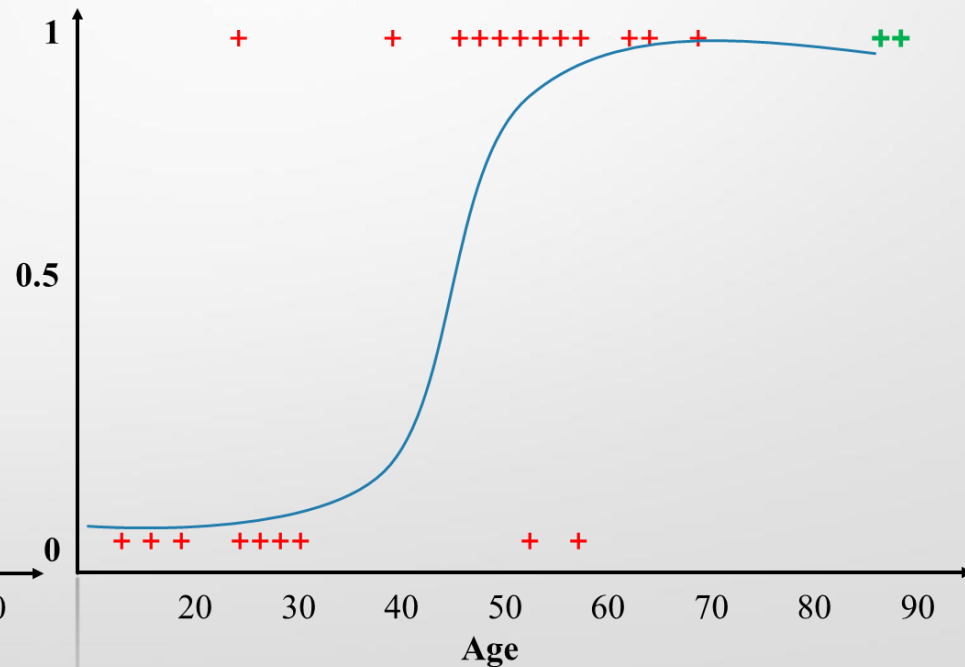
Age



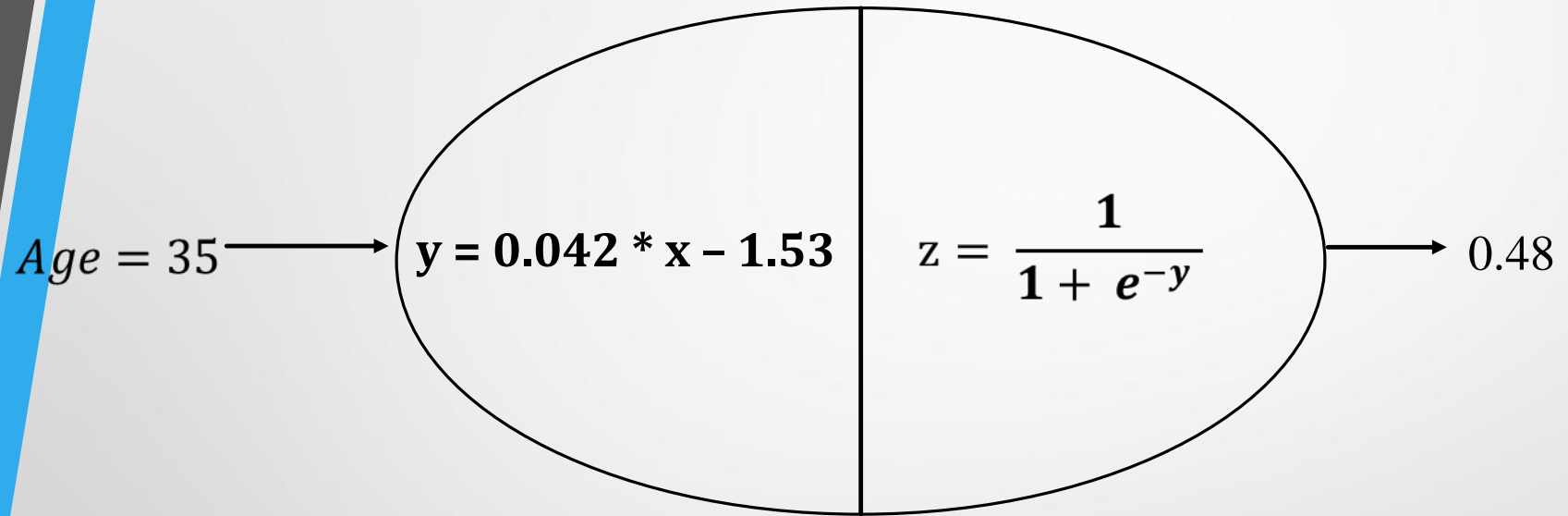
Step 2:

$z = \frac{1}{1 + e^{-y}}$

If person will buy insurance



Single Neuron Example



$z \geq 0.5 \Rightarrow$ Person will buy insurance

$z < 0.5 \Rightarrow$ Person will not buy insurance

Single Neuron Example

- For a single input (independent) variable:

$$y = 0.042 * x - 1.53$$

- For a Multiple inputs variable:

$$y = 0.042 * x_1 + 0.08 * x_2 + 0.3 * x_3 - 1.53$$

Age

Income

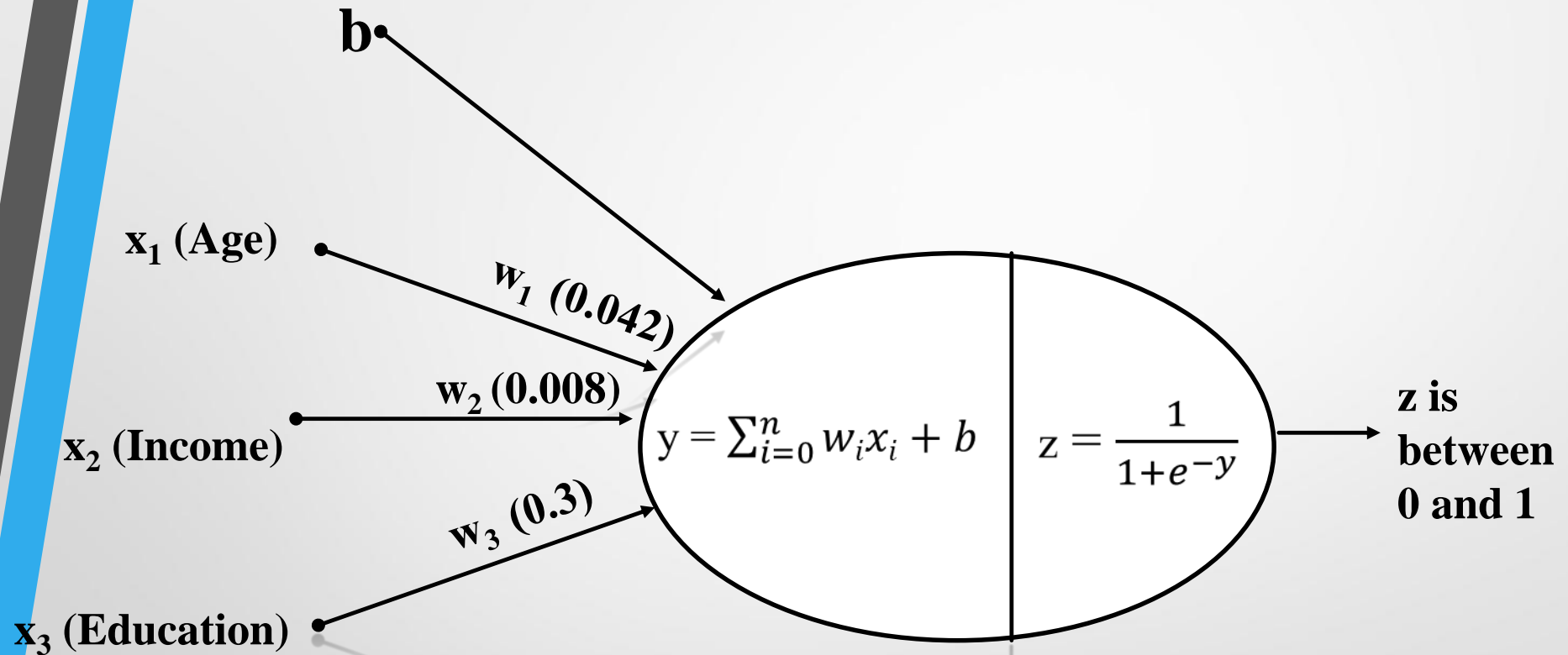
Education

-

$$y = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b$$

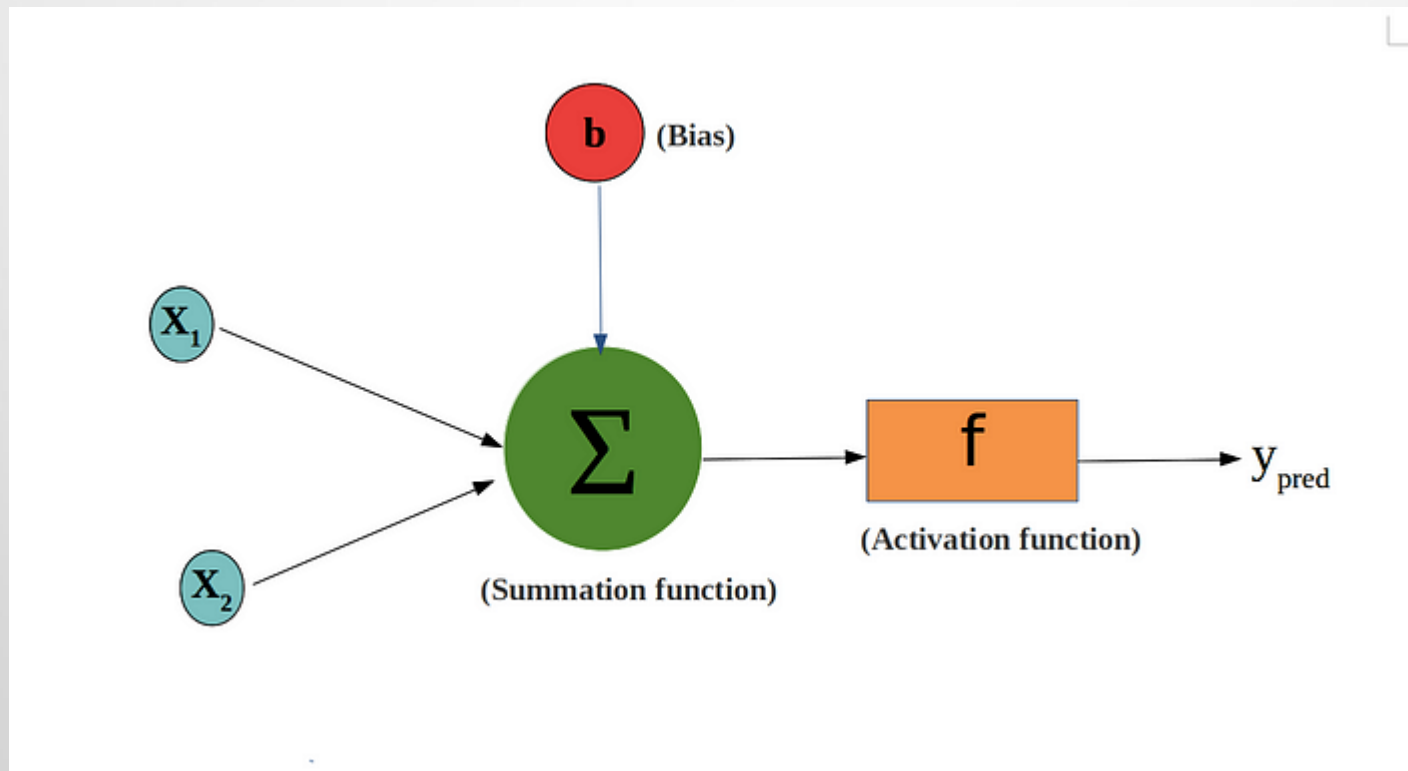
$$y = \sum_{i=0}^n w_i x_i + b$$

Multiple inputs



The Role of Weights and Bias in a NN?

- Let us consider there are only two inputs/features in a dataset (input vector $\mathbf{X} \in [\mathbf{x}_1, \mathbf{x}_2]$), and our task is to perform **binary classification**.



The Role of Weights and Bias in a NN?

- The summation function $g(x)$ sums up all the inputs and adds bias to it.

$$\begin{aligned} g(x) &= \sum_{i=1}^{i=2} x_i + b \\ &= x_1 + x_2 + b \end{aligned}$$

- So after summation function does its job we get a integer value as output, but since this is a binary classification problem we need to convert this integer value to a binary output, to do this we take help of a activation function that maps a integer value to a binary output.

$$\begin{aligned} y_{pred} &= 1 \text{ if } g(x) \geq 0 \\ &= 0 \text{ else} \end{aligned}$$

The Role of Weights and Bias in a NN?

- If we compare the activation function expression with the equation of a line:

$$y = mx + c \quad (1)$$

$$x_2 = -x_1 + b \quad (2)$$

- By carefully observing the above 2 expressions, we can infer that the **slope(m)** of the equation $x_2 = -x_1 + b$ is fixed that is **-1**, and it will not change in any case.
- Now you should have understood the problem, for any given dataset if there are no weights involved then the slope of the line that classifies a data point never changes and we are not able to draw a scalable line that separates two classes.

The Role of Weights and Bias in a NN?

- Example:

	x1	x2	y
0	-9	-5	0
1	1	6	1
2	-5	0	0
3	-3	-7	0
4	5	-8	1
5	-7	11	0
6	2	3	1
7	-2	10	0
8	7	0	1
9	-7	9	0
10	-1	6	0

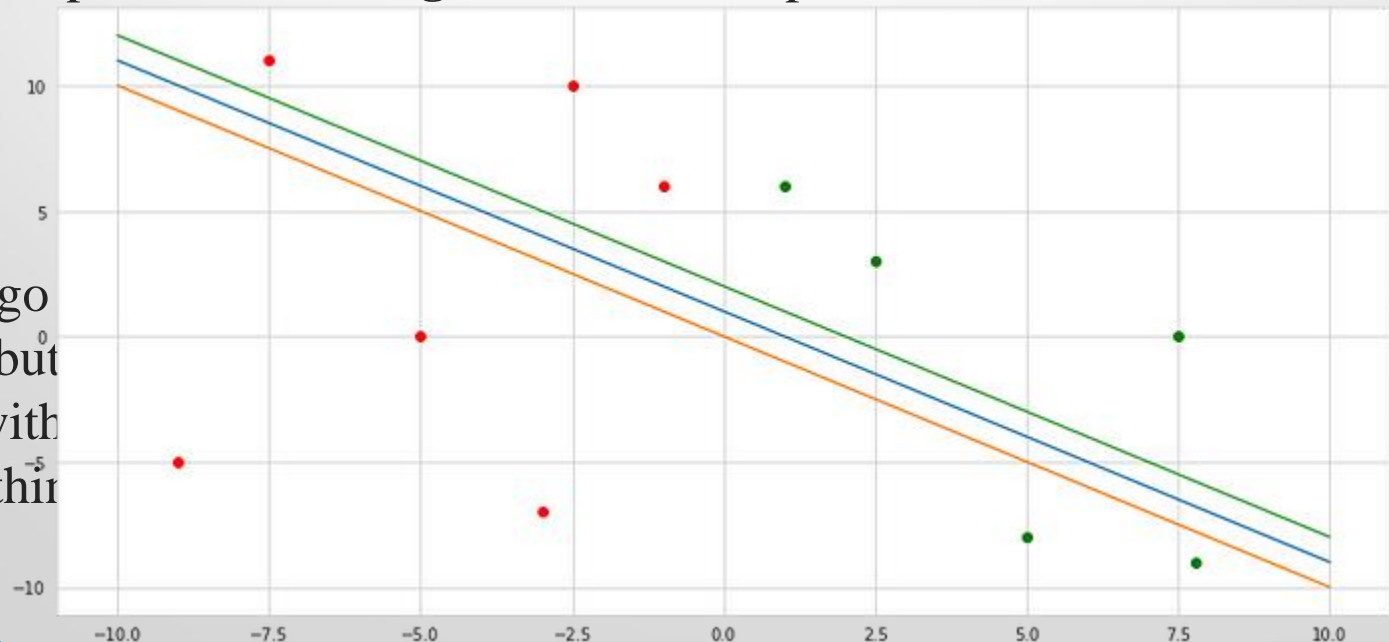
$$g(x) = \sum_{i=1}^{i=2} x_i + b$$
$$= x_1 + x_2 + b$$

- Next step is to map the integer output to a binary value using activation function that is $(x_1 + x_2 + b \geq 0)$ for the output to be 1, that can be viewed as $(x_2 \geq -x_1 - b)$ and the edge condition is **$x_2 = -x_1 - b$**
- Now if we try to fit the line equation ($x_2 = -x_1 + b$) for the different values of x_1 and b we will get this plot...

The Role of Weights and Bias in a NN?

- If we consider the value of b is 0, and for any value of x_1 the line equation looks like $(x_2 = -x_1 + 0)$, it is represented as the **orange** line in the plot.
- If we consider the value of b is 1, the line equation looks like $(x_2 = -x_1 + 1)$, it is represented as the **blue** line in the plot.
- If we consider the value of b is 2, the line equation looks like $(x_2 = -x_1 + 2)$, it is represented as the **green** line in the plot.

- If we go
lines, but
line, with
some thir



The Role of Weights and Bias in a NN?

- Lets get weights into action!

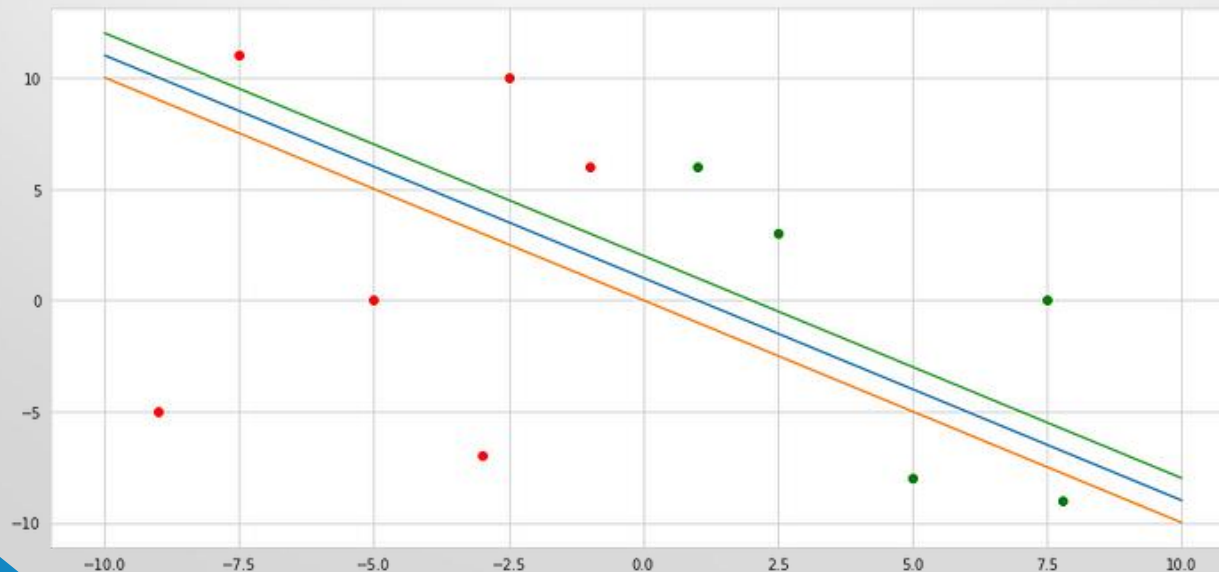
$$w_2x_2 = -w_1x_1 + b$$

- The line that better fits the given dataset is: $x_1 = 0$,

$$w_2x_2 = -w_1x_1 + b$$

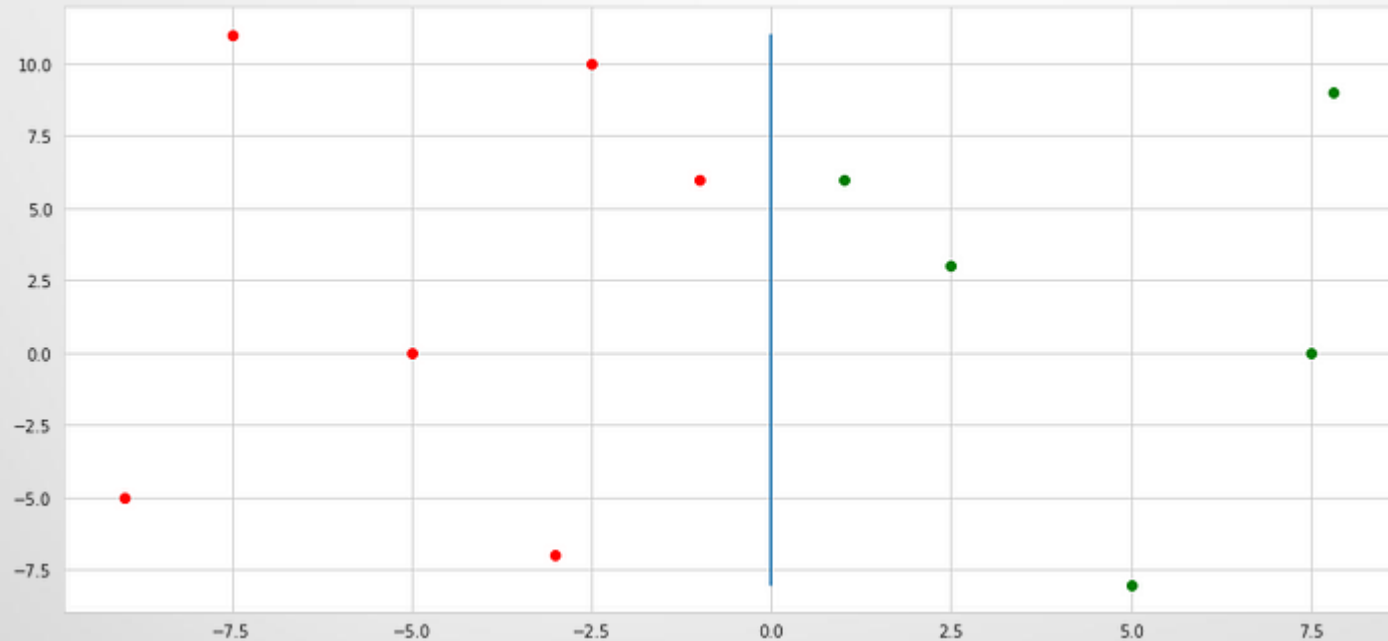
$$(0)x_2 = -(1)x_1 + 0$$

$$x_1 = 0$$



The Role of Weights and Bias in a NN?

- When $w_2 = 0$, $w_1 = 1$, $b = 0$, the equation fits the data set in a best way.



- From the plot, we can observe that the slope of the line has been changed with the introduction of weights in the equation.

What do the weights in a Neuron convey to us?

1. Importance of the feature: Weights associated with each feature, convey the importance of that feature in predicting the output value. Features with weights that are close to zero said to have lesser importance in the prediction process compared to the features with weights having a larger value.

2. Tells the relationship between a particular feature in the dataset and the target value: let us consider an example of finding the likelihood of buying a car, with the dataset containing two input features like

- i. Car price
- ii. Car Popularity
- let us suppose that people often tend to buy a car within their budget and the most popular one among many.

$$\text{buying car} \propto \text{Popularity of Car}$$

$$\text{Buying Car} \propto \frac{1}{\text{price of Car}}$$

What do the weights in a Neuron convey to us?

$$w_1 * (\text{price of car}) + w_2 * (\text{popularity of car}) + b$$

- If the price of the car increases, then the expression value also increases, that means “we are more likely to buy that car”, but we don’t want that to happen, so we have to compensate it with a negative valued weight (w_1) so that their product becomes negative and the value of expression decreases, that implies we are not interested in buying that car, in this way weights help us.
- So if the weight associated with a feature is positive it implies that there is a direct relationship between that feature and the target value, and if the weight associated with the feature is negative it implies that there is an inverse relationship between the feature and the target value.

Summary – Weights

- Weights play an important role in changing the orientation or slope of the line that separates two or more classes of data points.
- Weights tell the importance of a feature in predicting the target value.
- Weights tell the relationship between a feature and a target value

Use of Bias in the Neuron?

- Bias is used for shifting the activation function towards the left or right,
 - did not get this statement?
- let us consider a sigmoid activation function to demonstrate the use of bias, we can represent the sigmoid activation function with the mathematical expression as

$$\text{sigmoid function} = \frac{1}{1 + e^{-x}}$$

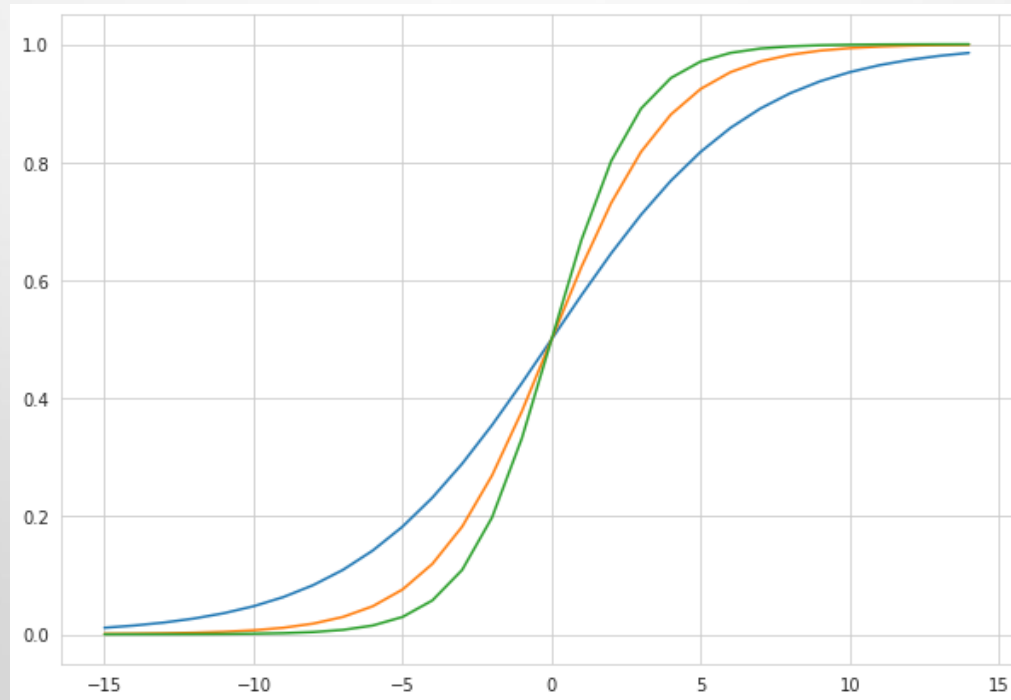
- replacing x with the equation of a line

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

Use of Bias in the Neuron?

1. Let us vary different values of w and fix the b value to 0

- $w = 0.3$ - the blue line in the plot
- $w = 0.5$ - the orange line in the plot
- $w = 0.7$ - the green line in the plot



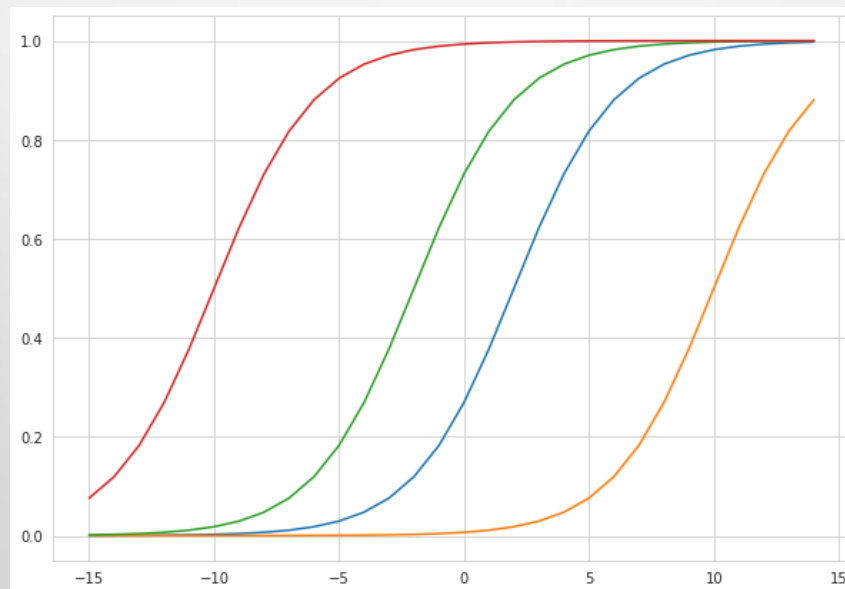
Use of Bias in the Neuron?

- Even while giving different values of w , we could not shift the center of the activation function, in this case, the sigmoid function.
- Changing the value of w only changes the steepness of the curve, but there is no way we can shift the curve towards left or right, the only way to shift the curve towards left or right is by changing the value of bias(b).

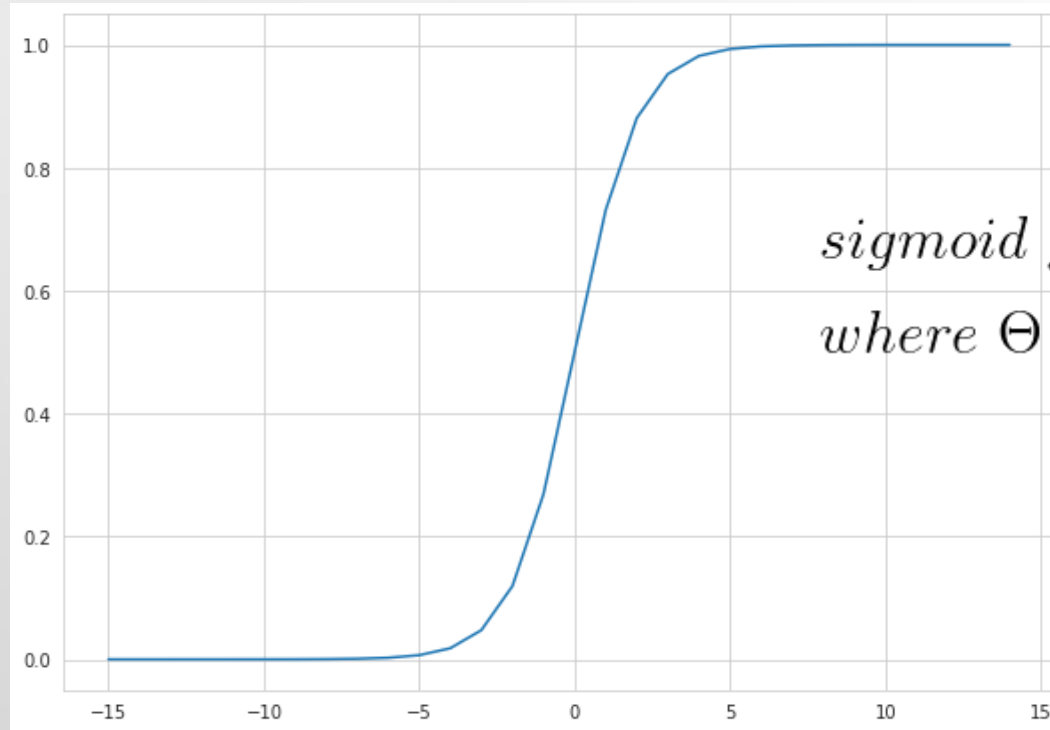
Use of Bias in the Neuron?

2. Let us vary different values of b and fix w value to 0.5

- $b = -1$ — the red line in the plot
- $b = -5$ — the green line in the plot
- $b = 1$ — the blue line in the plot
- $b = 5$ — the orange line in the plot



Why do we need to shift the activation function towards the left or right?



$$\text{sigmoid function} = \frac{1}{1 + e^{-(\Theta)}}$$

where $\Theta = (1)x + 0$

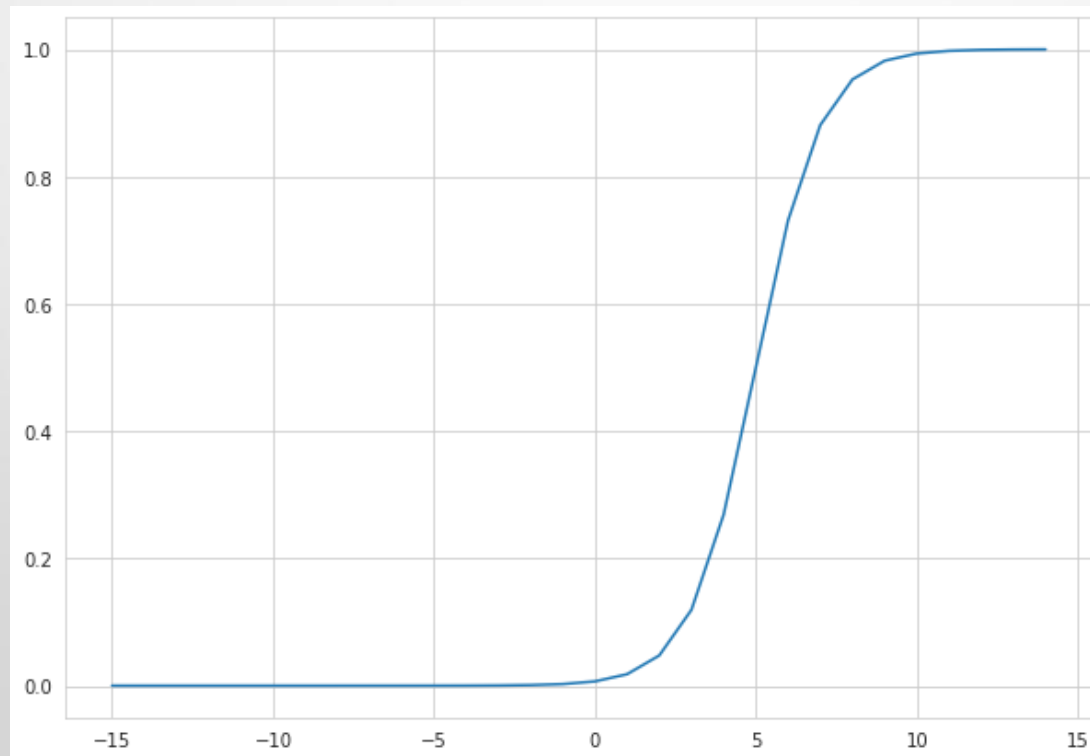
- From the plot, we infer that all the values towards the **right of 0** are **mapped to 1** and all the values towards **the left of 0** are mapped to 0.

Why do we need to shift the activation function towards the left or right?

- What if we want y value to be 0 when $x < 5$?

$$\text{sigmoid function} = \frac{1}{1 + e^{-(\Theta)}}$$

where $\Theta = (1)x - 5$

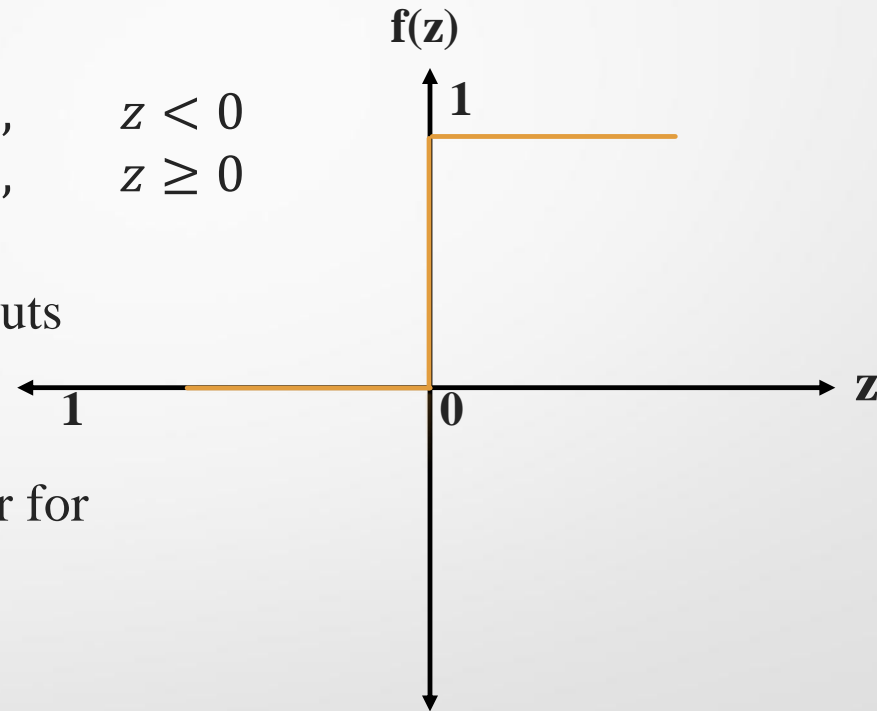


Activation functions (linear)

1. Step function:

$$f(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

- Range of function = possible outputs
= $\{0, 1\}$
- Used in: hidden layer, output layer for
classification (binary) problems

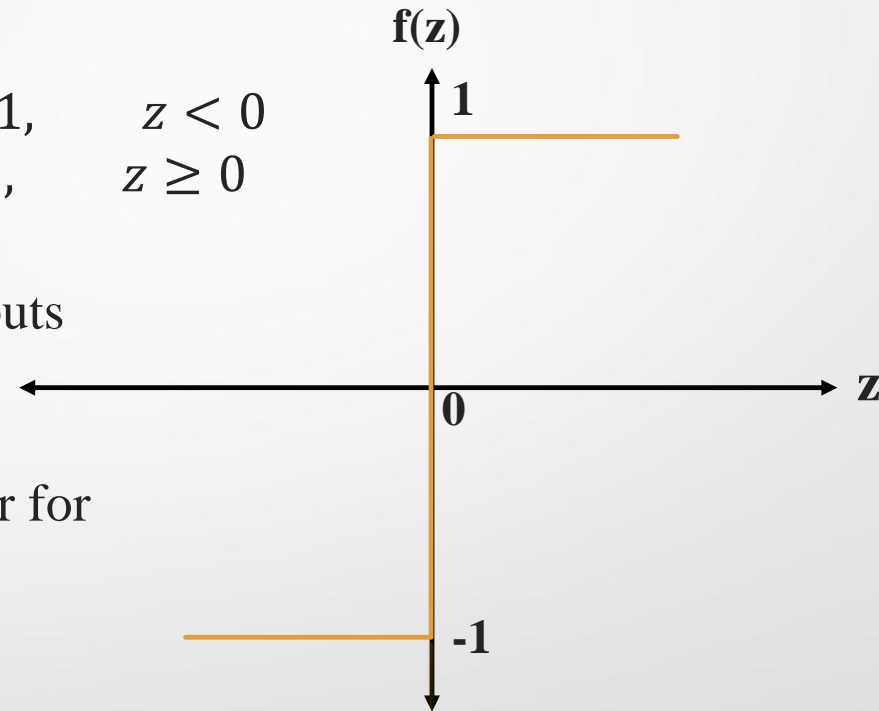


Activation functions (linear)

2. Signum or sgn or sign function:

$$f(z) = \begin{cases} -1, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

- Range of function = possible outputs
= $\{-1, 1\}$
- Used in: hidden layer, output layer for
classification (binary) problems

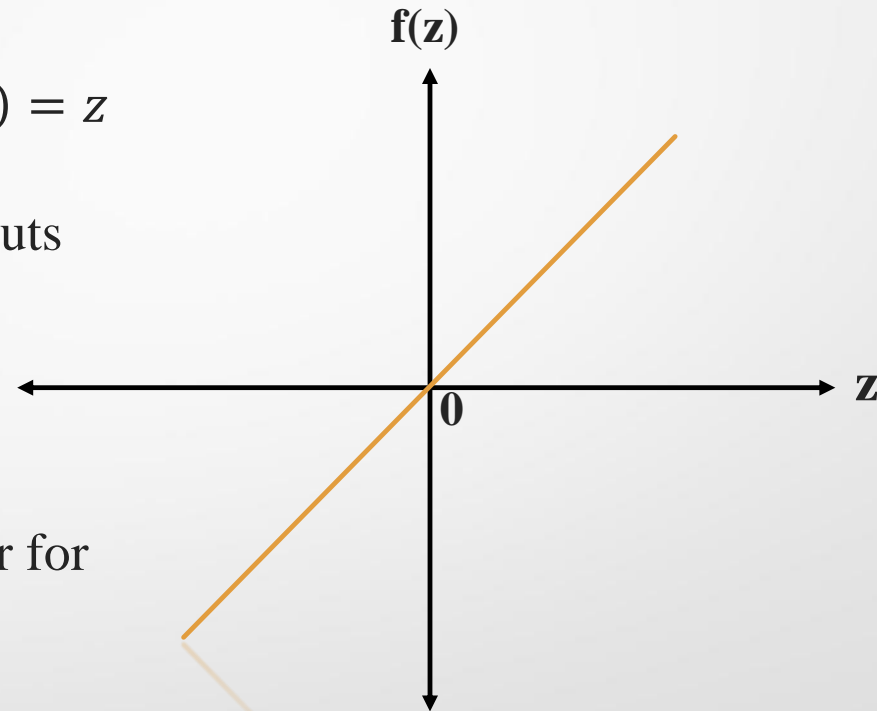


Activation functions (linear)

3. Linear function:

$$f(z) = z$$

- Range of function = possible outputs
= $\{-\infty, \infty\}$
- Used in: hidden layer, output layer for regression

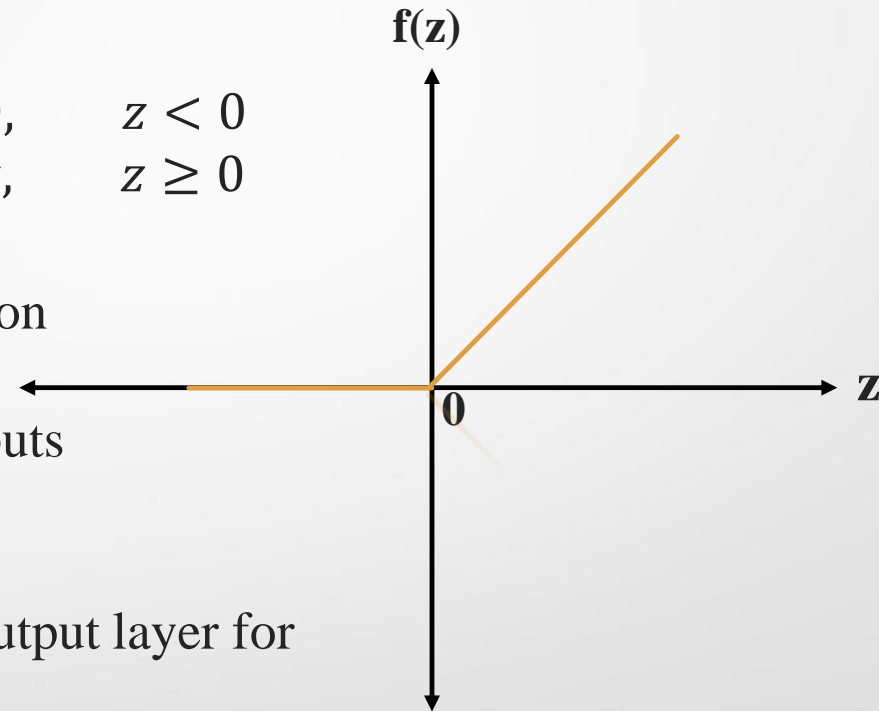


Activation functions (linear)

4. Rectifier Linear Unit (ReLU) function:

$$f(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$$

- Modified version of Linear function
- Range of function = possible outputs
= $[0, \infty)$
- Used in: hidden layer for CNN, output layer for regression (only +ve outputs)

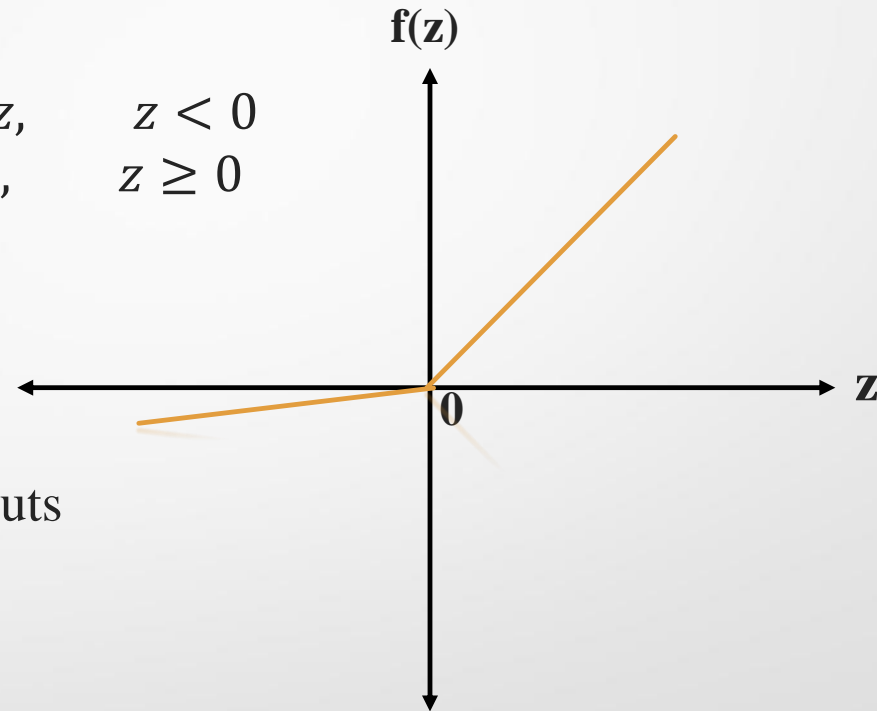


Activation functions (linear)

5. Leaky ReLU function:

$$f(z) = \begin{cases} az, & z < 0 \\ z, & z \geq 0 \end{cases}$$

- a is small +ve number
 - Control how much leakage is allowed for -ve numbers
- Range of function = possible outputs
= $\{-\infty, \infty\}$
- Used in: hidden layer for CNN

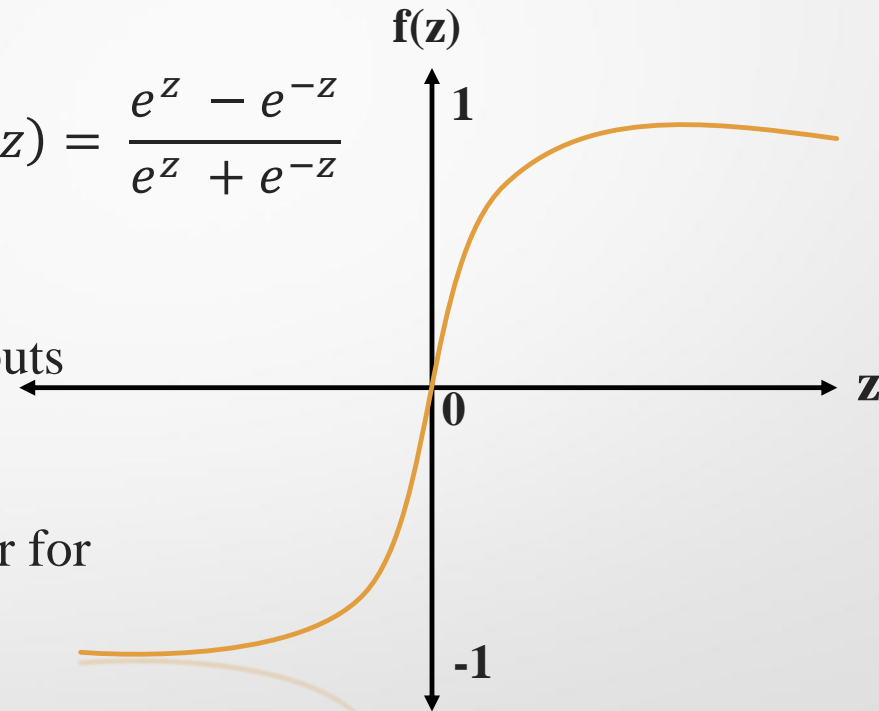


Activation functions (non-linear)

6. Hyperbolic Tangent (Tanh(z)) function:

$$f(z) = \text{Tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- e = Euler's number ≈ 2.71828
- Range of function = possible outputs
= $\{-1, 1\}$
- Used in: hidden layer, output layer for classification

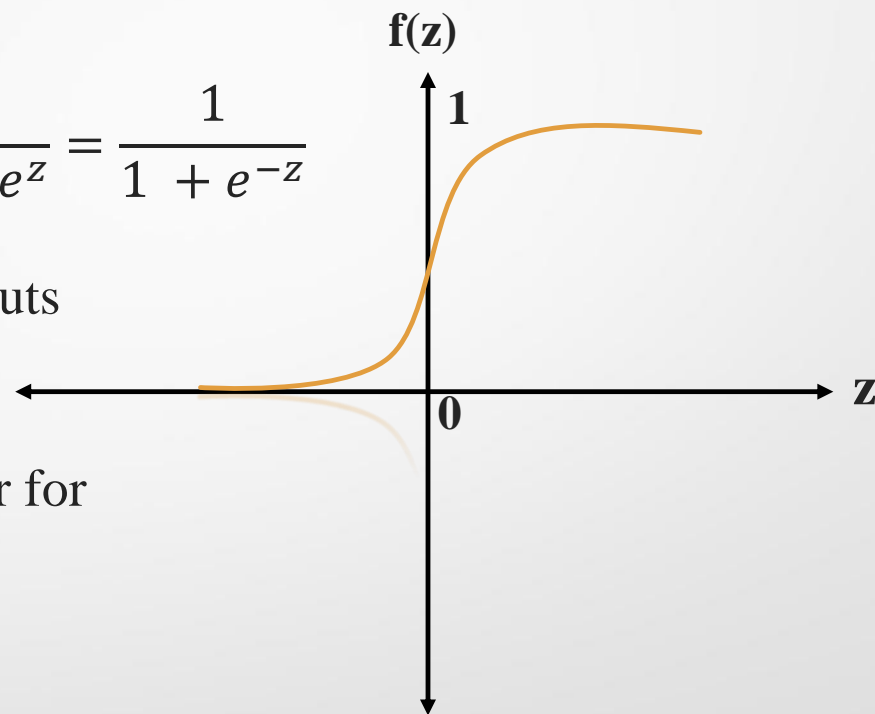


Activation functions (non-linear)

7. Sigmoid function:

$$f(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$

- Range of function = possible outputs
= $\{0, 1\}$
- Used in: hidden layer, output layer for classification



Activation functions (non-linear)

8. Softmax function:


- Multi-class extension of Sigmoid function
- Let z_1, z_2, \dots, z_n are the n inputs for output layers

$$f(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \dots + e^{z_n}}$$
$$f(z_i) = \frac{e^{z_i}}{e^{z_1} + e^{z_2} + \dots + e^{z_n}}$$

- Range of function = possible outputs
= $\{0, 1\}$
- Used in: output layer for multi-class classification

Cost functions in NN

1. Mean Absolute Error (MAE):

$$MAE = \frac{1}{n} \sum_{i=1}^n abs(y_i - \hat{y}_i)$$


The diagram shows the formula for Mean Absolute Error (MAE). Below the formula, there are two boxes: 'Actual' and 'Predicted'. An arrow points from the 'Actual' box to the y_i term in the formula, and another arrow points from the 'Predicted' box to the \hat{y}_i term.

- Total error in the model = error1 + error2 + + errorn
= $\sum_{i=1}^n abs(y_i - \hat{y}_i)$
- Individual errors are called **Loss**
- Cumulative error is called **Cost**

Cost functions in NN

2. Mean Square Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

3. Log Loss or Binary Cross Entropy:

$$Log Loss = -\frac{1}{n} \sum_{i=0}^n y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i)$$

- Used in Logistic regression

Gradient Descent for NN

- Gradient Descent is used during the training of NN.

x	y
2	-0.5
3	0
5	1
7	2
9	3

$$y = 0.5 * x - 1.5$$

Weight Bias

- It helps find these parameter (Weight and Bias)
- y is called a predication/classification function.

Gradient Descent for NN

Age	Affordability	Insurance?
22	1	0
25	0	0
47	1	1
52	0	0
46	1	1
56	1	1
55	0	0
60	0	1
62	1	1
61	1	1
18	1	0
28	1	0
27	0	1

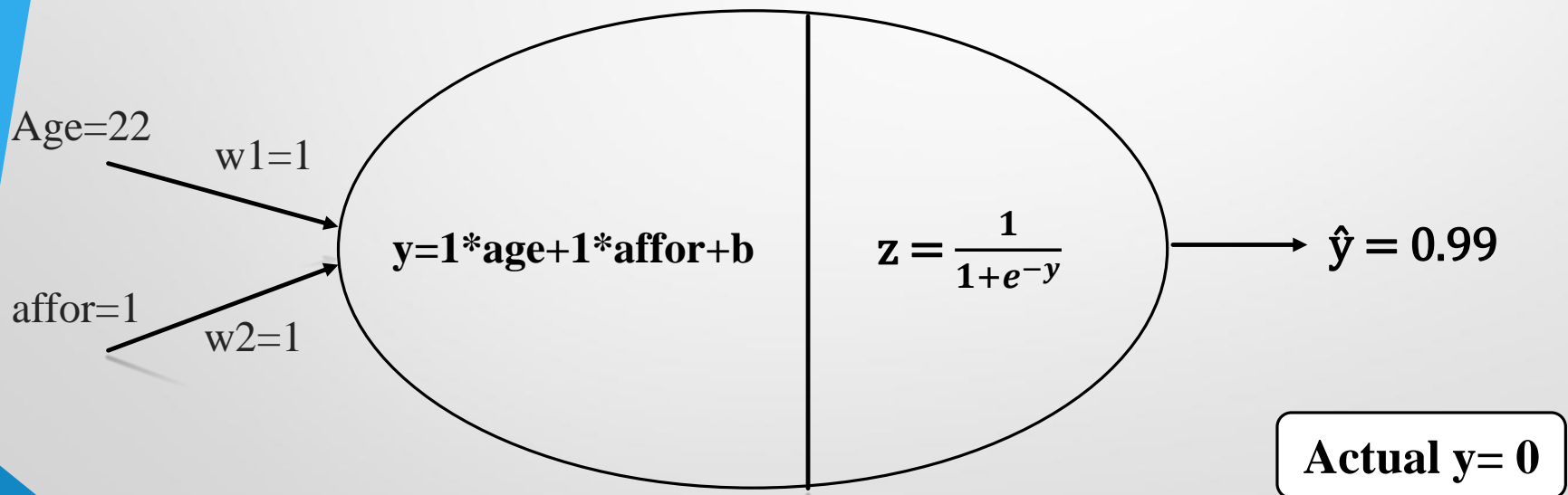
Problem Statement: Given an age and affordability, come up with a model/function that can predict/classify if person will buy insurance or not.

Gradient Descent for NN

$$y = w1*x1 + w2*x2 + \text{bias}$$

$$y = w1*\text{age} + w2*\text{affordability} + \text{bias}$$

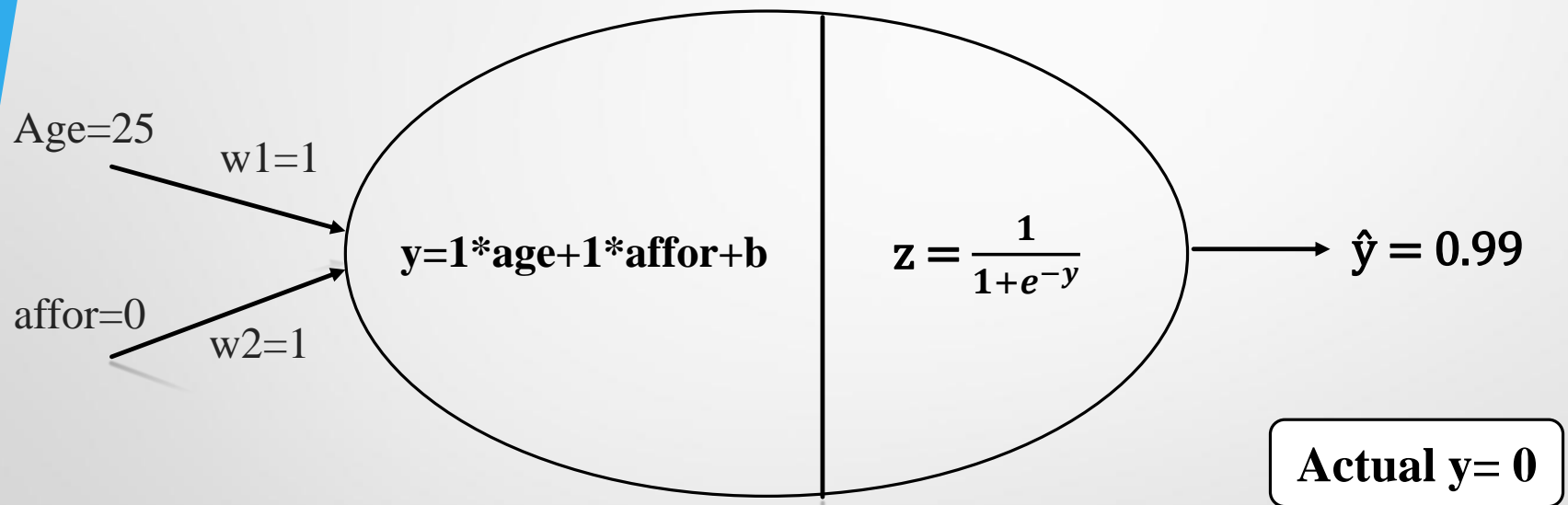
1. Age = 22 and affordability = 1



$$\begin{aligned} \text{error1} &= -(y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i)) \\ &= 4.6 \end{aligned}$$

Gradient Descent for NN

2. Age = 25 and affordability = 0

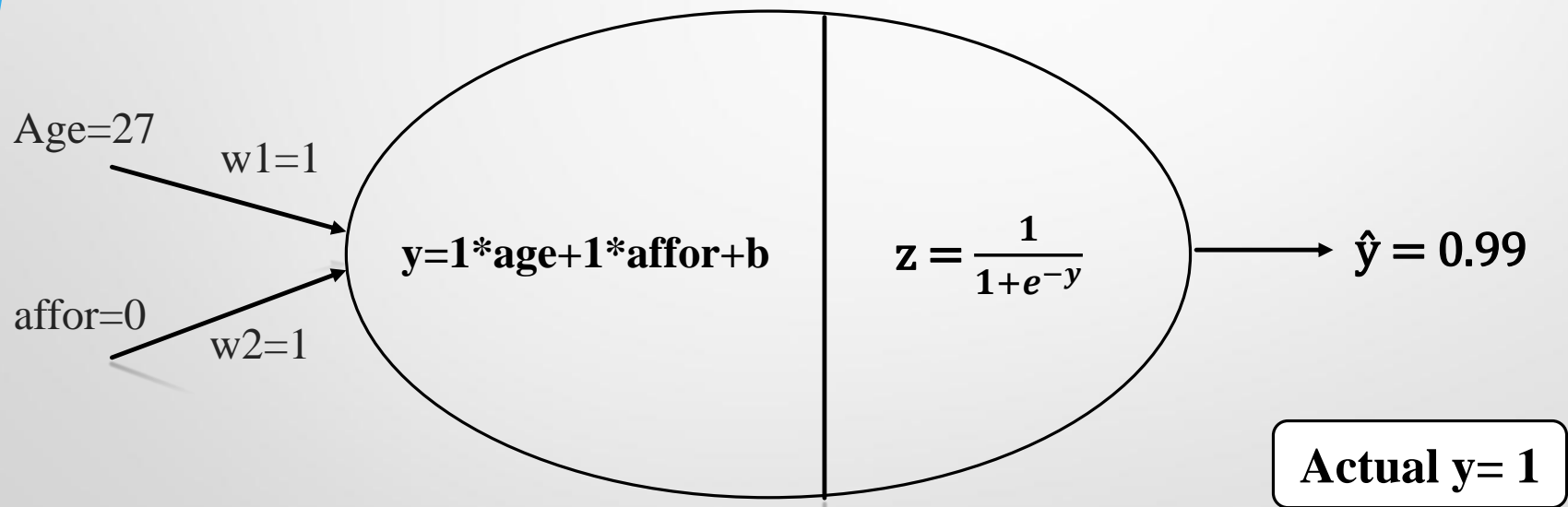


$$\begin{aligned} \text{error2} &= -(y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i)) \\ &= 4.6 \end{aligned}$$

Gradient Descent for NN

- Go through all the samples/rows of a dataset

13. Age = 27 and affordability = 0



$$\begin{aligned} \text{error13} &= -(y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i)) \\ &= 0.01 \end{aligned}$$

Gradient Descent for NN

- Total error = error1 + error2 + + error13
= 4.31
- This error comes after the first epoch.
- Now, back propagate the error (i.e. 4.31) so that w1 and w2 can be further adjusted.
 - i.e. adjust the weights w1 and w2 in such a way that the LogLoss is less than 4.31.
- How?????

Gradient Descent for NN

- From w_1 subtract/add something

$$w_1 = w_1 - \text{something}$$

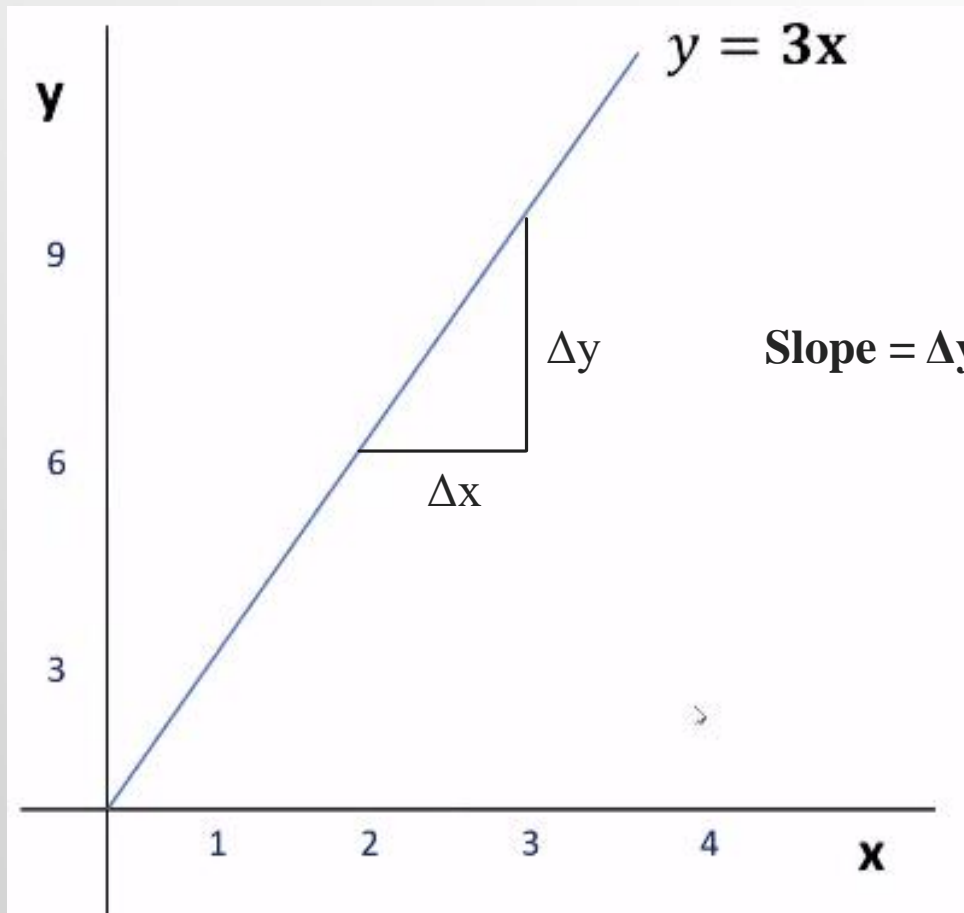
- What is the value of something/subtraction

$$w_1 = w_1 - \text{learning rate} * \frac{\partial}{\partial w_1}$$

**It's a small value (0.01)
so that weight do not
change drastically**

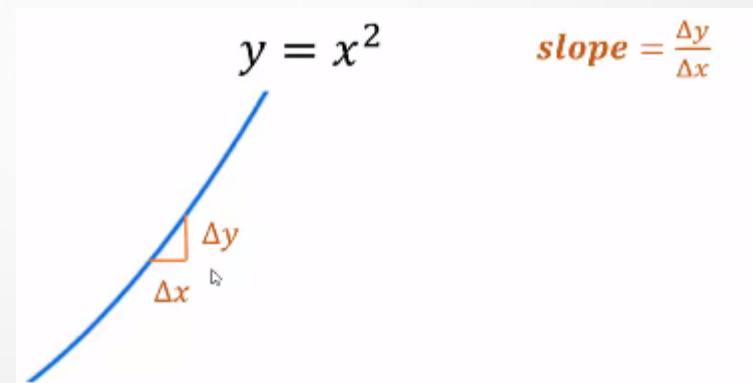
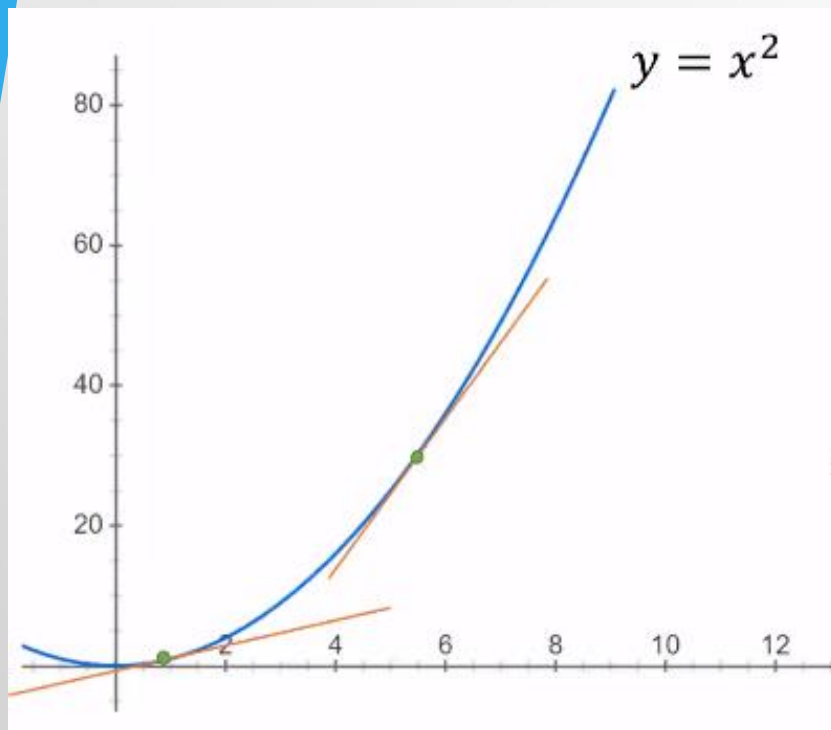
**Derivative of Loss
compare to w_1**

Derivatives

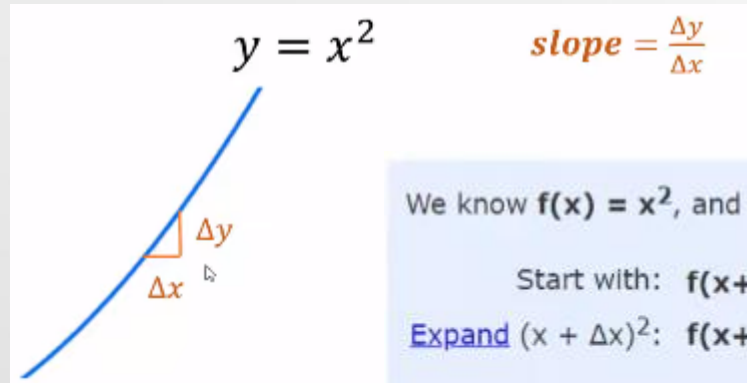


$$\text{Slope} = \Delta y / \Delta x = 3/1 = 3$$

Derivatives



Derivatives



We know $f(x) = x^2$, and we can calculate $f(x+\Delta x)$:

$$\text{Start with: } f(x+\Delta x) = (x+\Delta x)^2$$

$$\text{Expand } (x + \Delta x)^2: f(x+\Delta x) = x^2 + 2x \Delta x + (\Delta x)^2$$

$$\text{The slope formula is: } \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$\text{Put in } f(x+\Delta x) \text{ and } f(x): \frac{x^2 + 2x \Delta x + (\Delta x)^2 - x^2}{\Delta x}$$

$$\text{Simplify } (x^2 \text{ and } -x^2 \text{ cancel}): \frac{2x \Delta x + (\Delta x)^2}{\Delta x}$$

$$\text{Simplify more (divide through by } \Delta x): = 2x + \Delta x$$

$$\text{Then as } \Delta x \text{ heads towards } 0 \text{ we get: } = 2x$$

Result: the derivative of x^2 is $2x$

Derivatives

$$\frac{\partial}{\partial x} x^2 = 2x$$

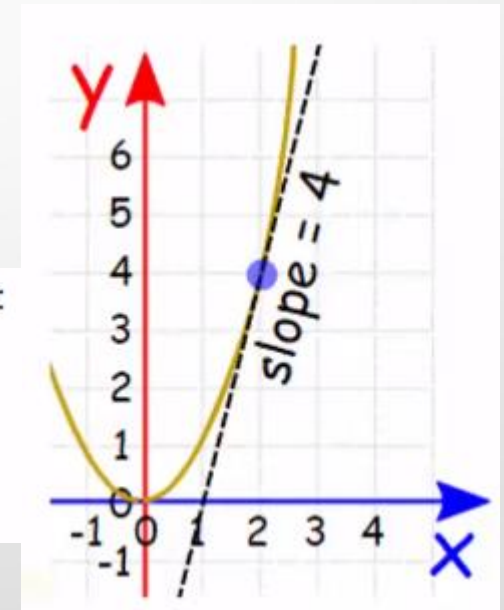
derivative of x^2 with respect to x

What does $\frac{d}{dx} x^2 = 2x$ mean?

It means that, for the function x^2 , the slope or "rate of change" at any point is **$2x$** .

So when **$x=2$** the slope is **$2x = 4$** , as shown here:

Or when **$x=5$** the slope is **$2x = 10$** , and so on.



Derivatives

$$\frac{\partial}{\partial x} x^3 = 3x^2$$

$$\frac{\partial}{\partial x} 7x^3 = 21x^2$$

$$\frac{\partial}{\partial x} 2x^3 + x^5 = 6x^2 + 5x^4$$

Derivatives

Slope

Used for linear equation

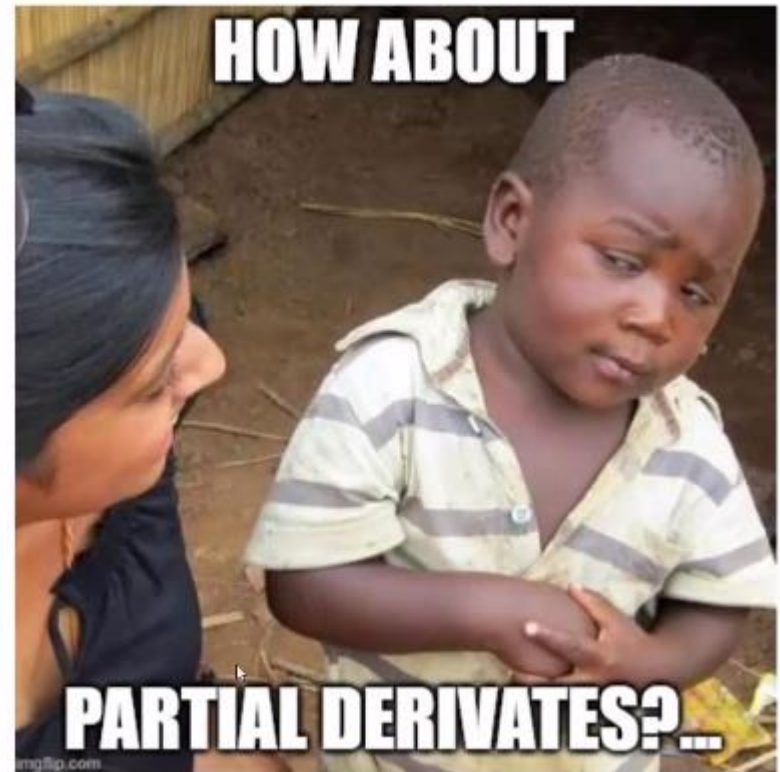
It is a constant

Derivatives

Used for non-linear equation

It is a function

Derivatives



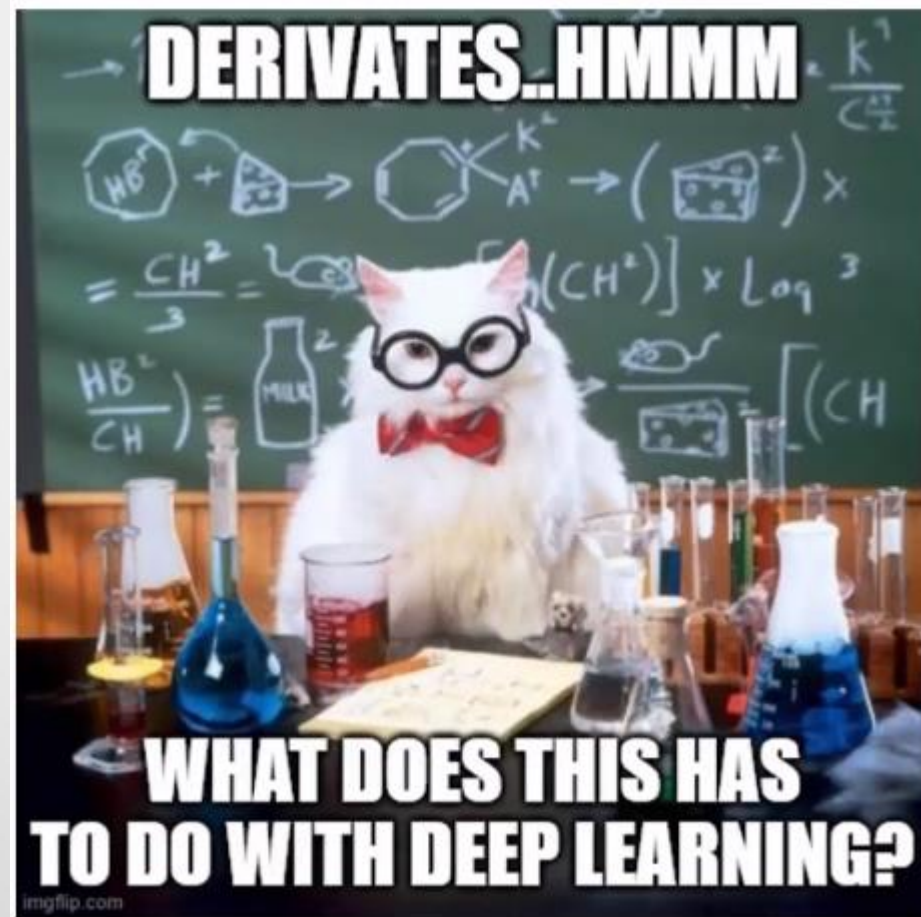
Derivatives

$$f(x, y) = x^3 + y^2$$

$$\partial f / \partial x = 3x^2 + 0 = 3x^2$$

$$\partial f / \partial y = 0 + 2y = 2y$$

Derivatives



Derivatives

$$f(x, y) = x^3 + y^2$$

$$\partial f / \partial x = 3x^2 + 0 = 3x^2$$

$$\partial f / \partial y = 0 + 2y = 2y$$

Derivatives

$$f(\text{bedrooms}, \text{sqr ft}) = \text{bedrooms}^3 + \text{sqr ft}^2$$

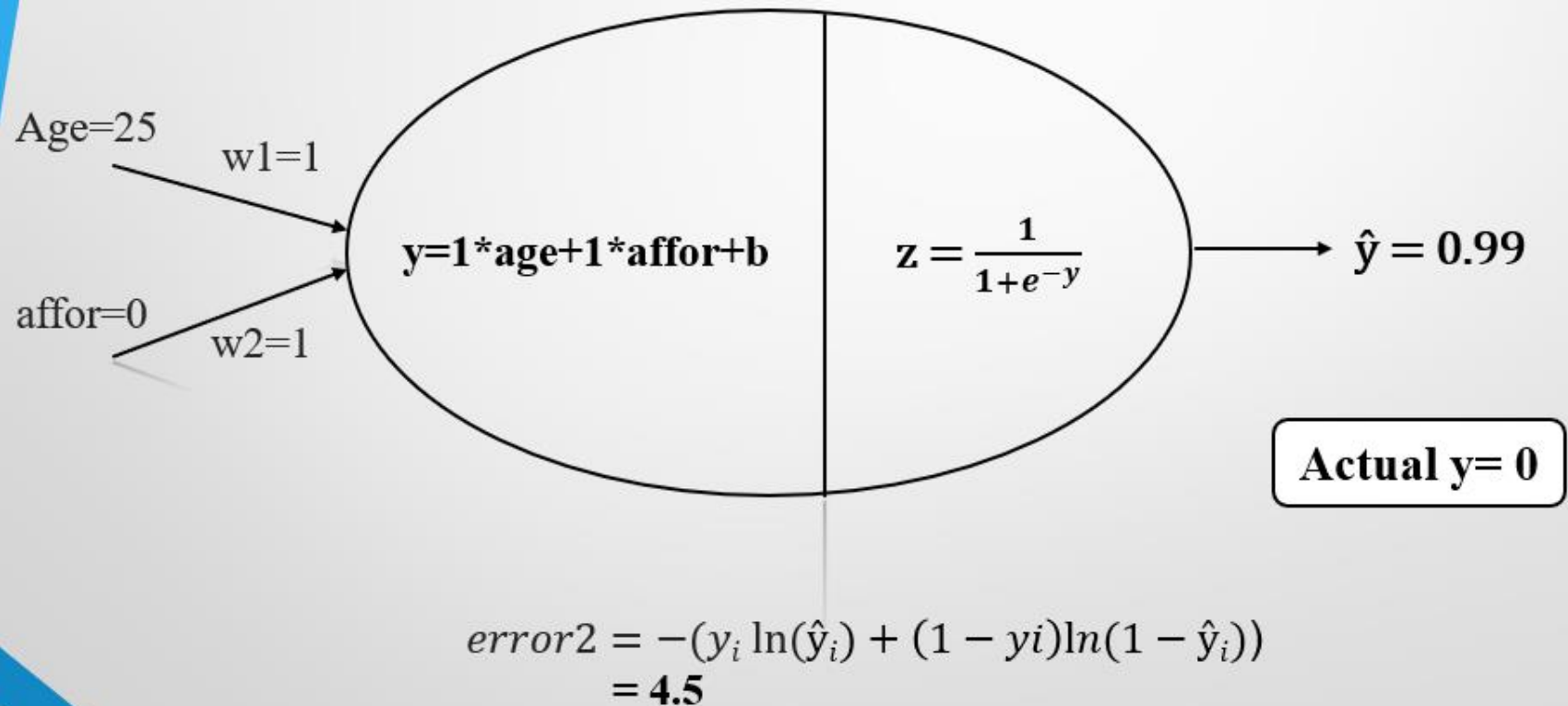
$$\frac{\partial(\text{Price})}{\partial(\text{bedrooms})} = 3 * \text{bedrooms}^2$$

How much a **price** is changing given a change in *bedrooms*

$$\frac{\partial(\text{Price})}{\partial(\text{sqr ft})} = 2 * \text{sqr ft}$$

How much a **price** is changing given a change in *sqr ft*

Derivatives



Based on the Error, we want to modify the weight.

Gradient Descent for NN

- From $w1$ subtract/add something

$$w1 = w1 - \text{something}$$

- What is the value of something/subtraction

$$w1 = w1 - \text{learning rate} * \frac{\partial}{\partial w1}$$

**It's a small value (0.01)
so that weight do not
change drastically**

**Derivative of Loss
compare to $w1$**

Gradient Descent for NN

$$w1 = w1 - \text{learning rate} * \frac{\partial}{\partial w1}$$

$$w2 = w2 - \text{learning rate} * \frac{\partial}{\partial w2}$$

$$b = b - \text{learning rate} * \frac{\partial}{\partial b}$$

Gradient Descent for NN

$$\frac{\partial}{\partial x} x^n = n x^{n-1}$$

$$\frac{\partial}{\partial x} \left(\frac{1}{x} \right) = \frac{\partial}{\partial x} (x^{-1}) = -\frac{1}{x^2}$$

$$\frac{\partial}{\partial x} \log x = \frac{1}{x}$$

$$\frac{\partial}{\partial x} e^x = e^x$$

$$\frac{\partial}{\partial x} e^{-x} = -e^{-x}$$

$$\frac{\partial}{\partial x} [f(x)]^n = n[f(x)]^{n-1} \times \frac{\partial}{\partial x} f(x)$$

$$\frac{\partial Cost}{\partial w} = ?$$

$$\frac{\partial Cost}{\partial b} = ?$$

$$Log Loss = -\frac{1}{n} \sum_{i=0}^n y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

$$\begin{aligned} \hat{y} &= a = \sigma(WX + b) \\ &= \sigma(z) \end{aligned}$$

$$z = (WX + b)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Gradient Descent for NN

$$\text{Log Loss} = -\frac{1}{n} \sum_{i=0}^n y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

$$\begin{aligned}\hat{y} &= a = \sigma(WX + b) \\ &= \sigma(z)\end{aligned}$$

$$z = (WX + b)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$L = -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

$$W = [w_1, w_2, \dots, w_n]$$

$$\frac{\partial \text{Cost}}{\partial w} \Rightarrow \frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_n}$$

$$\frac{\partial}{\partial x} x^n = n x^{n-1}$$

$$\frac{\partial}{\partial x} \left(\frac{1}{x} \right) = \frac{\partial}{\partial x} (x^{-1}) = -\frac{1}{x^2}$$

$$\frac{\partial}{\partial x} \log x = \frac{1}{x}$$

$$\frac{\partial}{\partial x} e^x = e^x$$

$$\frac{\partial}{\partial x} e^{-x} = -e^{-x}$$

$$\frac{\partial}{\partial x} [f(x)]^n = n[f(x)]^{n-1} \times \frac{\partial f(x)}{\partial x}$$

Gradient Descent for NN

$$L = -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

$$\hat{y} = a = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$z = (WX + b)$$

$$\frac{\partial \text{Cost}}{\partial W} \Rightarrow \frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_n}$$

$$\frac{\partial L}{\partial w} = \underbrace{\frac{\partial L}{\partial a} * \frac{\partial a}{\partial w}}_{\text{Chain rule}} = \underbrace{\frac{\partial L}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w}}_{\text{Chain rule}}$$

Chain rule

$$\frac{\partial}{\partial x} x^n = n x^{n-1}$$

$$\frac{\partial}{\partial x} \left(\frac{1}{x} \right) = \frac{\partial}{\partial x} (x^{-1}) = -\frac{1}{x^2}$$

$$\frac{\partial}{\partial x} \log x = \frac{1}{x}$$

$$\frac{\partial}{\partial x} e^x = e^x$$

$$\frac{\partial}{\partial x} e^{-x} = -e^{-x}$$

$$\frac{\partial}{\partial x} [f(x)]^n = n[f(x)]^{n-1} \times \frac{\partial f(x)}{\partial x}$$

Gradient Descent for NN

$$w1 = w1 - \text{learning rate} * \frac{\partial}{\partial w1}$$

$$w2 = w2 - \text{learning rate} * \frac{\partial}{\partial w2}$$

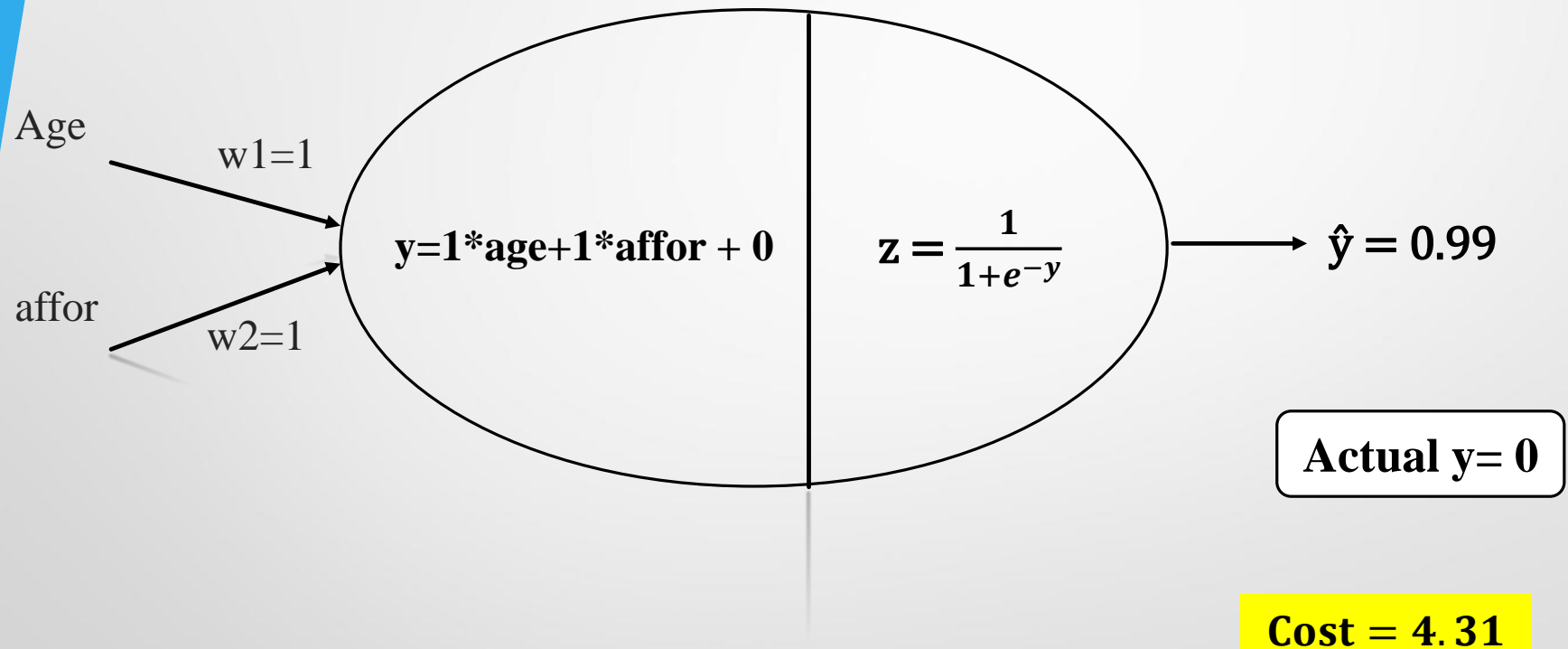
$$b = b - \text{learning rate} * \frac{\partial}{\partial b}$$

$$\frac{\partial}{\partial w1} = \frac{1}{n} \sum_{i=1}^n x_i (\hat{y}_i - y_i)$$

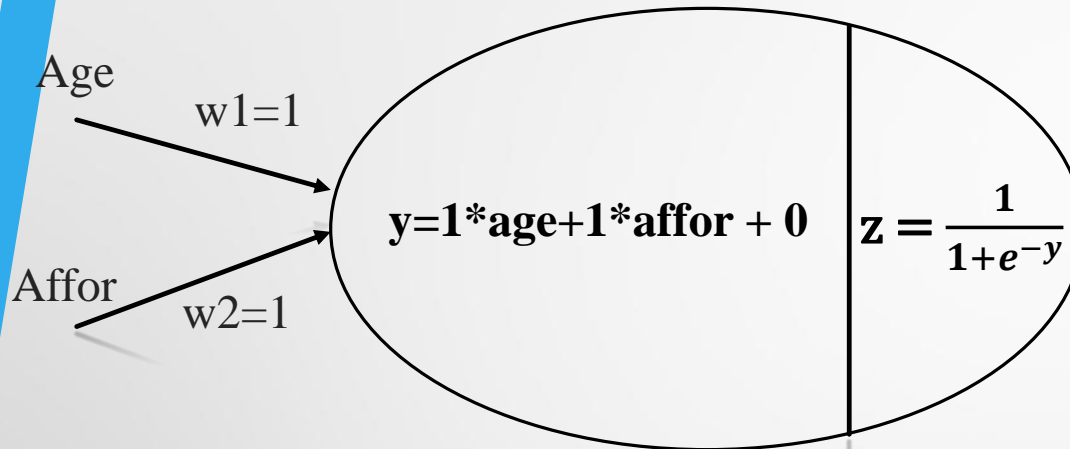
$$\frac{\partial}{\partial w2} = \frac{1}{n} \sum_{i=1}^n x_i (\hat{y}_i - y_i)$$

$$\frac{\partial}{\partial b} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)$$

Gradient Descent for NN



Gradient Descent for NN



$$w1 = w1 - \text{learning rate} * \frac{\partial}{\partial w1}$$

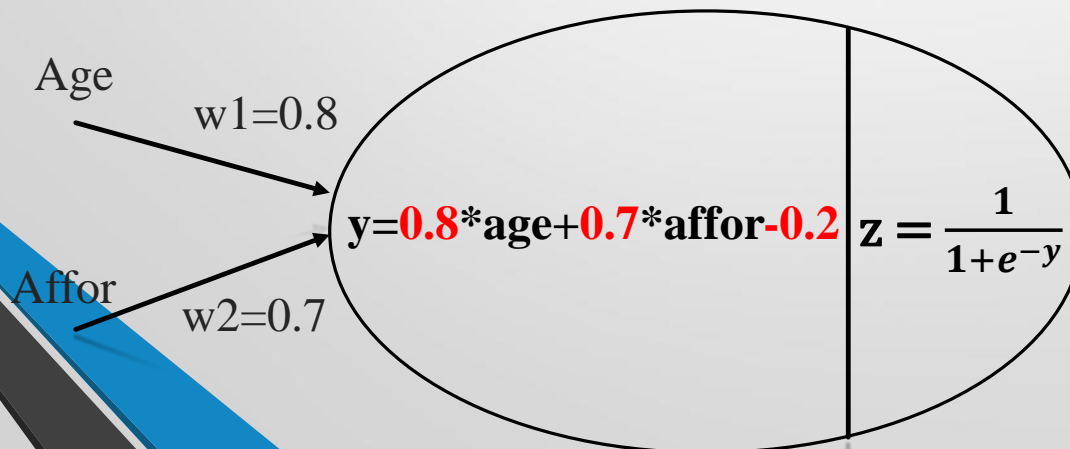
$$w1 = 1 - 0.2 = \mathbf{0.8}$$

$$w2 = w2 - \text{learning rate} * \frac{\partial}{\partial w2}$$

$$w2 = 1 - 0.3 = \mathbf{0.7}$$

$$b = b - \text{learning rate} * \frac{\partial}{\partial b}$$

$$b = 0 - 0.2 = \mathbf{-0.2}$$



Gradient Descent for NN

Age	Affordability	Insurance?
22	1	0
25	0	0
47	1	1
52	0	0
46	1	1
56	1	1
55	0	0
60	0	1
62	1	1
61	1	1
18	1	0
28	1	0
27	0	1

End of second epoch



Actual $y = 0$

$$\text{error1} = -(y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i))$$

$$\text{Total error} = \text{error1} + \text{error2} + \dots + \text{error13}$$

Gradient Descent for NN

- Repeat the process for multiple epochs.

When to stop?

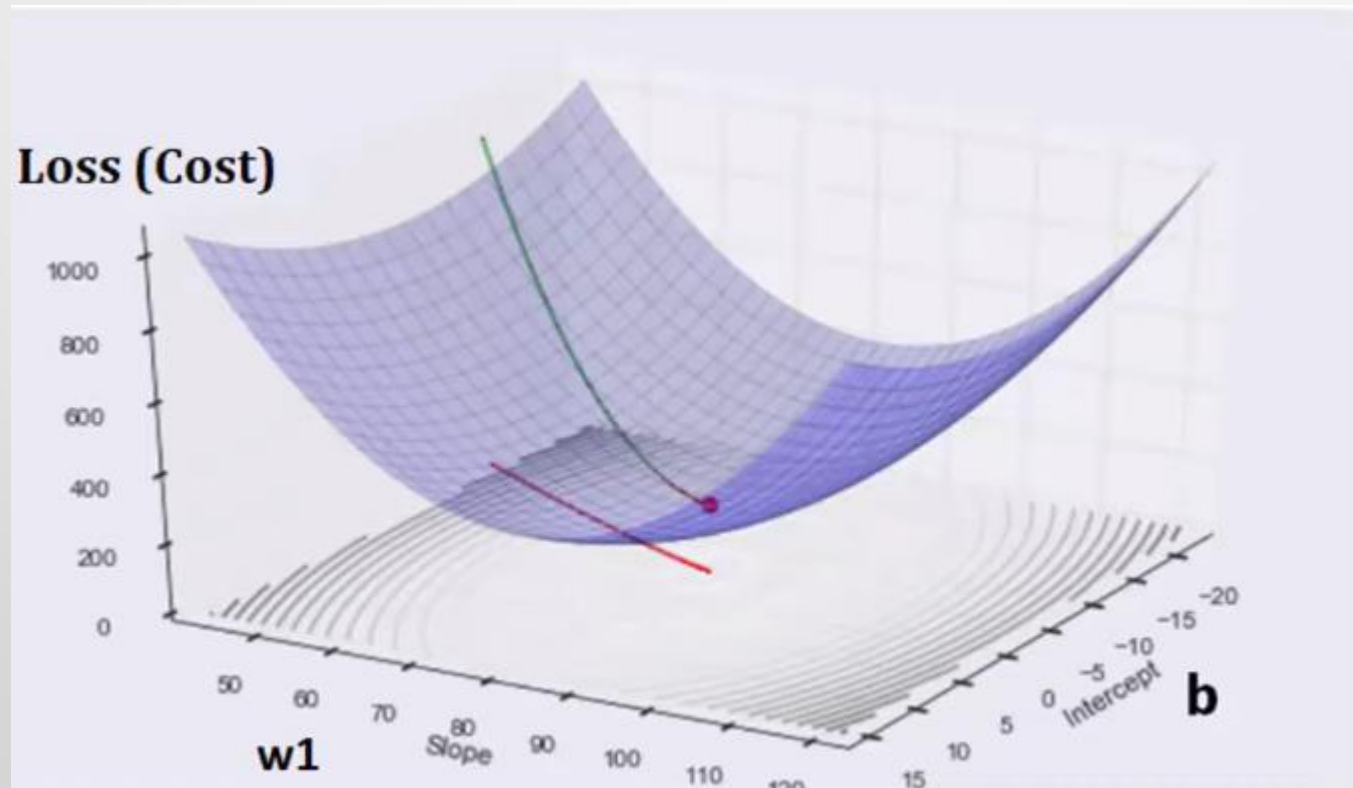
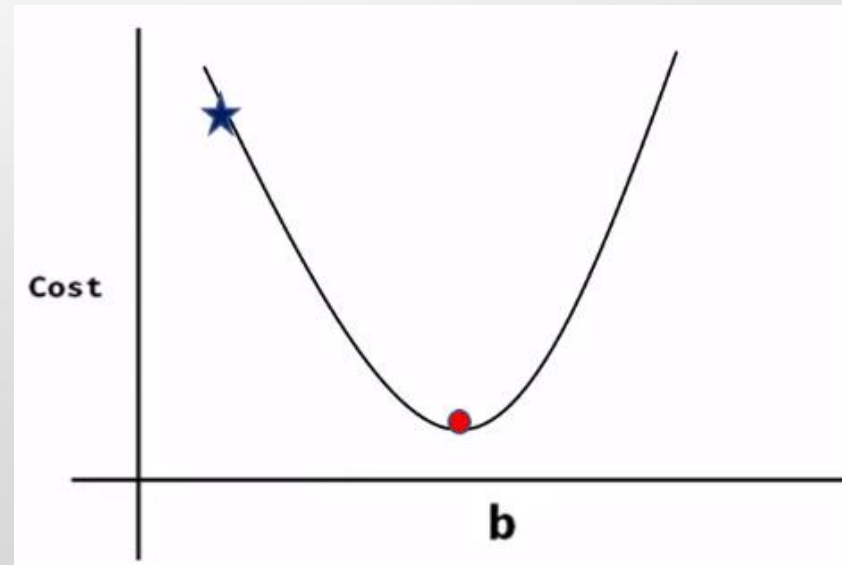
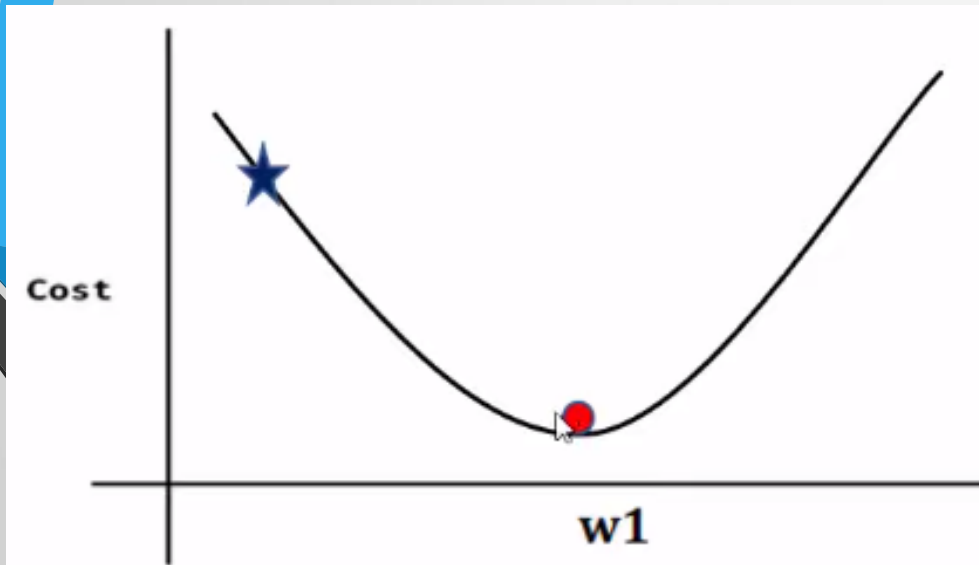
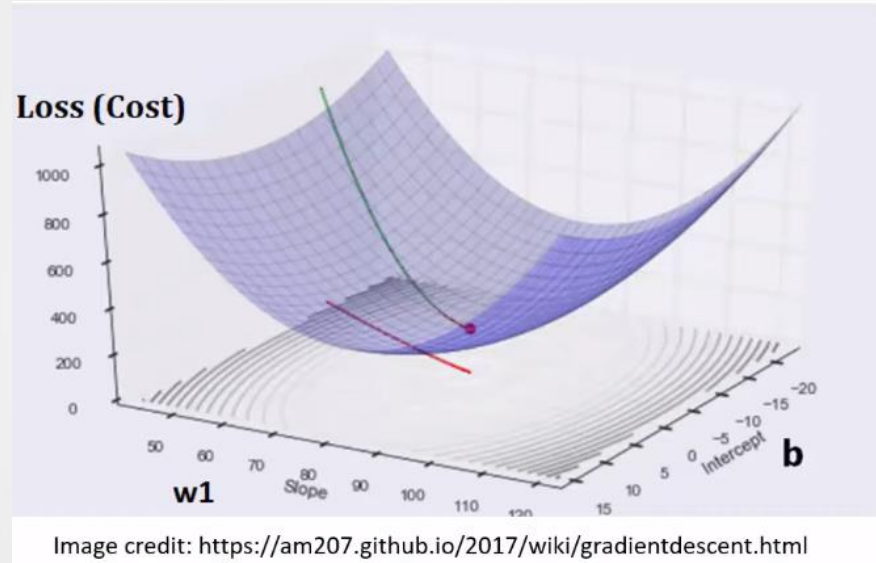
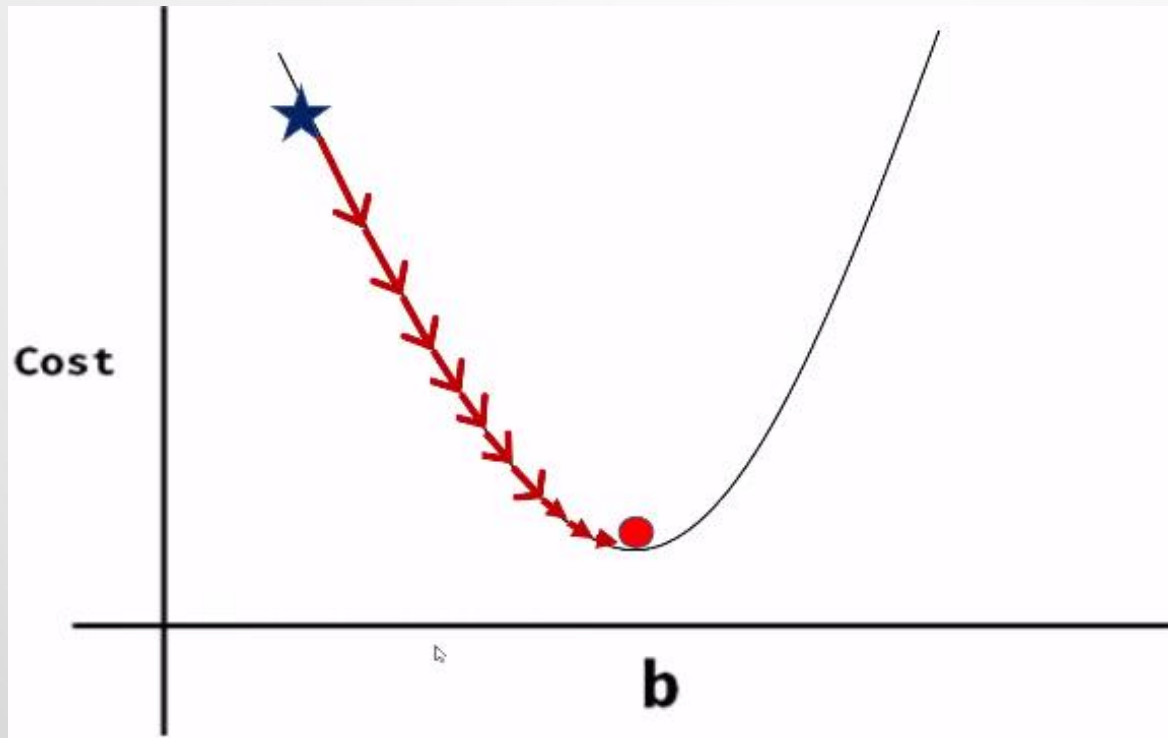


Image credit: <https://am207.github.io/2017/wiki/gradientdescent.html>

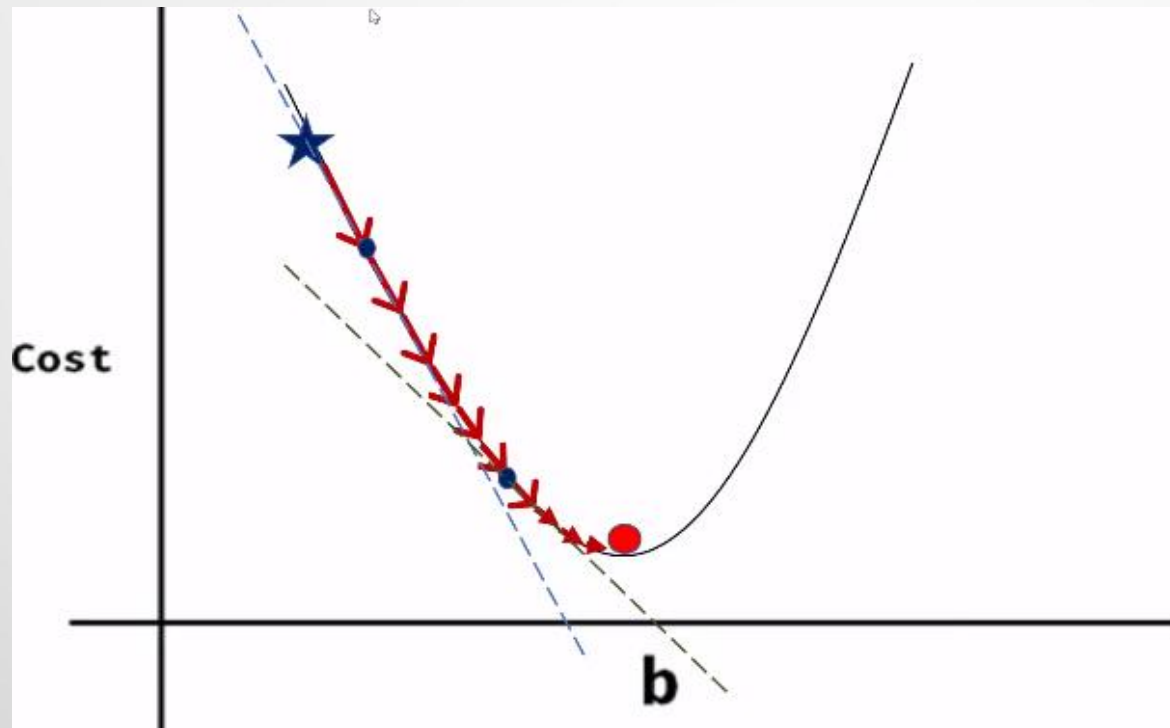
Gradient Descent for NN



Gradient Descent for NN



Gradient Descent for NN



Types of Gradient Descent

1. Batch Gradient Descent:

- Go through all training samples before adjusting the weights and calculate the error.
- Back propagate and adjust weight.
- **Problem:** What if we have 10 million samples
 - To find cumulative error for 1st epoch, we need to do a forward pass for 10 million samples.
 - If we have two features in dataset, it requires finding 20 million derivatives.

What is dataset has 200 features???

Types of Gradient Descent

2. Stochastic Gradient Descent:

- Randomly pick single data training sample and find error.
- Start adjusting the weights.
- Repeat for new randomly picked sample.
- Good for large training dataset.

Types of Gradient Descent

3. Mini Batch Gradient Descent:

- Instead of choosing one randomly picked training sample, we will use a **batch** of **randomly picked** training samples.
- Use a batch (randomly picked) training samples for a forward pass and then adjust the weights.

Dropout regularization

- Technique used to deal with overfitting in Deep learning.
- Adding of dropout layer will increase the model performance.
- **How dropout helps in dealing overfitting?**
 - It can't rely on one input as it might be dropped out at random.
 - Neurons will not learn redundant details of input.

How to Decide on a Network Topology?

- # of input nodes?
 - Number of features
- # of output nodes?
 - Suitable to encode the output representation
- Transfer (activation) function?
 - Suitable to the problem
- # of hidden nodes?
 - Not exactly known

NN design issues

- Data representation
- Network topology
- Network parameters
- Training

NN design issues

- **Data representation:**
 - Data representation depends on problem
 - In general, ANNs work on continuous (real values) attributes. Therefore, symbolic attributes need to be encoded into continuous ones.
 - Attributes of different types may have different ranges of values which affect the training process.
 - Normalization may be used for this problem.

NN design issues

- **Network topology:**
 - The number of layers and neurons depend on the specific task.
 - In practice, this issue is solved by trial and error.
 - Two types of adaptive algorithms can be used:
 - Start from large network and successively remove some neurons and links until network performance degrades.
 - Begin with a small network and introduce new neurons until model performance is satisfactory.

NN design issues

- **Network parameters:**
 - How are the weights initialized?
 - How is the learning rate chosen?
 - How many hidden layers and how many neurons?
 - How many examples in the training set?

NN design issues

- **Training:**
 - Rule of thumb: the number of training examples should be at least five to ten times the number of weights of the network.

Applications of ANNs

- ANNs have been widely used in various domains for:
 - Pattern recognition
 - Personal Assistants
 - Healthcare
 - Marketing and Sales
 - Social Media (People you may know)

Generalization and uniform convergence

- Generalization refers to your model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.
- Uniform convergence—the supremum difference between the training and test errors over a certain function class.

Concentration inequalities

- Concentration inequalities deal with deviations of functions of independent random variables from their expectation.
- Concentration inequalities furnish us bounds on how random variables deviate from a value (typically, expected value) or help us to understand how well they are concentrated.
- A random variable with high concentration is one that is close to its mean (or value) with high probability (more than a certain threshold).

VC dimension

- The Vapnik–Chervonenkis (VC) dimension is a measure of the complexity of a machine learning model.
- It is a number that helps quantify the difficulty of learning from examples.
- The VC dimension is often used to guide the model selection process when developing machine learning applications.