Developer Circles

# AI Session 02: Intro to Deep Learning

PyTorch


Daniel Bridera
Data Science

Joaquín Barotto
Data Science

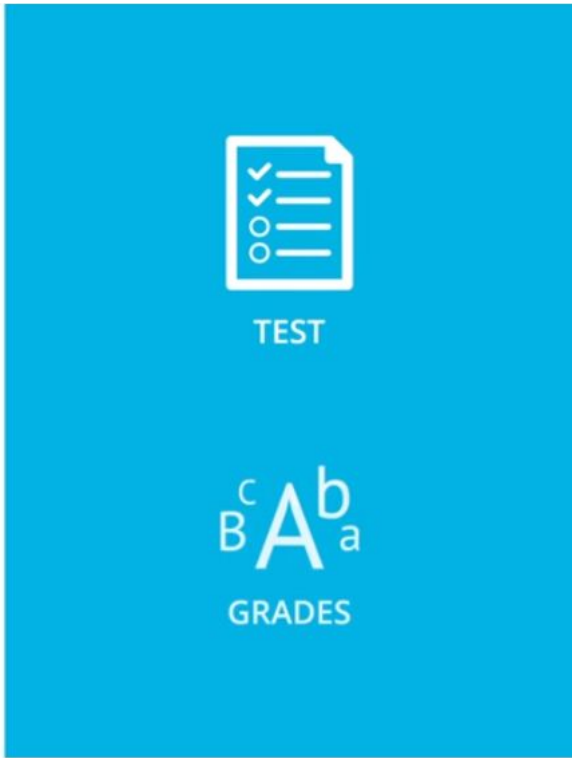

facebook for developers

@DataSmoke

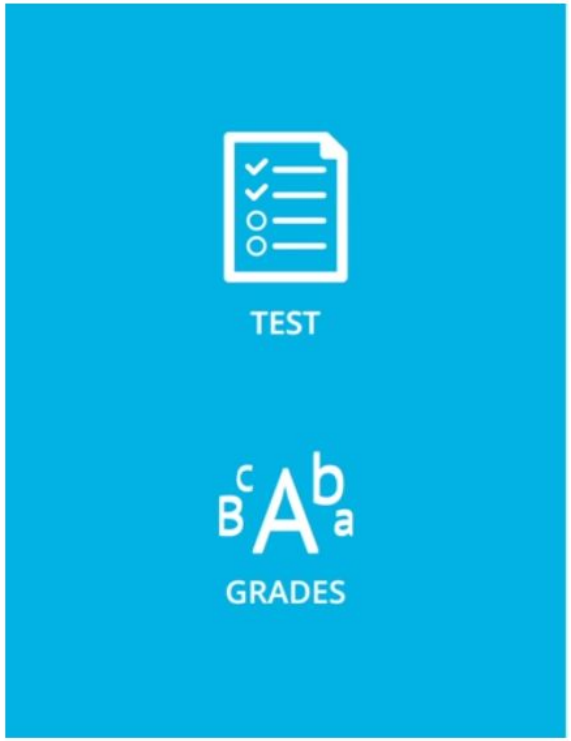
@JoaoDaBahia

# Classification Problems

Acceptance at a University



# Classification Problems

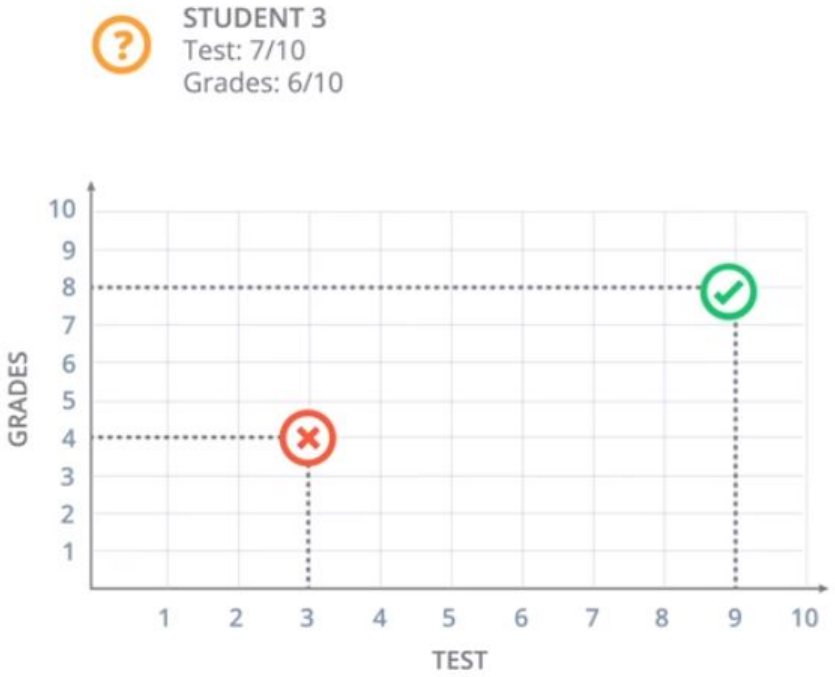



STUDENT 1
Test: 9/10
Grades: 8/10

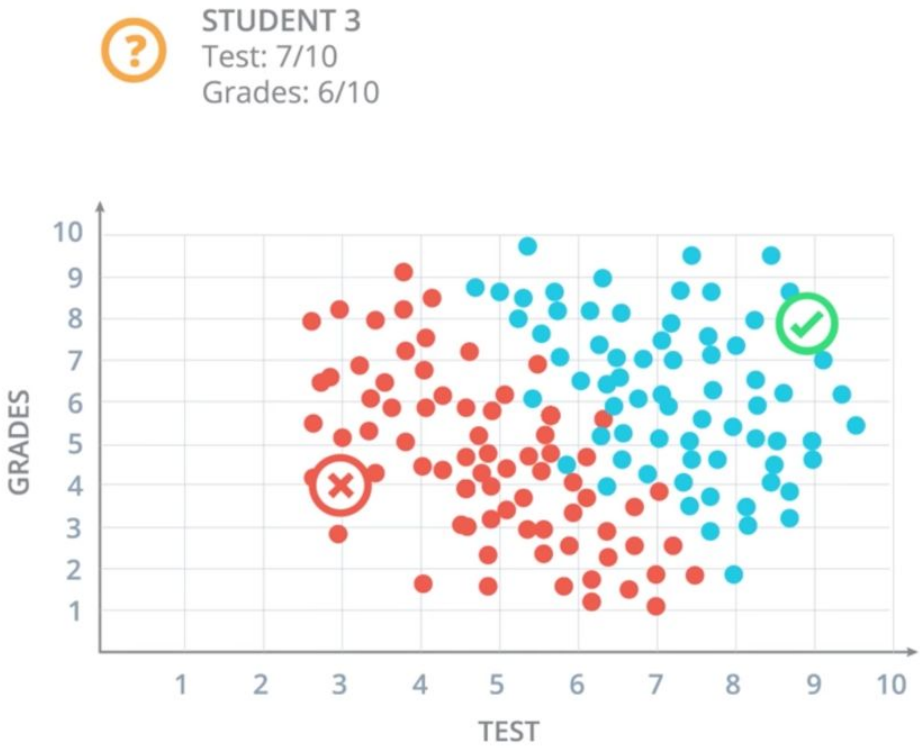
STUDENT 2
Test: 3/10
Grades: 4/10

STUDENT 3
Test: 7/10
Grades: 6/10

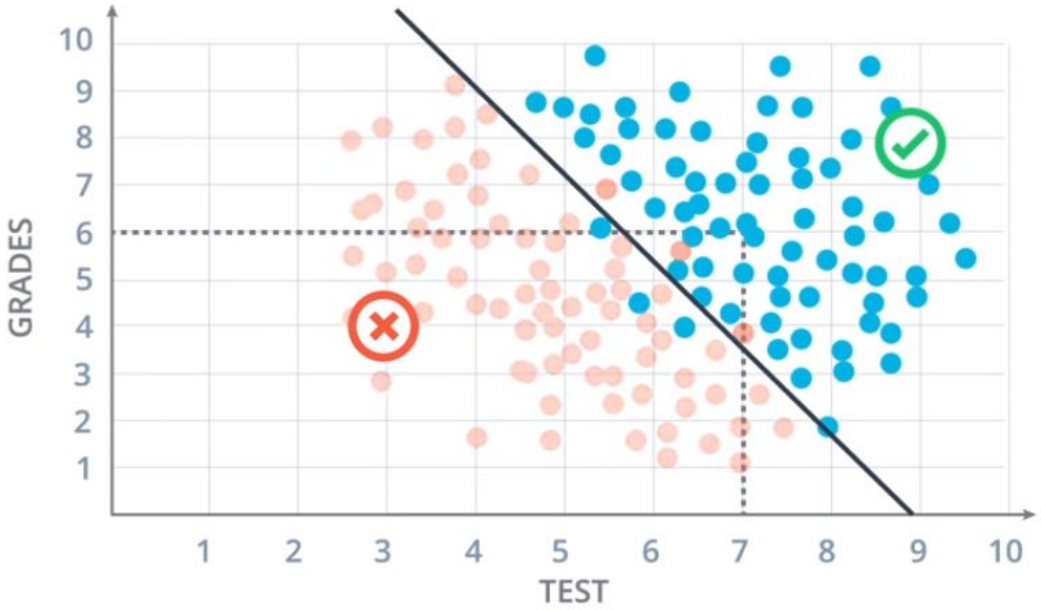
# Classification Problems



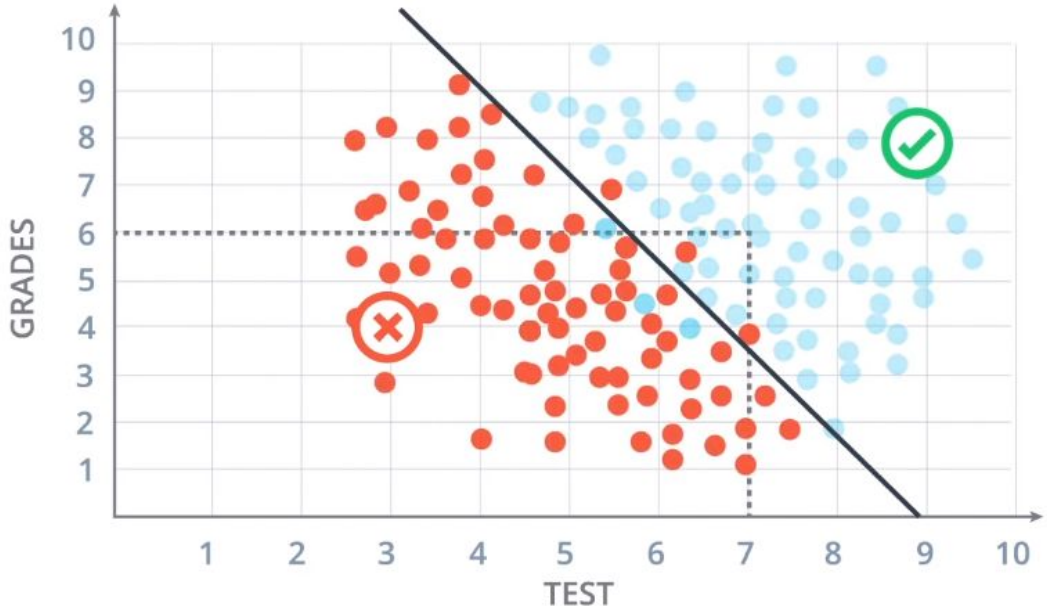
# Classification Problems



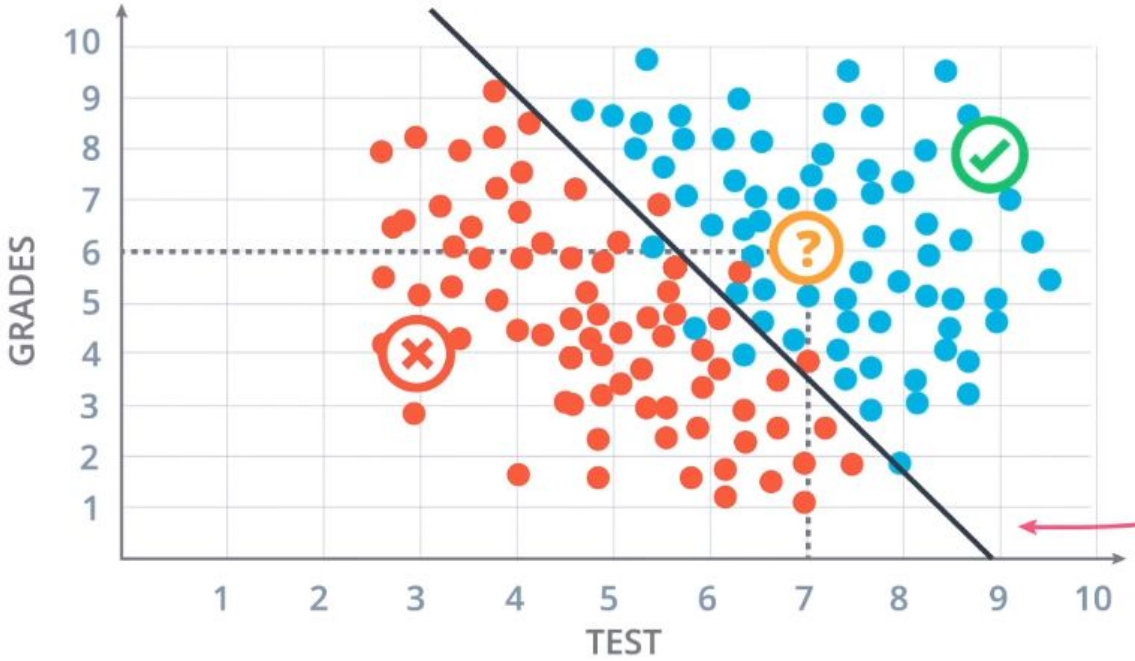
# Classification Problems



# Classification Problems



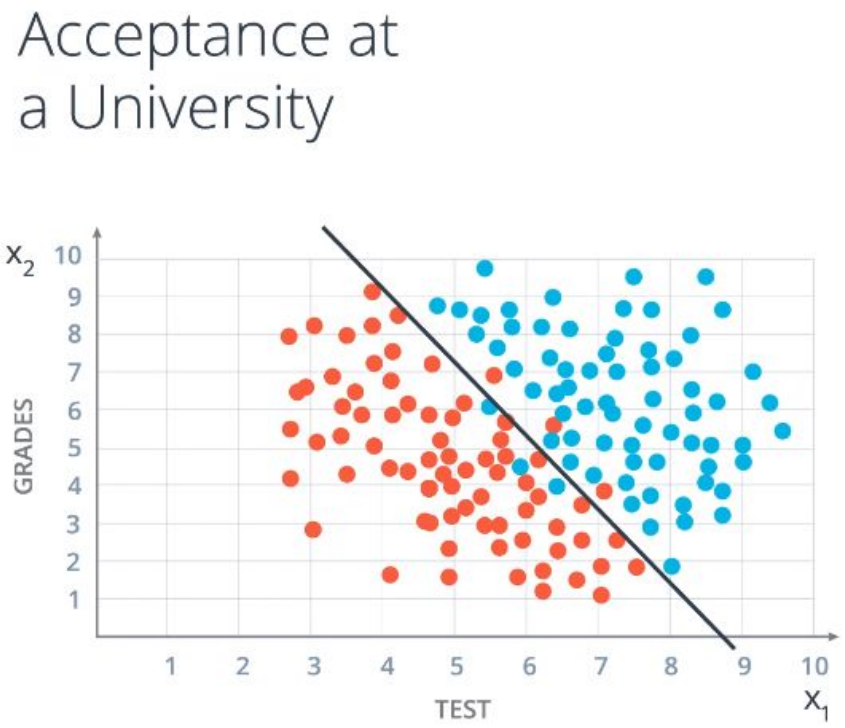
# Classification Problems




**QUESTION**

How do we find this line?

# Classification Problems

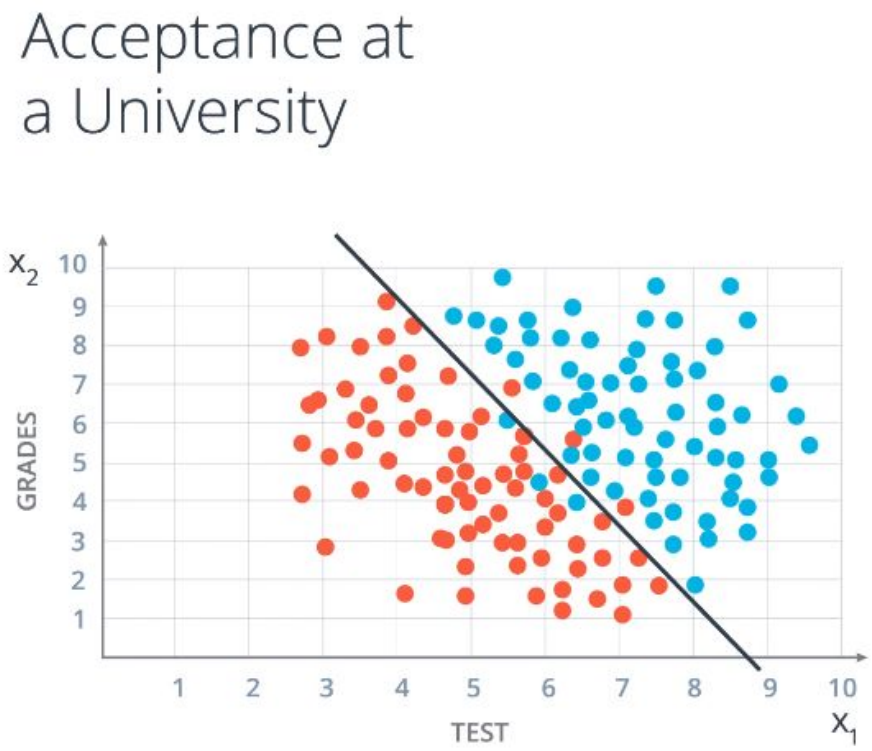



**BOUNDARY:**
**A LINE**
$2x_1 + x_2 - 18 = 0$

Score =
2*Test + Grades - 18

**PREDICTION:**
Score > 0: Accept
Score < 0: Reject

# Classification Problems




**BOUNDARY:**
**A LINE**

$w_1x_1 + w_2x_2 + b = 0$

$Wx + b = 0$

$W = (w_1, w_2)$

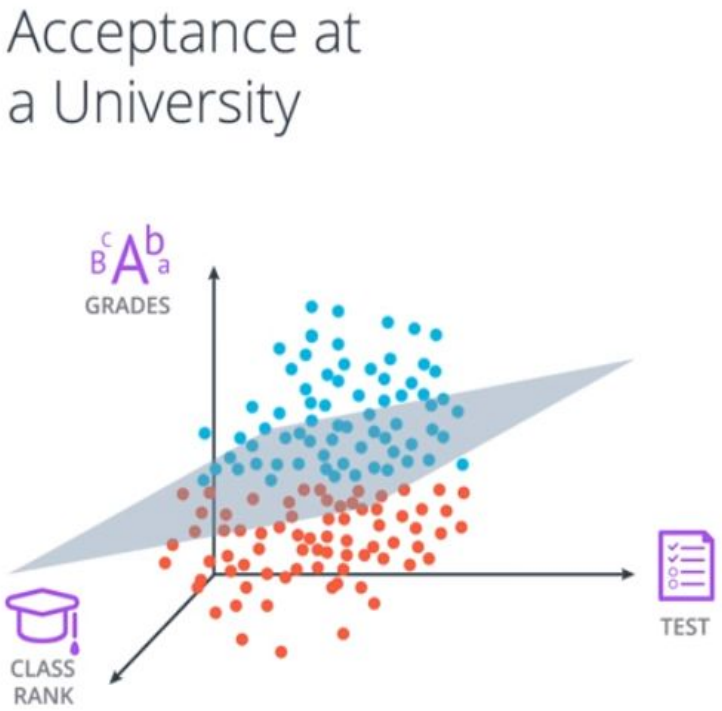
$x = (x_1, x_2)$

$y$ = label: 0 or 1

**PREDICTION:**

$$\hat{y} = \begin{cases} 1 \text{ if } Wx + b \geq 0 \\ 0 \text{ if } Wx + b < 0 \end{cases}$$

# Classification Problems




**BOUNDARY:**

**A PLANE**

$w_1x_1 + w_2x_2 + w_3x_3 + b = 0$

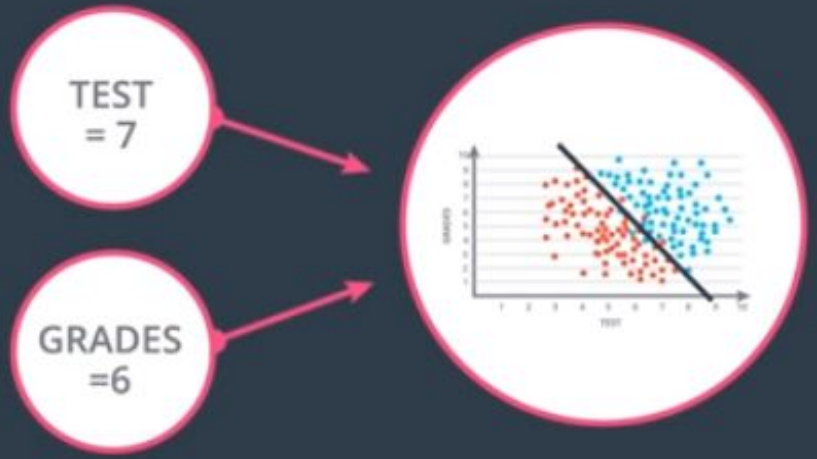
$Wx + b = 0$

**PREDICTION:**

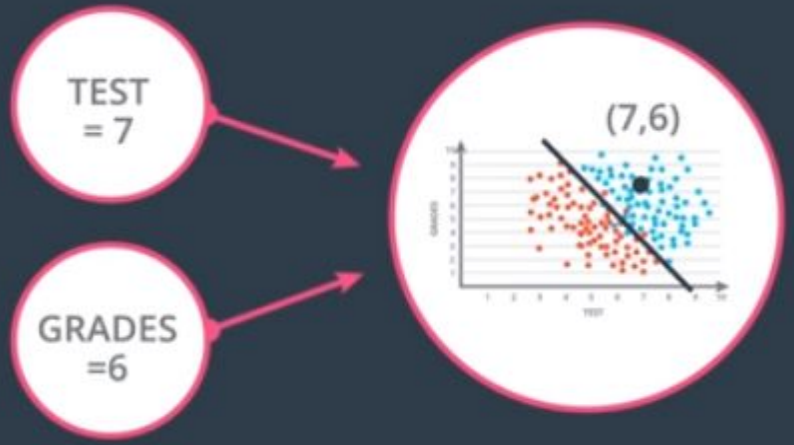
$$\hat{y} = \begin{cases} 1 \text{ if } Wx + b \geq 0 \\ 0 \text{ if } Wx + b < 0 \end{cases}$$

# Classification Problems

# Perceptron



# Perceptron

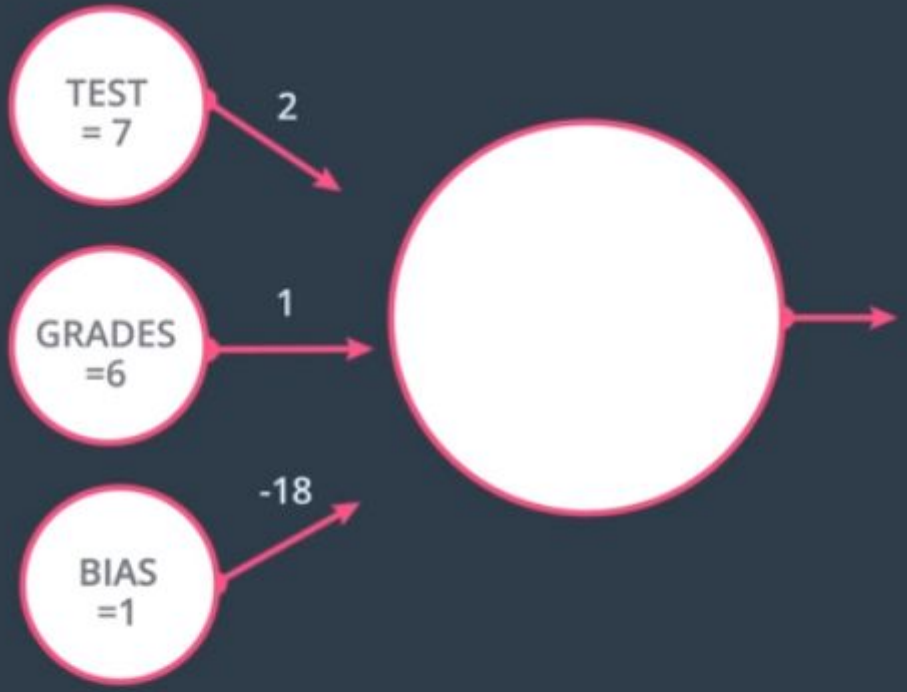


# Perceptron


Score=
2*Test + 1*Grades -18

**PREDICTION:**
Score ≥ 0 Accept
Score < 0 Reject

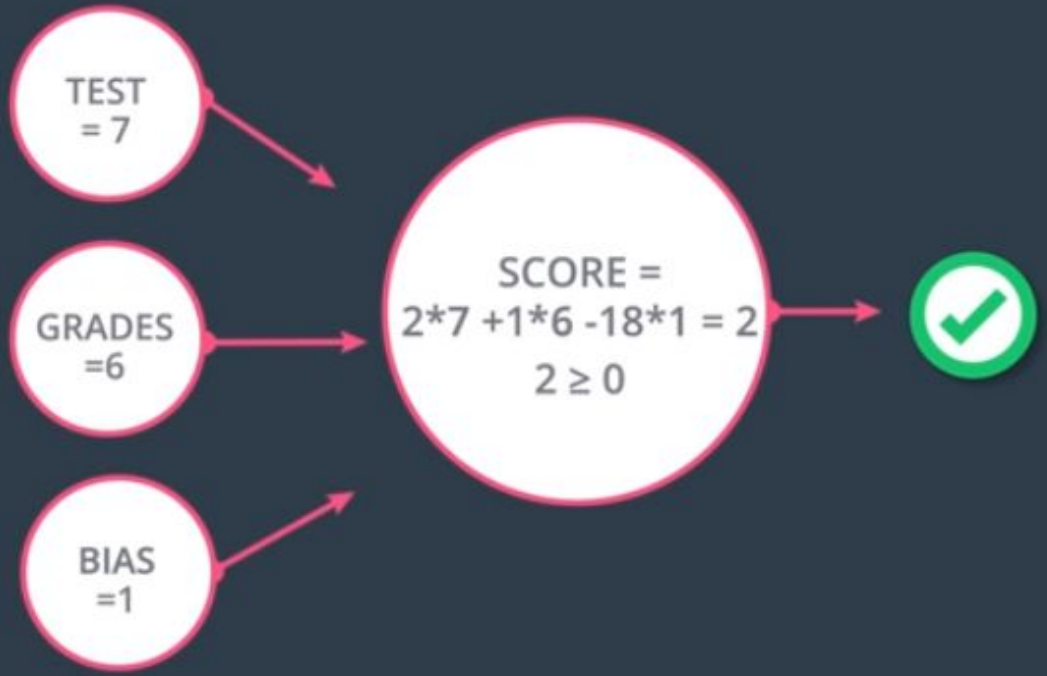
# Perceptron




Score=
2*Test + 1*Grades -18

**PREDICTION:**
Score ≥ 0 Accept
Score < 0 Reject

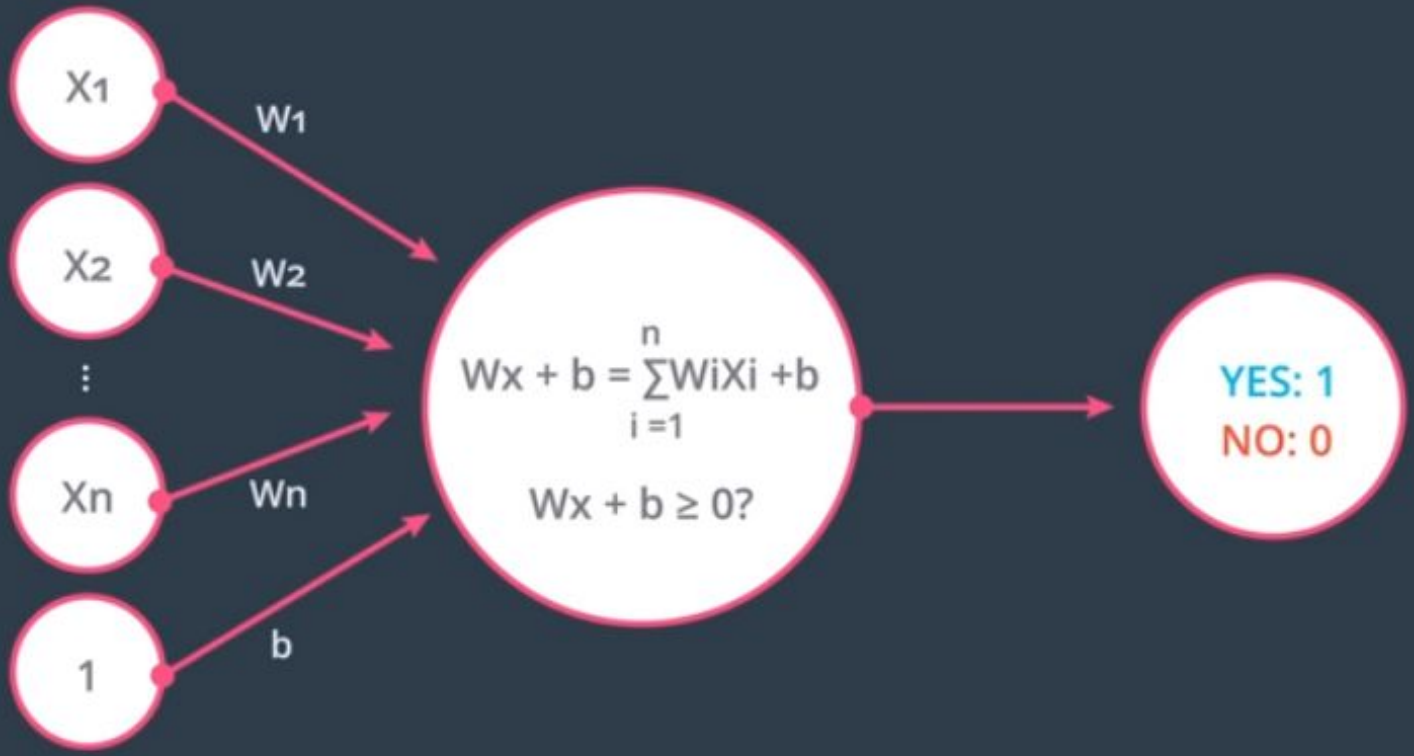
# Perceptron



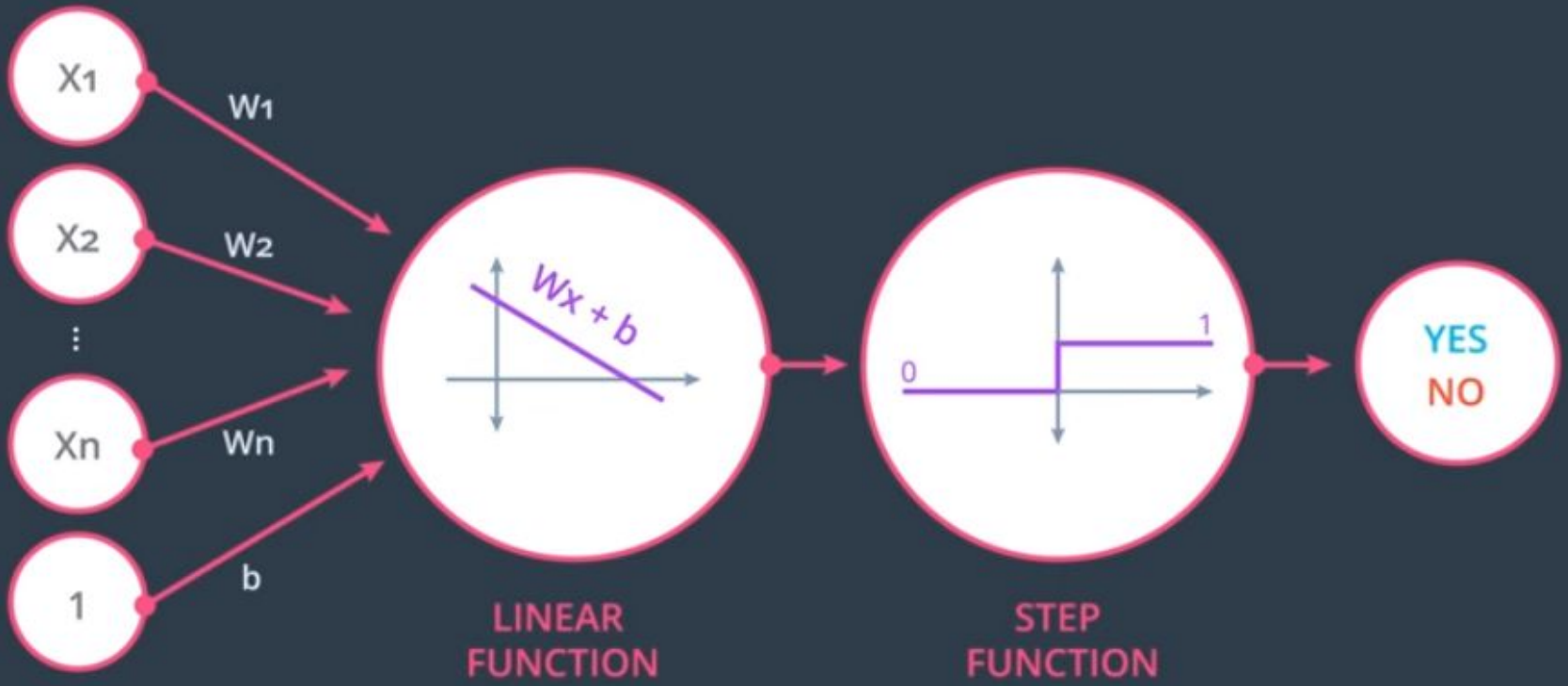

Score=
2*Test + 1*Grades -18

**PREDICTION:**
Score ≥ 0 Accept
Score < 0 Reject

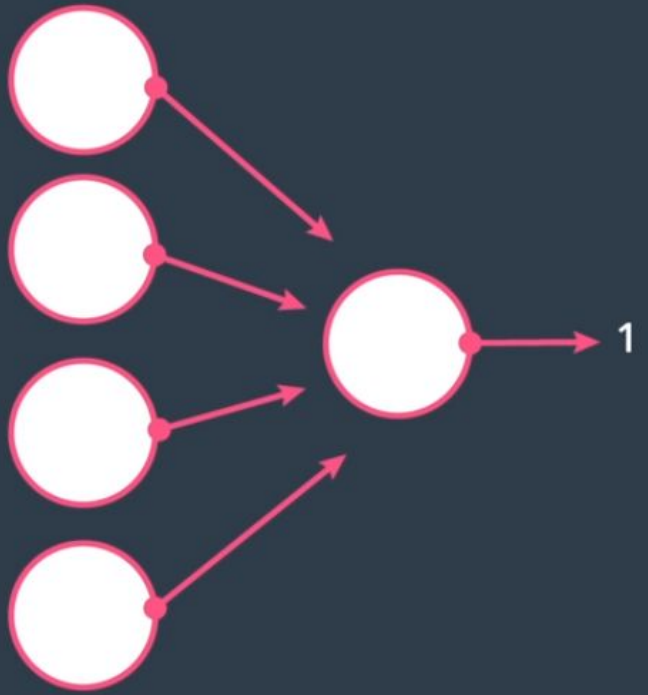
# Perceptron

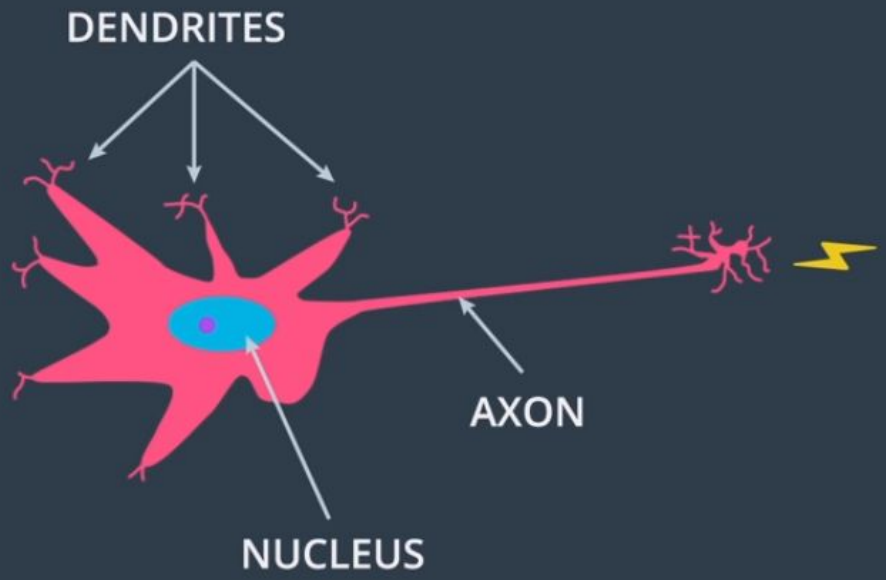


# Perceptron


X1
W1
X2
W2
Xn
Wn
1
b
Wx + b
LINEAR FUNCTION
0
1
STEP FUNCTION
YES
NO

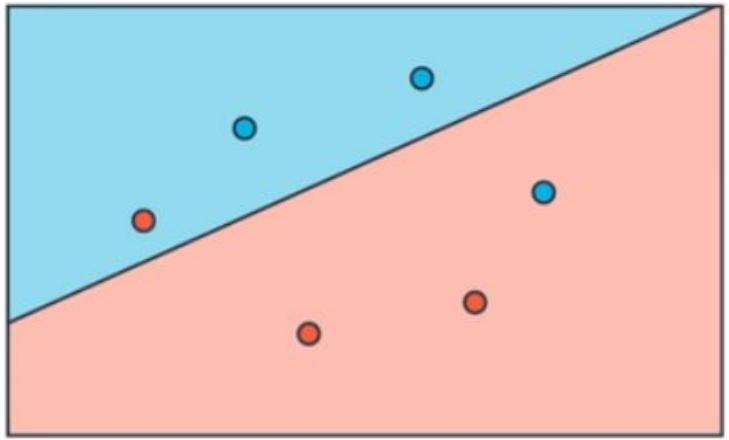
# Perceptron



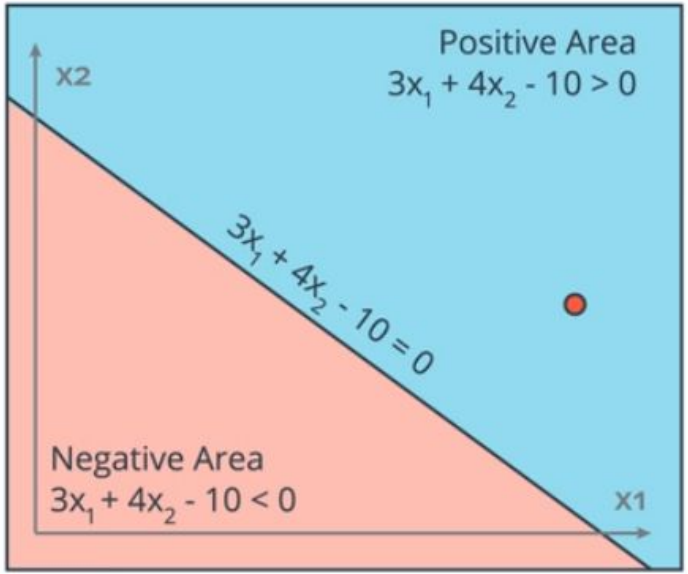



# Perceptron

Goal: Split Data



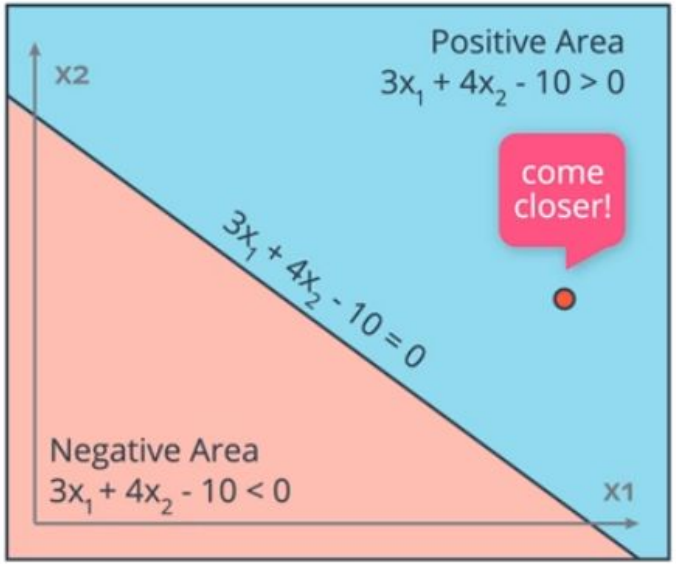
# Perceptron




**LINE:** $3x_1 + 4x_2 - 10 = 0$

**POINT:** (4,5)

# Perceptron

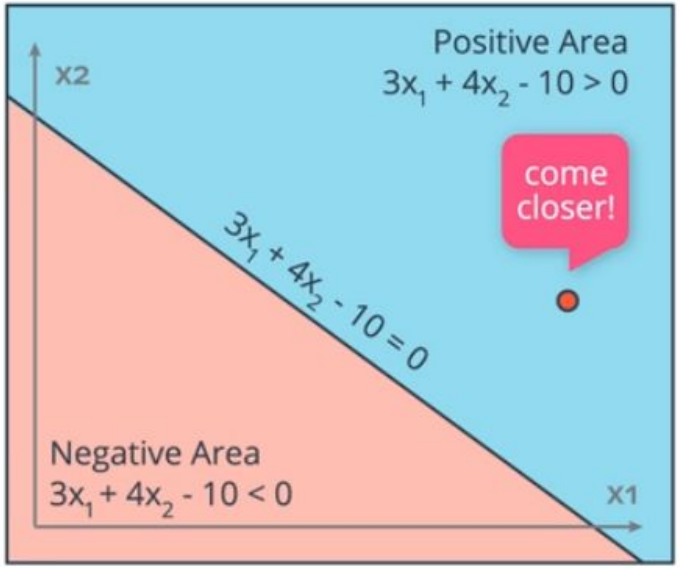



**LINE:** $3x_1 + 4x_2 - 10 = 0$

**POINT:** (4,5)

| | | | |
|---|---|---|---|
| | 3 | 4 | -10 |
| – | 4 | 5 | 1 |
| | -1 | -1 | -11 |

# Perceptron




**LINE:** $3x_1 + 4x_2 - 10 = 0$

**POINT:** (4,5)

**LEARNING RATE:** 0.1

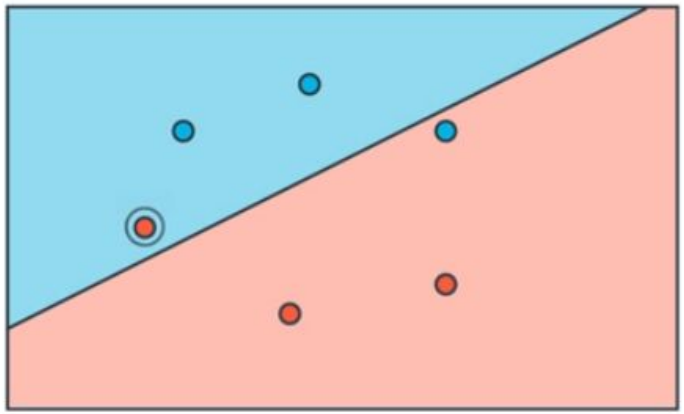
| | | | |
|---|---|---|---|
| | 3 | 4 | -10 |
| − | 0.4 | 0.5 | 0.1 |
| | 2.6 | 3.5 | -10.1 |

**NEW LINE**

$2.6x_1 + 3.5x_2 - 10.1 = 0$

# Perceptron

## Perceptron Algorithm



1. Start with random weights: $w_1, \ldots, w_n, b$

2. For every misclassified point $(x_1,\ldots,x_n)$:

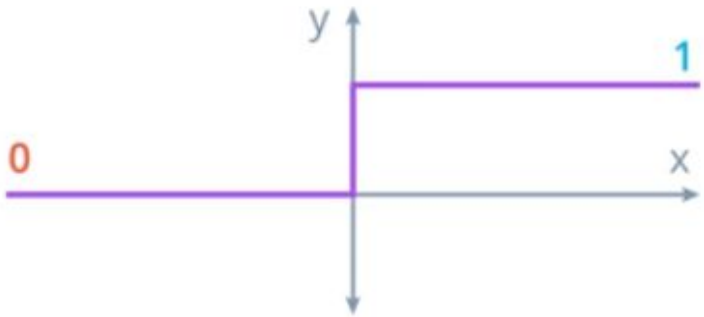
   2.1. If prediction = 0:

   - For i = 1 ...n
     - Change $w_i + \alpha x_i$
   - Change b to b + α

   2.2. If prediction = 1:

   - For i = 1 ...n
     - Change $w_i - \alpha x_i$
   - Change b to b - α

# Activation Functions



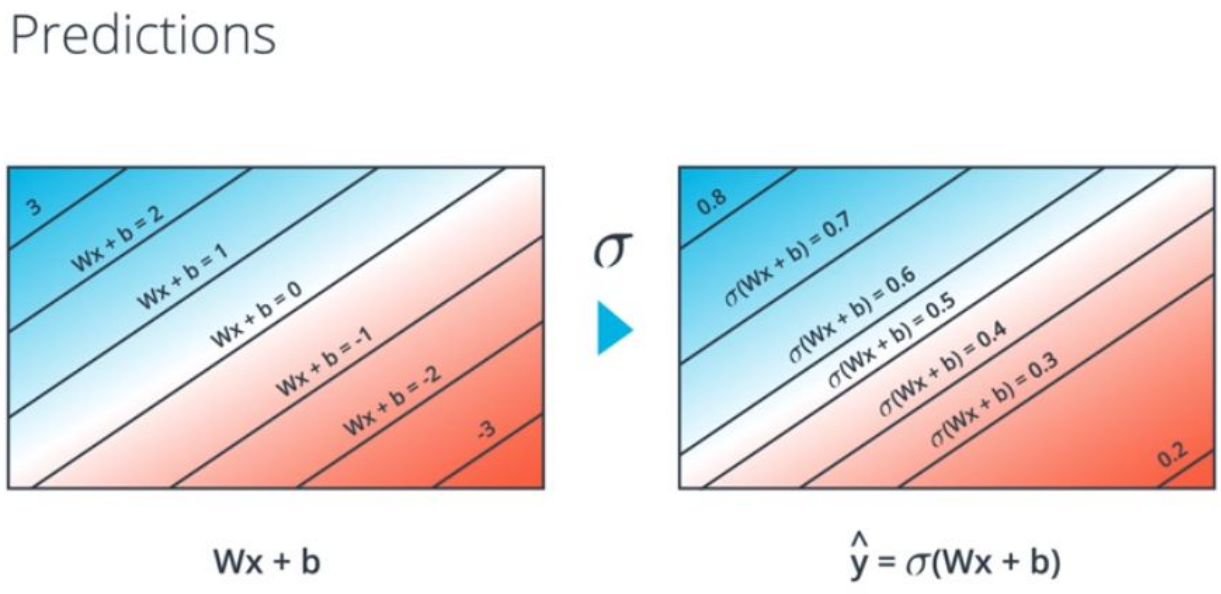

**DISCRETE:**
Step Function

$$y = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

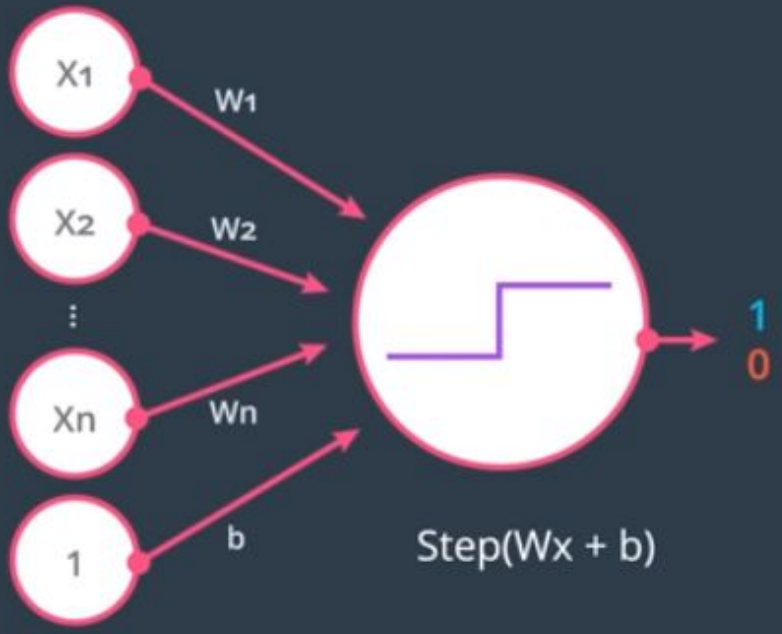

**CONTINUOUS:**
Sigmoid Function

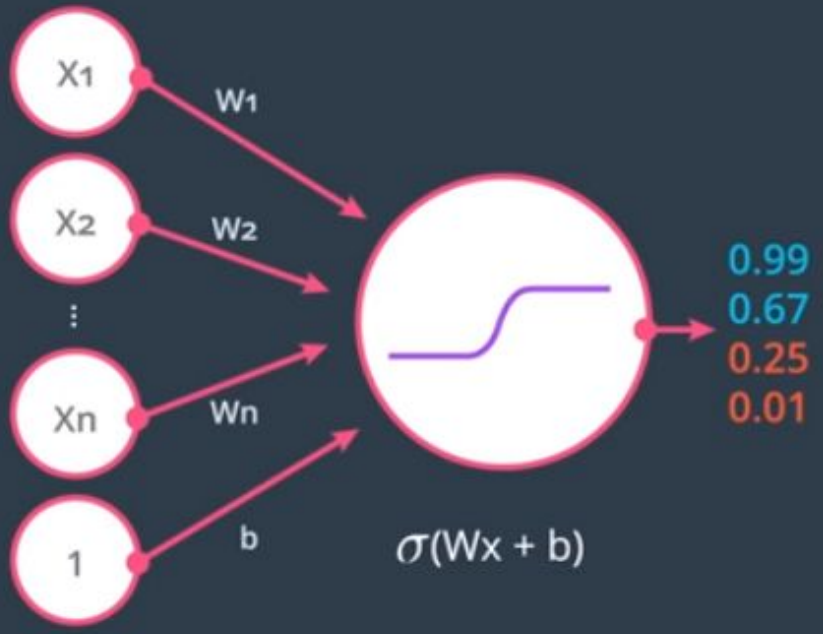
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

# Activation Functions

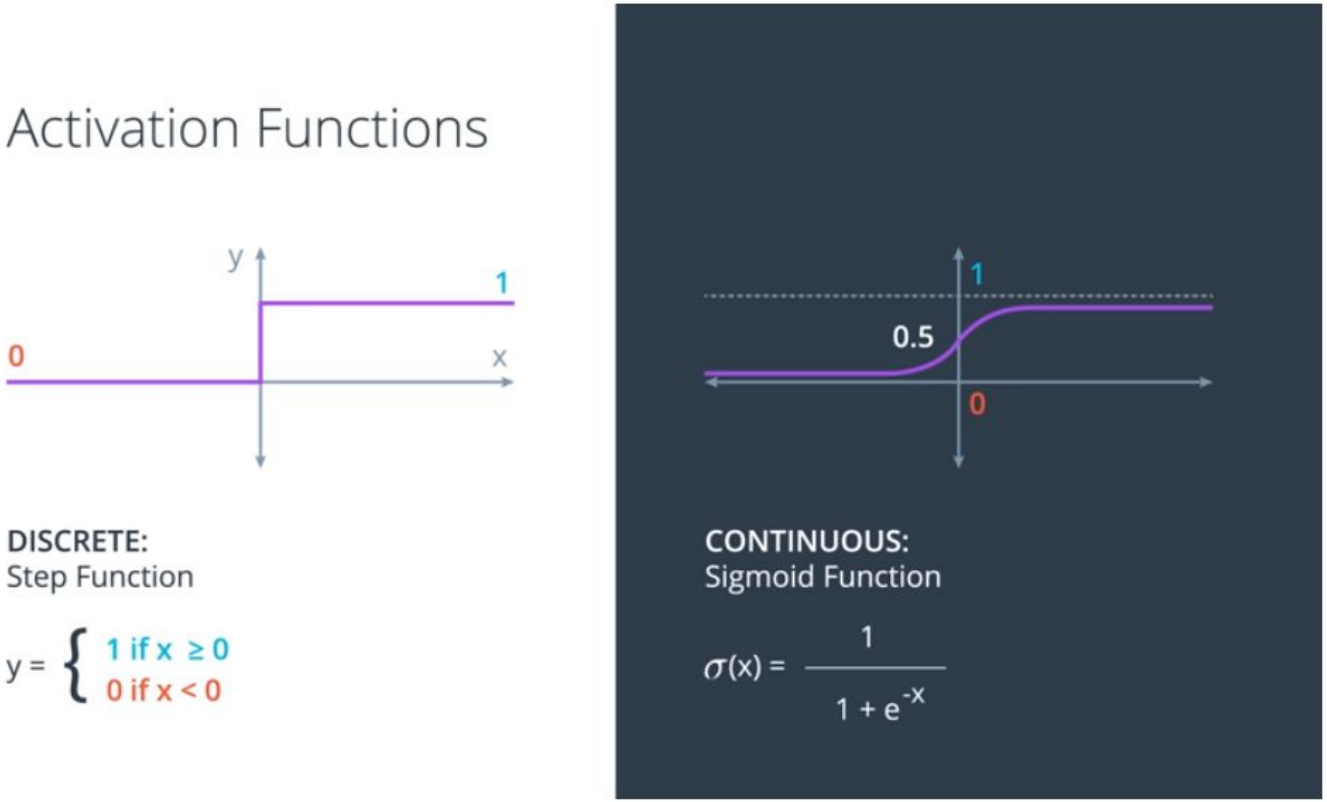


# Perceptron


X1
W1
X2
W2
Xn
Wn
1
b
Step(Wx + b)
1
0


X1
W1
X2
W2
Xn
Wn
1
b
σ(Wx + b)
0.99
0.67
0.25
0.01

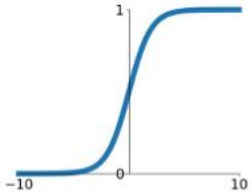
# Activation Functions



# Activation Functions

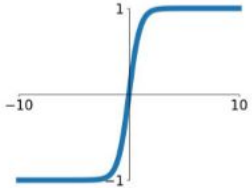
**Sigmoid**

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



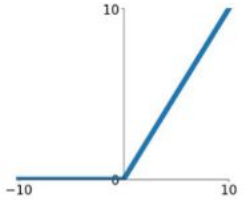

**tanh**

$$\tanh(x)$$



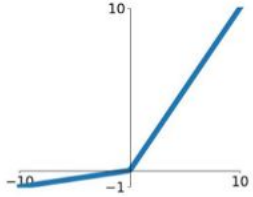

**ReLU**

$$\max(0, x)$$




**Leaky ReLU**

$$\max(0.1x, x)$$

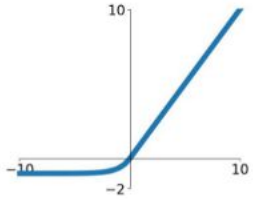



**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

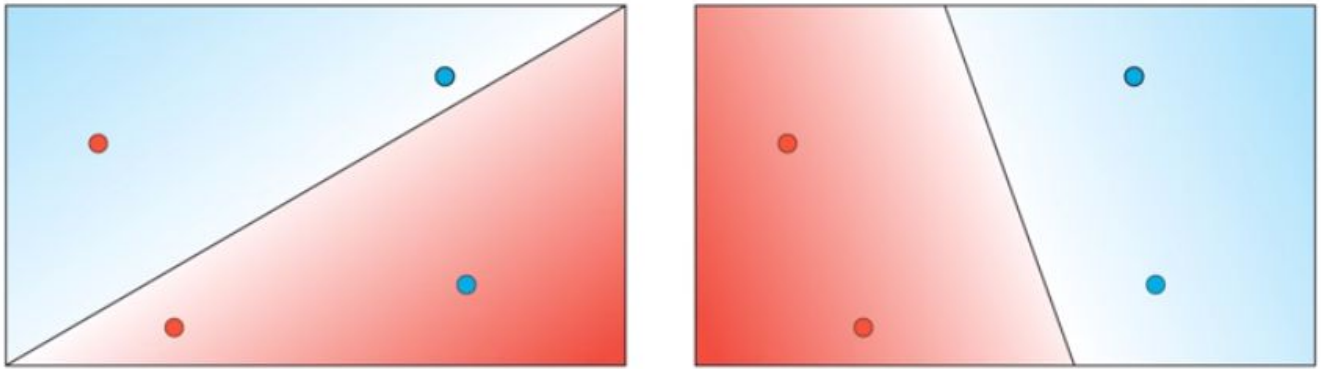
**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

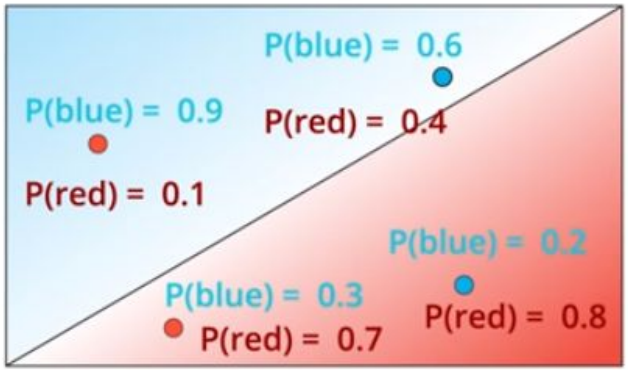



Okay gang, let's see what DATA SCIENCE really is.
STATISTICS
ADDTEXT.COM

# Probability



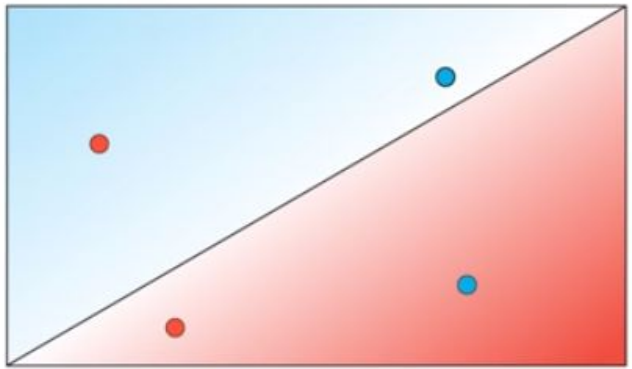
# Probability




$\hat{y} = \sigma(Wx+b)$

$P(\text{blue}) = \sigma(Wx+b)$

# Probability



$\hat{y} = \sigma(Wx+b)$

$P(\text{blue}) = \sigma(Wx+b)$

P(red) = 0.1

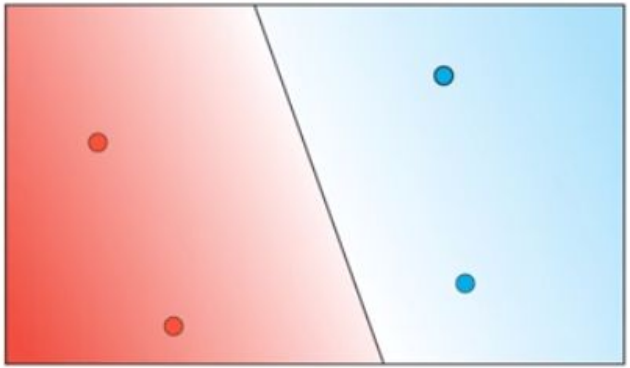
P(blue) = 0.6

P(red) = 0.7

x P(blue) = 0.2

P(all) = 0.0084

# Probability



0.6 * 0.2 * 0.1 * 0.7 = 0.0084

0.7 * 0.9 * 0.8 * 0.6 = 0.3024

**Maximum Likelihood**

# Products

**0.6 * 0.2 * 0.1 * 0.7 = 0.0084**

ln(0.6) + ln(0.2) + ln(0.1) + ln(0.7)

-0.51 -1.61 -2.3 -0.36

-ln(0.6) - ln(0.2) - ln(0.1) - ln(0.7) = 4.8

0.51 1.61 2.3 0.36

**0.7 * 0.9 * 0.8 * 0.6 = 0.3024**

ln(0.7) + ln(0.9) + ln(0.8) + ln(0.6)

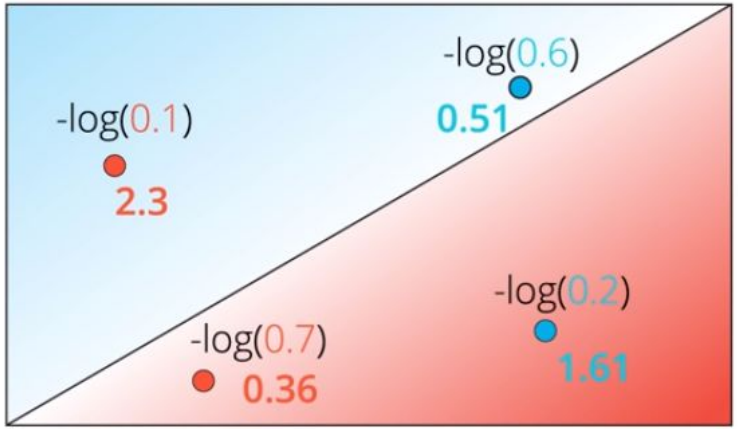
-0.36 -0.1 -.22 -0.51

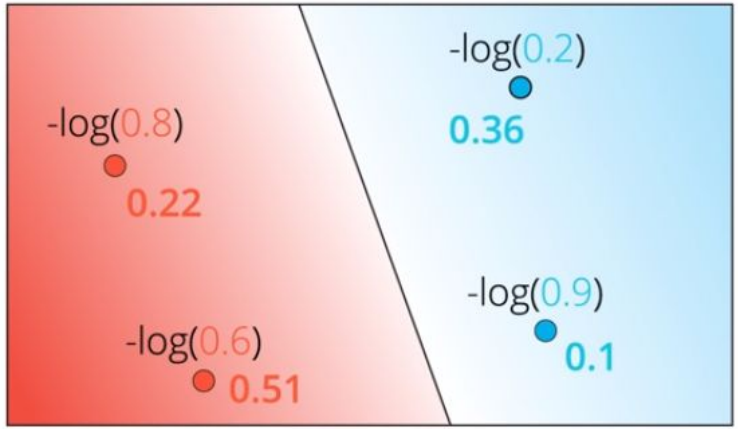
-ln(0.7) - ln(0.9) -ln(0.8) -ln(0.6) = 1.2

0.36 0.1 .22 0.51

**Cross Entropy**

# Cross Entropy




**0.6 * 0.2 * 0.1 * 0.7 = 0.0084**

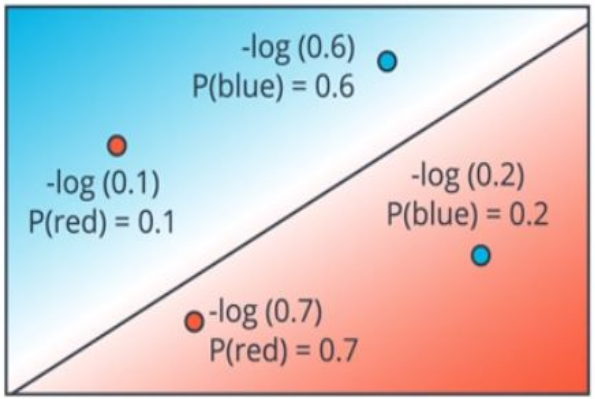
-log(0.6) - log(0.2) - log(0.1) - log(0.7) = 4.8

**0.7 * 0.9 * 0.8 * 0.6 = 0.3024**

-log(0.7) - log(0.9) - log(0.8) - log(0.6) = 1.2

**Goal: Minimize the Cross Entropy**

# Error Function




$-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7) = 4.8$

0.51 1.61 2.3 0.36

If $y = 1$

$P(\text{blue}) = \hat{y}$

$\text{Error} = -\ln(\hat{y})$

If $y = 0$

$P(\text{red}) = 1 - P(\text{blue}) = 1 - \hat{y}$

$\text{Error} = -\ln(1 - \hat{y})$

$\text{Error} = -(1-y)(\ln(1-\hat{y})) - y\ln(\hat{y})$

$$\text{Error Function} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)(\ln(1-\hat{y}_i)) + y_i\ln(\hat{y}_i)$$

# Error Function

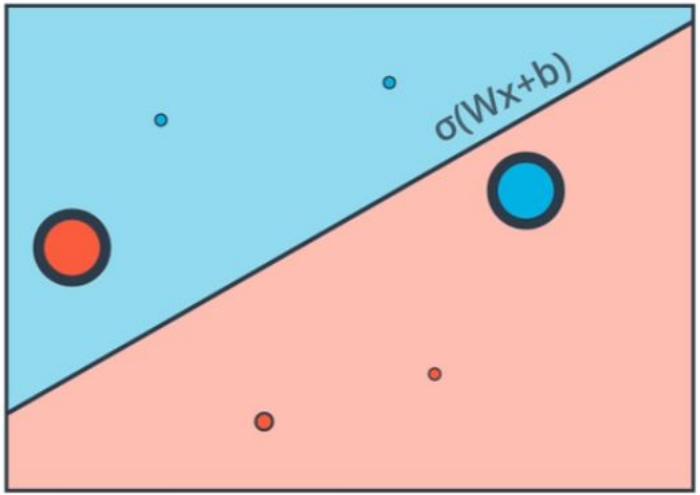
$$\text{Error Function} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)(\ln(1-\hat{y}_i)) + y_i \ln(\hat{y}_i)$$

$$E(W,b) = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)(\ln(1-\sigma(Wx^{(i)}+b)) + y_i \ln(\sigma(Wx^{(i)}+b)$$

**GOAL**

Minimize Error Function

# Gradient Descent Algorithm




1. Start with random weights:

   $w_1, \ldots, w_n, b$

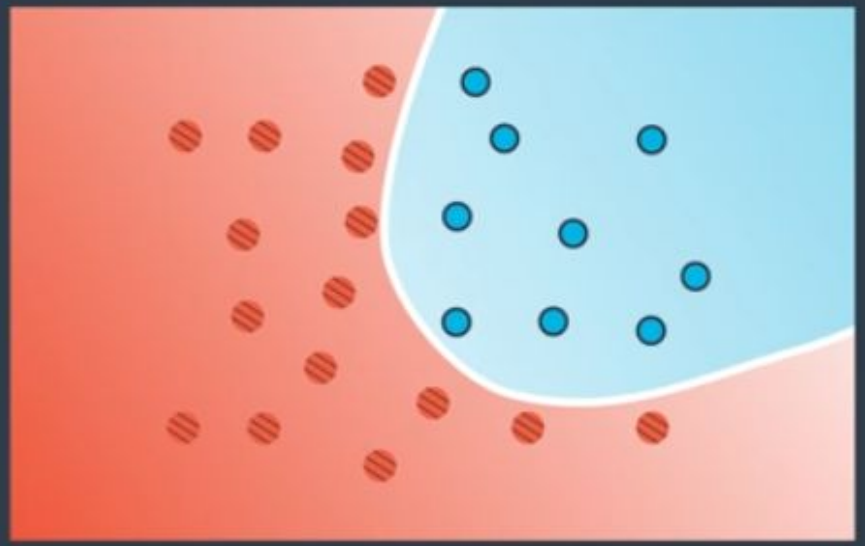
2. For every point $(x_1, \ldots, x_n)$:

   2.1. For i =1 . . . n

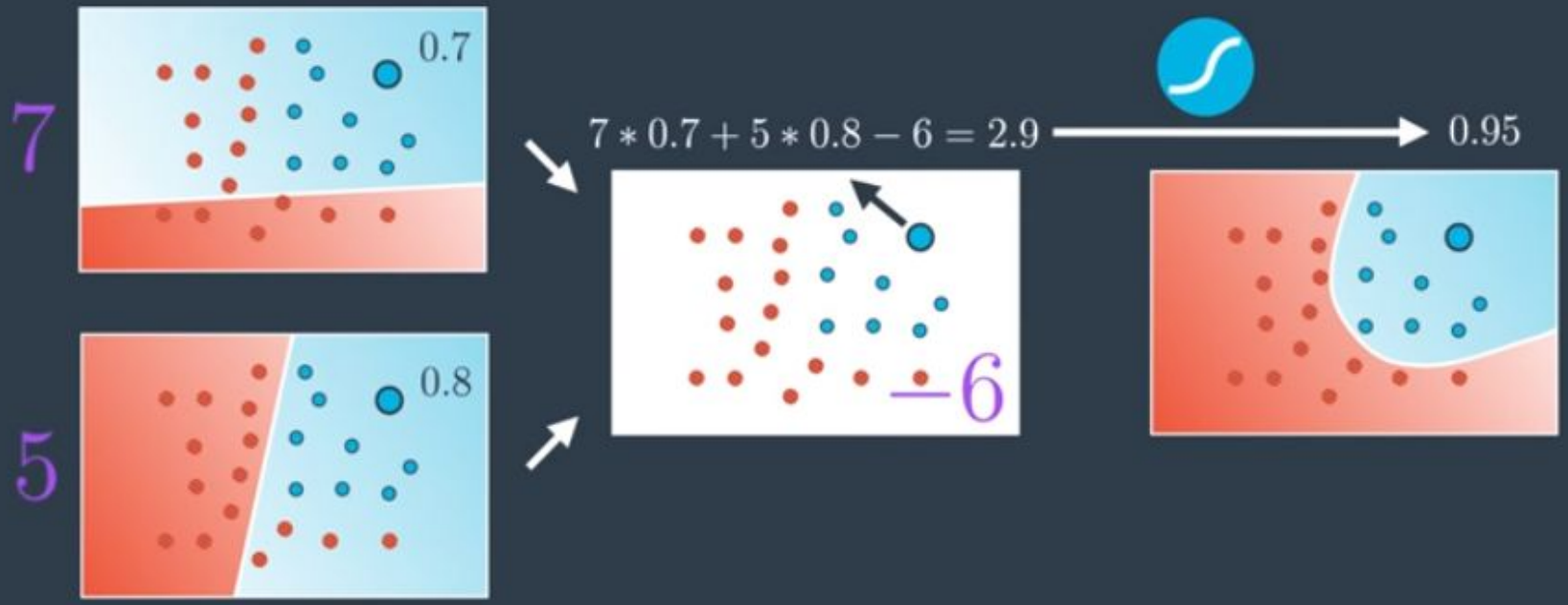
      2.1.1. Update $w_i' \longleftarrow w_i - \alpha (\hat{y}-y)x_i$

      2.1.2. Update $b' \longleftarrow b - \alpha (\hat{y}-y)$

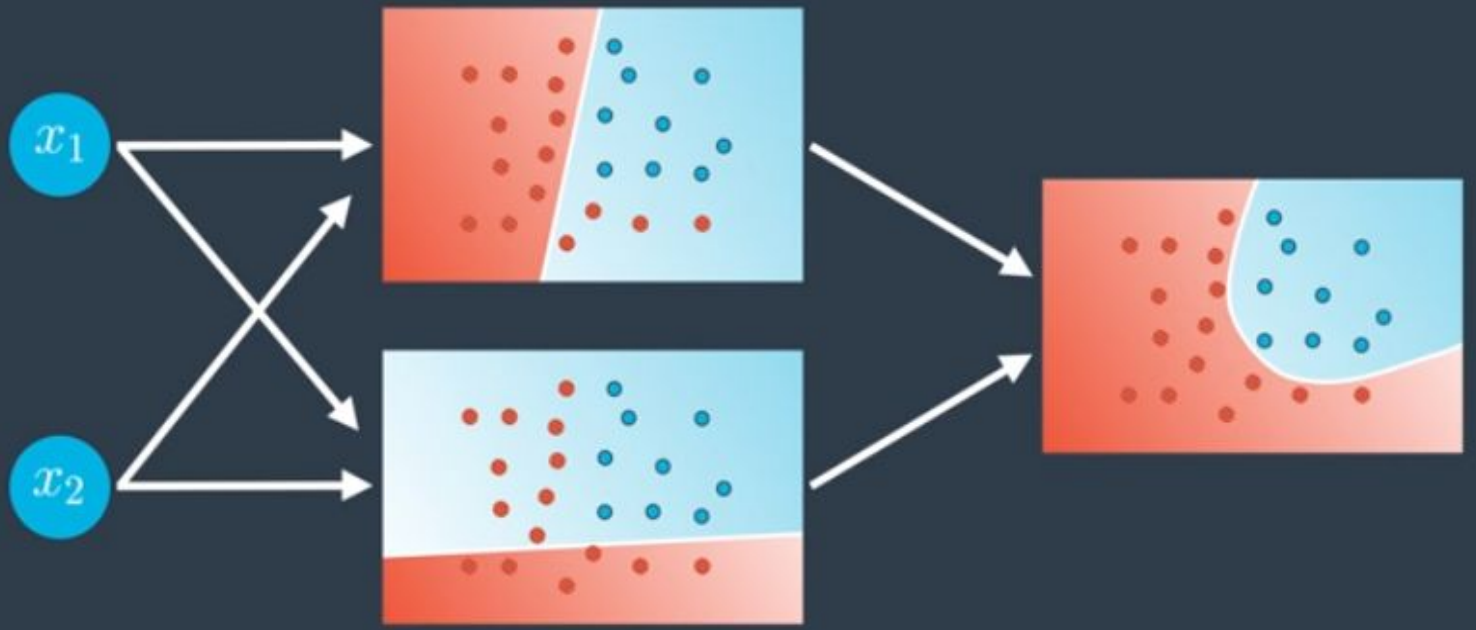
# Non-Linear Regions

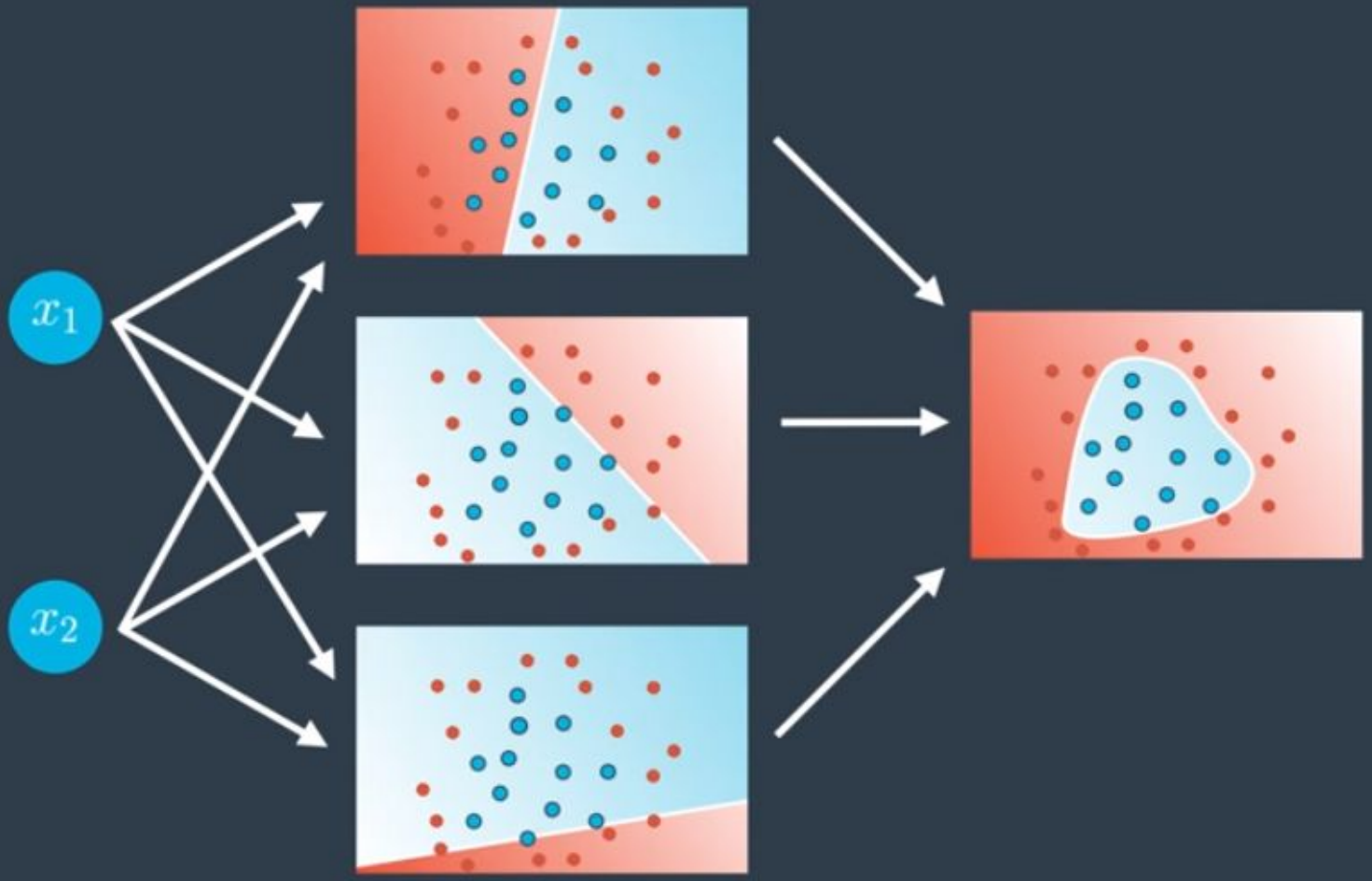


# Neural Network

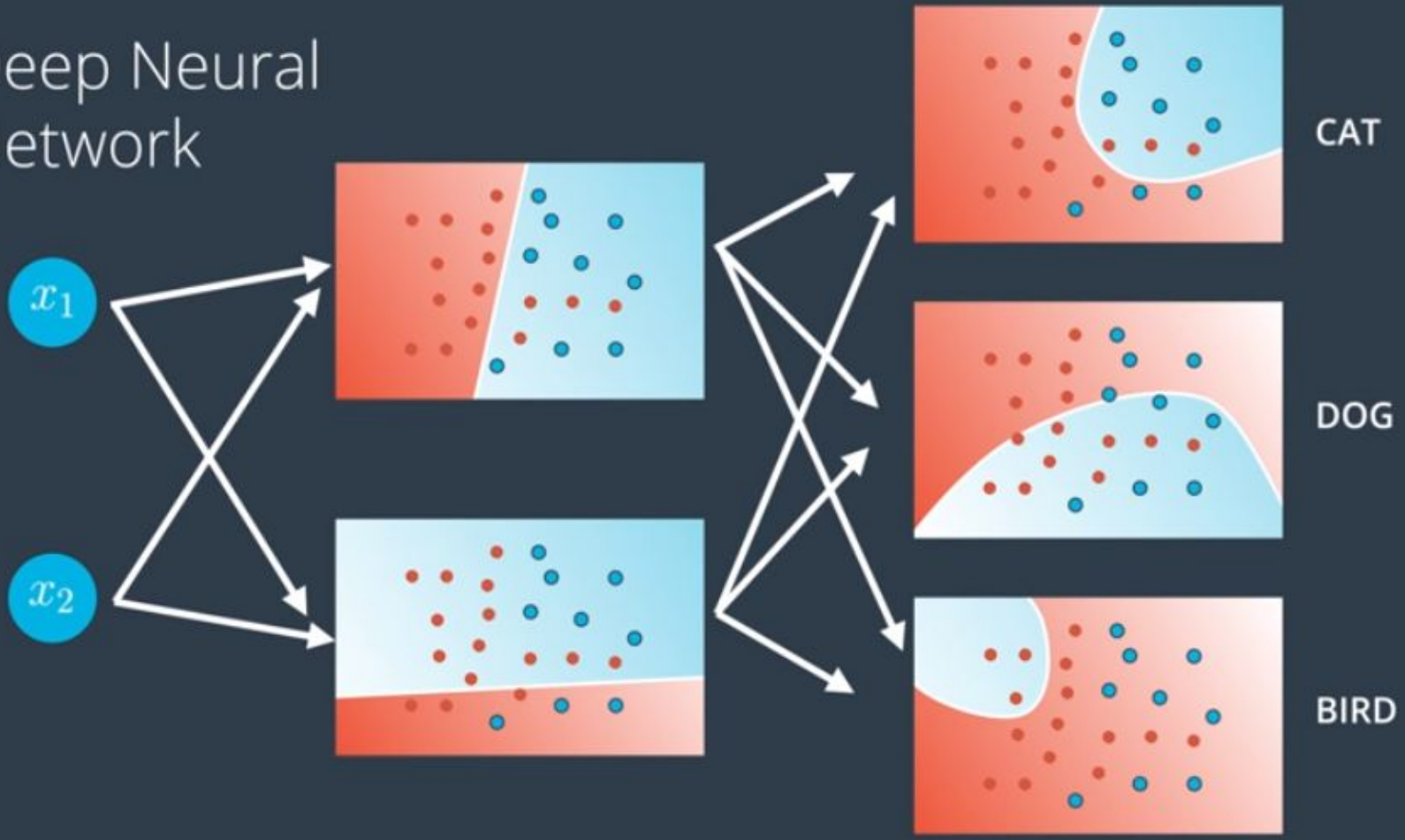


# Neural Network

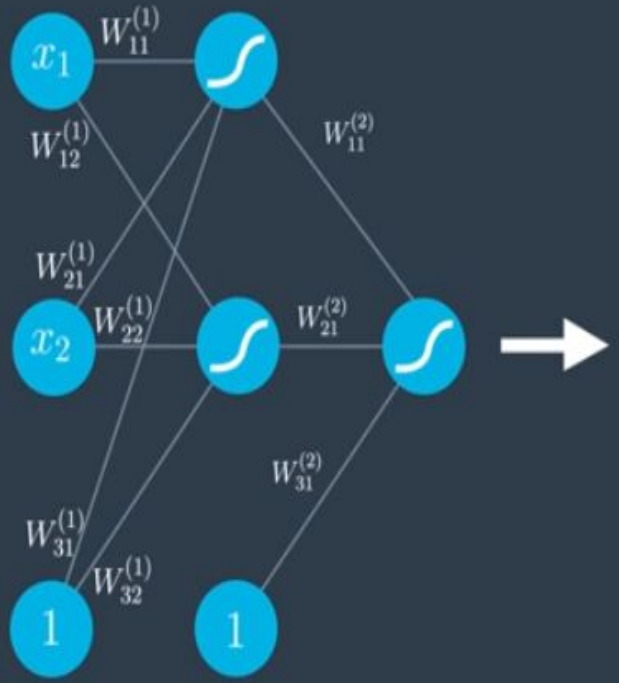



$x_1$
$x_2$

# Deep Neural Network

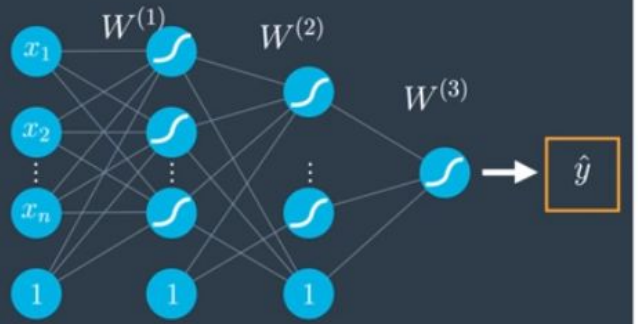


# Feedforward




$$\hat{y} = \sigma \begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix} \sigma \begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

# Multi-layer Perceptron




**PREDICTION**

$$\hat{y} = \sigma \circ W^{(3)} \circ \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

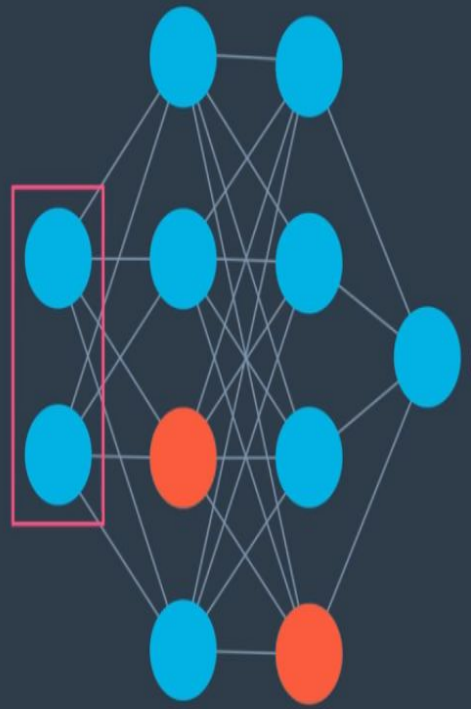
## Backpropagation

Now, we're ready to get our hands into training a neural network. For this, we'll use the method known as **backpropagation**. In a nutshell, backpropagation will consist of:

- Doing a feedforward operation.
- Comparing the output of the model with the desired output.
- Calculating the error.
- Running the feedforward operation backwards (backpropagation) to spread the error to each of the weights.
- Use this to update the weights, and get a better model.
- Continue this until we have a model that is good.

# When your network seems to be overfitting..

# DROPOUT



EPOCH 1

EPOCH 2

Joaco