EPISEN – ING3. SI

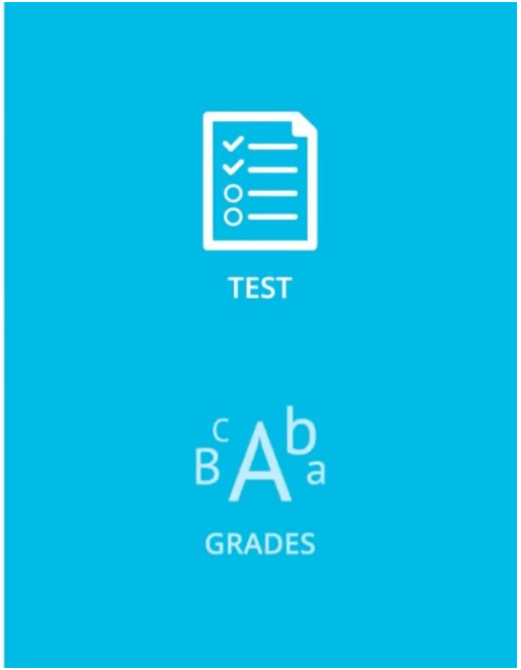
Machine Learning

Abdallah EL HIDALI
Tech Lead Sita For Aircraft
abdallah.el-hidali@sita.aero

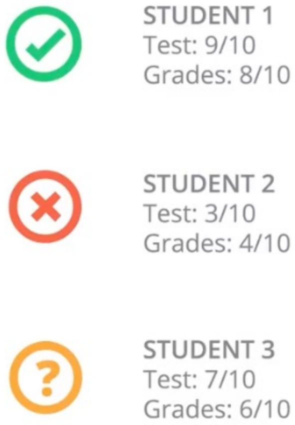
EPISEN
2024/2025

III. La classification

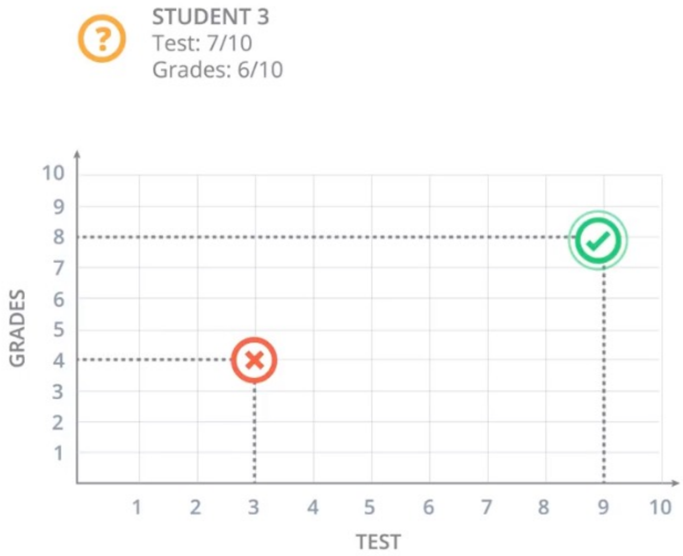
La classification - Introduction



La classification - Introduction

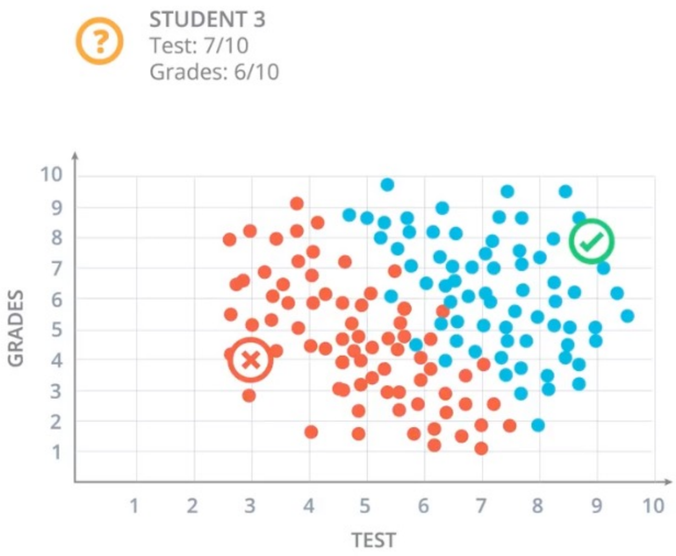


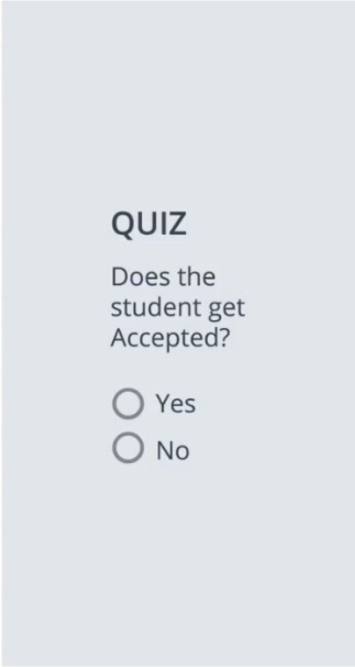
La classification - Introduction



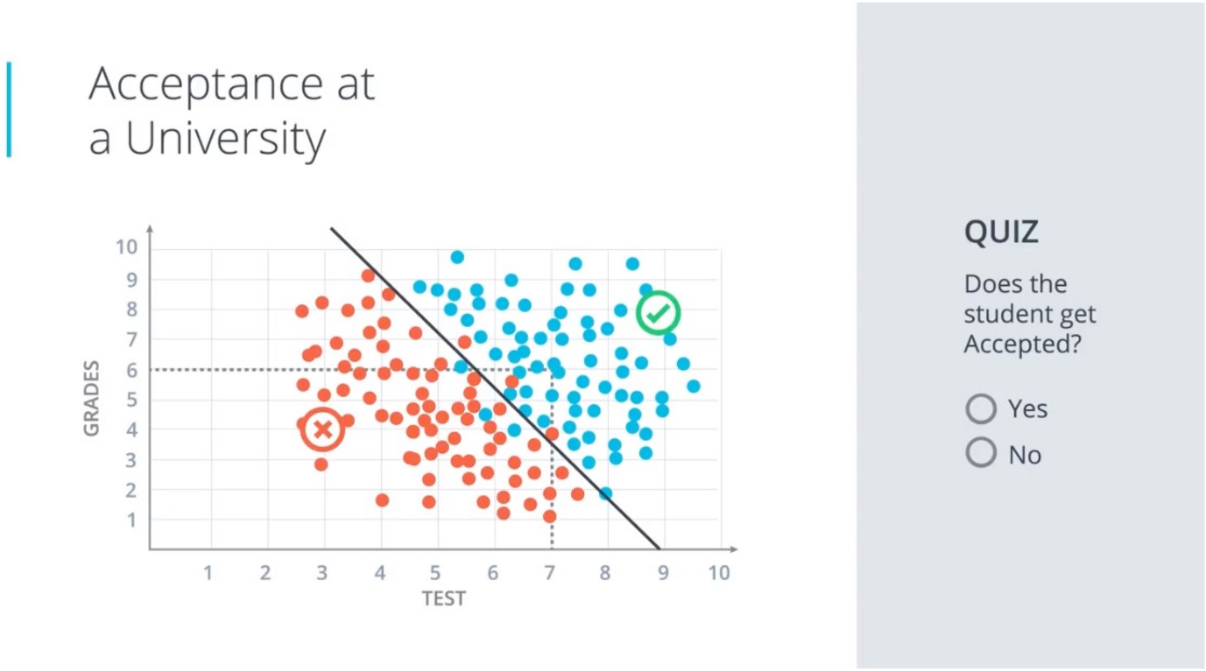
On analyse ensuite les données historiques

La classification - Introduction





La classification - Introduction



La classification - Introduction

Dans notre modèle de classification binaire (régression logistique) :

- La droite représente la frontière de décision.
- Les points au-dessus de la droite sont classés comme positifs (étudiants acceptés).
- Les points en dessous de la droite sont classés comme négatifs (étudiants rejetés).

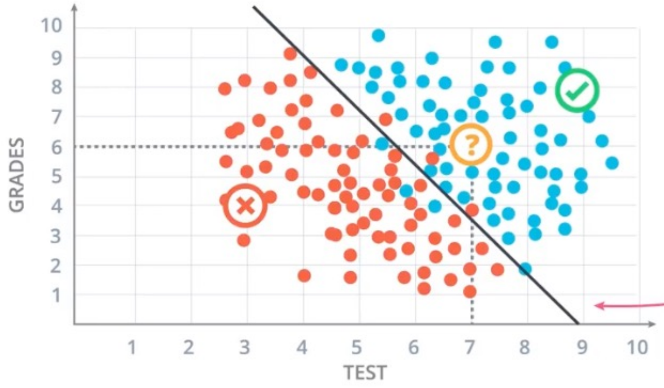
Cette droite est notre modèle de décision. Elle détermine si un étudiant est accepté ou non en fonction de sa position dans l'espace.

La distance d'un point à la droite indique le degré de confiance du modèle dans sa décision.

La classification - Introduction

La classification - Introduction

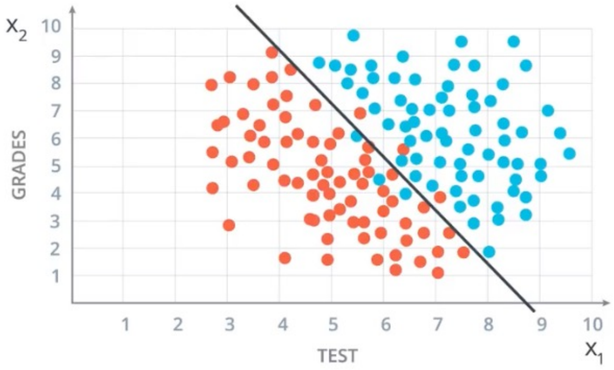
Question



Comment trouver cette ligne ?

La classification - Introduction

Acceptance at a University



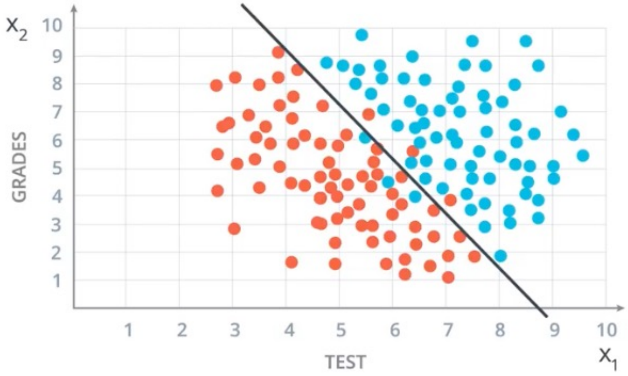
BOUNDARY:
A LINE
$2x_1 + x_2 - 18 = 0$

Score =
2*Test + Grades - 18

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

La classification - Introduction

Acceptance at a University



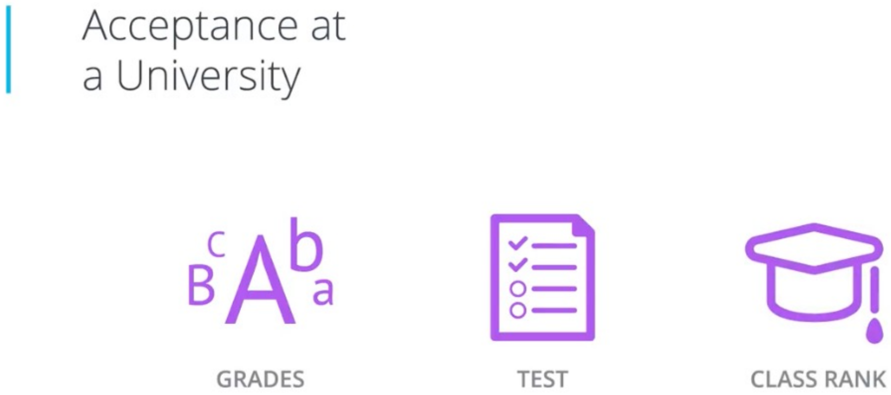
BOUNDARY:
A LINE

$w_1x_1 + w_2x_2 + b = 0$
$Wx + b = 0$
$W = (w_1, w_2)$
$x = (x_1, x_2)$
y = label: 0 or 1

PREDICTION:

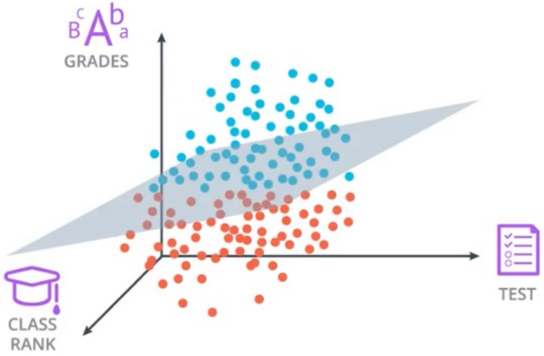
$$\hat{y} = \begin{cases} 1 \text{ if } Wx + b \geq 0 \\ 0 \text{ if } Wx + b < 0 \end{cases}$$

La classification – dimensions > 2



La classification – dimensions > 2

Acceptance at a University



BOUNDARY:
A PLANE

$w_1x_1 + w_2x_2 + w_3x_3 + b = 0$

$Wx + b = 0$

PREDICTION:

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

SITA

La classification – dimensions > 2

Acceptance at a University

	x_1	x_2	x_3		x_n	y
	EXAM 1	EXAM 2	GRADES	...	ESSAY	PASS?
STUDENT 1	9	6	5	...	6	1(yes)
STUDENT 2	8	4	8	...	3	0(no)
...	...	...	...	...	...	
STUDENT n	6	7	2	...	8	1(yes)

← n columns →

n-dimensional space

$x_1, x_2, \ldots, x_n$

BOUNDARY:

n-1 dimensional hyperplane

$w_1x_1 + w_2x_2 + w_nx_n + b = 0$

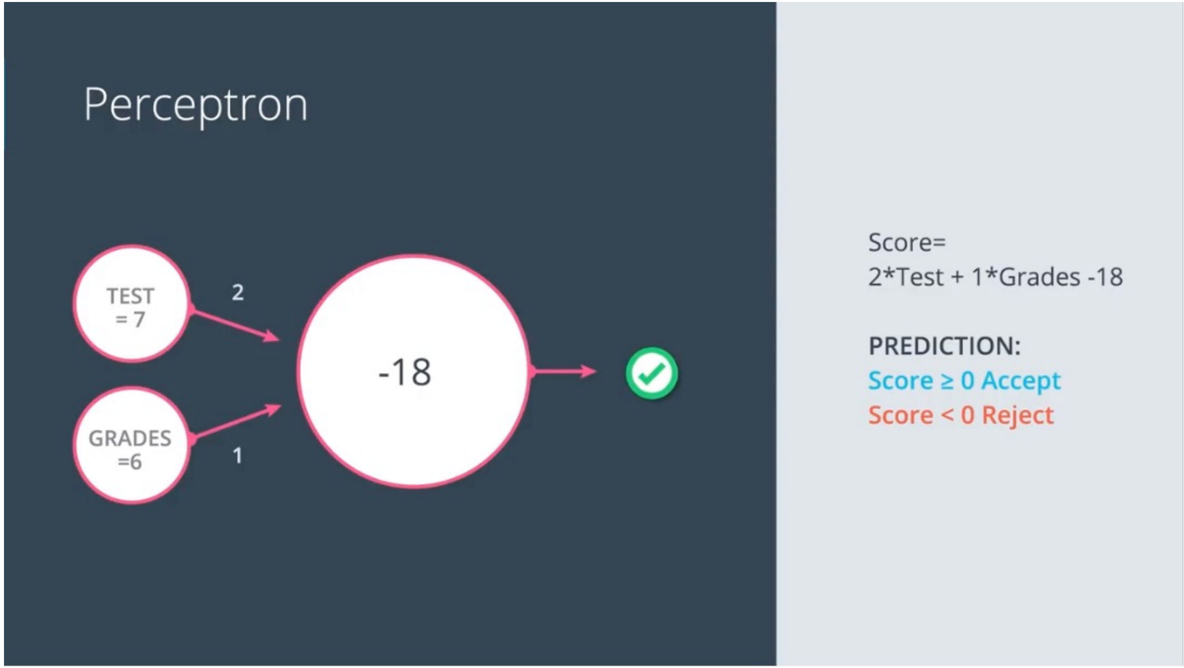
$Wx + b = 0$

PREDICTION:

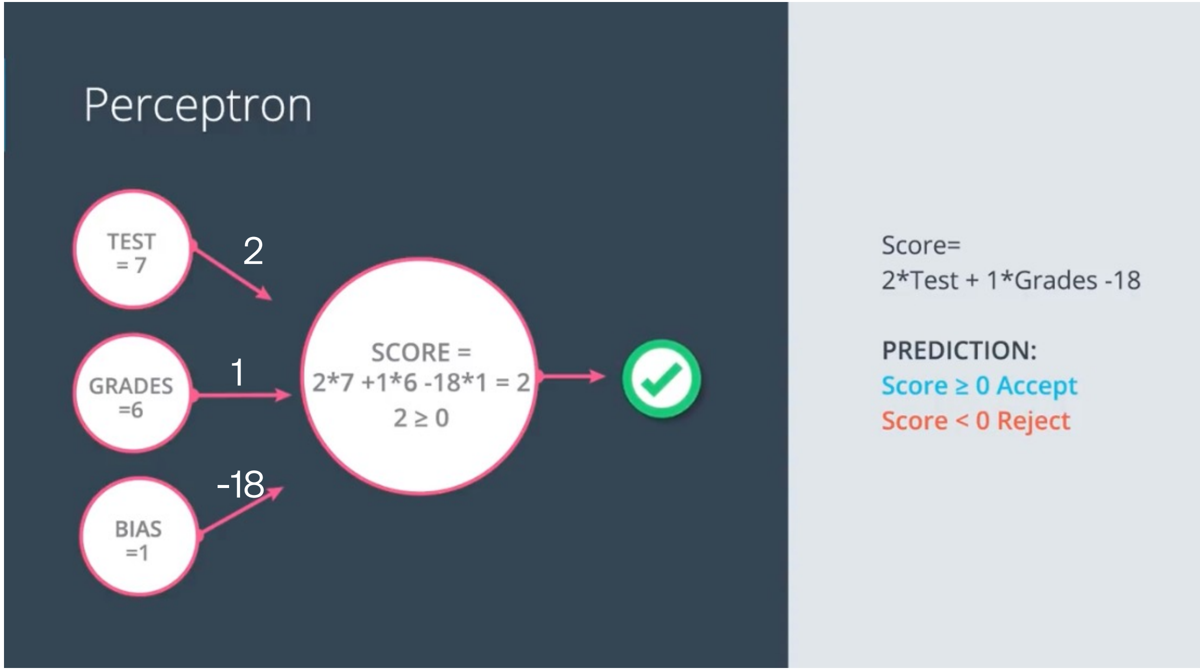
$$\hat{y} = \begin{cases} 1 \text{ if } Wx + b \geq 0 \\ 0 \text{ if } Wx + b < 0 \end{cases}$$

Les Perceptrons

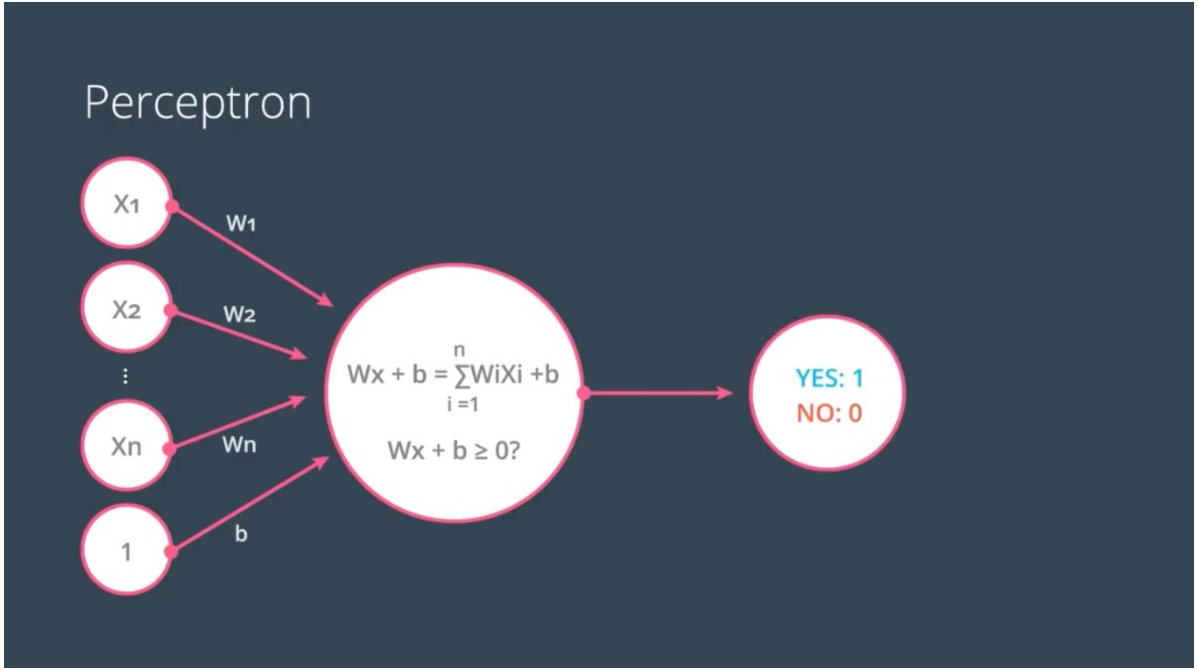
Les Perceptrons



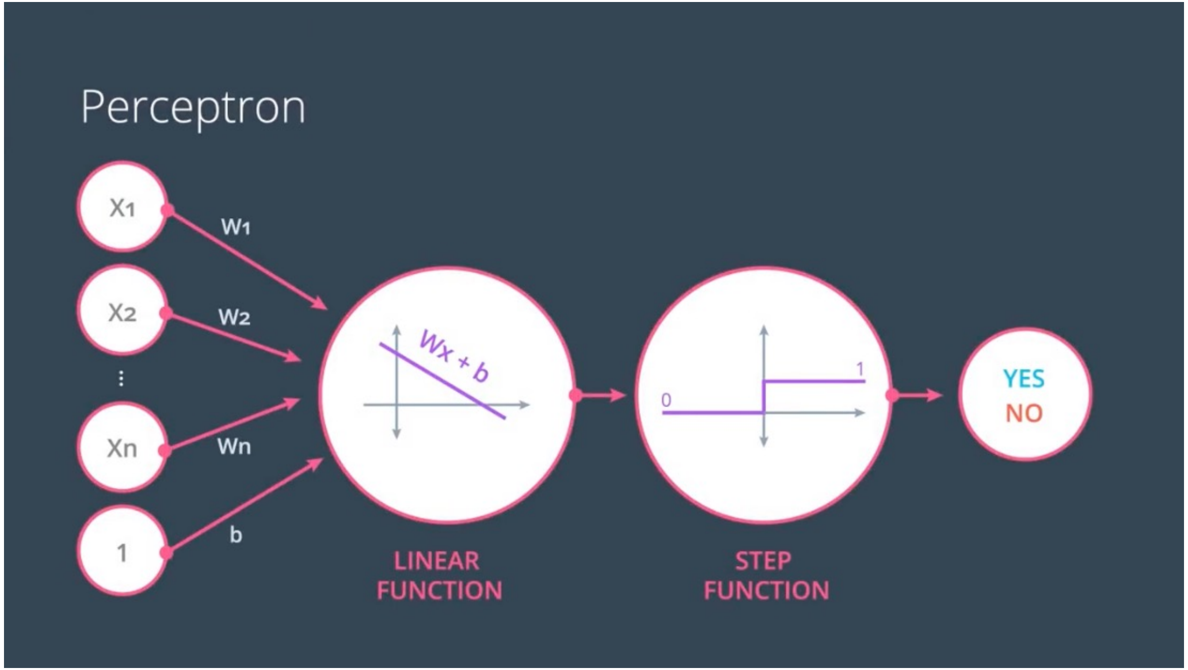
Les Perceptrons



Les Perceptrons

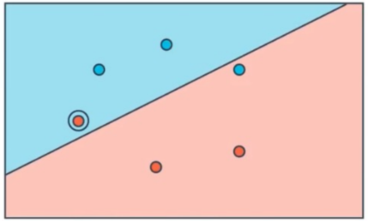


Les Perceptrons



Algorithme du Perceptron

Perceptron Algorithm



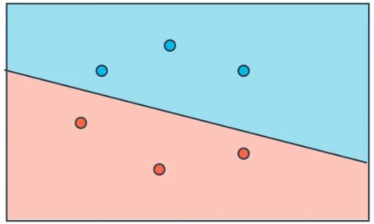
1. Start with random weights: $w_1, \ldots, w_n, b$

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 $b + \alpha$

   2.2. If prediction = 1:
   - For i = 1 ...n
     - Change $w_i - \alpha x_i$
   - Change b to $b - \alpha$

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 $b + \alpha$

   2.2. If prediction = 1:
   - For i = 1 ...n
     - Change $w_i - \alpha x_i$
   - Change b to $b - \alpha$

Algorithme du Perceptron - Exercice

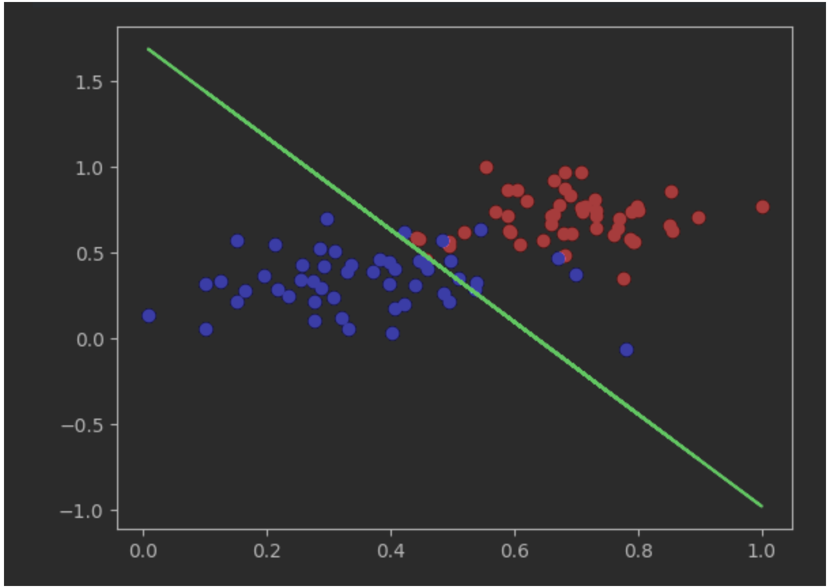
```
import numpy as np
# Setting the random seed, feel free to change it and see different solutions.
np.random.seed(42)

def stepFunction(t):
    if t >= 0:
        return 1
    return 0

def prediction(X, W, b):
    return stepFunction((np.matmul(X,W)+b)[0])

# TODO: Fill in the code below to implement the perceptron trick.
# The function should receive as inputs the data X, the labels y,
# the weights W (as an array), and the bias b,
# update the weights and bias W, b, according to the perceptron algorithm,
# and return W and b.
def perceptronStep(X, y, W, b, learn_rate = 0.01):
    # Fill in code
    return W, b

# This function runs the perceptron algorithm repeatedly on the dataset,
# and returns a few of the boundary lines obtained in the iterations,
# for plotting purposes.
# Feel free to play with the learning rate and the num_epochs,
# and see your results plotted below.
def trainPerceptronAlgorithm(X, y, learn_rate = 0.01, num_epochs = 25):
    x_min, x_max = min(X.T[0]), max(X.T[0])
    y_min, y_max = min(X.T[1]), max(X.T[1])
    W = np.array(np.random.rand(2,1))
    b = np.random.rand(1)[0] + x_max
    # These are the solution lines that get plotted below.
    boundary_lines = []
    for i in range(num_epochs):
        # In each epoch, we apply the perceptron step.
        W, b = perceptronStep(X, y, W, b, learn_rate)
        boundary_lines.append((-W[0]/W[1], -b/W[1]))
    return boundary_lines
```



https://github.com/elhidali/EPISEN-2024