[Big Data et IA] - Rendu TP 01

Rendu par `Vincent Lagogué`, `David Tejeda`, `Tom Thioulouse`, `Thomas Peugnet`.

Question 1 - Création du dataset pour l'entrainement du neurone

Code Python

```python
import random
import matplotlib.pyplot as plt
import numpy as np

def generate_class1(n):
    """
    Generates n points in the first class, abscisse in [0,10] and ordonnée in [0,10]
    """
    points = []
    for i in range(n):
        x = int(random.random()*10)
        y = int(random.random()*10)
        points.append((x,y))
    return points

def generate_class2(n):
    """
    Generates n points in the second class, abscisse in [-10,0] and ordonnée in [0,10]
    """
    points = []
    for i in range(n):
        x = int(random.random()*(-10))
        y = int(random.random()*10)
        points.append((x,y))
    return points


def get_points(n):
    """
    Generates n points for the training dataset
    """
    points = []
    points += generate_class1(n)
    points += generate_class2(n)

    # Shuffle the points
    random.shuffle(points)

    return points
```

```python
def dataset_entrainement(points):
    """
    Generates n points for the training dataset
    """

    # Get 80% of the points
    points = points[:int(len(points)*0.8)]
    return points

def dataset_test(points):
    """
    Generates n points for the testing dataset
    """

    # Get 20% of the points
    points = points[int(len(points)*0.8):]
    return points

points = get_points(250)

dataset_entrainement = dataset_entrainement(points)
dataset_test = dataset_test(points)

print("dataset_entrainement : ", dataset_entrainement)
print("dataset_test : ", dataset_test)
```

Code qui nous donne le rendu suivant:

```
dataset_entrainement :  [(0, 3), (7, 3), (0, 5), (6, 3), (9, 9), (-9, 3), (0, 9), (-3,
0), (6, 5), (-2, 7), (0, 6), (0, 2), (-3, 1), (1, 1), (-1, 1), (-4, 3), (-7, 3), (0,
5), (-6, 2), (-2, 1), (1, 2), (-2, 2), (2, 3), (-1, 7), (-5, 3), (-7, 4), (-3, 6), (-3,
5), (6, 7), (-8, 8), (9, 0), (-3, 0), (-6, 3), (-8, 7), (8, 0), (0, 4), (6, 7), (9, 4),
(-4, 0), (-9, 9), (3, 2), (2, 3), (5, 7), (3, 8), (-2, 6), (7, 1), (5, 2), (-8, 9),
(-9, 0), (0, 8), (3, 6), (3, 6), (9, 5), (5, 9), (5, 4), (0, 1), (-5, 6), (-3, 7), (6,
3), (-4, 9), (-3, 4), (-4, 6), (-5, 5), (-5, 7), (5, 5), (8, 3), (7, 7), (0, 0), (2,
5), (5, 8), (9, 0), (0, 3), (2, 4), (-5, 9), (1, 2), (-7, 0), (1, 9), (6, 1), (2, 9),
(1, 8), (-8, 3), (-8, 6), (0, 9), (1, 5), (6, 5), (-3, 6), (5, 3), (4, 3), (9, 6), (-5,
6), (-7, 6), (-2, 1), (2, 5), (-9, 6), (6, 1), (4, 5), (4, 7), (0, 5), (-9, 8), (-1,
4), (-1, 4), (1, 4), (-5, 0), (0, 0), (-7, 8), (-7, 9), (4, 8), (8, 2), (7, 7), (-6,
6), (0, 4), (0, 3), (5, 6), (-6, 4), (-6, 1), (-7, 3), (1, 0), (-6, 1), (7, 6), (2, 9),
(-6, 2), (7, 7), (8, 1), (4, 0), (-4, 7), (-7, 1), (-7, 4), (-1, 0), (6, 1), (0, 2),
(7, 1), (-8, 7), (0, 6), (-1, 7), (9, 7), (0, 5), (9, 3), (3, 6), (-2, 7), (5, 4), (0,
0), (8, 1), (7, 8), (-5, 0), (7, 2), (-8, 4), (0, 1), (2, 7), (-4, 6), (3, 6), (-3, 2),
(-5, 7), (6, 9), (3, 2), (8, 2), (6, 4), (-3, 8), (8, 3), (0, 2), (-9, 1), (8, 4), (-5,
3), (5, 1), (-4, 6), (-9, 0), (1, 5), (-1, 8), (0, 9), (9, 6), (9, 5), (-1, 2), (-7,
0), (-9, 7), (3, 6), (1, 3), (-3, 2), (-7, 6), (9, 2), (9, 1), (0, 1), (3, 0), (-1, 4),
(-9, 5), (-2, 1), (-8, 3), (1, 9), (-8, 7), (-1, 6), (-5, 6), (-2, 1), (-1, 0), (-9,
8), (6, 7), (-1, 7), (2, 8), (9, 3), (6, 4), (0, 2), (-5, 6), (-2, 9), (8, 7), (5, 8),
(3, 4), (-4, 7), (5, 0), (0, 0), (-7, 9), (7, 2), (-6, 2), (9, 7), (3, 3), (-3, 3), (8,
5), (2, 2), (9, 1), (6, 3), (5, 9), (7, 3), (4, 0), (-4, 2), (-1, 6), (3, 0), (7, 7),
(0, 3), (5, 5), (1, 7), (4, 6), (-9, 2), (9, 7), (4, 2), (-5, 9), (1, 5), (-1, 7), (6,
0), (5, 6), (-1, 5), (3, 4), (-2, 0), (0, 8), (8, 9), (4, 8), (-9, 2), (1, 5), (-1, 9),
(0, 4), (-1, 7), (2, 2), (0, 1), (2, 5), (-6, 8), (-1, 3), (9, 2), (4, 1), (-5, 0), (3,
0), (-7, 4), (2, 6), (-2, 1), (-8, 8), (3, 7), (2, 2), (5, 6), (3, 0), (-3, 2), (3, 8),
(-7, 8), (3, 8), (-5, 5), (-1, 5), (0, 0), (-6, 1), (8, 9), (6, 9), (-6, 8), (3, 5),
(-5, 7), (6, 3), (-8, 5), (-8, 1), (5, 6), (-9, 1), (8, 8), (-7, 0), (-2, 6), (8, 3),
(-3, 6), (-7, 2), (0, 3), (6, 6), (-1, 0), (5, 1), (0, 9), (-4, 1), (-8, 6), (0, 3),
(-4, 0), (9, 0), (-6, 5), (2, 9), (0, 7), (1, 3), (-7, 6), (5, 9), (5, 0), (-3, 5),
(-6, 8), (3, 7), (9, 3), (6, 1), (-1, 8), (-9, 0), (9, 6), (-9, 7), (6, 4), (3, 3),
(-9, 4), (-9, 0), (-4, 2), (6, 2), (9, 9), (-6, 3), (9, 5), (2, 7), (9, 6), (1, 9),
(-2, 1), (-2, 5), (-4, 7), (-6, 5), (-6, 1), (-5, 3), (-4, 9), (5, 8), (3, 2), (-2, 6),
(0, 0), (4, 3), (2, 0), (-9, 4), (0, 1), (0, 2), (0, 1), (6, 0), (-9, 3), (8, 8), (-8,
4), (-5, 0), (2, 9), (2, 5), (-4, 0), (-2, 0), (-5, 0), (1, 9), (6, 6), (-9, 7), (5,
8), (-7, 0), (6, 6), (-6, 5), (4, 5), (-7, 1), (-6, 7), (3, 8), (-5, 8), (0, 3), (-2,
1), (-1, 5), (-8, 9), (-6, 5), (0, 9), (3, 1), (3, 8), (2, 9), (0, 8), (6, 5), (4, 1),
(-8, 1), (3, 4), (2, 0), (-7, 3), (5, 2), (-1, 4), (-4, 4), (3, 6), (5, 8), (-2, 2),
(2, 7), (-1, 5), (-5, 8), (-4, 6), (6, 8), (-1, 0), (-1, 4), (8, 2), (9, 6), (1, 0),
(-8, 2), (-7, 2), (9, 3), (2, 8)]
```

```
dataset_test :  [(-6, 3), (-7, 1), (6, 4), (-2, 0), (-9, 5), (-8, 0), (9, 8), (2, 0),
(-8, 8), (4, 6), (2, 0), (2, 9), (2, 5), (-4, 4), (0, 0), (-4, 3), (2, 6), (-6, 6),
(-6, 6), (-3, 7), (6, 0), (-6, 1), (3, 6), (0, 5), (-7, 5), (-3, 8), (3, 6), (-4, 7),
(8, 6), (5, 2), (3, 8), (-5, 6), (-9, 7), (-9, 9), (-1, 4), (6, 8), (8, 2), (-2, 8),
(9, 6), (7, 6), (2, 5), (9, 2), (3, 8), (9, 7), (4, 1), (-8, 3), (-6, 5), (-9, 8), (9,
5), (-8, 7), (4, 0), (2, 8), (4, 2), (8, 1), (-7, 5), (-2, 5), (2, 0), (7, 6), (-2, 1),
(8, 8), (3, 8), (-7, 5), (-6, 8), (0, 7), (1, 4), (-3, 6), (-1, 4), (9, 5), (2, 2),
(-1, 4), (6, 7), (3, 2), (-2, 1), (1, 5), (1, 9), (-3, 0), (0, 4), (-8, 8), (8, 8), (6,
6), (-1, 8), (-2, 7), (-5, 2), (-4, 8), (-1, 5), (-6, 8), (2, 0), (-6, 2), (2, 6), (0,
7), (-9, 2), (4, 2), (-5, 5), (-9, 4), (-3, 7), (0, 6), (-6, 8), (-1, 8), (5, 4), (-4,
2)]
```

Après plusieurs tests, nous avons augmenté considérablement le nombre de points (250 -> 100 000) pour avoir davantage de précision.

Question 2 et 3 - Entrainez un neurone et mesurez la précision du modèle.

Voici le code qui nous permet d'entrainer notre neurone et d'en mesurer la précision.

```python
import random
import numpy as np

W = (0,0)
b = 0.5
alpha = 0.1

def generate_class1(n):
    """
    Generates n points in the first class, abscisse in [0,10] and ordonnée in [0,10]
    """
    points = []
    for i in range(n):
        x = int(random.random()*10)
        y = int(random.random()*10)
        points.append((x,y,0))
    return points


def generate_class2(n):
    """
    Generates n points in the second class, abscisse in [-10,0] and ordonnée in [0,10]
    """
    points = []
    for i in range(n):
        x = int(random.random()*(-10))
        y = int(random.random()*10)
        points.append((x,y,1))
    return points


def get_points(n):
    """
    Generates n points for the training dataset
    """
    points = []
    points += generate_class1(n)
    points += generate_class2(n)

    random.shuffle(points)

    return points


def dataset_entrainement_f(points):
    """
    Generates n points for the training dataset
    """

    # Get 80% of the points
    points = points[:int(len(points)*0.8)]
    return points


def dataset_test_f(points):
    """
    Generates n points for the testing dataset
    """

    # Get 20% of the points
    points = points[int(len(points)*0.8):]
    return points


def f(x,y):
    """
    Returns the value of the function f(x,y) = W_0 * x + W_1 * y + b
    """
    return W[0]*x + W[1]*y + b


# Test the model
def test_model(dataset):
    """
    Returns the accuracy of the model
    """
    correct_predictions = 0
    for e in dataset:
        x = e[0]
        y = e[1]
        label = e[2]

        if f(x,y) > 0:
            prediction = 1
        else:
            prediction = 0

        if prediction == label:
            correct_predictions += 1

    return correct_predictions / len(dataset)


points = get_points(1000000)
dataset_entrainement = dataset_entrainement_f(points)
dataset_test = dataset_test_f(points)

# Formule W_0 * x + W_1 * y + b

for e in dataset_entrainement:
    x = e[0]
    y = e[1]
    label = e[2]

    if f(x,y) > 0:
        prediction = 1
    else:
        prediction = 0

    if prediction != label:
        w0 = W[0] + alpha * (label - prediction) * x
        w1 = W[1] + alpha * (label - prediction) * y
        b = b + alpha * (label - prediction)
        W = (w0,w1)


print("W = ", W)
print("b = ", b)
print("Accuracy of the model : ", test_model(dataset_test))
print("Accuracy of the model with sigmoid : ", test_model2(dataset_test))
```

Ce qui nous donne le résultat suivant :

```
╰─➤  python3 exo1.py
                                                                        1 ↵
W =  (-8.29, -0.79)
b =  -2.77e-17
Accuracy of the model :  0.9501475
```

Nous donnant donc une précision proche de 95%.

Question 4 - Refaites l'expérience avec la fonction sigmoid

Voici le code de la fonction sigmoide ainsi que du second test que nous avons effectué :

```python
def sigmoid(x):
    """"
    Returns the sigmoid of x
    """
    return 1 / (1 + np.exp(-x))


def f2(x,y):
    """
    Returns the value of the function f(x,y) = sigmoid(W_0 * x + W_1 * y + b)
    """
    return sigmoid(W[0]*x + W[1]*y + b)


def test_model2(dataset):
    """
    Returns the accuracy of the model
    """
    correct_predictions = 0
    for e in dataset:
        x = e[0]
        y = e[1]
        label = e[2]

        if f2(x,y) > 0.5:
            prediction = 1
        else:
            prediction = 0

        if prediction == label:
            correct_predictions += 1

    return correct_predictions / len(dataset)
```

Nous donnant ainsi le résultat suivant :

```
╰─➤  python3 exo1.py
Accuracy of the model with sigmoid :  0.9501475
```

Nous pouvons constater que nous retrouvons précisément la même valeur que lors de la question précédente.