Assignment 1

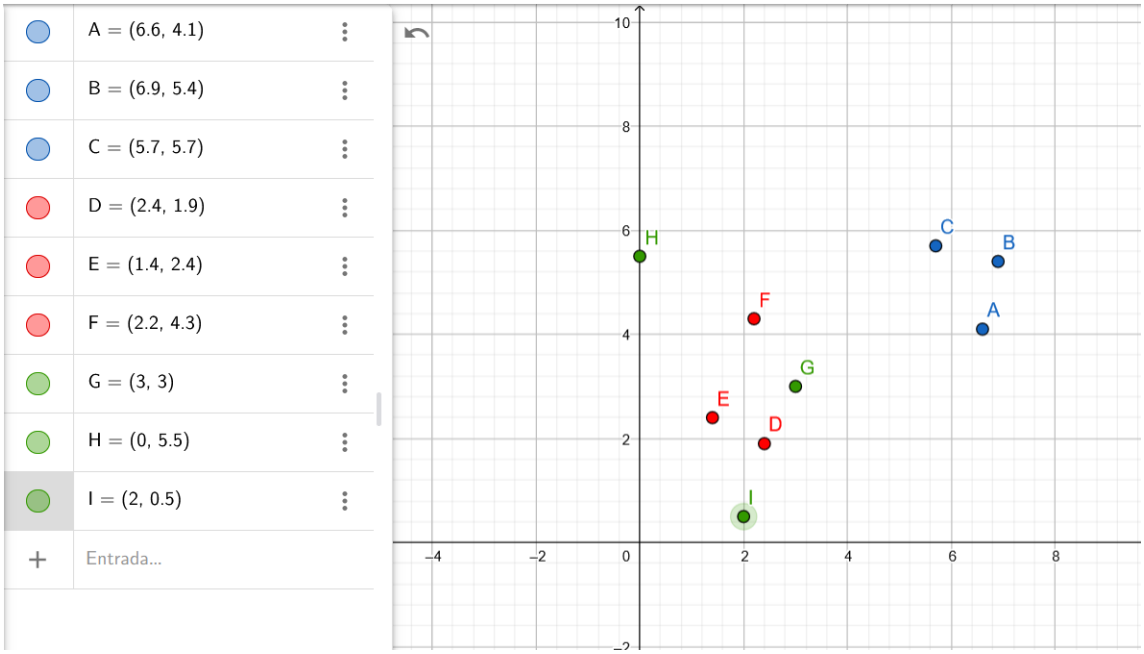
Elena Pérez-Ródenas Martínez
REDID: 827222533

September 2021

I, Elena Pérez-Ródenas Martínez, guarantee that this homework is my independent work and I have never copied any part from other resources. Also, I acknowledge and agree with the plagiarism penalty specified in the course syllabus.

1. Exercise

1.1.



First of all, being the red points of class 1 and the blue ones of class 0, we can see that the new green points are closer to the ones of class 1.

In order to check this, we can calculate the Euclidean distance between each of them and the new one and take the three closer points.

Let's start with point 1 (3,3)

$d(1,A) = \sqrt{(6{,}6-3)^2+(4{,}1-3)^2} = 3{,}7643$
$d(1,B) = \sqrt{(6{,}9-3)^2+(5{,}4-3)^2} = 4{,}5793$
$d(1,C) = \sqrt{(5{,}7-3)^2+(5{,}7-3)^2} = 3{,}8184$
$d(1,D) = \sqrt{(2{,}4-3)^2+(1{,}9-3)^2} = 1{,}253$
$d(1,E) = \sqrt{(1{,}4-3)^2+(2{,}4-3)^2} = 1{,}7088$

$d(1,F) = \sqrt{(2{,}2-3)^2 + (4{,}3-3)^2} = 1{,}5264$

Since the 3 closest neighbours are D, E and F with class 1, the new point's class will be 1.

Let's see point 2 (0,5.5)

$d(2,A) = \sqrt{(6{,}6^2 + (4{,}1-5{,}5)^2} = 6{,}7469$
$d(2,B) = \sqrt{6{,}9^2 + (5{,}4-5{,}5)^2} = 6{,}9007$
$d(2,C) = \sqrt{5{,}7^2 + (5{,}7-5{,}5)^2} = 5{,}7035$
$d(2,D) = \sqrt{2{,}4^2 + (1{,}9-5{,}5)^2} = 4{,}3267$
$d(2,E) = \sqrt{(1{,}4^2 + (2{,}4-5{,}5)^2} = 3{,}4015$
$d(2,F) = \sqrt{2{,}2^2 + (4{,}3-5{,}5)^2} = 2{,}506$

Since the 3 closest neighbours are D, E and F with class 1, the new point's class will be 1.

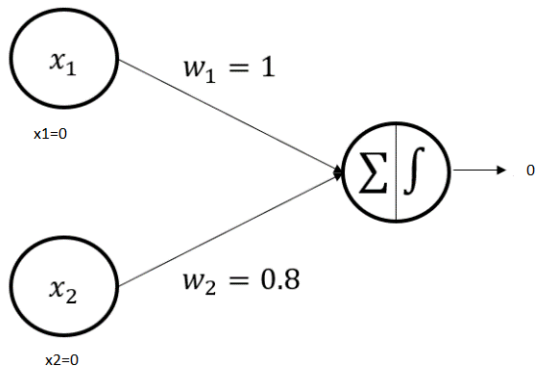
Let's see point 3 (2,0.5)

$d(3,A) = \sqrt{(6{,}6-2)^2 + (4{,}1-0{,}5)^2} = 5{,}8412$
$d(3,B) = \sqrt{(6{,}9-2)^2 + (5{,}4-0{,}5)^2} = 6{,}93$
$d(3,C) = \sqrt{(5{,}7-2)^2 + (5{,}7-0{,}5)^2} = 6{,}382$
$d(3,D) = \sqrt{(2{,}4-2)^2 + (1{,}9-0{,}5)^2} = 1{,}456$
$d(3,E) = \sqrt{(1{,}4-2)^2 + (2{,}4-0{,}5)^2} = 1{,}9925$
$d(3,F) = \sqrt{(2{,}2-2)^2 + (4{,}3-0{,}5)^2} = 3{,}8053$

Since the 3 closest neighbours are D, E and F with class 1, the new point's class will be 1.

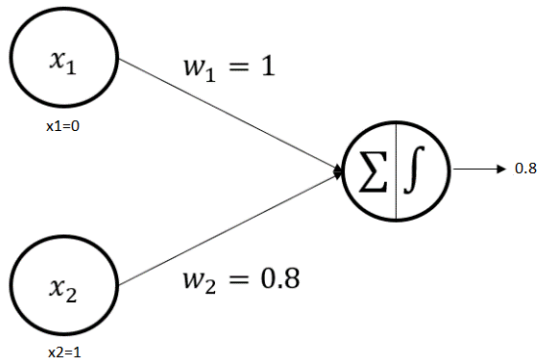
1.2.

For x={0,0}



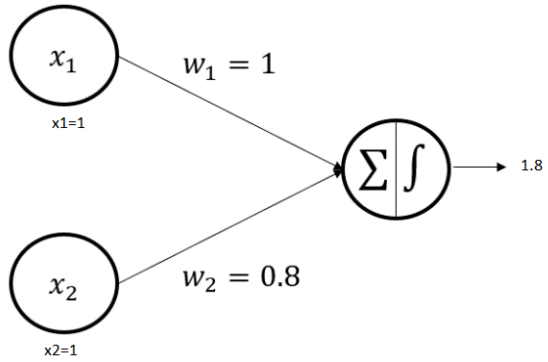
As we can see, the sum is 0, which is not greater than 1.5 so the class will be 0.

For x={0,1}



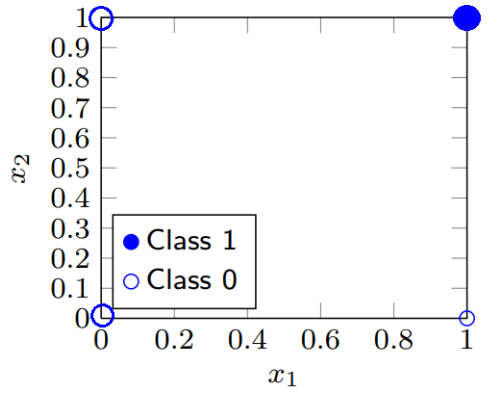
Now the sum is 0.8, which is not greater than 1.5 either, so the class will be 0 again.

For x={1,1}

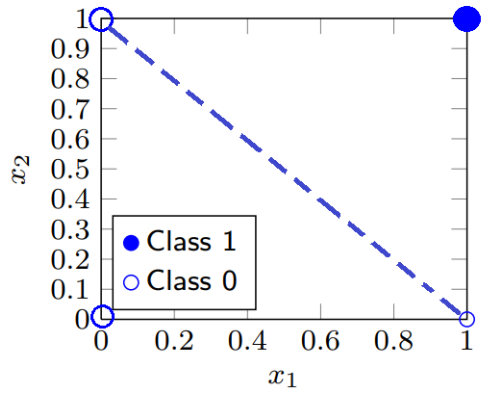


Now the sum is 1.8 which is greater than 1.5 so the class will be 1.

We can plot the outputs like this:



Geomtery meaning:



When $\theta = 1{,}5$ below the line the points will be of class 0 and above of class 1. The points in the line will have class 0 as well.
Actually,if the point satisfies the inequality $1 * (x_1) + 0{,}8 * (x_2) > \theta$, its class will be 1.

2. Coding Assignment KNN

2.1. Task 1

First of all we split data into attributes (paramA and paramB) and labels (Class).
After that, we can just import the function train_test_split and for example use a test size of 20 %

```
from sklearn.model_selection import train_test_split

def split_input_data(data):
    X = data[['paramA','paramB']]
    y = data['Class']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
    return X_train, X_test, y_train, y_test
```

2.2. Task 2

In order to complete KNN() we implement a classifier using the function KNeighborsClassifier of sklearn.neighbors being nneighbors=k=3 in this example. After that, we can just fit it with X_train and y_train and make the prediction for X_test.

```
def knn(nneighbors, X_train, y_train, X_test):
    clf = KNeighborsClassifier(nneighbors)
    clf.fit(X_train, y_train)
    display_contours(clf, nneighbors, X_train, y_train)
    predicted_y = clf.predict(X_test)
    return predicted_y
```

We can print the result and see the prediction made:

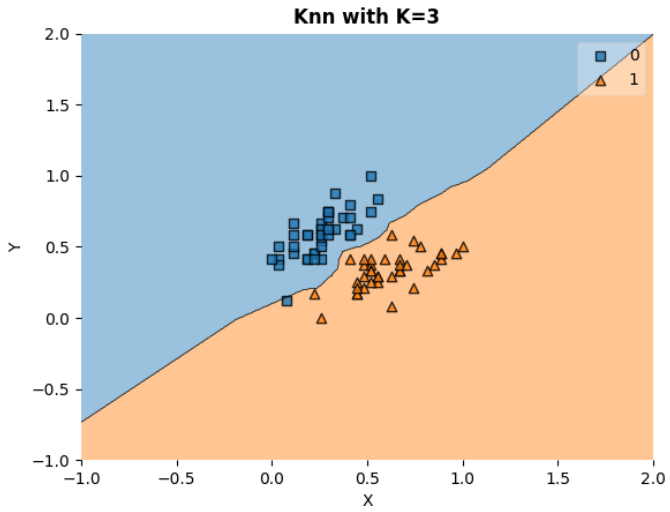
```
predicted_y
0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 0
```

2.3. Task 3

The function display_contours() es supposed to plot the decision region of the classifier. In order to do this we can use plot_decision_regions with the values of X_train in x axis and y_train in y axis. It is also possible to name the axes and put a title.

```
def display_contours(classifier, number_of_neighbors, X_train, y_train):
    plot_decision_regions(X=X_train.values, y=y_train.values, clf=classifier)
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title('Knn with K='+ str(number_of_neighbors), fontweight="bold")
    plt.show()
```

Here is the result of the final plot:



2.4. Task 4

Finally, we want to print the confusion matrix and the classification reports based on y_predicted and y_test.

```
def evaluateknn(y_predicted, y_test):
    print("Confusion matrix")
    print(confusion_matrix(y_predicted, y_test), "\n")
    print("Classification reports")
    print(classification_report(y_predicted, y_test), "\n")
```

```
Confusion matrix
[[10  0]
 [ 1  9]]

Classification reports
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        10
           1       1.00      0.90      0.95        10

    accuracy                           0.95        20
   macro avg       0.95      0.95      0.95        20
weighted avg       0.95      0.95      0.95        20
```