ELEC 4727/5727

Computer Vision: Introduction to Machine Learning

Hadi Abdelmawla

**KNN Analysis**

**[Part A] K-NN (K Nearest Neighbor Classification)- Perform 3-NN analysis for the following 2-dimensional training values**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **X** | **Y** | **Classification** |
| **A** | **1** | **1** | **Good** |
| **B** | **3** | **7** | **Bad** |
| **C** | **3** | **3** | **Good** |
| **D** | **2** | **1** | **Good** |
| **E** | **3** | **2** | **Bad** |

**Use the query attributes of the new point F: [X= 3, Y= 4]**

**Use Euclidean Distance as a measure and MAJORITY to designate the classification as: Good, Bad, No Classification. Clearly fill out the following table showing the 3 nearest neighbors (give data rank 1=min, 2=second min, …, 5=farthest point)**

|  |  |  |
| --- | --- | --- |
| **Data** | **Distance to Point F** | **Nearest Neighbor Rank** |
| **A** | **3/2** | **2** |
| **B** | **4** | **5** |
| **C** | **1** | **1** |
| **D** | **3** | **4** |
| **E** | **2** | **3** |

|  |  |
| --- | --- |
| **Classification** | **Good or Bad** |
| **K=1** | **good** |
| **K=3** | **bad** |

**KMeans Clustering**

**[Part A] Perform iterative K-Means (K=2) Clustering for the following 2-dimensional attributes.**

|  |  |  |
| --- | --- | --- |
| **Point** | **X** | **Y** |
| **A** | **2** | **1** |
| **B** | **3** | **1** |
| **C** | **5** | **2** |
| **D** | **6** | **0** |

Your final answer should designate the coordinates for 2 centroids and designate the cluster that each data point resides. You must show your work to get credit and your answer should be entered into the final table:

**Iteration 0:**

**Centroids:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Point** | **X-coordinate** | **Y-coordinate** |
| **Centroid 1** | **A** | **2** | **1** |
| **Centroid 2** | **B** | **3** | **1** |

**Distance Table: Points to Centroids**

**You do not need to use sqrt, just the magnitude of the sum of squared differences of x and y positions**

|  |  |  |
| --- | --- | --- |
| **Point** | **Centroid 1** | **Centroid 2** |
| **A** | **(2-2)^2+(1-1)^2=0** | **(2-3)^2+(1-1)^2=1** |
| **B** | **(3-2)^2+(1-1)^2=1** | **(3-3)^2+(1-1)^2=0** |
| **C** | **(5-2)^2+(2-1)^2=10** | **(5-3)^2+(2-1)^2=5** |
| **D** | **(6-2)^2+(0-1)^2=17** | **(6-3)^2+(0-1)^2=10** |

**Group Assignment based on Distance Table**

|  |  |
| --- | --- |
| **Point** | **Assign Group 1 or 2 based on closest to the centroids** |
| **A** | **1** |
| **B** | **2** |
| **C** | **2** |
| **D** | **2** |

**Derive the new centroids based on group assignment**

**Centroids:**

|  |  |  |
| --- | --- | --- |
|  | **X-coordinate** | **Y-coordinate** |
| **Centroid 1** | **2** | **1** |
| **Centroid 2** | **(3+5+6)/3=4 and 2/3** | **(1+2+0)/3 =1** |

**Distance Table: Points to Centroids**

**You do not need to use sqrt, just the magnitude of the sum of squared differences of x and y positions**

|  |  |  |
| --- | --- | --- |
| **Point** | **Centroid 1** | **Centroid 2** |
| **A** | **0** | **(2-4.66)^2+(1-1)^2=7.11** |
| **B** | **1** | **(3-4.66)^2+(1-1)^2=2.76** |
| **C** | **10** | **(5-4.66)^2+(2-1)^2=1.1156** |
| **D** | **17** | **(6-4.66)^2+(0-1)^2=2.7956** |

**Group Assignment based on Distance Table**

|  |  |
| --- | --- |
| **Point** | **Assign Group 1 or 2 based on closest to the centroids** |
| **A** | **1** |
| **B** | **1** |
| **C** | **2** |
| **D** | **2** |

**KNN (Graduate students)**

**Write a Python function for processing an Numpy array of (x,y,class) into the KNN**

**Example:**

D = np.array([[3,13,0],[2.75,5,0],[3.5,1,0],[3.75,9,0],[1,9,1],[1.5,5,1],[5.5,1,1],[5.25,5,1],[6.5,9,1],[6,13,1]])

The data is shown as [X,Y,class], in which case the class is a number, and the X,Y value. You can generalize the solution so that the last item in each data list is the class, in that way, you should work your solution to handle any dimension data features (Example 5 features: [X,Y,Z,W,Q, class]

run\_knn('example\_data.csv', num\_neighbors=3, input\_data=[3,4] )

Running the above code will return a class prediction (majority or max of the 3 nearest data.) You do not need to handle cases for number of K is greater than the size of the data. Just the base case of KNN from the first problem of this assignment using the data D.

Example:

[['3', '13', '0'], ['2.75', '5', '0'], ['3.5', '1', '0'], ['3.75', '9', '0'], ['1', '9', '1'], ['1.5', '5', '1'], ['5.5', '1', '1'], ['5.25', '5', '1'], ['6.5', '9', '1'], ['6', '13', '1']]

[0] => 0

[1] => 1

[[3.0, 13.0, 0], [2.75, 5.0, 0], [3.5, 1.0, 0]]

[0, 0, 0]

Data=[3, 4], Predicted: 0

**KMeans Clustering (Graduate students)**

**Write a Python function for processing an Numpy array of (x,y,class) into the Kmeans clustering algorithm**

**Example:**

D = np.array([[3,13,0],[2.75,5,0],[3.5,1,0],[3.75,9,0],[1,9,1],[1.5,5,1],[5.5,1,1],[5.25,5,1],[6.5,9,1],[6,13,1]])

For this problem, the class field can be ignored on the above data. The Kmeans algorithm will be run, and if the new centers (centroids) calculated do not change or iterations. Your code can have the following input use:

run\_kmeans(D, num\_clusters=3, num\_iterations=10)

You do not need to make a prediction for a new data points (that would compare the new point to the nearest of the K centers), you only need to output the final breakdown of centroids and which points of the dataset D belong to each clustered group.