

COMP 542 Machine Learning  
(Fall 2021)  
**Assignment 4**

Name: \_\_\_\_\_ Score: \_\_\_\_\_/200

Submit your assignment at CANVAS by uploading a file named  
**'assignment4\_your\_first\_name.ipynb'**.

**Due date: Wednesday, 24<sup>th</sup> of the November, 2021 by 11:59 PM**

-----

1. [100 points] **K-Mean++**. Implement K-Mean++ clustering algorithm in python as follows:

- Read input file 'as4\_1.txt' given in the Canvas course website. The file is composed of X and Y values in the first and second columns and label in the third column.
- Create **myInit()** that places the initial  $k$  centroids far away from each other in the 4 steps as shown below:
  1. Randomly select the first centroid from the data points
  2. For each data point compute its distance from the nearest, previously chosen centroid

Use following Euclidean distance function:

```
import numpy as np

def euclidean2D(point1, point2):
    x1 = point1[0]
    x2 = point2[0]
    y1 = point1[1]
    y2 = point2[1]

    return np.sqrt((x1 - x2)**2 + (y1 - y2)**2)
```

3. Select the point having maximum distance from the nearest centroid as the next centroid
4. Repeat steps 2 and 3 until  $k$  centroids have been sampled

- Create **myAssign()** that assigns each example to the nearest centroid,  $\mu^{(j)}$ ,  $j \in \{1, \dots, k\}$  where for every  $x^{(i)}$ ,  $\text{label}[x^{(i)}] = j$  which is  $\arg \min_j \|x^{(i)} - \mu^{(j)}\|^2$ .
- Create **myCentroid()** that calculates a new centroid of all points that are assigned to the same centroid.
- Create **myUpdateCentroid()** that moves the centroids to the center of the examples that were assigned to it
- Create **myKmeanPlusPlus()** that initially calls **myInit()**, and then repeats to call **myAssign()**, **myCentroid()**, and **myUpdateCentroids()** until the cluster assignments do not change or a user-defined tolerance or maximum number of iteration is reached. **myKmeanPlusPlus()** should ask user to receive the following arguments and use the same variable name in the parenthesis:
  1. The number of clusters ( $k$ )
  2. Tolerance ( $myTol$ )
  3. Maximum number of iterations ( $myMax$ )**myKmeanPlusPlus()** returns a list of new labels.
- Create **myPlot()** that visualizes plot of clustering result in different colors and markers. You can use any plot method.

2. [100 points] **DBSCAN**. Implement DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm in python as follows:
- Read input file named as 'as4\_2.csv' using numpy. The file is composed of X and Y values in the first and second columns and label in the third column.
  - Create '**getLabel()**' that receives *true labels* from the read file.
  - Create '**getData()**' that returns *vectors* from the read file.
  - Create '**getDBSCAN()**' that receives *vectors*, *epsilon*, and *minPoints* and returns *predicted label*. This function finds the points in the epsilon neighborhood of every point, and identifies the core points with more than *minPoints* neighbors. Secondly, this function finds the connected components of core points on the neighbor graph. Lastly, this function assigns each non-core point to a nearby cluster if the cluster is an epsilon neighbor, otherwise assign it to noise. **getDBSCAN()** returns a list of new labels.
- For the more detailed algorithm for DBSCAN, you can check out at the <https://en.wikipedia.org/wiki/DBSCAN>
- Create '**getAccuracy()**' that receives *predicted label* and *true label* and returns *accuracy*.
  - Create '**plotDBSCAN()**' that visualizes plot of clustering result in different colors and markers. You can use any plot method.
3. [Bonus 50 points] **Improved DBSCAN**. Implement improved DBSCAN method using one of existing algorithms introduced in the paper "S. Li, An Improved DBSCAN Algorithm Based on the Neighbor Similarity and Fast Nearest Neighbor Query, IEEE Access, 2020". This paper introduces many previous methods enhancing performance and accuracy. You can use any suggested methods in Section II, RELATED WORK as well as the method that the paper mainly suggests. Show the enhanced accuracy comparing to what you will be obtaining in the previous problem 2 (**DBSCAN**). You can use the same functions that you have implemented and dataset in problem 2. You can use different dataset from problem 2 to show better accuracy but when you compare the accuracy, the chosen dataset should be equally and additionally used in problem 2.

Write your 3 python programs in each cell of jupyter notebook and save into your shared directory as a name, '**assignment4\_your\_first\_name.ipynb**' where 'your\_first\_name' should be your real first name. Attach the file at the Assignment 4 in the Canvas.

**No code sharing is permitted**

**Please start your assignment early and submit your file within the deadline**