Techniques of AI project report: K-medoids Algorithm

Yiming Yan 0573780

Goals of project:

In this project, I will implement a K-Medoids clustering algorithm, and divide the provided dataset into clusters. The results will be shown with plots. Accuracy will be the main metric to evaluate the functionality of algorithm. Moreover, some other metrics like F-score will work for evaluation too.

The purpose of using multiple dataset is to discuss whether clustering is suitable for different data sets, that the distribution of data may vary. A comparison between K-means and K-medoids will be made, the evaluation will be based on classification accuracy. Moreover, the principle of selecting k value will be discusses with experiment.

Dataset:

Three different data set will be used in this project. The types of attributes are numeric, while no missing value exists in all of the data sets.

First a small data set with two features, in order to show the functionality of the cluster algorithm. We could consider this data set as a set of points in two-dimensions with x and y coordinate. The size of the data set is 40.Name of the file: ”exampledataset.txt”.

Then Iris data set will be used, it contains four attributes and every instance is labeled. Its size is 150. Iris is one of the most famous data set for machine learning, containing the following information:

Name of file:”iris.data”

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class: {Iris-Setosa, Iris-Versicolour , Iris-Virginica}

And “banknote authentication” data set downloaded from UCI Machine Learning Repository. It is a dataset whose data were extracted from images, to evaluate the authentication of banknotes. Four attributes and a column of labels are included. The size of data set is 1372.

Name:”data_banknote_authentication.txt”

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer): {0 (Fake), 1 (Authentic)}

We could see the difference between the format of storing data in different files. For second kind of file we need to implement a function to process the data, which will be mentioned later in this document.

```
0.196705753183788   0.266174967499455
0.413286989521383   0.355828352990633
0.338435546719209   0.435738258997923
0.103801517189990   0.164344805836115
0.159052363075132   0.325059012698889
0.0669054926780630  0.487418074001379
0.335731444739015   0.0379836806470678
0.285495537731203   0.293509583541386
0.0848835330132443  0.206943248886680
0.0738278885758684  0.154568213233134
0.238039859133728   0.131917020763398
0.454051208253475   0.379383132540102
0.276087513357917   0.497607990564876
0.0164699463749383  0.0932857220706846
0.0269314632177781  0.390572634267382
0.402531614279451   0.0978989905133660
0.225687427351724   0.496179486589963
0.191323114779979   0.401130784882144
0.394821851844845   0.212113354951653
0.182143434749897   0.364431934025687
1.49835358252355    1.40350138880436
1.80899026719904    1.93497908617805
1.35650893348105    1.47948454563248
1.07324343448981    1.23179161166312
1.59099145527485    1.39629024850978
```

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
```

Figure: Left: Example small data set; Right: Iris data set

As for the first example data set, different k values will be applied to the algorithm to show how the points are grouped, with different number of centroids.

In Iris data set, data are labeled with three classes, representing different kinds of subspecies, as it was shown before. Which means we will divide the data into three different clusters, as a prediction on given data. And then compare the prediction with original labels, to calculate the accuracy of classification.

And for banknote data set, the number of classes is reduced to two: authentic or fake. The meaning of applying this dataset is to test the functionality of the algorithm when the size of data set becomes much larger. Accuracy will show the effect of algorithm.

And by comparing performance of the algorithm dealing with different data set, we could judge whether it always works.

Learn Technique Used in This Project

K-medoids Algorithm

K-mediods is a clustering algorithm. Clustering is to group the objects with more similarity to each one within the cluster than the objects in other clusters. The common clustering algorithm can be divided into the following categories: distribution-based、hierarchical-based, density-based, grid-based and centroid-based. K-medoids is an algorithm based on centroid and aims to minimize the distances between points and the center of the cluster.

In clustering problems, we are usually provided with a dataset unlabeled. And we expect to implement an algorithm, to group the data into coherent subsets or clusters automatically. K-medoids is similar to k-means algorithm, and the latter is the most common clustering algorithm, which divide the data into k different clusters. It does two things: First is cluster assignment step, and second is to move centroids, and constantly repeat those two steps until no further improvement could be made.

But K-means is sensitive to outliers for the clustering is based on the mean value of data. Therefore we will take k-medoids algorithm as an alternative. When moving centroids, k-means algorithm will calculate the mean value of the whole cluster to select the new centroid of the cluster to assignment the points again, while in K-Medoids algorithm it will iterate to find the point that lead to smallest sum of distance. The drawback of k-medoid is higher time complexity (square), which make it difficult to deal with huge data sets, that the scale of calculation will become very large.

METRIC:

Sum of Euclidean distance within a cluster

When judging whether a new cluster is better than original one, we compare the sum of Euclidean distance between points and the centroid within a cluster. If the sum is smaller than old one, then replace the cluster with newer one.

WSS (within cluster sum of squares)

The WSS is the sum of the squared distances of all points in the cluster. It is a widely-used metric to evaluate the performance of algorithm. It could help to determine the number of clusters when solving unsupervised learning problems.

$$SSE = \sum_{i=1}^{k} \sum_{c \in C_i} |p - m_i|^2.$$

Among them, C_i is the i-th cluster, p is the sample point in C_i, m_i is the centroid of C_i (the mean of all samples in C_i), and SSE is the clustering error of all samples, representing the quality of the clustering effect.

Pseudo Code of the k-medoids algorithm

And in this project, the algorithm actually used is PAM algorithm, a type of k-medoids algorithm, which is the most commonly used. PAM tries out all of the objects in this cluster as a new medoid that will lead to lower WSS（within cluster sum of squares）.

```
//Clusters: The list to store the points by clusters.
//Centroids: The list to store the centroids.
Begin
    1.Centroids<-INITIALIZE(dataset,k)//Initialize k centroids,Centroids=
{c1,c2,...,ck,ci∈dataset}
    2.NearestCluster<-SELECTCENTROID(dataset,Centroids)//Calculate the euclidean
distance between every point and all the centroids, to choose the nearest one to
join
    3.CLASSIFY(Centroids,dataset,NearestCluster){//Calculate to get the nearest
centroid for each point in dataset
        Clusters[j].APPEND(Point_i) if IS_MINIMUM(DISTANCE(Point_i,Centroid_j))
        return Cluster
        }
        improvement==TRUE
        optimalcost<-METRIC(Cluster)
```

```
loop while improvement==TRUE:
    4.// Calculate to look for a better centroid again in every already formed
cluster
        for cluster in Clusters{
            Point_x<-RANDOM_SELECT(points_in_cluster):
            newCenrtoids<-SWAP(cluster,Point_x) in Centroids//Set the new
selected point as the new centroid
            newNearest<-SELECTCENTROID(dataset,newCentroids)
            newCluster<-CLASSIFY(newCentroids,dataset,newNearest)
            newCost<-METRIC(newCluster)
            if newCost<optimalCost{
                REFRESH(){
                Centroids<-newCenrtoids
                optimalcost<-newCost
                Cluster<-newCluster
                }
            }else{//Greedy search, when no further improvement could be made, we
think we get to an local optimal position then we stop
                improvement<-FALSE
            }
        }
END
```

The candidate points to be selected as the new centroids should be selected from current cluster, and the function to get the set of such points is as follows. This is python code extracted from the project implemented.

```
    def selectPoint(self, medoids, clusters):#Provide a set of points which is
not current centroid in current cluster.
        return [pts for PTS in clusters for pts in PTS if pts not in medoids]
```

Evaluation

In this project, to evaluate the performance of the algorithm, classification accuracy, time complexity, and human's intuitive perception of output images are taken into consideration.

K-medoid clustering is usually used for unsupervised learning and data sets are often not labeled. But if we have such a data set with labels, we could calculate its classification accuracy. The percentage of the results that predicted the correct results in the total sample:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

And for time complexity, the times of iterations could show the scale of calculation. In real world time complexity is also an important metric that influence the scope of application of different algorithms.

Data processing

First, load the dataset from local directory. For txt format files such as the first dataset utilized in this project, when its data np.loadtxt() could read and convert data into numpy arrays. But when the type of data is a string, using np.loadtxt() function will lead to value error, because it cannot convert string to float. We need to implement a method to process the data.

```

```

```
    # To process and extract data from dssp and stride. Returns: data(array),
labels(list), dataframe
def processIris(path):

    with open(path, 'r') as f: # read the file in lines and store lines in a
list
        data = f.readlines()
    chart=[]
    for l in data:
        l=l.strip('\n')
        chart.append(l.split(','))#Split the data connected with ','
    df = pd.DataFrame(chart,columns=['sepal_length', 'sepal_width',
'petal_length', 'petal_width', 'class'])
    res=[]
    labels=[]
    for instances in chart:#Store data and label seperately.
        res.append(instances[:len(instances)-1])
        labels.append(instances[-1])
    res=np.array([ [float(attr) for attr in inst] for inst in res])# convert
string items to float and convert the whole list to numpy array.
    return res,labels,df
```

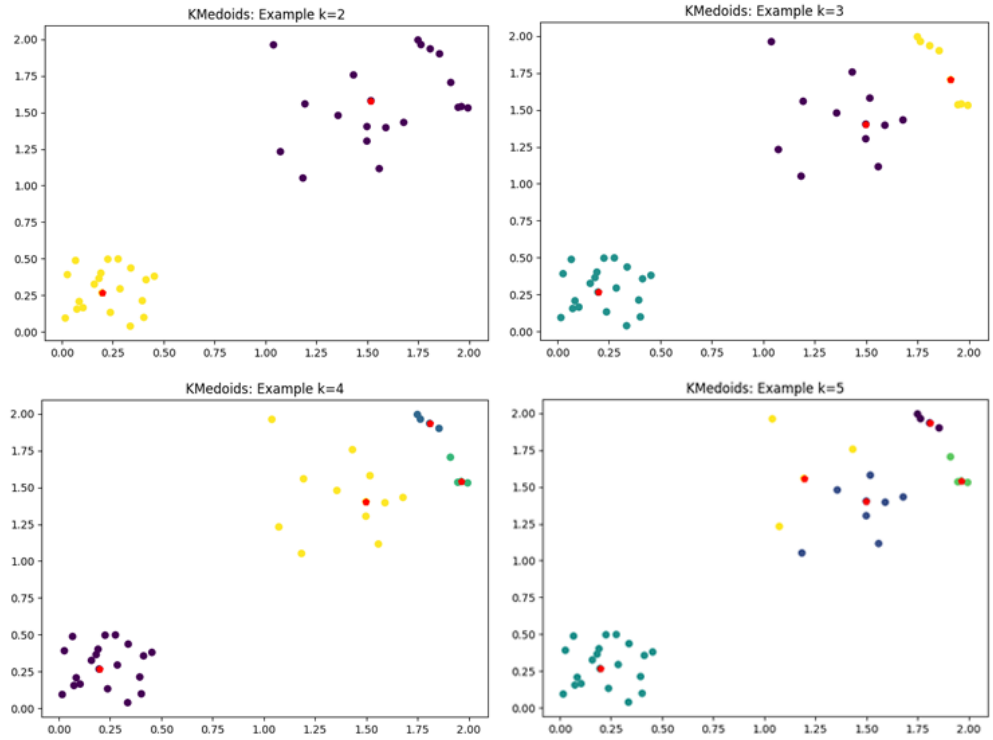
This function could read the data set file line by line, and split each line into list, finally convert to arrays made up with float numeric values. It returns the data numpy array, the labels in list and a dataframe to show the dataset directly.

Experiment

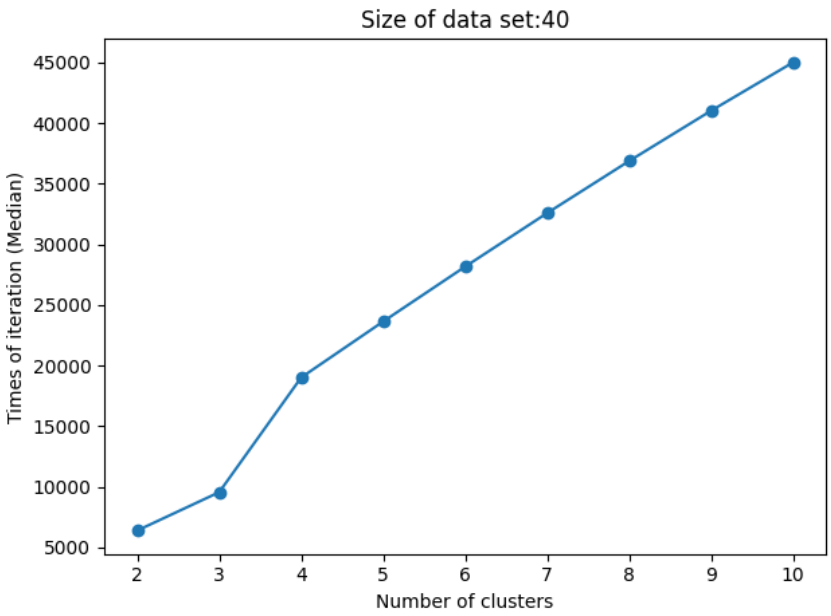
Environment: Pycharm with python 3.6. The experiment has several different parts. First test k-medoid (PAM) algorithm on three different data set, to show the performance of the algorithm when the scale of data set varies. Then compare time complexity and clustering effects between k-medoids and k-means, to discuss on their advantages and disadvantages. Finally experiment on k value selection, which is commonly called "elbow method".

Example small data set

Read and process the file “exampledataset.txt” with np.loadtxt() function, run k-medoids algorithm on this data set. Change k to different values, and it will lead to the change in the number of clusters generated.



As it is shown in the figure, according to the k value, the algorithm will define k centroids, and form clusters around the centroids in red. Different clusters are dyed different colors. And times of iteration changes, iteration here means the how many times the algorithm operate a loop.



Clustering for classification on Iris data set

```
We have labels in this data set, and therefore we could make classification with
clustering and see the performance of the algorithm.
```

As for Iris data set, there are three labels and thus we need to let the algorithm divide the dataset into 3 clusters.

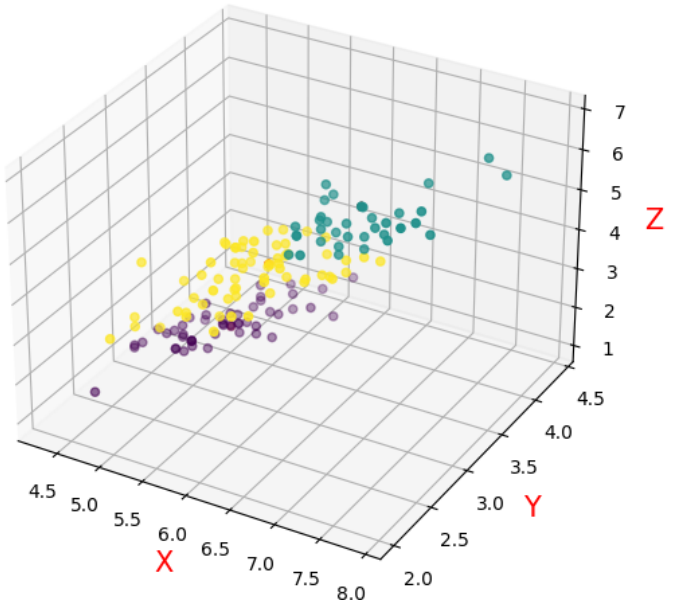
And we make prediction on the classes of iris. Accuracy is shown in the console (89.33%).

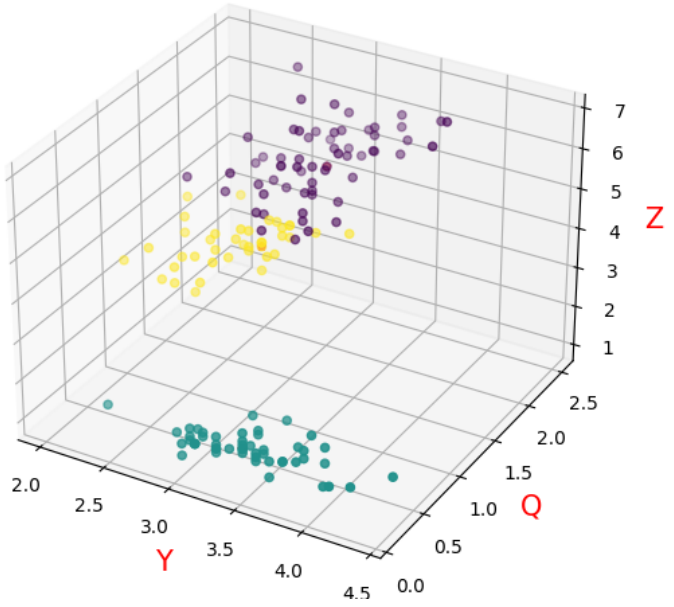
```
[IRIS]number of clusters: 3
[IRIS]Times of iteration: 67473
Original Labels:  ['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iri
Prediction:  ['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-set
The accuracy of classifying iris is 89.33333333333333%
```

But when clustering again, the result varies, and accuracy changes (90.67%). The reason for this phenomenon may be because the selection of centroid points is random, and according to the algorithm, the loop stops when no further improvement could be made, which means the result is a local optimal result.

```
[IRIS]number of clusters: 3
[IRIS]Times of iteration: 95013
Original Labels:  ['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa'
Prediction:  ['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Ir
The accuracy of classifying iris is 90.66666666666667%
```

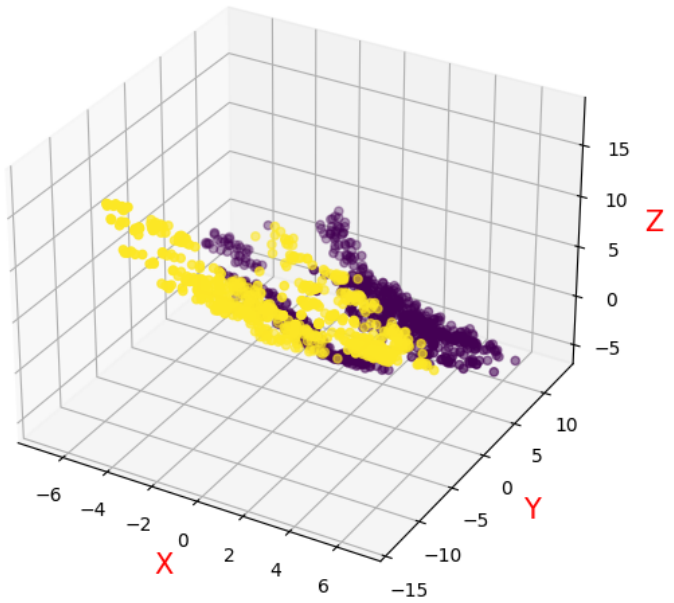
The data has four attributes, and therefore grouping is based on the distance between those 4 dimensions arrays. And the result is show as follows: (X=sepal length; Y=sepal width; Z=petal length; Q=petal width)





Banknote dataset

For some data set like Iris, points are clearly divided into distant groups in certain dimensions, then we may consider this algorithm is suitable for such data sets. But sometimes the relationship between the features and classification is more complex that it cannot be simply grouped by the distance of arrays, K-medoids algorithm may fail.



It seems that the samples are properly divided into two clusters, but when comparing original labels with prediction, and look at the accuracy (55.54%) then we could find that the prediction on this data set is not accurate.

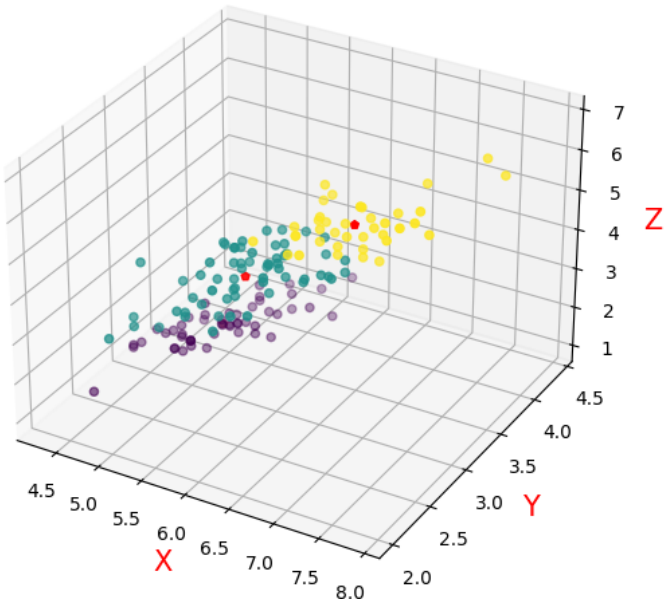
And we could discover from the output that, for a data set in size of more than 1000 instances, the scale of calculation will become extremely large. For allocating points to clusters requires constantly calculate the distance between the centroid and every other points.

```
C:\Users\Souta\Anaconda3\python.exe C:/Users/Souta/PycharmProjects/TAIwithDataset/main.py
[BankNote]number of clusters: 2
[BankNote]Times of iteration: 7502040
Original Labels:  ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '
Prediction:  ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '
The accuracy of classifying iris is 55.53935860058309%
D:\PyCharm 2020.2\plugins\python\helpers\pycharm_matplotlib_backend\backend_interagg.py:68: UserWarning:
  self.figure.tight_layout()
```

Compare with K-means

The main difference between k-medoids and k-means is, when selecting centroids, the position where is the mean value will be selected to be the center of cluster, while in k-medoids the algorithm should select a instance which represents the median value.

Compare with the result of k-medoid mentioned before, k-means could also work well and take much less time complexity. And therefore, as for iris data set, k means is also an appropriate learning algorithm.



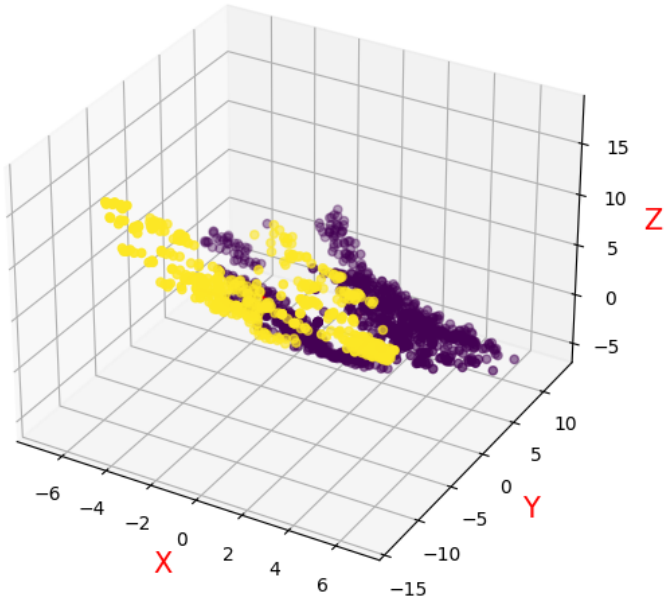

```
C:\Users\Souta\Anaconda3\python.exe C:/Users/Souta/PycharmProjects/
D:\PyCharm 2020.2\plugins\python\helpers\pycharm_matplotlib_backend
  self.figure.tight_layout()
number of clusters:3
times of iterations:3947

Process finished with exit code 0
```

And for the data set in larger size (more then 1000), the advantage on time complexity become more obvious. And could divide points into two cluster as what k-medoids did. Time complexity of k-means is much smaller than k-medoids.

```
Run:  main
C:\Users\Souta\Anaconda3\python.exe C:/Users/Souta/Pychar
D:\PyCharm 2020.2\plugins\python\helpers\pycharm_matplot
  self.figure.tight_layout()
number of clusters:2
times of iterations:56832

Process finished with exit code 0
```

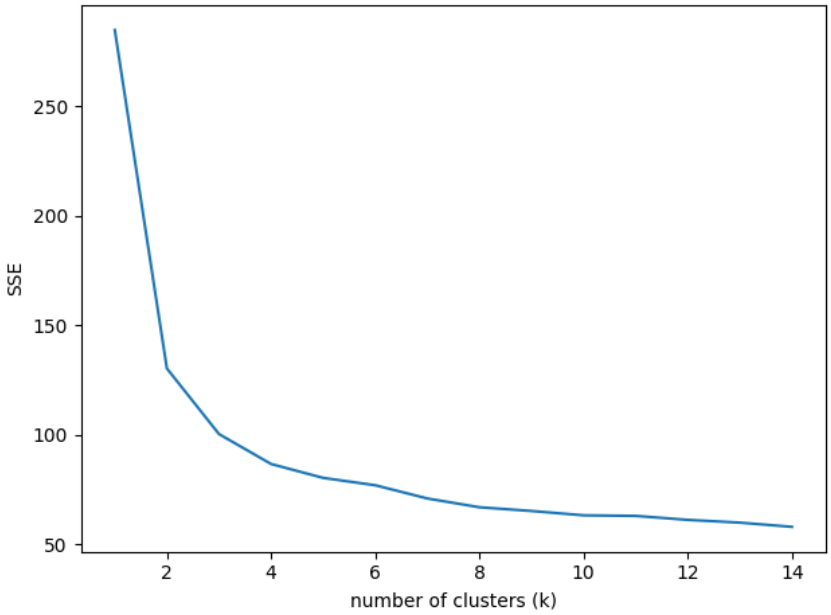


K-Value Selection

In clustering problems, it is usually the case that the more clusters we choose, the less value of SSE. But too many clusters will lead to over-fitting, which means that we need to stop increasing the number of clusters at a proper position.

To select a proper parameter for a data set in unsupervised learning problem, **elbow method** could help to select optimal values. We could use elbow method to get a proper k value when manipulating k-medoids clustering.

We ignore the actual number of classes of Iris data base and try to cluster the points with different k values, get the SSE of the clustering result, and plot the curve which shows the relationship between k and SSE. And it could be discovered from the curve that a turning point appears between 2 and 4, which means that 3 is the optimal k value for this data set when taking k-medoids clustering.



Discussion and Conclusion

In this project we implemented and tested on K-medoids algorithm by clustering data from three different data set, showing the functionality of this kind of clustering algorithm and its shortcomings as well.

Should we split the data set for training, validation and test? Clustering methods such as k-medoid are usually used for unsupervised learning, where there is no label and it should automatically classify or group the input data. And therefore there is no need for splitting the original data set into train and test set.

Not suitable for large data set. As we can discover from the experiment, for small data set like the first example data set and iris data set, the scale of calculation could be restrained to a low level, but for banknote data set the time spent will be much longer. When comparing with k-means the difference is more obvious. It is because k-medoids have a much bigger time complexity than k-means that k-means' time complexity is linear while k-medoids' is square. In k-means we get new centroids by calculating the mean value of objects in a cluster, while for k-medoids we need to select a point whose sum of distance to other points within the cluster is smallest. Selecting a point from the set of points increases the number of layers of the loop by one layer.

K-medoids reaches a local optimal rather than global optimal. It is shown in the algorithm that when no further improvements could be made, the searching stops. Which means that the algorithm is likely to give a solution which is not the best actually. This is one of the problems. And it may be one of the factors to influence the classification accuracy.

Elbow method And for k value selection, when we know the exact number of classes, we could set it as k value. When we need to experiment to get appropriate k value, elbow method could work, but whether elbow method works depends on the data set. If the data set we are working with has a clear distinction between groups, then we could observe a significant turning point, in this case we could determine k value easily. If the curve changes smoothly, then it will be difficult for us to decide which is better for k value. This is another problem for k-medoids algorithm.

Trouble with labels in classification problems. For classification when we have labels in our dataset, a challenge for the algorithm is, the output of the algorithm is a list of number, which represents the order of centroids in the list stores centroids, it is cluster number rather than the actual class of object. Therefore we have to convert cluster numbers to real classes. In this project we determine the class of a cluster by looking into the actual class of the centroid points. One feasible alternative solution is to count the actual labels of the points in the cluster, and take the category with the most occurrences as the class of the cluster.

All in all, k-medoid could perform well when dealing with small dataset, but when the size of data set is too large, time complexity will become a main constraint. And for different data set the effect of algorithm varies distinctly.

References