Report

K-means

1641469 Long Cheng

1. Introduction

This project implement the K-means approach based on two data sets from different problem domains, both these datasets are download from UCI Machine Learning Repository, one is *iris* and the other is *seeds*. As we know, k-means is a clustering approach, in this exercise, I plotted the clustering result to make it visible. In order to get more accurate result, the data preprocessing operations is also used in this exercise.

2. Dataset

In this exercise, I used two different datasets, both of them are suitable for clustering.

2.1 Iris data

The data I used is download from: http://archive.ics.uci.edu/ml/datasets/Iris, named *Iris Data set*. It has three kind of Iris plant, together 150 instances, each class have 50 instances. The first type is linearly separable from the others 2, but the latter are NOT linearly separable from each other. And this data has four

attributes, just some feature about different iris. The main characteristics is mentioned below:

Number of Instances	150
Number of Attributes	4
Missing Values	NO

2.2 Seeds Data Set

The data I used is download from: http://archive.ics.uci.edu/ml/datasets/seeds, named *seeds Data set*. This dataset examined three kind of seed's kernel, 70 elements each, together 210 instances. The seed's kernel information contains seven attributes, mostly calculate by some biologists. The main characteristics is mentioned below:

Number of Instances	210
Number of Attributes	7
Missing Values	NO

3. Data preprocessing operations

We know that the performance of k-means approach can be influenced by the dataset, if one data has two different classes and they are not linearly separable, the performance can't be guaranteed. In this exercise, I used two

kinds of method to enhance the performance. One is normalization, and the other is wipe out outliers.

3.1 Normalization

Both data we chose have many kinds of features, all of them are numbers, but have different ranges. If we want to do k-means on them, first we should normalize these data into same range, like [0,1]. There are many normalization methods, here I chose is min-max normalization. For each attribute, it's easy to calculate the minimum and maximum value. For each value x in this attribute X, use the formula below:

$$x = \frac{x \quad - \min(X)}{\max(\mathrm{X}) - \min(X)}$$

After deployed it on all the attribute, all the data we have is range from 0 to 1, which can give better performance of K-means.

3.2 Wipe out outliers

The aim of k-means is divide the data into different clusters. Intuitively, in one class, if there are some samples' data great different than others in this class, they will have great impact on the performance. In this exercise, I wiped out these outliers. The main process has four steps:

1. For each class, summary the data of all the instances, calculate the center point C.
2. For all the instances, calculate their Euclidean distance to C, marked as L.
3. For L, calculate its mean value M and variance V.

4. Wipe out the instance whose Euclidean distance is not in the range between M±10*V.

4. K-means steps

To implement K-means approach, there are four steps, randomly choose K points, divide all point into K closest, choose another K points to be the new center, calculate SSE (sum of the squared error) based on the closest and new center point, loop this process until the SSE stay invariant.

4.1 Random K point

From the dataset, we know the value of K, which is the number of categories in the instances. This save time and energy for us to calculate the K. For the first K center point, randomly choose them from all the points to get start.

4.2 Build K closest

After we have K center point, for all the point, we calculate the Euclidean (L2) distance to them. The L2 formula is:

$$\mathrm{L2} = \sqrt{\sum_{i=1}^{n}(x_i - c_i^k)^2}$$

Here n is the dimension of x, x_i is the dimension i-th value of x, c_i^k is one of K center point.

For all points, they have K L2 distances, choose the minimum one to join that closest. Traverse all the points to finish the division.

4.3 Find new K center point

Now we have K closest, from the SSE formula below, it can be shown that the mean of the points could minimizes the SSE of the cluster.

$$SSE = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mathbf{c}_i)^2$$

After calculate the derivation of SSE by $\mathbf{c}_i$, the new center point is calculated from:

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

Among these formula, x is a data object, C_i is the i-th cluster, $\mathbf{c}_i$ is the centroid of cluster C_i, d is the Euclidean (L_2) distance.

4.4 calculate SSE

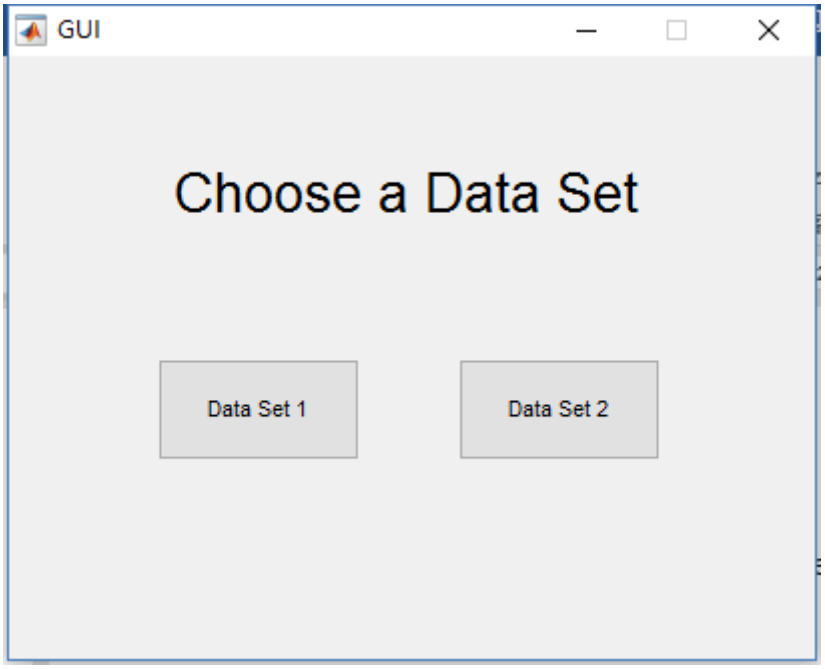
SSE means sum of the squared error, as we mentioned before, it's calculated from sum of all points' L2 distance from their center point.

Loop to 4.2, use new K center points to get new K closest. Loop this until SSE not change a lot. In my realization, I set the tolerance here is 1e-3.

5. Clustering Result

5.1 Main GUI

At first, run the GUI of this exercise, the GUI like below:

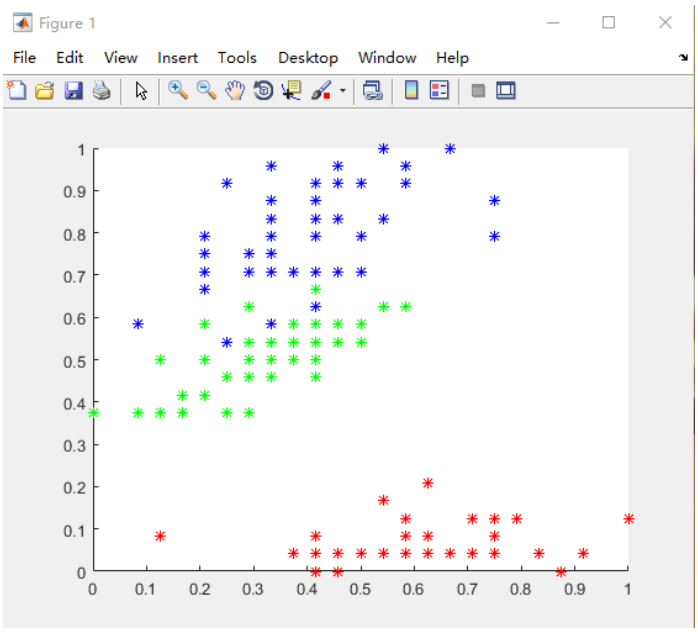


Choose the data set you want to demo.

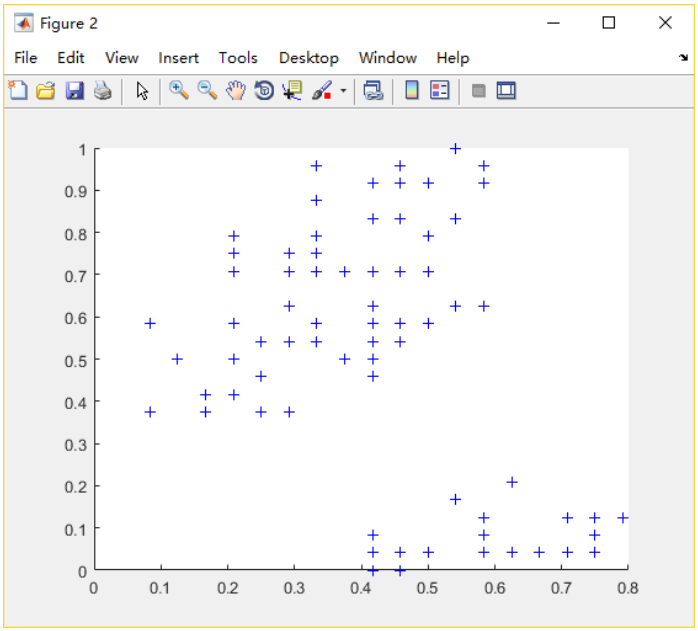
5.2 Result of Iris Data

In Iris, there are 4 dimensions, I used all of them in calculation. To show the result, I only choose two dimensions. The result shows as below:

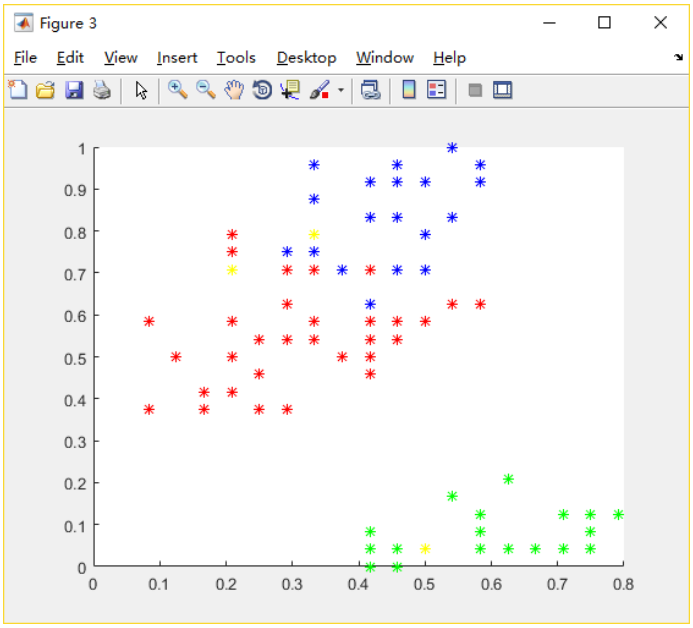
First, the original data, only two dimensions plotted:



Second, the remained points after wipe out outliers:



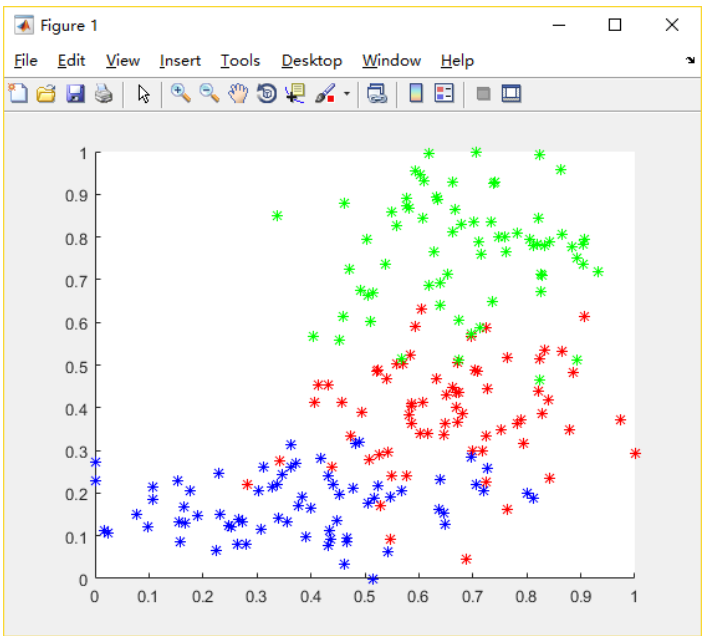
Third, the clustering result, different cluster point out by different color:



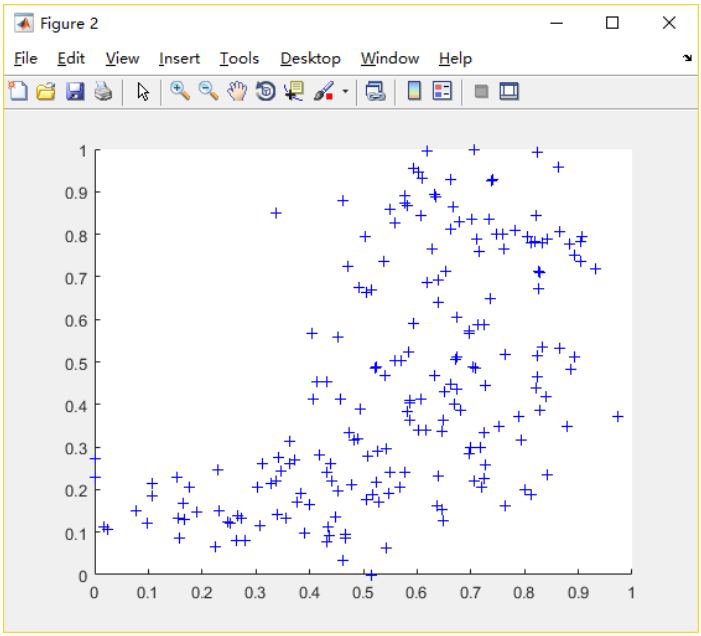
5.3 Result of Seeds Data

In seeds data, there are 7 dimensions, all of them was calculated by K-means. But to show the result, I only choose two dimensions. The result shows as below:

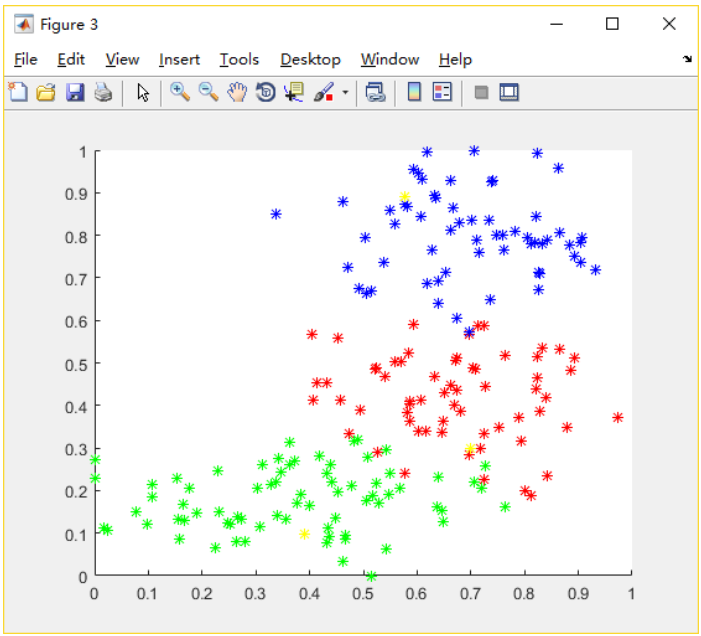
First, the original data, only two dimensions plotted:



Second, the remained points after wipe out outliers:



Third, the clustering result, different cluster point out by different color:



6. Limitations and Possible improvements

6.1 Limitations

For iris data, sometimes, in a very low chance, it will calculate two different class into one cluster, after long time debugging, I found that the reason of this mainly come from the first random center points. As I know, in Iris, one class can be linearly separable from the other. If the K point you randomly chose was both that class, the result is not very nice.

Because some classes have superposition part, for the point in that part, it's hard to divide them into correct cluster.

6.2 Possible improvements

We know that K-means choose first point randomly, maybe we can change it more reasonable, the first K point must have distance.

Deploy some hierarchical clustering to calculate the K value.

And also, for different data, try L1 distance may improve the time-consuming or accuracy rating.