CS 565 Project1

Kaikang Zhu

Analysis of K-means experiments

1.Distance function.

The Euclidean Distance measures the shortest distance between two points. Its formula is showed below.

$$d(x,y) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

The traditional K-Means algorithm is implemented by L2 distance. L1 just adds the difference between points. It's not very precise to reflect the distance between two points. And L2^2 distance is more sensitive to outliers. So I choose L2 distance to be the distance function of K-Means and K-Means++. And it turn out to achieve good results after implementation.

2.Normalize

For the two datasets, they need to be preprocessed differently. For the wine dataset, different columns have different value ranges, so we should normalize them into a given range. I choose (0,1). The normalization formula I use is the following one.

$$X_{scaled} = \frac{X - X.min(axis=0)}{X.max(axis=0) - X.min(axis=0)} \cdot (max - min) + min$$

To realize the normalization, I use some external library to normalize the values into (0,1). Then values can be compared better. The library I use to normalize is minmax_scale() (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html).

For the churn dataset, it has both categorical data and real value data. I first turn the categorical data into numerical data, then concatenate it with real-value data to form a processed churn dataset. After that, I also turn them into a (0,1) range.

3.Find the optimal k

We only find optimal k for the wine dataset.

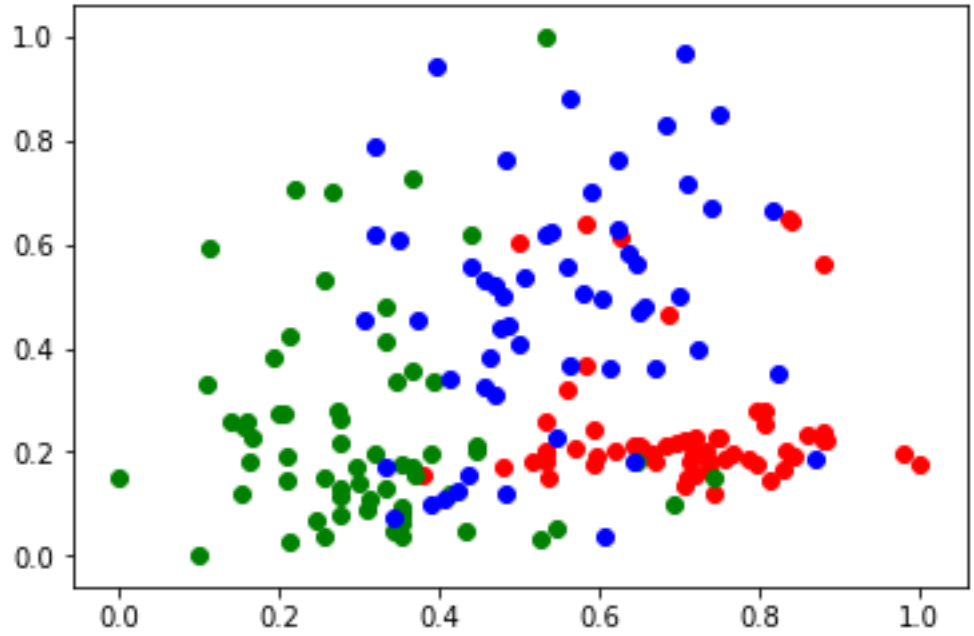
Every time k adds 1, the total error of k clusters decreases. When the k has achieved the best number, the decrease trend will slow down. This can be proved by the following chart.

k	error
2	103.36963302
3	88.75357266
4	85.28941325
5	83.88000748

As we can see from the chart, when k changes from 2 to 3, the error decreases a lot. After that, the decreasing trend slow down apparently. So the optimal k is 3. After implementation, the cluster result is also very stable.

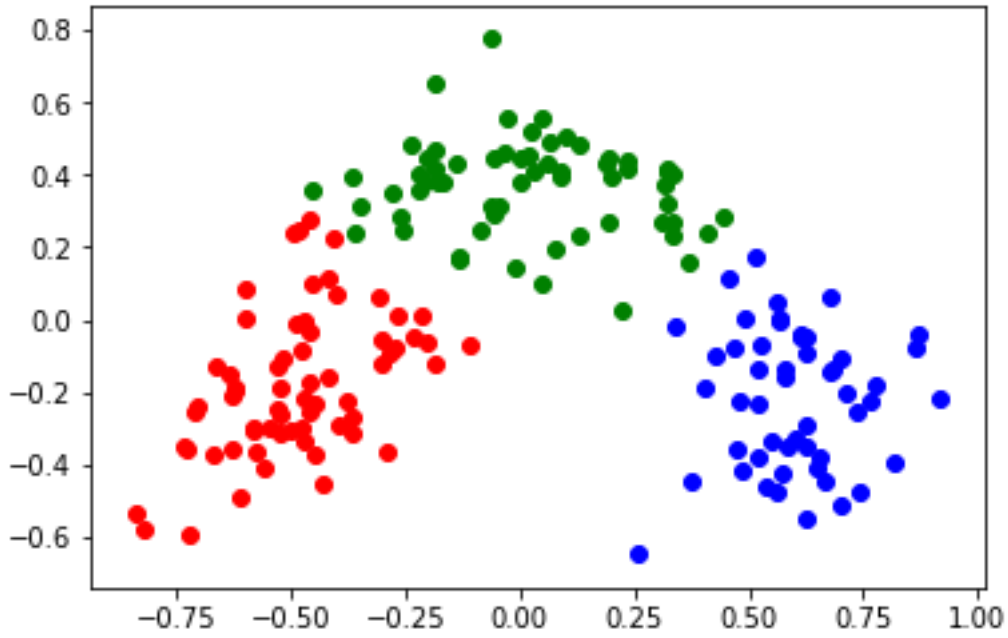
4.Implementation of K-Means
First of all, we should initialize k points randomly selected from the dataset. Then, we compare each point with k centers and select the smallest one. After one loop, we get the k clusters. Continue to loop to get clusters until the present clusters and former cluster are exactly the same. Finally, we get the correct clusters.

The following is the plotted image.



The points aren't clearly clustered into 3 clusters. The effect aren't very good.

5.PCA
Due to the original data is 13-dimensional. We use PCA to transform the data into 2-dimensional. The library I use here is sklearn.decompositon.PCA. It's easy to implement and it can successfully reduce the dimensions of each wines to 2. After using PCA, we can get a better clustering results.



Analysis of K-means++ experiments

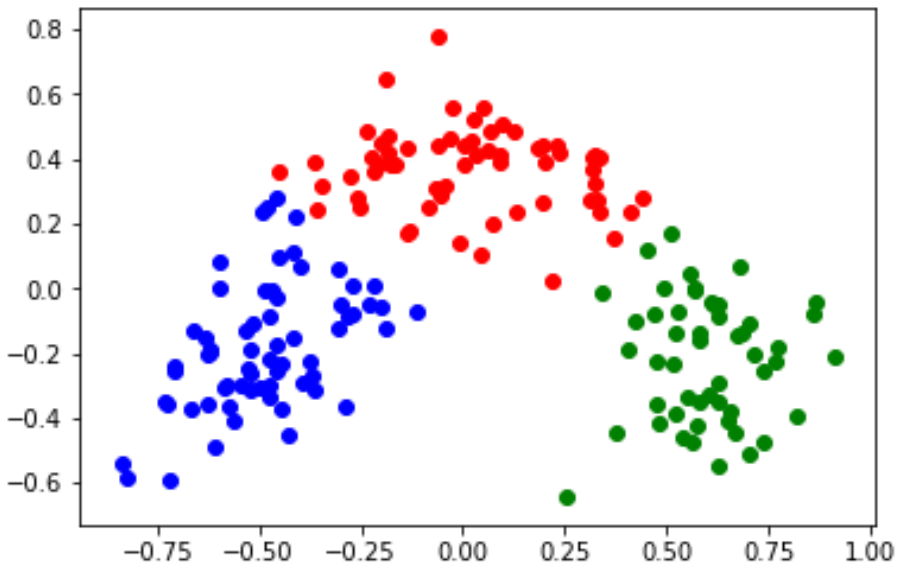
The only difference between K-Means and K-Means++ is the initialization of k centers. After randomly selecting a center as the first center. We calculate the closest distance of the rest points to the centers and calculate the probability of each point of being selected. The most farthest point has the biggest probability to be selected as the next center.

After implemented the K-means++, we can get smaller error and faster convergence speed.

k	error
2	61.61591852
3	39.34285001
4	34.25909546
5	30.77403395

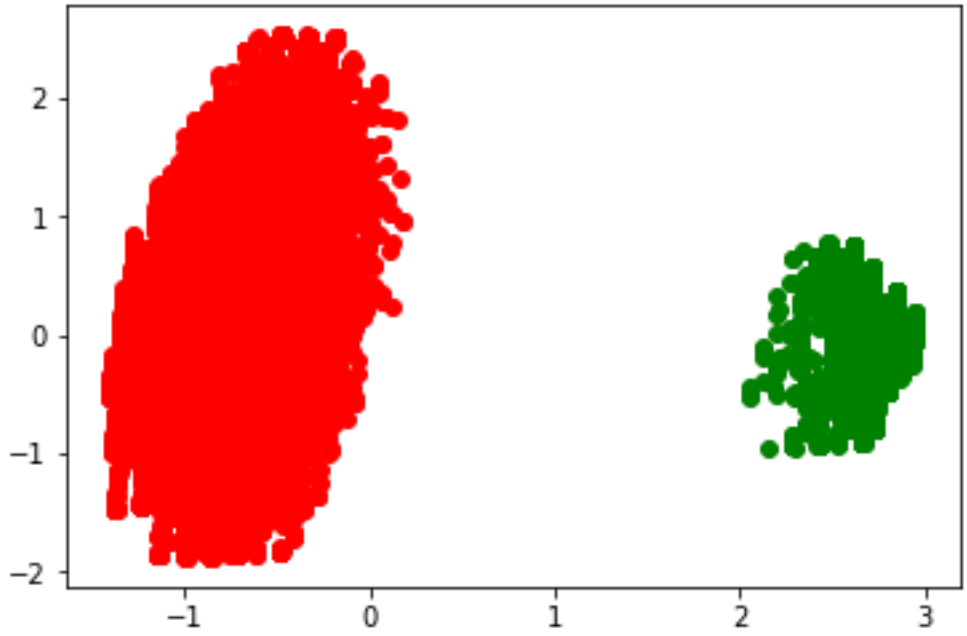
For the K-Means++, the optimal k is also 3.

The clustering result:

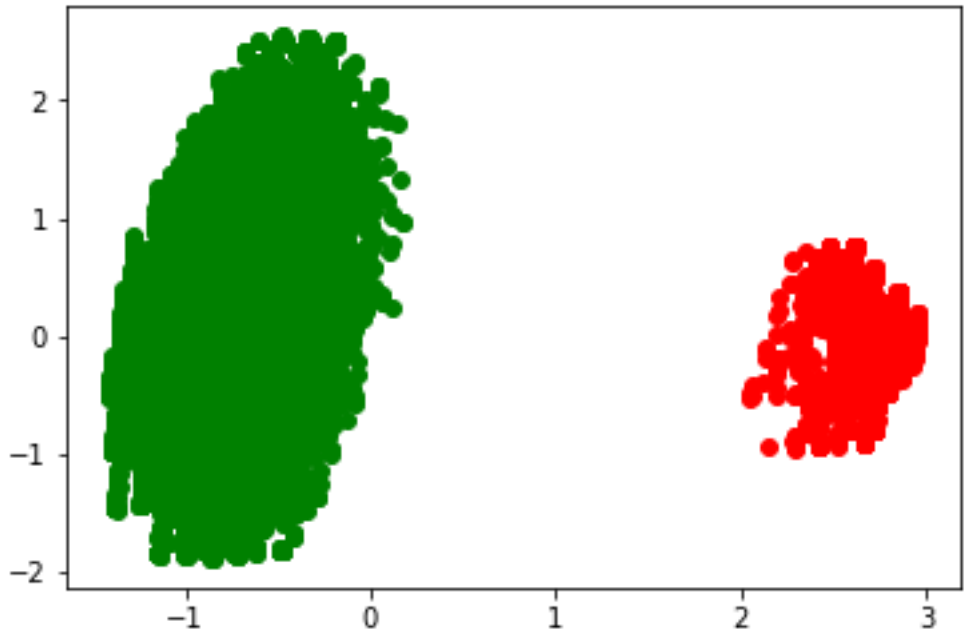


Implementation on churn dataset

We have to preprocess the churn dataset first. There are several two-valued, three valued columns. Using LabelBinarizer from the sklearn library, we can transform the categorical value into binary value (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelBinarizer.html).
After processing column by column, use np.concatenate() to concatenate each column to form a processed churn dataset. Then the process is the same as the wine dataset. And the k is given, so we can reuse the K-Means and K-Means++ code and implement the PCA.
We can get the following clusters:
K-Means:



K-Means++:



Observing from the two picture, the result of two methods is nearly the same.

Summary

In this project, I successfully implemented the K-Means and K-Means++ algorithms on two datasets and get reasonable results. There are still some drawbacks. The code of preprocessing of the churn dataset is a little too long and not neat. I should use some smarter way to preprocess the data. And the clustering result of churn dataset is not exactly the same as the truth dataset which means the algorithm is not good enough.