CS534 Project 3
Kmeans Clustering
Yaonan Zhong
May 26, 2014

1. Initialization. Run your Kmeans algorithm with k = 5 on the provided data1 with randomly initialized cluster centers for 200 times. Each time, you will randomly pick five points in the data to serve as the initial centers and run your kmeans algorithm to convergence. You will repeat this for 200 times and record all the centers that are found.

a. Create a single figure in which the original data points are plotted in one color and the found cluster centers in the 200 runs in a different color. Make sure that the cluster centers are clearly visible from the other points when printed in black and white.

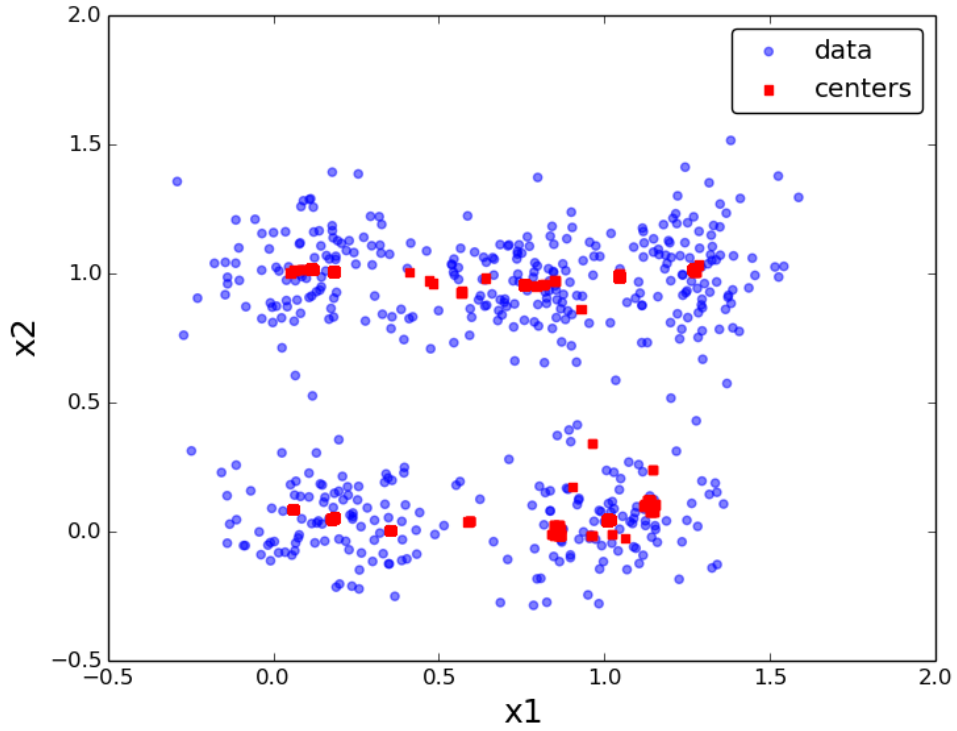

Fig. 1 The cluster centers(square) of 200 runs and the original data(circle)

b. Report the minimum, maximum, mean and standard deviation of the within-cluster sum of squared distances for the clustering found by your algoirthm in the 200 random runs. Please comment on your results regard the sensitivity of the kmeans algorithm to the initialization.

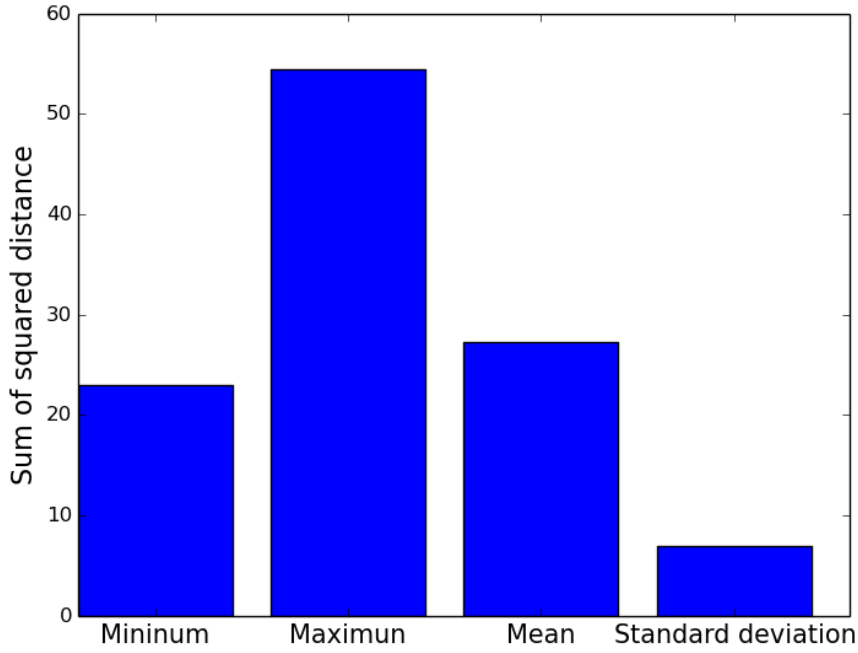

Fig. 2 The minimum, maximum, mean and standard deviation of the within-cluster sum of squared distances for the clusters found in the 200 random runs

From figure 1 and 2 we can see that the clustering results are highly sensitive to the initial seeds. In figure 1, the centers from 200 runs are distributed across the original data. They are inside or between the expected clusters. The difference between the minimum and maximum of sum of squared distance is 30, while the standard deviation is 7. Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum.

2. Finding K. Identifying how many clusters are in the data is a very challenging task. In this task, you will try out a commonly used heuristic on the provided data2. We will consider a variety of different k values (k = 2,3, ...,15). For each k value, you will run your kmeans algorithm with 10 different random initializations and record the lowest within-cluster sum of squared distances obtained for that k value. You will then plot them as a function of k.

a. What trend do you observe as the k increase from 2 to 15? Do you expect this trend to be generally true for all data as we increase k?

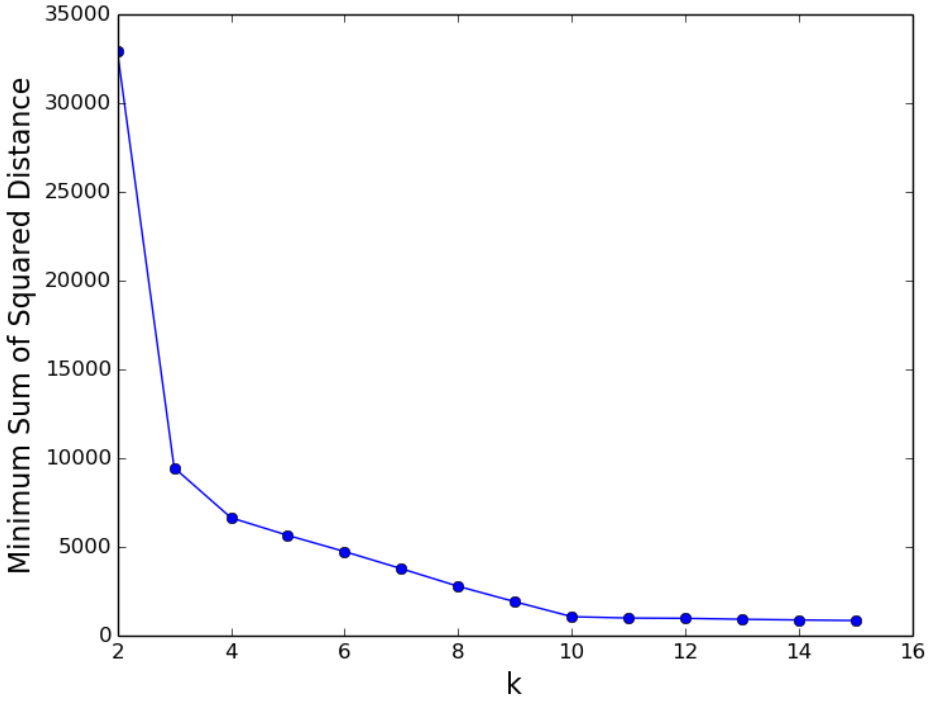

Fig. 3 The minimum sum of squared distance for different k values

Figure 3 shows that the minimum sum of squared distance decreases as the k increases from 2 to 15. And we can see there are four different slopes at 2 to 3, 3 to 4, 4 to 10, and 10 to 15. It is generally true that increasing k can reduce the sum of squared distance. And the extreme case is that k equals the number of data samples, which means one cluster per sample and leads to zero sum of squared distance.

b. A commonly used heuristic is to look for the “knee” in this curve, which is the value of k where the rate decreasing sharply reduces. Find the knees in your plot and provide an explanation to your finding. Why do we see multiple knees? what do they correspond to?

There are three knees in the above curve with k=3,4,10. This behavior is because the suboptimal partitions are found in our dataset. We can clearly identify this result after plotting the cluster centers at these knees as figure 4 shows. Our original data can be divided into three main clusters such that they are similar with the vertices of a triangle. At the same time it can also be divided into 10 smaller clusters for all three vertices. In the first case k=3, and the later one k=10. The 10 small clusters include 4 ones at the top vertex, and 3 ones for both bottom vertices. That’s why there is also a knee at 4 besides 3 and 10. When k=4, there are two centers at the top vertice, and two centers for the bottom vertices.

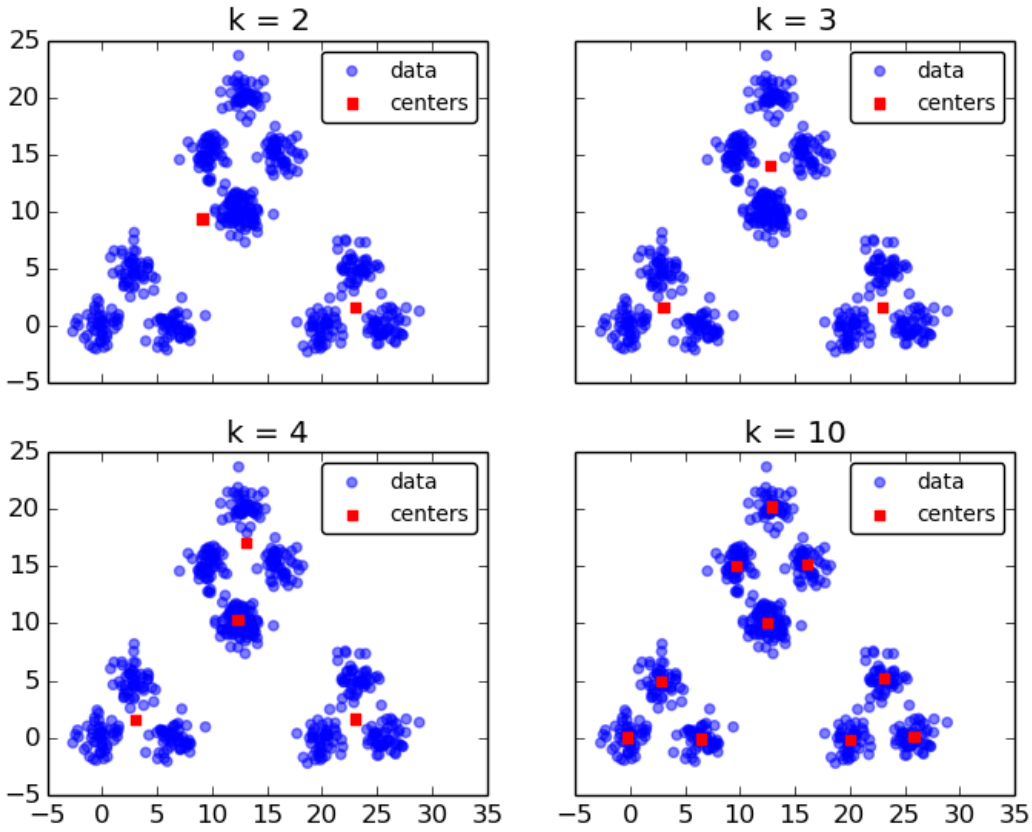

Fig. 4 The original data and cluster centers at k=2,3,4,10

3. Feature normalization. Kmeans clustering and many other clustering methods are distance based. Distances are sensitive to feature processing such as normalization. For the given data3, please do the following.

a. Apply your implemented kmeans algorithm with k = 2 for 200 different random initializations. Record all the cluster centers that are found in these 200 runs. Plot the original data in one color and plot the cluster centers in the same figure using a different color(and shape if that helps to differentiate them).

b. Now normalize the features of the given data set. That is, first center the data to have zero mean(by subtracting the mean from the data), then rescale each feature dimension to have unit variance. Redo part a with this normalized dataset. Report the difference that you observe. The results should be different with and without normalization. Note that this should not be taken to mean that data always need to be normalized. In some cases, the difference in scale can be truly meaningful and normalization could remove such useful information. In other cases, it is crucial to properly scale the data. The decision should be made in a case-by-case fashion.

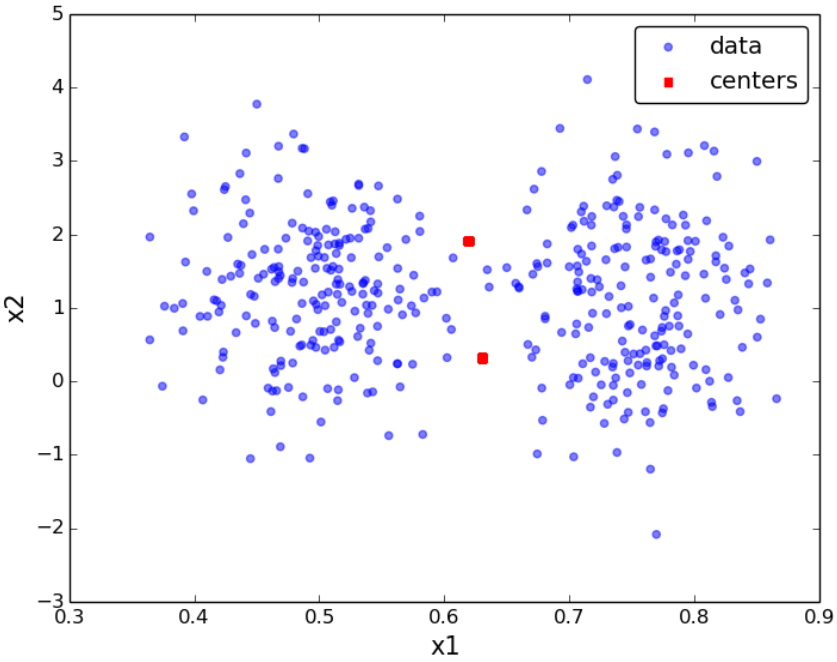

Fig. 5 The original data and cluster centers before feature normalization

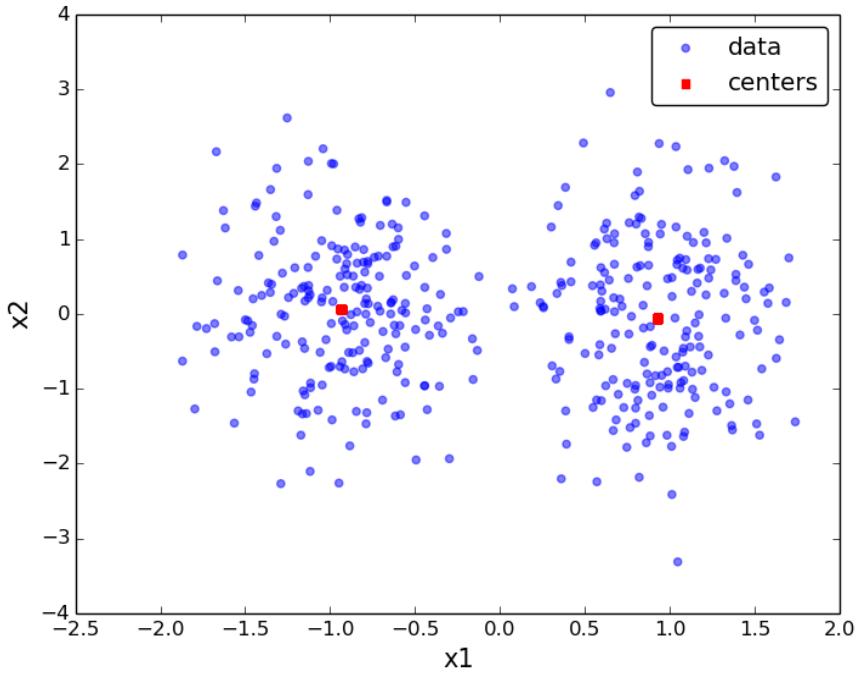

Fig. 6 The normalized data and cluster centers after feature normalization

Figure 5 show the original data and cluster centers before feature normalization. We can see the result centers are not inside the expected clusters, but in the gap between the two clusters. And

the clusters tend to be separated along the feature x2. It is because the two features have different scales. The value of feature x2 is much larger than that of x1, which leads to the unequal variances along both features. K-means clustering is "isotropic" in all directions of space and therefore tends to produce more or less round (rather than elongated) clusters. In this situation leaving variances unequal is equivalent to putting more weight on variables with smaller variance, so clusters will tend to be separated along variables with greater variance.

Figure 6 shows the normalized features and cluster centers after feature normalization. In such case both features have the same scale and the cluster centers are correctly found since the weights of both features are equal now.

Source:
1. initialization.py, initPlot.py, kmeans.py
2. findK.py, findKPlot.py
3. norm.py, normCenters.py, normPlot.py