**CS405 Machine Learning**

**Lab #2 Clustering**

**Introduction**

Different from *classification* which asks us to classify a set of data into several specific groups. Clustering requires us to put the similar data together, hold on here, make yourselves clear, after clustering, what you get is several clusters, you don’t know what the clusters represent.

Let’s see an example. There is a group of students, we want to divide them into subgroups, so that the members in each subgroup can be as similar as possible. Obviously, the result depends on how you define “similarity”. For example, the easiest way is to divide the students by sexuality. We use (M, F) to represent a student (boy is (1, 0), girl is (0, 1)). But this feature is too simple, we don’t even need to do clustering. What we usually do to use more features, like height, weight, grading, etc., and map the data to N dimension vector space before clustering.

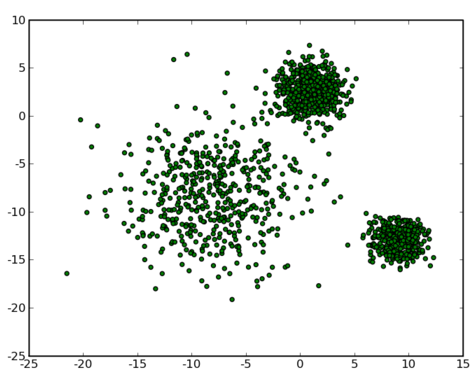


Fig.1

So now, our problem is what “similarity” is, and how to measure and qualify “similarity”. To simplify the problem, assume that we already map the data set to a 2D vector space, as shown in Fig. 1. We can easily tell there are roughly 3 clusters by intuition, two are dense, one is sparse. How computer knows there are three clusters. We have the following algorithms.

**K-means**

The basic assumption of K-means is in every cluster, we can find a center, so that the distance between every point in this cluster and the center is the minimum distance compare to other centers. In math word, it can be interpreted as,

There are N data points need to be clustered into K groups, K-means is to minimize

where

And is the center of *kth* cluster.

It’s not easy to minimize *J* directly, so we use an iterative refinement technique. First, we fix the center , and minimize . Easy to see that, if we cluster every point to their nearest center then *J* is the minimum. Next step is to fix , and minimize . Take derivative of *J* to , and make it zero, so that

which is the average of all the points in *cluster k*. Because for now every step we get the minimum of *J*, after every iteration *J* only can decrease to minima.

The procedure of *K-means* is

1. Initialize ;

2. Cluster every point to their nearest center;

3. update using average ;

4. repeat from step 1.

*Exercise 1.*

Implement K-means using Python (or MatLab).

Note: You are not allowed to use the existing function for k-means in Python (or MatLab).

Tips: if you choose to use Python, *matplotlib* and *SciPy* is needed.

**K-medoids**

If we see the objective minimize function of *K-means* again, we can find that *K-means* requires the data points in Euclidian space, and calculate the dissimilarity by Euclidian distance. For example, if the data is categorical types of data (like different kinds of dogs), there is no meaning when we calculate Euclidian distance.

In *K-medoids*, we change the Euclidian distance to a dissimilarity measure function

A common way is to use a dissimilarity matrix D to represents , and means the difference between *ith* and *jth* cluster.

*Exercise 2*

Revise your code in Exercise 1 to implement K-medoids. Compare the results of K-means and K-medoids, and comment on the performance (from complexity, and intuition point of view).

**Gaussian Mixture Model (GMM)**

**Spectral Clustering**

*Exercise 3 Image Compression*

Use *K-means* to make a simple image compression.