Possible Improvement of K-means Algorithm in clustering

Zhuyun Chen
CPS360 Machine Learning

Abstract

As a popular machine learning technique, k-means algorithm has been widely used for cluster analysis in data mining. However, k-means clustering has some significant weaknesses: the results of clustering can be highly effected by the initial selection of centers, and sensitive to noises. The motivation of this study is to combine different techniques with original k-means algorithm to strengthen the weaknesses of it and improve the result of clustering. In this study, I tried different methods that combing k-means algorithm with technique of initialization of centers, bisection, and medoid-shift, and compared the results generated by each algorithm.

Introduction

Cluster analysis, or clustering, is a common technique for statistical data analysis. It organizes similar objects in a dataset into same groups, which called clusters. As a main task of exploratory data mining, clustering algorithm has been a popular interest in the field of machine learning. K-means, as a method of vector quantization, is one of the most well-known clustering algorithms.

With that being said, k-means algorithm has many weaknesses. Selection of initial centers dhas a significant effect on the final clustering result, which makes the result unstable for k-means method. Meanwhile, the error of k-means usually can only converge to a local minimum that correlated to the region that is initially drawn by the randomly initialized centers. The method is also sensitive to noises, which causes it to perform poorly on small datasets. To improve the k-means algorithm, a common sense is to combine it with some other techniques that would help it to avoid those weaknesses.

In this study, I tried to combine the k-means method with initialization of centers, bisection, and medoid-shift to improve its performance on clustering problem, specifically image segmentation. The methods I used to combine with k-means algorithm will be describe in following section of this paper.

Dataset and Method Description

Since the main focus of this study will be on the k-means method and possible improvements of the algorithm, it worth a while to explain how the k-means algorithm works. The main idea of k-means is very simple. It takes parameter k as the number of clusters, and randomly assign k objects in the dataset to be the initial center of each cluster. Then for each object in the dataset, the algorithm would calculate its distance to the center, put it into the same cluster with the nearest center, and calculate the new centers with in the clusters. This would be repeated until the different between the new centers and the old centers of each cluster is below certain threshold.

Algorithm 1 k-means algorithm

Require: k, number of clusters; D, original dataset
  Initialize Arbitrarily choose k data points from D as initial centers of clusters
  Associate each data points to the closest mean
  repeat
    for all data points i and center c **do**
      Calculate Euclidean distance between i, c
      Assign i to the cluster with closest center
    Calculate new center for each cluster
  until no change in centers
  return clusters

The first possible improvement is try to avoid results in local minimum. This can be done by assigning the initial center with some reasonable guess rather than totally random selection through the whole dataset. For static, in general, we could assume objects that are farther to each others would more likely belong to different clusters. In image segmentation, data are pixels, which represent by their r, g, b values. Distance between pixels can be calculated using the Euclidean distance of r, g, b values. Below is the formula for calculating Euclidean distance between pixel x and pixel y:

$$d(x, y) = \sqrt{(r_x - r_y)^2 + (g_x - g_y)^2 + (b_x - b_y)^2}$$

The combination of k-means method with initialization technique would choose k pixels that gives the largest Euclidean distance as the initial centers, and perform the regular k-means algorithm.

A different approach to avoid local optimal result is bisection. Instead of getting k centers at the beginning, the

bisecting-kmeans algorithm would see the whole dataset as one cluster and bisecting it into two clusters. For each cluster, it will calculated the error function, and keep bisecting the cluster with larger error, until it gets k centers. The advantage of this method is that it would minimize errors for each bisection step, which would promise global optimal results rather than local optimal.

Algorithm 2 k-means with bisection

Require: k, number of clusters; D, original dataset
  Initialize Let D be the cluster, and find the mean of the cluster
  Associate each data points in the cluster with means
  clusterList = D
  numCluster = 1
  repeat
    tempCluster, tempMSE = -inf
    for all cluster c **do**
      clusterMSE = 0
      m = c.mean()
      for all data point i in c **do**
        Calculate Euclidean distance between i, m
        tempMSE += distance(i,m)
      if clusterMSE $>$ tempMSE **then**
        tempMSE = clusterMSE
        tempCluster = c
    clusterList.remove(tempCluster)
    newClusters = kmeans(2,tempCluster)
    ClusterList.append(newClusters)
  until numCluster $\leq k$
  return clusterList

Another weakness of k-means is the impact of noise in dataset. Just as in math, means values could be easily effected by extreme values. When the dataset is small, noise in the data would make significant changes on the clustering result. As a solution to that, Kaufman and Rousseeuw developed the k-medoid algorithm in 1987 (Kaufman and Rousseeuw 1987).

Algorithm 3 Partitioning Around Medoids (PAM) algorithm

Require: k, number of clusters; D, original dataset
  Initialize Arbitrarily choose k data points from D as initial centers of clusters
  Associate each data points to the closest median.
  repeat
    for all non-medoid data points i and medoid c **do**
      Swap(i, m)
      Associate each data point to the closest medoid
      Recompute the cost
      if total cost increased in the previous step **then**
        undo swap
  until no change in centers
  return clusters

Instead of using means of each cluster to be the center, k-medoid algorithm uses the median to be the center of cluster. There are different implementations of k-medoid algorithm, but the most commonly used one is the Partitioning Around Medoids (PAM) algorithm, which shown above. PAM uses a greedy search which may not find the optimum solution, but it is faster than exhaustive search.

The dataset I used are gif images I pulled from internet and the the gif files from previous homework. I use the cImage package in python to get the r, g, b value of pixels and use them as the test data for algorithms.

Experimental design

In this study, I record the time of each algorithm to compare their efficiency, and use mean square error (MSE) to compare the accuracy of the result. MSE measures the average squared difference between each data point and the center of the cluster that data point belongs to. It can be calculated by the formula below, while X_i is the i^{th} data point in the cluster, and $\hat{X}_i$ is center of the cluster:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2$$

For each dataset, I test each algorithm with k-value from 2 to 20, and compute the total MSE for of all clusters generated by each method. Therefore, the estimator θ of k would be:

$$totalMSE = \sum_{i=2}^{k}\frac{1}{n}\sum_{j=1}^{n}(X_{ij} - \hat{X_{ij}})^2$$

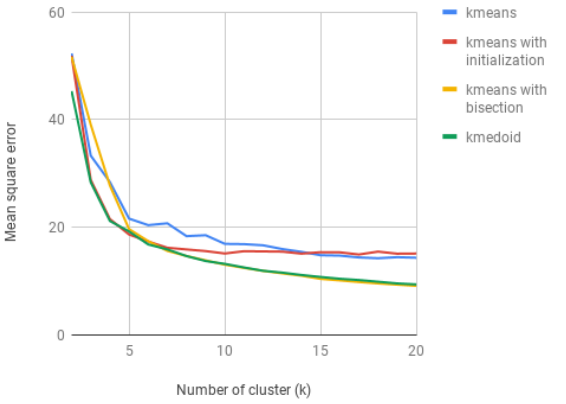
X_{ij} is the j^{th} data point in i^{th} cluster, and $\hat{X_{ij}}$ is center of the i^{th} cluster.

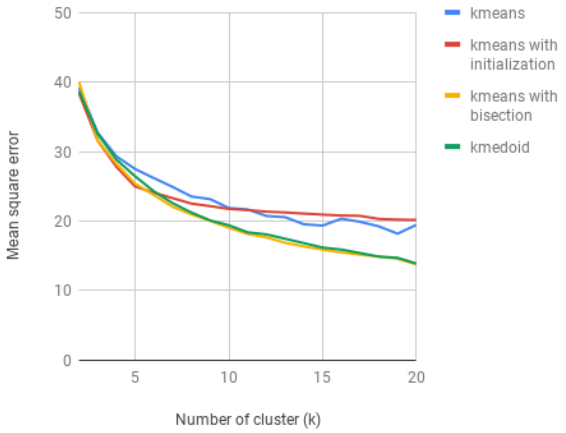
To make sure the universality of the conclusion, I test all the algorithms with multiple dataset, and compare the difference between them for each dataset. For every dataset I choose, I run each algorithm with it for 20 trials, and calculate the average of all trials.

Results analysis

I test the performance of each algorithm with multiple datasets, and in general, the result looks quite similar. Below are the results from three different datasets I choose.

Figure 1.1 to figure 1.3 below shows the graphs of accuracy with each method, and figure 2.1 to figure 2.3 shows the graphs of efficiency with each method. Three graphs from each group indicate the result generated by different dataset.





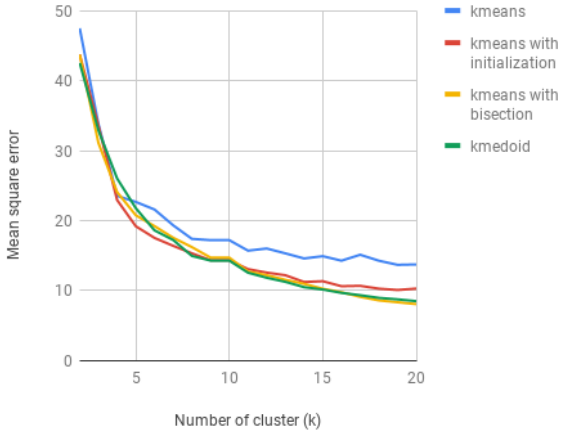
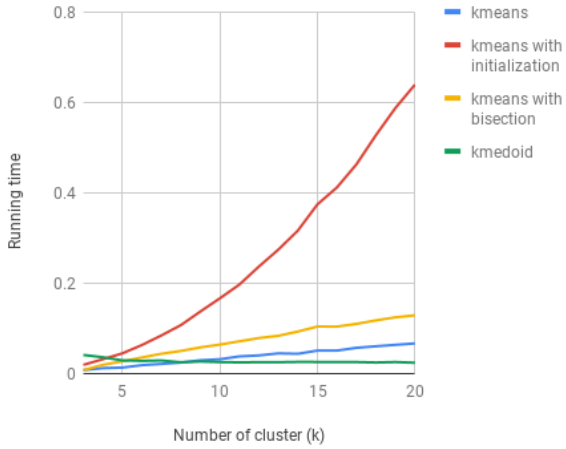
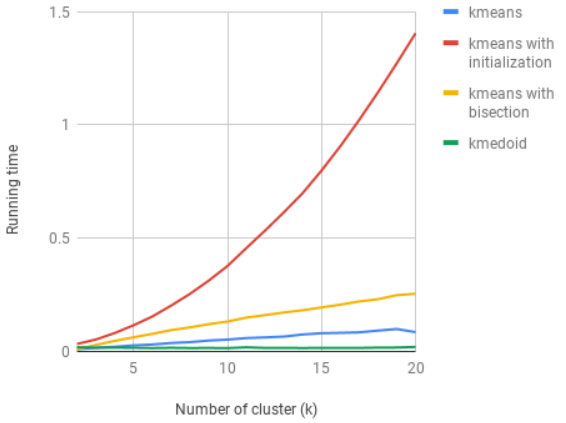


Figure 1: MSE of different k-means-based algorithms





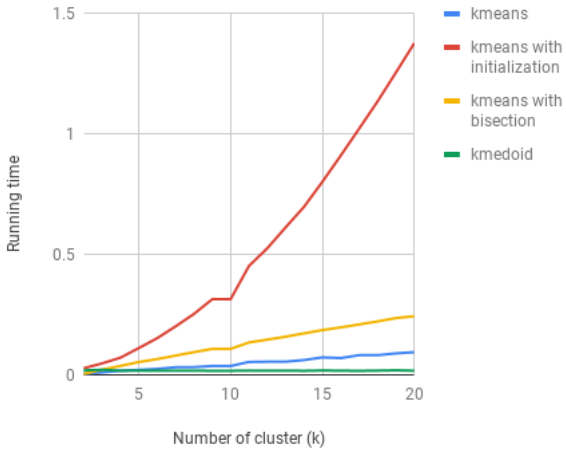


Figure 2: Run time of different k-means-based algorithms

According to the graphs, k-means with initialization seems give some improvements to the original k-means algorithm when the k is relatively small. But when k gets large, the runtime of initialization would increase significantly, and improvement for accuracy does not seems to be significant. Therefore, k-means with initialization could help to improve the result when k is relatively small, but have more cons than pro when k gets larger.

Both k-mean with bisection and k-medoid would improve the results. The run time of k-means with bisection is larger than original k-means algorithm, but k-medoid takes the smallest amount of run time among all algorithms.

Conclusion

In conclusion, all three methods would help to improve the result in some sense. Implementing initialization on k-means would be helpful when k-value is relatively small, but not quite useful for larger k because of computational time. The use of bisection technique in k-means would slightly increase the run time, but it would significantly improve the result. Among all methods, k-medoid seems to be the best in general. It gives better result than original k-means method, and is more efficient than k-means. However, k-medoid does not work for larger dataset due to computational space.

With that being said, among all algorithms I introduced in previous section, k-means with bisection would be the best method to use for larger datasets, while k-medoid would work best for small datasets.

Future work

In this study, I only compare different methods by themselves. However, some methods can be combined together to form new algorithm, and might make more improve to the result. Different combinations of those three method, described in previous section, could be tested in future works.

References

Kaufman, L., and Rousseeuw, P. 1987. *Clustering by means of Medoids, in Statistical Data Analysis Based on the L 1Norm and Related Methods*. North-Holland, 405416.