An Experimental Analysis of K-means Using Matlab

G.Chamundeswari[1], Prof. G. Pardasaradhi Varma[2], Prof. Ch. Satyanarayana[3]

[1]Assoc. Professor, Ramachandra College of Engineering, Eluru, A.P, India.
[2]Professor and HOD, Dept. of IT, SRKR Engg. College, Bhimavaram.
[3] Professor, JNTUK, Kakinada.

Abstract

Clustering problems arise in various areas like pattern recognition and pattern classification, image processing, bioinformatics etc. It is considered that the k-means algorithm is the best-known squared error based clustering algorithm. It is very simple and can be easily implemented in solving many practical problems. This paper presents the results of the analysis of running k-means algorithm in matlab with two UCI repository data sets, iris plant and haberman's survival data.

Keywords— k-means, clustering, UCI repository, k-means-Matlab.

1. Introduction

The aim of clustering is to partition a set of objects which have associated multi-dimensional attribute vectors into homogeneous groups such that the patterns within each group are similar. Several unsupervised learning algorithms have been proposed which partition the set of objects into a given number of groups according to an optimization criterion. One of the most popular and widely studied clustering methods is K-means. In this paper an emphasis on understanding and analysis of k-means in matlab view is presented.

This paper is organized as follows: following the introduction Section2 gives an overview of k-means algorithm, Section3 introduces matlab, the datasets used and interprets the implementation of k-means in matlab, Section4 the experimental results and finally conclusion in Section5.

2. Overview of k-means clustering algorithm

The k-means algorithm can work very well for compact and hyper spherical clusters. The time complexity of k-means is O(N K d)[4]. k-means can be used to cluster large data sets. It uses a two-phase iterative algorithm to minimize the sum of point-to-centroid distances, summed over all k clusters:

The *first phase* uses batch updates, where each iteration consists of reassigning points to their nearest cluster centroid, all at once, followed by recalculation of cluster centroids. This phase occasionally does not converge to solution that is a local minimum, that is, a partition of the data where moving any single point to a different cluster increases the total sum of distances. This is more likely for small data sets. The batch phase is fast, but potentially only approximates a solution as a starting point for the second phase.

The *second phase* uses online updates, where points are individually reassigned if doing so will reduce the sum of distances, and cluster centroids are recomputed after each reassignment. Each iteration during the second phase consists of one pass though all the points. The second phase will converge to a local minimum, although there may be other local minima with lower total sum of distances. The problem of finding the global minimum can only be solved in general by an exhaustive choice of starting points, but using several replicates with random starting points typically results in a solution that is a global minimum.

Given a data set, a desired number of clusters, k, and a set of k initial starting points, the k-means clustering algorithm finds the desired number of distinct clusters and their centroids. A centroid is defined as the point whose coordinates are obtained by computing the average of each of the coordinates of the points of the samples assigned to the cluster. Formally, the k-means clustering algorithm follows the following steps.

1. *Set k*: Choose a number of desired clusters, k.
2. *Initialization*: Choose k starting points to be used as initial estimates of the cluster centroids. These are the initial starting values.
3. *Classification*: Examine each point in the data set and assign it to the cluster whose centroid is nearest to it.
4. *Centroid Calculation*: When each point is assigned to a cluster, recalculate the new k centroids.
5. *Convergence condition*: Repeat steps 3 and 4 until no point changes its cluster assignment, or until a maximum number of passes through the data set is performed.

Before the clustering algorithm can be applied, actual data samples are collected. The features that describe each data sample in the database are required a priori. The values of these features make up a feature

vector (F_{i1}, F_{i2}, … , F_{im}), where F_{im} is the value of the mth feature of the ith job. The feature vector can be thought of as a point in M-dimensional space. Like other clustering algorithms, k-means requires that a distance metric between points be defined. This distance metric is used in step 3 of the algorithm given above. A common distance metric is the Euclidean distance. Given two sample points, pi and pj, each described by their feature vectors, pi = (F_{i1}, F_{i2}, … , F_{iM}) and pj = (F_{j1}, F_{j2}, … , F_{jM}), the distance, d_{ij}, between pi and pj is given by:

$$d_{ij} = \sqrt{\sum_{m=1}^{M} (F_{im} - F_{jm})^2}$$

If the different features being used in the feature vector have different relative values and ranges, the distance computation may be distorted and hence can be scaled.

The number of clusters to be found, along with the initial starting point values are specified as input parameters to the clustering algorithm. Given the initial starting values, the distance from each sample data point to each initial starting value is found using equation. Each data point is then placed in the cluster associated with the nearest starting point. New cluster centroids are calculated after all data points have been assigned to a cluster. Suppose that C_{im} represents the centroid of the mth feature of the ith cluster. Then,

$$C_{im} = \frac{\sum_{j=1}^{n_i} F^*_{i,jm}}{n_i}$$

where $F^*_{i,jm}$ is the mth feature value of the jth job assigned to the ith cluster and where ni is the number of data points in cluster i. The new centroid value is calculated for each feature in each cluster. These new cluster centroids are then treated as the new initial starting values and steps 3-4 of the algorithm are repeated. This continues until no data point changes clusters or until a maximum number of passes through the data set is performed.

3. Implementation of k-means in matlab

3.1 About Matlab

MathWorks is the leading developer of mathematical computing software for engineers and scientists. The product of Mathworks, matlab is a programming environment for algorithm development, data analysis, visualization, and numerical computation[7]. Using matlab, we can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran. We can use matlab in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. For a million engineers and scientists in industry and academia, matlab is the language of technical computing.

3.2 Data Sets

For testing of k-means in matlab, we used the well known UCI Machine Learning Repository. The UCI Machine Learning Repository is among other things, a collection of databases, which is widely used by the research community of Machine Learning, especially for the empirical algorithms analysis of this discipline[6]. The two data sets used for experimentation here are the iris plant dataset and the Haberman's Survival Data.

Iris plant Dataset: Total number of attributes is five of which four (Sepal Length, Sepal Width, Petal Length and Petal Width) are numeric and one the name of the class. The total number of instances are 150 (50 in each of the three classes). The three classes are Iris Setosa, Iris Versicolour, and Iris Virginica. One class is linearly separable from the other 2, the latter are not linearly separable from each other.

Haberman's Survival Data: The dataset contains cases from a study that was conducted at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. It contains 4 attributes of numerical type (Age of the patient, Year of operation, No. of positive axillary nodes detected and Survival Status). Total number of instances are 306.

3.3 Implementation Issues

The matlab function used for k-means clustering is *idx = kmeans(data,k),* which partitions the points in the n-by-p data matrix data into k clusters. This iterative partitioning minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. Rows of data correspond to points, columns correspond to variables. kmeans returns an n-by-1 vector idx containing the cluster indices of each point. By default, kmeans uses squared Euclidean distances. When data is a vector, kmeans treats it as an n-by-1 data matrix, regardless of its orientation.

```
>> opts = statset('Display','final');

[idx,ctrs] = kmeans(data,3,...
                    'Distance','city',...
                    'Replicates',5,...
                    'Options',opts);
```

Like many other types of numerical minimizations, the solution that kmeans reaches often depends on the starting points. It is possible for kmeans to reach a local minimum, where reassigning any one point to a new cluster would increase the total sum of point-to-centroid distances, but where a better solution does exist. However, you can use the optional 'replicates' parameter to overcome that problem.

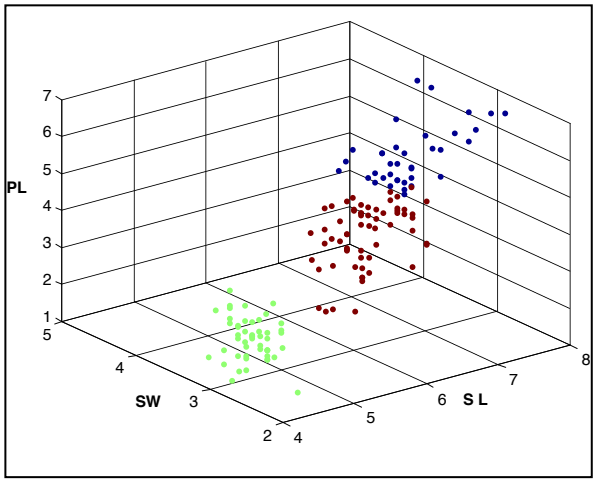
In iris dataset, for three clusters, specify five replicates, and use the 'display' parameter to print out the final sum of distances for each of the solutions.

5 iterations, total sum of distances = 159.3
5 iterations, total sum of distances = 159.3
5 iterations, total sum of distances = 159.3
5 iterations, total sum of distances = 159.3
4 iterations, total sum of distances = 159.3

[idx,ctrs] = kmeans(data,k) returns the k cluster centroid locations in the k-by-p matrix C.

In our experiment only three of the features for iris data set (sepal length, sepal width and petal length) are taken for convenience. To plot a scattered graph we use,

scatter3(data(:,1), data(:,2),data(:,3),10,idx,'filled')



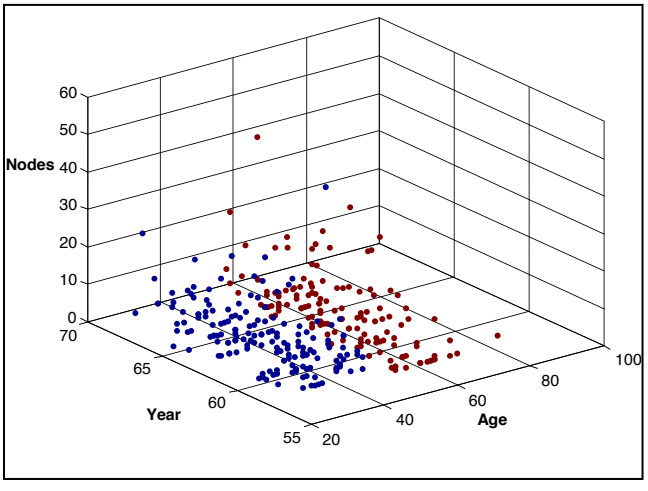
ĕI Ĕ

In haberman's dataset, for two clusters, specify five replicates, and use the 'display' parameter to print out the final sum of distances for each of the solutions.

```
[idx,ctrs] = kmeans(data,2,...
                    'Distance','city',...
                    'Replicates',5,...
                    'Options',opts);
```

4 iterations, total sum of distances = 3682
4 iterations, total sum of distances = 3682
5 iterations, total sum of distances = 3684
4 iterations, total sum of distances = 3684
6 iterations, total sum of distances = 4286

To plot a scattered graph for haberman's survival data year, age and number of axillary nodes are used for convenience.



ĕl Ĕ

4. Experimental Results

Tests are done on the number of iterations in k-means algorithm for reaching the minimum. The optional 'display' parameter is used to print information about each iteration. 'iter' is the number of iterations, 'phase' is the number of phases in the algorithm, 'num' is the number of points exchanged and 'sum' is the total sum of the distances.

For Iris data set with three clusters,
>> [u,v,sumd,D]= kmeans(data,3,'display','iter');

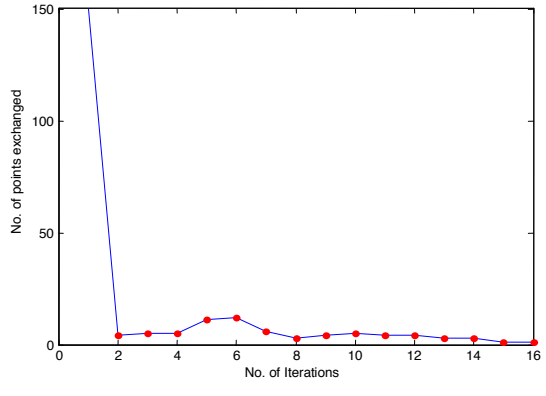
iter	*phase*	*num*	*sum*
1	1	150	144.909
2	1	4	143.181
3	1	5	140.999
4	1	5	131.651
5	1	11	107.511
6	1	12	89.6144
7	1	6	85.6414
8	1	3	84.4773
9	1	4	83.6055
10	1	5	82.3714
11	1	4	81.3672
12	1	4	80.3157
13	1	3	79.6817

14	1	3	79.1156
15	1	1	78.9451
16	2	0	78.9451

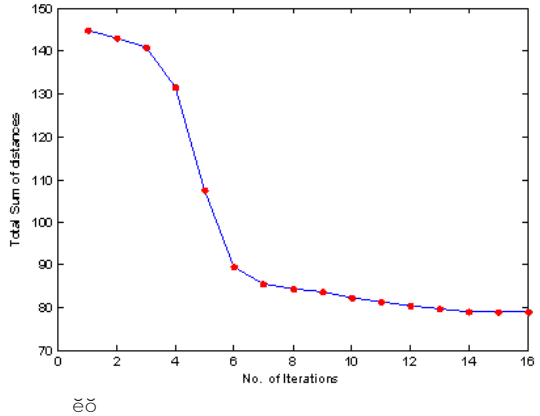
16 iterations, total sum of distances = 78.9408

Notice that the total sum of distances decreases at each iteration as kmeans reassigns points between clusters and recomputes cluster centroids. In this case, the second phase of the algorithm did not make any reassignments, indicating that the first phase reached a minimum. In some problems, the first phase might not reach a minimum, but the second phase always will.

>>plot(iter,num,'Marker','.','MarkerSize',15,'MarkerEdgeColor','r') plots the graph illustrating the number of iterations against the number of points exchanged.



>>plot(iter,sum,'Marker','.','MarkerSize',15,'MarkerEdgeColor','r') plots the graph illustrating the number of iterations against the total sum of distances.



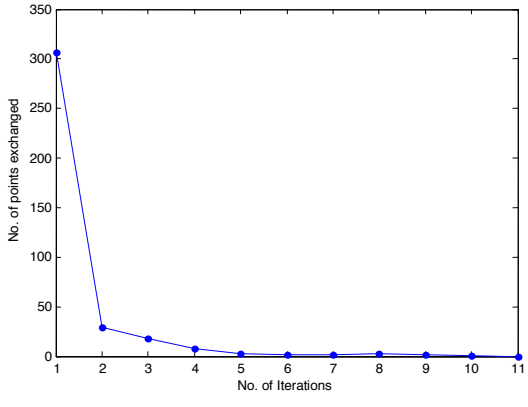
For Hyberman's Survival data set with two clusters,

>> [u,v,sumd,D]= kmeans(data,2,'display','iter');

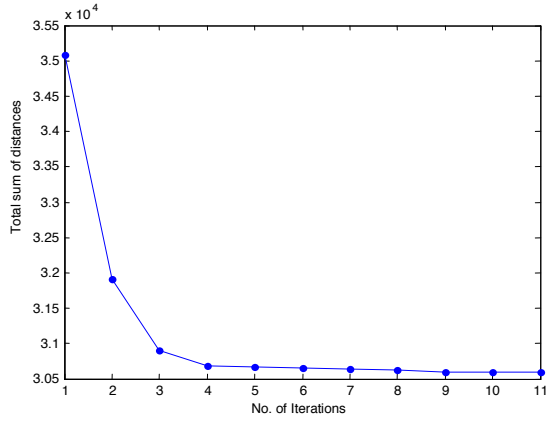
iter	*phase*	*num*	*sum*
1	1	306	35083.6
2	1	29	31905.2
3	1	18	30906.4
4	1	8	30686.4
5	1	3	30662.9
6	1	2	30651.4
7	1	2	30642.4
8	1	3	30618.1
9	1	2	30595.2
10	1	1	30592.8
11	2	0	30592.8

11 iterations, total sum of distances = 30592.8

>> plot(iter,num,'Marker','.','MarkerSize',15)



>> plot(iter,sum,'Marker','.','MarkerSize',15)



Test Results Show tha

- 90-95% of the points are located in the second iteration itself in both the cases.

- The algorithm reaches minimum in the first phase itself in both the cases.

5. Conclusion

Our work with matlab is simple and efficient for statistical analysis when compared to the work in java. Though k-means clusters datasets using heuristics, it is based on other clustering and location problems. Hence there is a need to experiment, analyse, improve and explore other algorithms. This can be done faster and effectively using matlab.

6. References

[1] Tapas Kanungo, David M Mount, *"An Efficient K-means Clustering Algorithm: Analysis and Implementation."* Pattern Analysis and Machine Intelligence, IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 24, No. 7 (July 2002).

[2] Krishna.K,M. Murty, *"Genetic K-means algorithm."* IEEE Trans. Syst., Man, Cybern. B., Cybern., vol. 29, no. 3, pp. 433 – 439, Jun. 1999.

[3] Kaufman L. and P. Rouseeuw, "*Finding Groups in Data: An Introduction to Cluster analysis.*" Wiley & Sons, 1990.

[4] Rui Xu, Donald Wunsch II, *"Survey of Clustering Algorithms."* IEE Transactions on Neural Networks, Vol.16, No.3, May 2005.

[5] G.Babu, M. Murthy, *" A near-optimal initial seed value selection in K-means algorithm using a genetic algorithm"*, Pattern Recognition, Vol.14, No.10, 1993.

[6] UCI. Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository Irvine, CA: University of California, School of Information and Computer Science. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[7] Matworks. http: //www.matworks.com

[8] C. Aggarwal and P. Yu, *"Redefining clustering for high-dimensional applications"* IEEE Trans. Knowl. Data Eng., vol.14, No. 2, Feb. 2002.

[9] P.S. Bradley, Usama M.Fayyad, *"Initial Points for k-Means Clustering."* Advances in Knowledge Discovery and Data Mining, MIT Press.

[10] Pang Tan,Vipin Kumar, Michael Steinbach, "*Introduction to Data Mining*" Pearson Publication.