The global k-means clustering algorithm

Saividhya Saibaba, Rukmani Ganapathy Seetharaman

Abstract— Normal k-means clustering algorithm takes the initial centroids randomly and performs a local optimistic search algorithm to find the centroids of the clusters. This algorithm depends on initial centroid values and its purity depends on it. An alternative approach is discussed in the paper [1] where a global optimistic search algorithm is proposed and each of the k centroids is determined incrementally by choosing N number of samples as initial position and performing N executions of the global k-means algorithm. To improve the runtime of the global k-means algorithm, another improvement is proposed which uses an error upper bound to stop the normal k-means algorithm and k-d trees to reduce the number of initial positions from N number of samples to number of bucket centers.

Index Terms—k-means, Clustering, global k-means, k-d trees, fast global k-means, purity, clustering error

1 Introduction

The clustering problem is one of the most common problems where clusters needs to be determined from a group of data samples. Data samples that are similar and homogeneous needs to be grouped together. A popular criterion to find optimized clusters is clustering error. Clustering error is the sum of the squared distance between data samples and the centroid to which it is assigned. One of the popular algorithm being used for clustering which minimized clustering error is the k-means algorithm. But, the performance of normal k-means algorithm depends on the initial position of the centroids chosen. Also, normal k-means performs a local search algorithm to find the centroids for the cluster based on initial centroids selected.

[1] proposes an incremental technique to choose the cluster centroids where each centroid is chosen in different iterations by choosing the number of samples as initial positions. In each iteration, normal k-means algorithm is used to find the cluster with minimized clustering error. To improve runtime of the algorithm [1] proposed another technique – fast global k-means algorithm with k-d trees which used bucket centers from k-d tree of the data samples as the initial position.

2 Methodology

2.1 Global k-means

Consider there are N data samples, X = {$x_1, x_2, x_3, \ldots x_n$} with d dimensions. The M clustering problem will be to group homogeneous data samples into M clusters which minimizes clustering error. The clustering error is determined using the formula in (1).

$$E(m_1,\ldots,m_M) = \sum_{i=1}^{N}\sum_{k=1}^{M} I(x_i \in C_k)\|x_i - m_k\|^2, \qquad (1)$$

Normal k-means algorithm performs an iterative approach to find the centroid of the clusters which minimizes the clustering error. It starts with randomly placed initial cluster centroids and finds a local optimum centroid of the cluster such that it minimizes the clustering error. But, the problem with normal k-means algorithm is it depends on the initial values of the centroid.

In [1], global k-means algorithm is used where the centroids are chosen incrementally and global search algorithm is done. The initial position of the centroids are not randomly selected. Instead, the data samples are used as initial position for the centroids and optimum centroids are chosen by performing normal k-means that minimizes clustering error. For example, if there are N data samples and the problem is to find M clusters then the M centroids are chosen incrementally as follows. For k=1, the mean of the data sample is selected as centroid and normal k-means algorithm is performed to find the optimal centroid for the problem M = 1. This becomes the first centroid in M clustering problem. For k = 2, we take the centroid from previous iteration and for the second centroid we consider all data samples and for each data sample we run normal k-means algorithm. Among these samples we select the centroids for which we got minimum clustering error. This will be the solution for M = 2 clustering problem. Similarly for M clustering problem, we take centroids we got for M-1 clustering problem and for the Mth centroid we choose from N data samples for which normal k means is run and the minimum clustering error is obtained.

2.2 Fast global k-means with k-d trees

The above algorithm is computationally expensive since we are running the normal k-means for each data sample and for M iterations. To improve the computational load of global k-means algorithm, two improvements were proposed in [1]. One improvement is to have an upper error bound. Instead of choosing the N number of data samples as centroid initial position and passing them to normal k-means, this approach takes the data sample for which the upper error bound is minimum. The Error bound is computed using formula in Fig. 1 [1]

$$E_n \leqslant E - b_n$$

Fig. 1. Error Bound

where E is the error from previous iteration and bn is the guaranteed reduction in error which is calculated using the formula in Fig. 2 [1]. The difference between the distance between the sample and centroid from previous iteration and the distance between the data sample and all the data samples is computed. Then the data sample which has maximum b_n or minimum E_n is chosen as the initial position for normal k-means.

$$b_n = \sum_{j=1}^{N} \max(d_{k-1}^{j} - \|x_n - x_j\|^2, 0),$$

$$i = \arg\max_n b_n,$$

Fig. 2. Formula for b_n

Another improvement proposed in [1] is to use k-d trees bucket centers as initial position instead of considering all the data samples. A k-d tree divides the data sample into multiple subsets. A node in the tree can either be a non-terminal node or leaf. All non-terminal nodes will have 2 children in k-d tree. At each non-terminal node the data space is divided into two regions by constructing a hyperplane and data on one side of the hyperplane is considered as left child subspace and data on other side will be in right child subspace. The k-d tree is constructed recursively and the final level of k-d tree above the leaves will have the bucket centers. A k-d tree is constructed using the data samples provided where it will categorizes and places the data samples into buckets and all data samples that are closer will fall under single bucket. These bucket centers are used as initial positions instead of the number of data samples. Hence, the computational load is reduced.

2.3 Interpretation

The global k-means and fast global k-means algorithms are deterministic cluster algorithms that globally cluster the data, without using random restarts. These algorithms solve the clustering problem incrementally by doing k -means on each individual cluster to get the best centroid for the cluster. This way, the optimal solution obtained for one cluster can be extended to find the optimal solution for the subsequent clusters. These methods are advantageous since they don't depend on any initial conditions to arrive at the optimal solution.

Since the global k-means algorithm computes the optimal solution for every cluster incrementally, the solution for M clusters would also consist of the solution for every cluster k that is less than M. This property can be leveraged while trying to cluster a dataset into its true cluster size without any additional computations. According to [1], this algorithm works based on the assumption that having found the optimal centres for the (k-1) clusters, the kth center can be discovered using local search.

Although, the idea of global k-means overcomes the problem of random restarts in the k-means clustering problem, it suffers from huge runtime complexity. [1] proposes fast global k-means and fast global k-means with kd-trees as optimizations. In the fast global k-means method, the converging criteria is modified to reduce the time complexity. An upper bound is computed for the clustering error and this quantity is used as the threshold for the number of iterations. [1] states that calculating the guaranteed reduction in error b_n as shown in Fig. 2 will help compute the upper bound of the error E - b_n that is got if the algorithm is run until convergence. Furthermore, storing the intermediate results like the pairwise distance between data points and clusters would help speed up the execution further.

The next suggested optimization technique of using kd-trees performs recursive PCA on the data to partition the data space into buckets. Since kd-trees partition the data space, using the bucket centres as potential centroids would be a reasonable approximation and since the number of bucket centres is much lesser than the number of data points, this algorithm runs faster than the global k-means algorithm.

The **EM algorithm** and k-means algorithm are highly inter-related in terms of trying to cluster the data. [3] lists the two as different methods to perform the clustering task for a given data point. Accordingly, the E and M steps for the k-means clustering involve the following -

```
Initialization: Random k centroids
Loop {
    Expectation step : Computing clustering error and cluster assignments
    Maximization step: Minimize delta between clustering error
} until delta in clustering error doesn't change much for subsequent iterations or
cluster assignments doesn't change
```

Fig. 3. EM algorithm for k-means clustering

After the random initialization of the k centroids, the k-means computation algorithm assigns the data points to the cluster centered at the closest centroid from the list of centroids and computes the clustering error. This can be considered the **E step** of the algorithm which involves with defining a rule for the repeated computation. The cluster assignment continues till the difference in the cluster SSE between iterations is minimized. This is equivalent to the **M step** of the EM algorithm which is defines the rule for selecting the best value from the values calculated in the E step. Accordingly, the K-means algorithm can be compared to the EM algorithm and implemented such that the cluster assignment results in minimum clustering error.

3 Potential areas of use

The study conducted in [1] would be useful in every scenario where clustering needs to be done. While performing unsupervised learning, k-means is one of the most common methods used. However, since the normal k-means algorithm's performance depends on the initial cluster chosen, the results would vary greatly for every run of the algorithm. [4] proposes the k-means algorithm with multiple restarts to overcome the initialization problem. [5] talks about different methods by which a global optimum can be found given a huge dataset of points that need to be clustered. [1] also states that many other stochastic global optimization implementations like simulated annealing, genetic algorithms have not gained wide acceptance

4 Implementation

We have used MATLAB to implement all three algorithms. The data sets which we have considered are MNIST data set and two artificially generated data set. We take 5000 samples from MNIST data set (by selecting 500 data samples from each cluster) and PCA is applied on the data set to reduce the number of dimensions. To select the 5000 samples from MNIST data set along with the labels we first concatenate data sample with the corresponding label as first column and form a matrix. From this matrix, we randomly choose 500 samples from each label and we use the first column as label vector and the remaining columns as the data sample. This is done to improve the runtime of the global k-means algorithm. For artificially generated data - we consider 3 Gaussian components where in one data set we have Gaussian components whose means are close to each other and the other data set has a well separated components. The Gaussian distribution is computed using gmdistribution method of MATLAB where the means and covariance of each components is given as input. A 2 dimensional Gaussian mixture is generated. For the well separated artificially generated data set we have included components with mean [-5 0;1 6;4 -5] and variance [1 0;0 1],[2 0;0 .5],[1 0;0 1]. For the overlapping artificially generated data set we have included components with mean [1 -3;-3 -5;3 -6] and variance [1 0;0 1],[2 0;0 .5],[1 0;0 1].

4.1 Normal k-means

The initial centroids are taken randomly using randperm function from MATLAB. The euclidean distance between each data sample and the centroids are calculated using pdist2 function. Each data sample is assigned to the centroid to which the distance is minimum by using min function in MATLAB which returns the minimum distance and the corresponding centroid value.. Clustering error is computed using the formula in (1). Clustering error is calculated by computing the squared distance between the data sample and the centroid to which the distance is minimum. The above method is run for multiple iterations until the stop condition is satisfied. We have taken the stop condition as exact match where the cluster assignment doesn't change between iterations. The centroids for the next iteration is computed by taking the mean of the data samples that has been assigned for each cluster. To implement this we have used splitapply function of MATLAB which splits the data into groups and applies the specified function and returns the result.

4.2 Global k-means

In Global k-means algorithm, for k=1 the mean of the data sample is selected as centroid. For k = 2, the centroid from previous iteration is taken and the second centroid is taken by choosing each of the data sample as initial centroid and normal k-means algorithm is run for each data sample. From the above data samples, the centroid for which minimum clustering error was obtained is considered as the solution for k=2. Similarly the above method is done for M clusters by fixing M-1 centroids and taking the Mth initial centroid as data sample and compute optimal centroids by running normal k-means.

4.3 Fast global k-means with k-d trees

In fast global k-means algorithm, initially a k-d tree is built from the data samples to find the bucket centers. K-d tree is constructed recursively by first choosing a principal component for the data set. The principal component of the data set is computed using pca function of MATLAB and we choose the principal component for which the eigen value is maximum. Then, the data samples are divided into two groups as left subspace and right subspace. The difference between the projection of each sample on the principal component and the median of the projections of the data samples are calculated and all the samples for which this difference is less than 0 are considered as samples in left subspace and all samples for which the difference is greater than or equal to 0 are considered as samples from right subspace. The same logic is applied recursively for left and right subspaces until the number of samples which can fall under a bucket (bucket size) is satisfied. The distance between the bucket centers and samples and the distance between samples are computed initially using pdist2 function in MATLAB and stored in memory so that we need not compute it each time in the iteration. Once the bucket centers are created, these are used instead of data samples to compute the new centroids. Similar to global k-means for k=1 the mean of the data sample is taken as centroid and passed to normal k-means algorithm. For k = 2, the centroid from previous iteration and the bucket centers are considered as initial centroid positions which is sent to normal k-means algorithm. To further reduce the computational overload the error bound is calculated for each bucket center and the bucket center for which there is minimum error bound is taken and sent to normal k-means algorithm instead of sending it for all bucket centers. The error bound is calculated by using formula in Fig 1 and 2. The difference between the distance of data sample and the centroid of previous iteration and the distance between bucket center and all data samples is calculated. The sum of the above difference is computed for each data sample and the bucket center for which the sum is maximum is considered as the initial centroid value.

5 Problems encountered and assumptions

The global k-means algorithm on the original MNIST data set was taking long time. So, to reduce the computational overload we performed PCA on the original data set to reduce the number of dimensions and also we got 500 samples from each cluster. So that our data set has 5000 samples of MNIST data. Also, while running the algorithms on original value we got huge clustering error. Hence, we normalized the data by dividing the values by 255. Another assumption we made was when constructing k-d tree, all the samples whose difference with the median is less than 0 are considered as one subset and the remaining samples are considered as another subset as discussed in paper [2].

6 Metrics Calculated

6.1 Clustering error

This is a cluster quality metric that is calculated as the sum of square error (SSE) for a given dataset and cluster centres. Once the data points are assigned to a cluster, the square of the distance between each data point and its cluster centre is calculated and summed for all data points. If the cluster assignment was correct, the SSE would be less.

6.2 Purity

This is a cluster quality metric that measure the level of homogeneity of data points in a cluster. This is calculated by considering the true labels of the data samples in a cluster. To get this measure, count of the most frequent class label in each cluster is added and the sum is divided by the total number of data points in the dataset. Ideally we would want the clusters to be highly homogenous such that the true labels of all points within a cluster are the same. Since, data samples sharing that are similar to each other tend to be located close to each other, we would expect our clusters to capture this and have very high intra cluster similarity and very low inter cluster similarity. If this is the case, the purity measure would be high, indicating highly homogenous clusters.

7 Results

7.1 Runtime

We ran all the three algorithms - normal k-means, global k-means and fast global k-means with MNIST data set. We selected 500 samples from each label and did PCA to reduce the number of dimensions. We conducted this experiment for different values of k = 3 to 15. The runtime for all the three algorithms for MNIST data set is plotted in Fig. 1.

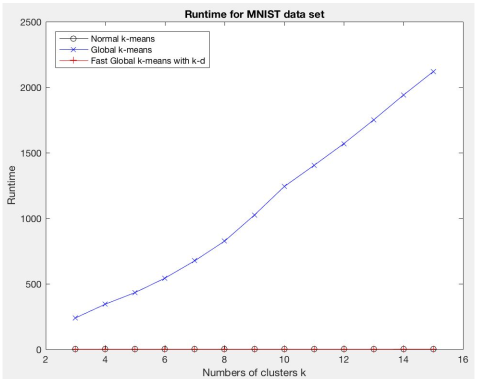
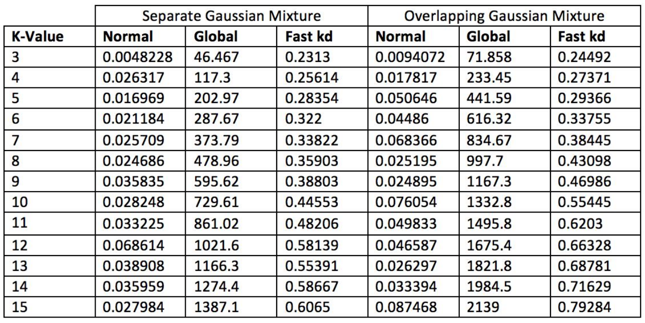


Fig. 1. Runtime for MNIST data set for all three algorithms

It can be seen that the runtime for both global and fast global k-means algorithm increases as the k value increases. The actual runtime values for both Gaussian mixture datasets are shown in Fig 4. The runtime for global increases drastically as compared to the other two algorithms since the global k-means runs the normal k-means algorithm for k X N times where N is the number of samples. Also, it can be seen that the fast global k-means takes a little more time than the normal k-means since the fast global k-means runs the normal k-means for k X B times where B is the number of bucket centers considered for initial positions.



	Separate Gaussian Mixture			Overlapping Gaussian Mixture		
K-Value	**Normal**	**Global**	**Fast kd**	**Normal**	**Global**	**Fast kd**
3	0.0048228	46.467	0.2313	0.0094072	71.858	0.24492
4	0.026317	117.3	0.25614	0.017817	233.45	0.27371
5	0.016969	202.97	0.28354	0.050646	441.59	0.29366
6	0.021184	287.67	0.322	0.04486	616.32	0.33755
7	0.025709	373.79	0.33822	0.068366	834.67	0.38445
8	0.024686	478.96	0.35903	0.025195	997.7	0.43098
9	0.035835	595.62	0.38803	0.024895	1167.3	0.46986
10	0.028248	729.61	0.44553	0.076054	1332.8	0.55445
11	0.033225	861.02	0.48206	0.049833	1495.8	0.6203
12	0.068614	1021.6	0.58139	0.046587	1675.4	0.66328
13	0.038908	1166.3	0.55391	0.026297	1821.8	0.68781
14	0.035959	1274.4	0.58667	0.033394	1984.5	0.71629
15	0.027984	1387.1	0.6065	0.087468	2139	0.79284

Fig. 2. Consolidated runtime for Separate and Overlapping Gaussian data set

7.2 Performance Vs Number of buckets

We ran the fast global k-means algorithm with k-d tree for all the 3 datasets and increased the number of buckets for the same datasets mentioned in Section 4. We ran this experiment for k=15 and the results have been plotted in Fig. 5 and 6. It can be seen that the clustering error decreases as the number of buckets is increased. Also it can be seen that the SSE remains constant after certain number of buckets. The same behavior was observed for all the 3 datasets - MNIST, Gaussian overlapping and well-separated mixtures.

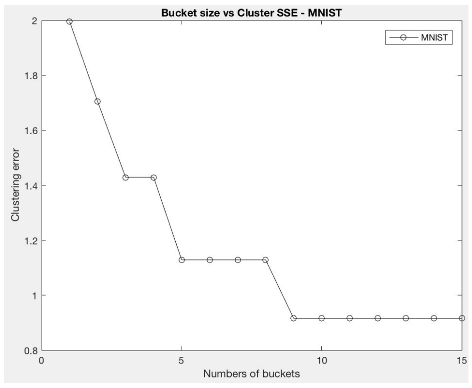


Fig. 3. Bucket size Vs Clustering Error for MNIST data set

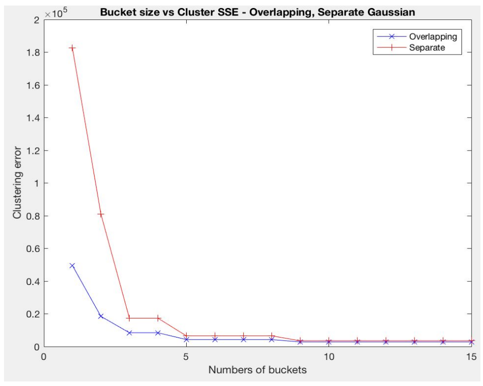


Fig. 4. Bucket size Vs Clustering Error for Gaussian data set

7.3 Clustering Error

We ran all the three algorithms - normal k-means, global k-means and fast global k-means with the MNIST dataset, Well separated and Overlapping Gaussian data mixtures with 3 components respectively. We varied the k values between 3 to 15. For the normal k-means algorithm, we ran 100 iterations for each k value and computed the average SSE for each k value. Fig 7 shows the Clustering Error vs K value for the MNIST dataset. It can be observed that the Clustering error for all 3 methods keeps decreasing as the number of clusters increases. This is because, as the number of clusters increases, more points get clustered into their respective clusters, thus reducing the SSE. However, there is not much of a difference in between the 3 algorithms for different k values. These could be because all 3 algorithms result in very similar cluster centres which is causing the clustering error to be very similar.

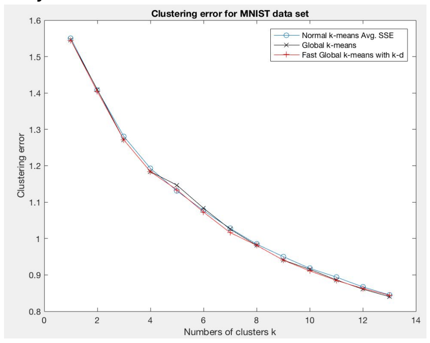


Fig. 5. Clustering Error for MNIST dataset

Fig 8 and 9 show the Clustering error vs number of clusters plots for the Well Separated and Overlapping Gaussian mixtures respectively. Here also it can be observed that the number of clusters increases, the clustering error decreases. Here, global and fast global seem to perform very similarly while normal k-means seems to perform marginally worse than the other two.

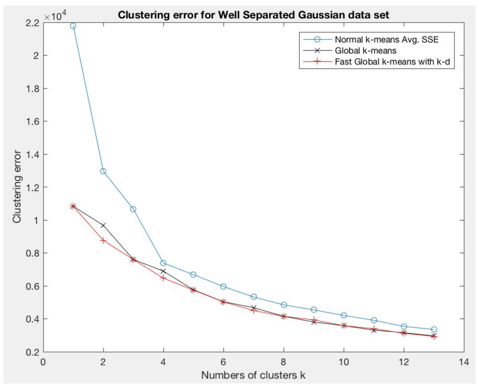


Fig. 6. Clustering Error for Well Separated Gaussian dataset

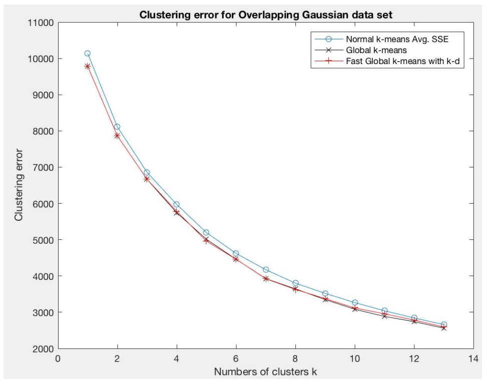


Fig. 7. Clustering Error for Overlapping Gaussian dataset

7.4 Purity

We computed the purity of the clusters that were got by running the normal k-means, global k-means and fast global k-means with KD-Trees on the MNIST dataset since this is the only dataset with true labels for the data points. Since normal k-means was run for 100 iterations for each k-value, the purity measured is the average of the purity values got during the 100 iterations for each k value. The plot for the purity values obtained for each k value for all the 3 algorithms is shown in Fig 10. It can be observed that the purity value for the clusters increases as the k value increases. This can be attributed to the fact that as the number of clusters increases, the number of points that are correctly clustered also increases. That is, similar points gets clustered together while dissimilar points are not clustered together. When this occurs, the homogeneity of the clusters also increases which is what the purity measures.

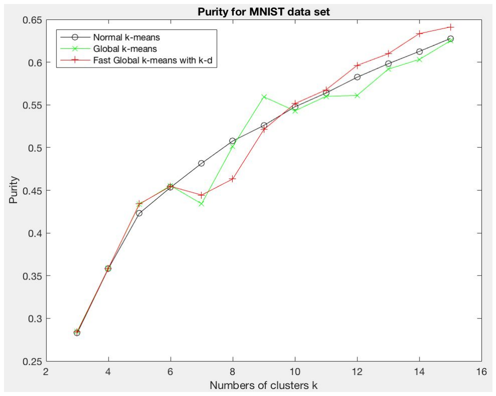


Fig. 8. Clustering Error vs Purity for MNIST data set

7.5 Cluster Assignments

The cluster assignments for a Well Separated Gaussian mixture dataset is shown for normal k-means, global k-means and fast global k-means with KD Trees in Fig 13, 14 and 15 respectively. These were run for k = 15. It can be observed that all the 15 cluster centers get positioned such that they can cluster the data completely.

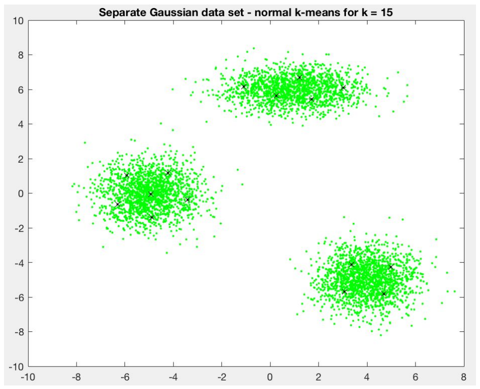


Fig. 9. Cluster Assignment for Normal k-means

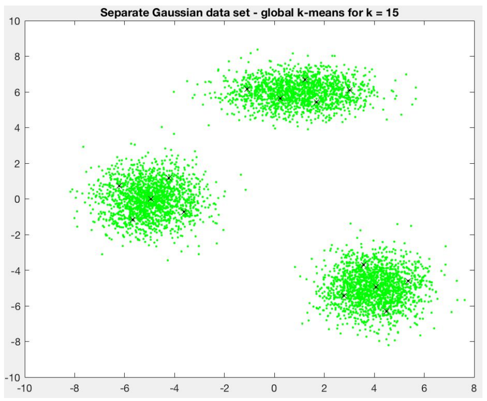


Fig. 10. Cluster Assignment for Global k-means

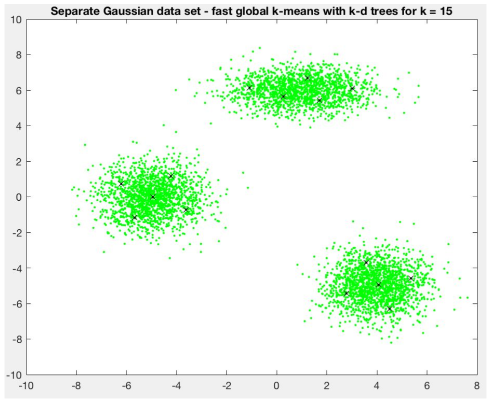


Fig. 11. Cluster Assignment for fast global k-means with KD-Trees

8 Discussion and Conclusion

Thus we understood and analyzed the concept of global k-means algorithm. We were able to observe the points that were mentioned in [1] regarding the performance of the normal k-means algorithm and why it is better to go for global optimum as opposed to local optimum. The graphs plotted for the SSE, purity, runtime all seem to be in line with the observations made in [1].

9 Future Scope

The methods proposed in [1] can be further generalized to obtain a generic algorithm for all unsupervised learning algorithms that depend on an initial state. The idea of considering all data samples as potential clusters, which was further optimized to using kd trees, which would be used as the potential centroids can be further extended to work for all kinds of data samples. Robust cluster quality metrics that measure the clustering power of the algorithm need to be defined such that the measure of correctness will be able to indicate how well the algorithm is performing on a given dataset.

Acknowledgment

We would like to thank Prof. Baoxin Li for giving us the opportunity to work on this project. It was a great learning experience to implement this project and interpret the results. We would also like to thank Vijetha Gattupalli, the TA, for clarifying our doubts in timely manner and for guiding us throughout the implementation of the project.

We also acknowledge that we understand the discussion and the requirements of this project have been satisfied without violating any academic integrity policies.

References

[1] [1] The global k-means clustering algorithm Aristidis Likas, Nikos Vlassis, Jakob J. Verbeek Pattern Recognition, 2003

[2] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK,

2008, pp. 1-8

[3] “Clustering with K-Means and EM: how are they related?” *Machine learning - Clustering with K-Means and EM: how are they related?-CrossValidated*, stats.stackexchange.com/questions/76866/clustering-with-k-means-and-em-how-are-they-related. Web. 01 Dec. 2017.

[4] M.N. Murty, A.K. Jain, P.J. Flynn, Data clustering: a review,ACM Comput. Surv. 31 (3) (1999) 264–323.

[5] Torn, A. A. "Clustering methods in global optimization." Stochastic Control. 1987. 247-252.