Optimizing k-means for Scalability

Akansha Agrawal
M.Tech CSE Scholar
Rajasthan College of Engineering for Women, Jaipur

Shreya Sharma
M.Tech, CSE
Amity University, Noida

ABSTRACT
Proposed decades ago, k-means is still the most popular algorithm for clustering. Despite the drawbacks of k-means, its advantages make it most attractive. Several researches have been conducted to alleviate the problems of k-means. We suggest here some simple modifications to optimize k-means for scalability without much sacrifice in the precision. Current shift in emphasis of data mining towards Big Data requires fast algorithms that can scale well. We propose an idea how time-tested techniques can be adapted to changing needs. The implementation results demonstrate the impact simple modifications can bring

General Terms
Clstering, k-means

Keywords
Data mining, Big Data, k-means

1. INTRODUCTION
Big data refers to the techniques that are employed to understand, analyze and utilize the knowledge hidden in massive amount of data that resides in Cloud. Cloud computing itself has an array of associated technologies specifically suited to the requirements. Yet, the impact of cloud computing is too vast to be grasped completely at present. The amount and velocity of data that is being generated due to Cloud computing has led to many new challenges of data mining. A straightforward requirement from any data mining algorithm to be designed for Big Data could be stated as: scalable, handling mixed data, handling missing data, handling streaming data, fast, secure, accurate. To ensure scalability, an algorithm might become complex itself, thus increasing runtime. We observe that in order to design a scalable yet fast algorithm for data analysis, the conventional algorithms should be changed slightly so as not to increase much programming effort and yet achieve desirable performances. Also, in context of many cloud applications involving data analysis speed of computation matters much more than the accuracy of results. So, we emphasize on reducing runtime of the algorithm by not much sacrificing the precision of clustering results.

The choice of k-means algorithm is due to its simplicity. Five decades ago K-means was proposed in [1]; it is still very popular [2] and one of the top 10 algorithms of clustering and data mining [3]. The algorithm was found to be sensitive to initial selection of cluster representatives; and research for better initialization varies from random selection to supervised learning based selection. Lozano et al[4] first attempted to compare the different versions of k-means random partition, Forgy[5], Kaufman[6] and MacQueen[1]; a more recent is by Erisoglu et al [7]. There are some research works like [8] which compare and combine the evolutionary approaches with k-means. Very recent works [9,10,11,12,13] to improve k-means emphasize the potential of this algorithm; hence we have selected it for our work. Some improve upon initialization methods, while other analyze using different distance metric to calculate distances between points.

2. BACKGROUND
The k-means algorithm used for clustering as proposed in [1,5] are all similar in the basic concept that any data point should be assigned to a cluster based on its distance from the centroid of the cluster. The centroid of a cluster is an artificial data point obtained by taking mean values of all data points belonging to the cluster. Hence, each time a data point is added to a cluster, the centroid of that cluster gets updated. Thus, the algorithm progresses iteratively; at each iteration a data point is considered and its cluster is decided based on the minimum distance from the cluster representatives and then updating the value of cluster representatives. At each iteration through every data point, the points may or may not change their cluster. The algorithm stops when no data point changes its cluster. The initial values of cluster representatives have to be picked randomly.

Factors that affect runtime and performance of k-means are: 1) initial seeds or cluster representatives, 2) number of instances, 3) number of clusters, 4) number of dimensions. Moreover, the small operations like measuring distance between a data point and cluster representative also affect the overall runtime of the algorithm.

3. PROPOSED SCHEME
We modify k-means in three aspects:1)Limiting criteria, 2) Selection of initial seeds and 3) Distance metric used to measure the distance between cluster representatives and data-points.

Limiting criteria

The popular versions of k-means run till there are no changes of clusters. This convergence is non-deterministic hence a potential source of time consumption. When adapting k-means for scalability, we need to make it faster. Here, we suggest that algorithm should stop when number of data points changing their cluster drops below a certain level. Let this level be termed as "threshold", denoted by δ. Another limiting criterion is more straightforward, that is fixing the number of iterations beforehand, to a number "max_iteration", denoted by μ. Deciding the values of these parameters is now an issue. We suggest that these should be decided based on the distribution of dataset so as to reflect the nature of dataset.

We decide value of threshold as follows:

Threshold = Standard-deviation of extraneous values, where any value which does not fall within the standard deviation range of mean is termed as extraneous.

The algorithm to decide threshold is:

Step a) Compute mean vi and standard deviation di of all dimensions

Step b) For every data value xl in dimension l, if $|x_l - v_l| > d_l$, the value x_l is extraneous

Step c) Count the number of extraneous values in each dimension and compute its standard deviation. Output this value.

The value of max_iteration should be decided according to the runtime required. The runtime of traditional k-means is O(nkt), where t is number of iterations. Keeping t=n will give a quadratic running time which is acceptable. To eliminate the factor of k, we can pick t=n/k. More reduction can be obtained by picking t=n/k*k. Thus, a tradeoff is to be established between runtime and cluster quality.

Initializing cluster seeds

Generally initial cluster representatives are selected at random in k-means algorithm. We observe that the time taken by the algorithm to converge increases if the initial cluster centroids are very far from the actual centroids. If the initial centroids are selected as close as possible to ideal cluster centroids then algorithm will take less time to converge. Instead of picking a real data-point as a representative, we pick certain values and form into a tuple to have an artificial data-point. For any dimension, its entire range of values can be divided into k sub-ranges, and the middle value of each sub-range is picked.

The algorithm to decide initial cluster representatives is:

Deterministic Mean Representatives

Step a) Compute width of sub-range of all m dimensions as $width_j = \frac{(max_j - min_j)}{k}, 1 \leq j \leq m$

Step b) Create k centroids as m-tuples of values $min_j + \frac{i * width_j}{2}, 1 \leq i \leq k, 1 \leq j \leq m$

Step c) Output the centroids

Distance metric

The distance between centroids and data-points is crucial operation, since it is performed many times during entire clustering process. Conventional k-means algorithm uses Euclidean distance which involves multiplication and calculation of square root. This is an expensive distance metric. We pick a less expensive distance metric, namely Manhattan distance that involves only addition and subtraction. Though this may impact the quality of cluster, yet if the compromise in quality is not much more than the gain in runtime, we can opt Manhattan distance safely. The formula for computing Manhattan distance between any two m-tuples is

$$Manhattan\ distance(p_1, p_2) = \sum_{i=1}^{m} |x_{1i} - x_{2i}|$$

Now, we present the variants of k-means that we use for clustering. It is much similar to the traditional k-means, except the three changes that we have proposed above.

4. SCALABLE k-MEANS WITH FIXED ITERATIONS

Step 1 – Select initial centroids as per Deterministic Means Representatives initialization technique

Step 2 – Compute Max_iterations $\mu = \frac{n}{k*k}$

Step 3 – For each data-point, assign a cluster according to minimum distance from the centroids. The distance used is Manhattan.

Step 4 – update values of centroids as mean of the data-points in the respective cluster.

Step 5 – Repeat steps 3 and 4 until Max_iterations is reached

5. SCALABLE k-means WITH Fast Convergence

Step 1 – Select initial centroids as per Deterministic Means Representatives initialization technique

Step 2 – Compute Max_iterations $\mu = \frac{n}{k*k}$, and Threshold δ according to algorithm for deciding threshold

Step 3 – For each data-point, assign a cluster according to minimum distance from the centroids. The distance used is Manhattan. Simultaneously record the number of data-points that change their cluster.

Step 4 – Update values of centroids as mean of the data-points in the respective cluster.

Step 5 – Repeat steps 3 and 4 until Max_iterations is reached or Number of changes fall below than δ.

6. IMPLEMENTATION RESULTS

The proposed variations in k-means have been implemented using MATLAB® to study the impact of various parameters involved. A comparison with original k-means, provided in MATLAB as built-in function has been drawn. Experiments have been performed over popular datasets and some synthetic datasets. The results over famous Iris dataset which has four dimensions and 150 instances belonging to 3 different classes are summarized in Tables 1 and 2. The last two columns compare the proposed algorithms with traditional k-means in terms of the precision-runtime trade off.

Table 1 Parameter values and implementation results for proposed algorithms and standard k-means over Iris dataset

	Max_iteration	**Threshold**	**Actual Iterations**	**Runtime**	**Precision**	**% loss in precision**	**%gain in runtime**
Fixed Iteration k-means			17	0.039 sec	0.8867	0	90.66
Fast convergence k-means	17	13	2	0.0053 sec	0.77	13.16	98.73

Standard k-means				0.418 sec	0.8867		

Another popular dataset is Ruspini[15]of two-dimensional points that could be grouped into two or four clusters, for which standard k-means gives maximum precision (value 1) is used to evaluate performance of proposed algorithms. The results for 2 cluster grouping are summarized in Table 2.

Table 2 Parameter values and implementation results for proposed algorithms and standard k-means over Ruspini dataset

	Max_iteration	**Threshold**	**Actual Iterations**	**Runtime**	**Precision**	**% loss in precision**	**%gain in runtime**
Fixed Iteration k-means			19	0.0158 sec	1.0	0	88.63
Fast convergence k-means	19	7	2	0.0021 sec	0.95	5	98.48
Standard k-means				0.139 sec	1.0		

Synthetic datasets, with random data values, of different dimensions and size were constructed and runtime performance of the algorithms for scalability was tested. The scale-up can be in number of instances, number of dimensions or number of clusters.. Table 3 lists the runtime for different settings

Size of Dataset			**Run-time (in sec)**		
n	**m**	**k**	**Fixed Iteration k-means**	**Fast convergence k-means**	**Standard k-means**
1000	2	2	0.24	0.0023	0.052
1000	5	2	0.27	0.0023	0.059
1000	10	2	0.332	0.003	0.061
1000	2	4	0.35	0.0054	0.0077
1000	5	4	0.353	0.007	0.015
1000	10	4	0.36	0.0072	0.029
5000	2	2	0.87	0.039	0.074
5000	5	2	0.96	0.058	0.105
5000	10	2	1.09	0.105	0.174
5000	2	4	1.26	0.034	0.019
5000	5	4	1.91	0.0349	0.175
5000	10	4	2.21	0.038	0.417
10000	2	2	3.4	0.113	0.051
10000	5	2	3.44	0.157	0.197
10000	10	2	3.558	0.182	0.311
10000	2	4	5.76	0.071	0.054
10000	5	4	5.91	0.073	0.203

10000	10	4	6.03	0.0735	0.569
50000	2	2	18.43	0.34	0.72
50000	5	2	19.05	0.36	1.08
50000	10	2	19.89	0.397	2.56
50000	2	4	21.72	0.445	0.843
50000	5	4	23.62	0.447	1.54
50000	10	4	27.88	0.544	2.98

It can be observed that the growth in runtime of the proposed algorithms is not much as compared to the traditional k-means, hence they are more scalable than k-means. Also, the proposed fast convergence k-means is much faster. The growth can be clearly understood by graphical representation given in Figure 1 for small size datasets. Figure 2 for medium size datasets. Figure 3 shows the runtime cost of propose algorithms vs. the Standard K-means algorithm for large size datasets.

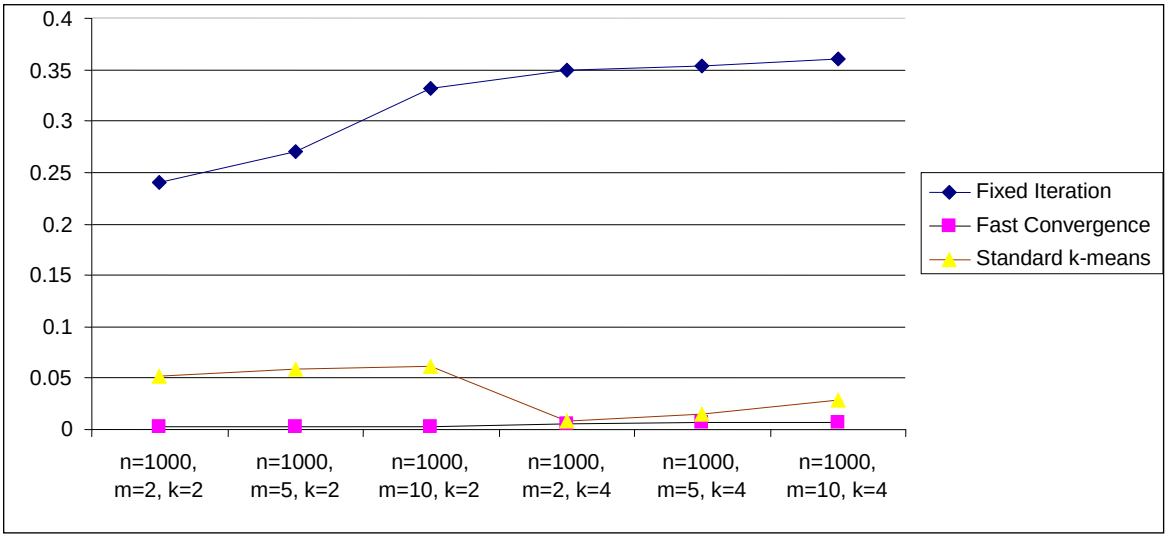

Figure 1 Growth of runtime of proposed algorithms for small dataset

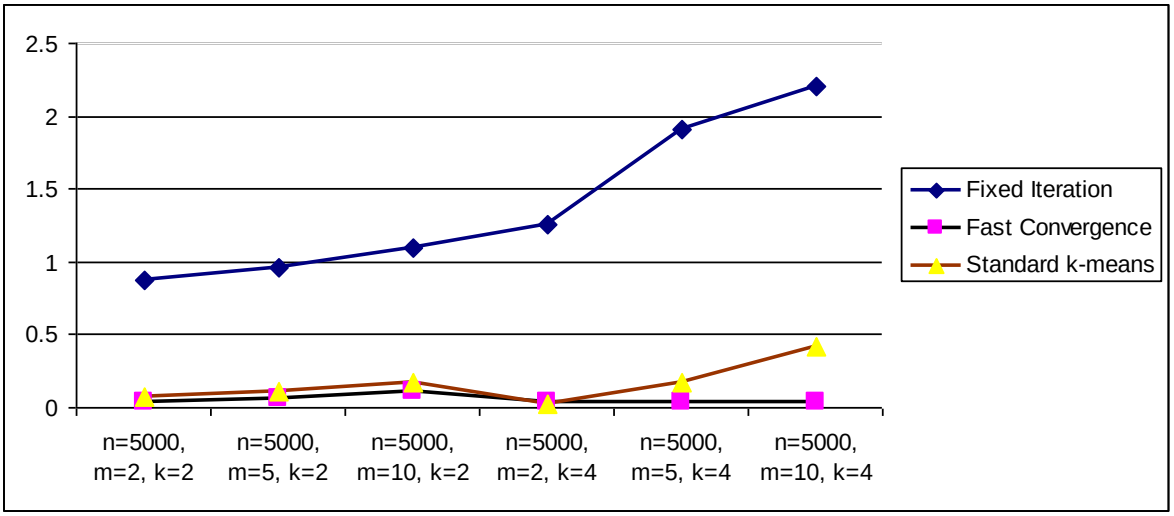

Figure 2 Growth of runtime of proposed algorithms for medium size dataset

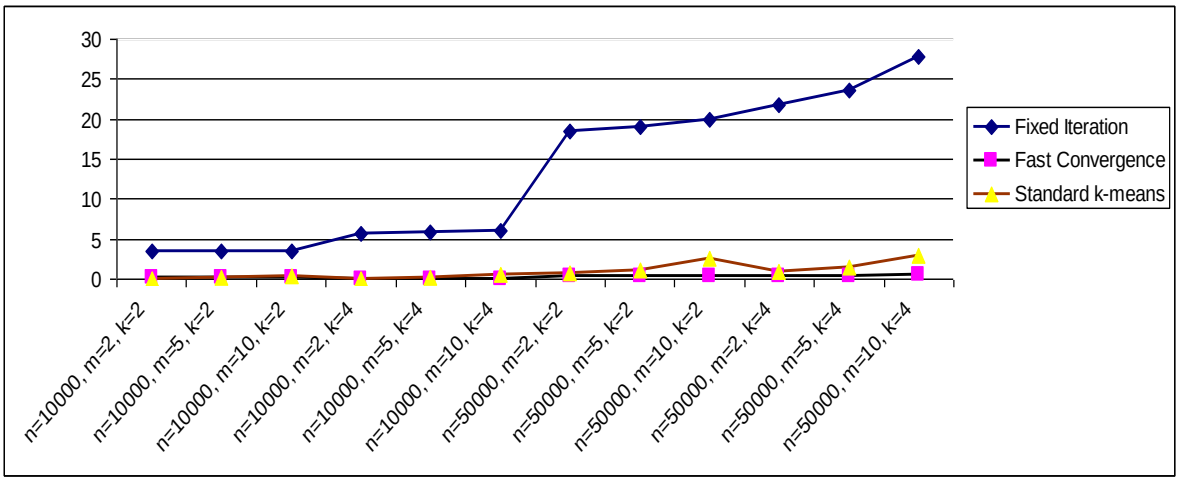

Figure 3 Growth of runtime of proposed algorithms for large size dataset

7. CONCLUSION

A simple idea how can traditional clustering algorithms be made scalable has been proposed. Experiments have been conducted to test the scalability performance of proposal on various synthetic datasets. Results over popular datasets and their comparison with standard k-means show that the trade-off of precision for speed is much in favour of the proposed algorithms. That is, the proposed fast convergence variant of k-means has a very slow growth of runtime with increasing number of clusters, even in large datasets.

As further research, the proposed idea can be combined with distance metrics for categorical data to check its behavior on heterogeneous datasets. Also, the proposed idea of fixing the number of iterations or number of changes can be used in other traditional incremental clustering algorithms. The performance of proposed variants for very large datasets, or very large number of clusters is also open for exploration.

8. REFERENCES

[1] J. MacQueen. Some methods for classification and analysis of multivariate observations. In Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967.

[2] A. K. Jain. Data clustering: 50 years beyond k-means. Pattern Recognition Letters, 31:651-666, 2010.

[3] X. Wu et al. Top 10 algorithms in data mining. Knowledge and Information Systems, 14(1):1-37, 2008.

[4] Lozano, J.A., Pena, J.M., Larranaga, P., 1999. An empirical comparison of four initialization methods for the k-means algorithm. Pattern Recognition Letters 20, 1027–1040.

[5] E.W. Forgy (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". Biometrics 21: 768–769.

[6] Kaufman, L., Rousseeuw, P. J., 1990. Finding Groups in Data. An Introduction to Cluster Analysis. Wiley, Canada.

[7] Erisoglu, M., Calis, N., Sakallioglu, S., 2011. A new algorithm for initial cluster centers in k-means algorithm. Pattern Recognition Letters 32, 1701–1705.

[8] C Liu, T Hu, Y Ge and H Xiong, "Which Distance Metric is Right: An Evolutionary K-Means View", Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012.

[9] Igor Melnykov, Volodymyr Melnykov. "On K-means algorithm with the use of Mahalanobis distances", Statistics and Probability Letters 84 (2014) 88–95. http://dx.doi.org/10.1016/j.spl.2013.09.026

[10] GrigoriosTzortzis, AristidisLikas. "The MinMax k-Means clustering algorithm", Pattern Recognition 47(2014)2505–2516. http://dx.doi.org/10.1016/j.patcog.2014.01.015

[11] Sadhana Tiwari and Tanu Solanki, "An Optimized Approach for k-means Clustering", International Journal of Computer Applications (0975 – 8887) 9th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine-2013)

[12] A Singh, A Yadav and A Rana, "K-means with Three different Distance Metrics", International Journal of Computer Applications (0975 – 8887) Volume 67– No.10, April 2013.

[13] M Ramakrishnan and DT Jayaraj, "Modified K-Means Algorithm for Effective Clustering of Categorical Data Sets", International Journal of Computer Applications (0975 – 8887) Volume 89 – No.7, March 2014.

[14] E. H. Ruspini (1970) Numerical methods for fuzzy clustering. Inform. Sci. 2, 319–350.