

# Evaluation of an Optimized K-Means Algorithm Based on Real Data

Cosmin M. Poteraş Faculty of Automation, Computers and Electronics University of Craiova, Romania Email: cpoteras@software.ucv.ro Mihai L. Mocanu Faculty of Automation, Computers and Electronics University of Craiova, Romania Email: mmocanu@software.ucv.ro

Abstract—In a previous paper [1] we introduced an optimized version of the K-Means Algorithm. Unlike the standard version of the K-Means algorithm that iteratively traverses the entire data set in order to decide to which cluster the data items belong, the proposed optimization relies on the observation that after performing only a few iterations the centroids get very close to their final position causing only a few of the data items to switch their cluster. Therefore, after a small number of iterations, most of the processing time is wasted on checking items that have reached their final cluster. At each iteration, the data items that might switch the cluster due to centroids' deviation will be re-checked. The prototype implementation has been evaluated using data generated based on an uniform distribution random numbers generator. The evaluation showed up to 70% reduction of the running time. This paper will evaluate the optimized K-Means against real data sets from different domains.

#### I. INTRODUCTION

THE MORE data continues to grow in both quantity (volume) and diversity, the more challenging clustering algorithms become. Clustering refers to identifying data items' common characteristics (features or attributes) and grouping the data items according to a quantitative estimation of these characteristics. The resulted groups are usually called clusters and they have to be strongly differentiated by their underlying characteristics.

A wide range of domains have successfully employed clustering. Paper [2] made use of clustering for analysing markets as well as recommendations. Papers [3] [4] apply clustering in medicine. Paper [5] uses clustering to analyse news articles and their comments for e-business related purposes. Paper [6] employs clustering for predicting students academic results, while paper [7] applies clustering to human activity recognition.

Unless a mathematical model is available, choosing the most suitable clustering algorithm might prove a hard decision. Arguments that might lead the decision can range from complex experimental results to our own intuition.

Among other challenging open clustering-related issues like: heterogeneity, volume or scalability, that are worth putting research efforts into, raised by clustering, the execution time plays a very important role.

Our proposed optimization focuses on improving the execution time of the K-Means algorithm while keeping the same output. In use for more than four decades, the K-Means algorithm has been applied in a wide area of fields, ranging from artificial intelligence to image processing or from neural networks to machine vision, or more specifically in unsupervised learning, pattern recognition, classification analysis a.o.

The K-Means algorithm uses a set of cluster centers (cluster centroids) and distributes the data items to the cluster with the closest centroid in terms of Euclidean distance. Picking up the best initial centroids is still an open issue. Different centroids lead to different output and has an important influence on the performance of the algorithm. Choosing the right centroids is beyond the scope of this paper.

The standard K-Means algorithm implies successive exploration of the entire data space with the goal of distributing data items to clusters. At the end of every iteration, the centroids are re-computed by averaging the data items inside the cluster. The next iteration will make use of the newly computed centroids and re-distribute the data items. The loop continues until the centroids no longer change or until a maximum number of iterations has been reached.

The optimization introduced in [1] and presented also in this paper is based on an easily noticeable fact: after a small number of iterations, most of the data items no longer change their cluster, and at the same time, the centroids' deviation reduces significantly. So, why exploring the entire data space if only a small number of data items are subjects to changing the cluster?

Our solution aims of drawing a line between the data items that will certainly not change their cluster, avoiding their exploration in the next iteration(s), and the data items that might switch their cluster which obviously are to be checked.

Exploring the data space only partially on every iteration will not affect the centroid computation. The influence of the data items that are not subject to changing the cluster, on the future centroids, will be preserved in the next iteration(s).

That being said, it becomes obvious that the optimization does not affect the output in any way.

The paper's structure is as follows: section II presents previous attempts for reducing the execution time of the K-Means algorithm, section III describes our proposed optimization for the K-Means algorithm, section IV experimentally evaluates the algorithm against real data sets, (unlike paper [1] where the evaluation is performed against a randomly generated data set with uniform distribution), while section V concludes the paper.

#### II. RELATED WORK

The K-Means algorithm was subject to many research studies covering a wide range of optimization approaches, from computational complexity reduction to parallel and distributed implementations.

In paper [8] the authors propose an optimization that relies on the assumption that if a data item got closer to the centroid on the previous iteration, it will not change the cluster. The assumption allowed the implementation to reduce the amount of computations necessary for computing new centroids.

Parallel and distributed solutions have been discussed in [9][10][11] by treating important topics specific to this kind of environments: synchronization, communication overhead, data availability, architecture (peer-to-peer, client-server, a. o.). The parallel and distributed implementations showed considerable improvement when dealing with very big data sets.

GPUs proved to be a good host for highly-parallel implementations of the K-Means algorithm. Such platforms are addressed in papers [12][13]

# III. OPTIMIZING K-MEANS

In this section we will introduce both the standard K-Means algorithm and the optimized version of it, as proposed in paper [1]. The same optimization strategy will be discussed here, for a better understanding and reading experience.

Algorithm 1 presents the main phases of the standard K-Means algorithm.

A	lgori	thm	1	Standard	K-Means
---	-------	-----	---	----------	---------

1. Load initial centroids

2. Visit all data items and distribute them to the cluster with the closest centroid

3. For each cluster compute the average of all data items and set the result as the cluster's centroid

4. If the exit criteria (no centroid changes or the maximum number of iteration is reached) are not met, go to step 25. Exit.

At a closer look, we can immediately identify step number 2 as the one requiring the most execution time. The time spent on step 2 increases proportionally with the size of the data set, as the entire data set is explored at every iteration.

Figure 1 illustrates an example of centroids evolution of a standard K-Means algorithm. Data items are represented as 2D points. Centroids A, B and C start from their initial positions  $A_1$ ,  $B_1$ ,  $C_1$ , and successively traverse positions  $A_i$ ,  $B_i$ ,  $C_i$ , where i = 1..6.  $A_6$ ,  $B_6$  and  $C_6$  are the final positions of the centroids. One can easily notice that after only few iterations, the centroids get very close to their final position, which means that after a few iterations, the number of data items that are subject to changing the cluster reduces considerably.

This observation plays a key role in improving step 2 of algorithm 1; it states that after a small number of iterations,



Fig. 1. Centroids Deviation



Fig. 2. Point P in an Arbitrary Iteration

the number of points that must be visited (that might change the cluster) is reduced considerably. This lead us to defining specific criteria for splitting the data set in two collections: the former would be made of all data items that are subject to changing the cluster (let's call that the border collection) and the latter would be made of all other points (not changing the cluster). Before providing the mathematical criteria behind the two data item collections, let's examine figure 2.

Figure 2 is a snapshot of an arbitrary iteration around an arbitrary point P. Point P is part of cluster C as a consequence of the fact that the distance from P to C  $(d_{PC})$  is less than the distance to A  $(d_{PA})$  and the distance to B  $(d_{PB})$ . We are interested in evaluating the "safety distance" of point P, that is, the distance that P is missing from jumping to the next closest cluster (the cluster represented by centroid A). Let's call this distance  $e_P$  (distance to the cluster's edge). We can state that P is  $e_P$ -away from the next closest cluster.

$$e_P = min(d_{PA} - d_{PC}, d_{PB} - d_{PC})$$
(1)

At the end of the iteration, centroids A, B and C would be re-computed causing them to jump to new positions A', B', C'. The worst scenario for P is the following: A moves closer to P by |AA'|, B also moves closer by |BB'| while C moves away from P by |CC'|. The conditions that  $e_P$  must fulfil in order for P to remain in cluster C is:

$$e_P > |CC| + |AA| \tag{2}$$

and

$$e_P > |CC| + |BB| \tag{3}$$

For the sake of computation reduction, we can merge conditions 2 and 3 into the following condition:

$$e_P > 2 * max(|AA|, |BB|, |CC|) \tag{4}$$

Inequality 4 help us determining whether a certain point might change the cluster or not, but doesn't save us from visiting the entire data set. The solution is to group the data items by the value of  $e_P$ . We can split the range of values that e can take into intervals. Each group would be associated an interval. As long as  $e_P$  is greater than the interval's lower bound and lower or equal to the interval's upper bound, Pwould become part of the group associated with that interval. This allows us to change step 2 of algorithm 1 so that instead of visiting all points, we could visit the groups and check inequality 4. If inequality 4 returns false, it means the points inside that group have to be re-visited. Otherwise, all points inside the group will hold the cluster.

At the end of each iteration, new centroids are to be computed, which might cause centroids to deviate from their current position. To keep the groups synchronized, we will need to also update the lower and upper bounds of the associated intervals. That is, we will assume the worst case scenario presented above, which means the interval bounds would shift towards 0 by at most twice the maximum centroid deviation. More precisely, the bounds of the intervals would be reduced by 2 \* max(|AA|, |BB|, |CC|).

Algorithm 2 resumes the solution above.

Choosing the WIDTH constant has a big impact on the performance of the algorithm. The number of groups could explode if the value of WIDTH is to small. The other way around, if the value of WIDTH is to big, the intervals for  $e_P$  would be very wide causing them to be marked for re-visiting very often. Both extreme scenarios might reduce considerably the improvement brought by algorithm 2.

It is very hard to accurately define a general approach for choosing the right value of WIDTH. It depends a lot on the data distribution. However, in our research, we defined WIDTH's value as the average distance between adjacent data elements. Such a value for WIDTH, would increase the chances of balancing the groups. Algorithm 3 explains the procedure for defining the value of WIDTH, as it was used in our researches:

Therefore, the data set must be analysed prior to defining WIDTH, but if the data set is to big, this might require more time than we gain through the proposed optimization. A good compromise would be to analyse only a sample of the data set.

It can be easily noticed that the optimized K-Means has the same output as the standard K-Means. The quality of the clusters is preserved.

#### Algorithm 2 Optimized K-Means

1. Define constant WIDTH

2. Define group intervals

 $I_i = (i * WIDTH, (i+1) * WIDTH]$ 

3. Mark the entire data set to be visited

4. For each point to be visited

5.  $e = min(d_{PC_l} - d_{PC_w})$  where  $C_w$  is the center of the closest (winner) cluster and  $C_l, l = 1..k, l <> w$  stands for all other centroids

6. Map all points with  $i * WIDTH < e \le (i+1) * WIDTH$ to interval (i \* WIDTH, (i+1) \* WIDTH] where *i* is a positive integer

7. Compute new centroids  $C_j$ , where j = 1..k and their maximum deviation  $D = max(|C_iC_j|)$ 

8. If D = 0 or the maximum number of iterations was reached, move to 11

9. Update  $I_i$  s boundaries by subtracting 2 \* D (points owned by this interval got closer to the edge by 2 \* D)

10. Pick up all points that are mapped to an interval whose lower bound became less than or equal to 0, mark them for re-visiting, then go to 4

11. Exit

# Algorithm 3 Defining the WIDTH

1. Extract a 5-10% sample of the dataset

2. Traverse each data element in the dataset and compute the distance to it's neighbours.

 While traversing, average the distances to neighbours that do not differ by more than 50% than the current average value.
 Assign *WIDTH* the result of 3.

#### IV. EXPERIMENTAL EVALUATION

Experiments were carried out for three publicly available real data sets posted on UC Irvine Machine Learning Repository [14]. The execution environment was made of a Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, with 8GB of RAM memory, Ubuntu 14.04 operating system. We have selected three data sets that will be described below. A number of 2, 4, 8 and 12 centroids were randomly selected. Multiple runs were carried out for each scenario. The execution times shown below represent the average of all measured execution times for each scenario.

# A. US Census (1990) Data Set

The US Census 1990 [15] is a raw 1% sample data set (2458285 records) of the official US Census 1990, obtained of from the (U.S. Department of Commerce) Census Bureau website. For our experiments we've clustered the records by the age information. The WIDTH used for experiments, computed according to algorithm 3, was 1.

Data distribution is shown in figure 3

Results are shown in table I

One could expect such results considering that the data distribution on a wide segment (ages 0-70) is almost uniform. The improvement raises to 61.39% for two centroids, but it



Fig. 3. US Census 1990 Data Distribution

 TABLE I

 Results for US Census Data (1990) Data Set

	Number of Centroids			
	2	4	8	12
Time(s) - Standard K-Means	1611	3016	6079	7846
Time(s) - Optimized K-Means	622	1843	4998	6539
Improvement (%)	61.39	38.89	17.78	16.66

TABLE II Results for 3D Road Network Data Set

	Number of Centroids			
	2	4	8	12
Time(s) - Standard K-Means	469	2543	9274	17894
Time(s) - Optimized K-Means	196	833	3438	8104
Improvement (%)	58.16	67.24	62.92	54.70

drops as the number of centroids grow, to as low as 16.66% in case of 12 centroids.

#### B. 3D Road Network Data Set

The 3D Road Network data set [16] gives the elevation information for a 2D road network in North Jutland, Denmark. There are 434874 road segments. The road segments were clustered using both standard and optimized K-Means algorithms by their elevation. After applying the algorithm 3, the resulted value for WIDTH was 0.07.

Data distribution is shown in figure 4. It is not a uniform distribution, but the differences between adjacent data intervals are smooth.

Results are shown in table II

Results for the 3D Road Network Data set are very encouraging. Improvement was between 54% and 67% for a range of 2 to 12 centroids.



Fig. 4. 3D Road Network Data Distribution



Fig. 5. Bag of Words Data Distribution

## C. Bag of words Data Set

Bag of words [17] is a text collection obtained from five different sources. We've run our experiments against the New York Times collections which is made of words extracted from New York Times articles. The vocabulary is made of unique words resulted after removing stop-words and truncating the collection by only keeping words that occurred more than ten times. For each word, the number of occurrences was extracted, resulting a collection of 102660 words. The words were clustered using both standard and optimized K-Means algorithms by their number of appearances. Algorithm 3 indicated a value of 33.65 for the *W1DTH* constant.

Data distribution is shown in figure 5

The distribution shows a concentration of words by their appearance in the interval [0 - 12000] where approximately 100000 words fit, so more than 97% of the data set.

Results are shown in table III

For the Bag of Words Data Set, the Optimized K-Means, together with algorithm 3 for determining the value of the WIDTH constant, proved to be totally inefficient. They brought an increase of the execution time between 17.7% and 67.4%. The main reason for this increase seems to be the data distribution, which causes algorithm 3 to determine

-67.4

	Number of Centroids			
	2	4	8	12
Time(s) - Standard K-Means	125	339	1274	2697
Time(s) - Ontimized K-Means	148	399	1768	4514

-18.6

-17.7

-38.7

Improvement (%)

TABLE III Results for Bag of Words Data Set

an inappropriate value for the WIDTH. The bigger the distance between adjacent words (in terms of the number of appearances), the greater the value of WIDTH would be. A greater value for WIDTH would result into wider word groups, which once invalidated, result into re-visiting a bigger number of words. Also, the wider the group, the greater the chances are for that group to be invalidated and re-visited, even on small deviations.

### V. CONCLUSIONS

In this paper we evaluated an optimization for the K-Means algorithm proposed in a previous paper [1], by using real publicly available data sets. Two of the data sets (US Census (1990) and 3D Road Network) shown important improvements, very close the performance resulted by running the algorithm on the randomly generated uniform data set used in [1]. The common feature of the two data sets is that there is no sharp trend on their distribution chart. This is a very important feature when trying to calibrate the Optimized K-Means algorithm (computing the WIDTH constant). The third data set used (Bag of Words), as opposed to the ones mentioned above, showed an unacceptable loss of performance. The main reason for this loss is related to the data distribution, namely, more than 97% of the data elements are concentrated into less than 5% of their distance range (number of word appearances range). Calibrating the algorithm under these circumstances requires a more intensive analysis of the data set. Our future research efforts will focus on improving algorithm 3 to cover a wider range of data distributions.

#### REFERENCES

- Cosmin Marian Poteraş, Marian Cristian Mihăescu, Mihai Mocanu: An optimized version of the K-Means clustering algorithm, Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, ACSIS, Vol. 2, pages 695–699, 2014, DOI: 10.15439/2014F258.
- [2] Dolnicar, S: Using cluster analysis for market segmentation—typical misconceptions, established methodological weaknesses and some recommendations for improvement, Australasian Journal of Market Research, 2003, 11(2), 5–12.

- [3] Ng, H. P.; Ong, S. H.; Foong, K. W. C.; Goh, P. S.; Nowinsky, W. L.: Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm, 7th IEEE Southwest Symposium on Image Analysis and Interpretation, March 26-28, 2006, Denver, Colorado, pages 61-66
- [4] Agnieszka Wosiak, Danuta Zakrzewska: On Integrating Clustering and Statistical Analysis for Supporting Cardiovascular Disease Diagnosis, Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, ACSIS, Vol. 5, pages 303–310 (2015) DOI: 10.15439/2015F151, http://dx.doi.org/10.15439/2015F151
- 10.15439/2015F151, http://dx.doi.org/10.15439/2015F151
  [5] Hongwei Xie, Li Zhang; Jingyu Sun, Xueli Yu: Application of K-means Clustering Algorithms in News Comments The International Conference on E-Business and E-Government, May 2010, Guangzhou, China, pages 451-454
- [6] kK Oyelade, O. J, Oladipupo, O. O, Obagbuwa, I. C: Application of K-Means Clustering algorithm for prediction of Students' Academic Performance, (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 1, 2010, pages 292 - 295
- [7] Szymon Wawrzyniak, Wojciech Niemirom: Clustering Approach to the Problem of Human Activity Recognition using Motion Data, Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, ACSIS, Vol. 5, pages 411–416 (2015), DOI: 10.15439/2015F424, http://dx.doi.org/10.15439/2015F424
- [8] Souptik Datta, Chris Giannella, Hillol Kargupta: K-Means Clustering Over a Large, Dynamic Network, Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA. SIAM 2006 ISBN 978-0-89871-611-5, pages 153—164.
- [9] Yufang Zhang, Zhongyang Xiong, Jiali Mao, Ling O: The Study of Parallel K-Means Algorithm, Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21–23, 2006, Dalian, China, pages 5868–5871.
- [10] Jing Zhang, Gongqing Wu, Xuegang Hu, Shiying Li, Shuilong Hao: A Parallel K-means Clustering Algorithm with MPI, 4th Internation Symposium on Parallel Architectures, Algorithms and Programming, ISBN 978-0-7695-4575-2, pages 60-64, 2011.
- [11] Jitendra Kumar, Richard T. Mills, Forrest M. Hoffman, William W. Hargrove: Parallel k-Means Clustering for Quantitative Ecoregion Delineation Using Large Data Sets, Proceedings of the International Conference on Computational Science, ICCS 2011, Procedia Computer Science 4 (2011) 1602–1611.
- [12] Reza Farivar, Daniel Rebolledo, Ellick Chan, Roy Campbell: A Parallel Implementation of K-Means Clustering on GPUs, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2008, Las Vegas, Nevada, USA, July 14-17, 2008, 2 Volumes. CSREA Press 2008 ISBN 1-60132-084-1, pages 340–345.
- [13] Mario Zechner, Michael Granitzer: Accelerating K-Means on the Graphics Processor via CUDA, The First International Conference on Intensive Applications and Services INTENSIVE 2009, 20–25 April, Valencia, Spain, pages 7–15, ISBN 978-1-4244-3683-5.
- [14] M. Lichman: UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2013, http://archive.ics.uci.edu/ml
- [15] The USCensus1990raw data set, U.S. Department of Commerce Census Bureau - http://dataferrett.census.gov/
- [16] Manohar Kaul: Building Accurate 3D Spatial Networks to Enable Next Generation Intelligent Transportation Systems, Proceedings of International Conference on Mobile Data Management (IEEE MDM), June 3-6 2013, Milan, Italy
- [17] http://archive.ics.uci.edu/ml/datasets/Bag+of+Words