# K-Means clustering variants and improvements

Stefan Sebastian, 242

November 15, 2018

**Abstract**

TODO make abstract

# Contents

# 1 Introduction

## 1.1 Motivation

In recent times there has been an explosive growth of data. This new, digital resource has become so important that a recent article from The Economist magazine has called it 'the new oil'[1]. Some important sources of data, gathered by Bernard Marr[2], a business consultant in big data and AI technologies, are: Social Media (every minute 456000 tweets are sent on Twitter, 46740 photos are uploaded on Instagram, 4146600 YouTube videos are being watched), Communication (16 million text messages and 156 million emails are sent every minute), Services(45788 trips are made through Uber per minute), Internet of Things(8 million people use voice control every month). The amount of data in the world is predicted to reach 163 zettabytes by 2025[3], where one zettabyte represents a trillion gigabytes.

The main reason for choosing the K-means method for this paper is because I believe that the development of a computationally efficient unsupervised learning method will be necessary to keep up with future's data, which will continue to grow and sometimes lack meaningful labels.

## 1.2 Clustering

Clustering is a field of data analysis whose purpose is to organize points, or objects into natural clusters. In other words given N objects described by a set of features, find a number of groups, such that objects in the same cluster are similar to each other and different from objects in other clusters[4]. Given that there is no rigorous definition of similarity and that clusters in real data can overlap there are multiple algorithms and difference measures proposed in research.

## 1.3 K-means

The K-means algorithm was discovered independently in multiple scientific fields around the same time and its name comes from a paper published by MacQueen in 1967[4]. Despite being so old, this method is still widely used in clustering tasks because of its efficiency, simplicity and proven success.

The main goal of this algorithm is to minimize the distance between cluster centers and points assigned to them, which is a NP-hard problem. For this reason K-means is a greedy algorithm and has a chance to get stuck in local minimum instead of the best solution[4].

A basic version of the K-means algorithm is[5]:

1. Select an initial partition with K clusters and repeat 2 and 3 until the point membership no longer changes
2. Assign each point to the closest center
3. Update cluster centers based on the new assignment

# 2  Improvements

## 2.1  Execution time optimizations

As illustrated in the introductory section, data sets are growing larger and more varied. Consequently there is a need for algorithms that scale well and obtain good results without having huge computational costs. K-means is such an algorithm, since it is characterized by simplicity and has been proven to work empirically with a multitude of datasets[4]. This success has driven researchers to scale the algorithm for use on larger datasets. Next, I will present some of these execution time optimizations for the algorithm.

### 2.1.1  Enhanced K-means

A simple improvement for the basic K-means has been proposed by Fahim et al.[6] under the name of "enhanced K-means". The main idea is that some time can be saved during the computation of distances between points and cluster centers in the update phase of the original algorithm. This distance is calculated at every iteration of the algorithm but the information from previous iterations could be used, by employing an additional data structure that memorizes the previous distance between a point and its cluster's center. At every iteration the distance to the new cluster mean is calculated and if it's less or equal to the previous distance stored for that point then it is left in its cluster and there's no need to calculate the other k-1 distances.

The complexity of this algorithm can be approximated to O(nk) compared to the complexity of the classic algorithm which is O(nki), where n represents the number of points, k the number of clusters and i the maximum number of iterations. In brief, the time complexity when updating the position of a point is O(1) if it stays in its cluster and O(k) otherwise. Since the algorithm converges to a local minimum then the number of points updated in each iteration decreases, meaning the expected complexity is $nk \sum_{j=1}^{i} 1 \, / \, j$.

The performance of the algorithm was tested on 3 different datasets, with the following number of records and features: (20000, 16), (4177, 7), (6574, 15) and was compared with the classic K-means and the CLARA algorithm. The enhanced K-means obtains similar results on smaller number of clusters (approximatively less than 50) but outperforms the other methods on larger values being about twice as fast as CLARA and 5 times as fast as the classic variant on more than 100 clusters. In conclusion this algorithm can be a useful optimization for working with datasets in which we expect to find a large number of clusters.

### 2.1.2  K-means clustering using MapReduce

MapReduce[7] is a framework for processing large datasets over a cluster of computing nodes. It offers a programming model based on map and reduce functions and it has gained popularity in the big data community due to its simplicity, fault tolerance and scalability. This framework is a good starting point for clustering implementations that can work at large scale.

Zhao et al.[8] provide a detailed description of a simple parallel K-means implementation using MapReduce. The algorithm, called 'PKMeans' uses only one MapReduce job: a map function that assigns each sample to the closest center and a reduce function which updates the cluster centers. The input dataset is stored on a distributed file system

as a sequence of (key, value) pairs, where the key is the position of record in the data file, and the value is a string content of the record. Center values are stored on a global variable, so map functions can calculate distances independently. After each map task a combine function is applied which computes partial sums of the points assigned to the same cluster and their count. The reduce function takes as input the result of the combine task and uses it to compute the new means. The algorithm was tested with a number of computers varying between 1 and 4 and datasets of 1 up to 8GB and reported a linear speedup in computation time, meaning a system with m systems runs the algorithm m times faster. However, this speedup should decrease on larger systems due to increases in communication costs.

### 2.1.3   K-means algorithm on the GPU

A Graphical Processing Unit[9] is a special circuit designed for computer graphics and image processing. In contrast to CPUs, they have over a hundred cores that can handle thousands of threads at the same time, meaning GPUs are optimized for applying the same operation over a large batch of data. For this reason, a lot of data analysis algorithms have been implemented with this technology.

A sample implementation of K-means on the GPU has been published by Li et al.[10]. The algorithm can use two strategies, depending on the dimension of the data.

If the features of a data point can be stored on the GPU registers then, for the distance calculation step, each of the n points is dispatched to a thread. In the other case, a flow similar to matrix multiplication is used with the following matrices: data[n][d], centroid[d][k] and result[n][k]. To optimize latency, the matrices are processed in tiles, instead of individual elements. After this operation, in order to find the closest centroid, the task of analyzing a row in 'result' to find the max value is distributed across n threads.

For the step of updating the values of the means a divide and conquer strategy is used. The data is first divided into $n' = n/M$ groups, where M is the number of multiprocessors on the GPU and each group is reduced to get temporary centroids. The obtained data is then split again into M groups and the same process loops until the resulting number of loops is lower than M.

The first strategy is tested on datasets with up to 4 million points, 400 clusters and 8 feature dimension, finishing the computation in just 2.2 seconds. An impressive result for the second strategy is processing 494.080 points, with a 100 clusters and 34 features in just over a second.
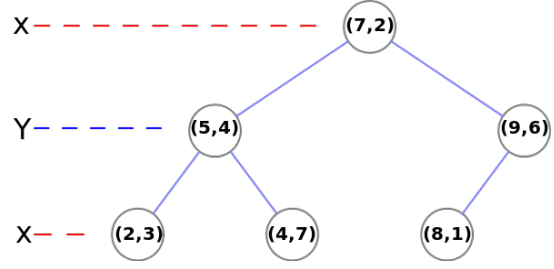
## 2.2   Seed selection

The K-means algorithm converges to a local optima[5], meaning it can be strongly influenced by the seed selection process. If the initial seeds are near a local optima then the algorithm is unlikely to find the best cluster centers. For this reason, various strategies have been proposed for efficient seed initialization.

### 2.2.1   K-d trees

A k-d tree, introduced by Jon L. Bentley[11], meaning k-dimensional tree is a data structure used to organize points in k-dimensional space, which is useful for partitioning data and optimizing multidimensional queries. The algorithm for building such a tree is recursive: select a point in the current dataset, choose a dimension, create a node of the
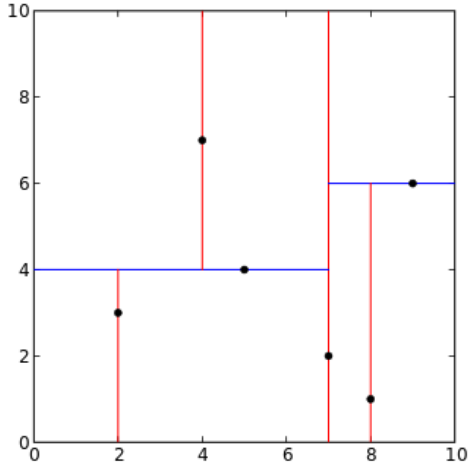
(a) An example of a 2d k-d tree[12]



(b) The data structure[13]

Figure 1: K-d tree graphic representation

selected point with a left subtree made out of the points with a lower value on the chosen dimension and a right subtree with the others.

An implementation of seed selection based on the k-d structure is proposed by Redmond and Heneghan[14]. The reason for using this data structure is to gain information about the density of the dataset. In order to suit the problem some decisions have been made while building the tree. At each step the dataset is split among the longest dimension of the bounding box. This box can be computed by taking the minimal and maximal values for each feature in the dataset, and building two new points. The longest dimension is simply the feature for which there is the greatest difference between the two points calculated in the previous step. The chosen condition for stopping and making a leaf node is when the size of the considered dataset is $\frac{n}{10K}$ or lower, which allows approximatively 10 leaf buckets for each cluster.

After the k-d tree is built some density estimations are made using the obtained leaf buckets. For each bucket the value of the density is calculated as $N/V$, where N is the number of points it contains and V is the volume of the bounding box. The volume is computed as the product of the differences between each feature of the two bounding points, or the geometric mean in case of two features being equal.

The heuristic for choosing the seeds is to get points that have a large density and are separated by a reasonable distance. Thus, the first seed is equal to the mean of the bucket with the highest density. The other k-1 seeds are chosen to be the ones that maximize the value g, where g is the product between the density of the bucket mean and the minimum distance to other bucket means. An optimization to avoid outliers is to compute the seeds, remove 20% of the least dense buckets, and then compute another pair of seeds.

The algorithm was compared to another 3 initialization methods (Forgy, Bradley and Fayyad, Katsavounidis), on 36 datasets and obtained better Distortion values on over 30 compared to each of the other methods. Distortion is the sum of the distances between each point and its cluster center.

# 3 Variants

## 3.1 Bisecting K-means

bisecting kmeans

## 3.2 Evolutionary K-means

genetic kmeans

# 4 Conclusions

TODO something conclusive

# References

[1] "The most valuable resource." https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data. Accessed: 2018-11-11.

[2] "How much data do we create every day? the mind-blowing stats everyone should read." https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read#7f96d4a060ba. Accessed: 2018-11-11.

[3] "What will we do when the world's data hits 163 zettabytes in 2025?." https://www.forbes.com/sites/andrewcave/2017/04/13/what-will-we-do-when-the-worlds-data-hits-163-zettabytes-in-2025/#4ff934f6349a. Accessed: 2018-11-11.

[4] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[5] R. C. D. Anil K. Jain, *Algorithms for clustering data*, ch. Clustering Methods and Algorithms, pp. 96–97. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1988.

[6] A. Fahim, A.-B. M.Salem, F. A. Torkey, and M. Ramadan, "Efficient enhanced k-means clustering algorithm," *Journal of Zhejiang University*, vol. 7, no. 10, pp. 1626–1633, 2006.

[7] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 137–150, 2004.

[8] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Cloud Computing* (M. G. Jaatun, G. Zhao, and C. Rong, eds.), (Berlin, Heidelberg), pp. 674–679, Springer Berlin Heidelberg, 2009.

[9] K. Krewell, "Whats the difference between a cpu and a gpu?." https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/, 2009. Accessed: 2018-11-14.

[10] Y. Li, K. Zhao, X. Chu, and J. Liu, "Speeding up k-means algorithm by gpus," *Journal of Computer and System Sciences*, vol. 79, no. 2, pp. 216 – 229, 2013. 10th IEEE International Conference on Computer and Information Technology, 2010.

[11] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, Sept. 1975.

[12] KindDragon33, "Kdtree figure 1." https://en.wikipedia.org/wiki/File:Kdtree_2d.svg, 2006. Accessed: 2018-11-15.

[13] MYguel, "Kdtree figure 2." https://en.wikipedia.org/wiki/File:Tree_0001.svg, 2008. Accessed: 2018-11-15.

[14] S. J. Redmond and C. Heneghan, "A method for initialising the k-means clustering algorithm using kd-trees," *Pattern Recognition Letters*, vol. 28, no. 8, pp. 965 – 973, 2007.