EXPLORING K-MEANS

Through Research and Experiment

Samuel Heersink – 7846520
CSI 4105 – University of Ottawa
December 4th, 2018

Contents

Abstract 2

Part 1 – Research 2

1.1 – On Clustering 2

1.2 – K-means 3

1.2.1 – The Problem 3

1.2.2 – The Algorithm 3

1.2.3 – Two Variations 4

Part 2 – Experiment 5

2.1 – Introduction 5

2.2 – Methodology 6

2.3 – Results 6

2.4 – Observations 8

2.5 – Considerations 8

2.6 – Conclusions 8

References 10

Abstract

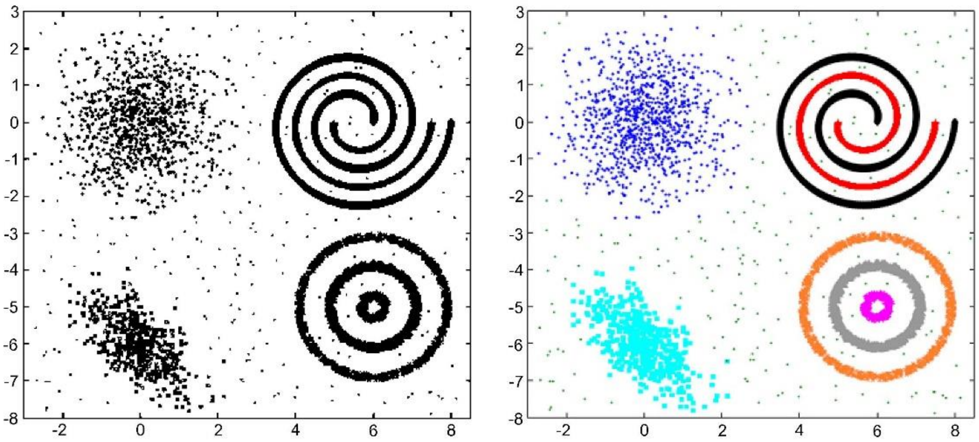
This paper attempts to explore the nature of the well-known K-means algorithm through a presentation of background research and a simple experiment. After a brief overview of the problem of clustering in general, the algorithm itself is presented. The paper examines the performance limitations of the algorithm and describes two variant algorithms, Bisecting K-means and K-means++. These variations are explored in detail with an experiment which attempts to explore the differences in performance of the two variations on several benchmark datasets. This experiment concludes that Bisecting K-means++ split on cluster size performs far better on larger (100k+ points or more) datasets, while K-Means++ is the best choice for smaller datasets.

Part 1 – Research

1.1 – On Clustering

The question of data clustering ties into the natural human desire to understand through classification. In every area of human knowledge, one finds diverse systems for classifying, ranking and sorting information. Though this problem seems to be a natural fit to be solved computationally, determining the intuitive divisions of information is a task not so easily solved by computers. Data clustering is a formalization of this problem, and many attempts have been made to develop clusterings that both reflect human intuition and are easily implemented by a computer.

Clustering is an analysis technique for recognizing the patterns in large amounts of unlabelled data. We label these patterns as clusters, and this labelling provides insight that we use to predict the labels of new data. The cluster shapes are entirely dependent on the nature of the data, and the problem of clustering lies in choosing an algorithm to find that natural clustering. Consider the seven clusters in Fig. 1; while easily recognized by a human, an algorithm may not successfully identify all of them due to their diversity in shape. Getting a computer to recognize the clusters is a matter of formulating the right question of what a cluster means in this context.



(a) Input data

(b) Desired clustering

Figure 1: Diversity of Clusters, from [1]

1.2 – K-means

1.2.1 – The Problem

K-means is likely the most widely used approach to the clustering problem. The term itself refers to both the formalization of the problem and the most common algorithm used to solve it. The formal question posed by K-means is this: given a set of data points and an integer K, what are the K centroid locations that minimize the distance from each point to its closest centroid? Solving this results in a partitioning of the data into K roughly spherical clusters of points. We can express the question as a pair of formulae:

Figure 2: K-means formula and the centroid calculation, from [2].
In this formula, K is the number of clusters, x is a single point in a cluster, π_x is the weight of that point, n_k is the number of points in a cluster, m_k is the mean of a cluster and C_k is a single cluster.

$$\min_{\{m_k\},1\leq k\leq K} \sum_{k=1}^{K} \sum_{\boldsymbol{x}\in C_k} \pi_{\boldsymbol{x}} \mathrm{dist}(\boldsymbol{x}, \boldsymbol{m}_k)$$

$$\boldsymbol{m}_k = \sum_{\boldsymbol{x}\in C_k} \frac{\pi_{\boldsymbol{x}} \boldsymbol{x}}{n_k}$$

Though Euclidean distance is most typically employed, the distance formula used is subject to interpretation.

1.2.2 – The Algorithm

The K-means problem is known to be NP-Hard [9], so no algorithm is known to solve it in polynomial time. However, the heuristic approximation algorithm that bears its name is simple, practical and flexible.

K-means the algorithm was first published by Lloyd in 1957 and is often referred to as Lloyd's Algorithm [2]. The main steps of the algorithm are as follows:

1. Randomly select K initial centroids
2. Assign each point to the cluster of the nearest centroid
3. Recalculate each centroid as the mean of all points in the cluster
4. Repeat steps 2 and 3 until the centroids do not change

K-means is guaranteed to terminate, though the upper bound for this termination is an exponential number of iterations. [6] This upper bound is a carefully crafted example that demonstrates the close link between this algorithm's effectiveness and the structure of the data that it operates on; in practice, on data displaying clustered structure, K-means terminates in far fewer iterations than this upper bound suggests.

Due to the greedy nature of the algorithm, K-means is subject to terminating with a local minimum offering no guaranteed approximation to the optimal solution. In addition, the simple definition of a cluster used by K-means leaves it unable to process non-spherical cluster shapes like those seen in fig. 1. Despite these shortcomings, K-means is simple to implement and extremely effective in practice, and many augmentations of and variations on K-means have been developed.

1.2.3 – Two Variations

K-means++ is an augmentation of the K-means algorithm by which the initial cluster centers are deliberately chosen to increase the algorithm's chances at terminating faster and with more optimal centroid locations. The random centroids in the first step of the algorithm are chosen one by one using a weighting proportional to their distance from those already chosen. Spreading the initial centroids out in this manner has been shown to outperform traditional K-means, and the method has been proven to obtain an O(log k) approximation of the optimal solution [3].

Bisecting K-means is another variation on the algorithm. It finds K centroids by repeatedly splitting clusters into two using traditional K-means until the desired number of clusters is reached. Starting with a single cluster of all the data, Bisecting K-means proceeds this way:

1. Pick a cluster to split
2. Perform K-means on this cluster, with K = 2
3. Repeat step 2 until K clusters have been found

The method for selecting a cluster to split is subject to implementation. Various scores are available for comparing the quality of clusters, but each of these has computational requirements that affect the execution time of the algorithm. Generally, the largest cluster is chosen to be split. Bisecting K-means has been shown to outperform traditional K-means as well as several other clustering methods when classifying documents. [4]

K-means++ and Bisecting K-means are two of the simplest variations on the traditional K-means algorithm, but they have both shown to have significant effects on its performance. An understanding of the behaviour of these variations when provided with differing structures of data is critical when choosing the correct variation to apply.

Part 2 – Experiment

2.1 – Introduction

During my research into variations on K-means I discovered two pieces of research that compared the performance of Bisecting K-means (BKM) and traditional K-means (KM). Steinbach et al. [4] compared KM and BKM with various agglomerative hierarchical clustering techniques at the task of document classification. Celebi and Marshall [5] applied KM and BKM to the task of colour quantization against several other color quantization methods including popularity, median-cut, and split-and-merge. Each of these projects reported different findings about the performance of these algorithms. Steinbach et al. found that BKM performed better than both KM and all the other clustering methods they applied. Celebi and Marshall found that KM achieved better results than BKM on average, despite having a higher standard deviation. This difference in findings between the two researchers' projects is likely due in part to the nature of the tasks at hand.

Colour quantization is the task of reducing the colour space for an image so that it can be manipulated more efficiently. True-colour images can contain 16 million possible colours, and each pixel in the image must be capable of representing any one of those colours. By plotting the colours of each pixel in three dimensions, k clusters can be found to reduce the number of possible colours from 256^3 to k. The better the clustering, the less colour distortion in the quantized image. [5]

Document classification is another common application for clustering algorithms such as K-means. A collection of documents is classified into K classes according to their similarity with respect to their shared content. Each word in a collection of documents is represented as a feature of the space, which results in an extremely high dimensionality for the data.

The difference between the applications of the algorithms in these two reports is quite clear. Celebi and Marshall's task of colour quantization dealt with many points in 3 dimensions and up to 256 clusters. Steinbach et al.'s document classification dealt with data in up to 31,472 dimensions but no more than 25 clusters. In this report I attempt to learn more about the differences in performance in these two algorithms, as well as that of the K-means++ initialization, when applied to data of varying natures.

2.2 – Methodology

In order to compare the relative effectiveness of traditional K-means, K-means++ and Bisecting K-means I used a collection of clustering benchmark datasets used for comparing clustering algorithms, obtained from [7]. These datasets each possessed different qualities in terms of dimensionality, scale, and cluster count, shown in Fig. 3. With inspiration from the methods displayed in [8], I used scikit-learn and python to run six different variations of K-means on each dataset. I tested traditional K-means, K-means++, and four variations of Bisecting K-means: splitting on either the largest or worst clusterings and implementing either K-means or K-means++. I recorded the execution time for each algorithm and measured several scores evaluating cluster accuracy: Mutual Information Score, V-Measure, Folkes-Mallows Score and the Adjusted Rand Index.

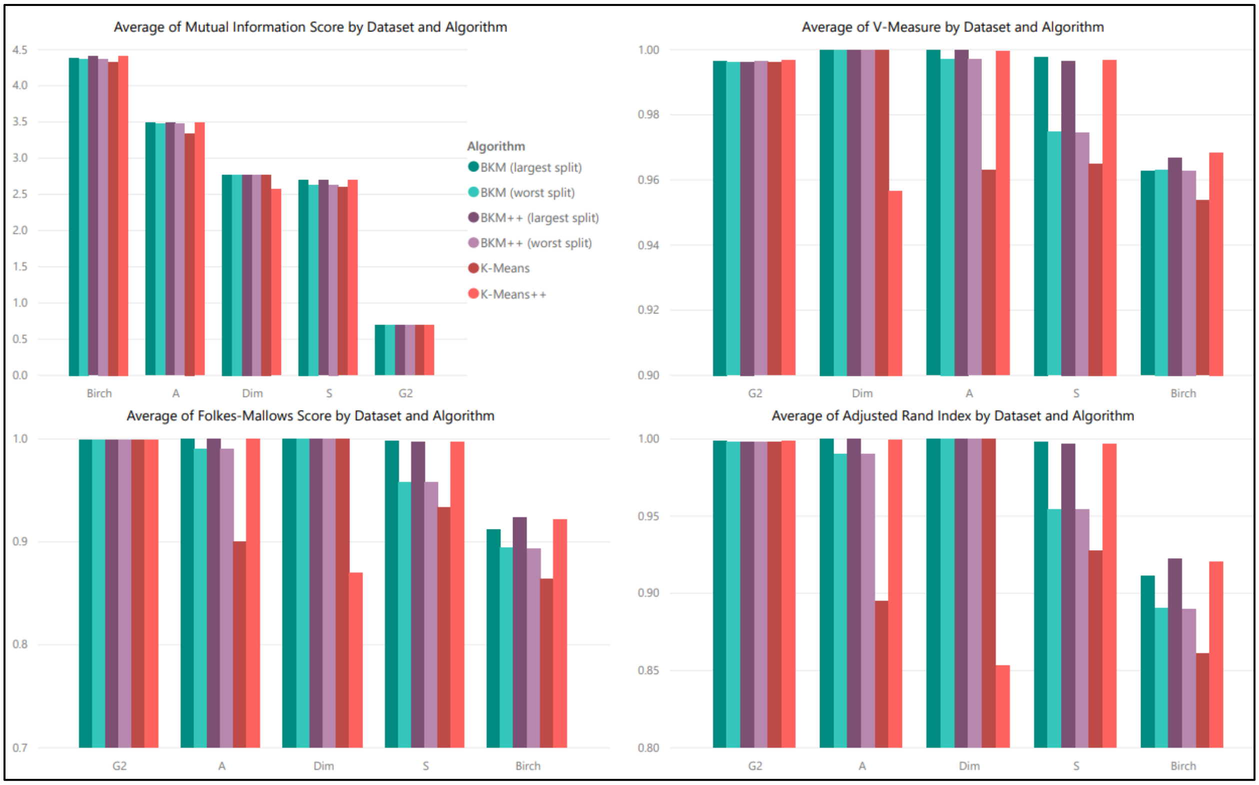
Dataset	Subsets	Dimensions	Clusters	Points
S	3	2	15	5000
A	3	2	20 – 50	3k – 7.5k
Birch	3	2	100	100k
G2	110	1 – 1024	2	2048
Dim	6	32 – 1024	16	1024

Figure 3: Differing qualities of the datasets used

2.3 – Results

I present my results in the form of two figures, shown on the following page.

Figure 4: Averages of each clustering evaluation score per algorithm and dataset



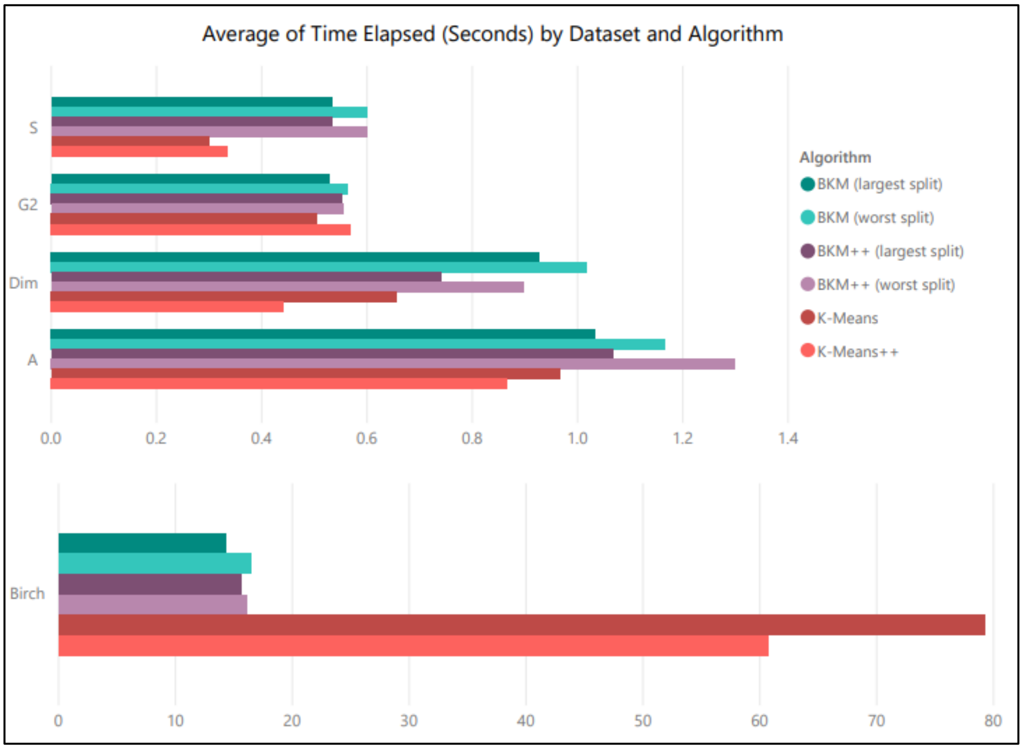


Figure 5: Average time elapsed per dataset and algorithm

2.4 – Observations

There are some key observations to be made from this data:

- Performance with respect to scores is almost the same for all methods.
- All the scores agreed with each other in terms of relative performance of the algorithms.
- Most algorithms displayed perfect or near-perfect scores for the G2 dataset due to its simple 2-clustering.
- Bisecting K-means split on cluster size outperformed the split on cluster quality in every way and for both traditional K-means and K-means++ implementations – it was superior on all scores as well as the time elapsed.
- With one exception, K-means++ and Bisecting K-means++ performed better than their traditional counterparts.
- The combination of high dimensionality and moderate cluster count of the Dim set made it the exception to the previous observation. All algorithms performed equally save for K-means++, which had the worst scores in all categories.
- The Mutual Information Score was not a normalized metric like the others, so scores across datasets could not be compared.
- While the simpler K-means and K-means++ had lower execution times than their bisecting counterparts on the smaller datasets, they far exceeded their times on the larger Birch dataset.

2.5 – Considerations

Regrettably, the results recorded did not include the details for each individual subset. Including the differing qualities such as cluster size, number of points and dimensionality along with the other results would have allowed for valuable observations to be made with respect to those qualities. In addition, normalizing the mutual information score like the others would have made it a more valuable metric in this context. Lastly, multiple test runs may have offered insight into the deviation of traditional K-means reported by Celebi and Marshall.

2.6 – Conclusions

To draw conclusions from these observations, K-means or K-means++ would be the most effective algorithms to apply on a relatively small dataset with few clusters (given a moderate cluster count and high dimensionality, traditional K-means may perform better than K-means++). Since the accuracy of all the algorithms is comparable within these parameters, the lower execution times of K-means and K-means++ make them the optimal choice. With large datasets such as Birch, Bisecting K-means split on cluster size is far preferred to the other algorithms due to near-equal performance and significantly lower execution time.

References

[1] A. Jain, "Data clustering: 50 years beyond K-means", *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651-666, 2010.

[2] J. Wu, *Advances in K-means clustering*. Springer, 2014.

[3] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding", Stanford, 2006.

[4] M. Steinbach, G. Karypis and V. Kumar, "A Comparison of Document Clustering Techniques", University of Minnesota, Minneapolis, 2000.

[5] M. Celebi and S. Marshall, "Comparison of Conventional and Bisecting K-means Algorithms on Color Quantization", Louisiana State University, Shreveport, 2012.

[6] A. Vattani, "K-means Requires Exponentially Many Iterations Even in the Plane", *Discrete & Computational Geometry*, vol. 45, no. 4, pp. 596-616, 2011.

[7] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets", *Applied Intelligence*, vol. 48, no. 12, pp. 4743-4759, 2018.

[8] J. VanderPlas, *The Python Data Science Handbook*. O'Reilly, 2016.

[9] M. Mahajan, P. Nimborkhar and K. Varadarajan, "The Planar K-means Problem is NP-hard", in *Proceedings of the 3rd International Workshop on Algorithms and Computation*, Kolkata, India, 2009, pp. 274-285.