K-Means Clustering in Database Systems

Tyler Casella
Department of Computer Science
Cal Poly San Luis Obispo
tcasella@calpoly.edu

December 2012

Abstract

With the increasingly influencial challenges faced by big data, methods for extracting useful information from within the large quantity of information have significant implications on the future. This survey examines one of these methods: K-means clustering. K-means clustering aims to partition data into several groups based on similarity to eachother. This paper examines how K-means clustering works, its potential in databases, and will explore contributions made related to the algorithm..

1 Introduction

As we continue further into the Information Age, information has become a major commodity. We are presented with a plethora of available data sources, but often useful information is not explictly discernible within the data. Extracting potentially useful information from big data is an incredible challenge but yields valuable results. This survey examines one tool often used to perform this difficult task: K-means clustering.

K-means clustering is a method by which elements of dataset are divided into K different groups based on similarity to one another. This approach is very similar to classification, however is not the same [4]. In classification, a distinct set of classes are specified and each data element is partitioned into each class which is pertains to. K-means clustering differs, however, in that classes are not specified at all. Instead only the number of classifications is specified and K-means clustering will determine what each classification should be to derive the closest fitting clusters that are most dissimilar to one another. From this, associations and similarities otherwise hidden can be extracted from the dataset.

The goal of the K-means clustering algorithm is to minimize the distance function between each data element and its subsequent cluster centroid. This is calculated as [4]:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} \|d(X_i, C_j)\|$$

Where $\|d(X_i, C_j)\|$ is the distance between the data element X_i and cluster centroid C_j.

During each pass of the algorithm, every data element is assigned to the nearest partition based on this measure of similarity. Euclidean distance is often the used for $d(X_i, C_j)$. The process is then repeated updated, allowing for elements to change partitions as their centroids evolve [4].

The algorithm can be summarized as follows [4]:

1. K cluster centroids are initalized randomly.
2. Assign each data element to the nearest centroid to establish partitions.
3. Recalculate centroids of each partition.
4. Repeat steps 2 & 3 until each subsequent change in centroid is negligible.

2 Related Work

Due to the NP-hard computational difficulty the K-means cluster algorithm presents, a large amount of work has been conducted on maximizing the efficiency of the algorithm.

2.1 Far Efficient K-Means Algorithm

[4] presents an revision to the standard K-means algorithm to allow said efficiency to be obtained. The proposed algorithm changes two key aspects of the standard K-means algorithm: the initial centroid selection process and the iterative process of selecting clusters for data.

The altered algorithm is summarized as follows [4]:

1. Find the farthest pair of data (d_1 and d_2) and select as the initial two cluster centers.
2. Find the data element closest to d_1 and add it to the d_1 cluster. Remove the element from the data set
3. Repeat step 2 until the number of data elements in d_1 reaches a threshold of 50% of (N/K). Calculate the arithmetic mean of the d_1 cluster to derive its centroid c_1.
4. Repeat steps 2 & 3 for the d_2 cluster to obtain centroid c_2.
5. Select a third cluster center as the center of the data element d_i such that:
$$max(min(distance(\{d_i, c_1\}, \{d_i, c_2\})))$$
6. Find the data element closest to the new cluster and add it to the cluster before removing from data set. Repeat until threshold is reached.
7. Calculate d_i cluster centroid c_i.
8. Repeat steps 5 through 7 until the desired number of clusters is reached.

With these revisions, [4] found significant performance increases based on both the Davies-Bouldin and Dunn index.

3 K-Means Clustering of Heterogeneous Data

With the advent of big data, data available is becoming more and more heterogeneous as the number of potential sources expands. [1] examines the potential for K-means clustering to perform extraction of potentially beneficial information.

When applying the K-means clustering algorithm to such data, pre-processing must be performed to transform the documents into an algebraic model such as the vector model space (VSM). A common method for storing text in vector form is via Term Frequency - Inverse Term Frequency(TF-IDF) [1]. TF-IDF is defined as:

$$TF-DF(t_k, d_j) = \frac{Freq(t_k, d_j)}{\sum_k Freq(t_k, d_j)} xLog \frac{N_{doc}}{N_{doc}(t_k)}$$

Where $Freq(t_k, d_k)$ represents the frequency that the term t_k occurs in the document d_j. $\sum_k Freq(t_k, d_j)$ is the total number of all terms within the document d_j. N_{doc} denotes the number of documents in the corpus and $N_{doc}(t_k)$ is the number of documents containing the term t_k within the corpus [1].

The same can be applied to numeric data using the analagous representation Bin Frequency - Inverse Document Bin Frequency (BF-IDBF) [1], which is defined identically.

Once these representations are obtained, the data can be converted to a *unified vectorization*, combining the both types into a single vector. Depending on how the tokenization is performed, whether on a per word basis or more, the dimensionality can become too burdensome for the K-means clustering computation. [1] achieves dimension reduction via Random Projection (RP) due to its simplicity and speed. With these preprocessing steps complete, the dataset is then ready to be processed by the K-means clustering algorithm.

4 K-Means Clustering in SQL

As the realm of big data advances, mining important information from a dataset becomes more of a stepping-stone within a large list of required procedures. As such, having a light-weight method for applying K-means clustering to a DBMS via SQL is extremely desirable, as it is widely available.

[5] explores an implementation of the K-means clustering algorithm in SQL. The input of the implementation is a kxd dimension table representing the dataset. The output is three matrices W, C, R representing k weights, wehre k is the number of clusters, k means (centroids), and k variances (squared distances) [5]. Particular optimizations can be made by allowing synchronized scans in which multiple queries can be submitted and performed within a single scan.

5 Applications

One of the shining abilities of K-means clustering is its ability to shine light on information that may have not been realized otherwise. This makes it an extremely applicable tool for many industries.

One such application is detecting market trends. By maintaining transaction information within stock houses, market trends can be detected via clustering which can reveal which products will likely sell well and which are likely to bust. This application is explored by [6].

Another large application seeing increased use today is image recognition. For many years K-means clustering has been used within computer graphics to perform texture synthesis in which a small image is transformed in some way (often in size) while seamlessly mantaining the same patterns as the original image. Similar concepts can be applied in reverse to find images which are similar in nature to an input image. [2] proposes a method for using K-means clustering to perform image retrievals from a database.

6 Conclusion

As big data continues to grow, data mining will become an even more integral part of our lives. K-means clustering is on the forefront of this movement, allowing for efficient use of vast collections of data. We've explored a number of methods for implementing and utilizing K-means clustering, however we are only starting to see its potential role in the future as the amount of potentially useful data available grows.

References

[1] F. Bourennani, M. Guennoun, and Y. Zhu, "Clustering relational database entities using k-means," in Advances in Databases Knowledge and Data Applications (DBKDA), 2010 Second International Conference on, april 2010, pp. 143 –148.

[2] H. Liu and X. Yu, "Application research of k-means clustering algorithm in image retrieval system," in Proceedings of the Second Symposium International Computer Science and Computational Technology, ser. ISCSCT 09, 2009, pp. 274–277.

[3] X. Liu, C. Zhou, L. Zhang, H. Sheng, and J. Li, "Quick searching based on l-k means hierarchical clustering in a huge scale face database," in Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for, nov. 2008, pp. 42 –47.

[4] B. K. Mishra, A. Rath, N. R. Nayak, and S. Swain, "Far efficient k-means clustering algorithm," in Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ser. ICACCI '12. New York, NY, USA: ACM, 2012, pp. 106–110. [Online]. Available: http://doi.acm.org/10.1145/2345396.2345414

[5] C. Ordonez, "Integrating k-means clustering with a relational dbms using sql," Knowledge and Data Engineering, IEEE Transactions on, vol. 18, no. 2, pp. 188 – 201, feb. 2006.

[6] D. Shalini, M. Shashi, and A. Sowjanya, "Mining frequent patterns of stock data using hybrid clustering," in India Conference (INDICON), 2011 Annual IEEE, dec. 2011, pp. 1 –4.