International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume I, Issue III, July - 2014

COMPARISON OF K-MEAN ALGORITHM & APRIORI ALGORITHM – AN ANALYSIS

Dr.C.Kumar Charliepaul [1]
Principal
A.S.L Pauls College of Engg & Tech,
Coimbatore .
charliepaul1970@gmail.com

G.Immanual Gnanadurai [2]
Assistant professor / CSE
Dhaya College of Engineering,
Madurai
imman8601@gmail.com

ABSTRACT

Data mining is a relatively new term, the technology is not. Companies have used powerful computers to sift through volumes of supermarket scanner data and analyze market research reports for years. However, continuous innovations in computer processing power, disk storage, and statistical software are dramatically increasing the accuracy of analysis while driving down the cost. While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks.

KEY WORDS: *Clustering, K-Mean Algorithm, Apriori Algorithm*

INTRODUCTION

K-MEAN is a Data Mining project in which we are going to use Clustering techniques such as global k-means and x- means. Global K-mean and X-mean are the useful algorithms for all clustering problems. Global K-mean and X-mean algorithms simply are defined as techniques to group same objects into an existing category. The main aim of this final year Microsoft project is to group similar data items into one group and separates dissimilar objects into different groups so as to make the search of required information easy and save the user time. These crystal reports are generated as the output of each query.
k-means clustering is a data mining/machine learning algorithm used to cluster observations into groups of related observations without any prior knowledge of those relationships. The k-means algorithm is one of the simplest clustering techniques and it is commonly used in medical imaging, biometrics and related fields.

Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

- In computer science and data mining, **Apriori** is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation).
- The algorithm attempts to find subsets which are common to at least a minimum

number C (the cutoff, or confidence threshold) of the item sets.

K-MEAN ALGORITHM

The main idea from the K-Means algorithm is to provide the classification of a lot of information based on its own data. This classification, that it will be shown next, is based on analysis and comparison between numerical values from the data. Thus, the algorithm automatically will provide a autonomous classification without human supervision, that is, with no existing classification. Because of this characteristic, the K-Means is considered as an unsupervised data mining algorithm.

To understand how the algorithm works, let's imagine that we have a table distributed with lines and columns that contains a lot of samples to be classified. In this table, each column is called of dimension and each line contains information for each dimension, which can be also called of ocurrences or dots. Generally, this algorithm works with continously samples, but it can also treat discrete data, provided that they must be mapped to corresponding numerical values.

As i said earlier, the algorithm will analyse all the samples of this table and generate clusters (classifications). So, the algorithm will classify the data into one cluster and indicate which lines (patterns) belong to this cluster (class). The user or the developer must provide to the algorithm the number of clusters (k) that the data must be partitioned. This number of clusters (K) remembers the first letter of the algorithm: K-means.

To generate the clusters and classify the samples, the algorithm makes a comparison between each value of the line based on a distance measure. Generally, it's used the euclidian distance to calculate how "far" the attribute of the pattern is from each other. How to evaluate this distance depends on how many attributes exist from the provided table. After the calculation of the distances, the algorithm computes the centroid for each one of the clusters. While the algorithm goes through each step, the value of each centroid is recomputed based on the mean of the values of each attribute of each pattern that belongs to this centroid. Thus, the algorithm results with k centroids and put the paterrns of the table in accordance to its distance of centroids.To simplify all the explanation of how the algorithm works, i will present the K-means process at the following steps:

Step 01: **Begin with a decision on the value of k = number of clusters.**

In this step, the k centroids must be initiated. You may assign the training samples randomly, or systematically as the following:

1. Take the first k training samples of the table as single-element clusters
2. Assign each of the remaining (N-k) training samples to the cluster with the nearest centroid. After each assignment, recomputed the centroid of the gaining cluster.

Step 02: **Create a distance matrix between each pattern and the centroids.**

In this step, for each sample in sequence compute its distance from the centroid of each of the clusters. The drawback of this step is the heavy calculation, since we have N samples and k centroids, the algorithm will have to evaluate NxK distances.

Step 03: **Put each sample in the cluster with the closest centroid (minimal distance).**

Here, the samples are classified based on its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster. Notice that the algorithm will end when no data is moving to another cluster anymore.

Step 04: Update the new centroids for each cluster.

At this moment, the centroid location is updated. For each centroid of the cluster that gained or lost a sample, its location is updated through calculating the mean of each attribute of all samples that belong to the respective cluster.

Step 05: Repeat until the convergence condition satisfied.

The algorithm comes back to the **Step 02** , repeating the adjustment process of the location of each centroid until convergence is achieved, that is until a pass through the training sample causes no new assignments.

The fluxogram of all steps described above can be ilustrated at the Figure 01 below.

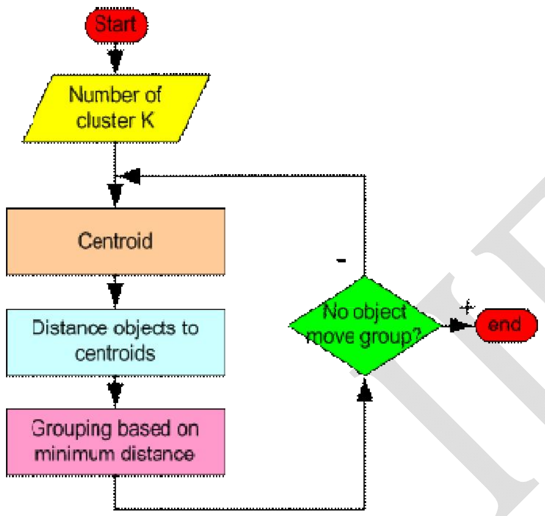

Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Figure 01. K-means Algorithm Process

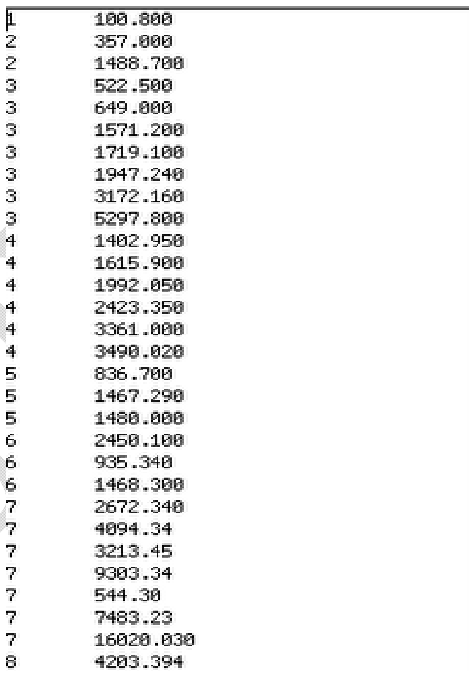
One can note that we will have a classification that puts each sample at only one cluster. Thus, we can conclude that this algorithm generates a "hard clustering," once each sample can only be classified at only one class. Other algorithms work with the "soft" classification concept, where there is one metric that measures the degree of the sample belong to each class.

If you want to read more about the algorithm K-Means, you can look further at the link below and even download other implementations of the algorithm:

Now that we introduced the algorithm, let's see a practical example using the K-means technique.

Practical Example of the use of K-means.

In this example, let's suppose a company that sells products to clients by orders made up of a list of itens (products). To make easy the comprehension of the problem and the data scenary, we will consider objects (X clients) and each object have two attributes or features as shown in Figure o2 below.



1	100.800
2	357.000
2	1488.700
3	522.500
3	649.000
3	1571.200
3	1719.100
3	1947.240
3	3172.160
3	5297.800
4	1402.950
4	1615.900
4	1992.050
4	2423.350
4	3361.000
4	3490.020
5	836.700
5	1467.290
5	1480.000
6	2450.100
6	935.340
6	1468.300
7	2672.340
7	4094.34
7	3213.45
7	9303.34
7	544.30
7	7483.23
7	16020.030
8	4203.394

Figure 02. Data Set samples

Based on this model, the marketing department desires to segment the clients to offer exclusive discounts and other benefits. Our goal is to group these clients of the marketing data set into three categories: Gold , Silver and Bronze Clients. The client classification criteria must consider only the two attributes: the total of orders of each client and the total cost of the client in his all orders without discounts. Obviously that the clients that have more orders and with higher total costs will be classified as Golden Clients.

With the all objects shown at the table at the Figure 2, each client will be represented as one point with two attributes (X,Y) where X = Number of Orders and Y = Total Cost. The Figure 02 shows the chart based on those attributes mapped into coordinates.

We will use the algorithm K-means to classify the data set in accordance to the marketing department wants. As it was only specified two attributes (Total Cost and number of orders) , those will be used to classify the clients. In real problems the algorithm K-Means could also work with any number of attributes to classify the objects.

Analyzing the data of the Figure 03, we can predict that the three clients will be classified as Golden Clients, therefore it's easy to see the distance between these clients and the others. However it's not so easy the classification of the rest of the clients into Silver and Bronze Clients categories.

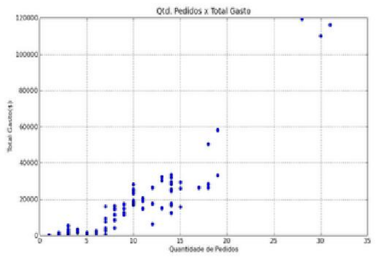


Figure 03. Scatter Plot (Total of Orders x Total Cost)

To help the classification of these clients, we will use a implementation of the K-means algorithm that will work with only two attributes. This implementation in Python named *k_means.py* and can be seen here.

To execute the algorithm we the module at the console with the following parameters:

%% python k_means.py <"dataSet pathFile">

EX: %%python k_means.py "C:/dataSet.txt" 3

Running the algorithm with the data set we presented earlier, the results were very satisfactory. In our example, the K-means classified the data into three classes: class 1, 2 and 3. Based on the definition of the client type, we associate the class 1 to Bronze Client, the class 2 to Silver Client and class 3 to the Golden Client. Putting these samples at scatter plot, we can visualize clearly the classification of the clients. This chart is shown at the Figure 04.

Interpreting the chart presented at the Figure 04, the clients represented by the color green are the Golden Clients, the clients at color red are the Silver Clients and the clients at blue color are the Bronze Clients. The three triangles in yellow point are the centroids calculated by the algorithm.

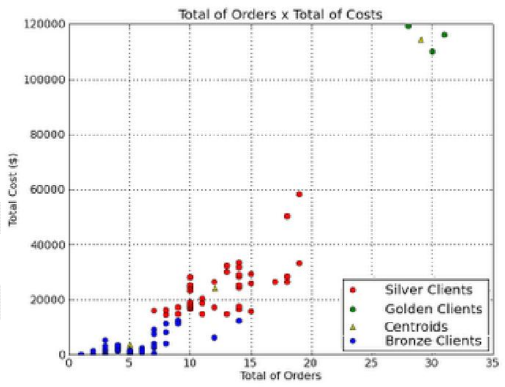


Figure 04. Clustered Data after running the K-means algorithm.

With the use of the K-means algorithm, it's possible to classify the current clients in accordance to their number of orders and the total cost in all orders, as the company marketing department desired. To classify a new client, just execute again the implementation and verify which is its classification. Thus, all clients will be again analyzed and classified.

One future feature could also be implemented is compare the attributes of a new client to the centroids ones before including it at the data set. This comparison is done by calculating the distance between the values of the new client and values of all centroids provided by the algorithm. Thus, the new client is said belong to the cluster that has minimum distance from this data.

APRIORI ALGORITHM

Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing). Each transaction is seen as a set of items (an *itemset*). Given a threshold C, the Apriori algorithm identifies the item sets which are subsets of at least Ctransactions in the database.

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as *candidate generation*), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori uses breadth-first search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length $k$from item sets of length $k-1$. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.

The pseudo code for the algorithm is given below for a transaction database T, and a support threshold of ϵ. Usual set theoretic notation is employed, though note that $T$is a multiset. $C_k$is the candidate set for level k. At each step, the algorithm is assumed to generate the candidate sets from the large item sets of the preceding level, heeding the downward closure lemma. $count[c]$accesses a field of the data structure that represents candidate set c, which is initially assumed to be zero. Many details are omitted below, usually the most important part of the implementation is the data structure used for storing the candidate sets, and counting their frequencies.

```
Apriori($T, \epsilon$)
    $L_1 \leftarrow \{\text{large } 1-\text{itemsets}\}$
    $k \leftarrow 2$
    while $L_{k-1} \neq \emptyset$
        $C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \in \bigcup L_{k-1} \wedge b \notin a\}$
        for transactions $t \in T$
            $C_t \leftarrow \{c \mid c \in C_k \wedge c \subseteq t\}$
            for candidates $c \in C_t$
                $count[c] \leftarrow count[c] + 1$
        $L_k \leftarrow \{c \mid c \in C_k \wedge \; count[c] \geq \epsilon\}$
        $k \leftarrow k + 1$
    return $\bigcup_k L_k$
```

Examples
Example 1

Consider the following database, where each row is a transaction and each cell is an individual item of the transaction:

alpha	beta	epsilon
alpha	beta	Theta
alpha	beta	epsilon
alpha	beta	Theta

The association rules that can be determined from this database are the following:
1. 100% of sets with alpha also contain beta
2. 50% of sets with alpha, beta also have epsilon
3. 50% of sets with alpha, beta also have theta

we can also illustrate this through a variety of examples

Example 2
Assume that a large supermarket tracks sales data by stock-keeping unit (SKU) for each item: each item, such as "butter" or "bread", is identified by a numerical SKU. The supermarket has a database of transactions where each transaction is a set of SKUs that were bought together.
Let the database of transactions consist of following itemsets:
Itemsets

{1,2,3,4}
{1,2,4}
{1,2}
{2,3,4}
{2,3}
{3,4}
{2,4}

We will use Apriori to determine the frequent item sets of this database. To do so, we will say that an item set is frequent if it appears in at least 3 transactions of the database: the value 3 is the support threshold.

The first step of Apriori is to count up the number of occurrences, called the support, of each member item separately, by scanning the database a first time. We obtain the following result

Item	Support
{1}	3
{2}	6
{3}	4
{4}	5

All the itemsets of size 1 have a support of at least 3, so they are all frequent.

The next step is to generate a list of all pairs of the frequent items:

Item	Support
{1,2}	3
{1,3}	1
{1,4}	2
{2,3}	3
{2,4}	4
{3,4}	3

The pairs {1,2}, {2,3}, {2,4}, and {3,4} all meet or exceed the minimum support of 3, so they are frequent. The pairs {1,3} and {1,4} are not. Now, because {1,3} and {1,4} are not frequent, any larger set which contains {1,3} or {1,4} cannot be frequent. In this way, we can prune sets: we will now look for frequent triples in the database, but we can already exclude all the triples that contain one of these two pairs:

Item	Support
{2,3,4}	2

In the example, there are no frequent triplets -- {2,3,4} is below the minimal threshold, and the other triplets were excluded because they were super sets of pairs that were already below the threshold.

We have thus determined the frequent sets of items in the database, and illustrated how some items were not counted because one of their subsets was already known to be below the threshold.

Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets (the algorithm attempts to load up the candidate set with as many as possible before each scan). Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset S only after all of its proper subsets.

APRIORIALGORITHMS ADVANTAGES

- Uses large itemset property
- Easily parallelized
- Easy to implement

APRIORIALGORITHMSDIS ADVANTAGES

- Assumes transaction database is memory resident.
- Requires many database scans.

K-MEANS ADVANTAGES:

1) If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.
2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

K-MEANS DISADVANTAGES:

1) Difficult to predict K-Value.
2) With global cluster, it didn't work well.
3) Different initial partitions can result in different final clusters.

4) It does not work well with clusters (in the original data) of Different size and Different density

CONCLUSION:

Association Rules form an very applied data mining approach. Association Rules are derived from frequent item sets.The Apriori algorithm is an efficient algorithm for finding all frequent item sets. The Apriori algorithm implements level-wise search using frequent item property.The Apriori algorithm can be additionally optimized.There are many measures for association rules.Initially, the number of clusters must be known, or chosen, to be K say. The initial step is the choose a set of K instances as centers of the clusters. Often chosen such that the points are mutually “farthest apart”, in some way.Next, the algorithm considers each instance and assigns it to the cluster which is closest. The cluster centroids are recalculated either after each instance assignment, or after the whole cycle of re-assignments. This process is iterated

REFERENCES

[1].Andrew Moore: “K-means and Hierarchical Clustering - Tutorial Slides” http://www2.cs.cmu.edu/~awm/tutorials/kmeans.html

[2]. Brian T. Luke: “K-Means Clustering”http://fconyx.ncifcrf.gov/~lukeb/kmeans.html

[3].Tariq Rashid: “Clustering” http://www.cs.bris.ac.uk/home/tr1690/documentation/fuzzy_clustering_initial_report/node11.html

[4].Hans-Joachim Mucha and Hizir Sofyan: “Nonhierarchical Clustering” http://www.quantlet.com/mdstat/scripts/xag/html/xaghtmlframe149.ht Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in knowledge discovery and data mining* (pp. 307–s328). Menlo Park: AAAI Press.

[5] . http://untroubled.org/spam/Mining Model Content for Association Models (Analysis Services - Data Mining)

[6].Fast Algorithms for Mining Association Rules R. Agrawal and R. Srikant *Proc. 20th Int. Conf. on Very Large Databases (VLDB 1994, Santiago de Chile)*,487-499 Morgan Kaufmann, San Mateo, CA, USA 1994

Authors Biography:

Kumar Charlie Paul, Principal of A.S.L Pauls College of Engineering & Technology. Had did many National and International Conferences and published many papers in journals. He also guided many students for their Ph.D project works. Having more than 23 years of experience in teaching field.

Immanual Gnandurai, Assistant Professor/ CSE of Dhaya college of Engineering. Had did many National and International Conferences and published many papers in journals. He also guided many students for their UG and PG project works.