

Methodology for Data Mining

Miss S. G. Anantwar

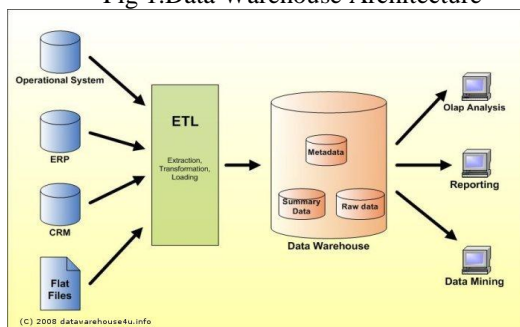
Prof. R.R. Shelke

Abstract: A data warehouse is a copy of transaction data specifically structured for query and analysis. Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from data warehouse in different perspectives and summarizing it into useful information. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases (data warehouse). Web mining is the use of data mining techniques to automatically discover and extract information from web documents and services. There are various algorithms used to extract the data from the data warehouse. The most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. In this paper we are dealing with the two best known algorithms like Apriori and K-mean Algorithm. This paper focuses on the review of comparison within these algorithms.

I. INTRODUCTION

A data warehouse is a copy of transaction data specifically structured for query and analysis. A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision-making process. A data warehouse is a centralized repository that stores data from multiple information sources and transforms them into a common, multi-dimensional data model for efficient querying and analysis.

Fig 1. Data Warehouse Architecture



A data warehouse is a:

- i) Subject-oriented
- ii) Integrated
- iii) Time varying
- iv) Non-volatile collection of data in support of the management's decision-making process.

Data Mining: Data mining is the process of finding correlations or patterns among dozens of fields in large

relational databases (data warehouse). While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining techniques are the result of a long process of research and product development. Data mining takes evolutionary process beyond retrospective data access and navigation to prospective and proactive information delivery. Data mining is ready for application in the business community because it is supported by three technologies that are now sufficiently mature:

- Massive data collection
- Powerful multiprocessor computers
- Data mining algorithms

II. AN ARCHITECTURE FOR DATA MINING

Many data mining tools currently operate outside of the warehouse, requiring extra steps for extracting, importing, and analyzing the data. Furthermore, when new insights require operational implementation, integration with the warehouse simplifies the application of results from data mining. The resulting analytic data warehouse can be applied to improve business processes throughout the organization, in areas such as promotional campaign management, fraud detection, new product rollout, and so on. Figure 2. illustrates an architecture for advanced analysis in a large data warehouse.

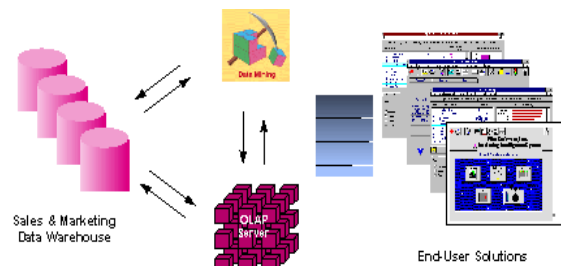


Figure 2. - Integrated Data Mining Architecture

The ideal starting point is a data warehouse containing a combination of internal data tracking all customer contact coupled with external market data about competitor activity. Background information on potential customers also provides an excellent basis for prospecting. This warehouse can be implemented in a variety of relational database systems: Sybase, Oracle, Redbrick, and so on, and should be optimized for flexible and fast data access. An OLAP (On-Line Analytical Processing) server enables a more sophisticated end-user business model to be applied when navigating the data warehouse.

Discovery in Database (KDD) in that knowledge discovery and analysis can be performed from many information and raw data in databases. Nowadays data

mining technique has many ways to implement and apply to discover knowledge from raw data; this is up to the types of data analysis and usage. The data mining processes used in this paper has many integrated data mining techniques including k-means clustering algorithms and Apriori algorithm.

III. THE K-MEANS ALGORITHM

The k-means algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters, k . A detailed history of k-means along with descriptions of several variations are given in [7] Gray and Neuhoﬀ [6] provide a nice historical background for k-means placed in the larger context of hill-climbing algorithms. The algorithm operates on a set of d -dimensional vectors, $D = \{\mathbf{x}_i \mid i = 1, \dots, N\}$, where $\mathbf{x}_i \in \mathbb{R}_d$ denotes the i th data point. The algorithm is initialized by picking k points in \mathbb{R}_d as the initial k cluster representatives or “centroids”. Techniques for selecting these initial seeds include sampling at random from the dataset, setting them as the solution of clustering a small subset of the data or perturbing the global mean of the data k times. Then the algorithm iterates between two steps till convergence:

Step 1: Data Assignment. Each data point is assigned to its *closest* centroid, with ties broken arbitrarily. This results in a partitioning of the data.

Step 2: Relocation of “means”. Each cluster representative is relocated to the center (mean) of all data points assigned to it. If the data points come with a probability measure (weights), then the relocation is to the expectations (weighted mean) of the data partitions. The algorithm converges when the assignments (and hence the \mathbf{c}_j values) no longer change. The algorithm execution is visually depicted in Fig. 3. Note that each iteration needs $N \times k$ comparisons, which determines the time complexity of one iteration. The number of iterations required for convergence varies and may depend on N , but as a first cut, this algorithm can be considered linear in the dataset size. One issue to resolve is how to quantify “closest” in the assignment step. The default measure of closeness is the Euclidean distance, in which case one can readily show that the non-negative cost function,

$$\sum_{i=1}^N (\operatorname{argmin}_j \|x_i - c_j\|_2^2)$$

Will decrease whenever there is a change in the assignment or the relocation steps, and hence convergence is guaranteed in a finite number of iterations. The greedy-descent nature of k-means on a non-convex cost also implies that the convergence is only to a local optimum, and indeed the algorithm is typically quite sensitive to the initial centroid locations. Figure 4 illustrates how a poorer result is obtained for the same dataset as in Fig. 4 for a different choice of the three initial centroids. The local minima problem can be countered to some extent by running the algorithm multiple times with different initial

centroids, or by doing limited local search about the converged solution.

Fig. 3 Changes in cluster representative locations (indicated by ‘+’ signs) and data assignments (indicated by

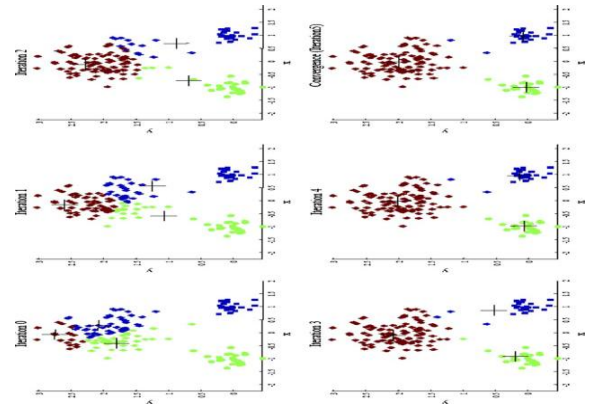
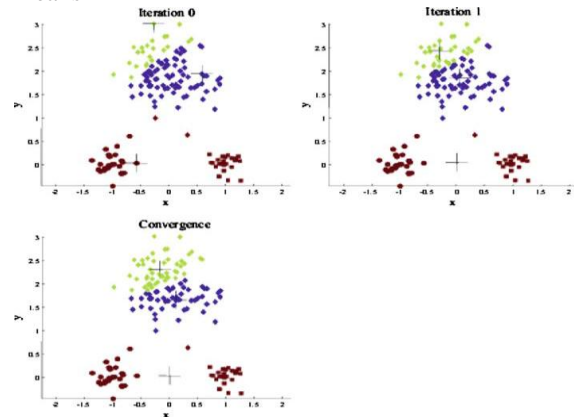


Fig.4 Effect of an inferior initialization on the k-means result means algorithm color) during an execution of the k-means



Limitations: 1. The k-means algorithm suffers from several other problems. Observe that k-means is a limiting case of fitting data by a mixture of k Gaussians with identical, isotropic covariance matrices ($\Sigma = \sigma^2 \mathbf{I}$). This problem may be alleviated by rescaling the data to “whiten” it before clustering, or by using a different distance measure that is more appropriate for the dataset.

2. This approach also makes the solution less sensitive to initialization, and since the hierarchical method provides results at multiple resolutions, one does not need to pre-specify k either. The cost of the optimal solution decreases with increasing k till it hits zero when the number of clusters equals the number of distinct data-points. This makes it more difficult to:

(a) Directly compare solutions with different numbers of clusters.

(b) To find the optimum value of k . If the desired k is not known in advance, one will typically run k-means with different values of k , and then use a suitable criterion to select one of the results.

Advantages: 1. The essential properties of k-means, including guaranteed convergence, linear separation boundaries and scalability, are retained [3].

2. This result makes k-means effective for a much larger class of datasets so long as an appropriate divergence is used.

3. k-means can be paired with another algorithm to describe non-convex clusters. One first clusters the data into a large number of groups using k-means. These groups are then agglomerated into larger clusters using single link hierarchical clustering, which can detect complex shapes.

One can progressively increase the number of clusters, in conjunction with a suitable stopping criterion. Bisecting k-means [3] achieves this by first putting all the data into a single cluster, and then recursively splitting the least compact cluster into two using 2-means. The celebrated LBG algorithm [3] used for vector quantization doubles the number of clusters till a suitable code-book size is obtained. Both these approaches thus alleviate the need to know k beforehand. The algorithm is also sensitive to the presence of outliers, since “mean” is not a robust statistic. A preprocessing step to remove outliers can be helpful. Post-processing the results, for example to eliminate small clusters, or to merge close clusters into a large cluster, is also desirable. Ball and Hall’s ISODATA algorithm effectively used both pre- and post-processing on k-means.

Generalizations and connections: As mentioned earlier, k-means is closely related to fitting a mixture of k isotropic Gaussians to the data. Moreover, the generalization of the distance measure to all Bregman divergences is related to fitting the data with a mixture of ki components from the exponential family of distributions. Another broad generalization is to view the “means” as probabilistic models instead of points in Rd . Here, in the assignment step, each data point is assigned to the most likely model to have generated it. In the “relocation” step, the model parameters are updated to best fit the assigned datasets. Such *model-based* k-means allow one to cater to more complex data, e.g. sequences described by Hidden Markov models. One can also “kernelize” k-means [4]. Though boundaries between clusters are still linear in the implicit high-dimensional space, they can become non-linear when projected back to the original space, thus allowing kernel k-means to deal with more complex clusters. Dhillon et al. [4] have shown a close connection between kernel k-means and spectral clustering. The *K-medoid* algorithm is similar to k-means except that the centroids have to belong to the data set being clustered. Fuzzy c-means is also similar, except that it computes fuzzy membership functions for each clusters rather than a hard one. Despite its drawbacks, k-means remains the most widely used partitioning clustering algorithm in practice. The algorithm is simple, easily understandable and reasonably scalable, and can be easily modified to deal with streaming data. To deal with very large datasets, substantial effort has also gone into further speeding up k-means, most notably by using kd-trees or

exploiting the triangular inequality to avoid comparing each data point with all the centroids during the assignment step. Continual improvements and generalizations of the basic algorithm have ensured its continued relevance and gradually increased its effectiveness as well

IV. THE APRIORI ALGORITHM

4.1 Description of the algorithm One of the most popular data mining approaches is to find frequent itemsets from a transaction dataset and derive association rules. Finding frequent itemsets (itemsets with frequency larger than or equal to a user specified minimum support) is not trivial because of its combinatorial explosion. Once frequent itemsets are obtained, it is straightforward to generate association rules with confidence larger than or equal to a user specified minimum confidence. Apriori is a seminal algorithm for finding frequent itemsets using candidate generation [1]. It is characterized as a level-wise complete search algorithm using anti-monotonicity of itemsets, “if an itemset is not frequent, any of its superset is never frequent”. By convention, Apriori assumes that items within a transaction or item set are sorted in lexicographic order. Let the set of frequent item sets of size k be Fk and their candidates be Ck . Apriori first scans the database and searches for frequent item sets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent itemsets.

1. Generate $Ck+1$, candidates of frequent itemsets of size $k+1$, from the frequent itemsets of size k .
2. Scan the database and calculate the support of each candidate of frequent itemsets.
3. Add those itemsets that satisfies the minimum support requirement to $Fk+1$.

The Apriori algorithm is shown in Fig. 3. Function apriori- in line 3 generates $Ck+1$ from Fk in the following two step process:

1. Join step: Generate $RK+1$, the initial candidates of frequent itemsets of size $k+1$ by taking the union of the two frequent itemsets of size k , Pk and Qk that have the first $k-1$ elements in common.

$$RK+1 = Pk \cup Qk = \{i_{tem1}, \dots, i_{temk-1}, i_{temk}, i_{temk_}\}$$

$$Pk = \{i_{tem1}, i_{tem2}, \dots, i_{temk-1}, i_{temk}\}$$

$$Qk = \{i_{tem1}, i_{tem2}, \dots, i_{temk-1}, i_{temk_}\}$$

$$\text{where, } i_{tem1} < i_{tem2} < \dots < i_{temk} < i_{temk_}.$$

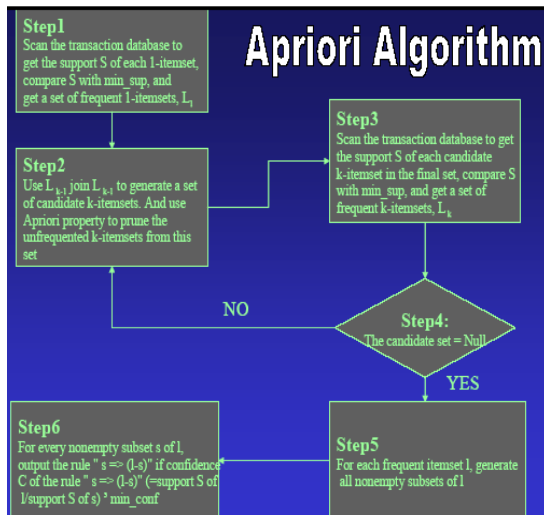
2. Prune step: Check if all the itemsets of size k in $Rk+1$ are frequent and generate $Ck+1$ by removing those that do not pass this requirement from $Rk+1$. This is because any subset of size k of $Ck+1$ that is not frequent cannot be a subset of a frequent itemset of size $k+1$. Function subset in line 5 finds all the candidates of the frequent itemsets included in transaction t . Apriori, then, calculates frequency only for those candidates generated this way by scanning the database. It is evident that Apriori scans the database at most $k_{max}+1$ times when the maximum size of

frequent itemsets is set at k_{max} . The Apriori achieves good performance by reducing the size of candidate sets (Fig. 3). However, in situations with very many frequent itemsets, large itemsets, or very low minimum support, it still suffers from the cost of generating a huge number of candidate sets and scanning the database repeatedly to check a large set of candidate itemsets. In fact, it is necessary to generate 2100 candidate itemsets to obtain frequent itemsets of size 100.

Apriori algorithm in pseudocode

```

L1 = {frequent items};
for (k = 2; Lk-1 != ∅; k++) do begin
    Ck = candidates generated from Lk-1 (that is:
    cartesian product Lk-1 × Lk-1 and eliminating any
    k-1 size itemset that is not frequent);
    for each transaction t in database do
        increment the count of all candidates in
        Ck that are contained in t
    Lk = candidates in Ck with min_sup
end
return ∪k Lk;
    
```



The impact of the algorithm : Many of the pattern finding algorithms such as decision tree, classification rules and clustering techniques that are frequently used in data mining have been developed in machine learning research community. Frequent pattern and association rule mining is one of the few exceptions to this tradition. The introduction of this technique boosted data mining research and its impact is tremendous. The algorithm is quite simple and easy to implement. Experimenting with Apriori-like algorithm is the first thing that data miners try to do.

Current and further research: Since Apriori algorithm was first introduced and as experience was accumulated, there have been many attempts to devise more efficient algorithms of frequent itemset mining. Many of them share the same idea with Apriori in that they generate

candidates. These include hash-based technique, partitioning, sampling and using vertical data format. Hash-based technique can reduce the size of candidate itemsets. Each itemset is hashed into a corresponding bucket by using an appropriate hash function. Since a bucket can contain different itemsets, if its count is less than a minimum support, these itemsets in the bucket can be removed from the candidate sets. A partitioning can be used to divide the entire mining problem into n smaller problems. The dataset is divided into n non-overlapping partitions such that each partition fits into main memory and each partition is mined separately. Since any itemset that is potentially frequent with respect to the entire dataset must occur as a frequent itemset in at least one of the partitions, all the frequent itemsets found this way are candidates, which can be checked by accessing the entire dataset only once. Sampling is simply to mine a random sampled small subset of the entire data. Since there is no guarantee that we can find all the frequent itemsets, normal practice is to use a lower support threshold. Trade off has to be made between accuracy and efficiency. Apriori uses a horizontal data format, i.e. frequent itemsets are associated with each transaction. Using vertical data format is to use a different format in which transaction IDs (TIDs) are associated with each itemset. With this format, mining can be performed by taking the intersection of TIDs. The support count is simply the length of the TID set for the itemset. There is no need to scan the database because TID set carries the complete information required for computing support.

The most outstanding improvement over Apriori would be a method called FP-growth (frequent pattern growth) that succeeded in eliminating candidate generation [3]. It adopts a divide and conquer strategy by

(1) Compressing the database representing frequent items into a structure called FP-tree (frequent pattern tree) that retains all the essential information and

(2) Dividing the compressed database into a set of conditional databases, each associated with one frequent itemset and mining each one separately. It scans the database only twice. In the first scan, all the frequent items and their support counts (frequencies) are derived and they are sorted in the order of descending support count in each transaction. In the second scan, items in each transaction are merged into a prefix tree and items (nodes) that appear in common in different transactions are counted. Each node is associated with an item and its count. Nodes with the same label are linked by a pointer called node-link. Since items are sorted in the descending order of frequency, nodes closer to the root of the prefix tree are shared by more transactions, thus resulting in a very compact representation that stores all the necessary information. Pattern growth algorithm works on FP-tree by choosing an item in the order of increasing frequency and extracting frequent itemsets that contain the chosen item by recursively calling itself on the conditional FP-tree. FP-growth is an order of magnitude faster than the original Apriori algorithm. There are several other

dimensions regarding the extensions of frequent pattern mining. The major ones include the followings:

- (1) Incorporating taxonomy in items [2]: Use of taxonomy makes it possible to extract frequent itemsets that are expressed by higher concepts even when use of the base level concepts produces only in frequent itemsets.
- (2) Incremental mining: In this setting, it is assumed that the database is not stationary and a new instance of transaction keeps added. The algorithm in [2] updates the frequent itemsets without restarting from scratch.
- (3) Using numeric valuable for item: When the item corresponds to a continuous numeric value, current frequent itemset mining algorithm is not applicable unless the values are discretized. A method of subspace clustering can be used to obtain an optimal value interval for each item in each itemset [3].
- (4) Using other measures than frequency, such as information gain or χ^2 value: These measures are useful in finding discriminative patterns but unfortunately do not satisfy anti-monotonicity property. However, these measures have a nice property of being convex with respect to their arguments and it is possible to estimate their upper bound for supersets of a pattern and thus prune unpromising patterns efficiently. Apriori SMP uses this principle [5].
- (5) Using richer expressions than itemset: Many algorithms have been proposed for sequences, tree and graphs to enable mining from more complex data structure [4].
- (6) Closed itemsets: A frequent itemset is closed if it is not included in any other frequent itemsets. Thus, once the closed itemsets are found, all the frequent itemsets can be derived from them. LCM is the most efficient algorithm to find the closed itemsets [3]

Use of Apriori algorithm

- Initial information: transactional database D and user-defined numeric minimum support threshold min_sup
- Algorithm uses knowledge from previous iteration phase to produce frequent itemsets
- This is reflected in the Latin origin of the name that means "from what comes before"

Limitations of Apriori algorithm

- Needs several iterations of the data
- Uses a uniform minimum support threshold
- Difficulties to find rarely occurring events
- Alternative methods (other than apriori) can address this by using a non-uniform minimum support threshold

Some competing alternative approaches focus on partition and sampling

CONCLUSION


The preprocessing added the association rules obtained from Apriori-based algorithm in feature selection to get better feature set. In the clustering process we used random data to generate initial centroids that works better for data sets. The experimental results show that the Pre-KMA model helped data mining to process more accurate than traditional method with decrease SSE value. It is even more efficient with concurrent processing with the decrease in processing time and loop of clustering are decreased too.

This research has some improvable points in that feature selection technique proposed in this paper is appropriate for the selected data. For other data sets, the Pre-KMA model needs more experimentation and efficiency confirmation. The concurrent technique can also be improved for a better parallelization performance.

REFERENCES

- [1] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th VLDB conference, pp 487–499
- [2] Ahmed S, Coenen F, Leng PH (2006) Tree-based partitioning of data for association rule mining. *Knowl Inf Syst* 10(3):315–331
- [3] Banerjee A, Merugu S, Dhillon I, Ghosh J (2005) Clustering with Bregman divergences. *J Mach Learn Res* 6:1705–1749
- [4] Bezdek JC, Chuah SK, Leep D (1986) Generalized k-nearest neighbor rules. *Fuzzy Sets Syst* 18(3):237–256. [http://dx.doi.org/10.1016/0165-0114\(86\)90004-7](http://dx.doi.org/10.1016/0165-0114(86)90004-7)
- [5] Bloch DA, Olshen RA, Walker MG (2002) Risk estimation for classification trees. *J Comput Graph Stat* 11:263–288
- [6] Hastie T, Tibshirani R (1996) Discriminant adaptive nearest neighbor classification. *IEEE Trans Pattern Anal Mach Intell* 18(6):607–616
- [7] Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice-Hall, Englewood Cliffs

AUTHOR'S PROFILE

	<p>Miss Swati G. Anantwar Student of M.E. (CSE) H.V.P.M.C.O.E.T Amravati</p>
--	---

Passport Size Latest Color Photo	Prof.R.R.Shelke Asst. Professor ,M.E.(CSE)
----------------------------------	---