ISSN:0975-9646



# A Survey on Data Mining Algorithms and Future Perspective

N.K. Sharma
*Assistant Professor*
*Ujjain Engineering College*
*Ujjain, M.P., India*

Dr. R.C. Jain
*Director*
*S.A.T.I. Engineering College*
*Vidisha, M.P., India*

Manoj Yadav
*Software Consultant*
*P.I.S.T.*
*Bhopal, M.P., India*

**Abstract -Data mining is a powerful and new method of analyzing data and finding out new patterns from large set of data. The objective of data mining is to pull out knowledge from a data set in an understandable format. Data mining is the process of collecting, extracting and analyzing large data set from different perspectives.**

**There is an enormous amount of data stored in databases and data warehouse due to enormous technological advancements in computing and Internet. It is therefore, required, to develop powerful tools for analysis of such huge data and mining valuable information out of it. One of the main challenges in database mining is developing fast and efficient algorithms that can handle large volumes of data as most of the mining algorithms perform computation over the entire databases, often very large. Data mining is a convenient way of extracting patterns, which represents knowledge implicitly stored in large data sets and focuses on issues relating to their feasibility, usefulness, effectiveness and scalability. It can be viewed as an essential step in the process of knowledge discovery. Data are normally preprocessed through data cleaning, data integration, data selection, and data transformation and prepared for the mining task. Data mining can be performed on various types of databases and information repositories, but the kind of patterns to be found are specified by various data mining functionalities like class description, association, correlation analysis, classification, prediction, cluster analysis etc. This paper gives an overview of the existing data mining algorithms required for the same.**

**Keywords: Data mining, KDD, Clustering, Association Rule, Classification, Sequential and parallel Algorithms**

## 1. INTRODUCTION

The literature available for data mining contains many definitions [1][2][5][9]. Some of them depend on the application and how data has been organized into a database whereas some of them depend on the discovery of new information from the facts in a database. Data mining is a process by which one can extract interesting and valuable information from large data using efficient techniques. Data Clustering, Data Classification, Detection of Outliers and Association Rule Mining are useful basic data mining techniques depending upon the type of information sought from databases. Basic data mining techniques are data clustering, data classification, detection of outliers (or deviations) and association rule discovery [1][3].

1.1 Data Clustering: It is the process of grouping or partitioning a set of data into different groups or clusters based on the principle that the similarity between the data points in one cluster is maximized and the similarity between data points in different clusters is minimized. For example, grouping of companies with similar stock behavior or similar growth to identify genes and proteins have similar functions.

1.2 Data Classification: It is the process to classify a set of data based on their values. For example, A car dealer has to classify its customers according to their preference for cars so that catalogs of new models can be mailed directly to the customers to maximize business opportunity.

1.3 Detection of outliers (or Deviations): Finds data points that differ significantly from the majority of the data points in a given data set as needed in Medical diagnosis and credit card detection.

1.4 Association Rule Discovery: It finds all rules that correlate the presence of one set of items with that of another set of items. For example, one may find, from a large set of transaction data, such an association rule as if a customer buys (one brand of ) milk, he/ she usually buys ( another brand of ) bread in the same transaction.

The objective of this paper is to provide various data mining algorithms with reference to clustering, classification and association rules. In following section a survey of clustering algorithms is illustrated. Section 3, consist various association algorithms. In section 4, existing classification algorithms have been discussed. In section 5 various algorithms we expound have been collated, while in section 6 we discussed development of new algorithms to overcome the various existing issues.

## 2. CLUSTERING ALGORITHMS

Clustering algorithms can be categorized based on their cluster model. As there are probably a few dozen published clustering algorithms. Not all provide models for their clusters and can thus not easily be categorized [12][13][14].

### *2.1 Connectivity based clustering (Hierarchical clustering)*

Connectivity based clustering, also known as hierarchical clustering, is based on the core idea of objects being more related to nearby objects than to objects farther away. As such, these algorithms connect "objects" to form "clusters" based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster. At different distances, different clusters will form, which can be represented using a dendrogram, which explains where the common name "hierarchical clustering" comes from. These algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances. In a dendrogram, the y-axis marks the distance at

which the clusters merge, while the objects are placed along the x-axis such that the clusters don't mix.

Apart from the usual choice of distance functions, the user also needs to decide on the linkage criterion (since a cluster consists of multiple object, there are multiple candidates to compute the distance to) to use. Popular choices are known as single-linkage clustering (the minimum of object distances), complete linkage clustering (the maximum of object distances) or UPGMA ("Unweighted Pair Group Method with Arithmetic Mean", also known as average linkage clustering). Furthermore, hierarchical clustering can be computed agglomerative (starting with single elements and aggregating them into clusters) or divisive (starting with the complete data set and dividing it into partitions).

While these methods are fairly easy to understand, the results are not always easy to use, as they will not produce a unique partitioning of the data set, but a hierarchy the user still needs to choose appropriate clusters from. The methods are not very robust towards outliers, which will either show up as additional clusters or even cause other clusters to merge (known as "chaining phenomenon", in particular with single-linkage clustering). In the data mining community these methods are recognized as a theoretical foundation of cluster analysis, but often considered obsolete. They did however provide inspiration for many later methods such as density based clustering.

### *2.2 Centroid-based Clustering*

In centroid-based clustering, clusters are represented by a central vector, which must not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

The optimization problem itself is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximate method is often actually referred to as "k-means algorithms". It does however only find a local optimum, and is commonly run multiple times with different random initializations. Variations of k-means often include such optimizations as choosing the best of multiple runs, but also restricting the centroids to members of the data set (k-medoids), choosing medians (k-medians clustering), choosing the initial centers less randomly (K-means++) or allowing a fuzzy cluster assignment (Fuzzy c-means). Most k-means-type algorithms require the number of clusters - k - to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. K-means has a number of interesting theoretical properties. On one hand, it partitions the data space into a structure known as Voronoi diagram. On the other hand, it is conceptually close to nearest neighbor classification and as such popular in machine learning. Third, it can be seen as a variation of model based classification, and Lloyd's algorithm as a variation of the Expectation-maximization algorithm.

### *2.3 K-Means Clustering Algorithm*

This nonhierarchical method initially takes the number of components of the population equal to the final required number of clusters. In this step itself the final required number of clusters is chosen such that the points are mutually farthest apart. Next, it examines each component in the population and assigns it to one of the clusters depending on the minimum distance. The centroid's position is recalculated every time a component is added to the cluster and this continues until all the components are grouped into the final required number of clusters.

The k-means algorithm works as follows:

a) Randomly select k data object from dataset D as initial cluster centers.

b) Repeat

i. Calculate the distance between each data object $d_i(1 \le i \le n)$ and all k cluster centers $c_j(1 \le j \le n)$ and assign data object $d_i$ to the nearest cluster.

ii. For each cluster $j(1 \le j \le k)$, recalculate the cluster center.

iii. Until no changing in the center of clusters. The most widely used convergence criteria for the k-means algorithm is minimizing the SSE.

### *2.4 Distribution-based Clustering*

The clustering model most closely related to statistics is based on distribution models. Clusters can then easily be defined as objects belonging most likely to the same distribution. A nice property of this approach is that this closely resembles the way artificial data sets are generated: by sampling random objects from a distribution. While the theoretical foundation of these methods is excellent, they suffer from one key problem known as overfitting, unless constraints are put on the model complexity. A more complex model will usually always be able to explain the data better, which makes choosing the appropriate model complexity inherently difficult. The most prominent method is known as expectation-maximization algorithm. Here, the data set is usually modeled with a fixed (to avoid overfitting) number of Gaussian distributions that are initialized randomly and whose parameters are iteratively optimized to fit better to the data set. This will converge to a local optimum, so multiple runs may produce different results.

Distribution-based clustering is a semantically strong method, as it not only provides you with clusters, but also produces complex models for the clusters that can also capture correlation and dependence of attributes. However, using these algorithms puts an extra burden on the user: to choose appropriate data models to optimize, and for many real data sets, there may be no mathematical model available the algorithm is able to optimize (e.g. assuming Gaussian distributions is a rather strong assumption on the data).

### *2.5 Density-based Clustering*

In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points.The most popular density based clustering method is DBSCAN.

In contrast to many newer methods, it features a well-defined cluster model called "density-reach ability". Similar to linkage based clustering; it is based on connecting points within certain distance thresholds. However, it only connects points that satisfy a density criterion, in the original variant defined as a minimum number of other objects within this radius. A cluster consists of all density-connected objects (which can form a cluster of an arbitrary shape, in contrast to many other methods) plus all objects that are within these objects range. Another interesting property of DBSCAN is that its complexity is fairly low - it requires a linear number of range queries on the database - and that it will discover essentially the same results (it is deterministic for core and noise points, but not for border points) in each run, therefore there is no need to run it multiple times.
The key drawback of DBSCAN is that it expects some kind of density drop to detect cluster borders.

***2.6 Graph-based approaches***

To address the problem of clustering, k nearest neighbors (kNNs) are used to identify k most similar points around each point and by way of conditional merging, clusters are generated. There are various variants of kNN clustering and they differ at the conditional merging part of the solution. For a given point p, its kNNs are found out. If the distance between p and any of the points in kNN(p) set (say q) is less than $\epsilon$, then point q is merged into the cluster of p. This algorithm requires tuning of $\epsilon$ and k values to get clusters. In a method proposed, to construct strongly connected components from a given directed graph where each edge is associated with a weight (usually distance between points).
This method is highly sensitive to the presence of noise and cannot handle clusters of different densities.

## 3. ASSOCIATION RULE MINING ALGORITHMS

Given a set of transactions, where each literal (called items), association rules is an expression of the form X Y, where X and Y are set of items[1,3,6,8].
There are two important measures for association rules, support (s) and confidence ($\alpha$), can be defined as follows:
**Support** is the ratio (in percent) of the records that contain $X \cup Y$ to the total number of records in the database.
**Confidence** is the ratio (in percent) of the number of records that contain $X \cup Y$ to the number of records that contain X.
For example, let us consider customer's purchase data shown in Table 1.

Table 1: Consumer purchased Data

| Trans ID (TID) | Item |
|---|---|
| 1 | egg, bread, milk |
| 2 | egg, milk |
| 3 | egg |
| 4 | egg, bread, milk |
| 5 | cheese |

For example, if you were a owner of the supermarket, you would like to think of the layout of the store. In that case, the rules in Table 2 can be useful.

Table 2: Association Rules

| Rule | Support | Confidence |
|---|---|---|
| egg, bread => milk | 40% | 100% |
| egg => bread | 40% | 50% |

The problem of mining association rules can be decomposed into two sub problems:
1. Find all sets of items(itemset (IS)) whose support is greater than the user-specified minimum support(MS):large item set or frequent Item sets (FIS).
For example, if MS is 40% then {egg, bread}(40%),{egg, bread, milk}(40%).
2. Use the frequent item set to generate the desired rules
If ABCD, AB is FISs, then AB -> CD : conf = support(ABCD)/support(AB) and
if conf >= minimum confidence(MC), then the rule holds

For example, if MC is 75%, then {egg, bread} {milk}(100%) holds.

Notation used in this paper is summarized below
k-item set---An item set having k items
$L_k$ --- Set of frequent k-item sets(those with minimum support). Each member of this set has two fields : i) item set and ii) support count.
$C_k$ --- Set of candidate k-item sets (potentially frequent item sets). Each member of this set has two fields : i) item set and ii) support count.

3. Algorithms for discovering large itemsets make multiple passes over the data. In the first pass, we count the support of individual items and determine which of them are large, i.e. have minimum support. In each subsequent pass, we start with a seed set of itemsets found to be large in the previous pass. We use this seed set for generating new potentially large itemsets, called candidate itemsets, and count the actual support for these candidate itemsets during the pass over the data. At the end of the pass, we determine which of the candidate itemsets are actually large, and they become the seed for the next pass. This process continues until no new large itemsets are found.

*3.1 AIS Algorithm*

The AIS algorithm [3] was the first published algorithm developed to generate all large itemsets in a transaction database. It focused on the enhancement of databases with necessary functionality to process decision support queries. This technique is limited to only one item in the consequent. That is, association rules are in the form of $X \Rightarrow I_k \mid c$, where X is a set of items and $I_k$ is a single item in the domain I, and c is the confidence of the rule. The AIS algorithm makes multiple passes over the entire database. During each pass, it scans all transactions. In the first pass, it counts support of individual items and determines which of them are large or frequent in the database. Large itemsets of each pass are extended to generate candidate itemsets. After scanning a transaction, the common itemsets between large itemsets of previous pass and items of this transaction are determined. These common itemsets are extended with other items in the transaction to generate new candidate itemsets. The

estimation tool and pruning techniques determine candidate sets by omitting unnecessary itemsets from the candidate sets. Then, the support of each candidate set is computed. Candidate sets having supports greater than or equal to minimum support are chosen as large itemsets. These large itemsets are extended to generate candidate sets for the next pass. The process terminates when no more large itemsets are found. The main disadvantage of this algorithm is that it results in unnecessarily generating and counting too many candidate itemsets that turn out to be small.

*3.2 SETM Algorithm*

This algorithm [3] used SQL to calculate large itemsets. In this each member of the set large itemsets, $L_k$, is in the form <TID, itemset> where TID is the unique identifier of a transaction. Candidate itemsets are generated on the fly as the database is scanned, but counted at the end of the pass.

1. New candidate itemsets are generated the same way as in AIS algorithm, but the TID of the generating transaction is saved with the candidate itemset in a sequential structure.
2. At the end of the pass, the support count of candidate itemsets is determined by aggregating this sequential structure Several passes are made on the database. When no more itemsets are found, the algorithm terminates.

The main disadvantage of this algorithm is due to the number of candidate sets $C_k$. Since for each candidate itemset there is a TID associated with it, it requires more space to store a large number of TIDs but sorting is needed on itemsets. Another disadvantage is that for each candidate itemset, there are as many entries as its support value.

*3.3Apriori Algorithm*

The Apriori algorithm [5],[6] is a great achievement in the history of mining association rules. It is by far the most well-known association rule algorithm. It uses Apriori Property that states that any subset of large itemset must also be large. It differs from AIS & SETM in the manner candidate itemsets are generated. The Apriori generates the candidate itemsets by joining the large itemsets of the previous pass and deleting those subsets that are small in the previous pass without considering the transactions in the database. By only considering large itemsets of the previous pass, the number of candidate large itemsets is significantly reduced. The large itemset of the previous pass is joined with itself to generate all itemsets whose size is higher by 1. Each generated itemset, that has a subset, which is not large, is deleted. The remaining itemsets are the candidate ones. The only drawback of this algorithm is that counting support of candidate itemsets is a time consuming process since it requires scanning the entire database.

*3.4 AprioriTid Algorithm*

Similar to Apriori , AprioriTid [5 ],[6] alorithm uses the Apriori candidate generating function to determine candidate sets before beginning of a pass. The main difference from Apriori is that the database is not used at all for counting the support of candidate itemsets after the first pass. Another set C' is generated of which each member has the TID of each transaction and the large itemsets present in this transaction. This set is used to count the support of each candidate itemset.

Steps involved:

- The entire database is scanned and $C_1$' is obtained in terms of itemsets.
- Large itemsets with 1-item $L_1$ are calculated by counting entries of $C_1$'.
- Apriori is the used to obtain $C_2$.
- Entries of $C_2$' corresponding to a transaction is obtained by considering members of $C_2$ which are present in T
- $L_2$ is then obtained by counting support in $C_2$'
- This process continues until the candidate itemsets are found to be empty.

The advantage of using this encoding function is that in later passes the number of entries in C' may be smaller than the number of transactions in the database. The disadvantage is that during initial passes candidate itemsets generated are very large equivalent to size of database.

*3.5 AprioriHybrid Algorithm*

This algorithm [6] is based on the idea that it is not necessary to use the same algorithm in all the passes over data. Apriori does better in the earlier passes. AprioriTid does better than Apriori in the later passes. Hence, a hybrid algorithm can be designed that uses Apriori in the initial passes and switches to AprioriTid when it expects that the set C' will fit in memory. Therefore, an estimation of C' at the end of each pass is necessary. Also, there is cost involvement of switching from Apriori to AprioriTid**.**

*3.6 Partition Algorithm*

This algorithm[7] is fundamentally different from the previous algorithms in that it scans the database at most two times to generate all significant association rules. Contrast this with the previous algorithms, where the database is not only scanned multiple times but the number of scans cannot even be determined in advance. Surprisingly, the savings in I/O is not achieved at the cost of increased CPU overhead. The reason the database needs to be scanned multiple number of times is because the number of possible itemsets to be tested for support is exponentially large if it must be done in a single scan of database. However, suppose we are given a small set of potentially large itemsets, say a few thousand itemsets. Then the support for them can be tested in one scan of database and the actual large itemsets can be discovered. This approach will work only if the given set contains all actual large itemsets. Partition algorithm accomplices this in two scans of the database. In one scan it generates a set of all potentially large itemsets by scanning the database once. The set is a superset of all large itemsets, i.e. it may contain false positives. But no false negatives are reported. During the second scan counters for each of these itemsets are set up and their actual support are measured in one scan of the database.

*3.7 SEAR Algorithm*

This algorithm is abbreviated as Sequential Efficient Association Rule mining. This algorithm [8] [10] was developed to see the effect of different data representation. It is basically called modified Apriori. Like Apriori, each

pass of SEAR consists of a candidate generation phase followed by a counting phase. However, it uses the prefix data structure for itemsets, which has been developed to improve the data structure used by Apriori. Apriori stores candidates in tree-like data structure. Candidate sets are stored in leaf nodes, each of which accommodates several candidates. The purpose of internal nodes is to direct the search for a candidate to the proper leaf. In contrast, SEAR employs a prefix-tree data structure, in which nodes do not contain sets, but only information about sets. Prefix trees store both frequent sets and candidate sets in the same tree. Infrequent sets are either not created in the first place or are deleted immediately. It also makes use of pass bundling. In this, several levels of the tree are expanded before counting databases. It is implemented by setting a lower limit to the number of candidates that have to be created before a counting step is allowed.

*3.8 SPEAR Algorithm*

This algorithm [8] [10] is partitioned version of SEAR and non-TID list version of Partition [7]. It uses the horizontal data format, but makes two scans over database. First it gathers potentially frequent itemsets, and then it obtains their global support. Each partition is loaded into memory and processed completely using SEAR. The advantage of using item-lists is that the data size doesn't grow during course of algorithm so no buffer management is required. The counting phase of SPEAR is a single pass of SEAR during which all the active sets in the tree are considered as candidates.

*3.9 SPINC Algorithm*

SPINC [8] is a version of SPEAR that uses an incremental partitioning technique. This incremental method does not process partitions independently like SPEAR, but uses the partial results from preceding partitions. The basic principle is to count a set in every partition following the one in which it was found frequent. It produces immediate savings in that a set will not be counted twice in the partitions in which it was frequent. This directly reduces the amount of computation. Furthermore, the last partition does not have to be counted al all in phase II, which saves the IO operations and allows the algorithm to use less that two passes over the data.

*3.10 DHP Algorithm*

DHP [9] [10] stands for Direct Hashing and Pruning. It has two main features: one is efficient generation for large itemsets and the other is effective reduction on transaction database size. It utilizes a hash technique for efficient generation of candidate large itemsets, in particular for large 2-itemsets. In addition, it employs effective pruning techniques to progressively reduce the transaction database size. DHP uses the technique of hashing to filter out unnecessary itemsets for next candidate itemset generation. DHP reduces database size progressively by not only trimming each individual transaction size but also pruning number of transactions in the database. DHP achieves a shorter execution time than Apriori even in later iterations. Parallel Algorithms provide scalability to massive data sets and improving response time. Its execution time depends not only on input size but also on the architecture of the parallel Computer and the no. of processors. All the algorithms proposed for parallel in shared nothing architecture can also be implemented for distributed architecture. Distributed architecture is same as parallel shared nothing architecture with a slow communication network. Therefore, the emphasis in distributed systems is on reducing the communication overhead.

*3.11 Count Distribution (CD)*

It is a simple parallelization of Apriori algorithm. All processors generate the entire candidate set from $L_{k-1}$. Each processor can thus independently get partial supports of the candidates from its local database partition and sum up to get global counts by exchanging local counts with other processors [21],[22]. Once global frequent sets have been determined, next candidate item sets can be determined in parallel at all processors. The focus is on minimizing communication. It does so even at the expense of carrying redundant computations in parallel. Aggregate memory of the system is not exploited effectively.

*3.12 Data Distribution (DD)*

It is designed to exploit better the total system's memory as the number of processor is increased. It uses the total system memory by generating disjoint candidate sets on each processor [21],[22]. However, to generate the global support, each processor must scan the entire database (its local partition and all remote partitions) in all iterations.It is designed to exploit better the total system's memory as the number of processor is increased but Contention is a major problem and Processors may remain idle at the time of communication.

*3.13 Candidate Distribution*

This algorithm partitions the candidates during iteration l, so that each processor can generate disjoint candidates independent of other processors [21],[22]. The partitioning uses heuristics based on support, so that each processor gets an equal amount of work. The choice of the redistribution pass involves a tradeoff between decoupling processor dependence as soon as possible and waiting until sufficient load balance can be achieved.

No local data send, only global values exchanged and data is received asynchronously and the processors do not wait for the complete pruning information to arrive from all the processors but repartitioning is expensive.

*3.14 Fast Distributed*

It is built on count distribution and proposes new techniques to reduce the number of candidates considered for counting. In this way, it minimizes communication. Each site generates candidates using the Global frequents from all the sites and assigns a home site for each candidate. Then, each site computes the local support for all candidates. Next comes the *local pruning* step, Any item-set X that is not locally frequent at the current site, is removed. This is because if X is globally frequent then it must be frequent at some other site.The next step is *count-polling* optimization. Each home site requests, for all candidates assigned to it, local count from all other sites and computes their global support. The home site then broadcasts the global support to all other sites.The algorithm is specifically designed for distributed environment and unlike Count Distribution where local counts of all the candidates are broadcast to everyone else,

it sends to only one home site per candidate but algorithms require multiple database scans. Number of scans is equal to the size largest frequent itemset. All the processors have to communicate the results of every database scan and wait for the results from other nodes to proceed for the next level. Hence, the number of synchronizations is equal to the size largest frequent item set.

## 4. CLASSIFICATION ALGORITHMS

### *4.1 FP-Growth Algorithm*

FP stands for Frequent Patterns. This algorithm [11] makes use of an FP-tree structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns. FP-Growth is an efficient method for mining complete set of frequent patterns by pattern fragment growth. Efficiency of mining is achieved with three techniques. First, a large database is compressed into a highly condensed, much smaller data structure, which avoids costly repeated database scans. Second, FP tree-based mining adopts a pattern fragment growth method to avoid the costly generation of a large number of candidate sets. And third, a partitioning based, divide and conquer method is used to decompose mining task into a set of smaller tasks for mining confined patterns in conditional databases, which dramatically reduces the search space. FP-Growth is efficient for mining both long and short frequent patterns.

It avoids costly candidate generation and it has better performance and efficiency than Apriori like algorithms but it takes two complete scans of database and it uses a recursive routine to mine patterns from a conditional pattern base.

### *4.2 P-Tree Algorithm*

A Pattern Tree [17], unlike FP Tree, which contains the frequent items only, contains all the items that appear in the original database. We can obtain a P-tree through one scan of database and get the corresponding FP-tree from the P-tree later. An FP tree is a sub-tree of the P-tree with a specified support threshold, which contains those frequent items that meet this threshold and hereby excludes infrequent items. We do this by checking the frequency of each node along the path from root to leaves. It uses the same mining process as used by FP-Growth algorithm [11]. It scans the original database only once and in case support threshold changes, we need not re-scan the database but it uses a recursive mining process.

### *4.3 Inverted Matrix Algorithm*

This association rule-mining algorithm is based on the conditional pattern concept [11]. The algorithm [19][20] is divided into two main phases. The first one, considered pre-processing, requires two full I/O scans of the dataset and generates a data structure called Inverted Matrix. In the second phase, the Inverted Matrix is mined using different support levels to generate association rules. The mining process might take in some cases less than one-full I/O scan of the data structure in which only frequent items based on the support given by the user are scanned and participate in generating the frequent patterns. The Inverted Matrix layout combines the horizontal and vertical layouts with the purpose of making use of the best of the two approaches and reducing their drawbacks as much as possible. The idea of this approach is to associate each item with all transactions in which it occurs (i.e. an inverted index), and to associate each transaction with all its items using pointers.For computing frequencies, it relies first on reading sub-transactions for frequent items directly from the Inverted Matrix [19]. Then it builds independent relatively small trees for each frequent item in the transactional database. Each such tree is mined separately as soon as they are built, with minimizing the candidacy generation and without building conditional sub-trees recursively.

It uses a simple and non-recursive association rule mining process and the inverted matrix can be made disk resident, so it performs well for large datasets but it makes two scans of original database and the complexity of developing inverted matrix is a bit high.

## 5. COMPARISON OF ALGORITHMS

| Algorithm | Scan | Data structure | Comments |
|---|---|---|---|
| Apriori | m+1 | Lk-1 : Hash table | Transaction database with Ck: Hash tree moderate cardinality; Base algorithm for parallel algorithms |
| Apriori-TID | m+1 | Lk-1 : Hash table | Very slow with larger number $C_k$: array indexed by TID of Ck |
| Partition of Data | 2 | | Hash Table Suitable for large DB with high cardinality of data; Favors homogenous data distribution. |
| CD | m+1 | Hash table and tree | Data Parallelism. |
| DD | m+1 | Hash table and tree | Task Parallelism; round- robin partition |
| IDD | m+1 | Hash table and tree | Task Parallelism; partition by the first items |
| HD | m+1 | Hash table and tree | Hybrid data and task parallelism; grid parallel architecture |

## 6. LIMITATIONS & FUTURE SCOPE

Database needs to be scanned number of times, because the number of possible itemsets to be tested for support is exponentially large, if it must be done on a single scan of database. There are several advances made in high performance algorithms but still there are several areas where immediate attention is required mentioned below [7][10][15][16][18][23]

I. Larger databases: Databases with hundreds of fields and tables, millions of records, and multi- gigabyte size are quite commonplace, and terabyte databases are beginning to appear.
II. High dimensionality: Not only is there often a very large number of records in the database, but there can also be a very large number of fields (attributes, variables) so that the dimensionality of the problem is high. A high dimensional data set creates problems in terms of increasing the size of the search space for model induction in a combinatorial explosive manner. In addition, it increases the chances that a data mining algorithm will find spurious patterns that are not valid in general.
III. Over fitting: When the algorithm searches for the best parameters for one particular model using a limited set of data, it may model not only the general patterns in the data but also any noise specific to that data set, resulting in poor performance of the model on test data.
IV. Assessing statistical significance: A problem (related to over fitting) occurs when the system is searching over many possible models. For example, if a system tests N models at the 0.001 significance level, then on average, with purely random data, N=1000 of these models will be accepted as significant. This point is frequently missed by many initial attempts at KDD.
V. Changing data and knowledge: Rapidly changing data may make previously discovered patterns invalid. In addition, the variables measured in a given application database may be modified, deleted, or augmented with new measurements over time.
VI. Complex relationships between fields: Hierarchically structured attributes or values, relations between attributes, and more sophisticated means for representing knowledge about the contents of a database will require algorithms that can effectively utilize such information.
VII. Understandability of patterns: In many applications it is important to make the discoveries more understandable by humans.
VIII. Rule refinement strategies can be used to address a related problem: the discovered knowledge may be implicitly or explicitly redundant.
IX. User interaction and prior knowledge: Many current KDD methods and tools are not truly inter-active and cannot easily incorporate prior knowledge about a problem except in simple ways. The use of domain knowledge is important in all of the steps of the KDD process.
X. Integration with other systems: A stand-alone discovery system may not be very useful. Typical integration issues include integration with a DBMS (e.g. via a query interface), integration with spreadsheets and visualization tools, and accommodating real time sensor readings.

**Possible future scopes to overcome above illustrated limitations are expounded below:**

I. Large scale data sets are usually logically and physically distributed. Organizations need a decentralized approach if they are geographically distributed. Modern organizations are emphasizing on distributed nature of the data rather the size of the data to be mined. Presently most current work concerns only horizontal partition (were different sites have different transaction), one has to work on vertically partitioned (where different sites have different item).
II. Most of the current algorithms are iterative and scan data many times, hence they are not scalable. Similarly in large databases, the candidates will certainly not fit in aggregate system memory. This means that candidates must be written out of disk and divided into partitions small enough to be processed in memory results further data scan. Hence, limits scalability of the most current algorithms.
III. Current algorithms require multiple passes over the databases. For disk resident databases, this requires reading the database completely for each pass. Current algorithms can handle only a few thousand items. Second iteration counts the frequency of all 2-itemsets and essentially has quadratic complexity because one must consider all item pairs and no pruning is possible at this stage.
IV. Present algorithms used only a static load balancing based on the initial data decomposition, and they assumed a homogeneous dedicated environment. A typical database parallel server has multiple users and transient loads. Dynamic load balancing is also crucial in a hetrogenious environment. Such an environment might include metaclusters and superclusters, with machines ranging from ordinary workstations to super computer.
V. The main focus of the current algorithm has been frequent itemset discovery. There is no attention on rule generation because the assumption was that there were only a few different itemsets, which led researchers to believe that rule generation was cheap. One can extract literally millions of frequent itemsets. The complexity of the rule generation step is $0(r.2^l)$, where r is the number of frequent itemsets and l is the longest frequent pattern.
VI. Most of the methods partition the data base horizontally in equal size – blocks. The number of frequent itemsets generated from each block can be heavily skewed. That is, although one block might contribute many frequent itemsets, the other might have very few, implying that the processor responsible latter block will be ideal most of the time. The Effect of data skew can be eliminated to large extent by randomizing the data allocated to each partition.

## 7. CONCLUSIONS

This paper summarized survey of clustering algorithms, association rules algorithms from a large amount of data, sequential/parallel algorithms and classification algorithms based on it are describe with their merits and demerits. These algorithms are not suitable in many applications. Some of the limitations are given and also suggested the research area for data mining system.

## REFERENCES


[1] R. Agrawal, T. Imienski and A. Swamy, Database Mining : A Performance Perspective, IEEE Tran. On Knowledge and Data Engg., December,1991.

[2] M-S Chen, J Han and P. S. Yu, Data Mining : An Overview from a Database Perspective, IEEE Tran. On Knowledge and Data Engg., December,1996.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data,* Washington, D.C., May 1993

[4] Agarwal and R.Srikant, "*Fast Algorithms for Mining Association Rules*," Proc. 20th International Conference. Very Large Databases, Santiago, Chile, Sept 1994.

[5] A.Y. Zomya, T.E. Ghazawi and O. Frieder, Parallel and Distributed Computing for Data Mining, IEEE Concurrency, Oct./Nov. 1999.

[6] R. Agrawal and Ramakrishnan Srikant. Fast Algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases,* Santiago, Chile, August 29- September 1 1994.

[7] Ashok Savasere, Edward Omiecinski and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st VLDB Conference,* Zurich, Switzerland, 1995.

[8] Mueller A. Fast sequential and parallel algorithms for association rule mining: A comparison. Technical Report CS-TR-3515, Department of Computing Science, University of Maryland, College Park, MD, 1995.

[9] Jong Soo Park, Ming-Syan Chenand Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of 1995 ACM-SiGMOID international Conference on Management of Data,* 1995.

[10] Mohammed J. Zaki, Rensselaer polytechnic Institute. Association Mining: A Survey. IEEE Concurrency, 1999.

[11] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. Technical Report CMPT99-12, School of Computing Science, Simon Fraser University, 1999.

[12] Mining Frequent Itemsets by Transaction Decomposition with Itemset Clustering I-En Liao, Ke-Chung Lin, Hong-Bin Chen Int'l Conf. Data Mining | DMIN'09 |

[13] A NEW TERM WEIGHTING SCHEME FOR DOCUMENT CLUSTERING A. Keerthiram Murugesan1 and B. Jun Zhang1 Int'l Conf. Data Mining | DMIN'11 |

[14] MRC: Multi Relational Clustering approach Majid Rastegar-Mojarad, Behrouz Minaei-Bidgoli Int'l Conf. Data Mining | DMIN'09 |

[15] Agarwal and R.Srikant, "*Fast Algorithms for Mining Association Rules*," Proc. 20th International Conference. Very Large Databases, Santiago, Chile, Sept 1994.

[16] D. W. Cheung, V.T. Ng, A.W. Fu and Y Fu, Efficient Mining of Association Rules in Distributed Databases, IEEE Tran. On Knowledge and Data Engg. December,96.

[17] A.Y. Zomya, T.E. Ghazawi and O. Frieder, Parallel and Distributed Computing for Data Mining, IEEE Concurrency, Oct./Nov. 1999.

[18] Skillicorn, Strategies for Parallel Data Mining.IEEE Concurrency, Nov. 1999.

[19] A. Mueller," *Fast and Sequential Algorithms for Association Rule Mining."* A comparison, Tech Report CS-TR-3515, Univ. of Maryland, College Park. Md. 1995.

[20] J.S. Park, M.Chen and P.S. Yu, "Effective *Hash-Based Algorithm for Mining Association Rules*", ACM International Conf. Information and Knowledge Management, ACM Press, May 1995.

[21] Margaret H. Dunham and Yongqiao Xiao, Southern Methodist University, Dallas, Texas and Le Gruenwald, Zahid Hossain, University of Oklahoma, Norman UK, " *A survey of Association Rules*"

[22] Mohammed J. Zaki, Rensselaer Polytechnic Institute, "*Parallel and Distributed Association Mining: A Survey*", IEEE Concurrency 1999

[23] Data Mining in the Real World: Experiences, Challenges, and Recommendations Gary Weiss Int'l Conf. Data Mining | DMIN'09 |