Mining Frequent Item set Using Genetic Algorithm

Hardik Patel¹ Prof. Jigar Patel²
¹, ²Department of Computer Science
1, 2Alpha College of Engg and technology Khatraj, Ahmadabad 382 481, India

Abstract— By applying rule mining algorithms, frequent itemsets are generated from large data sets e.g. Apriori algorithm. It takes so much computer time to compute all frequent itemsets. We can solve this problem much efficiently by using Genetic Algorithm(GA). GA performs global search and the time complexity is less compared to other algorithms. Genetic Algorithms (GAs) are adaptive heuristic search & optimization method for solving both constrained and unconstrained problems based on the evolutionary ideas of natural selection and genetic. The main aim of this work is to find all the frequent itemsets from given data sets using genetic algorithm & compare the results generated by GA with other algorithms. Population size, number of generation, crossover probability, and mutation probability are the parameters of GA which affect the quality of result and time of calculation.

I. INTRODUCTION

Studies of Frequent Itemset (or pattern) Mining is recognized in the data mining field because of its large applications in mining association rules, correlations, and graph pattern constraint based on frequent patterns, sequential patterns, and many other data mining tasks. Capable algorithms for mining frequent itemsets are critical for mining association rules as well as for many other data mining tasks. The major challenge found in frequent pattern mining is a large number of result patterns. As the minimum threshold becomes lower, an exponentially large number of itemsets are generated. So, pruning unimportant patterns can be done effectively in mining process and that becomes one of the main topics in frequent pattern mining. Therefore, the main aim is to optimize the process of finding patterns which should be efficient and can detect the important patterns which can be used in various ways.

Genetic algorithms (GAs), inspired by biological evolution, are efficient domain independent search methods. That is, these methods could help us in effectively solving problem in different application domain. The goals of Holland's research have been twofold: First, to abstract and rigorously explain the adaptive processes of nature systems. Second, to design artificial systems software that retains the important mechanisms of nature system. These methods are capable of applying in many fields and perform well. From the viewpoint of AI research, Holland’s method provides a good mechanism of learning.

GAs are population-based search techniques that maintain populations of potential solutions during searches. A string with a fixed bit-length usually represents a potential solution. In order to evaluate each potential solution, GAs need a payoff (or reward, objective) function that assigns scalar payoff to any particular solution. Once the representation scheme and evaluation function is determined, a GA can start searching. Initially, often at random, GAs create a certain number, called the population size, of strings to form the first generation. Next, the payoff function is used to evaluate each solution in this first generation. Better solutions obtain higher payoffs. Then, on the basis of these evaluations, some genetic operations are employed to generate the next generation. The procedures of evaluation and generation are iteratively performed until the optimal solution(s) is (are) found or the time allotted for computation ends.

The goal of this paper is to review mining frequent item sets using different methods.

II. DIFFERENT METHODS OF MINING FREQUENT ITEM SETS

A. An Algorithm for Frequent Pattern Mining Based On Apriori:

Frequent pattern mining is a heavily researched area in the field of data mining with wide range of applications. Mining frequent patterns from large scale databases has emerged as an important problem in data mining and knowledge discovery community number of algorithms has been proposed to determine frequent pattern. Apriori algorithm is the first algorithm proposed in this field. With the time a number of changes proposed in Apriori to enhance the performance in term of time and number of database passes. In this paper three different frequent pattern mining approaches (Record filter, Intersection and Proposed Algorithm) are given based on classical Apriori algorithm. In these approaches Record filter approach proved better than classical Apriori Algorithm, Intersection approach proved better than Record filter approach and finally proposed algorithm proved that it is much better than other frequent pattern mining algorithm. In last we perform a comparative study of all approaches on dataset of 2000 transaction.

Conclusion- Association rule mining has a wide range of applicability such as market basket analysis, medical diagnosis/ research, website navigation analysis, homeland security and so on. In this method, we surveyed the list of existing association rule mining techniques and compare these algorithms with our modified approach. The conventional algorithm of association rules discovery proceeds in two and more steps but in our approach discovery of all frequent item will take the same steps but it will take the less time as compare to the conventional algorithm. We can conclude that in this new approach, we have the key ideas of reducing time. As we have proved above how the proposed Apriori algorithm take less time than that of classical apriori algorithms. That is really going to be fruitful in saving the time in case of large database. This key idea is surely going to open a new gateway for the upcoming researcher to work in the filed of the data mining.
B. Efficient Algorithm For Mining Frequent Itemsets Using Clustering Techniques

Now a days, Association rule plays an important role. The purchasing of one product when another product is purchased represents an association rule. The Apriori algorithm is the basic algorithm for mining association rules. This paper presents an efficient Partition Algorithm for Mining Frequent Itemsets (PAFI) using clustering. This algorithm finds the frequent itemsets by partitioning the database transactions into clusters. Clusters are formed based on the similarity measures between the transactions. Then it finds the frequent itemsets with the transactions in the clusters directly using improved Apriori algorithm which further reduces the number of scans in the database and hence improve the efficiency.

In this method, the Partition Algorithm for Frequent Itemset (PAFI) is proposed before applying Improved Apriori Algorithm. This algorithm reduces the number of scans in the database and improves efficiency and computing time by taking the advantage of clustering technique. By experiment results, it can obtain higher efficiency.

C. Efficient Hardware Data Mining For Frequent Item-set With Apriori Algorithm

The Apriori algorithm is a popular correlation-based data mining kernel. However, it is a computationally expensive algorithm and the running times can stretch up to days for large databases, as database sizes can extend to Gigabytes. Through the use of a new extension to the systolic array architecture, time required for processing can be significantly reduced. Our array architecture implementation on a Xilinx Virtex-II Pro 100 provides a performance improvement that can be orders of magnitude faster than the state-of-the-art software implementations. The system is easily scalable and introduces an efficient systolic injection method for intelligently reporting unpredictably generated mid-array results to a controller without any chance of collision or excessive stalling.

FPGA implementations of the Apriori algorithm can provide significant performance improvement over software-based approaches. We are also interested in implementing some of the more recent (and more control-intensive and memory-intensive) approaches in hardware, including hash-based strategies such as DHP and trie-based approaches. It may be possible to increase the bandwidth of the system by processing several sub-partitions of a set in parallel. We are also interested in leveraging our experience with high-performance string matching for autonomous pattern generation for network security.

D. Mining Frequent Item set for Non Binary Data set using Genetic Algorithm

Frequent itemset mining is a basic problem in data mining and knowledge discovery. The discovered patterns can be used as input for Association rules, which are useful in many application domains. We have considered a large database of customer transactions from a super market. Each transaction consists of items purchased by a customer in a visit. We present an efficient algorithm that generates all significant association rules between items in the database.

In general the association rule mining algorithms like Apriori, partition, pincer-search, incremental, border algorithm etc, does not consider negation occurrence of the attribute in them and also take more time to compute all the frequent itemsets. By using Genetic Algorithm (GA), we can improve the scenario and the system can predict the rules which contain negative attributes in the generated rules, even with more than one attribute in consequent part. The major advantage of using GA in the discovery of frequent itemsets is that they perform global search and its time complexity is less compared to that of other algorithms which are based on the greedy approach. The main aim of this method is to find all possible frequent item sets from given dataset using the genetic algorithm.

III. INTRODUCTION TO GENETIC ALGORITHM

Genetic algorithms (GAs), inspired by biological development, are efficient domain self-sufficient search methods. That is, these methods could help us in effectively solving problem in different application domain. The goals of Holland's research have been twofold: First, to conceptual and strictly explain the adaptive processes of character systems. Second, to design artificial systems software that retains the important mechanisms of nature system. These methods are capable of applying in many fields and execute well. From the viewpoint of AI research, Holland's method provides a good mechanism of learning.

Genetic Algorithms are population-based search techniques that maintain populations of probable solutions during searches. A string with a fixed bit-length usually represents a probable solution. In order to assess each potential solution, GAs need a payoff (or reward, objective) functions that assigns scalar induce to any particular solution. Once the representation scheme and estimate function is determined, a GA can start searching. Initially, often at casual, GAs create a positive number, called the population size, of strings to form the first generation. Next, the payoff function is used to evaluate each solution in this first generation. Better solutions obtain higher payoffs. Then, on the basis of these evaluation, some genetic operations are employed to generate the next generation.

A. SIMPLE GENETIC ALGORITHM

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
3. [New population] Create a new population by repeating following steps until the new population is complete
   a) [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
   b) [Crossover] With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
   c) [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
d) **[Accepting]** Place new offspring in the new population
4. **[Replace]** Use new generated population for a further run of the algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
**[Loop]** Go to step 2

IV. GENETIC ALGORITHM vs TRADITIONAL METHODS

The following list gives the essential differences between GAs and other forms of optimization.

1. Genetic algorithms a coded form of the function values (parameter set), rather than with the actual values themselves. So, for example, if we want to find the minimum of the function \( f(x) = x^3 + x^2 + 5 \), the GA would not deal directly with \( x \) or \( y \) values, but with strings that encode these values. For this case, strings representing the binary \( x \) values should be used.

2. Genetic algorithms use a set, or population, of points to conduct a search, not just a single point on the problem space. This gives GAs the power to search noisy spaces littered with local optimum points. Instead of relying on a single point to search through the space, the GAs looks at many different areas of the problem space at once, and uses all of this information to guide it.

3. Genetic algorithms use only payoff information to guide themselves through the problem space. Many search techniques need a variety of information to guide themselves. Hill climbing methods require derivatives, for example. The only information a GA needs is some measure of fitness about a point in the space (sometimes known as an objective function value). Once the GA knows the current measure of "goodness" about a point, it can use this to continue searching for the optimum.

4. GAs are probabilistic in nature, not deterministic. This is a direct result of the randomization techniques used by GAs.

5. GAs are inherently parallel. Here lies one of the most powerful features of genetic algorithms. GAs, by their nature are very parallel, dealing with a large number of points (strings) simultaneously.

<table>
<thead>
<tr>
<th></th>
<th>Ad hoc approach (analytical, specific)</th>
<th>Genetic approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Depending on solution, generally good</td>
<td>Median or low</td>
</tr>
<tr>
<td>Performance</td>
<td>Depending on solution</td>
<td>Fair to excellent</td>
</tr>
<tr>
<td>Problem understanding</td>
<td>Necessary</td>
<td>Not necessary</td>
</tr>
<tr>
<td>Human work needed</td>
<td>A few minutes to a few theses</td>
<td>A few days</td>
</tr>
<tr>
<td>Applicability</td>
<td>Low: Most interesting problems have no usable mathematical expression, or are non-computable, or &quot;NP-complete&quot; (too many solutions to try them all)</td>
<td>General</td>
</tr>
<tr>
<td>Intermediary steps</td>
<td>not solutions (you must wait until the end of computation)</td>
<td>are solutions (the solving process can be interrupted at any time, though the later the better)</td>
</tr>
</tbody>
</table>

Table. 1: Comparison GA with traditional Algorithms.