Journal of Indian Research (ISSN: 2321-4155)
Vol.1, No.4, October-December, 2013, 71-83.

Computation Science

HYBRID DATA MINING ALGORITHM: AN APPLICATION TO WEATHER DATA

B. Athiyaman*
R Sahu**
A. Srivastava****

ABSTRACT

Data mining is an attitude that business actions should be based on learning, that informed decisions are better than uninformed decisions, and measuring results is highly beneficial to analyze the large data sets. Association rule mining is the most commonly used techniques in Data mining. The application rule involves combining algorithms from apriori algorithm and clustering mining applying different kinds of association rules and its extensions on large data sets like weather analysis data . This process is used to analyze meteorological data to find hidden patterns so that the information retrieved can be transformed into usable knowledge. The expansion of earth science data in recent years have been made possible by advances in parallel, high-throughput technologies in the area of earth science. This has ushered in a new era in the area of earth science information system. The paper aims to build an effective and accurate tool to analyze and extract hidden knowledge from this huge data.

Keywords: Efficient Association, Rule Mining, Market Basket Analysis

INTRODUCTION

Data mining has recently attracted tremendous amount of attention in database research because of its applicability in many areas. In the data mining, the database can be transaction databases or analytical databases. Analytical databases can work as transaction databases by using the predicate logic. The mining method of association rule in the transaction databases is suitable for analytical databases.

Meteorological data mining is a form of Data mining concerned with finding hidden patterns inside largely available meteorological data, so that the information retrieved can be transformed into usable knowledge. We try to extract useful knowledge from daily historical weather data collected locally for Indian major cities. The data includes monthly period. After data preprocessing, we apply outlier analysis association clustering rules mining techniques. After each mining technique, we present the extracted knowledge and describe its importance in meteorological field.

With wide application of computers and automated data collection tools, massive amount of data has been continuously collected and stored in databases, which creates an imminent need and great opportunities for mining interesting knowledge from data. Association rule mining is one kind of data mining technique which discovers association or correlation relationships among data. The discovered rules may help market basket or cross-sale analysis decision making and

---

***B. Athiyaman** is Scientist & Head Computer & Networking Division at NCMRWF, Ministry of Earth Sciences, New Delhi.

****R Sahu** is Professor & Director, IIIT, Basar, AP, India-504101.

***** **Dr.Azad Srivastava** is the Head, High Performance Computing & Open Source Development Practices, CMC Ltd , Noida, India.

business management. Mining association rule means that, given a database of transactions, to discover all associations among items such that the presence of some items in a transaction will imply the presence of other items in the same transaction. The mining of association rules can be mapped into the problem of discovering large item sets where a large itemset is a group of items that appear in a sufficient number of transactions. Usually large itemset generation is dominating factor for the overall data mining performance.

2. Literature Survey – Single dimensional association rule mining algorithms

There are two main algorithms that are in common use:

i) The Apriori Algorithm: mining frequent itemsets using candidate generation (*Agrawal & Srikant*, 1994)

ii) AprioriTid algorithm

2.1 The Apriori Algorithm

The apriori-gen function takes as argument Lk-1 the set of all large (k-1) item sets. It returns a superset of the set of all large k-itemsets. The function works in two steps. These two steps are similar to the join and prune steps, respectively. However, in general, first step produce a superset of the candidates produced by the join step. The major drawbacks of this approach are as follow:

i) *Generating large number of frequent itemsets is expensive*: 106 frequent 1-itemsets require testing of 5*10^11 candidate 2-itemsets.

ii) *Not good for long patterns*: A frequent itemset of size 100 requires testing of 2100 ≈1030 smaller candidate itemsets.

iii) Repeated scans of database are expensive.

iv) The main bottleneck is the candidate generation mechanism.

Analysis of Apriori and AprioriTid algorithm

In this work Apriori and AprioriTid algorithm, which are used to construct the frequent itemset, are analyzed. On the basis of analysis, it is found that too many data due to those items is repeatedly saved in the AprioriTid algorithm. An efficient AprioriTid algorithm is presented. The implementation shows the new algorithm is more effective in decreasing data size and execution times than AprioriTid algorithm.

2.2. AprioriTid algorithm

As against Apriori algorithm, the AprioriTid algorithm [**4**] has the additional property that the database is not used at all for counting the support of candidate itemsets after the first pass. Rather, an encoding of the candidate itemsets used in the previous pass is employed for this purpose. In later passes, the size of this encoding can become much smaller than the database, thus saving much reading effort. The AprioriTid algorithm also uses the Apriori-gen function to determine the candidate itemsets before the pass begins. The interesting feature of this algorithm is that the database D is not used for counting support after the first pass. Rather, the set Ck is used for this purpose. Each member of the set Ck is of the form < TID,{Xk} > , where each Xk is a potentially large k-itemset present in the transaction with identifier TID. For k=1, C1 corresponds to the database D, although conceptually each item I is replaced by the itemset {i}. For k>1, Ck is generated by the algorithm. The member of Ck corresponding to transaction t is < t.TID, {c∈Ck | c contained in t} >. If a transaction does not contain any candidate k-itemset, then Ck will not have an entry for this transaction. Thus, the number of entries in Ck may be smaller than the number of transactions in the database, especially for large values of k. In addition, for large values of k, each entry may be smaller than the corresponding transaction because very few candidates may be contained in the transaction. However, for small values for k, each entry may be larger than the corresponding transaction because an entry in Ck includes all candidate k-itemsets contained in the transaction.

From above analysis, using Ck is efficient method to avoid scanning DB in AprioriTid algorithm **[5, 6**]. However, same item may appear in many candidate itemsets contained in a transaction and appear many times in Ck , which will make item to be repeated stored and improve size of query data. In next fraction, I will give an idea for reducing size of data stored in Ck and a new efficient AprioriTid algorithm.

3. Efficient AprioriTid Algorithm

3.1. Proposed method

Method: **If c∈Ck-1 and c.support<minsup, c is useless in Ck-1.**

Explanation: As large itemset (Lk-1) are generated from candidateset(Ck-1),and Lk-1={c∈Ck-1 |c.count>=minsup}. It is also known that Ck is generated from Lk-1 , thus it can be said that any *itemset in Ck-1 whose support is less than minimum support, will not appear in Ck.* Hence, no need to consider those itemsets to build Ck-1, and this will reduce the size of Ck-1 in initial passes.

3.2 Efficient AprioriTid Algorithm

Efficient AprioriTid algorithm is improved AprioriTid algorithm. The basic idea is to only use itemsets of Ck-1, whose supports are equal to or greater than minsup,to build Ck-1; So a new relationship of transaction t with entry, is defined as Ck =<t.TID,{c∈Ck |c∈ t and c.support >=minsup}> ; This will consumedly decrease size of stored data in Ck ; Moreover, because searching data scale is reduced when we compute support of itemsets in Ck. At the same time, it will reduce much time of I/O and running.

In AprioriTid algorithm, from step 6 to step 11, it is to compute new Ck and support of itemset in Ck. So we can add the sentence, which will delete itemset that support is smaller than minsup in Ck, to optimize the algorithm after step 11.

```
1)  L1={large-Itemsets};
2)  C1=database D;
3)  For(k=2;Lk-1≠ϕ;K) do begin
4)  Ck=generate Lk from Lk-1 ;//New
    candidates
5)  Ck=ϕ;
6)  For all C ∈Ck do begin
7)  Tc={t.TID|t∈ Ck-1,c-c[k])∈ t.set-of-itemset
            Λ (c-c[k-1]∈ t.set-of-itemsets)}
8)  If|Tc| >minsup then begin
9)  Lk=LkU{C}
10) For all p∈ Tc do
11) Ck=CkUp,c>;
12) End
13) End
14) End
15) Answer=ULk;
```

Figure 1: AprioriTid algorithm

The material process is shown:

Step 1 : Firstly, Confirming itemset c in Ck, then transaction set Tc, presented with TID, including the item of c, is computed;

Step 2 :The number of entry is computed in Tc, defined as |Tc|, which is support of itemset c;

Step3: if |Tc| > minsup , c is included into Lk and Ck , otherwise deleting c.

Through above process, with the computed support in Ck, void itemset can be directly deleted from Ck or added to Ck and Lk. In following, the efficient AprioriTid algorithm is given.

From Step 8 to 12 of algorithm, it can be seen that eligible itemset c is added into Lk and all transactions of Tc are added into Ck.

4. Experiments and conclusions

4.1. Experiments

For validating the effect of Efficient A algorithm, I had taken a course database to mine association rule of course relationship. In this database, total records are 3486 minsup is support (%)of total records. For different values of support comparison in execution times of both algorithms is given in **Table** 1. The number of records in Ck and Ck are given in **Table** 2.

Table1: Comparison of execution time in sec.

Support	**AprioriTid Algorithm(sec.)**	**Efficient AprioriTid Algorithm(sec.)**
10	38.02	17.89
20	11.14	4.2
30	4.31	1.54
40	1.92	0.88
50	0.79	0.51
60	0.47	0.05

Table 2. Against Record Number

Passes(K)		**AprioriTid Algorithm**	**Efficient AprioriTid Algorithm**
K=2	C2	74	32
	C2	801	766
K=3	C3	85	18
	C3	883	734
K= 4	C4	14	1
	C4	445	239

From the **Table**1 and **Table** 2, it can be seen that efficient AprioriTid algorithm is valid in reducing data scale and execution time. The number of records is averagely reduced to 47%.With the searching scales decreasing; the algorithm complexity of efficient aprioriTid algorithm is reduced than AprioriTid algorithm.

With the above transaction database, experiment with different values of support to contrast execution time and record number is made between Efficient AprioriTid and AprioriTid algorithm. The experiment is performed on a workstation with a CPU clock rate of 2.0 GHz, 1 GB of main memory. The result is shown as follow:

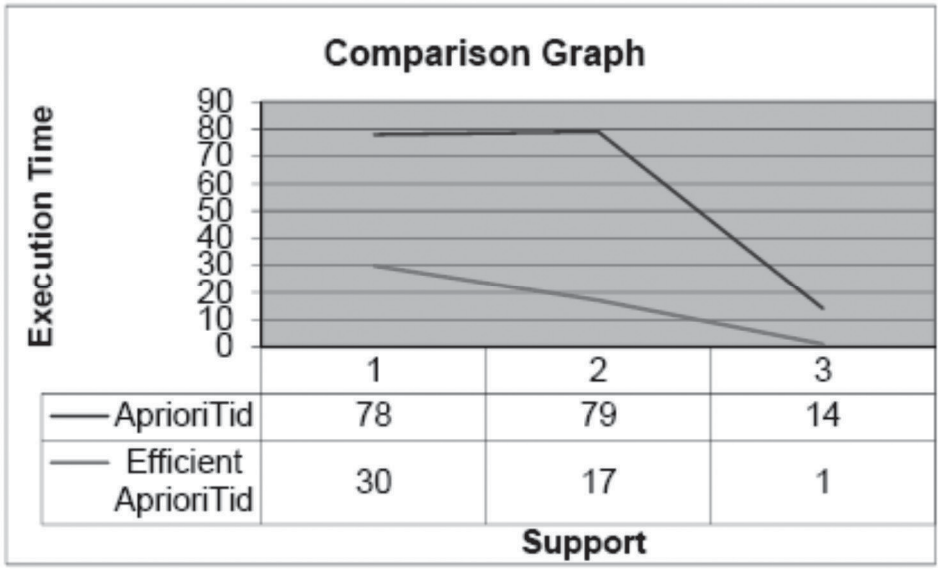
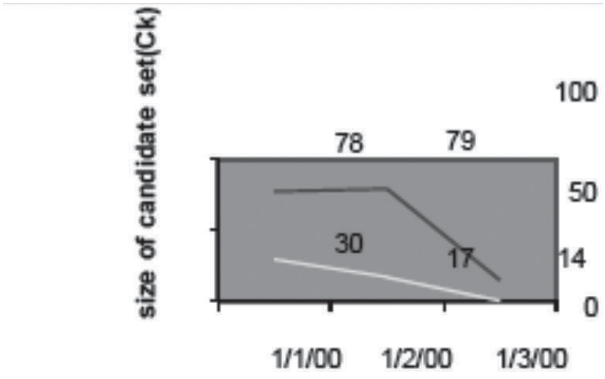


Figure 2: Execution times

	1	2	3
AprioriTid(Ck')	768	736	445
Efficient prioriTid(C	759	632	239

Figure 3. Record number



	1	2	3
	1	2	3
AprioriTid(Ck)	78	79	14
EfficientA prioriTid(C	30	17	1

Figure 4: Record Number

Figure 2 shows that with the minsup decreasing, the execution times of two algorithms increase because of increase in the total number of candidate and large itemsets. However, the Efficient AprioriTid Algorithm execution time is smaller than AprioriTid for searching data scale if reduced in Ck. When the candidate itemsets in *Ck* are rapidly increased with minsup reduced step by step, Efficient AprioriTid can save more execution times than AprioriTid algorithm.

First detail analyzes the Apriori and AprioriTid algorithm, then gives new idea to reduce the scale of candidate itemsets. On the basis of idea, i gives an improved AprioriTid algorithm. The experiment results show that Efficient AprioriTid can gain a better performance in the execution times and complexity of computing than AprioriTid algorithm.

To take care of the issues of large databases, following is the proposed methodology:

Phase 1: Data Cleaning

Phase 2: Static discretization of quantitative attributes using concept hierarchy

Phase 3: Association Rule Mining

The above phases are sequential in nature. These suggest that each phase produces certain deliverables which is taken sequentially as input by the next phase.

4.1 Phase 1: Data Cleaning

After selecting the attribute for analysis, the data related to those attributes is cleaned. The various steps of data cleaning are:

1 **Eliminating all the tuples with large missing values –** When a lot many values are missing in a tuple, the data authenticity cannot be verified. Due to large dataset, the tuples having more than 33% of missing values to be eliminated.

2 **Dataset partitioning based on locations** – The dataset is collected from various sites. As a particular region exhibit similar behavior, one needs to partition all the tuples based on *stationId*.

To achieve this, various intermediate temporary tables are created in database to hold the tuples belonging to a particular stationId. Running a single database scan and segregating all the tuples belonging to a particular *stationId* can be done.

This step is very important as the segregation according to *stationId* is going to be very helpful while filling the missing values.

Filling Missing Values: After deleting tuples having more than 33% missing attributes, the tuples with low missing attributes are filled with as follows:

Assumption: The value of attributes like pressure, temperature depends upon the corresponding values at its surrounding stations.

Using the above assumption, an algorithm for filling missing values has been proposed called *Nearby Station Algorithm*. The various parameters are discussed below.

The surrounding stations are defined as all stations which are within the radius *r* of 100 miles from *base station*. *Base station* is the one for which value of attributes are missing. The value of radius *r* can vary depending upon the nature of climate change patterns in the country.

For filling missing values of attributes of a particular station find out all surrounding stations which are within given radius *r* (we choose 100 miles). Assign them weight according to their distance to base station.

The input parameters taken by algorithm are: currentStationId which is *Base Station.* The *surrounding stations* are stored in an array neighbourStationId. The radius *r* is used to determine how many stations is neighbouring Base station and total neighbour stations are stored in numberOfStation.

```
**********************************************************************************************************
* /**************************
```

*** Nearby Station Algorithm**

```
**************************/Input: currentStationId,neighbourStattionId [] ,numberOfStation
```

Output: missingValueattribute

Procedure : sum

=0; totalWeight

=0;

While(numberOfStation >0)

{d =

findDistance(currentStationId,neighbourStationId[numberOfStation]);

totalWeight + = (1-d/const);

sum += weight*neighbourStationId[numberofStation];

numberOfStation--;

}

missingValueattribute=sum/totalWeight;

realfindDistance(station1,station2){x = 69.1 *(station2.latitute -station1.latitute)

y = 69.1 *(station2.longitute -station1.longitte)*cos(station1.latitue/57.3)

return sqrt(x*x+y*y)

}

Filling Missing Values: After deleting tuples having more than 33% missing attributes, the tuples with low missing attributes are filled with as follows:

Assumption: The value of attributes like pressure, temperature depends upon the corresponding values at its surrounding stations.

Using the above assumption, an algorithm for filling missing values has been proposed called Nearby Station Algorithm. The various parameters are discussed below.

The surrounding stations are defined as all stations which are within the radius r of 100 miles from base station. Base station is the one for which value of attributes are missing. The value of radius *r* can vary depending upon the nature of climate change patterns in the country.

For filling missing values of attributes of a particular station find out all surrounding stations which are within given radius **r** (we choose 100 miles). Assign them weight according to their distance to base station.

The input parameters taken by algorithm are: currentStationId which is BaseStation. The surrounding stations are stored in an array neighbourStationId. The radius *r* is used to determine how many stations is neighbouring Base station

and total neighbour stations are stored in numberOfStation.

The problem of finding association rules falls within the purview of database mining [*Agrawal et al.*, 1993c; *Anwar et al.*, 1992; *Holsheimer* and *Siebes*, 1994; *Michalski et al.*, 1992; *Agrawal et al*., 1993b; and *Sur*, 1990], also called Knowledge Discovery in databases *Han et al.*, 1992; *Lubinsky*, 1989; and *Shapiro*, 1991. Related but not directly applicable, work includes the induction of classification rules [*Breiman et al*., 1984; *Catlett*, 1991, *Fayyad et al.*, 1993, *Han et al.*, 1992; and *Ross*, 1993]. The other works in the machine learning literature is the KID3 algorithm presented in *Piatestky* (1991). If used for finding associations this algorithm will make as many passes over the data as the number of combinations of items in the antecedent, which is exponentially large. Related work in the database literature is the work on inferring functional dependencies from data [*Bitton*, 1992; and *Mannila et al*., 1987].

Table 4: Associations rules for UNA weather data

#	Rule	Conf.
1	[RH=mid Temp=warm Wind=Moderate] ==> [Rain=no rain]	0.99
2	[RH=high Temp=warm] ==> [Rain=no rain]	0.99
3	[Temp=warm Wind=Moderate] ==> [Rain=no rain]	0.99
4	[month = 2]==> [temp = cold]	0.96
5	[month = 1] ==> [temp = cold]	0.96
6	[month = 12] ==> [temp = cold]	0.95
7	[Wind=Light] ==> [Rain=no rain]	0.91
8	[Wind=light Temp=cold rain]==>[RH=moderate]	0.91
9	[Rain= Heavy Rain]==>[Temp = cold]	0.88
10	[Temp = cold]==>[Wind = Moderate]	0.74
11	[RH= low Wind = Moderate Temp=warm] ==>[Rain=Light Rain]	0.65
12	[Wind = Moderate] --> [RH = mid]	0.60

The implementation of algorithm shown in **Figure** 1 works in the following fashion:

1 It assigned the maximum weight to the station which is much nearer to the *base station*.

2 Weights of each station are calculated on the basis of distance using their latitude and longitude values with respect to *base station*.

3 On particular date weighted mean of corresponding attribute values with the above mentioned weight is taken.

The above method can work effectively only when the minimum number of stations called *minStation* is 10. The value of *minStation* can be varied according to density of weather stations in a particular geographical location. The value of radius *r* and *minStation* are taken considering the climatic conditions in the Indian subcontinent.

In case, numbers of *surrounding stations* are less than *minStation.* Then the tuples with low missing attributes are filled as:

1 Take the average of weighted mean of corresponding attribute values from *surrounding stations.* This value is called *weightedMean1*.

2 Take average of values of the same attribute derived from the tuples belonging to same *base station*. This value is called *Mean2*.

3 Take the average of *weighted Mean1* and *Mean2*.

The intermediate tables formed during the two step of this phase are merged together to form the original database. Thus the important task of filling the missing values has been done.

4.2 Phase 2: Discretization of quantitative attributes using concept hierarchy

The cleaned dataset obtained in previous phase is used to discretize the data. For discretization of attributes in intervals, statistical properties of data have been used. Various steps used in this phase are as follws:

1. **Finding mean and standard deviation of** *all the attributes in the dataset* – Determine the mean and standard deviation of each attribute residing in the dataset since the attribute values are numeric, is fairly easy to compute and can store the statistical parameters of data. Any standard statistical tool is quite capable of calculating mean and standard deviation of large amount of data. Any such tool like SPSS, SAS can be used or one can use a simple computer program to achieve this task.

2. **Replacing original values with their corresponding** *Normally distributed Ranges* – This is a fairly complex step, but ensures great benefits in the final outcome. The previous ways of discretization of the range of attributes into intervals does not utilize the statistical properties of data.

Common partitioning techniques used earlier were based upon dividing range of values with equal intervals, division based on equal frequency of tuples in an interval etc.

Large number of partitions will result in intervals having no significance. Having very few partitions will result in intervals occurring in most rules, i.e. the distinct rules will be very less and important patterns may be lost in large intervals. Thus, the attributes have been divided to cover the extreme values as well as distribute properly the majority of values.

Since the dataset is large, one has a good reason to believe that the data to follow Normal Distribution **[9]**. **Figure** 1 visually represents the partitions. The range ($\mu - \sigma$ toμ)and(μ to$\mu + \sigma$) has been divided into 4 intervals each. The range ($\mu - 2\sigma$ to $\mu - \sigma$) and ($\mu + \sigma$ to $\mu + 2\sigma$) has been divided into two intervals each and remaining one interval for ($\mu - 3\sigma$ to $\mu - 2\sigma$)and($\mu + 2\sigma$ to $\mu + 3\sigma$).

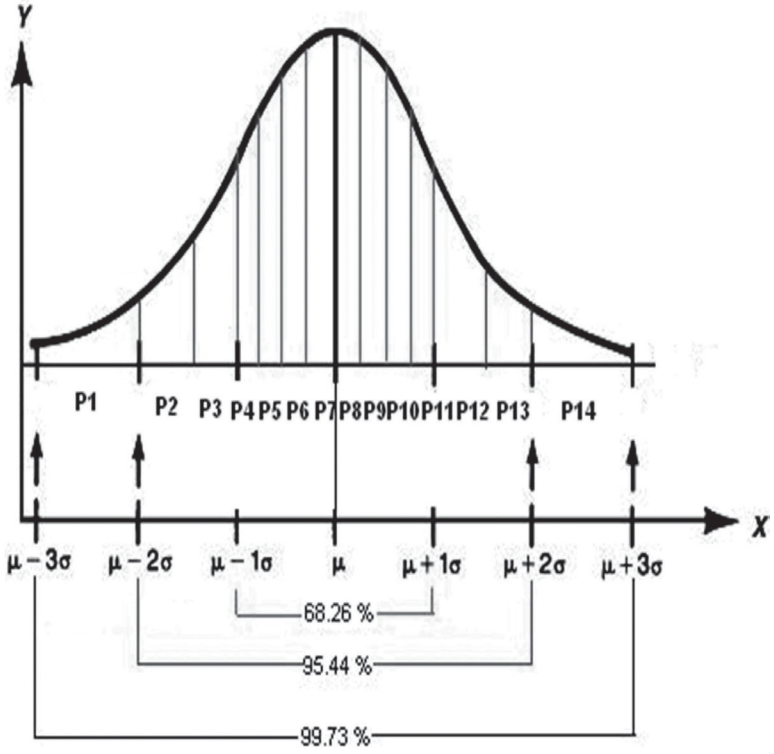


Figure 5: Discretization of Data

Now corresponding to each attribute in the dataset, we define a set of 14 representative ranges. Each range is being represented with a mnemonic of the form *Pi* where *P* being an English letter resembling the attribute and *i* being a number and its value lies in the range 1 to 14.

To calculate the representative range for each tuple one can follow the steps given below:

1. For mean (μ) and standard deviation (σ) corresponding to an attribute calculate following 14 ranges:

^1 P1 = **[** (μ -3σ),(μ -2σ) **]**

^1 P2 = **[** (μ -2σ),(μ -3σ/2) **]**

^1 P3 = **[** (μ -3σ/2),(μ -σ) **]**

^1 P4 = **[** (μ -σ),(μ -3σ/4) **]**

^1 P5 = **[** (μ -3σ/4),(μ -σ/2) **]**

^1 P6 = **[** (μ -σ/2),(μ -σ/4) **]**

^1 P7 = **[** (μ -σ/4), μ **]**

^1 P8 = **[** μ ,(μ + σ/4) **]**

^1 P9 = **[** (μ + σ/4),(μ + σ/2) **]**

^1 P10 = **[** (μ + σ/2) , (μ + 3σ/4) **]**

^1 P11 = **[** (μ +3σ/4),(μ + σ) **]**

^1 P12 = **[** (μ + σ),(μ +3σ/2) **]**

^1 P13 = **[** (μ +3σ/2),(μ +2σ) **]**

^1 P14 = **[** (μ +2σ),(μ +3σ) **]**

2. Now replace each of the values in every tuple corresponding to attribute considered in above step by the ***Pi,*** Where *i* denotes the range number to which the value of the attribute belongs to.

3. Any tuple having the value not falling in any of the ranges above can be represented by either P1 or P14 whichever is closest.

After the completion of the above two steps, the dataset contains the nominal values and no numeric values. Though at first sight , there is loss of original data, but the process makes sure that the nominal values closely resemble the original numeric value, since it uses the Normal Distribution.

This is a crucial and important phase of data transformation. The work done in this phase directly translates into the quality of output in upcoming phases. Though the processing needed during the phase demands certain level of expertise with some of the statistical tools and technologies, but the underlining principle is fairly easy to grasp and deploy.

The resultant dataset is now in nominal form having values of the attributes defined in discrete sets of the form { P1, P2, P3, ….. , P14 }, where P represents the attribute, and i takes a value from 1 to 14. Because of this nominal nature of the dataset, it can now be analyzed with any standard association mining technique to derive the important results.

4.3 Phase 3: Association Rule Mining

The dataset prepared in the previous phase now have categorical attributes instead of numeric values. These categorical attributes can be generalized to higher conceptual levels.

The steps involved in deducting the association rules are as follow:

Mining Multidimensional Association Rules – Weather is continuous, data intensive, multidimensional, dynamic and chaotic [2]. The normal methods for mining Association Rules involves Apriori algorithm [6], but dataset for

association rule mining is different than traditional data set used in Apriori . Modifications in approach are required to mine multidimensional association rules which are inspired from *Kamber et al.* [7].

Due to large size of the database, efficiency needs to be improved. One such method used in this methodology is Transaction reduction [8].

The support and confidence values can either be determined statistically or it can be predicted with the help of domain experts. We proposed that support value be determined after generation of first candidate set by taking the mean of all the values of support of tuples. The initial support should be 1% of total number of tuples so as to cover sufficiently large tuples.

The confidence value initially proposed is as low as 30%, and then it is gradually increased high up to 90% because as the database is large, very large number of rules will be generated.

The output of this phase is set of multi dimensional association rules, which can give us the underlining description of the inter-relationship of attributes in the dataset.

Algorithm:

Input: A sample weather dataset 8 numeric and 2 categorical. The numeric attributes were STATION NAME, visibility, cloud clover, wind Direction, wind Speed, temperature, dew Point Temperature, station, Pressure, sea level Pressure, minimum support threshold, minimum joint profit.

5.1 Phase 3: Association Rule Mining

The original dataset was collected from over 4000 different stations from India. The considerable amounts of values of different attributes were missing from the dataset. Thus, the first step in the methodology proved to be very crucial. The dataset obtained after filling the missing values were used for partitioning of the attributes based on the 14 representative ranges defined in the two phase of methodology. The purpose of this phase was to obtain categorical attributes for the corresponding numeric attributes in the dataset. **Figure** 3 displays the dataset obtained after the completion of phase two. The Association Rule Mining is applied over the dataset in the three phases. Due to large dataset, the transaction reduction technique discussed in [**8**] was very helpful. The count of original dataset was 443146 which reduced dramatically during intermediate steps in Association rule mining.

The support was taken to be 30 and minimum confidence was initially 30% but it was increased to 80% during the mining process.

For eight attributes, time taken to generate candidate and frequent predicate sets is very high. Therefore, the rules displayed are generated after the five frequent predicate set and time taken to generate rules is about 60 hours. The rules are shown in **Figure** 4.

Output:

DISPLAYING THE RULES: CONFIDENCE

1. CLC4^STP11^TEMP10^DPT9 ->WSP4 69.0

2.CLC8^SLP3^TEMP9^VIS6 ->WSP4 42.0

3.VIS4^SLP10^TEMP7^WID5 ->WSP4 72.0

4.WID4^DPT9^TEMP13^WSP5 ->WSP4 31.0

CONCLUSION & SCOPE OF FUTURE WORK

The proposed Methodology provides a comprehensive knowledge about how to deal with large datasets. The methodology is easy but requires good knowledge of data mining.

Given the fact that methodology works with only after cleaning the data which involves filling the missing values using the nearby station algorithm developed , then as a next step one has to use normal distribution for obtaining respective categorical attributes. This converts the dataset into nominal values so that Association rule mining algorithms

can be adapted. In the last step, Apriori algorithm is applied with some modifications suitable for mining multi association rules.

The proposed methodology provides an easy way of weather analysis for large data or similar works. The present approach to extract important association rules though complete, but it can be expanded to add enhanced functionality. To exemplify, the rules mined at the end of last step it can be further classified based on station-Id. This way we can use rules specific to each city, rather than using rules which apply in general to all cities.

REFERENCES

1. Agrawal, R. , T. Imielinski, and A. Swami(1993). "Mining Association Rules Between Sets of Items in Large Databases," in *ACM SIGMOD International Conference on Management of Data,* May 1993
2. Agrawal, R. and R. Srikant(19940). "Fast algorithms for mining association rules," in *20th VLDB Conference*, Santiago, Chile, September 1994.
3. Mannila, H., R. Shrikant, H. Toivonan, and A. I. Verkamo(1994). "Efficient algorithms for discovering association rules," in *AAAI Workshop, Knowledge Discovery in Databases*, July 1994.
4. Klemettinen, M., H. Mannila, P. Ponkainen, H. Toivonen, and A. I. Verkamo(1994). "Finding Interesting Rules from Large Sets from Discovered Association Rules," in *3rd International Conference on Information and Knowledge Management,* November 1994.
5. Savasere, A., E. Omiecinski and S. Navathe(1995). An Efficient Algorithm for Mining Association Rules for Large Databases, In *Proceedings of 21st Conference of VLDB,* Zurich, Switzerland, September 1995.
6. Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy(1995). "*Advances in Knowledge Discovery and Data Mining,"* AAAI/MIT Press, 1995.
7. Agrawal, R.and J. Shafer(1996). Parallel mining of association rules, in *IEEE trans., on Knowledge and data Engg.,* 1996.
8. Agrawal, R., H. Mannila, R. Shrikant, H. Toivonan, and A. I. Verkamo(1996). "Fast discovery of association rules," in U. Fayyad et el., eds. *Advances in Knowledge Discovery and Data Mining*, MIT Press.
9. Toivonen, H (1996) "Sampling Large Databases for Association Rules," in *22nd VLDB Conference*.
10. Feldman, R., Y. Aumann, A. Amir, and H. Mannila(1997). "Efficient algorithms for discovering frequent sets in incremental databases," in *Proceedings of ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*, 1997.
11. Cheuang, D., S. D. Lee, and B. Kao(1997). "A General Incremental Technique for Maintaining Discovered Association Rules," In *Proceedings of the 5th International Conference on Database Systems for Advance Applications (DASFAA '97)*, pp.185-194, Melbourne, Australia, April 1997.
12. Thomas, S., S. Bodagala, K. Alsabti, and S. Ranka(1997). "An Efficient algorithm for the incremental updating of association rules in large database," In the *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp. 263-266, Newport Beach, California, USA, August 1997.
13. Agrawal, R. C., C. Agrawal, and V. V. V. Prasad(2000). A Tree Projection Algorithm for Generation of Frequent Itemsets. *Journal of Parallel and Distributed Computing* (Special Issue on High Performance Data Mining), 2000.
14. Ayad, Ahmed, Nagwa El-Makky, and Yoursy Taha(2001). "Incremental Mining of Constructed Association Rules," *First SIAM International Conference on Data Mining*, April 5-7, 2001, Chicago, USA.
15. Agrawal ,Charu C., Cecilia Procopiuc, and Philip S. Yu(2001). Finding Localized Associations Show-Jane Yen and Arbee L. P. Chen. A Graph – Based Approach for Discovering Various Types of Association Rules. *IEEE Transactions on Knowledge and Data Engineering,* Vol. 13(5):839-845, September / October 2001, pp 839-845.
16. Agrawal, Charu C. and Philip S. Yu.(2001). A New Approach to Online Generation of Association Rules. *IEEE Transactions on Knowledge and Data Engineering,* Vol. **13(**4) :527-540., July / August 2001.
17. Das, A. K., D. K. Bhattacharyya(2003). "Efficient Rule mining: Dynamic Database," in *Proceedings of Conference ITPC* 2003, vol. 1, Nepal, May 23 – 26, 2003.
18. Zhang; Yu-Fang , Jia-Li Mao; Zhong-Yang Xiong(2003). An efficient clustering algorithm Machine Learning and Cybernetics, *2003 International Conference* Vol. 1:261 -265, 2-5 Nov. 2003 .

19. Janujaz , E ., Kriegel , H.-P., Peifle, M(2004). Dbdc :Density based Distributed Clustering in *Proc Intl. Conf Extending Database Technology (EDBT)* 2004,.pp 88-105

20. Santhanam, S.(2004). "Weather Data Mining Using Independent Component Analysis"; *The Journal of Machine Learning Research,* 5, pp.239-253.

21. Amini, Amineh, Teh Ying Wah(2011). Density Micro-Clustering Algorithms on Data Streams *Proceedings of International Multi Conference of Engineer and Computer Scientists* 2011, Vol. I, March16-18,2011, Hong Kong.

22. Zhongnan, Zhang, Weili Wu and Yaochun Huang(2008). "Mining Dynamic Interdimension Association Rules for Local-Scale Weather Prediction". In *Proceedings of the 28th Annual International Computer Software and Applications Conference* -Workshops and Fast Abstracts -(COMPSAC'04) –Vol.(2): 146-149.