

Mining the Future: Predicting Itemsets' Support of Association Rules Mining

Shenoda Guirguis
Computer Science Dept, University of
Pittsburgh, Pittsburgh PA, USA.
shenoda@cs.pitt.edu

Khalil M. Ahmed, Nagwa M. El Makky,
Alaaeldin M. Hafez,
Computer Science Dept., Alexandria
University, Alexandria, Egypt

Abstract

This paper proposes a novel research dimension in the field of data mining, which is mining the future data before its arrival, or in other words: predicting association rules ahead before the arrival of the data. To achieve that, we need only predict the itemsets' support, upon which association rules could be easily produced. A time series analysis approach (MFTP) is proposed to perform itemsets' support prediction task. The proposed technique outperforms other prediction techniques for short history. The conducted performance study showed good prediction accuracy and response time. This, we provide a new tool to provide more information in the decision support field.

1. Introduction

Data mining analyzes huge data and extracts useful hidden knowledge in the data to support decision maker. In the famous basket-mining model, the mined association rules hints the super market owner (decision maker) how to group items to increase the sales. However, it is true in many applications that the item grouping changes over time and it is not necessary that the grouping that was valid in the last month will be valid in the next month. In this research, we propose to analyze and detect useful hidden knowledge in the data of the future ahead before its arrival, or in other words, to predict the mining rules of the future. For example, in the basket-mining model, the decision maker can run our proposed technique to detect the items grouping that is predicted to be valid in the following month, along with the items grouping that was valid in the past.

In the following subsection we define the problem statement. Then the rest of the paper is organized as follows. Section 2 provides information about the related work. Sections 3 and 4 discuss the proposed technique and the performance evaluation respectively. Then we conclude the results and provide some suggestions to extend this research in section 5.

1.2 Problem Statement

The problem the proposed technique solves can be stated as follows:

Definition 1: Given an incremental database environment, it is required to efficiently generate association rules for the most recent database increment, given a certain minimum support (minsup), and to predict the support of frequent itemsets of the next database increment ahead before its arrival. An incremental database environment is one where the database faces additions as well as deletions.

That is for an incremental database environment the itemsets' support values in the forthcoming database increment are predicted. Once the supports are predicted, mining rules could be easily deduced.

2. Related Work

One of the greatest challenges we faced when we started this research is that the research dimension of this paper is new; no previous work suggested mining the future. However, prediction is an old issue that is well studied. And since we abstract our problem to predict itemsets' support, we surveyed all prediction techniques and adopted the technique that suits the association mining context. We also surveyed the incremental association mining techniques, since we propose our technique to work in an incremental mining environment.

Many prediction approaches are applicable in this context; such as Neural Networks, Expert Systems, and Time Series analysis. Although expert systems are robust to new data, they are not general in the sense that it requires domain experts to write down domain specific rules (the knowledge base). Neural networks have proven to be excel-lent tools for forecasting. However, a certain neural network is designed to model and forecast, a certain dataset and requires a long history to allow training; the same limitation of statistical time series approaches.

A time series approach that requires no long history is the trend analysis approach which is simple and general solution. Therefore, a trend analysis approach is proposed. The proposed approach transforms the time series of each itemset into a string of trend indicators, and mines them for maximal-frequent-trend-pattern. It is therefore called the Maximal Frequent Trend Pattern (MFTP) algorithm.

Many incremental mining algorithms have appeared, such as “Fast Update”, “Difference Estimation for Large Itemsets”, “When to Update”, “Update with Early Pruning”, “BORDER”, and “Dynamic Data Mining”. Among all, Efficient Counting Using TIDLists (ECUT) [11] is one of the best known incremental algorithms. Therefore, the ECUT algorithm is proposed to be used.

3. The Proposed Technique

3.1 Definitions and Terminology

In this section, a set of definitions of terms that are used in the proposed technique is given. Before moving to the definitions, Table 1 below lists a set of symbols to be used hereafter.

Table 1: Symbols of the algorithm

Symbol	Definition
DB	The original database.
db	The database increment that has just arrived
DB'	The updated database = $DB \cup db$
L^x	Set of large itemset of a database x
$NBd(y)$	Negative border of the set of a large itemsets y
$minsup$	Minimum support ($0 \leq minsup \leq 1$)
$minhl$	System parameter: Minimum History Length; which is set by the system manager.

In the following set of definitions, let $I = \{i_1, i_2, \dots, i_m\}$ be a set of literals called items, and let DB be a database of transactions where each transaction $T \subseteq I$ and has a unique identifier (TID). Note: any subset of I is called an itemset.

Definition 2: An association rule R is an implication of the form $R: S_a \Rightarrow S_c$, where both S_a (rule antecedent) and S_c (rule consequent) are itemsets, and $S_a \cap S_c = \Phi$.

Definition 3: A rule $R: S_a \Rightarrow S_c$ has support s iff $s\%$ of the transactions in DB contain $S_a \cup S_c$. and an itemset S has support s iff $s\%$ of the transactions in DB contain S .

Definition 4: A rule $R: S_a \Rightarrow S_c$ has confidence c iff $c\%$ of the transactions containing S_a also contain $S_a \cup S_c$.

Definition 5: An itemset is large or frequent iff its support exceeds a certain predefined support threshold called minimum support ($minsup$)

Definition 6: Given a set of itemsets $L \subseteq P(R)$; where R is a set of items, and $P(R)$ is the power-set of R , and L is closed with respect to the set inclusion relation, the negative border set $NBd(L)$ of L consists of the minimal itemsets $X \in P(R)$ not in L [21]. Alternatively, $NBd(L) = P(R) - L$.

Definition 7: The History Log refers to the set of large itemsets and the set of negative border itemsets of each database increment (db_i) and the original database (DB).

Definition 8: The history length is the number of database increments that exists DB .

Definition 9: The moving average is a simple mathematical technique used primarily to eliminate aberrations and reveal the real trend in a collection of data points.

There are five popular types of moving averages: simple, exponential, triangular, variable, and weighted. The only significant difference between them is the weight assigned to the most recent data. In the proposed technique, a triangular moving average is applied to smooth the data before detecting the trends. In a series of values $\{x_1, x_2, \dots, x_n\}$, the triangular moving average for value number i (x'_i) is computed as:

$$x'_i = \frac{x_{i-1} + 2x_i + x_{i+1}}{4} \quad (1)$$

Definition 10: The Support Trend Range (STR) is a range of support values defined for each trend indicator, as given in definition 11.

Definition 11: Trends are the identification of value movements on individual time series sequences. Trend Indicators are calculated by relating the current value of a phenomenon to its previous value. If the support value of a certain itemset changed from a certain value x_i at database increment number i , to a value x_{i+1} at database increment number $i+1$, and x_{i+1} lays within the range of a certain trend indicator (TI), then it said that this support transition is of trend TI . A trend scale is used to determine the size of the space used to choose the values of the trend indicators.

In the proposed technique a trend scale of “4” is used, which means that the trend indicators can take the values $\{-2, -1, 0, 1, 2\}$. That is for a support transition –of a certain itemset– from s_i to s_{i+1} , the trend indicator takes the values $\{-2, -1, 0, 1, 2\}$ defined as follows:

$$\begin{aligned} TI_i = 2 &\Rightarrow s_{i+1} \text{ is in a two steps higher range than } s_i \\ TI_i = 1 &\Rightarrow s_{i+1} \text{ is in a one step higher range than } s_i \\ TI_i = 0 &\Rightarrow s_{i+1} \text{ is in the same range as } s_i \\ TI_i = -1 &\Rightarrow s_{i+1} \text{ is in a one step lower range than } s_i \\ TI_i = -2 &\Rightarrow s_{i+1} \text{ is in a two step lower range than } s_i \end{aligned}$$

Specifying whether the trend is “0” or so, is related to the STR size, or alternatively STR . Therefore, several

experiments were conducted to recommend how to define the *STR*.

Definition 12: A trend-pattern is a string of trend-indicators. A frequent-trend-pattern is a sub-pattern that appears frequently in the trend-pattern. A maximal-frequent-trend-pattern is the longest frequent-trend-pattern found in the time series.

3.2 The ECUT and MFTP algorithms

The proposed technique is divided into two components; the incremental data mining component, and the prediction component as shown in Figure 1.

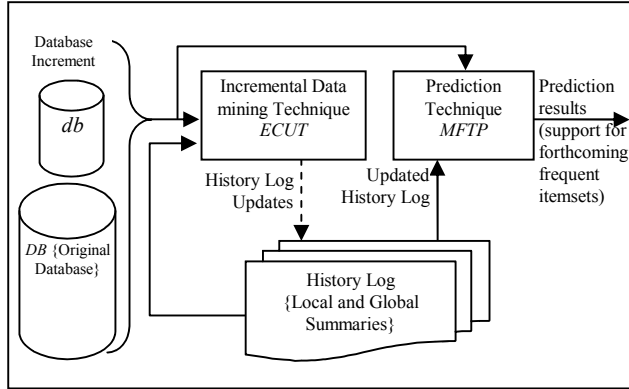


Figure 1: Block Diagram of the Proposed Technique

Upon arrival of a new database increment (*db*), the technique goes as follows: the ECUT is first called to update the history log, then MFTP is called for each itemset that is large or belongs to the negative border, as shown in Figure 2.

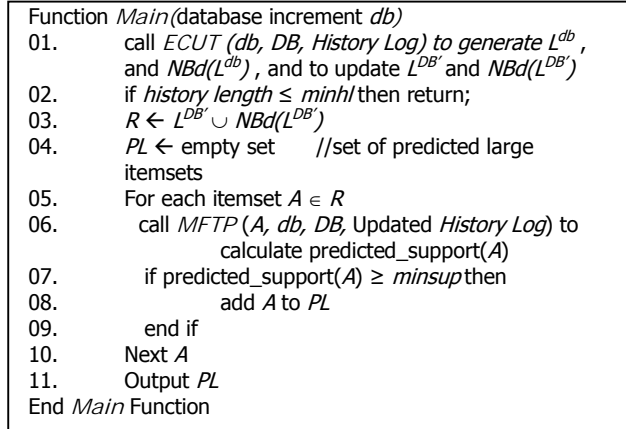


Figure 2: Main function.

The *ECUT* algorithm (Figure 3) is applied to compute the set of large itemsets and the negative border of *db* (L^{db} and $NBd(L^{db})$) and DB' ($L^{DB'}$ and $NBd(L^{DB'})$).

The *ECUT* was modified to first generates the L^{db} and $NBd(L^{db})$ to add them to the history log. Then, it proceeds normally updating the support of all itemsets of L^{DB} and $NBd(L^{DB})$ against the database increment. All itemsets that are found to be large are saved and added to the $L^{DB'}$. All itemsets that were in $NBd(L^{DB})$ and became large are added to a *winners* set. This *winners* set is used with $L^{DB'}$ to generate the *candidates* set. This *candidates* set is

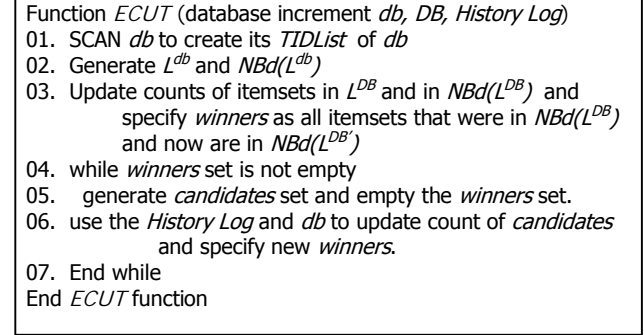


Figure 3: ECUT function.

filtered against $NBd(L^{DB'})$, based on the theory proved in [11] that a winner itemset should have at least one subset that belongs to $NBd(L^{DB'})$. After filtering the *candidates* set, new *winners* and new *candidates* are generated and so on till no more *winners* could be generated.

For each itemset in $L^{DB'}$ and $NBd(L^{DB'})$, the *MFTP* algorithm (Figure 4) constructs the time series, from the history log, applies the triangular moving average to smooth the series, and then transforms it into a trend pattern, which is mined for maximal frequent trend pattern. The maximal frequent trend pattern that matches the current pattern is then used to predict the forthcoming trend, and hence, the forthcoming support range. To match a frequent pattern of length *l* with the current pattern, a match-factor is used. That is if (*match-factor*)% of the first *l*-1 indicators of the frequent pattern matches their corresponding indicators in the last *l*-1 indicators of the whole pattern, then they are matched and the *l*th indicator is the predicted trend.

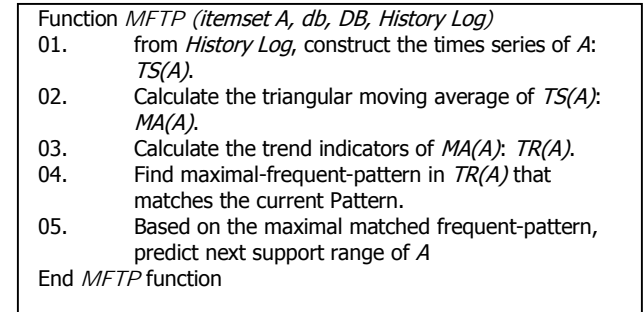


Figure 4: MFTP function.

4. Performance Study

4.1 Implementation, Performance Indexes, and Error measures

Visual C++ 6.0 was used in implementation. The validity of the implemented *ECUT* algorithm was tested by comparing the results with that of a brute force Apriori program downloadable from Christian Brogelt's web-pages [5]. The performance indexes are the response time and the accuracy. The error measures defined following.

4.1.1. Error Measures. The proposed algorithm predicts a range of support values, within which the actual value is believed to be. Therefore, the mid point of the predicted range is used to compare to the actual value. If the predicted range is $[v_1, v_2]$, then the predicted value is:

$$predicted = (v_1 + v_2) / 2 \quad (2)$$

Hence, the Mean Squared Error (*MSE*) is calculated by the formula (where "N" is the number of predictions):

$$MSE = \sum (actual - predicted)^2 / N \quad (3)$$

While the Normalized Mean Squared Error (*NMSE*) is calculated by the formula:

$$NMSE = MSE / \sigma^2 \quad (4)$$

Where " σ^2 " is the variance of the actual itemset's support. Similarly the Root Mean Squared Error (*RMSE*) and Normalized Root Mean Square Errors where used:

$$RMSE = \sqrt{MSE} \quad (5)$$

$$NRMSE = RMSE / \sigma \quad (6)$$

The Mean Absolute Percentage Error (*MAPE*) is calculated by the formula:

$$MAPE = \frac{100}{N} \sum \frac{Absolute(actual - predicted)}{actual} \quad (7)$$

4.2 Experimental Environment

The algorithm was implemented and tested on an IBM compatible PC (Pentium II®) with 1.2 GHz processor, 128 MB main memory, and 512 KB cache memory. The program was the only major job running on the machine throughout all experiments. Kernel and background processes were occupying 1% of the CPU capacity, and 9% of the main physical memory. Synthetic data were used as test data, having properties chosen to assimilate a reasonable retailing environment. The dataset properties are:

- Total number of transactions = 199,947 transaction.
- Average transaction length = 10 items.
- Number of items = 100 items.

The test data was generated using a program developed in IBM Almaden research center. It is available from the IBM quest website. The database was divided manually, by splitting the generated file, into 200 database increments.

4.3 The Complexity of the *MFTP* Algorithm

The time complexity of the *MFTP* algorithm is studied as a function of the History Length "N". The *MFTP* first goes through the database increments retrieving the support values, smoothing the series using the moving average, calculate differences and derives the trends. These steps are $O(N)$ operations. Then, the algorithm mines the trend pattern for maximal frequent trend pattern.

In best case, the algorithm would find a matched frequent trend pattern of length $N/2$ in first test, i.e. $O(N)$ comparisons. In worst case, the algorithm would fail, i.e. the algorithm will check all pattern lengths and will not find a match. The algorithm starts from length $N/2$ and decreases this length by one each time. For each length k , the algorithm checks $(N-2k)$ different patterns whether they match the current trend pattern of length k , and each check of two trend patterns requires k different comparisons. Thus, the total number of checked patterns in worst case (*TN*) is:

$$\begin{aligned} TN &= \sum_{k=1}^{\frac{N}{2}} k(N-2k) \\ &= \sum_{k=1}^{\frac{N}{2}} Nk - 2 \sum_{k=1}^{\frac{N}{2}} k^2 \\ &= N \times O(N^2) - 2 \times O(N^3) = O(N^3) \end{aligned}$$

That is the *MFTP* is of complexity $O(N)$ in best case, and of complexity $O(N^3)$ in worst case. Figure 5 shows the response time of the *MFTP* algorithm against the history length; the relation is linear. This means that the worst case occurs rarely. It is also noted that large drops in the response time occurs randomly when the algorithm be lucky and finds a matched frequent trend pattern very early.

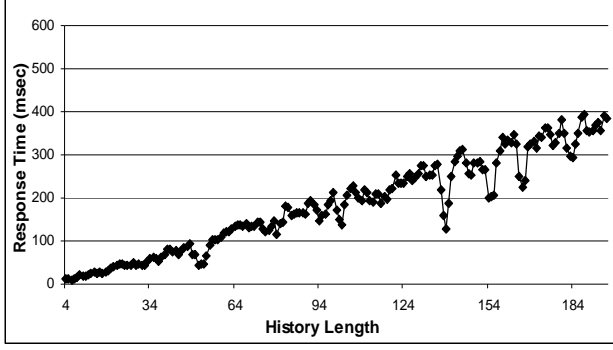


Figure 5: The MFTP Response Time versus the History Length.

4.4 Performance Analysis of the MFTP Algorithm

4.4.1. Varying the Support Trend Range of the MFTP.

The Support Trend Range (STR) is calculated for each itemset depending on features of its time series; to achieve most accurate possible prediction. Different schemes for calculating the STR were tested. In all the cases, for the time series formed by the support values $S = \{s_i; i=1, 2, \dots, \text{history length}\}$, the support transition Δ_i is defined as:

$$\Delta_i = s_{i+1} - s_i \quad i=1, 2, \dots, \text{history length}-1 \quad (8)$$

The Trend Indicator (TI_i) takes the values $\{-2, -1, 0, 1, 2\}$ as defined in section 2.2. In first experiment, the range of the support transitions was used as the STR ; i.e.

$$STR = \max_i (\Delta_i) - \min_i (\Delta_i) \quad (9)$$

The results were good for the frequent itemset case and better for the infrequent itemset case. Then, the standard deviation of the support transitions was used, that is.

$$STR = \sigma_{\Delta} = \sqrt{\frac{\sum_{i=1}^{\text{historylength}-1} (\Delta_i - \bar{\Delta})^2}{(\text{historylength}-2)}} \quad (10)$$

The trend indicator (TI_i) for the movement of the support of a certain itemset from s_i to s_{i+1} , is calculated as follows:

$$TI_i = \begin{cases} 2 & \text{if } \Delta_i > 2 \sigma_{\Delta} \\ 1 & \text{if } \sigma_{\Delta} < \Delta_i < 2 \sigma_{\Delta} \\ 0 & \text{if } -\sigma_{\Delta} < \Delta_i < \sigma_{\Delta} \\ -1 & \text{if } -2 \sigma_{\Delta} < \Delta_i < -\sigma_{\Delta} \\ -2 & \text{if } \Delta_i < -2 \sigma_{\Delta} \end{cases}$$

The results are shown in figures 6 and 7, and in tables 2 and 3. For an infrequent itemset case the performance was similar to that when the STR was the range of differences while it improved for the frequent case.

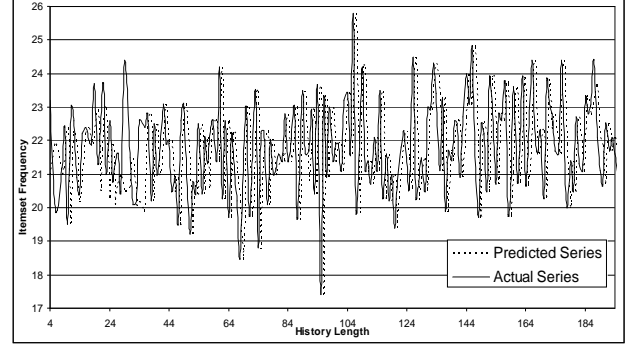


Figure 6: The frequent itemset case with STR is the standard deviation of differences

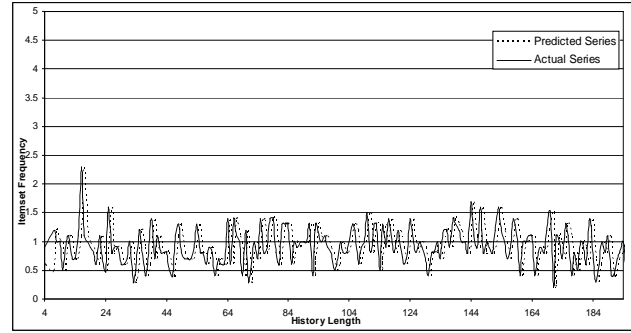


Figure 7: The infrequent itemset case with STR is the standard deviation of differences

Table 2: Errors when STR is the standard deviation of differences: frequent itemset case

MSE	NMSE	RMSE	NRMSE	MAPE
4.05049	2.25350	2.01258	1.50116	7.6198%

Table 3: Errors when STR is the standard deviation of differences: infrequent itemset case

MSE	NMSE	RMSE	NRMSE	MAPE
0.20528	2.0149	0.45308	1.419474	46.723%

4.4.2. Comparing the Proposed MFTP Algorithm to Statistical Time Series Analysis Methods (Box-Jenkins Method) on Empirical Data. Empirical datasets were used to compare the performance of the proposed $MFTP$ algorithm to that of the Box-Jenkins method, a standard and very powerful statistical time series analysis method, using SPSS tool.

a. US Quarterly Unemployment Rates Dataset (between 1948-1978): downloadable from <http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/labour.html>. The Box-Jenkins model found to best model this dataset in [16] was an AR(2) model, which was suggested

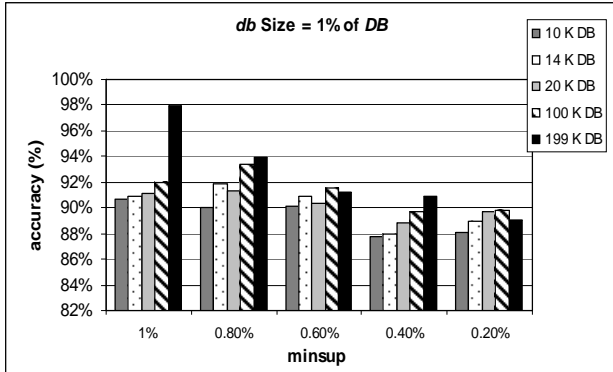


Figure 8: Percentage of valid predicted large itemsets versus DB size.

to SPSS program, without parameters values, leaving SPSS to calculate them. The proposed method outperformed the Box-Jenkins method for history length of 120 points and 30 points. For short history lengths, less than 30 points, the Box-Jenkins method failed to predict.

b. Daily Readings of the Viscosity of a Chemical Product Dataset: downloadable from <http://www.math.mun.ca/~aoyet/stat3540.html>. Similarly, an AR(2) model was used. The Box-Jenkins method slightly outperformed *MFTP* for history length of 94 points, while *MFTP* outperformed it for history length of 35 points. However, Box-Jenkins method failed to predict for short history lengths, less than 35 points.

4.5 Performance Analysis of the Entire Proposed Technique

4.5.1. Validation of the Implemented *ECUT*. To validate the implemented *ECUT*, its output, the set of large itemsets, was compared to that of the Apriori algorithm (implemented by Christian Brogelts, is available on the web [5]) [because no outputs (in terms of set of large itemsets and negative border) were available for the *ECUT* algorithm in the literature]. The same large itemsets were generated by both algorithms for randomly selected sizes of the original database.

4.5.2. Accuracy of the Entire Proposed Technique. The accuracy is measured as the percentage of validly predicted large itemsets to the actual ones, and the percentage of invalidly predicted large itemsets. In Figure 8, the accuracy is studied as DB size, and the *minsup* changes, while fixing the percentage of *db* to 1% of *DB*, and it was high; between 86% and 98%. The accuracy increases as the *minsup* increases, because the sensitivity

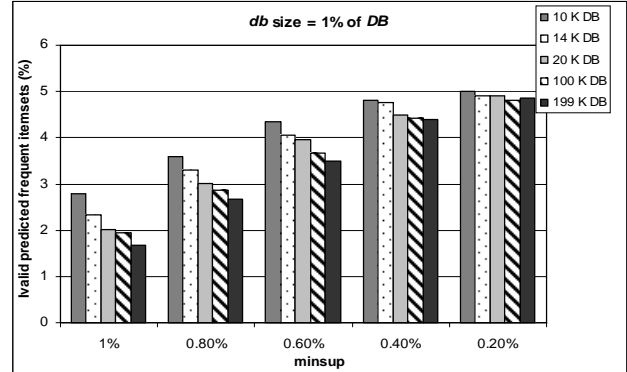


Figure 9: Percentage of invalid predicted large itemsets versus DB size.

to errors increases at smaller *minsup* values, and increases as the *DB* size increases. On the other hand, Figure 9 shows the percentage of the invalid predictions, for the same variations of parameters. Low percentage, between 1.5% and 5%, were found and it increases as the *minsup* decreases, and decreases as the *DB* size increases.

4.5.3. Error Measures of the Proposed Technique versus the History Length. To specify the minimum history length (*minhl*) after which *MFTP* can be used, the error measures were studied as the history length grows, for frequent and infrequent itemset cases. Figures 10 and 11 show the error measures of the frequent and infrequent itemset cases, respectively.

In this experiment, the *STR* was the standard deviation of the differences between support values. For the frequent itemset case, the error measures stabilize when the history length is 25. For the infrequent itemset case, all error measures, except the NMSE, stabilize when the history length is 30, while the NMSE stabilizes when the history length is 70. These results recommend a *minhl* between 25 and 30 database increments, for datasets with characteristics that are similar to the dataset used in experiments.

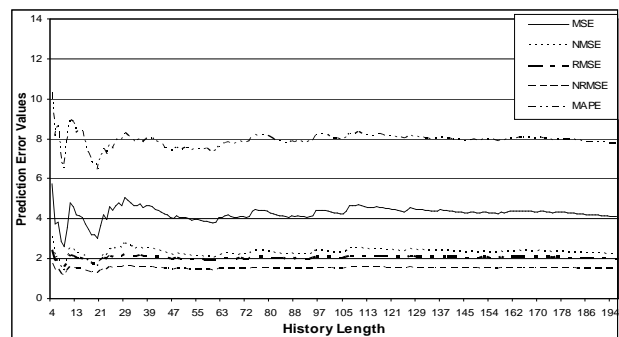


Figure 10: Prediction error measures versus history length: The frequent itemset case

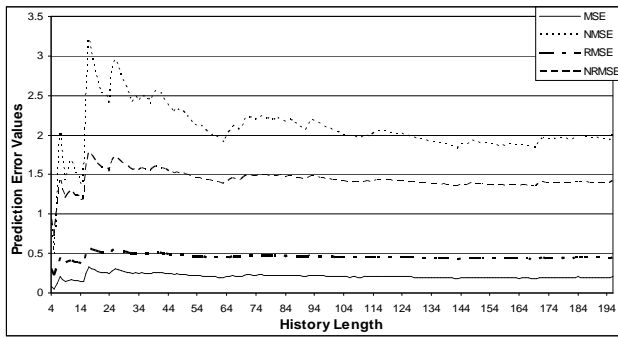


Figure 11: Prediction error measures versus history length: The infrequent itemset case

5. Conclusions and Future Work

This paper proposes a new technique to predict the frequent itemsets' support in an incremental database environment, which applies a time series analysis algorithm (*MFTP*) for prediction, and (*ECUT*) for the incremental mining. The experiments showed that *MFTP* runs in $O(N)$, where N is the history length, using the standard deviation of the support transitions as the support trend range achieves best prediction accuracy, and that *MFTP* accuracy is high compared to statistical forecasting methods for short history lengths. *MFTP* also showed high performance in terms of response time and percentage of actual large itemsets that were predicted to be large. Experiments recommended using *minhl* of 25 database increments to achieve reliable results for datasets with similar characteristics of the used dataset.

Suggested future research extensions. It is needed to establish a mechanism to automatically adjust the moving average weights and to examine other approaches of prediction such as Bayesian modeling and compare it to the proposed *MFTP* algorithm. Then, implementing a mining dependency between time series algorithm, which enables taking into consideration, when predicting, the effect of external time series (out of business scope) on the time series of the itemsets and the addition of a new degree of freedom, namely, the data span dimension, proposed in [11]. This allows user-defined selections of a temporal subset of the database to be used in the mining process. Finally is to integrate this work with a real life application such as Bioinformatics.

6. References

- [1] Abraham A., Nath B., & Mahanti P K, "Hybrid Intelligent Systems for Stock Market Analysis", 2001.
- [2] Abraham A., Sajith N. P., & Saratchandran P., "Modeling Chaotic Behavior of Stock Indices Using Intelligent Paradigms", 2003.
- [3] Ayad A., El-Makky N., & Taha Y., "A New Algorithm for Incremental Mining of Constrained Association", *SIAM'01*, Chicago USA, 2001.
- [4] Ayan N.F., Tansel A. U., & Arkun M. E., "An Efficient Algorithm to Update Large Itemsets With Early Pruning", *Proceedings of SIGKDD'99*, San Diego, 1999.
- [5] Borgelt C, "*Christian Borgelt's Web-pages: Apriori - Association Rule Induction/Frequent Itemset Mining*", <http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html>, 2005.
- [6] Chen, M. S., Han, J., & Yu, P.S., "Data Mining: An Overview from a Databases Perspective", *IEEE Transactions on Knowledge and Data Engineering*, 1996.
- [7] Cohen P. R., "A Survey of the Eighth National Conference on Artificial Neural Intelligence: Pulling Together or Pulling Apart?", *AI Magazine*, 1992, Vol. 12, No. 1.
- [8] David W. C., Han J., Vincent T., & Wong C. Y., "Maintenance of Discovered Association Rules in large databases: An Incremental Updating Technique", *Proceedings of ICDE'96*, New Orleans, Louisiana, USA, 1996.
- [9] David W. C., Lee S.D., & Kao B., "A General Incremental Technique for Maintaining Discovered Association Rules", *In Proceedings of DASFAA'97*, Melbourne, Australia, 1997.
- [10] Fayyad U. M., Piatetsky-Shapiro G., Smyth P., & Uthurusamy R., "Advances in Knowledge Discovery and Data Mining", *AAAI/MIT Press*, 1996.
- [11] Ganti V., Gehrke J., & Ramakrishnan R., "DEMON: Mining and Monitoring Evolving Data", *Proceedings of ICDE'00*, San Diego, 2000.
- [12] George F. L. & William A. S., "*Artificial Intelligence: Structures and Strategies for Complex Problem Solving*", Addison Wesley Longman Inc. 3rd edition, 1997.
- [13] Hafez A. M., "Association Mining of Dependency Between Time Series", *Proceedings of SPIE Vol. 4384*, SPIE AeroSense, 2001.

- [14] Han J. & Kamber M., "Data Mining: Concepts and Techniques", *Academic Press*, Morgan Kaufmann Publishers, 2000.
- [15] Kashef R., "Negative Border with Partitioning Algorithm for Incremental Mining", M.Sc. Thesis, *The Arab Academy of Science and Technology*, Alexandria, 2004.
- [16] Laila M., "Neural Networks Time Series Forecasting With Applications", M.Sc. Thesis, *Faculty of Science*, Alexandria University, 2004.
- [17] Lee S.D., & David W. C., "Maintenance of Discovered Association Rules: When to Update?", *In Proceedings of the ACM-SIGMOD Workshop DMKD-97*, Tucson, Arizona, 1997.
- [18] Raghavan V. & Hafez A. M., "Dynamic Data Mining", *In Proceedings of the IEA/AIE*, 2000.
- [19] Sarda N. L. & Srinivas N. V., "An Adaptive Algorithm for Incremental Mining of Association Rules", *In Proceedings of DEXA Workshop'98*, pp. 240-245, 1998.
- [20] Sima J., Orponen P., & Antti-Poika T., "A Computational Taxonomy and Survey of Neural Network Models", *Neural Computation*, 2001.
- [21] Thomas S., Bodagala S., Alsabti K., & Ranka S., "An Efficient Algorithm for the Incremental Update of Association Rules in Large Databases", *In Proceedings of the KDD 97*, New Port Beach, California, 1997.
- [22] Toivonen H., "Sampling Large Databases for Association Rules", *VLDB'96*, pp. 134-145, Mumbai, 1996.