Mining Patterns in Long Sequential Data with Noise

Wei Wang
IBM T. J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
ww1@us.ibm.com

Jiong Yang
IBM T. J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
jiyang@us.ibm.com

Philip S. Yu
IBM T. J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
psyu@us.ibm.com

ABSTRACT

Pattern discovery in time series data has been a problem of great importance in many fields, e.g., computational biology, performance analysis, consumer behavior, etc. Recently, considerable amount of research has been carried out in this area. The facts that the input data is typically very large and noises may present in various formats pose great challenge to the mining process. Recently, we have made several new research advances in this area. In this paper, we present some of them. We will survey new models proposed to address different types of noises as well as scalable algorithms developed for efficiently mining patterns under each model.

1. INTRODUCTION

In many applications, data typically present in the form of sequence(s). This may be either a consequence of employing a natural temporal ordering among individual data (e.g., financial market data) or a result of complying to some inherent physical structure (e.g., protein sequences in chromosomes). The input sequence is usually very long, which demands high scalability of the pattern discovery process. As an important research direction of the data mining field, mining patterns in long sequential data has been widely studied during recent years, which includes but is not limited to the following application domains.

- *Bio-Medical Study.* Each human gene consists a sequence of (usually over a half million) amino acids. A certain combination of amino acids may uniquely define a specific aspect of the biological function or appearance of a cell. Identifying such kind of meaningful combinations plays a crucial role in understanding the fundamental of life towards a deeper level, which has been the goal of many biologists for many years. By viewing the chromosomes as sequences of amino acids, the above task can be transformed into mining sequential patterns that satisfying some user-specified criteria.

- *Performance Analysis.* Many system-monitoring applications involve collecting and analyzing attributes whose values evolve over time. Patterns of system state transition has been proved to be very useful in predicting system behavior from a recent state history and in preventing disastrous circumstances from occurring.

- *Client Profile.* User profiles can be built based on the discovered pattern on trace logs. Such knowledge can either be utilized to develop an optimal proxy caching scheme or be used to provide better market targeting tools.

As a matter of fact, noises exist in most applications, which adds considerable challenges to the pattern mining process simply because many important patterns may be concealed if the model employed fails to accommodate noises properly. Depending on the type of application and the user's interests, tolerable noises may present in different formats and hence require different models accordingly.

1. *Injection of noise.* A typical example of noise injection is that, a client may accidentally access some irrelevant web page by mistake when he/she surfs on the Internet. Such access entries may be regarded as random noises inserted in the long traces during the process of mining meaningful patterns from the collected trace logs.

2. *Over-population of uninteresting patterns.* Different symbols (or events) may occur at vastly different frequencies by nature. For example, the sales of lamps is typically much higher than that of big furniture. However, even though patterns involving less expansive goods (such as lamps) occur more frequently, they may be considered not very interesting if the occurrences of such patterns are within people's expectation. In contrast, unexpected patterns involving furniture, if any, may be of great interests (probably because of a considerably high profit margin) even though such patterns may have relatively small number of occurrences in the data. Unfortunately, the significance of an infrequent but interesting pattern can be easily diluted by the "over-populated" unwanted patterns. These unwanted patterns can be viewed as "noise" in a broad sense because their presence *obstructs* (to some extent) the discovery of interesting infrequent patterns.

To address above issues, powerful model(s) that suits each specific purpose is demanded to provide clear separation between useful patterns and noises, and efficient mining algorithms are also needed to make these new models applicable to long data sequences. As mentioned before, the sequence can be very long and can easily range to more than hundreds of millions of symbols. This may result in very long patterns that may contain thousands of symbols. Therefore, any pattern discovery algorithm has to scale well with respect to both the length of the input sequence and the length of potential patterns. In this paper, we will focus on each type of noise in a separate section and present some recent advances to meet the challenge.

2. INJECTION OF NOISES

In this section, we discuss some recent development aiming at handling noises in the form of injection. Periodicity detection on sequence data is a challenging problem of great importance in many real applications. Two models, namely *asynchronous patterns* [16] and *meta patterns* [17], are proposed recently to address the issues of accommodating insertion of random noise and characterizing change of behavior.

2.1 Asynchronous Patterns

Most previous research in mining periodic patterns assumed that the disturbance within a series of repetitions of a pattern, if any, would not result in the loss of synchronization of subsequent occurrences of the pattern with previous occurrences [10]. For example, "Joe Smith reads newspaper every morning" is a periodic pattern. Even though Joe might not read newspaper in the morning occasionally, this disturbance will not affect the fact that Joe reads newspaper in the morning of the subsequent days. In other words, disturbance is allowed only in terms of "missing occurrences" but not as general as any "insertion of random noise events". However, this assumption is often too restrictive since we may fail to detect some interesting pattern if some of its occurrences is misaligned due to inserted or deleted noise events. Consider the application of *inventory replenishment*. The history of inventory refill orders can be regarded as a symbol sequence. Assume that the time between two replenishments of cold medicine is a month normally. The refill order is filed at the beginning of each month before a major outbreak of flu which in turn causes an additional refill at the 3rd week. Afterwards, even though the replenishment frequency is back to once each month, the refill time shifts to the 3rd week of a month (not the beginning of the month any longer). Therefore, it would be desirable if the pattern can still be recognized when the disturbance is within some reasonable threshold. In addition, the system behavior may change over time. Some pattern may not be present all the time (but rather within some time interval). Therefore, it is important to mine periodic patterns that are significant within a subsequence of symbols which may contain disturbance of length up to a certain threshold.

In [16], we proposed a more flexible model — *Asynchronous Periodic Pattern*. Two parameters, namely *min_rep* and *max_dis*, are employed to qualify valid patterns and the symbol subsequence containing it, where this subsequence in turn can be viewed as a list of **valid segments** of perfect repetitions interleaved by disturbance. Each valid segment is required to be of at least *min_rep* contiguous repetitions of the pattern and the length of each piece of disturbance is allowed only up to *max_dis*. The intuition behind this is that a pattern needs to repeat itself at least a certain number of times to demonstrate its significance and periodicity. On the other hand, the disturbance between two valid segments has to be within some reasonable bound. Otherwise, it would be more appropriate to treat such disturbance as a signal of "change of system behavior" instead of random noise injected into some persistent behavior. The parameter *max_dis* acts as the boundary to separate these two phenomena. Obviously, the appropriate values of these two parameters are application dependent and need to be specified by the user. For patterns satisfying these two requirements, our model will return the subsequence with the maximum overall repetitions. Note that, due to the presence of disturbance, some subsequent valid segment may not be well synchronized with the previous ones. (Some position shifting occurs.) This in turn would impose a great challenge in the mining process.

A pattern may be partially filled to enable a more flexible model. For instance, (*cold_medi*, *, *, *) is a partial monthly pattern showing that the cold medicine is reordered on the first week of each month while the replenishment orders in the other three weeks do not have strong regularity. However, since we also allow the shifted occurrence of a valid segment, this flexible model poses a difficult problem to be solved. For a give pattern P, its associated valid segments may overlap. In order to find the valid subsequence with the most repetitions for P, we have to decide which valid segment and more specifically which portion of a valid segment should be selected. While it is relatively easy to find the set of valid segments for a given pattern, substantial difficulties lie on how to assemble these valid segments to form the longest valid subsequence. As shown in Figure 1, with $min\_rep = 3$, S_1, S_2, and S_3 are three valid segments of the pattern $P = (d_1, *, *)$. If we set $max\_dis = 3$, then X_1 is the longest subsequence before S_3 is considered, which in turn makes X_2 the longest one. If we only look at the symbol sequence up to position j without looking ahead in the sequence, it is very difficult to determine whether we should switch to S_2 to become X_1 or continue on S_1.

This indicates that we may need to track multiple ongoing subsequences simultaneously. Since the number of different assemblages (of valid segments) grows exponentially with increasing period length, the process to mine the longest subsequence becomes a daunting task (even for a very simple pattern such as $(d_1, *, *)$). To solve this problem, for a given pattern, an efficient algorithm is developed in [16] to identify subsequences that may be extended to become the longest one and organize them in such a way that the longest valid subsequence can be identified by a single scan of the input sequence and at any time only a small portion of all extendible subsequences needs to be examined.

Another innovation of the mining algorithm is that it can discover all periodic patterns regardless of the period length. Most previous research in this area focused on patterns for some pre-specified period length [10; 14] or some pre-defined calendar [15]. Unfortunately, in practice, the period is not always available a priori (It is also part of what needs to be mined out from the data). The stock of different merchandises may be replenished at different frequencies (which may be unknown ahead of time[1] and may also varies from time to time). A period may span over thousands of symbols in a long time series data or just a few symbols. A distance-

[1] The replenishment order of a merchandise may not be prescheduled but rather be filed whenever the inventory is low.

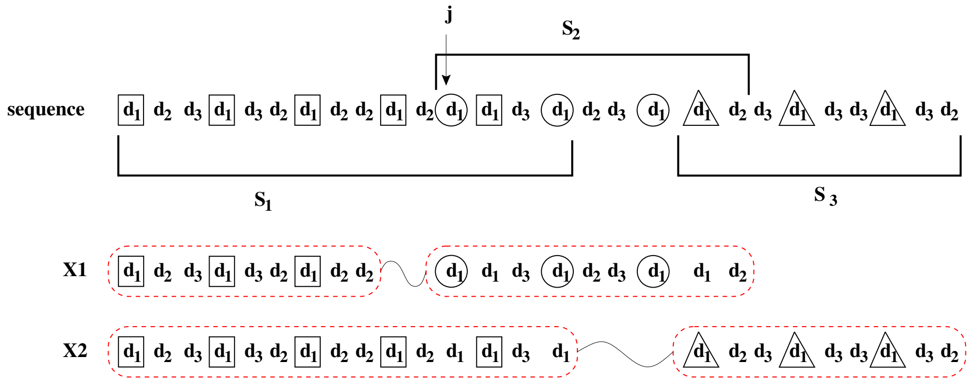

Figure 1: Example of Symbol Sequence

based pruning mechanism is first introduced to discover all possible periods and the set of symbols that are likely to appear in some pattern of each possible period. In order to find the longest valid subsequence for all possible patterns, a level-wise approach is employed. The Apriori property also holds on patterns of the same period. That is, a valid segment of a pattern is also a valid segment of any pattern with fewer symbols specified in the pattern. For example, a valid segment for $(d_1, d_2, *)$ will also be one for $(d_1, *, *)$. Then, for each likely period, all valid patterns with their longest supporting subsequences can be mined via an iterative process efficiently.

2.2 Meta Patterns

As we mentioned earlier, due to the changes of system behavior, some pattern may be only notable within a portion of the entire data sequence and different patterns may present at different places. The evolution among patterns may also follow some regularity. Such regularity, if any, would be of great value in understanding the nature of the system and building prediction models. Consider the application of *Internet user profile*. The sequence of web pages that a user accesses is often used to construct the user profile. An accurate profile is significant in many application domains including personalization and recommendation systems. During a period of time, a user may access some web sites repetitively. Such a behavior may be represented by a periodic pattern that can be put into user's profile. Moreover, a user's Internet access pattern may change over time. For instance, during a normal business day, one may surf financial web sites mostly when the stock market is open and may switch to sports oriented web sites for the rest of the day. At a coarser level, we may also find that such a pattern holds during weekdays whereas a total different pattern presents during weekends. Such a weekly pattern can be represented in the form of *meta-pattern* [17] which may take occurrences of patterns/meta-patterns (of lower granularity) as components. Because of the hierarchical nature of the meta-pattern, the concept of *level* is introduced to represent the "depth" of a meta-pattern. In contrast, we refer to the patterns that contain only raw event(s) is referred to as the *basic patterns* (or patterns of *level* 1), which may be viewed as special cases of meta-patterns.
However, most previous research in this area focused on mining patterns that only take basic events as their components and did not address the above issue. The meta-pattern proposed in [17] is a more general model for periodicity than any previous model. The recursive nature of meta-pattern not only can provide a more compact representation of complicated patterns but also can capture the hierarchies of pattern evolutions, which may not be expressible by previous models. As some tolerable noise is usually allowed within a series of pattern repetitions [16] to accommodate a certain degree of imperfectness, two portions (of a data sequence) where a pattern is notable may have different layout of pattern occurrences. As a result, there may not exist any common representation in terms of raw events. For example, two patterns $(a, b, *)$ and (b, c) appear in the sequence alternately in Figure 2(a). (Here, a pattern may be only partially filled and "*" is used to substitute the "don't care" position(s).) The length of each portion where $(a, b, *)$ is notable is 19 and each portion where (b, c) is notable contains 6 symbols. In addition, each gap between notable portions of $(a, b, *)$ and (b, c) consists of 2 positions while a three-position gap presents after each notable portion of (b, c). All of these can be represented by a meta-pattern of four components $((a, b, *) : [1, 19], * : [20, 21], (b, c) : [22, 27], * : [28, 30])$. The numbers in the brackets indicate the offset of the component within the meta-pattern. Let's take a closer look at those two portions where the pattern $(a, b, *)$ is notable: one is from position 1 to 19 and the other is from position 31 to 49. Note that both portions contain some noise that impairs the perfectness on repetition of $(a, b, *)$. Neither of them can match a single basic pattern format (i.e., $(a, b, *, a, b, *, a, b, *, a, b, *, a, b, *, a, b, *)$). Since the locations and durations of the noise are different in these two portions, $(a, b, *, a, b, *, a, b, *, *, *, *, *, a, b, *, a, b, *)$ and $(a, b, *, a, b, *, a, b, *, *, a, b, *, a, b, *, a, b, *)$ do not match with each other. In general, the noise could occur anywhere, be of various duration, and even occur multiple times within the portion where a pattern is notable as long as the noise is below some threshold. Even though the allowance of noise plays a positive role in characterizing system behavior in a noisy environment, it prevents such a meta-pattern from being represented in the form of an (equivalent) basic pattern. The model of meta-pattern is therefore proposed [17] to provide a more powerful means to periodicity representation.
However, the flexibility of meta-pattern poses serious challenges in the discovery process, which may not be encountered in mining basic patterns.

- While a basic pattern has two degrees of freedom: the period (i.e., the number of components in the pattern) and the choice of symbol/event for each component, a meta-pattern has an additional degree of freedom: the length of each component in the pattern. It is incurred by the fact that a component may occupy multiple positions. This extra degree of freedom would increase the number of potential meta-pattern candidates dramatically.

- Many patterns/meta-patterns may collocate or overlap for any given portion of a sequence. For example, both of $(a, b, a, *)$ and $(a, *)$ are valid within the subsequence shown in Figure 2(b). As a result, during the meta-pattern mining process, there could be a large number of candidates for each component of a (higher level) meta-pattern. This also aggravates the mining complexities.

Therefore, how to identify the "proper" candidate meta-patterns is very crucial to the overall efficiency of the mining process. To tackle this problem, a so called *component property* is identified and, in addition to the traditionally used Apriori property, to prune the search space. This is inspired by the observation that a pattern may participate in a meta-pattern only if its notable portions exhibit a certain cyclic behavior. A *segment based* algorithm is devised to identify the potential period of a meta-pattern and for each component of a possible period, the potential pattern candidate(s) and its length within the meta-pattern. The set of all meta-patterns can be categorized according to their structures and are evaluated in a designed order illustrated in Figure 3. For example, the candidate patterns at level 1 with 1 non-eternal component is examined first. Based on these patterns, the candidate patterns at level 1 with two non-eternal components are generated and examined. This procedure continues until all candidate patterns at level 1 are examined. After that, the patterns at the second level is generated and examined in the same manner as the patterns in level 1. Next, the candidate patterns at level 3, 4, and so on are generated and examined successively. As a result, the pruning power provided by both properties can be fully utilized. The pruning effects of the Apriori property and the component property are indicated by solid arrows and dashed arrows, respectively.

3. OVER-POPULATION OF UNINTERESTING PATTERNS

In this section, we discuss some research achievement in mining interesting patterns involving infrequent event(s) in long data sequences. As we mentioned in the previous section, the support (number of occurrences) has been widely used as the metric to identify the significant patterns from the rest [9; 10]. A qualified pattern in the support model must occur sufficient number of times. In some applications, such a model is proved to be very meaningful and important. However, in some other applications, the number of occurrences may not represent the significance of a pattern. Consider the following examples.

- *Computational Biology.* A human being chromosome consists of a sequence of genes. Researchers are interested in gene expressions (i.e., a portion of gene sequence) which are statistically significant rather than gene expressions that occur frequently. The statistical significance of a gene expression is defined as how likely such a gene combination would occur in an equivalent random data set [5]. In other words, we want to find the gene expressions that have very low probability to occur by chance, but actually occur in some chromosome. It is obvious that a statistically significant gene expression may not necessarily occur frequently, thus, the support metric is not an appropriate measurement of the significance of a gene expression.

- *Web server load.* Consider a web server cluster consisting of 5 servers. The workload on each server is measured by 4 ranks: *low*, *relatively low*, *relatively high*, and *high*. Then there are $4^5 = 1024$ different events, one for each possible combination of server states. Some preliminary examination of the cluster states over time might show that the state fluctuation complies with some periodic behavior. Obtaining such knowledge would be very beneficial for understanding cluster's behavior and improving its performance. Although the high workload on all servers may occur at a much lower frequency than other states, patterns involving it may be of more interests to some system administrators.

- *Earthquake.* Earthquakes occur very often in California. It can be classified by its magnitude and type. Scientist may be interested in knowing whether there exists any inherent seismic period so that prediction can be made. Note that patterns involving big earthquake is much more valuable even though it occurs at a much lower frequency than smaller ones.

In the above examples, we can see that users may be interested in not only the patterns with high occurrences, but also the patterns that do not occur frequently by chance as well. A large number of occurrences of an "expected" frequent pattern sometimes may not be as interesting as a few occurrences of an "expected" rare pattern. The support model is not ideal for these applications because, in the support model, the occurrence of a pattern carries the same weight (i.e., 1) towards its significance, regardless of its likelihood of occurrences. Intuitively, in above applications, the occurrence of a rare pattern should accumulate more weights/significance than a frequent pattern. To address this problem, we propose a new model in [18] to characterize the *statistically significant* patterns instead of frequent patterns.

The significance metric of an occurrence of a pattern should have the following properties. (1) The significance of an occurrence is anti-monotonic with respect to the likelihood that a pattern may occur by chance (or by prior knowledge). (2) The metric should have some physical meaning, i.e., not arbitrary created. It is fortunate that the *information* metric [4] which is widely studied and used in the communication field can fulfill both requirements. Intuitively, information is a measurement of how likely a pattern will occur or the amount of "surprise" when a pattern actually occurs. If a pattern is expected to occur frequently based on some prior knowledge or by chance, then an occurrence of that pattern carries less information. Thus, we use information as the measure of significance for an occurrence of a pattern. The *information gain* metric is introduced to represent the

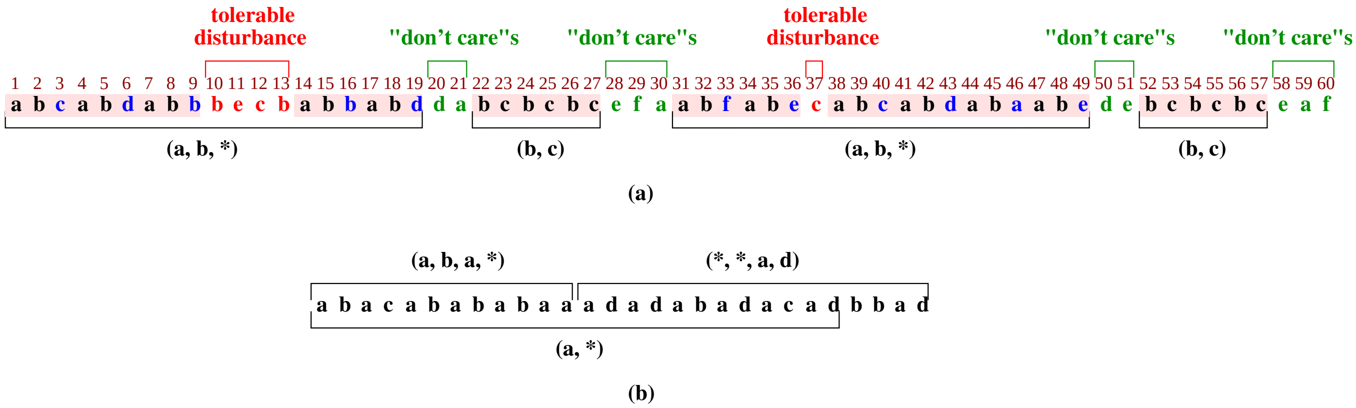

Figure 2: Meta Pattern

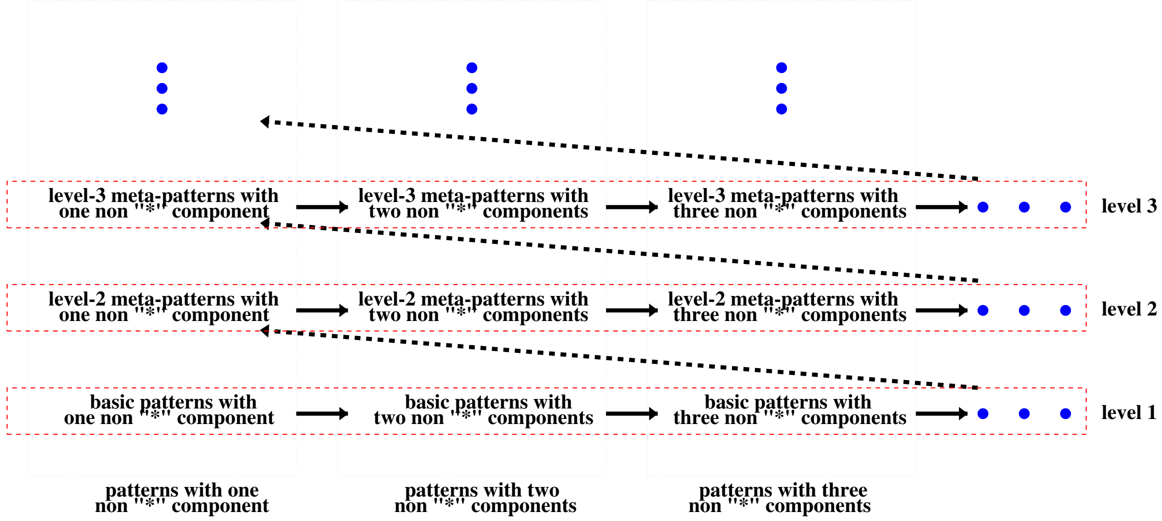

Figure 3: Pruning Directions

accumulated information of a pattern in an event sequence. We refer to this model as the *information model* in [18].

The information model is different from the support model. For a given minimum information gain threshold, let Ψ be the set of patterns that satisfy this threshold. Under the support model, in order to find all patterns in Ψ when event occurrence frequencies are vastly different, the minimum support threshold has to be set very low. A major problem could rise from this: too many patterns discovered. Table 1 shows a comparison between the support model and the information model. The example sequences are constructed from real traces. In Table 1, two traces (Scour and IBM traces) are used. Scour is a web search engine that is specialized for multimedia contents. The web URL of the Scour search engineer is "http://www.scour.net". The scour servers consist of a cluster of machines. Every five minutes, a distributed sensor records the CPU utilization of every node and the average is taken as the global CPU utilization level. We discretize the global CPU utilization level into events, event A stands for the utilization level between 0 and 0.05, B stands for the utilization level between 0.05 and 0.1, and so on. The scour trace consists of 28800 occurrences of 19 events (100 days). The event that corresponding to the utilization between 0.95 and 1 does not occur in the trace.

The IBM Intranet traces consist of 160 critical nodes, e.g., file servers, routers, etc., in the IBM T. J. Watson Intranet. Each node issues a message in response of certain situation, e.g., CPU saturation, router interface down, etc. There are total 20 types of messages. We treat a certain message from a particular node as a distinct event, thus there are total 500 distinct events in the trace because a certain type of node may only send out 4 or 5 types of messages. The IBM Intranet trace consists of 10,000 occurrences of the events. By applying Infomation model on this trace, we found some statistically significant patterns that are also interesting. For example, the pattern $(node_a\_fail, *, node_b\_saturated, *)$ has the eighth highest information gain. This pattern means that a short time after a router ($node_a$) fails, the CPU on another node ($node_b$) is saturated. Under a thorough investigation, we found that $node_b$ is a file server and after $node_a$ fails, all requests to some files are sent to $node_b$, thus causes the bottleneck.

In order to find the pattern with most information gain, the support threshold has to be set at 0.000234 and there are over 16,000 satisfied patterns in one sequence. It is obvious that the support threshold has to be set very low to

Table 1: Support threshold vs. information gain threshold

Number of Patterns	Scour Trace		IBM Trace	
Satisfied Info. Thresh.	Support Thresh.	Num. of satisfied patterns	Support Thresh.	Num. of satisfied patterns
1	0.000234	16,123	0.0035	637
10	0.000212	16,953	0.0031	711
100	0.000198	17,876	0.0024	987

discover a small number of patterns with high information gain. This means that the patterns with most information gain are buried in a sea of patterns with relatively low information gain. This could be a large burden for the end user to distinguish the significant patterns (i.e., patterns with most information gain) from the rest. In addition, since a large number of patterns has to be discovered, the support model may yield an inefficient algorithm.

Although the information gain is a more meaningful metric for the problems addressed previously, it does not preserve the *downward closure* property (as the *support* does). For example, the pattern (a_1, a_2) may have enough information gain while both $(a_1, *)$ and $(*, a_2)$ do not. We cannot take advantage of the standard pruning technique (e.g., Apriori algorithm) developed for mining association rules [1; 3; 11] and temporal patterns [2; 7; 13]. Our observation that the *subadditivity property* (where the information gain of (a_1, a_2) can not exceed the summation of that of $(a_1, *)$ and $(*, a_2)$) is still preserved by the information gain motivates us to devise a recursive algorithm as the core of our pattern discovery tool, InfoMiner [18]. More specifically, the InfoMiner uses a depth-first, projection-based approach that starts from the patterns with only one filled position and then proceeds to more complicated patterns gradually, and in the mean time utilizes the subadditivity property to continuously refine the candidate event list associated with each yet-open position in the pattern. It has been demonstrated in [17] that the response time of InfoMiner is linearly proportional to the length of the input sequence.

4. CONCLUSIONS

In this paper, we discuss several recent research advances for mining patterns in time series data given the presence of noise. They not only provide more suitable models to measure interesting patterns but also enable scalable solutions to mine patterns under the proposed models with respect to the size of input data. Interested readers are welcome to refer to the individual papers for further information.

5. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. 20th Int. Conf. on Very Large Data Bases*, 487-499, 1994.

[2] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proc. Int. Conf. on Data Engineering (ICDE)*, Taipei, Taiwan, 3-14, March 1995.

[3] R. J. Bayardo Jr. Efficiently mining long patterns from databases. *Proc. ACM SIGMOD Conf. on Management of Data*, 85-93, 1998.

[4] R. Blahut. *Principles and Practice of Information Theory*, Addison-Wesley Publishing Company, 1987.

[5] A. Califano, G. Stolovitzky, and Y. Tu. *Analysis of gene expression microarrays: a combinatorial multivariate approach*, IBM T. J. Watson Research Report, 1999.

[6] R. Durbin, S. Eddy, A. Krough, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[7] M. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: sequential pattern mining with regular expression constraints. *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, 223-234, 1999.

[8] National Center for Biotechnology Information. Available at "http://www.ncbi.nlm.nih.gov".

[9] J. Han, W. Gong, and Y. Yin. Mining segment-wise periodic patterns in time-related databases. *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, 214-218, 1998.

[10] J. Han, G. Dong, and Y. Yin. Efficient mining partial periodic patterns in time series database. *Proc. Int. Conf. on Data Engineering*, 106-115, 1999.

[11] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD)*, 1-12, 2000.

[12] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, vol 58, 13-30, 1963.

[13] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, vol. 1, no. 3, 259-289, 1997.

[14] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. *Proc. 14th Int. Conf. on Data Engineering*, 412-421, 1998.

[15] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. *Proc. 24th Intl. Conf. on Very Large Data Bases (VLDB)*, 368-379, 1998.

[16] J. Yang, W. Wang, and P. Yu. Mining asynchronous periodic patterns in time series data. *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 275-279, 2000.

[17] J. Yang, W. Yang, and P. Yu. Meta-patterns: revealing hierarchy of periodic patterns. *IBM Research Report*, 2001.

[18] J. Yang, W. Yang, and P. Yu. InfoMiner: mining significant periodic patterns with rare events in time series data. *IBM Research Report*, 2001.