THE DISCOVERY OF TOP-K DNA FREQUENT PATTERNS WITH APPROXIMATE METHOD

Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering and Knowledge Engineering Research Units, School of Computer Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand

nittaya@sut.ac.th, kittisakThailand@gmail.com

Abstract

Top-k frequent pattern discovery is indeed an association analysis concerning automatic extraction of the k most correlated and interesting patterns from large databases. Current studies in association mining concentrate on how to effectively find all objects that are frequently co-occurring. Given a set of objects with m features, there are almost 2^m frequent patterns to consider. For DNA data that are normally very high in dimensionality, frequent pattern discovery from genetic data is obviously a computationally expensive problem. We therefore devise an approximate approach to tackle this problem. We propose an approximate method based on the window sliding concept to estimate data density and obtain data characteristics from a small set of samples. Then we draw a set of representatives with reservoir sampling technique. These representatives are subsequently used in the main process of frequent pattern mining. Our designed algorithm had been implemented with the Erlang language, which is the functional programming paradigm with inherent support for pattern matching. The experimental results confirm the efficiency and reliability of our approximate method.

Keywords*: Top-k frequent patterns, Approximate method, DNA patterns, Window sliding, Reservoir sampling, Erlang language*

1. Introduction

Frequent pattern discovery is an essential operation for association analysis, which is the discovery process concerning an automatic extraction of interesting patterns and correlations from a large database. These patterns can reveal implicit relationships among set of objects (or items) that lead to the generation of association rules in a form of "if *antecedents* then *consequences*." These rules have the potential use in medical diagnosis, customer behavioural forecast, financial decision support, and many other applications. The process of finding all frequent itemsets in a database is computationally expensive because it involves the search for all item combinations. For a data set with high dimensionality such as the genetic data, finding only top-k frequent itemsets is more practical than searching for all itemsets that meet the minimum support threshold. Top-k frequent pattern discovery (Han *et al*., 2002) limits the search space to the k most frequently occurred patterns across the database.

In this paper, we study the top-k frequent pattern discovery in the data streaming scenario. The discovery of frequent patterns from a stream is considered a hard problem because of a continuously generated nature of stream that does not allow a revisit over passing data element. Moreover, the discovery process has been required to be fast to produce immediate results. From these requirements, we thus devise an approximate approach to solve the problem of top-k pattern discovery over continuous stream using the DNA data as an illustration. Our approximate algorithm is intended to be applied to process a stream prior to the pattern discovery process. The organization of this paper is as follows. After the literature review regarding association analysis and frequent pattern mining in section 2, we present our method in section 3. The experimental results are demonstrated in section 4. We conclude our paper in section 5 with the discussion of future research direction.

2. Literature Review

Since the introduction of the AIS (Agrawal-Imielinski-Swami) algorithm (Agrawal and Srikant, 1994b) by the three members of IBM Almaden Research Center in 1993 (Agrawal *et al*., 1993), the concept of association rule mining from transactional databases has received much interest from many data mining researchers. A year later, Rakesh Agrawal and Ramakrishnan Srikant (1994a; 1994b) improved the algorithm by reducing its search space with apriori property of the search through a frequent itemset lattice. This new algorithm has been named Apriori. The advent of Apriori algorithm is a major milestone of advancement in association analysis.

Apriori algorithm has been widely used as a basis for subsequent improvement proposed by a number of research teams. Park *et al*. (1995a) proposed to use hashing technique for the improvement of frequent itemset search. Han and Fu (1995) introduced the idea of discovering multiple levels of association rules. For a very large transactional database, Savasere *et al*. (1995) suggested to split the database and then search for associative relationships in a reduced data set. Toivonen (1996) tackled the large database problem with a sampling idea to search for interesting association from data representatives. Cheung *et al*. (1996a) considered an incremental approach for gradually learning of association among itemsets. Parallel computation is another mainstream of research to speed up association rule mining (Park *et al*., 1995b; Agrawal and Shafer, 1996; Zaki *et al*., 1997).

For a non-Apriori based association mining algorithm, the FP-growth algorithm that uses a tree structure to store frequent itemsets is an efficient method for extracting frequent patterns. The algorithm had been proposed by Han *et al*. (2000) and gained popularity since then (Agrawal *et al*., 2001; Pei *et al*., 2001; Liu *et al*., 2002; Grahne and Zhu, 2003).

In the emerging era of cloud technology, distributed computation of frequent patterns can be effectively accomplished. The research along this line has started since the last two decades (Cheung *et al*., 1996b) and it is still an active research area (Coenen and Leng, 2006; Tseng *et al*., 2010; Zhu *et al*., 2011; Lin *et al*., 2013; Cuzzocrea *et al*., 2014; Elayyadi *et al*., 2014).

With the advanced mobile devices, data collection and broadcasting occur at a very high speed. The frequent pattern discovery algorithms have to deal with the new kind of data, i.e., streaming data. A data stream is a sequence of digitally encoded data that are continuously transmitted from distributed sources (Guha *et al*., 2001; Babcock *et al*., 2002; Gaber *et al*., 2005; Jiang and Gruenwald, 2006). Kargupta *et al*. (2004) developed the VEDAS system to monitor vehicles at real time. Cai *et al*. (2004) designed the MAIDS system to mine incidents from data streams. Halatchev and Gruenwald (2005) proposed an estimation technique to guess missing values in sensor data streams. Finding frequent itemsets over data stream is a research problem studied by several researchers (Chang and Lee, 2004; Charikar *et al*., 2004; Chi *et al*., 2004; Gaber *et al*., 2004; Ghoting and Parthasarathy, 2004; Li *et al*., 2004; Teng *et al*., 2004; Yu *et al*., 2004; Lin *et al*., 2005; Mao *et al*., 2005).

The work presented in this paper is also along the line of distributed data stream processing to find the top-k patterns from DNA data. To estimate the frequency of top-k patterns, we adapted the Monte Carlo approximate method (Kerdprasop *et al*., 2006). The details of our design will be discussed in the next section.

3. Approximate Method for Top-k Pattern Discovery

A framework of our approximate top-k frequent pattern discovery is presented in figure 1. Contribution of our work is the design and implementation of the approximation-via-sliding-window (figure 2) and density-biased-sampling (figure 3) algorithms, whereas the frequent pattern discovery is Apriori-based algorithm (Agrawal and Srikant, 1994). Our sampling technique is based on the reservoir concept (Vitter, 1985; Kerdprasop et al., 2005), but data representatives will be drawn only from the dense area. Thresholds for minimum density and area size can be adjusted by user.

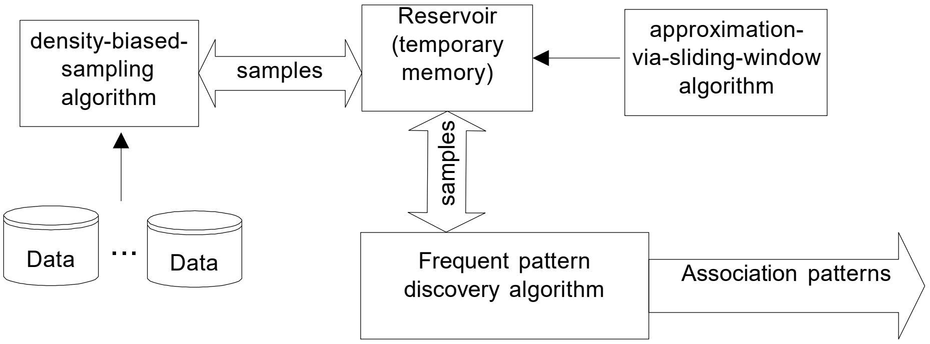

density-biased-sampling algorithm
samples
Reservoir (temporary memory)
approximation-via-sliding-window algorithm
samples
Data
...
Data
Frequent pattern discovery algorithm
Association patterns

Figure 1. A framework of approximate method for top-k pattern discovery

```
Input:    a set of data points represented as vectors
Output: a new set of transformed data points annotated with density value

% Initialize windows
Interact with user to obtain dimension value
Generate window grid of size W along dimension axes
% Count density
Sequential move on each window and count number of data points, N, in the window
Record a list of window's central point and its N value in a file F
Return F as a set of transformed data
```

Figure 2. Pseudocode of the approximation-via-sliding-window algorithm

```
Input:    a set of high density data from the approximation-via-sliding-window algorithm
Output: a new set of data samples

(1)   Extract data from a condense form and obtain a desired sampling choice from user
(2)   If choice = 'Density-biased Reservoir+Hashing', then
(3)         Interactive with user to obtain reservoir size
(4)         Hash each data point to store in a reservoir R
(5)         If collision occurs, then stored data item is replaced by a new one
(6)         Repeat steps 4-5 until there is no more data point, and return R as output
(7)   If choice = 'Density-biased Reservoir+Simple Random Sampling', then
(8)         Interact with user to obtain the bin size
(9)         Randomly select data point to store in a reservoir R  //sampling without replacement
(10)        Repeat step 9 until R is full, and return R as an output
(11)  If choice = 'Density-biased Reservoir+Rejection Sampling', then
(12)        Interact with user to obtain the bin size and interval I, I  [0.0..0.5]
(13)        Randomly select data point D                       // sampling without replacement
(14)        Generate a uniform random number U from the range [0.0 .. 1.0]
(15)         If U is within the range [0.5-I .. 0.5+I], then store D in R
(16)                Otherwise, reject and discard D
(17)         Repeat steps 13-16 until R is full, and return R as an output
```

Figure 3. Pseudocode of the density-biased-sampling algorithm

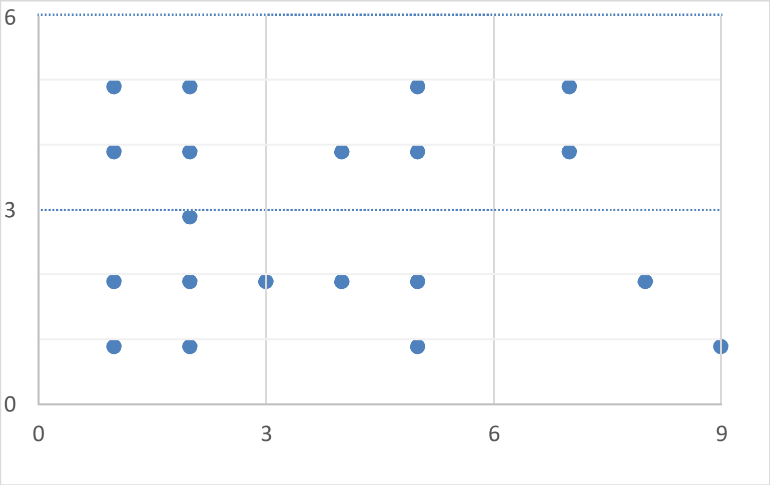


Figure 4. Twenty data points distributed within six windows of size 33

Our density-biased sampling technique (an algorithm in figure 2) has been designed to handle streaming in which input data are continuously processed by the system. To analyse each and every data item is almost impossible. We thus instead consider frequent patterns from the representatives. The intuitive idea of selecting representative data with the approximation- via-sliding-window algorithm can be demonstrated through a simple situation of processing a two-dimensional data set containing 20 data points, which are shown in figure 4. For the purpose of concise demonstration, we assume that the data points in this example limit themselves within the scale 9x6 along the horizontal and vertical axes, respectively.

The first step of a stream data density estimation is to decide the size of small grids, which we call windows in our algorithm. Suppose we choose the size 33. The boundaries of each window can be listed with intervals in the <x,y> coordinates as follows (note that the interval such as [0,3) represents the values ranging from zero up to 3, but does not include 3) :

Range along <x,y> axes   Range along <x,y> axes
window :   < [0,3), [0,3) >   window :   < [0,3), [3,6] > window :
< [3,6), [0,3) >   window :   < [3,6), [3,6] > window :   < [6,9],
[0,3) >   window :   < [6,9], [3,6] >

Data points in each window will be counted and condensed to the representation format that consumes less memory. The condensed form is per window, instead of per data point. In this condensed form, we store the central location of a window together with the number of data points existing in that window. For instance, all five data points in window will be packed and stored as { <1.5,4.5> , 5 }, where <1.5,4.5> is the central point of this window. All 20 data points will be transformed as shown in Figure 5. These transformed data points that meet the minimum density requirement are the output of the approximation-via-sliding-window algorithm, and also the input for the density-biased sampling algorithm.

	Raw data			Transformed data		Output
<1,1>	<2,4>	<5,4>		{ <1.5,1.5>, 4 }		
<1,1>	<2,5>	<5,5>	*window*	{ <4.5,1.5>, 4 }	*density*	{ <1.5,1.5>, 4 }
<1,4>	<3,2>	<7,4>	*size=33*	{ <7.5,1.5>, 2 }	*threshold=4*	{ <4.5,1.5>, 4 }
<1,5>	<4,2>	<7,5>	⇨	{ <1.5,4.5>, 5 }	⇨	{ <1.5,4.5>, 5 }
<2,1>	<4,4>	<8,2>		{ <4.5,4.5>, 3 }		
<2,2>	<5,1>	<9,1>		{ <7.5,4.5>, 2 }		
<2,3>	<5,2>	<5,4>				

Figure 5. The transformation from raw data to the {central-point, density} format and the final output of approximation-via-sliding-window algorithm

Condensed data		Data representatives
{ <1.5,1.5>, 4 }		<1.5,1.5>, <1.5,1.5>, <1.5,1.5>, <1.5,1.5>
{ <4.5,1.5>, 4 }	⇨	<4.5,1.5>, <4.5,1.5>,<4.5,1.5>,<4.5,1.5>,
{ <1.5,4.5>, 5 }		<1.5,4.5>, <1.5,4.5>,<1.5,4.5>,<1.5,4.5>,<1.5,4.5>,

Figure 6. Data representatives that are generated back from the condensed format

The first step of the density-biased-sampling algorithm is the extraction of data points that are stored in the condensed form. After the extraction process, we obtain the representative data points as illustrated in figure 6. In the sampling step, user can choose different schemes of sample draw and temporary memory maintenance as follows:

Density-biased reservoir + Hashing
Density-biased reservoir + Simple random sampling
Density-biased reservoir + Rejection sampling

A set of samples drawn from streaming data is then forwarded to the Apriori-based frequent pattern discovery algorithm (Agrawal and Srikant, 1994).

4. Experimental Results

A. DNA Data Set

The proposed approximate method has been applied to find top-k frequent patterns from the DNA data set (available at http://archive.ics.uci.edu/ml/datasets/). This data set contains 3,186 instances. We split the data into two parts: the first 2,000 instances to be used as a training data and the rest 1,186 instances are for testing correctness of the discovered patterns. Each data instance is a sequence of 60 genetic codes (A=adenine, T=thymine, C=cytosine, G=guanine) obtained from different location of a gene. Some data samples are displayed in figure 7.

These genetic codes can be categorized as either exon/intron, intron/exon, or none. The exon/intron is the border region of genetic codes that links the exon part to the intron part. The intron/exon can be interpreted in the same manner, but vice versa. Exon is the part containing genetic codes that control the protein synthesis. Intron is the intervening area between exons and it will later be discarded before the synthesis of proteins. The none category is the genetic string that does not bear genetic codes for protein synthesis. The structure of exon and intron in a gene is schematically shown in figure 8.

```
T,T,C,T,A,T,G,A,G,A,A,A,C,G,T,G,G,C,A,T,T,G,T,G,C,G,C,A,A,G,G,T,G,G,G,C,C,C,
        C,G,C,G,G,G,A,C,G,G,G,G,C,A,G,C,T,C,C,G,G,G,exon/intron

C,T,C,C,C,C,A,C,C,C,A,C,C,T,G,T,C,C,A,C,C,C,G,C,C,C,G,C,A,G,A,T,C,G,C,T,T,C,C,
        T,G,G,A,G,C,C,A,G,G,C,A,A,G,A,A,C,T,C,C,A,intron/exon

C,T,G,A,C,T,A,A,G,C,C,G,C,C,C,C,T,T,G,T,C,C,C,T,T,C,T,C,A,G,A,T,T,A,T,G,T,T,T,
        G,A,G,A,C,C,T,T,C,A,A,C,A,C,C,C,C,G,G,C,C,intron/exon

G,A,G,G,A,G,C,T,A,G,A,C,A,A,G,T,A,C,T,G,G,T,C,T,C,A,G,C,A,G,G,T,G,C,G,T,G,A,
        G,G,G,G,A,G,G,G,G,A,T,G,G,C,T,G,C,C,A,A,G,G,exon/intron

A,A,G,G,C,T,C,A,G,G,A,G,G,A,G,G,G,A,G,A,T,C,A,A,C,A,T,C,A,A,C,C,T,G,C,C,C,C,G,
        C,C,C,C,C,T,C,C,C,C,A,G,C,C,T,G,A,T,A,A,A,none
```

Figure 7. Some DNA data instances

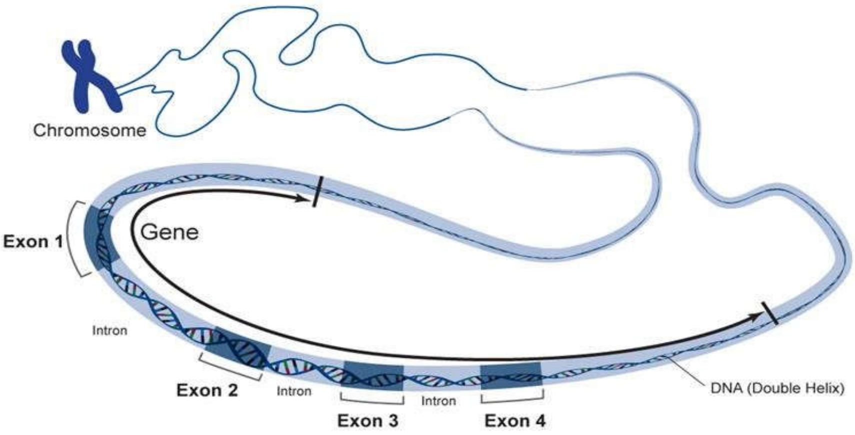


Figure 8. Structure of a gene with exon and intron parts (http://genome.gov/Glossary/)

B. *Testing Scheme*

We test the performance of our approximate method by simulating the DNA data set as a data stream, then feeding a stream to the density approximation and sampling algorithms. Data representatives are stored in a temporary memory area, called a reservoir. The representatives are finally processed by the frequent pattern discovery algorithm to find the top-k patterns. Completeness of the approximately discovered patterns is justified by the comparison against the frequent patterns that are discovered without the application of approximate method.

C. *Program Running Results*

We implemented our approximate frequent pattern discovery method with the Erlang programming language. The running result of the main function is shown in figure 9. Our approximate frequent pattern discovery program finds the frequent patterns of a specific class. In figure 9, we show the frequent patterns of a class intron/exon with the minimum support = 80%. At this level of support value, there are 3 frequent patterns of length 1 (k=1, or 1-item sets), 3 frequent patterns of length 2 (k=2, or 2-item sets), and 1 frequent pattern of length 3 (k=3, or 3-item set). These seven patterns (shown inside the red square in figure 9) can be interpreted as follows:

["AM"] means occurrence of the adenine base (A) at location 29 (ASCII code of M) in a DNA string

["CL"] means occurrence of the cytosine base (C) at location 28 (ASCII code of L) in a DNA string

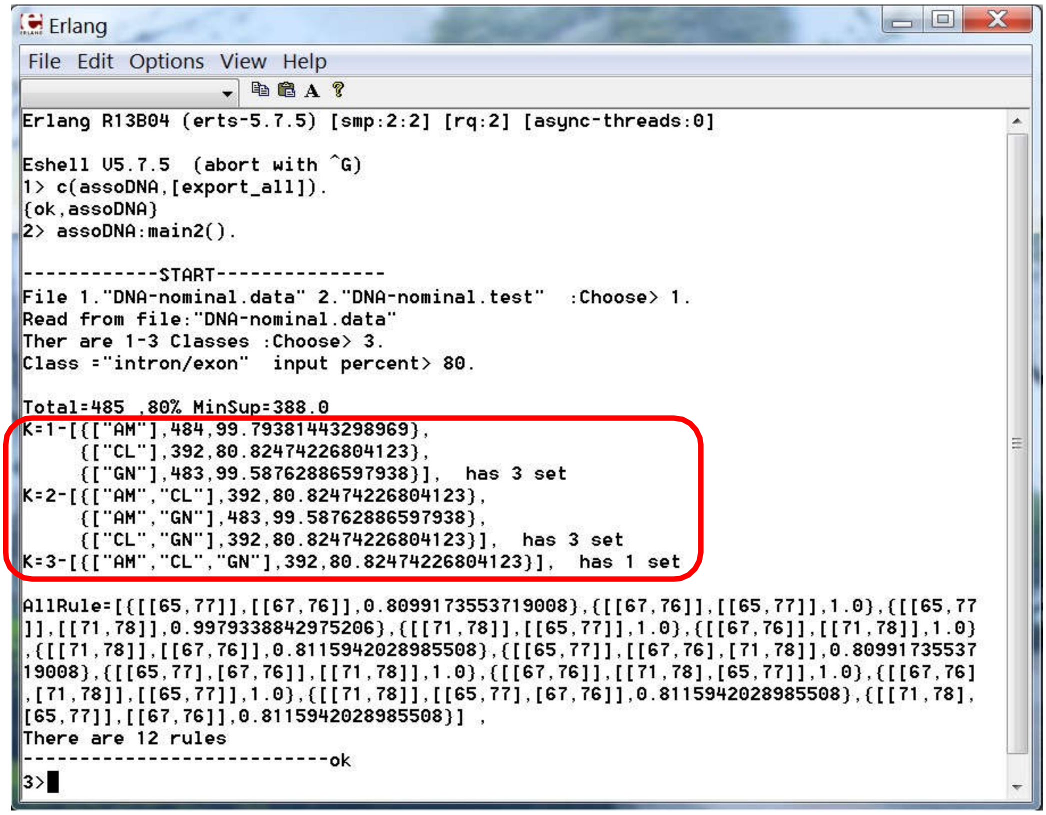
["GN"] means occurrence of the guanine base (G) at location 30 (ASCII code of N) in a DNA string

["AM", "CL"] means co-occurrence of the adenine base at location 29 and cytosine base at location 28 in a DNA string

["AM", "GN"] means co-occurrence of the adenine base at location 29 and guanine base at location 30 in a DNA string

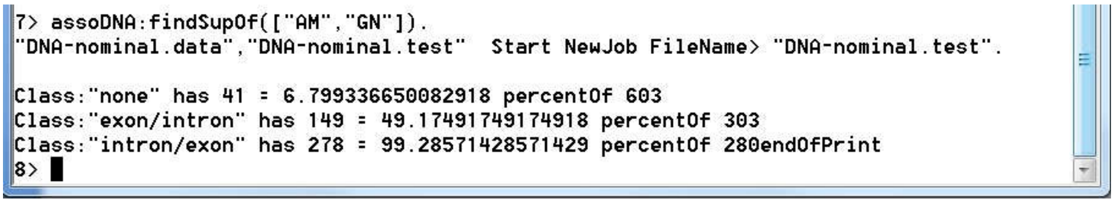
["CL", "GN"] means co-occurrence of the cytosine base at location 28 and guanine base at location 30 in a DNA string

["AM", "CL", "GN"] means co-occurrence of the adenine base at location 29, cytosine base at location 28, and guanine base at location 30 in a DNA string



```
Erlang R13B04 (erts-5.7.5) [smp:2:2] [rq:2] [async-threads:0]

Eshell V5.7.5  (abort with ^G)
1> c(assoDNA,[export_all]).
{ok,assoDNA}
2> assoDNA:main2().

------------START---------------
File 1."DNA-nominal.data" 2."DNA-nominal.test"  :Choose> 1.
Read from file:"DNA-nominal.data"
Ther are 1-3 Classes :Choose> 3.
Class ="intron/exon"  input percent> 80.

Total=485 ,80% MinSup=388.0
K=1-[{["AM"],484,99.79381443298969},
     {["CL"],392,80.82474226804123},
     {["GN"],483,99.58762886597938}],  has 3 set
K=2-[{["AM","CL"],392,80.82474226804123},
     {["AM","GN"],483,99.58762886597938},
     {["CL","GN"],392,80.82474226804123}],  has 3 set
K=3-[{["AM","CL","GN"],392,80.82474226804123}],  has 1 set

AllRule=[{[[65,77]],[[67,76]],0.8099173553719008},{[[67,76]],[[65,77]],1.0},{[[65,77
]],[[71,78]],0.9979338842975206},{[[71,78]],[[65,77]],1.0},{[[67,76]],[[71,78]],1.0}
,{[[71,78]],[[67,76]],0.8115942028985508},{[[65,77]],[[67,76],[71,78]],0.80991735537
19008},{[[65,77],[67,76]],[[71,78]],1.0},{[[67,76]],[[71,78],[65,77]],1.0},{[[67,76]
,[71,78]],[[65,77]],1.0},{[[71,78]],[[65,77],[67,76]],0.8115942028985508},{[[71,78],
[65,77]],[[67,76]],0.8115942028985508}] ,
There are 12 rules
---------------------------ok
3>
```

Figure 9. Running result of intron/exon frequent patterns with at least 80% of occurrence frequency (that is, minimum support = 80%)



```
7> assoDNA:findSupOf(["AM","GN"]).
"DNA-nominal.data","DNA-nominal.test"  Start NewJob FileName> "DNA-nominal.test".

Class:"none" has 41 = 6.799336650082918 percentOf 603
Class:"exon/intron" has 149 = 49.17491749174918 percentOf 303
Class:"intron/exon" has 278 = 99.28571428571429 percentOf 280endOfPrint
8>
```

Figure 10. The result of comparing the pattern [“AM”,”GN”] against the test data

Correctness of the discovered frequent patterns can be confirmed through the use of “findSupOf” function to predict the probable area of a gene in the test data set. Figure 10 shows the confirmation of the pattern [“AM”,”GN”], which is one of the discovered frequent patterns of a class intron/exon, through the search and comparison of this pattern against the whole test set. We found that this pattern matched 41 sub-patterns in the class none, 149 sub-patterns in the class exon/intron, and 278 sub-patterns in the class intron/exon. Based on the majority matching, we thus conclude that the discovered pattern [“AM”,”GN”] correctly represents the top frequent patterns of the class intron/exon.

For completeness confirmation, we compared the patterns discovered from our approximate method with those obtained from the traditional method that does not apply the density approximation and sampling technique. With varied percentages of minimum support value, our approximate method can discover patterns very close to the traditional method. The results are summarized in table 1.

Table 1. Comparative results of number of patterns discovered from our approximate method with those discovered from traditional method.

Minimum support	Traditional pattern discovery method				Approximate method				#Matched patterns
	# 1-item	# 2-item	# 3-item	# 4-item	# 1-item	# 2-item	# 3-item	# 4-item	
Class = “none”									
50%	0	0	0	0	0	0	0	0	0
45%	0	0	0	0	0	0	0	0	0
40%	0	0	0	0	0	0	0	0	0
35%	0	0	0	0	0	0	0	0	0
30%	1	0	0	0	1	0	0	0	1
25%	117	0	0	0	111	0	0	0	111
Class = “exon/intron”									
85%	3	2	0	0	3	2	0	0	5
80%	4	5	2	0	4	5	2	0	11
75%	4	5	2	0	4	5	2	0	11
70%	4	6	3	0	4	6	3	0	13
65%	5	8	5	1	5	8	5	1	19
60%	5	9	7	2	5	8	5	1	19
Class = “intron/exon”									
85%	2	1	0	0	2	1	0	0	3
80%	3	3	1	0	3	3	1	0	7
75%	3	3	1	0	3	3	1	0	7
70%	3	3	1	0	3	3	1	0	7
65%	3	3	1	0	3	3	1	0	7
60%	3	3	1	0	3	3	1	0	7

Table 2. Averaging summary of matched patterns against the test data.

Class	Matched patterns (traditional method)		Matched patterns (approximatie method)		Difference (traditional vs approximate)	
	2-item patterns	3-item patterns	2-item patterns	3-item patterns	2-item patterns	3-item patterns
none	--	--	--	--	--	--
exon/intron	91.24%	84.35%	90.98%	83.27%	0.26	1.08
intron/exon	90.11%	87.62%	90.09%	86.89%	0.02	0.73

The comparative results shown in table 1 have been performed on the training data set. When matching the discovered patterns against the DNA patterns in the test data set, we found that the difference of patterns matched by our approximate method to the ones that matched by traditional method is only 0.52% (averaging from the difference values: 0.26, 1.08, 0.02, 0.73). We therefore conclude from this empirical study that the discovery of frequent patterns from randomly selected representatives from a data stream yields the patterns as complete and accurate as the standard method that finds patterns from the whole large data set.

5. Conclusions

Frequent pattern discovery is an essential operation for association analysis. The discovery process concerns an automatic extraction of interesting patterns and correlations from a large database. These patterns can reveal implicit relationships among set of objects (or items) that lead to the generation of association rules to be used for decision support, financial forecast, medical diagnosis, and many other applications. Current studies in association rule mining concentrate on how to effectively find all objects frequently co-occurring. Given *m* objects, there are as much as 2^m frequent patterns to consider. Frequent pattern discovery is thus a computationally expensive problem. For the case of data streaming, this problem is even harder because a continuously generated nature of stream does not allow a revisit on each data element, but the discovery process must produce results in a reasonable short period of time.

With such a strict requirement, we therefore propose an approximate approach to tackle the frequent pattern discovery over continuous stream problem. Our approximate algorithm is intended to be a pre-processing step prior to the discovery process. We propose a stochastic method to get a good guess of the stream characteristics, and then draw a set of representatives from the incoming stream. These representatives are subsequently used in the process of frequent pattern mining. Our design had been implemented with the functional programming paradigm and the experimental results confirm the efficiency and reliability of our method. For a massive database, parallel method is a solution for the scalability problem. That is the main direction of our future research.

Acknowledgement

This research has been supported by grants from the Thailand Research Fund and the National Research Council of Thailand. The authors have been supported by Suranaree University of Technology through the funding of the Data Engineering Research Unit and the Knowledge Engineering Research Unit.

References

Agrawal R., Aggarwal C. and Prasad V., A tree projection algorithm for generation of frequent itemsets, *Journal of Parallel and Distributed Computing*, Vol. 61, pp. 350-371 (2001).

Agrawal R., Imielinski T. and Swami A., Mining association rules between sets of items in large databases, In *Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'93)*, pp. 207-216 (1993).

Agrawal R. and Srikant R., Fast algorithm for mining association rules in large databases, *Research Report RJ 9839*, IBM Almaden Research Center, San Jose, CA. (1994a).

Agrawal R. and Srikant R., Fast algorithms for mining association rules, In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pp. 487-499 (1994b).

Agrawal R. and Shafer J., Parallel mining of association rules: Design, implementation, and experience, *IEEE Trans. Knowledge and Data Engineering*, Vol. 8, pp. 962-969 (1996).

Babcock B., Babu S., Datar M., Motwani R. and Widom J., Model and issues in data stream systems, In *Proc. ACM Symp. Principles of Database Systems (PODS'02)*, pp. 1-16 (2002).

Cai Y., Pape G., Han J., Welge M. and Auvil L., MAIDS: Mining alarming incidents from data streams, In *Proc. Int. Conf. on Management of Data*, pp. 919-920 (2004).

Chang J. and Lee W., A sliding window method for finding recently frequent itemsets over online data streams, *Journal of Information Science and Engineering*, Vol. 20, No. 4, pp. 753-762 (2004).

Charikar M., Chen K. and Farach-Colton M., Finding frequent items in data streams, *Theoretical Computer Science*, Vol. 312, Issue 1, pp. 3-15 (2004).

Cheung D., Han J., Ng V., Fu A. and Fu Y., A fast distributed algorithm for mining association rules, In *Proc. 1996 Int. Conf. Parallel and Distributed Information Systems*, pp. 31-44, (1996a).

Cheung D., Han J., Ng V. and Wong C., Maintenance of discovered association rules in large databases: An incremental updating technique, In *Proc. 1996 Int. Conf. Data Engineering (ICDE'96)*, pp. 106-114 (1996b).

Chi Y., Wang H., Yu P. and Muntz R., Moment: Maintaining closed frequent itemsets over a stream sliding window, In *Proc. IEEE Int. Conf. on Data Mining*, pp. 59-66 (2004).

Coenen F. and Leng P., Partitioning strategies for distributed association rule mining, *The Knowledge Engineering Review*, Vol. 21, Issue 1, pp. 25-47 (2006).

Cuzzocrea A., Leung C. and MacKinnon R., Mining constrained frequent itemsets from distributed uncertain data, *Future Generation Computer Systems*, Vol. 37, pp. 117-126 (2014).

Elayyadi I., Benbernou S., Ouziri M. and Younas M., A tensor-based distributed discovery of missing association rules on the cloud. *Future Generation Computer Systems*, Vol. 35, pp. 49-56 (2014).

Gaber M., Zaslavsky A. and Krishnaswamy S., Resource-aware knowledge discovery in data streams, In *Proc. Int. Workshop on Knowledge Discovery in Data Streams*, pp. 649-656 (2004).

Gaber M., Zaslavsky A. and Krishnaswamy S., Mining data streams: A review, *ACM SIGMOD Record*, Vol. 34, Issue 2, pp. 18-26 (2005).

Ghoting A. and Parthasarathy S., Facilitating interactive distributed data stream processing and mining, In *Proc. IEEE Int. Symposium on Parallel and Distributed Processing Systems* (2004).

Grahne G. and Zhu J., Efficiently using prefix-trees in mining frequent itemsets, In *Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, pp. 123-132 (2003).

Guha S., Koudas N. and Shim K., Data streams and histograms, In *Proc. ACM Symposium on Theory of Computing*, pp. 471-475 (2001).

Halatchev M. and Gruenwald L., Estimating missing values in related sensor data streams, In *Proc. Int. Conf. on Management of Data*, pp. 83-94 (2005).

Han J. and Fu Y., Discovery of multiple-level association rules from large databases, In *Proc. 1995 Int. Conf. Very Large Data Bases (VLDB'95)*, pp. 420-431 (1995).

Han J., Pei J. and Yin Y., Mining frequent patterns without candidate generation, In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pp. 1-12 (2000).

Han J., Wang J., Lu Y. and Tzvetkov P., Mining top-k frequent closed patterns without minimum support, in *Proc. Int. Conf. on Data Mining*, pp. 211-218 (2002).

Jiang M. and Gruenwald L., Research issues in data stream association mining, *ACM SIGMOD Record*, Vol. 35, Issue 1, pp. 14-19 (2006).

Kargupta H., Bhargava R., Liu K., Powers M., Blair P., Bushra S., Dull J., Sarkar K., Klein M., Vasa M. and Handy D., VEDAS: A mobile and distributed data stream mining system for real-time vehicle monitoring, In *Proc. SIAM Int. Conf. on Data Mining*, pp. 300-311 (2004).

Kerdprasop K., Kerdprasop N. and Sattayatham P., Density-biased clustering based on reservoir sampling, In *Proc. 16th Int. Workshop on Database and Expert Systems Applications (DEXA)*, pp. 1122-1126 (2005).

Kerdprasop K., Kerdprasop N. and Sattayatham P., A Monte Carlo method to data stream analysis, *Enformatika Transactions on Engineering, Computing and Technology*, Vol.14, pp. 240-245 (2006).

Li H., Lee S. and Shan M., An efficient algorithm for mining frequent itemsets over the entire history of data streams, In *Proc. Int. Workshop on Knowledge Discovery in Data Streams* (2004).

Lin C., Chiu D., Wu Y. and Chen A., Mining frequent itemsets from data streams with a time-sensitive sliding window, In *Proc. SIAM Int. Conf. on Data Mining* (2005).

Lin Y., Hu X., Li X., and Wu X., Mining stable patterns in multiple correlated databases, *Decision Support Systems*, Vol. 56, pp. 202-210 (2013).

Liu J., Pan Y., Wang K. and Han J., Mining frequent item sets by opportunistic projection, In *Proc. 2002 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'02)*, pp. 239-248 (2002).

Mao G., Wu X., Liu C., Zhu X., Chen G., Sun Y. and Liu X., Online mining of maximal frequent item sequences from data streams, *Technical Report CS-05-07*, University of Vermont, U.S.A. (2005).

Park J., Chen M. and Yu P., An effective hash-based algorithm for mining association rules, In *Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'95)*, pp. 175-186 (1995a).

Park J., Chen M. and Yu P., Efficient parallel mining for association rules, In *Proc. 4th Int. Conf. Information and Knowledge Management*, pp. 31-36 (1995b).

Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U. and Hsu M., PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth, In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pp. 215-224 (2001).

Savasere A., Omiecinski E. and Navathe S., An efficient algorithm for mining association rules in large databases, In Proc. 1995 Int. Conf. Very Large Data Bases (VLDB'95), pp. 432-443 (1995).

Teng W., Chen M. and Yu P., Resource-aware mining with variable granularities in data streams, In *Proc. SIAM Int. Conf. on Data Mining* (2004).

Toivonen H., Sampling large databases for association rules, In *Proc. 1996 Int. Conf. Very Large Data Bases (VLDB'96)*, pp. 134-145 (1996).

Tseng F., Kuo Y. and Huang Y., Toward boosting distributed association rule mining by data de-clustering, *Information Sciences*, Vol. 180, pp. 4263-4289 (2010).

Vitter J., Random sampling with a reservoir, *ACM Transaction on Mathematical Software*, Vol. 11, No.1, pp. 37-57 (1985).

Yu J., Chong Z., Lu H. and Zhou A., False positive or false negative: Mining frequent itemsets from high speed transactional data streams, In *Proc. Int. Conf. on Very Large Databases* (2004).

Zaki M., Parthasarathy S., Ogihara M. and Li W., Parallel algorithm for discovery of association rules, *Data Mining and Knowledge Discovery*, Vol.1, pp. 343-374 (1997).

Zhu X., Li B., Wu X., He D. and Zhang C., CLAP: collaborative pattern mining for distributed information systems, *Decision Support Systems*, Vol. 52, Issue 1, pp. 40-51 (2011).