Data and Event Stream Mining

Kavya Naveen (PhD)
Asst. Prof, Dept of CSE
RNSIT, Bangalore.

Dr. M.V.Sathyanarayana
Prof and Head E&C Engg Dept
MCE, Hassan

N.C. Naveen
Asst. Prof, Dept of ISE
RVCE Bangalore

Summary

Management and analysis of streaming data has become crucial with its applications on web and for transactions data . Data streams consist of mostly numeric data but what is more interesting are the events derived from the numerical data that need to be monitored. The events obtained from streaming data form event streams. Event streams have similar properties to data streams, i.e., they are seen only once in order as a continuous stream. Events appearing in the event stream have time stamps associated with them at a certain time granularity, such as second, minute, or hour. One type of frequently asked queries over event streams are count queries, i.e., the frequency of an event occurrence over time. Count queries can be answered over event streams easily; however, users may ask queries over deferent time granularities as well. For example, a banking consultant may ask how many times a certain type of transaction is increased in the same time frame, where the time frames specified could be an hour, day, or both. Such types of queries are challenging especially in the case of event streams where only a window of an event stream is available at a certain time instead of the whole stream. In this paper, we propose a technique for predicting the frequencies of event occurrences in event streams at multiple time granularities.

Key Words: Mining, event streams, constraint –based mining, data streams, knowledge

1. Introduction

A transaction T supports an item set I if I is contained in T . The support for an item set I is defined as the ratio of the number of transactions that supports the item set I to the total number of transactions. If the support for an item set I satisfies the user-specified minimum support threshold, then I is called frequent item set, and a frequent item set of length k a frequent k-item set. The confidence of a rule X=>Y is defined as the ratio of the support for the item sets X union Y to the support for the item set X. If item set Z=X union Y is a frequent item set and the confidence of X=>Y is no less than the user-specified minimum confidence, then the rule X=>Y is an association rule.
Traditionally, data are collected and stored in a repository and queried or mined for useful information upon request .However, in the case of applications like sensor networks and stock market, data continuously .known as a stream and thus need to be queried or analyzed . Streaming data (or data streams) brought another dimension to data querying and data mining research[1]. This is due to the fact that, in data streams, as the data continuously generated only a window of the data is available at a certain time. The events occurring in a stream of data constitute an event stream, and an event stream has the same characteristics as a data stream, i.e., it is continuous and only a window of the stream can be seen at a time. Basically, an event stream is a collection of events that are collected from a data stream over a period of time. Experts would like to extract information from an event stream, such as the value of an event at a specific time-tick; frequency of certain events, correlations between different events, regularities within a single event; or future behavior of an event. Relationships among the events can be captured from event streams via online data mining tools.

Many constraint-based mining[2] methods have been proposed. Hipp and Guntzer (2002) presented that data mining process should be an initial unconstrained and costly mining run. The mining queries are answered from the initial mining result such that response time can be minimized. However, the discovered association rules may become invalid or inappropriate since the transactions are increasing any time. It is very costly to re-run the unconstrained mining algorithm[5] to obtain the up-to-date initial mining result. An
item constraint specifies what is the particular individual or group of items that should or should not be presented in the pattern, that is, the items in the discovered patterns have to be contained in the specified item set[3].

In this paper, we successfully integrate two kinds of patterns and use the similar style of the data mining language proposed in (Yen and Chen, 1997). Besides, we also propose efficient data mining methodology to find associations among certain items for stream data .
In this paper the constraint is event stream mining over time granularity.

2. Data mining for streams in a customer transaction

The data mining language is defined as follows. Users can query association rules or sequential patterns by
specifying the related parameters in the data mining language.

```
Mining <Data Mining Transaction>
From <ES>
With <(D1),(D2), .,(Dm)>
Support <S%>
Confidence <C%>
```

In the Mining clause, <Data Mining Transaction> can be association rules[10] or sequential patterns. The former is to discover association rules and the later is to discover sequential patterns.
In the From clause, <ES> is used to specify the database name to which users query the association rules or sequential patterns where ES represents Event Stream. In the With clause, if the

<Data Mining Transaction> is sequential patterns, <(D1),(D2), .,(Dm)> are user specified item sets which ordered by increasing purchasing time, and (Di) can be the notation '*' which represents any sequences. If the <Data Mining Transaction> is association rules then m is equal to 2, and D1 and D2 are the item sets in the antecedent and consequent, respectively, of the discovered rules. Besides, (Di) and the items in Di can be the notation '*' which represents any items. Support clause is followed by the user-specified minimum support s%.
Confidence clause is followed by the user-specified minimum confidence c% if the Data Mining Transaction is association rules. If the Data Mining Transaction is sequential patterns, this clause is ignored.

In order to find the interesting association rules and sequential patterns efficiently, we need to transform the original transaction data into another type and capture streams in the form of event streams[12]. Each event in each customer sequence is transformed into a bit string. The length of a bit string is the number of the transactions in the customer sequence. If the ith transaction of the customer sequence contains an item, then the ith bit in the bit string for this item is set to 1. Otherwise, the ith bit is set to 0. For example, in Table 1, the bit string for item A in CID 1 is 011. Hence, we can transform the customer sequence data base (Table 1) into the bit-string database (Table 2). From the bit-string database, we can easily compute the number of the transactions in a customer sequence, which contain an item set. For example, in Table 1, if we want to know how many transactions in CID 1 support the item set (A,C,E). We can perform logical AND operations on the bit strings for items A, C and E in CID 1. The number of 1's in the resultant bit string is the number of the transactions which contain the item set (A,C,E) in CID 1.
Suppose a transaction of customer contains the two events S1 and S2 We can do operation called event bit string operation to check if the events S1S2 is also contained in this transaction of customer sequence. The process of the event bit-string operation is described as follows: Let the bit string for sequence S1 in customer sequence c is B1, and for sequence S2 is B2. Bit string B1 is scanned from left to right until a bit value 1 is visited. We set this bit and all bits on the left hand side of this bit to 0 and set all bits on the
right hand side of this bit to 1, and assign the resultant bit string to a template Tb. Then, the bit string for sequence S1S2 in c can be obtained by performing logical AND operation on bit strings Tb and B2. If the number of 1's in the bit string for sequence S1S2 is not zero, then S1S2 is contained in customer sequence c. Otherwise, the customer sequence c does not contain S1S2.
For example, consider Table 1. We want to check if sequence <(A)(C)> is contained in customer sequence in CID 1. From Table 2, we can see that the bit string for items A and C in CID 1 are BAZ011 and BCZ111, respectively, and the template bit string TbZ001. By performing logical AND operation on Tb and BC, we can obtain that the bit string for sequence h(A)(C)i in customer sequence CID 1 is 001.

Event stream database (ESD)

CID	Customer sequence
1	h(C)(A,C)(A,C,E)i
2	h(A,E)(A)(A,C,E)(C,E)i
3	h(C)(E)(E)(C,E)i
4	h(B,D)(A,E)(B,C)(A,E)(A,B,E)(F)i
5	h(D)(D,E,F)(C,E,F)(A,D)(B,D)(D,F)i

Table 1

Bit-string database

CID	Transaction items	Bit string for each item
1	A, C, E	011,111,001
2	A, C, E	1110,0011,1011
3	C, E	1001,0111
4	A, B, C, D, E, F	010110,101010,001000,100000,010110,000001
5	A, B, C, D, E, F	000100,000010,001000,110111,011000,011001

Table 2

3. Mining events association rules

In this section, we focus on mining event streams association rules[6] according to proposed data mining language. We divide the query into two cases.
Case 1: there are items in the antecedent of the discovered rules specified.
Case 2: there are items in the consequent of the discovered rules specified, but the item in the antecedent is not specified, which can be any items. Suppose that the event set specified in the antecedent of the discovered rules in
Case 1 is X, and the event set specified in the consequent of the discovered rules is Y. We propose an efficient algorithm called MEAR (Mining Events Association Rules) to find all the interesting association rules according to the user requirements, which is described as follows:

Step 1. Scan the bit-string database once to compute the support for the specified event set, and then find all the frequent 1-event sets.
Step 2. Generate candidate (k+1)-event sets (k is the length of event set X (or X union Y) for Case 1, and k is the length of event set Y for Cases 2), scan the bit-string database to find the frequent (k+1)- event sets, which contain the event set X (or X union Y) for Case 1, the event set Y for Case 2, and generate the (k+1)-event set database.
Step 3. The frequent event sets are generated for each iteration. In the (h–k)th iteration (h>=k+1), generate candidate (h+1)-event sets, scan the h-event database to generate (h+1)-event set database and find all the frequent (h+1)-event sets).

3.1. Applying Time constraint for events

Data analysis at multiple time granularities was already explored in the context of sequential pattern mining by Bettini et al. [5]. Our target, however, is to .find frequencies of event occurrences at multiple time granularities without any time restriction. Temporal aggregation queries were well studied and several approaches have been proposed recently [10,13]. However, all these works consider only a single time granularity, where this granularity is usually the same as the granularity used to store the time attributes. To the best of our knowledge, the only work exploring the aggregate queries of streaming data in the time dimension at multiple time granularities appeared Zhang et al. present specialized indexing schemes for maintaining aggregates using multiple levels of temporal granularities: older data is aggregated using coarser granularities while more recent data is aggregated with .near detail. If the dividing time between different granularities should be advanced, the values at the near granularity are traversed and the aggregation at coarser granularity[8] is computed. However, we scan the stream only once and estimate the frequency of the event at any arbitrary time granularity without storing any information at intermediate time granularities.

Definition 1: Let G and H be two granularities. Then, G is said to be finer than H, denoted as G _ H, if for each time-tick i in G, there exists a time-tick j in H such that G(i) _ H(j). If G _ H, then H is said to be coarser than G. For example, day is .near than week, and coarser than hour, because every day is a subset of a week and every hour is a subset of a day.

Definition 2. A 0/1 event stream is an event stream where each time-tick records the state of the event at that time-tick, which is equal to 1 if the event occurs, and 0 otherwise.

When we transform an event stream at time granularity to another event stream Sg1 at granularity , we obtain a deferent set of time-ticks and different sets of events associated with these time-ticks. Before we give the formal definition on of transformation of a stream, the following two concepts need to be introduced.

Definition 3: A Transformation Operation is a mapping P:Ec -> E that takes event states at c successive time-ticks where c is the transformation coefficient, and returns a single event state according to the particular operation in use. Some common transformation operations are Min Merge, Max Merge, Avg-Merge, Sum Merge, Union, and Intersection. For example, Min Merge operation returns the minimum event value from the set of c events, where c is the transformation coefficient. The other operations are defined similarly. In this paper, we are interested in 0/1 (Boolean) event stream where the universal set of event states is {0, 1}, and we use merge OR operation that logically ORs the event values at corresponding time-ticks. Besides merge OR, some other transformation operations can also be used as long as their output is also a Boolean event
stream.

For example , Stream Query 1 means that the user would like to find all the association rules whose antecedent and consequent contain items A and C, respectively, from the customer sequence database ESD (Table 1). The minimum support and the minimum confidence are set to 5 and 20%, respectively.

Stream event Query 1:
Mining <Association Rules>
From <ESD>
With <(A,*),(C,*)>
support <5%>
confidence <20%>

After performing step 1, we can find all the frequent 1-eventsets and their supports. The set of the frequent 1-eventsets are {A, B, C, D, E, F}. Because (A, C) is a frequent event set, we go on Step 2. According to step 2, we can obtain the candidate 3-itemsets (A, B, C), (A, C, D), (A, C, E) and (A, C, F). After scanning bit-string database, the
generated 3-eventset database are shown in Table 3, and the frequent 3-eventset is (A, C, E). Finally, the frequent event sets that contain the event set (A, C) are (A, C) and (A, C, E). According to step 4, because there is the specified event set (A) in the antecedent and event set (C) in the consequent of the discovered rules in Query 1, we can find three association rules: (A)=>(C, E), (A, E)=>(C) and (A)=>(C).

4. Mining event sequential patterns

In this section, we describe the proposed algorithm MESP (Mining Event Sequential Patterns) for finding all the interesting sequential patterns according to the user requirements. For example, in Query 2, the user would like to find all the sequential patterns which contain the sequence <(E)(A)(B)>from the customer sequence database
(Table 1) and the minimum support threshold is set to 40%.

Sream event Query 2:
Mining <Sequential Patterns>
From <ESD>
With <*,(E),*,(A),*,(B),*> support <40%> Suppose the user specifies a sequence which contains m event sets D1, D2, .and Dm in the With clause and S=<(D1) (D2).(Dm)>. We divide the algorithm MESP for this type of queries into two steps: the first step is to find (m+1)- frequent sequences which contains sequence S, and the second step is to find all the q-frequent sequences (q>=m+ 2) which contains sequence S. In the following, we describe the two steps:

Step 1. Find all the frequent (m+1)-sequences Step 1.1. Scan the bit-string database, if all events in S are contained in a record, then output the events in this record and the bit string for each item into 1-eventset database. If S is a frequent sequence, then find all frequent 1-eventsets. The frequent event sets are found in each iteration. For the kth iteration (k>=1), the candidate (k+1)-event sets are generated, and scan the (k+1)-event set database to find (k+1)- frequent event sets .Finally, we output the frequent k-event sets and its bit string in each record into the frequent event set database.

Step 1.2. Each frequent event set (i.e. frequent 1-sequence) is given a unique number, and replace the frequent event sets in the frequent event set database with their numbers to form a 1-sequence database.

For example, in Table 2, the records which contain the sequence h(E)(A)(B)i in the With clause in Query 2 are CID 4 and CID 5, Hence, we can generate the frequent event sets
(A), (B), (C), (D), (E), (F) and (B, D), and the numbers for the frequent event sets are 1, 2, 3, 4, 5, 6, and 7, respectively.

3-eventset database for Query 1

CID	3-eventsets	Bit string for each 3-eventset
1	(A, C, E)	001
2	(A, C, E)	0010

Table 3

The 1-sequence database is shown in Table 4.

1-sequence database for Query 2

CID	1-sequence	Bit string for each 1-Sequence
4	1, 2, 3, 4, 5, 6, 7	010110, 101010, 001000, 100000, 010110, 000001, 100000
5	1, 2, 3, 4, 5, 6, 7	000100, 000010, 001000, 110111, 010000,010001,000010

Table 4

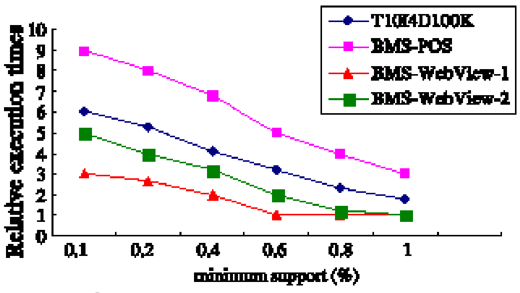

FIGURE 1

Relative execution times for different event sets over time granularity

Conclusion

In this paper , we propose how to mine stream data by slicing streams as events and then applying Mining events with association rule method and Mining events with sequential pattern methods. The concept for bit –string database and items within even sets are considered even though they cost extra memory space, the event mining query response time can be reduced more efficiently.

References

[1] .Agrawal, R. & Srikant, R. (1995). Mining sequential patterns. In Proceedings of international conference on data engineering (ICDE). pp. 3–14.

[2] Han, J.&Pei, J. (2000).Mining frequent patternsby pattern-growth:Methodology and implications. In Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining. pp. 30–36.

[3] Hipp, J., & Guntzer, U. (2002). Is pushing constraints deeply into the mining algorithms really what we want?—an alternative approach for association rule mining. In SIGKDD Explorations, 4(1), 50–55.

[4] Meo, R., Psaila, G., & Ceri, S. (1996). A new SQL-like operator for mining association rules. In Proceedings of international conference on very large data base. pp. 122–133.

[5] Ng, R., Lakshmanan, L.S., Han, J., & Mah, T. (1999). Exploratory mining via constrained frequent set queries. In Proceedings of ACM SIGMOD. pp. 556–558.

[6] Pei, J. & Han, J., (2000). Can we push more constraints into frequent pattern mining? In Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining. pp. 350–354.

[7] Pei, J, & Han, J., (2002). Constrained frequent pattern mining: A patterngrowth Vi ew. In SIGKDD Explorations, 4(1), 31–39.

[8] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M.C. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proceedings of international conference on data engineering. pp. 215–224.

[9] Pei, J., Han, J., & Wang, W. (2002). Mining sequential patterns with constraints in large databases. In Proceedings of ACM conference on information and knowledge management (CIKM). pp. 18–25.

[10] Yen, S.J., & Chen, A.L.P. (1997). An efficient data mining technique for discovering interesting association rules. In Proceedings of international conference on database and expert systems applications (DEXA). pp. 664–669.

[11] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the ACM SIGMOD Conference on Management of Data, 1993, pp. 207–216.

[12] M. Atallah, R. Gwadera, W. Szpankowski, Detection of signi.cant sets of episodes in event sequences: algorithms, analysis and experiments, in: roceedings of the 4th IEEE International Conference Data Mining, 2004, pp. 3–10.

[13] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data streams, in: Proceedings of the ACM PODS Symposium on Principles of Database Systems, 2002..