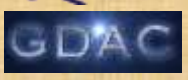
CMRULES: An Efficient Algorithm for Mining Sequential Rules Common to Several Sequences

Philippe Fournier-Viger[1]
Usef Faghihi[1]
Roger Nkambou[1]
Engelbert Mephu Nguifo[2]

[1]University of Québec in Montréal (Canada)
[2]University Blaise-Pascal(France)



May, 20th 2010

Introduction

- **Discovering temporal relationships** between events stored in large databases is important in many domains. It helps to understand the relationships between events and sets a basis for the prediction of events
- For example:
  - predicting stock market
  - predicting consumer behavior
- In this paper, we focus on temporal relationships in sequences of discrete events

Sequential Pattern Mining

- One of the most popular set of techniques for discovering temporal relations between events in sequences of event is **sequential pattern mining**.
- It finds subsequences that are common to several sequences in a sequence database.
- However, knowing that a sequence of events appear frequently in a database **is not sufficient for the prediction of events**. For example, it is possible that some event *y* appears frequently after an event *x* but that there are also many cases where *x* is not followed by *y*.
- For **prediction**, we need a measurement of the confidence that if *x* occurs *y* will occur afterward.

3

Sequential Rule Mining

- The alternative that addresses prediction is **Sequential Rule Mining.**
- **Sequential rules** are also called episode rules, prediction rule or temporal rules.
- A sequential rule typically has a *confidence* and a *support.*
- Manila et al. (1997) discovers rules of the form X→Y appearing frequently **in a single sequence of events**. X and Y are sets of events.
- Other works that find rules **in a single sequence of events**: Hamilton & Karimi (2005), Hsieh (2006), Deogun (2005), etc.
- **Several applications:** stock market analysis (Das et al., 1998; Hsieh et al., 2006), weather observation (Hamilton & Karimi, 2005), drought management (Harms et al. 2002), alarm analysis, etc.

4

Sequential Rule Mining (continued)

- Algorithms for finding rules occurring **frequently in several sequences**:
  Das et al. (1998), $x \rightarrow y$ where x,y are single events
  Harm et al. (2002). $X \rightarrow Y$ where X, Y are sets of events
- However, these algorithms are **not** appropriate for finding **rules that are common to several sequences**.
- The reason is that they are designed for mining rules appearing frequently in sequences. With these algorithms, if a rule appears many times in a sequence, it can be considered frequent even if it does not appear in any other sequences.
- In this paper, **we address the problem of mining sequential rule common to several sequences**.

5

Our proposal

- We propose an algorithm for mining rules **common to several sequences**. This can be useful for example for mining rules that are common to several customers.
- The algorithm is based on association rule mining.
- The idea is to find associations rules between items to prune the search space to items that occur jointly in many sequences. Then it eliminates association rules that are not sequential rules. We prove that this correctly generates all sequential rules.

6

Association Rule Mining

- **A transaction database** D is a set of transactions $T=\{t_1,t_2\ldots t_n\}$ and a set of items $I=\{i_1, i_2,\ldots i_n\}$, where $t_1,t_2,\ldots,t_n \subseteq I$.
- **Association rule:** $X \rightarrow Y$, such that $X,Y \subseteq I$, $X \cap Y=\emptyset$,
- The **support** of a rule $X \rightarrow Y$ is defined as $sup(X \cup Y) / |T|$. The **confidence** of a rule is defined as $conf(X \rightarrow Y) = sup(X \cup Y) / sup(X)$.
- **Association rule mining**: finding all rules such that their support and confidence are higher to thresholds *minsup* and *minconf*.

ID	transactions
1	a b c e f g
2	a b c d e f
3	a b e f
4	b f g

(2) Association rule mining

Association rules

ID	rule	Sup.	Conf.
1	c → a b e f	0.5	1.0
2	a b c → e	0.5	1.0
3	a → c e f	0.5	0.6
4	a b → e f	0.75	1.0
5	b → e f	0.75	0.75
6	a → e f	0.75	1.0
7	c → f	0.5	1.0
8	a → b	0.75	1.0
	…	…	…

7

Definition of a Sequence Database

- A **sequence database** SD is defined as a set of sequences $SD=\{s_1, s_2\ldots s_n\}$ and a set of items $I=\{i_1, i_2,\ldots i_n\}$, where each sequence s_x is an ordered list of transactions $s_x=\{X_1, X_2, \ldots X_n\}$ such that $X_1, X_2, \ldots X_n \subseteq I$.
- An example:

ID	sequences
1	(a b), (c), (f), (g), (e)
2	(a d), (c), (b), (e f)
3	(a), (b), (f) (e)
4	(b), (f g)

8

Mining Sequential Rules Common to Several Sequences

- A **sequential rule** X⇒Y is a relationship between two itemsets X,Y such that X,Y ⊆ I and X∩Y = ∅.
- **Sequential support:** seqSup(X ⇒ Y) = sup(X■Y) / |SD|.
- **Sequential confidence:** seqConf(X ⇒ Y) = sup(X■Y) / sup(X).
- sup(X■Y) denotes the number of sequences from a sequence database where all the items of X appear before all the items of Y.
- **Sequential rule mining:** finding all rules with sequential support and a sequential confidence higher than some thresholds *minSeqSup* and *minSeqConf*.

9

An example of Sequential Rule Mining

Some rules found for minSeqSup= 0.5 and minSeqConf= 0.5:

ID	sequences
1	(a b), (c), (f), (g), (e)
2	(a d), (c), (b), (e f)
3	(a), (b), (f) (e)
4	(b), (f g)

→

ID	rule	S-Sup.	S-Conf.
1	a b c⇒e	0.5	1.0
2	a⇒c e f	0.5	0.66
3	a b ⇒ e f	0.5	1.0
4	b⇒ e f	0.75	0.75
5	a⇒ e f	0.75	1.0
6	c⇒ f	0.5	1.0
7	a⇒b	0.5	0.66
	…	…	…

10

The Relationship between Association Rules and Sequential Rules

- A **sequence database S can be transformed into a transaction database S'** by removing time information.
- Each sequential rule r: X⇒Y of S has a corresponding association rule r': X→Y in S'.
- Since sup(X■Y) is always lower or equal to sup(X∪Y) these relationships hold:
  - sup(r') ≥ seqSup(r)
  - conf(r') ≥ seqConf(r),

  for any sequential rule r and its corresponding association rule r'.

11

The CMRules algorithm

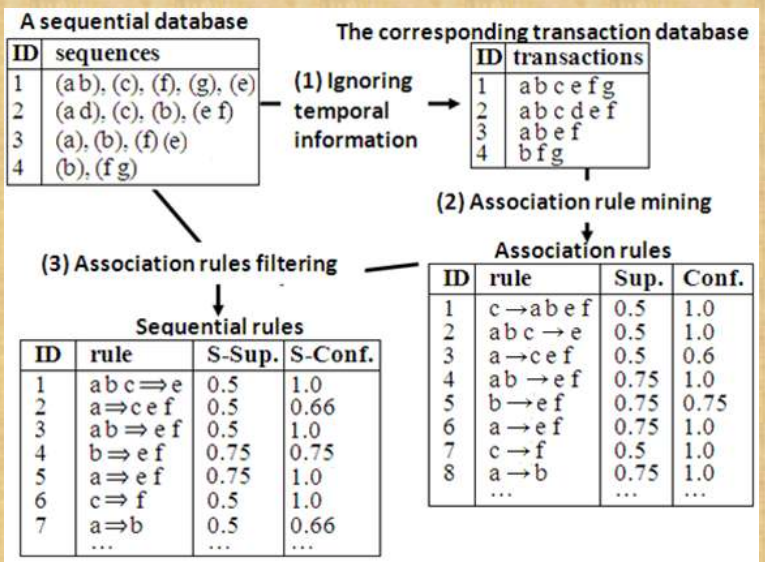
INPUT : a sequence database, *minSeqSup*, *minSeqConf*
OUTPUT : the set of all sequential rules
PROCEDURE:

1. Consider the sequence database as a transaction database
2. Find all association rules from the transaction database by applying an association rule mining algorithm such as Apriori (Agrawal et al., 1993). Select *minsup* = *minSeqSup* and *minconf* = *minSeqConf*.
3. Scan the original sequence database to calculate the sequential support and sequential confidence of each association rule found in the previous step. Eliminate each rule *r* such that:
   a. seqSup(r) < *minSeqSup*
   b. seqConf(r) < *minSeqConf*
4. Return the set of rules

see the article for the proof of correctedness.

12

A Sample Execution of the CMRules algorithm



13

Optimizations

- **How to calcultate calculate** sup(X■Y) for a rule X→Y ?
  - The naïve approach is to check all sequences.
  - Association rule mining algorithms first find frequent itemsets X and Y and then generate rules of the form: X→ Y-X
  - Algorithms such as AprioriTID, Eclat, H-Mine, etc. can annotate itemset X and Y with the sequences that contains them.
  - If we use such algorithm, we can only check sequence containing X for calculating sup(X■Y) .
  - This can improve performance by up to 50 %.
- Also, we don't need to keep the association rules. We can discard each rule immediately after it is found if it is not a sequential rule. This reduces memory consumption.

14

Analysis of the Time Complexity

- Step 1: converting a sequence database in a transaction database is done in linear time.
- Step 2: association rule mining. Two substeps:
  1. Discovering frequent itemsets:
     Apriori : $O(d^2 n)$ d= number of diff. items, n= number of transactions
  2. Generating association rules from frequent itemsets: less costly, thus can be ignored.
- Step 3: calculating sequential conf. and support.
  For each rule, **best case**: $|S| \times$ *minsup* sequences to check, **worst case**: |S| sequences to check. Checking if a rule is included in a sequence is done in linear time.

15

A Second Algorithm: CMDeogun

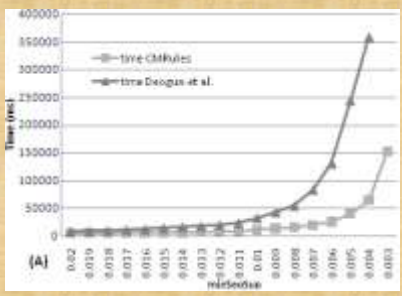
- It is an adaptation of the algorithm of Deogun et al. for the case of several sequences.
- The approach is similar to the Apriori algorithm.
- The process:
  - **First** find all rules with an antecedent and consequent of size 1.
  - **Then**, recursively combine rules of smaller size to discover larger rules by performing left expansion and right expansion.

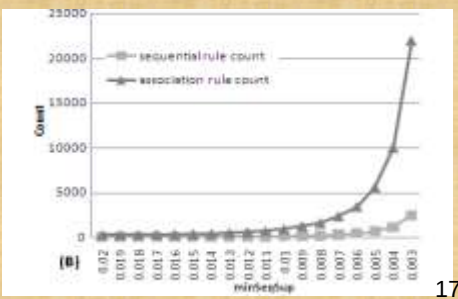
16

Comparaison of CMRules and CMDeogun on the Kosarak dataset

Kosarak Dataset : 990 000 click-stream data from the logs of online news portal.

Experiment, we selected the 70 000 first sequences of the dataset. These sequences contain an average of 7.97 items each, from a set of 21 144 different items.
We applied the algorithms on this dataset with *minSeqConf* = 0.3 and with *minSeqSup* = 0.02, 0.019, ..., 0.003.





17

Discussion of results

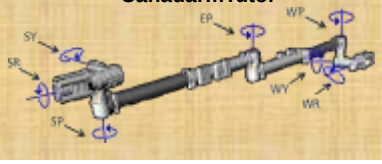
- The percentage of sequential rules can be quite high in real datasets.
  - For *minSeqSup* of 0.02, there is 312 association rules and 64 sequential rules (20.51 %).
  - For a *minSeqSup* of 0.003, there is 21 947 association rules and 2 512 sequential rules (11.44%).
- The execution time of CMRules seems to grow proportionally to the number of association rules.
- CMRules has a better scalability for this dataset.
- The time for converting the sequence database in a transaction database is negligible (~300 ms).

18

An Application of CMRules in a Cognitive Agent

- The **CTS Cognitive Agent** has some predefined behaviors.
- Each execution of CTS is recorded as a sequence in a sequence database
- Items = actions and perceptions of CTS
- Itemset = one cognitive cycle
- Rules are used for making predictions about which behavior will be successful with learners, and adapt the behavior of CTS accordingly.

CanadarmTutor



19

An Application of CMRules in a Cognitive Agent

- We did 100 executions of CTS (100 sequences)
- Sequence length: average of 25 itemsets
- CMRules was executed after each execution of CTS
- On average 31.05% of the association rules were sequential rules.
- Execution time was less than 50 ms.
- We have found several relevant rules. For example:
  - "the learner move the wrong joint ⇒ the learner makes the robotic arm pass too close to the space station"
  - "the learner lacks motivation ⇒ the learner is inactive"

20

Conclusion

Summary

- CMRules: an algorithm for mining sequential rules common to several sequences.
- CMRules can discover sequential rules and association rules at the same time
- Extensions by using another association rule mining algorithm (incremental mining, parallel mining, etc.).
- Extensions by modifying the definition of sup(X■Y)
- CMDeogun: a second algorithm.

Future works

- Further experiments will be done to measure empirically the performance of CMRules and CMDeogun. We will also try to define other algorithms.

21

Thank you. Questions?

Acknowledgement to:

- Current and past members of the GDAC and PLANIART research teams for their collaboration.
- NSERC and FQRNT funding programs

Fonds de recherche sur la nature et les technologies Québec

CRSNG NSERC