C3: More on Association Rules:
Data Stream, FP-Tree, Vertical Mining

October 2, 2018

Ming-Syan Chen

Agenda for Classes 2018
(Might be revised as we progress)

- Class 1 – (9/11) Overview of data mining
- Class 2 – (9/18) R (I), Wush Wu
- Class 3 – (9/25) Association, Apriori and its related issues
- Class 4 – (10/2) Data stream mining, FP Tree, Vertical mining (announcing HW1)
- Class 5 – (10/9) Classification: decision tree, GPGPU
- Class 6 – (10/16) Description of Data, Project announcement (announcing HW2)
- Class 7 – (10/23) R (II), Wush Wu

Tentative Class Agenda (cont'd)

- Class 8 – (10/30) Data exploration, more on decision trees, rule-based classifiers
- Class 9 – (11/6) Scikit learn, LibSVM, Preparation for HW3 and HW4
- Class 10 – (11/13) KNN, Bays, Neural network, Concept of SVM
- Class 11 – (11/20) Abstract presentation, SVM, Clustering, K-means, PAM
- Class 12 – (11/27) More on clustering; Sequential pattern mining;
- Class 13 – (12/4) Web mining, PageRank, etc.

Tentative Class Agenda (cont'd)

- Class 14– (12/11) When Database and Data Mining Meet, Prof. Mingling Lo
- Class 15 – (12/18) Project presentation I
- Class 16 – (12/25) Project presentation II

  (Final Exam according to Univ. Schedule)

  (Project due 1/24/2019)
- Happy New Year!

Data Stream:
Too fast and too huge to capture

Related Papers

- W.-G. Teng, M.-S. Chen and P. S. Yu, ``A Regression-Based Temporal Pattern Mining Scheme for Data Streams," *Proc. of the 29th Intern'l Conf. on Very Large Data Bases (VLDB-2003)*, September 9-12, 2003.
- J. Han, J. Pei, Y. Yin, R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach" DMKD, Vol. 8, No. 1, Jan. 2004. (SIGMOD 2000)
- M. J. Zaki, Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering 12 (3), 372-390, 2000

Data Streams

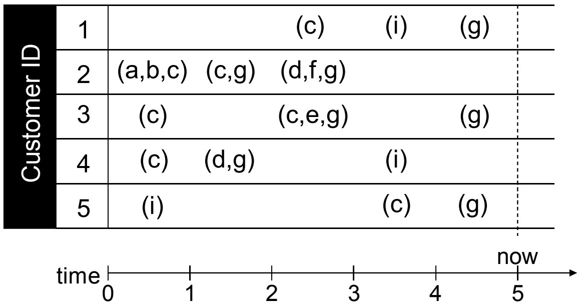
- Traditional DBMS – data stored in finite, persistent data sets
- Emerging Applications – data input as continuous, ordered data streams
  - Network monitoring and traffic
  - Telecom call detail records (CDR)
  - ATM operations in banks
  - Sensor networks
  - Web logs and click-streams
  - Transactions in retail chains
  - Mobile computing
  - One of the most important issues in data mining

Data Streams (cont'd)

- Definition
  - Continuous, unbounded, rapid, time-varying streams of data elements
- Application Characteristics
  - Massive volumes of data (can be several terabytes)
  - Records arrive at a rapid rate
- Goal
  - Mine patterns, process queries and compute statistics on data streams in real-time

Online Transaction Flows

- Example data of a market-basket application
  - In the form of <TxTime, CustomerID, Itemset>

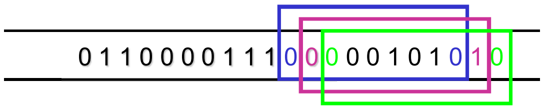


Support Framework for Temporal Patterns

- Goal: to mine all frequent temporal patterns satisfying the time constraint
- Occurrence frequency (or support) of patterns is the metric commonly used
  - However, the definition of support varies from one application to another
- With the sliding window model, the support of a temporal pattern X is generally (newly) defined as

$$Sup_t(X) = \frac{\text{\# of customers having pattern X at time t}}{\text{\# of all customers}}$$

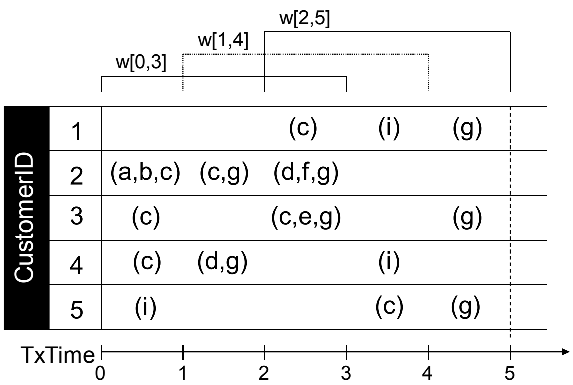
Sliding Window Model



- A fixed-width window is utilized so that only data elements within the current time window are taken into consideration
- Why?
  - Approximation technique for bounded memory
  - Natural in applications (emphasizes recent data)
  - Well-specified and deterministic semantics

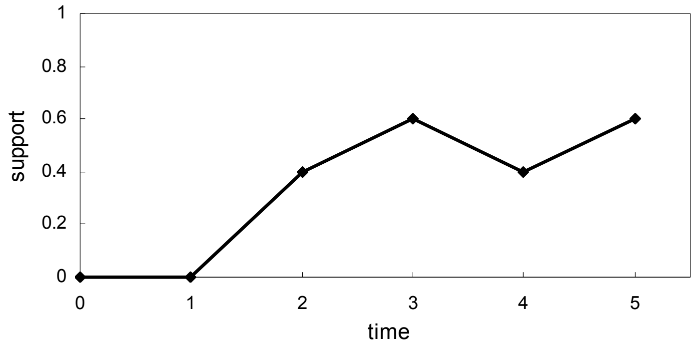
Example of the Support Counting

- Consider the itemset {c, g} (wSize=3)



Example of Support Counting (cont'd)

- Support variations of {c, g}



One Scan for Statistics Collection

- Levelwise generation of patterns
  - In the 1st window, only singleton items are counted
  - Frequent item(set)s are retained and joined to form longer candidate itemsets
  - Candidate itemsets are counted in the next window, while infrequent ones are discarded

Example of One Scan Support Counting ({c,g} not the same as shown before)

- (accumulated supports)/(# of recorded windows)

t=1	
{c}	0.6 / 1

t=2	
{c}	1.2 / 2
{g}	0.4 / 1

t=3	
{c}	2 / 3
{d}	0.4 / 1
{g}	1 / 2
{c, g}	0.6 / 1

t=4	
{c}	2.8 / 4
{d}	0.8 / 2
{g}	1.6 / 3
{i}	0.4 / 1
{c, g}	1 / 2
{d, g}	0.4 / 1

t=5	
{c}	3.4 / 5
{g}	2.4 / 4
{i}	0.8 / 2
{c, g}	1.6 / 3

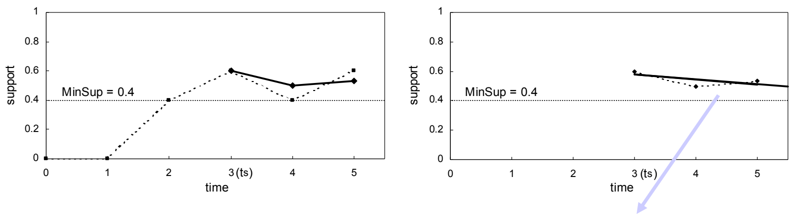
MinSup=0.4

Delayed Pattern Recognition

- E.g., for {c} t=2, add four 0.2's (1-4;to t=3), add four 0.2's (1,2,3,5; to t=4), add three 0.2's (1,3,5; to t=5)
- Delayed pattern recognition occurs, e.g.,
  - Actually, {c, g} is frequent at t=2
  - But since items c & g are not both frequent until t=2, {c, g} is generated & counted at t=3
  - Only could transient patterns be neglected
  - Patterns with supports near the threshold are examined and identified if so qualified

Regression-Based Analysis

- To maintain support variations of frequent patterns with lower space cost
  - Make the series averaged & find a fit line



$y = -0.0333x + 0.6778$

Construct the Fit Line

- To construct the fit line f=A+Bt, we have
  - $B = S_{tf}/S_{tt}$, and
  - $A = (\sum f)/n - B(\sum t)/n$,

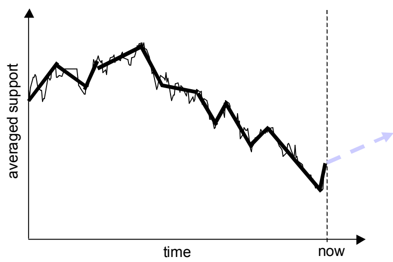
  where
  - $S_{tt} = \sum t^2 - (\sum t)^2/n$
  - $S_{ff} = \sum f^2 - (\sum f)^2/n$
  - $S_{tf} = \sum tf - (\sum t)(\sum f)/n$
  - Note: n(# of points), $(\sum t)$ and $(\sum t^2)$ can be calculated from t_s and t_{now}

Compact ATF Representation

- Accumulated Time & Frequency form
  - (t_s, $\sum tf$, $\sum f$, $\sum f^2$) can losslessly represent the fit line
    - t_s: starting time
    - t: time index
    - f: occurrence frequency of the pattern at t
  - Easy to update this form by accumulating corresponding (product) entries

Piecewise Linear Representation

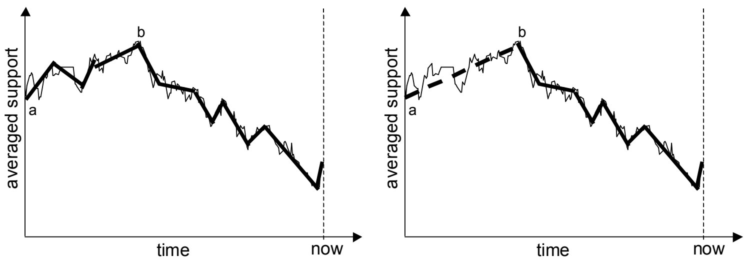
- Not only a single fit line but a set of fit lines are used to represent the original series
- Use an ATF list to maintain multiple entries of successive ATF entries



Problem!
Unbounded memory space is required as time advances!

Segment Relaxation

- To make memory space bounded
- Temporal granularity is introduced
  - People are interested in recent changes at a fine scale, but old changes at a coarse scale



FP-Tree: Another way for association rule mining (in addition to Apriori)

Related Papers

- W.-G. Teng, M.-S. Chen and P. S. Yu, ``A Regression-Based Temporal Pattern Mining Scheme for Data Streams," *Proc. of the 29th Intern'l Conf. on Very Large Data Bases (VLDB-2003)*, September 9-12, 2003.
- J. Han, J. Pei, Y. Yin, R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach" DMKD, Vol. 8, No. 1, Jan. 2004. (SIGMOD 2000)
- M. J. Zaki, Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering 12 (3), 372-390, 2000

Apriori-like Algorithm

- Essential idea
  - Generate the set of candidate patterns of length (k+1) from the set of frequent patterns of length k.
  - Check the corresponding occurrence frequencies of the candidate patterns in the database.

What does Apriori-like algorithm suffer from?

- In situations with many frequent patterns, long patterns, or quite low minimum support threshold, it is costly.
- It is tedious to repeatedly scan the database and check a large set of candidates by patterns matching.

FP-Tree: To avoid generating candidate patterns

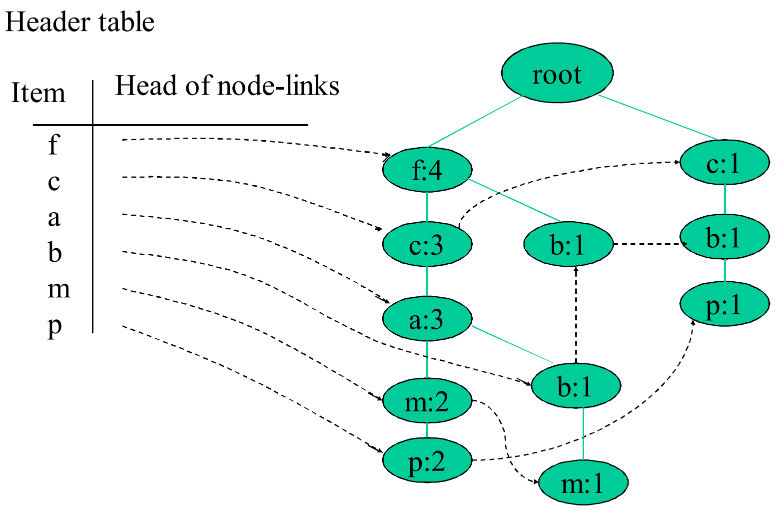
- A highly compact data structure: frequent pattern tree
- An FP-tree-based pattern fragment growth mining method
- The paper on SIGMOD 2000 has spawned many subsequent studies
- Note however when the database is large, it becomes not feasible to put the whole tree into the memory (i.e., FP-tree is usually good for small db only)

Example 1 (minsup=3)

Tx ID	Items Bought	(ordered) Frequent Items
100	f,a,c,d,g,i,m,p	f,c,a,m,p
200	a,b,c,f,l,m,o	f,c,a,b,m
300	b,f,h,j,o	f,b
400	b,c,k,s,p	c,b,p
500	a,f,c,e,l,p,m,n	f,c,a,m,p

List of frequent items:
(f:4),(c:4),(a:3),(b:3),(m:3),(p:3)

FP-tree



Frequent pattern tree (FP-tree)

- 3 parts
  - One root labeled as “null”
  - A set of item prefix subtrees
  - Frequent item header table

FP-tree(cont’)

- Each node in the prefix subtree consists of
  - Item-name
  - Count
  - Node-link
- Each entry in the frequent-item header table consists of
  - Item-name
  - Head of node-link

FP-tree construction: step 1

- Scan the transaction database DB once (the first time), and derives a list of frequent items.
- Sort frequent items in frequency descending order.
- This ordering is important since each path of a tree will follow this order

List of frequent items:
(f:4),(c:4),(a:3),(b:3),(m:3),(p:3)

FP-tree construction: step 2

- Create a root of a tree, label with "null"
- Scan the database the second time.

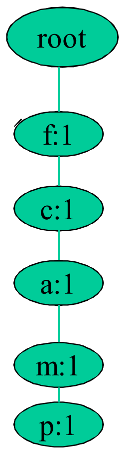
The scan of the first tx leads to the construction of the first branch of the tree.

FP-tree construction: step 2(cont')

Scan of 1st Transaction : f,a,c,d,g,i,m,p

The 1st branch of the tree

<(f:1),(c:1),(a:1),(m:1),(p:1)>



FP-tree construction: step 2(cont')

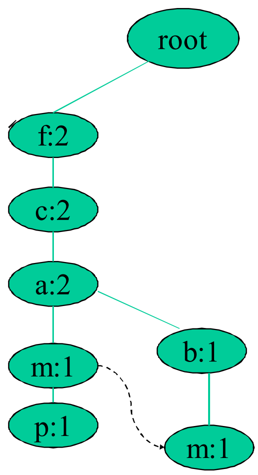
Scan of 2nd Transaction :

a,b,c,f,l,m,o

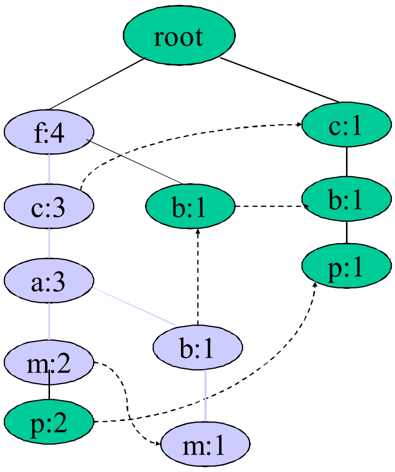
(-> f, c, a, b, m)

Two new nodes:

(b:1) (m:1)



Mining Process: for node “m”

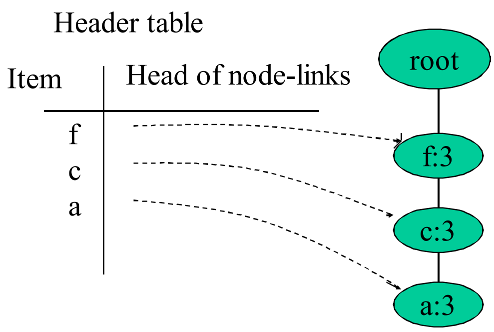


Two paths:
<f:4,c:3,a:3,m:2>
<f:4,c:3,a:3,b:1,m:1>

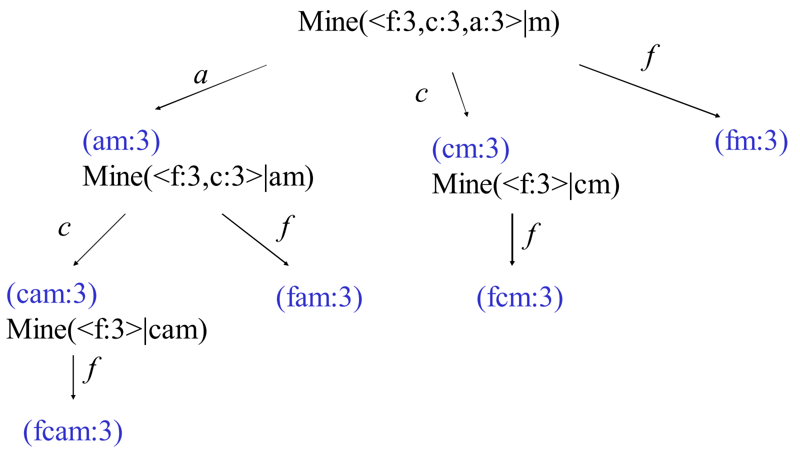
Conditional patterns based of “m”

(f:2, c:2, a:2)
(f:1, c:1, a:1, b:1)

Conditional FP-tree of “m”



Mining m’s conditional FP-tree



Mining m’s conditional FP-tree

- The whole set of frequent patterns

{(m:3),(am:3),(cm:3),(fm:3),(cam:3),(fam:3),(fcam:3),(fcm:3)}

Note: We perform the identification of frequent patterns from the leaf part of the tree first.

Remarks

- Association rule mining starts from 1993 (term used) and has been well received by the community.
- Both Apriori and FP-Tree approaches have spawned many subsequent studies (and variations)

Vertical Mining

Related Papers

- W.-G. Teng, M.-S. Chen and P. S. Yu, ``A Regression-Based Temporal Pattern Mining Scheme for Data Streams," *Proc. of the 29th Intern'l Conf. on Very Large Data Bases (VLDB-2003)*, September 9-12, 2003.
- J. Han, J. Pei, Y. Yin, R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach" DMKD, Vol. 8, No. 1, Jan. 2004. (SIGMOD 2000)
- M. J. Zaki, Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering 12 (3), 372-390, 2000

Frequent Itemset Mining (FIP)

- Notation
  - Items I, Transaction Database T
  - Itemset: combination of distinct items (e.g. AD, CW …)
  - $\sigma(X)$, support of an itemset, is the number of transactions in which X occur
- Find all itemsets with support > *min_sup*

DATABASE

Transcation	Items
1	A C T W
2	C D W
3	A C T W
4	A C D W
5	A C D T W
6	C D T

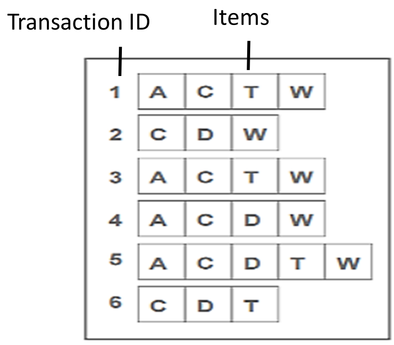
ALL FREQUENT ITEMSETS

MINIMUM SUPPORT = 50%

Support	Itemsets
100% (6)	C
83% (5)	W, CW
67% (4)	A, D, T, AC, AW CD, CT, ACW
50% (3)	AT, DW, TW, ACT, ATW CDW, CTW, ACTW

Database Representation

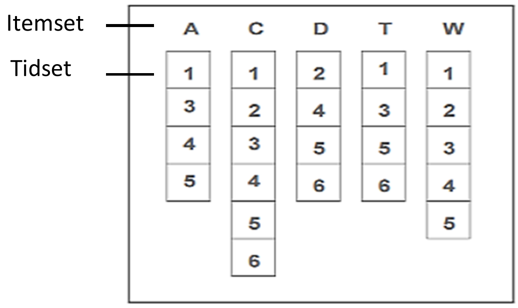
- Horizontal items



Drawback: Need to traverse overall dataset to count support

Database Representation(cont'd)

- Vertical Transaction ID set (Tidset)



Efficient for counting support !

Vertical Mining

Eclat[M. J. Zaki et al., TKDE 2000**]**

- 1. Intersect each two Tidset with the same itemset prefix
- 2. Count support and prune infrequent itemset
- 3. Continue until no new frequent itemset