CS8803 / CS4365 Big Data Systems and Analytics

Lecture 9

Frequent Pattern Mining

Ling Liu
Professor
College of Computing
Georgia Institute of Technology

Adapted from Slides by J. Han,
Data Mining: Concepts and Techniques

Course Administravia

- Project Proposal Due
  - This Friday midnight, graceful no penalty extension to Saturday morning
- Second Homework Assignment
  - Delay the second home starting date to Monday of Week 6 instead of this week

Outline

- Last Lecture: Crowd Computing
- Four Classical Analytic Models
  - Collaborative Filter (User-Based CF, item based CF)
  - Clustering (unsupervised)
  - Classification (Supervised)
  - **Frequent Patten Mining**

Today's Lecture

Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient frequent itemset mining methods
- Constraint-based association mining
- Summary

What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - First proposed by Agrawal, Imielinski, and Swami [AIS93]
- Motivation: **Finding inherent regularities in data**
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - **Pattern analysis** in spatiotemporal, multimedia, time-series, and stream data
  - Classification: **associative classification**
  - Cluster analysis: **frequent pattern-based clustering**
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Frequent Pattern /Association rule mining

- Proposed by Agrawal et al in 1993.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for Market Basket Analysis to find how items purchased by customers are related.

  Bread → Butter [sup = 5%, conf = 100%]

The AR model: data

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of *items*.
- Transaction t :
  - t a set of items, and $t \subseteq I$.
- Transaction Database T: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$.

Transaction data: supermarket data

- Market basket transactions:
  - t1: {bread, cheese, milk}
  - t2: {apple, eggs, salt, yogurt}
  - …   …
  - tn: {biscuit, eggs, milk}
- Concepts:
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Transaction data representation

- A simplistic view of shopping baskets,
- Some important information not considered. E.g,
  - the quantity of each item purchased
  - the price paid
  - … …

Transaction data: a set of documents

- **A text document data set. Each document is treated as a "bag" of keywords**

doc1: Student, Teach, School
doc2: Student, School
doc3: Teach, School, City, Game
doc4: Baseball, Basketball
doc5: Basketball, Player, Spectator
doc6: Baseball, Coach, Game, Team
doc7: Basketball, Team, City, Game

The AR model: rules

- A transaction *t* contains *X*, a set of items (itemset) in *I*, if $X \subseteq t$.
- An association rule is an implication of the form:
  $X \rightarrow Y$, where $X, Y \subset I$, and $X \cap Y = \emptyset$
- Example: {milk, bread, cereal} → {Butter}
- An itemset is a set of items.
  - E.g., X = {milk, bread, cereal} is an itemset.
- A *k*-itemset is an itemset with *k* items.
  - E.g., {milk, bread, cereal} is a 3-itemset

Rule strength measures

- Support: The rule holds with support *sup* in *T* (the transaction data set) if sup% of transactions contain $X \cup Y$.
  - $sup = \Pr(X \cup Y)$.
- Confidence: The rule holds in *T* with confidence *conf* if *conf*% of tranactions that contain *X* also contain *Y*.
  - $conf = \Pr(Y \mid X)$
- An association rule is a pattern that states when *X* occurs, *Y* occurs with certain probability.

Support and Confidence

- Support count: The support count of an itemset *X*, denoted by *X.count*, in a data set *T* is the number of transactions in *T*, which contain *X*. Assume *T* has *n* transactions.
- The Formula:

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Association Rules Example

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

I = { Beer, Bread, Jelly, Milk, PeanutButter}

Support of Bread→PeanutButter is 60%

3/5 transactions buy bread and peanutbutter

Confidence of Bread→PeanutButter is 75%

among 4 buy bread, 3/4 buys bread and peanutbutter

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
  - Completeness: find all rules.
  - No target item(s) on the right-hand-side
  - Mining with data on hard disk (not in memory)

Another example

t1:	Beef, Chicken, Milk
t2:	Beef, Cheese
t3:	Cheese, Boots
t4:	Beef, Chicken, Cheese
t5:	Beef, Chicken, Clothes, Cheese, Milk
t6:	Chicken, Clothes, Milk
t7:	Chicken, Milk, Clothes

- Transaction data
- Assume:
  minsup = 30%
  minconf = 80%

- An example frequent *itemset*:
  {Chicken, Clothes, Milk} [sup = 3/7] ~ 43% > minsup
- Association rules from the itemset:
  Clothes → Milk, Chicken [sup = 3/7~43%, conf = 3/3 = 100%]
  … …
  Clothes, Chicken → Milk, [sup = 3/7 ~ 43%, conf = 3/3 = 100%]

Association Rule Mining: Two Steps

1. Find Large Frequent Itemsets (candidate item sets generation and test).
2. Generate rules from frequent itemsets.

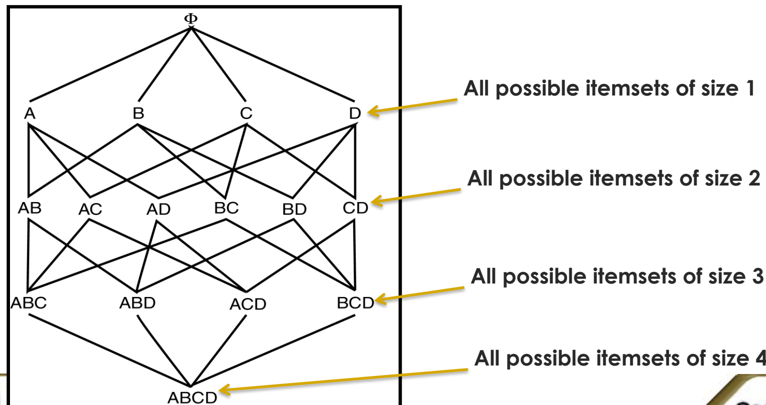
Sup(X→) = Sup(XUY) > minsupport

$$\text{conf}(X \rightarrow Y) = \frac{\text{sup}(X \cup Y)}{\text{sup}(X)} > \text{minconf.}$$

Complexity: A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27 \times 10^{30}$ sub-patterns!

How to find frequent itemsets

- **Naïve Approach**
  - Enumerate all possible itemsets and then count each one



Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient frequent itemset mining methods
- Constraint-based association mining
- Summary

Many FP mining algorithms

- There are a large number of them!!
- They use different **strategies** and **data structures**.
- Their resulting sets of **rules** are all the **same**.
  - Given a transaction data set *T*, and a minimum support and a minimum confident, the set of association rules existing in *T* is uniquely determined.
- Deterministic
  - Any algorithm should find the same set of rules
  - although their computational efficiencies and memory requirements may be different.

Popular Methods for Mining Frequent Patterns

- Three major approaches
  - Apriori (Agrawal & Srikant@VLDB' 94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD' 00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM' 02)
- We study only one in this lecture:
  the Apriori Algorithm

The Apriori algorithm

- **The best known algorithm**
- **Two steps**:
  - Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets).
  - Use frequent itemsets to generate rules.

- E.g., a frequent itemset

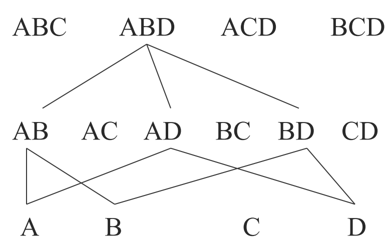
  {Chicken, Clothes, Milk} [sup = 3/7]

  and one rule from the frequent itemset

  Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

Step 1: Mining all frequent itemsets

- A frequent *itemset* is an itemset whose support is ≥ minsup.
- Key idea: The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets



The apriori property

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
- Example:
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}

Apriori: A Candidate Generation-and-Test Approach

- **Apriori pruning principle**:
  - If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB' 94, Mannila, et al. @ KDD' 94)
- **Method:**
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example

$Sup_{min}=2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

The Algorithm

- Iterative algo. (also called level-wise search): Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on.
  - In each iteration *k*, only consider itemsets that contain some *k*-1 frequent itemset.

- Find frequent itemsets of size 1: F_1
- From k = 2 to l /* l *is the size of the total items in T*
  - C_k = candidates of size *k*: those itemsets of size *k* that could be frequent, given F_{k-1}
  - F_k = those itemsets that are actually frequent, $F_k \subseteq C_k$ (need to scan the database once).

Apriori candidate generation

- The candidate-gen function takes $F_{k\text{-}1}$ and returns a superset (called the candidates) of the set of all frequent *k*-itemsets. It has two steps
  - *join* step: Generate all possible candidate itemsets C_k of length *k*
  - *prune* step: Remove those candidates in C_k that cannot be frequent (*using minsupport filter*).

The Apriori Algorithm

- Pseudo-code:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction *t* in database do
    increment the count of all candidates in C_{k+1} that are contained in *t*
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into $\boldsymbol{C_k}$

  select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

  from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$

  where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$
- Step 2: pruning

  forall ***itemsets c in*** $\boldsymbol{C_k}$ do

  forall ***(k-1)-subsets s of c*** do

  if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - L_3=*{abc, abd, acd, ace, bcd}*
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4=*{abcd}*

Another example

- F_3 = {{1, 2, 3}, {1, 2, 4}, {1, 3, 4}, {1, 3, 5}, {2, 3, 4}}
- After join
  - C_4 = {{1, 2, 3, 4}, {1, 3, 4, 5}}
- After pruning:
  - C_4 = {{1, 2, 3, 4}}
    because {1, 4, 5} is not in F_3 ({1, 3, 4, 5} is removed)

Georgia Institute of Technology

33

Georgia Tech College of Computing

Example: Closure of all itemsets

Level 4 (frequent quadruples): {ABDF}

Level 3 (frequent triplets): {ABD}, {BDF}

Level 2 (frequent pairs): {AB}, {AD}, {BF}, {BD}, {DF}

Level 1 (frequent items): {A}, {B}, {D}, {F}

Finding frequent item-sets for a given set of transactions is computationally expensive

Georgia Institute of Technology

Georgia Tech College of Computing

Apriori Optimization: An Example

Level 4 (frequent quadruples): {ABDF}

Level 3 (frequent triplets): {ABD}, {BDF}

Level 2 (frequent pairs): {AB}, {AD}, {BF}, {BD}, {DF}

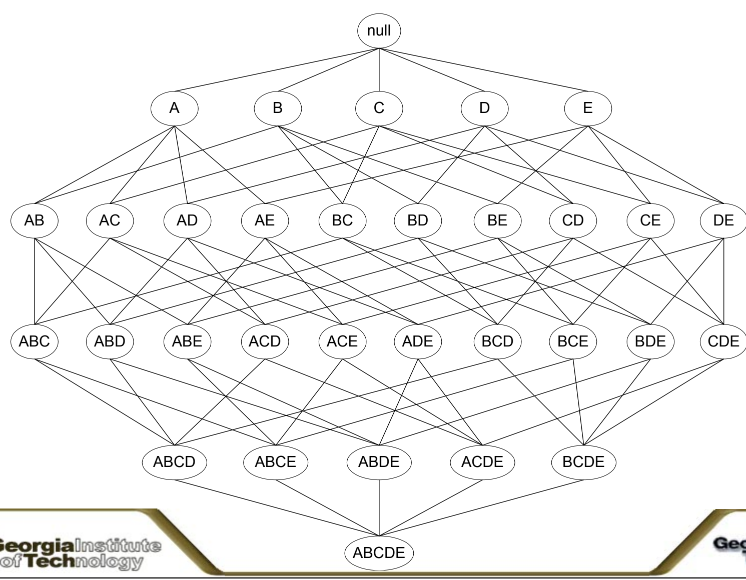
Level 1 (frequent items): {A}, {B}, {D}, {F}

Apriori Principle:
All subsets of a frequent itemset must be frequent

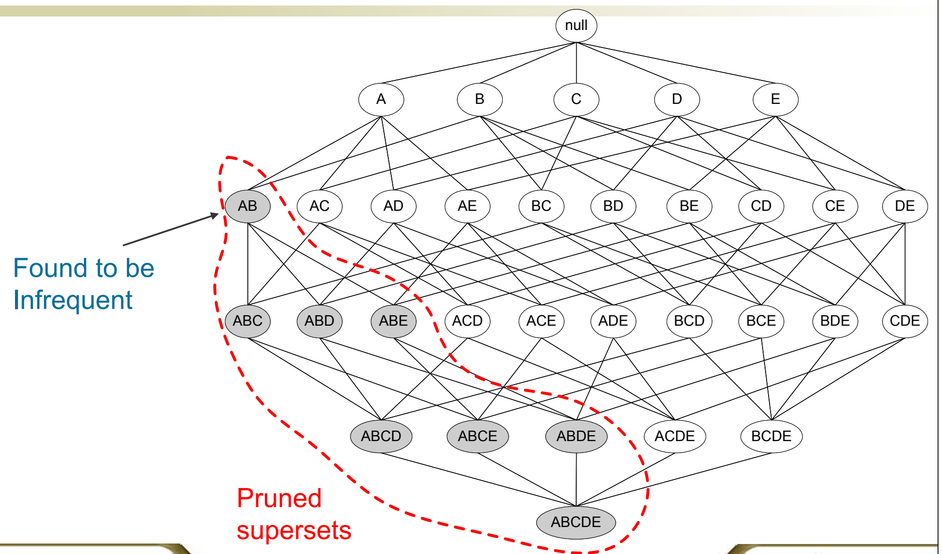
Question: Can ADF be frequent if F is not frequent?

NO: because F (also AF) are not frequent

Frequent Itemset Generation



Illustrating Apriori Principle



Factors Affecting Complexity

- Choice of minimum support threshold
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Georgia Institute of Technology · Georgia Tech College of Computing

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, 14 candidate rules:

ABC →D,	ABD →C,	ACD →B,	BCD →A,
A →BCD,	B →ACD,	C →ABD,	D →ABC
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		

- If |L| = k, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

Generating rules: an example

- Suppose {2,3,4} is frequent, with sup=50%
  - Proper nonempty subsets: {2,3}, {2,4}, {3,4}, {2}, {3}, {4}, with sup=50%, 50%, 75%, 75%, 75%, 75% respectively
  - The item set {2,3,4} generate 6 ($2^3 - 2$) association rules:
    - 2,3 → 4, confidence=100%
    - 2,4 → 3, confidence=100%
    - 3,4 → 2, confidence=67%
    - 2 → 3,4, confidence=67%
    - 3 → 2,4, confidence=67%
    - 4 → 2,3, confidence=67%
    - All rules have support = 50%

Rule Generation

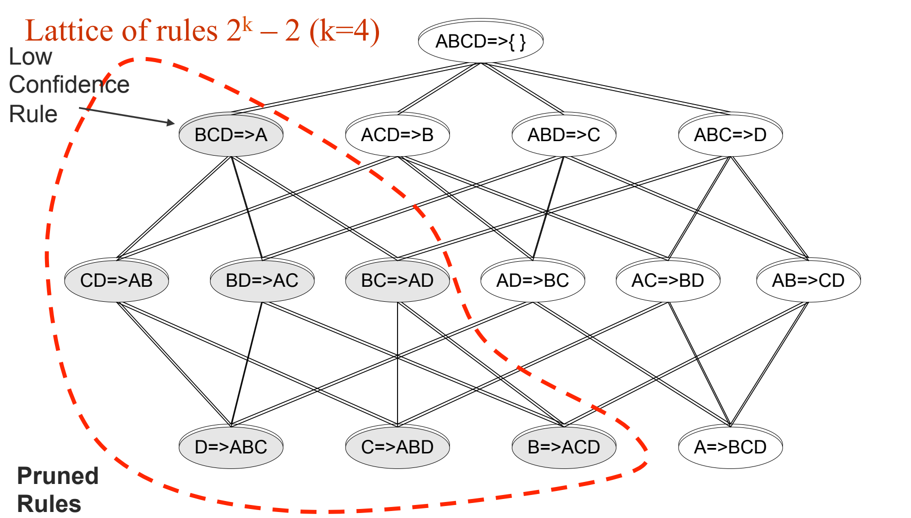
- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

    $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$

    - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm



Apriori Summary: Reducing Association Rule Complexity

- Two properties are used to reduce the search space for association rule generation.
  - **Downward Closure**
    - A subset of a large itemset must also be large
  - **Anti-monotonicity**
    - A superset of a small itemset is also small. This implies that the itemset does not have sufficient confidence-support to be considered for rule generation.

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- **Improving Apriori: general ideas**
  - **Reduce passes of transaction database scans**
  - **Shrink number of candidates**
  - **Facilitate support counting of candidates**

Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find missed frequent patterns

H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

Bottleneck of Frequent-pattern Mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1 i_2 \ldots i_{100}$
    - # of scans: 100
    - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100}-1 = 1.27 \times 10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Constraint-based association mining (supervised learning)
- Summary

Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: explores such constraints for efficient mining—**constraint-based mining**

Constraints in Data Mining

- **Knowledge type constraint**:
  - classification, association, etc.
- **Data constraint** — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.' 02
- **Dimension/level constraint**
  - in relevance to region, price, brand, customer category
- **Rule (or pattern) constraint**
  - small sales (price < $10) triggers big sales (sum > $200)
- **Interestingness constraint**
  - strong rules: min_support ≥ 3%, min_confidence ≥ 60%

Georgia Institute of Technology | Georgia Tech College of Computing

Constrained FP Mining

- Constrained mining vs. **constraint-based search/ reasoning**
  - Both are aimed at reducing search space
  - Finding all patterns satisfying constraints vs. finding some (or one) answer in constraint-based search in AI
  - Constraint-pushing vs. heuristic search
  - An interesting research problem: how to integrate them
- Constrained mining vs. **query processing in DBMS**
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing

Georgia Institute of Technology | Georgia Tech College of Computing

The Apriori Algorithm — Example

$Sup_{min}=1$

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D → C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

→ L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D → C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

→ L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D → L_3

itemset	sup
{2 3 5}	2

Naïve Algorithm: Apriori + Constraint

$Sup_{min}=2$

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D → C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

→ L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
~~{1 5}~~
{2 3}
~~{2 5}~~
~~{3 5}~~

Scan D → C_2

itemset	sup
{1 2}	1
{1 3}	2
~~{1 5}~~	~~1~~
{2 3}	2
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

→ L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D → L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price} < 5

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Mining sequential and structured patterns
- Extensions and applications

Frequent-Pattern Mining: Research Problems

- Mining fault-tolerant frequent, sequential and structured patterns
  - Patterns allows limited faults (insertion, deletion, mutation)
- Mining truly interesting patterns
  - Surprising, novel, concise, …
- Application exploration
  - E.g., DNA sequence analysis and bio-pattern classification
  - "Invisible" data mining

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Constrained FPM
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Different data formats for mining

- The data can be in transaction form or table form

Transaction form:

a, b
a, c, d, e
a, d, f

Table form:

Attr1	Attr2	Attr3
a,	b,	d
b,	c,	e

- Table data need to be converted to transaction form for association mining

From a table to a set of transactions

Table form:

Attr1	Attr2	Attr3
a,	b,	d
b,	c,	e

⇒ Transaction form:

(Attr1, a), (Attr2, b), (Attr3, d)

(Attr1, b), (Attr2, c), (Attr3, e)

candidate-gen can be slightly improved. Why?

Georgia Institute of Technology

Georgia Tech College of Computing

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Georgia Institute of Technology

Georgia Tech College of Computing

Problems with the association mining

- Single minsup: It assumes that all items in the data are of the same nature and/or have similar frequencies.
- Not true: In many applications, some items appear very frequently in the data, while others rarely appear.

  E.g., in a supermarket, people buy *food processor* and *cooking pan* much less frequently than they buy *bread* and *milk*.

Rare Item Problem

- If the frequencies of items vary a great deal, we will encounter two problems
  - If minsup is set too high, those rules that involve rare items will not be found.
  - To find rules that involve both frequent and rare items, minsup has to be set very low. This may cause combinatorial explosion because those frequent items will be associated with one another in all possible ways.

Multiple minsups model

- The minimum support of a rule is expressed in terms of *minimum item supports* (MIS) of the items that appear in the rule.
- Each item can have a minimum item support.
- By providing different MIS values for different items, the user effectively expresses different support requirements for different rules.
- To prevent very frequent items and very rare items from appearing in the same itemsets, we introduce a **support difference constraint**.

  $max_{i \in s}\{sup\{i\} - min_{i \in s}\{sup(i)\} \leq \varphi$,

Minsup of a rule

- Let MIS(i) be the MIS value of item i. The *minsup* of a rule R is the lowest MIS value of the items in the rule.
- I.e., a rule R: $a_1, a_2, \ldots, a_k \rightarrow a_{k+1}, \ldots, a_r$ satisfies its minimum support if its actual support is ≥

  $\min(\text{MIS}(a_1), \text{MIS}(a_2), \ldots, \text{MIS}(a_r))$.

An Example

- Consider the following items:

  bread, shoes, clothes

  The user-specified MIS values are as follows:

  MIS(*bread*) = 2%   MIS(*shoes*) = 0.1%

  MIS(*clothes*) = 0.2%

  The following rule doesn't satisfy its minsup:

  clothes → *bread* [sup=0.15%,conf =70%]

  The following rule satisfies its minsup:

  clothes → *shoes* [sup=0.15%,conf =70%]

Downward closure property

- In the new model, the property no longer holds (?)

E.g., Consider four items 1, 2, 3 and 4 in a database. Their minimum item supports are

MIS(1) = 10%   MIS(2) = 20%

MIS(3) = 5%   MIS(4) = 6%

{1, 2} with support 9% is infrequent, but {1, 2, 3} and {1, 2, 4} could be frequent.

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Mining class association rules (CAR)

- Normal association rule mining does not have any target.
- It finds all possible rules that exist in data, i.e., any item can appear as a consequent or a condition of a rule.
- However, in some applications, the user is interested in some targets.
  - E.g, the user has a set of text documents from some known topics. He/she wants to find out what words are associated or correlated with each topic.

Problem definition

- Let T be a transaction data set consisting of n transactions.
- Each transaction is also labeled with a class y.
- Let I be the set of all items in T, Y be the set of all class labels and $I \cap Y = \emptyset$.
- A **class association rule** (**CAR**) is an implication of the form

  $X \rightarrow y$, where $X \subseteq I$, and $y \in Y$.
- The definitions of **support** and **confidence** are the same as those for normal association rules.

Georgia Institute of Technology

67

Georgia Tech College of Computing

An example

- **A text document data set**

doc 1:	Student, Teach, School	: Education
doc 2:	Student, School	: Education
doc 3:	Teach, School, City, Game	: Education
doc 4:	Baseball, Basketball	: Sport
doc 5:	Basketball, Player, Spectator	: Sport
doc 6:	Baseball, Coach, Game, Team	: Sport
doc 7:	Basketball, Team, City, Game	: Sport

- Let *minsup* = 20% and *minconf* = 60%. The following are two examples of class association rules:

  Student, School $\rightarrow$ Education [sup= 2/7, conf = 2/2]

  game $\rightarrow$ Sport [sup= 2/7, conf = 2/3]

Georgia Institute of Technology

68

Georgia Tech College of Computing

Mining algorithm

- Unlike normal association rules, CARs can be mined directly in one step.
- The key operation is to find all **ruleitems** that have support above *minsup*. A **ruleitem** is of the form:

  (*condset*, *y*)

  where **condset** is a set of items from *I* (*i.e., condset* $\subseteq I$), and $y \in Y$ is a class label.
- Each ruleitem basically represents a rule:

  condset $\rightarrow$ *y*,
- The Apriori algorithm can be modified to generate CARs

Multiple minimum class supports

- The multiple minimum support idea can also be applied here.
- The user can specify different minimum supports to different classes, which effectively assign a different minimum support to rules of each class.
- For example, we have a data set with two classes, Yes and No. We may want
  - rules of class Yes to have the minimum support of 5% and
  - rules of class No to have the minimum support of 10%.
- By setting minimum class supports to 100% (or more for some classes), we tell the algorithm not to generate rules of those classes.
  - This is a very useful trick in applications.

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Sequential pattern mining

- Association rule mining does not consider the order of transactions.
- In many applications such orderings are significant. E.g.,
  - in market basket analysis, it is interesting to know whether people buy some items in sequence,
    - e.g., buying bed first and then bed sheets some time later.
  - In Web usage mining, it is useful to find navigational patterns of users in a Web site from sequences of page visits of users

Basic concepts

- Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items.
- **Sequence:** An ordered list of itemsets.
- **Itemset/element**: A non-empty set of items $X \subseteq I$. We denote a sequence s by $\langle a_1 a_2 \ldots a_r \rangle$, where a_i is an itemset, which is also called an **element** of s.
- An element (or an itemset) of a sequence is denoted by $\{x_1, x_2, \ldots, x_k\}$, where $x_j \in I$ is an item.
- We assume without loss of generality that items in an element of a sequence are sorted in **lexicographic order**.

Basic concepts (contd)

- **Size**: The **size** of a sequence is the number of elements (or itemsets) in the sequence.
- **Length**: The **length** of a sequence is the number of distinct items in the sequence.
  - A sequence of length k is called ***k*-sequence**.
- A sequence $s_1 = \langle a_1 a_2 \ldots a_r \rangle$ is a **subsequence** of another sequence $s_2 = \langle b_1 b_2 \ldots b_v \rangle$, if r <=v and there exist integers $1 \le j_1 < j_2 < \ldots < j_{r-1} < j_r \le v$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, $\ldots$, $a_r \subseteq b_{jr}$. We also say that s_2 **contains** s_1.

An example

- Let I = {1, 2, 3, 4, 5, 6, 7, 8, 9}.
- Sequence ⟨{3}{4, 5}{8}⟩ is **contained** in (or is a **subsequence** of) ⟨{6} {3, 7}{9}{4, 5, 8}{3, 8}⟩
  - because {3} ⊆ {3, 7}, {4, 5} ⊆ {4, 5, 8}, and {8} ⊆ {3, 8}.
  - The size of the sequence ⟨{3}{4, 5}{8}⟩ is 3, and the length of the sequence is 4.
  - However, ⟨{3, 8}⟩ is not contained in ⟨{3}{8}⟩.

Objective

- Given a set S of input data sequences (or sequence database), the problem of mining sequential patterns is to find all the sequences that have a user-specified minimum support.
- Each such sequence is called a **frequent sequence**, or a **sequential pattern**.
- The **support** for a sequence is the fraction of total data sequences in S that contains this sequence.

Example

Table 1. A set of transactions sorted by customer ID and transaction time

Customer ID	Transaction Time	Transaction (items bought)
1	July 20, 2005	30
1	July 25, 2005	90
2	July 9, 2005	10, 20
2	July 14, 2005	30
2	July 20, 2005	40, 60, 70
3	July 25, 2005	30, 50, 70
4	July 25, 2005	30
4	July 29, 2005	40, 70
4	August 2, 2005	90
5	July 12, 2005	90

Example (cond)

Table 2. Data sequences produced from the transaction database in Table 1.

Customer ID	Data Sequence
1	⟨{30} {90}⟩
2	⟨{10, 20} {30} {40, 60, 70}⟩
3	⟨{30, 50, 70}⟩
4	⟨{30} {40, 70} {90}⟩
5	⟨{90}⟩

Table 3. The final output sequential patterns

	Sequential Patterns with Support ≥ 25%
1-sequences	⟨{30}⟩, ⟨{40}⟩, ⟨{70}⟩, ⟨{90}⟩
2-sequences	⟨{30} {40}⟩, ⟨{30} {70}⟩, ⟨{30} {90}⟩, ⟨{40, 70}⟩
3-sequences	⟨{30} {40, 70}⟩

GSP mining algorithm

- Very similar to the Apriori algorithm

```
Algorithm GSP(S)
C1 ← init-pass(S);                              // the first pass over S
F1 ← {⟨{f}⟩| f ∈ C1, f.count/n ≥ minsup};       // n is the number of sequences in S
for (k = 2; Fk–1 ≠ ∅; k++) do                   // subsequent passes over S
    Ck ← candidate-gen-SPM(Fk–1);
    for each data sequence s ∈ S do             // scan the data once
        for each candidate c ∈ Ck do
            if c is contained in s then
                c.count++;                      // increment the support count
        end
    end
    Fk ← {c ∈ Ck | c.count/n ≥ minsup}
end
return ∪k Fk;
```

Fig. 12. The GSP Algorithm for generating sequential patterns

Candidate generation

Function candidate-gen-SPM(F_{k-1})

1. **Join step.** Candidate sequences are generated by joining F_{k-1} with F_{k-1}. A sequence s_1 joins with s_2 if the subsequence obtained by dropping the first item of s_1 is the same as the subsequence obtained by dropping the last item of s_2. The candidate sequence generated by joining s_1 with s_2 is the sequence s_1 extended with the last item in s_2. There are two cases:
   - the added item forms a separate element if it was a separate element in s_2, and is appended at the end of s_1 in the merged sequence, and
   - the added item is part of the last element of s_1 in the merged sequence otherwise.

   When joining F_1 with F_1, we need to add the item in s_2 both as part of an itemset and as a separate element. That is, joining $\langle\{x\}\rangle$ with $\langle\{y\}\rangle$ gives us both $\langle\{x, y\}\rangle$ and $\langle\{x\}\{y\}\rangle$. Note that x and y in $\{x, y\}$ are ordered.
2. **Prune step**. A candidate sequence is pruned if any one of its $(k-1)$-subsequence is infrequent (without minimum support).

Fig. 13. The candidate-gen-SPM() function

An example

Table 4. Candidate generation: an example

Frequent 3-sequences	Candidate 4-sequences after joining	Candidate 4-sequences after pruning
⟨{1, 2} {4}⟩	⟨{1, 2} {4, 5}⟩	⟨{1, 2} {4, 5}⟩
⟨{1, 2} {5}⟩	⟨{1, 2} {4} {6}⟩	
⟨{1} {4, 5}⟩		
⟨{1, 4} {6}⟩		
⟨{2} {4, 5}⟩		
⟨{2} {4} {6}⟩		

Summary

- Association rule mining has been extensively studied in the data mining community.
- So is sequential pattern mining
- There are many efficient algorithms and model variations.
- Other related work includes
  - Multi-level or generalized rule mining
  - Constrained rule mining
  - Incremental rule mining
  - Maximal frequent itemset mining
  - Closed itemset mining
  - Rule interestingness and visualization
  - Parallel algorithms
  - …

Questions