ANU MLSS 2010: Data Mining

Part 2: Association rule mining

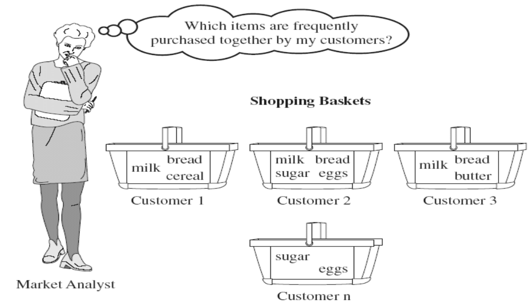
Lecture outline

- What is association mining?
- Market basket analysis and association rule examples
- Basic concepts and formalism
- Basic rule measurements
- The *Apriori* algorithm
- Performance bottlenecks in *Apriori*
- Multi-level and multi-dimensional association mining
- Quantitative association mining
- Constraint based mining
- Visualising association rules

What is association mining?

- Association mining is the task of finding frequent rules / associations / patterns / correlations / causual structures within (large) sets of items in transactional (relational) databases
- *Unsupervised* learning techniques (*descriptive* data mining, not *predictive* data mining)
- The main applications are
  - Market basket analysis (customers who buys X also buys Y)
  - Web log analysis (click-stream)
  - Cross-marketing
  - Sale campaign analysis
  - DNS sequence analysis

Market basket analysis



Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

Association rules examples

- Rules form: *body* ⇒ *head [support, confidence]*
- Market basket:
  buys(X, `beer') ⇒ buys(X, `snacks') [1%, 60%]
  - If a customer X purchased `beer', in 60% she or he also purchased `snacks'
  - 1% of all transactions contain the items `beer' and `snacks'
- Student grades:
  major(X, `MComp') and takes(X, `COMP8400') ⇒ grade(X, `D') [3%, 60%] *
  - If a student X, who's degree is `MComp', took the course `COMP8400' she or he in 60% achieved a grade `D'
  - The combination `MComp', `COMP8400' and `D' appears in 3% of all transactions (records) in the database

Basic concepts

- Given:
  - A (large) database of transactions
  - Each transaction contains a list of one or more items (e.g. purchased by a customer in a visit)
- Find the rules that correlate the presence of one set of items with that of another set of items
- Normally one is only interested in rules that are *frequent*
  - For example, 70% of customers who buy tires and car accessories also get their car service done

    Question: How can this be improved to 80%? Possibly offer special deals like a 15% reduction of tire costs when the service is done

Formalism

- Set of items $X = \{x_1, x_2, ..., x_k\}$
- Database D containing transactions
- Each transaction T is a set of items, such that T is a subset of X
- Each transaction is associated with a unique identifier, called *TID* (for example, a unique number)
- Let A be a set of items (a subset of X)
- An association rule is an implication of the form $A \Rightarrow B$, where A is a subset of X and B is a subset of X, and the intersection of A and B is empty
  - No item in A can be in B, and vice versa
  - No rule of the form: {`beer', `chips'} $\Rightarrow$ {`chips', `peanuts'}

Basic rule measurements

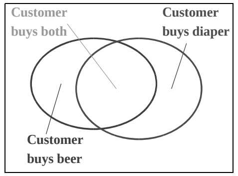
- A rule $A \Rightarrow B$ holds in a database D with *support s,* with s being the percentage of transactions in D that contain A and B

  support($A \Rightarrow B$) = P(A U B)
- The rule $A \Rightarrow B$ has a *confidence c* in a database D if c is the percentage of transactions in D containing A that also contain B

  confidence($A \Rightarrow B$) = P(B|A) = P(A U B) / P(A)

  confidence($A \Rightarrow B$) = support($A \Rightarrow B$) / support(A)

Rule measurements example



- Find all the rules {X, Y} $\Rightarrow$ Z with minimum confidence and support
- Support, s, is the probability that a transaction contains {X, Y, Z}
- Confidence, c, is the conditional probability that a transaction having {X, Y} also contains Z

Transaction ID	Items Bought
2000	a, b, c
1000	a, c
4000	a, d
5000	b, e, f

Let minimum support = 50%, and minimum confidence = 50%, so we have ([s, c]):

- a $\Rightarrow$ c [50%, 66.67%]
- c $\Rightarrow$ a [50%, 100%]

Source: Han and Kamber, DM Book, 1st Ed.

Rule measurements example (2)

Transaction ID	Items Bought
2000	a, b, c
1000	a, c
4000	a, d
5000	b, e, f

Itemset	Support
a	75.00%
b	50.00%
c	50.00%
a, c	50.00%

- Minimum support = 50% and confidence = 50%
- Rule a $\Rightarrow$ c
  - support (a $\Rightarrow$ c): 50%
  - confidence (a $\Rightarrow$ c) = support(a $\Rightarrow$ c) / support(a) = 50% / 75% = 66.67%

Mining frequent item sets

- Key step: Find the *frequent sets of items* that have *minimum support* (appear in at least xx% of all transactions in a database)
- Basic principle (*Apriori* principle): A sub-set of a frequent item set must also be a frequent item set
  - For example, if {a,b} is frequent, both {a} and {b} have to be frequent (if `beer' and 'chips' are purchased frequently together, then `beer' is purchased frequently and `chips' are also purchased frequently)
- Basic approach: Iteratively find frequent item sets with cardinality from 1 to k (k-item sets), $k > 1$
- Use the frequent item sets to generate association rules
  - For example, frequent 3-item set {a,b,c} contains rules: a $\Rightarrow$ c, b $\Rightarrow$ c, a $\Rightarrow$ b, {a,b} $\Rightarrow$ c, {a,c} $\Rightarrow$ b, {b,c} $\Rightarrow$ a, etc.
- We are normally only interested in longer rules (with all except one element on the left-hand side)

The *Apriori* algorithm (Agrawal & Srikant, VLDB'94)

- C_k : Candidate item set of size k

  L_k : Frequent item set of size k
- Pseudo-code:

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 do
return $_k L_k$;

The *Apriori* algorithm – An example (sup=50%)

Database D

TID	Items
100	a,c,d
200	b,c,e
300	a,b,c,e
400	b,e

Scan D

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}

Scan D

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}

Scan D

C_3

itemset	sup
{b, c, e}	2

L_3

itemset	sup
{b, c, e}	2

The *Apriori* algorithm – An example (2)

Database D

TID	Items
100	a,c,d
200	b,c,e
300	a,b,c,e
400	b,e

L_3

itemset	sup
{b, c, e}	2

- Minimum support = 50% and minimum confidence = 50%
- Rules:
  - b ⇒ c [50%, 66.67%]
  - b ⇒ e [75%, 100%]
  - c ⇒ e [50%, 66.67%]
  - {b, c} ⇒ e [50%, 100%]
  - {b, e} ⇒ c [50%, 66.67%]
  - {c, e} ⇒ b [50%, 100%]

Important details of the *Apriori* algorithm

- How to generate candidate sets?
  - Step 1: Self-joining L_k (C_k is generated by joining L_{k-1} with itself)
  - Step 2: Pruning (any *(k-1)*-item set that is not frequent cannot be a subset of a frequent *k*-item set)
- Example of candidate generation:
  - L_3 = *{{a,b,c}, {a,b,d}, {a,c,d}, {a,c,e}, {b,c,d}}*
  - Self-joining: $L_3 * L_3$ (*{a,b,c,d}* from *{a,b,c}* and *{a,b,d}*, and *{a,c,d,e}* from *{a,c,d}* and *{a,c,e}*)
  - Pruning: *{a,c,d,e}* is removed because *{a,d,e}* is not in L_3
  - C_4 = {*{a,b,c,d}*}
- How to count supports for candidates?

How to generate candidate item-sets?

- Suppose the items in L_{k-1} are listed in an order (e.g. *a* < *b*)
- Step 1: Self-joining L_{k-1}

  insert into C_k

  select $p.item_1$, $p.item_2$, …, $p.item_{k-1}$, $q.item_{k-1}$

  from L_{k-1} p, L_{k-1} q

  where $p.item_1 = q.item_1$, …, $p.item_{k-2} = q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$
- Step 2: Pruning

  forall item sets c **in** C_k **do**

  forall (k-1)-sub-sets s of c **do**

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

Apriori performance bottlenecks

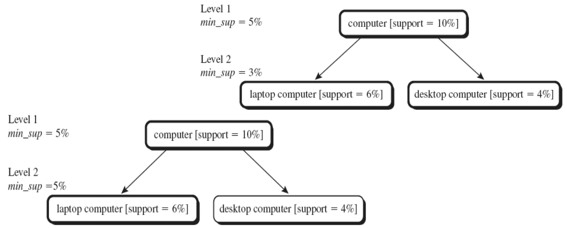
- The core of the *Apriori* algorithm is to
  - Use frequent *(k-1)* item sets to generate candidate frequent *k* item sets
  - Use database scan and pattern matching to collect counts for candidate item sets
- Candidate generation is the main bottleneck
  - 10^4 frequent 1-item sets (sets of length 1) will generate 10^7 candidate 2-item sets!
  - To discover a frequent pattern of size 100 (for example $\{a_1, a_2, ..., a_{100}\}$) one needs to generate $2^{100} = 10^{30}$ candidates
  - Multiple scans of the database are needed (*n+1* scans if the longest pattern is *n* items long)

Methods to improve *Apriori's* efficiency

- Reduce the number of scans of the database
  - Any item set that is potentially frequent in the database must be frequent in at least one of the partitions of the database
  - Scan 1: Partition database and find local frequent patterns
  - Scan 2: Consolidate global frequent patterns
- Shrink number of candidates
  - Select a sample of the database, mine frequent patterns within sample using *Apriori*
  - Scan database once to verify frequent item sets found in sample
  - Scan database again to find missed frequent patterns
- Facilitate support of counting candidates
  - For example, use special data structures like Frequent-Pattern tree (FP-tree)

Multi-level association mining

- Items often form hierarchies
- Items at lower levels are expected to have lower support
  - Flexible *support* setting (uniform, reduced, or group-based (user specific))



Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

Multi-level association mining (2)

- Some rules may be redundant due to *ancestor* relationships between items
- For example:
  buys(X, `milk')* $\Rightarrow$ *buys(X, `bread')   [8%, 70%]
  buys(X, `skim milk')* $\Rightarrow$ *buys(X, `bread')   [2%, 72%]
  - The first rule is said to be an *ancestor* of the second rule
- A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor
  - For example, if around 25% of all milk purchased is `skim milk', then the second rule above is redundant, as it has a ¼ of the support of the first, more general rule (and similar confidence)

Multi-dimensional association mining

- Single-dimensional rules: *buys(X, `milk')* $\Rightarrow$ *buys(X, `bread')*
- Multi-dimensional rules: Two or more dimensions or predicates (or attributes)
  - Inter-dimension association rules (*no repeated predicates*): *age(X, `19-25')* and *occupation(X, `student')* $\Rightarrow$ *buys(X, `coke')*
  - Hybrid-dimension association rules (*repeated predicates*): *age(X, `19-25')* and *buys(X, `popcorn')* $\Rightarrow$ *buys(X, `coke')*
- Categorical Attributes: finite number of possible values, no ordering among values (data cube approach)
- Quantitative Attributes: numeric, implicit ordering among values (discretisation, clustering, etc.)

Quantitative association mining

- Techniques can be categorised by how numerical attributes, such as *age* or *income,* are treated
- Static discretisation based on predefined concept hierarchies
- Dynamic discretisation based on data distribution
  - A_{quant1} and $A_{quant2} \Rightarrow A_{cat}$
  - Example: *age(X, `19-25')* and *income(X, `40K-60K')* $\Rightarrow$ *buys(X, `HDTV')*
- For quantitative rules, do discretisation such that (for example) the confidence of the rules mined is maximised

Mining interesting correlation patterns

- Flexible support
  - Some items might be very rare but are valuable (like diamonds)
  - Customise $support_{min}$ specification and application
- Top-*k* frequent patterns
  - It can be hard to specify $support_{min}$, but top-*k* rules with $length_{min}$ are more desirable
  - Achievable using special data structures, like Frequent-Pattern (FP) tree
  - Dynamically raise $support_{min}$ during FP-tree construction phase, and select most promising to mine

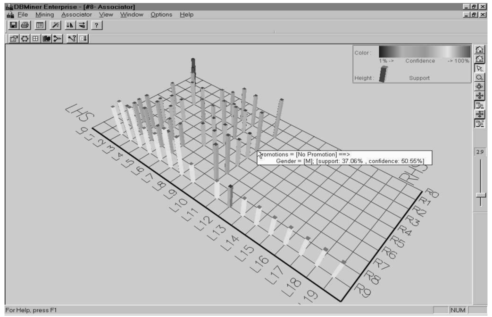
Constraint based data mining

- Finding *all* the frequent rules or patterns in a database autonomously is unrealistic
  - The rules / patterns could be too many and not focussed
- Data mining should be an *interactive* process
- The user directs what should 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 (and what not)
  - System optimisation: explores such constraints for efficient mining

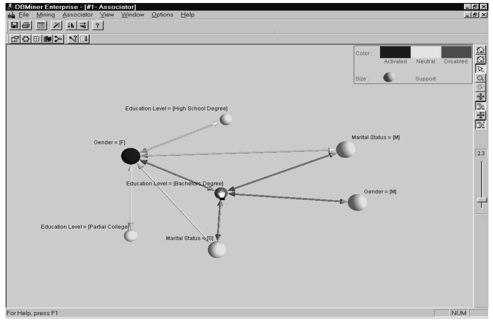
Constraints in data mining

- Knowledge type constraint
  - Correlation, association, etc.
- Data constraint (use SQL like queries)
  - For example: *Find product pairs sold frequently in both stores in Sydney and Melbourne*
- Dimension / level constraint
  - In relevance to region, price, brand, customer category, etc.
- Rule or pattern constraint
  - Small sales (price < $10) trigger big sales (sum > $200)
- Interestingness constraint
  - Strong rules only: $support_{min}$ > 3%, $confidence_{min}$ > 75%

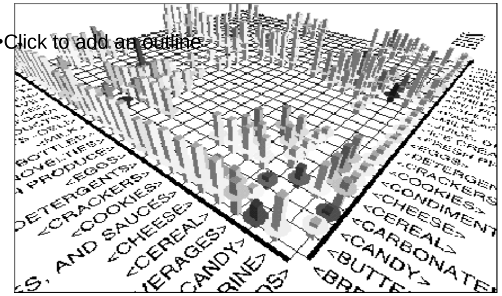
Visualisation of association rules (1)



Visualisation of association rules (2)



Visualisation of association rules (3)



•Click to add an outline