PQHS 471

Lecture 4: Unsupervised Learning (3)

Frequent Pattern Mining

Market Basket Analysis

The goals of **market basket analysis** are to mine:

- *Frequent item sets:* Find items that are frequently purchased together.
- *Association rules:* Find simple prediction models (called association rules) that have both good frequency (i.e., high support) and good prediction (i.e., high confidence and/or lift).

Transaction Data

A special type of record data, where each record (transaction) involves a set of items.
For example, the set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

TID	***Items***
1	**Bread, Coke, Milk**
2	**Beer, Bread**
3	**Beer, Coke, Diaper, Milk**
4	**Beer, Bread, Diaper, Milk**
5	**Coke, Diaper, Milk**

Frequent Pattern Analysis

Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set.

Frequent Pattern Analysis

Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set.
Motivation: Finding inherent regularities in data

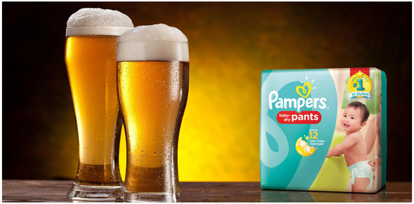
- What products were often purchased together?— Beer and diapers?!
- What are the subsequent purchases after buying a PC?
- What kinds of genomic components are sensitive to this new drug?
- Can we automatically classify web documents?

Frequent Itemset Mining

Frequent Itemset Mining: Frequent set of items in a transaction data set.

Frequent Itemset Mining

Frequent Itemset Mining: Frequent set of items in a transaction data set.



Walmart Friday night sale data mining.

Basic Concepts

- *Itemset:* $X = \{x_1, x_2, ..., x_k\}$ (k-itemset)
- *Support count:* (absolute support) count of transactions containing X
- *Frequent itemset:* Those X with at least minimum support count chosen.

Basic Concepts

- *Itemset:* $X = \{x_1, x_2, ..., x_k\}$ (k-itemset)
- *Support count:* (absolute support) count of transactions containing X
- *Frequent itemset:* Those X with at least minimum support count chosen.

Association rule: $A \rightarrow B$ with minimum support and confidence

- *Support:* probability that a transaction contains $A \cup B$

$$s = P(A \cup B)$$

- *Confidence:* conditional probability that a transaction having A also contains B.

$$c = P(B|A)$$

Examples

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

Frequent itemsets (minimum support count $= 3$) ?

Examples

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

Frequent itemsets (minimum support count = 3) ?

$$\{A : 3, B : 3, D : 4, E : 3, AD : 3\}$$

Association rules (minimum support = 50%, minimum confidence = 50%) ?

Examples

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

Frequent itemsets (minimum support count = 3) ?

$$\{A : 3, B : 3, D : 4, E : 3, AD : 3\}$$

Association rules (minimum support = 50%, minimum confidence = 50%) ?

$$A \rightarrow D(60\%, 100\%)$$

$$D \rightarrow A(60\%, 75\%)$$

Frequent itemset mining

- Brute force approach? Enumerating all possible itemsets?

Frequent itemset mining

- Brute force approach? Enumerating all possible itemsets? $\times$
- Set enumeration tree $\checkmark$
- Apriori (Agrawal & Srikant @VLDB'94) and variations
- Frequent pattern growth (FPgrowth—Han, Pei& Yin @SIGMOD'00)

Apriori — Apriori Property

- Any nonempty subset of a frequent itemset must be frequent
  If $\{beer, diaper, nuts\}$ is frequent, so is $\{beer, diaper\}$

Apriori — Apriori Property

- Any nonempty subset of a frequent itemset must be frequent
  If $\{beer, diaper, nuts\}$ is frequent, so is $\{beer, diaper\}$
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Bottom up search strategy

Apriori — Level-Wise Search Method

- Initially, scan database once to get frequent 1-itemset
- Generate length (k+1) candidate itemsets from length k frequent itemsets
- Test the candidates against database
- Terminate when no frequent or candidate set can be generated

Apriori algorithm

1. Search all singletons (item sets of size 1) to create L_1
2. For $m = 2, 3, ...,$, generate L_m:
   - Join step: For any two sets $p, q \in L_{m-1}$ that share $m-2$ items, create their union C;
   - Prune step: If C has a size-$m-1$ subset that is not in L_{m-1}, drop C;
   - Check support: If C has support $> t$, keep it.
3. For every item set C with support $> t$, consider every possible split of C into two subsets, A and B, and if the confidence (and/or lift) of $A \rightarrow B$ is above a predetermined threshold, keep it.

Apriori algorithm

1. Search all singletons (item sets of size 1) to create L_1
2. For $m = 2, 3, ...,$, generate L_m:
   - Join step: For any two sets $p, q \in L_{m-1}$ that share $m-2$ items, create their union C;
   - Prune step: If C has a size-$m-1$ subset that is not in L_{m-1}, drop C;
   - Check support: If C has support $> t$, keep it.
3. For every item set C with support $> t$, consider every possible split of C into two subsets, A and B, and if the confidence (and/or lift) of $A \rightarrow B$ is above a predetermined threshold, keep it.

Note: Here the first two steps are to mine frequent item sets, and the third step is to mine association rules.

Apriori algorithm

<u>Pseudo-code</u>:

C_k: Candidate k-itemset
L_k : frequent k-itemset

L_1 = frequent 1-itemsets;
for (k = 2; L_{k-1} !=$\varnothing$; k++)
  C_k = generate candidate set from L_{k-1};
  for each transaction t in database
    find all candidates in C_k that are subset of t;
    increment their count;
  L_k = candidates in C_k with min_support
return $\cup_k L_k$;

Apriori algorithm example

$Sup_{min} = 2$

Transaction DB

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

Candidate set generation and pruning

Example: Generate C_4 from L_3={*abc, abd, acd, ace, bcd*}

- Step 1: Self-joining: L_3*L_3
  - *abcd* from *abc* and *abd; acde* from *acd* and *ace*
- Step 2: Pruning:
  - *acde* is removed because *ade* is not in L_3

C_4={*abcd*}

Improve efficiency of Apriori

Supermarket transaction data may contain billions of transactions (i.e. $n \sim 10^9$) and million features. Consider:

- Product categories (e.g. milk), $p \sim 10^4$
- Brands (Horizon milk), $p \sim 10^5$
- UPCs (Horizon organic milk, 16oz), $p \sim 10^6$

Improve efficiency of Apriori

Bottlenecks:

- Multiple scans of transaction database
- Huge number of candidates
- Tedious workload of support counting for candidates

Improving Apriori: general ideas

- Shrink number of candidates
- Reduce passes of transaction database scans
- Reduce number of transactions
- Facilitate support counting of candidates

Partitioning: Reduce Number of Scans

Any itemset that is potentially frequent in database must be frequent (relative support) in at least one of the partitions of database

- Scan 1: partition database in n partitions and find local frequent patterns (minimum support count?)
- Scan 2: determine global frequent patterns from the collection of all local frequent patterns

Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within samples using Apriori
- Scan database once to verify frequent itemsets found in sample
- Use a lower support threshold than minimum support
- Tradeoff accuracy against efficiency

Textbook chapters

- ESL: chapter 14: 14.1 - 14.2