Mining Frequent Itemsets

CPS 296.1
Topics in Database Systems

Data mining

- Data → knowledge
- DBMS meets AI and statistics
- Clustering, prediction (classification and regression), association analysis, outlier analysis, evolution analysis, etc.
  - ➤Usually complex statistical "queries" that are difficult to answer → often specialized algorithms outside DBMS

➤We will focus on papers related to association rule/frequent itemset mining

Mining frequent itemsets

- Given: a large database of transactions, each containing a set of items
  - Example: market baskets
- Find all frequent itemsets
  - A set of items *X* is frequent if no less than s_{min}% of all transactions contain *X*
  - Examples: {diaper, beer}, {scanner, color printer}

TID	items
T001	diaper, milk, candy
T002	milk, egg
T003	milk, beer
T004	diaper, milk, egg
T005	diaper, beer
T006	milk, beer
T007	diaper, beer
T008	diaper, milk, beer, candy
T009	diaper, milk, beer
...	...

A naïve algorithm

- First try
  - Keep a running count for each possible itemset
  - For each transaction *T*, and for each itemset *X*, if *T* contains *X* then increment the count for *X*
  - Return itemsets with large enough counts
- Problem: The number of itemsets is huge!
  - 2^n, where *n* is the number of items
- Think: How do we prune the search space?

The Apriori property

- All subsets of a frequent itemset must also be frequent
  - Because any transaction that contains *X* must also contains subsets of *X*

➤If we have already verified that *X* is infrequent, there is no need to count *X*'s supersets because they must be infrequent too

The Apriori algorithm

➤ Agrawal and Srikant. "Fast Algorithms for Mining Association Rules." *VLDB* 1994

- Multiple passes over the transactions
- Pass *k* finds all frequent *k*-itemsets (itemset of size *k*)
- Use the set of frequent (k – 1)-itemsets found in the previous pass to narrow the search for *k*-itemsets

Pseudo-code for Apriori

Scan the transactions to find L_1, the set of all frequent 1-itemsets, together with their counts;

for ($k = 2$; $L_{k-1} \neq \varnothing$; k++) {

Generate C_k, the set of candidate k-itemsets, from L_{k-1}, the set of frequent $(k-1)$-itemsets found in the previous step;

Scan the transactions to count the occurrences of itemsets in C_k;

Find L_k, a subset of C_k containing k-itemsets with counts no less than ($s_{\min}$% · total # of transactions); }

Return $L_1 \cup L_2 \cup \ldots \cup L_k$;

Candidate generation

From L_{k-1} to C_k

- Join: combine frequent $(k-1)$-itemsets to form candidate k-itemsets
- Prune: ensure every size-$(k-1)$ subset of a candidate is frequent

Example: pass 1

TID	*Items*
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions

$s_{\min}$% = 20%

L_1

itemset	*count*
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

Itemset { F } is infrequent

Example: pass 2

TID	*Items*
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions

$s_{\min}$% = 20%

Generate candidates → Scan and count → Check min. support

L_1

itemset	*count*
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

C_2

itemset
{A, B}
{A, C}
{A, D}
{A, E}
{B, C}
{B, D}
{B, E}
{C, D}
{C, E}
{D, E}

C_2

itemset	*count*
{A, B}	4
{A, C}	4
{A, D}	1
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2
{C, D}	0
{C, E}	1
{D, E}	0

L_2

itemset	*count*
{A, B}	4
{A, C}	4
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2

Example: pass 3

TID	*Items*
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions

$s_{\min}$% = 20%

Generate candidates → Scan and count → Check min. support

L_2

itemset	*count*
{A, B}	4
{A, C}	4
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2

C_3

itemset
{A, B, C}
{A, B, E}
~~{A, C, E}~~
~~{B, C, D}~~
~~{B, C, E}~~
~~{B, D, E}~~

C_3

itemset	*count*
{A, B, C}	2
{A, B, E}	2

L_3

itemset	*count*
{A, B, C}	2
{A, B, E}	2

Example: pass 4

TID	*Items*
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions

$s_{\min}$% = 20%

Generate candidates

L_3

itemset	*count*
{A, B, C}	2
{A, B, E}	2

C_4

itemset
~~{A, B, C, E}~~

C_4 and L_4 are empty

Example: final answer

L_1

itemset	count
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

L_2

itemset	count
{A, B}	4
{A, C}	4
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2

L_3

itemset	count
{A, B, C}	2
{A, B, E}	2

Other tricks and extensions

- Transaction reduction
  - If a transaction does not contain any frequent *k*-itemset, remove it from further consideration
    - » AprioriTid, AprioriHybrid, from the same paper
- Dynamic itemset counting
  - Why only introduce candidate itemsets at the end of a scan? Start counting them whenever there is enough support from smaller itemsets
  - Fewer passes over data
    - » Brin et al., *SIGMOD* 1997
- Parallelization, sampling, incremental mining, etc.