Introduction to Data Mining

Lecture #12: Frequent Itemsets

U Kang
Seoul National University

Association Rule Discovery

Supermarket shelf management – Market-basket model:

- **Goal:** Identify items that are bought together by sufficiently many customers
- **Approach:** Process the sales data collected with barcode scanners to find dependencies among items
- **A classic rule:**  

  - If someone buys diaper and milk, then he/she is likely to buy beer 
  - Don't be surprised if you find six-packs next to diapers!

The Market-Basket Model

- A large set of **items**
  - e.g., things sold in a supermarket
- A **large set** of **baskets**
- Each basket is a **small subset of items**
  - e.g., the things one customer buys on one day
- Want to discover **association rules**
  - People who bought {x,y,z} tend to buy {v,w}
    - Amazon!

Input:

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

Output:

Rules Discovered:
{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

Applications – (1)

- **Items** = products; **Baskets** = sets of products someone bought in one trip to the store
- **Real market baskets:** Chain stores keep TBs of data about what customers buy together
  - Tells how typical customers navigate stores, lets them position tempting items
  - Suggests tie-in "tricks", e.g., run sale on diapers and raise the price of beer
  - Need the rule to occur frequently, or no $$'s
- **Amazon's people who bought *X* also bought *Y***

Applications – (2)

- **Baskets** = sentences; **Items** = documents containing those sentences
  - How can we interpret items that appear together too often?
  - Items that appear together too often could represent plagiarism
  - Notice items do not have to be "in" baskets

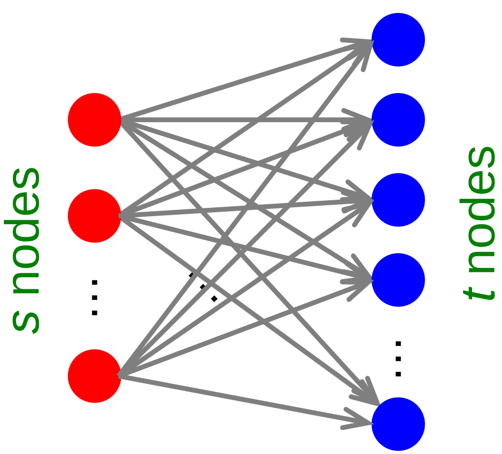
- **Baskets** = patients; **Items** = biomarkers(genes, proteins), diseases
  - How can we interpret frequent itemset (disease and biomarker)?
  - Frequent itemset consisting of one disease and one or biomarkers suggests a test for the disease

More generally

- **A general many-to-many mapping (association) between two kinds of things**
  - But we ask about connections among "items", not "baskets"
- **For example:**
  - Finding communities in graphs (e.g., Twitter)

Example:

- **Finding communities in graphs (e.g., Twitter)**
- **Baskets** = nodes; **Items** = outgoing neighbors
  - Searching for complete bipartite subgraphs $K_{s,t}$ of a big graph



A dense 2-layer graph

- **How?**
  - View each node ***i*** as a basket B_i of nodes ***i*** points to
  - $K_{s,t}$ = a node set ***Y*** of size ***t*** that occurs in *s* baskets B_i
  - Looking for $K_{s,t}$ → all frequent sets of size ***t*** that appears ***s*** times

ROADMAP

First: Define

- **Frequent itemsets**
- **Association rules:**
  - Confidence, Support, Interestingness

Then: Algorithms for finding frequent itemsets

- **Finding frequent pairs**
- **A-Priori algorithm**
- **PCY algorithm + 2 refinements**

Outline

- ☐ **Frequent Itemsets**
- ☐ Finding Frequent Itemsets

Frequent Itemsets

- **Simplest question:** Find sets of items that appear together "frequently" in baskets
- ***Support*** for itemset I: Number of baskets containing all items in I
  - (Often expressed as a fraction of the total number of baskets)
- Given a ***minimum support s***, then sets of items that appear in at least s baskets are called ***frequent itemsets***

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

Support of {Beer, Bread} = 2

Example: Frequent Itemsets

- **Items** = {milk, coke, pepsi, beer, juice}
- **Minimum support** = 3 baskets

  B_1 = {m, c, b}   **B_2** = {m, p, j}

  B_3 = {m, b}   **B_4** = {c, j}

  B_5 = {m, p, b}   **B_6** = {m, c, b, j}

  B_7 = {c, b, j}   **B_8** = {b, c}

- **Frequent itemsets:**
  - {m}, {c}, {b}, {j}, {m,b}, {b,c}, {c,j}

Association Rules

- **Association Rules:**
  If-then rules about the contents of baskets
- $\{i_1, i_2,\ldots,i_k\} \rightarrow j$ means: "if a basket contains all of $i_1,\ldots,i_k$ then it is ***likely*** to contain j"
- **In practice there are many rules, want to find significant/interesting ones!**
- ***Confidence*** of this association rule is the probability of j given $I = \{i_1,\ldots,i_k\}$

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

Interesting Association Rules

- **Not all high-confidence rules are interesting**
  - The rule $X \rightarrow \boldsymbol{milk}$ may have high confidence for many itemsets X, because milk is just purchased very often (independent of X) and the confidence will be high
- **Interest** of an association rule $I \rightarrow j$: difference between its confidence and the fraction of baskets that contain j

$$\text{Interest}(I \rightarrow j) = \text{conf}(I \rightarrow j) - \Pr[j]$$

  - Interesting rules are those with high positive or negative interest values (usually above 0.5)

Interesting Association Rules

- **Interest** of an association rule $I \rightarrow j$: difference between its confidence and the fraction of baskets that contain j

$$\text{Interest}(I \rightarrow j) = \text{conf}(I \rightarrow j) - \Pr[j]$$

  - Interesting rules are those with high positive or negative interest values (usually above 0.5)
  - E.g.
    - conf[I -> j] is large, but Pr[j] is small ?
    - conf[I -> j] is small, but Pr[j] is large ?

Example: Confidence and Interest

B_1 = {m, c, b} **B_2 = {m, p, j}**
B_3 = {m, b} **B_4= {c, j}**
B_5 = {m, p, b} **B_6 = {m, c, b, j}**
B_7 = {c, b, j} **B_8 = {b, c}**

- **Association rule: {m, b} → c**
  - **Confidence** = 2/4 = 0.5
  - **|Interest|** = |0.5 – 5/8| = 1/8
    - Item ***c*** appears in 5/8 of the baskets
    - Rule is not very interesting!

Finding Association Rules

- **Problem: Find all association rules with support $\geq s$ and confidence $\geq c$**
  - **Note:** Support of an association rule is the support of the set of items on the left side
- **Hard part: Finding the frequent itemsets!**
  - If $\{i_1, i_2, \ldots, i_k\} \rightarrow j$ has high support and confidence, then both $\{i_1, i_2, \ldots, i_k\}$ and $\{i_1, i_2, \ldots, i_k, j\}$ will be "frequent"

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

Mining Association Rules

- **Step 1:** Find all frequent itemsets I
  - (we will explain this next)
- **Step 2: Rule generation**
  - For every subset A of I, generate a rule $A \rightarrow I \setminus A$
    - Since I is frequent, A is also frequent
    - **Variant 1:** Single pass to compute the rule confidence
      - confidence(***A,B→C,D***) = support(**A,B,C,D**) / support(**A,B**)
    - **Variant 2:**
      - **Observation:** If **A,B,C→D** is below confidence, so is **A,B→C,D**
      - Can generate "bigger" rules from smaller ones!
  - **Output the rules with confidence $\geq c$**

Example

B_1 = {m, c, b} B_2 = {m, p, j}
B_3 = {m, c, b, n} B_4= {c, j}
B_5 = {m, p, b} B_6 = {m, c, b, j}
B_7 = {c, b, j} B_8 = {b, c}

- **Minimum support** $s = 3$, **confidence** $c = 0.75$
- **1) Frequent itemsets:**
  - {b,m} {b,c} {c,m} {c,j} {m,c,b}
- **2) Generate rules:**
  - ~~**b**→**m**: c=4/6~~ **b**→**c**: c=5/6 ~~**b,c**→**m**: c=3/5~~
  - **m**→**b**: c=4/5 … **b,m**→**c**: c=3/4
  - ~~**b**→**c,m**: c=3/6~~

Compacting the Output

- **To reduce the number of rules we can post-process them and only output:**
  - **Maximal frequent itemsets:**
    No superset is frequent
    - E.g., if {A}, {A, B}, {B,C} are frequent, {A,B}, {B,C} are maximal
    - Gives more pruning

  or

  - **Closed itemsets:**
    No immediate superset has the same count (> 0)
    - E.g., if {A}, {A,C} both have support 5, {A} is not closed
    - Stores not only frequent information, but exact counts

Example: Maximal/Closed

	Support	Maximal(s=3)	Closed
A	4	No	No
B	5	No	Yes
C	3	No	No
AB	4	Yes	Yes
AC	2	No	No
BC	3	Yes	Yes
ABC	2	No	Yes

Frequent, but superset BC also frequent.

Frequent, and its only superset, ABC, not freq.

Superset BC has same count.

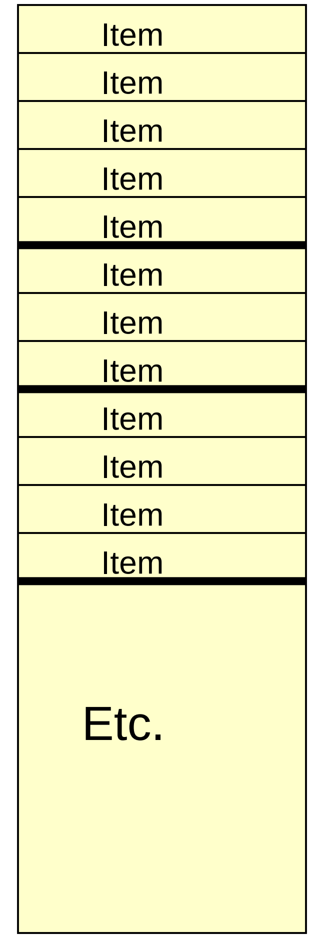
Its only super-set, ABC, has smaller count.

Outline

- ☑ Frequent Itemsets
- ☐ **Finding Frequent Itemsets**

Itemsets: Computation Model

- **Back to finding frequent itemsets**
- Typically, data are kept in flat files rather than in a database system:
  - Stored on disk
  - Stored basket-by-basket
  - Baskets are **small** but we have many baskets and many items
    - Expand baskets into pairs, triples, etc. as you read baskets
    - Use ***k*** nested loops to generate all sets of size ***k***



Items are positive integers, and boundaries between baskets are –1.

Note: We want to find frequent itemsets. To find them, we have to count them. To count them, we have to generate them.

Computation Model

- The true cost of mining disk-resident data is usually the **number of disk I/Os**
- In practice, association-rule algorithms read the data in ***passes*** – all baskets read in turn
- We measure the cost by the **number of passes** an algorithm makes over the data
  - 1-pass, 2-pass, …

Main-Memory Bottleneck

- For many frequent-itemset algorithms, **main-memory** is the critical resource
  - As we read baskets, we need to count something, e.g., occurrences of pairs of items
  - The number of different things we can count is limited by main memory
  - Swapping counts in/out from/to disk is a disaster (why?)

Finding Frequent Pairs

- **The hardest problem often turns out to be finding the frequent pairs of items $\{i_1, i_2\}$**
  - **Why?** Freq. pairs are common, freq. triples are rare

- **Let's first concentrate on pairs, then extend to larger sets**
- **The approach:**
  - We always need to generate all the itemsets
  - But we would only like to count (keep track of) those itemsets that in the end turn out to be frequent

Naïve Algorithm

- **Naïve approach to finding frequent pairs**
- Read file once, counting in main memory the occurrences of each pair:
  - From each basket of ***n*** items, generate its ***n(n-1)/2*** pairs by two nested loops
- **Fails if $(\#items)^2$ exceeds main memory**
  - **Remember:** #items can be 100K (Wal-Mart) or 10B (Web pages)
    - Suppose 10^5 items, counts are 4-byte integers
    - Number of pairs of items: $10^5(10^5-1)/2 \approx 5*10^9$
    - Therefore, $2*10^{10}$ (20 gigabytes) of memory needed

Counting Pairs in Memory

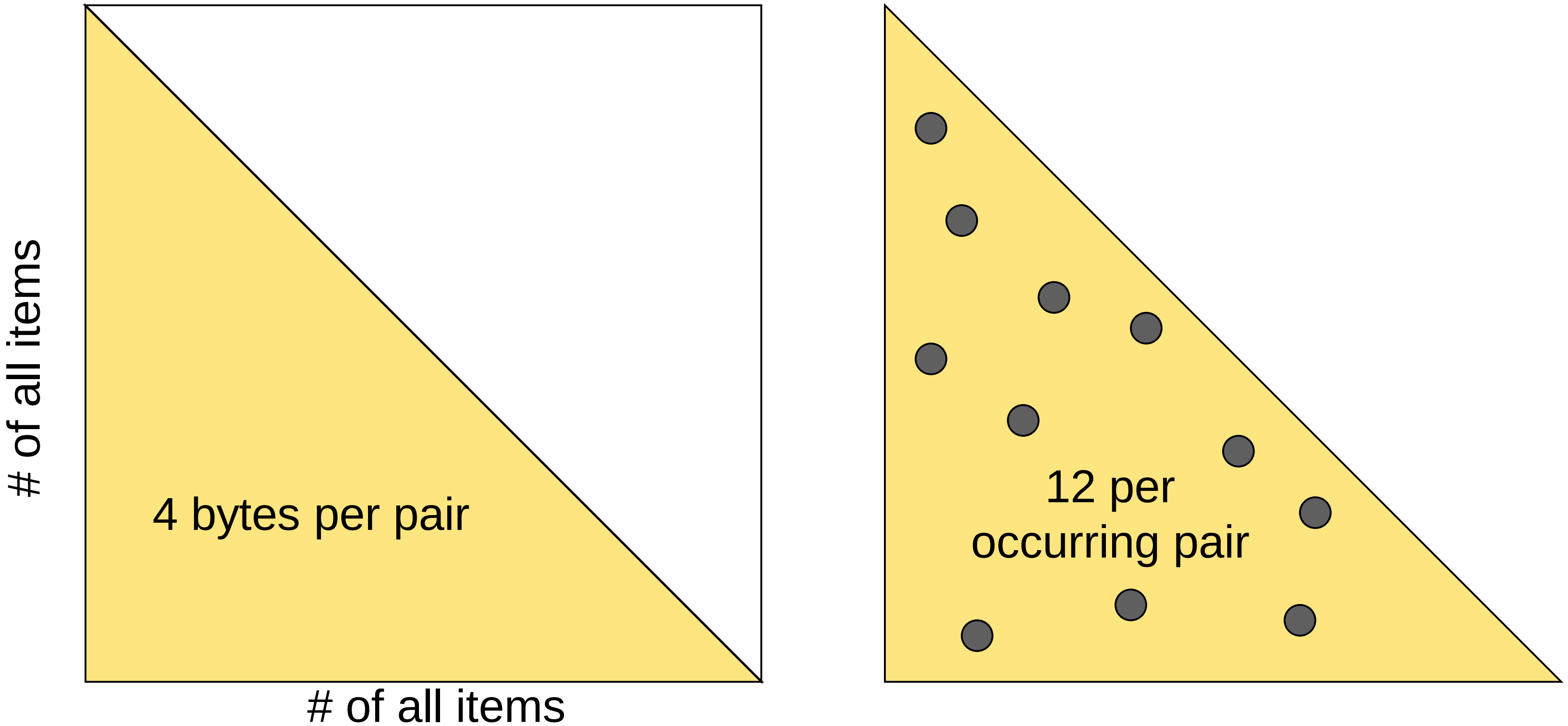
Two approaches:

- **Approach 1:** Count all pairs using a matrix
- **Approach 2:** Keep a table of triples [*i*, *j*, *c*] = "the count of the pair of items {*i*, *j*} is *c*." (where c>0)
  - If integers and item ids are 4 bytes, we need approximately 12 bytes for pairs with count > 0
  - Plus some additional overhead for the hashtable

Note:

- **Approach 1** only requires 4 bytes per pair
- **Approach 2** uses 12 bytes per pair (but only for pairs with count > 0)

Comparing the 2 Approaches



Triangular Matrix
(Approach 1)
Store all (i,j) where i<j

Triples
(Approach 2)
Store (i,j) whose sup>=1

Comparing the two approaches

- **Approach 1: Triangular Matrix**
  - **n** = total number items
  - Count pair of items {*i*, *j*} only if *i*<*j*
  - Can use one-dimensional array to store the tri. matrix
    - Keep pair counts in lexicographic order:
      - {1,2}, {1,3},…, {1,*n*}, {2,3}, {2,4},…,{2,*n*}, {3,4},…
    - Pair {*i*, *j*} is at position $(i-1)(n-i/2)+j-i$ *(array index starts from 1)*
      - Proof: $\left(\sum_{k=1}^{i-1}(n-k)\right)+(j-i)$
  - Total number of pairs $\boldsymbol{n(n-1)/2}$; total bytes ~ $\boldsymbol{2n^2}$
  - **Triangular Matrix** requires 4 bytes per pair
- **Approach 2** uses **12 *bytes*** per occurring pair *(but only for pairs with count > 0)*
  - When should we prefer Approach 2 over Approach 1?
    - If less than **1/3** of possible pairs actually occur

Comparing the two approaches

- **Approach 1: Triangular Matrix**
  - **n** = to
  - Co
  - Ca … rix
    -
    - … *starts from 1)*
  - To
  - **Tri**
- **Appr**
  (but only for pairs with count > 0)
  - When should we prefer Approach 2 over Approach 1?
    - If less than **1/3** of possible pairs actually occur

Problem is if we have too many items so the pairs do not fit into memory.

Can we do better?

Questions?