About midterm

• Exam will be closed book, closed notes
• The questions are designed such that you will not need a calculator to get the answers
• Extra office hour tomorrow 10 – 12
Frequent pattern mining: association rules

CS434
What Is Frequent Pattern Mining?

- **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 DNA are sensitive to this new drug?

- **Broad applications**
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis
  - Web log (click stream) analysis
  - DNA sequence analysis
Association rules

Data: Market-Basket transactions

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs, Coke},
{Beer, Bread} → {Milk},

Implication means **co-occurrence, not causality!**

Given a set of transactions, find rules that will **predict the occurrence of an item based on the occurrences of other items** in the transaction
Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: \{Milk, Bread, Diaper\}
  - k-itemset
    - An itemset that contains k items

- **Support count (\(\sigma\))**
  - Frequency of occurrence of an itemset
  - E.g. \(\sigma(\{\text{Milk, Bread,Diaper}\}) = 2\)

- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. \(s(\{\text{Milk, Bread, Diaper}\}) = 2/5\)

- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsуп* threshold

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>
Definition: Association Rule

- **Association Rule**
  - An implication expression of the form \( X \rightarrow Y \), where \( X \) and \( Y \) are itemsets
  - Example:
    \{Milk, Diaper\} \rightarrow \{Beer\}

- **Rule Evaluation Metrics**
  - Support (\( s \))
    - Fraction of transactions that contain both \( X \) and \( Y \): \( P(X \cap Y) \)
  - Confidence (\( c \))
    - Measures how often items in \( Y \) appear in transactions that contain \( X \): \( P(Y | X) \)

Example:
\{Milk, Diaper\} \Rightarrow Beer

\[
\begin{align*}
  s &= \frac{\sigma(Milk, Diaper, Beer)}{|T|} = \frac{2}{5} = 0.4 \\
  c &= \frac{\sigma(Milk, Diaper, Beer)}{\sigma(Milk, Diaper)} = \frac{2}{3} = 0.67
\end{align*}
\]
Problem definition: Association Rules Mining

- **Inputs:**
  - Itemset \( X = \{ x_1, \ldots, x_k \} \),
  - thresholds: \( \text{min}_\text{sup}, \text{min}_\text{conf} \)

- **Output:**
  - All the rules \( X \rightarrow Y \) having:
    - support \( P(X \cap Y) \geq \text{min}_\text{sup} \)
    - confidence \( P(Y | X) \geq \text{min}_\text{conf} \)

Let \( \text{min}_\text{sup} = 50\%, \text{min}_\text{conf} = 50\% \):

- \( A \rightarrow C \) (50%, 66.7%)
- \( C \rightarrow A \) (50%, 100%)

<table>
<thead>
<tr>
<th>Transaction-id</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, B, C</td>
</tr>
<tr>
<td>20</td>
<td>A, C</td>
</tr>
<tr>
<td>30</td>
<td>A, D</td>
</tr>
<tr>
<td>40</td>
<td>B, E, F</td>
</tr>
</tbody>
</table>
Brute-force solution

• List all possible association rules
• Compute the support and confidence for each rule
• Prune rules that fail the $min_{sup}$ and $min_{conf}$ thresholds

⇒ Computationally prohibitive!
Mining Association Rules

Example of Rules:

- \{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\} (s=0.4, c=0.67)
- \{\text{Milk, Beer}\} \rightarrow \{\text{Diaper}\} (s=0.4, c=1.0)
- \{\text{Diaper, Beer}\} \rightarrow \{\text{Milk}\} (s=0.4, c=0.67)
- \{\text{Milk, Beer}\} \rightarrow \{\text{Diaper}\} (s=0.4, c=0.67)
- \{\text{Milk}\} \rightarrow \{\text{Diaper, Beer}\} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  \{\text{Milk, Diaper, Beer}\}

- Rules originating from the same itemset have identical support but can have different confidence

- Thus, we may decouple the support and confidence requirements

- We can first find all frequent itemsets that satisfy the support requirement
Mining Association Rules

• Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq \minsup$

  2. Rule Generation
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

• Frequent itemset generation is still computationally expensive
Frequent Itemset Generation

Given \( d \) items, there are \( 2^d \) possible candidate itemsets
Frequent Itemset Generation

• Brute-force approach:
  – Each itemset in the lattice is a candidate frequent itemset
  – Count the support of each candidate by scanning the database
  – Match each transaction against every candidate
  – Complexity $\sim O(NMw) \Rightarrow \text{Expensive since } M = 2^d !!!$

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>
Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
  - If \{beer, diaper, nuts\} is frequent, so is \{beer, diaper\}
  - i.e., every transaction having \{beer, diaper, nuts\} also contains \{beer, diaper\}

- Apriori principle holds due to the following property of the support measure:

\[ \forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y) \]

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support
Illustrating Apriori Principle

Found to be Infrequent

Pruned supersets
Illustrating Apriori Principle

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>4</td>
</tr>
<tr>
<td>Coke</td>
<td>2</td>
</tr>
<tr>
<td>Milk</td>
<td>4</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Diaper</td>
<td>4</td>
</tr>
<tr>
<td>Eggs</td>
<td>1</td>
</tr>
</tbody>
</table>

Items (1-itemsets)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread,Milk}</td>
<td>3</td>
</tr>
<tr>
<td>{Bread,Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Bread,Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Milk,Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Milk,Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Beer,Diaper}</td>
<td>3</td>
</tr>
</tbody>
</table>

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Min Support count = 3

If every subset is considered, $C_1^6 + C_2^6 + C_3^6 = 41$

With support-based pruning, $6 + 6 + 1 = 13$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread,Milk,Diaper}</td>
<td>3</td>
</tr>
</tbody>
</table>

Triplets (3-itemsets)
The Apriori Algorithm

• Method:
  
  – Let k=1
  – Generate frequent itemsets of length 1
  – Repeat until no new frequent itemsets are identified
    • Generate length (k+1) candidate itemsets from length k frequent itemsets
    • Prune candidate itemsets containing subsets of length k that are infrequent
    • Count the support of each candidate by scanning the DB
    • Eliminate candidates that are infrequent, leaving only those that are frequent
The Apriori Algorithm

- **Pseudo-code:**
  
  $C_k$: Candidate itemset of size $k$
  
  $L_k$: frequent itemset of size $k$

  \[ L_1 = \{ \text{frequent items} \}; \]

  \[ \text{for} \ (k = 1; \ L_k \neq \emptyset; \ k++) \ \text{do begin} \]
  
  \[ C_{k+1} = \text{candidates generated from } L_k; \]

  \[ \text{for each transaction } t \text{ in database do} \]

  \[ \text{increment the count of all candidates in } C_{k+1} \]
  
  \[ \text{that are contained in } t \]

  \[ L_{k+1} = \text{candidates in } C_{k+1} \text{ with min}\_\text{support} \]

  \[ \text{end} \]

  \[ \text{return } \bigcup_k L_k; \]
How to Generate Candidates?

• Suppose the items in $L_k$ are listed in an order (e.g., alphabetic ordering)

• Step 1: self-joining $L_k$
  
  For all itemsets $p$ and $q$ in $L_k$ such that
  
  $$p.item_i = q.item_i \text{ for } i = 1, 2, ..., k-1 \text{ and } p.item_k < q.item_k$$
  
  Add to $C_{k+1}$
  
  $$p.item_1, p.item_2, ..., p.item_k, q.item_k$$

• Step 2: pruning
  
  For all itemsets $c$ in $C_{k+1}$ do
  
  For all (k)-subsets $s$ of $c$ do
  
  if ($s$ is not in $L_k$) then delete $c$ from $C_{k+1}$
Important Details of Apriori

Self-joining rule:
1. we join two itemsets if and only if they only differ by their last item
2. When joining, the items are always ranked based on a fixed ordering of the items (e.g., alphabetic ordering)

• Example of Candidate-generation
  – \( L_3 = \{abc, abd, acd, ace, bcd\} \)
  – Self-joining: \( L_3 \times L_3 \)
    • \( abcd \) from \( abc \) and \( abd \)
    • \( acde \) from \( acd \) and \( ace \)
  – Pruning:
    • \( acde \) is removed because \( ade \) is not in \( L_3 \)
  – \( C_4 = \{abcd\} \)

Why not abd, and acd -> abcd?
Why should this work?

• How can we be sure we are not missing any possible itemset?

• This can be seen by proving that for every possible frequent k+1-itemset, it will be included using this self-joining process

Proof
For any k +1 item set S (with items ranked), it will be included by joining the following two subsets:
1. \( S_k = \{ \text{the first } k \text{ items of } S \} \)
2. \( S'_k = S \text{ with the } k\text{-th item removed} \)

Clearly \( S_k \) and \( S'_k \) are frequent, and differ by only the last item. So they must satisfy the self-join condition and \( S_k \cap S'_k = S \)
The Apriori Algorithm—An Example

\( \text{Sup}_{\min} = \frac{2}{4} \)

Database TDB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
</tr>
</tbody>
</table>

\( C_1 \) 1st scan

\( L_1 \)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{D}</td>
<td>1</td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

\( C_2 \) 2nd scan

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, B}</td>
<td>1</td>
</tr>
<tr>
<td>{A, C}</td>
<td>2</td>
</tr>
<tr>
<td>{A, E}</td>
<td>1</td>
</tr>
<tr>
<td>{B, C}</td>
<td>2</td>
</tr>
<tr>
<td>{B, E}</td>
<td>3</td>
</tr>
<tr>
<td>{C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

\( C_3 \) 3rd scan

<table>
<thead>
<tr>
<th>Itemset</th>
<th>{A, B, C}?</th>
</tr>
</thead>
</table>

\( L_3 \)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>
Mining Association Rules

• Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
     – Enumerate all possible rules from the frequent itemset and out these of high confidence
Example: Generating rules

- **Min_conf = 80%**

Database TDB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

L2

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, C}</td>
<td>2</td>
</tr>
<tr>
<td>{B, C}</td>
<td>2</td>
</tr>
<tr>
<td>{B, E}</td>
<td>3</td>
</tr>
<tr>
<td>{C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

L3

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

Rules:

- \( A \rightarrow C: 100\% \)
- \( C \rightarrow A: 66.7\% \)
- \( B \rightarrow C: 66.7\% \)
- \( C \rightarrow B: 66.7\% \)
- \( B \rightarrow E: 100\% \)
- \( E \rightarrow B: 100\% \)
- \( C \rightarrow E: 66.7\% \)
- \( E \rightarrow C: 66.7\% \)
- \( B, C \rightarrow E: 100\% \)
- \( B, E \rightarrow C: 66.7\% \)
- \( C, E \rightarrow B: 100\% \)
Frequent-Pattern Mining: Summary

• Frequent pattern mining—an important task in data mining
• “Scalable” frequent pattern mining methods
  – Apriori (Candidate generation & test)
    ▪ The Apriori property has also been used in mining other type of patterns such as sequential and structured patterns
    ▪ Problem: frequent patterns are not necessarily interesting patterns
      ▪ Bread -> milk is not really interesting although it has high support and confidence
      ▪ Many other measures of interestingness exist to address this problem
        ▪ Such as “unexpectedness”
Comparing Association rule with Supervised learning

• Supervised learning
  – Have predefined class variable
  – Focus on difference one class from another

• Association rule mining
  – Do not have predefined target class variable
  – Right hand side of the rule can have many items
  – We could place the class variable C on the right hand side of a rule, but it does not focus on differentiating classes, but more on characterizing a class
What you need to know

• What is an association rule?
• What are the support and confidence of a rule?
• The apriori property
• How to find frequent itemset using the apriori property
  – The Candidate Generation: self-join, and prune
  – Why is it correct?
• How to produce association rules based on frequent itemsets?