Data Mining

• Broadly speaking, data mining is the process of semi-
  automatically analyzing large databases to find useful patterns
• Like knowledge discovery in artificial intelligence data mining
discovers statistical rules and patterns
• Differs from machine learning in that it deals with large volumes
  of data stored primarily on disk.
• Some types of knowledge discovered from a database can be
  represented by a set of rules.
  • e.g.: “Young women with annual incomes greater than $50,000 are
    most likely to buy sports cars”
• Other types of knowledge represented by equations, or by
  prediction functions
• Some manual intervention is usually required
  • Pre-processing of data, choice of which type of pattern to find,
    postprocessing to find novel patterns
Applications of Data Mining

• **Prediction** based on past history
  • Predict if a credit card applicant poses a good credit risk, based on some attributes (income, job type, age, ..) and past history
  • Predict if a customer is likely to switch brand loyalty
  • Predict if a customer is likely to respond to “junk mail”
  • Predict if a pattern of phone calling card usage is likely to be fraudulent
• Some examples of prediction mechanisms:
  • **Classification**
    • Given a training set consisting of items belonging to different classes, and a new item whose class is unknown, predict which class it belongs to
  • **Regression formulae**
    • Given a set of parameter-value to function-result mappings for an unknown function, predict the function-result for a new parameter-value

Applications of Data Mining (Cont.)

• **Descriptive Patterns**
  • **Associations**
    • Find books that are often bought by the same customers. If a new customer buys one such book, suggest that he buys the others too.
    • Other similar applications: camera accessories, clothes, etc.
    • Associations may also be used as a first step in detecting causation
      • E.g. association between exposure to chemical X and cancer, or new medicine and cardiac problems
  • **Clusters**
    • E.g. typhoid cases were clustered in an area surrounding a contaminated well
    • Detection of clusters remains important in detecting epidemics
Definition: Rules that state a statistical correlation between the occurrence of certain attributes in a database table.

Given a set of transactions, where each transaction is a set of items, \(X_1, \ldots, X_n\) and \(Y\), an association rule is an expression \(X_1, \ldots, X_n \Rightarrow Y\).
This means that the attributes \(X_1, \ldots, X_n\) predict \(Y\).

Intuitive meaning of such a rule: transactions in the database which contain the items in \(X\) tend also to contain the items in \(Y\).

Support:
- Given the association rule \(X_1, \ldots, X_n \Rightarrow Y\), the support is the percentage of records for which \(X_1, \ldots, X_n\) and \(Y\) both hold.
- The statistical significance of the association rule.

Confidence:
- Given the association rule \(X_1, \ldots, X_n \Rightarrow Y\), the confidence is the percentage of records for which \(Y\) holds, within the group of records for which \(X_1, \ldots, X_n\) hold.
- The degree of correlation in the dataset between \(X\) and \(Y\).
- A measure of the rule’s strength.
Problem:
Given a transaction table $D$, find the support and confidence for an association rule $B,D \Rightarrow E$.

Database $D$

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>A B E</td>
</tr>
<tr>
<td>02</td>
<td>A C D E</td>
</tr>
<tr>
<td>03</td>
<td>B C D E</td>
</tr>
<tr>
<td>04</td>
<td>A B D E</td>
</tr>
<tr>
<td>05</td>
<td>B D E</td>
</tr>
<tr>
<td>06</td>
<td>A B C</td>
</tr>
<tr>
<td>07</td>
<td>A B D</td>
</tr>
</tbody>
</table>

Answer:
support = 3/7, confidence = 3/4

Apriori Algorithm

An efficient algorithm to find association rules.

• Procedure
  1. Find all the frequent itemsets:
     A frequent itemset is a set of items that have support greater than a user defined minimum.
  2. Use the frequent itemsets to generate the association rules
**Notation**

<table>
<thead>
<tr>
<th>$D$</th>
<th>The sample transaction database</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>The set of all frequent items.</td>
</tr>
</tbody>
</table>

$k$-itemset | An itemset having $k$ items.

$L_k$ | Set of candidate $k$-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count.

$C_k$ | Set of candidate $k$-itemsets (potentially frequent itemsets). Each member of this set has two fields: i) itemset and ii) support count.

---

**Example**

Database $D$

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>A, C, D</td>
</tr>
<tr>
<td>200</td>
<td>B, C, E</td>
</tr>
<tr>
<td>300</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>400</td>
<td>B, E</td>
</tr>
</tbody>
</table>

*(k = 1) itemset*

<table>
<thead>
<tr>
<th>C1</th>
<th>Support</th>
<th>L1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>.50</td>
<td>Y</td>
</tr>
<tr>
<td>(B)</td>
<td>.75</td>
<td>Y</td>
</tr>
<tr>
<td>(C)</td>
<td>.75</td>
<td>Y</td>
</tr>
<tr>
<td>(D)</td>
<td>.25</td>
<td>N</td>
</tr>
<tr>
<td>(E)</td>
<td>.75</td>
<td>Y</td>
</tr>
</tbody>
</table>

*(k = 2) itemset*

<table>
<thead>
<tr>
<th>C2</th>
<th>Support</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A, B)</td>
<td>.25</td>
<td>N</td>
</tr>
<tr>
<td>(A, C)</td>
<td>.75</td>
<td>Y</td>
</tr>
<tr>
<td>(A, E)</td>
<td>.25</td>
<td>N</td>
</tr>
<tr>
<td>(B, C)</td>
<td>.50</td>
<td>Y</td>
</tr>
<tr>
<td>(B, E)</td>
<td>.75</td>
<td>Y</td>
</tr>
</tbody>
</table>

*(k = 3) itemset*

<table>
<thead>
<tr>
<th>C3</th>
<th>Support</th>
<th>L3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(B, C, E)</td>
<td>.50</td>
<td>Y</td>
</tr>
</tbody>
</table>

*(k = 4) itemset*

<table>
<thead>
<tr>
<th>C4</th>
<th>Support</th>
<th>L4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A, B, C, E)</td>
<td>.25</td>
<td>N</td>
</tr>
</tbody>
</table>

* Suppose a user defined minimum = .49

* n items implies $O(n^2 - 2)$ computational complexity?
Procedure

Apriorialgo()
{
F = ∅;
Lk = {frequent 1-itemsets};
k = 2; /* k represents the pass number. */
while (Lk-1 != ∅)
{
F = F ∪ Lk;
Ck = New candidates of size k generated from Lk-1;
for all transactions t ∈ D
increment the count of all candidates in Ck that are contained in t ;
Lk = All candidates in Ck with minimum support ;
k++;
}
return ( F ) ;
}

Candidate Generation

Given Lk-1, the set of all frequent (k-1)-itemsets,
generate a superset of the set of all frequent k-itemsets.

Idea : if an itemset X has minimum support, so do all subsets of X.

1. Join Lk-1 with Lk-1
2. Prune: delete all itemsets c ∈ Ck such that some (k-1)-subset of c is not in Lk-1.

ex) L2 = { {A,C}, {B,C}, {B,E}, {C,E} }

2. Prune : { {A,B,C}, {A,C,E}, {B,C,E} }

{ A, B } ∉ L2  { A, E } ∉ L2

[Instead of 5C1 = 10, we have only 1 candidate.]
Association rules are always defined on binary attributes. 
⇒ Need to flatten the tables.

ex) Phone Company DB.

<table>
<thead>
<tr>
<th>CID</th>
<th>Gender</th>
<th>Ethnicity</th>
<th>Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>CID</td>
<td>M</td>
<td>W</td>
<td>A</td>
</tr>
<tr>
<td>CID</td>
<td>F</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>CID</td>
<td>W</td>
<td>H</td>
<td>I</td>
</tr>
</tbody>
</table>

- Support for Asian ethnicity will never exceed .5.
- No need to consider itemsets {M,F}, {W,B} nor {D,I}.
- M ⇒ F or D ⇒ I are not of interest at all.

* Considering the original schema before flattening may be a good idea.

When item constraints are considered, the Apriori candidate generation procedure does not generate all the potential frequent itemsets as candidates.

◆ Procedure
1. Find all the frequent itemsets that satisfy the boolean expression $B$.

2. Find the support of all subsets of frequent itemsets that do not satisfy $B$.

3. Generate the association rules from the frequent itemsets found in Step 1. by computing confidences from the frequent itemsets found in Steps 1 & 2.
**Additional Notation**

| B       | Boolean expression with m disjuncts:  
|         | $B = D_1 \lor D_2 \lor ... \lor D_m$  
| D_i     | N conjuncts in $D_i$  
|         | $D_i = a_{i,1} \land a_{i,2} \land ... \land a_{i,n}$  
| S       | Set of items such that any itemset that satisfies  
|         | B contains an item from S.  
| $L_s(k)$| Set of frequent $k$-itemsets that contain an item in S.  
| $L_b(k)$| Set of frequent $k$-itemsets that satisfy B.  
| $C_s(k)$| Set of candidate $k$-itemsets that contain an item in S.  
| $C_b(k)$| Set of candidate $k$-itemsets that satisfy B.  

**Direct Algorithm**

**Procedure**

1. Scan the data and determine $L_1$ and $F$.
2. Find $L_b(1)$
3. Generate $C_b(k+1)$ from $L_b(k)$
   3-1. $C_{k+1} = L_k \times F$
   3-2. Delete all candidates in $C_{k+1}$ that do not satisfy $B$.
   3-3. Delete all candidates in $C_{k+1}$ below the minimum support.
   3-4. For each $D_i$ with exactly $k + 1$ non-negated elements,  
        add the itemset to $C_{k+1}$ if all the items are frequent.
Given $B = (A \land B) \lor (C \land \neg E)$

$C_1 = \{ (A), (B), (C), (D), (E) \}$

$L_1 = \{ (A), (B), (C), (E) \}$ $\xrightarrow{\text{step 1 & 2}}$ $L_{b(1)} = \{ C \}$

$C_2 = L_{b(1)} \times F$

$= \{ (A,C), (B,C), (C,E) \}$ $\xrightarrow{\text{step 3-2}}$ $C_{b(2)} = \{ (A,C), (B,C) \}$

$L_2 = \{ (A,C), (B,C) \}$ $\xrightarrow{\text{step 3-4}}$ $L_{b(2)} = \{ (A,B), (A,C), (B,C) \}$

$C_3 = L_{b(2)} \times F = \{ (A,B,C), (A,B,E), (A,C,E), (B,C,E) \}$ $\xrightarrow{\text{step 3-2}}$ $C_{b(3)} = \{ (A,B,C), (A,B,E) \}$

$L_3 = \emptyset$ $\xrightarrow{\text{step 3-4}}$ $L_{b(3)} = \emptyset$

Multiple Joins and Reorder algorithms to find association rules with item constraints will be added.
Given a database \( D \) of customer transactions, the problem of mining sequential patterns is to find the maximal sequences among all sequences that have certain user-specified minimum support.

- Transaction-time field is added.
- Itemset in a sequence is denoted as 
  \(<s_1, s_2, \ldots, s_n>\)

### \( D \) and Sequential version of \( D' \)

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>Transaction Time</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jun 25 93</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>Jun 30 93</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>Jun 10 93</td>
<td>10,20</td>
</tr>
<tr>
<td>2</td>
<td>Jun 15 93</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>Jun 20 93</td>
<td>40,60,70</td>
</tr>
<tr>
<td>3</td>
<td>Jun 25 93</td>
<td>30,50,70</td>
</tr>
<tr>
<td>4</td>
<td>Jul 25 93</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>Jun 30 93</td>
<td>40,70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>Customer Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(&lt;30), (90)&gt;</td>
</tr>
<tr>
<td>2</td>
<td>(&lt;10, 20), (30), (40, 60, 70)&gt;</td>
</tr>
<tr>
<td>3</td>
<td>(&lt;30, 50, 70)&gt;</td>
</tr>
<tr>
<td>4</td>
<td>(&lt;30), (40, 70), (90)&gt;</td>
</tr>
<tr>
<td>5</td>
<td>(&lt;90)&gt;</td>
</tr>
</tbody>
</table>

* Customer sequence: all the transactions of a customer is a sequence ordered by increasing transaction time.

Answer set with support > .25 = \{ \(<30), (90)>, \(<30), (40, 70)> \}
Definitions

**Def 1.** A sequence \(<a_1, a_2, \ldots, a_n>\) is contained in another sequence \(<b_1, b_2, \ldots, b_m>\) if there exists integers \(i_1 < i_2 < \ldots < i_n\) such that \(a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}\)

ex) \(<(3), (4 5), (8)>\) is contained in \(<(7), (3 8), (9), (4 5 6)\>\). 
\(<(3), (5)>\) is contained in \(<(3 5)\>\).

**Def 2.** A sequence \(s\) is maximal if \(s\) is not contained in any other sequence.

- \(T_i\) is transaction time.
- itemset\((T_i)\) is transaction the set of items in \(T_i\).
- itemset : an item set with minimum support.

### Procedure

1. Convert \(D\) into a \(D'\) of customer sequences.

2. Itemset mapping

3. Transform each customer sequence into a itemset representation. \(<s_1, s_2, \ldots, s_n> \Rightarrow <l_1, l_2, \ldots, l_n>\>

4. Find the desired sequences using the set of itemsets.
   4-1. AprioriAll
   4-2. AprioriSome
   4-3. DynamicSome

5. Find the maximal sequences among the set of large sequences.
   for\((k = n; k > 1; k--)\)
   foreach k-sequence \(s_k\)
   delete from \(S\) all subsequences of \(s_k\).
Data Mining (Apriori Algorithm)  

### Example

<table>
<thead>
<tr>
<th>Large Itemsets</th>
<th>Mapped to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(30)</td>
<td>1</td>
</tr>
<tr>
<td>(40)</td>
<td>2</td>
</tr>
<tr>
<td>(70)</td>
<td>3</td>
</tr>
<tr>
<td>(40 70)</td>
<td>4</td>
</tr>
<tr>
<td>(90)</td>
<td>5</td>
</tr>
</tbody>
</table>

#### step 2

**Customer Sequence**

<table>
<thead>
<tr>
<th>CID</th>
<th>Customer Sequence</th>
<th>Transformed Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;(30),(90)&gt;</td>
<td>&lt;{(30)}{(90)}&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;(10 20),(30),(40 60 70)&gt;</td>
<td>&lt;{(30)}{(40),(70),(40 70)}}</td>
</tr>
<tr>
<td>3</td>
<td>&lt;(30 50 70)&gt;</td>
<td>&lt;{(30),(70)}&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;(30),(40 70),(90)&gt;</td>
<td>&lt;{(30)}{(40),(70),(40 70)}{(90)}&gt;</td>
</tr>
<tr>
<td>5</td>
<td>&lt;(90)&gt;</td>
<td>&lt;{(90)}&gt;</td>
</tr>
</tbody>
</table>

**Mapping**

<table>
<thead>
<tr>
<th>CID</th>
<th>Transformed Sequence</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;{(30)}{(90)}&gt;</td>
<td>&lt;{1}{5}&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;{(30)}{(40),(70),(40 70)}}</td>
<td>&lt;{1}{2,3,4}&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;{(30),(70)}&gt;</td>
<td>&lt;{1}{3}&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;{(30)}{(40),(70),(40 70)}{(90)}&gt;</td>
<td>&lt;{1}{2,3,4}{5}&gt;</td>
</tr>
<tr>
<td>5</td>
<td>&lt;{(90)}&gt;</td>
<td>&lt;{5}&gt;</td>
</tr>
</tbody>
</table>

---

Data Mining (Apriori Algorithm)  

### AprioriAll

```java
AprioriAll()
{
    Lk = {frequent 1-itemsets};
    k = 2; /* k represents the pass number. */
    while (Lk ! = ∅)
    {
        F = F ∪ Lk;
        Ck = New candidates of size k generated from Lk-1;
        for each customer-sequence c ∈ D
            increment the count of all candidates in Ck that are contained in c;
        Lk = All candidates in Ck with minimum support;
        k++;
    }
    return (F);
}
```

---

PAGE UNIVERSITY
AprioriSome and DynamicSome algorithms to find association rules with sequential patterns will be added.