Association Rule Mining

Instructor: Jesse Davis

Slides from: Chris Clifton, Pedro Domingos, Jeff Ullman
Announcements

- No class next week
  - Office hours next Tuesday 5:30-7:30/8
- Homework 3 is graded
- Homework 4 is due next Tuesday by midnight
Outline

- Homework 3 review
- Association rule mining
- Take away messages from class
Problem 1

- Accuracy 98-99% after several dozen iterations

- Generally slower than NB but higher accuracy
Problem 1: 2 BIG (RELATED) MISTAKES

- Setting bias by hand (e.g., $w_0x_0 = 0$)
  - Every input vector should have the same $x_0$ (say, 1)
  - Weight $w_0$ should be *learned* like any other weight

- Not normalizing feature values to range [0,1].
  - Notice that if $w_0x_0$ is fixed at 0 then $\sum w_i x_i > 0$ iff $n\sum w_i x_i > 0$, so normalization would indeed be unnecessary
  - If $w_0x_0 \neq 0$ you must normalize to ensure that model generalizes!
Bagging vs. Boosting

- Both techniques will improve performance of decision stumps

- Boosting should help more because it is better at reducing the ‘bias’ portion of error in addition to variance portion of error

- Bagging is better for handling variance
Bias
Bagging vs. Boosting - Errors

- Error 1: Bagging would help more
- Error 2: Boosting would help more
  - Explained why boosting is good
  - Didn't explain why bagging would be worse
GA Crossover

00110

11100

00111

10111

10110
If sum inputs > 0, then output is 1, else 0
X1 XOR X2

If sum inputs > 0, then output is 1, else 0

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) 0 0</td>
<td>0</td>
</tr>
<tr>
<td>b) 0 1</td>
<td>1</td>
</tr>
<tr>
<td>c) 1 0</td>
<td>1</td>
</tr>
<tr>
<td>d) 1 1</td>
<td>0</td>
</tr>
</tbody>
</table>
Genetic Algorithm For Sudoku

Goal: Generate Grid

Constraints:
1) Can’t change givens
2) 1-9 in each 3x3 subgrid
3) 1-9 in each row
4) 1-9 in each column

Solution components:
1) Initialization
2) Representation
3) Crossovers
4) Mutations
5) Fitness function
Sudoku: Initialization

Ensure that each 3x3 subgrid has 1—9 appearing exactly once!
Sudoku: Representation
Sudoku: Crossovers

Crossover only at subblock boundaries
Sudoku: Mutations

Disallow if swap involves a given
Sudoku: Fitness Function

- Representation and operators enforce these constraints:
  - Givens are not moved around
  - Each sub-block has 1--9 appearing exactly once
- Ignore these constraints:
  - Each column has 1--9 appearing exactly once
  - Each row has 1--9 appearing exactly once
- Fitness function: Penalize these states
  - Fewer violated constraints, the fitter the solution
  - Could penalize based on “how far off” solution is, i.e., row of all 9’s is worse than row with two 9’s
Outline

- Homework 3 review
- Association rule mining
  - Introduction and definitions
  - Naïve algorithm
  - Apriori
  - PCY
  - Limiting disk I/O
  - Presenting results, other metrics
- Take away messages from class
Association Rule Mining

Given: Set of transactions
Find: Rules that predict the occurrence of an item based on other items in the transaction

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

Implication means co-occurrence, not causality!
Why Association Rule Mining

- Motivation: Finding regularities in data
  - What products were often purchased together?
  - What kinds of DNA are sensitive to this new drug?

- Foundation for many data mining tasks
  - Association
  - Correlation
  - Causality

- Algorithms do not require labeled data or for a user to specify a predefined target concept
Market-Basket Model

- A large set of *items*, e.g., things sold in a supermarket
- A large set of *baskets (transactions)*, each of which is a small set of the items, e.g., the things one customer buys on one day

<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, Milk, 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>
Really a general many-many mapping (association) between two kinds of things

We ask about connections among “items,” not among “baskets”

The technology focuses on common events, not rare events (“long tail”)
Definition: Item Set

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

<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, Milk, 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>
Simplest question: find sets of items that appear “frequently” in the baskets

*Support count* for itemset $I$ = the number of baskets containing all items in $I$

*Support* Fraction of transactions that contain an itemset

Given a *support threshold* $s$, sets of items that appear in at least $s$ baskets are called *frequent itemsets*
### Example Support

<table>
<thead>
<tr>
<th>Items</th>
<th>Itemset</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread, Milk</td>
<td>{Br,M}</td>
<td>4</td>
</tr>
<tr>
<td>Bread, Milk, Diaper, Beer, Eggs</td>
<td>{Br,D}</td>
<td>3</td>
</tr>
<tr>
<td>Milk, Diaper, Beer, Coke</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bread, Milk, Diaper, Beer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bread, Milk, Diaper, Coke</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Support(\{Br,M\}) = \frac{4}{5} = 0.8
Support(\{Br,D\}) = \frac{3}{5} = 0.6
Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- 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\},
Example: Frequent Itemsets

- **Items**={milk, coke, pepsi, beer, juice}.
- **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: $\{m\}$, $\{c\}$, $\{b\}$, $\{j\}$, $\{m,b\}$
Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- 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\}
Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- 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: $\{m\}$, $\{c\}$, $\{b\}$, $\{j\}$, $\{m, b\}$, $\{b, c\}$
Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- 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\}
Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- Support = 3 baskets.
  - $B_1 = \{m, c, b\}$
  - $B_3 = \{m, b\}$
  - $B_5 = \{m, p, b\}$
  - $B_7 = \{c, b, j\}$
  - $B_2 = \{m, p, j\}$
  - $B_4 = \{c, j\}$
  - $B_6 = \{m, c, b, j\}$
  - $B_8 = \{b, c\}$
- Frequent itemsets: $\{m\}$, $\{c\}$, $\{b\}$, $\{j\}$, $\{m,b\}$, $\{b,c\}$, $\{c,j\}$
Definition: Association Rules

- If-then rules about the contents of baskets
- Given:
  - Set of *items*: \( I = \{i_1, i_2, \ldots, i_m\} \)
  - Set of *transactions*: \( D = \{d_1, d_2, \ldots, d_n\} \)
- An *association rule*: \( A \Rightarrow B \), where
  - \( A \subset I \)
  - \( B \subset I \)
  - \( A \cap B = \emptyset \)
  - \( \{i_1, i_2, \ldots, i_k\} \rightarrow j \) means: “if a basket contains all of \( i_1, \ldots, i_k \) then it is *likely* to contain \( j \).”
Definition: Confidence

- **Confidence** of this association rule is the conditional probability of $j$ given $i_1, \ldots, i_k$.
  - This gives a measure of how accurate the rule is.
  - $\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\sup\{\{A,B\}\}}{\sup(A)}$

<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, Milk, 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: Confidence

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

- An association rule: $\{m, b\} \rightarrow c$.
- Confidence $= \frac{2}{4} = 50\%$. 
Applications – (1)

- **Items** = products; **baskets** = sets of products someone bought in one trip to the store.

- **Example application**: given that many people buy beer and diapers together:
  - Run a sale on diapers; raise price of beer.

- Only useful if many buy diapers & beer.
Applications – (2)

- **Baskets** = sentences; **items** = documents containing those sentences.
- Items that appear together too often could represent plagiarism.
- Notice items do not have to be “in” baskets.
Applications – (3)

- **Baskets** = Web pages; **items** = words.
- Unusual words appearing together in a large number of documents, e.g., “Brad” and “Angelina,” may indicate an interesting relationship.
Outline

- Homework 3 review
- Association rule mining
  - Introduction and definitions
  - Naïve algorithm
  - Apriori
  - PCY
  - Limiting disk I/O
  - Presenting results, other metrics
- Take away messages from class
Scale of the Problem

- WalMart sells 100,000 items and can store billions of baskets

- The Web has billions of words and many billions of pages

- We have access to lots and lots of data...
Association Rule Mining Goal

- **Question:** “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

- **Hard part:** finding the frequent itemsets
  - **Note:** 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”
Creating Associating Rules

Given: Support s, confidence c

Step 1: Find all itemsets with support s

Step 2: For each frequent itemset L

  For each non-empty subset s of L

    Output the rule s → \{l-s\} if its confidence ≥ c
Example: Association Rule

For rule $A \Rightarrow C$:

$$\text{support} = \text{support}(\{A\} \cup \{C\}) = 50\%$$

$$\text{confidence} = \text{support}(\{A\} \cup \{C\})/\text{support}(\{A\}) = 66.6\%$$
Example: Itemset to Association Rule

<table>
<thead>
<tr>
<th>Items</th>
<th>Itemset</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread, Milk</td>
<td>{Br,M}</td>
<td>4</td>
</tr>
<tr>
<td>Bread, Milk, Diaper, Beer, Eggs</td>
<td>{Br,D}</td>
<td>3</td>
</tr>
<tr>
<td>Milk, Diaper, Beer, Coke</td>
<td>{M,Be}</td>
<td>3</td>
</tr>
<tr>
<td>Bread, Milk, Diaper, Beer</td>
<td>{M,D}</td>
<td>3</td>
</tr>
<tr>
<td>Bread, Milk, Diaper, Coke</td>
<td>{Br,M,D}</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>{M,D,Be}</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\{\text{Br}\} \rightarrow \{\text{M}\}, \ s = 0.8, \ c = 1.0 \\
\{\text{M}\} \rightarrow \{\text{Br}\}, \ s = 1.0, \ c = 0.8
\]

...

\[
\{\text{Br},\text{M}\} \rightarrow \{\text{D}\}, \ s = 0.8, \ c = 0.75 \\
\{\text{Be}\} \rightarrow \{\text{M},\text{D}\}, \ s = 0.6, \ c = 1.0
\]
Computation Model

- Typically, data is kept in flat files rather than in a database system
  - Stored on disk
  - Stored basket-by-basket
  - Expand baskets into pairs, triples, etc. as you read baskets
    - Use $k$ nested loops to generate all sets of size $k$. 
The true cost of mining disk-resident data is usually the number of disk I/O’s.

In practice, association-rule algorithms read the data in *passes* – all baskets read in turn.

Thus, we measure the cost by the number of passes an algorithm takes.
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
  - The number of different things we can count is limited by main memory
  - Swapping counts in/out is a disaster (why?)
Finding Frequent Pairs

- The hardest problem often turns out to be finding the **frequent pairs**
  - Often frequent pairs are common, frequent triples are rare
  - Probability of being frequent drops exponentially with size
  - number of sets grows more slowly with size
- We’ll concentrate on pairs, then extend to larger sets
Naïve Algorithm

- 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 \((\#\text{items})^{2}\) exceeds main memory
- Remember: \#items can be 100K (Wal-Mart) or 10B (Web pages)
Example: Counting Pairs

- Suppose $10^5$ items
- Suppose counts are 4-byte integers
- Number of pairs of items: $10^5(10^5-1)/2 = 5*10^9$ (approximately)
- Therefore, $2*10^{10}$ (20 gigabytes) of main memory needed
Two approaches:
1. Count all pairs, using a triangular matrix.
2. Keep a table of triples \([i, j, c]\) = “the count of the pair of items \([i, j]\) is \(c\).”

1. requires only 4 bytes/pair
2. requires 12 bytes, but only for those pairs with count > 0
Approaches Pictorially

Method (1) 4 per pair

Method (2) 12 per occurring pair
Approach 1

- Assign each item a number
- Count \( \{i, j\} \) only if \( i < j \)
- Keep pairs in the order
  - \( \{1, 2\} \)
  - \( \ldots \)
  - \( \{1, n\} \)
  - \( \{2, 3\} \)
  - \( \ldots \)
  - \( \{n-1, n\} \)
- Pair \( \{i, j\} \) at the position: \((i-1)(n-i/2) + j - i\)
Approach 2

- Total bytes used is about $12p$, where $p$ is the number of pairs that actually occur.

- Beats triangular matrix if at most $1/3$ of possible pairs actually occur.

- Require extra space for retrieval structure.
Outline

- Homework 3 review
- Association rule mining
  - Introduction and definitions
  - Naïve algorithm
  - Apriori
  - PCY
  - Limiting disk I/O
  - Presenting results, other metrics
- Take away messages from class
Apriori Algorithm

- Generate and test approach for discovering frequent itemsets

- Iterative approach
  - Find all frequent itemsets of size $k$ before finding frequent itemsets of size $k+1$
  - One pass through the data for each frequent itemset size
Apriori’s Key Idea

- **Apriori Principle (monotonicity):** if an itemset appears at least $s$ times, so do all its subsets
- **Contrapositive for pairs:** if item $i$ does not appear in $s$ baskets, then no pair including $i$ can appear in $s$ baskets
- Apriori principle holds due to the following property of the support measure:

$$ \forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y) $$
A-Priori Algorithm: Frequent Pairs

- **Pass 1**: Read baskets and count in main memory the occurrences of each item
  - Requires memory proportional to #items
  - *Frequent items*: those that appear s times

- **Pass 2**: Read baskets again and count in main memory only those pairs both of which were found in Pass 1 to be frequent
  - Requires memory proportional to square of *frequent* items, plus a list of the frequent items
  - *Frequent itemsets*: those that appear s times
The Apriori Algorithm

- **Join Step**: $C_k$ is generated by joining $L_{k-1}$ with itself
- **Prune Step**: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- **Pseudo-code**:

  $C_k$: Candidate itemset of size $k$
  $L_k$: frequent itemset of size $k$

  $L_1 = \{\text{frequent items}\}$

  for ($k = 1; L_k \neq \emptyset; k++$) do begin
    $C_{k+1} =$ candidates generated from $L_k$
    for each transaction $t$ in database do
      increment the count of all candidates in $C_{k+1}$ that are contained in $t$
    end
    $L_{k+1} =$ candidates in $C_{k+1}$ with min_support
  end

  return $\bigcup_k L_k$.
## Apriori: Pass 1

Given: Min support is 2

### Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
</tr>
</tbody>
</table>

### $C_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

Scan D
Apriori: Pass 1

Given: Min support is 2

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
</tr>
</tbody>
</table>

Scan D

$C_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

$L_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
</tbody>
</table>

Prune
Apriori: Pass 2

Given: Min support is 2

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
</tr>
</tbody>
</table>

$L_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

$C_2$

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
</tr>
<tr>
<td>{1,3}</td>
</tr>
<tr>
<td>{1,5}</td>
</tr>
<tr>
<td>{2,3}</td>
</tr>
<tr>
<td>{2,5}</td>
</tr>
<tr>
<td>{3,5}</td>
</tr>
</tbody>
</table>

$L_2$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
<td>1</td>
</tr>
<tr>
<td>{1,3}</td>
<td>2</td>
</tr>
<tr>
<td>{1,5}</td>
<td>1</td>
</tr>
<tr>
<td>{2,3}</td>
<td>2</td>
</tr>
<tr>
<td>{2,5}</td>
<td>3</td>
</tr>
<tr>
<td>{3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

Scan D
Apriori: Pass 2

Given: Min support is 2

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
</tr>
</tbody>
</table>

$L_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

$C_2$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
<td></td>
</tr>
<tr>
<td>{1,3}</td>
<td></td>
</tr>
<tr>
<td>{1,5}</td>
<td></td>
</tr>
<tr>
<td>{2,3}</td>
<td></td>
</tr>
<tr>
<td>{2,5}</td>
<td></td>
</tr>
<tr>
<td>{3,5}</td>
<td></td>
</tr>
</tbody>
</table>

$L_2$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
<td>1</td>
</tr>
<tr>
<td>{1,3}</td>
<td>2</td>
</tr>
<tr>
<td>{1,5}</td>
<td>1</td>
</tr>
<tr>
<td>{2,3}</td>
<td>2</td>
</tr>
<tr>
<td>{2,5}</td>
<td>3</td>
</tr>
<tr>
<td>{3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

Scan D

Prune

$L_2$
Apriori: Pass 2

Given: Min support is 2

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
</tr>
</tbody>
</table>

\(L_1\)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

\(C_2\)

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
</tr>
<tr>
<td>{1,3}</td>
</tr>
<tr>
<td>{1,5}</td>
</tr>
<tr>
<td>{2,3}</td>
</tr>
<tr>
<td>{2,5}</td>
</tr>
<tr>
<td>{3,5}</td>
</tr>
</tbody>
</table>

\(L_2\)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,3}</td>
<td>2</td>
</tr>
<tr>
<td>{2,3}</td>
<td>2</td>
</tr>
<tr>
<td>{2,5}</td>
<td>3</td>
</tr>
<tr>
<td>{3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

Scan D
Given: Min support is 2

### Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
</tr>
</tbody>
</table>

### L₂

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,3}</td>
<td>2</td>
</tr>
<tr>
<td>{2,5}</td>
<td>3</td>
</tr>
<tr>
<td>{3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

### C₃

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2,3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

### L₃

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2,3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>
Apriori: Join Step

- Suppose the items in \( L_{k-1} \) are listed in an order.
- Join each element in \( L_{k-1} \) with itself.
- If \( l_1, l_2 \in L_{k-1} \), they are joinable if:
  - The first \( k-2 \) items in \( l_1 \) and \( l_2 \) are the same.
  - \( \ldots \) AND
  - \( l_1[k-2] = l_2[k-2] \)
For each candidate itemsets $C_k$
  - Look at each subset of size $k-1$ [i.e., drop one item from the candidate]
    - If ANY one of these subsets isn’t frequent, discard this candidate
    - Application of the Apriori principle
Example: 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 $ace$
  - Note: other joins (i.e., $abc$ and $acd$, $abc$ and $ace$, etc. are illegal)

- Pruning:
  - $acde$ is removed because $ade$ is not in $L_3$
- $C_4 = \{abcd\}$
Outline

- Homework 3 review
- Association rule mining
  - Introduction and definitions
  - Naïve algorithm
  - Apriori
  - PCY
  - Limiting disk I/O
  - Presenting results, other metrics
- Take away messages from class
Aside: Hash-Based Filtering

- Simple problem: I have a set $S$ of one billion strings of length 10.
- I want to scan a larger file $F$ of strings and output those that are in $S$.
- I have 1GB of main memory.
  - So I can’t afford to store $S$ in memory.
Solution – (1)

- Create a **bit array** of 8 billion bits, initially all 0’s.
- Choose a hash function $h$ with range $[0, 8 \times 10^9]$, and hash each member of $S$ to one of the bits, which is then set to 1.
- Filter the file $F$ by hashing each string and outputting only those that hash to a 1.
Solution – (2)

File $F$

Filter

0010001011000

To output; may be in $S$.

Drop; surely not in $S$. 
PCY Algorithm

- During Pass 1 of A-priori, most memory is idle.
- Idea: Use tmemory for a hash table
  - Hash pairs of items that appear in a transaction – we need to generate these
  - Just the count, not the pairs themselves
  - Interested in the presence of a pair AND whether it is present at least $s$ (support) times
PCY Algorithm: Pass 1

FOR (each basket) {
    FOR (each item in the basket) {
        add 1 to item’s count;
    }
    FOR (each pair of items) {
        hash the pair to a bucket;
        add 1 to the count for that bucket
    }
}
Observation About Buckets

- A bucket that a frequent pair hashes to meets minimum support threshold
  - Cannot eliminate any member of this bucket

- Even without any frequent pair, a bucket can be frequent
  - Cannot eliminate any member of this bucket

- Best case: Count for a bucket is less than minimum support
  - Eliminate all pairs hashed to this bucket even if the pair consists of two frequent items
PCY: Pass 1

Given: Min support is 2

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>{1,3}, {1,4}, {3,4}</th>
<th>{2,3}, {2,5}, {3,5}</th>
<th>{1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}</th>
<th>{2,5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scan D

$C_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

Bucket

<table>
<thead>
<tr>
<th>Bucket</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
### PCY: Between Passes

**Given:** Min support is 2

**Database D**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,3,4</td>
</tr>
<tr>
<td>2</td>
<td>2,3,5</td>
</tr>
<tr>
<td>3</td>
<td>1,2,3,5</td>
</tr>
<tr>
<td>4</td>
<td>2,5</td>
</tr>
</tbody>
</table>

**C₁**

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

**C₂**

**Itemset**

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
</tr>
<tr>
<td>{1,3}</td>
</tr>
<tr>
<td>{1,5}</td>
</tr>
<tr>
<td>{2,3}</td>
</tr>
<tr>
<td>{2,5}</td>
</tr>
<tr>
<td>{3,5}</td>
</tr>
</tbody>
</table>

**Bucket**

<table>
<thead>
<tr>
<th>Bucket</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Between Passes

- Replace the buckets by a bit-vector:
  - 1 means the bucket is frequent
  - 0 means it is not frequent

- 4-byte integers are replaced by bits, so the bit-vector requires 1/32 of memory

- Also, decide which items are frequent and list them for the second pass
Picture of PCY

- Item counts
  - Hash table
  - Pass 1
- Frequent items
  - Bitmap
  - Counts of candidate pairs
  - Pass 2
PCY Algorithm: Pass 2

- Count all pairs \(\{i, j\}\) that meet the conditions for being a candidate pair:
  1. Both \(i\) and \(j\) are frequent items.
  2. The pair \(\{i, j\}\), hashes to a bucket number whose bit in the bit vector is 1.

- Notice all these conditions are necessary for the pair to have a chance of being frequent.
Outline

- Homework 3 review
- Association rule mining
  - Introduction and definitions
  - Naïve algorithm
  - Apriori
  - PCY
  - Limiting disk I/O
  - Presenting results, other metrics
- Take away messages from class
All (Or Most) Frequent Itemsets in \( \leq 2 \) Passes

- A-Priori, PCY, etc., take \( k \) passes to find frequent itemsets of size \( k \)
- Other techniques use 2 or fewer passes for all sizes:
  - Simple algorithm
  - SON (Savasere, Omiecinski, and Navathe)
  - Toivonen
Simple Algorithm

- Take a random sample of the market baskets that fits in main memory
- Run a-priori or one of its improvements in main memory, so you don’t pay for disk I/O each time you increase the size of itemsets
  - Be sure you leave enough space for counts

| Copy of sample baskets | Space for counts |
Algorithm Details

- Scale back support threshold a suitable number
  - E.g., if sample is 1/100 of the baskets, use $s/100$ as your support threshold instead of $s$
- Optional: Verify that your guesses are truly frequent in the entire data set by a second pass

- Miss sets frequent in whole but not in sample
  - Smaller threshold, e.g., $s/125$, helps limit misses, but requires more space
Toivonen’s Algorithm

- Use simple algorithm, but lower the threshold \( s \) for the sample
  - Example: if the sample is 1% of the baskets, use \( s/125 \) vs. \( s/100 \).
  - Goal: Avoid missing truly frequent itemsets

- Add to the itemsets that are frequent in the sample the negative border of these itemsets.

- An itemset is in the negative border if it is not deemed frequent in the sample, but all its immediate subsets are
Example: Negative Border

- \(ABCD\) is in the negative border if and only if:
  1. It is not frequent in the sample, but
  2. All of \(ABC, BCD, ACD,\) and \(ABD\) are.

- \(A\) is in the negative border if and only if it is not frequent in the sample.
  - Because the empty set is always frequent.
  - Unless there are fewer baskets than the support threshold (silly case).
Picture of Negative Border

Negative Border

Frequent Itemsets from Sample

... tripletons
doubletons
singletons
Toivonen’s Algorithm Continued

- In a second pass, count all candidate frequent itemsets from the first pass, and also count their negative border

- If no itemset from the negative border turns out to be frequent, then the candidates found to be frequent in the whole data are exactly the frequent itemsets
Toivonen’s Algorithm Continued

- What if we find that something in the negative border is actually frequent?
- We must start over again!
- Try to choose the support threshold so the probability of failure is low, while the number of itemsets checked on the second pass fits in main-memory.
If Something in the Negative Border is Frequent . . .

We broke through the negative border. How far does the problem go?

... tripletons
doubletons
singletons

Frequent Itemsets from Sample

Negative Border
Theorem:

- If there is an itemset that is frequent in the whole, but not frequent in the sample, then there is a member of the negative border for the sample that is frequent in the whole.
Proof

- Suppose not; i.e.;
  1. There is an itemset $S$ frequent in the whole but not frequent in the sample, and
  2. Nothing in negative border is frequent in the whole

- Let $T$ be a **smallest** subset of $S$ that is not frequent in the sample

- $T$ is frequent in the whole ($S$ is frequent + monotonicity)

- $T$ is in the negative border (else not “smallest”)
Outline

- Homework 3 review
- Association rule mining
  - Introduction and definitions
  - Naïve algorithm
  - Apriori
  - PCY
  - Limiting disk I/O
  - Presenting results, other metrics
- Take away messages from class
Compacting the Output

1. **Maximal Frequent itemsets**: no immediate superset is frequent

2. **Closed itemsets**: no immediate superset has the same count (> 0).
   - Stores not only frequent information, but exact counts
### Example: Maximal/Closed

<table>
<thead>
<tr>
<th></th>
<th>Count</th>
<th>Maximal (s=3)</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AB</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AC</td>
<td>2</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BC</td>
<td>3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ABC</td>
<td>2</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Frequent, but superset BC also frequent
- Frequent, and its only superset, ABC, not freq
- Superset BC has same count
- Its only superset, ABC, has smaller count
Interestingness Measurements

- Two popular objective measurements:
  - *support*
  - *confidence*

- Subjective measures: A rule (pattern) is interesting if it is:
  - *Unexpected* (surprising to the user)
  - *Actionable* (the user can do something with it)
Criticism of Support and Confidence

- Example: 5000 students
  - 3000 play basketball
  - 3750 eat cereal
  - 2000 both play basketball and eat cereal

- \( \text{play basketball} \Rightarrow \text{eat cereal} \) [40\%, 66.7\%]
  - misleading as the overall percentage of students eating cereal is 75\% which is higher than 66.7\%

- \( \text{play basketball} \Rightarrow \text{not eat cereal} \) [20\%, 33.3\%]
  - More accurate, but lower support and confidence

<table>
<thead>
<tr>
<th></th>
<th>basketball</th>
<th>not basketball</th>
<th>sum(row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cereal</td>
<td>2000</td>
<td>1750</td>
<td>3750</td>
</tr>
<tr>
<td>not cereal</td>
<td>1000</td>
<td>250</td>
<td>1250</td>
</tr>
<tr>
<td>sum(col.)</td>
<td>3000</td>
<td>2000</td>
<td>5000</td>
</tr>
</tbody>
</table>
Statistical Measures

- $P(S \land B) = P(S) \times P(B) \Rightarrow$ Statistical independence
- $P(S \land B) > P(S) \times P(B) \Rightarrow$ Positively correlated
- $P(S \land B) < P(S) \times P(B) \Rightarrow$ Negatively correlated

- Lift($A \Rightarrow B$) = $\frac{P(B \mid A)}{P(B)}$
Example: Lift

![Table]

<table>
<thead>
<tr>
<th></th>
<th>Coffee</th>
<th>Coffee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tea</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tea</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>10</td>
</tr>
</tbody>
</table>

Association Rule: Tea $\rightarrow$ Coffee

Confidence $= P(\text{Coffee}|\text{Tea}) = 0.75$

but $P(\text{Coffee}) = 0.9$

$\Rightarrow$ Lift $= 0.75/0.9 = 0.8333$ (< 1, therefore is negatively associated)
## Presentation of Association Rules (Table Form)

<table>
<thead>
<tr>
<th>Body</th>
<th>Implies</th>
<th>Head</th>
<th>Supp (%)</th>
<th>Conf (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>revenue(x) = '0.00~500.00'</td>
<td>28.45</td>
<td>40.4</td>
</tr>
<tr>
<td>2</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>revenue(x) = '500.00~1000.00'</td>
<td>20.46</td>
<td>29.05</td>
</tr>
<tr>
<td>3</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>59.17</td>
<td>34.04</td>
</tr>
<tr>
<td>4</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>revenue(x) = '1000.00~1500.00'</td>
<td>10.45</td>
<td>14.84</td>
</tr>
<tr>
<td>5</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>region(x) = 'United States'</td>
<td>22.56</td>
<td>32.04</td>
</tr>
<tr>
<td>6</td>
<td>cost(x) = '1000.00~2000.00'</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>12.91</td>
<td>69.34</td>
</tr>
<tr>
<td>7</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>revenue(x) = '0.00~500.00'</td>
<td>28.45</td>
<td>34.54</td>
</tr>
<tr>
<td>8</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>cost(x) = '1000.00~2000.00'</td>
<td>12.91</td>
<td>15.67</td>
</tr>
<tr>
<td>9</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>region(x) = 'United States'</td>
<td>25.9</td>
<td>31.45</td>
</tr>
<tr>
<td>10</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>59.17</td>
<td>71.86</td>
</tr>
<tr>
<td>11</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>product_line(x) = 'Tents'</td>
<td>13.52</td>
<td>16.42</td>
</tr>
<tr>
<td>12</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>revenue(x) = '500.00~1000.00'</td>
<td>19.67</td>
<td>23.88</td>
</tr>
<tr>
<td>13</td>
<td>product_line(x) = 'Tents'</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>13.52</td>
<td>98.72</td>
</tr>
<tr>
<td>14</td>
<td>region(x) = 'United States'</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>25.9</td>
<td>31.94</td>
</tr>
<tr>
<td>15</td>
<td>region(x) = 'United States'</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>22.56</td>
<td>71.39</td>
</tr>
<tr>
<td>16</td>
<td>revenue(x) = '0.00~500.00'</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>28.45</td>
<td>100</td>
</tr>
<tr>
<td>17</td>
<td>revenue(x) = '0.00~500.00'</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>28.45</td>
<td>100</td>
</tr>
<tr>
<td>18</td>
<td>revenue(x) = '1000.00~1500.00'</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>10.45</td>
<td>96.75</td>
</tr>
<tr>
<td>19</td>
<td>revenue(x) = '500.00~1000.00'</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>20.46</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>revenue(x) = '500.00~1000.00'</td>
<td>order_qty(x) = '0.00~100.00'</td>
<td>19.67</td>
<td>96.14</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>revenue(x) = '0.00<del>500.00' AND order_qty(x) = '0.00</del>100.00'</td>
<td>28.45</td>
<td>40.4</td>
</tr>
<tr>
<td>24</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>revenue(x) = '0.00<del>500.00' AND order_qty(x) = '0.00</del>100.00'</td>
<td>28.45</td>
<td>40.4</td>
</tr>
<tr>
<td>25</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>revenue(x) = '500.00<del>1000.00' AND order_qty(x) = '0.00</del>100.00'</td>
<td>19.67</td>
<td>27.93</td>
</tr>
<tr>
<td>26</td>
<td>cost(x) = '0.00~1000.00'</td>
<td>revenue(x) = '500.00<del>1000.00' AND order_qty(x) = '0.00</del>100.00'</td>
<td>19.67</td>
<td>27.93</td>
</tr>
<tr>
<td>27</td>
<td>cost(x) = '0.00<del>1000.00' AND order_qty(x) = '0.00</del>100.00'</td>
<td>revenue(x) = '500.00~1000.00'</td>
<td>19.67</td>
<td>33.23</td>
</tr>
</tbody>
</table>
Visualization of Association Rule Using Rule Graph

- Education Level = [High School Degree]
- Gender = [F]
- Marital Status = [M]
- Education Level = [Bachelors Degree]
- Marital Status = [S]
- Education Level = [Partial College]

For Help, press F1
Outline

- Homework 3 review
- Association rule mining
- Take away messages from class
Take Away: Feature Construction

Real World

Feature Space

Concepts/ Classes/ Decisions

Feature construction is crucial!!!

Worth spending time on
Take Away: Empirical Evaluation

Use statistical techniques such as 10-fold cross validation to get meaningful results.
Take Away: Empirical Evaluation

- Often, an ML system has to choose when to stop learning, select among alternative answers, etc.
- One wants the model that produces the highest accuracy on future examples (“overfitting avoidance”)
- It is a “cheat” to look at the test set while still learning
- Better method
  - Set aside part of the training set
  - Measure performance on this “tuning” data to estimate future performance for a given set of parameters
  - Use best parameter settings, train with all training data (except test set) to estimate future performance on new examples
Take Away: Empirical Evaluation

- Accuracy only can be misleading
- Look at alternative measures
  - True positive rate/recall
  - False positive rate
  - Precision
  - Area under the curve
Take Away: Be Wary of Assumptions

Simplification: Assumed investments were independent.

Reality: All similar type of bet.
Take Away: Simple Methods

- Simple approaches often work reasonable well in practice
  - 1-nn
  - Naïve Bayes
  - Perceptron

- Often worth trying first
Take Away: Ensembles

1) Many classifiers often better than single classifier
2) Bagging/boosting are simple and very effective
3) Worth trying!
Summary

- Association rules: Efficient way to mine interesting information very large databases
  - Get probabilities
  - Don’t require user guidance for interesting patterns
- Apriori algorithm and it’s extensions allow the user to gather a good deal of information without too many passes through data
Questions?