Data mining - detailed outline

- Problem
  - Getting the data: Data Warehouses, DataCubes, OLAP
  - Supervised learning: decision trees
  - Unsupervised learning
    - association rules
    - (clustering)

Problem

Given: multiple data sources
Find: patterns (classifiers, rules, clusters, outliers...)

NY
- sales(p-id, c-id, date, $price)
- customers( c-id, age, income, ...)

PGH

SF
Data Ware-housing

First step: collect the data, in a single place (= Data Warehouse)

How?
How often?
How about discrepancies / non-homegeneities?

How? A: Triggers/Materialized views
How often? A: [Art!]
How about discrepancies / non-homegeneities? A: Wrappers/Mediators

Step 2: collect counts. (DataCubes/OLAP)
Eg.:
OLAP

Problem: “is it true that shirts in large sizes sell better in dark colors?”

<table>
<thead>
<tr>
<th>sales</th>
<th>ci-d</th>
<th>p-id</th>
<th>Size</th>
<th>Color</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>C10</td>
<td>Shirt</td>
<td>L</td>
<td>Blue</td>
<td>30</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td>C10</td>
<td>Pants</td>
<td>XL</td>
<td>Red</td>
<td>50</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>C20</td>
<td>Shirt</td>
<td>XL</td>
<td>White</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

DataCubes

‘color’, ‘size’: DIMENSIONS
‘count’: MEASURE

<table>
<thead>
<tr>
<th>color</th>
<th>size</th>
<th>C / S</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blue</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOT</td>
<td>23</td>
<td>6</td>
<td>18</td>
<td>47</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### DataCubes

- **Dimensions**: `color`, `size`
- **Measure**: `count`

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td>Blue</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td><strong>TOT</strong></td>
<td>23</td>
<td>6</td>
<td>18</td>
<td>47</td>
</tr>
</tbody>
</table>
DataCubes

'color', 'size': DIMENSIONS
'count': MEASURE

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td>Blue</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>TOT</td>
<td>25</td>
<td>6</td>
<td>18</td>
<td>47</td>
</tr>
</tbody>
</table>

DataCube

SQL query to generate DataCube:
• Naively (and painfully:)
  select size, color, count(*)
  from sales where p-id = 'shirt'
  group by size, color

  select size, count(*)
  from sales where p-id = 'shirt'
  group by size

  ...

DataCubes

SQL query to generate DataCube:
• with 'cube by' keyword:
  select size, color, count(*)
  from sales
  where p-id = 'shirt'
  **cube by** size, color
DataCubes

DataCube issues:
Q1: How to store them (and/or materialize portions on demand)
Q2: Which operations to allow

A: ROLAP/MOLAP
A: roll-up, drill down, slice, dice

[More details: book by Han+Kamber]
DataCubes

Q1: How to store a dataCube?
A1: Relational (R-OLAP)

<table>
<thead>
<tr>
<th>Color</th>
<th>Size</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>Blue</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Tot</td>
<td>23</td>
<td>6</td>
</tr>
</tbody>
</table>

Q1: How to store a dataCube?
A2: Multi-dimensional (M-OLAP)
A3: Hybrid (H-OLAP)

<table>
<thead>
<tr>
<th>C/S</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td>Blue</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Tot</td>
<td>23</td>
<td>6</td>
<td>18</td>
<td>47</td>
</tr>
</tbody>
</table>

Pros/Cons:
ROLAP strong points: (DSS, Metacube)
DataCubes

Pros/Cons:
ROLAP strong points: (DSS, Metacube)
• use existing RDBMS technology
• scale up better with dimensionality

DataCubes

Pros/Cons:
MOLAP strong points: (EssBase/hyperion.com)
• faster indexing
  (careful with: high-dimensionality; sparseness)
HOLAP: (MS SQL server OLAP services)
• detail data in ROLAP; summaries in MOLAP

DataCubes

Q1: How to store a dataCube
Q2: What operations should we support?
DataCubes

Q2: What operations should we support?

<table>
<thead>
<tr>
<th></th>
<th>C/S</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Red</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Blue</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>TOT</td>
<td>23</td>
<td>6</td>
<td>18</td>
<td>47</td>
</tr>
</tbody>
</table>

Roll-up

<table>
<thead>
<tr>
<th></th>
<th>C/S</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Red</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Blue</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>TOT</td>
<td>23</td>
<td>6</td>
<td>18</td>
<td>47</td>
</tr>
</tbody>
</table>

Drill-down

<table>
<thead>
<tr>
<th></th>
<th>C/S</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Red</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Blue</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>TOT</td>
<td>23</td>
<td>6</td>
<td>18</td>
<td>47</td>
</tr>
</tbody>
</table>
DataCubes

Q2: What operations should we support?

- Slice
- Dice
- Roll-up
- Drill-down
- Slice
- Dice
- (Pivot/rotate; drill-across; drill-through
- top N
- moving averages, etc)
D/W - OLAP - Conclusions

- D/W: copy (summarized) data + analyze
- OLAP - concepts:
  - DataCube
  - R/M/H-OLAP servers
  - ‘dimensions’; ‘measures’

Outline

- Problem
- Getting the data: Data Warehouses, DataCubes, OLAP
- Supervised learning: decision trees
- Unsupervised learning
  - association rules
  - (clustering)

Decision trees - Problem

<table>
<thead>
<tr>
<th>Age</th>
<th>Chol-level</th>
<th>Gender</th>
<th>CLASS-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>150</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

??
Decision trees

- Pictorially, we have
  
  num. attr#1 (eg., ‘age’)
  
  num. attr#2 (eg., chol-level)

- and we want to label ‘?’

- so we build a decision tree:

  num. attr#1 (eg., ‘age’)
  
  num. attr#2 (eg., chol-level)

  50
Decision trees

- so we build a decision tree:

\[
\begin{array}{c}
\text{age} < 50 \quad \text{Y} \\
+ \quad \text{chol.} < 40 \quad \text{N} \\
- \quad \text{...}
\end{array}
\]

Outline

- Problem
- Getting the data: Data Warehouses, Data Cubes, OLAP
- Supervised learning: decision trees
  - problem
  - approach
  - scalability enhancements
- Unsupervised learning
  - association rules
  - (clustering)

Decision trees

- Typically, two steps:
  - tree building
  - tree pruning (for over-training/over-fitting)
Tree building

• How?

num. attr#1 (eg., 'age')

num. attr#2
(eg., chol-level)

+  +  -
+  +  -
+  +  -
+  +  -
+  +  -
+  +  -
+  +  -
+  +  -

• A: Partition, recursively - pseudocode:
  
  Partition (Dataset S)
  if all points in S have same label
  then return
  evaluate splits along each attribute A
  pick best split, to divide S into S1 and S2
  Partition(S1); Partition(S2)

Tree building

• Q1: how to introduce splits along attribute $A_i$
• Q2: how to evaluate a split?
Tree building

• Q1: how to introduce splits along attribute $A_i$

• A1:
  – for num. attributes:
    • binary split, or
    • multiple split
  – for categorical attributes:
    • compute all subsets (expensive!), or
    • use a greedy algo

• Q2: how to evaluate a split?

• A: by how close to uniform each subset is - ie., we need a measure of uniformity:
Tree building

entropy: $H(p^+, p^-)$

Any other measure?

Tree building

entropy: $H(p_+, p_-)$

'gini' index: $1-p_+^2 - p_-^2$

Tree building

entropy: $H(p_+, p_-)$

'gini' index: $1-p_+^2 - p_-^2$

(How about multiple labels?)
Tree building

Intuition:
- entropy: #bits to encode the class label
- gini: classification error, if we randomly guess '+' with prob. $p_+$

Thus, we choose the split that reduces entropy/classification-error the most: Eg.:

Before split: we need $(n_+ \times n_+ + n_- \times n_-) \times H(p_+, p_-) = (7+6) \times H(7/13, 6/13)$ bits total, to encode all the class labels

After the split we need:
- 0 bits for the first half and
- $(2+6) \times H(2/8, 6/8)$ bits for the second half
Tree pruning

- **What for?**

  num. attr#1 (eg., 'age')
  num. attr#2 (eg., chol-level)

  ![Diagram](image)

**Shortcut for scalability: DYNAMIC pruning:**

- stop expanding the tree, if a node is ‘reasonably’ homogeneous
  - ad hoc threshold [Agrawal+, vldb92]
  - ( Minimum Description Language (MDL) criterion (SLIQ) [Mehta+, edbt96] )

**Q: How to do it?**

- A1: use a ‘training’ and a ‘testing’ set - prune nodes that improve classification in the ‘testing’ set. (Drawbacks?)
- A2: or, rely on MDL (= Minimum Description Language)
Outline

• Problem
• Getting the data: Data Warehouses, DataCubes, OLAP
• Supervised learning: decision trees
  – problem
  – approach
  – scalability enhancements
• Unsupervised learning
  – association rules
  – (clustering)

Scalability enhancements

• Interval Classifier [Agrawal+, vldb92]: dynamic pruning
• SLIQ: dynamic pruning with MDL; vertical partitioning of the file (but label column has to fit in core)
• SPRINT: even more clever partitioning

Conclusions for classifiers

• Classification through trees
• Building phase - splitting policies
• Pruning phase (to avoid over-fitting)
• For scalability:
  – dynamic pruning
  – clever data partitioning
Outline

- Problem
- Getting the data: Data Warehouses, DataCubes, OLAP
- Supervised learning: decision trees
  - problem
  - approach
  - scalability enhancements
- Unsupervised learning
  - association rules
  - (clustering)

Association rules - idea

[Agrawal+ SIGMOD93]

- Consider 'market basket' case:
  (milk, bread)
  (milk)
  (milk, chocolate)
  (milk, bread)
- Find 'interesting things', eg., rules of the form:
  milk, bread -> chocolate | 90%

Association rules - idea

In general, for a given rule
  Ij, Ik, ... Im | c

'c' = 'confidence' (how often people by Ix, given that they have bought Ij, ... Im)
's' = support: how often people buy Ij, ... Im, Ix
Association rules - idea

Problem definition:
• given
  – a set of ‘market baskets’ (=binary matrix, of \(N\) rows/baskets and \(M\) columns/products)
  – min-support ‘s’ and
  – min-confidence ‘c’
• find
  – all the rules with higher support and confidence

Closely related concept: “large itemset”
\(I_j, I_k, \ldots, I_m, I_x\)
is a ‘large itemset’, if it appears more than ‘min-support’ times

Observation: once we have a ‘large itemset’, we can find out the qualifying rules easily (how?)
Thus, let’s focus on how to find ‘large itemsets’

Naive solution: scan database once; keep \(2^{||I||}\) counters
Drawback?
Improvement?
Association rules - idea

Naive solution: scan database once; keep $2^{2*|I|}$ counters
Drawback? $2^{**1000}$ is prohibitive...
Improvement? scan the db $|I|$ times, looking for 1-, 2-, etc itemsets

Eg., for $|I|$=3 items only (A, B, C), we have

```
A  B  C
100 200 2
min-sup:10
```

Association rules - idea

```
A  B  C
100 200 2
```

min-sup:10

Association rules - idea

```
A  B  C
100 200 2
```

min-sup:10
Association rules - idea

Anti-monotonicity property:
if an itemset fails to be ‘large’, so will every superset
of it (hence all supersets can be pruned)

Sketch of the (famous!) ‘a-priori’ algorithm
Let $L(i-1)$ be the set of large itemsets with $i-1$
elements
Let $C(i)$ be the set of candidate itemsets (of size $i$)

Association rules - idea

Compute $L(1)$, by scanning the database.
repeat, for $i=2,3...$,
   “join” $L(i-1)$ with itself, to generate $C(i)$
   two itemset can be joined, if they agree on their first $i-2$ elements
   prune the itemsets of $C(i)$ (how?)
   scan the db, finding the counts of the $C(i)$ itemsets - set
   this to be $L(i)$
   unless $L(i)$ is empty, repeat the loop
(see example 6.1 in [Han+Kamber])

Association rules - Conclusions

Association rules: a new tool to find patterns
• easy to understand its output
• fine-tuned algorithms exist
• still an active area of research
Overall Conclusions

- Data Mining: of high commercial interest
- DM = DB+ ML+ Stat
- Data warehousing / OLAP: to get the data
- Tree classifiers (SLIQ, SPRINT)
- Association Rules - 'a-priori' algorithm
- (clustering: BIRCH, CURE, OPTICS)

Reading material


Additional references

- Jiawei Han and Micheline Kamber, Data Mining, Morgan Kaufman, 2001, chapters 2.2-2.3, 6.1-6.2, 7.3.5