Data Mining
A crash course!

C. Faloutsos

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)

SF customers(c-id, age, income, ...)

Data Ware-housing
First step: collect the data, in a single place (= Data Warehouse)
How?
How often?
How about discrepancies / non-homegeneities?

Data Ware-housing
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>Size</th>
<th>Color</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Blue</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</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

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

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

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

...
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>3</td>
</tr>
<tr>
<td>Blue</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>0</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)

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

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

Q1: How to store a dataCube
Q2: What operations should we support?
Q3: How to index a dataCube?
Q2: What operations should we support?

- **Roll-up**
  - C/S: Red 20, Blue 3, Gray 0
  - Size: M 3, L 5

- **Drill-down**
  - C/S: Red 20, Blue 3, Gray 0
  - Size: M 3, L 5

- **Slice**
  - C/S: Red 20, Blue 3, Gray 0
  - Size: M 3, L 5

- **Dice**
  - C/S: Red 20, Blue 3, Gray 0
  - Size: M 3, L 5

- (Pivot/rotate; drill-across; drill-through; top N; moving averages, etc)
DataCubes

Q1: How to store a dataCube
Q2: What operations should we support?
Q3: How to index a dataCube?

A1: Bitmaps

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td></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>

A2: Join indices (see [Han+Kamber])

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td></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>

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>+</td>
</tr>
</tbody>
</table>

...?

Decision trees

- Pictorially, we have

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

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

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

...?

Decision trees

- and we want to label '?'

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

?  +  -
+  +  +
+  +  -
-  +  -

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

Decision trees

- so we build a decision tree:

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

40 ...

?  +  -
+  +  +
+  +  -
-  +  -

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

50

Decision trees

- so we build a decision tree:

age<50

Y N
+  Y
N 

chol. <40

Outline

- Problem
- Getting the data: Data Warehouses, DataCubes, 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#2 (eg., chol-level)
   + + + -
   + + - -
   + - - -
   num. attr#1 (eg., 'age')

Tree building

- How?

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 $A_i$
- Q2: how to evaluate a split?
- A: by how close to uniform each subset is - i.e., we need a measure of uniformity:

```
<table>
<thead>
<tr>
<th>$p_+$</th>
<th></th>
<th>$p_-$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
```

Any other measure?

```
Tree building
entropy: $H(p_+, p_-)$

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

(How about multiple labels?)
```

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.:

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

num. attr#1 (eg., ‘age’)
```
Tree building

- Before split: we need
  \( (n_+ + n_-) \times H(p_+, p_-) = (7+6) \times H(7/13, 6/13) \)
  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?

\[
\begin{array}{c|c|c|c|c}
\text{num. attr#2} & \text{num. attr#1 (eg., 'age')} \\
\hline
\text{(eg., chol-level)} & \text{...} \\
\end{array}
\]

Tree pruning

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) [Mehra+, edbt96]

Tree pruning

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) - in detail.

Tree pruning

-envision the problem as compression (of what?)

- and try to min. the # bits to compress
  (a) the class labels AND
  (b) the representation of the decision tree
a brilliant idea - eg.: best $n$-degree polynomial to compress these points:

the one that minimizes (sum of errors + $n$)

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

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
\[ I_j, I_k, ... I_m \rightarrow I_x | c \]
'c' = 'confidence' (how often people by \( I_x \), given that they have bought \( I_j, ... I_m \))
's' = support: how often people buy \( I_j, ... I_m, I_x \)

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

Association rules - idea

Closely related concept: "large itemset"
\[ I_j, I_k, ... I_m, I_x \]
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'

Association rules - idea

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

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

\[ \begin{array}{ccc}
A & B & C \\
100 & 200 & 2 \\
\end{array} \]
first pass
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 itemsets 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

- Agrawal, R., T. Imielinski, A. Swami, 'Mining Association Rules between Sets of Items in Large Databases', SIGMOD 1993,
Additional references

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