Roadmap

1) Roots: System R and Ingres
2) Implementation: buffering, indexing, q-opt
3) Transactions: locking, recovery
4) Distributed DBMSs
5) Parallel DBMSs: Gamma, Alphasort
6) OODB DBMS
7) Data Analysis - data mining
   data cubes
   association rules
8) Benchmarks
9) Vision statements
   extras (streams/sensors, graphs, multimedia, web, fractals)

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)

PGH

???

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-homogeneities?
Data Warehousing

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

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

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/M/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 3 5</td>
<td>28</td>
</tr>
<tr>
<td>C10</td>
<td>Pants</td>
<td>XL</td>
<td>Red</td>
<td>50</td>
<td>3 3 8</td>
<td>14</td>
</tr>
<tr>
<td>C20</td>
<td>Shirt</td>
<td>XL</td>
<td>White</td>
<td>20</td>
<td>0 0 5</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23 6 18</td>
<td>47</td>
</tr>
</tbody>
</table>

DataCubes

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

\[\begin{array}{c|ccc|c}
\text{C/S} & \text{S} & \text{M} & \text{L} & \text{TOT} \\
\hline
\text{Red} & 20 & 3 & 5 & 28 \\
\text{Blue} & 3 & 3 & 8 & 14 \\
\text{Gray} & 0 & 0 & 5 & 5 \\
\text{TOT} & 23 & 6 & 18 & 47 \\
\end{array}\]

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

\[\begin{array}{c|ccc|c}
\text{C/S} & \text{S} & \text{M} & \text{L} & \text{TOT} \\
\hline
\text{Red} & 20 & 3 & 5 & 28 \\
\text{Blue} & 3 & 3 & 8 & 14 \\
\text{Gray} & 0 & 0 & 5 & 5 \\
\text{TOT} & 23 & 6 & 18 & 47 \\
\end{array}\]
DataCubes

`color`, `size`: DIMENSIONS
`count`: MEASURE

<table>
<thead>
<tr>
<th>size</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>20 3 5</td>
</tr>
<tr>
<td>M</td>
<td>3 3 8</td>
</tr>
<tr>
<td>L</td>
<td>0 0 5</td>
</tr>
<tr>
<td>TOT</td>
<td>23 6 18</td>
</tr>
</tbody>
</table>

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

DataCube issues:
- 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 them (and/or materialize portions on demand)
Q2: How to index them
Q3: Which operations to allow
DataCubes

DataCube issues:
Q1: How to store them (and/or materialize portions on demand) A: ROLAP/MOLAP
Q2: How to index them A: bitmap indices
Q3: Which operations to allow A: roll-up, drill down, slice, dice
[More details: book by Han+Kamber]

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>

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>

A3: Hybrid (H-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)

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

Q2: What operations should we support?

<table>
<thead>
<tr>
<th>Roll-up</th>
<th>C</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></td>
<td>28</td>
</tr>
<tr>
<td>Blue</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>TOT</td>
<td>23</td>
<td>6</td>
<td>18</td>
<td></td>
<td>47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drill-down</th>
<th>C</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></td>
<td>28</td>
</tr>
<tr>
<td>Blue</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>Gray</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>TOT</td>
<td>23</td>
<td>6</td>
<td>18</td>
<td></td>
<td>47</td>
</tr>
</tbody>
</table>
Data Cubes

Q2: What operations should we support?

Dice

<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>

Q3: How to index a dataCube?

A1: Bitmaps

<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>

Q2: What operations should we support?

- Roll-up
- Drill-down
- Slice
- Dice

Q3: How to index a dataCube?

A2: Join indices (see [Han+Kamber])
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>
<tr>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>····</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>····</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>····</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>?</td>
</tr>
</tbody>
</table>

Decision trees

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

- and we want to label ‘?’
  - num. attr#1 (eg., ‘age’)
  - num. attr#2
    - (eg., chol-level)

- so we build a decision tree:
  - num. attr#2
    - (eg., chol-level)
  - 40
  - num. attr#1 (eg., ‘age’)
  - 50
**Outline**

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

**Decision trees**

- so we build a decision tree:

```
+ Y age<50

N

+ Y chol <40

N

- ...
```

**Tree building**

- How?

```
+ + +

+ - -

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

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

**Decision trees**

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

**Tree building**

- How?
- Q1: how to introduce splits along attribute $A_i$
- Q2: how to evaluate a split?

**Tree building**

- skip

**Tree building**

- skip
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?

  `gini' index: $1 - p$

- use a greedy algo

Any other measure?

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

- A: by how close to uniform each subset is -
  i.e., we need a measure of uniformity:

- multiple split

(How about multiple labels?)

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

'gini' index: $1 - p^2 - p^2$
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.:

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

Tree building

- Before split: we need \( (n_+ + n_-) \cdot H(p_+, p_-) = (7+6) \cdot H(7/13, 6/13) \) bits total, to encode all the class labels
- After the split we need:
  - 0 bits for the first half and
  - \((2+6) \cdot H(2/8, 6/8)\) bits for the second half

Tree building

- What for?

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) [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) - in detail:
Tree pruning

- envision the problem as compression (of what?)

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

(MDL)

- a brilliant idea - eg.: best n-degree polynomial to compress these points:
- the one that minimizes (sum of errors + n)

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
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