An Amateur’s Introduction to Multirelational Data Mining

Raghu Ramakrishnan
Computer Sciences Department
University of Wisconsin at Madison

Caveat

◆ The title should be parsed as “An Intro to MRDM from an Amateur” rather than “An Intro to MRDM for Amateurs”

◆ Database-tinted glasses being worn!
Talk Outline

- A summary of my thoughts on MRDM/ILP
- Relational clustering as a case study
- Issues in DBMS integration
  - Along with some comments on “multi” table vs. single-table mining

MRDM MUSINGS
MRDM Accomplishments

- ILP origins, hypothesis discovery
- Classification
- Clustering
- Frequent itemsets
- Equational discovery
- Subgroup discovery
- Extensions of Bayesian nets to multiple relations via key-foreign key traversals

Positives

- Impressive! Quite a range of patterns/models are shown to be expressible in this formalism
  - Importantly, the added expressiveness allows new kinds of patterns to be naturally formulated by a user
- There is a (more or less) common computational structure consisting of
  - Space of patterns to search
  - Measure of support for a pattern
  - Enumeration and pruning strategy over search space

What tangible benefit can we derive from this generality?
Issues

- Can we indeed capture the semantics exactly for each of these classes of patterns/models?
  - Taking into account the details of the underlying evaluation algorithm!
- Is the performance comparable to specialized algorithms? Is it acceptable for a broad range of applications?

Challenges, Opportunities

- If ILP notation is roughly analogous to relational calculus, what is the appropriate algebra?
  - Equivalences, compositionality
  - Cost-based optimization to find “optimal” evaluation plans
- What kind of user input/domain knowledge can be used to focus computation, or help with optimization?
General Observations

- “Multirelational” seems to be a misplaced emphasis
  - Artifact of how “cases” are viewed in typical learning settings
  - If mining is seen as a generalization of querying, multiple tables are the norm; tables and views (over several tables) are interchangeable
- For enterprise class applications, important to integrate with database systems
  - Relational underpinnings a plus; should exploit this

CATEGORICAL CLUSTERING
Problem Statement

- **Goal:** Discover clusters of attribute-values
- **Data:** A table $T$ with attributes drawn from domains $D_1, \ldots, D_n$

$$\begin{array}{ccc}
A & B & C \\
\text{a}_1 & \text{b}_1 & \text{c}_1 \\
\text{a}_2 & \text{b}_2 & \text{c}_2 \\
\text{a}_3 & \text{b}_3 & \text{c}_3 \\
\text{a}_4 & \text{c}_4 \\
\end{array}$$

Note: We expect sizes of $D_1, \ldots, D_n$ to be small

- Thus, a tuple of $T$ consists of a value from each domain, e.g., $(a_1, b_2, c_1)$
- $T$ could be an arbitrary view over several tables!

STIRR (Gibson, Kleinberg, Raghavan, VLDB 98)

- **Intuition:** Want to detect that “Honda and Toyota are related because unusually high numbers of both were sold in August.”
  - If we also find that many Hondas and Nissans are sold in Sept, and many dealers sell both Hondas and Acuras, this leads to a cluster best described as "late-summer sales of Japanese cars"
- **Approach:** Techniques for spectral graph partitioning, generalized to hypergraphs.
  - Attr values as weighted vertices in a graph; edges based on co-occurrence. Weights propagate along links, leading to a non-linear dynamical system.
CACTUS (Ganti, Gehrke, Ramakrishnan, KDD 99)

- Same motivation, different problem formulation and approach
- Precise definition of cluster, deterministic algorithm that computes all clusters
- Very efficient, scalable, SQL-based algorithm

**Similarity Between Attributes**

- "similarity" between $a_i$ and $b_i$
  \[ \text{support}(a_i, b_i) = \text{number of tuples containing } (a_i, b_i) \]
- $a_i$ and $b_i$ are strongly connected if support($a_i, b_i$) is higher than expected
- \{a₁,a₂,a₃,a₄\} and \{b₁,b₂\} are strongly connected if all pairs are

[Diagram showing connected nodes and edges demonstrating similarity between attributes]
Similarity Within an Attribute

- $\text{sim}_A(b_1, b_2)$: Number of values of $A$ which are strongly connected with both $b_1$ and $b_2$

<table>
<thead>
<tr>
<th>sim*(B)</th>
<th>thru A</th>
<th>thru C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b_1, b_2)</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(b_1, b_3)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>(b_1, b_4)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b_2, b_3)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>(b_2, b_4)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Cluster Definition

- Region: A cross-product of sets of attribute values: $C_1 \times \ldots \times C_n$
- $C = C_1 \times \ldots \times C_n$ is a cluster iff
  1. $C_i$ and $C_j$ are strongly connected, for all $i, j$
  2. $C_i$ is maximal, for all $i$
  3. Support($C$) > expected

$C_i$: cluster projection of $C$ on $A_i$
The CACTUS Algorithm

- **Summarize**
  - Inter-attribute summaries: Scan dataset
  - Intra-attribute summaries: Query IA summaries
- **Clustering phase**
  - Compute cluster projections
  - Level-wise synthesis of cluster projections to form candidate clusters
- **Validation**
  - Requires a scan of the dataset

Inter-Attribute Summaries

- Supports of all strongly connected attribute value pairs from different attributes
  - Similar in nature to “frequent” 2-itemsets
  - So is the computation

\[
\begin{array}{c}
A \rightarrow B \rightarrow C \\
i_1 \rightarrow b_1 \rightarrow c_1 \\
i_2 \rightarrow b_2 \rightarrow c_2 \\
i_3 \rightarrow b_3 \rightarrow c_3 \\
i_4 \rightarrow b_4 \rightarrow c_4 \\
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
IJ(A,B) & IJ(A,C) & IJ(B,C) \\
\hline
(a_1,b_1) & (a_1,c_1) & (b_1,c_1) \\
(a_2,b_2) & (a_2,c_2) & (b_2,c_2) \\
(a_3,b_3) & (a_3,c_3) & (b_3,c_3) \\
\vdots & \vdots & \vdots \\
\hline
\end{array}
\]
**Intra-Attribute Summaries**

- \( \text{sim}_A(B) \): Similarities thru A of attribute value pairs of B

\[
\begin{align*}
\text{sim}_A(A,B,C) & \quad \text{thru A} \quad \text{thru C} \\
(b_1,b_2) & \quad 4 \quad 2 \\
(b_1,b_3) & \quad 0 \quad 2 \\
(b_1,b_4) & \quad 0 \quad 0 \\
(b_2,b_3) & \quad 0 \quad 2 \\
(b_2,b_4) & \quad 0 \quad 0 
\end{align*}
\]

**Computing Intra-Attribute Summaries**

- **SQL query to compute \( \text{sim}_A(B) \)**
  - Select T1.B, T2.B, count(*)
  - From IJ(A,B) as T1(A,B), IJ(A,B) as T2(A,B)
  - Where T1.B \( \neq \) T2.B and T1.A = T2.A
  - Group By T1.A, T2.A
  - Having count > 0;

- **Note**: Inter-attribute summaries are sufficient
  - Dataset is not accessed!
Memory Requirements for Summaries

- Attribute domains are small
  - Typically less than 100
  - E.g., the largest attribute value domain in the UC-Irvine collection is 100 (Pendigits dataset)
- 50 attributes, domain sizes 100, and 100 MB of main memory
  - Only one scan of the dataset for computing inter-attribute summaries

Clustering Phase

1. Compute cluster projections on each attribute
2. Join cluster projections across attributes—cluster generation

Identify the cluster projections:
\{a_1,a_2\}, \{b_1,b_2\}, \{c_1,c_2\};

Then form the cluster:
\{a_1,a_2\} \times \{b_1,b_2\} \times \{c_1,c_2\}
Computing Cluster Projections: Algorithm

- For the attribute $A_1$, compute cluster projections from clusters on $(A_1,A_2)$, $(A_1,A_3),..., (A_1,A_n)$: $S_1^2, ..., S_1^n$
  - NP-hard, but efficient under certain assumptions (under 10% of summarization cost)
- Intersection join on $S_1^2, ..., S_1^n$:
  
  $$S_1 = \{s: (\exists s' \in S_i^2 \ni s \subseteq s'),...,(\exists s' \in S_i^n \ni s \subseteq s')\}$$

- Lemma: $C = C_1 \times ... \times C_n$ be a cluster. Then $C_i$ is the intersection of $\{C_i': (C_i',C_k) \text{ is a cluster on } A_i, A_k\}$

Computing Cluster Projections: Example

From A, B: $S_A^B = \{a_1, a_2, a_3, a_4\}$

From A, C: $S_A^C = \{a_1, a_2\}$

$\{a_1, a_2\} = S_A^B \cap S_A^C$

$= \{a_1, a_2, a_3, a_4\} \cap \{a_1, a_2\}$
Cluster Generation

- Cluster projections $S_1, \ldots, S_n$ on $A_1, \ldots, A_n$
- Cross product $S_1 \times \ldots \times S_n$
- Level-wise synthesis: $S_1 \times S_2$, prune, then add $S_3$ and so on.
- May contain some dubious clusters!

![Diagram of cluster generation]

Experimental Evaluation

- Compare CACTUS with STIRR [GKR98]
- Synthetic datasets
  - Quasi-random data [GKR98:STIRR]
  - Fix domain of each attribute
  - Randomly generate tuples from these domains
  - Identify clusters and plant additional (5%) data within the clusters
Synthetic Datasets

{0,...,9} x {0,...,9}
{10,...,19} x {10,...,19}  

Both CACTUS and STIRR identified the two clusters exactly

Synthetic Dataset (contd.)

{0,...,9} x {0,...,9} x {0,...,9}
{10,...,19} x {10,...,19} x {10,...,19}
{0,...,9} x {10,...,19} x {10,...,19}

Cactus identifies the 3 clusters

STIRR returns:
{0,...,9} x {0,...,19} x {0,...,9}
{10,...,19} x {0,...,19} x {10,...,19}
Scalability with #Tuples

![Graph showing time vs. #Tuples]

CACTUS is 10 times faster.

Scalability with #Attributes

![Graph showing time vs. #Attributes]

CACTUS is significantly faster than STIRR.
Scalability with Domain Size

Bibliographic Data

- Database and theory bibliographic entries [Wie]—38500 entries
- Attributes: first author, second author, conference/journal, and year
- Example cluster projections on the conference attribute:

1. ACM Sigmod, VLDB, ACM TODS, ICDE, ACM Sigmod Record
2. ACMTG, CompGeom, FOCS, Geometry, ICALP, IPL, JCSS, ...
3. PODS, Algorithmica, FOCS, ICALP, INFCTRL, IPL, JCSS, ...
Connections to MRDM Work

- Applications of Koller et al.'s Bayesian approaches based on PRMs
  - Learn a PRM, add latent variables for clusters
  - Tuple-level clusters; apriori knowledge of number of clusters
- Neville and Jensen 2000
  - Iterative relational classification

ROCK [Guha, Rastogi, Shim, ICDE 99]

- Each tuple is a node, and two nodes are linked if within a threshold distance.
- Similarity between two nodes is the number of common neighbors.
- ROCK does agglomerative hierarchical clustering based on similarity.
Comments

- There may be people out there doing relational mining without realizing it!
- The same problem is attacked in very different ways by different researchers
  - Can these different approaches all be approximated in the MRDM/ILP framework?
  - If so, how efficient is this approximation?
  - What is the added expressive power of an MRDM formulation?
- Note that none of the DB-influenced methods really cares about the number of tables
  - The input is simply a collection of tuples; not “cases” that are somehow mapped to tuples

DBMS INTEGRATION
Why Integrate?

Data

- Copy
- Extract

Models

- Mine

Integration Objectives

- Avoid isolation of querying from mining
  - Difficult to do "ad-hoc" mining
- Provide simple programming approach to creating and using DM models
- Make it possible to add new models
- Make it possible to add new, scalable algorithms

Analysts (users) | DM Vendors
Microsoft SQL Server: Key Design Decisions

- Adopt relational data representation
  - A Data Mining Model (DMM) as a "tabular" object (externally; can be represented differently internally)
- Language-based interface
  - Extension of SQL
  - Standard syntax

DM Concepts to Support

- Representation of input (cases)
- Representation of models
- Specification of training step
- Specification of prediction step

Should be independent of specific algorithms
What are “Cases”?

- DM algorithms analyze “cases”
- The “case” is the entity being categorized and classified
- Examples
  - Customer credit risk analysis: Case = Customer
  - Product profitability analysis: Case = Product
  - Promotion success analysis: Case = Promotion
- Each case encapsulates all we know about the entity

Cases as Records: Example

<table>
<thead>
<tr>
<th>Cust ID</th>
<th>Age</th>
<th>Marital Status</th>
<th>Wealth</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>M</td>
<td>380,000</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>S</td>
<td>50,000</td>
</tr>
<tr>
<td>3</td>
<td>57</td>
<td>M</td>
<td>470,000</td>
</tr>
</tbody>
</table>
Types of Columns

<table>
<thead>
<tr>
<th>Cust ID</th>
<th>Age</th>
<th>Marital Status</th>
<th>Wealth</th>
<th>Product</th>
<th>Quantity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>M</td>
<td>880,000</td>
<td>TV</td>
<td>1</td>
<td>Appliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Coke</td>
<td>6</td>
<td>Drink</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ham</td>
<td>3</td>
<td>Food</td>
</tr>
</tbody>
</table>

- **Keys**: Columns that uniquely identify a case
- **Attributes**: Columns that describe a case
  - Value: A state associated with the attribute in a specific case
  - Attribute Property: Columns that describe an attribute
    - Unique for a specific attribute value (TV is always an appliance)
  - Attribute Modifier: Columns that represent additional "meta" information for an attribute
    - Weight of a case, Certainty of prediction

More on Columns

- **Properties describe attributes**
  - Can represent generalization hierarchy
- **Distribution information associated with attributes**
  - Discrete/Continuous
  - Nature of Continuous distributions
    - Normal, Log_Normal
  - Other Properties (e.g., ordered, not null)
Representing a DMM

◆ Specifying a Model
  - Columns it should predict
  - Algorithm to use
  - Special parameters
◆ Model is represented as a nested table
  - Specification = Create table
  - Training = Inserting data into the table
  - Predicting = Querying the table

CREATE MINING MODEL

CREATE MINING MODEL [Age Prediction]
(
[Customer ID] LONG KEY,
[Gender] TEXT DISCRETE ATTRIBUTE,
[Hair Color] TEXT DISCRETE ATTRIBUTE,
[Age] DOUBLE CONTINUOUS ATTRIBUTE PREDICT,
)
USING [Microsoft Decision Tree]
CREATE MINING MODEL

CREATE MINING MODEL [Age Prediction]

(  
  [Customer ID] LONG KEY,  
  [Gender] TEXT DISCRETE ATTRIBUTE,  
  [Age] DOUBLE CONTINUOUS ATTRIBUTE PREDICT,  
  [ProductPurchases] TABLE (  
    [ProductName] TEXT KEY,  
    [Quantity] DOUBLE NORMAL CONTINUOUS,  
    [ProductType] TEXT DISCRETE RELATED TO [ProductName]  
  )  
)

USING [Microsoft Decision Tree]

Note that the ProductPurchases column is a nested table. SQL Server computes this field when data is “inserted.”

Training a DMM

- Training a DMM requires passing it “known” cases
- Use an INSERT INTO in order to “insert” the data to the DMM
  - The DMM will usually not retain the inserted data
  - Instead it will analyze the given cases and build the DMM content (decision tree, segmentation model)

  » INSERT [INTO] <mining model name>  
    [(columns list)]  
    <source data query>
**INSERT INTO**

```sql
INSERT INTO [Age Prediction]
(
[Cust ID], [Gender], [Hair Color], [Age]
)
OPENQUERY([Provider=MSOLESQ..., 'SELECT
    [Cust ID], [Gender],
    [Hair Color], [Age]
FROM [Customers]'
)
```

**Executing Insert Into**

- The DMM is trained
  - The model can be retrained or incrementally refined
- Content (rules, trees, formulas) can be explored
- Prediction queries can be executed
What are Predictions?

- Predictions apply the trained model to estimate missing attributes in a data set
- Predictions = Queries
- Specification:
  - Input data set
  - A trained DMM (think of it as a truth table, with one row per combination of predictor-attribute values; this is only conceptual)
  - Binding (mapping) information between the input data and the DMM

Prediction Join

SELECT [Customers].[ID],
       MyDMM.[Hair Color],
       PredictProbability(MyDMM.[Hair Color])
FROM
    MyDMM PREDICTION JOIN [Customers]
ON    MyDMM.[Gender] = [Customers].[Gender] AND
    MyDMM.[Age] = [Customers].[Age]
Conclusions

- MRDM offers a nice conceptual framework encompassing a range of models and patterns.
  - Novel patterns are expressible
  - Potential for even more: Could it lead to algebraic formulations that facilitate cost-based optimization, and compositional mining?
  - For many applications, database integration is important, and needs to be though through.