Multi-Relational Data Mining

Outline [Dzeroski 2003] [Dzeroski & De Raedt 2003]

Introduction

Inductive Logic Programming (ILP)

Relational Association Rules

Relational Decision Trees

Relational Distance-Based Approaches

Advanced Methods
Introduction

The Single Table Assumption

• Existing data mining algorithms expect data in a single table
• But in reality, DBs consist of multiple tables
• Naive solution: join all tables into a single one (universal relation) and apply (single-relational) data mining algorithm

<table>
<thead>
<tr>
<th>Client#</th>
<th>Date</th>
<th>Item</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2765</td>
<td>02/25/2005</td>
<td>A</td>
<td>5</td>
</tr>
<tr>
<td>3417</td>
<td>02/26/2005</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>1005</td>
<td>02/26/2005</td>
<td>C</td>
<td>12</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>02/25/2005</td>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>3417</td>
<td>02/26/2005</td>
<td>B</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client#</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1005</td>
<td>Jones</td>
<td>35</td>
</tr>
<tr>
<td>1010</td>
<td>Smith</td>
<td>52</td>
</tr>
<tr>
<td>1054</td>
<td>King</td>
<td>27</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Introduction

**The Single Table Assumption**

- Universal relation

<table>
<thead>
<tr>
<th>Client#</th>
<th>Date</th>
<th>Item</th>
<th>Quantity</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1005</td>
<td>02/26/2005</td>
<td>C</td>
<td>12</td>
<td>Jones</td>
<td>35</td>
</tr>
<tr>
<td>1005</td>
<td>02/28/2005</td>
<td>B</td>
<td>2</td>
<td>Jones</td>
<td>35</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2765</td>
<td>02/25/2005</td>
<td>A</td>
<td>5</td>
<td>Bornman</td>
<td>23</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There are no more client entities!
What if rule depends on how many different items were purchased by a client?
Introduction

Aggregating Related Tables

- Enhancing „target table“ by aggregates of the related tuples in other tables

- Aggregation operators: COUNT, SUM, MIN, AVG, . . .

<table>
<thead>
<tr>
<th>Client#</th>
<th>Name</th>
<th>Age</th>
<th>Overall Quantity of Item A</th>
<th>Overall Quantity of Item B</th>
<th>. . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>1005</td>
<td>Jones</td>
<td>35</td>
<td>0</td>
<td>10</td>
<td>. . .</td>
</tr>
<tr>
<td>1010</td>
<td>Smith</td>
<td>52</td>
<td>35</td>
<td>0</td>
<td>. . .</td>
</tr>
<tr>
<td>. .</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>. . .</td>
</tr>
</tbody>
</table>

More meaningful! But what aggregates to consider? And what if attributes of the other clients that have purchased the same item are relevant?
Introduction

**Multi-Relational Data Mining**

- Data mining methods for multi-table databases
- Pattern search space much larger than for single tables
- Testing the validity of a pattern more expensive
- Similar data mining tasks
  - classification, clustering, association rules, . . .
  - . . . plus some tasks specific to the multi-relational case
- Single table (propositional) algorithms can be upgraded to multiple tables (first order predicate logic)
Inductive Logic Programming

Inductive Logic Programming (ILP)

- Goal: learn logic programs from example data
- Knowledge representation is expressive and understandable
- Examples: tuples from multiple tables
- Hypotheses: sets of rules
- Use of background knowledge
  also set of rules
Inductive Logic Programming

Logic Programs and Databases

- *Logic program*: set of clauses
- *Clause*: rule of the form "Head $\leftarrow$ Body"
  where Head / Body consist of atoms connected using the logical operators $\land$, $\lor$ and $\neg$
- *Atom*: predicate applied to some terms
- *Predicate*: boolean function with arguments (terms)
- *Term*: constant (e.g., mary), variable (e.g., X), function symbol applied to some term
Inductive Logic Programming

Logic Programs and Databases

- Example rule
  \[
  \text{father}(X,Y) \lor \text{mother}(X,Y) \leftarrow \text{parent}(X,Y)
  \]

- Definite clauses: exactly one atom in the head
  \[
  \text{parent}(X,Y) \leftarrow \text{father}(X,Y) \lor \text{mother}(X,Y)
  \]

- Horn clauses
  - One (positive) atom in the head, conjunction of body atoms
    \[
    \text{mother}(X,Y) \leftarrow \text{parent}(X,Y) \land \text{female}(Y)
    \]
Inductive Logic Programming

Classical Rule Induction Task

- Given:
  - set $P$ of examples from target relation (positive examples)
  - set $N$ of examples not from target relation (negative examples)
  - background predicates $B$
  - hypothesis (rule) language
- Find a set of rules that explains all positive and none of the negative examples

consistent and complete set of rules
Example

Training examples

daughter(mary, ann) +
daughter(eve, tom) +
daughter(tom, ann) -
daughter(eve, ann) -

Background knowledge

parent(ann, mary) female(ann)
parent(ann, tom) female(mary)
parent(tom, eve) female(eve)
pARENT(tom, ian)

Hypothesis language

definite clauses

Resulting rule

\[ daughter(X, Y) \leftarrow parent(Y, X) \land female(X) \]
The Sequential Covering Algorithm

Hypothesis (H) := {}  

Repeat  

  find a clause c that covers some positive and no negative examples;  
  add c to H;  
  delete all positive examples implied by c \( B \cup H \cup \{c\} \)  

Until no more (uncovered) positive examples

Construction of new clauses:  

  search of the space of clauses  
  applying some refinement operator
Inductive Logic Programming

Structuring the Space of Clauses

- Substitution \( \theta = \{V_1 / t_1, \ldots, V_n / t_n\} \)
  - assignment of terms \( t_i \) to variables \( V_i \)
- Clauses as sets of atoms (literals)
  \[ \text{Head} \leftarrow \text{Body} \iff \text{Head} \lor \neg \text{Body} \]
  - e.g., \( \text{daughter}(X, Y) \leftarrow \text{parent}(Y, X) \):
    \[ \{\text{daughter}(X, Y), \neg \text{parent}(Y, X)\} \]
- Clause \( c \theta - \text{subsumes} \) clause \( c' \)
  - if there exists a substitution \( \theta \) such that \( c\theta \subseteq c' \)
Inductive Logic Programming

Structuring the Space of Clauses

- Examples
  \[ c = \text{daughter}(X, Y) \leftarrow \text{parent}(Y, X) \]
  \[ \theta = \{ X / \text{mary}, Y / \text{ann} \} \]
  \[ c\theta = \text{daughter}(	ext{mary}, \text{ann}) \leftarrow \text{parent}(\text{ann}, \text{mary}) \]

  \[ c = \text{daughter}(X, Y) \leftarrow \text{parent}(Y, X) \]
  \[ c' = \text{daughter}(X, Y) \leftarrow \text{female}(X) \land \text{parent}(Y, X) \]
  \[ \theta = \{ \} \]
  \[ c\theta = c \subseteq c', \text{ i.e. } c \theta - \text{subsumes } c' \]
Inductive Logic Programming

**Structuring the Space of Clauses**

- Syntactic notion of generality
  
  clause \( c \) is *at least as general as* clause \( c' \) (\( c \leq c' \)) iff
  
  \( c \theta - \text{subsumes} \ c' \)

  \( c \) is *more general than* clause \( c' \) iff
  
  \( c \leq c' \land \neg(c' \leq c) \)

  \( c \) is a *generalization* of \( c' \), \( c' \) a *specialization* of \( c \)

  - If \( c \) does not cover an example, none of its specializations do
  - If \( c \) covers an example, all of its generalizations do
Inductive Logic Programming

Searching the Space of Clauses

- Top-down approach:
  start from most general clauses
  recursively apply refinement operators

- Refinement operator
  $\theta$ – subsumption-based
  returns all most general specializations of a given clause

- Types of refinements
  apply a substitution to a clause or
  add a literal to the body of the clause
Inductive Logic Programming

Example

daughter(X,Y) ←

daughter(X,Y) ← X=Y
daughter(X,Y) ← female(X)

dughter(X,Y) ← female(X), female(Y)

dughter(X,Y) ← parent(Y,X)

Refinement graph (lattice)
Inductive Logic Programming

Top-Down Search of Refinement Graphs

Hypothesis \( (H) := \{ \} \)

repeat

\[ \text{clause } c := p(X_1, \ldots, X_n) \leftarrow \]

repeat

build the set \( S \) of all refinements of \( c \);

\( c := \text{the best element of } S \) (according to some heuristic)

until stopping criterion satisfied (\( c \) is consistent with \( B \cup H \))

add \( c \) to \( H \);

delete all positive examples implied by \( c \) (using \( B \cup H \));

until no more (uncovered) positive examples (i.e., \( H \) complete)
Inductive Logic Programming

**FOIL** [Quinlan 1990]

- Top-down search of refinement graph
- Weighted information gain as heuristic to choose best clause
- Heuristic can be modified to allow clauses covering (some) negative examples
  - Handling of noisy data
- Declarative bias to reduce search space
  - Syntactic restrictions on clauses to be considered to be provided by the user
Inductive Logic Programming

Declarative Bias

• Argument types / domains (relational DBS)
• Input / output modes of arguments
  argument must / must not be instantiated when predicate added
• Parametrized language bias
  e.g., maximum number of variables, literals, . . . per clause
• Clause templates
  Ex.: \[ P(X,Y) \leftarrow Q(X,Z) \land R(Z,Y) \]
  where P, Q, R denote predicate variables
Declarative bias difficult to specify for user (syntactic!)
Inductive Logic Programming

**PROGOL** [Muggleton 1995]

- Refinement graph can be infinite
- Degree of graph typically very large
- Consider only clauses that cover a given example (pair)
- Bottom-up search of the refinement graph
  
  start with bottom clause $c$ covering a given positive example
  
  search the set of all clauses $\theta$-subsuming $c$

  heuristic to choose best such clause consistent with training examples

- Complete search
  
  slow, but does not miss potentially good clauses
Relational Association Rules

*Introduction* [Dehaspe and Toivonen 2001]

- Datalog queries: definite clause without function symbols
- Query: clause with empty head
  
  ```
  ?- person(X), parent(X,Y), hasPet(Y,Z)
  
  select Person.ID, Parent.KID, HasPet.AID
  from Person, Parent, HasPet
  where Person.ID = Parent.PID and Parent.KID = HasPet.PID
  ```

- Datalog queries viewed as multi-relational version of itemsets

  
  itemset \{person, parent, child, pet\}
  + constraints

  person = parent, person is parent of child, pet belongs to child
Relational Association Rules

Notions

- How to determine the frequency of a query?
- **Key**: a user-specified atom that needs to be present in each query
  
e.g. `?- person(X)` as key,
  
  `?- person(X), parent(X,Y), hasPet(Y,Z)` as query
- **Answer set** of a query: set of all substitutions making the query true
- **Absolute frequency** (**support**) of a query:
  
cardinality of answer set
- **Relative frequency** of a query:
  
  absolute frequency of query / absolute frequency of key
Relational Association Rules

Notions

• Query extension:

Given two frequent queries \( Q_1 = ?-l_1,\ldots,l_m \) and \( Q_2 = ?-l_1,\ldots,l_m,l_{m+1},\ldots,l_n \)
where \( Q_1 \theta \)-subsumes \( Q_2 \)

Query extension \( ?-l_1,\ldots,l_m \rightarrow ?-l_1,\ldots,l_m,l_{m+1},\ldots,l_n \)

• Example

\( Q_1 = ?- \person(X), parent(X,Y) \) absolute frequency 40
\( Q_2 = ?- \person(X), parent(X,Y), hasPet(Y,Z) \) absolute frequency 30
\( E = \person(X), parent(X,Y) \rightarrow hasPet(Y,Z) \)

support 30, confidence 30/40 = 75%

• Goal: finding all frequent query extensions of the given key
Relational Association Rules

*Method*

- **Declarative language bias:**
  
  key atom, e.g., key(person(-))
  
  input / output modes for predicates
  
  e.g., parent(+,-), hasPet(+,cat), hasPet(+,dog)

- **Level-wise search** of the pattern space
  
  similar to the Apriori-algorithm

- Start with the query \(?-\) key (level 1)

- At level \(l\), apply refinement operators to frequent queries obtained at level \(l-1\)
Relational Association Rules

Example

- Level 1: \( \text{person}(X) \)
- Level 2: \( \text{person}(X), \text{parent}(X,Y) \)
  \( \text{person}(X), \text{hasPet}(X,\text{cat}) \)
  \( \text{person}(X), \text{hasPet}(X,\text{dog}) \)
- Level 3: \( \text{person}(X), \text{parent}(X,Y), \text{parent}(Y,Z) \)
  \( \text{person}(X), \text{parent}(X,Y), \text{hasPet}(X,\text{cat}) \)
  \( \text{person}(X), \text{parent}(X,Y), \text{hasPet}(X,\text{dog}) \)
  \( \text{person}(X), \text{hasPet}(X,\text{cat}), \text{parent}(X,Y) \)
  \( \text{person}(X), \text{hasPet}(X,\text{cat}), \text{hasPet}(X,\text{dog}) \)
  \( \text{person}(X), \text{hasPet}(X,\text{dog}), \text{parent}(X,Y) \)
  \( \text{person}(X), \text{hasPet}(X,\text{dog}), \text{hasPet}(X,\text{cat}) \)
Relational Association Rules

Discussion

+ Finds frequent Datalog queries and query extensions
+ Generalization of different types of propositional association rules
  frequent itemsets, episodes, hierarchical itemsets, . . .
+ Declarative bias can significantly reduce the search space
- Declarative bias hard to specify by user
- Rule format quite restricted
  no consideration of aggregation operators
- Scalability for large databases?
  Integration with DBMS?
Relational Decision Trees

Introduction [Blockeel and De Raedt 1998]

- Construct decision tree, not flat set of rules
- Tests correspond to the evaluation of a predicate
- Decision tree is binary
  true / false
- Variables shared within a given subtree
- Relational decision tree can be converted into decision lists
  Ordered sets of clauses
Target predicate: maintenance (M,A)
where M denotes a machine and A the maintenance action to be taken
Relational Decision Trees

Example

Equivalent logic program

维持(M,send_back) ← haspart(M,X), worn(X), irreplacable(X)

维持(M,repair_in_house) ← haspart(M,X), worn(X)

维持(M,no_maintenance) ←

Depth-first traversal of decision tree

Order of clauses matters
Relational Decision Trees

Algorithm

• Major differences to propositional algorithm
  tests: predicates or substitutions
  test can consist of two atoms (one predicate and one substitution)
  possible tests context-sensitive

• Possible tests depend on the set of variables appearing in predicates
  along the path from the root to a given node

• Example:
  at node irreplacable(X), variables M and X have appeared

• One of the most efficient algorithms of Multi-Relational DM
Relational Distance-Based Approaches

Introduction [Kirsten, Wrobel and Horvath 2001]

• Many (propositional) data mining methods are distance-based
  k-nearest neighbor classification
  k-means / k-medoid clustering
  hierarchical clustering

• Standard distance functions $\text{dist}(X,Y)$ compare only attributes of objects (tuples) $X$ and $Y$

• Generalized distance function also compares attributes of tuples related to $X$ / $Y$

• Existing data mining methods can immediately be applied
Relational Distance-Based Approaches

The RIBL Distance Measure

- RIBL: Relational Instance Based Learning
- Propositional distance function
  \[ dist(x, y) = \frac{\sum_{i=1}^{d} \text{diff}(x_i, y_i)}{d} \]
  where \( \text{diff}(x_i, y_i) = \begin{cases} |x_i - y_i| & \text{for numerical attribute} \\ 0 & \text{for categorical attribute and } x_i = y_i \\ 1 & \text{for categorical attribute and } x_i \neq y_i \end{cases} \)
- RIBL distance measure
  depth 0: consider only attributes of \( x \) / \( y \)
  depth 1: consider also attributes of objects directly related to \( x \) / \( y \)
  depth 2: consider also attributes of objects directly related to depth 1 objects
Relational Distance-Based Approaches

Example

\begin{align*}
\text{member}(\text{person1}, 45, \text{male}, 20, \text{gold}) & \quad \text{p1} \\
\text{member}(\text{person2}, 30, \text{female}, 10, \text{platinum}) & \quad \text{p2} \\
\text{house}(\text{person1}, \text{murgle}, 1987, 560) & \quad \text{h1} \\
\text{house}(\text{person1}, \text{montecarlo}, 1990, 210) & \quad \text{h2} \\
\text{house}(\text{person2}, \text{murgle}, 1999, 430) & \quad \text{h3} \\
\text{district}(\text{montecarlo}, \text{famous}, \text{large}, \text{monaco}) & \quad \text{d1} \\
\text{district}(\text{murgle}, \text{famous}, \text{small}, \text{slovenia}) & \quad \text{d2}
\end{align*}

Depth 0: 
\[ \text{dist}(p_1, p_2) = \frac{1}{5} \cdot (\text{diff}(\text{person1}, \text{person2}) + \text{diff}(45, 30) + \text{diff}(\text{male}, \text{female}) + \text{diff}(20, 10) + \text{diff}(\text{gold}, \text{platinum})) \]
\[ = \frac{1}{5} \cdot (1 + (45/100) + 1 + (10/50) + 1) = 0.67 \]

Depth 1: 
\[ \text{diff}(\text{person1, person2}) = \text{diff(\{H_1\},\{H_2\})} \]
\[ = \min_{a \in H_1, b \in H_2} \text{diff}(a, b) \text{ where } H_1 = \{h_1, h_2\} \text{ and } H_2 = \{h_3\} \]

Depth 2: 
\[ \text{diff}(h_1, h_3) = \frac{1}{4} \cdot (1 + \text{diff}(d_1, d_2) + (12/100) + (130/1000)) \]
\[ \cdots \]
Relational Distance-Based Approaches

Discussion

+ RIBL distance also takes into account attributes of related objects
+ RIBL can be extended to deal with list-valued attributes etc.
+ Distance-based data mining methods immediately applicable
- RIBL measure is not metric
+ But metric measures have been proposed in literature
- Distance-based methods require a single distance function
  but each depth results in different distance values
- Distance measure relies on appropriate normalization of
  all attributes and appropriate aggregation operators
Multi-Relational Data Mining

*Advanced Methods*

- Probabilistic Relational Models
  - First-order generalization of Bayes Networks
  - Probabilistic model
  - Can deal with aggregations
  - Can be used for classification and clustering
  - [Taskar, Segal & Koller 2001]

- Improvements of efficiency
  - Survey [Blockeel, Sebag 2003]
  - Exploitation of DBMS technology [Yin, Han, Yang & Yu 2004]
    - Tuple-ID propagation
    - Materialize class histograms
Multi-Relational Data Mining

Advanced Methods

• Multi-relational clustering with user‘s guidance
  [Yin, Han & Yu 2005]
  Another kind of „semi-supervised“ clustering
  User specifies set of attributes relevant for cluster labels
  Algorithm adds other relevant attributes that are correlated to
  the user-provided attributes
Probabilistic Relational Models [Taskar, Segal & Koller 2001]

- Probabilistic Relational Model = PRM
- First-order generalization of Bayes Networks (BNs)
- Combines strengths of first-order logic (clean semantics) and probabilistic models (dealing with uncertainty)
- Can deal with aggregation over all referenced tuples in another table
- Can be used for classification and clustering
Advanced Methods

Bayesian Networks

- Directed graph
- *Nodes*: random variables
- *Edges*: direct probabilistic influences
- Probability distribution of a node is assumed to be independent from all non-descendants, given the direct predecessors, e.g. *XRay* conditionally independent of *Pneumonia* given *Infiltrates*
## Advanced Methods

### Bayesian Networks

- Associated with each node $X_i$ is a *conditional probability distribution* $P(X_i|Pa_i; \Theta)$ — distribution of $X_i$ for each assignment to parents.
- e.g., for Lung Infiltrates

| Pneu. | Tub. | $P(\text{Inf.} | \text{Pneu., Tub.})$ |
|-------|------|-------------------------------|
| T     | T    | 0.8                           |
| T     | F    | 0.6                           |
| F     | T    | 0.2                           |
| F     | F    | 0.01                          |

```
Pneumonia
Tuberculosis
Lung Infiltrates
XRay
Sputum Smear
```
Advanced Methods

Semantics

conditional independencies in BN structure + local probability models = full joint distribution over domain

Example:

\[ P(\bar{P}, T, I, X, \bar{S}) = P(\bar{P})P(T)P(I | \bar{P}, T)P(X | I)P(\bar{S} | T) \]

- Natural and compact representation
- If nodes have \( \leq k \) parents, then \( 2^k n \) instead of \( 2^n \) parameters
Bayesian Networks

- BNs can be learned from training data
- Structure learning
  - which nodes need to be connected via edges?
    - combinatorial search
- Parameter learning
  - determine conditional probability distributions for each node
    - numerical optimization
- Objective function: trade-off between number of edges (the fewer, the better) and likelihood of observed variable values (the higher the better)
Relational Database Schema

- **Patient**
  - Homeless
  - HIV-Result
  - Ethnicity
  - Disease-Site

- **Contact**
  - Contact-Type
  - Close-Contact
  - Transmitted
  - Age

- **Strain**
  - Unique
  - Infectivity

**Relationships**
- Infected with
- Interacted with

**Tables**
- **Patient**
- **Contact**
- **Strain**
Advanced Methods

**Probabilistic Relational Model**

Edges between nodes of same or different tables
Probabilistic Relational Model

PRM: (possible) relationships at the schema level

Relational skeleton: actual relationships at the instance (tuple) level
Advanced Methods

Probabilistic Relational Model

(part of the) corresponding BN
Advanced Methods

Aggregation

Patient
- POB
- Homeless
- HIV-Result
- Age
- Disease Site

Contact
- Contact-Type
- Close-Contact
- Age
- Transmitted

Contact #5077
- Contact-Type: coworker
- Close-Contact: no
- Age: middle-aged
- Transmitted: false

Contact #5076
- Contact-Type: spouse
- Close-Contact: yes
- Age: middle-aged
- Transmitted: true

Contact #5075
- Contact-Type: friend
- Close-Contact: no
- Age: middle-aged
- Transmitted: false

Mode

Patient

\[
\begin{array}{c|ccc}
A = y & A = m & A = o \\
y & 0.4 & 0.4 & 0.2 \\
m & 0.2 & 0.6 & 0.2 \\
o & 0.1 & 0.3 & 0.6 \\
\end{array}
\]
Advanced Methods

Semantics

\[ P(I \mid \sigma, S, \Theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A \mid parents_{s,\sigma}(x.A)) \]

probability distribution over instances I

PRM \quad S \quad + \quad \text{relational skeleton } \sigma \quad =
References


• Dzeroski S., De Raedt L.: “Multi-Relational Data Mining”, *ACM SIGKDD 2003 Tutorial*.


References

• Taskar B., Segal E., Koller D.: "Probabilistic Classification and Clustering in Relational Data", *Proc. IJCAI 2001*.


• Xiaoxin Yin, Jiawei Han, Philip Yu: "Cross-Relational Clustering with User's Guidance", *Proc. ACM SIGKDD 2005*. 