Inductive Databases: A personal view and overview

Luc De Raedt

Institut für Informatik
Albert-Ludwigs Universität Freiburg
Germany
Two Messages

• Data Mining Systems
  - MineRule: integrate SQL and association rules
  - LDL++: integrating deductive databases and mining
  - MolFea: Mining features in Molecules
  - MineSeqLog: Mining logical sequences

• Inductive databases
  - Integrate data mining with databases
  - Querying for patterns using constraints
  - Constraint programming analogy
Inductive databases

- Data mining
  - search for interesting and understandable patterns in data
- State-of-the-art in data mining ~ databases in the early days
- A theory of data mining is lacking
- View by Mannila and Iemielinski (CACM 96)
  - Make first class citizens out of patterns
  - Query not only the data but also the patterns
Inductive databases

• Many databases around
• The need to mine / analyze them
  - “understandable” patterns needed
  - Scientist/User wants control of mining process
  - Constraint based mining
    • Constraints specify patterns of interest
    • E.g. find all patterns that occur in at least 30 % the actives but in none of the inactives.
  - Mining becomes a querying process
Part I

Examples of (preliminary) inductive databases

MineRule
LDL++
MolFea
MineSeqLog
MINE RULE (1)

- A SQL-like operator on transactional DB

Table Purchase

<table>
<thead>
<tr>
<th>Tid</th>
<th>Customer</th>
<th>Item</th>
<th>Date</th>
<th>Price</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c1</td>
<td>ski-pants</td>
<td>12/1</td>
<td>55</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>c1</td>
<td>beer</td>
<td>12/1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>c2</td>
<td>shirts</td>
<td>12/1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>c2</td>
<td>jackets</td>
<td>12/1</td>
<td>115</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>c1</td>
<td>diapers</td>
<td>12/1</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
MINE RULE (2)

MINE RULE exemple as
SELECT DISTINCT 1..n Item as BODY, 1..1 Item as HEAD, SUPPORT, CONFIDENCE
WHERE HEAD.Item=`umbrellas`
FROM Purchase
GROUP BY Tid
HAVING COUNT(*)<6
EXTRACTING RULES WITH SUPPORT: 0.06,
CONFIDENCE: 0.9

E.g., jacket flight_Dublin $\Rightarrow$ umbrellas (0.08,0.93)
MINE RULE (3)

MINE RULE WordOfMouth as
SELECT DISTINCT 1..1 Customer as BODY,
1..n Customer as HEAD,
SUPPORT, CONFIDENCE
WHERE BODY.Date <= HEAD.Date
FROM Purchase
GROUP BY Item
EXTRACTING RULES WITH SUPPORT: 0.01,
CONFIDENCE: 0.9

E.g., c7 ⇒ c3 c12 (0.02,0.93)
MINE RULE

• Data pre-processing can be realized elegantly in SQL
• Rule post-processing facilities.
• Further details,
  - See Meo et al. DMKD 98
Molecular Feature Mining

• What?
  - Find fragments (substructures) of interest in sets of molecules

• Why?
  - Discover new knowledge
  - Use in predictive models
    • SAR (Structure Activity Relationship)
Molecules and Fragments

- 2D-structure
  - essentially Graphs
- Fragments
  - substructures
  - We: linear fragments
  - Sequence of atoms and bonds
- Linear fragments
  - 'o', 'c', 'cl', 'n', 's',... denote elements
  - '-'... single bond
  - '='... double bond
  - '#'... triple bond
  - ':'... aromatic bond
  - (hydrogens implicit)
- Smarts encoding

\[ \text{O-c:c:c:c-Cl} \]
Smiles encoding

- Smiles
  - Compact encoding of molecular structure
  - Used by computational chemists
  - Supported by many tools (e.g. Daylight)
  - Very compact!
  - Very efficient matching

\[ N - c1 : c : c : c (-O - c2 : c : c : c (-Cl) : c : c2) : c : c1 \]
Constraint-based Data Mining

• What?
  - Use constraints to specify which fragments/patterns are interesting
  - E.g. Frequency and syntax

• Why?
  - Declarative Querying
  - Interactive Process
  - Inductive database idea
Constraint-based data mining

• Generality
  - One fragment is more general than another one if it is a substructure of the other one
  - Notation: \( g \leq s \) (\( g \) is more general than \( s \); i.e. \( g \) will match a graph/string whenever \( s \) does)
  - Graphs: \( \sim \) subgraph relationship
  - Strings: substring / subsequence relationship
    • E.g. aabbcc is more general than ddaabbccee (substring)
    • E.g. abc is more general than aabbcc (subsequence)
  - (Item)sets: subset relation, e.g. \( \{a,b\} \) subset \( \{a,b,c\} \)
Strings Partial Order

\[ 
\varepsilon \\
 a \ b \\
aa \ ab \ ba \ bb \\
aaa \ aab \ aba \ baa \ abb \ bab \ bba \ bbb \\
\ldots 
\]
Primitives

- **Generality MolFea Symmetry!**
  - \( g \) is equivalent to \( s \) (syntactic variants) only when they are a reversal of one another
    - E.g., \( C-O-S' \) and \( S-O-C' \) denote the same substructure
  - \( g \) is more general than \( s \) if and only if \( g \) is a subsequence of \( s \) or \( g \) is a subsequence of the reversal of \( s \)
    - E.g., \( Cl-O-S' \leq Cl-O-S-c:c:c' \)
    - E.g., \( O-Cl' \leq Cl-O-S' \)

- **Frequency of a fragment \( f \) on a data set \( D \)**
  - The percentage of data points in \( D \) that \( f \) occurs in
  - E.g let \( f \) be aa and let \( D=\{abaa,acc,caa\} \); \( freq(f,D) = .66=2/3 \)
Primitive Constraints

- $f \leq P$, $P \leq f$, not ($f \leq P$) and not ($P \leq f$):
  - $f$ ... unknown target fragment,
  - $P$ ... a specific fragment
  - e.g. abbaa $\leq f$

- $\text{freq}(f, D)$
  - relative frequency of a fragment $f$ on a data set $D$

- $\text{freq}(f, D1) \geq t$, $\text{freq}(f, D2) \leq t$,
  - $t$ ... positive real number between 0 and 1
  - $D1$, $D2$ ... Data sets
  - e.g. $\text{freq}(f, Pos) \geq 0.20$
Example query

- Let $E_1 = \{aabbcc, abbc, bb\}$
- Let $E_2 = \{abc, bc, cc\}$
- $\text{Freq}(f, E_1) \geq 2$ and $\text{freq}(f, E_2) = 0$ and "a" < f
- Solutions: abb and abbc
Example Queries

- \( (\text{`N-O'} \leq f) \land (\text{freq}(f, \text{Act}) \geq 0.1) \land (\text{freq}(f, \text{Inact}) \leq 0.01) \)

- \( \text{not (F' } \leq f) \land \text{not (Cl' } \leq f) \land \text{not (Br' } \leq f) \land \text{not (I' } \leq f) \land (\text{freq}(f, \text{Act}) \geq 0.05) \land (\text{freq}(f, \text{Inact}) \leq 0.02) \)

- Queries are conjunctions of primitive constraints
Representing Solutions

• Traditional min. frequency constraint
  – Let \( c \) be \( \text{freq}(f, \text{Act}) \geq x \)
  – \( c \) satisfies Anti Monotonicity property
    • If we have a fragment \( g \leq s \),
    • Then if \( s \) is a solution then \( g \) is a solution as well
  – Imposes a lower border \( S \) on the space of solutions
A String Example

freq(f, D) △ 2 where D =

\[
\begin{align*}
\varepsilon & \\
A & B & C & D & F \\
AB & AC & BD & \\
ABC &
\end{align*}
\]

Consider \( E \)

\( E \) is not frequent,

Therefore no string containing \( E \) is frequent

Consider \( ABC \)

\( ABC \) is frequent

Therefore all substrings of \( ABC \) are frequent

Characterized by \( S = \{ ABC, BD, F \} \)
Another String Example

Let $f \leq ABD$

Characterized by $S = \{ABD\}$
Representing Solutions

• Traditional max frequency constraint
  – Let $c = \text{freq}(f, \text{Act}) < x$
  – $c$ satisfies Monotonicity property
    • If we have a fragment $g \leq s$,
    • Then if $g$ is a solution then $s$ is a solution as well
  – Imposes an upper border $G$ on the space of solutions
Consider “B” ≤ f and freq(f, D) ⊳ 2 with D = ABCDBDEF

Characterized by S = {ABC}

Characterized by S = {ABC, BD, F}

and G = {C}
Mitchell’s Version Space

• Consider now two constraints:

\[
\begin{align*}
    c_1 &= \text{freq}(f, D) \geq x \\
    c_2 &= \text{freq}(f, E) \leq y 
\end{align*}
\]

• We want to compute

\[
sol(c_1 \land c_2) = \{ f \mid \exists s \in S, g \in G : g \leq f \leq s \}
\]

where $S$ and $G$ are defined w.r.t. $c_1 \land c_2$
Mitchell’s Version Spaces

Too frequent w.r.t. $c_2$
Too general

Solutions

Infrequent w.r.t. $c_1$
Too specific

Is more general
## Constraints

<table>
<thead>
<tr>
<th>Anti-monotonic</th>
<th>Monotonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>$freq(f, D) \geq x$</td>
<td>$freq(f, D) \leq x$</td>
</tr>
<tr>
<td>$f \leq P$</td>
<td>$f \geq P$</td>
</tr>
<tr>
<td>$not(\ P \leq f \ )$</td>
<td>$not(\ P \geq f \ )$</td>
</tr>
</tbody>
</table>

### In ML

<table>
<thead>
<tr>
<th>$f \leq P$</th>
<th>$not(f \leq P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sim$</td>
<td>$\sim$</td>
</tr>
</tbody>
</table>

$P$ is a positive example

$P$ is a negative example
Computing Borders

• Borders completely characterize the set of solutions

• Combination of well-known algorithms
  - Level wise algorithm by Agrawal et al., Mannila and Toivonen
  - Mitchell’s and Mellish’s version space algorithms
The HIV Data Set

- Developmental Therapeutics Program’s AIDS Antiviral Screen Database (http://dtp.nci.nih.gov)
- One of the largest public domain databases of this type
- Measures protection of human CEM cells from HIV-1 infection using a soluble formazan assay
- We retained 41768 compounds (after pre-processing the whole data set of 43382 ones)
  - 40282 Confirmed Inactive
  - 1069 Confirmed Moderately Active
  - 417 Confirmed Active
AZT (Azidothymidine)

The majority of these fragments are derivatives of AZT.

Gives insight into the structural requirements for anti-HIV activity.

A rediscovery that proves the principle

Post-processing Combine fragments?
Use of Fragments : SAR

• Use as fingerprints/descriptors for SAR model building
• Feed data into your favorite data mining/statistical package
  - Neural Nets
  - Decision Trees
  - (Logistic) Regression
  - Support Vector Machines
  - Bayesian Methods
  - ...
Mining SeqLog

- SeqLog: simple “vanilla” datalog like language for structured sequences

User Modelling of Unix Users

latex(ilp, tex) xdvi(ilp, dvi) dvips(ilp, dvi) lpr(ilp, ps)

Secondary Structure of Proteins

hel(ProtID, HelixID, Wing?, HelixType, Length)
str(ProtID, StrandID, SheetID, Wing?, Orientation, Length)

Example protein:

hel('pdb1ssb', 1, 'A', h(right, alpha), 11)  hel('pdb1ssb', 2, 'A', h(right, alpha), 11)
str('pdb1ssb', 3, 'S1', 'A', 1, 0, 7)  hel('pdb1ssb', 4, 'A', h(right, alpha), 11)
str('pdb1ssb', 5, 'S1', 'A', 2, -1, 22)  str('pdb1ssb', 6, 'S1', 'A', 3, -1, 17).
SEQLOG Principles

A simple sequence is a possibly empty set of logical atoms

A sequence is a possibly empty set of logical atoms separated by operators
We currently use two operators:
- : directly follows
- <: follows (i.e. transitive closure of follows)

A sequential clause is a formula of the form $h \leftarrow s$ where $h$ is an atom and $s$ is a sequence
E.g. $latex2ps(X, tex) \leftarrow latex(X, tex) \leftarrow dvips(X, dvi)$

Sequential clauses can be recursive
They encode background knowledge
This implies e.g. $latex2ps(ilp, tex) \leftarrow lpr(ilp, ps)$

A SeqLog program consists of a set of sequential clauses and a set of sequences.
 SeqLog for Mining

• It is possible to define a notion of
  - Substring
    \[
    \text{xdvi}(\text{FileName}, \text{dvi}) \text{ dvips}(\text{FileName}, \text{dvi}) \text{ is a substring of}
    \]
    \[\text{latex}(\text{ilp}, \text{tex}) \text{ xdvi}(\text{ilp}, \text{dvi}) \text{ dvips}(\text{ilp}, \text{dvi}) \text{ lpr}(\text{ilp}, \text{ps})\]
  - Subsequence
    \[
    \text{xdvi}(\text{FileName}, \text{dvi}) < \text{lpr}(\text{FileName}, \text{ps}) \text{ is a subsequence of}
    \]
    \[\text{latex}(\text{ilp}, \text{tex}) \text{ xdvi}(\text{ilp}, \text{dvi}) \text{ dvips}(\text{ilp}, \text{dvi}) \text{ lpr}(\text{ilp}, \text{ps})\]
  - Resolution and Fixpoint

For SeqLog Programs
MineSeqLog

• Apply the idea of inductive databases to SeqLog, i.e. use constraints such as
  - Minimum and maximum frequency
  - Generality
    • Related to subsequence / substring matching
  - Background knowledge

To specify patterns (sequences) of interest
Other idea

- Use SeqLog as a dedicated representation language for data mining (a la Inductive Logic Programming)
- RDM (De Raedt ILP 00) is an earlier design for an IDB based on Warmr setting (was never fully implemented)
## Table Purchase

<table>
<thead>
<tr>
<th>Tid</th>
<th>Customer</th>
<th>Item</th>
<th>Date</th>
<th>Price</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c1</td>
<td>ski-pants</td>
<td>12/1</td>
<td>55</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>c1</td>
<td>beer</td>
<td>12/1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>c2</td>
<td>shirts</td>
<td>12/1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>c2</td>
<td>jackets</td>
<td>12/1</td>
<td>115</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>c1</td>
<td>diapers</td>
<td>12/1</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
LDL++

- Use LDL (deductive database language)
- Implement special « aggregate » primitives in LDL++ that can be used to implement data mining
- Various domains and tasks have been addressed

```
trans_size(T,count(I)) :- purchase(T,C,I,D,Q,P).
large_trans(T,S):- trans_size(T,S), S > 6.
```

In general

```
q(Z₁, ..., Zₖ, u_d_aggr<X₁, ..., Xₙ>) <- r(Y₁, ..., Yₘ).
p(X₁, ..., Xₙ, patterns<Y,m_s, m_c>) <- r(Z₁, ..., Zₘ).
```

computes

```
p(t₁, ..., tₙ,lhs,rhs,f,c) See Ph.D. G. Manco(2001)
Gianotta - Manco PKDD 99
```
Clustering

- Some work on constraint based clustering by
  - Jiawei Han et al.
  - Claire Cardie et al.

- Idea:
  - Cluster membership constraints
    - X and Y must belong to same cluster
    - X and Y must belong to different clusters
    - ...

Design issues

• Specification of the data part
  - Different data types
  - Pre-processing features

• Specification of the pattern part
  - Different pattern domains
  - Different constraints
  - Different mining tasks
  - Post-processing features

• The closure property
Part II

Inductive database principles

(a personal and logical view)
Inductive database principles

• What is an inductive database?
  - A set of data sets
  - A set of pattern sets

• IDB languages
  - A query language that generates data sets
  - An inductive query language that generates pattern sets

• Closure principle!

• A set and logic oriented view

• Not a universal framework, though quite general
Any monotonic or anti-monotonic constraint \( c \),
and any membership function (e.g. \( f \in P \))
is an atom.

An **inductive query** is a boolean formula over atoms.
E.g. \((f \in P) \text{ and } [\text{freq}(f, D1) > x \text{ or } \text{freq}(f, D2) < y] \text{ and } f < a b b b c c c c\)

The **query evaluation** problem
Given
an inductive database
an inductive query \( q \)
Find a characterisation of \( sol(q) \)
Manipulation

- create data set D as query
- create view data set D as query
- create pattern set P as query
- create pattern view P as query
- Insert / Delete / Update statements
create data set D4 as \{aa,ab,bb\}
create pattern view P2 as
  \text{freq}(f,D4) \geq 2
  
At this point : P2 = \{a,b\}
update data set D4 insert \{abc\}
  
Update P2 too : P2 = P2 U \{ab\}
  
• Incremental data mining !

• Insert \{ab\} into pattern view P2
• Pattern view update problem
An analogy with constraint programming

- Constraint logic programming integrates
  - LP + Constraint handling over domain
- IDB integrates
  - DB + Constraints over pattern domains
  - Pattern domains include: association rules, MolFea patterns, Seqlog, Warmr patterns, etc.

Cf. LDL++ examples
LDL++ has been used to implement/emulate many different pattern domains
Example Queries

?- Patt ≤ \{Item-1, Item-2, ..., Item-n\},
    frequency(Patt,'poslist') > 1000,
    frequency(Patt,'neglist') < 10 .

(covers at least 1000 pos / at most 10 neg)

could be used to define

pos(T) :- subset(Patt,T).
Example queries

?- Patt ≤ {Item-1, Item-2, ..., Item-n},
   frequency(Patt,[trans1]) = 1, seed
   frequency(Patt,[trans2])= 0 ,
   negative seed
   frequency(Patt,‘listofpositives’,P),
   frequency(Patt,‘listofnegs’,N),
   N < 10, Acc is P / (P + N), Acc > 0.9.

Predictive data mining
CDM

Constraint Based Data Mining can easily be extended to cover introduction new predicates a constraint data mining tool cf. LDL++, RDM
Same questions as in CLP pattern domain primitives semantics solvers
Memory organisation

- Consider
  - $q_1 : \text{freq}(f,D) > m$
  - $q_2 : \text{freq}(f,D \cup M) > m$ ($q_1 \models q_2$)
  - $q_3 : \text{freq}(f,D) > m \text{ OR } \text{freq}(f,M) > m$ ($q_3 \models q_2$)

- Scenario’s
  - $q_1$ answered and stored; $q_2$ asked
  - $q_2$ answered and stored; $q_1$ asked

- Keep track of subset relations among pattern sets / data sets
- Keep track of relations among patterns (generality - lattice structure) within given pattern set
- Operations on solution sets? On border sets?
Claim (subsumption)
Let $q_1$ and $q_2$ be two queries such that $q_1|=q_2$.
Then $sol(q_1) \subseteq sol(q_2)$

Background knowledge can also be used in this process.
E.g. $freq(f, D) > x$ and $x \geq y \rightarrow freq(f, D) > y$
E.g. $freq(f, D1) > x$ and $D1 \subseteq D2 \rightarrow freq(f, D2) > x$
E.g. $freq(f_2, D) > x$ and $f_1 \leq f_2 \rightarrow freq(f_1, D) > x$

Useful:
axioms about sets, generality, number theory

Subsumption is useful in the light of interactive querying and reuse of the results of previous queries
Query evaluation

• How to evaluate boolean inductive queries?
• Observe:
  - MolFea: conjunction of anti-monotonic and monotonic constraints
    • can be answered using level wise version space algorithm
    • solutions form a version space, can be represented by border sets.
Query Evaluation

**Theorem**
Let $q$ be an inductive query.
Then $sol(q)$ can be represented using a set of versionspaces
(a set of versionspaces represents the union of the versionspaces)

**Proof**
Write $q$ in Disjunctive Normal Form, i.e.
in the form of disjunction of conjunctions of the form $a_1 \land ... a_k \land m_1 \land ... m_n$
Each conjunction corresponds to a versionspace
$sol(q)$ can be represented using disjunctive versionspace (Cf. Gunther Sablon)
Divide and conquer approach

To evaluate/solve a query rewrite in DNF
for each conjunct in DNF
call level wise version space algo.
Claim
Let $q_1$ and $q_2$ be two queries that are logically equivalent. Then $sol(q_1) = sol(q_2)$

Using logical rewrites to optimize the mining process.
E.g. $(a_1 \lor a_2) \land (m_1 \lor m_2)$ is logically equivalent to

$$(a_1 \land m_1) \lor (a_2 \lor m_1) \lor (a_1 \land m_2) \lor (a_2 \lor m_2)$$

One versionspace versus the disjunction of four
Properties of inductive queries

\( AM \): the set of anti-monotonic queries
\( MO \): the set of monotone queries

We know that:
\[ \forall a \in AM : \neg a \in MO \]
\[ \forall m \in MO : \neg m \in AM \]
\[ \forall a_1, a_2 \in AM : a_1 \lor a_2 \in AM \]
\[ \forall a_1, a_2 \in AM : a_1 \land a_2 \in AM \]
\[ \forall m_1, m_2 \in MO : m_1 \lor m_2 \in MO \]
\[ \forall m_1, m_2 \in MO : m_1 \land m_2 \in MO \]
How many version spaces do we need?

Define

The class of queries $\Theta_k = \{ q \mid sol(q) \}$ can be represented as a union of $k$ version spaces but not as the union $k - 1$ versions spaces.

Open Question

How to decide the minimal $i$ for which $q \in \Theta_i$?

Theorem (DeRaedt, Jaeger, Mannila)

$\Theta_1 = \{ q \mid q \text{ can be written as } A_1 \wedge \ldots \wedge A_n \wedge M_1 \wedge \ldots \wedge M_m \}
\quad \text{where } A_i \text{ is a disjunction of anti-monotonic atoms}
\quad \text{where } M_j \text{ is a disjunction of monotonic atoms}$

$\Theta_k = \{ q \mid q \text{ can be written as the disjunction of } k \Theta_1 \text{ formulae} \}$

Theorem

Deciding minimal $k$ is co-NPcomplete
Operations on version spaces

- Logical operations on primitives have a set oriented counterpart?

- E.g. \((a_1 \lor a_2) \land (m_1 \lor m_2)\) is logically equivalent to
  \[(a_1 \land m_1) \lor (a_2 \lor m_1) \lor (a_1 \land m_2) \lor (a_2 \land m_2)\]

- What can we say about the corresponding operations on solution sets?
  - We assume solution sets are version spaces
  - Version spaces closed under intersection but not for union!
Let $sol(q_1)$ and $sol(q_2)$ be boundary set representable, i.e. representable using a versionspace.

Then in general $G(q_1 \lor q_2) \neq G(q_1) \lor G(q_2)$ and $S(q_1 \lor q_2) \neq S(q_1) \lor S(q_2)$

Counter Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>AC</td>
</tr>
<tr>
<td></td>
<td>CD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ABC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACD</td>
</tr>
</tbody>
</table>
A set and logic oriented view of inductive databases

• Key assumption:
  - Inductive queries are logical expressions over monotonic and anti-monotonic prims.
  - What about other primitives? Accuracy?

• Perspective:
  - Reasoning about query answering and optimisation
  - (first) elements of a theory given
  - Border set (Version space) representations useful
  - Operations on version spaces
  - A lot of opportunities for further work
Ongoing work

- String version space data structure
- Operations on string version spaces
- Efficient computation of string version space

- Elaborate theory and implementation

- Plenty of opportunities for interesting research (Ph.D.s)
Conclusions

• Inductive databases and constraint based mining
  - MineRule
  - LDL++
  - MolFea
  - MineSeqLog and RDM

• Solving inductive queries
  - Very general framework for query formulation
  - Problems of query evaluation and optimisation raised
Thanks
also to
Jean-Francois Boulicaut, Lee Sau Dan, Christoph Helma, Manfred Jaeger, Stefan Kramer, Heikki Mannila
and the cInQ project
• Announcements
  - Monday afternoon:
    • Long tutorial by Boulicaut and De Raedt on
      - Inductive databases
  - Tuesday
    • Full day workshop on
      - Knowledge Discovery with Inductive Databases
      - Talk by Fosca Gianotti
    • Workshop on EU Projects
      - Presentation on cInQ by Boulicaut