From Inductive Databases to Declarative Modeling for Data Mining

Lab for Declarative Languages and Artificial Intelligence

Joint work with especially
Tias Guns and Siegfried Nijssen
and the EU FET ICON project
and Jean-Francois Boulicaut and the former EU IQ project

(c) Luc De Raedt
We typically ...

1. Formalize learning / mining task
2. Design algorithm / technique to use
3. Implement the algorithm
4. Use and distribute the software
And do it again ...

**TASK 1**
- Input
- Specialized Algorithm 1
- Output

**TASK 2**
- Input
- Specialized Algorithm 2
- Output

**TASK 3**
- Input
- Specialized Algorithm 3
- Output
Problems

- Data mining is hard
  - focus on developing high-performance algorithms
  - only little re-use of existing solutions
  - many variations exist
  - little help with formalizing problems
- Common frameworks are missing (other than Weka, Orange, ...)
- A theory of data mining is missing (Mannila)
Two frameworks

• Is there a unifying framework, a practical theory, for data mining?

• Inductive databases (Iemielinski and Mannila 96)
  • exploit an analogy with databases
  • data mining as a querying process

• Declarative data mining
  • exploit an analogy with constraint programming and optimization
  • model data mining problems and solve them

• Caveat: this talk focusses on pattern mining
Inductive Databases
Cannot we simplify this ...?

1. Formalize learning / mining task
2. Design algorithm / technique to use
3. Implement the algorithm
4. Use and distribute the software
Inductive databases

- 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
  - Tightly integrate data mining and databases
The vision

• Supporting KDD processes by means of queries
  
  «There is no such thing as real discovery, just a matter of the expressive power of the query languages» Imielinski & Mannila, CACM Nov. 1996

• Make first class citizens out of patterns

• Examples queries
  
  • Give a decision tree that tests upon at most 5 attributes including blood pressure and sex, and that has accuracy at least 90 % on the training data
  
  • Give all fragments of molecules that appear in at least 20% of the actives, and in at most 1% of the inactives, and that do not contain a benzene ring.
A long-term perspective

• Why is the relational model so successful?
  • A general purpose query language with « nice » properties
    • simple theoretical foundations
    • declarative semantics
    • closure principle
  • The same is needed for KDD applications
  • The ultimate goal of IDBs is to find the equivalent of Codd’s relational database model for use in data mining
The Inductive Database framework

Extensional data
Intensional data
Intensional/extensional patterns

Data Mining System
Inductive database abstraction

- What is an inductive database? [De Raedt, 2002]
  - 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!
  - The result of a query should be a pattern set, a data set or a combination thereof
Abstraction

- Patterns domains specify
  - **Language of patterns** (e.g., itemsets, association rules, sequences, graphs, dependencies, decision trees, clusters)
  - **Evaluation functions** (e.g., frequency, closures, generality, validity, accuracy)
  - **Primitive constraints** (e.g., minimal and maximal frequency, freeness, syntactical constraints, minimal accuracy)
Mine Rule Example

- An 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>
</tbody>
</table>

Meo et al. 96
MINE RULE example 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 ⇒ umbrellas (0.02,0.93)
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 Example

- Advantages
  - Data selection by means of « full » SQL
  - Query evaluation can be effective

- Limitations
  - Dedicated to association rules
  - Poor possibilities for expressing background knowledge
  - No specific mechanism for rule post-processing (results are stored in relational tables)
DMQL Han & Kamber 2001 (m-k)

- A typical example of « syntactic sugar » for using many different data mining algorithms
- incorporate special primitives to couple with a decision tree learner, a clustering algorithm, etc...
- DMQL as an interface
- But what are the fundamental primitives?
- Where is the theoretical framework?
Mining views

• Blockeel et al., DAMI 2012
• Key concept = intensional patterns
• store patterns in virtual table and query
• Nice (theoretical) framework
Fig. 2. The PlayTennis data table and its corresponding Concepts table.

```
Day | Outlook | Temp | Humidity | Wind | Play
--- | ------- | ---- | -------- | ---- | ----
D1  | Sunny   | Hot  | High     | Weak | No
D2  | Sunny   | Hot  | High     | Strong| No
D3  | Overcast| Hot  | High     | Weak | Yes
D4  | Rain    | Mld  | High     | Weak | Yes
D5  | Rain    | Cool | Normal   | Weak | Yes
D6  | Rain    | Cool | Normal   | Strong| No
...

Fig. 3. Framework for an Inductive Database
```

```
(A) select R.rid, C1.*, C2.* from Sets S, Rules R, Concepts C1, Concepts C2
from Sets S, Rules R, where T.cid = C.cid
and C1.cid = R.cida and C2.cid = R.cidc and S.sz >= 30
and R.conf >= 80 and S.sz <= 4

(B) select T.treeid, C.* from Treescharac_Play D,
Trees_Play T, Concepts C
where T.cid = C.cid
and D.treeid = T.treeid and D.sz <= 5
and D.accc = (select max(acc)
from Treescharac_Play D1
where D1.sz <= 5)
```
A modern inductive database

Slides courtesy Dino Pedreschi and Fosca Gianotti (Pisa)
How do people get into town? What is the map of their trips in space and time?
A modern inductive query language

CREATE MODEL MilanODMatrix AS MINE ODMATRIX
FROM (SELECT t.id, t.trajectory FROM TrajectoryTable t),
(SELECT orig.id, orig.area FROM MunicipalityTable orig),
(SELECT dest.id, dest.area FROM MunicipalityTable dest)

CREATE RELATION CenterToNESuburbTrajectories USING ENTAIL
FROM (SELECT t.id, t.trajectory FROM TrajectoryTable t, MilanODMatrix m
WHERE m.origin = Milan AND m.destination IN (Monza, ..., Brugherio))

CREATE MODEL ClusteringTable AS MINE T-CLUSTERING
FROM (Select t.id, t.trajectory from CenterToNESuburbTrajectories t)
SET T-CLUSTERING_FUNCTION = ROUTE_SIMILARITY AND T-CLUSTERING_EPS = 400 AND T-CLUSTERING_MIN_PTS = 5

M-ATLAS -- Gianotti et al. VLDB J. 11

Mobility mining
Slide courtesy Pisa group
Inductive databases - perspective

- Put data mining on the same methodological grounds as databases
- Data mining as an interactive querying process
- Progress
  - theory -- mining views
  - practice -- interesting domain specific databases
- Limitations
  - no generally applicable IDBs today
  - primitives **built in** and integrated in DB,
  - **no user defined constraints**, search built-in...
  - **theory still weak?** (what about query optimization, expressivity, rich representations ... )
Declarative Modeling
The Challenge

Cannot we simplify this ... ?

1. Formalize learning / mining task
2. Design algorithm / technique to use
3. Implement the algorithm
4. Use and distribute the software
Key Claim

I am going to argue that POTENTIALLY WE CAN by adopting a Declarative Modeling paradigm

- Specify WHAT the problem IS
- Often as a constraint satisfaction or optimization problem

Instead of Procedural approaches

- Specify HOW the problem should be SOLVED
- Specify programs

Declarative Modeling has never been fully applied to ML/DM. Yet all the necessary ingredients are there.
The What, Why and How of Declarative Modeling
What is declarative modeling?

Model

array [1..9, 1..9] of var 1..9: sq;
predicate row_diff(int: r) =
    all_different (c in 1..9) (sq[r, c]);
predicate col_diff(int: c) =
    all_different (r in 1..9) (sq[r, c]);
predicate subgrid_diff(int: r, int: c) =
    all_different (i, j in 0..2) (sq[r + i, c + j]);

constraint forall (r in 1..9) (row_diff(r));
constraint forall (c in 1..9) (col_diff(c));
constraint forall (r, c in {1, 4, 7}) (subgrid_diff(r, c))
solve satisfy;

Zinc family of languages
How does it work?

MODEL specifies task = constraints + optimization criterion

Data = Input

State WHAT the problem is different SOLVERs possible
Why declarative modeling?

**DECLARATIVE**
- few lines of code
- easy to understand, maintain, change
- can be used with multiple “solvers”, e.g., exact and approximate
- formal verification possible

**PROCEDURAL**
- 1000s of lines of code
- hard to understand, maintain or change
- solver is built in the program

Here - CONSTRAINT PROGRAMMING
Also -- ANSWER SET PROGRAMMING
Constraint Programming

Given

- a set of variables \( V \)
- the domain \( D(x) \) of all variables \( x \) in \( V \)
- a set of constraints \( C \) on values these variables can take

Find an assignment of values to variables in \( V \) that satisfies all constraints in \( C \)

Zinc [Garcia de la Banda et al. CP 06]
Constraint Satisfaction

var P1, P2, P3, P4: {1,2};
constraint P1 != P2;
constraint P3 != P4;
constraint P1 != 1;
solve satisfy;
Solvers for CP

Two key ideas

• propagation of constraints, e.g., from

\[ D(P1) = \{1\} \text{ and } D(P2) = \{1,2,3\} \text{ and } P1 \neq P2 \text{ infer that } 1 \notin D(P2) \text{ and simplify } D(P2) = \{2,3\} \]

propagator: if \( D(x) = \{d\} \) and \( x \neq y \) then delete \( d \) from \( D(y) \)

• if you cannot propagate, instantiate (or divide) and recurse, e.g.,

\[ \text{call with } D(P2) = \{2\} \text{ and } \text{ with } D(P2) = \{3\} \]

\[ P2 = 2 \quad \text{ and } \quad P2 = 3 \]
Constraint Programming

There is a lot more to say

- about types of constraints and domains used
- about modeling languages
- about propagators -- how to modify domains
- about choosing the next variable to instantiate
- about implementations ...
- about their incorporation in programming languages ...
- about their performance ...
What about ML/DM?
Observation 1

Machine learning and data mining are essentially constraint satisfaction and optimization problems.
Data Mining

Given

• a database containing instances or transactions \( D \)
  the set of instances

• a hypothesis space or pattern language \( L \)

• a selection predicate, query or set of constraints \( Q \)

Find \( Th(Q,L,D) = \{ h \in L \mid Q(h,D) = \text{true} \} \)

[Mannila and Toivonen, 96]
Observation 1

Machine learning and data mining are essentially constraint satisfaction and optimization problems well-known in ML and DM.

good news
Observation 2

Use of solvers is very common in statistical learning (and SVMs)

• convex optimization and mathematical programming solvers

graphical models

• knowledge compilation packages and belief propagation

An important factor for their success
Observation 2

There has been a paradigm shift in the field of AI from programming to solving (Hector Geffner at ECAI 2012)

Today AI uses solvers for crisp computational problems
- SAT, ASP, CSP, CP, maxSAT, weighted model counting, ...
- many problems are reduced to these basic problems ... and solved efficiently

Still less common in other areas of DM/ML
Observation 3

There has been an enormous progress in solver technology for basic constraint satisfaction and optimization problems.

Solver technology facilitates the development of high-level declarative modeling languages:

- specify the **WHAT** -- not the **HOW**

Examples include:

- ZINC, Essence, Comet, OPL, FO(·), ...

Very flexible approach ...

*Still less common in DM/ML (except Matlab ?)*
Evidence
The case of Pattern mining
Pattern Mining

A. frequent pattern

- which patterns are frequent?

\[ Th(\mathcal{L}, Q, \mathcal{D}) = \{ p \in \mathcal{L} \mid Q(p, \mathcal{D}) = true \} \]

B. Correlated pattern mining = subgroup discovery

- which patterns are significant w.r.t. classes? all patterns? k-best patterns?

\[ Th(\mathcal{L}, Q, \mathcal{D}) = \arg_{p \in \mathcal{L}} \max_k \phi(p, \mathcal{D}) \]

C. pattern set mining

- which pattern set is the best concept-description for the actives? for the inactives?

\[ Th(\mathcal{L}, Q, \mathcal{D}) = \{ P \subseteq \mathcal{L} \mid Q(P, \mathcal{D}) = true \} \]
Pattern Mining

A. frequent pattern

- which patterns are frequent?

\[ \text{Th}(L, Q, D) = \{ \text{patterns}\} \]

B. Correlated pattern

- which pattern is significant w.r.t. classes

\[ \text{arg}_{\text{patterns}} \max_k \phi(p, D) \]

C. pattern set

- which pattern set is the best concept-description for the actives?

\[ \text{Th}(L, Q, D) = \{ P \subseteq L | Q(P, D) = \text{true}\} \]

We have been using off-the-shelf CP SOLVERS for these tasks, cf. Guns, Nijssen, De Raedt [AAAI 10, AIJ 11]

One solver for all of these

Easy to combine different constraints

Now looking at modeling level

KDD 08

KDD 09

IEEE TKDE 11
A. Frequent Pattern Mining
A. Frequent Itemset Mining

Given

- \( \mathcal{I} = \{1, \cdots, NrI\} \)
  set of items

- \( \mathcal{T} = \{1, \cdots, NrT\} \)
  set of transactions identifiers

- \( \mathcal{D} = \{(t, I) | t \in \mathcal{T}, I \subseteq \mathcal{I}\} \)
  Dataset

- \( Items \subseteq \mathcal{I} \) and \( Trans \subseteq \mathcal{T} \)

Find \( Items \) such that
\[ |\text{covers}(Items, \mathcal{D})| > \text{freq} \]

where \( \text{covers}(Items, \mathcal{D}) = \{t \in \mathcal{T} | (t, I) \in \mathcal{D} \text{ and } Items \subseteq I\} \)
A. Frequent Itemset Mining

Given

- \( \mathcal{I} = \{1, \cdots, NrI\} \)
  set of items
- \( \mathcal{T} = \{1, \cdots, NrT\} \)
  set of transactions identifiers
- \( \mathcal{D} = \{(t, I)| t \in \mathcal{T}, I \subseteq \mathcal{I}\} \)
  Dataset
- \( \text{Items} \subseteq \mathcal{I} \) and \( \text{Trans} \subseteq \mathcal{T} \)

Find \( \text{Items} \) such that

\[
|\text{covers(Items, D)}| > \text{freq}
\]

where \( \text{covers(Items, D)} = \{t \in \mathcal{T}|(t, I) \in \mathcal{D} \text{ and Items} \subseteq I\} \)
Frequent Itemset Mining

math like notation

user defined functions and constraints

solver independent (standardized)

efficiently solvable

```plaintext
int: Freq;
int: NrI;
int: NrT;

array[1..NrT] of set of 1..NrI: D;

var set of 1..NrI: Items;
var set of 1..NrT: Trans;

constraint card(Trans) > Freq;
constraint Trans = covers(Items, D);

solve satisfy;

function var set of int: cover(Items, D) =
  let {
    var set of int: Trans,
    constraint forall (t in ub(Trans))
      (t in Trans ↔ Items subset D[t])
  } in Trans;
```
Closed Itemset Mining

```
function var set of int: cover_inv(Trans,D) =
  let {
    var set of int: Items,
    constraint forall (i in ub(Items))
      (i in Items ↔ Trans subset D'[i])
  } in Items;

function var set of int: cover(Items, D) =
  let {
    var set of int: Trans,
    constraint forall (t in ub(Trans))
      (t in Trans ↔ Items subset D[t])
  } in Trans;
```

```
int: Freq;
int: NrI;
int: NrT;
array[1..NrT] of set of 1..NrI: D;

var set of 1..NrI: Items;
var set of 1..NrT: Trans;
constraint card(Trans) > Freq;
constraint Trans = covers(Items, D);
constraint Items = cover_inv(Trans, D);
solve satisfy;
```
Further Constraints

* exact coverage:
  t in Trans <-> Items subset D[t]

* freq:
  i in Items -> card(Trans intersect D'[i]) >= Freq

* maximal:
  i in Items <-> card(Trans intersect D'[i]) >= Freq

* closed:
  i in Items <-> Trans subset D'[i]

* delta-closed:
  i in Items <-> card(Trans intersect D'[i]) <= Delta * card(Trans)

easy to model
How does it work?

MODEL specifies task = constraints + optimization criterion

Data = Input

Only state WHAT the problem is
• CP based
• Map to standard Solvers offered by Zinc
• Like Gecode and Comet
  • Gecode -- sound and complete
  • Comet -- local search ...
• CHALLENGE
  • how to encode this efficiently?
Encoding in Zinc

int: Freq;
int: Nrl; int: NrT;
array [1..NrT] of set of int: D;

array [1..Nrl] of var bool: Items;
array [1..NrT] of var bool: Trans;

constraint % encode D: every Trans complement has no supported Items
  forall(t in 1..NrT) (  
    Trans[t] <-> sum(i in 1..Nrl) ( Items[i]*(1 - (i in D[t])) ) <= 0  
  );

constraint % frequency: every Item is supported by sufficiently many Trans
  forall(i in 1..Nrl) (  
    Items[i] -> sum(t in 1..NrT) ( Trans[t]*(i in D[t]) ) >= Freq  
  );

solve satisfy;

\forall t : T_t = 1 \iff \sum_i I_i(1 - D_{ti}) = 0

\sum_t T_t \geq \text{minsуп} \iff \forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq \text{minsуп}
Resulting Search Strategy akin to Zaki’s Eclat [KDD 97]

see Guns et al AIJ 11
Use a Data Mining System as solver
Results with LCM [Uno et al.] within Zinc
CHALLENGE
• how to recognize that DM system applies?
• possibly add post-processing ...
B. Correlated Pattern Mining
= Subgroup Discovery
= Discriminative patterns
Top-k Correlated Pattern Mining
Subgroup Discovery

- $\mathcal{D}$ now consists of two datasets, say $P$ and $N$
- a correlation function $\phi(p, \mathcal{D})$, e.g., $\chi^2$
- $Th(\mathcal{L}, Q, \mathcal{D}) = \arg_{p \in \mathcal{L}} \max_k \phi(p, \mathcal{D})$
Modeling perspective

Alternative opt. functions, for example:

\[
\text{solve maximize } \chi_2(\text{Trans}, \text{pos}, \text{neg});
\]

with:

\[
\text{function float: } \chi_2(\text{Trans}, \text{pos}, \text{neg})
\]
C. Pattern Set Mining
Pattern Sets

\[ Th(\mathcal{L}, Q, \mathcal{D}) = \{ P \subseteq \mathcal{L} \mid Q(P, \mathcal{D}) = true \} \]

One is not interested in all solutions to a pattern mining task, typically post-processing needed.

So, why not apply constraint based mining to pattern sets directly? [Zimmermann 09] [Guns et al, IEEE TKDE 11]
Pattern Sets

Consider a set of itemsets

\{\{a, b, c\}, \{b, d, e\}, \{c, e, f\}\}

Can be interpreted as DNF expression

\((a \land b \land c) \lor (b \land d \land e) \lor (c \land e \land f)\)

Useful for concept-learning and clustering

from local to global pattern mining
Pattern Sets

$$Th(\mathcal{L}, Q, \mathcal{D}) = \{P \subseteq \mathcal{L} | Q(P, \mathcal{D}) = true\}$$

What are meaningful constraints?

- local constraints on $$I \in P$$ such as $$freq(I, \mathcal{D}) \geq \text{minsup}$$
- constraints on all pairs of patterns $$I_1, I_2 \in P$$, e.g. $$|\text{covers}(I_1, \mathcal{D}) \cap \text{covers}(I_2, \mathcal{D})| \leq t$$
- global constraints $$freq(P, \mathcal{D}) \geq t'$$
- correlation, top-k, ...
k-Pattern Set Mining (|P|=k)

\[
\begin{align*}
\text{int: } & \text{ NrI; } \text{ int: } \text{ NrT; } \quad \text{int } K; \\
\text{array}[1..\text{NrT}] \text{ of set of int: } & \text{ D; } \\
\text{set of int: } & \text{ pos; set of int: } \text{ neg; } \\
\%
\quad \text{pattern set} \quad \\
\text{array}[1..K] \text{ of var set of 1..NrI: } & \text{ Items; } \\
\text{constraint } & \text{ lexleq(Items); } \quad \%	ext{ remove symmetries} \\
\%	ext{ every pattern is closed 'on the positives'} \\
\text{constraint} \quad & \text{let } \{ \text{ Dp = } [\text{D}[t] \mid t \text{ in pos}] \} \text{ in} \\
& \text{forall } (d \text{ in } 1..K) ( \\
& \quad \text{Items}[d] = \text{cover_inv(cover(Items}[d], \text{Dp}, \text{Dp)})}; \\
\%	ext{ accuracy of pattern set} \\
\text{solve maximize} \quad \\
& \text{let } \{ \text{ Trans = } \text{union}(d \text{ in } 1..K) (\text{cover(Items}[d], \text{D})) \} \text{ in} \\
& \text{card(Trans intersect pos) - card(Trans intersect neg); }
\end{align*}
\]
Generality

Can model instantiations/versions of:

- Concept learning (k-term DNF learning)
- Conceptual clustering
- k-Tiling
- Redescription mining
- ...
Pattern Mining

A. frequent pattern

- which patterns are frequent?

\[ \text{Th}(\mathcal{L}, Q, D) = \{ \} \]

B. Correlated pattern

- which patterns are significant w.r.t. classes?

\[ \text{Th}(\mathcal{L}, Q, D) = \{ \} \]

C. pattern set mining

- which pattern set is the best concept-description for the actives?

\[ \text{Th}(\mathcal{L}, Q, D) = \{ \} \]

We have been using off-the-shelf CP SOLVERS for these tasks, cf. Guns, Nijssen, De Raedt [AAAI 10, AIJ 11]

One solver for all of these tasks, cf. Guns, Nijssen, De Raedt [AAAI 10, AIJ 11]

Easy to combine different constraints

Now looking at modeling level
Constraint Programming for Itemset Mining (CP4IM) is a declarative approach to constraint-based itemset mining.

Instead of hand-crafting imperative algorithms, in constraint programming you declaratively specify a problem by means of the constraints it needs to satisfy. A generic solver will then effectively search for the solutions that satisfy the constraints.

In L. De Raedt, T. Guns, S. Nijsen. Constraint programming for itemset mining, KDD 2008, we have shown that this is a viable approach for many pattern mining problems such as frequent, closed, cost-based itemset mining and more. An accessible introduction is available in T. Guns, S. Nijsen, L. De Raedt. Itemset mining: A constraint programming perspective, Artificial Intelligence 175(12-13), 2011.

This website hosts an overview of our publications, the software we have developed as well as the datasets we used. We additionally aim to gather information on the use of constraint programming for pattern mining in general.

Contact Tias Guns with questions or comments regarding this website.

**Constraint-Based Mining**
Mining all (or top-k) itemsets that satisfy the constraints.

- L. De Raedt, T. Guns, S. Nijsen, Constraint programming for itemset mining, KDD 2008. [PDF](https://example.com), related [video](https://example.com)
  Presents FIMCP, the most flexible itemset mining framework to date. Some capabilities of the framework:
  - Different interestingness measures including frequent itemsets, discriminating itemsets and emerging itemsets.
  - Different condensed representations including closed itemset mining (formal concept learning), delta-closed and maximal itemset mining.
Perspective
All this is fine but...

what about

• efficiency and scalability ?

• other types of data and patterns (sequences, trees, graphs ...) ? relational

• other DM/ML tasks ? probabilistic, statistical learning, kernels / distances ...
Efficiency / Scalability

- Trade-off efficiency / generality
- Current experiments (with ONE solver)
  - Often a constant factor slower
  - Some cases much faster (correlated)
  - Avoid with specialized solvers [Nijssen and Guns, ECMLPKDD 10]
- Feature of Declarative Modeling
  - many solvers available (complete, approximate, ...)
  - one can even work with portfolio’s (Satzilla)
- Challenge is to build efficient solvers and translations

The new role of DM/ML scientists if we succeed?
Task / Representation

Rich representations ~ relational, graphs  

Task level ~ unsupervised, regression, clustering, probabilistic...  

Let us have a look at Statistical Relational Learning

Markov Logic [Domingos et al.]

ProbLog [De Raedt et al.]

...

kLog [Frasconi et al.]  kernel based
kLOG [Frasconi, Costa, DR, De Grave 12]

Diagram:
- **Input Database**
  - **Graphicalize**
- **Graphs**
- **Feature Vectors Kernel Matrix**
  - **Feature Generation Graph Kernel**
  - **Statistical Learner**
- **Classifier**
Surgical excision of CNV may allow stabilisation or improvement of vision.

Graphicalization
kLOG

- **Input Database**
- **Graphs**
- **Feature Vectors Kernel Matrix**
- **Classifier**

- **Graphicalize**
- **Feature Generation**
- **Graph Kernel**
- **Statistical Learner**
kLOG

Task MODEL specifies task = constraints + optimization criterion
CHALLENGE
Define the KERNEL declaratively
Define the LOSS function ... and SOLVE

As for many other ML/DM systems?
What if we succeed?
Our work today ...

We typically ...

1. Formalize learning / mining task
2. Design algorithm / technique to use
3. Implement the algorithm
4. Use and distribute the software

Specialized Algorithm

TASK
Input
Output
hard
Our work tomorrow ...

The user/application perspective...

1. Formalize learning / mining task
2. Model the problem easy
3. Select the right solvers
4. Use and distribute the software

More opportunities for re-use ...

A de facto standard language for DM / ML as Zinc?
Our work tomorrow ...

The scientist’s perspective...

Designing modeling languages
Studying task properties
Studying translations
Producing and adapting solvers

more fun

Larger impact of results in larger framework?
Conclusions

Declarative modeling languages for ML / DM has high potential
Conclusions

All the **necessary ingredients are available** to realize declarative modeling languages for ML/DM

- machine learning & data mining
- declarative modeling and constraint programming
- programming language technology
- should work for unsupervised, clustering ... as well

So let’s do it ...
Questions ?