Languages for Mining and Learning

Luc De Raedt
IJCAI, ML-track, 2015
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
This talk

• Survey + illustrate
  • programming languages for ML/DM
  • modelling languages for ML/DM (constraint programming and kernel perspective)
  • inductive query languages (database perspective)
Can we design programming languages containing machine learning primitives?

Can a new generation of computer programming languages directly support writing programs that learn?

Why not design a new computer programming language that supports writing programs in which some subroutines are hand-coded while others are specified as “to be learned.” Such a programming language could allow the programmer to declare the inputs and outputs of each “to be learned” subroutine, then select a learning algorithm from the primitives provided by the programming language.
Dynamics: Evolving Networks

• *Travian*: A massively multiplayer real-time strategy game

• Commercial game run by TravianGames GmbH

• ~3,000,000 players spread over different “worlds”

• ~25,000 players in one world

[Thon et al., MLJ 11, ECML 08]
Probabilistic (Logic) Programming

how does the world change over time?

one of the effects holds at time $T+1$

$$0.4::\text{conquest(Attacker,C,T+1)} \leftarrow \text{city(C,Owner,T)}, \text{city(C2,Attacker,T)}, \text{close(C,C2,T)}.$$  

if cause holds at time $T$
Probabilistic (Logic) Programming

0.3::stress(X) :- person(X).
0.2::influences(X,Y) :- person(X), person(Y).

smokes(X) :- stress(X).
smokes(X) :- friend(X,Y), influences(Y,X), smokes(Y).

0.4::asthma(X) :- smokes(X).

person(1).
friend(1,2).
person(2).
friend(2,1).
Probabilistic Programming

Key idea:

- modeling / programming language

- extend with probabilistic primitives to define probability distribution over possible worlds or execution traces

- extend with solvers / execution strategies to answer probabilistic queries (marginal, conditional probabilities, MAP and MPE)

- extend with learning strategies (EM or Bayesian inference) to parameters (structure)

Many other probabilistic programming languages including

- PRISM (Sato), Blog (Russell), Figaro (Pfeffer), Church (Goodmann), Factory (McCallumn), ...
Take Away Message

• What do languages for ML/DM bring us ?
  • expressive power
  • ease of modelling
  • separation of model and solving/execution

• a lot of interesting AI Challenges
Declarative modeling

Specify a wide range of data mining and machine learning problems.

Support high-level and natural modeling of pattern mining tasks; that is, the models should closely correspond to the definitions of data mining problems found in the literature; should support user-defined constraints and criteria such that existing problem formulations can be extended and modified and novel mining tasks can be specified.

Inspiration from Constraint Programming and Knowledge Representation. [Guns, Nijssen, DR]
Itemset mining

Data

Frequent patterns

4
2
3
Frequent Itemset Mining

Given

- $\mathcal{I} = \{1, \cdots, NrI\}$
  set of items
- $\mathcal{T} = \{1, \cdots, NrT\}$
  set of transactions
- $\mathcal{D} = \{(t, I) \mid t \in \mathcal{T}, I \subseteq \mathcal{I}\}$
  dataset
- $\mathcal{I} \subseteq \mathcal{I}$ and $\mathcal{T} \subseteq \mathcal{T}$

Find $\mathcal{I}$ such that

$|\text{covers}(\mathcal{I}, \mathcal{D})| > \text{freq}$

where $\text{covers}(\mathcal{I}, \mathcal{D}) = \{t \in \mathcal{T} \mid (t, I) \in \mathcal{D} \text{ and } \mathcal{I} \subseteq I\}$

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;

MiningZinc [Guns et al IJCAI 13]
Frequent Itemset Mining

math like notation

user defined functions and constraints

solver independent (standardized)

efficiently solvable

```
int: Freq;
int: Nrl;
int: NrT;

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

var set of 1..Nrl: 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;
```
Discriminative Pattern Mining

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

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

constraint Trans = cover(Items, D);

solve maximize
card(Trans intersect pos) – card(Trans intersect neg) neg);

Alternative opt. functions, for example:

solve maximize chi2(Trans, pos, neg);

with:

function float: chi2(Trans, pos, neg)
Declarative Modeling

Structured Data
much harder to model,
much harder to solve efficiently
Declarative Modeling

What about modeling graph kernels?

related to kLog [Frasconi, Journal Track IJCAI 15]
GIKs [Orsini, IJCAI 15]

Graph Invariant Kernels make an all versus all comparison of the vertices of two graphs G and G'. Vertices are similar if they are similar in both their attributes and structure.
Iemielinski and Mannila (1995)

The concept of data mining as a querying process.

- Make first class citizens of patterns.
- Query not only the data but also the patterns.
- Tightly integrate databases and data mining.
- Search for the equivalent of Codd’s relational algebra for data mining

“From the user perspective, there is no such thing as a real discovery, just a matter of the expressive power of the query language.”
An inductive database example
Virtual Mining Views (Blockeel et al. 12)

<table>
<thead>
<tr>
<th>Beer</th>
<th>Brand</th>
<th>Color</th>
<th>Alcohol%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Westmalle Tripel</td>
<td>Blonde</td>
<td>9.5</td>
</tr>
<tr>
<td>2</td>
<td>Orval</td>
<td>Dark</td>
<td>6.2</td>
</tr>
<tr>
<td>3</td>
<td>Straffe Hendrik</td>
<td>Gold</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Straffe Hendrik</td>
<td>Dark</td>
<td>11</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cid</th>
<th>Brand</th>
<th>Color</th>
<th>Alcohol%</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Westmalle Tripel</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>c2</td>
<td>Westmalle Tripel</td>
<td>Blonde</td>
<td>?</td>
</tr>
<tr>
<td>c3</td>
<td>?</td>
<td>Blonde</td>
<td>?</td>
</tr>
<tr>
<td>c4</td>
<td>?</td>
<td>Blonde</td>
<td>9.5</td>
</tr>
<tr>
<td>c5</td>
<td>Straffe Hendrik</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cid</th>
<th>frequency</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>c5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
An inductive database example
Virtual Mining Views (Blockeel et al. 12)

```sql
SELECT C.*, S.supp, S.sz,
S.supp * S.sz AS area
FROM BeerConcepts C, BeerSets S
WHERE (C.cid = S.cid AND (S.freq * S.sz > 60))
OR (S.freq > 10)
```

<table>
<thead>
<tr>
<th>Cid</th>
<th>Brand</th>
<th>Color</th>
<th>Alcohol%</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Westmalle Tripel</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>c2</td>
<td>Westmalle Tripel</td>
<td>Blonde</td>
<td>?</td>
</tr>
<tr>
<td>c3</td>
<td>?</td>
<td>Blonde</td>
<td>?</td>
</tr>
<tr>
<td>c4</td>
<td>?</td>
<td>Blonde</td>
<td>9.5</td>
</tr>
<tr>
<td>c5</td>
<td>Straffe Hendrik</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cid</th>
<th>frequency</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>c5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
AI Challenges

• Which primitives to choose?
• How to embed these in the underlying language?
  • integrate in the underlying execution mechanism, or
  • transform so that solver can be used
    • general purpose or specific?
Our work tomorrow ... 

We typically ...

1. Formalize learning / mining task

2. Specify the (declarative) model

4. Use and distribute the software
Our work tomorrow ...

We typically ...

1. Develop languages / primitives / compilers

2. Develop solvers

4. Use and distribute the software

ML Research

High potential for reuse / for standardization