Inductive Logic Programming

Joana Côrte-Real
jcr@dcc.fc.up.pt

CRACS & INESC TEC
Faculty of Sciences
University of Porto

University of Aizu
5th December 2014

Overview

Section 1

Introduction

Inductive vs Deductive Reasoning

Deductive Reasoning derives new rules or facts from a pre-defined set of rules (and other background knowledge).

Inductive Reasoning can learn a rule from examples and a set of facts which describe the example (or background knowledge)

Example of Induction and Deduction

Deduction

```
parent(X,Y):-mother(X,Y).
parent(X,Y):-father(X,Y).
```

$\cup$

```
mother(mary,anne).
mother(mary,john).
father(tom,anne).
father(tom,john).
```

$\models$

```
parent(mary,anne)
parent(mary,john)
parent(tom,anne)
parent(tom,john)
```

Induction

```
parent(mary,anne)
parent(mary,john)
parent(tom,anne)
parent(tom,john)
```

$\cup$

```
mother(mary,anne).
mother(mary,john).
father(tom,anne).
father(tom,john).
```

$\models$

```
parent(X,Y):-mother(X,Y).
parent(X,Y):-father(X,Y).
```

Main characteristics of ILP

Inductive **L**ogic **P**rogramming

- induces rules which explain examples and BK
- based on Logic Programming (Prolog)
- machine learning technique
- can be used for prediction and/or description
- also used to interface with experts of other areas of knowledge

Section 2

The train example

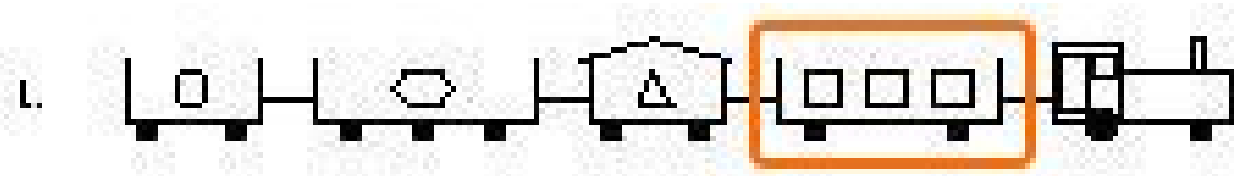
Introduction to **trains**

- **Michalski's trains** is a classic dataset in Machine Learning
- the short version is composed of 10 trains (5 *east*bound and 5 *west*bound)
- each train is composed of several cars, which can have different characteristics

AIM: find a classifier (Prolog rule) which indicates if a train is headed east, e.g.:

eastbound(T):-has_car(T, C), long(C).

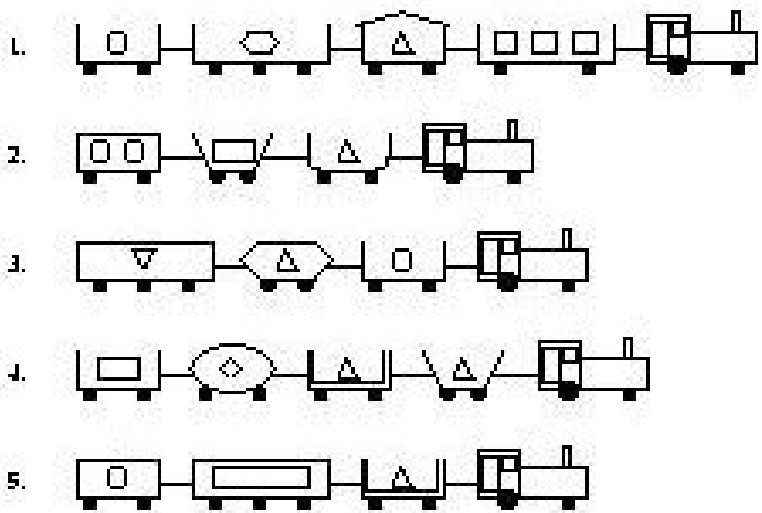
Car description



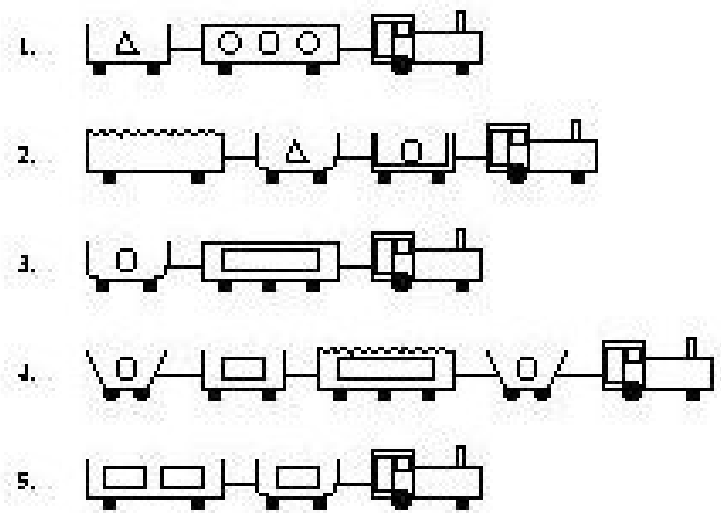
```
short(car_12).
closed(car_12).
long(car_11).
long(car_13).
short(car_14).
open_car(car_11).
open_car(car_13).
open_car(car_14).
shape(car_11,rectangle).
shape(car_12,rectangle).
shape(car_13,rectangle).
shape(car_14,rectangle).
load(car_11,rectangle,3).
load(car_12,triangle,1).
load(car_13,hexagon,1).
load(car_14,circle,1).
wheels(car_11,2).
wheels(car_12,2).
wheels(car_13,3).
wheels(car_14,2).
has_car(east1,car_11).
has_car(east1,car_12).
has_car(east1,car_13).
has_car(east1,car_14).
```

East and westbound trains

Positive examples
eastbound trains



Negative examples
westbound trains



Section 3

The ILP Algorithm

So what is ILP?

Composed of:

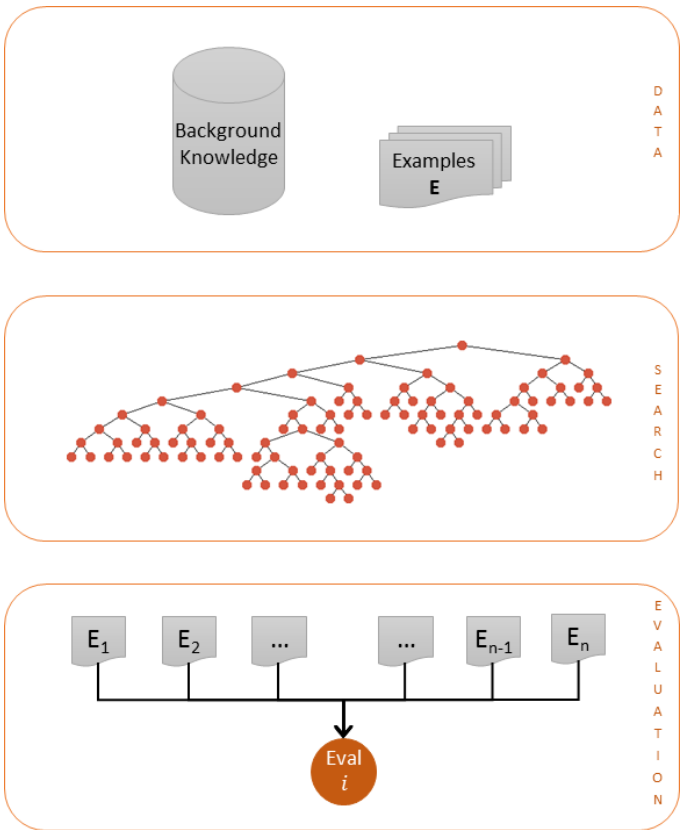
BK	facts, rules, etc.
Examples	positive and sometimes negative.
Search	method to explore possible rules.
Evaluation	criteria to choose the 'best' rule.

Generates theory which can *explain* both the BK and the positive examples together.
This theory should *not* explain negative examples, if they exist.

ILP Algorithm illustrated

There are three main parts in the ILP algorithm:

1. Load examples (+ and -) and background knowledge
2. Traverse search space of possible rules until stop criterion is met
3. Evaluate each rule according to evaluation criteria



A common approach

Use a greedy covering algorithm.

- Repeat while some positive examples remain uncovered (not entailed):
  1. Find a good clause (one that covers as many positive examples as possible but no/few negatives).
  2. Add that clause to the current theory, and remove the positive examples that it covers.

ILP algorithms use this approach but vary in their method for finding a good clause.

Rule evaluation

Generated rules are tested against examples.

Classic evaluation metrics:

Sensitivity = True Positives / Positives

Specificity = True Negatives / Negatives

Precision = True Positives / True and False Positives

Optimizations are used to avoid searching all syntactic space (language bias).

Rule generation

There are two classic ways to traverse search space:

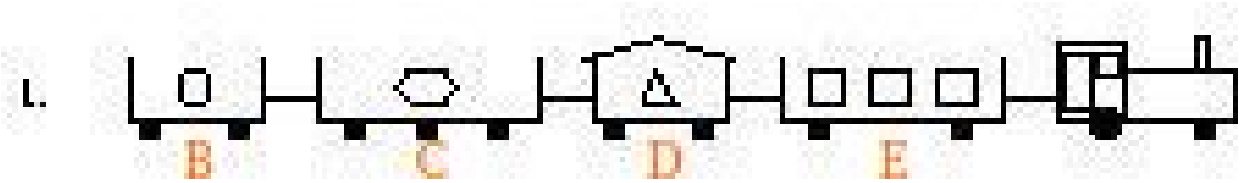
bottom-up start from a very specific clause (sometimes *saturated*) and generalize.

top-down start from the most general clause and specify.

The top-down algorithm will be applied to **trains** to illustrate rule generation.

Saturation

Saturation consists of picking a positive example and building the most *specific* possible rule from it by replacing constants with variables.




```
eastbound(A):-
    has_car(A,B),has_car(A,C),has_car(A,D),has_car(A,E),
    short(B),short(D),closed(D),long(C),long(E),
    open_car(B),open_car(C),open_car(E),
    shape(B,rectangle),shape(C,rectangle),
    shape(D,rectangle),shape(E,rectangle),
    wheels(B,2),wheels(C,3),wheels(D,2),wheels(E,2),
    load(B,circle,1),load(C,hexagon,1),
    load(D,triangle,1),load(E,rectangle,3).
```

Search - level 0

- Add literals to most general clause **eastbound(A):-true**.
- In this case, all literals containing **A** are possible choices for level 0 of the search.

```
eastbound(A):-
    has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
    short(B), short(D), closed(D), long(C), long(E),
    open_car(B), open_car(C), open_car(E),
    shape(B, rectangle), shape(C, rectangle),
    shape(D, rectangle), shape(E, rectangle),
    wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
    load(B, circle ,1), load(C, hexagon ,1),
    load(D, triangle ,1), load(E, rectangle ,3).
```

Search - level 1

- Add literals to each level 0 rule.
- Literals highlighted in yellow contain variables **A** and **B** and are the children of level 0 rule **eastbound(A):-has_car(A, B)**.
- Literals highlighted in blue contain variables **A** and **C** and are the children of level 0 rule **eastbound(A):-has_car(A, C)**.

```
eastbound(A):-
    has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
    short(B), short(D), closed(D), long(C), long(E),
    open_car(B), open_car(C), open_car(E),
    shape(B,rectangle), shape(C,rectangle),
    shape(D,rectangle), shape(E,rectangle),
    wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
    load(B,circle,1), load(C,hexagon,1),
    load(D,triangle,1), load(E,rectangle,3).
```

Search - and so on

- continue specifying hypothesis until most specific clause is generated or until stop criterion is met.
- note that the search space is a lattice.
- in order to decrease the search space, remove permutations (**eastbound(A):-has_car(A, B), has_car(A, C), short(B), long(C)** = **eastbound(A):-has_car(A, B), has_car(A, C), long(C), short(B)**).
- there are other ways to reduce the search space such as a language bias or user-defined pruning predicates.

Section 4

Conclusion

Summing up

Advantages of ILP:

- uses a language which is easy to interpret for experts from other areas of knowledge
- very concise classifiers
- great representative capacity: can represent relations and not only tables

BUT:

- can generate a very large search space
- corresponds to a discrete optimization problem
- generates non-probabilistic classification

Some ILP systems

- ALEPH[1] (top-down search): saturates first uncovered positive example, and then performs top-down admissible search of the lattice above this saturated example.
- GILPS[2] several searching engines, including bottom-up.
- and many others (FOIL, LINUS/DINUS, Tilde, Claudien, IndLog, etc)

[1] http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html
[2] http://www.doc.ic.ac.uk/ jcs06/GILPS/

Real-world application - Breast Cancer

This rule shows that a breast cancer case is malignant IF:

1. A is classified as BI-RADS 5 and had a mass present in a patient who was between the ages of 65 and 70 and had two prior mammograms (B, C)
2. prior mammogram (B) had no mass shape described and no punctate calcifications
3. prior mammogram (C) was classified as BI-RADS 3

```
is_malignant(A):-
    'BIRADS_category'(A,b5),
    'MassPAO'(A,present),
    'Age'(A,age6570),
    previous_finding(A,B),
    'MassesShape'(B,none),
    'Calc_Punctate'(B,
        notPresent),
    previous_finding(A,C),
    'BIRADS_category'(C,b3).
```

Thank you

Joana Côrte-Real