Logical & Relational Learning:

Introduction to Inductive Logic Programming (ILP)

Bernard ESPINASSE
Aix-Marseille Université (AMU), France.

20 of November 2017

- Introduction to Machine Learning
- Logical & Relational Learning
- Inductive Logic Programming (ILP)

References

- **Books, articles and reports :**
  - S. Dzeroski, J. Cussens, S. Manandhar, « An Introduction to Inductive Logic Programming and Learning Language in Logic », LLL'99, LNAI 1925, pp. 3–35, 2000.
  - S. Muggleton, J. Santos, and A. Tamaddoni-Nezhad, « ProGolem: A System Based on Relative Minimal Generalisation », ILP 2009, LNCS 5989, pp. 131–148, 2010.
  - Nédellec C., Rouveirol C., Adé H., Bergadano F. et Tausend B (1996). « Declarative Bias in ILP », *In Advances in Inductive Logic Programming*, p. 82-103, De Raedt L. (Ed.), IOS Press.
  - Muggleton S. (1991). Inductive Logic Programming. *New Generation Computing* 8 (4), 29.
  - De Raedt L. (2010). Inductive Logic Programming. *Encyclopedia of Machine Learning,* p. 529-537.
  - Saso Dzeroski, James Cussens, Suresh Manandhar, « An Introduction to Inductive Logic Programming and Learning Language in Logic »,
- **Courses/tutorials :**
  - Course of Krishnaprasad Thirunarayan (T.K.Prasad), Wright University, Dayton, Ohio (USA).
  - Cours de N. Lavrac, J. Stefan Institute, Ljbljan (Slovenia).

Outline

1. Introduction to Machine Learning

- Machine Learning Problem
- Supervised & Unsupervised ML
- Structure Learning Vs Parameters Learning
- Input and output representations in ML

Machine Learning: Definitions

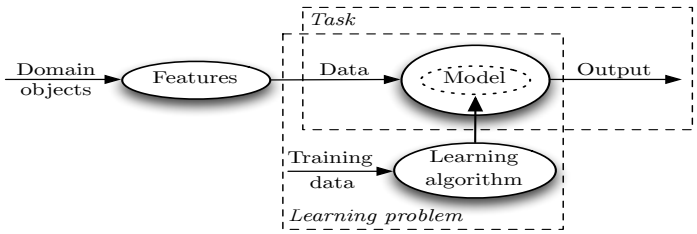
- **Machine Learning (ML) is:**
  - The process by which relatively permanent changes occur in behavioural potential as a result of experience. (*Anderson*)
  - Learning is constructing or modifying representations of what is being experienced. (*Michalski*)
  - A computer program is said to **learn** from experience ***E*** with respect to some class of tasks ***T*** and performance measure ***P***, if its performance at tasks in ***T***, as measured by ***P***, **improves** with experience ***E***. (*Mitchell*)
  - …

Supervised & Unsupervised ML

- **Supervised ML** *(naive Bayes classifiers, SVM, Kernels, ILP, …):*
  - task of inferring a *function* from a set of ***labeled*** *training examples* (x_i, $Classe_i$)
  - each example is a *pair* of an *input* (a vector) and a desired *output*
  - an supervised learning algorithm :
    - *analyzes* the training data and produces an inferred function which try to determine the class labels for training examples
    - *generalize* from the training data to unseen examples
    - according to a set of *assumptions (inductive bias)* to *predict* outputs given inputs that it has not encountered.
- **Unsupervised ML** *(clustering, neural networks, …)*
  - task of inferring a function to describe hidden structure from a set of ***unlabeled*** *training examples* (without desired output)
  - there is no objective evaluation of the accuracy of the structure that is output by the relevant algorithm

Supervised Machine Learning Problem

- Suppose we want that a machine learns from a set of *examples* from a certain domain, concerning specific *features* of this domain and *output* related,
- The ***task*** this machine have to do is to devise a *mapping* from these *features* to the *output (label of class)* this mapping is also called ***model***,
- A machine learning system can learn a *model* using a *learning algorithm* considering a *training dataset*, which consists of examples for which the *output* is already known:



"Machine learning is concerned with using the right features to build the right models that achieve the right tasks" [Flach, 2012]

Structure & Parameters Learning in ML

We distinguish 2 types of learning:

- **Parameters Learning**:
  - Given the structure (the rules) of this learning model ***M*** and we just want to infer the ***relevant parameters of M*** from a training set of examples *(Ex : parameters of a classifier or variables of a rule or a set of rules, …)*
- **Structure Learning**:
  - This learning consists to learn :
    - ***The structure of the learning model M*** from a training set of examples *(Ex : a classifier, a rule or a set of rules, …)*
    - ***The relevant parameters of M*** *(Ex : parameters of a classifier, variables of rule or a set of rules, …)*

Complexity of Structure learning > Complexity of Parameter learning

Input and Output Representations in ML

- **Input representation :**
  - The inputs in the learning process, training set examples, can be represented as:
    - A ***propositional*** *representation* (Ex : a vector, …)
    - A ***symbolic*** *representation* : (Ex : in First Order Logic, in Prolog, …)
- **Output representation :**
  - The outputs of the learning process, a model (structure + parameters) can be expressed in:
    - A ***numerical value:*** *probabilities* : Ex: P(Xi /Classi), *regressions*, …
    - A ***symbolic*** *representation* :
      - *a propositional rule (without variable), decision tree, …*
      - *a first order rule with variables (Prolog rules)*

Representations and Learning in ML

	Example representation	Learning model representation	Structure learning	Parameters learning
STATISTICAL LEARNING	table representations (Propositional)	Probability	NO	YES
DEEP LEARNING	table representations in hierarchical views	Probability	NO	YES
INDUCTIVE LOGIC PROGRAMMING - ILP (ALEPH, GILPS, ...)	Symbolic (First Order Logic)	Symbolic Rules (First Order Logic)	YES (Any kind of structure)	NO (Not necessary)
PROBABILISTIC INDUCTIVE LOGIC PROGRAMMING - PILP (KLog /KLogNLP)	Symbolic (First Order Logic)	Probability + (symbolic)	NO (probabilities+rules)	YES

2. Logical & Relational Learning

- **Forms of reasoning**
- **Deduction Vs Induction**
- **Principle of Logical and Relational Learning**

Forms of Reasoning

- **Deduction**: From causes to effect (*logic inference*)

  fact a, rule a => b

  INFER b (*First-order logic*)
- **Induction**: From correlated observations to rules (*Learning*)

  observe correlation between a1, b1, ... an, bn

  LEARN a -> b
- **Abduction**: From effects to possible causes (*Explanation*)

  rule a => b, observe b

  AN EXPLANATION a

*We will now consider only **Deduction** and **Induction**.*

Reasoning illustration: the “Rock-paper-scissors” game

(from Joana Corte-Real et al.)

The game:

Classical rules of the game:

- R1: The **rock** beats the **scissors** *(the rock crushes the scissors)*
- R2: The **scissors** beats the **paper** *(scissors cut the paper)*
- R3: The **paper** beats the **rock** *(the rock wrap the rock)*

Reasoning illustration: the “Rock-paper-scissors” game

(from Joana Corte-Real et al.)

Rules	beats (Round , PlayerA , PlayerB) :– plays (Round , PlayerA , rock) , plays (Round , PlayerB , scissors)
Facts (Background Knowledge - BK)	plays(1, ines , rock) plays(1, joana , scissors)
Examples	beats(1, ines , joana)

- **Rules** : rule(s) to win the game
- **Facts (BK)** : who and what each player plays at each round
- **Example** : who win at each round

Deduction illustration in “Rock-paper-scissors” game

- **Deductive Reasoning**: derives new rules or facts from a pre-defined set of rules (and other Background Knowledge):

Rules	beats (Round , PlayerA , PlayerB) :– plays (Round , PlayerA , rock) , plays (Round , PlayerB , scissors)
+	
Facts (BK)	plays(1, ines , rock) plays(1, joana , scissors)
=	
Examples	beats(1, ines , joana)

Induction illustration in “Rock-paper-scissors » game

- **Inductive Reasoning** : can learn a rule from examples and a set of facts which describe the example (or Background Knowledge - BK)

Facts (BK)	plays(1, ines , rock) plays(1, joana , scissors)
+	
Examples	beats(1, ines , joana)
=	
Rules	beats (Round , PlayerA , PlayerB) :– plays (Round , PlayerA , rock) , plays (Round , PlayerB , scissors)

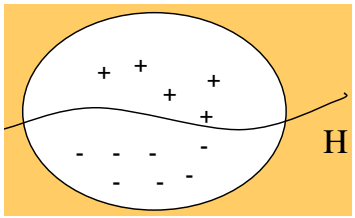
Predictive versus Descriptive Induction

(Source Lavrac)

Predictive induction: Inducing classifiers, aimed at solving classification/prediction tasks

- *Classification rule learning, Decision tree, learning, ...*
- *Bayesian classifier, ANN, SVM, …*

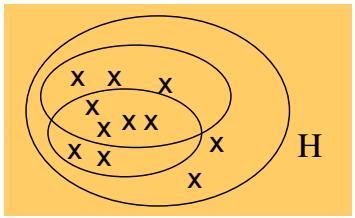
-> Data analysis through hypothesis generation and testing



Descriptive induction: Discovering regularities, uncovering patterns, aimed at solving KDD tasks

- *Symbolic clustering, Association rule learning, Subgroup discovery, ...*

-> Exploratory data analysis



Logical and Relational Learning

Goal: to find a hypothesis h, i.e., a logic program, from a set of positive & negative examples:

- **Given:**
  - a **set of training examples *T*** expressed in a language chosen for representing the examples L_E,
  - a **background knowledge B**,
  - a **hypothesis language L_H** that specifies the clauses that are allowed in the hypotheses set ***H***,
  - a **relation *covers(e, H, B)*** which determines the classification of an example *e* with respect to *H* and *B*,
- **Find** a **hypothesis $h \in H$** that :
  - **covers all positive training examples** and
  - **none of the negative ones**

  with respect to background theory B.

Logical and Relational Learning: learning settings

- Specific **learning setting** is determined by L_E language together with the **covers** relation [De Raedt, 1997]
- Most **popular learning settings** are:
  - **Learning from entailment [Plotkin 1970]: the examples are definite clauses :**

    - An hypothesis h covers an example e with respect to the background knowledge B ***if and only if*** $B \cup H \models e$

    - An example can consist of just a single fact.

  - **Learning from interpretations [De Raedt and Dzeroski, 1994]: the examples are Herbrand interpretations:**

    - An hypothesis h covers an example e with respect to the background knowledge B ***if and only if e is a model of*** $B \cup H$

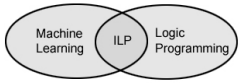
    - All facts that hold in the example are known, more information is available to the learner.

3. Inductive Logic Programming (ILP)

- Definition of Inductive Logic programming
- Predictive ILP and Descriptive ILP
- ILP interests: multiples relations and structured data
- Rule learning in ILP: global process
- ILP Systems Strategies for Hypothesis Search
- ILP Applications

Inductive Logic Programming – ILP

- **ILP is a technique related to Logical and Relational Learning:**
  - At the intersect of **Machine Learning** & **Logic Programming** domains



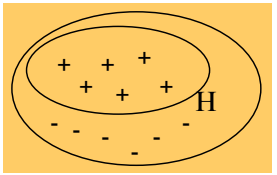
  - Which **learns logic rules from examples and background knowledge (BK)**

    Ex : learn the rule for grand parents, given background knowledge of parents and examples of grandparents

  - **Induces rules** which **explain examples and BK**
  - based on **Logic Programming** (Prolog)
- **ILP can be used for :**
  - ***Classification*** and ***Prediction***
  - to ***interface with experts*** of other areas of knowledge

Predictive ILP: Classification (1)

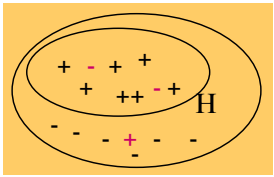
(Source N. Lavrac)



- **Given :**
  - A set of observations:
    - a set of ***positive examples E+***
    - a set of ***negative examples E-***
  - A ***Background Knowledge B***
  - An hypothesis language L_H
  - A ***covers relation***
- **Find an *hypothesis H* $\in$ L_H** such that (given B) H covers ALL positive and NO negative examples
- ***In logic,*** find H such that:
  - $\forall e \in E+ : B \cup H \models e$ **(H is complete)**
  - $\forall e \in E- : B \cup H \not\models e$ **(H is consistent)**
- ***In ILP***, E are ground fact, B and H are (set of) ***definite clauses***.

Predictive ILP: Classification (2)

(Source N. Lavrac)

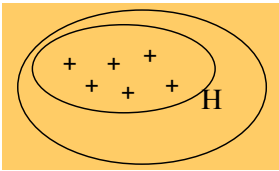


- **Given :**
  - A set of observations:
    - a set of ***positive examples E+***
    - a set of ***negative examples E-***
  - A ***Background Knowledge B***
  - An hypothesis language L_H
  - A *covers relation*
  - **A *quality criterion***
- **Find an *hypothesis H* $\in$ L_H** such that (given B) H is optimal w.r.t. some quality criterion : max. predictive accuracy A(H)

  (instead find a hypothesis H $\in$ L_H such that (given B) H covers ALL positive and NO negative examples)

Descriptive ILP: Discovery

(Source Lavrac)



- **Given :**
  - A set of observations:
    - a set of ***positive examples E+***
  - A ***Background Knowledge B***
  - An hypothesis language L_H
  - A ***covers relation***
- **Find : Maximally specific *hypothesis H* $\in$ L_H** such that (given B) H covers ALL positive examples
- ***In logic,*** find *H* such that $\forall c \in H$, *c is true in some preferred model of B $\cup$ E (e.g. least Herbrand model M (B $\cup$ E))*
- ***In ILP***, E are ground fact, B and H are (set of) ***general clauses***.

A Sample Problem: Learning of Family Relations (1)

(Source Lavrac)

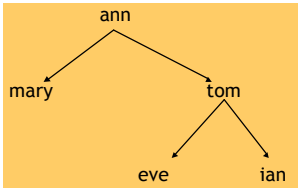
Observations:

E + = {daughter(mary,ann), daughter(eve,tom)}

E - = {daughter(tom,ann), daughter(eve,ann)}

Background Knowledge:

B = {mother(ann,mary), mother(ann,tom), father(tom,eve), father(tom,ian),
female(ann), female(mary), female(eve), male(pat), male(tom),
parent(X,Y) ← mother(X,Y),
parent(X,Y) ← father(X,Y)}



A Sample Problem: Learning of Family Relations (2)

(Source Lavrac)

E + = {daughter(mary,ann), daughter(eve,tom)}
E - = {daughter(tom,ann), daughter(eve,ann)}
B = {mother(ann,mary), mother(ann,tom), father(tom,eve), father(tom,ian),
female(ann), female(mary), female(eve), male(pat), male(tom),
parent(X,Y) ← mother(X,Y), parent(X,Y) ← father(X,Y)}

Predictive ILP:

induce a definite clause:

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

or a set of definite clauses:

daughter(X,Y) ← female(X), mother(Y,X)
daughter(X,Y) ← female(X), father(Y,X)

Descriptive ILP: *induce a set of (general) clauses:*

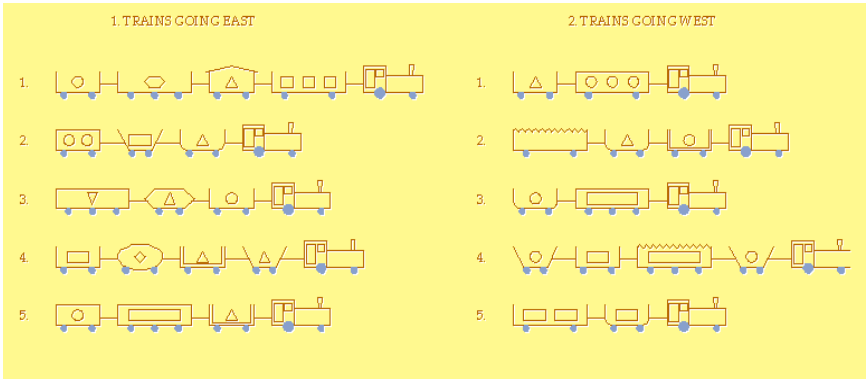
← daughter(X,Y), mother(X,Y)
female(X) ← daughter(X,Y)
mother(X,Y)
father(X,Y) ← parent(X,Y)

ILP interest: Multiples relations

- **Most ML techniques cannot use more than one relation:**
  - e.g., decision trees, neural networks, ...
- **ILP technique permit to use multiple relations:**
  - Ex:
    - Given known relations :
      father(Old,Young) and *mother(Old,Young)*
      male(Somebody) and *female(Somebody)*
    - ILP can learn new relations :
      parent(X,Y) :- father(X,Y)
      parent(X,Y) :- mother(X,Y)
      brother(X,Y) :- male(X), father(Z,X), father(Z,Y).

ILP interest: Structured Data (1)

- Example of East-West trains (Michalski) :



Question : What makes a train to go eastward ?

ILP interest: Structured Data (2)

- **Structured data = multiple relations:**

has_car relation

Train	Car
t1	c11
t1	c12
t1	c13
t1	c14
t2	c21
…	…

and

car_properties relations

Car	Length	Shape	Axle	Roof	...
c11	short	rectangle	2	none	…
c12	long	rectangle	3	none	…
c13	short	rectangle	2	peaked	…
c14	long	rectangle	2	none	…
c21	short	rectangle	2	flat	…
…	…	…	…	…	…

ILP interest: Structured Data (3)

Induction of a classifier for the East-West trains example :

- **BK**:
  - relation ***has_car***
    Ex : has_car(t1, c11), …
  - relation ***car_properties*** *(length, roof, shape, axle, roof, …)*
    Ex : lenght(c11, short), …
- **Examples**:
  the trains t1 to t10
- **Classes**:
  east, west
- Possible **Hypothesis (Theory):**
  east(T) :- has_car(T,C), length(C,short), roof(C,_)

Rule learning in ILP: global process (1)

Problem: learn *grandparent* rule:

- **Background knowledge *BK* :**
  parent_of(charles, george)
  parent_of(george, diana)
  parent_of(bob, harry)
  parent_of(harry, elizabeth).
- **Positive examples E+ :**
  grandparent_of(charles, diana)
  grandparent_of(bob, elizabeth).
- **Generate hypothesis H :**
  grandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,Y).

Rule Learning in ILP: global process (2)

How to come up with a rule for *grandparent_of(X,Y)?*

1. Take the example *grandparent_of(bob,elizabeth).*
2. Find the **subset of Background Knowledge (BK) relevant** to this example:
   parent_of(bob,harry)
   parent_of(harry,elizabeth) .
3. **Form a rule** from these facts :
   grandparent_of(bob,elizabeth) :-
   parent_of(bob,harry), parent_of(harry,elizabeth).
4. **Generalize the rule** :
   grandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,Y).
5. **Check if this rule is valid** for the positive and not valid for the negative examples

=> Elaboration and Exploitation of the Hypothesis Space

Rule Learning in ILP: global process (3)

Subtasks performed by an ILP implementation regarding ILP as a search problem in the Hypothesis Space:

1. **Structuring the Hypothesis Space,**
2. **Searching the Hypothesis Space,**
3. **Bounding the Search,**
4. **Evaluating the Hypotheses.**

Rule Learning in ILP: global process (4)

Procedure: ***traverses*** the hypothesis space, ***generating*** and ***testing*** the candidate hypothesis implemented by a ***covering algorithm*** which ***construct iteratively*** a set of clauses :

Covering (E)

Input: set of examples E
Output: a set of consistent rules

1. Learned_Rules = ∅
2. E^+ = Positives(E)
3. **while** $E^+ \neq \emptyset$
4. R = learn_rule(E^+)
5. Learned_Rules = Learned_Rules ∪ R
6. $E^+ = E^+$ - {examples covered by R}
7. **end while**
8. **return** Learned_Rules

- Starting with a empty set of rules (*Line 1*), the algorithm then generates and evaluates a clause on the positive examples (*Line 4*),
- if this clause satisfies some criteria, it adds the clause to the hypothesis (*Line 5*) and
- removes the positive examples covered by the clause (*Line 6*).
- These steps are repeated until all positive examples have been covered (*loop while Line 3*).
- The *learn_rule(e)* procedure in *Line 4* constructs individual clauses by (heuristically) searching the space of possible clauses, structured by a *specialization* or *generalization* operator.

1. Search starts with a very general rule (clause with no conditions in the body),
2. Proceeds to add literals (conditions) to this clause until it only covers positive examples, i.e., the clause is consistent.

ILP Systems Strategies for Hypothesis Search (1)

- **Progol** [Muggleton, 1995] : an iterative **top-down ILP system** that performs batch learning : all of the examples and the BK must be defined before starting the algorithm.
- **ALEPH** [Srinivasan] uses functionalities from various ILP systems like: Progol, FOIL, FORS, Indlog, MIDOS, SRT, Tilde and WARMR …
- **GILPS** [Santos 2010] : implements TopLog, ProGolem, The BK and mode declarations definitions of GILPS are identical to Aleph and Progol.
- **Golem** [Muggleton & Feng, 1990]: a bottom-up ILP system, which constraints the search space with the *relative least general generalization (rlgg)* [Plotkin, 1971].
- **Progolem** (Muggleton et al., 2010): Combine Golem and Progol strategies .
- **Toplog** (Muggleton et al., 2008): uses a declarative bias called Top-Directed Hypothesis Derivation (TDHD), where each clause issued as a candidate hypothesis must be derived from a precise logical program called top theory T.

Some ILP systems (2)

- **FOIL** [Quinlan93] learns multiple predicates from a non-interactive and non-incremental mode, realizing a top-down search in the hypothesis space
- **MIS** [xxx] : an interactive system and theory reviewer. It learns a definition of multiple predicates in a incremental way. Realize top-down search and it was the first ILP system that accept background knowledge from an intentional and extensional way.
- **Tilde** [xxx] : it's a learning system based on decision trees. These trees can be used to classify new examples or transformed in a logical program.
- **LINUS** [xxx] : an empirical ILP system, non-interactive and non-incremental. It transforms ILP systems to a attribute-value representation.
- **…**

Characteristics of various ILP systems

(Source : [Conceiçao 2008])

System	TD	BU	Predictive	Descriptive	Inc	N-Inc	Int	N-Int	Mult-Pred
Aleph		✓	✓			✓	✓		
Cigol		✓			✓		✓	✓	
Claudien	✓			✓		✓		✓	
Clint					✓		✓		✓
FOIL	✓		✓			✓		✓	
FORS	✓		✓			✓		✓	
GOLEM		✓	✓			✓		✓	
LINUS						✓	✓		
MARVIN		✓					✓	✓	
MIS	✓				✓		✓		✓
MOBAL	✓		✓			✓		✓	✓
Progol	✓		✓			✓		✓	✓
Tilde				✓					
WARMR				✓					

- **TD**: if the system uses a top-down search
- **BU**: if the system uses a bottom-up search
- **Predictive**: if the finding task of knowledge is predictive : then classification rules can be generated
- **Descriptive**: if the finding task of knowledge is descriptive : then only true properties from the examples are observed
- **Inc and N-Inc**: if the system uses incremental or non-incremental learning, respectively
- **Int and N-Int**: if the system is the type interactive or non-interactive, respectively
- **Mult-Pred**: if the system can learn multiple predicates.

Applications of ILP (1)

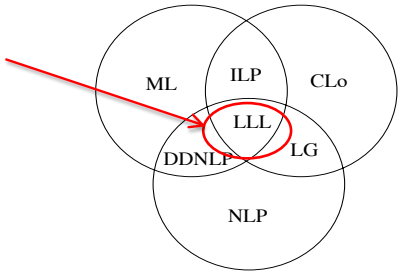
- **Application in NLP (Natural Language Processing)**
  - Information extraction from text (Named Entity Recognition, Relation Extraction, …)
  - Constructing Biological Knowledge Bases by Extracting Information from Text Sources
  - …
- **Applications to Chemoinformatics and Bioinformatics**
  - Learning drug structure-activity rules:
  - Learning rules for predicting mutagenesis, carcinogenesis
  - Learning to identify pharmacophores on small molecules (with Pfizer UK and Prolifix Ltd. Made available soon)
  - Learning rules for predicting protein secondary structure
  - Learning qualitative models for functional genomics (with the Computational Biology Group, Aberystwyth)
  - Learning to identify neuropeptide precursors (with the Machine Learning Group, University of York and the Bioinformatics Group, SmithKline-Beecham. Made available soon)
  - …

Applications of ILP (2)

- **Applications to Medicine**
  - Learning rules for selecting the best embryos for transfer in In Vitro fertilisation
  - Learning to identify diabetics susceptible to renal disease
  - Learning qualitative models of the human lung
  - …
- **Applications to other areas**
  - Learning rules from chess databases
  - Inductive Learning of Chess Rules Using Progol
  - Learning rules for finite element mesh design
  - Learning diagnostic rules for qualitative models of satellite power supplies
  - Learning qualitative models of the U-tube system
  - Learning to identify over-performing stocks
  - Learning simplified civil-service procedures
  - …
- **More from UT-ML group (Ray Mooney)**
  http://www.cs.utexas.edu/~ml/publication/ilp.html

ILP and Natural Language Processing (NLP)

Application of ILP to NLP led to the research domain of ***Learning Language in Logic (LLL)***, intersection of Machine Learning (ML), NLP and Computational Logic (CLo) :



Source [Dzeroski et al., 1999]

(with CLo = computational logic, ML = machine learning, DDNLP = data-driven NLP, LG = logic grammars, NLP = natural language processing, ILP = inductive logic programming.)

A contribution : OntoILPER [Lima el al., 2013] using GIPS ILP system [Santos 2010]

ILP: Some Limitations ...

- **Limitation related to Prolog language:**
  - Specification of a finite set ensemble of constants,

    ⇒ Numerical domains have to be borned.

    If D = {0, 1, 2, 3}, what is value of succ(3, ?)
  - Inadequate representation of numerical data.
- **Time processing:**

  ⇒ *Parallel Prolog, Map Reduce, …*
- **Unable to treat uncertainty:**

  ⇒ *PILP (Probabilistic ILP)*