SADC
Sistemas de Apoio à Decisão Clínica
(Clinical Decision Support Systems)

Mestrado em Informática Médica

5 ECTS

Inês Dutra, João Gama e Alípio Jorge

ines@dcc.fc.up.pt, jgama@fep.up.pt, amjorge@liacc.up.pt

Inductive Logic Programming: The Problem Specification

- *Given*:
  - *Examples:* first-order atoms or definite clauses, each labeled positive or negative.
  - *Background knowledge:* in the form of a definite clause theory.
  - *Language bias:* constraints on the form of interesting new clauses.

ILP Specification (Continued)

- *Find:*
  - A hypothesis *h* that meets the language constraints and that, when conjoined with *B*, entails (implies) all of the positive examples but none of the negative examples.
- To handle real-world issues such as noise, we often relax the requirements, so that *h* need only entail significantly more positive examples than negative examples.

Algoritmo?

- Algoritmo: lista de instruções bem definidas utilizadas para executar uma determinada tarefa
- Dado um estado inicial, o algoritmo passa por uma série de estados sucessivos bem definidos, eventualmente terminando
- A transição de um estado para outro não precisa ser determinística
- Alguns algoritmos são probabilísticos e incorporam aleatoriedade

ILP: A Common Approach

- Use a greedy covering algorithm.
  - Repeat while some positive examples remain uncovered (not entailed):
    - Find a *good clause* (one that covers as many positive examples as possible but no/few negatives).
    - 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*.

Some ILP Systems

- PROGOL, ALEPH (top-down): saturates first uncovered positive example, and then performs top-down admissible search of the lattice above this saturated example.
- GOLEM (bottom-up), FOIL (top-down), LINUS/DINUS.
- Tilde, Claudien, IndLog, ...

ILP Saturation

- Consists of building a *bottom clause* (seed)
- Incorporates background knowledge to an atomic formula
- Example:

```
metabolism(A) :-
    essential(A,'Non-Essential'), motif(A,'PS00510'), chromosome(A,'14'),
    interaction(A,B,C,E),
    essential(B,'Non-Essential'), motif(B,'PS00188'), chromosome(B,'2'),
    interaction(A,F,D,G),
    intertype(C,'Genetic'), intertype(D,?),
    interaction(B,A,C,E),
    interaction(B,H,C,I),
    interaction(F,A,D,G),
    interaction(H,B,C,I), interaction(H,_,_,_).
```

ILP: Aleph

- Procedure to extract theories from examples
- Complete (branch-and-bound) search for best clause in the ***whole*** space
- Search subject to several user control settings
  - Max clause length
  - Max chaining length
  - Minacc
  - Max nodes
  - Search strategy, etc.

ILP: Aleph

- Aleph
  - Desenvolvido na Universidade de Oxford por Ashwin Srinivasan

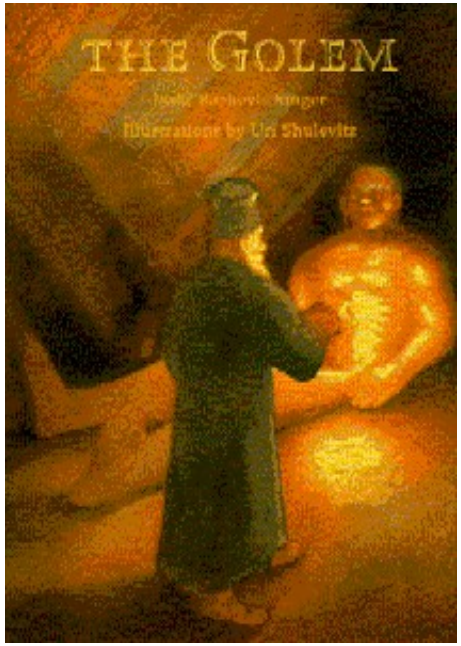
http://www.comlab.ox.ac.uk/oucl/research/areas/i

ILP: Aleph

Then the Rabbi said,

"*Golem, you have not been completely formed, but I am about to finish you now…You will do as I will tell you.*"

Saying these words, Rabbi Leib finished engraving the letter Aleph. Immediately the golem began to rise.



Aleph: algoritmo

- Estado inicial:
  - Exemplos ou observações
  - Descrições: conhecimento prévio ou background knowledge (BK)
- Estado final: hipótese ou teoria ou modelo
- Transições: hipóteses intermediárias

Aleph: algoritmo

- Select example
- Build most-specific-clause (**bottom clause**)
- Search. Find a clause more general than the bottom clause
- Remove redundant. The clause with the best score is added to the current theory, and all examples made redundant are removed. This step is sometimes called the "**cover removal**" step. Note here that the best clause may make clauses other than the examples redundant
- Return to first step

Aleph: Knowledge Representation

Input Files: Prolog Syntax

dtp.b: BK
dtp.f: pos examples
dtp.n: neg examples

Representation: BK

```
chromosome('G234064','1').
chromosome('G234065','1').
chromosome('G234070','1').
chromosome('G234073','1').
chromosome('G234074','1').
chromosome('G234076','1').
chromosome('G234084','2').
chromosome('G234085','2').
chromosome('G234089','2').
```

Representation: BK

```
interaction('G234062','G235011','Physical',?).
interaction('G234064','G234126','Genetic-
                                Physical','0.9141').
interaction('G234064','G235065','Genetic-
                                Physical','0.7515').
interaction('G234064','G235571','Physical','0.9691').
interaction('G234065','G234073','Physical','0.7492').
interaction('G234065','G235042','Physical','-0.4659').
```

Representation: Examples

metabolism('G239098').

metabolism('G234980').

metabolism('G235245').

metabolism('G234108').

metabolism('G238387').

metabolism('G240504').

metabolism('G236733').

Example of clause learned

```
metabolism(A) :-
        chromosome(A,'15'),
        interaction(A,B,_,_),
        complex(B,'Transcription
                complexes/Transcriptosome').
```

A and *B* are variables that represent genes

Aleph: algoritmo

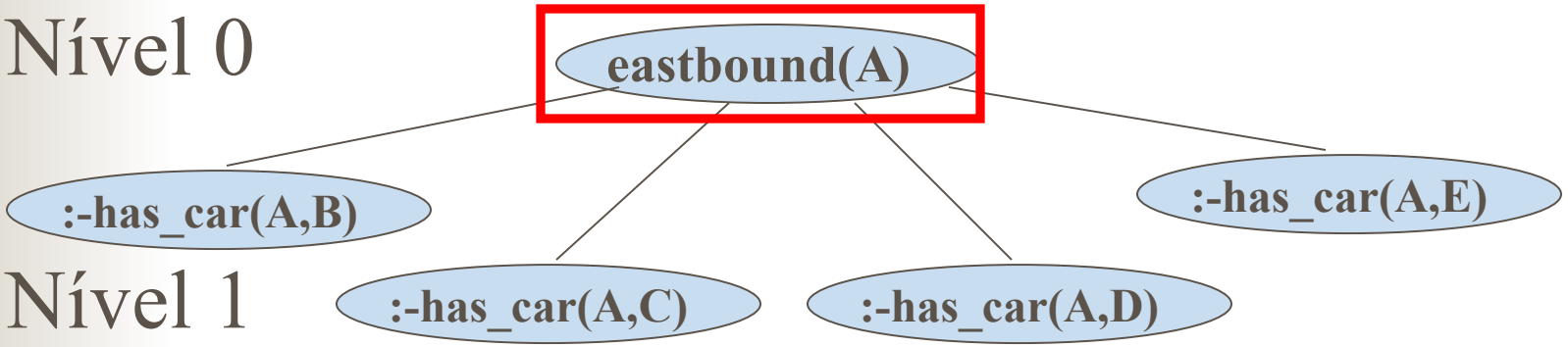
- Exemplo: trens que vão para leste e trens que vão para oeste

Aleph: algoritmo

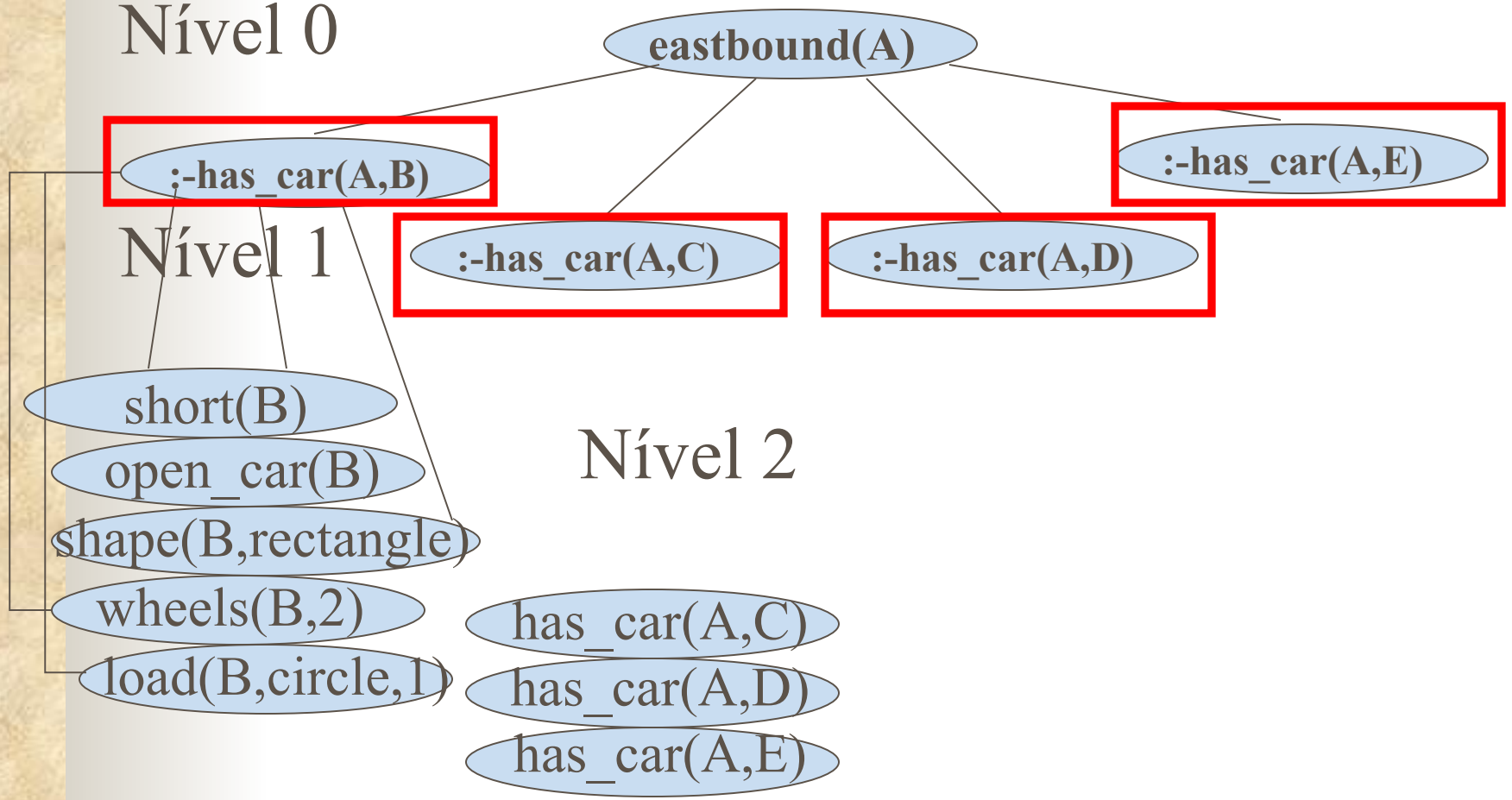
- Saturação:

```
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).
```

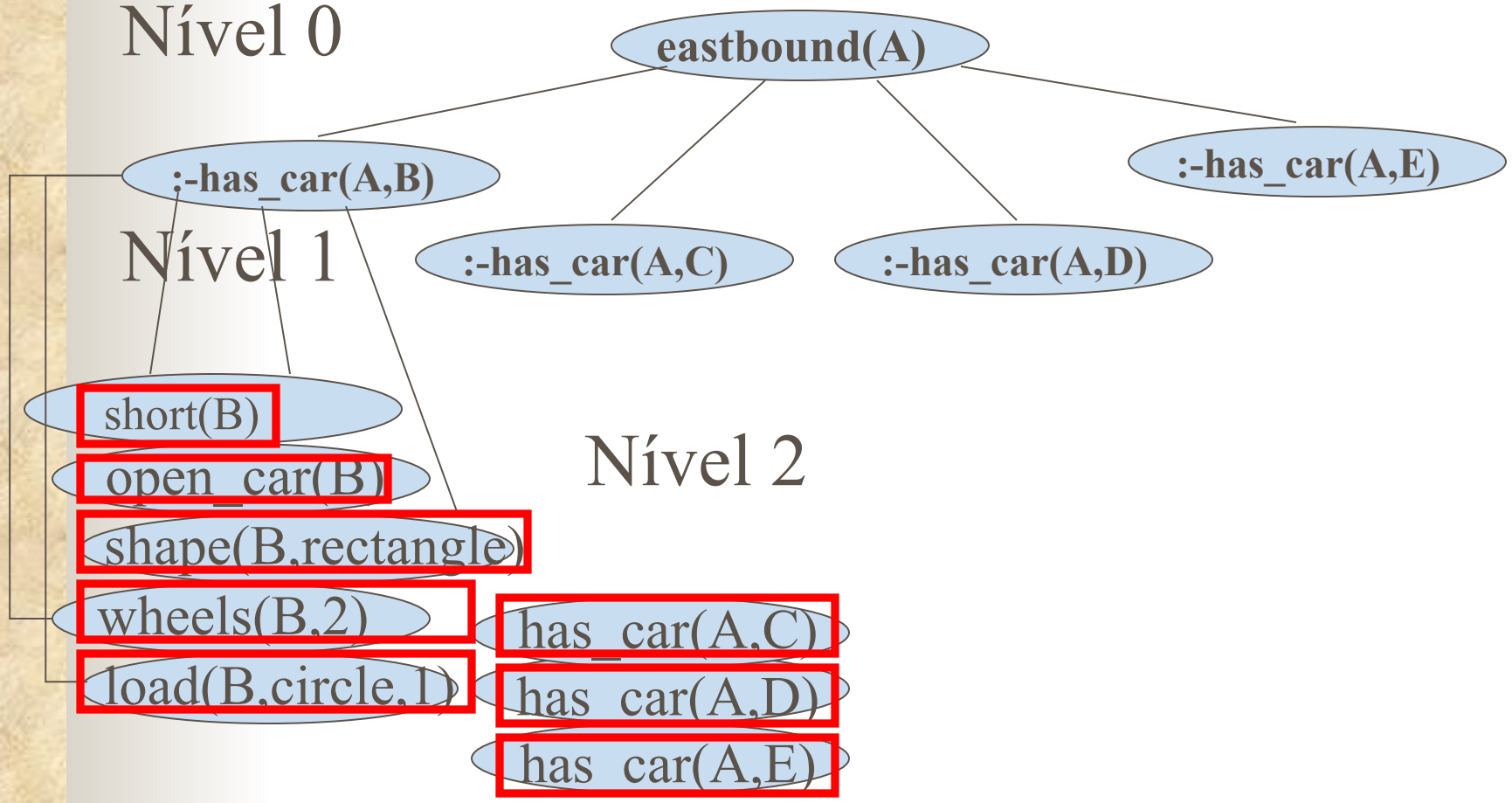
Aleph: Busca


Nível 0
eastbound(A)
:-has_car(A,B)
:-has_car(A,E)
Nível 1
:-has_car(A,C)
:-has_car(A,D)

Aleph: Busca


Nível 0
eastbound(A)
:-has_car(A,B)
:-has_car(A,E)
Nível 1
:-has_car(A,C)
:-has_car(A,D)
short(B)
open_car(B)
shape(B,rectangle)
wheels(B,2)
load(B,circle,1)
Nível 2
has_car(A,C)
has_car(A,D)
has_car(A,E)

Aleph: Busca


Nível 0
eastbound(A)
:-has_car(A,B)
:-has_car(A,E)
Nível 1
:-has_car(A,C)
:-has_car(A,D)
short(B)
open_car(B)
shape(B,rectangle)
wheels(B,2)
load(B,circle,1)
Nível 2
has_car(A,C)
has_car(A,D)
has_car(A,E)

Aleph: algoritmo

- Busca: cláusula mais geral

```
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).
```

Aleph: algoritmo

- Busca: adiciona “filhos” possíveis (literais candidatos)

```
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).
```

Aleph: algoritmo

- Busca: adiciona “filhos” possíveis ao primeiro filho

```
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).
```

Aleph: algoritmo

- Busca: segundo filho de nível 1

```
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).
```

Aleph: algoritmo

- Busca: filhos do segundo filho de nível 1

```
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).
```

Aleph: example of run

➔ aleph_trains

Aleph: how to run?

- You need to have a Prolog system
  - Yap: http://yap.sourceforge.net OU
  - SWI: http://www.**swi-prolog.org**
- Aleph:
  http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/
- Files: .b, .f, .n
- To make things easier: everything in the same directory!

Aleph: Comandos básicos

- read_all
- reduce
- induce

Aleph: Parameters

```
:- set(clauselength,5).
:- set(depth, 200).
:- set(i,3).
:- set(noise,0).
:- set(minacc,0.7).
:- set(nodes,1000000).
:- set(m,20).
:- set(evalfn,mestimate).
:- set(test_pos,'/u/dutra/Protein/prot_test_set.f').
:- set(test_neg,'/u/dutra/Protein/prot_test_set.n').
:- set(optimise_clauses,true).

:- set(record,true).
:- set(recordfile,'prot_train_set.out').
:- set(samplesize,0).
```

Strength estimate = (*support* + *m* * *prior*) / (*coverage* + *m*)

M → 0, strength → precision

Support = True positives

Coverage = True positives + false negatives

Aleph: Modes and Types

```
:- modeh(1,eastbound(+train)).
:- modeb(1,short(+car)).
:- modeb(1,closed(+car)).
:- modeb(1,long(+car)).
:- modeb(1,open_car(+car)).
:- modeb(1,double(+car)).
:- modeb(1,jagged(+car)).
:- modeb(1,shape(+car,#shape)).
:- modeb(1,load(+car,#shape,#int)).
:- modeb(1,wheels(+car,#int)).
:- modeb(*,has_car(+train,-car)).
```

```
:- determination(eastbound/1,short/1).
:- determination(eastbound/1,closed/1).
:- determination(eastbound/1,long/1).
:- determination(eastbound/1,open_car/1).
:- determination(eastbound/1,double/1).
:- determination(eastbound/1,jagged/1).
:- determination(eastbound/1,shape/2).
:- determination(eastbound/1,wheels/2).
:- determination(eastbound/1,has_car/2).
:- determination(eastbound/1,load/3).
```

Aleph: Modes and Types

```
:- modeh(1,metabolism(+gene)).

:- modeb(1,essential(+gene,#essential)).
:- modeb(1,class(+gene,#class)).
:- modeb(1,complex(+gene,#complex)).
:- modeb(1,phenotype(+gene,#phenotype)).
:- modeb(1,motif(+gene,#motif)).
:- modeb(1,chromosome(+gene,#chromosome)).
:- modeb(*,gte(+number,#number)).
:- modeb(*,interaction(+gene,-gene,-intertype,-number)).
:- modeb(1,intertype(+intertype,#intertype)).
```

Case study 1: Learning rules for early diagnosis of rheumatic diseases

- Correct diagnosis in the early stage of a rheumatic disease is a difficult problem [Pirnat et al. 1989]
- Having passed all investigations, many patients can not be reliably diagnosed after their first visit to the specialist
- Two reasons:
  - symptoms, clinical manifestations, laboratory and radiological findings of various rheumatic diseases are very similar and not specific
  - subjective interpretation of anamnestic, clinical, laboratory and radiological data

Case study 1: rheumatic disease

- Application of LINUS to the problem of learning rules for early diagnosis of rheumatic diseases.
- Given: attribute-value descriptions of patient data, bk provided by a medical specialist in the form of typical co-ocurrences of symptoms
- Experiments: LINUS with CN2
- Showed that the noise-handling mechanism of CN2 and the ability of LINUS to use bk affect the performance (classification accuracy and information content) and the complexity of the induced diagnostic rules

Case study 1: rheumatic disease

- Data about 462 patients (Univ medical center of ljubljana)
- Over 200 rheumatic diseases that can be grouped into 3, 6, 8 or 12 diagnostic classes
- 8 classes: suggested by a specialist

Case study 1: rheumatic disease

Class	Name	Num patients
A1	Degenerative spine diseases	158
A2	Degenerative joint diseases	128
B1	Inflammatory spine diseases	16
B234	Other inflammatory diseases	29
C	Extra-articular rheumatism	21
D	Crystal-induced synovitis	24
E	Non-specific rheumatism manifestations	32
F	Non rheumatic diseases	54

Case study 1: rheumatic disease

- Experiments on anamnestic data without patient´s clinical manifestations, laboratory and radiological findings
- 16 anamnestic attributes:

  sex, age, family anamnesis, duration of present symptoms, duration of rheumatic diseases, joint pain (arthrotic or arthritic), number of painful joints, number of swollen joints, spinal pain, other pain, duration of morning stifness, skin manifestations, mucosal manifestations, eye manifestations, other manifestations and therapy.

- From 462 patients, 8 were incomplete, 12 attribute values missing (sex and age) (not a problem since LINUS with CN2 handles missing data)

Case study 1: rheumatic disease

- Medical bk: aumengted the patient data with typical co-ocurrences of symptoms
- 6 typical groups suggested by the specialist:

Case study 1: rheumatic disease

Joint pain	Morning stifness
No pain	≤ 1h
arthrotic	≤ 1h
arthritic	> 1h

spinal pain	Morning stifness
No pain	≤ 1h
spondylotic	≤ 1h
spondylitic	> 1h

sex	Other pain
male	thorax
male	heels

Joint pain	Spinal pain
No pain	spondylotic
arthrotic	No pain
No pain	spondylitic
arthritic	spondylitic
arthritic	No pain
No pain	No pain

Case study 1: rheumatic disease

Joint pain	Spinal pain	Painful joints
No pain	spondylotic	0
arthrotic	No pain	$1 \leq$ joints ≤ 30
No pain	spondylitic	0
arthrotic	spondylitic	$1 \leq$ joints ≤ 5
arthritic	No pain	$1 \leq$ joints ≤ 30
No pain	No pain	0

Swollen joints	Painful joints
0	0
0	$1 \leq$ joints ≤ 30
$1 \leq$ joints ≤ 10	$0 \leq$ joints ≤ 30

Case study 1: rheumatic disease

bk	Signif test	Acc (%)	Relative inf score (%)	Num of rules	Num of literals
no	no	62.8	49	96	302
no	yes	51.7	22	30	102
yes	no	72.9	59	96	301
yes	yes	52.4	30	38	120

Medical evaluation

- Specialist evaluated the entire set of induced rules
- For each of the conditions in a rule:
  - +1 if the condition favours the diagnosis made by the rule
  - -1 if the condition was against the diagnosis
  - 0 if the condition is irrelevant
- Mark of a rule: sum of the points for all conditions in the rule
- Actual marks range from -1 to 3
  - 3: rules which are very characteristic for a disease
  - 2: good, correct rules
  - 1: not wrong, but not too characteristic for the disease
  - 0: by chance
  - -1: misleading rules

Medical evaluation: sem BK

class	Num rules with mark					rules	avgm
	3	2	1	0	-1		
A1			4	1	2	7	0.29
A2			3	2	1	6	0.33
B1		1	2			3	1.33
B2			3	1		4	0.75
C			1	2		3	0.33
D		1	2			3	1.33
E				3		3	0.00
F				1		1	0.00

Medical evaluation: com BK

class	Num rules with mark					rules	avgm
	3	2	1	0	-1		
A1		3	2	2		7	1.14
A2	1	1	3	1	1	7	1.00
B1		1	2			3	1.33
B2	1	2	4			7	1.57
C				3		3	0.00
D		1	2			3	1.33
E				4		4	0.00
F	1	1	1	1		4	1.50

Medical evaluation

- Use of bk provided by specialist helps to guide the search to obtain new knowledge
- System can work and infer the specialist´s knowledge plus new knowledge, but it will probably take much more time ☹

Case study 2: drug discovery

- Given:
  - Molecules active and inactive for dtp
  - Their description in terms of coordinates and bonds
- Find small structures that model active molecules

Case study 2: drug discovery

- Examples of dtp groups:

```
hydrophobic(m752,
        hyphob([a2, a3, a5, a8, a7, a4, a2],
        2.16452, -0.833917, 3.6379)).
hacc(m9706,
    hacc(a10, -6.2969, -1.3684, -0.4631)).
```

Case study 2: drug discovery

- Utilisation of refinement operator

```
refine(false,Clause):-
    member(Point1, [hydrophobic(M,P1), hdonor(M,P1),halogen(M,P1),hacc(M,P1)]),
    member(Point2,[hydrophobic(M,P2),hdonor(M,P2),halogen(M,P2),hacc(M,P2)]),
    Clause = (active(M) :- Point1, Point2, dist(M,P1,P2,D1,E)).

refine(Clause1,Clause2):-
        Clause1 = (active(M) :- Point1,Point2, dist(M,P1,P2,D1,E)),      member(Point3,
        [hydrophobic(M,P3),hdonor(M,P3),halogen(M,P3),hacc(M,P3)]),
        Clause2 = (active(M) :- Point1, Point2, dist(M,P1,P2,D1,E),
                        Point3, dist(M,P1,P3,D2,E), dist(M,P2,P3,D3,E)).
```

- Reduce search space!!!

Como avaliar resultados?

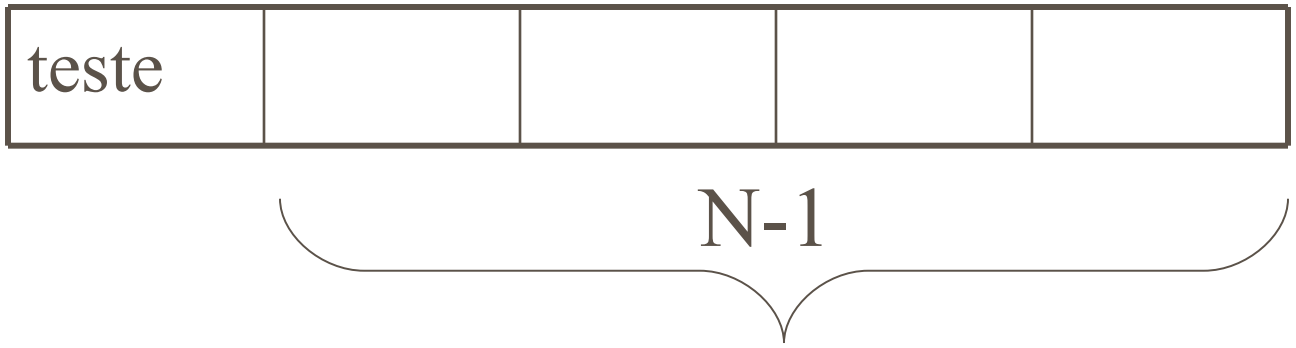
- Conjunto de treino?
- Como verificar se o classificador encontrado (teoria) comporta-se bem para novos exemplos (que nunca foram vistos antes?)
- Conjunto de ajuste (tuning set)
- Métricas:
  - Accuracy
  - Receiver operating characteristic (ROC)
  - Precision-recall (PR)
  - Area under the curve (AUC)

Como avaliar resultados?

- Classificadores separam:
  - TP: True positives
  - TN: True negatives
  - FP: False positives
  - FN: False negatives

Como avaliar resultados?

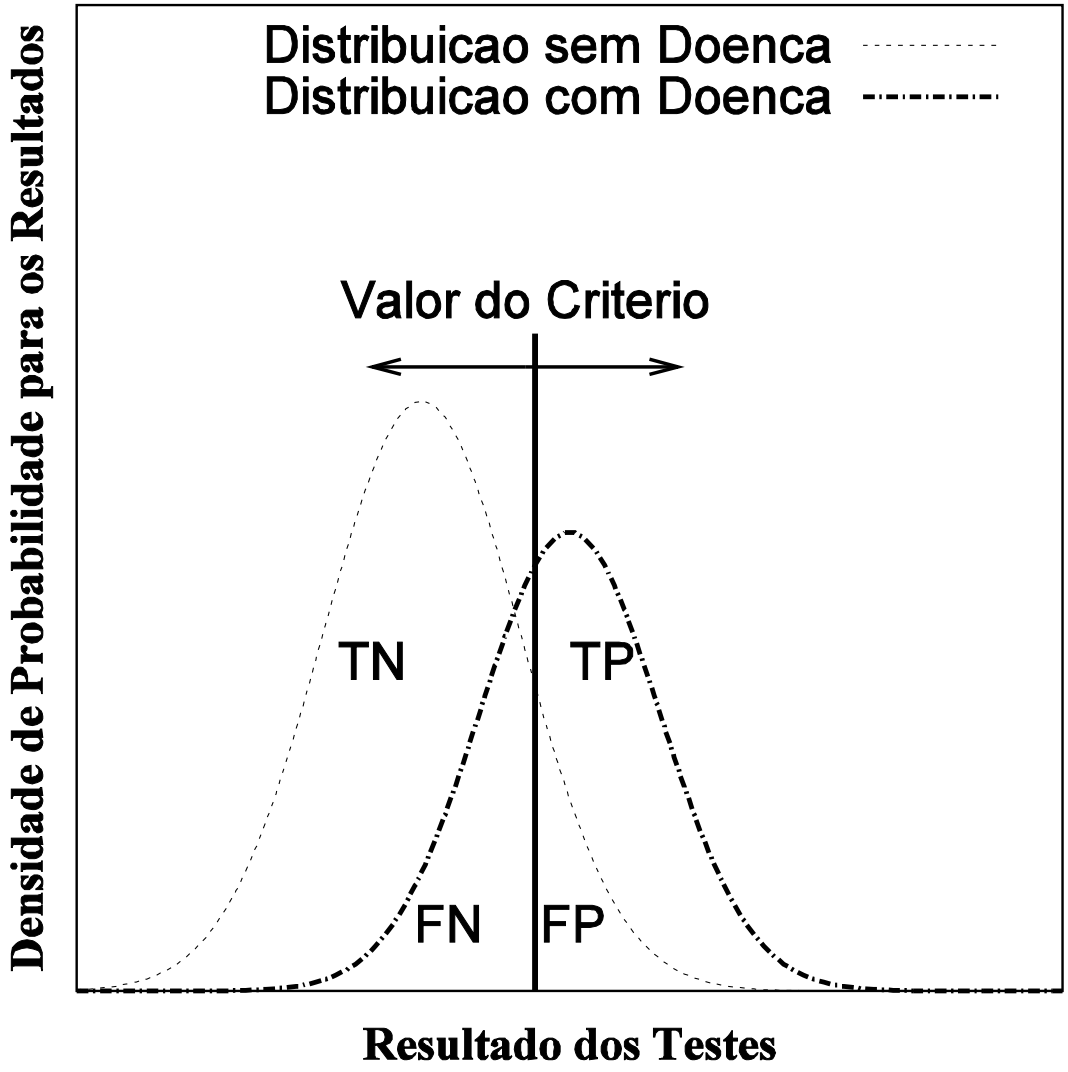
- Para minimizar erro do classificador em exemplos nunca vistos: cross-validation
- Particiona o conjunto de treino em n partes iguais. Treina em n-1 e testa no n-ésimo conjunto. Repete n vezes



Como avaliar resultados?

- Leave-one-out: cross-validation onde temos n exemplos, treinamos em n-1 e deixamos 1 único exemplo para teste
- Problemas com cross-validation: sobreposição de exemplos em cada conjunto de treino
- Segundo Dietterich: 5 times 2-fold cross-validation should be used

Avaliação

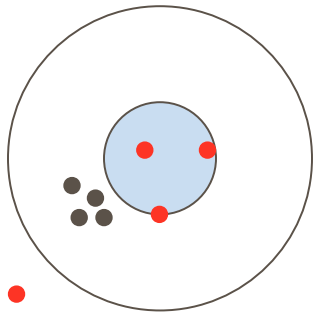

Distribuicao sem Doenca
Distribuicao com Doenca
Valor do Criterio
TN
TP
FN
FP
Densidade de Probabilidade para os Resultados
Resultado dos Testes

Como avaliar resultados?

- Tuning set?
- Geralmente utilizado para estimar parâmetros

Métricas

- Accuracy x Precision



Accuracy

Precision

$$P = TP / (TP+FP)$$

$$Acc1 = (TP+TN)/Totex$$

$$Acc2 = (TP/(TP+FP) + TN/(TN+FN)) / 2$$

Tx acerto pos

Tx acerto neg

A

TP=63	FP=28	91	
FN=37	TN=72	109	
100	100	200	

TPR = 0.63

FPR = 0.28

ACC = 0.68

B

TP=77	FP=77	154	
FN=23	TN=23	46	
100	100	200	

TPR = 0.77

FPR = 0.77

ACC = 0.50

C

TP=24	FP=88	112	
FN=76	TN=12	88	
100	100	200	

TPR = 0.24

FPR = 0.88

ACC = 0.18

C'

TP=88	FP=24	112	
FN=12	TN=76	88	
100	100	200	

TPR = 0.88

FPR = 0.24

ACC = 0.82