Temporal Aspects of Inductive Logic Programming

Fabrizio Riguzzi

Department of Mathematics and Computer Science – University of Ferrara
fabrizio.riguzzi@unife.it
http://ml.unife.it/tailp/

Dipartimento di Matematica e Informatica

Outline

Dipartimento di Matematica e Informatica

Machine Learning

- Classification
- Positive and negative examples
- Representation of data and classifier

Dipartimento
di Matematica
e Informatica

Representation Langauges

- Propositional or attribute-value languages
- First order/logical/relational languages

Dipartimento
di Matematica
e Informatica

Propositional Languages

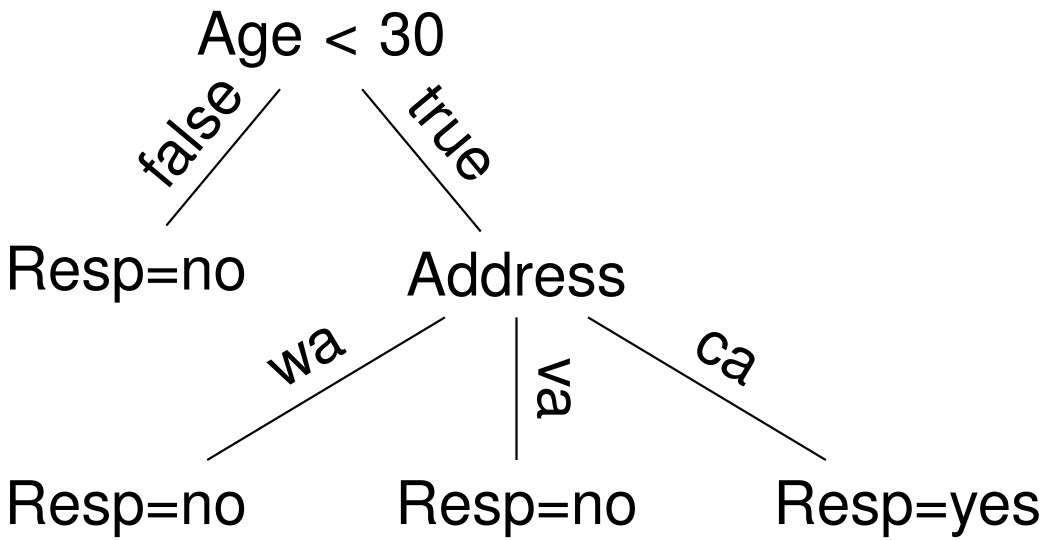
- Each instance described by a fixed set of attributes
- Data: table
- One target attribute
- Model/Classifier: production rules, decision trees, Bayesian networks

Example: Targeted Mailing

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	yes
ann	29	f	wa	no
steve	31	m	va	no

Decision Trees



customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	yes
ann	29	f	wa	no
steve	31	m	va	no

Dipartimento di Matematica e Informatica

Production Rules

If Age<30 and Address=ca then Resp=yes

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	yes
ann	29	f	wa	no
steve	31	m	va	no

Dipartimento di Matematica e Informatica

First-Order Languages

- Instances described by logic theories
- They allow to easily represents parts of objects, attributes of parts and relations among parts

Dipartimento
di Matematica
e Informatica

Targeted Mailing

customer

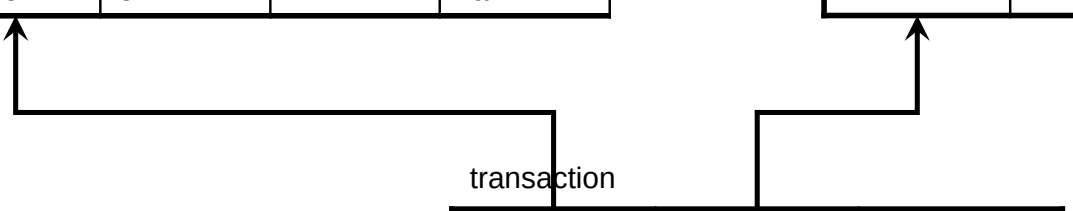
Name	Age	Sex	Address
john	35	m	ca
mary	25	f	ca
ann	29	f	wa
steve	31	m	va

respond

Name
ann

article

Name	Category	Size	Price
bike_1	sport	l	1000
jacket_2	clothing	l	150
tent_2	outdoor	m	250



transaction

Name	Article	Quantity
john	bike_1	2
ann	jacket_2	1
steve	bike_1	1
john	tent_2	1
ann	bike_1	3

Logic

- Useful to model domains with complex relationships among entities
- Various forms:
  - First Order Logic
  - Logic Programming
  - Description Logics
  - Temporal Logics

Logic Programming

- Closed World Assumption
- Turing complete
- Prolog

$$flu(bob).$$
$$hay\_fever(bob).$$
$$sneezing(X) \leftarrow flu(X).$$
$$sneezing(X) \leftarrow hay\_fever(X).$$

Inductive Logic Programming

- Aim:
  - classifying instances of the domain, i.e.
  - predicting the class
- Two settings:
  - Learning from entailment
  - Learning from interpretations

Learning from Entailment

- Given
  - A set of positive example E^+
  - A set of negative examples E^-
  - A background knowledge B
  - A space of possible programs $\mathcal{H}$
- Find a program $P \in \mathcal{H}$ such that
  - $\forall e^+ \in E^+, P \cup B \models e^+$ (completeness)
  - $\forall e^- \in E^-, P \cup B \not\models e^-$ (consistency)

Targeted Mailing

customer

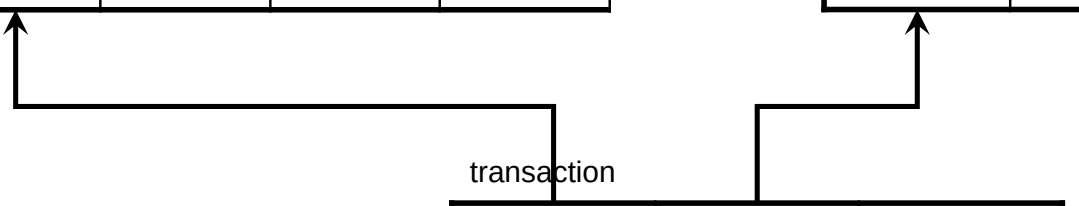
Name	Age	Sex	Address
john	35	m	ca
mary	25	f	ca
ann	29	f	wa
steve	31	m	va

respond

Name
ann

article

Name	Category	Size	Price
bike_1	sport	l	1000
jacket_2	clothing	l	150
tent_2	outdoor	m	250



transaction

Name	Article	Quantity
john	bike_1	2
ann	jacket_2	1
steve	bike_1	1
john	tent_2	1
ann	bike_1	3

Mailing Example

- Positive examples $E^+ = \{respond(ann)\}$
- Negative examples
  $E^- = \{respond(john), respond(mary), respond(steve)\}$
- Background $B =$ facts for relations *customer*, *transaction* and *article*
  $customer(john, 35, m, ca).$
  $customer(mary, 25, f, ca).$
  $customer(ann, 29, f, wa)....$
  $transaction(john, bike\_1, 2).$
  $transaction(ann, jacket\_2, 1)....$
  $article(bike\_1, sport, l, 1000).$
  $article(jacket\_2, clothing, l, 150)....$

Mailing Example

- Space of programs $\mathcal{H}$: programs containing clauses with
  - in the head $respond(Customer)$
  - in the body a conjunction of literals from the set
    $\{customer(Customer, Age, Sex, Address),$
    $transaction(Customer, Article, Quantity),$
    $article(Article, Category, Price),$
    $Age = constant, Sex = constant, \ldots\}$
- Possible solution
  $respond(Customer) \leftarrow transaction(Customer, Article, \_Quantity),$
  $article(Article, Category, \_Size, \_Price),$
  $Category = clothing$

Definitions

- $covers(P, e) = true$ if $B \cup P \models e$
- $covers(P, E) = \{e \in E | covers(P, e) = true\}$
- A theory P is more general than Q if $covers(P, U) \supseteq covers(Q, U)$
- Specialization opposite of generalization
- If $B \cup P \models Q$ then $B \cup Q \models e \Rightarrow B \cup P \models e$ so P is more general than Q
- A clause C is more general than D if $covers(\{C\}, U) \supseteq covers(\{D\}, U)$
- If $B, C \models D$ then C is more general than D
- If a clause covers an example, all of its generalizations will (*covers* is antimonotonic with respect to specialization)
- If a clause does not cover an example, none of its specializations will

Dipartimento di Matematica e Informatica

Theta Subsumption

- A clause
  $h \leftarrow b_1, \ldots, b_n$
  can be seen as a set of literals $\{h, not\ b_1, \ldots, not\ b_n\}$
- A substitution θ is a replacement of variable with terms:
  $\theta = \{X/a, Y/b\}$
- C θ-subsumes D ($C \geq D$) if there exists a substitution θ such that $C\theta \subseteq D$ [Plotkin 70]
- $C \geq D \Rightarrow C \models D \Rightarrow B, C \models D \Rightarrow C$ is more general than D
- $C \models D \not\Rightarrow C \geq D$

Dipartimento di Matematica e Informatica

Examples of Theta Subsumption

- $C1 = father(X, Y) \leftarrow parent(X, Y)$
- $C2 = father(X, Y) \leftarrow parent(X, Y), male(X)$
- $C3 = father(john, steve) \leftarrow parent(john, steve), male(john)$
- $C1 = \{father(X, Y), not\ parent(X, Y)\}$
- $C2 = \{father(X, Y), not\ parent(X, Y), not\ male(X)\}$
- $C3 = \{father(john, steve), not\ parent(john, steve), not\ male(john)\}$
- $C1 \geq C2$ with $\theta = \emptyset$
- $C1 \geq C3$ with $\theta = \{X/john, Y/steve\}$
- $C2 \geq C3$ with $\theta = \{X/john, Y/steve\}$

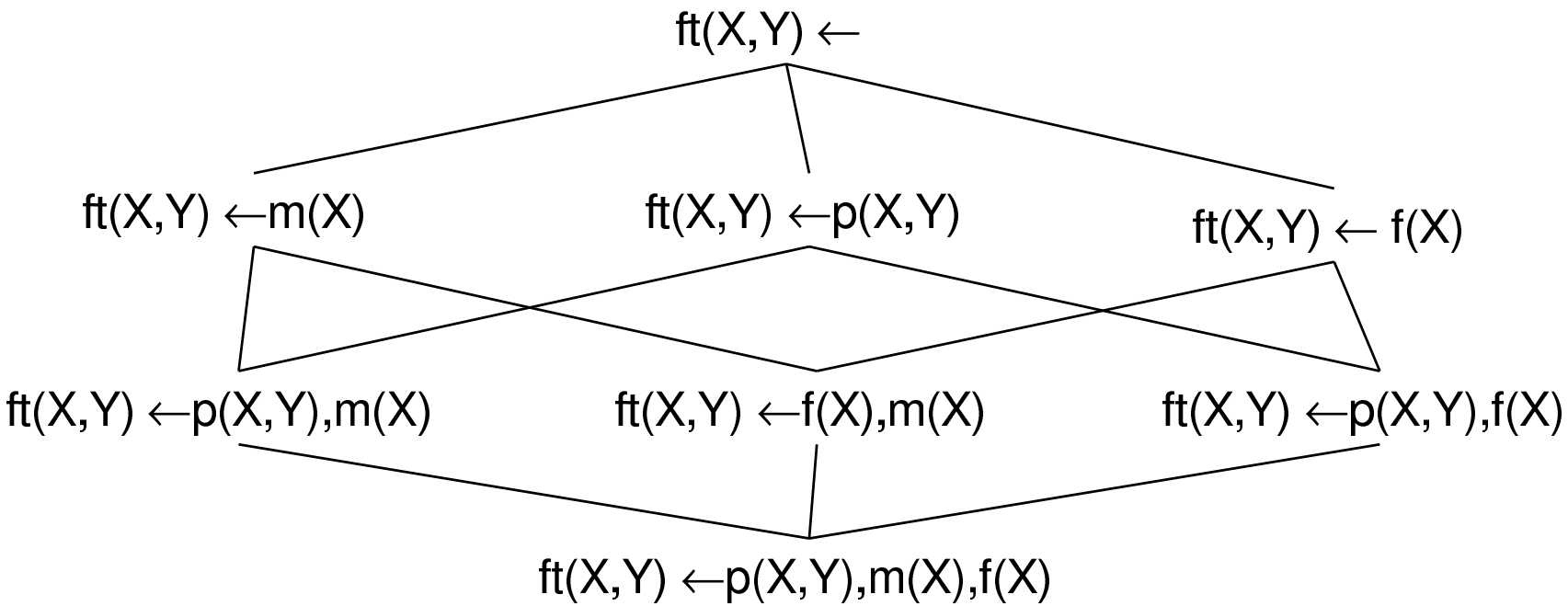
In Practice

- Coverage test: SLD or SLDNF resolution
  - Try to derive e from $B \cup P \cup \{C\}$
- Generality order:
  - θ-subsumption

Properties of Theta Subsumption

- θ-subsumption induces a lattice in the space of clauses
- Every set of clauses has a least upper bound (lub) and a greatest lower bound (glb)
- This is not true for the generality relation based on logical consequence

Lattice



Dipartimento di Matematica e Informatica

Top-Down Systems

- Covering loop
- Search for a clause from general to specific

Learn(E, B)
$P := 0$
repeat /* covering loop */
    $C :=$GenerateClauseTopDown(E, B)
    $P := P \cup \{C\}$
    Remove from E the positive examples covered by P
until Sufficiency criterion
return P

Dipartimento di Matematica e Informatica

Top-down Systems

GenerateClauseTopDown(E,B)
Beam := $\{p(X) \leftarrow true\}$
BestClause := *null*
repeat /* specialization loop */
    Remove the first clause *C* of *Beam*
    compute $\rho(C)$
    score all the refinements
    update *BestClause*
    add all the refinements to the beam
    order the beam according to the score
    remove the last clauses that exceed the dimension *d*
until the Necessity criterion is satisfied
return *BestClause*

Typical Stopping Criteria

- Sufficiency criteria:
  - $E^+ = \emptyset$
  - GenerateClauseTopDown returns *null*
  - a disjunction of the above
- Necessity criteria
  - the number of negative examples covered by *BestClause* is 0
  - the number of negative examples covered by *BestClause* is below a threshold
  - *Beam* is empty
  - a disjunction of the above

Refinement Operator

- $\rho(C) = \{D | D \in L, C \geq D\}$
- where L is the space of possible clauses
- A refinement operator usually generates only minimal specializations
- A typical refinement operator applies two syntactic operations to a clause
  - it applies a substitution to the clause
  - it adds a literal to the body

Heuristic Functions

- n^+, n^- number of positive and negative examples in the training set, $n = n^+ + n^-$
- $n^+(C), n^-(C)$ number of positive and negative examples covered by clause C
- $n(C) = n^+(C) + n^-(C)$
- Accuracy: $Acc = P(+|C)$ (more accurately Precision), $P(+|C)$ can be estimated by
  - relative frequency: $P(+|C) = \frac{n^+(C)}{n(C)}$
  - m-estimate: $P(+|C) = \frac{n^+(C)+mP(+)}{n(C)+m}$, where $P(+) = n^+/n$
  - Laplace: m-estimate with $m = 2, P(+) = 0.5$ $P(+|C) = \frac{n^+(C)+1}{n(C)+2}$

Heuristic Functions

- Coverage: $Cov = n^+(C) - n^-(C)$
- Informativity: $Inf = \log_2(Acc)$
- Weighted relative accuracy: $WRAcc = P(C)(P(+|C) - P(+))$, where $P(C) = n(C)/n$

FOIL [Quinlan 90]

- Top-down system with
  - Dimension of the beam: 1
  - Heuristic: (approximately) weighted gain of Inf: $H = n(C')(Inf(C') - Inf(C))$
  - Refinement operator: addition of a literal or unification of two variables
  - Sufficiency criterion: $E^+ = \emptyset$
  - Necessity criterion: $n^-(BestClause) = 0$

Progol [Muggleton 95] & Aleph [Srinivasan 07]

- Top-down system with
  - Dimension of the beam: user defined
  - Heuristic: Compression: $Comp = n^+(C) - n^-(C) - |C|$ + branch and bound (Aleph)
  - Refinement operator: adds a literal from the most specific clause (*bottom clause*) $\perp$ after having replaced some of the constants with variables
  - Sufficiency criterion: $E^+ = \emptyset$
  - Necessity criterion: $Beam = \emptyset$ or a maximum number of iterations of the loop is reached

Language Bias

- Mode declarations
- Syntax

```
modeh(RecallNumber,PredicateMode).
modeb(RecallNumber,PredicateMode).
```

- `RecallNumber` can be a number or *. Usually *. Maximum number of answers to queries to include in the bottom clause

Mode Declarations

- `PredicateMode` template of the form:

```
p(ModeType, ModeType,...)
```

- Some examples:

```
modeb(1,mem(+number,+list)).
modeb(1,dec(+integer,-integer)).
modeb(1,mult(+integer,+integer,-integer)).
modeb(1,plus(+integer,+integer,-integer)).
modeb(1,(+integer)=(#integer)).
modeb(*,has_car(+train,-car))
```

Mode Declarations

- `ModeType` can be:
  - Simple:
    - `+T` input variables of type `T`;
    - `-T` output variables of type `T`; or
    - `#T` constants of type `T`.
  - Structured: of the form `f(..)` where `f` is a function symbol and every argument can be either simple or structured. For example:

```
modeb(1,mem(+number,[+number|+list])).
```

Bottom Clause $\perp$ [Muggleton 95]

- Most specific clause covering an example e
- Form: $e \leftarrow B$
- B: set of ground literals that are true regarding the example e
- B obtained by considering the constants in e and querying the predicates of the background for true atoms regarding these constants
- A map from types to lists of constants is kept, it is enlarged with constants in the answers to the queries and the procedure is iterated a user-defined number of times
- Values for output arguments are used as input arguments for other predicates

Dipartimento di Matematica e Informatica

Bottom Clause $\bot$

- Example:

$e = father(john, mary)$
$B = \{parent(john, mary), parent(david, steve),$
$parent(kathy, mary), female(kathy), male(john), male(david)\}$
$modeh(*, father(+person, +person)).$
$modeb(*, parent(+person, -person)).$
$modeb(*, parent(-person, +person)).$
$modeb(*, male(+person)).$
$modeb(*, female(\#person)).$
$e \leftarrow B = father(john, mary) \leftarrow parent(john, mary), male(john),$
$parent(kathy, mary), female(kathy).$

Bottom Clause $\bot$

- The resulting ground clause $\bot$ is then processed by replacing each term in a + or - placemarker with a variable
- An input variable (+T) must appear as an output variable with the same type in a previous literal or as an input variable in the head and a constant (#T) is not replaced by a variable.

$\bot = father(X, Y) \leftarrow$
$parent(X, Y), male(X), parent(Z, Y), female(kathy).$

Aleph Example

`http://cplint.eu/p/grandfather_sol.pl`

- Learning the grandfather relation from knowledge of `parent/2`, `male/1` and `female/1`

Learning from Interpretations

- Interpretation = set of ground atoms.
- Aim: learning a classifier for logical interpretations
- Classifier: a set of disjunctive clauses T
- Disjunctive clause
  $C = h_1 \vee h_2 \vee \ldots \vee h_n \leftarrow b_1, b_2, \ldots, b_m$
  can be seen as a set of literals
  $\{h_1, \ldots, h_n, not\ b_1, \ldots, not\ b_m\}$
- $head(C) = h_1 \vee h_2 \vee \ldots \vee h_n$ or $\{h_1, \ldots, h_n\}$
- $body(C) = b_1, b_2, \ldots, b_m$ or $\{b_1, \ldots, b_m\}$
- $body^+(C) =$ set of positive literals of $body(C)$
- $body^-(C) =$ set of atoms of negative literals of $body(C)$

Dipartimento di Matematica e Informatica

Learning from Interpretations

- Set of clauses as a classifier
  - an interpretation I is positive if all the clauses of T are true in the interpretation ($I \models T$)
  - an interpretation I is negative if there exists at least one clause of T that is false in it ($I \not\models T$)
- A clause C is true in an interpretation I ($I \models C$) if for all grounding substitutions θ of C:
  $I \models body(C)\theta \Rightarrow head(C)\theta \cap I \neq \emptyset$
  or
  $body^+(C)\theta \subseteq I \wedge body^-(C)\theta \cap I = \emptyset \Rightarrow head(C)\theta \cap I \neq \emptyset$

Dipartimento di Matematica e Informatica

Test of the Truth of a Clause

- Range restricted clause: all the variables of the clause appear in positive literals in the body
- Range restricted clause C, finite interpretation I: run the query $? - body(C), not\ head(C)$ against a logic program containing I
- If $C = h_1 \vee h_2 \vee \ldots \vee h_n \leftarrow b_1, b_2, \ldots, b_m$ then the query is $? - b_1, b_2, \ldots, b_m, not\ h_1, not\ h_2, \ldots, not\ h_n$
- If the query succeeds, C is false in I. If the query fails, C is true in I [De Raedt, Bruynooghe 93]

Example

- $I = \{female(liz), male(richard), gorilla(liz), gorilla(richard)\}$
- $C = male(X) \vee female(X) \leftarrow gorilla(X)$: the clause is true in I because the query $? - gorilla(X), not\ male(X), not\ female(X)$ fails
- $C = male(X) \leftarrow gorilla(X)$: the clause is false in I because the query
  $? - gorilla(X), not\ male(X)$ succeeds with $\theta = \{X/liz\}$.

Learning from Interpretations

- **Given**
  - a space of possible clausal theories $\mathcal{H}$
  - a set P of interpretations
  - a set N of interpretations
- **Find**: a clausal theory $H \in \mathcal{H}$ such that
  - for all $p \in P$, $p \models H$
  - for all $n \in N$, $n \not\models H$
- Less expressive than learning from entailment: no recursive definitions

Dipartimento di Matematica e Informatica

Test with Background

- Background: a normal program B
- Truth of a clause C in the interpretation $M(B \cup I)$ where M is the model according to the chosen semantics and I is an interpretation (i.e. $B \cup I \models C$)
- Range restricted clause C, normal program B containing only range restricted clauses, interpretation I: run the query $? - body(C), not\ head(C)$ against the logic program $B \cup I$.
- If the query succeeds, C is false in $M(B \cup I)$ ($B \cup I \not\models C$). If the query fails, C is true in $M(B \cup I)$ ($B \cup I \models C$)

Learning from Int. with Background

Given

- a space of possible clausal theories $\mathcal{H}$
- a set P of interpretations
- a set N of interpretations
- a background theory B

Find: a clausal theory $H \in \mathcal{H}$ such that

- for all $p \in P$, $B \cup p \models H$
- for all $n \in N$, $B \cup n \not\models H$

Generality Relation

- $cover(\{C\}, e) = true$ if $e \models C$
- $C \geq D \Rightarrow C \models D \Rightarrow D$ is more general than C
- the relation is reversed
- Example:

$false \leftarrow true$
$false \leftarrow gorilla(X)$
$female(X) \leftarrow gorilla(X)$
$female(X) \vee male(X) \leftarrow gorilla(X)$

Dipartimento di Matematica e Informatica

ICL [De Raedt, Van Laer, 95]

- Dual version of a top down entailment algorithm:
  - coverage loop is performed on negative examples
- Updates CN2 to first order

ICL(P, N, B)
$H := \emptyset$
repeat
    $C :=$FindBestClause(P, N, B)
    if $C \neq$ *null* then
        add C to H
        remove from N all interpretations that are false for C
until $C =$ *null* or N is empty
return H

Dipartimento di Matematica e Informatica

ICL FindBestClause

FindBestClause(P, N, B)
$Beam := \{false \leftarrow true\}$, $BestClause := null$
while $Beam$ is not empty do
    $NewBeam := \emptyset$
    for each clause C in $Beam$ do
        for each refinement Ref of C do
            if Ref is better than $BestClause$ and Ref is
                statistically significant then
                $BestClause := Ref$
            if Ref is not to be pruned then
                add Ref to $NewBeam$
                if size of $NewBeam > MaxBeamSize$ then
                    remove worst clause from $NewBeam$
    $Beam := NewBeam$
return $BestClause$

Dipartimento di Matematica e Informatica

ICL Heuristics

- $n(\overline{C})$= number of interpretations (positive and negative) where C is false
- $n^-(\overline{C})$= number of negative interpretation where C is false
- $H(C) = p(-|\overline{C}) = \frac{n^-(\overline{C})+1}{n(\overline{C})+2}$ = precision over negative class

Process Mining

- Every organization performs a number of business processes in order to achieve its mission.
- Information system can log all the actions performed in a process
- Problem: how to mine a process model from the log
- Research area: process mining, e.g. [Agrawal et al. 1998], [Aalst et al. 2003] [Greco et al. 2006]
- We propose:
  - A novel representation language for describing process models.
  - An approach to Process Mining that uses learning from interpretations techniques from ILP.

Dipartimento di Matematica e Informatica

Processes

- A *process trace* t is a sequence of events or a list of time-stamped events.
- An example of a trace is
  $\langle a, b, c \rangle$
- A *process model* PM is a formula in a language.
- Interpreter of the language:
  - $t \models PM$: compliant trace
  - $t \not\models PM$: non compliant trace
- A bag of process traces L is called a *log*.

Dipartimento di Matematica e Informatica

Traces Representation with Logic Programming

- A trace can be represented as an interpretation:
  - Each event is represented with an atom whose predicate is the event type .
  - An extra argument is added to the atom indicating the position in the sequence or its time.
- For example, the trace: a, b, c
  can be represented with the interpretation
  $\{a(1), b(2), c(3)\}$.
- Background knowledge: a normal logic program B.
- $M(B \cup t)$ rather than simply t

Models Representation

- Subset of the $\mathcal{S}$CIFF language [Alberti et al. 2007] for specifying and verifying interaction in open agent societies.
- A process model is a set of integrity constraints
- Integrity Constraint (IC):

$$Body \rightarrow \exists(ConjP_1) \vee \ldots \vee \exists(ConjP_n) \vee \forall\neg(ConjN_1) \vee \ldots \vee \forall\neg(ConjN_m)$$

Dipartimento di Matematica e Informatica

Models Representation

- *Body*, $ConjP_i$ and $ConjN_j$: conjunctions of literals
- The variables of the body are implicitly universally quantified with scope the entire formula.
- The quantifiers in the head apply to all the variables not appearing in the body.
- $\exists(ConjP_i)$: *P disjunct*
- $\forall\neg(ConjN_j)$: *N disjunct*

IC Example

$$\begin{aligned} & a(bob, T), T < 10 \\ \rightarrow & \exists T1(b(alice, T1), T < T1) \\ & \vee \\ & \forall T1 \neg (c(mary, T1), T < T1, T1 < T + 10) \end{aligned} \tag{1}$$

- Meaning: if *bob* has executed action *a* at a time $T < 10$, then *alice* must execute action *b* at a time $T1$ later than T or *mary* must not execute action *c* for 9 time units after T.

Truth of an IC

- An IC C is true in an interpretation $M(B \cup t)$, written $M(B \cup t) \models C$, if, for every substitution θ for which $Body$ is true in $M(B \cup t)$, there exists a disjunct $\exists(ConjP_i)$ or $\forall\neg(ConjN_j)$ that is true in $M(B \cup t)$.

Test of the Truth of an IC

- Similarly to disjunctive clauses, the truth of an IC in an interpretation $M(B \cup t)$ can be tested by running the query:

$$? - Body, not(ConjP_1), \ldots not(ConjP_n), \\ not(not(ConjN_1)), \ldots, not(not(ConjN_m))$$

  in a database containing the clauses of B and atoms of t as facts.
- If the query fails, the IC is true in the interpretation.
- If the query succeeds, the IC is false in the interpretation.

Example

- For the example above we have the query

$$?- a(bob, T), T < 10, not(b(alice, T1), T < T1), \\ not(not(c(mary, T1), T < T1, T1 < T + 10))$$

- fails in $\{a(bob, 2), b(alice, 3)\}$
- fails in $\{a(bob, 2), c(mary, 14)\}$
- succeeds in $\{a(bob, 2), c(mary, 6)\}$
- succeeds in $\{a(bob, 2), b(alice, 1), c(mary, 6)\}$

Truth of a Theory

- A process model H is true in an interpretation $M(B \cup t)$ if every IC is true in it and we write $M(B \cup t) \models H$.

Learning Problem

- Adaptation to ICs of the learning from interpretation setting of ILP:
- **Given**
  - a space of possible process models $\mathcal{H}$
  - a set I^+ of positive traces;
  - a set I^- of negative traces;
  - a background theory B.

  Find: a process model $H \in \mathcal{H}$ such that
  - for all $i^+ \in I^+$, $M(B \cup i^+) \models H$;
  - for all $i^- \in I^-$, $M(B \cup i^-) \not\models H$;
- If $M(B \cup i) \models C$ we say that IC C covers the trace i
- if $M(B \cup i) \not\models C$ we say that C rules out the trace i.

Learning IC Theories

Definition (Subsumption)

An IC D subsumes an IC C, written $D \geq C$, iff it exists a substitution θ for the variables in the body of D or in the N conjunctions of D such that

- $Body(D)\theta \subseteq Body(C)$ and
- $\forall ConjP(D) \in HeadSet(D),\ \exists ConjP(C) \in HeadSet(C)$: $ConjP(C) \subseteq ConjP(D)\theta$ and
- $\forall ConjN(D) \in HeadSet(D),\ \exists ConjN(C) \in HeadSet(C)$: $ConjN(D)\theta \subseteq ConjN(C)$

Dipartimento di Matematica e Informatica

Learning IC Theories

Theorem

$D \geq C \Rightarrow D \models C.$

- Thus C is more general than D.

Language Bias

- Set of IC templates. Each template specifies
  - a set of literals *BS* allowed in the body,
  - a set of disjuncts *HS* allowed in the head. For each disjunct, the template specifies:
    - whether it is a *P* or an *N* disjunct,
    - the set of literals allowed in the disjunct.

Refinement Operator

- Given an IC D, the set of refinements $\rho(D)$ of D is obtained by performing one of the following operations
  - adding a literal from the IC template for D to the body;
  - adding a disjunct from the IC template for D to the head: the disjunct can be
    - $\exists d_1 \wedge \ldots \wedge d_k$ where $\{d_1, \ldots, d_k\}$ is the set of literals allowed by the IC template for D for the P disjunct,
    - $\forall \neg d$ where d is allowed by the IC template for D for a N disjunct;
  - removing a literal from a P disjunct in the head;
  - adding a literal to an N disjunct in the head. The literal must be allowed by the language bias.

Learning Algorithm

- DPML (Declarative Process Model Learner), an adaptation of ICL.

function DPML(I^+, I^-, B)
initialize $H := \emptyset$
do
    $C :=$ FindBestIC(I^+, I^-, B)
    if $C \neq \emptyset$ then
        add C to H
        remove from I^- all interpretations that are false for C
while $C \neq \emptyset$ and I^- is not empty
return H

Learning Algorithm

- **function** FindBestIC(I^+, I^-, B)
  initialize *Beam* := $\{false \leftarrow true\}$
  initialize *BestIC* := $\emptyset$
  while *Beam* is not empty do
    initialize *NewBeam* := $\emptyset$
    for each IC *C* in *Beam* do
      for each refinement *Ref* of *C* do
        if *Ref* is better than *BestIC* then *BestIC* := *Ref*
        if *Ref* is not to be pruned then
          add *Ref* to *NewBeam*
          if size of *NewBeam* $>$ *MaxBeamSize* then
            remove worst clause from *NewBeam*
    Beam := *NewBeam*
  return *BestIC*

Learning Algorithm

- FindBestIC: beam search with $p(\ominus|\overline{C})$ as a heuristic function,
- $p(\ominus|\overline{C})$ is the probability that an input trace is negative given that is ruled out by the IC C.
- $p(\ominus|\overline{C}) = \frac{n(\ominus|\overline{C})}{n(\ominus|\overline{C})+n(\oplus|\overline{C})}$

Learning Algorithm

- Differently from ICL we do not perform pruning of the ICs that are not statistically significant

Combining Logic and Probability

- Logic does not handle well uncertainty
- Graphical models do not handle well relationships among entities
- Solution: combine the two
- Many approaches proposed in the areas of Logic Programming, Uncertainty in AI, Machine Learning, Databases, Knowledge Representation

Dipartimento di Matematica e Informatica

Probabilistic Logic Programming

- Distribution Semantics [Sato ICLP95]
- A probabilistic logic program defines a probability distribution over normal logic programs (called instances or possible worlds or simply worlds)
- The distribution is extended to a joint distribution over worlds and interpretations (or queries)
- The probability of a query is obtained from this distribution

Dipartimento di Matematica e Informatica

Probabilistic Logic Programming (PLP) Languages under the Distribution Semantics

- Probabilistic Logic Programs [Dantsin RCLP91]
- Probabilistic Horn Abduction [Poole NGC93], Independent Choice Logic (ICL) [Poole AI97]
- PRISM [Sato ICLP95]
- Logic Programs with Annotated Disjunctions (LPADs) [Vennekens et al. ICLP04]
- ProbLog [De Raedt et al. IJCAI07]
- They differ in the way they define the distribution over logic programs

PLP Online

- `http://cplint.eu`
  - Inference (knwoledge compilation, Monte Carlo)
  - Parameter learning (EMBLEM)
  - Structure learning (SLIPCOVER)
- `https://dtai.cs.kuleuven.be/problog/`
  - Inference (knwoledge compilation, Monte Carlo)
  - Parameter learning (LFI-ProbLog)

Logic Programs with Annotated Disjunctions

`http://cplint.eu/e/sneezing_simple.pl`

$$sneezing(X) : 0.7 \ ; \ null : 0.3 \leftarrow flu(X).$$
$$sneezing(X) : 0.8 \ ; \ null : 0.2 \leftarrow hay\_fever(X).$$
$$flu(bob).$$
$$hay\_fever(bob).$$

- Distributions over the head of rules
- *null* does not appear in the body of any rule
- Worlds obtained by selecting one atom from the head of every grounding of each clause

Dipartimento di Matematica e Informatica

ProbLog

$sneezing(X) \leftarrow flu(X), flu\_sneezing(X).$
$sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).$
$flu(bob).$
$hay\_fever(bob).$
$0.7 :: flu\_sneezing(X).$
$0.8 :: hay\_fever\_sneezing(X).$

- Distributions over facts
- Worlds obtained by selecting or not every grounding of each probabilistic fact

Distribution Semantics

- Case of no function symbols: finite Herbrand universe, finite set of groundings of each switch/clause/fact
- Atomic choice: selection of the i-th atom for grounding $C\theta$ of switch/clause C
  - represented with the triple (C, θ, i)
- Example $C_1 = sneezing(X) : 0.7 \; ; \; null : 0.3 \leftarrow flu(X).$, $(C_1, \{X/bob\}, 1)$
- A ProbLog fact $p :: F$ is interpreted as $F : p \vee null : 1 - p$.

Distribution Semantics

- Selection σ: a total set of atomic choices (one atomic choice for every grounding of each clause)
- A selection σ identifies a logic program w_σ called world
- The probability of w_σ is $P(w_\sigma) = \prod_{(C,\theta,i)\in\sigma} P_0(C,i)$
- Finite set of worlds: $W_T = \{w_1, \ldots, w_m\}$
- $P(w)$ distribution over worlds: $\sum_{w\in W_T} P(w) = 1$

Distribution Semantics

- Ground query Q
- $P(Q|w) = 1$ if Q is true in w and 0 otherwise
- $P(Q) = \sum_w P(Q, w) = \sum_w P(Q|w)P(w) = \sum_{w \models Q} P(w)$
- You can see $P(Q)$ as the probability that Q is true in a world sampled at random from $P(w)$
  - for each choice, sample a value to get a world
  - test the query in the world

Example Program (LPAD) Worlds

`http://cplint.eu/e/sneezing_simple.pl`

$sneezing(bob) \leftarrow flu(bob).$
$sneezing(bob) \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_1) = 0.7 \times 0.8$

$null \leftarrow flu(bob).$
$sneezing(bob) \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_2) = 0.3 \times 0.8$

$sneezing(bob) \leftarrow flu(bob).$
$null \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_3) = 0.7 \times 0.2$

$null \leftarrow flu(bob).$
$null \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_4) = 0.3 \times 0.2$

$$P(Q) = \sum_{w \in W_{\mathcal{T}}} P(Q, w) = \sum_{w \in W_{\mathcal{T}}} P(Q|w)P(w) = \sum_{w \in W_{\mathcal{T}}: w \models Q} P(w)$$

- $sneezing(bob)$ is true in 3 worlds
- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$

Dipartimento di Matematica e Informatica

Example Program (ProbLog) Worlds

- 4 worlds

$sneezing(X) \leftarrow flu(X), flu\_sneezing(X).$
$sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).$
$flu(bob).$
$hay\_fever(bob).$

$flu\_sneezing(bob).$
$hay\_fever\_sneezing(bob).$ $hay\_fever\_sneezing(bob).$
$P(w_1) = 0.7 \times 0.8$ $P(w_2) = 0.3 \times 0.8$
$flu\_sneezing(bob).$
$P(w_3) = 0.7 \times 0.2$ $P(w_4) = 0.3 \times 0.2$

- $sneezing(bob)$ is true in 3 worlds
- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$

Logic Programs with Annotated Disjunctions

`http://cplint.eu/e/sneezing.pl`

$strong\_sneezing(X) : 0.3 \ ; \ moderate\_sneezing(X) : 0.5 \leftarrow flu(X).$
$strong\_sneezing(X) : 0.2 \ ; \ moderate\_sneezing(X) : 0.6 \leftarrow hay\_fever(X).$
$flu(bob).$
$hay\_fever(bob).$

- 9 worlds
- $strong\_sneezing(bob)$ is true in 5
- $P(strong\_sneezing(bob)) =$
  $0.3 \cdot 0.2 + 0.3 \cdot 0.6 + 0.3 \cdot 0.2 + 0.5 \cdot 0.2 + 0.2 \cdot 0.2 = 0.44$

Monty Hall Puzzle

- A player is given the opportunity to select one of three closed doors, behind one of which there is a prize.
- Behind the other two doors are empty rooms.
- Once the player has made a selection, Monty is obligated to open one of the remaining closed doors which does not contain the prize, showing that the room behind it is empty.
- He then asks the player if he would like to switch his selection to the other unopened door, or stay with his original choice.
- Does it matter if he switches?

Monty Hall Puzzle

http://cplint.eu/e/monty.swinb

```
:- use_module(library(pita)).
:- endif.
:- pita.
:- begin_lpad.
prize(1):1/3; prize(2):1/3; prize(3):1/3.

open_door(2):0.5 ; open_door(3):0.5:- prize(1).
open_door(2):- prize(3).
open_door(3):- prize(2).

win_keep:- prize(1).

win_switch:-
  prize(2),
  open_door(3).

win_switch:-
  prize(3),
  open_door(2).
:- end_lpad.
```

Examples

Throwing coins `http://cplint.eu/e/coin.swinb`

```
heads(Coin):1/2 ; tails(Coin):1/2 :-
  toss(Coin),\+biased(Coin).
heads(Coin):0.6 ; tails(Coin):0.4 :-
  toss(Coin),biased(Coin).
fair(Coin):0.9 ; biased(Coin):0.1.
toss(coin).
```

Russian roulette with two guns `http://cplint.eu/e/trigger.pl`

```
death:1/6 :- pull_trigger(left_gun).
death:1/6 :- pull_trigger(right_gun).
pull_trigger(left_gun).
pull_trigger(right_gun).
```

Dipartimento di Matematica e Informatica

Examples

Mendel's inheritance rules for pea plants

`http://cplint.eu/e/mendel.pl`

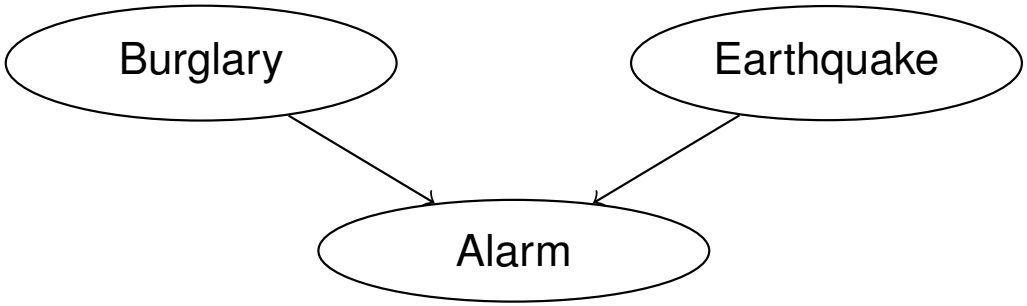
```
color(X,purple):-cg(X,_A,p).
color(X,white):-cg(X,1,w),cg(X,2,w).
cg(X,1,A):0.5 ; cg(X,1,B):0.5 :-
  mother(Y,X),cg(Y,1,A),cg(Y,2,B).
cg(X,2,A):0.5 ; cg(X,2,B):0.5 :-
  father(Y,X),cg(Y,1,A),cg(Y,2,B).
```

Probability of paths `http://cplint.eu/e/path.swinb`

```
path(X,X).
path(X,Y):-path(X,Z),edge(Z,Y).
edge(a,b):0.3.
edge(b,c):0.2.
edge(a,c):0.6.
```

Dipartimento di Matematica e Informatica

Encoding Bayesian Networks



alarm	t	f
b=t,e=t	1.0	0.0
b=t,e=f	0.8	0.2
b=f,e=t	0.8	0.2
b=f,e=f	0.1	0.9

burg	t	f
	0.1	0.9

earthq	t	f
	0.2	0.8

http://cplint.eu/e/alarm.pl

```
burg(t):0.1 ; burg(f):0.9.
earthq(t):0.2 ; earthq(f):0.8.
alarm(t):-burg(t),earthq(t).
alarm(t):0.8 ; alarm(f):0.2:-burg(t),earthq(f).
alarm(t):0.8 ; alarm(f):0.2:-burg(f),earthq(t).
alarm(t):0.1 ; alarm(f):0.9:-burg(f),earthq(f).
```

Dipartimento di Matematica e Informatica

Expressive Power

- All languages under the distribution semantics have the same expressive power
- LPADs have the most general syntax
- There are transformations that can convert each one into the others

Function Symbols

- What if function symbols are present?
- Infinite, countable Herbrand universe
- Infinite, countable Herbrand base
- Infinite, countable grounding of the program T
- Uncountable W_T
- Each world infinite, countable
- $P(w) = 0$
- Semantics not well-defined

Game of dice

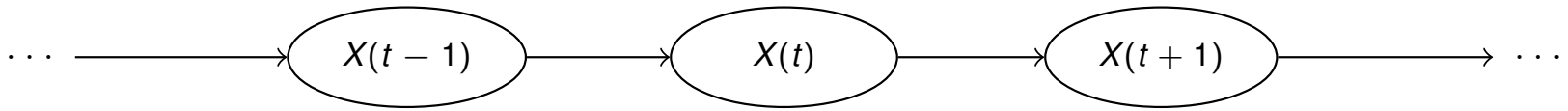
http://cplint.eu/e/threesideddice.pl

```
on(0,1):1/3 ; on(0,2):1/3 ; on(0,3):1/3.
on(T,1):1/3 ; on(T,2):1/3 ; on(T,3):1/3 :-
  T1 is T-1, T1>=0, on(T1,F), \+ on(T1,3).
```

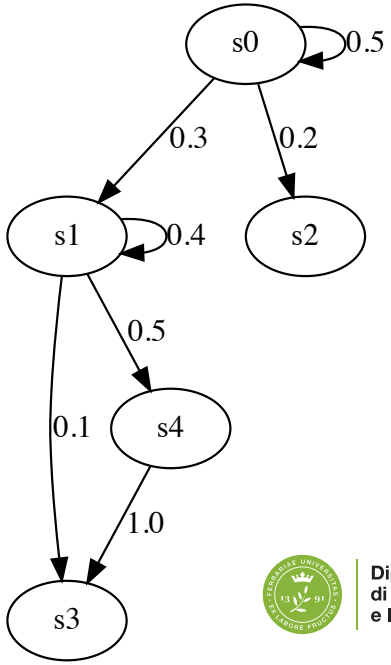
Markov chain

- Model checking: we want to know what is the likelihood that on an execution of the chain from a start state s, a final state t is reached

http://cplint.eu/e/markov_chain.pl



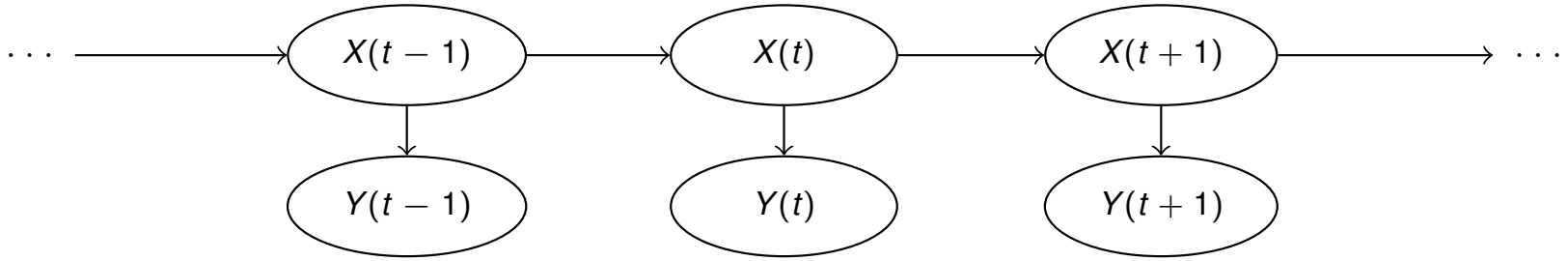

```
% reach(S, I, T) starting at state S at instance I,
% state T is reachable.
reach(S, I, T) :-
  trans(S, I, U),
  reach(U, next(I), T).
reach(S, _, S).
% trans(S,I,T) transition from S at instance I goes to T
trans(s0,S,s0):0.5; trans(s0,S,s1):0.3; trans(s0,S,s2):0.2.
trans(s1,S,s1):0.4; trans(s1,S,s3):0.1; trans(s1,S,s4):0.5.
trans(s4,_,s3).
```



Dipartimento di Matematica e Informatica

Hidden Markov Models

http://cplint.eu/e/hmm.pl




```
hmm(S,O):-hmm(q1,[],S,O).
hmm(end,S,S,[]).
hmm(Q,S0,S,[L|O]):-
  Q\= end,
  next_state(Q,Q1,S0),
  letter(Q,L,S0),
  hmm(Q1,[Q|S0],S,O).
next_state(q1,q1,_S):1/3;next_state(q1,q2_,_S):1/3;
  next_state(q1,end,_S):1/3.
next_state(q2,q1,_S):1/3;next_state(q2,q2,_S):1/3;
  next_state(q2,end,_S):1/3.
letter(q1,a,_S):0.25;letter(q1,c,_S):0.25;
  letter(q1,g,_S):0.25;letter(q1,t,_S):0.25.
letter(q2,a,_S):0.25;letter(q2,c,_S):0.25;
  letter(q2,g,_S):0.25;letter(q2,t,_S):0.25.
```

Dipartimento di Matematica e Informatica

Reasoning Tasks

- Inference: we want to compute the probability of a query given the model and, possibly, some evidence, or find assignments of the random variables with the highest probability
- Weight learning: we know the structural part of the model (the logic formulas) but not the numeric part (the weights) and we want to infer the weights from data
- Structure learning we want to infer both the structure and the weights of the model from data

Inference for PLP under DS

- EVID: compute an unconditional probability $P(e)$, the probability of evidence (also query in this case).
- COND: compute the conditional probability distribution of the query given the evidence, i.e. compute $P(q|e)$
- MPE or *most probable explanation*: find the most likely value of all non-evidence atoms given the evidence, i.e. solving the optimization problem $\arg\max_q P(q|e)$
- MAP or *maximum a posteriori*: find the most likely value of a set of non-evidence atoms given the evidence, i.e. finding $\arg\max_q P(q|e)$. MPE is a special case of MAP where $Q \cup E = H_T$.
- DISTR: compute the probability distribution or density of the non-ground arguments of a conjunction of literals q, e.g., computing the probability density of X in goal $mix(X)$ of the Gaussian mixture

Dipartimento di Matematica e Informatica

Weight Learning

- Given
  - model: a probabilistic logic model with unknown parameters
  - data: a set of interpretations
- Find the values of the parameters that maximize the probability of the data given the model
- Discriminative learning: maximize the conditional probability of a set of outputs (e.g. ground instances for a predicate) given a set of inputs
- Alternatively, the data are queries for which we know the probability: minimize the error in the probability of the queries that is returned by the model

Dipartimento di Matematica e Informatica

Structure Learning

- Given
  - language bias: a specification of the search space
  - data: a set of interpretations
- Find the formulas and the parameters that maximize the likelihood of the data given the model
- Discriminative learning: again maximize the conditional likelihood of a set of outputs given a set of inputs

Inference for PLP under DS

- EVID
- Knowledge compilation:
  - compile the program to an intermediate representation
    - Binary Decision Diagrams (BDD) (ProbLog [De Raedt et al. IJCAI07], `cplint` [Riguzzi AIIA07,Riguzzi LJIGPL09], PITA [Riguzzi & Swift ICLP10])
    - deterministic, Decomposable Negation Normal Form circuit (d-DNNF) (ProbLog2 [Fierens et al. TPLP15])
    - Sentential Decision Diagrams
  - compute the probability by weighted model counting

Inference for PLP under DS

- Bayesian Network based:
  - Convert to BN
  - Use BN inference algorithms (CVE [Meert et al. ILP09])
- Lifted inference

Knowledge Compilation

- Assign Boolean random variables to the probabilistic rules
- Given a query Q, compute its explanations, assignments to the random variables that are sufficient for entailing the query
- Let K be the set of all possible explanations
- Build a Boolean formula $F(Q)$
- Build a BDD representing $F(Q)$

Approximate Inference

- Inference problem is #P hard
- For large models inference is intractable
- Approximate inference
  - Monte Carlo: draw samples of the truth value of the query
  - Iterative deepening: gives a lower and an upper bound
  - Compute only the best k explanations: branch and bound, gives a lower bound

Parameter Learning

- Problem: given a set of interpretations, a program, find the parameters maximizing the likelihood of the interpretations (or of instances of a target predicate)
- The interpretations record the truth value of ground atoms, not of the choice variables
- Unseen data: relative frequency can't be used

Parameter Learning

- An Expectation-Maximization algorithm must be used:
  - Expectation step: the distribution of the unseen variables in each instance is computed given the observed data
  - Maximization step: new parameters are computed from the distributions using relative frequency
  - End when likelihood does not improve anymore

EMBLEM

- EM over Bdds for probabilistic Logic programs Efficient Mining
- Input: an LPAD; logical interpretations (data); *target* predicate(s)
- All ground atoms in the interpretations for the target predicate(s) correspond to as many queries
- BDDs encode the explanations for each query Q
- Expectations computed with two passes over the BDDs

Structure Learning for LPADs

- Given a trivial LPAD or an empty one, a set of interpretations (data)
- *Find the model and the parameters* that maximize the probability of the data (log-likelihood)
- SLIPCOVER: Structure LearnIng of Probabilistic logic program by searching OVER the clause space EMBLEM [Riguzzi & Bellodi TPLP 2015]
  1. Beam search in the space of clauses to find the promising ones
  2. Greedy search in the space of probabilistic programs guided by the LL of the data.
- *Parameter learning* by means of EMBLEM

SLIPCOVER

- Cycle on the set of predicates that can appear in the head of clauses, either target or background
- For each predicate, beam search in the space of clauses
- The initial set of beams is generated by building a set of *bottom clauses* as in Progol [Muggleton NGC 1995]

SLIPCOVER

- The initial beam associated with predicate P/Ar will contain the clause with the empty body $h : 0.5.$ for each bottom clause $h :- b_1, \ldots, b_m$
- In each iteration of the cycle over predicates, it performs a beam search in the space of clauses for the predicate.
- The beam contains couples $(Cl, LIterals)$ where $Literals = \{b_1, \ldots, b_m\}$
- For each clause Cl of the form $Head :- Body$, the refinements are computed by adding a literal from $Literals$ to the body.

SLIPCOVER

- The tuple (Cl', $Literals'$) indicates a refined clause Cl' together with the new set $Literals'$
- EMBLEM is then executed for a theory composed of the single refined clause.
- LL is used as the score of the updated clause (Cl'', $Literals'$).
- (Cl'', $Literals'$) is then inserted into a list of promising clauses.
- Two lists are used, *TC* for target predicates and *BC* for background predicates.
- These lists ave a maximum size

SLIPCOVER

- After the clause search phase, SLIPCOVER performs a greedy search in the space of theories:
  - it starts with an empty theory and adds a target clause at a time from the list *TC*.
  - After each addition, it runs EMBLEM and computes the LL of the data as the score of the resulting theory.
  - If the score is better than the current best, the clause is kept in the theory, otherwise it is discarded.
- Finally, SLIPCOVER adds all the clauses in *BC* to the theory and performs parameter learning on the resulting theory.

Event Calculus

- [Kowalski Sergot 1986] is a logic programming formalism for representing the effects of events on properties
- Event: an action that may occur in the world, e.g., a person who is arriving in the kitchen
- Fluent: time-varying property that might be the effect of an event; for example, e.g. a person is located in the kitchen (after arriving in the kitchen)
- Time point: an instant of time and indicates when an event happens or when a fluent holds

Dipartimento di Matematica e Informatica

Event Calculus

Simple EC [Shanahan 1999]

- `initiates(a,b,t)` Fluent b starts to hold after action a at time t
- `terminates(a,b,t)` Fluent b ceases to hold after action a at time t
- `initially(b)` Fluent b holds from time 0
- `t1 < t2` Time point t1 is before time point t2
- `happens(a,t)` Action a occurs at time t
- `holdsAt(b,t)` Fluent b holds at time t
- `clipped(t1,b,t2)` Fluent b is terminated between times t1 and t2

Event Calculus Axioms

```
holdsAt(F, T) :-
  initially(F),
  \+ clipped(0, F, T).
holdsAt(F, T2) :-
  initiates(F, T1),
  T1 < T2 ,
  \+ clipped(T1 ,F, T2 ).
clipped(T1 ,F, T3) :-
  terminates(F, T2),
  T1 < T2 , T2 < T3.
% effect axioms
initiates(<fluent> T) :-
  happens(<event>, T),
  <conditions>.
terminates(<fluent>, T) :-
  happens(<event>, T),
  <conditions>.
initially(<fluent>).
```

Example

```
initiates(locatedIn(A, B), T) :-
  happens(arrive(A, B), T).
terminates(locatedIn(A, D), T) :-
  happens(arrive(A, B), T),
  B \= D.
initially(locatedIn(bob, garden)).
happens(arrive(bob, kitchen), 3).
happens(arrive(bob, garage), 5).
```

- `holdsAt(locatedIn(bob,garden),2)` true
- `holdsAt(locatedIn(bob,garden),4)` false
- `holdsAt(locatedIn(bob,kitchen),4)` true
- `holdsAt(locatedIn(bob,garage),6)` true

`http://cplint.eu/p/event_calculus.pl`

Dipartimento di Matematica e Informatica

Probabilistic Event Calculus

```
initiates(locatedIn(A, B), T):0.66 :-
  happens(arrive(A, B), T).
terminates(locatedIn(A, D), T):0.66 :-
  happens(arrive(A, B), T),
  B \= D.
initially(locatedIn(bob, garden)).
happens(arrive(bob, kitchen), 3).
happens(arrive(bob, garage), 5).
```

- `holdsAt(locatedIn(bob,garden),2)` 1.0
- `holdsAt(locatedIn(bob,garden),4)` 0.34
- `holdsAt(locatedIn(bob,kitchen),4)` 0.66
- `holdsAt(locatedIn(bob,garage),6)` 0.66

`http://cplint.eu/p/prob_event_calculus.pl`

Dipartimento di Matematica e Informatica

Probabilistic Event Calculus

You may also have probabilities on events

```
initiates(locatedIn(A, B), T):0.66 :-
  happens(arrive(A, B), T).
terminates(locatedIn(A, D), T):0.66 :-
  happens(arrive(A, B), T),
  B \= D.
initially(locatedIn(bob, garden)).
happens(arrive(bob, kitchen), 3):0.95.
happens(arrive(bob, garage), 5):0.99.
```

- `holdsAt(locatedIn(bob,garden),2)` 1.0
- `holdsAt(locatedIn(bob,garden),4)` 0.373
- `holdsAt(locatedIn(bob,kitchen),4)` 0.627
- `holdsAt(locatedIn(bob,garage),6)` 0.6534

`http://cplint.eu/p/prob_event_calculus2.pl`

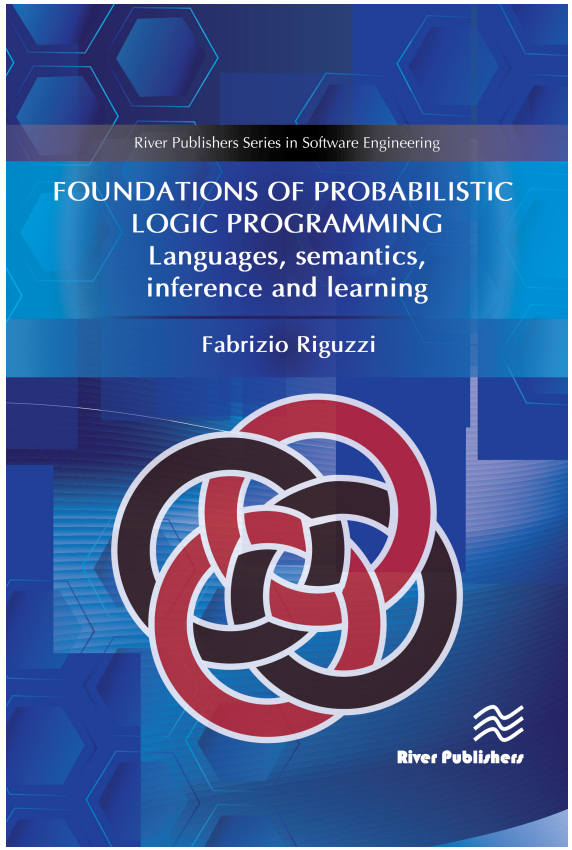
Dipartimento di Matematica e Informatica

Learning Probabilistic Event Calculus

- Given a number of histories including the values of fluents and the events at each time
- Find the effect axioms [Schwitter 2018]
- Problem: no examples of `initiates/2` and `terminates/2`
- Solution: generate them from the histories.
- For a fluent that initiates, a positive example for `initiates/2`
- For a fluent that terminates, a negative example for `terminates/2`
- For a fluent that stays the same, a negative example for `initiates/2` and `terminates/2`

Conclusions

- ILP suitable for temporal data
- PLP useful to deal with uncertainty
- Parameter learning
- Structure learning
- Research directions:
  - Dealing with intervals
  - Learning restricted and cheaper languages



Resources

- Online course on cplint
  - Moodle https://edu.swi-prolog.org/
  - Videos of lectures https://www.youtube.com/playlist?list=PLJPXEH0boeND0UGWJxBRWs7qzzKpC-FkN
- ACAI summer school on Statistical Relational AI http://acai2018.unife.it/
- Videos of lectures https://www.youtube.com/playlist?list=PLJPXEH0boeNDWTNwWTWnVffXi5XwAj1mb
- Videos of lecture Probabilistic Inductive Logic Programming
  - Part 1 https://youtu.be/mLdPGSlgNxU
  - Part 2 https://youtu.be/DRlOft0Y_Ng
- cplint in Playing with Prolog https://www.youtube.com/playlist?list=PLJPXEH0boeNAik6QnfvGlAGRQxFY_LCE3

Dipartimento di Matematica e Informatica

THANKS FOR LISTENING AND ANY QUESTIONS ?

Dipartimento di Matematica e Informatica

References

- Bellodi, E. and Riguzzi, F. (2012). Learning the structure of probabilistic logic programs. In Inductive Logic Programming 21st International Conference, ILP 2011, London, UK, July 31 - August 3, 2011. Revised Papers, volume 7207 of LNCS, pages 61-75, Heidelberg, Germany. Springer.
- Bellodi, E. and Riguzzi, F. (2013). Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intelligent Data Analysis, 17(2).
- Gutmann, B., Thon, I., and Raedt, L. D. (2011). Learning the parameters of probabilistic logic programs from interpretations. In European Conference on Machine Learning and Knowledge Discovery in Databases, volume 6911 of LNCS, pages 581-596. Springer.

Dipartimento di Matematica e Informatica

References

- Muggleton, S. (1995). Inverse entailment and progol. New Generation Comput., 13(3&4):245-286.
- Riguzzi, F. (2007). A top down interpreter for LPAD and CP-logic. In Congress of the Italian Association for Artificial Intelligence, number 4733 in LNAI, pages 109-120. Springer.
- Riguzzi, F. (2009). Extended semantics and inference for the Independent Choice Logic. Logic Journal of the IGPL.
- Riguzzi, F. and Swift, T. (2010). Tabling and Answer Subsumption for Reasoning on Logic Programs with Annotated Disjunctions. In Hermenegildo, M. and Schaub, T., editors, Technical Communications of the 26th Int'l. Conference on Logic Programming (ICLP10), volume 7 of Leibniz International Proceedings in Informatics (LIPIcs), pages 162-171, Dagstuhl, Germany. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Dipartimento di Matematica e Informatica

References

- Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In International Conference on Logic Programming, pages 715-729.
- Thon, I., Landwehr, N., and Raedt, L. D. (2008). A simple model for sequences of relational state descriptions. In Daelemans, W., Goethals, B., and Morik, K., editors, Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II, volume 5212 of Lecture Notes in Computer Science, pages 506-521. Springer.
- Robert Kowalski and Marek Sergot. A Logic-Based Calculus of Events. In New Generation Computing, Vol. 4, pp. 67–95, 1986.
- Murray Shanahan. The Event Calculus Explained. In M.J. Wooldridge and M. Veloso (eds), Artificial Intelligence Today, LNAI, Vol. 1600, Springer, pp. 409–430, 1999.

Dipartimento
di Matematica
e Informatica

References

- Lamma, E., Mello, P., Riguzzi, F., Storari, S. (2007, June). Applying inductive logic programming to process mining. In International Conference on Inductive Logic Programming (pp. 132-146). Springer, Berlin, Heidelberg.
- Schwitter, Rolf. "Learning effect axioms via probabilistic logic programming." Technical Communications of the 33rd International Conference on Logic Programming (ICLP 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Dipartimento di Matematica e Informatica