Supportive Textbooks in Course

Semantics of Knowledge Systems
(UINF/SZS1/04)
Study programme: Informatics
Teacher: RNDr. Tomáš Horváth
Institute of Computer Science
Pavol Jozef Šafárik University in Košice

The ESF project[1] no. SOP HR 2005/NP1-051 11230100466

[1]The project is cofinanced with the support of the European Union

About the subject

- Duration: 2+1 hours per week, 28+14 hours per study
- Guarant: prof. RNDr. Villiam Geffert, DrSc.
- Assessment: *Examinaiton*
- Final examination: *Written test*
- Requirements:
  - At least 80% presence on practices during the semester
  - At least 60% success on written test at final examination
- Course objective:
  - To introduce basic concepts of logic programming and inductive logic programming.
  - To acquire knowledge on fuzzy logic programming from the deductive and inductive point of view.

Syllabus of the subject

- Important notions from classical logic
  - Deduction
    - Resolution, Derivation, Deduction
    - Subsumption theorem
    - Refutation theorem
  - Induction
    - Examples, Background Knowledge, Hypothesis
    - Learning task
    - Hypothesis space

Syllabus of the subject

- Motivation for fuzzy logic
    - truth values, many-valued operators, ...
- Truth functional fuzzy logic deduction
    - Language of fuzzy logic programming
    - computation rules
    - fuzzy Herbrand interpretations and models
    - computed answer, correct answer
    - T_P operator and the fixpoint semantics
    - soundness and completeness of fuzzy logic programming
- Induction in fuzzy logic
    - Examples, Background Knowledge, Hypothesis
    - The learning task
    - Generalized Annotated Programs
    - Truth completion

Language

Our language consist of

- Attributes $\mathcal{A}_1, \ldots, \mathcal{A}_m \in \mathcal{A}$ with domains $\mathcal{D}_1, \ldots, \mathcal{D}_m \in \mathcal{D}$.
  For each attribute $\mathcal{A}_i$ we have
  - Variables $\mathcal{V}^{\mathcal{A}_i}$
  - Constants $\mathcal{C}^{\mathcal{A}_i}$
- Functions $f(\mathcal{A}_1, \ldots, \mathcal{A}_n) \in \mathcal{F}$
- Predicates $p(\mathcal{A}_1, \ldots, \mathcal{A}_o) \in \mathcal{P}$
  - $p(\mathcal{A}_1, \ldots, \mathcal{A}_o) : \mathcal{D}_{\mathcal{A}_1} \times \ldots \times \mathcal{D}_{\mathcal{A}_o} \mapsto \{0, 1\}$

Language

The frequently used operators of our language are:

- Conjunction $\wedge$
- Disjunction $\vee$
- Implication $\rightarrow$
- Aggregation $@$

A combination of $\wedge, \vee and @$ is again an aggregation.

Truth values and connectives

The set of truth values is the set $\{0,1\}$
The truth functions of the operators are

- Conjunctor $\wedge^\bullet$
  - $\wedge^\bullet(0,0) = \wedge^\bullet(0,1) = \wedge^\bullet(1,0) = 0$
  - $\wedge^\bullet(1,1) = 1$
- Disjunctor $\vee^\bullet$
  - $\vee^\bullet(0,1) = \vee^\bullet(1,0) = \vee^\bullet(1,1) = 1$
  - $\vee^\bullet(0,0) = 0$
- Implicator $\rightarrow^\bullet$
  - $\rightarrow^\bullet (0,0) = \rightarrow^\bullet (0,1) = \rightarrow^\bullet (1,1) = 1$
  - $\rightarrow^\bullet (1,0) = 0$
- Aggregator $@^\bullet$
  - assume $@^\bullet(0,\dots,0) = 0$ and $@^\bullet(1,\dots,1) = 1$

Herbrand structures

Let L be first-order language.

Herbrand universe U_L is the set of all ground terms created from constants and function symbols of the language L. In case that L does not contains constants, we add an arbitrary sign, e.g. "*a*" what enables us to create ground terms.

Herbrand base B_L is the set of all ground atoms created from perdicate symbols of the language L and terms from the Herbrand universe U_L.

Herbrand structures

Let L be first-order language.

Herbrand interpretation f is the function $f : B_L \mapsto \{0, 1\}$

Let f be a Herbrand interpretation and Σ a set of (closed) formulas of L.
If f is a model of Σ then f is a Herbrand model of Σ.

Clauses

A clause (rule) is a formula of the form $\forall x_1, \ldots, \forall x_s (L_1 \vee \ldots \vee L_m)$, where L_i are literals and $x_1, \ldots, x_s$ are variables appering in $L_1 \vee \ldots \vee L_m$.

The notation $\forall x_1, \ldots, \forall x_s (A_1 \vee \ldots \vee A_k \vee \neg B_1 \vee \ldots \vee \neg B_n)$ is equivalent to $\forall x_1, \ldots, \forall x_s (B_1 \wedge \ldots \wedge B_n \rightarrow A_1 \vee \ldots \vee A_k)$. Instead of this complicated notation we will use an abbreviated notation $A_1, \ldots, A_k \leftarrow B_1, \ldots, B_n$.
$A_1, \ldots, A_k$ is the head and $B_1, \ldots, B_n$ is the body of the clause.

Clauses

A program clause is a clause of the form $A \leftarrow B_1, \ldots, B_n$, what contains exactly one literal (A) in the head and zero or more literals in the body.

A program is a set of program clauses.

A ground clause is a clause of the form $A \leftarrow$, i.e. a program clause with an empty body.

A goal is a clause of the form $\leftarrow B_1, \ldots, B_n$, i.e. a program clause with an empty head.

An empty clause is a clause with emty body and empty head. We denote it by $\Box$. A Horn clause is either a program clause or a goal.

Theorems on Herbrand models

Let Σ be a set of clauses and Σ has a model. Then Σ has a Herbrand model, too.

Let Σ be a set of clauses. Then Σ is unsatisfiable iff Σ has no Herbrand model.

Let Σ be a set of clauses. Then Σ has a model iff Σ has a Herbrand model.

Minimal Herbrand model

Let Π be a program and $\left\{M_i\right\}_{i\in I}$ is a nonempty set of Herbrand models of Π. Then $M_\Pi = \bigcap_{i\in I} M_i$ is a Herbrand model of Π.

We call M_Π a minimal herbrand model of Π.

If Π is a program, then $M_\Pi = \{A \in B_\Pi | \Pi \models A\}$

The T_Π operator

Let Π be a program. A function $T_\Pi : B_\Pi \mapsto B_\Pi$ is defined as follows:

Let I be a Herbrand interpretation. Then
$T_\Pi(I) = \{A \in B_\Pi | A \leftarrow A_1, \ldots, A_n$ is a ground instance of a clause in Π and $\{A_1, \ldots, A_n\} \subseteq I\}$

Let Π be a program and I a Herbrand interpretation of Π. Then I is a model of Π iff $T_\Pi(I) \subseteq I$.

Resolution

Let $C_1 = L_1 \vee \ldots \vee L_i \vee \ldots \vee L_m$ and $C_2 = M_1 \vee \ldots \vee M_j \vee \ldots \vee M_n$ be clauses with no common variables. If the substitution θ is the most general unifier (mgu) of $\{L_i, M_j\}$, then the clause $C = (L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_m \vee M_1 \vee \ldots \vee M_{j-1} \vee M_{j+1} \vee \ldots \vee M_n)\theta$ is the binary resolvent of C_1 and C_2 resolved upon the literals L_i and M_j

Let C be a clause, $L_1, \ldots, L_n (n \geq 1)$ are unifiable literals in C and θ is the mgu of $\{L_1, \ldots, L_n\}$. Then the factor of C is obtained by removing $L_2\theta, \ldots, L_n\theta$ from $C\theta$.

Correctness of resolution

If C is a resolvent of C_1 and C_2 then $\{C_1, C_2 \models C$.

Derivation

Let Σ be a set of clauses and C a clause. Derivation of C from Σ is a finite sequance of clauses $R_1, \ldots, R_k = C$, such that every clause R_i either belongs to Σ or is a resolvent of two clauses from $\{R_1, \ldots, R_k\}$. If such a derivation exist, we denote it by $\Sigma \vdash_r C$.

The derivation of an empty clause $\square$ from Σ we call refutation of Σ.

Deduction

Let C and D be clauses. C subsumes D ($C \succeq D$) if there exists a substitution θ such that $C\theta \subseteq D$. Notice, that in this definition the notation $\{A, \neg B_1, \ldots, \neg B_k\}$ is used for a clause $A \leftarrow B_1, \ldots, B_k$.

Correctness of subsumption

If C subsumes D then $C \models D$.

Let Σ be a set of clauses and C a clause. There exist a deduction of C from Σ (denote $\Sigma \vdash_d C$), if C is a tautology or there exist a clause D, such that $\Sigma \vdash_r D$ and D subsumes C.

Theorems

The subsumption theorem for resolution

Let Σ a set of clauses and C a clause. Then $\Sigma \models C$ iff $\Sigma \vdash_d C$.

Refutation theorem for resolution

Let Σ a set of clauses. Then Σ is unsatisfiable iff $\Sigma \vdash_r \Box$.

The subsumtion theorem for resolution and the Refutation theorem for resolution are equivalent.

SLD-derivation

Let Σ be a set of Horn clauses and C a Horn clause. SLD-derivation of C from Σ is a finite sequence of Horn clauses $R_0, \ldots, R_k = C$ such that $R_0 \in \Sigma$ and every $R_i for 1 \leq i \leq k$ is a binary resolvent of R_{i-1} and a program clause $C_i \in \Sigma$ in which the resolved literals are the head of C_i and a selected atom in the body of R_{i-1}. R_0 is called the *top* clause and C_i are the *input clauses* of SLD-derivation. We denote SLD-derivation of C from Σ by $\Sigma \vdash_{sr} C$.

The SLD-derivation of $\Box$ from Σ we call SLD-refutation of Σ (denote $\Sigma \vdash_{sr} \Box$).

SLD-deduction

Let Σ be a set of Horn clauses and C a Horn clause. There exist an SLD-deduction of C from Σ, if C is a tautology or there exist a Horn clause D, such that $\Sigma \vdash_{sr} D$ and D subsumes C. We denote SLD-derivation of C from Σ by $\Sigma \vdash_{sd} C$.

Theorems

The subsumption theorem for SLD-resolution

Let Σ a set of Horn clauses and C a Horn clause. Then $\Sigma \models C$ iff $\Sigma \vdash_{sd} C$.

Refutation theorem for SLD-resolution

Let Σ a set of Horn clauses. Then Σ is unsatisfiable iff $\Sigma \vdash_{sr} \Box$.

The subsumtion theorem for SLD-resolution and the Refutation theorem for SLD-resolution are equivalent.

Examples

Let f be a Herbrand interpretation of the language L.

a concept predicate is an arbitrary predicate $p_c(\mathcal{V}^{\mathcal{A}_1}, \dots, \mathcal{V}^{\mathcal{A}_n})$

a concept is defined as $C_f = \{p_c(d_1, \dots, d_n) \in B_L \mid d_1 \in \mathcal{D}_1, \dots, d_n \in \mathcal{D}_n$ and $p(d_1, \dots, d_n)$ is defined in $f\}$.

If $e \in C_f$ and $f(e) = 1$ then e is called a positive example of a concept C_f denoted by e^+.
If $e \in C_f$ and $f(e) = 0$ then e is called a negative example of a concept C_f denoted by e^-.
an example set $E = E^+ \cup E^-$, where $E^+ = \{e^+\}$ and $E^- = \{e^-\}$

Background knowledge and hypothesis

Let f be a Herbrand interpretation of the language L.

The background knowledge $\beta = \{C \mid$ where $C \in B_L$ or C is a Horn clause $\}$.

The hypothesis is defined as $\Sigma = \{p_c(x_{1\,1}, \dots, x_{1\,n}) \leftarrow p_{\beta_1}(y_{1\,1}, \dots, y_{1\,k}), \dots, p_{\beta_m}(y_{m\,1}, \dots, y_{m\,l}) \mid x_{i\,j}, y_{r\,s}$ are variables or constants of the language L, p_c is the concept predicate, p_{β_i} are predicates of the language $L\}$.

Rule covering

Let $e \in C_f$, $C \in \Sigma$ and β be background knowledge.

The covering of an example $e \in C_f$ by a rule C w.r.t. background knowldge β is defined as a function *covers* : $\Sigma \times \beta \times C_f \mapsto$ *true*, *false* as follows:

$$covers(C, \beta, e) = \begin{cases} true & \text{if } C \cup \beta \models e \\ false & \text{if } C \cup \beta \not\models e \end{cases}$$

Hypothesis covering

Let $e \in C_f$, Σ be a hypothesis and β be background knowledge.

The covering of an example $e \in C_f$ by a hypothesis Σ w.r.t. background knowldge β is defined as a function *covers* : $\Sigma \times \beta \times C_f \mapsto$ *true*, *false* as follows:

$$covers(\Sigma, \beta, e) = \begin{cases} true & \text{if } (\exists C \in \Sigma)\ C \cup \beta \models e \\ false & \text{if } (\forall C \in \Sigma)\ C \cup \beta \not\models e \end{cases}$$

Inductive Logic Programming task

Given are E^+, E^- and β. The task is to find a hypothesis Σ, such that the following conditions hold:

- completeness
  - $(\forall e^+ \in E^+)\ covers(\Sigma, \beta, e^+) = true$
- consistency
  - $(\forall e^- \in E^-)\ covers(\Sigma, \beta, e^-) = false$

Specialization

If a rule $C \in \Sigma$ is not consistent we have to weaken it, i.e. find a rule D what is specific according to C. This process is called specialization.

The specialization operator is a function $\upsilon : \Sigma \mapsto \Sigma$, where Σ is a set of Horn clauses

- applies some substitution θ to a clause
- adds a literal to the body of a clause

Generalization

If a rule $C \in \Sigma$ is not complete we have to strengthen it, i.e. find a rule D what is general according to C. This process is called generalization.

The generalization operator is a function $\upsilon : \Sigma \mapsto \Sigma$, where Σ is a set of Horn clauses

- applies some inverse substitution θ^{-1} to a clause
- removes a literal from the body of a clause

Theorem

Relation of specialization and generalization to covering

Let $e \in E$ be an example and C, D be a Horn clauses. Let C be an arbitrary specialization of D and D an arbitrary generalization of C. Then

- If $D \not\models e$ then $C \not\models e$
- If $C \models e$ then $D \models e$

Hypothesis space

A hypothesis space is the set of all Horn clauses.

An acceptable hypothesis space for a given example set E and background knowledge β is the set of all Horn clauses created from the predicates in E, and β.

Ordering on hypothesis space

As an ordering on hypothesis space we use *subsumption*.

Let C be a Horn clause and S be a set of Horn clauses.

- If $C \succeq D$ for all $D \in S$ then C is a generalization of S under subsumption.
- If for all generalization C' of S under subsumption holds that $C' \succeq C$ then C is the least generalization of S under subsumption.

- If $D \succeq C$ for all $D \in S$ then C is a specialization of S under subsumption.
- If for all specializations C' of S under subsumption holds that $C \succeq C'$ then C is the greatest specialization of S under subsumption.

Theorems

Let H be the set of all Horn clauses. $\top$ denote the most general clause of H and $\bot$ denote the most specific clause of H.

Existence of least generalizations on hypothesis space

For every finite subset S of H there exists the least generalization of S under subsumption.

Existence of greatest specialization on hypothesis space

For every finite subset S of H there exists the greatest specialization of S under subsumption.

The hypothesis space H ordered by subsumption is a lattice.

Recommended reading

- J. W. Lloyd: *Foundations of Logic Programming*. Springer, Berlin, 1987.
    - An introduction to crisp Logic Programming.
- S-H. Nienhuys-Cheng, R. de Wolf: *Foundations of Inductive Logic Programming*. Springer, Berlin, 1997.
    - Theoretical basis of classical Inductive logic programming
    - Basics from Logic Programming

Drawbacks of Crisp framework

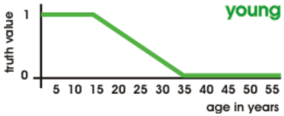
In a standard logical framework information are **absolutely** *true* or *false*.

- In real-world application information are often
  - *Imperfect* (imprecise, uncertain, vague, noisy, ...)
    - "Toyotas are probably better cars as Opels", "Hollywood stars are mainly lean", etc.
  - Expressed in *natural language* with *linguistic hedges*
    - "Very young girl", "Not too big building", "Middle aged", "Low price", etc.
  - *Aggregated* in accordance with their importance
    - "Managers prefer hotels near to business center while the cheapest hotels which are more far from the city center are better for students", etc.
- We **need a framework** to represent and deal with imperfect information.

Fuzzy framework

Unable to represent and deal with imperfect information.

- More truth values (from interval $[0, 1]$) representing the belief that an individual being to a concept.
- Representation of concepts of natural language (with linguistic hedges).



- Expression of preferences of information by aggregation function.

$$goodhotelformanager = \frac{2.near + cheap}{3}$$

Conjunctor

Function $C : [0, 1]^2 \mapsto [0, 1]$ is called *conjunctor* if

- $C(1, 1) = 1$
- $C(0, 1) = 0$
- $C(1, 0) = 0$
- $C(0, 0) = 0$
- $\forall r > 0 : C(1, r) > 0$
- *C is left continuous in second parameter*

Variables of conjunctor we denote $C(b, r)$

Implicator

Function $I : [0,1]^2 \mapsto [0,1]$ is called *implicator* if

- $C(1,1) = 1$
- $C(0,1) = 1$
- $C(1,0) = 0$
- $C(0,0) = 1$
- $\forall h < 1 : I(1,h) < 1$
- *I is right continuous in second parameter*

Variables of implicator we denote $I(b,h)$

Properties of conjunctors and implicators

property $a(C, I)$

- $(\forall b, h, r \in [0,1])\ I(b,h) \geq r$ iff $C(b,r) \leq h$

Properties of conjunctors and implicators

property $\phi_1(I)$

- $\phi 1(I)$ iff I is non-increasing in the first and non-decreasing in the second coordinate.

property $\phi_2(C, I)$

- $(\forall b, h \in [0, 1])\ C(b, I(b, h)) \leq h$

property $\phi_3(C, I)$

- $(\forall b, r \in [0, 1])\ I(b, C(b, r)) \geq r$

Properties of conjunctors and implicators

relation between a and ϕ_2, ϕ_3

$a(C, I)$ iff ($\phi_2(C, I)$ and $\phi_3(C, I)$)

Residuality

If $C(b, r) \leq h$ only if $I(b, h) \geq r$ then we call the tuple (C and I) residual

- The residual conjunctor to implicator I is $C_I(b, r) = inf\{h : I(b, h) \geq r\}$
- The residual implicator to conjunctor C is $I_C(b, h) = sup\{r : C(b, r) \leq h\}$

Properties of residual operators

property 1

Let I be an implicator. Then $C_I(b, r)$ is a conjunctor.

property 2

Let C be a conjunctor. Then $I_C(b, h)$ is an implicator.

property 3

Let I be an implicator and C be a conjunctor. Then the tuples (C_I, I) and (I_C, C) are residual.

Language

Our language consist of

- Attributes $\mathcal{A}_1, \ldots, \mathcal{A}_m \in \mathcal{A}$ with domains $\mathcal{D}_1, \ldots, \mathcal{D}_m \in \mathcal{D}$. For each attribute $\mathcal{A}_i$ we have
    - Variables $\mathcal{V}^{\mathcal{A}_i}$
    - Constants $\mathcal{C}^{\mathcal{A}_i}$
- Functions $f(\mathcal{A}_1, \ldots, \mathcal{A}_n) \in \mathcal{F}$
- Predicates $p(\mathcal{A}_1, \ldots, \mathcal{A}_o) \in \mathcal{P}$
    - $p(\mathcal{A}_1, \ldots, \mathcal{A}_o) : \mathcal{D}_{\mathcal{A}_1} \times \ldots \times \mathcal{D}_{\mathcal{A}_o} \mapsto [0, 1]$

Language

Our language have finitely many

- Conjunctions $\wedge_1, \ldots, \wedge_k$
- Disjunctions $\vee_1, \ldots, \vee_l$
- Implications $\rightarrow_1, \ldots, \rightarrow_m$
- Aggregations $@_1, \ldots, @_n$

A combination of $\wedge, \vee and @$ is again an aggregation.

Truth values and connectives

The set of truth values is the set of real numbers from the unit interval [0,1]
The truth functions of the operators are

- Conjunctors $\wedge_1^\bullet, \dots, \wedge_k^\bullet$
- Disjunctors $\vee_1^\bullet, \dots, \vee_l^\bullet$
- Implicators $\rightarrow_1^\bullet, \dots, \rightarrow_m^\bullet$
- Aggregators $@_1^\bullet, \dots, @_n^\bullet$
    - assume $@^\bullet(0, \dots, 0) = 0$ and $@^\bullet(1, \dots, 1) = 1$

Assume $\wedge^\bullet, \vee^\bullet and @^\bullet$ are **left continuous.**

Classical operators

- Let x the truth value of an event X and y the truth value of an event Y. (if X and Y are *disjunctive*)
  - Lukasiewicz
    - $\wedge_L^\bullet(x, y) = max\{0, x + y - 1\}$
    - $\vee_L^\bullet(x, y) = min\{1, x + y\}$
    - $\rightarrow_L^\bullet(x, y) = min\{1, 1 - x + y\}$

Classical operators

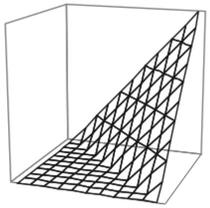
- Let x the truth value of an event X and y the truth value of an event Y.
  - Godel (if X and Y are *inclusive*)
    - $\wedge^{\bullet}_{G}(x, y) = min\{x, y\}$
    - $\vee^{\bullet}_{G}(x, y) = max\{x, y\}$
    - $\rightarrow^{\bullet}_{G}(x, y) = \begin{cases} y, & x > y \\ 1, & else \end{cases}$

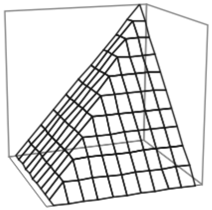
Classical operators

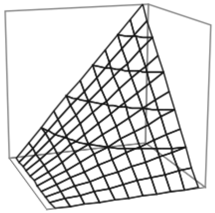
- Let x the truth value of an event X and y the truth value of an event Y.
    - Product (if X and Y are *independent*)
        - $\wedge^{\bullet}_{P}(x, y) = x.y$
        - $\vee^{\bullet}_{P}(x, y) = x + y - x.y$
        - $\rightarrow^{\bullet}_{P}(x, y) = min\{1, \frac{y}{x}\}$

Comparison of classical conjunctors

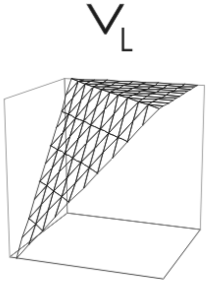
$\wedge_L$ $\wedge_G$ $\wedge_P$

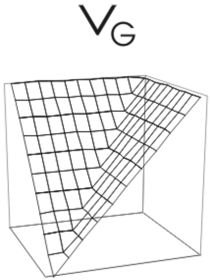


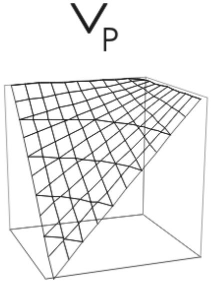




Comparison of classical disjunctors

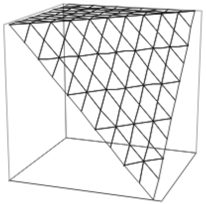




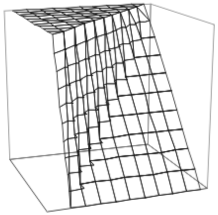


Comparison of classical implicators

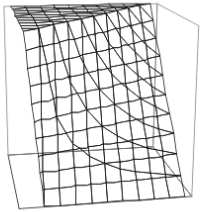
$\rightarrow_L$



$\rightarrow_G$



$\rightarrow_P$



Fuzzy structures

Let $\mathcal{B}_L$ be the Herbrand base.

- A mapping $f : \mathcal{B}_L \mapsto [0, 1]$ is said to be a fuzzy Herbrand interpretation
- f can be extended to $\overline{f}$ all formulas along the complexity of formula using the truth function of connectives

Fuzzy theories

Let $\mathcal{B}_L$ be the Herbrand base.

- A graded formula $(\varphi.x)$ is true in an interpretation f $(f \models_{FLP} \varphi.x)$ if $\overline{f}(\varphi) \geq x$
- f is a model of a theory P if for all formulas $\varphi \in dom(P)$ we have $f(\varphi) \geq P(\varphi)$

Fuzzy modus ponens

The procedural semantics is based on the backward usage of fuzzy modus ponens

- $\frac{(B.b),(H \leftarrow_I B.r)}{(H.C_I(b,r))}$, where C_I is the residual conjunctor to implicator $\leftarrow_I$
- if the body holds with grade b and the rule holds with grade r then the truth of the head of the rule is equal to $C_I(b, r)$

Fuzzy modus ponens

Soundness of fuzzy modus ponens

Fuzzy modus ponens is a sound rule, i.e. if $f(B) \geq b$ and $f(H \leftarrow_I B) \geq r$ then $f(H) \geq C_I(b, r)$.

Proof of soundness of fuzzy modus ponens

From premises of the theorem we have
$f(B) \geq b$
$f(B \rightarrow_I H) = I(f(B), f(H)) \geq r$

So,
$C(b, r) \leq C(f(B), f(B \rightarrow_I H))$

Since $C = C_I$ and the property $\phi_2(C, I)$ holds, we get
$C_I(f(B), f(B \rightarrow_I H)) = C_I(f(B), I(f(B), f(H))) = C_I(b, I(b, h) \leq f(H)$

We see, that $C_I(b, r)$ is the largest sound evaluation of modus ponens.

Declarations

- *fact* is a graded atom $(B.b)$, where $b \in [0, 1]$
- *body* is of the form $B = @(B_1, \dots, B_n)$
- *rule* is a graded implication $(H \leftarrow @(B_1, \dots, B_n).r)$, where $r \in [0, 1]$
    - $H \leftarrow @(B_1, \dots, B_n)$ is the *logical part* of the rule
    - $r \in [0, 1]$ is the *quantitative part* of the rule
- A finite set P of positively graded rules and facts is said to be a *fuzzy logic program* if there are no two rules (facts) with the same logical parts and different quantitative parts

Correct answer

A pair $(x; \vartheta)$ consisting of a real number $0 < x \leq 1$ and a substitution ϑ is a correct answer for a program P and a query "$? - A$" if for arbitrary interpretation f, which is a model of P, we have $\overline{f}(\forall(A\vartheta)) \geq x$.

Computation rules

We define the 1. admissible rule as follows:

- From $((XA_mY); \upsilon)$ infer $((XC(B, r)Y)\theta; \upsilon \circ \theta)$ if
  - A_m is an atom (called the selected atom)
  - θ is an mgu of A_m and H
  - $P(H \leftarrow B) = r$ and B is a (nonempty) body

Computation rules

We define the 2. admissible rule as follows

- From (XA_mY) infer $(X0Y)$ if in an aggregation an argument is missing

Computation rules

We define the 3. admissible rule as follows

- From $((XA_mY);\upsilon)$ infer $((XrY)\theta;\upsilon\circ\theta)$ if
  - A_m is an atom (called the selected atom)
  - θ is an mgu of A_m and A
  - $P(A) = r$ (i.e. A is a fact)

Computation rules

We define the 4. admissible rule as follows

- If the word does not contain any predicate symbols rewrite all connectives ($\vee$'s, $\wedge$'s and @'s) to $\vee^\bullet$, $\wedge^\bullet$ and $@^\bullet$. As this word contains only some additional C's and reals evaluate it (of course the substitution remains untouched).

Computed answer

A pair $(r;\theta)$ consisting of a (rational) number r and a substitution θ is said to be a *computed answer* for a program P and a goal "$? - A$" if there is a sequence $G_0, \ldots, G_n$ such that

- every G_i is a pair consisting of a word and a substitution
- $G_0 = (A, id)$
- every G_{i+1} is inferred from G_i by one of the inference rules (we do not forget the usual Prolog renaming of variables along derivation)
- $G_n = (r, \theta')$ and $\theta = \theta$' restricted to variables of A

Soundness of Fuzzy Logic Programming

Soundness of the semantics

Every computed answer for a definite fuzzy logic program P and goal "$? - A$" is a correct answer.

Proof of soundness of fuzzy logic programming

Take a goal A with a computation of length $k+1$ starting with a rule $P(A \leftarrow B) = r$. Suppose that the result holds for all computed answers due to computations of length $\leq k$. For each atom D from the body B there is a computation of length $\leq k$, hence computed answer $d \leq f(D)$ in every model of P. But then $f(B) \geq b$, where b is the computed answer for the whole body. This is because conjunctions, disjunctions and aggregations are monotone in both coordinates. Hence, f being a model of P means $f(A \leftarrow B) \geq P(A \leftarrow B) = r$ and by the soundness of modus ponens we get $f(A) \geq C(b, r)$.

The T_P operator

$F_P = \{f \mid f : B_P \mapsto [0,1]\}$

- F_P is a lattice with coordinatewise lattice ordering
- define the operator $T_P : F_P \mapsto F_P$ as follows

$T_P(f)(A) = max\{sup\{C_i(f(B), r) \mid (A \leftarrow iB.r)$ is a ground instance of a rule in the program $P\}$, $sup\{b \mid (A.b)$ is a ground instance of a fact in the program $P\}\}$.

The fixpoint theorem

Assume that all implications fulfill properties $\phi 1 - \phi 3$. Moreover, assume that all C's, $\wedge$'s, $\vee$'s and @'s are lower semicontinuous. Then

- T_P is continuous (i.e. T_P preserves joins of upward directed sets of interpretations)
- f is a model of P iff $T_P(f) \leq f$ (hence the minimal fixpoint of T_P is a model of P)

Proof of the fixpoint theorem

The continuity of T_P is straightforward, using mootonicity and lower continuity of all connectives in the body and conjunctors evaluating fuzzy modus ponens.

Proof of the fixpoint theorem

The second part of the theorem we prove as follows: Denote $\rightarrow_i^{\bullet} = I_i$. Assume that f is a model of P. For an $H \leftarrow_i B$ a ground instance of some $C \in dom(P)$ we would like to show that $f(H) \geq C_i(f(B), P(C))$.
We have
$f(H \leftarrow_i B) \geq f(C) \geq P(C)$,
the first inequality holds because $H \leftarrow_i B$ is a ground instance of C, the second, because f is a model of P.

Proof of the fixpoint theorem

By $(\phi_1 - \phi_2)$ we have
$C_i(f(B), P(C)) \leq C_i(f(B), f(H \leftarrow_i B)) =$
$C_i(f(B), I_i(f(B), f(H))) \leq f(H)$.
Now assume $T_P(f) \leq f$. It sufficies for all ground instances
$H \leftarrow_i B$ of C to show $f(H \leftarrow_i B) \geq P(C)$. But
$f(H) \geq T_P(f)(H) \geq C_i(f(B), P(C))$
gives (by ϕ_1 and the above)
$f(H \leftarrow_i B) = I_i(f(B), f(H)) \geq I_i(f(B), C_i(f(B), P(C))) \geq P(C)$.
The last inequality is ϕ_3.
Hence the fixpoint theorem works even without any further assumptions on conjunctors (definitely they must not be commutative and associative).

Completeness of fuzzy logic programs

Assume that all implications fulfill properties $\phi 1 - \phi 3$. Moreover, assume that all C's, $\wedge$'s, $\vee$'s and @'s are lower semicontinuous. Then for every correct answer $(x; \vartheta)$ for P and "$? - A$" and for every $\varepsilon > 0$ there is a computed answer (r, θ) for P and "$? - A$" such that $x - \varepsilon < r$ and $\vartheta = \theta\gamma$ (for some γ)

Proof of completeness of fuzzy logic programs

The previous theorem practically states the completeness for ground instances of atoms, because the correct answer for a ground query A holds also in the minimal model which is a countable iteration of the T_P operator starting from the 0-interpretation.
Now having a positive ε there is an n such that $T_P^n(0)(A) > T_P^\omega(0) - \varepsilon$. Looking for the contributions to the value of $T_P^n(0)(A)$, there one also greater than $T_P^\omega(0) - \epsilon$, this is obtained through the application of a rule or a fact, in any case we can trace the computation backwards. The only difference from the classical case is that a zero value of an atom can appear in a nonzero value of a disjunction or aggregation (that is why we need the admissible rule 2).

Proof of completeness of fuzzy logic programs

To extend it for general case we can extend the language by constants c_X for every variable and define the truth value of a formula with c_X as the truth value of the formula with universally quantified variable X, that is as the infimum of truth values on all constants. Such a structure of the language is again a model of the theory P, because where in the theory there was a free variable, truth value is computed as though it was universally quantified, hence all values on constants are greater or equal and hence also on c_X. Hence, the validity on ground atoms directly follows from the fixed point theorem and using the *lifting lemma* and *mgu lemma* we obtain the full result.

Fuzzy logic paradox

Assume A is a proposition with truth values in a structure $f(A) = f(\neg A) = \frac{1}{2}$.
Then the following holds:
$f(A \wedge A) = \wedge^{\bullet}(f(A), f(A)) = \wedge^{\bullet}(f(A), f(\neg A)) = f(A \wedge \neg A)$,
which is never fulfilled in real world applications.

Fuzzy logic paradox

Our solution to this paradox is that it is wrong to describe conjunction of A and A and with $\neg A$ with the same conjunction. Our system offers sound and complete deduction with many connectives, which can be chosen to fit the real world situation. In the above case it can look like

$$\frac{1}{2} = f(A \wedge_G A) = \wedge_G^{\bullet}(f(A), f(A)) > \wedge_L^{\bullet}(f(A), f(\neg A)) = f(A \wedge_L \neg A) = 0.$$

Examples, Background Knowledge and Hypothesis

A fuzzy example set E^F consists of fuzzy facts (i.e. a graded atoms). Its elements $e.\alpha$ are called graded examples with grade α.

A fuzzy background knowledge B^F is a fuzzy logic program.

A fuzzy hypothesis H^F is a set of fuzzy rules.

Fuzzy Inductive Logic Programming task

Given are E^F and B^F. The task is to find a fuzzy hypothesis H^F, such that the following conditions hold:

- flp-completeness
  - $(\forall e.\alpha \in E^F)\ (H^F \cup B^F) \models_{FLP} e.\alpha$
- flp-consistency
  - $(\forall e.\alpha \in E^F)(\forall \beta > \alpha)\ (H^F \cup B^F) \not\models_{FLP} e.\beta$

Theorem

Generality of FILP

The formal model od fuzzy inductive logic programming task is a generalization of inductive logic programming task.

Proof

Reduce the truth value interval $[0,1] \cap Q$ to $\{0,1\}$. Thus, the example set consists of fuzzy examples $e_i.0$ or $e_i.1$ which correspond to crisp negative and positive examples. In case of truth values from $\{0,1\}$ fuzzy operators are generalizations of classical crisp operators, so the fuzzy background knowledge and the fuzzy hypothesis correspond to crisp background knowledge and crisp hypothesis.

Proof

The flp-completeness condition holds, if $(H \cup B)(e) \geq \alpha$ for every $e.\alpha \in E^F$. If $\alpha = 1$ then $(H \cup B)(e) \geq \alpha = 1$ and since the truth values can be 0 or 1, $(H \cup B)(e) = 1$. It means, all positive examples ($e.\alpha = e.1$ are entailed by $H \cup B$.
The flp-consistency condition holds, if $(H \cup B)(e) < \beta$ for every $e.\alpha \in E$) and $\beta > \alpha$. Since the truth values are 0 or 1, if $\alpha = 0$ the only value for β can be 1, so $(H \cup B) = 0$. It means that e does not belong to the minimal model of $H \cup B$, so non of the negative examples ($e.\alpha = e.0$) are entailed by $(H \cup B)$.

So, if a hypothesis is flp-complete (flp-consistent) then it is complete (consistent), too.

Generalized Annotated Programs

The language consists of

- qualitative part
  - language of predicate logic
- quantitative part
  - for each predicate p there is a (possibly different) truth values set T_P with ordering $\leq_p$.
  - consists of annotation terms which are assigned to an annotation function what are assumed to be total continuous (hence monotonic) in the sense of lattice theory.

Generalized Annotated Programs

If A is an atomic formula and α is an annotation term, then $A : \alpha$ is an annotated atom. If $\alpha \in [0, 1]$ then $A : \alpha$ is c-annotated. When α is a variable, then $A : \alpha$ is v-annotated.
If $A : \rho$ is a possibly complex annotated atom and $B_1 : \mu_1, \ldots, B_k : \mu_k$ are variable-annotated atoms, then $A : \rho(\mu_1, \ldots, \mu_k) \leftarrow B_1 : \mu_1 \wedge \ldots B_k : \mu_k$ is an annotated clause.

We stress, that atoms in the body of a rule are only v-annotated and only facts can be c-annotated. We assume that variables in the annotation of the head also appear as annotations of the body literals and different literals in the body are annotated with different variables.

Generalized Annotated Programs

Let B_L be the Herbrand base of the qualitative part of the GAP language. A mapping $f : B_L \mapsto [0,1]$ is said to be a Herbrand interpretation for annotated logic.
The satisfaction is defined along the complexity of formulas as in the classical logic.
Suppose $f : B_L \mapsto [0,1]$ is an interpretation, $\mu \in [0,1]$ and A is ground atom, then $f \models_{GAP} A : \mu$, i.e. f is a model of $A : \mu$ iff $f(A) \geq \mu$.

$f \models_{GAP} A : \rho(\mu_1, \ldots, \mu_k) \leftarrow B_1 : \mu_1, \ldots, B_k : \mu_k$
if for all assignments v of annotation variables we have
$f(A) \geq_A \rho(v(\mu_1), \ldots, v(\mu_k)) \leftarrow f(B_1) \geq_{B_1} v(\mu_1) \wedge \ldots$
$\ldots \wedge f(B_k) \geq_{B_k} v(\mu_k)$

Generalized Annotated Programs

FLP and GAP transformations

Assume $C = A : \rho \leftarrow B_1 : \mu_1, \dots, B_k : \mu_k$ is an annotated clause. Then *flp*(C) is the fuzzy rule $A \leftarrow \rho(B_1, \dots, B_k).1$, here ρ is understood as an n-ary aggregator operator.

Assume $D = A \leftarrow_i @(B_1, \dots, B_n).r$ is a fuzzy logic program rule. Then *gap*(D) is the annotated clause
$A : C_i(@(x_1, \dots, x_n), r) \leftarrow B_1 : x_1, \dots, B_n : x_n$.

Generalized Annotated Programs

FLP and GAP equivalence

Assume C is an annotated clause, D is a fuzzy logic program rule and f is a fuzzy Herbrand interpretation. Then

- f is a model of C iff f is a model of $flp(C)$.
- f is a model of D iff f is a model of $gap(D)$.

Induction in GAP

A GAP example set E^G consists of GAP facts (i.e. a graded atoms). Its elements $e.\alpha$ are called graded examples with grade α.

A GAP background knowledge B^G is a generalized annotated logic program.

A GAP hypothesis H^G is a set of generalized annotated program rules.

Inductive Generalized Annotated Programming task

Given are E^G and B^G. The task is to find a GAP hypothesis H^G, such that the following conditions hold:

- gap-completeness
  - $(\forall e.\alpha \in E^G)\ (H^G \cup B^G) \models_{GAP} e.\alpha$
- gap-consistency
  - $(\forall e.\alpha \in E^G)(\forall \beta > \alpha)\ (H^G \cup B^G) \not\models_{GAP} e.\beta$

Observation

Fuzzy ILP and IGAP transformation

The Fuzzy Inductive Logic Programming task can be transformed to Inductive Generalized Annotated Program task.

The Inductive Generalized Annotated Program task can be transformed to Fuzzy Inductive Logic Programming task.

Truth completion on examples

Let $E^G = \{e.\alpha | \alpha \in [0, 1]\}$ be an annotated example set.

For a $\beta \in [0, 1]$ we divide E^G into two parts

- $E^{G^{\beta +}} = \{e.\alpha | \alpha \geq \beta\}$
- $E^{G^{\beta -}} = \{e.\alpha | \alpha < \beta\}$

The sets $E^{G^{\beta +}}$ and $E^{G^{\beta -}}$ correspond to positive and negative example sets (as in classical meaning) relevant to the grade β.

Truth completion on background knowledge

Every fact $p_i(x_1, \dots, x_n) : \alpha$ in the background knowledge we transform to classical, two-valued form $p_i(x_1, \dots, x_n, \alpha)$.

For every predicate symbol p_i we add to background knowledge

- monotonicity rules
  - $p_i(x_{i1}, \dots, x_{i i_k}, \upsilon) \quad \leftarrow \quad \leq_{p_i} (\upsilon, \delta), \quad p_i(x_{i1}, \dots, x_{i i_k}, \delta)$
- ordering relations
  - $\leq_{p_i} (\alpha_1, \alpha_2), \leq_{p_i} (\alpha_2, \alpha_3), \dots$
    $\dots \leq_{p_i} (\alpha_{k_{i-2}}, \alpha_{ki-1}), \leq_{p_i} (\alpha_{k_{i-1}}, \alpha_{ki})$
  - where $\{\alpha_1, \dots, \alpha_i\}$ is the set of distinct truth values appearing in the annotations of the predicate p_i.

Solving IGAP via classical ILP

Input: E^G and B^G
Output: H^G vspace0.3cm The IGAP algorithm

- initialize hypothesis $H^G = \emptyset$
- make truth completion on B^G
- for every $\beta \in TV_{E^G} = \{\alpha_1, \ldots, \alpha_k\} \setminus \{\alpha_1\}$, where TV_{E^G} is the set of distinct truth values in the annotations of examples do the following:
    - make truth completion on E^G relevant to β
    - compute with classical two-valued ILP the hypothesys H_β on truth completed examples and background knowledge.
    - Transform H_β to annotated form H^G_β
    - $H^G = H^G \cup H^G_\beta$

Theorem

gap-consistency of IGAP algorithm

Given annotated background knowledge and annotated example set. The IGAP algorithm finds a gap-consistent annotated hypothesis.

Proof

Let E, B and H be annotated example set, background knowledge and hypothesis. The gap-consistency condition requires that for every $e : \alpha \in E$ the following holds $(\forall \beta > \alpha)(H \cup B) \not\models_{GAP} e : \beta$, so the minimal model $M_{H \cup B}$ of $H \cup B$ can not assign a truth value β (higher than α) to example $e : \alpha$. By contradiction, assume that our algorithm assigns a truth value β higher than α to an example $e : \alpha$. From the construction of positive and negative example sets in our algorithm, and from the classical consistency of ILP (the hypothesis can not cover negative examples) it is clear that an example $e : \alpha$ can be covered only with a hypothesis of the truth value $\delta \leq \alpha$. So, it is not possible that minimal model $M_{H \cup B}$ assigns truth value β to example $e : \alpha$.

Corollary

flp-consistency and gap-consistency relation

Given fuzzy background knowledge B^F and fuzzy example set E^F. Let us transform the E^F and B^F to annotated example set E^A and annotated background knowledge B^A. Let compute with IGAP algorithm an annotated hypothesis H^A and transform it to fuzzy hypothesis H^F. Then H^F is flp-consistent.

Recommended reading

- P. Vojtáš: *Fuzzy logic programming*. Fuzzy Sets and Systems. 124, 3 (2004), p: 361-370
    - Contains all themes of the deductive part of fuzzy logic programming covered by the subject.
- H. T. Nguyen, E. A. Walker: *A First Course in Fuzzy Logic*. Chapman and Hall/CRC, USA, 2006.
    - An introduction to the theory of fuzzy sets.
- T. Horváth, P. Vojtáš: *Fuzzy induction via generalized annotated programs*. In: 8th International Conference on Computational Intelligence (Fuzzy Days, Dortmund '04), Dortmund, Germany, 2004: Springer, 2005, p: 419-433.
    - A Fuzzy Inductive Logic Programming approach covered by the subject.

- Expert systems
- Similarities and the problem of fuzzy unification
- Fuzzy databases
- Fuzzy resolution
- Fuzzy logic programming abduction

- Computing with words
- Multi-criterial decision making
- Semantic web applications
  - Flexible search
  - User preferences
- Fuzzy data mining
  - Hard task because of many (possible unknown) operators and many truth values
- Use of fuzzy logic in areas with many imperfect information
  - Medicine, Genetics, Economics, ...