Languages for learning

Attribute-value languages

$L = \{A_1 = V_1, ..., A_n = V_n | V_1 \in V_{A_1}, ..., V_n \in V_{A_n}\}$,

where V_{A_1} is the set of possible values for A_i, $i = 1, ..., n$.

For example, $e =$\{`color = green`, `shape = rectangle`\}.

Predicate representation (propositional logic)

$p_1 =$ `(color = green)`

$p_2 =$ `(shape = rectangle)`

$e = p_1 \wedge p_2$

Ordering examples/hypotheses

Generality (subsumption, covering) relation, $\geq$

Nominal attributes (no ordering between their values exists): $X \geq Y$, if $X \subseteq Y$. For example, {`shape = rectangle`} $\geq$ {`color = green`, `shape = rectangle`}.

Linear (numeric) attributes (a full order on the atribute values exists): $X = \{A_1 = X_1, ..., A_n = X_n\}$, $Y = \{A_1 = Y_1, ..., A_n = Y_n\}$. Then $X \geq Y$, if $X_i \geq Y_i$ (relation between numbers) $(i = 1, ..., n)$.

Structural attributes (a partial order on the atribute values exists): $X \geq Y$, if $X_i \geq Y_i$ $(i = 1, ..., n)$, where $X_i \geq Y_i$ means that Y_i is a successor of X_i in a taxonomic tree.

Language of hypotheses

L + *disjunction*:

$L_H = \{C_1 \vee C_2 \vee ... \vee C_n | C_i \in L, i \geq 1\}$.

$H \rightarrow E$, if there exists a conjunct $C_i \in H$, so that $C_i \geq E$.

Semantic subsumption: $H \geq_{sem} H'$, if $H \rightarrow E, H' \rightarrow E', E \supseteq E'$.

Syntactic subsumption: $H \geq H'$, if $\forall C_i \in H, \exists C_j \in H' : C_i \geq C_j$.

if $H \geq H'$, then $H \geq_{sem} H'$. (What about the reverse?)

Representing hypotheses as rules

$H = \{C_1 \vee C_2 \vee ... \vee C_n\}$

`if` C_1 `then` $+$`,`
`if` C_2 `then` $+$`,`
`...`
`if` C_n `then` $+$

Multi-concept learning

$E = \cup_{i=1}^{k} E^i$

i^{-th} problem $\Rightarrow E^+ = E^i,\ E^- = E \backslash E^i$

Rules: `if` C_i `then` $Class_j$

Least general generalization (lgg)

$H = lgg(H_1, H_2)$ if:

- $H \geq H_1$ and $H \geq H_2$
- $\forall H'$: $H' \geq H_1,\ H' \geq H_2 \Rightarrow H' \geq H$.

Examples

- Nominal attributes: $lgg(H_1, H_2) = H_1 \cap H_2$.
- Linear attributes: minimal intervals including both attribute values.
- Structural attributes: closest common parents for both attribute values in the taxonomy.

Relational languages

A sample from the MONK examples:

---

example(1,pos,[hs=octagon, bs=octagon, sm=no, ho=sword, jc=red, ti=yes]).
example(2,pos,[hs=square, bs=round, sm=yes, ho=flag, jc=red, ti=no]).
example(3,pos,[hs=square, bs=square, sm=yes, ho=sword, jc=yellow, ti=yes]).
example(4,pos,[hs=round, bs=round, sm=no, ho=sword, jc=yellow, ti=yes]).
example(5,pos,[hs=octagon, bs=octagon, sm=yes, ho=balloon, jc=blue, ti=no]).
example(6,neg,[hs=square, bs=round, sm=yes, ho=flag, jc=blue, ti=no]).
example(7,neg,[hs=round, bs=octagon, sm=no, ho=balloon, jc=blue, ti=yes]).

Propositional representation

```
if [hs=octagon, bs=octagon] then +
if [hs=square, bs=square] then +
if [hs=round, bs=round] then +
if [jc=red] then +
```

For class ”−” we need 18 rules (why?).

Relational rules

```
if [hs=bs] then +
if [jc=red] then +
if [hs≠bs,jc≠red] then -
```

First-Order Logic atoms for positive examples

$monk(octagon, octagon, no, sword, red, yes)$
$monk(square, round, yes, flag, red, no)$
$monk(square, square, yes, sword, yellow, yes)$
$monk(round, round, no, sword, yellow, yes)$
$...$

First-Order Logic atoms for hypothesis "+"

$monk(A, A, B, C, D, E)$
$monk(A, B, C, D, red, E)$

Prolog

$class(+, X) : -hs(X, Y), bs(X, Y).$
$class(+, X) : -jc(X, red).$
$class(-, X) : -not\ class(+, X).$

First-Order Logic – alphabet

- Variables: alphanumerical strings beginning a capital – X, Y, $Var1$.
- Constants: alphanumerical strings beginning with a lower case letter (or just numbers) – a, b, c, $const1$, 125.
- Functions: f, g, h, or other constants.
- Predicates: p, q, r, $father$, $mother$, $likes$, or other constants.
- Logical connectives: $\wedge$ (*conjunction*), $\vee$ (*disjunction*), $\neg$ (*negation*), $\leftarrow$ or $\rightarrow$ (*implication*) and $\leftrightarrow$ (*equivalence*).
- Quantifiers: $\forall$ (*universal*) and $\exists$ (*existential*)
- Punctuation symbols: (,) and ,

First-Order Logic – terms

- a variable is a term;
- a constant is a term;
- if f is a n-argument function ($n \geq 0$) and $t_1, t_2, ..., t_n$ are terms, then $f(t_1, t_2, ..., t_n)$ is a term.

First-Order Logic – formulas

- if p is an n-argument predicate ($n \geq 0$) and $t_1, t_2, ..., t_n$ are terms, then $p(t_1, t_2, ..., t_n)$ is a formula (called *atomic formula* or *atom*;)
- if F and G are formulas, then $\neg F$, $F \wedge G$, $F \vee G$, $F \leftarrow G$, $F \leftrightarrow G$ are formulas too;
- if F is a formula and X – a variable, then $\forall X F$ and $\exists X F$ are also formulas.

First-Order Logic – examples

"For every man there exists a woman that he loves."
(classes of objects ⇒ variables):

$\forall X \exists (Y man(X) \rightarrow woman(Y) \wedge loves(X, Y))$

"John loves Mary." (concrete objects ⇒ constants):

$loves(john, mary)$

"Every student likes every professor." :

$\forall X \forall Y (is(X, student) \wedge is(Y, professor) \rightarrow likes(X, Y))$

Or (universal quantifiers may be skipped):

$is(X, student) \wedge is(Y, professor) \rightarrow likes(X, Y)$

Language of logic programming – Horn clauses

- Literal: an atom or its negation.
- Complementary literals: A and $\neg A$.
- Clause: a disjunction of literals.
- Horn clause: a clause with no more than one positive literal.
- Empty clause ($\Box$): a clause with no literals (logical constant "false").

Language of logic programming – Prolog notation

$A \vee \neg B_1 \vee \neg B_2 \vee \ldots \vee \neg B_m$
$(p \leftarrow q = p \vee \neg q)$

$A \leftarrow B_1, B_2, \ldots, B_m$

Program clause (rule):

$A : -B_1, B_2, \ldots, B_m$

Goal:

$: -B_1, B_2, \ldots, B_m$
or
$? - B_1, B_2, \ldots, B_m$

Fact:

A (single atom)

Substitutions

$\theta = \{V_1/t_1, V_2/t_2, ..., V_n/t_n\}$

$V_i \neq V_j \ \forall i \neq j,\ t_i \neq V_i,\ \ i = 1, ..., n$

Example:

$t_1 = f(a, b, g(a, b))$, $t_2 = f(A, B, g(C, D))$

$\theta = \{A/a, B/b, C/a, D/b\}$

$t_1\theta = t_2$, $t_2\theta^{-1} = t_1$ (inverse substitution)

Term generality (covering, instance relation)

$t_1 \geq t_2 \Leftrightarrow \exists\theta\ (\theta^{-1}) : t_1\theta = t_2\ (t_2\theta^{-1} = t_1)$

Term unification

$t_1 = f(X, b, U)$, $t_2 = f(a, Y, Z)$

Unifiers of t_1 and t_2: $\theta_1 = \{X/a, Y/b, Z/c\}$, $\theta_2 = \{X/a, Y/b, Z/U\}$

$t_1\theta_1 = t_2\theta_1 = f(a, b, c)$

$t_1\theta_2 = t_2\theta_2 = f(a, b, U)$ (most general unifier - mgu)

Term anti-unification, lgg

$$f(X, g(a, X), Y, Z) = lgg(f(a, g(a, a), b, c), f(b, g(a, b), a, a)$$

Anti-unification, lgg, lattice of terms

```
               V
              ...
   anti-unifiers of t1 and t2
              ...
          lgg(t1,t2)
              /\
             /  \
            /    \
           /      \
          t1       t2
           \      /
            \    /
             \  /
              \/
           mgu(t1,t2)
              ...
     unifiers of t1 and t2
              ...
```

(the lower part of the lattice may not exists)

Semanics of logic programs

Herbrand base (B_P): all ground atoms that can be built by predicates from P with arguments – functions and constants from P.

Model of clause (M_C): Let $C = A$:- $B_1, ..., B_n$ $(n \geq 0)$ belong to P and $M_C \subseteq B_P$. M_C is a *model* of C, if for all ground instances $C\theta$, either $A\theta \in M$ or $\exists B_j, B_j\theta \notin M$.

Empty clause $\square$ has no model.

Least Herbrand model of logic program P **(M_P):** the intersection of all models of P.

Intuition:

- Express when a clause or a logic program is true?
- Depends on the model (the context where the clause appears).
- This model is represented by a set of facts.

Logical consequence

$P_1 \models P_2$, if every model of P_1 is also a model of P_2.

P is *satisfiable* (consistent, true), if P has a model.
Otherwise P is *unsatisfiable* (inconsistent, false).

If $P \models \Box$, then P is unsatisfiable.

Deduction theorem: $P_1 \models P_2 \;\Leftrightarrow\; P_1 \wedge \neg P_2 \models \Box$.

Majot result in LP: $M_P = \{A | A\ is\ a\ ground\ atom, P \models A\}$

How to find M_P?

- Find all models of P.
- Use inference rules: procedures I for transforming one formula (program, clause) P into another one Q, denoted $P \vdash_I Q$.
- I is *correct and complete*, if $P \vdash_I P \;\Leftrightarrow\; P_1 \models P_2$.

Resolution (correct and complete inference rule)

- C_1 and C_2 are clauses
- There exist $L_1 \in C_1$ and $L_2 \in C_2$ that can be made complementary by applying an *mgu*, i.e. $L_1\mu = \neg L_2\mu$.
- Then $C = (C_1 \backslash \{L_1\} \cup C_2 \backslash \{L_2\})\mu$ is called *resolvent* of C_1 and C_2.
- Most importantly, C follows from C_1 and C_2, i.e. $C_1 \wedge C_2 \models C$.

Example:

$C_1 = grandfather(X, Y) : -parent(X, Z), father(Z, Y).$
$C_2 = parent(A, B) : -father(A, B).$

$\mu = \{A/X, B/Z\}, \;\; parent(A, B)\mu = \neg parent(X, Z)$

Then, the resolvent of C_1 and C_2 is:

$C = grandfather(X, Y) : -father(X, Z), father(Z, Y),$

Prolog

Question: Given logic program P and atom A, find if A logically follows from P.

```
grandfather(X,Y) :- parent(X,Z), father(Z,Y).
parent(A,B) :- father(A,B).
father(john,bill).
father(bill,ann).
father(bill,mary).
```

Is John a grandfather of Ann?

```
?- grandfather(john,ann).
yes
?-
```

Who are the grandchildren of John?

```
?- grandfather(john,X).
X=ann;
X=mary;
no
?-
```