Learning from Interpretations

B4M36SMU

In this tutorial, we will get familiar with the basics of first-order logic needed for this course. We will briefly talk about the generalizing agent from the lecture in the context of inductive logic programming (ILP). In particular, we will focus on learning from interpretations, that is learning hypothesis $\mathcal{H}$ such that:

$$o \models \mathcal{H} \; \forall o \in O^+$$
$$o \not\models \mathcal{H} \; \forall o \in O^-$$

In other words, the target hypothesis is a model for all positive observations and is not a model for all negative observations.

Motivation

What is the motivation for combining first-order logic (FOL) with machine learning? Is finite vector representation expressive enough to represent all kinds of problems? Consider the following domains:

- graphs with the variable number of nodes, e.g. molecules
- relationship representation, e.g. taxonomies
- structured data, e.g. semantic tree of a sentence

Logic

This section contains basic definitions of elements of first-order predicate logic needed for this tutorial.

Term is a constant, variable, or a compound term. Constant symbols represent objects in the domain, e.g. *Adam*. A variable ranges over the domain's objects, e.g. x. A compound term is a function symbol, e.g. $fatherOf/1$, applied on a-tuple of terms where a is its arity, e.g. $fatherOf(Adam)$. Predicate symbols express relations among objects in the domain of their attributes, e.g. $sibling/2$. An atom is a predicate symbol of arity a applied to an a-tuple of terms, e.g. $sibling(x, Adam)$. Literal is an atom or its negation, e.g. $\neg sibling(x, Adam)$. Formulae are constructed from literals using logical connectives ($\wedge$, $\vee$, $\implies$, ...) and quantifiers ($\forall$, $\exists$). A *ground* term, literal, clause, theory, etc., does not contain any variable. A clause is a universally quantified disjunction of literals, e.g. $\forall x, y : \neg sibling(Adam, x) \vee sibling(x, y)$. However, the quantifier part of a clause is often omitted. CNF is a conjunction of disjunction, e.g. $(sibling(x, y) \vee \neg human(Eva) \vee \neg human(Adam)) \wedge (\neg human(x) \vee mortal(x))$; see that the quantifier part is also omitted.

A clause is *range-restricted* if any variable occurring in a positive literal of it, also occurs in a negative literal of it, e.g. $path(x, z) \vee \neg path(x, y) \vee \neg path(y, z)$. An st-clause is such a clause which has got at most s literals, each one having at most t occurrences of predicates, functions, constants or variables; e.g. $edge(succ(E), x)$ is 1-4-clause.

Substitution is an assignment of terms to variables, e.g. $\theta = \{x \mapsto A, y \mapsto f(A, z)\}$. By applying a substitution to a clause, literal, or term, variables in clause, literal, or term are replaced by their images in the substitution, e.g. consider previous θ and $\gamma = l(x, y, w)$, then $\gamma\theta = l(A, f(A, z), w)$. The substitution θ for which $\gamma\theta$ is ground is called *grounding substitution.* Unification is a process which for two terms or atoms l_1 and l_2 finds such substitution θ that $l_1\theta = l_2\theta$; note that unification of two expression does not need to exist, e.g. $l_1 = p(x)$ and $l_2 = q(A)$.

An interpretation defines which atoms are true and which are false. An interpretation o is a model of clause γ ($o \models \gamma$) iff γ is true in the interpretation. The previous definition of an interpretation is rather an informal one. In order to do it formally, we would have to define a universe $\mathcal{U}$ and a mapping which maps each constant to an element of $\mathcal{U}$; for each predicate of arity a, it defines upon which a-tuple of elements $\mathcal{U}$ it holds, etc. However, we do not need such formal formulation, because it is sufficient for us to use *Herbrand interpretations* only. In Herbrand interpretations, each symbol maps to itself, i.e. it uses *Herbrand universe* and part of the interpretation's mapping is given, e.g. constant *Adam* maps to *Adam*. *Herbrand universe* consists of all ground terms which can be composed constant and function symbols. *Herbrand base* is a set of all ground atoms which can be formed out of predicate symbols and terms from Herbrand universe.

Note that we use the notation in which constants start with uppercase while the rest (e.g. variables) starts with lowercase. Constants may be seen as a special case of function symbols, precisely as function symbols of arity 0. However, that is not the case of the provided codes for the ILP homework. Also note that a predicate is defined by its name and arity; the same holds for a function.

For more information see [1, 2].

Exercise

- What is the difference between a term and an atom, e.g. $fatherOf(Adam)$ and $human(Adam)$?
- Is the clause $p(x, y) \lor \neg p(x, z)$ range-restricted?
- Is the st-clause $p(x, y) \lor p(A, B) \lor p(f(A), f(x)) \lor \neg q(f(x, A), B)$ a 3-3-clause?
- Show Herbrand universe and base for constants $\mathcal{C} = \{A, B\}$ and predicate $\mathcal{P} = \{p/1\}$.
- Show Herbrand universe and base for constants $\mathcal{C} = \{A, B\}$, functions $\mathcal{F} = \{f/2\}$ and predicates $\mathcal{P} = \{p/1\}$.
- Given theory $\Phi = \{\neg edge(x, y) \lor edge(y, x), \neg edge(x, x)\}$ find two Herbrand interpretations, from which the first one is a model of Φ and the second is not.
- Unify $\gamma_1 = edge(x, C)$ with $\gamma_2 = edge(A, B)$.
- Unify $\gamma_1 = path(x, f(x))$ with $\gamma_2 = path(g(y), y)$.

Θ-subsumption

One of the core concepts of ILP is θ-subsumption[1]. We say that γ_1 *θ-subsumes* γ_2, $\gamma_1 \subseteq_\theta \gamma_2$, iff there exists substitution θ such that $\gamma_1\theta \subseteq \gamma_2$ in the sense of literals. For example, for $\gamma_1 = p(x, y)$ and $\gamma_2 = p(A, z)$ it holds that $\gamma_1 \subseteq_\theta \gamma_2$. To check this by the definition, firstly find a substitution, e.g. $\theta = \{x \mapsto A, y \mapsto z\}$, and check the sets of literals, e.g. $\gamma_1\theta = \{p(A, z)\} \subseteq \{p(A, z)\} = \gamma_2$. Note here that the $=$ relation is a little bit overloaded, meaning that a clause is the same as a set of literals it contains. On the contrary, $\gamma_2 \not\subseteq_\theta \gamma_1$. But in general, it may happen that two clauses are *subsume-equivalent*, i.e. $\gamma_i \approx_\theta \gamma_j$ iff $\gamma_i \subseteq_\theta \gamma_j$ and $\gamma_j \subseteq_\theta \gamma_i$.

Exercise

List all pairs s.t. $\gamma_1 \subseteq_\theta \gamma_2$ from the following clauses:

(a) $p(x)$

(b) $q(y) \lor p(y)$

(c) $p(A) \lor p(B) \lor p(z)$

(d) $q(A)$

(e) $q(x) \lor p(z) \lor q(z)$

[1]We will need this concept in the following labs.

Generalizing Agent

In order to have our agent being able on-line efficiently PAC-learn, as explained in the lectures [3], we need two things for the agent to be polynomial. Firstly, Γ must be polynomial in the size of the problem, i.e, in $|\mathcal{P}|$, $|\mathcal{F}|$, and $|\mathcal{C}|$. Secondly, the update step must take at most polynomial time, thus the relation $\models$ must be computable in polynomial time as well.

For deciding whether $o \models \gamma$, it is enough to find one grounding substitution θ for which the following holds: i) all negative literals of $\gamma\theta$ are in o and ii) no positive literal of $\gamma\theta$ is in o. It is a hard task in generall, but in our range-restricted case, it can be decided in polynomial time as shown in the lecture. The algorithm, suited for our setting, therefore firstly finds all substitutions θ for which negative literals of $\gamma\theta$ are in o. Among these substitutions, it searches for one which violates the second condition, i.e. no positive literal of $\gamma\theta$ is in o. The first part can be organized as a tree search whose leaves have $\gamma\theta$ with some substitution θ; however, we are only interested in leaves with grounding substitutions, i.e. the leaves with ground clauses. The tree search for finding ground substitutions processes as follows given γ, θ and o

```
1. if $\gamma\theta$ is ground, end the tree search; test whether $\gamma\theta \models o$
2. select one non-ground literal $l$ from $\gamma$ which consists of atom $a$
       $\forall a_g \in o$
           if $a$ and $a_g$ can be unified, then repeat the search with $\gamma\theta'$ where $\theta' = unify(a\theta, a_g)$
```

Note that while searching a grounding substitution, we may select literal l (step 2) from negative literals only. This is possible because the input γ is a range-restricted clause. Thus, a substitution grounding negative literals grounds positive literals as well.

References

[1] Luc De Raedt. *Logical and relational learning.* Springer Science & Business Media, 2008.

[2] Shan-Hwei Nienhuys-Cheng and Ronald De Wolf. *Foundations of inductive logic programming.* Vol. 1228. Springer Science & Business Media, 1997.

[3] Filip Železný and Jiří Kléma. *SMU textbook.* 2017.