SMU ILP: Learning from Interpretations

In this tutorial, we will focus on learning the generalizing agent from the lecture in the context of inductive logic programming (ILP). In particular, we will focus on learning from interpretation, that is learning hypothesis H such that:

$$o \models H \ \forall o \in O^+$$
$$o \not\models H \ \forall o \in O^-$$

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 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. 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 ($\wedge$, $\vee$,...) 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. $path(X, Z) \vee \neg path(X, Y) \vee \neg path(Y, Z)$. A *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.

A 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 a 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 universum $\mathcal{U}$ 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 formed out of 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 constant, function, and predicate symbols start with lowercase, and variables start with uppercase. 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 tutorials. 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. $fatherOf(adam)$, $human(adam)$?
- Show Herbrand universe/base for constants a, b, functions $f/2$ and pred-

icate $p/1$.

- Is the following clause range-restricted? $p(X,Y) \vee \neg p(X,Z)$
- Is the following st-clause a 3-3-clause? $p(X,Y) \vee p(a,b) \vee p(f(a),f(X)) \vee \neg q(f(X,a),b)$
- Unify $\gamma_1 = edge(X,c)$ with $\gamma_2 = edge(a,b)$.
- Unify $\gamma_1 = path(X,f(X))$ with $\gamma_2 = path(g(Y),Y)$.

Generalizing Agent

For this and the upcoming sections, suppose that o is an observation, i.e. interpretation, γ is a FOL clause, θ is a substitution. The basic building blocks of the agent can be summerized in the following list:

- Γ – a set of all possible range-restricted *st*-clauses given predicates, constants, functors
- $\Phi = \left\{ \bigwedge_{i \subseteq I} \gamma_i \middle| I \subseteq [1 : |\Gamma|] \right\}$
- start with Φ hypothesis, i.e. the most specific CNF
- if agent's prediction of an observation is different from the observation's class, the agent udpates his hypothesis according to the delete operation
- $\text{delete}\left(\bigwedge_{i \in I'} \gamma_i, o \right) = \bigwedge_{\substack{i \in I' \\ o \models \gamma_i}} \gamma_i$

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 $|P|$, $|F|$, and $|C|$. Secondly, the update step must take at most polynomial time, thus the relation $\models$ must be computable in polynomial time as well; see following section.

Polynomial Testing of $\models$

Note that the following algorithm is polynomial since we use range-restricted clauses.

We recall that $o \models \gamma$ does not hold if and only if there is a ground instance $\gamma\theta$ of γ such that

1. atoms of all negative literals of $\gamma\theta$ are in o, and
2. no positive literal of $\gamma\theta$ is in o.

So, to determine $o \models \gamma$, the agent first finds all substitutions θ satisfying condition 1, and then checks if each of them satisfies condition 2. 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 γ 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.

Tutorial Task

The task for this tutorial is to implement a generalizing agent as described in the previous section (lecture) to the provided code. Firstly, download and unzip the archive with the source codes for this tutorial; install required SW (see *readme*). Your task is to implement the online learning agent into class *GeneralizingAgent* in file *agent.py*. The agent starts with a CNF composed of range-restricted *st*-clauses. At any time, the agent may be supplied by an observation by calling on his method *seeObservation*, which takes a *Sample*, i.e. an interpretation with label (*pos*/*neg*). Also at any time, the agent may be asked to give his current hypothesis out via his method *getHypothesis*. Thus, the place for your implementation of the agent behavior and $\models$ operator (by the search tree) is in the method *seeObservation*.

Selecting a dataset, i.e. file composed of lines representing a sample (*+/- singer(jackson),band(toto)...*), and a proper setting of parameters t and s is done at the end of file *tutorial1.py*. Creating your own dataset, changing t or s are also things you may try. You can also see the process of feeding the agent with observations in the file. Currently, the used range-restricted *st*-clauses' generator does not support function symbols. However, for some problems/datasets, they are not needed.

List of tasks:

1. Implement the agent (the search tree for $\models$ operator).
2. Run the agent on the lecture (triangle) instance. Recall that there are negative and positive samples of oriented graphs and the discriminative pattern is a (non-)presence of an unoriented triangle. See chapter 4 of [3], page 37–39, for more details about the example.

3. See the agent's final hypothesis – is it what you expected? Can it be made shorther? If yes, then how? Otherwise why not?

4. Compute by hand how many constant-free range-restricted 3-3-clauses for the triangle dataset are there. Is the number same as the length of the most specific range-restricted 3-3-clause? If not, explain.

5. Implement a brute force grounding for $\models$ (generating all possible groundings); python's package *itertools* may help you.

6. Compare the runtimes of the search tree grounder and the brute force one.

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.