Generalizing Logic Programs via Subsumption ⋆

Miguel A. Gutiérrez-Naranjo, José A. Alonso-Jiménez, and Joaquín Borrego-Díaz

Dept. of Computer Science and Artificial Intelligence – University of Sevilla
{magutier,jalonso,jborrego}@us.es

1 Introduction

In a Machine Learning system based on clausal logic, the main operation lies on applying an operator to one or more clauses with the hope that the new clauses give a better classification for the training set. Most of the systems use the subsumption order to carry out the transformation of programs. In the literature, different classes of operators on clauses can be found (e.g. [4, 5]). In these works, the emphasis is put on the specialization operators, which are operators such that the obtained clause is implied or subsumed by the original clause, and the generalization operators are considered the *dual* of the first ones.

In [1, 2] we present some *new* generalization operators for clausal learning, the Learning Operators under Subsumption (LOS) which allow us to generalize clauses in the subsumption order, i.e., if C is a clause, $\{\Delta/x\}$ is a LOS and $C\{\Delta/x\}$ is the output clause, then $C\{\Delta/x\}$ subsumes C.

The main property of these operators is that the LOS represent a *characterization* by operators of the subsumption relation between clauses (see Theorem 1). Based on these operators, we compute an algebraic expression of the subsumption order via a mapping which quantify the subsumption relation.

In this paper, we extend the study to programs. We define a new class of operators, the *composed LOS*, which act on the set of programs and they also represent a characterization of the subsumption relation between programs. Analogously to the clausal case, the minimum of the length of the chains of operators between two programs is the basis of a weak metric to quantify the closeness between programs.

2 Operators for clauses

Before defining the LOS, we need some previous definitions.

Definition 1. *A mapping $\Delta : Lit \rightarrow \mathcal{P}\mathcal{P}\mathbb{N}^+$ is called an* assignment *if there exists a term t such that, for every literal L, and for all set of positions $P \in \Delta(L)$, P is a subset of the set of positions of t in L.*

⋆ Work partially supported by the MCyT project TIC 2000-1368-C03-0 and the project TIC-137 of the *Plan Andaluz de Investigación*.

The idea of our operators is to graft a variable in the literals of the clause according to an assignment Δ.

Definition 2. *Let Δ be an assignment and x a variable. We say that the mapping $\{\Delta/x\} : \mathbb{C} \longrightarrow \mathbb{C}$*

$$C \mapsto C\{\Delta/x\} = \bigcup_{L \in C} L\{\Delta(L)/x\}$$

is a Learning Operator under Subsumption (LOS) *if for all literal L, if $\Delta(L) \neq \emptyset$ then $x \notin Var(L)$.*

For example[1], if $C_2 = sum(s(x), s(0), s(z)) \leftarrow sum(s(0), 0, s(0)), sum(x, s(0), z)$ we can get $C\{\Delta/y\} = sum(s(x), y, s(z)) \leftarrow sum(x, y, z)$ for an appropriate operator $\{\Delta/y\}$. The main property of these operators is that they are a *characterization* of the subsumption relation, as the following theorem claims.

Theorem 1. *Let C_1 and C_2 be clauses. Then $C_1 \succeq C_2$ iff there exists a finite sequence $\{\Delta_1/x_1\}, \ldots, \{\Delta_n/x_n\}$ of LOS such that $C_1 = C_2\{\Delta_1/x_1\} \ldots \{\Delta_n/x_n\}$*

We can quantify the closeness between clauses based on the number of operators of the shortest path of operators between clauses if subsumption holds.

Definition 3. *We define the mapping $dc : \mathbb{C} \times \mathbb{C} \to [0, +\infty]$ as follows:*

$$dc(C_2, C_1) = \begin{cases} min\{|\mathcal{C}| \, : \, \mathcal{C} \in \mathbf{L}(C_2, C_1)\} & \textit{if } C_1 \succeq C_2 \\ +\infty & \textit{otherwise} \end{cases}$$

The mapping dc is a *computable* quasi-metric and we have implemented an algorithm to compute it. We also provide upper and lower bounds for the quasi-metric under the assumption of subsumption.

Theorem 2. *Let C_1 and C_2 be two clauses such that $C_1 \not\subseteq C_2$. If $C_1 \succeq C_2$ then*

$$|Var(C_1) - Var(C_2)| \leq dc(C_2, C_1) \leq min\{2 \cdot |Var(C_1)| \, , \, |Var(C_1)| + |Var(C_2)|\}$$

3 Operators for programs

We extend the study of operators to generalize programs.

Definition 4. *The program P_1 subsumes the program P_2, $P_1 \succeq P_2$, if there exists a mapping $F : P_1 \to P_2$ such that $F(C) \succeq C$, for all $C \in P_1$.*

Definition 5. *A* composed LOS *is a finite set $\Theta = \{\Delta_1/x_1, \ldots, \Delta_n/x_n\}$ where $\{\Delta_i/x_i\}$ is a LOS for all $i \in \{1, \ldots, n\}$.*

For applying a composed LOS to a program we need an auxiliary mapping which associates one LOS to each clause of the program.

[1] We use the Prolog notation $A \leftarrow B_1, \ldots, B_n$ instead of $\{A, \neg B_1, \ldots, \neg B_n\}$

Definition 6. *Let P be a program and Θ a composed LOS. An* auxiliary mapping for applying (amfa) *is a mapping $a : P \to \Theta$ such that for all clause $C \in P$, the clause $C\{a(C)\}$ is not the empty clause. The program $P_a\Theta = \{C\{a(C)\} \,|\, C \in P\}$ is the program obtained by applying Θ to P via the* amfa a.

For example, if P is the program $P = \{C_1, C_2, C_3\}$ with

$$\begin{array}{l} C_1 = sum(0, s(s(0)), s(s(0))) \leftarrow sum(0, s(0), s(0)) \\ C_2 = sum(s(x), s(0), s(z)) \leftarrow sum(s(0), 0, s(0)), sum(x, s(0), z) \\ C_3 = sum(s(y), y, s(z)) \leftarrow sum(0, y, y), sum(y, y, z) \end{array}$$

we can get $P_a\Theta = \{D_1, D_2\}$ with D_1, D_2 the clauses $D_1 \equiv sum(0, x, x) \leftarrow$ and $D_2 \equiv sum(s(x), y, s(z)) \leftarrow sum(x, y, z)$ for appropriate a and Θ. Analogously to the clausal case, the composed LOS represent an operational characterization of the subsumption relation among programs, as the next theorem shows.

Theorem 3. *Let P_1 and P_2 be two programs. $P_1 \succeq P_2$ iff there exists a finite chain of composed LOS $\Theta_1 \ldots \Theta_n$ such that $P_2\Theta_1 \ldots \Theta_n \subseteq P_1$, where each Θ is applied via an appropriate* amfa a_i.

We also present a mapping for quantifying the closeness among programs. In this case the idea is also to take the shortest path of operators.

Definition 7. *We define the mapping $dp : \mathbb{P} \times \mathbb{P} \to [0, +\infty]$ as follows:*

$$dp(P_1, P_2) = \begin{cases} min\{|\mathcal{C}| \,:\, \mathcal{C} \in \mathbf{L}(P_1, P_2)\} & \textit{if } P_1 \succeq P_2 \\ +\infty & \textit{otherwise} \end{cases}$$

The mapping dp is *computable* and the set $\mathbb{P}$ endowed with dp is a *quantitative domain*. Since programs are finite set of clauses, the next theorem provides a method to compute the pseudo–quasi–distance of two programs

Theorem 4. *Let P_1 and P_2 be two programs.*

$$dp(P_1, P_2) = \max_{D \in P_2} \left\{ \min_{C \in P_1} \{dc(C, D)\} \right\}$$

Note that the mapping $dp^*(P_1, P_2) = \max\{dp(P_1, P_2), dp(P_2, P_1)\}$ is the Hausdorff metric based on the quasi-metric dc.

4 Related work and examples

In [3], Nienhuys-Cheng defines a distance for ground atoms and the Hausdorff metric d_h based on it to define a metric on sets of ground atoms. This metric can be easily extended to literals. The following example allows us to compare this metric with our quasi-metric. For all $n \geq 0$, let us consider the clauses

$$\begin{array}{ll} C_n \equiv sum(s^{n+1}(x_1), s^n(y_1), s^{n+1}(z_1)) & \leftarrow sum(s^n(x_1), s^n(y_1), s^n(z_1)) \\ D_n \equiv sum(s^{2n+1}(x_2), s^{2n}(y_2), s^{2n+1}(z_2)) & \leftarrow sum(s^{2n}(x_2), s^{2n}(y_2), s^{2n}(z_2)) \end{array}$$

The next table shows the values of the quasi-metric $dc(C_n, D_n)$ and the metric of Hausdorff $d_h(C_n, D_n)$ for several values of N as well as the time of computation on a Pentium III 800 Mhz. in an implementation for SWI-Prolog 4.0.11.

N	$dc(C_n, D_n)$		$d_h(C_n, D_n)$	
	Sec	Q–dist	Sec	Dist
64	0.02	3	0.11	$\sim 2.7\,10^{-20}$
128	0.06	3	0.21	$\sim 1.4\,10^{-39}$
256	0.1	3	0.43	$\sim 4.3\,10^{-78}$
512	0.26	3	0.93	$\sim 3.7\,10^{-155}$
1024	0.67	3	2.03	$\sim 2.7\,10^{-309}$

For all $n \geq 0$, $dc(C_n, D_n) = 3$. If we use the Hausdorff metric d_h based on d_{nc} we have that, for all $n \geq 0$, $d_h(C_n, D_n) = \frac{1}{2^n+1}$, which tends to zero in spite of the subsumption relation holds for all n.

5 Conclusions and future work

The operators presented in this paper might provide a general framework to specify learning process based on Logic Programming [1]. We present a family of operators which represents an operational characterization of the subsumption between programs. These operators provide a weak metric which captures the idea of closeness among programs based on subsumption.

The operators are not related to a specific system and they can be easily implemented and used in any system. The process of quantifying qualitative relations (as subsumption) is a hard and exciting problem which arises in many fields of Computer Science which is far from a complete solution. We present a contribution to its study by defining a computable weak metric on the set of logic programs in a natural way, as the minimum number of operators which map a clause into another. These operators and its associated weak metric can be a useful tool for the design of new learning algorithms based on subsumption.

References

1. M.A. Gutiérrez-Naranjo. Operadores de generalización para el aprendizaje clausal. Ph.D. thesis, Dept. of Computer Science and Artificial Intelligence, University of Seville, 2002
2. M.A. Gutiérrez-Naranjo, J.A. Alonso-Jiménez and J. Borrego-Díaz: A quasi-metric for Machine Learning. Proceedings VIII Iberoamerican Conference on Artificial Intelligence. Springer Verlag, 2002. To appear.
3. S-H. Nienhuys-Cheng: Distance between Herbrand interpretations: a measure for approximations to a target concept. Technical Report EUR–FEW–CS–97–05. Department of Computer Science, Erasmus University, the Netherlands, 1997. `http://www.few.eur.nl/few/research/pubs/cs/1997/eur-few-cs-97-05.pdf`
4. E.Y. Shapiro: Inductive Inference of Theories from Facts. Technical Report 624, Department of Computer Science, Yale University, New Haven, CT, 1981
5. P.R.J. van der Laag, S.-H. Nienhuys-Cheng: Completeness and properness of refinement operators in Inductive Logic Programming. Journal of Logic Programming, Vol 34, n.3, pp.: 201–225, March 1998.