An algebraic approach to inductive learning

Zdravko Markov
Department of Computer Science, Central Connecticut State University
1615 Stanley Street, New Britain, CT 06050, U.S.A.
E-mail: markovz@ccsu.edu

Abstract

The paper presents an approach to inductive machine learning based on a consistent integration of the generalization-based (such as inductive learning from examples) and metric-based (such as agglomerative clustering) approaches. The approach stems from the natural idea (formally studied within lattice theory) to estimate the similarity between two objects in a hierarchical structure by the distances to their closest common parent. The hierarchies used are subsumption lattices induced by generalization operators (e.g. lgg) commonly used in inductive learning. Using some results from the theory the paper defines a unified framework for solving basic inductive learning tasks. An algorithm for this purpose is proposed and its performance is illustrated by examples.

Introduction

Inductive learning addresses mainly classification tasks where a series of training examples (instances) are supplied to the learning system and the latter builds an intensional or extensional representation of the examples (hypothesis). The approaches to inductive learning are based mainly on generalization/specialization or similarity-based techniques. Two types of systems are considered here – *inductive learning from examples* and *conceptual clustering*. They both generate inductive hypotheses made by abstractions (generalizations) from specific examples and differ in the way examples are presented to the system (whether or not they are pre-classified). The hypotheses generated by these systems usually form a partially ordered set under some generality ordering. The properties of partially ordered sets are well studied in lattice theory. One concept from this theory is mostly used in inductive learning – this is the *least general generalization (lgg)* which given two hypotheses builds their most specific common generalization. The existence of an *lgg* in a hypothesis space implies that this space is a semi-lattice (the lgg plays the role of infimum). The idea behind the lgg is to make "cautious" (minimal) generalization. However this property of the lgg greatly depends on how similar are the hypotheses/examples used to build the lgg. For example there exist elements in the hypothesis space whose lgg is the top element (empty hypothesis). An obvious solution of this problem is to use a distance (metric) over the hypothesis/examle space in order to evaluate the similarity between the hypotheses/examples. Then the pair of hypotheses/examples with a minimal distance between them must be used for the lgg, thus producing possibly the minimal generalization over the whole set of hypotheses/examples. Various distance measures can be used for this purpose. The best choice however is a distance that is well coupled with the lgg used, that is the pair of the *closest hypotheses* must produce the *minimal lgg*. This is the problem we address in the present paper.

The next section introduces the algebraic notions used throughout the paper. Section 3 describes a general language independent algorithm called GSL for building lattice structures on a given set of examples. Section 4 illustrates the use of the algorithm within a series of languages commonly used in ML. Section 5 discusses related work and Section 6 concludes and gives directions for future work.

Quasi-Metric on semi-lattices

In this section we introduce a height-based distance measure on a join semi-lattice. (for a survey of metrics on partially ordered sets see (Monjardet, 1981)).

Definition 1 (Semi-distance, Quasi-metric). A *semi-distance (quasi-metric)* is a mapping $d : O \times O \to \Re$ on a set of objects O with the following properties $(a, b, c \in O)$:

1. $d(a, a) = 0$ and $d(a, b) \geq 0$.
2. $d(a, b) = d(b, a)$ (symmetry).
3. $d(a, b) \leq d(a, c) + d(c, b)$ (triangle inequality).

Definition 2 (Order preserving semi-distance). A semi-distance $d : O \times O \to \Re$ on a partially ordered set $(O, \preceq)$ is *order preserving* iff for all $a, b, c \in O$, such that $a \preceq b \preceq c$ it follows that $d(a, b) \leq d(a, c)$ and $d(b, c) \leq d(a, c)$

Definition 3 (Join/Meet semi-lattice). A *join/meet semi-lattice* is a partially ordered set $(A, \preceq)$ in which every two elements $a, b \in A$ have an infimum/supremum.

Definition 4 (Diamond inequality). Let $(A, \preceq)$ be a join semi-lattice. A semi-distance $d : A \times A \to \Re$ satisfies the *diamond inequality* iff the existence of $sup\{a,b\}$ implies the following inequality: $d(inf\{a,b\}, a) + d(inf\{a,b\}, b) \leq d(a, sup\{a,b\}) + d(b, sup\{a,b\})$.

Definition 5 (Size function). Let $(A, \preceq)$ be a join semi-lattice. A mapping $s : A \times A \to \Re$ is called a *size function* if it satisfies the following properties:

S1. $s(a,b) \geq 0, \forall a, b \in A$ and $a \preceq b$.

S2. $s(a,a) = 0, \forall a \in A$.

S3. $\forall a, b, c \in A$, such that $a \preceq c$ and $c \preceq b$ it follows that $s(a,b) \leq s(a,c) + s(c,b)$ and $s(c,b) \leq s(a,b)$.

S4. Let $c = inf\{a,b\}$, where $a, b \in A$. For any $d \in A$, such that $a \preceq d$ and $b \preceq d$ it follows that $s(c,a) + s(c,b) \leq s(a,d) + s(b,d)$.

Consider for example the partially ordered set of first order atoms under θ-subsumption. A size function $s(a,b)$ on this set can be defined as the number of different functional symbols (a constant is considered a functional symbol of arity zero) occurring in the substitution θ mapping a onto b ($a\theta = b$).

Theorem 1. Let $(A, \preceq)$ be a join semi-lattice and s – a size function. Let also $d(a,b) = s(inf\{a,b\}, a) + s(inf\{a,b\}, b)$. Then d is a *semi-distance* on $(A, \preceq)$.

A widely used approach to define a semi-distance is based on an order preserving size function and the diamond inequality instead of property $S4$. The use of property $S4$ however is more general because otherwise we must assume that (1) all intervals in the lattice are finite and (2) if two elements have an upper bound they must have a least upper bound (supremum) too.

Further, a size function can be defined by using the so called *height functions*. The approach of height functions has the advantage that it is based on estimating the object itself rather than on its relations to other objects.

Definition 6 (Height function). A function h is called *height* of the elements of a partially ordered set $(A, \preceq)$ if it satisfies the following two properties:

H1. For every $a, b \in A$ if $a \preceq b$ then $h(a) \leq h(b)$ (isotone).

H2. For every $a, b \in A$ if $c = inf\{a,b\}$ and $d \in A$ such that $a \preceq d$ and $b \preceq d$ then $h(a) + h(b) \leq h(c) + h(d)$.

Theorem 2. Let $(A, \preceq)$ be a join semi-lattice and h be a height function. Let $s(a,b) = h(b) - h(a), \forall a \preceq b \in A$. Then s is a *size function* on $(A, \preceq)$.

Corollary 1. Let $(A, \preceq)$ be a join semi-lattice and h be a height function. Then the function $d(a,b) = h(a) + h(b) - 2h(inf\{a,b\}), \forall a, b \in A$ is a *semi-distance* on $(A, \preceq)$.

General algorithm

Using a given set of examples E the algorithm builds a semi-lattice G, where E is the set of all maximal elements of G. The algorithm hereafter referred to as *GSL (Generalizing Semi-Lattice)* is as follows:

1. Initialization: $G = E$, $M = E$;
2. If $|M| = 1$ then exit;
3. Let T be the set of the least general generalizations (lgg) of all pairs of elements from the set M;
4. Let $h_{min} \in T$ be an element of T satisfying some minimality condition $min(T)$;
5. $DC = \{h_i | h_i \in M, h_{min} \preceq h_i\}$;
6. $M = M \setminus DC$;
7. $G = G \cup \{h_{min}\}$, $M = M \cup \{h_{min}\}$;
8. go to step 2.

The *computational complexity* of the algorithm is $O(n^3)$ excluding the lgg and min operations. The algorithm *terminates* if the chosen language does not contain *infinite descending chains*. This holds for propositional languages and atomic formulae (see next section).

The GSL algorithm can be used to solve two types of inductive learning tasks:

- *Conceptual clustering.* In this case the semi-lattice G will represent the concept hierarchy, where the successors of the root concept represent the splitting of the initial set of examples E. Note that the algorithm allows concept overlapping. This is possible because the set T_{min} (in the modified version of the algorithm) could contain more than one elements which possibly can cover a single element.
- *Concept learning.* In this case a class membership is supplied for each example from the set E. This information can be used to stop the further generalizations if a minimal element covering examples from different classes has been generated. Then this element is just skipped (not added to set M and its coverage – not excluded from M) and added to the set of induced hypotheses. The algorithm proceeds with other minimal elements (if such exist). Thus the generalization process stops before reaching the top element of the lattice (the latter clearly will be an overgeneralization).

The construction of the lgg and the computation of the function min (estimating the distance between the elements as defined in Corollary 1) are language specific and will be discussed in the following sections.

Languages

Attribute-value (propositional) language

An example or a hypothesis in this language consists of conjunctions of propositions (attribute-value pairs)

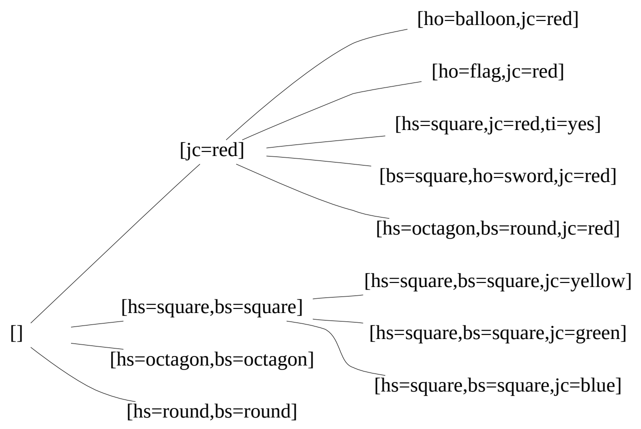
[ho=balloon,jc=red]
[ho=flag,jc=red]
[hs=square,jc=red,ti=yes]
[bs=square,ho=sword,jc=red]
[hs=octagon,bs=round,jc=red]
[jc=red]
[hs=square,bs=square,jc=yellow]
[hs=square,bs=square]
[hs=square,bs=square,jc=green]
[]
[hs=octagon,bs=octagon]
[hs=square,bs=square,jc=blue]
[hs=round,bs=round]

Figure 1: Propositional hypotheses for MONK1

$H = p_1 \wedge \ldots \wedge p_n$ which can also be represented as a set of propositions, i.e. $H = \{p_1, \ldots, p_n\}$. The partial ordering $\preceq$ here (usually called *covering*) is the subset inclusion $\subseteq$ between the hypotheses (examples). The lgg (the infimum) of two hypotheses is defined as their intersection, i.e. $lgg(H_1, H_2) = H_1 \cap H_2$. Then the minimality $min(T)$ condition chooses elements $h \in T$, which have been produced as lgg's of pairs of elements with minimal distances. That is $min(T) = \{h | h \in T, h = lgg(x,y), d(x,y) = min_{u,v \in T} d(u,v)\}$.

Two types of height functions can be used here. The first one is $v(H) = |H|$. This is the standard way of representing generality with nominal attributes by *dropping condition*. Another useful function (although in the general case not satisfying property $H2$ of height functions) can be defined by using the *coverage* of the hypothesis in terms of all maximal elements of the semi-lattice (the set E), that is $v(H) = |\{e | e \in E, H \preceq e\}|$. In contrast to the size of the hypothesis, this is a kind of *semantic* evaluation which reflects the generality of the hypothesis with respect to the examples used to generate it.

An illustration of the algorithm is shown in Figure 1, where some of the smaller elements of the set G are shown. The set E consists of the 61 *positive* examples of the well-known MONK1 (Thrun et al., 1991) database (the training sample). As a height function the size of the coverage of the hypothesis is used. Note that the produced lattice can be used both for concept learning (it contains the target hypothesis {hs=bs or jc=red}) and for conceptual clustering since the classifications of the examples were not used (the negative examples were skipped).

Atomic formulae

The language of the first order atomic formulae can be seen as an intermediate step between the propositional and the full relational language (e.g. Horn clauses). Its main advantage with respect to the propositional language is the possibility to define explicitly the *equality* of attribute values. The algebraic properties of this language are studied in (Reynolds, 1970), where the author shows that the set of atoms with same functors and arity augmented by adding a 'universal atom' and a 'null atom' forms a *complete lattice.* In this lattice the partial ordering is the *instance relation* ($\geq$) and the meet and the join operations are the well known *greatest common instance* (obtained by unification) and *least general generalization* (*lgg*, obtained by anti-unification).

Within the language of atomic formulae the GSL algorithm builds a semi-lattice, whose maximal elements are E and the atom with all distinct variables (the universal atom) is its minimal element. As in the propositional case the minimality condition chooses elements $h \in T$, which have been produced as lgg's of pairs of elements with minimal distances. That is $min(T) = \{h | h \in T, h = lgg(x,y), d(x,y) = min_{u,v \in T} d(u,v)\}$.

Several types of height functions can be used here. Some of them actually do not conform to the formal properties of height according to definition 6.

In (Reynolds, 1970) a function evaluating the generality of atoms is proposed. It is called $size(A)$ (note that this is not the formal size function according to Definition 5) and is defined as the number of symbol occurrences in A minus the number of distinct variables occurring in A. $A \geq B$ implies $size(B) \geq size(A)$ and $A \simeq B$ implies $size(A) = size(B)$. Also, for any A and B there is no chain from A to B whose length is greater than $size(B) - size(A)$. Unfortunately the Reynolds' $size$ function does not satisfy the second formal property of a height function and consequently the function d is not a quasi-metric.

A simplified version of the Reynolds' $size$ function is proposed in (Hutchinson, 1997). It is based on the number of functional symbols in the atom. Though formally a height, this function does not account properly for the variables in the atoms and consequently it is improper for the minimality condition in the algorithm.

Similarly to the propositional case a coverage-based function can be used for a height within the language of atoms. It is defined as $v(H) = |\{e | e \in E, H \geq e\}|$, where $\geq$ is the instance relation.

The algorithm shows a better performance with the Reynolds' size function and the coverage-based height compared with the Hutchinson's function. Figure 2 shows part of the resulting set G built by the algorithm running on a version of the MONK1 training sample represented as atoms (the same set of 61 positive examples used in the propositional case).

Horn clauses

Within the language of Horn clauses the GSL algorithm can be used with the θ-subsumption-based lgg (Plotkin, 1970). The formal background for this is the fact that under θ-subsumption as partial ordering the set of all Horn clauses with same head predicates forms a *semi-lattice.*

When a *background knowledge* is used the corresponding version of the *relative lgg* (*rlgg*) can be applied. That is, given a set of instances $\{P_1, P_2, ..., P_n\}$ of the

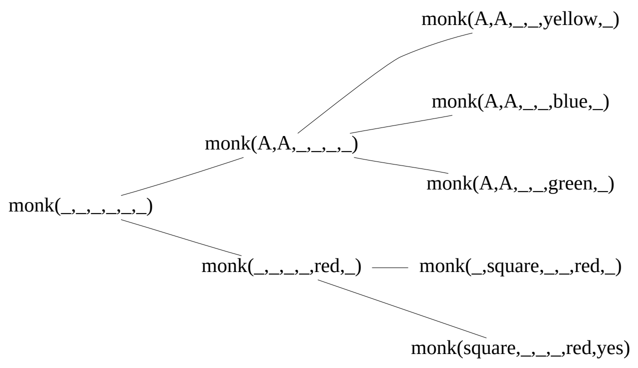

Figure 2: Relational hypotheses for MONK1

target predicate and a set of ground atoms BK as background knowledge the input of the GSL algorithm can be constructed as $E = \{P_1 \leftarrow BK, P_2 \leftarrow BK, ..., P_n \leftarrow BK\}$.

The major problem in this setting however is finding a proper height function. It is known that the Horn clause θ-subsumption semi-lattice contains *infinite* chains. Therefore the definition of a formal height function is not a trivial task. Practically the GSL algorithm needs an evaluation function representing the similarity (or distance) between the clauses with respect to their role in the concept learning problem. Thus similarly to the case of propositional and atomic languages a coverage-based height function could be used too. Currently we are investigating an approach to consistent integration of the *syntactical join operation* (θ-subsumption lgg) and a *semantic height function* using implication based coverage.

There are also direct approaches to define a proper metric on Horn clauses without height functions. Some of the most recent ones are (Hutchinson, 1997; Nienhuys-Cheng, 1997; Ramon, Bruynooghe, & Laer, 1998a, 1998b). These approaches define a simple metric on atoms and then extend it to sets of atoms (clauses or models) using the Hausdorff metric or other similarity functions. Because of the complexity of the functions involved and the problems with the computability of the models these approaches are usually computationally hard.

Related work

The algebraic approach to inductive learning is a very natural way to study the inherent to the area generalization and specialization hierarchies. These hierarchies represent hypothesis spaces that in most cases are partially ordered sets under some generality ordering. One of the first and most popular works within this approach is the Version Space framework (Mitchell, 1982). In this framework the space of all correct conjunctive hypotheses is maintained by using the boundary sets S and G, representing correspondingly the most specific and most general hypotheses. The version space is actually an equivalence class of hypotheses with respect to the inductive task conditions, i.e. covering all positive examples and no negative ones. Thus the goal of the system is by acquiring more examples to reduce this class eventually to a single hypothesis.

In the presence of background knowledge and in case of more complex hypothesis languages usually the more general approach of *refinement operators* is applied. Refinement operators are constructive means to build generalizations or specializations of hypotheses with respect to some generality ordering. In contrast to the Version Space approach refinement operators are used to search the hypothesis space containing not only correct (not covering negative examples) and complete (covering all positive examples) hypotheses. Thus is the case of top-down refinement the system starts from the most general hypothesis and further specializes it in order to avoid covering of negative examples. In this process some positive examples can also be excluded, therefore the search proceeds with other disjunctive components of the hypothesis. Conversely the upward refinement operators are used to generalize an initial too specific hypothesis in order to ensure that it covers as many as possible positive examples. The first study of refinement operators is (Shapiro, 1983), where the so called Model Inference System is introduced. This system performs downward refinement of clauses based on θ-subsumption ordering. An in-depth overview of the refinement operators used in inductive logic programming can be found in (van der Laag, 1995).

Another type of refinement operators used in ML are those which take as input two hypotheses and produce their common generalization or specialization. The most popular among these is the *least general generalization (lgg)* operator which given two hypotheses build their most specific common generalization. The existence of an *lgg* in a hypothesis space (a partially ordered set) directly implies that this space is a semi-lattice. Consequently some algebraic notions as finiteness, modularity, metrics etc. can be used to investigate the properties of the hypothesis space. A complete study of least generalizations and greatest specializations within the language of clauses can be found in (Nienhuys-Cheng & de Wolf, 1996).

Lgg's exist for most of the languages commonly used in ML. However all practically applicable lgg's (i.e. computable) are based on *syntactical* ordering relations. A relation over hypotheses is syntactical if it does not account for the background knowledge and for the coverage of positive/negative examples. For example dropping condition for nominal attributes, instance relation for atomic formulae and θ-subsumption for clauses are all syntactical relations. On the other hand the evaluation of the hypotheses produced by an lgg operator is based on their coverage of positive/negative examples with respect to the background knowledge, i.e. it is based on *semantic* relations (in the sense of the inductive task). This discrepancy is a source of many problems in ML, where overgeneralization is the most difficult one.

There exists a general semantic relation over hypotheses in all languages. It can be defined by the *set inclusion* relation between the sets of examples covered by the hypotheses. In (Champesme, Brézellec, & Soldano, 1995) it is called *empirical subsumption* relation. The empirical subsumption is a *preorder* and can be easily extended to a partial order by using the equivalence classes as elements. Unfortunately the lgg w.r.t. this relation does not exists in the general case (actually the intersection of two sets is their *lgg*, however it does not always have an explicit representation in the underlying language). In (Champesme et al., 1995) the empirical subsumption is used for reducing the class of equivalent hypotheses under the corresponding syntactical relation. Generally this kind of semantic relation is used as a preference criterion for evaluation of the hypotheses generated by refinement operators or *lgg*'s based on syntactical relations.

Concluding remarks

The paper is an attempt to combine theortical and practical research in ML. We use basic results from lattice theory to develop a unified inductive learning framework. We also introduce an algorithm and illustrate by examples its application in two areas of ML – conceptual clustering and concept learning. Compared to similar ones our approach has two basic advantages:

- It is language independent, i.e. it can be applied both within propositional (attribute-value) languages and within first order languages.
- It allows consistent integration of generalization operators with a semantic distance measure.

Clearly more theoretical and practical work is needed to investigate the advantages and drawbacks of our approach. In this respect we see the following directions for future work:

- On the theoretical side further efforts should be put into investigating the general conditions which a coverage-based evaluation function should satisfy in order to be a correct height function.
- The practical learning data often involve numeric attributes. Proper relations, lgg's and covering functions should be investigated in order to extend the approach for handling numeric data.
- Though the algorithm is well founded it still uses heuristics. This is because building the complete lattice is exponential and we avoid this by employing a hill-climbing strategy (choosing a single minimal element in Step 4). Obviously this leads to incompleteness. Therefore other strategies should be investigated or perhaps the semantic relation should be refined to incorporate these additional heuristics.
- Finally, more experimental work needs to be done to investigate the behavior of the algorithm in real domains (involving large amount of examples and noise).

Acknowledgments

Many thanks to the anonymous referees for the valuable comments and suggestions that improved the paper.

References

Champesme, M., Brézellec, P., & Soldano, H. (1995). Empirically conservative search space reduction. In Raedt, L. D. (Ed.), *Proceedings of ILP-95*, pp. 387–401. Dept. of Computer Science, K.U.Leuven.

Hutchinson, A. (1997). Metrics on terms and clauses. In van Someren, M., & Widmer, G. (Eds.), *Machine Learning: ECML-97*, Vol. 1224 of *Lecture Notes in Artificial Intelligence*, pp. 138–145. Springer-Verlag.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, *18*, 203–226.

Monjardet, B. (1981). Metrics on partially ordered sets – a survey. *Discrete Mathematics*, *35*, 173–184.

Nienhuys-Cheng, S.-H. (1997). Distance between herbrand interpretations: a measure for approximations to a target concept. Tech. rep. EUR-FEW-CS-97-05, Erasmus University.

Nienhuys-Cheng, S.-H., & de Wolf, R. (1996). Least generalizations and greatest specializations of sets of clauses. *Journal of Artificial Intelligence Research*, *4*, 341–363.

Plotkin, G. D. (1970). A note on inductive generalization. *Machine Intelligence*, *5*, 153–163.

Ramon, J., Bruynooghe, M., & Laer, W. V. (1998a). Distance measure between atoms. Tech. rep. CW 264, Katholieke Universiteit Leuven.

Ramon, J., Bruynooghe, M., & Laer, W. V. (1998b). A framework for defining a distance between first-order logic objects. Tech. rep. CW 263, Katholieke Universiteit Leuven.

Reynolds, J. C. (1970). Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence*, *5*, 135–152.

Shapiro, E. Y. (1983). *Algorithmic program debugging.* MIT Press.

Thrun, S. B., et al. (1991). The MONK's problems - a performance comparison of different learning algorithms. Tech. rep. CS-CMU-91-197, Carnegie Mellon University.

van der Laag, P. R. J. (1995). *An Analysis of Refinement Operators in Inductive Logic Programming.* Ph.D. thesis, Tinbergen Institute Research.