Shift of bias without operators *

CLAUDIO CARPINETO
Fondazione Ugo Bordoni
Via Baldassarre Castiglione 59, 00142 - Rome
E-mail: fubdpt5@itcaspur.bitnet

Abstract

Just as the order defined by generality over concepts allows concept induction to be performed without generalisation/specialisation operators, so too the order defined by set inclusion over concept languages may allow the shift to better concept languages during learning to be performed without language shift operators. We are currently investigating this idea and in this paper we report first results. We present a framework for sided language shift adapted from Mitchell's version space strategy, and discuss its theoretical requirements and practical limitations. We illustrate the ordering-driven versus operator-driven strategy to language shift with a problem example (i.e., version-space induction over variable-factored conjunctive concept languages) to which our framework has been successfully applied.

1. Introduction

One of the preferred strategies to incorporate bias in a concept learner is careful vocabulary choice of its concept language (e.g., (Utgo86), (Subr89), (Berg91)) .Any concept language designed prior to learning, however, may later turn out to be too small (strong bias) or too large (weak bias). Various approaches have been proposed to shift the bias in the course of learning (e.g. [Math89], [Paga89], (Utgo86)). Even though these approaches apply to different learning algorithms (e.g, decision trees, version spaces) and try to improve different quality measures associated with learning (e.g., consistency, conciseness, accuracy, efficency) they usually tackle the problem in the same manner, i.e. defining a set of language shift operators and carrying out a depth-first search through the space of possible extensions/restrictions of the current concept language aimed at finding a better language for the learning task at hand. Furthermore, since the number of admissible extensions/restrictions is generally intractably large, most of the approaches to inductive language shift rely on various heuristics to reduce the number of candidates and/or to cut down the search.

* Work carried out within the framework of the agreement between the Italian PT Administration and the Fondazione Ugo Bordoni

This is a common research paradigm; in machine learning it has been extensively used to solve the much more simple and well-understood problem of inducing concepts from examples. For this task, however, other methods have been proposed that rather than being based on the use of generalisation/specialisation operators are based on the ordering defined by generality over the concept space. Of the latter methods, the Candidate Elimination (CE) algorithm (Mitc82) is perhaps the best known. It maintains and updates two boundary sets, the set S containing the maximally specific concepts consistent with data and the set G containing the maximally general concepts consistent with data, which can be used to represent and find *all* consistent concepts (i.e. the version space). Hereafter we shall assume the reader is familiar with the basic CE algorithm.

As methods driven by ordering may present significant advantages over methods driven by operators, it seems to be important to address the question of whether they can be applied also to shift of bias. As we shall see, there are in fact structural similarities between concept induction and concept language induction that justify changing the CE algorithm with the aim of inducing concept languages rather than concepts.

2. A framework for sided language shift

What makes the CE algorithm work is that (a) the hypothesis space is partially ordered (a concept c_1 is *more general than* a concept c_2 if the set of instances covered by c_1 is a proper superset of the set of instances covered by c_2) and (b) "data" can be used to prune upper and lower elements of such space (positive instances rule out too specific concepts and negative instances rule out too general concepts). These two features have a natural counterpart in the shift-of-bias setting. First, the concept language space is partially ordered by set inclusion: we shall say that a language L_1 is *larger than* a language L_2 if the set of concepts expressible in L_1 is a proper superset of the set of concepts expressible in L_2. Second, the results of the learning process can be used to prune too large or too small languages. There are several ways in which this may happen. For instance, small languages may be not enough expressive, thus giving rise to inconsistency with data. Conversely, large languages may cause the learning algorithm to become unacceptably costly (for instance in terms of the number of training instances necessary for convergency), or may degrade its predictive capability (Berg91). The problem of sided language shift can be formulated more precisely in this way:

Given

- A set of training instances {I}.
- An inductive learning algorithm.
- A set of concept languages {L}.
- An evaluation function E that analyses the result of induction and states whether either the relative concept language is too small (E = -1) and has to be enlarged, or is the

right size (E = 0) and may be left unchanged, or is too large (E =+1) and has to be restricted.

Find

The concept languages that return E = 0.

To solve this problem the procedure described in fig.1, adapted from the version space strategy used in concept induction, can be employed.

Initialize the sets S and G, respectively, to the sets of maximally small and maximally large concept languages in {L}.

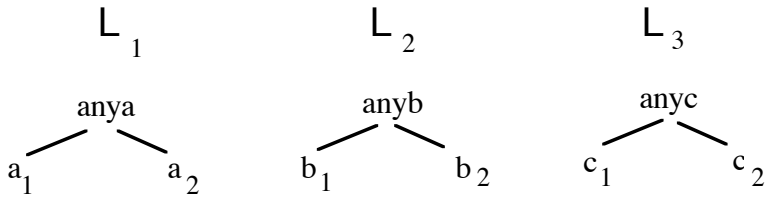
For each instance i in {I}
  For each language l in S
    Run the inductive algorithm
    If E = 0 **Then** continue
    else If E = -1 **Then** replace l with its larger languages, only to the amount required so that they return E = 0, and in such ways that each remains smaller than some language in G and there are no smaller languages in S.
    else If E = +1 **Then** drop l from S.
  For each language l in G
    Run the inductive algorithm
    If E = 0 **Then** continue
    else If E = +1 **Then** replace l with its smaller languages, only to the amount required so that they return E = 0, and in such ways that each remains larger than some language in S and there are no larger languages in G.
    else If E = -1 **Then** drop l from G.

Fig.1 The sided language shift algorithm

Besides the computational problems, a few of which will be outlined later, such a framework poses a fundamental theoretical requirement : the monotonicity of the evaluation function E. More precisely, the property to be satisfied is the following: if E = -1 for a language L then E = -1 for any language L*cL; likewise, if E = +1 for a language L then E = +1 for any language L*⊃L. This must hold for any instance distribution. The monotonicity of **E** depends on several factors : the particular inductive algorithm considered, the set of concept languages employed, the criteria used to restrict or enlarge the language. In the next section we introduce a problem example whose parameters guarantees the monotonicity of E.

3. A problem example: version space induction over variable-factored conjunctive concept languages

Our problem example is characterized by the following parameters. The inductive algorithm is the CE algorithm. Each instance is a conjunction of n attributes. The concept languages initially available are n tree-structured attribute languages, their leaves being the values present in the training instances. We shall use three very simple attribute languages:



The criteria used to shift the concept language are consistency (i.e. when the version space becomes empty the language has to be enlarged) and efficiency (i.e. when the size of the sets S or G exceeds a given threshold[1], which we assume equal to 2, the language must be restricted). We will examine both approaches to language shift in turn.

Language shift based on operators

We assume there are three language-shift operators (op_1, op_2, op_3), the application of op_i causing the current language (any attribute language, initially) to be multiplied[2] by the i-th attribute language. Since all chosen attribute languages contain the superconcept 'any', any operator application produces a larger language, as more usually done.

Suppose we begin with the attribute language L_1 {anya,a_1,a_2} and that the first instance (the first instance must be positive in the CE algorithm) is $a_1b_1c_1^+$.The corresponding version space is formed by S={a_1} and G={anya}. Suppose the next instance is $a_2b_2c_2^-$; the version space remains unchanged. If the algorithm is next given the instance $a_1b_2c_1^-$, the version space becomes empty. At this point a language-shift operator is selected (op_2), the new language L_{12} is generated and the version space in L_{12} is computed. It consists of the set S={a_1b_1} and the set G={anya-b_1}. Then suppose the algorithm is given $a_1b_1c_2^-$; this instance makes L_{12} inconsistent. Another operator is applied (op_3), which returns the language L_{123}. While this language is consistent with the data (its final version space is formed by S={$a_1b_1c_1$} and G={anya-b_1c_1}), it violates the size constraint, in that the version space in L_{123} corresponding to the first two instances is bounded by a three-valued set (G={anya-anyb-c_1, anya-b_1-anyc, a_1-anyb-anyc}). Because no more operators are applicable, we have to backtrack and change the operator applied to

---

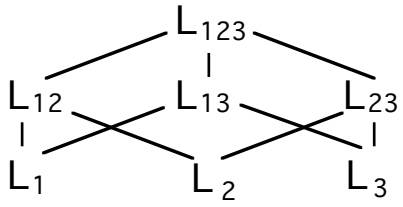
[1]The complexity of the CE algorithm depends on the square of the boundary sets' size (Mitc82).

[2]The *product* $L_{1,2}$ of two factor languages L_1 and L_2 is the set of concepts formed from the conjunctions of concepts from L_1 and L_2 (examples of product concepts are 'anya-anyb', 'anya-b_1', etc). The number of concepts in the product language is therefore the product of the number of concepts in its factors.

the initial language L_1, this once choosing the operator op_3. As also the newly-generated language L_{13} is inconsistent we have to backtrack again changing the initial attribute language until the language L'_{23} is generated, whose version space is non-empty (it contains the concept b_1c_1) and does not violate the size constraint.

Sided language shift

We now solve the same problem using the sided language shift strategy. The set of candidate concept languages is formed by all the languages that can be generated multiplying the attribute languages by one another. With n initial factor languages it is possible to generate $\sum_{k=1,n} n! / (n - k)! k! = 2^n - 1$ product languages; moreover, each product language is *larger* than any of its factor languages (note that this holds in general, because it is always possible to add the superconcept 'any' to each factor language). In our case the ordering graph over the product languages is:



Having chosen this set of languages we are sure the function E varies monotonically, i.e. the algorithm in fig.1 is correct. To prove this we have to show that both consistency and efficiency vary monotonically (in two inverse ways) with the ordering over the concept language space. The former is trivial: since a version space contains by definition all concepts consistent with data, then, for any instance distribution, if a language produces an empty version space then any *smaller* language will also produce an empty version space, and viceversa. The latter is more tricky because in general the efficiency of the CE algorithm is only weakly related to its concept language's size. In this case however it can be easily seen [Carp91] that the structure of product languages is such that for any pair of boundary sets in the factor languages the corresponding boundary sets in their product language are larger. Therefore inducing version spaces over product languages is never (i.e. for any instance distribution) less costly than over factor languages, and viceversa. Of course, the algorithm correctness does not guarantee that there will be some solution; this depends on the existence, for any training set, of at least one concept language in the hypothesis space that satisfies both criteria. In our example the solution exists (the language L'_{23}) and the algorithm in fig.1 will find it. The steps necessary to shift to the solution language are shown in fig.2; in each row of the matrix the version spaces associated with the languages in the sets S and G after any training instance are pictured (a,b,c are short respectively for anya,anyb,anyc).

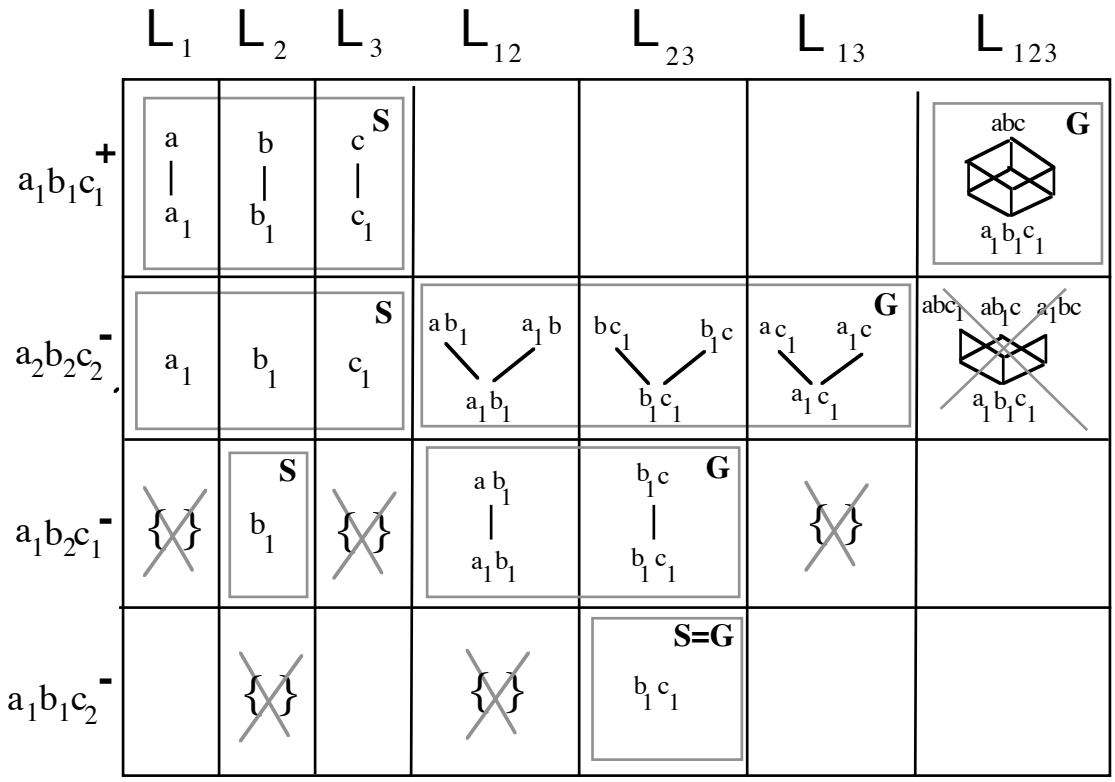


Fig.2 Version-space induction with sided language shift

4. Discussion

Sided language shift presents advantages and disadvantages over operator-based language shift. As in any method based on hypothesis space ordering, the advantages seem to chiefly rely on the possibility of discarding more candidate hypotheses (languages) and on the ease of selecting new candidates when the current hypothesis does not fit data. On the other hand, there appears to be two main disadvantages: limited representational capabilities due to a restricted set of concept languages, and exponential growth of the size of S or G. The latter seems to be particular important because the size of the boundary sets controls the number of times the two more costly operations involved (i.e., learning evaluation and concept induction after any language shift) have to be performed.

There are however specific cases in which these costs can be reduced. Elsewhere (Carp91) we proposed an approach to version-space induction over variable-factored conjunctive concept languages that can be seen as employing a one-sided language shift strategy (in fact, this research has originated from it). The use of a simple evaluation criteria (consistency) and of a specifically designed algorithm which was able to induce the new version space in any product language without reprocessing the instances already

seen in its factor languages allowed the overall complexity to be kept low. In fact, the method can be used to improve the efficiency of the standard CE algorithm when the initial tree-structured conjunctive concept language is consistent with data.

5. Conclusion

We have taken a first step toward a general alternative framework for inductive language shift which is not based on operators. Among the many possible dimensions along which this early model of sided language shift can be improved, exploring its applicability to other learning algorithm and other evaluation criteria (for instance the expressiveness/predictivity trade-off associated with classification learning) seems to be a crucial one.

Acknowledgements

I would like to thank Derek Sleeman, Pete Edwards, Filippo Fabrocini and Renato Petrioli for useful discussions on this topic.

References

[Berg91] Bergadano, F., Esposito, F., Rouveirol, C., Wrobel, S. (1991). Evaluating and Changing Representation in Concept Acquisition. *In Proceedings of EWSL-91*, Porto, Springer-Verlag.

[Carp91] Carpineto, C.(1991). Efficient induction of version spaces through constrained language shift. Technical Report of Fondazione Ugo Bordoni, 5T03591, Rome. To appear in *Proceedings of The Fifth Generation Computer Systems*, Tokyo, 1992

[Math89] Matheus, C.J., Rendell, L.A. (1989). Constructive induction on decision trees. In *Proceedings of the 11th IJCAI*, Detroit, Morgan Kaufmann.

[Mitc82] Mitchell, T.M.(1982). Generalization as Search. *Artificial Intelligence*, *18*.

[Paga89] Pagallo, G.(1989). Learning DNF by Decision Trees. In *Proceedings of the 11th IJCAI*, Detroit, Morgan Kaufmann.

[Subr89] Subramanian, D. (1989). Representational Issues in Machine Learning. In *Proceedings of the 6th International Workshop in Machine Learning*. Ithaca, Morgan Kaufmann.

[Utgo86] Utgoff, P. (1986). Shift of bias for inductive concept learning. In R. Michalski et al. (Eds), *Machine Learning* vol *II*. Morgan Kaufmann.