INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 01, January 2024, pp : 558-560

VERSION SPACES: A CANDIDATE ELIMINATION APPROACH TO RULE LEARNING

F. Arockiya Leticia[1]

[1]M.Sc., Department of Computer Science, Fatima College, Madurai, India.

ABSTRACT

The version space is progressively constructed by the candidate elimination method given a hypothesis space H and a set of instances E. One by one, the examples are added; by eliminating the assumptions that contradict the example, each example may reduce the version space. This is accomplished by the candidate elimination procedure, which updates the general and particular boundaries with every new case.

1. INTRODUTION

The recent focus on building knowledge-based systems within the artificial intelligence (AI) field has made rule and concept learning become increasingly essential tasks for AI researchers [2], [6]. For the challenging task of creating knowledge bases for use in such systems, a software that can extract general rules from training instances would be beneficial. We are still far from knowing effective, dependable ways for managing the combinatorics inherent in the learning job, despite some progress in this area [1], [9]. The process of learning concepts or rules. The National Science Foundation and the Advanced Research Projects Agency, under contract DAHC 15-73-C-0435, provided funding for this work.

2. DEFINITION AND REPRESENTATION

The term version space is used in this paper to refer to the set of current hypotheses of the correct statement of a rule which predicts some fixed action, A. In other words, it is the set of those statements of the rule which cannot be ruled out on the basis of training instances observed thus far. It is easy to see that this version space contains the set of all plausible rule revisions which may be made by a search algorithm in response to some new training instance. More exactly, assume that there is some rule R which correctly predicts action A for the class of training instances I+, but not for instances in the class I . In terms of their predictions on 1+ and I-, the rule version space connected to action A and the instances 1+ and I-is an equivalency class of rules. The rules that anticipate the same action but have different patterns indicated on their left hand sides are the elements that make up the version space.

We need to have a compact data representation for version spaces before we can write algorithms that reason in terms of them. In general, when the language of patterns for rules is complex, the number of probable alternatives can be very high, possibly infinite. Observing that the pattern matching technique defines a general-to-specific ordering on the rule pattern space is the key to an efficient representation of version spaces.

While condition (2) guarantees that the rule will match every training instance in I+, condition (1) ensures that the rule cannot match any training instances in 1-.

(1) and (?) will be required as well as sufficient requirements for membership of a rule statement in the version space, since the sets MGV and MSV are by definition complete.

A Version Space Represented by it 's Extrenal Sets

Rule Subgraph			Constraints on Subgraph Node Attributes			
Rule	subgraph	node name	atom type	number of non-hydrogen neighbors	number of hydrogen neighbors	number of unsaturated electrons
MSV1	v-w-x-y-z	v	carbon	1	3	0
		w	carbon	2	2	0
		x	carbon	2	2	0
		y	carbon	2	2	0
		z	carbon	>=1	any	0
MGV1	v-w-x	v	carbon	1	any	any
		w	any	2	any	any
		x	any	>=1	2	any
MGV2	v-w-x	v	carbon	1	any	any
		w	any	2	any	any
		x	any	2	any	any

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 01, January 2024, pp : 558-560

2.1 Version Spaces and Rule Learning:

Recall the rule learning task discussed earlier . A program is given examples of two classes of training instances, 1+ and I-. The program must determine some rule which will produce a given action, A, for each training instance in I+, but for no instance in I - . The candidate elimination algorithm presented here operates on the version space of all plausible rules at each step, beginning with the space of all rule versions consistent with the first positive training instance, and modifying the version space to eliminate candidate versions Knowledge Ar . q . - l30 Mitchell found to conflict instances. with subsequent training . The chief difference between the candidate elimination approach and the search approach discussed above is that search techniques select and modify a current best. Rather than select a single best rule version, the candidate elimination algorithm represents the space of all plausible rule versions, eliminating from consideration only those versions found to conflict with observed training instances. Thus, the candidate elimination approach separates the deductive step of determining which rule versions are plausible, from the inductive step of selecting a current best- hypothesis. At any step, the same heuristics used by search methods to infer the current best hypothesis may be applied to infer the best element contained in the version space. However, by refraining from committing itself to this inductive step, the candidate elimination algorithm completely avoids the need to backtrack to undo past decisions or reexamine old training instances. At the Rule version spaces for each Distinct predicted action are then generated from the training instances associated with the action. Subsequent data may be analyzed to modify the version space in a manner guaranteed to be consistent with the original data. The candidate elimination algorithm operates on the maximally general and maximally specific sets representing the version space. The set of maximally general rule versions (MGV) is initialized to a single pattern consisting of the most general statement in the language of rule patterns (a single atom graph with no constrained node at tributes) . The set of maximally specific versions (MSV) is initialized to a rule which contains as its pattern the first instance in I+. The initial version space represented by these external

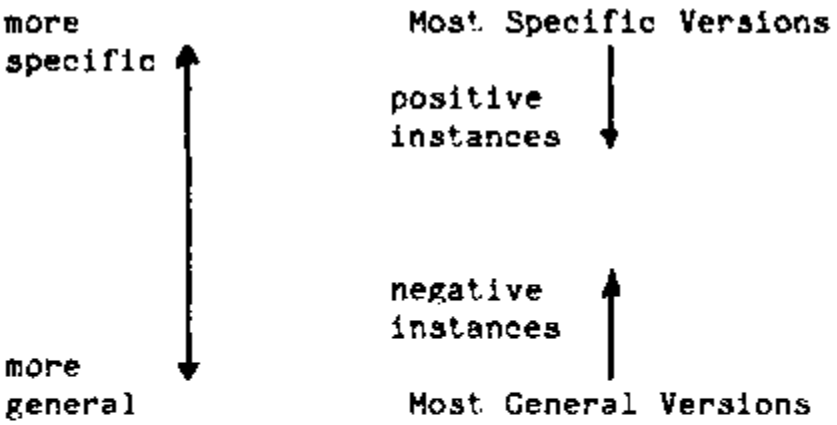

Figure 2

Effect of Positive and Negative
Training Instances on Version Space Boundaries

3. REASONING WITH VERSION SPACES

It seems that there are additional applications for an explicit representation of the space of plausible rule versions than what has been previously mentioned. The limitations on version spaces' general applicability are covered in this section along with some intriguing new applications.

3.1 Applicapility and Limitations:

The above-discussed version space approach to rule learning has some limitations. The algorithm is predicated on the idea that training instances should be assigned to 1+ and Ii s consistence, meaning that at least one rule in the rule space can produce the provided classification of training instances. This may be true in some domains, but in some "noisy" domains the procedure of There could be irregularities in the training cases themselves or in the instances'

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 01, January 2024, pp : 558-560

classification. The method will exclude all rule variants from consideration and force backtracking if the set of training cases is inconsistent. However, this is not worse than what is anticipated from other non-statistical techniques, as they all necessitate going backward in particular, it means that there isn't a rule in the provided rule language that can distinguish between 1+ and I-. This might happen when there is noisy data or when the rule language isn't complicated enough to express the given dichotomy between 1+ and I-. Eliminating only candidate versions that conflict with a fixed number of training instances greater than one* could be one way to make the candidate elimination algorithm more tolerant of noisy data. The cost of this extension would be a slower rate at which the version space boundaries converge toward one another. The nature of the partial ordering of version spaces may represent a second restriction on their universal application. Despite the complicated language of rule patterns, Meta-DENDRAL suggests that the size of these sets may be manageable for basic molecules. * That being said, it's likely that the size of these external sets may grow significantly for some domains. Knowledge of the interdependencies between the node characteristics is used in Meta-DENDRAL to remove rule statements that are conceptually comparable but syntactically different. Therefore, redundancy in the rule language has no negative effect on the size of the external sets. Adding domain-specific restrictions on the elements of the version space that are permitted is a second potential strategy for reducing the size of the maximum general and maximally specific version sets. AI programs frequently make use of task domain knowledge. In contrast, the efficiency of search procedures and their need for backtracking appear to be adversely affected by both the number of branches in the partial ordering and the depth of the branches.

3.2 Other Uses for Version Spaces:

Version spaces provide an explicit representation of the range of plausible rules. With this explicit representation, the program acquires the ability to reason more abstractly about its actions. The program is aware of more than the current best hypothesis - it has available the entire range of plausible choices. This view of version spaces suggests their use for tasks other than the particular rule learning task

4. CONCLUSION

The importance of careful selection of training instances for efficient and reliable learning has been stressed . Final result is presentation of Since version spaces represent the range of rule versions which cannot be resolved by the current training data, they also summarize the range of unencountered training that will be useful in selecting among competing rule versions. By constructing a training instance which matches some, but not all , of the maximally general versions, the program may be able to determine which of several potentially important attributes should be specified in a rule . On the other hand, by constructing training instances which match a given most general version, but not its most specific counterpart, the program may determine how specific the constraint on a given attribute must be useful.

5. REFERENCES

[1] B. G. Buchanan and T. M. Mitchell, "Model- Directed Learning of Production Rules", Proceedings of the Workshop on Pattern-

[2] Directed Inference Systems, Honolulu, Hawaii, May 1977,

[3] E. A. Feigenbaum, B. G. Buchanan, and J.Lederberg, "On Generality and Problem Solving: A Case Study Using the DENDRAL Program", in

[4] Machine Intelligence j5, B. Meltzer and D. Michie, eds., American Elsevier, 1971, pp. 165-190.

[5] M. Minsky, "Steps Toward Artificial Intelligence " , in Computers and Thought. E. A. Feigenbaum and J. Feldman, eds., McGraw-Hill,

[6] N. Y. , 1963, pp. 406-450.

[7] U. T. M. Mitchell and G* M. Schwenzer, "A Computer Program for Automated Empirical 13C-NMR Rule Formation1', Heuristic Programming Project Working Paper HPP-77-^, Stanford University, February, 1977.

[8] R. J. Popplestone, "An Experiment in Automatic Induction", in Machine Intelligence £, B. Meltzer and D. Michie, eds., Edinburgh University Press, 1969.

[9] E. Shortliffe , MYCIN: Computer-Based Medical

[10] Consultations. American Elsevier, N* Y», 1970.