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

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 01, January 2024, pp : 192-195

e-ISSN : 2583-1062

Impact Factor : 5.725

A CANDIDATE ELIMINATION APPROACH TO RULE LEARNING IN VERSION SPACE

K. Dharshini[1]

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

ABSTRACT

The study of strategies for picking up broad concepts or rules from a set of training instances is a key area of research in artificial intelligence. A guaranteed method is provided for solving this problem that finds all rule versions that are consistent with a collection of positive and negative instances for training without going backtracking. The proposed algorithm makes use of a representation of that rule space that is in agreement with the observed training data. Candidate rule versions that are discovered to clash with each new instance are removed from this "rule version space" in response to new training instances. The application of version spaces is explained in relation to Meta-DENDRAL, a software that acquires knowledge about chemical spectroscopy rules.

Keywords: Candidate elimination, machine learning, concept learning, Version space, rule learning, concept formation Meta-DENDRAL.

1. INTRODUCTION

Rule learning and concept learning are crucial AI research goals, particularly in constructing knowledge-based systems. Extracting general rules from training instances is essential for such systems. Despite some success, there is still a lack of efficient methods for controlling combinatorics inherent in learning tasks. The task involves determining a rule P -> A, where P is a set of conditions or constraints from a predefined language that must be satisfied for action A to be invoked, and the learned rule must apply to all instances from I+ but not to I-. The search paradigm is a popular rule learning approach where a rule is determined based on the first training instance and revised based on legal alterations. This process is viewed as a concept formation problem, where the goal is to learn the correct statement of a rule, which is then applied to the correct set of instances. The search paradigm faces two challenges: determining possible rule changes based on training instances must align with past performance, and backtracking may be necessary when incorrect decisions are discovered in subsequent instances, requiring a different search tree branch. This paper introduces a candidate elimination algorithm for learning rules that goes beyond the search paradigm. It uses a method to represent and update the space of all rule versions, eliminating rules that conflict with observed training instances. The algorithm ensures that all rules are consistent with observed data without backtracking or reexamining past instances.

2. VERSION SPACES

This section presents a rule learning candidate elimination method that preserves and updates a representation of every conceivable rule version. It offers ways to express and work with this space, making sure that no matter which order training instances are presented, all rule versions match the observed training data without going back.

2.1 Definition and Representation

The version space refers to the set of current hypotheses of the correct statement of a rule that predicts a fixed action, A. It contains all plausible rule revisions made by a search algorithm in response to a new training instance. For example, if a rule R correctly predicts action A for the class of training instances I+ but not for instances in the class I-, the rule version space of I+ and I- is defined as the set of all such rules. To write programs that reason in terms of version spaces, a compact data representation is necessary. The key to an efficient representation lies in observing that a general-to-specific ordering is defined on the rule pattern space by the pattern matching procedure used for applying rules. The version space can be represented in terms of its maximal and minimal elements according to this ordering. The general-to-specific ordering is generally a partial ordering, meaning that given any two rules, one cannot always say that one is more general than the other. Therefore, when all elements of the version space are ordered according to generality, there may be several maximally general versions(MGV) and maximally specific versions(MSV).

A rule statement belongs to the version space of I+ and I- if it is less general than or equal to one of the maximally general versions and less specific than or equal to one of the maximally specific versions.

2.2 An Example: Meta-DENDRAL

An algorithm for describing version spaces and learning production rules to characterize molecular behavior in chemical spectroscopy has been implemented by the Meta-DENDRAL program. The version of this program that establishes guidelines for molecular substructures connected to carbon-13 nuclear magnetic resonance spectrum data is the main focus of this work.

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 : 192-195

Rule subgraph			Contraints on Subgraph Node Attribute			
Rule	Subgraph	Node name	Atom Type	Number of non-hytrogen Neighbors	Number of hytrogen 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	Carbon	2	any	any
		X	Carbon	>=1	2	any
MGV2	v-w-x	v	Carbon	1	any	any
		W	Carbon	2	any	any
		X	Carbon	2	any	any

Figure 1

The figure 1 depicts a version space represented by its extreme sets.

In terms of maximally general rule versions (rules MGV1 and MGV2) and maximally specialized rule versions (rule MSV1), the program represents a version space. Each node in the rule pattern represents an atom in a molecular structure, and it is described in a complex network language of chemical subgraphs. There are four limited attribute values for each subgraph node. The occurrence of a peak connected to atom "v" in a specific spectral range is the outcome that all rules anticipate. The subgraph needs to fit into the molecule graph and have attribute constraints that match the relevant node in the molecule graph in order to match the rule pattern to a training instance (molecular graph). Classes 1+ and I- in this software denote groupings of molecules for which the stated spectral. In the general-to-specific ordering, Figure 1 depicts a version space with several hundred rule versions, comprising three versions and other versions in between. It can be described as one maximally specific version (MSV1) and two maximally general variants (MGV1 and MGV2). All nodes and node attribute constraints that are consistent with all training instances in I+ are present in the single most specific version. Since neither of the general versions is "above" the other in the general-to-specific partial ordering, two general versions are needed in this instance. Any rule that matches an element of I- will be more generic than either MGV1 or MGV2. The maximally generic versions' superset of nodes, found in MSV1, corresponds with the general-to-specific ordering.

2.3 Version Spaces and Rule Learning

The rule learning task involves a program determining a rule that produces a specific action for each training instance in 1+ but not in I-. The candidate elimination algorithm is presented, which operates on the version space of all plausible rules at each step, starting with the first positive training instance and modifying it to eliminate conflicting versions.

The method known as the candidate elimination strategy divides the process into two parts: the inductive phase of choosing a current best hypothesis and the deductive step of identifying probable rule variants. It excludes just those rule versions that are inconsistent with training instances seen, thereby representing the space of all conceivable rule versions. By using this method, all correct variants of the rule are discovered once all training data has been presented, eliminating the need to go back and review previous training examples. This method provides two significant learning benefits.

2.4 Revisiting the Meta-DENDRAL Example

The meta-DENDRAL program implements a candidate elimination algorithm using version spaces. It determines predicted actions for training instance classes 1+ and I-, based on positive and negative training instances. Rule version spaces for each action are generated from the training instances, and subsequent data analysis ensures consistency with the original data. This algorithm is used to predict molecule graphs' peak appearances. The candidate elimination algorithm uses maximally general and maximally specific sets to represent the version space. The maximally general rule versions (MGV) are initialized to a single pattern, while the maximally specific versions (MSV) are initialized to a rule with the first instance in I+. This initial version space contains all rules matching the first training instance.

Training instances are considered individually, and conflicting rule versions are removed from the version space by shifting. As seen in figure 2 the maximally specific and maximally general boundaries of the version space towards each other.

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

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 01, January 2024, pp : 192-195

e-ISSN : 2583-1062

Impact Factor : 5.725

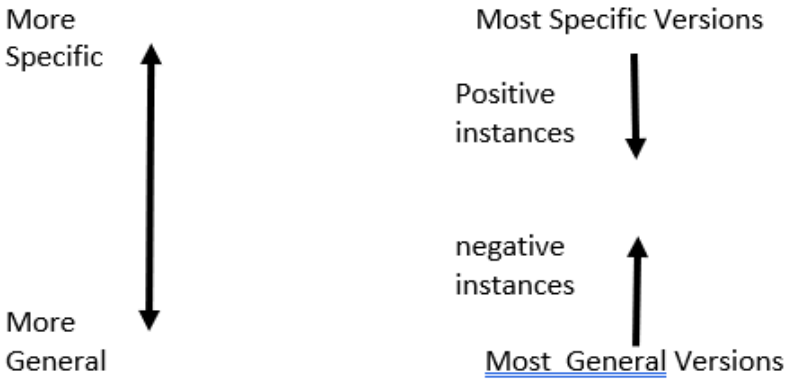

Figure 2

The study explores the impact of positive and negative instances on version space boundaries.

The generality of MSV elements is increased by positive training examples, whereas the specificity of MGV elements is increased by negative training instances. Since any version outside the present version space bounds is inconsistent with earlier training data, the maximally specific set cannot be replaced by a more generic or specific set.

It is important to replace elements in a model with versions that are at least slightly more particular in order to replace them with a new training instance. To achieve consistency with prior 1+ instances, constraints derived from model elements are added. Since negative items are already maximally specific, they must be removed from the collection. Elements from the model must be substituted with elements that fit the new positive instance and are at least more general if the new training instance is part of I*. Elements that are not more precise than a minimum of one element from the model are excluded. All rules that are incompatible with the new training instance will be eliminated as a result of the new maximum general and maximally specific sets bounding the space of fill rules consistent with the observed data.

3. USING VERSION SPACES FOR REASONING

This section discusses the potential uses of an explicit representation for the space of plausible rule versions, while also highlighting limitations on their general applicability.

3.1 Applicability and Limitations

The version space method to rule learning that training instances are consistently assigned to 1+ and I-. This assumption holds true in certain noisy domains, but inconsistent training examples or an unreliable procedure could occur in others. Backtracking will be necessary if the algorithm removes all rule versions due to inconsistent training examples. Still, compared to other nonstatistical techniques that necessitate backtracking, this is not all that bad. When noisy data or complex rule language are present, the version space may collapse to the null space, indicating that no rule in the provided rule language can distinguish between 1+ and I-.

The candidate elimination algorithm can be improved by eliminating candidate versions that conflict with a fixed number of training instances, resulting in a decrease in the rate at which version space boundaries converge.

The partial ordering of rule versions may result in version spaces with certain constraints. The maximally general and maximally specialized sets of versions are tiny for simple languages. According to Meta DENDRAL, these sets might be doable for basic molecules but might grow too big for some domains. To ensure that duplication in the rule language does not effect the size of extremal sets, syntactically distinct rule statements are eliminated using information about the interdependencies of node characteristics. The size of maximum general and maximally specialized version sets may also be restricted by domain-specific restrictions on components that are permitted in the version space.

The version space approach is limited by the need for reliable training data and the risk of large maximally general and maximally specific sets. It assumes a partial ordering over rule patterns, which is always satisfied by the pattern matcher. The approach is most efficient in domains with a non-branchy partial ordering but deep branches. The efficiency of the algorithm and extremal sets remains unaffected by branch depth, but is negatively affected by the number of branches. Search procedures' efficiency and need for backtracking are affected by both branch depth and partial ordering.

3.2 Additional Uses of Version Spaces

Version spaces offer an explicit representation of the range of plausible rules, enabling a program to reason more abstractly about its actions. This allows it to be aware of more than the current best hypothesis, offering the entire range of plausible choices. This view suggests their use for other tasks.

3.3 Selecting New Training Instances

Version spaces represent the range of rule versions that cannot be resolved by current training data and summarize the range of unencountered training institutions that can be useful in selecting among competing rule versions. By constructing a training instance that matches some but not all maximally general versions, a program can determine which potentially important attributes should be specified in a rule. Generating training instances can also help limit the size of maximally specific and maximally general sets, allowing examples to discriminate among competing extreme versions.

3.4 Combining Separately Obtained Outcomes

Version spaces are a method for merging sets of rules from different training data sets. By intersecting version spaces of two rules, the result is consistent with the union of the data sets, similar to the candidate elimination algorithm's result when running on the union of the training data.

Version spaces offer a concise summary of past training instances and a representation of all plausible rule versions. They allow a program to indicate its knowledge about the correct rule form, making it useful for intelligent training instances selection and merging sets of independently generated rules.

4. CONCLUSION

Version spaces, represented by maximally general and maximally specific versions, are ideal for learning rules from sequentially presented training instances. A candidate elimination algorithm is demonstrated, finding rule versions consistent with all instances, without backtracking.

Version spaces offer a concise summary of past training instances and a representation of all plausible rule versions, allowing a program to indicate its uncertainty about the correct rule form, making it useful for intelligent training selection.

5. REFERENCES

[1] M. Minsky, "Steps Toward Artificial Intelligence", in Computers and Thought. E. A. Feigenbaum and J. Feldman, eds., McGraw-Hill, N. Y. , 1963, pp. 406-450.

[2] T. M. Mitchel l 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.

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

[4] E. Shortliffe, MYCIN: Computer-Based Medical Consultations. American Elsevier, N* Y», 1976.

[5] H. A. Simon and G. Lea, "Problem Solving and Rule Induction: A Unified View", in L. W. Gregg, ed. , Knowledge and Cognition. Lawrence Erlbaum Associates, Potomac, Maryland, 197**, pp. 105-127.