Submitted for the 9th International Machine Learning Workshop, Scotland, 1992

Transformation of Version Space with Constructive Induction: The VS* Algorithm

Janusz Wnek
Center for Artificial Intelligence
George Mason University
4400 University Dr., Fairfax, VA 22030, USA

JanuszWnek@aic.gmu.edu
tel. (1-703-993 1717) fax (1-703-993 3729)

Abstract

In inductive learning, learned descriptions can be either overgeneralized or undergeneralized, and finding the proper balance is difficult, particularly when only insufficient data is available. For this reason, the classification of instances which are not covered by any rule, or covered by many rules, is often postponed to the testing phase. During this phase, different measures are applied to decide how to classify particular instances. To improve this situation, a method of inductive learning is proposed which enables the change of representation space in the learning process through the extension of attribute domains. Constructive induction and generalized descriptions of remaining classes are used in the method to improve classification rules of a given class. Two types of generalization are used: generalization of training examples, and generalization of the representation space. The generalization of the representation space is in the form of hypothesis-driven constructive induction (HCI). HCI takes inductive rules generated from primary training examples, and constructs examples for additional learning. Empirical evidence shows that the method cuts overgeneralized areas and fills up undergeneralized gaps of classification rules obtained by selective learning. The paper explains the method using diagrammatic visualization.

Key words: Concept learning, Constructive induction, Diagrammatic visualization, Empirical evaluation

1. Introduction

The area of learning from examples has been widely studied for many years, and has received much attention in recent years, as is indicated by the large number of references in the machine learning bibliography (Stefanski, Wnek & Zhang, 1990). Although learning form examples is the best understood type of machine learning, the ultimate goal, *to learn from experience as effectively as humans do*, is still in the distant future.

In inductive learning paradigm, a learning system receives a set of training examples, each labeled as belonging to a particular class. The system's goal is to produce general classification rules that will correctly classify new unknown examples. The process is artificially guided, or biased, by imposing general restrictions on the representation language and on the search space. In this paper, a method is presented which improves the effectiveness of the search of the space of possible hypotheses.

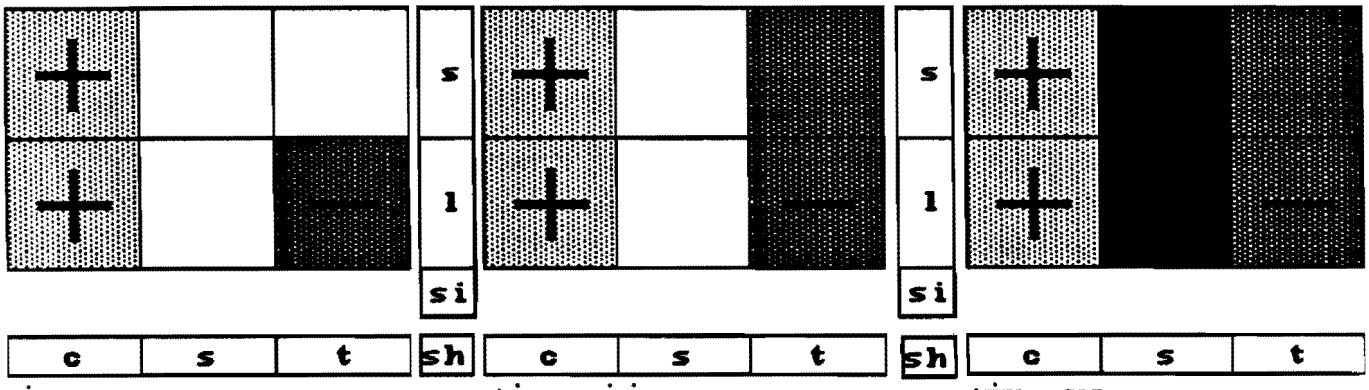
In the case of inductive learning, an inductive system must reason from specific instances to general rules. An effective search means a tradeoff between the quality of the classification rule found and the time consumed. This is the rationale to primarily concentrate the initial search to the neighborhood of the training examples as opposed to exhaustive search techniques, or random search techniques. The generalized examples in the form of classification rules are used to classify unseen instances. To determine the class of an unknown instance, many matching techniques have been developed. Classification is simple if only one classification rule covers the instance. But if more then one rule covers the instance or none of the rules covers this part of the instance space, the decision has to be based on partial, or flexible matching. These techniques take into consideration the distance between the description of the instance and the generalized descriptions of learned classes. The class which is the closest is chosen. In the case of a tie, another distance is calculated, or simply the most "popular" class is chosen.

There are many reasons for applying sophisticated classification techniques, e.g., imprecise concepts, noise in data, or a concept is not represented by training examples. In simple cases, however, the classification rules should precisely discriminate all instances. Therefore, it is proposed to verify the rules obtained by further searching the hypotheses space. The search is continued assuming that the classification rules become examples in a changed representation space. The verified rules not only discriminate precise concepts but they also increase performance accuracy when applied along with flexible matching techniques in more complex problems (Wnek, 1992).

Since the features used in the changed representation are generated from inductive hypotheses, the process is yet another kind of *hypothesis-driven constructive induction (HCI)* (Wnek & Michalski, 1991). However, the nature of the change in representation is different. Wnek & Michalski proposed (1991) the addition of constructed attributes and/or removal of irrelevant attributes. This type of induction was called HCI/a. Here, the extension of domains of individual attributes is proposed and therefore the induction is called HCI/v.

The proposed method is general. In this paper, however, examples and the hypotheses space are described in the VL1 language, an extension of the propositional calculus. VL1 is a fairly rich language that includes conjunction, disjunction, and set-membership operators. Consequently, the rule space of all possible VL1 discrimination rules is quite large. To search this space, the STAR algorithm is used (Michalski, 1983). The STAR algorithm has been implemented in several systems, such as Rigel (Gemello, Mana & Saitta, 1991), POSEIDON (Zhang, 1990), CN2 (Clark & Niblett, 1989), and AQ15 (Michalski et al., 1986). Here only some of features defined in the AQ15 system are used. One of them is the "trim" operator that controls levels of generalization of consistent descriptions, (Fig. 1). The operator can be set to one of three values: "gen," "spec," or "mini." In short, "gen" builds the most general descriptions, "spec" builds the most specific descriptions, and "mini" stands for minimal, or simplest descriptions.

Three examples of generalization in a simple domain are given in Fig.1. The domain is described by two attributes: **size** and **shape**. The **size** attribute can have two nominal values: small and large, the **shape** attribute can have three nominal values: circle, square, and triangle. Two positive examples are marked with +, one negative example is marked with -. Light gray areas mark "positive" concept description, dark gray areas mark "negative" concept description. Black areas mark both "positive" and "negative" concepts, i.e. both concepts cover instances represented by these areas. White areas do not belong to any concept.



trim = spec
The most specific descriptions

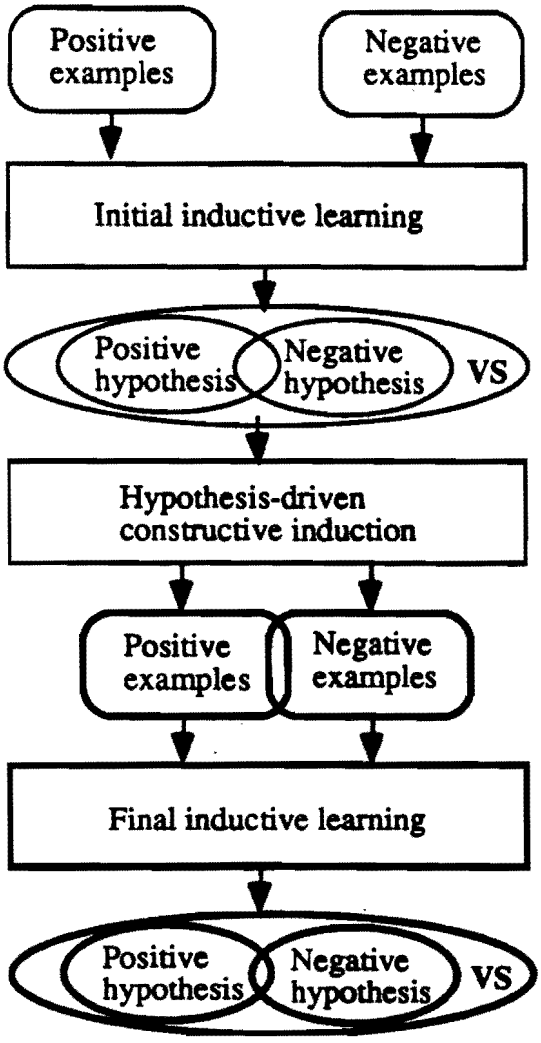
trim = mini
The minimal descriptions

trim = gen
The most general descriptions

Figure 1. Different types of generalization obtained by AQ15 under control of the "trim" operator.

2. VS* Learning Method

VS* method (version space with the STAR) is based on a three-step inductive generalization process (Fig.2).



Original representation language:
conjunctions of attribute values

Original representation space element:
point

Examples:
[x1 = 8] & [x2 = 4] : Positive
[x1 = 3] & [x2 = 10] : Negative

HCI/v transformed representation language:
conjunctions with internal
disjunction of attribute values

HCI/v transformed representation space element :
set of points

Examples:
[x1 = 6..13] & [x2 = 5..8, 10..14] : Positive
[x1 = 0..19] & [x2 = 16..19] : Negative

Figure 2. Three-step inductive learning with HCI/v constructive induction

In the first step, a version space[1] is built (Mitchell, 1978). The learning goal is to build descriptions of the "positive" class with the smallest possible number of errors of commission. This step also prepares the "negative" class description as an upper bound for the next inductive step. Thus, the method builds the most specific classification rules for the "positive" concept, and the most general classification rules for the "negative" concept. The rules are represented in conjunctive form with internal disjunction allowed.

[1] There are some differences with the original concept of version space. Firstly, the upper bound (G set) is represented by the most general "negative" class description. It covers all negative examples, and no positive examples. Secondly, the HCI/v method uses different strategy to build the version space. Both lower and upper bound ("positive" and "negative" concept) are learned using STAR method (Michalski, 1983).

In the second step, the description space is generalized. Each classification rule becomes an example in the new representation. The domain of new instances and class membership is inherited from the domain of classification rules and related class descriptions, respectively. In this step the contribution of constructive induction to the proposed method is revealed. Hypothesis-driven constructive induction, HCI/v, extends domains of attributes with new multi-valued features and selects the only relevant features to create new training examples.

The third inductive learning step finds final descriptions. The final descriptions are searched in the version space transformed by the HCI/v method. Since the positive examples now reflect larger areas of the description space, information that was initially missing is now assessed. The negative training examples, the most general upper bound, control proper generalization.

The VS* method finds a balance between a specific description of a given concept and a general description of a complementary concept. This is achieved first, by the determination of both the specific and the general descriptions and, second, after a representational change, finding the best description in between. It is worth noting that the same final result could be obtained without any change in the representation space. One could generate a training set that consists of all instances represented in classification rules, and use an inductive learner to obtain final classification rules. This would be however, a highly ineffective process, both in terms of time and memory.

3. An Illustrative Learning Problem

Given

- background knowledge: two attributes, vector representation space (x1 x2). Each of the attributes has 20 linear values (0..19) (Fig. 3);
- a set of positive examples of a concept (cells marked with + in Fig. 3);
- a set of negative examples of the same concept (cells marked with - in Fig. 3).

Determine

- a concept description which is a generalization of the positive examples that does not cover any of the negative examples.

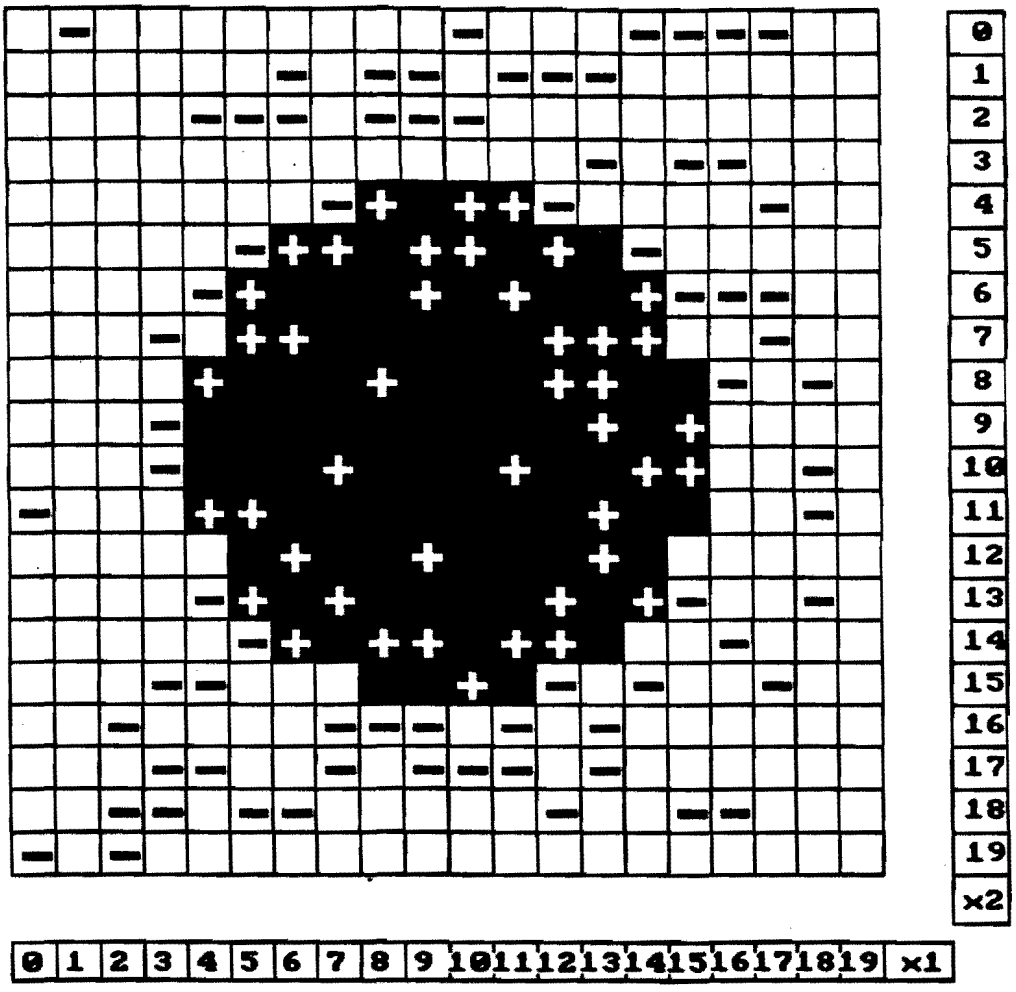


Figure 3. The learning domain with positive and negative examples of the CIRCLE concept.

The problem has many solutions that are consistent with the training set. Two of them are represented in Fig. 4, and were obtained using the learning program AQ15. The first solution resulted from setting the trim parameter to "spec" value, the second was obtained with "gen" value. Both solutions carry significant error. The first, a too specific description, has 16 errors of omission, the second, a too general description, has even more errors of commission. The question arises whether the program was given too little data to learn the target concept, or the solution lies somewhere between the specific and general descriptions. If it does, can we find a justifiable solution, anyway? How can we generalize the "specific" description, or specialize the "general" description? The task is not simple, because while learning, the algorithm is not aware of any errors of commission, or omission. At this point, the only confirmation of correctness of a solution is the consistency with the training set. Hence, let us prepare a new training set (based on obtained classification rules) that will consist of positive and negative training examples and generalize them again. Once again, the special feature available in AQ15 is used, which is due to

the same representation of training examples and learned concepts[1]. The result of this additional search is the same as in figure 2. The learned concept matches exactly the target concept.

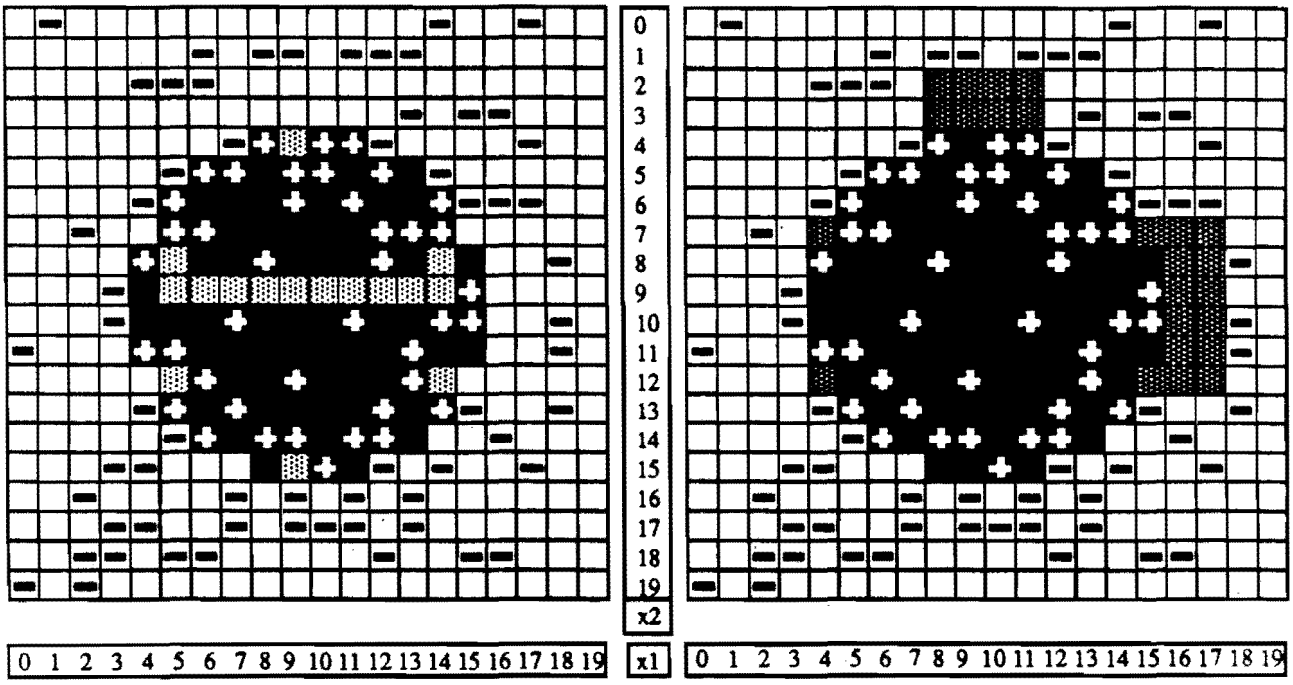
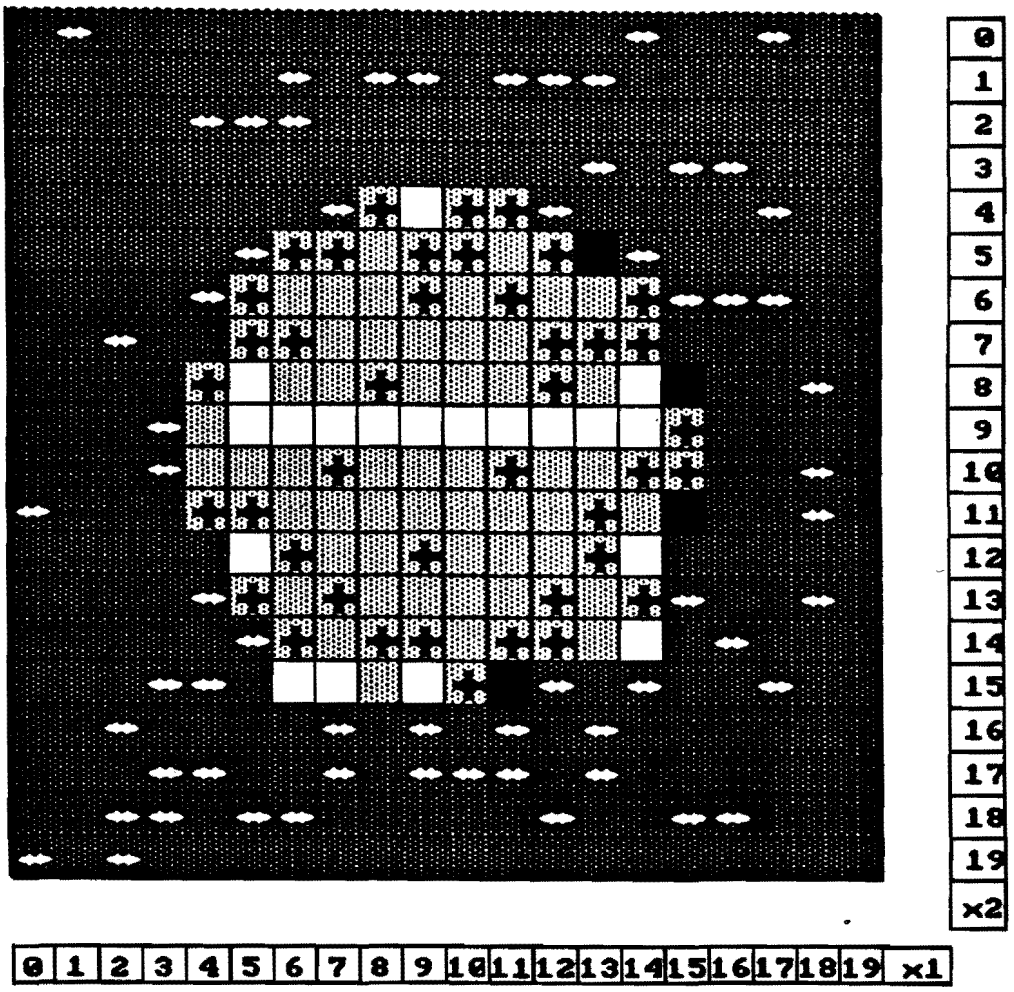


Figure 4. CIRCLE learned from examples.
The most specific description (left), the most general description (right).

4. The CIRCLE Learning Problem Solved by VS* Method

Figure 5 shows initial representation space, with positive and negative examples, and classification areas of "positive-spec" and "negative-gen" concepts that define the version space. Below the diagram, the classification rules are listed.

[1] The same mechanism allows AQ11 (the predecessor of AQ15) to treat the concept descriptions themselves as negative examples while it is building the nonoverlapping concept descriptions for subsequent classes in learning multiple concepts. In order to obtain a nonoverlapping set of discrimination rules, AQ11 takes as its positive instances all known instances in class i, and as its negative instances all known instances in class j ($j \neq i$) *plus* all conjuncts that make up the discrimination rules for previously processed classes k ($k < i$) (Cohen & Feigenbaum; 1982).



Light gray areas - "positive-spec" concept
Dark gray areas - "negative-gen" concept

White areas are not covered by any concept
Black areas are covered by both concepts

"Positive-spec" concept hypothesis:
[x1=6..13] & [x2=5,6,7,8,10,11,12,13,14] or
[x1=5,14] & [x2=6,7,10,11,13] or
[x1=8,10,11] & [x2=4,15] or
[x1=4,15] & [x2=8..11]

"Negative-gen" concept hypothesis:
[x2=0..3,16..19] or
[x1=0..3,16..19] or
[x1=0..5] & [x2=14..19] or
[x1=0..7,12..19] & [x2=0..4] or
[x1=15..19] & [0..8, 11..19] or
[x1=11..19] & [x2=15..19] or
[x1=0..4] & [x2=0..7,12..19] or
[x1=0..5,13..19] & [x2=0..5]

Figure 5. The concepts defining version space

The *white* and *black* instances do not have any unique classification. The *white* instances are not covered by any rule, whereas the black instances are covered by both the "positive" and "negative" rules. If the learning process is stopped at this point, the classification of these instances would rely on a partial matching with "positive" and "negative" classification rules.

The inductive system, however, continues learning. The new positive examples reflect areas that were covered by the "positive" concept (light gray), and the new negative examples reflect areas covered by the "negative" concept (dark gray)[2]. After making the conversion from classification rules to training examples in the changed representation language, the final inductive learning step of VS* yields rules describing the CIRCLE (Fig. 6) and the corresponding diagram (Fig. 3). It can be observed that not all "white" instances were generalized to the final concept description.

[x1=8..11] & [x2=4..15] or
[x1=4..15] & [x2=8..11] or
[x1=5..14] & [x2=6..13] or
[x1=6..13] & [x2=5..14]

Figure 6. The final description of the CIRCLE concept

5. Conclusions

The VS* method integrates four important components of inductive learning paradigm: version space (Mitchell, 1978), STAR methodology (Michalski, 1983), constructive induction (Michalski, 1978; Matheus, 1989; Rendell & Seshu, 1990), and inductive bias (Utgoff, 1986). VS* utilizes strong inductive bias in the form of version space to control inductive generalization processes. VS* uses STAR methodology to build the version space and to search it after imposing HCI/v constructive transformations. In pictorial terms, VS* cuts overgeneralized areas and fills up undergeneralized gaps of classification rules obtained by selective learning. The VL1 language and the single-representation feature of the AQ15 learning system help in facilitating the implementation of the VS* method. The VS* method works in multidimensional attribute spaces and with multiple concepts (Wnek, 1992).

In the future, the HCI/v component of the VS* method will first be combined with another type of hypothesis-driven constructive induction, HCI/a, and later with a data-driven constructive induction method. DCI methods find interesting patterns within the raw data before any hypothesis is created. This protects the learning system from losing important information.

---

[2] The system can treat the ambiguous instances (black) as positive, negative or empty examples (Michalski et al., 1986). Here we treat these instances as empty, i.e., not covered by any classification rule.

Acknowledgement

It is a pleasure to thank Thomas Arciszewski and Mike Hieb for their comments on an earlier draft of this paper.

This research was done in the GMU Center for Artificial Intelligence. Research of the Center is supported in part by the Defense Advanced Research Projects Agency under the grants administered by the Office of Naval Research No. N00014-87-K-0874 and No. N00014-91-J-1854, in part by the Office of Naval Research under grants No. N00014-88-K-0226, No. N00014-88-K-0397 and No. N00014-91-J-1351, and in part by the National Science Foundation grant No. IRI-9020266.

References

Cohen, P.R., and Feigenbaum, E.A., The Handbook of Artificial Intelligence, Vol.3, pp.424-425, 1982.

Clark, P., and Niblett, T., "The CN2 Induction Algorithm," *Machine Learning*, Vol. **3**, pp. 261-284, 1989.

Gemello, R., Mana, F. and Saitta, L., "Rigel: An Inductive Learning System," *Machine Learning*, Vol. **6**, pp. 7-35, 1991.

Matheus, C., "Feature Construction: An Analytic Framework and Application to Decision Trees," *Ph.D. Thesis*, University of Illinois, 1989.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," *Artificial Intelligence*, Vol. **20**, pp. 111-161, 1983. (Reprinted in Readings in Machine Learning, J.W. Shavlik and T.G. Dietterich, eds., Morgan Kaufmann Pubs., 1990).

Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N., "The AQ15 Inductive Learning System: An Overview and Experiments," ISG 86-23, Dept. of Computer Science, University of Illinois, 1986.

Mitchell, T.M., "Version Spaces: An Approach to Concept Learning," *Ph.D. Thesis*, Stanford University, 1978.

Rendell, L., and Seshu, R., "Learning Hard Concepts Through Constructive Induction: Framework and Rationale," *Computational Intelligence*, Vol. **6**, pp. 247-270, 1990.

Stefanski, P., Wnek, J., and Zhang, J., "Bibliography of the Recent Machine Learning Research 1985-1989," in *Machine Learning: An Artificial Intelligence Approach*, Vol.3, Y. Kodratoff and R.S. Michalski (eds.), Morgan Kaufmann Pubs., 1990.

Utgoff, P.E., Machine Learning of Inductive Bias, Kluwer Academic Publ., 1986.

Wnek, J., "Transformation of Representation Space using Multiple Valued Attributes," *Reports of Machine Learning and Inference Laboratory*, MLI 92-01, Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1992.

Wnek, J., and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments," in *Proceedings of the IJCAI-91 Workshop on Evaluating and Changing Representation in Machine Learning*, pp. 13-22, Sydney, Australia, 1991.

Zhang, J., "Learning Flexible Concepts form Examples: Employing the Ideas of Two-tiered Concept Representation," *Ph.D. Thesis*, University of Illinois, 1990.