Correspondence

Splitting and Merging Version Spaces to Learn Disjunctive Concepts

Tzung-Pei Hong, *Member*, *IEEE Computer Society*, and Shian-Shyong Tseng, *Member*, *IEEE*

Abstract—We have modified the original version space strategy in order to learn disjunctive concepts incrementally and without saving past training instances. The algorithm time complexity is also analyzed, and its correctness is proven.

Index Terms—Disjunctive concept, incremental learning, merge, split, version space.

1 Introduction

A famous incremental learning strategy, called the "version-space" strategy [5], [6], has been used in some real learning systems. However, it is mainly applicable only to learning conjunctive concepts. For learning disjunctive concepts, Mitchell proposed the "multiple-version-spaces" strategy [5], which maintains several boundary sets at the same time. Manago and Blythe integrated the top-down and bottom-up learning strategies to learn disjunctive concepts [4]. Both the above mentioned strategies must process training instances in batch fashion and are, thus, unsuited to incremental learning. Murray proposed the "multiple-convergence" strategy [7], which generalizes the idea of version space boundary sets to obtain disjunctive concepts. Hong and Tseng [2] proposed the "incremental multiple version-space" strategy, a modification of the multiple version-space strategy for incremental learning. However, the latter methods require saving previous positive training instances for reprocessing during the learning process. Since incremental learning is suited to cases in which training instances arrive intermittently, the number of instances may grow very large.

In this paper, we propose a new version-space-based learning strategy, the "version-space split-merge" (VSSM) learning strategy, which is able to learn disjunctive concepts well incrementally, but without keeping track of past training instances. This strategy simultaneously maintains several version spaces, each of which excludes all negative training instances and includes some positive training instances. Whenever a new training instance arrives, these version spaces are then split and merged to form new ones, satisfying the requirement of being able to learn disjunctive concepts.

2 The "Version-Space Split-Merge" Learning Strategy

The algorithm is presented in detail below:

Initialize G_0 to contain only the most general hypothesis in the entire hypothesis space.

REPEAT

Await the next training instance I;
IF I is positive THEN
IF a version space whose S set covers I already exists THEN do nothing;
ELSE IF a version space exists whose S set after including I is still consistent with its corresponding G set THEN minimally generalize the S set of this version space to cover I;
ELSE create a new version space whose S set contains only I and whose G set consists of the hypotheses in G_0 that are more general than I;
IF I is negative THEN
minimally specialize G_0 to exclude I;
FOR each version space V DO
minimally specialize G to exclude I;
IF V becomes inconsistent after excluding I THEN
split S set into disjuncts that exclude I;
FOR each disjunct D DO call the merge procedure to process D;
UNTIL all training instances have been exhausted.

The merge procedure is described below:

IF a version space whose S set is more general than D already exists THEN do nothing;
ELSE IF a version space exists whose S set after merging with D is still consistent with its corresponding G set THEN minimally generalize the S set of this version space to cover D;
ELSE create a new version space whose S set contains only D and whose G set consists of the hypotheses in G_0 that are more general than D.

3 Example

Each member of a given set of instances can be described as an unordered pair of simple objects characterized by three nominal attributes: shape (e.g., circle, triangle), color (e.g., red, blue), and size (e.g., large, small). Assume VSSM processes the following training instances:

1. {(Large Blue Circle) (Large Red Triangle)} positive,
2. {(Large Blue Triangle) (Large Blue Triangle)} positive,
3. {(Large Blue Triangle) (Small Blue Triangle)} negative,
4. {(Large Red Triangle) (Small Blue Circle)} positive, and
5. {(Large Blue Circle) (Large Blue Triangle)} negative.

The learning process is shown in Fig. 1. Note that each instance can be thought of as a most specific hypothesis.

4 Time Complexity Analysis

Define a unit operation as a specialization or generalization process [1], [3] for the purpose of having a processing time criterion. Let s_{max} and g_{max} denote, respectively, the maximum numbers of hypotheses in sets S and G appearing in a version space. Also assume the maximum number of version spaces appearing in the entire learning process is v_{max} and the maximum number of disjuncts split from S is d_{max}. Let T_p and T_n denote, respectively, the time complexity the VSSM learning algorithm requires to deal with one positive training instance and one

- *T.-P. Hong is with the Department of Information Management, I-Shou University, Kaohsiung County, Taiwan, 84008, Republic of China. E-mail: tphong@csa500.isu.edu.tw.*
- *S.-S. Tseng is with the Department of Computer and Information Science, National Chiao-Tung University, Hsinchu, Taiwan, 30050, Republic of China. E-mail: sstseng@cis.nctu.edu.tw.*

Manuscript received 24 July 1996; revised 13 Mar. 1997.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 104498.

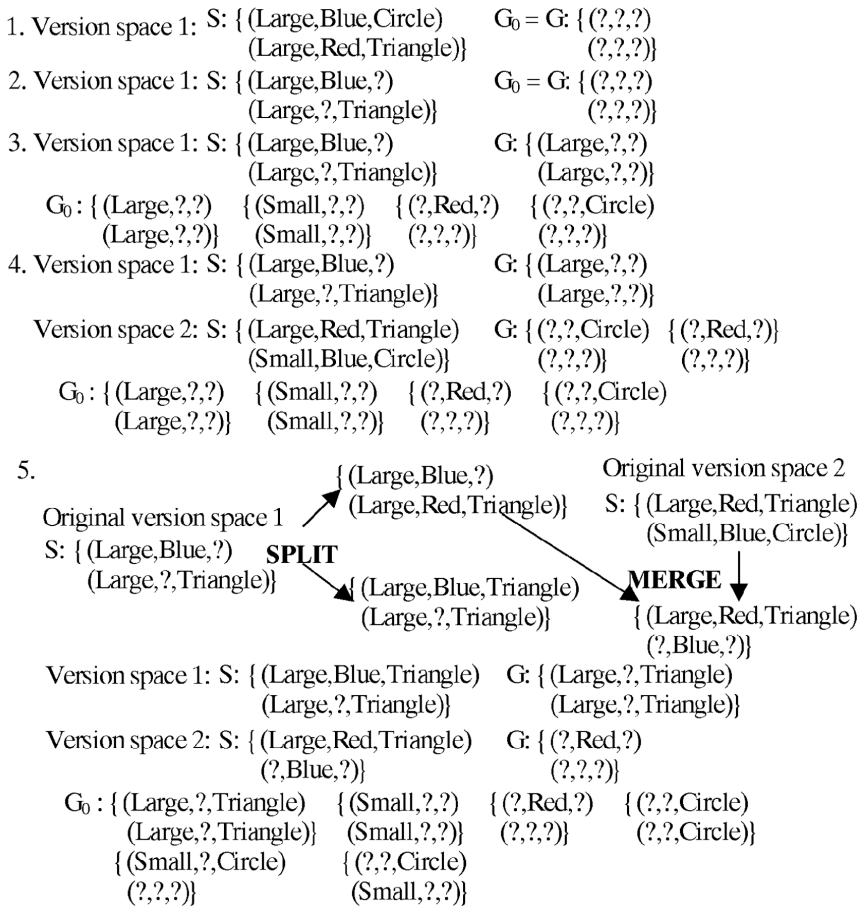

Fig. 1. Illustration of the VSSM learning strategy.

negative training instance, and let M denote the time complexity the merge procedure requires to deal with one disjunct. From the above algorithm, we obtain:

$$M = O\begin{pmatrix} v_{max} \times s_{max} + v_{max} \times \\ (s_{max} + g_{max} \times s_{max}) \\ + s_{max} + s_{max}^2 + g_{max} \end{pmatrix} = O\big(v_{max} \times g_{max} \times s_{max} + s_{max}^2\big),$$

$$T_p = O\big(v_{max} \times g_{max} \times s_{max} + s_{max}^2\big),$$

and

$$T_n = O\big(v_{max} \times \big(g_{max}^2 + d_{max} \times \big(v_{max} \times g_{max} \times s_{max} + s_{max}^2\big)\big)\big).$$

Note that this is a higher degree of time complexity than that of the original version space learning algorithm [5] since the problem VSSM solves involves learning disjunctive concepts. This is not necessarily, however, a higher degree of time complexity than those required by the other version-space-based algorithms for learning disjunctive concepts since it depends on the value of d_{max}.

5 Correctness of the "Version-Space Split-Merge" Learning Algorithm

The following theorems can easily be proven:

Theorem 1. *The VSSM learning algorithm can always terminate.*

Theorem 2. *When an inconsistent version space appears, previous positive training instances included in S are also included in the union of disjuncts D obtained by splitting S.*

Theorem 3. *Each version space derived by the VSSM learning algorithm will exclude all negative training instances, and all the version spaces gathered together will include all positive training instances.*

6 Discussion and Conclusion

Several points concerning the VSSM learning algorithm are discussed in more detail below:

1. When inconsistencies occur, the split disjuncts, rather than the positive training instances they include, are processed. Generalization of the S sets of the other version spaces are then processed to a higher level of abstraction.
2. When a new positive training instance arrives, two or more version spaces may still be consistent after including the newly arrived positive training instance. Nevertheless, the choice of version spaces for generalization doesn't affect the final correctness of the VSSM learning algorithm. Taking any hypothesis from each version space as disjunct still enables learning of disjunctive concepts.
3. The learning process is order-dependent in terms of efficiency, but not in correctness of concept learning. It is preferable to have negative training instances processed as early as possible to reduce the amount of reprocessing of past training instances. Positive training instances with high degrees of similarity should also be processed in consecutive order so generalization of these positive training instances does not vary much from them, thus reducing the need for later reprocessing. When inconsistencies do occur, each split disjunct will then be quite close to already included positive training instances.

4. The whole G_0 set, which excludes all the negative training instances, must be retained. When a new version space is created, its G set can then be easily formed by choosing hypotheses in G_0 that are more general than the positive training instance being considered. If the G_0 set is not retained, a G set for the new version space must be generated by considering all past negative training instances one after another, and the G_0 set is generated again in order to form the G set. Retaining the G_0 set is thus worthwhile since it avoids redundant computation.

Whether VSSM is worth using depends on the given training instances. If many training instances are to be processed and some of them will be incrementally obtained, the VSSM learning algorithm is then a good candidate. If, however, training instances can be completely collected at the beginning of learning, it might then be more beneficial to use one of the batch learning algorithms.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their very constructive comments. This research was supported by the National Science Council of the Republic of China under Contract No. NSC89-2213-E-214-003.

REFERENCES

[1] T.P. Hong and S.S. Tseng, "Learning Concepts in Parallel Based Upon the Strategy of Version Space," *IEEE Trans. Knowledge and Data Eng.*, vol. 6, no. 6, pp. 857–867, Nov. 1994.
[2] T.P. Hong and S.S. Tseng, "Learning Disjunctive Concepts by the Version Space Strategy," *Proc. Nat'l Science Council (Part A)*, vol. 19, no. 6, pp. 564–573, Republic of China, 1995.
[3] T.P. Hong and S.S. Tseng, "A Generalized Version Space Learning Algorithm for Noisy and Uncertain Data," *IEEE Trans. Knowledge and Data Eng.*, pp. 336-340, vol. 9, no. 2, Mar-Apr. 1997.
[4] M. Manago and J. Blythe, "Learning Disjunctive Concepts," *Lecture Notes in Artificial Intelligence*, vol. 347, pp. 211–230. Berlin, Heidelberg, Germany: Springer-Verlag, 1989.
[5] T.M. Mitchell, "Version Spaces: An Approach to Concept Learning," PhD thesis, Stanford Univ., 1978.
[6] T.M. Mitchell, "Generalization As Search," *Artificial Intelligence*, vol. 18, pp. 203–226, 1982.
[7] K. Murray, "Multiple Convergence: An Approach to Disjunctive Concept Acquisition," *Proc. Int'l Joint Conf. Artificial Intelligence*, pp. 297–300, 1987.