Version Space Learning with DNA Molecules

Hee-Woong Lim[1], Ji-Eun Yun[1], Hae-Man Jang[2], Young-Gyu Chai[2], Suk-In Yoo[1], and Byoung-Tak Zhang[1]

[1]Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University, Seoul 151-742, Korea

[2]Department of Biochemistry and Molecular Biology
Han-Yang University, Ansan, Kyongki-do 425-791, Korea

{hwlim,jeyun,hmzhang,ygchai,siyoo,btzhang}@bi.snu.ac.kr

Abstract. Version space is used in inductive concept learning to represent the hypothesis space where the goal concept is expressed as a conjunction of attribute values. The size of the version space increases exponentially with the number of attributes. We present an efficient method for representing the version space with DNA molecules and demonstrate its effectiveness by experimental results. Primitive operations to maintain a version space are derived and their DNA implementations are described. We also propose a novel method for robust decision-making that exploits the huge number of DNA molecules representing the version space.

1. Introduction

Learning can be formulated as a search for a hypothesis in the space of possible hypotheses that is consistent with the training examples. Version space learning was proposed as a method for representing the hypothesis space [7]. It maintains the general boundary and specific boundary to represent the consistent hypotheses consisting of conjunctions of attribute values. However, Haussler [4] shows that the size of the boundaries can increase exponentially in some cases. Hirsh [6] shows that if the consistency problem, i.e. whether there exists a hypothesis in the hypothesis space that is consistent with the example, is tractable, the boundaries are not needed and new examples can be classified only by positive examples and negative examples.

In this paper, we present a DNA computing method that implements the version space learning without maintaining boundary sets. We exploit the huge number of DNA molecules to maintain and search the version space. To use the massive parallelism of DNA molecules efficiently, the encoding scheme is important. We present an efficient and reliable method to express the hypothesis in the version space. In this encoding scheme, the number of necessary sequences increases linearly, not exponentially, with the number of attributes. We verify the reliability of this encoding scheme by bio-lab experiments. We also show that the version space learning can be reduced to two primitive set operations of intersection and difference. Experimental

methods are described for these primitive operations as well as for predicting the classification of new examples.

Our work is related with Sakakibara [9] and Hagiya [3] in the sense that they try to learn a concept of predefined form from training examples and adopt the general framework of generate-all-solution and search, like in Adleman [1]. Sakakibara suggested a method to express k-term DNF with DNA molecules, to evaluate it, and to learn a consistent k-term DNF with the given examples. Hagiya *et al*. developed the methods to evaluate μ-formula and to learn consistent μ-formula with whiplash PCR.

This paper is organized as follows. In Section 2, it is shown that the process of maintaining a version can be formulated as a set operation and that the version space learning can be performed by two primitive set operations. Section 3 describes the method for version space learning with DNA molecules. In Section 4, the experimental results of generating initial version spaces are presented. Finally, the conclusion and future work are given in Section 5.

2. Version Space Learning as a Set Operation

An excellent description of version space learning can be found in [7]. Here we describe the basic terminology for our purposes. Attributes are features that are used to describe an object or concept. A hypothesis h is a set of restrictions on the attributes. Instance x is a set of attribute values that describe it and all attributes have its values. In this paper, we assume that an attribute takes binary values.

For the purpose if illustrating, let us consider a concept: "An office that has recycling bin" in [2]. If all offices can be described by attributes, *departmentt* (cs or ee), *status* (faculty or staff), *floor* (four or five). "An office on the fourth floor belonging to an faculty" can be expressed as {*status*=faculty, *floor*=four}, or {faculty, four} in abbreviated form.

We define the version space *VS* as a set of hypotheses consistent with training examples. In this paper, this does not include the concept of general and specific boundaries. VS_e denotes a version space for a single positive example e, i.e. a set of all hypotheses that classify an example e as positive. By above definition, an instance x is classified as positive by hypothesis h if and only if $h \subset x$. Therefore, a power set of instance x is equivalent to a set of hypotheses that classify the instance x as positive. In the case of above example, an instance {cs, faculty, four} is classified as positive by a hypothesis {faculty, four}, and is classified as negative by a hypothesis {faculty, five}.

Version space learning can be viewed as a process that refines the version space by removing inconsistent hypotheses and searches as the training examples are observed (Fig. 1). Thus, this process can be reduced to the following set operations. If a positive example is given, we select all hypotheses that classify the example as positive in version space. In contrast, if a negative example is given, we eliminate all hypotheses that classify the example as positive in the version space. Therefore, the version space learning can be performed by intersection and difference operations.

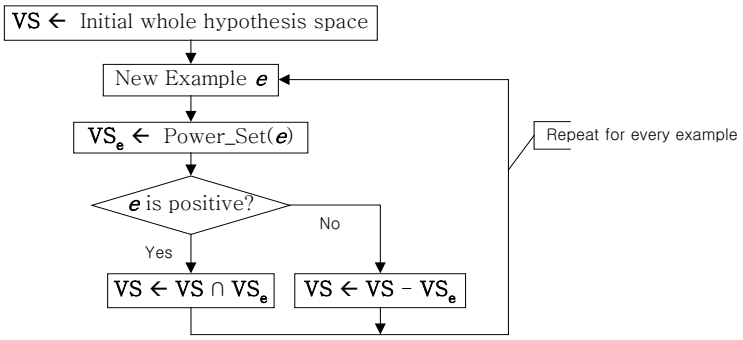

Fig. 1. Procedure for maintaining a version space

3. DNA Implementation

3.1 Encoding

Basically, a hypothesis is represented as a single strand, and the strand is composed by ligation of the sequences that correspond to the attribute values. When generating the initial whole hypothesis space, double strands with sticky ends are used for conjunctions of attribute values. The difference of the sticky ends between the attributes determines the order of attribute values in the hypothesis, and in consequence, a hypothesis can takes no more than one attribute value of the same kind of attribute. In addition to the attribute value strand, another double strand with a sticky end is used equivalently to encode the "don't-care symbol" to make intersection and difference operations easy.

The process to create the initial version space is as follows. At first, we put all the double strands with sticky ends, which correspond to each attribute values, into a test tube and let them hybridize each other. Next, ligation is performed to make the conjunctions of attribute values. Finally, after extracting single strands that we want, the initial version space is completed. This extraction of single strands can be done by affinity separation with magnetic beads.

By this process, we can generate all the possible hypotheses and the length of each hypothesis is all the same. The advantage of this encoding method is that the number of sequences that is needed is

the number of attribute values + the number of attributes

and it increases linearly, not exponentially, with the number of attributes. And by concatenating another common strand to the first attribute and the last attribute for primer, we can easily amplify the version space with PCR.

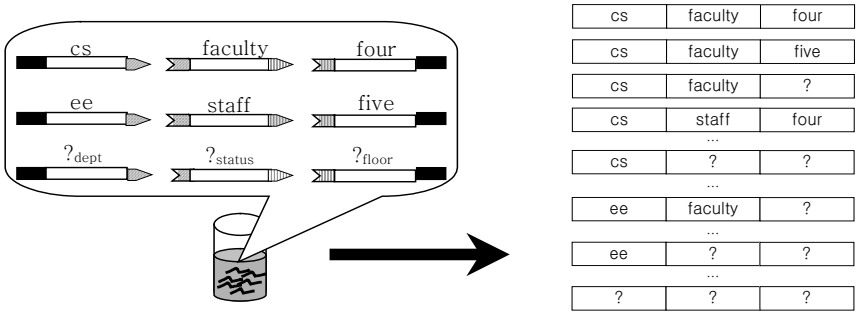

Fig. 2. All attribute strands are put into a test tube, and they are hybridized and ligated. This generates all possible hypothesis strands.

In addition to the above encoding scheme, we can also use the Adleman [1] style encoding scheme. Assume that an attribute value corresponds to a city in the Hamiltonian path problem, and the conjunction of two attribute values corresponds to the road that connects two cities. Then we can make an order of attribute values and restrict the ligation of the attribute values by the existence of an edge strand. The graphs below show two possible directed assembly graphs. Graph (a) encodes "don't care symbol" with the absence of attribute value. So, the length of the hypothesis strands is variable. Graph (b) encodes "don't-care symbol" with another DNA strand like an attribute value. In this case, the length of hypothesis strand is fixed.

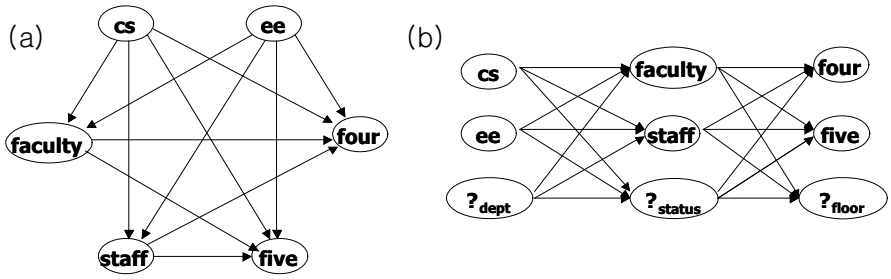

Fig. 3. Two possible directed assembly graphs to encode the hypotheses.

However, the number of the necessary strands is the number of nodes plus the number of edges in the graph, and it increases exponentially with the number of attribute values. So, this encoding scheme is infeasible.
In this paper, we use the first encoding method, considering the number of needed strands and the implementation of intersection and difference operation.

3.2 Primitive Operations

Basically, an affinity separation with magnetic beads is used to perform the set intersection and difference that are the primitive operations to maintain the version space. We designed an experiment that does not need the power set of the examples

explicitly, but uses only the individual attribute values separately. To do this, magnetic beads with DNA single strands are needed and the number of the beads is the same as the number of attribute values including "don't-care symbol". For example, in our example problem, there are nine kinds of bead. "$?_a$" denotes the "don't-care symbol" for attribute *a*.

3.2.1 Intersection

In case of a positive example, we should select the hypotheses that is composed of the only attribute values the example has and "don't-care symbol". Therefore, affinity separation with beads that have each of the attribute values or "don't-care symbol" must be performed by the order of each attribute. For example, if the input example is <cs, faculty, four> and it is positive, the hypotheses that must be selected are as follows:

<cs, faculty, four>, <?, faculty, four>, <cs, ?, four>, <cs, faculty, ?>,
<cs, ?, ?>, <?, faculty, ?>, <?, ?, four>, <?, ?, ?>.

To obtain theses hypotheses, at first, an affinity separation with the beads "cs" and "$?_{department}$" is performed simultaneously, and we select all the DNA strands that are hybridized with magnetic beads. And then with the beads "faculty" and "$?_{status}$", and, finally, with the beads "four" and "$?_{floor}$".

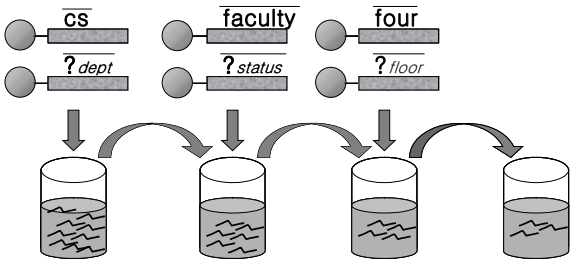

Fig. 4. Intersection operation for a positive example {cs, faculty, four}. DNA strands that are hybridized with beads are selected step by step.

When the number of attributes increases, the experimental process may be complicated because the affinity separation must be performed as many times as the number of attributes. However, the experimental step increases only linearly with the number of attributes.

3.2.2 Difference

Because a difference operation is equivalent to the selection of the hypotheses that are not elements of the intersection set, we can consider the hypotheses that are not selected by the intersection operation, as the result of difference operation. However, because of the reversibility of the chemical processes involved in our experiments, there can be remaining molecules that are not selected even though they are elements of the intersection set. So, choosing the rest of the intersection as difference can be a

primary factor of error. Therefore, difference operation must be performed through affinity separation, e.g. using the beads that are not used in intersection. In difference operation, the beads that are different kinds of attribute can be used simultaneously.

For example, if the new example is <cs, faculty, four> and it is negative, we must select the hypotheses whose *department* is "ee", *status* is "staff", or *floor* is "five". To get such hypotheses, the beads "ee", "staff" and "five" can be used to perform affinity separation. And all DNA strands that are hybridized with one or more of the three beads are selected.

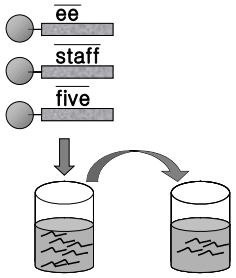

Fig. 5. Difference operation for a negative example {cs, faculty, four}. The beads are used simultaneously and the DNA strands that are hybridized with beads are selected.

3.3 Classification

So far, the method to maintain the version space for observed examples is presented. The question now to ask is: How to use this version space? How to classify new examples? There are various ways to classify new examples using current version space [7]. On the one hand, the version space classifies the new example as positive if it has at least one hypothesis that classifies the example as positive, and classifies it as negative when there are none. On the other hand, version space can use a majority-voting method. In this case, the new example is classified as the majority of hypotheses in the version space decide. In this paper the latter method is implemented as follows.

First, we divide the solution into two parts with an equal volume. And then in one tube, we perform intersection operation with the new example, and in the other tube, we perform difference operation. Now, assuming that the mass of each hypothesis strand is fixed and the distribution is uniform, we can classify the example to the part that has the more DNA molecules. That is to say, if the tube of intersection has more intensity, then we classify the new example as positive, otherwise as negative. To compare the intensity of hypothesis molecules, we can use gel-electrophoresis or fluorescence. If one tube has more DNA molecules, the band in gel-electrophoresis will be thicker, or the intensity of fluorescence will be stronger.

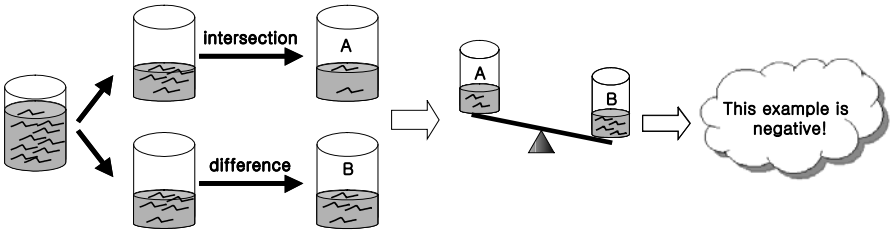

Fig. 6. Classification process. The intersection and difference are performed respectively. The remaining molecules are compared by the fluorescence intensity of DNA molecules. Then, the majority part is selected.

4. Experimental Results on Hypothesis Space Generation

In this section, we show the experimental results of generating the initial version space. In this experiment, we used the example shown in Section 2. We needed to design nine DNA sequences for nine attribute values, and two DNA sequences for two sticky ends. We used the sequence generator NACST [10] to design the DNA sequences for experiment. When generating, we restricted the Tm and GC contents so that all sequences have a similar probability to hybridize and the distribution of hypothesis has uniform distribution. Similarity and H-measure are considered to prevent incorrect hybridization. The designed sequences are given in Table 1.

#	Sequence	Tm (°C)	GC%
1	CTCCGTCGAATTAGCTCTAA	57.17	45
2	AGTCAGTTGGTGACCGCAGA	61.13	55
3	GCATATCAGGCGAGTAGGTG	61.85	55
4	ACAAGGGCTCAGAACCAATG	60.07	50
5	CAGTACTCGGTTTCCGCTAA	59.1	50
6	CGTATGCGCATCCGTTTCAT	62.07	50
7	TTCTTGTGTA**CAACCGCGGC**	62.44	55
8	ATCATGTAGGAACTGTCGCA	59.43	45
9	ACTCCGTATCGGGTAGCTTT	60	50
10	GGAGTTGACACTATCGTCGT	58.82	50
11	**ATAGCCTCGA**GGGACGAATA	61.23	50

Table 1. Sequences that are designed by the sequence generator [4].

We used the right half of sequence no. 7, and the left half of sequence no. 11 as sticky ends, and used the rests as attribute values, because the similarity between the no. 7 and the no. 11 was low. All 18 sequences that were used in the experiment are shown in Table 2.

Primer	Sequence	Tm (°C)
cs	5'- CTCCG TCGAA TTAGC TCTAA ATAGC CTCGA -3'	65.2
	3'- GAGGC AGCTT AATCG AGATT -5'	48.9
ee	5'- ACAAG GGCTC AGAAC CAATG ATAGC CTCGA -3'	69.3
	3'- TGTTC CCGAG TCTTG GTTAC -5'	53.7
?$_{dept}$	5'- ATCAT GTAGG AACTG TCGCA ATAGC CTCGA -3'	66.9
	3'- TAGTA CATCC TTGAC AGCGT -5'	50.0
faculty	5'- AGTCA GTTGG TGACC GCAGA CAACC GCGGC -3'	77.1
	3'- TATCG GAGCT TCAGT CAACC ACTGG CGTCT - 5'	70.3
staff	5'- CAGTA CTCGG TTTCC GCTAA CAACC GCGGC -3'	73.7
	3'- TATCG GAGCT GTCAT GAGCC AAAGG CGATT - 5'	66.9
?$_{stat}$	5'- ACTCC GTATC GGGTA GCTTT CAACC GCGGC -3'	74.0
	3'- TATCG GAGCT TGAGG CATAG CCCAT CGAAA - 5'	66.9
four	5'- GCATA TCAGG CGAGT AGGTG -3'	52
	3'- GTTGG CGCCG CGTAT AGTCC GCTCA TCCAC -5'	76.5
five	5'- CGTAT GCGCA TCCGT TTCAT -3'	58
	3'- GTTGG CGCCG GCATA CGCGT AGGCA AAGTA -5'	79
?$_{floor}$	5'- GGAGT TGACA CTATC GTCGT -3'	48.5
	3'- GTTGG CGCCG CCTCA ACTGT GATAG CAGCA -5'	75.2

Table 2. DNA strands that are used in the experiment.

Each sequence was synthesized as a single strand by Bioneer Corporation (Tae-Jeon, Korea) and has been purified by PAGE and 5' phosphorylated.

a. Hybridization of Molecules

First, all 18 single strands which were adjusted to 100 pmol were mixed with 10$\mu\ell$. Initial denaturation was performed at 95 °C for 5 minutes and then it was cooled down to 16 °C by the rate of 1 °C per 1 cycle in iCycler thermal cycler (Bio-rad, USA).

b. Ligation of Molecules

Ligation was performed with T4 DNA ligase at 16 °C overnight. The reaction buffer was 50 mM Tris-HCl (pH 7.8), 10 mM MgCl2, 5 mM DTT, 1 mM ATP, and 2.5 μg/mℓ BSA.

c. Native Gel-electrophoresis (Confirmation)

Ligation mixture was defined by 3 % native gel-electrophoresis. The running buffer consists of 40 mM Tris-acetate, 1 mM EDTA, and pH 8.0 (TAE). Gel was run on Bio-rad Model Power PAC 3000 electrophoresis unit at 60 W (6 V/cm), and constant power.

d. Separation with magnetic probe

Affinity separation was performed with magnetic beads “ee”, “staff”, and “five” sequentially. (Selecting the hypothesis <ee, staff, four>)

e. Confirmation by PCR & Gel-electrophoresis

We divided the selected DNA solutions into four test tubes and amplified by PCR respectively. First, with primers “ee” and “five”, Second, with primers “$?_{dept}$” and “four”, Third, “$?_{dept}$” and “$?_{floor}$”, and the Last, no primers. And the PCR products are defined by 3 % native gel-electrophoresis.

We confirmed the generation of the initial version space by gel-electrophoresis. As a result, the band of double stranded 80 bp DNA (60 bp for three attribute values, 20 bp for two sticky ends) was generated (Fig. 7(a)). And we also tested the bead separation by PCR with different primers and gel-electrophoresis (Fig. 7(b)).

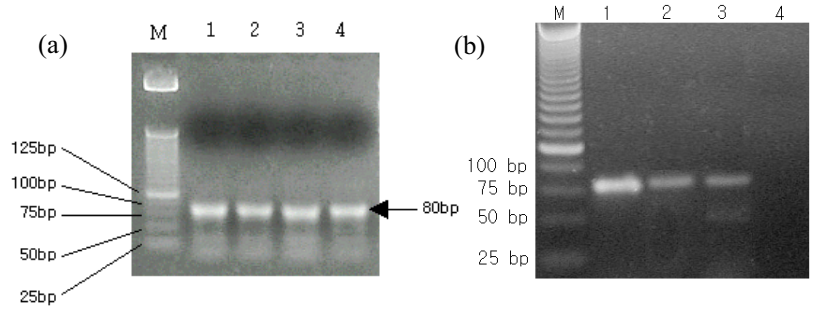

Fig. 7. (a) Result of gel-electrophoresis for the version space generated. Lane 1 and 2: all 18 single strands are mixed simultaneously to generate initial version space. Lane 3 and 4: the 9 double strands for attribute values are mixed after making the attribute value double strands with two single strands. (b) Result of gel-electrophoresis for the PCR products. Lane 1: primers “ee” and “five”. Lane 2 and 3: primers “$?_{dept}$” and “four”, and primers “$?_{dept}$” and “$?_{floor}$”. Lane 4: no primer

5. Conclusion

In this paper, we proposed an experimental method to implement version space learning with DNA molecules. An efficient encoding scheme for representing version spaces is presented, where the number of necessary DNA molecules increases only linearly with the number of attribute values. We confirmed that hypothesis strands of correct length were generated by an experimental result, and tested the magnetic bead separation method for primitive operations in simple experimental process. We also showed that the version space learning is reduced to a set operation on hypothesis sets, and defined two primitive operations for maintaining version spaces. Simple experimental methods to maintain a version space were proposed, along with a method to predict the class of a new example with the current version space by majority voting. The number of experimental steps to perform primitive operations and to predict the new example increases only linearly with the number of attributes and the number of attribute values.

The more experimental verification of the methods for the primitive operations for version space maintenance and for prediction of new examples remains as future

work. The success of experiments depends on the accuracy of affinity separation using magnetic beads. However, this error of experimental process may make version space learning robust to the noisy training examples. Some theoretical and experimental work on this possibility is still necessary. Another point to consider is the case in which an attribute can have more than two values. In this case, it must be specified how the generalization should be performed [5]. Finally, because all experimental processes described in this paper need only affinity separation by magnetic beads, it seems very plausible that this learning process can be automated by the network of microreactor as described in [8].

Acknowledgement

This research was supported in part by the Ministry of Education under the BK21-IT program, by the Ministry of Commerce through the Molecular Evolutionary Computing (MEC) project, and by the project, RIACT 04212000-0008.

References

1. L. M. Adleman, Computing with DNA, *Scientific American*, pages 34-41, August, 1998
2. T. Dean, J. Allen, and Y. Aloimonos, *Artificial Intelligence*, Addison-Wesley, 1995
3. M. Hagiya, M. Arita, D. Kiga, K. Sakamoto, S. Yokoyama, Towards parallel evaluation and learning of Boolean μ-formulas with molecules, *DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 48, pages 57-72, 1999
4. D. Haussler, Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artificial Intelligence*, Vol. 36, pages 177-221, 1988
5. H. Hirsh, Generalizing version spaces, *Machine Learning*, Vol. 17(1), pages 5-45, Kluwer Academic Publishers, 1994
6. H. Hirsh, N. Mishra, L. Pitt, Version spaces without boundary sets, *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI97)*, AAAI Press/MIT Press 1997
7. T. M. Mitchell, *Machine Learning*, 1997, McGraw-Hill
8. D. van Noort, F.-U. Gast, J. S. McCaskill, DNA computing in microreactors, In *Proceedings of 7th International Meeting On DNA Based Computers*, pages 128-137, 2001
9. Y. Sakakibara, Solving computational learning problems of Boolean formulae on DNA computers, *DNA Computing*, Springer-Verlag, Heidelberg, pages 220-230, 2000
10. S.-Y. Shin, D.-M. Kim, I.-H. Lee, and B.-T. Zhang, Evolutionary sequence generation for reliable DNA computing, In *Congress on Evolutionary Computation 2002*, 2002. (accepted)