DNA 분자를 이용한 Version Space 학습

임희웅[0] 장해만[2] 채영규[2] 유석인[0] 장병탁[1]
서울대학교 인공지능연구실[0]
서울대학교 바이오지능연구실[1]
한양대학교 생화학.분자생물학과[2]
{hwlim, hmjang, ygchai, siyoo, btzhang}@bi.snu.ac.kr

Version Space Learning with DNA Molecules

Hee-Woong Lim[0] Hae-Man Jang[2] Young-Gyu Chai[2] Suk-In Yoo[0] Byoung-Tak Zhang[1]
Artificial Intelligence Lab. Seoul National University
Bio Intelligence Lab. Seoul NAtional University
Dept. of Biochemistry and Molecular Biology, Han-Yang University

요 약

Version space는 목표 개념이 속성 값에 대한 제한조건의 연언(conjunction)으로 표현될 수 있는 귀납적 개념학습에서 가설공간을 표현하기위해 사용된다. Version space의 크기는 속성 값의 수에 대해 지수적으로 증가하는데, 우리는 DNA 분자를 이용하여 version space를 표현하는 효율적인 방법을 제시한다. 또한 version space를 유지하기위한 기본 연산과, 이를 DNA 분자를 이용하는 구현 방법이 제시된다. 또한 DNA 분자로 표현된 version space를 활용하여 새로운 example에 대한 분류를 예측하는 방법을 제시한다.

1. 서 론

개념학습은 가설공간에서 학습 데이터와 모순 되지 않는 가설을 탐색하는 과정으로 생각할 수 있다. Version space학습은 이러한 학습데이터와 모순 되지 않는 가설들을 표현하고 유지하는 방법으로 제안되었다 [1]. 그것은 가설의 형태가 속성 값에 대한 제한 조건의 연언으로 표현되는 가설공간에서 학습데이터와 모순 되지 않는 가설들을 표현하기위해 general boundary와 specific boundary를 사용한다. 그러나 Haussler [2]는 어떤 경우, 그 boundary의 크기가 지수적으로 증가할 수 있음을 보였다.

본 논문에서 우리는 이러한 boundary set을 사용하지 않고 version space를 표현하고 유지, 갱신하는 방법을 제시하는데, 이를 위해 무수히 많은 DNA 분자와 이들 간의 초 병렬적인 화학반응을 이용한다. DNA 분자에서의 초 병렬성을 사용하기위해서는 인코딩 방법이 매우 중요하다. 따라서 우리는 version space를 표현하기 위한 효율적이고, 신뢰성 있는 방법을 제시하는데, 이 방법에서 필요로 하는 DNA 시퀀스의 수는 속성 값의 수에 대해 선형적으로 증가하며, 우리는 bio-lab 실험 방법을 통해 인코딩 방법을 검증한다. 또한 version space를 통해 학습하는 과정이 집합 연산인 교집합과 차집합으로 이루어 질 수 있음을 보이고 이러한 연산을 구현하는 실험적인 방법을 제시할 뿐 아니라, 이렇게 유지된 version space를 사용하여 새로운 예에 대한 분류를 예측하는 방법 또한 제시한다.

2. 집합 연산으로서의 Version Space 학습

Version space 학습에 대한 자세한 설명은 [1]에 잘 나와 있다. 먼저 몇 가지 기본적인 용어에 대한 설명을 하면, 속성이란 object나 개념을 기술하기위한 특성을 말한다. 그리고 가설 h는 개념을 구성하기위한 속성 값들에 대한 제약조건의 집합이다. Instance x는 그것을 기술하는 속성 값들의 집합이며, 모든 속성들이 각자의 값을 가지고 있다. 본 논문에서는 각 속성들이 두 가지 값 가운데 하나를 가질 수 있다고 가정한다.

예를 들어 [3]에서 예로든 "An office that has recycling bin"를 생각해보자. 만일 모든 office들이 세 가지 속성, *department* (cs or ee), *status* (faculty or staff), *floor* (four or five) 에 의해서 완전히 기술될 수 있다고 가정해보자. 이 때, "An office on the fourth floor belonging to a faculty"는 {*status*=faculty, *floor*=four}로 표현 될 수 있으며, 간략하게 {faculty, four}로 나타낼 수 있다.

우리는 version space VS를 학습 데이터와 모순 되지 않는 가설들의 집합으로 정의 하며, general boundary와 specific boundary의 개념은 포함되지 않는다. VS는 positive example e와 모순 되지 않는, 다시 말해서 e를 positive로 분류하는 가설들의 집합을 나타낸다. 위와 같은 정의에 따르면, 어떤 instance x가 가설 h에 의해서 positive로 분류되기 위한 필요충분조건은 $h \subset x$ 이다. 따라서 instance x를 positive로 분류하는 가설들의 집합은 x의 멱집합과 동등하다. 앞에서 언급한 예의 경우, {cs, faculty, four}는 가설 {faculty, four}에 의해 positive로 분류되고, 가설 {faculty, five}에 의해서는 negative로 분류된다.

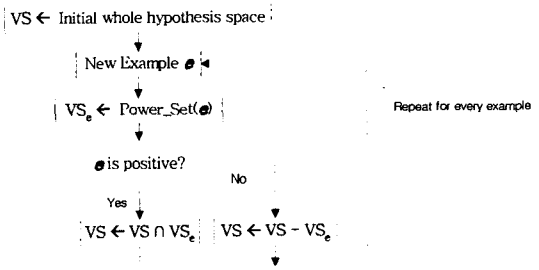

그림 1. Version space의 유지, 갱신 과정

Version space 학습은 학습데이터가 들어옴에 따라 그것과 모순

되는 가설들을 제거함으로써 가설공간을 갱신해나가는 과정으로 생각할 수 있다(그림. 1). 따라서 이 과정은 다음과 같은 집합 연산 과정으로 해석될 수 있다. 만일 positive example이 주어지면 현재 version space에 있는 가설 가운데 그 example을 negative로 분류하는 가설들을 삭제하고, 반대로 negative example이 주어지면, 그것을 positive example로 분류하는 가설들을 제거해야 한다. 그러므로 version space 학습은 교집합과 차집합 연산으로 수행될 수 있다.

3. DNA 분자를 이용한 구현

3.1 인코딩

기본적으로, 한 개의 가설은 각 속성에 해당하는 DNA 서열이 연결된 형태의 한 개의 DNA 단일 가닥으로 표현된다. 초기의 전체 가설공간을 생성할 때, 속성 값들 사이의 연언을 위해 sticky end를 가진 DNA 이중가닥을 사용한다. 각 속성마다 서로 다른 sticky end를 사용하여 가설을 이루는 속성들 사이에 순서를 가지도록 하였으며, 각 속성마다 오직 한 개의 값만을 가지도록 했다. 또한, 교집합과 차집합 연산의 수행을 용이하게 하기위해 각 속성 값에 더하여 "don't care symbol"을 인코딩 하기 위해 또 다른 DNA 이중 가닥을 추가로 사용하였다.

초기의 version space를 생성하기위한 과정은 다음과 같다. 먼저 속성 값들을 인코딩 하고 있는 sticky end를 가진 이중 가닥들을 시험관 안에 섞은 뒤 hybridization 및 ligation을 시킨다. 그 후, 우리가 원하는 단일 가닥을 추출해내면 초기 version space가 완성되는데, 이 추출과정은 magnetic bead를 이용한 affinity separation을 이용하여 수행될 수 있다.

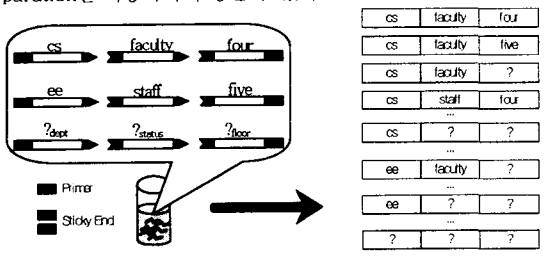

그림 2. 초기 version space의 생성

이 과정에 의해 모두 동일한 길이를 가진 모든 가능한 가설들을 생성할 수 있다. 이 인코딩 방법의 장점은 필요한 DNA 가닥의 종류가

속성 값의 수 + 속성의 수

이며, 속성의 수에 따라 선형적으로 증가하는데 있다. 그리고 각 가설의 양 끝에 동일한 primer를 붙이면 PCR 방법을 통해 version space를 쉽게 증폭할 수 있다.

3.2 기본연산

교집합

Positive example의 경우 우리는 그 example을 구성하는 속성 값과 "don't care symbol"로만 이루어진 가설들을 골라내야 한다. 따라서 그 속성 값들과 "don't care symbol"을 가진 bead들을 사용하여 각 속성에 대해 affinity separation을 순차적으로 수행해야 한다. 예를 들어 <cs, faculty, four>가 positive example로 주어진다면 선택 되어야 하는 가설들은 다음과 같다.

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

이 가설들을 얻기 위해, 먼저 "cs" bead와 "?$_{department}$" bead를 동시에 사용하여 affinity separation을 수행한 다음, "faculty"와 "?$_{status}$"를, 그리고 마지막으로 "four"와 "?$_{floor}$"를 사용하여 분리작업을 수행한다.

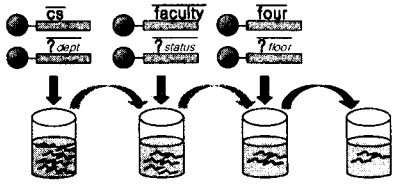

그림 3. Positive example {cs,faculty,four}에 대한 교집합 연산

차집합

차집합 연산은 앞의 교집합 연산에서 선택되지 않고 남아있는 가설들을 선택하는 것과 동등하게 생각할 수 있지만, 화학 반응의 가역적인 특성으로 인해 교집합 연산 시 선택되어야 함에도 불구하고 남아있는 가설이 있을 수 있으므로 차집합의 경우에도 마찬가지로 positive selection을 수행해야한다. 따라서 차집합을 수행할 때는 교집합에서 사용하지 않는 bead들을 사용해야 하며, 서로 다른 종류의 속성에 해당하는 bead들이 모두 동시에 사용될 수 있다.

예를 들어, <cs, faculty, four>가 negative example로 주어진다면, *department* 속성이 "ee"이거나, *status* 속성이 "staff"이거나, 혹은 *floor* 속성이 "five"인 가설을 골라야 한다. 그러한 가설들을 얻기 위해 "ee", "staff" 그리고 "five"에 해당하는 bead를 동시에 사용하여 affinity separation을 수행하면 된다.

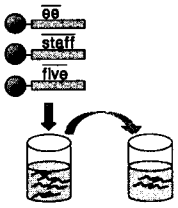

그림 4. Negative example {cs, faculty, four}에 대한 차집합 연산

3.3 분류 및 예측

지금까지 version space를 생성하고 학습 과정에 따라 유지, 갱신하는 방법을 알아보았다. 그런데 만일 학습이 완전히 이루어지지 않아 version space내에 여러 개의 가설이 남아있을 경우 이를 어떻게 활용하여 답을 모르는 새로운 example의 분류 결과를 예측할 것인가? 이를 위한 방법은 여러 가지로 생각해볼 수 있다[1]. 만일 현재 version space에 새로운 example을 positive로 분류하는 가설이 적어도 한 개 존재할 경우 그 example을 positive로 분류하는 방법을 생각 할 수 있고, 그 반대의 경우 또한 생각할 수 있다. 본 논문에서는 다수결의 방법을 사용하는데, 이는 현재 version space에 존재하는 가설 가운데 새로운 example을 positive로 분류하는 것과 negative로 분류하는 가설들의 수를 비교하여 더 많은 쪽으로 분류하는 것이다.

이 방법을 구현하는 실험 방법은 다음과 같다. 먼저 현재 version space가 담겨있는 용액을 같은 부피를 가진 두 부분으로 나눈다. 그리고 나서, 한 쪽 시험관에서는 새로운 example에 대해 교집합 연산을 수행하고, 다른 한 쪽에서는 차집합 연산을 수행한다. 이 때, 가설 생성 시 각 가설들의 분포가 균등하고, 기본

연산과정의 수율이 일정하다고 가정하면, 우리는 그 새로운 example에 대해 더 많은 DNA 분자를 가진 쪽으로 분류한다. 다시 말해서, 만일 교집합 연산의 결과가 더 많은 DNA 분자를 가진다면 그 새로운 example을 positive로 분류하고, 차집합 쪽이 더 많다면 negative로 분류하는 것이다. 이러한 비교는 각 가설 분자에 대해 fluorescence를 붙여 각 결과에서의 형광의 세기를 비교하여 이루어 질 수 있다.

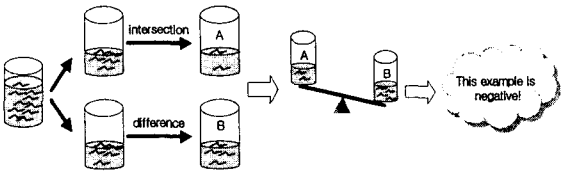

그림 5.분류, 예측 과정

4. 실험 결과

섹션 2에 제시된 예제를 실험하기위해, 속성값을 인코딩하는 9개의 DNA 서열과, sticky end를 위한 2개의 DNA 서열이 필요하다. 실험을 위한 DNA 서열의 디자인을 위해 서열 생성기인 NACST/Seq [4]를 사용하였는데, 이 때 잘못된 hybridization을 방지하기위해 서열 간의 유사도와 H-measure를 고려하였으며, 또한 가설의 분포를 균일하게 맞추기 위해 Tm과 GC함량을 비슷한 수준으로 맞추었다. 각각의 속성을 20 bp의 DNA로 인코딩 하고 sticky end로 10 bp의 DNA를 사용했다. 따라서 각각의 가설은 3개의 속성으로 이루어져 있으므로 sticky end부분을 포함하여 총 80 bp의 길이를 가지게 된다. 초기 version space의 생성 결과가 아래 그림 6의 왼쪽 그림에 제시되어 있으며, 원하는 길이인 80 bp의 가설들이 생성되었음을 확인 할 수 있다.

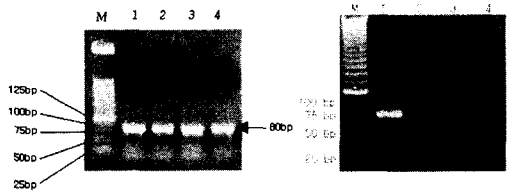

그림 6. 전기 영동 결과

그리고 version space 학습의 기본연산에 필요한 magnetic bead를 이용한 *affinity separation*을 테스트 한 결과가 그림 6의 오른편에 제시되어 있다. 초기 version space로부터 "ee", "staff", 그리고 "five"의 bead를 순차적으로 사용하여 분리 작업을 수행하여 <ee, staff, five>라는 가설을 골라낸 다음 서로 다른 primer set을 사용하여 PCR을 수행하여 실제로 <ee, staff, five>가 선택되었는가를 확인하였다. 정상적인 primer set인 "ee", "five"를 사용한 1번 레인의 경우 가장 많이 증폭되었고, 잘못된 primer set인 "$?_{dept}$" ,"four"(2번 레인), 와 "$?_{dept}$" , "$?_{floor}$"(3번 레인)을 사용한 경우 증폭이 제대로 이루어지지 않았음을 볼 수 있다.

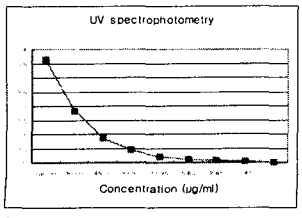

그림 7. UV spectrophotometry

그리고 마지막으로 DNA 분자수를 비교하여 다수결의 방법을 수행할 수 있는지를 확인하기 위해 DNA 농도에 따른 형광 강도의 변화를 측정하였다(그림 7).

5. 결론

지금까지, DNA 분자를 이용하여 *version space learning*을 구현하기 위한 실험 방법들을 알아보았다. Version space를 표현하기 위해 필요한 DNA 분자의 수가 속성의 수에 대해 선형적으로 증가하는 효율적인 인코딩 방법을 제안하였으며, 또한 version space 학습이 가설 공간에서의 집합 연산을 통해 이루어 질 수 있음을 보이고, 이를 위해 필요한 기본 연산을 정의 하였다. 그리고 이러한 연산을 수행하기 위한 실험 방법을 제시하고, 현재의 version space를 활용하여 답을 모르는 새로운 example이 주어졌을 때 그 분류 결과를 예측하는 방법 또한 제시하였다. 간단한 실험을 통해 초기 version space의 생성을 확인 하였고, 기본 연산의 수행에 필요한 magnetic bead를 이용한 affinity separation에 대한 테스트를 수행하였다. 또한 실험을 통해 다수결의 방법의 수행의 가능성을 확인하였다.

그러나 version space의 유지, 갱신을 위한 기본연산과, 분류 예측을 위한 다수결의 방법에 대한 실험적인 검증이 더 수행되어야 한다. 그리고 주어진 학습 데이터들을 이용한 전체적인 학습 과정에 대한 검증이 이루어 져야 할 것이다. 화학 반응의 특성상 실험 과정의 오차가 존재 할 수 있지만, 이러한 오차를 활용하면 noise를 포함한 학습데이터에 대해 robust한 성질을 가질 수 있도록 할 수도 있다. 따라서 사용되는 실험과정들에 대한 정량적인 해석과 모델링이 필요하다. 그리고 본 논문에서 제시된 기본 연산은 오직 magnetic bead를 이용한 affinity separation만을 사용하기 때문에, [5]에 제시된 microreactor의 network을 이용한 자동화도 가능할 것으로 생각 된다.

Acknowledgement

본 연구는 교육인적자원부의 BK21-IT 프로그램, 산업자원부의 Molecular Evolutionary Computing(MEC) 프로젝트, 그리고 RIACT 04212000-0008 프로젝트에 의해 지원되었습니다. 그리고 서울대학교의 컴퓨터 신기술 공동 연구소가 연구에 필요한 기자재를 제공하였습니다.

참고문헌

1. T. M. Mitchell, *Machine Learning*, 1997, McGraw-Hill
2. D. Haussler, Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artificial Intelligence*, Vol. 36, pages 177-221, 1988
3. T. Dean, J. Allen, and Y. Aloimonos, *Artificial Intelligence*, Addison-Wesley, 1995
4. S.-Y. Shin, D.-M. Kim, I.-H. Lee, and B.-T. Zhang, Evolutionary sequence generation for reliable DNA computing, *Proc. of Congress on Evolutionary Computation 2002*, pages 79-84, 2002.
5. 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