제2회 컴퓨테이셔널 인텔리젼스 합동학술대회

분자 컴퓨팅을 이용한 패턴 완성

Molecular Computational Methods for Pattern Completion

김주경*, 장병탁

서울대학교 컴퓨터공학부 {jkkim*, btzhang}@bi.snu.ac.kr

요약

본 논문은 대표적인 분자 컴퓨팅 방법 중 하나인 DNA Computing을 이용하여 기계학습을 가능하게 하는 확률라이브러리모델(Probabilistic Library Model)을 소개하고, 이를 이용하여 기존 컴퓨터로는 처리속도 및 용량의 한계 때문에 처리하기 쉽지 않은 기계학습 문제 중 하나인 훼손된 패턴을 원래의 패턴에 최대한 가깝게 복원하는 패턴 완성(pattern completion)을 수행할 수 있는 새로운 방법을 제시한다.

1. 서론

패턴 완성이란 원본 패턴들을 입력 받아 학습한 후 일부분 훼손된 패턴이 주어졌을 때 원래의 패턴에 가깝게 복원하는 것이다 [4]. 기존의 패턴 완성 방법 중 대표적인 것으로 neural network 기반인 Hopfield network [3]이나 그것의 binary, stochastic 버전인 Boltzmann machine [2]이 있다. 이들 neural network를 사용한 방법의 경우 node 수의 약 13%이하만큼의 원본 패턴을 학습할 경우에만 통계적으로 안정적으로 작동함이 밝혀져 있다 [1,3,5]. 이들은 fully connected recurrent network 구조이기 때문에 network상에서 필요한 edge 수는 $O(|node|^2)$가 되므로 node수가 많이 커지는 경우 기존의 컴퓨터로는 처리하기 힘들어 진다.

패턴 완성뿐 아니라 기계학습관련 많은 문제들은 기존의 방법과 컴퓨터로는 계산속도 및 용량의 한계로 인해 규모가 어느 정도 커지면 해결하기 힘들어진다. 그러므로 기존 컴퓨터의 제약을 벗어나고자 여러 가지 기술이 제시되고 있다. 본 논문에서는 분자컴퓨팅 방법 중 protein 등보다 안정적이고 제어가 용이한 DNA 분자들의 대용량 병렬처리 기능을 사용하여 컴퓨팅을 가능하게 하는 DNA Computing으로 기계학습의 여러 가지 문제들에 접근 할 수 있는 확률라이브러리모델(Probabilistic Library Model, PLM) [6,7]을 소개하고 패턴 완성에 적용해 본다.

실험에서는 0부터 9까지의 클래스를 가지는 UCI machine learning deposit의 Optical Recognition of Handwritten Digits[1] 데이터에 대하여 클래스 별 패턴 개수를 동일하게 하기 위해 각 클래스 별 376개 패턴을 선정하고 64차원 벡터로 변환한 총 3760개의 패턴을 이용하여 패턴 완성을 수행해 보인다.

그림1은 8×8 행렬로 나타낸 학습할 패턴들의 각 클래스 별 평균 이미지, 그림2는 모든 library 원소들을 합친 평균 이미지이다.

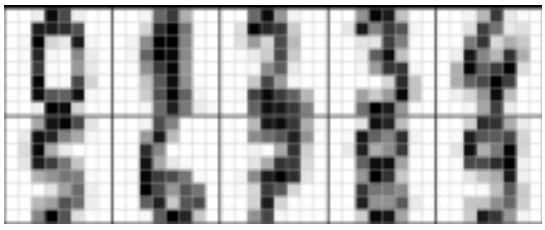

그림 1: 학습할 패턴들의 클래스 별 평균 이미지

[1] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/optdigits/

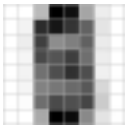

그림2: library 원소들의 평균 이미지

여기에서 진하게 표시되는 성분일수록 해당 클래스에서 빈번하게 true로 나타난 것이다.

2. PLM을 이용한 패턴 완성

2.1 PLM 소개

PLM은 기본적으로 특정 order의 term들로 이뤄지는 수많은 clause들의 집합인 library를 바탕으로 구성된다. 이 때 library는 Weighted Disjunctive Normal Form (wDNF) [6]로 나타낼 수 있다. 예를 들어 term의 order가 3인 경우 library는 $(x_1 \wedge \neg x_2 \wedge x_4) \vee (x_2 \wedge x_4 \wedge \neg x_5) \vee (x_3 \wedge x_3 \wedge \neg x_5) \vee \dots (x_3 \wedge x_5 \wedge \neg x_6)$ 식으로 표현될 수 있다. 이 때 clause들이나 한 clause내의 term들은 중복이 허용한다. 입력공간(input space)의 차원이 I라고 할 때 x_i $(1 \le i \le I)$는 입력패턴의 각 성분을 나타낸다. wDNF식의 term에서 x_i는 i번째 성분이 true, $\neg x_i$는 false임을 나타낸다.

2.2 패턴 완성 알고리즘

PLM을 이용한 패턴 완성의 대략적인 알고리즘은 다음과 같다.

- 1. random으로 또는 training set의 원소들을 subsampling한 것으로 초기 library를 구성
- 2. training set의 원소들을 library의 원소들과 비교하여 일정 hamming distance 이내에서 일치하는 경우 library상에서 해당 element의 개수를 일정 수 또는 비율만큼 증가시킴
- 3. library의 원소들을 분석해서 입력공간에서의 각 성분 별로 true인 것의 합, false인 것의 합을 구함. 순서대로 At_i, Af_i라 한다
- 4. 일정 hamming distance이내에서 library의 원소에 포함된 입력패턴의 성분들의 값이 입력패턴에서의 해당 성분들의 값과 일치하면 남기고 불일치하면 library에서 삭제
- 5. 3번째 과정처럼 library들을 분석해서 입력공간에서의 각 성분 별로 true인 것의 합을 Bt_i, false인 것의 합을 Bf_i라고 하여 계산
- 6. 각 성분마다 $y_i = \frac{Bt_i}{Bf_i} - \frac{At_i}{Af_i} + \alpha Bt_i$ 값을 구한다. 여기서 α는 실수 상수. 이때 i번째 성분의 결과값은 특정 threshold 값 t를 설정해서 y_i가 t이상이면 해당 성분은 true, 미만이면 false인 것으로 구하거나 y_i값의 크기를 바탕으로 그 성분이 true가 될 확률 값으로 결과를 구할 수 있다.

이 패턴 완성 알고리즘에서 가장 중요한 부분은 4번째 과정에서 완벽하게 일치 하지 않아도 원소들이 일정 hamming distance내에서 일치하는 경우 삭제하지 않고 남기는 것이다. 만약 완벽하게 일치하는 원소들만 남겨놓는다면 false로 주어진 입력의 성분 값을 true로 만드는 결과를 얻을 수 없다. 즉 어떤 성분의 값이 원래 true였는데 false로 된 경우 복원할 수 없다. 이를 해결하기 위해 약간의 오차를 허용하는 것이 필요하다. 예를 들어 library원소들의 order가 5고 해당 원소에 주어진 성분들의 값과 입력패턴에서의 해당 성분들의 값을 비교 시 4개의 성분은 같고 한 성분이 입력패턴에서는 false였는데 library 원소에는 true로 되어있다고 가정하자. 이 경우 성분 값이 4개 일치하기 때문에 이 원소는 입력패턴의 원본에서 subsampling된 것일 가능성이 높다. 이 경우 해당 성분은 원래 true인데 입력패턴에서 훼손되어서 false가 되어 있었을 가능성이 높기 때문에 이 성분을 true로 만드는 것이 원래 패턴으로 복원되게 하는 것일 가능성이

높다고 할 수 있다. 이 과정을 모든 library 원소에 대해서 수행한 후 결과적으로 남아 있는 library 원소들의 성분 값들을 평균적으로 나타내어 원래 패턴에 가깝도록 복원하는 것이다.

하지만 이 방법만으로 수행하는 경우 입력 패턴의 영향을 어느 정도 받긴 하지만 학습이 끝난 시점의 library 원소들의 평균 성분 값에 가까운 결과가 나오게 될 가능성이 높다. 예를 들어 그림2에서 진하게 표시되는 성분은 입력에 상관없이 결과도 true가 될 가능성이 높다. 그 이유는 library원소에 주어진 성분들의 값과 입력패턴에서 해당하는 성분이 4개는 같고 하나만 다를 때 위의 예제처럼 그 다른 값이 복원 되야 하는 값이 될 수도 있지만 해당 library원소가 우리가 복원하고자 하는 패턴과 클래스가 다르거나 모양이 많이 다른 패턴에서 subsampling된 것이었다고 해도 우연히 4개의 성분 값이 입력패턴과 일치할 수 있기 때문이다. 이 경우엔 성분 값이 다른 하나의 성분을 복원시 의도되지 않은 패턴이 결과로 나올 수 있다. 그리고 그 성분이 학습이 끝난 library 원소들의 분포를 봤을 때 빈번하게 true로 나타난 성분이었다면 위의 예처럼 4개의 성분값이 같고 하나가 다를 때 그 다른 하나에 해당할 가능성이 높고 우리가 원하는 방향이 아닌 단순히 전체 패턴에서의 빈도가 높기 때문에 true로 설정되는 잘못된 방향으로의 복원을 하게 될 가능성이 높다고 할 수 있다.

이러한 현상을 막기 위해 6번째 과정이 필요하다. 즉 $\frac{Bt_i}{Bf_i} - \frac{At_i}{Af_i}$ 를 통해 학습이 끝난 시점에서의 true, false의 비율과 완성이 끝난 시점에서의 true, false 비율을 비교해서 학습 끝난 시점보다 true의 비율이 더 높은 경우에만 그 성분을 true로 처리하게 한다. 여기에 αBt_i가 필요한 이유는 단순히 비율만 가지고 계산시 library내에서 true인 경우가 매우 적은 성분의 경우 비율계산만 고려 시 의도와 다르게 결과가 true값으로 될 수 있기 때문이다. 그러므로 true인 경우의 절대 빈도수도 어느 정도 감안을 해야 올바른 패턴 완성을 할 수 있다.

2.3 DNA Computing을 이용한 구현

위의 알고리즘을 현재까지 개발된 DNA Computing 기법으로 구현하는 경우에는 다음과 같은 과정을 거친다.

- 1. 입력공간의 각 성분들을 DNA로 인코딩한 후 PCR을 통해 증폭시킴. 그 후 원하는 길이에 맞게 ligation을 통해 library 원소들을 생성
- 2. training set의 원소 하나를 역시 DNA로 인코딩하고 이를 PCR로 증폭하여 여러 개로 만듬. 그 후 이들을 chopping한 후 library 원소들과 반응되게 하고 일정오차 이내에서 hybridization된 strand에 대해 PCR을 수행하여 해당 원소의 수가 증가되게 하고 완전히 hybridization되지 않은 strand의 수를 줄임 (경우에 따라 따로 감소처리 하지 않아도 무방). 이를 모든 training set의 원소들에 대해 수행
- 3. 학습이 된 library의 원소들의 성분검사
- 4. 입력패턴을 DNA로 인코딩하고 이를 PCR로 증폭하여 여러 개로 만든다. 그리고 이들을 chopping한 후 library 원소들과 반응하게 하고 일정 오차 이내에서 hybridization 된 strand들만 남김.
- 5. 3번처럼 성분검사
- 6. 3,5번으로 구해진 값들을 이용해 y_i계산

3. 시뮬레이션

0~9의 클래스를 가지는 64차원 벡터로 된 패턴 3760개를 학습하고 해당 패턴들을 일부 훼손 시킨 것을 입력으로 받아서 패턴 완성을 수행한다. training set의 3760개 원소들에 대해 각각 총 150번 order를 5로 random subsampling한 원소들을 만들고 동일한 원소가 10000개가 되도록 복제하여 초기 library를 구성한다. Library의 원소와 training set의 원소가 일치하는 경우 그 library원소의 개수가 500개 증가되도록 하였고 알고리즘상의 α값은 9, threshold 값은 $4\times10^{-11}\times \arg\max_{i\in I}(Bt_i)$로 설정하였다. 몇 가지 0부터 9까지의 패턴을 일부 훼손시킨 후 패턴 완성을 수행해 본 결과는 다음과 같다.

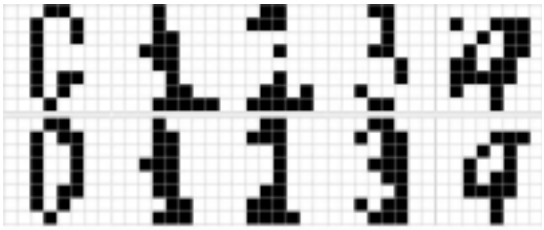

그림 3: 0부터 4까지의 패턴 완성 결과

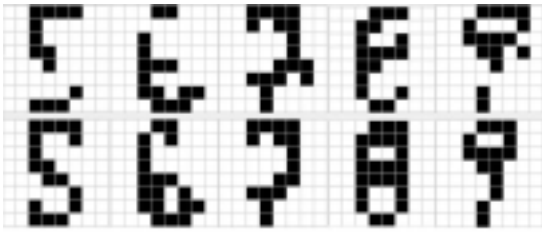

그림 4: 5부터 9까지의 패턴 완성 결과

그림 3은 0부터 4, 그림 4는 5부터 9까지의 임의의 숫자패턴을 일부 훼손 시킨 후 복원한 결과이다. 각 이미지에서 위쪽은 훼손된 패턴, 아래쪽은 복원한 패턴을 나타낸다. 입력으로 받은 훼손된 패턴의 모양이 반영되어있으면서 각각 정상적인 숫자 패턴 모습으로 복원되는 걸 볼 수 있다.

4. 결론

본 논문에서는 PLM을 기반으로 패턴 완성을 수행하는 방법과 실제 몇몇 패턴에 대한 시뮬레이션 결과를 제시하였다.

입력공간이 수십 차원 정도인 경우에는 기존의 컴퓨터를 가지고 시뮬레이션 하더라도 충분히 의미가 있는 결과를 볼 수 있었다. 하지만 입력공간이 수백, 수천 차원 이상으로 커지게 된다면 library의 크기가 상당히 커져야 정상적인 패턴 완성을 수행할 수 있으므로 그런 경우 효과적인 대규모 병렬처리를 가능하게 하는 DNA Computing 기술로 구현하면 기존의 컴퓨터 성능으로는 구현하기 힘든 대규모의 문제도 해결할 수 있을 것이다. 아직 DNA Computing 기술은 toy problem을 푸는 수준에 있지만 현재 꾸준히 실험기술, 알고리즘 등이 연구, 실험되고 있으므로 장차 패턴 완성을 비롯한 기존의 컴퓨터로 해결하기 어려운 문제를 풀려고 할 때 대안 중 하나로 사용될 수 있을 것이다.

감사의 글

이 논문은 교육부 BK21 사업 및 산업자원부 차세대 신기술 과제에 의해 지원되었음.

참고문헌

[1] Amit, D.J., *Modeling Brain Function: The World of Attractor Neural Networks*, New York: Cambridge University Press, 1989.

[2] Hinton, G.E., Sejnowski, T. J., "Learning and Relearning in Boltzmann Machines," In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, Cambridge: MIT Press, 1986, pp. 282-317.

[3] Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences, USA*, vol.79, 1982, pp. 2554-2558.

[4] MacKay, David J.C., *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[5] Műller, B., and J.Reinhardt, *Neural Networks: An Introduction*, New York: Springer-Verlag, 1990.

[6] Zhang, B.-T. and Jang, H.-Y., "Molecular learning of wDNF formulae," *Preliminary Proceedings of the Eleventh International Meeting on DNA Computing (DNA 11)*, 2005, pp. 185-195.

[7] Zhang, B.-T. and Jang, H.-Y., "A Bayesian algorithm for in vitro molecular evolution of pattern classifiers," *Lecture Notes in Computer Science*, 3384, 2005, pp. 458-467.