헬름홀츠 머신 기반의 탐색점 분포 학습에 의한 최적화

신수용, 장병탁
서울대학교 컴퓨터공학부
{syshin, btzhang}@scai.snu.ac.kr

Optimization by Helmholtz Machine-Based Learning of the Distribution of Search Points Using Helmholtz Machine

Soo-Yong Shin and Byoung-Tak Zhang
School of Computer Science and Engineering
Seoul National University

요 약

많은 최적화 문제에서 해답들의 구조는 서로 의존성을 가지고 있다. 이러한 경우 기존의 진화연산이 사용하는 빌딩 블록 개념으로는 문제를 해결하는데 많은 어려움을 겪게 된다. 이를 극복하기 위해서 헬름홀츠 머신(Helmholtz machine)을 이용해서 데이터의 분포를 예측한 후 최적화를 수행하는 방법을 제안한다. 기존의 진화 연산을 바탕으로 하지만 교차연산이나 돌연변이 연산을 사용하는 대신에, 헬름홀츠 머신을 이용해서 데이터의 분포를 파악하고, 이를 이용해서 새로운 데이터를 생성하는 과정을 통해 최적화 과정을 수행한다. 진화연산으로 해결하는 데 곤란을 겪고 있는 여러 함수들을 해결하여 이를 검증하였다.

1. 서 론

대부분의 최적화 문제들은 해답의 각 요소들이 서로 의존성을 가지고 있고, 그 의존성은 기존의 유전자 알고리즘에서 사용하는 빌딩 블록 개념으로는 표현할 수 없는 것들이 많다. 왜냐하면 빌딩 블록은 자신과 자신 주변의 관계밖에 표시하지 못하는 한계를 가지고 있고, 또한 교차 연산에 의해서 쉽게 파괴되어 버리기 때문이다. 따라서 이러한 단점을 극복하기 위해서 여러 가지 방법들이 제안되고 있다.

크게 보면 대략 2가지 접근방법으로 분류할 수 있다. 첫 번째 방법은 문제의 표현방법을 변화시켜 해답의 각 요소들간의 의존성(linkage information)을 표현할 수 있도록 하거나, 기존의 연산자들을 변형시켜 의존성을 이용할 수 있도록 하는 방법을 취하고 있다[1].

두 번째 방법은 주어진 데이터를 가지고 문제를 표현하는 모델을 구성하고, 그 모델을 이용해 원하는 해답을 찾는 방법이다. 데이터를 학습해 주어진 문제를 표현할 수 있는 모델을 형성하고, 그 모델로 새로운 데이터를 생성한 후, 생성된 데이터로 모델을 수정해 나가는 방법이다. 지금까지 많은 연구들이 있어 왔고, 모델을 구성하는 방법에서 조금씩 차이를 보이고 있다. 주변 분포(marginal distribution)를 계산하는 방법[2], 체인을 이용해서 분포를 표시하는 것[3], 트리 형태를 이용하는 연구[4], 주어진 문제를 작은 부분문제로 분해하는 방법[5], 베이지안 네트워크를 사용하는 것[6] 등이 대표적이다.

본 논문에서는 두 번째 방법을 참고로 하여, 생성 모델(generative model) 중 하나인 헬름홀츠 머신(Helmholtz machine)을 이용해서 분포를 파악한 후 최적화 과정을 수행하는 알고리즘을 개발하였다.

다음절에서 헬름홀츠 머신에 대해서 설명하고, 3절에서 구체적인 알고리즘을 소개한 후, 4절에서 여러 가지 실험 및 결과를 보이고, 5절에서 결론을 내리고자 한다.

2. 헬름홀츠 머신

헬름홀츠 머신은 대표적인 생성 모델 중 하나로, 비감독 학습(unsupervised learning)을 하며 Hinton 등이 요인(factor) 분석을 위해서 개발한 모델이다[7]. 구체적인 헬름홀츠 머신의 형태는 그림 1에 설명되어 있다. 입력을 받는 입력 노드(visible variable)와 은닉 노드(latent variable)가 있다. 주어진 데이터의 분포를 학습하는 데 사용되는 인식 연결(recognition connection)은 점선으로 표시되어 있으며, 새로운 패턴을 생성하는 데 사용되는 생성 연결(generative connection)은 실선으로 표시되어 있다.

일반 신경망 모델과는 다르게 가중치가 생성 가중치(generative weights)와 인식 가중치(recognition weights), 2 종류가 존재한다. 따라서 학습 방법도 일반적인 역전파(backpropagation) 방법으로는 안되고 수면-기상(wake-sleep) 알고리즘을 사용해야 한다[8]. 수면-기상(wake-sleep) 알고리즘은 2단계 알고리즘으로 EM 알고리즘(Expectation-Maximization)과 유사하게 가중치를 학습한다. 기상(wake) 단계에서는 인식 가중치를 고정한 후에 생성 가중치만을

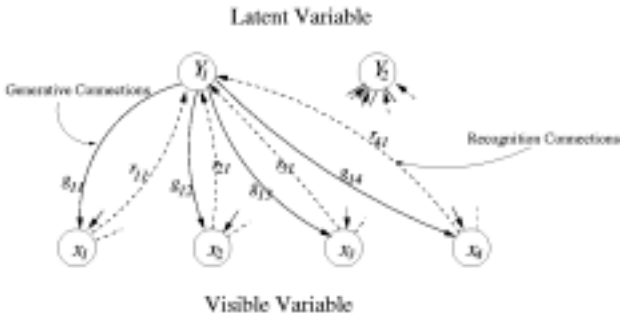

그림 1. 헬름홀츠 머신

학습하고, 수면(sleep) 단계에서는 반대로 생성 가중치를 고정한 후 인식 가중치를 학습한다.

Hinton 등의 실험 결과를 보면[7, 8], 숫자 인식 후 새로운 숫자 패턴 생성 또는 숨겨진 특정 패턴 등의 분석 등을 성공적으로 하는 것을 알 수 있다. 이와 같은 결과를 볼 때 헬름홀츠 머신은 패턴 분석에 적절한 모델이고, 또한 생성 모델이기 때문에 새로운 패턴을 생성하는 데에도 사용될 수 있다는 것을 알 수 있다.

3. 최적화 알고리즘

알고리즘에 대한 개략적인 설명이 그림 2에 나타나 있다. 우선 초기 데이터를 생성하고, 헬름홀츠 머신을 초기화한다. 그리고 헬름홀츠 머신을 2절에서 설명한 수면-기상 알고리즘을 이용해서 학습을 시킨다. 학습이 종료되면 새로운 데이터 집합을 생성한 후, 다음 세대 학습에 사용될 데이터를 선택한다. 다시 헬름홀츠 머신을 초기화한 후 학습을 반복한다.

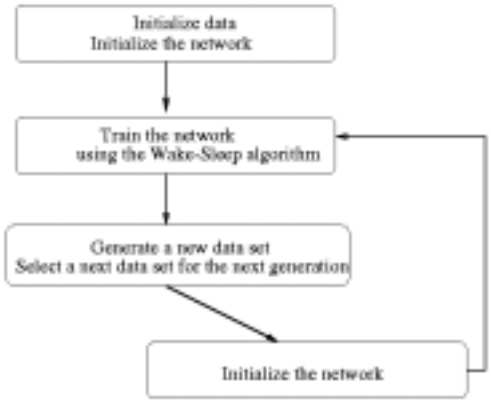

그림 2. 개략적인 알고리즘

초기 데이터는 외부에서 주어질 수도 있고, 헬름홀츠 머신을 초기화한 후 헬름홀츠 머신을 이용해서 생성할 수도 있다. 수면 기상 알고리즘을 학습을 할 때 가중치 변경은 델타 규칙(delta rule)을 사용하였고, 식 (1)~(4)에 설명되어 있다. 식 (1)은 생성 가중치를 학습하는 과정이고, 식 (2)은 인식 가중치를 학습하는 과정이다. 식 (3)은 은닉 노드의 값을 결정하는 식이고, 식 (4)은 새로운 데이터를 생성하는 식이다.

$$\mathbf{G}' = \mathbf{G} + \eta(\mathbf{x}^{(c)} - \mathbf{G}\mathbf{y}^{(c)})\mathbf{y}^{(c)} \tag{1}$$

$$\mathbf{R}' = \mathbf{R} + \eta(\mathbf{y}^{(f)} - \mathbf{G}\mathbf{x}^{(f)})\mathbf{x}^{(f)} \tag{2}$$

$$\mathbf{y} = \mathbf{R}\mathbf{x} \tag{3}$$

$$\mathbf{x} = \mathbf{G}\mathbf{y} \tag{4}$$

여기서, η는 학습 비율(learning rate)이고, $\mathbf{R}$은 인식 가중치 벡터, $\mathbf{G}$는 생성 가중치 벡터이다.

새로운 데이터는 지금까지 학습된 헬름홀츠 머신을 이용해서 생성하였다. 데이터는 기본적으로 식 (4)에 의해서 생성되나 확률적으로 그 값이 결정된다. 생성된 데이터와 학습에 사용된 데이터를 모두 합한 후 (($\mu + \lambda$) 방법) 절단 선택(truncation selection)으로 다음 세대의 학습에 사용될 데이터들을 선택한다.

4. 실험 결과

모두 4가지 서로 다른 문제에 대해서 실험을 해 보았고, 그 결과를 유전자 알고리즘과 비교하였다. 비교 대상이 된 유전자 알고리즘은 기본적인 형태의 유전자 알고리즘으로 일점 교차 연산(one-cut point crossover)과 일점 돌연변이 연산(one point mutation)을 사용하며, 적합도 비례 선택 방법(roulette-wheel selection)으로 선택하였다. 구체적인 유전자 알고리즘의 변수들은 표 1과 같고, 실험 함수 3개에 대해서 모두 동일한 조건으로 하였다. 헬름홀츠 머신의 조건은 표 2에 나타나 있다.

표 1. 유전자 알고리즘 조건

parameters	값
최대 세대 수	10,000,000
개체군 크기	50
교차 연산자 비율	0.5
돌연변이 연산자 비율	0.01
염색체 크기	표 3의 문제 크기 참조

표 2. 헬름홀츠 머신을 이용한 최적화 방법 변수

parameters	값
데이터 집합 크기	50
신경망 학습 반복 횟수	1,000
최대 세대 수	10,000
은닉 노드 수	1, 2, 3, 5
입력 노드 수	염색체 크기와 동일
학습율(learning rate)	0.01
최대 세대 수	100,000

실험을 한 문제는 원맥스 (onemax) 문제, 이차 함수 (quadratic function), 3차 디셉티브 함수 (deceptive function of order 3)이다. 자세한 함수의 정의는 식 (5), (6), (7)에 나타나 있다. 이차 함수와 디셉티브 함수의 경우에는 입력 배열의 변경(permutation)이 가능한데, 식 (8)에 의하여 변경하였다.

$$f_{OneMax}(x) = \sum_{i=0}^{n-1} x_i \tag{5}$$

$$f_{quadratic}(x,\pi) = \sum_{i=0}^{\frac{n}{2}-1} f_2(x_{\pi(2i)}, x_{\pi(2i+1)}) \tag{6}$$

$$f_{3deceptive}(x,\pi)=\sum_{i=0}^{\frac{n}{3}-1} f_3(x_{\pi(3i)},x_{\pi(3i+1)},x_{\pi(3i+2)}) \quad (7)$$

$$\pi_k(i)=\left\lfloor \frac{n(i \bmod k)+i}{k} \right\rfloor \quad (8)$$

각 문제의 각각의 경우에 대해서 모두 10번씩 실험을 하였으며, 전체 개체군 중 10% 정도가 최적해를 발견하면 수렴한 것으로 간주하여 실험을 중단하였다.

각각의 경우에 대한 실험 결과가 표 3에 설명되어 있다.

표 3. 헬름홀츠 머신을 사용한 최적화 방법(HM)과 단순한 유전자 알고리즘(sGA)의 비교

문제	문제 크기	Succ %		세대 수	
		HM	sGA	HM	sGA
One Max	20	100	100	23.2	930.7
	40	100	100	100.3	16,904.7
	60	100	50	196.2	5,691,291.0
	80	100	0	543.6	-
	100	100	0	991.1	-
Quadratic	20	100	100	18.4	12,073.3
	40	100	10	136.3	6,532,588.0
	60	100	0	418.1	-
	80	100	0	782.3	-
	100	100	0	1,265.8	-
3-Deceptive	15	100	100	177.2	308,547.3
	30	100	80	4,409.9	3,470,749.3
	45	100	0	44,192.8	-

표에서 알 수 있는 것처럼, 헬름홀츠 머신을 사용한 최적화 방법(HM)이 단순한 유전자 알고리즘(sGA)보다 훨씬 좋은 결과를 보여주는 것을 알 수 있다. 최적해를 찾는 비율을 볼 때, 헬름홀츠 머신을 이용한 방법은 거의 모든 경우에서 해를 찾고 있으나, 단순 유전자 알고리즘은 문제의 크기가 조금만 커져도 해를 찾지 못하는 것을 알 수 있다. 계산 효율 측면에서 세대수(generation)를 비교해 보더라도 그림 3에서 볼 수 있는 것처럼, 제안된 방법이 훨씬 더 우수하다는 것을 알 수 있다. 비록 여기서, 반복 횟수가 적합도를 계산한 횟수만을 고려한 것이라서 직접적인 비교는 되지 않으나, 헬름홀츠 신경망 학습 시간(학습 반복 횟수)을 고려하더라도 여전히 우수하다는 것을 알 수 있다.

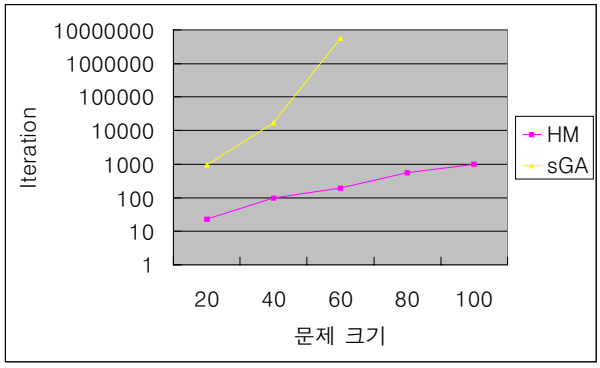

그림 3. One-max 문제에서의 반복 횟수 비교

5. 결 론

본 논문에서는 새로운 최적화 방법의 하나로 헬름홀츠 머신을 이용한 방법을 제안하였다. 기존의 진화 연산처럼 교차 연산자, 돌연변이 연산자를 사용하는 대신에 헬름홀츠 머신을 이용해서 분포를 파악하고 파악된 분포를 통해 학습 데이터를 새로이 생성하는 방법으로 최적화를 수행한다. 실험 결과에서 확인할 수 있는 것처럼 단순한 유전자 알고리즘보다는 최적화를 찾는 비율과 계산 효율 측면에서 훨씬 우수하다는 것을 알 수 있었다.

감사의 글

본 연구는 과학기술부 뇌연구개발사업(BR-2-1-G-06)에 의하여 일부 지원되었음.

참고문헌

[1] A. A. Salman, K. Mehrotra, and C. K, Mohan, "Linkage Crossover For Genetic Algorithms", *Proceedings of the Genetic and Evolutionary Computation Conference (GECOO'99)*, pp. 564 - 571, Morgan kaufmann, 1999.

[2] M. Pelikan and H. Mühlenbein, "The Bivariate Marginal Distribution Algorithm", *Advances in Soft Computing - Engineering and Design and Manufacturing*, pp.521-535, Springer, 1999.

[3] J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding Optima by Estimating Probability Densities", *Advances in Neural Information Processing Systems, Volume 9*, pp.424-432, The MIT Press, 1997.

[4] S. Baluja and S. Davies, "Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space", *Proceedings of the 14th International Conference on Machine Learning*. pp.30-38, Morgan Kaufmann, 1997.

[5] H. Mühlenbein and T. Mahnig, "FDA - A scalable evolutionary algorithm for the optimization of additively decomposed functions", *Evolutionary Computation* **7**(4): 1999.

[6] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Baysian Optimization Algorithm", *Proceedings of the Genetic and Evolutionary Computation Conference (GECOO'99)*, pp. 525 - 532, Morgan kaufmann, 1999.

[7] P. Dayan, G. E. Hinton, R. Neal, and R. S. "Zemel, Helmholtz Machine", *Neural Computation*, **7**: 1022-1037, 1995.

[8] G. E. Hinton, P. Dayan, B. J. Frey, R. M. Neal, "The "Wake-Sleep" Algorithm for Unsupervised Nerual Network", Science **268**: 1158-1161, 1995.