인공 신경망의 계산 복잡도

조 ○성배, 김 진형

한국과학기술원 전산학과 인공지능연구실

Computational Complexity of Artificial Neural Networks

Sung-Bae Cho and Jin H. Kim

Artificial Intelligence Lab., Dept. of Computer Science, KAIST

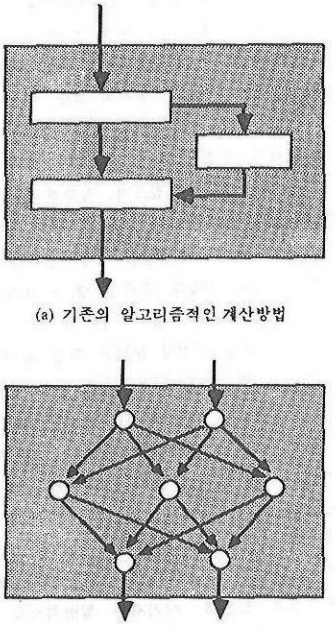
요 약

기존의 계산이 순서적인 절차에 의한 알고리즘에 근거하여 문제를 해결하는 방법이라면, 신경망은 단순한 연산을 수행하는 처리기들의 대규모 상호 연결을 통하여 문제를 해결하고자 하는 새로운 계산 모형이다. 이러한 신경망은 최근 기존 계산 방식의 한계성을 극복할 수 있는 대안의 하나로 여러가지 응용분야에 적용되고 있기는 하지만, 신경망을 현실적인 문제에 이용하기 위해서는 그 자체의 능력과 한계를 연구할 필요가 있다. 본 논문에서는 신경망의 계산 능력을 밝히고, 신경망의 계산 복잡도로서 신경망의 학습 문제가 NP-complete임을 증명하여 신경망의 연결강도를 조정하기 위한 polynomial time 알고리즘이 없음을 보인다.

I. 서 론

인공 지능(Artificial Intelligence)은 지난 30 여년 동안 놀라운 속도로 발전하여 우리가 일상 생활에서도 그 실체를 느낄 수 있을 정도로 산업 전반에 까지 그 응용 분야를 넓히고 있다. 하지만 현재의 인공 지능을 사람의 지능과 비교한다면 아직 많은 차이가 있고, 고작해야 아주 좁은 분야에서만 사람의 지능을 흉내내고 있는 실정이다. 이와 같이 인간은 비교적 쉽게 해내는 물체나 음성의 인식과 같은 문제를 기계가 하기 어려운 것은 그 처리 방식에 있어서 인간과 기계사이에 차이점이 존재하기 때문이다. 이러한 문제를 해결하기 위한 접근 방법으로 인간의 두뇌 구조를 분석하고 그 처리 메카니즘을 규명하여, 그와 같은 구조를 갖는 컴퓨터를 만들고자 하는 연구의 결과로서 신경망(Neural Network)이 새로운 계산 모형으로 등장하였다 [1,2,11,12].

주어진 입력에 대하여 해당하는 출력을 내는 과정을 계산이라고 볼때, 기존의 계산은 절차적인 순서에 의한 알고리즘에 근거하여 문제를 해결하는 방법인 반면에, 신경망 계산 기법은 <그림 1>과 같이 인간의 두뇌 신경 조직을 모델로 하여 단순한 기능을 하는 처리기들의 대규모 상호 연결을 통하여 문제를 해결하고자 하는 것이다. 이러한 신경망은 프로그래머가 문제 해결에 대한 정확한 지식을 갖지 못했더라도 상황에 따라 그 문제에 대한 적절한 알고리즘을 자체의 적응 정보처리 능력으로 만들어 내기 때문에 공학적인 측면에서 매우 가치있는 것임에



(a) 기존의 알고리즘적인 계산방법

(b) 신경망의 계산기법

<그림 1> 신경망 계산 기법과 기존 방법의 비교

입력없다고 하지만 현실세계의 실제문제에 있어서는 신경망이 기존의 알고리즘을 사용하는 방법에 비하여 항상 우수한 성능을 보이지는 않는다. 따라서 신경망 자체의 능력과 그 한계를 연구할 필요가 있다.

본 논문에서는 신경망의 계산능력을 밝히고, 신경망의 계산 복잡도에 관한 연구로서 신경망의 학습문제를 분석해 보고자 한다. 즉, 신경망을 학습시키는 방법이 존재하는가에 대한 computability와 그것이 적당한 시간내에 가능한가에 대한 complexity를 밝히기 위하여 신경망 학습 문제가 NP-complete임을 증명하고, 이를 통하여 신경망의 연결강도(connection weight)를 조정하기 위한 polynomial time의 알고리즘이 존재하지 않음을 보이고자 한다. II장에서는 신경망 자체의 이론적인 배경을 기술하고, III장에서는 신경망의 계산 능력을 밝힌후 학습 알고리즘이 NP-complete임을 증명한다. 그리고, 마지막으로 IV장에서는 결론을 기술한다.

II. 신경망 이론의 개요

2.1 기본 구조

신경망은 기본적으로 노드(node)와 이들간의 연결(connection)로 구성된 그래프로서, 노드는 처리기(PE)라 하며, 연결은 연결강도(connection weight)를 갖는다. 임의의 처리기 PE_j는 <그림 2.1>과 같이 다른 처리기 PE_i의 출력 y_j를 그 입력 x_j로 받아 PE_j와 PE_i의 연결강도 w_{ij}에 대한 각 입력의 가중치 합 net_j를 다음과 같이 구하고,

$$net_j = \Sigma w_{ij}x_i$$

이 합을 정해진 전이함수(transfer function)에 적용시켜서 얻은 그 결과를 PE_j의 출력 y_j로 내보낸다.

$$y_j = f(net_j + \theta_j)$$

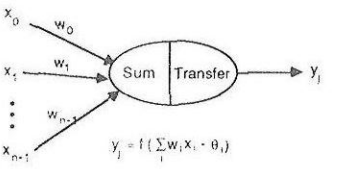
여기에서 θ_j는 PE_j의 고유 offset이다. 전이함수는 <그림 2.2>에서와 같이 주로 3가지 종류의 함수, 즉 hard limit nonlinearity, threshold logic, sigmoid nonlinearity funtion이 쓰인다.

전체의 구조는 이와 같은 단순한 PE들이 <그림 2.3>과 같이 상호 연결되어 있으며, 각각의 PE는 신경망 전체의 입출력 면에서 볼때 입력 PE, 출력 PE, 내부 PE등의 3가지로 나뉘어 진다. 이때, 신경망을 학습시킨다는 것은 신경망이 주어진 입력에 대하여 적당한 출력을 낼 수 있도록 각 노드 사이의 연결강도를 조정하는 것을 의미한다.

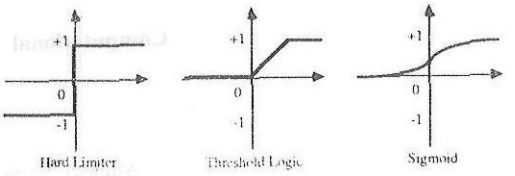
일반적으로 신경망 모델은 학습 능력, 분산 지식 표현, 병렬 처리 능력, 결함 극복 능력 등의 공통적인 특성을 갖는데, 이러한 특성으로 인하여 기존의 기호처리 중심의 인공지능 기법들이 잘 해결하지 못하는 분야에서 좋은 성능을 발휘하고 있다. 신경망의 구분은 신경망의 연결 형태, 즉 망의 구조와 학습 알고리즘의 차이에 따라서 여러가지로 나눌 수 있는데, 대표적인 것으로는 Hopfield Network, Boltzmann Machine, 다층 Perceptron등이 있다. 각 모델들은 나름대로의 장단점과 적합한 응용분야가 있는데, 여기서는 일반적으로 많이 사용되는 다층 Perceptron에 대해서만 설명하기로 한다.

2.2 다층 Perceptron

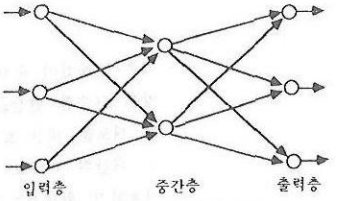
다층 구조 Perceptron은 입력층과 출력층 사이에 하나 이상



<그림 2.1> 처리기의 기능



<그림 2.2> 전이 함수의 종류



<그림 2.3> 신경망의 구조

의 중간층을 갖는 feed-forward 망이며[2], 추가된 중간층은 입력과 출력에 의하여 동시에 직접 연결되지 않은 중간 노드들을 갖는다. 비선형 다층 구조의 신경망 모델에 대한 학습 알고리즘을 만들기 위해 활성화 함수에 미분가능한 성질을 가미한 것이 sigmoid(semilinear threshold) unit이고, 이러한 unit을 가지고 있을 때의 학습 알고리즘이 Rumelhart 등이 제안한 Error Backpropagation 알고리즘이다[2,11]. Backpropagation 알고리즘은 지도 학습(supervised learning) 알고리즘으로서 이상적인 출력과 실제 출력사이의 평균 제곱 오차에 해당하는 비용함수의 값을 최소화하기 위하여 gradient search 기법을 사용한다. 이때, 임의의 작은 값으로 인접층 노드간의 연결강도를 초기화한 후 훈련 데이타들을 입력층에 반복적으로 제공함으로써 훈련을 시킨다. 입력 데이타에 의하여 계산된 출력층에서의 값과 기대하고 있던 값의 차이, 즉 에러값을 하위층으로 전파시킴으로써 하위층과의 연결강도를 재조정해 나가면서, 신경망 전체의 연결강도가 수렴하여 총 에러가 매우 작아질 때까지 반복적으로 훈련시킨다.

Backpropagation 알고리즘은 단층 구조 Perceptron에서 처럼 수렴한다고 증명되지는 않았지만 XOR 문제, 음성 합성 및 인식, 영상 패턴의 인식 등의 관심 있는 많은 문제 영역에서 비교적 성공적임이 실험적으로 판명되고 있다. 하지만, Backpropagation 알고리즘에서 사용하는 gradient search 방법은 global minimum을 찾는 대신에 local minimum에 빠지는 경우가 종종 있는데, 이러한 문제점을 피하기 위하여 임의의 다른 초기값을 갖는 연결강도 집합을 사용하여 여러번 훈련을 시키거나, 여분의 중간 노드들을 두며, 연결강도를 조절하기 위하여 사용되는 학습률의 값을 낮추는 등의 여러 방법이 제시되고 있다. 또한 학습시 연결강도의 수렴 속도를 개선하기 위한 좀 더 복잡한 적응 알고리즘에 대한 연구가 한창 진행중이다[11].

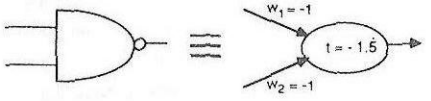
III. 신경망 학습의 계산 복잡도

기존의 신경망에 대한 연구들은 대부분 제안된 신경망 모델이 얼마나 생물학적으로 그럴듯 한가(biologically plausibility)에 촛점을 맞추고 있다. 하지만, 공학적인 면에서 현실적인 문제를 해결하기 위한 계산기법으로 신경망을 사용하기 위해서는 먼저 신경망의 능력에 대한 이론적인 연구가 필요하다. 즉, 신경망이 원하는 기능을 할 수 있도록 학습 시킬 수 있는 방법이 존재하는가 하는 computability와 그러한 학습이 적당한 시간 내에 완료될 수 있는가 하는 complexity를 밝혀야 할 것이다.

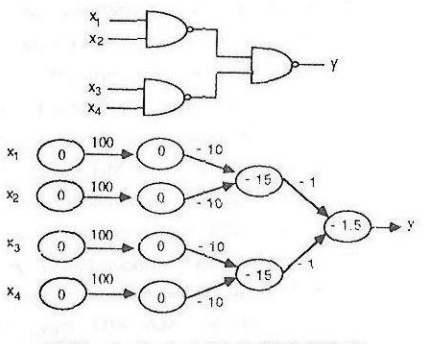
3.1 신경망의 Computability

일반적인 계산기법으로서 신경망의 능력과 그 한계를 결정하기 위해서는 신경망이 풀 수 있는 문제와 더 나아가 효율적으로 해결할 수 있는 문제를 규명해야 할 것이다. 이제, 기존의 모든 계산 가능한 문제는 신경망을 사용해도 풀 수 있음을 보이고자 한다. 우리가 풀고자 하는 문제의 입력이 $x = x_1 \cdots x_m$ 과 같은 이진 스트링으로 주어졌을때, 원하는 출력이 $y = y_1 \cdots y_k$ 라면, 이 문제는 $y_1 = f_1(x), \ldots, y_k = f_k(x)$ 와 같은 k개의 Boolean 함수로 규정 지을 수 있다. 따라서, 신경망의 계산능력을 보이기 위해서는 임의의 Boolean 함수가 적당한 신경망에 의하여 모의 실험(simulation)될 수 있음을 보이면 된다.

그런데, 스위칭 이론(switching theory)에 의하면 NAND 게이트가 Boolean 함수의 complete basis를 이룬다는 것이 증명되어 있으므로[8], 신경망이 NAND 게이트의 기능을 할 수 있음을 보임으로써 이를 증명하기로 한다. NAND 게이트는 모든 입력이 1 일때만 그 출력이 0이 되는 게이트이며, 이것이 complete basis를 이룬다는 것은 모든 임의의 Boolean 함수가 NAND 게이트의 combinational circuit에 의하여 처리될 수 있음을 의미한다. NAND 게이트는 <그림 3.1>에 나타난 것과 같이 연결강도를 $w_1 = w_2 = -1$로, threshold $t = -1.5$로 갖는 threshold 함수로 모의 실험될 수 있다. 그런데, 신경망에서 각 노드의 기능이 결국 가중 입력의 합에 대한 threshold 함수이므로, 임의의 Boolean 함수는 신경망에 의하여 쉽게 모의 실험될 수 있다. 즉, NAND 게이트로 이루어진 combinational 회로에 대하여 각각의 게이트를 그림과 같은 threshold 함수의 기능을 하는 노드로 대치하면 된다. <그림 3.2>에 한 예가 있다. 이와 같이 변환된 신경망의 크기는 입력의 수에 따라서 지수적으로(exponentially) 증가하긴 하지만, 기존의 계산기법에 의해 해결되는 모든 문제는 신경망을 사용해서도 풀 수 있음을 알 수 있다.



<그림 3.1> NAND 게이트의 기능을 하는 신경망의 노드



<그림 3.2> Combinational 회로의 신경망 구현 예

일반적인 컴퓨터로 문제를 풀때 고려하는 complexity는 time complexity와 space complexity, 그리고 알고리즘의 크기를 나타내는 Kolmogorov complexity가 있는데, 신경망에서는 학습을 위해 연결강도를 변경시키기 위한 반복 횟수로 time complexity를 가늠하고, 노드의 수로 space complexity를, 그리고 실제적인 알고리즘을 나타내는 연결강도내의 정보의 양에 의하여 Kolmogorov complexity를 측정한다. 이때, 노드의 수가 N이면 필요한 반복 횟수는 $O(N^2)$이고, 연결강도의 양은 $O(N^3)$ 비트이다. 따라서 신경망을 사용하여 문제를 해결하려고 할 때에는 이와 같은 세가지의 complexity 모두가 최소로 되도록 하여야 한다.

3.2 학습 알고리즘의 NP-completeness

신경망의 가장 큰 장점중의 하나는 프로그래머가 문제에 대한 해결지식을 명확히 갖지 못한 경우에도 단순히 입력 데이타와 그에 해당하는 출력 결과를 신경망에 보여줌으로써 풀고자 하는 문제의 입력과 출력간의 적절한 사상(mapping)을 신경망이 자체적으로 형성한다는 것이다. 신경망의 이와 같은 특성은 문제에 따라 각 노드 사이의 연결강도(connection weight)를 적당히 조정하는 것에 기인하는데, 이것을 신경망의 학습이라고 한다[3,4]. 이 절에서는 신경망 학습 문제의 computability와 complexity를 분석해 보고자 한다.

어떤 알고리즘의 computational complexity를 결정하는 방법으로는, complexity를 알고 있는 기존의 문제로 부터 polynomial time 내에 변환 가능한지를 검사하는 것이 있는데[9,10], 다음의 정리는 이와 같은 방법을 사용하여 신경망을 학습시키는 문제(LEARN)가 NP-complete 임을 보인다[6,7,8].

[정리 1] 임의의 다층구조 신경망에 대하여 일련의 입력/출력 쌍이 주어져 있을때, 각 입력에 대해 올바른 출력을 내도록 하는 신경망의 연결강도가 존재하는가 결정하는 문제는 NP-complete이다.

<증 명>

(1) LEARN ⊆ NP

연결강도를 guess 하고 선택한 연결강도에 의해 신경망이 올바로 동작하는지 결정하는 것은 nondeterministic 알고리즘이 polynomial time 내에 check 할 수 있으므로 신경망의 학습 문제는 NP class 에 속한다.

(2) 이제, 이미 NP-complete로 알려진 CNF satisfiability (SAT) 문제[9,10]로 부터 신경망의 학습(LEARN) 문제로 변환하고자 한다.

SATISFIABILITY

INSTANCE : Boolean 변수의 집합 U 와 이에 의해 구성된 절의 집합 C

QUESTION : C를 위한 적당한 truth assignment가 존재하는가? (즉, C로 구성된 임의의 Boolean 함수를 tautology로 만드는

truth assignment가 존재하는가 ?)

(3) Boolean 변수의 집합을 $U = \{u_1, u_2, \ldots, u_n\}$ 이라 하고, SAT의 한 instance를 구성하는 절의 집합을 $C = \{c_1, c_2, \ldots, c_m\}$ 이라고 하자. 또, 신경망의 연결강도를 $W = \{w_1, w_2, \ldots, w_t\}$ 라 하고, 각 층의 집합을 $L = \{L_1, L_2, \ldots L_k\}$ 라 하자. 이제 C가 satisfiable 하면(if and only if), 신경망도 올바른 출력을 내도록 각 층 L의 연결강도의 집합 W를 결정할 수 있도록 L과 W를 구성하고자 한다.

L과 W를 <그림 3.3>과 같이 C와 U로 부터 입력과 출력을 갖는 다층의 망으로 구성하여 보자. 이때, 각각의 연결강도 w_i를 다음 세가지 경우로 나누어 구성한다.

경우 1) $w_i \subseteq L_1$: $w_i = 1$ if u_i is used
$w_i = 0$ if u_i is not used

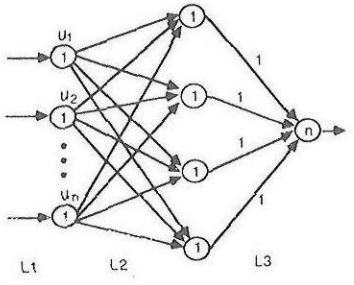
경우 2) $w_i \subseteq L_2$: $w_i = 1$ if u_i is true
$w_i = -1$ if u_i is false
$w_i = 0$ if u_i is not used

경우 3) $w_i \subseteq L_3$: $w_i = 1$

즉, L_1의 연결강도는 SAT의 instance에 따라서 가변적이고, threshold는 <그림 3.3>과 같이 1로 고정되어 있다. 이때, L_1층은 변수가 참인가 거짓인가를 결정하고, L_2층은 L_3층의 필요에 따라 negation을 만들어 낸다. 즉, L_2층의 연결강도 값은 변수 u_i가 참이면 1, 거짓이면 -1, L_3층에서 사용되지 않으면 0으로 한다. 예를 들어, CNF의 처음 항이 $(u_1 \vee u_2 \vee \sim u_3)$ 이면, L_3층의 첫번째 노드는 L_2층의 u_1, u_2 노드와의 연결강도를 각각 +1로 하고, u_3 노드와는 -1로 연결시킨다. 그리고, 그 이외의 노드들과의 연결강도는 0으로 한다. 결국 L_3의 노드들은 각각 OR 기능을 하여 $C_i = \vee_j u_j$의 역할을 하고, 최종의 출력 노드는 AND 기능을 하여 들어온 입력이 모두 1이어야 1을 출력하는 $\wedge_i c_i$의 역할을 한다.

이와 같은 변환이 올바른지 검사하기 위해서는 신경망이 올바른 출력을 내면(if and only if), C가 satisfiable 하다는 것을 증명하여야 한다. 그런데, 신경망이 어떤 입력에 대해 해당하는 출력을 내도록 학습되어 있다면 그러한 기능을 하는 Boolean 함수가 참이 되어 SAT의 해당 instance가 존재하게 되고, 반대로 그러한 SAT의 instance가 존재하지 않으면 신경망도 결코 수렴하지 않는다. 따라서, 이러한 변환은 올바르다.

(4) 이와같은 변환은 단순한 mapping이므로 쉽게 polynomial transformation임을 알 수 있다. 따라서, 이미 NP-complete 임을 알고 있는 CNF satisfiability 문제에서 신경망의 연결강도 조정 문제로 polynomial time reduction이 존재하므로, 신경망의 학습 문제는 NP-complete 이다. ∎



<그림 3.3> Satisfiability 신경망

위의 증명은 단순한 local replacement로 매우 쉽게 되어, 신경망의 polynomial time 학습 알고리즘은 존재하지 않음을 알았다. 따라서, 현존하는 모델은 신경망을 사용하여 풀고자 하는 문제에 약간의 제한을 가하고, 얻을 수 있는 출력의 최적성(optimality)을 약간 완화시켜 학습 알고리즘을 개발하고 있다.

IV. 결 론

신경망을 현실적인 문제에 적용하기 위해서는 먼저 신경망 자체의 계산 능력과 그 한계에 대한 연구가 필수적으로 수행되어야 한다. 따라서, 본 논문에서는 신경망이 기존의 모든 계산 가능한 문제를 해결할 수 있는 능력이 있음을 보이고, 신경망을 학습시키는 문제는 NP-complete라는 사실로 그 한계를 보였다. 결국 신경망의 능력을 최대한으로 잘 사용하기 위해서는 효과적인 학습방법이 필요하며, 이에 대한 연구가 절실한 실정이다.

참고 문헌

[1] Teuvo Kohonen, "An Introduction to Neural Computing," *Neural Networks*, Vol. 1, pp. 3 - 16, 1988.

[2] Lippmann, R. O., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4 - 22, April, 1987.

[3] Hinton, G. E., "Connectionist Learning Procedures," *Carnegie-Mellon University Technical Report CMU-CS-87-115*, 1987.

[4] Hinton, G. E. and Sejnowski T. J., "Neural Network Architectures for AI," *the 6th National Conference on AI*, Tutorial Material, 1987.

[5] Robert Hecht-Nielsen, "Neurocomputer Applications," *NATO ASI Series F: Computer and Systems Sciences*, Vol. 41, pp. 445 - 453, 1988.

[6] John F. Kolen, "Faster Learning Through A Probabilistic Approximation Algorithm," *International Conference on Neural Networks*, Vol. I, pp. 449 - 454, 1987.

[7] Peter Berke, "That Does Not Compute - e.g., Neural Nets," *International Conference on Neural Networks*, Vol. IV, pp. 819 - 826, 1987.

[8] Y. S. Abu-Mostafa, "Neural Networks for Computing ?," *Neural Networks for Computing*, AIP Conference No. 151, pp. 1 - 6, 1986.

[9] S. A. Cook, "The complexity of theorem-proving procedures," *Proceedings of the 3rd ACM Symposium on Theory of Comuting*, 1971, pp. 151 - 158.

[10] M. R. Garey and D. S. Johnson, Computers and Intractibility, *Freeman and Company*, San Francisco, 1979.

[11] Rumelhart, D. E. and McClelland, J. L., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986.

[12] Marvin Minsky and Seymour Papert, *Perceptrons : An Introduction to Computational Geometry*, Expanded Edition, Cambridge, MA: MIT Press, 1988.