論文 96-1-1-22

효율적 고차 신경회로망을 이용한 비선형 함수 근사에 대한 연구

正會員 辛 堯 安*

Nonlinear Function Approximation Using Efficient Higher-order Feedforward Neural Networks

Yoan Shin* Regular Member

본 논문은 1995년도 숭실대학교 교내학술연구비 지원에 의하여 연구되었음

要 約

본 논문에서는 다차원 유클리드 공간의 compact 부분집합에 정의된 임의의 비선형 연속 함수를 근사하기 위해 제안된 새로운 형태의 고차 비궤환 신경회로망 (higher-order feedforward neural network)인 ridge polynomial network (RPN)의 수학적 근사 능력을 검토하고, 실제 비선형 다변수 함수 근사와 패턴 분류 문제에 응용한다. RPN은 *pi-sigma network* (*PSN*)[13]의 일반화된 모델로서, 특수한 형태의 릿지 다항식 (ridge polynomial)에 기반을 두고 있다. 본 논문에서는 이러한 새로운 형태의 릿지 다항식이 임의의 다변수 다항식 (polynomial)을 정확히 표현할 수 있음을 증명하고, 이 결과로부터 임의의 연속 함수에 대한 RPN의 근사 정리를 얻는다. RPN은 Gabor-Kolmogorov 다항식에 기반을 둔 일반적인 고차 신경회로망의 단점인 많은 가변 계수의 수를 급격히 감소시키면서, 그의 장점인 빠른 학습, 우수한 비선형 함수 근사 능력은 보존하는 모델이며, PSN의 기본 학습 알고리즘을 바탕으로 하여 점진적으로 주어진 함수를 근사하여 나간다. 컴퓨터 모의 실험 결과는 제안된 RPN의 구조와 점진적 학습 알고리즘이 빠른 수렴과 우수한 비선형 함수 근사 능력을 가짐을 보인다.

ABSTRACT

In this paper, a higher-order feedforward neural network called *ridge polynomial network* (*RPN*) which shows good approximation capability for nonlinear continuous functions defined on compact subsets in multi-dimensional Euclidean spaces, is presented. This network provides a more efficient and regular structure as compared to ordinary higher-order feedforward networks based on Gabor-Kolmogorov polynomial expansions, while maintaining their fast learning property. The ridge polynomial network is a generalization of the pi-sigma network (PSN)[13] and uses a special form of ridge polynomials. It is shown that any multivariate polynomial can be exactly represented in this form, and thus realized by a RPN. The approximation capability of the RPNs for

* 숭실대학교 공과대학 전자공학과
論文番號 : 95102-0311
接受日字 : 1995年 3月 11日

arbitrary continuous functions is shown by this representation theorem and the classical Weierstrass polynomial approximation theorem. The RPN provides a natural mechanism for incremental function approximation based on learning algorithm of the PSN. Simulation results on several applications such as multivariate function approximation and pattern classification assert nonlinear approximation capability, fast learning, and proper incremental approximation capability of the RPN.

Ⅰ. 서 론

일반적인 신경회로망의 구분으로 뉴론 연결부(weight connection)의 형태에 따라 비궤환형(feedforward)과 궤환형 (feedback)으로 나눌 수 있다. 비궤환 구조는 뉴론간의 연결이 입력층부터 출력층으로 단일 방향 연결을 가지며, 이 구조의 대표적인 모델로는 단계층 퍼셉트론 (single-layered perceptron)[26], Widrow의 Adaline[38], multi-layered perceptron (MLP)[25], radial basis function network[28]등을 들 수 있다. 한편, 궤환 구조는 뉴론간의 연결에서 루프를 가지는 형태이며, 대표적인 모델로는 Hopfield network[15], MLP를 기본으로 한 recurrent network[25], Boltzmann machine[1]등이 있다.

가장 단순한 비궤환 구조 신경회로망 모델인 단계층 퍼셉트론과 Adaline은 선형 논리 유니트 (linear threshold logic unit)를 기본 뉴론 구조로 하며, 각기 매우 빠르고 간단한 학습 알고리즘을 이용한다. 단계층 퍼셉트론의 경우, 퍼셉트론 수렴 정리는 선형 분리가 가능한 (linearly separable) 데이터가 퍼셉트론 학습 알고리즘을 통해 유한한 반복 횟수 안에 완벽히 분리됨을 보장한다[26]. 또한, 자승 오차의 최소화에 기반을 둔 Adaline의 경우 gradient descent 방법인 LMS 알고리즘에 의해 이론적으로 오차의 global minima를 찾을 수 있으며[39], 적절히 학습률을 조절하여 빠른 수렴을 얻을 수 있다. 하지만, 이러한 간단한 신경회로망 구조의 문제점으로는 제한된 근사 능력 (approximation capability)을 들 수 있다. 즉, 이들 신경회로망은 앞서 언급한 대로 선형 분리가 가능한 문제에 대하여만 효과적으로 사용될 수 있으며, 일반적으로 우리가 접하는 비선형 문제에 대해서 한계를 갖고 있다.

현재 여러 응용 분야에 가장 널리 사용되는 비궤환 구조 신경회로망인 MLP는, 이러한 단계층 신경회로망의 제한된 근사 능력을 향상하기 위해 중간층 혹은 은닉층을 사용한다. 한 개의 중간층을 가지는 MLP는 충분히 많은 중간층 내 뉴론이 있다고 가정되면, 일반적 형태의 sigmoid 함수를 뉴론 특성 함수로 사용하여 우리가 관심 있는 대부분의 비선형 함수를 충분히 작은 오차 내로 근사할 수 있음이 증명되었고[16], gradient descent 방법의 변형인 backpropagation 학습 알고리즘[25]의 개발로 실제 여러 응용 분야에 효과적으로 사용되어 왔다. 하지만, MLP의 가장 큰 문제점으로는 backpropagation 알고리즘의 느린 학습 속도를 들 수 있다. 이는 이 알고리즘에서 사용되는 다중 계층에서의 오차의 역전파 과정 때문이다.

또 다른 비궤환 구조인 radial basis function network은 가우시안과 같은 원형 대칭인 뉴론 특성 함수를 사용하는 중간층을 두며, 일반적으로 지도(supervised) 학습과 무지도 (unsupervised) 학습을 결합한 형태의 학습 알고리즘을 사용한다. 이 RBF network 역시 MLP와 유사한 매우 우수한 근사 능력을 가지고 있음이 증명되었으나[28], 실제 응용에 있어 여러 가지 문제점을 갖는다. 예를 들어, 중간층 뉴론의 센터 벡터를 결정하는데 주로 사용되는 clustering 방법인 k-means 알고리즘을 위한 계산이 필요하고, 또한 알고리즘의 초기에 지정해 주는 중간층 뉴론의 수에 따라 성능의 기복이 매우 심하다. 또한 backpropagation 알고리즘과 같은 지도 학습 방법의 사용도 가능하나 이때는 MLP와 마찬가지로 학습 속도가 매우 느리게 된다.

다른 형태의 비궤환 신경회로망 모델로서 본 논문에서 중점적으로 다루어질 고차 신경회로망 (higher-order feedforward network)은, 입력 변수들의 다항식으로 고차 상관 (higher-order correlations)을 구하고 이들을 이용하여 비선형 함수에 대한 모델을 구성한다. 이러한 고차 신경회로망의 기본이 되는 구조는 Giles와 Maxwell에 의해 연구된 고차 처리 유니트 (higher-

order processing unit, HPU)[14]이다. HPU는 비선형 함수의 근사를 위해 중간층 뉴론들을 갖는 수직적 구조의 MLP와는 달리, 다변수 (multivariate) 입력들의 다항식 항들을 구성하여 이들을 새로운 입력으로 하는 단계층 퍼셉트론을 사용하는 수평적 구조의 모델이다. 이러한 HPU의 기본 기능은 아래와 같은 식으로 표현될 수 있다.

$$y = \sigma\Big(w_0 + \sum_j w_j x_j + \sum_{j,k(j\le k)} w_{jk} x_j x_k + \sum_{j,k,\ell(j\le k\le \ell)} w_{jk\ell} x_j x_k x_\ell + \cdots\Big) \quad (1)$$

여기서, y는 HPU의 출력, $\sigma(\cdot)$는 sigmoid나 선형 함수와 같은 뉴론 특성 함수이고, x_j는 d개의 다변수 입력 벡터 $\boldsymbol{x} = (x_1, \cdots, x_j, \cdots, x_d)$의 j번째 입력, $w_{jk\ell}$와 w_0는 입력 x_j, x_k, x_ℓ들의 곱으로부터 출력으로의 연결 강도 및 바이어스를 각각 나타내며, 학습 알고리즘에 의해 변화되는 값들이다. 위 식으로부터 n차 HPU의 출력은 상수항부터 d개의 입력 변수의 n차까지의 조합 가능한 모든 다항식 항들로 구성된 (만약에 sigmoid와 같은 뉴론 특성 함수를 사용하면, 다항식에 이 뉴론 특성 함수가 적용된) 비선형 함수로 표시된다. 즉, HPU는 기본적으로 다변수 입력에 대한 주어진 차수의 Gabor-Kolmogorov 다항식[18]이라 생각할 수 있다. 여기서 주의하여야 할 것은, HPU의 구조 자체는 이제 이 다항식 항들을 새로운 입력으로 하는 단계층 퍼셉트론으로 볼 수 있으며, 따라서 단계층 퍼셉트론에 사용되는 빠르고 간단한 LMS 알고리즘과 같은 학습 알고리즘을 사용할 수 있으며, 동시에 비선형 함수의 근사에도 적용 가능하게 되었다는 점이다.

HPU의 수학적인 근사 능력은 Stone-Weierstrass 정리 혹은 이의 한 변형인 Weierstrass 다항식 근사 정리에 의하여 충분히 보장된다[30]. 즉, Weierstrass 다항식 근사 정리에 의하면, d차원 유클리드 공간의 compact 부분집합 K에 정의되는 임의의 연속 함수에 대해서 무한히 작은 절대 오차를 갖는 다항식이 존재하며, HPU는 이러한 주어진 미지의 연속 함수를 유한한 차수를 갖는 다항식으로 근사함을 의미한다. 여기서, 각 다항식 항의 계수 (즉, 연결 강도)를 적응적 학습 알고리즘을 사용하여 찾아내는 것이 바로 신경회로망의 측면에서 본 HPU라 할 수 있다.

식 (1)과 같이 주어지는 HPU는 비선형 신호 처리 분야에서는 이미 오래 전부터 사용되어 왔다. 잘 알려진 Volterra 급수에 기초한 다항식 필터[24],[32]는 비선형 동적 시스템의 입력, 출력 관계를 식 (1)과 같은 형태로 모델화하고, 각 다항식 계수를 변형된 Wiener-Hopf 방정식[39]을 풀거나, LMS 알고리즘 혹은 RLS 알고리즘과 같은 반복 알고리즘을 이용하여 구하는 방법을 사용하여 왔다. 따라서, LMS 알고리즘과 같은 반복 적응 학습 알고리즘을 사용하는 다항식 필터의 경우 HPU와 기능적으로 동일함을 알 수 있다. 하지만, 이들이 사용되는 응용 분야에 있어서, 다항식 필터의 경우 주로 동적 시스템 인식 (dynamical system identification) 분야에 사용되고 있고, HPU의 경우 시스템 인식 뿐이 아니라 패턴 분류 등과 같은 정적 (static) 문제에 사용되고 있다.

위에서 살펴본 것처럼, HPU는 비선형 함수에 대한 근사 능력, 빠른 학습 알고리즘의 사용 등 기존의 비궤환 신경회로망의 장점을 보존하고 단점은 제거할 수 있는 우수한 모델이나, 입력 변수의 수 d와 필요한 다항식 차수 n이 증가함에 따라 요구되는 뉴론간 연결 수가 지수적으로 증가한다는 큰 단점이 있다. 즉, d개의 입력 변수를 갖는 n차 HPU에 필요한 연결수는 $\sum_{i=0}^{n}\binom{d+i-1}{i} = \binom{d+n}{n}$ 이 되며, 우리가 흔히 접하는 일반적인 다변수 함수 근사 응용 문제에 다루기 어려울 정도로 많은 연결이 필요하게 되기 쉽다. 따라서, 실제 응용에서는 2차나 3차 다항식으로 모델을 제한하는 경우가 많으며, 이는 바로 근사 능력의 한계를 의미한다. 최근에는 이런 문제를 해결하기 위해, 직교 탐색 (orthogonal search)[21]등을 통해 불필요한 항들을 제거하는 방법이 많이 연구되고 있다.

본 논문에서는 HPU를 기반으로 한 일반적인 고차 신경회로망들의 단점을 극복함과 동시에 그의 장점을 보존하는 새로운 형태의 고차 신경회로망인 ridge polynomial network (RPN)[34]의 수학적 근사 능력을 고찰하고, 이를 비선형 다변수 함수 근사와 패턴 분류 문제에 응용한다. 이를 위해 다음 장에서는 RPN의 기본 구성 모델인 pi-sigma network (PSN)[13]의 이론과 그의 학습 알고리즘에 대하여 살펴본다. 3 장에서는 RPN의 구조를 릿지 다항식 (ridge polynomial)의 새로운 형태로부터 유도해 내고, 일반적 비선형 연속 함수에 대한 근사 정리를 증명한다. 이 장에서는 또한

RPN을 PSN의 일반화된 형태로 표현할 수 있음을 보인다. PSN의 학습 알고리즘에 바탕을 둔 RPN의 점진적 학습 알고리즘 (constructive learning algorithm)에 대한 기술 역시 3 장에서 다루어 진다. 4 장에서는 HPU외의 다른 고차 신경회로망 모델들, 통계학에서 이용되고 있는 projection pursuit regression 방법 등과 RPN을 비교, 분석한다. 다음 5 장에서는 컴퓨터 모의 실험을 통해 RPN을 다변수 함수의 근사 및 패턴 분류 문제 등에 응용하며, 마지막으로 6 장에서 결론을 내린다.

Ⅱ. Pi-sigma Network (PSN)

1 장에서 살펴본 것과 같이, HPU는 우수한 근사 능력, 빠른 학습 알고리즘의 사용 등과 같은 장점을 가지고 있으나, 실제 응용에 있어서 모델 내에 너무나 많은 계수를 필요로 하는 단점이 있다. 이러한 문제를 해결하기 위해 제안된 pi-sigma network (PSN)[13]은, d개 입력 변수에 대한 n차 다변수 다항식을 구현하는데 있어, HPU에서와 같이 입력 변수들의 곱 (pi)들의 합 (sigma)를 취하는 대신, d개 입력들의 선형 결합 (sigma)들을 n개 구성 후, 이들의 곱 (pi)을 출력으로 한다. 즉, n차 PSN의 출력 y: $\boldsymbol{R}^d \rightarrow \boldsymbol{R}$는 아래의 식과 같이 주어지며, 그림 1과 같이 표현될 수 있다.

$$y = \sigma\Big(\prod_{i=1}^{n}\Big\{\sum_{j=1}^{d}(w_{ij}x_j + w_{i0})\Big\}\Big) = \sigma\Big(\prod_{i=1}^{n} h_i\Big) \tag{2}$$

여기서, $\sigma(\cdot)$는 앞서와 같이 sigmoid나 선형 함수와 같은 뉴론 특성 함수, h_i는 i번째 선형 결합, w_{ij}와 w_{i0}는 각각 학습에 의해서 변화되는 뉴론간 연결 강도 및 바이어스이다. 위 식 (2)에서 필요한 연결 강도의 수는 바이어스를 포함하여 모두 $(d+1)n$이 되며, 이는 HPU에서 요구되는 연결 강도의 수보다 월등히 적음을 쉽게 알 수 있다. 여기서 주의하여야 할 것은, PSN은 식 (2)에 의해 d개 입력 변수에 대한 n차 다변수 다항식 내의 모든 계수를 생성한다는 점이다. 이런 것이 가능한 것은 다항식 내의 모든 항들의 계수가 식 (2)의 w_{ij} 와 w_{i0}의 적절한 합과 곱으로 표시될 수 있기 때문이며, 이것이 바로 PSN이 매우 적은 연결 강도를 가질 수 있는 이유가 된다. 따라서 식 (2)로 표현되는 PSN 내의 연결 강도들은 HPU내의 계수들과는 달리 상호 독립적이지 않으며, 이런 의미에서 근사 능력과 수학적 표현 능력의 측면에서 PSN은 HPU의 제한적인 구현이 된다.

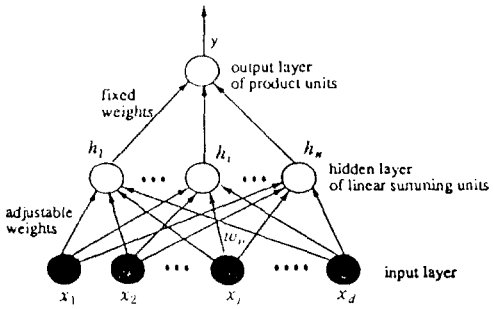


그림 1. 하나의 출력을 갖는 pi-sigma network.
Fig. 1. A pi-sigma network with one output.

그림 1에서 보면, 중간의 선형 결합인 h_i는 MLP에서와 같이 중간 "은닉층"을 형성하는 것처럼 보이나, 실제 이 층으로부터 출력층으로의 연결이 변화하지 않고 고정된 값들이므로 실제 PSN내에서 학습을 통해 변화시켜 주어야 하는 연결은 입력층과 중간층 사이의 연결만이다. 따라서 PSN을 위해 단계층 퍼셉트론 등에 사용되는 간단하고 빠른 학습 알고리즘을 사용할 수 있다. 실제로 PSN을 위한 학습 알고리즘으로는 아래와 같은 평균 자승 오차 (mean squared error) ε을 성능 지표로 하는 LMS 알고리즘의 변형된 형태를 사용한다.

$$\varepsilon \equiv \frac{1}{L}\sum_{p=1}^{L}(f^p - y^p)^2 \equiv \frac{1}{L}\sum_{p=1}^{L}\varepsilon^p \tag{3}$$

여기서, f^p와 $y^p = \sigma(\prod_i h_i^p)$는 각각 p번째 입력 패턴에 대한 목표 출력 및 PSN의 실제 출력, L은 학습 데이터의 개수이다. HPU등의 모델과는 달리 PSN은 출력층에서 곱셈 연산자를 사용하므로, 만약 동시에 식 (2)의 모든 w_{ij}와 w_{i0}를 변화시킬 경우 우리가 원하는 gradient 방향을 취할 수 없게 된다. 이를 해결하기 위해, [13]에서는 비동기 (asynchronous) LMS 알고리즘을 제안하였다. 즉, 하나의 학습 데이터 입력-출력 쌍에 대해 하나의 선형 결합에 사용되는 연결 강도들 (고정된 i에 대한 w_{ij}와 w_{i0})만을 학습 알고리즘을 사용하여 변화 시키고, 이 변화된 연결 강도를 이용하여 동일한 입력에 대해 새로운 실제 출력값을 계산, 이를 이용

하여 다른 선형 결합의 연결 강도를 변화 시키며, 이런 과정을 n개의 선형 결합의 연결 강도가 변화될 때까지 반복한다. 일단 알고리즘을 통해 변형될 연결 강도 w_{ij}와 w_{j0}가 결정되면, 순시 자승 오차 (instantaneous squared error) ε^p를 평균 자승 오차 ε의 추정치로 하여 LMS 알고리즘을 적용한다. 즉,

$$\Delta w_{ij} \propto -\frac{\partial \varepsilon^p}{\partial w_{ij}} \quad (j=0,1,\cdots,d) \tag{4}$$

와 같은 관계로부터, 다음의 개선식이 유도된다.

$$\Delta w_{ij} = \eta \cdot (f^p - y^p) \cdot (y^p)' \cdot \left(\prod_{k \neq i} h_k^p\right) \cdot x_j \quad (j=0,1,\cdots,d) \tag{5}$$

여기서 η는 학습률, $(y^p)'$는 출력의 1차 미분, 그리고 x_0는 1이다. 이 학습 알고리즘은 LMS 알고리즘의 경우와 같이 매우 작은 학습률이 가정되면 수렴하는 것이 증명되었고[13], 실제 응용에 있어서 매우 빠른 학습 수렴 속도를 보인다. PSN은 수학적으로 HPU의 제한된 구현이지만, 실제 패턴 인식[13], 문자 인식[35], 제어[19], Boolean 함수 구현[33], 화자 인식[36], ATM에서 비디오 신호를 위한 동적 대역폭 할당[4], 로보트용 보정기[40]등의 비선형 응용 분야에서 성공적으로 사용되고 있다.

Ⅲ. Ridge Polynomial Network (RPN)

3.1. RPN의 정의 및 근사 정리

d차원 유클리드 공간의 compact 부분집합 K에 정의되는 함수 f $(\langle \cdot, \boldsymbol{w} \rangle)$: $K \rightarrow \boldsymbol{R}$를 릿지 함수 (ridge function)라 한다. 여기서 $\boldsymbol{w} \in \boldsymbol{R}^d$이고 $\langle \boldsymbol{x}, \boldsymbol{w} \rangle = \sum_{j=1}^{d} x_j w_j$는 두개의 d차원 벡터 $\boldsymbol{x}$와 $\boldsymbol{w}$에 대한 내적 (inner product), $f(\cdot)$: $\boldsymbol{R} \rightarrow \boldsymbol{R}$는 연속 함수이다. 릿지 다항식 (ridge polynomial)은 내적을 새로운 입력으로 하는 다항식인 릿지 함수로서 아래와 같이 표현될 수 있다.

$$\sum_{i=0}^{n} \sum_{j=1}^{m} a_{ij} \langle \boldsymbol{x}, \boldsymbol{w}_{ij} \rangle^i \tag{6}$$

여기서 $a_{ij} \in \boldsymbol{R}$, $\boldsymbol{w}_{ij} \in \boldsymbol{R}^d$이다. 다음의 정리는 식 (6)과 같은 릿지 다항식으로 식 (1)에서와 같은 임의의 다변수 다항식을 정확히 표현할 수 있음을 보인다.

정리 1 (Chui & Li[5]) d개의 입력 변수에 대해 최고차 항의 차수가 k인 다항식들의 집합을 P_k^d라 하고, $\boldsymbol{x}^{jm}$는 d개의 입력 변수에 대해 전체 차수가 j인 다항식 항을 나타낸다고 하자. 즉, 주어진 임의의 m에 대해서 $\boldsymbol{x}^{jm} = x_1^{i_{jm1}} \cdots x_d^{i_{jmd}}$이고 $i_{jm1} + \cdots + i_{jmd} = j$이다. 이때 P_k^d에 속한 임의의 다항식

$$p(\boldsymbol{x}) = \sum_{j=0}^{k} \sum_{m=1}^{n_j} c_{jm} \boldsymbol{x}^{i_{jm}} \quad (c_{jm} \in R) \tag{7}$$

에 대해, 다음의 식을 만족하는 릿지 다항식의 계수 $a_{jm} \in \boldsymbol{R}$, $\boldsymbol{v}_{jm} \in \boldsymbol{R}^d$이 존재한다.

$$p(\boldsymbol{x}) = \sum_{j=0}^{k} \sum_{m=1}^{n_j} a_{jm} \langle \boldsymbol{x}, \boldsymbol{v}_{jm} \rangle^j \tag{8}$$

여기서 $n_j = \binom{d+j-1}{j}$로서 d개 입력 변수 j차 다항식 항의 수이다. ■

위의 정리에서 예를 들어 $d = 3$ 즉, x_1, x_2, x_3)이고 최고차 항이 $k = 4$이면, $x_1^3 x_2$, $x_1 x_2 x_3$ 등이 $\boldsymbol{x}^{i_{3m}}$으로 표현될 수 있으며,

$$p(\boldsymbol{x}) = 2x_2 - x_1^2 + 5x_1^3 x_3 - 3x_1 x_2^2 x_3$$
$$p(\boldsymbol{x}) = x_1 x_3 + 10 x_2^4$$
$$\vdots$$

등이 P_3^4에 속하는 다항식들의 예라 할 수 있다.

정리 1을 바탕으로, 좀 더 일반적이고 효율적인 형태를 갖는 릿지 다항식을 구성할 수 있다. 다음의 정리 2는 이런 새로운 형태의 릿지 다항식으로 표현되는 ridge polynomial network (RPN)이 2 장에서 살펴 본 pi-sigma network (PSN)의 단순한 합으로 표시될 수 있으며, 식 (1) 또는 (7)로 표시되는 일반적인 다변수 다항식과 같이 좋은 근사 능력을 가짐을 의미한다[34].

정리 2 P_k^d에 속한 임의의 다항식

$$p(\boldsymbol{x}) = \sum_{j=0}^{k} \sum_{m=1}^{n_j} c_{jm} \boldsymbol{x}^{i_{jm}} \quad (c_{jm} \in R)$$

에 대해, 다음의 식을 만족하는 $w_{ij} \in \boldsymbol{R}$, $w_{ij} \in \boldsymbol{R}^d$이 존재한다.

$$p(\boldsymbol{x}) = (\langle \boldsymbol{x}, \boldsymbol{w}_{11} \rangle + w_{11}) + (\langle \boldsymbol{x}, \boldsymbol{w}21 \rangle + w_{21}) \cdot (\langle \boldsymbol{x}, \boldsymbol{w}_{22} \rangle + w_{22}) + (\langle \boldsymbol{x}, \boldsymbol{w}_{31} \rangle + w_{31}) \cdot (\langle \boldsymbol{x}, \boldsymbol{w}_{32} \rangle + w_{32}) \cdot (\langle \boldsymbol{x}, \boldsymbol{w}_{33} \rangle + w_{33})$$

$$\vdots \qquad (9)$$

$$+((\langle \boldsymbol{x}, \boldsymbol{w}_{N1}\rangle + w_{N1}) \cdot (\langle \boldsymbol{x}, \boldsymbol{w}_{N2}\rangle + w_{N2}) \cdot (\langle \boldsymbol{x}, \boldsymbol{w}_{NN}\rangle + w_{NN})$$

$$= \sum_{i=1}^{N} \prod_{j=1}^{i} (\langle \boldsymbol{x}, \boldsymbol{w}_{ij}\rangle + w_{ij})$$

여기서 $N = \sum_{i=0}^{k} n_i$이다. ■

증명 식 (8) 우변이 식 (9) 우변의 특별한 형태임을 보임으로서 증명한다. 식 (8)로부터,

$$\sum_{j=0}^{k} \sum_{m=1}^{n_j} a_{jm} \langle \boldsymbol{x}, \boldsymbol{v}_{jm}\rangle^j = \sum_{m=1}^{n_0} a_{0m} + \sum_{m=1}^{n_1} a_{1m} \langle \boldsymbol{x}, \boldsymbol{v}_{1m}\rangle + \cdots + \sum_{m=1}^{n_k} a_{km} \langle \boldsymbol{x}, \boldsymbol{v}_{km}\rangle^k = \sum_{i=0}^{k} p_i(\boldsymbol{x}) \qquad (10)$$

여기서,

$$\sum_{m=1}^{n_0} (\langle \boldsymbol{x}, 0\rangle + a_{0m})(\langle \boldsymbol{x}, 0\rangle + 1)^{m-1}$$

이고 $\cdots$, k에 대해

$$\sum_{m=1}^{n_i} (\langle \boldsymbol{x}, a_{im} \boldsymbol{v}_{im}\rangle + 0)(\langle \boldsymbol{x}, \boldsymbol{v}_{im}\rangle + 0)^{i-1}(\langle \boldsymbol{x}, 0\rangle + 1)^{n_0 + \cdots + n_{i-1} + m - i}$$

여기서 $\boldsymbol{x}, \boldsymbol{v}_{jm} \in \boldsymbol{R}^d$, $a_{jm} \in R$, $a_{jm}\boldsymbol{v}_{jm} = (a_{jm}v_{jm1}, \cdots, a_{jm}v_{jmd}) \in \boldsymbol{R}^d$, $\boldsymbol{0} = (0, \cdots, 0) \in \boldsymbol{R}^d$이다. $p_0(\boldsymbol{x})$에서는 $j \equiv m$, $p_i(\boldsymbol{x})$ ($i = 1, \cdots, k$)에서는 $j \equiv n_0 + \cdots + n_{i-1} + m$로 변수 변환하면 식 (10)은 아래와 같이 정리된다.

$$\sum_{j=0}^{k} \sum_{m=1}^{n_j} a_{jm} \langle \boldsymbol{x}, \boldsymbol{v}_{jm}\rangle^j = \sum_{i=0}^{k} p_i(\boldsymbol{x}) = \sum_{j=1}^{n_0} (\langle \boldsymbol{x}, 0\rangle + a_{0j})(\langle \boldsymbol{x}, 0\rangle + 1)^{j-1} + \sum_{j=n_0+1}^{n_0+n_1} (\langle \boldsymbol{x}, a_{1,j-n_0} \boldsymbol{v}_{1,j-n_0}\rangle + 0)(\langle \boldsymbol{x}, 0\rangle + 1)^{j-1}$$

$$\vdots$$

$$+ \sum_{j=n_0+\cdots+n_{k-1}+1}^{n_0+\cdots+n_k} (\langle \boldsymbol{x}, a_{k,j-(n_0+\cdots+n_{k-1})} \boldsymbol{v}_{k,j-(n_0+\cdots+n_{k-1})}\rangle + 0)^k (\langle \boldsymbol{x}, 0\rangle + 1)^{j-k} \qquad (11)$$

또한 식 (9)로부터 아래 식이 성립된다.

$$\sum_{i=1}^{N} \prod_{j=1}^{i} (\langle \boldsymbol{x}, \boldsymbol{w}_{ij}\rangle + w_{ij}) = \sum_{i=0}^{n_0} \prod_{j=1}^{i} (\langle \boldsymbol{x}, \boldsymbol{w}_{ij}\rangle + w_{ij}) + \sum_{n_0+1}^{n_0+n_1} \prod_{j=1}^{i} (\langle \boldsymbol{x}, \boldsymbol{w}_{ij}\rangle + w_{ij}) \qquad (12)$$

$$\vdots$$

$$+ \sum_{n_0+\cdots+n_{k-1}+1}^{n_0+\cdots+n_k} \prod_{j=1}^{i} (\langle \boldsymbol{x}, \boldsymbol{w}_{ij}\rangle + w_{ij})$$

식 (11)이 식 (12)의 한 형태이므로 위 정리가 증명된다. ■

정리 2로부터, 다차원 유클리드 공간의 compact 부분집합 K에 정의된 임의의 연속 함수 f에 대하여, 식 (9)로 정의된 RPN의 근사 능력은 아래와 같이 쉽게 보여 진다.

정리 3 f를 compact 부분집합 K에 정의된 임의의 연속 함수라 하고, 임의의 상수 $\varepsilon > 0$이 주어져 있다고 하자. 이 때, 모든 $\boldsymbol{x} \in K$에 대해 $|f(\boldsymbol{x}) - p(\boldsymbol{x})| < \varepsilon$을 만족하는 식 (9)의 형태를 갖는 RPN p가 존재한다. ■

증명 Weierstrass 다항식 근사 정리[30]에 의해서, 모든 $\boldsymbol{x} \in K$에 대해 $|f(\boldsymbol{x}) - p(\boldsymbol{x})| < \varepsilon$을 만족하는 식 (7)과 같은 다변수 다항식 p가 존재한다. 정리 2로부터, 이 다항식 $p(\boldsymbol{x})$는 식 (9) 우변의 RPN으로 정확히 표현된다. ■

위에서 살펴본 바와 같이, RPN은 수학적인 근사 능력에 있어서 일반적 다변수 다항식인 HPU와 동일함을 알 수 있다. 여기서 우리가 주목하여야 할 점은, RPN을 표현하는 식 (9)내 덧셈 연산자 안의 각 곱셈항들은 다름 아닌 i차 PSN이며, 따라서 RPN은 저차의 PSN으로부터 고차의 PSN을 더하여 구하여진 PSN의 일반화된 모델이라 할 수 있다. 즉 PSN의 단점으로 2 장에서 지적되었던 제한된 수학적 근사 능력을 RPN은 그들의 단순한 합을 통해 극복하였음을 알 수 있다.

PSN들의 단순한 합으로 RPN을 구성하는데는 다음과 같은 의미가 있다고 할 수 있다. 첫째, 고차의 새로운 PSN을 더하므로써 전체 RPN이 이 더해진 PSN과 같은 차수를 갖게 되어 주어진 문제의 고차 비선형성을 좀 더 잘 근사하게 된다. 둘째, 예를 들어 k차 PSN을 새로 더할 경우 k차 항만이 생성되는 것이 아니고 상수부터 k-1차까지의 항도 역시 생성되므로 이미 있던 k-1차 PSN에 의해 형성된 불완전한 k-1까지의 항을 보완하게 된다 (5.1 절의 모의 실험 결과 참고). 셋째, RPN이 단순한 PSN의 합으로 표현되므로 2 장에서 논의된 PSN을 위한 효율적인 학습 알고리즘을 그대로 사용 가능하다 (3.2 절 참고).

위의 정리 2와 3에서 우리가 주의해야 할 점은, 이 정리들이 RPN이 PSN의 일반화된 형태라는 점을 보이

고 이의 근사 능력의 증명을 위한 하나의 limiting case로 이해되어야 한다는 것이다. 실제로 MLP의 근사 능력도 중간 은닉층이 무한히 많은 뉴론을 갖는다는 가정에서 얻어지는 결과이며[16], 실제 문제의 적용에 있어 무한히 많은 뉴론을 사용할 수는 없다는 점에서 위와 유사한 논의가 될 수 있다. 따라서 이러한 결과는 RPN이나 MLP라는 구조의 근사 능력만을 보여주는 것으로서 이런 성질이 없는 다른 신경회로망 구조들보다는 월등히 우수한 구조임을 증명하는 사실로 이용될 수는 있으나, 실제 이러한 좋은 구조를 만족하는 계수들을 얻는 것은 학습 알고리즘 등이 연관되는 전혀 별개의 문제로 다루어져야 한다. 다시 말해 정리 2와 3의 결과는 저차부터 고차까지 여러 PSN을 더함으로써 얻게 되는 RPN으로 표현되는 함수들이 충분히 많은 PSN이 더해진다는 가정 하에서 좋은 근사 능력을 갖는다는 limiting case에 대한 증명이다.

위에서 또한 주의하여야 할 점은 식 (9) 혹은 (12)로 표현되는 RPN에 필요한 계수의 수는 $O(dN^2)$로서 이는 HPU의 $O(N)$보다 많은 수라는 것이다. 하지만 이는 위에서 언급된 대로 근사 능력의 증명을 위한 경우이며, 아래의 3.2절에서 살펴볼 것처럼 실제 문제에 적용시 우리가 RPN을 이용하여 n차 다항식을 구현하고자 할 때 1차부터 n차까지의 PSN을 더하여 구현하게 된다. 이러한 구현은 n차 HPU와 정확히 같지 않은 이의 근사된 형태지만 계수의 수는 $O(dn^2)$로 HPU의 경우에 비해 상당히 감소된 수가 되며, 단순한 n차 PSN의 여러 장점을 보존하면서 약간의 계수 증가만으로 나은 구조를 얻게 된다는 장점을 갖게 된다. 즉 우리는 RPN을 이용하여 HPU와 동일한 구조를 얻고자 하는 것이 아니라, 정리 2와 3의 결과에 근거해 PSN을 추가시켜 나가는 점진적 학습 알고리즘을 통하여 원하는 오차 내의 결과를 얻고자 하는 것이다. 따라서 RPN이 "효율적"이다라는 표현은 만약 RPN과 HPU의 차수가 같다면 RPN의 파라미터의 수가 적다는 의미로 이해되어야 한다.

3.2. RPN을 위한 점진적 학습 알고리즘

신경회로망의 분류 기준으로 "학습 중 모델의 구조(structure) 변화 가능성"을 사용할 경우, 정적 구조와 동적 구조로 나눌 수 있다. 여기서 모델의 구조란 사용되는 뉴론의 수, 뉴론간 연결 등을 의미하며, 일반적으로 정적 구조를 갖는 모델은 학습이 끝날 때까지 이들이 변화하지 않고, 변화하는 것은 뉴론간 연결 강도의 값들뿐이다. 1 장에서 살펴 본 대부분의 비귀환 신경회로망 모델이 이에 해당한다.

정적 구조와는 대조적으로, 동적 구조를 갖는 모델들은 학습 중 뉴론간 연결 강도뿐이 아니라, 뉴론의 수나 연결 자체가 생성 혹은 소멸되는 구조를 말한다. 이 구조는 다시 점진적 방법(constructive 또는 incremental method)과 소멸형 방법 (destructive 또는 pruning method)으로 구분되며, 전자의 경우로 dynamic node creation 알고리즘[3], cascade-correlation 알고리즘[9], resource allocation network[29]등이 있으며, 후자의 예로는 weight decay 알고리즘[22], optimal brain damage 알고리즘[23]등이 있다.

동적 구조를 갖는 모델들은 정적 구조에 비해 다음과 같은 장점을 갖는다. 첫째, 사용되는 뉴론의 수나 연결 강도의 수에 의해 정해지는 모델의 복잡도에 따라서 모델은 매우 다른 근사 성능을 보인다. 만약에 어떤 신경회로망 모델이 주어진 문제에 비해 적은 복잡도를 갖고 있다면, 이 함수를 제대로 근사할 수 없게 되고, 이와 반대로 너무 복잡하다면 주어진 학습 데이터를 overfitting하기 쉬워진다[12]. 정적 구조의 모델들은 사용자가 학습 전에 문제의 복잡도를 추측하여 모델의 구조를 정하여 주어야 하지만, 동적 구조를 갖는 모델들은 주어진 문제에 따라 모델 자체의 구조를 학습 중간에 바꾸어 최적화 할 수 있다. 둘째, 정적 구조 모델의 경우 만약 원하는 결과를 얻지 못할 때는 다른 복잡도를 갖는 새로운 모델을 이용하여 다시 결과를 얻게 되며, 이는 바로 학습 시간의 연장을 의미한다. 하지만 동적 구조를 갖는 모델, 특히 점진적 방법의 경우 학습 알고리즘의 진행에 따라 이 문제가 자동적으로 조절될 수 있다.

식 (9)의 일반적 표현에 기반하여, 주어진 임의의 함수 f는 최고차가 n인 RPN으로 다음과 같이 근사된다.

$$f(\boldsymbol{x}) \approx \sigma\left(\sum_{i=1}^{n} P_i(\boldsymbol{x})\right) \tag{13}$$

여기서 $\sigma(\cdot)$는 뉴론 특성 함수, $P_i(\boldsymbol{x}) = \prod_{j=1}^{i}(\langle \boldsymbol{x}, \boldsymbol{w}_j \rangle + w_{j0})$ $(i = 1, \cdots, n)$은 선형 함수를 출력층 뉴론 특성 함수로 하는 i차 PSN의 출력값이다. 아래의 그림 2는 위 식 (13)으로 표현되는 RPN을 나타낸다. 여기서

PSN_i는 선형 함수를 출력층 뉴론 특성 함수로 하는 i차 PSN의 출력값이다. 그림 2에서와 같이 단계층 퍼셉트론과 동일한 1차 PSN으로부터 차수를 1차씩 증가시켜 최종적으로 n차의 PSN까지 사용하여 RPN을 구성할 경우, 실제 필요한 뉴론 연결의 수는 앞서 논의된 대로 $O(dn^2)$가 되며, 이는 PSN의 $(d+1)n$ 보다는 증가되었지만, HPU 경우에 비해 월등히 적은 수이다.

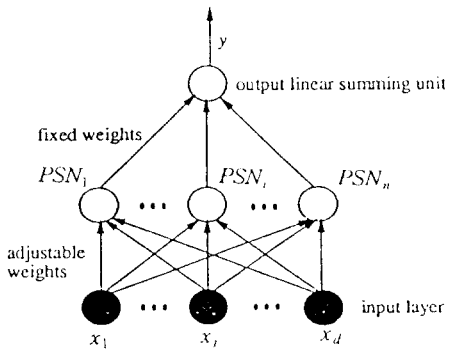


그림 2. 하나의 출력을 갖는 ridge polynomial network.
Fig. 2. A ridge polynomial network with one output.

실제 RPN의 학습 알고리즘은 위에서 살펴 본 분류 가운데 점진적 방법을 사용하여 그림 2와 같은 구조를 형성하여 나간다. 즉 RPN 구성의 처음 단계에서 1차와 같은 저차의 PSN으로 함수를 근사하고, 이 PSN의 학습이 끝나면 실제 함수와의 오차를 좀 더 고차의 PSN으로 근사하여 나간다. 결국 알고리즘 수행의 각 단계에서 임의의 함수는 PSN만을 사용하여 근사되고, 따라서 2 장에서 살펴 본 PSN을 위한 빠른 학습 알고리즘을 그대로 사용할 수 있게 된다. 이를 통해 RPN은 주어진 임의의 함수를 overfitting 없이 빠르게 근사할 수 있다. 표 1은 RPN을 위한 점진적 학습 알고리즘을 정리하여 보여준다. 이를 위해 다음과 같은 표기를 사용한다 - ε_{th} : 학습 종료 여부의 결정을 위한 임계 평균 자승 오차, ε_c, ε_p : 학습 중 현재 및 바로 직전 epoch의 평균 자승 오차, r: 다음 차수의 PSN 사용 여부를 결정하기 위한 임계치, η: 초기 학습률, δ_r, δ_η: r과 η를 위한 감소 계수 ($0 < \delta_r, \delta_\eta < 1$), P_n: n번째 알고리즘 스텝에서의 n차 PSN, t: 사용된 학습 epoch의 수, tth: 학습 종료 여부의 결정을 위한 학습 epoch 수의 임계치, $\hat{P}_n$: 학습을 마친 후 모델 내의 모든 계수가 학습 종료 시의 값을 그대로 유지하는 n차 PSN P_n, y_n: n번째 알고리즘 스텝에서의 RPN 출력값, 즉 $y_n = \sigma(\sum_{j=n_0-1}^{n-1} \hat{P}_j + P_n)$. 이러한 학습 알고리즘을 이용하여 미지의 함수 f는 다음과 같이 점진적으로 근사되어 나간다.

표 1. RPN을 위한 점진적 학습 알고리즘.
Table 1. A constructive learning algorithm for RPN.

1. 초기화: $n \leftarrow n_0 (\geq 1)$, $\hat{P}_{n_0-1} \leftarrow 0$, $t \leftarrow 0$, ε_{th}, ε_p, r, η, δ_r, δ_η, t_{th}에 적절한 값 지정.
2. n차 PSN $P_n(\boldsymbol{x}) = \prod_{i=1}^{n} (\langle \boldsymbol{x}, \boldsymbol{w}_i \rangle + w_{i0})$ 구성. $\boldsymbol{w}_i$와 w_{i0}은 적절히 랜덤한 값으로 초기화.
3. 모든 학습 데이터에 대해 다음의 과정 반복:
(a) RPN의 출력 y_n 계산.
(b) PSN의 비동기 LMS 알고리즘인 아래의 개선식을 이용하여 $i = 1, \cdots, n$에 대해 연결 강도와 바이어스인 $\boldsymbol{w}_i$와 w_{i0} 개선.

$$\Delta w_{ij} = \eta \cdot (f - y_n) \cdot (y_n)^{i} \cdot \left(\prod_{r \neq i}^{n} (\langle \boldsymbol{x}, \boldsymbol{w}_r \rangle + w_{r0}) \right) \cdot x_j \quad (j = 0, 1, \cdots, d)$$

4. 각 학습 epoch의 끝에 (즉, 모든 학습 데이터를 한번씩 사용하여 위의 과정을 수행 후), 현재 epoch의 평균 자승 오차 ε_c 계산.
5. 만약 $\varepsilon_c < \varepsilon_{th}$이거나 $t > t_{th}$이면, 학습 종료.
6. 아닐 경우 다음의 과정 반복:
(a) 만약 $\left| \frac{\varepsilon_c - \varepsilon_p}{\varepsilon_p} \right| \geq r$이면, $t \leftarrow t+1$, $\varepsilon_p \leftarrow \varepsilon_c$, 3번 스텝으로 이동.
(b) 아닐 경우, $\hat{P}_n \leftarrow P_n$, $r \leftarrow r \cdot \delta_r$, $\eta \leftarrow \eta \cdot \delta_\eta$, $t \leftarrow t+1$, $\varepsilon_p \leftarrow \varepsilon_c$, $n \leftarrow n+1$, 2번 스텝으로 이동.

$f \approx \sigma(P_{n_0})$

$f \approx \sigma(\widehat{P}_{n0} + P_{n0+1})$

$f \approx \sigma(\widehat{P}_{n_0} + \widehat{P}_{n_0+1} + P_{n_0+2})$

⋮

Ⅳ. RPN과 타 고차 신경회로망 모델과의 비교

본 장에서는 HPU외에 신경회로망 및 비선형 신호처리 분야에서 사용되고 있는 다항식에 기반을 둔 고차 모델들을 살펴보고, 이들의 장, 단점, RPN과의 차이점 등을 논의 하고자 한다. 또한 통계학에서 사용되고 있는 projection pursuit regression과의 연관성도 살펴보기로 한다. 이를 위해, 먼저 3 장에서 살펴 본 것처럼 고차 모델들을 크게 정적 구조와 동적 구조로 분류하여 논의하고, 다음으로 projection pursuit regression에 대해 토의한다. RPN과의 비교는 이 모델이 동적 구조를 가지므로, 주로 동적 구조의 고차 모델들과의 차이점에 대하여 논의하도록 한다.

4.1. 정적 구조를 갖는 고차 모델

4.1.1. Sigma-pi Unit

Feldman과 Ballard에 의해서 처음으로 논의된 sigma-pi unit[10]는 선택된 몇 개의 입력 변수들의 곱(pi)들을 더한 (sigma) 형태의 뉴론 출력을 갖는다. 즉 HPU는 주어진 차수의 다항식의 모든 항들을 고려하는데 반해, sigma-pi unit는 이 중 주어진 문제에 유용하다고 생각되는 몇 개의 항만을 선택적으로 사용하며, 따라서 이 모델은 HPU의 특수한 경우라 할 수 있다. 이 모델은 단층으로 HPU처럼 사용되거나, 좀 더 나은 비선형 특성을 얻기 위해 MLP 내의 선형 논리 유니트를 대체하여 사용될 수 있다. 하지만, 전자의 경우 이 모델은 $x_j^k(k)1)$와 같은 항을 허용하지 않으므로 근사 능력에 한계가 있고, 후자의 경우 backpropagation 알고리즘에 의해 느린 학습 속도를 갖는 단점이 있다. 또한 사용되는 다항식 항들의 결정이 체계적이지 못한 점도 지적된다.

4.1.2. Product Unit

Sigma-pi unit의 단점을 보완하기 위해, Durbin과 Rumelhart가 제안한 product unit[8]는 다항식의 단일 항을 구현하며, i번째 product unit의 출력 z_i는 다음과 같이 표시된다.

$$z_i = \prod_{j=1}^{d} x_j^{p_{ij}}$$

여기서, 지수 p_{ij}는 일반적인 다항식에서와 같이 정수로 국한되는 것이 아니라, gradient descent 알고리즘을 통해 적응적으로 결정될 수 있는 임의의 실수값이다. 이는 MLP에서의 중간층과 같은 역할을 하며, 여러 개의 product unit로 구성된 신경회로망의 출력 y는 아래와 같이 구해진다.

$$y = \sum_i w_i z_i$$

주어진 문제의 성격에 따라서 product unit은 매우 빠르고 좋은 결과를 얻을 수 있으나 (예를 들어 parity check problem), 기본적으로 backpropagation 형태의 학습 알고리즘을 사용하여야 함으로 수렴 속도가 늦어질 수 있다. 또한 실수 (real-valued) 문제에의 적용 가능성이 충분히 검증되지 않은 상태이다.

4.1.3. Functional Link Net

입력 변수들로부터 고차 비선형 함수를 얻는 방법으로 Pao가 제안한 functional link net[27]는 d차원 입력 $\boldsymbol{x} = (x_1, \cdots, x_d)$에 대해 주어진 미지의 함수를 $\sum_i w_i \phi_i(\boldsymbol{x})$의 형태로 근사하며, 여기서 $\phi_i(\boldsymbol{x})$는 주어진 문제에 따라 적절하게 선택된 기저함수 (basis function)로서, 예를 들어 입력 변수들 자체, 입력 변수들의 곱, 정현 함수 등 임의의 형태가 가능하다. 이 방법은 기본적으로 기저함수 $\phi_i(\boldsymbol{x})$들을 새로운 입력으로 하는 단층 퍼셉트론의 구조이며, 만약 기저함수들을 다항식의 고차항들을 사용하면 HPU 또는 sigma-pi unit와 유사한 기능을 갖게 되고, 단층 퍼셉트론의 여러 가지 장점과 비선형 특성을 동시에 얻을 수 있다. 하지만 이 방법의 문제점으로는 주어진 문제에 대해 최적의 기저함수들을 선택하는 것이다. 다항식의 고차항들을 고정적인 기저함수로 사용하는 HPU와는 달리, 이 방법은 주어진 문제마다 최적의 기저함수들을 찾아야 하며, 이는 실제 매우 어려운 문제이다. 예를 들어 parity check problem을 다루는 경우, 만약 기저함수들로 입력 변수들의 적절한 다항식 항들을 사용하면 이 문제는 매우 쉽게 풀릴 수 있으나, 만약 정현 함수를 기저함수로 사용하게 되면 매우 어려운 문제가 된다.

4.2. 동적 구조를 갖는 고차 모델

4.2.1. GMDH 알고리즘

Ivakhnenko에 의해 제안된 GMDH (group method of data handling) 알고리즘[18]은 다변수 입력에 대해 주어진 미지의 비선형 함수를 점진적으로 근사하는 방법이다. 즉, 처음에는 주어진 함수를 간단한 모델로 근사하고, 알고리즘의 진행에 따라 점점 복잡한 모델을 사용하여 근사해 나간다. GMDH 알고리즘은 기본적인 근사 모델로서 아래와 같은 2개의 입력 변수에 대한 2차 다항식을 사용한다.

$$a_0+a_1x_i+a_2x_j+a_3x_i^2+a_4x_j^2+a_5x_ix_j \tag{14}$$

여기서 a_0, …, a_5는 학습에 의해 결정되는 계수들이고, x_i, x_j는 알고리즘 전 단계의 입력들 중 선택된 2개의 입력이다.

GMDH 알고리즘의 동작 원리는 다음과 같다. 먼저 주어진 데이터를 학습 데이터와 테스트 데이터로 나눈다. 만약 원래의 입력 변수가 d개라면, GMDH 알고리즘은 먼저 학습 데이터만을 사용하여 $\binom{d}{2}$개의 입력 조합에 대해 식 (14)로부터 최소 자승 오차를 만족하는 최적의 계수 a_0, …, a_5를 변형된 Wiener-Hopf 방정식을 풀거나 LMS 알고리즘과 같은 반복 알고리즘을 이용하여 구한다. 이렇게 구해진 계수들을 사용하여, 이번에는 테스트 데이터에 대하여 식 (14)를 계산하고 계산된 값 (즉, 모델의 출력값)과 대응되는 원래의 출력값과 비교하여 테스트 데이터에 대한 오차의 합이 미리 정해진 기준보다 큰 입력 조합을 버리고, 남은 입력 조합을 다음 단계의 입력으로 사용한다. 이러한 과정은 테스트 데이터에 대한 현 단계의 오차가 전 단계보다 커질 때까지 반복된다. 결국, 이 알고리즘을 통해 미지의 함수가 입력 변수들의 고차 polynomial들로 점진적으로 근사되게 된다.

GMDH 알고리즘의 장점으로는 점진적인 근사를 통해 주어진 입력, 출력 데이터에 대한 overfitting을 막을 수 있고, HPU의 단점으로 지적된 너무나 많은 계수의 수효를 줄이면서 매우 뛰어난 근사 결과를 얻을 수 있다는 점이다. 이러한 결과는 간단한 2차 다항식들을 기본 모델로 사용하고, 알고리즘의 각 단계에서 불필요한 항들을 제거함으로써 얻어진다. 하지만, 알고리즘 각 단계에서 사용되는 계수의 계산과 불필요한 항의 제거 등 때문에 계산량이 증가하는 단점이 있다. 이 알고리즘과 RPN을 비교해 보면, 먼저 GMDH 알고리즘은 주어진 문제에 따라 중간 은닉층과 노드 (혹은 뉴론)들이 임의적으로 증가하는데 반해, RPN은 매우 규칙적인 구조를 갖는다. 이러한 구조적인 차이점 외에 다른 중요한 차이점으로, RPN의 경우 GMDH 알고리즘과는 달리 최적의 모델 구조를 얻기 위해 학습 데이터를 이용한 전처리 과정이 필요치 않은 일반적 구조를 유지하나, 이것이 주어진 특별한 문제에 대해 RPN의 구조가 최적이 아닐 수도 있음을 의미할 수 있다.

4.2.2. SONN (Self-organizing Neural Network)

SONN[37]은 GMDH 알고리즘과 유사한 점진적 근사 방법으로 이 알고리즘의 확장된 형태로 생각할 수 있다. 이들 방법의 큰 차이점으로는 먼저 식 (14)와 같은 하나의 2차 다항식을 사용하는 GMDH 알고리즘과는 달리, SONN에서는 다음과 같이 2개 입력 변수에 대한 4개의 2차 이하 다항식들 중에서 가장 적당한 것을 선택하여 사용한다.

$a_0+a_1x_i+a_2x_j$

$a_0+a_1x_i+a_2x_j+a_3x_ix_j$

$a_0+a_1x_j+a_2x_i^2$

$a_0+a_1x_i+a_2x_j+a_3x_i^2+a_4x_j^2+a_5x_ix_j$

또한 학습 알고리즘으로는 simulated annealing[20]을 사용하므로 오차의 global minima를 찾는데 매우 뛰어난 능력을 보인다. 마지막으로, 평균 자승 오차를 성능의 지표로 사용하는 GMDH 알고리즘과 달리, Akaike의 최소 표현 길이 (minimum description length)[2]의 변형된 형태를 지표로 사용함으로써, 근사의 정확도와 모델의 복잡도를 적절히 절충한 좋은 결과를 얻을 수 있다. 하지만, SONN의 가장 큰 문제점으로는 simulated annealing을 사용함으로써 수렴 속도가 매우 느리다는 것이다. GMDH 알고리즘과 마찬가지로 이 알고리즘 역시 주어진 문제에 따라 중간 은닉층과 노드들이 임의적으로 증가한다.

4.2.3. LMS Tree 알고리즘

Sanger가 제안한 LMS tree 알고리즘[31]도 만약 기저함수로 다항식을 사용하면, 동적 구조의 고차 신경회로망으로 분류가 가능하다. LMS tree 알고리즘은 초기에 하나의 입력 변수에 대한 기저함수들을 이용하여

임의의 다변수 함수를 근사하기 시작하며, "최대 오차 분산"을 갖는 기저함수 아래에 tree구조에서처럼 sub-network을 구성한다. 이 sub-network은 이전에 사용되지 않은 새로운 입력 변수에 대한 기저함수들을 사용하며, 이런 과정을 반복하므로써 결국 각 입력 변수들에 대한 기저함수의 곱으로 이루어진 함수들로 미지의 함수를 근사하게 된다. 이 알고리즘 역시 GMDH 알고리즘이나 SONN에서와 같이 불규칙적인 구조로 모델이 형성되어 가는 점이 RPN과 다르며, RPN과는 다르게 각 입력 변수들에 대한 기저함수의 곱으로 형성된 다항식을 이용한다.

4.3. Projection Pursuit Regression

식 (9)나 (13)과 같이 표현되는 RPN은 통계적 방법인 projection pursuit regression[11],[17]과 밀접한 관련을 가지고 있다. 개략적으로 말하면, projection pursuit regression은 고차원 데이터들로부터 어떠한 성능 지표를 최적화해 주는 저차원 사영 (projection)을 구하여, 이들을 이용해 임의의 고차원 다변수 함수를 근사하는 방법이다. 이렇게 저차원 사영을 이용하므로써 이 방법은 "curse of dimensionality"[17]를 피할 수 있으며, 주로 다음 식 (15)와 같이 릿지 함수들의 합으로 d개의 입력 변수를 갖는 미지의 다변수 연속 함수 f를 근사한다.

$$f(\boldsymbol{x}) \approx \sum_{i=1}^{n} a_i g_i(\langle \boldsymbol{x}, \boldsymbol{w}_i \rangle) \tag{15}$$

여기서 $g_i(\cdot) : \boldsymbol{R} \rightarrow \boldsymbol{R}$은 연속 함수, $a_i \in \boldsymbol{R}$, $\boldsymbol{w}_i \in \boldsymbol{R}^d$이다. 일반적으로, g_i는 저차원 사영에서 1차원 데이터 smoothing을 통해 주어진 데이터로부터 직접 결정되며, a_i, $\boldsymbol{w}_i$는 평균 자승 오차와 같은 주어진 성능 지표를 최적화하는 값들을 취하게 된다. Projection pursuit regression은 위에 기술한 방법을 반복적으로 적용하여 g_i, a_i, $\boldsymbol{w}_i$를 구하게 되며, 이를 일반적으로 "backfitting"[11],[17]이라 부른다.

RPN과 projection pursuit regression은 표현의 측면에서 매우 유사하다. 식 (15)는 만약 kernel g_i에 미리 정해진 다항식을 사용하면 식 (9)와 같은 형태가 된다. 하지만, 실제에 있어 고정된 특별한 형태의 릿지 다항식을 사용하고 그 안의 연결 강도만을 학습을 통해 변화시키는 RPN과는 달리, projection pursuit regression은 kernel g_i를 데이터 smoothing을 통해 주어진 문제마다 구해야 한다 (이러한 차이점은 projection pursuit regression과 MLP 사이에서도 존재한다).

Ⅴ. 컴퓨터 모의 실험

본 장에서는 점진적 학습 알고리즘을 이용한 RPN을 다음과 같은 비선형 문제에 응용함으로써, 지금까지 우리가 살펴 본 RPN의 비선형 함수 근사 능력을 실험적으로 검증하고자 한다. 이를 위해 다음과 같은 세 가지 문제가 고려되었다.

처음 살펴 볼 문제로서 RPN과 이의 점진적 학습 알고리즘이 1 장의 식 (1)의 형태로 표현되는 임의의 다변수 다항식을 얼마나 잘 구현 (representation) 할 수 있는가를 알아본다. 이 실험을 통해 정리 2의 결과를 실험적으로 검증하게 된다. 두 번째 살펴 볼 문제는 다변수 비선형 연속 함수의 근사이다. 이를 위해 컴퓨터에 의해 인위적으로 생성된 2차원 Gabor 함수가 사용되었다. 이 실험을 통해 정리 3에서 보여진 RPN의 근사 능력이 점진적 학습 알고리즘을 통해 어떻게 실제 함수 근사 문제에 적용될 수 있는지를 보여준다. 마지막으로 다룰 문제는 실제 해저의 여러 가지 음원으로부터 얻어진 수동 소나 신호를 분류하는 패턴 분류 문제이다. 이 문제는 연속 함수가 아닌 적분 가능한 measurable 함수로 모델화된다 할 수 있으며, 역시 함수 근사 문제로 볼 수 있다.

5.1. 다변수 Polynomial의 구현

실험에 사용된 다변수 다항식은 아래와 같이 $K = [-1,1]^5$에 정의된 5개 입력 변수에 대해 최고차 항이 3차인 다항식이다.

$$f(x_1, x_2, x_3, x_4, x_5) = 2 + 3x_1x_2 + 4x_3x_4x_5 \tag{16}$$

다음 절에서 살펴 볼 다른 두개의 응용 문제가 함수의 근사나 패턴 분류 등과 같이 RPN의 근사 능력을 확인하는 것과는 달리, 이 문제는 식 (16)과 같은 일반적인 다변수 다항식의 계수들을 RPN의 릿지 다항식 구조로 얼마나 잘 구현할 수 있는가에 초점을 맞춘다. 따라서, 학습 데이터와 테스트 데이터 두 종류로 데이터를 나누는 대신 학습 데이터 한 종류를 가지고 RPN의 학습 알고리즘을 수행, 학습이 끝난 후의 연결 강도들을 이용하

여 식 (9)와 같이 표현되는 RPN을 전개하여 얻어지는 일반적 형태의 다항식들의 계수와 식 (16)의 계수를 비교함으로써 성능을 평가한다.

학습 데이터로는 $[-1,1]^5$로부터 랜덤하게 선택된 200개의 입력과 이에 대응하는 식 (16)의 함수값을 사용하였다. RPN내의 연결 강도들은 [-1,1] 사이에서 균일 분포를 갖는 랜덤한 값들로, 점진적 학습 알고리즘에 사용되는 파라미터 η, r, δ_η, δ_r들은 0.02, 0.0001, 0.8, 0.1로 각각 초기화 되었다. RPN은 1차 PSN으로부터 구성되어 점진적 학습 알고리즘을 수행하였고, 각 PSN의 출력 뉴론 특성 함수로는 선형 함수 $\sigma(x)=x$를 사용하였다.

학습 알고리즘 수행 중에 6번째, 22번째, 65번째 epoch에서 자동적으로 2차, 3차, 4차 PSN이 RPN 구조에 추가되었고, 임계 평균 자승 오차 ϵ_{th} = 0.02에 대해서 238번째 epoch에서 학습이 종료되었다. 학습이 끝난 후 얻어진 RPN의 연결 강도를 이용하여 이를 전개 후 얻은 최종적인 다항식 $\hat{f}_{RPN}$는 다음과 같다.

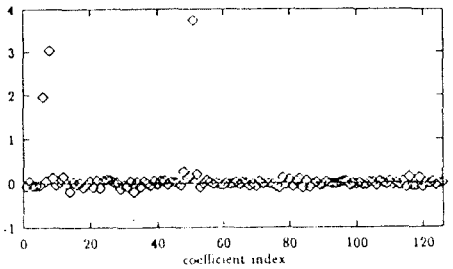
$$\hat{f}_{RPN}(x_1, x_2, x_3, x_4, x_5) \approx 1.96 + 3.03x_1x_2 + 3.74x_3x_4x_5 \quad (17)$$

식 (17)로부터 우리는 RPN의 점진적 학습 알고리즘이 적은 수의 학습 데이터를 이용했음에도 주어진 함수에 대해 상당히 정확한 표현을 얻었음을 알 수 있다. 다음의 그림 3 (a)는 RPN 전개 후 얻어진 다항식의 126개 모든 계수를 나타낸다. 여기서, "126"은 5개의 입력 변수에 대한 4차 다항식이 가질 수 있는 모든 계수의 수이다. 불필요한 계수 중에서 가장 큰 값을 갖는 것은 $x_3^2x_4$항의 계수 0.27이었다. 여기서 주의하여야 할 것은, 주어진 다항식은 최고차 항이 3차이지만, 본 학습 알고리즘은 4차의 PSN까지 이용하였다는 점이다. 하지만 4차 PSN의 4차 항들의 계수는 매우 적어 무시할 만하며, 결국 알고리즘은 4차 PSN 내의 3차 항들을 가지고, 3차 PSN의 3차 항들의 계수값들을 보정하여 주었음을 알 수 있다.

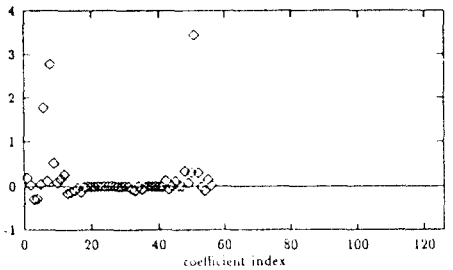
이러한 점진적 학습 알고리즘의 특성을 보기 위해, 그림 3 (b)에서는 3차 PSN까지만을 이용하여 RPN을 구성했을 때 (즉, 4차 PSN을 더하기 직전인 64번째 epoch에서) 얻어지는 계수들을 보여준다. 여기서는 RPN을 전개 후 모두 56개의 계수를 얻을 수 있다. 이런 전개를 통해 얻은 다항식은 다음과 같다.

$$\hat{f}_{RPN}(x_1, x_2, x_3, x_4, x_5) \approx 1.76 + 2.77x_1x_2 + 0.53x_1x_3 + 3.45x_3x_4x_5 \quad (18)$$

위에서 알 수 있듯이, 점진적 학습 알고리즘에 의한 고차 PSN의 추가를 통해 미지의 함수에 대한 좀 더 정확한 근사가 가능해짐을 알 수 있다. 마지막으로 그림 4는 학습 알고리즘의 학습 곡선을 나타내며, 매우 안정적인 학습 특성을 보임을 알 수 있다.



(a)



(b)

그림 3 (a) 4차까지의 PSN을 이용하여 구성한 RPN을 전개 후 얻어진 계수값, (b) 3차까지의 PSN을 이용하여 구성한 RPN을 전개 후 얻어진 계수값. 여기서, "coefficient index"는 RPN을 전개 후 얻는 (a) 126개 혹은 (b) 56개의 계수에 대한 임의의 순서를 나타내는 지수이고, 6번째, 8번째, 51번째 지수가 각각 상수, x_1x_2, $x_3x_4x_5$ 항의 계수를 나타낸다.

Fig. 3. (a) The coefficients obtained after expanding the RPN of orders up to 4. (b) The coefficients obtained after expanding the RPN of orders up to 3. Here, "coefficient index" simply indicates each coefficient out of the total of (a) 128 or (b) 56 coefficients after expanding the RPN. The 6th, 8th, and 51st indices correspond to constant, x_1x_2, $x_3x_4x_5$ terms, respectively.

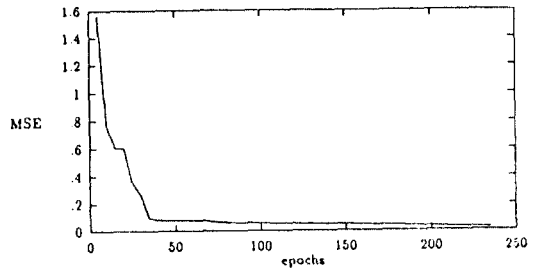


그림 4. 4차까지의 PSN을 이용한 RPN의 학습 곡선.
Fig. 4. The learning curve for the RPN of orders up to 4.

5.2. 비선형 다변수 연속 함수의 근사

이 실험에서는 주어진 다변수 연속 함수의 입력-출력 관계로부터 얻은 적은 수의 학습 데이터를 이용해 학습된 RPN이, 미지의 테스트 데이터에 대해 얼마나 잘 일반화 (generalization)할 수 있나를 살펴보는데 주안점을 둔다. 이를 위해 2차원 Gabor 함수를 택했다. Gabor 함수는 기본적으로 정현파에 의해 변조된 가우시안 함수이며, 포유류의 시신경 내 고도의 방향성을 갖는 단순 셀 수용체 (simple cell receptive fields)를 매우 정확히 근사한다고 알려져 있다[7]. 또한 Gabor 함수는 물리학에서 중요한 역할을 하는데 이는 이 함수가 공간-주파수 영역의 불확실성 척도의 곱이 가질 수 있는 lower bound를 만족하기 때문이다[7]. 이 실험에서 사용된 함수는 복소수 지수 함수 대신 코사인 함수를 이용하여 변조한 실수 2차원 함수를 택하였고, 다음 식과 같이 표현된다.

$$h(x_1, x_2) = \frac{1}{2\pi\lambda\sigma^2} \cdot e^{-\frac{(x_1/\lambda)^2 + x_2^2}{2\sigma^2}} \cdot \cos 2\pi(u_0 x_1 + v_0 x_2) \tag{19}$$

식 (19)에서 λ는 종횡비 (aspect ratio), σ는 스케일 지수 (scale factor), u_0, v_0는 변조 파라미터이며, 실험에서는 $\lambda = 1$, $\sigma = 0.5$, $u_0 = 1$, $v_0 = 1$로 정하여 사용하였다. 그림 5 (a)는 실험에 쓰인 $[-0.5, 0.5]^2$에 정의된 2차원 Gabor 함수를 보여 준다. 입력 데이터로는 $[-0.5, 0.5]^2$상에 등간격의 16×16 grid를 취하고 이들 256개 입력으로부터 랜덤하게 128개를 골라 학습 데이터로 하고, 나머지 128개는 테스트 데이터로 하였다. RPN의 출력 뉴론 특성 함수 $\sigma(x) \equiv \tanh(x) \in (-1, 1)$로 하여 좀 더 매끄러운 일반화를 꾀하였으며, 연결 강도들은 $[-0.5, 0.5]$내의 값을 랜덤하게 취하여 초기화 하였다. 또한 학습 알고리즘에 사용되는 파라미터 η, r, δ_η, δ_r들은 0.4, 0.0001, 0.6, 0.1로 각각 초기화 되었다.

학습은 초기에 4차 PSN을 가지고 시작되어, 5차, 6차 PSN을 각각 76번째, 613번째 epoch에 추가하며 진행되었고, 학습 epoch 수의 임계값 t_{th} = 10,000에 대하여 마지막으로 8차 PSN까지 사용하였다. 이때 최종 평균 자승 오차는 학습 및 테스트 데이터에 대해 각각 0.0003과 0.0004였다. 그림 5 (b)는 RPN의 학습 후 출력을 보여주며, 여기서 우리는 비교적 적은 수의 학습 데이터를 가지고 학습하였지만 RPN이 미지의 테스트 데이터에 대하여 매우 좋은 일반화를 얻었음을 알 수 있다. 그림 6은 1,000 epoch까지의 학습 곡선을 나타내며, 여기서 우리는 안정적인 학습이 이루어졌고, 고차 PSN를 더하여 감에 따라 (예를 들어, 76번째와 613번째 epoch) 평균 자승 오차가 급격히 줄어들음 알 수 있다.

비교를 위해 6차의 HPU를 이용하여 실험하였다. 이때 HPU 내 각 항들의 계수는 RPN의 경우와 유사하게 랜덤한 값으로 초기화 되었고 LMS 알고리즘을 이용하여 적응적으로 개선되었다. 학습은 0.1부터 1.0까지의 여러 가지 학습률에 대해 시도되었으나 모든 경우에 있어 5000 epoch 까지의 학습에도 평균 자승 오차가 학습 및 테스트 데이터 모두에 대해 0.01 이상의 큰 값을 갖고 더 이하의 값으로의 수렴이 어려운 결과를 보였다.

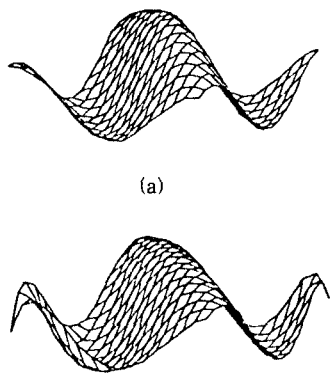


그림 5. (a) 실험에 사용된 2차원 Gabor 함수, (b) 학습 후의 RPN 출력.
Fig. 5. (a) 2D Gabor function to be approximated, (b) Output of the RPN.

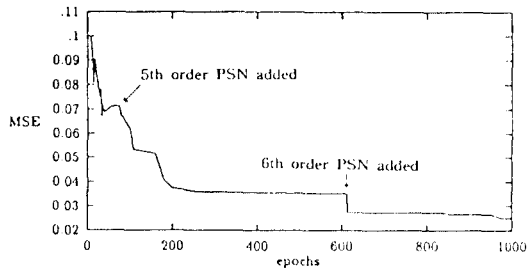


그림 6. Gabor 함수 근사를 위한 RPN의 1,000 epoch까지의 학습 곡선. 4차 PSN으로 시작되어, 5차, 6차 PSN이 각각 76번째, 613번째 epoch에 추가되었다.

Fig. 6. The learning curve up to 1,000 epochs for the approximation of Gabor function using the RPN. Starting with a 4th order PSN, the learning algorithm added the 5th and 6th order PSNs at the 76th and 613th epoch, respectively.

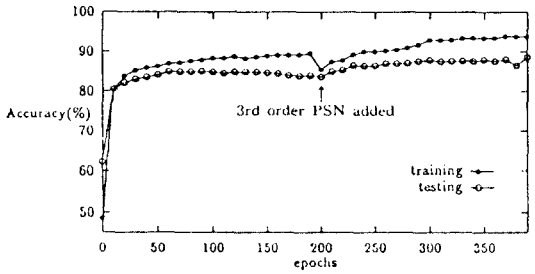


그림 7. RPN을 이용한 해저 수동 소나 신호 분류의 학습에 따른 정확도 변화.

Fig. 7. Classification accuracy for oceanic passive sonar signals using the RPN.

5.3. 해저 수동 소나 신호의 분류

이 실험에 사용된 데이터로는, 어류, 해저 지형의 균열, 배경 잡음 등과 같은 해저 내 자연 음원으로부터 발생된 지속 시간이 짧은 수동 소나 신호 (passive sonar signal)로부터 추출된 25차원 특징 벡터를 입력으로 하고, 각 신호가 속한 6개중 하나의 클래스 소속 정보를 출력으로 한다. 특징 벡터 추출을 위한 전처리 과정은 각 클래스 내의 신호들 사이의 차이는 최소화 하면서, 다른 클래스 내의 신호들과의 차이는 최대화할 수 있어야 하며, 본 실험에서 사용된 25차원 특징 벡터는 Gabor wavelet 계수[6] (16개), 신호의 길이 (1개), 그리고 시간 및 주파수 특성 (8개)으로 구성되었다. 6개 클래스가 가지고 있는 학습 데이터의 수는 각각 116, 116, 78, 116, 148, 116이고 (총 690), 테스트 데이터의 수는 각각 284, 175, 39, 129, 251, 127이다 (총 1005). 학습 알고리즘에 사용된 파라미터 η, r, δ_η, δ_r들은 0.1, 0.0001, 0.8, 0.1로 각각 초기화 되었고, RPN 내의 연결 강도들은 앞 절의 실험에서와 유사하게 [-1,1] 사이의 값들을 랜덤하게 취하여 초기화 되었다. 입력 벡터는 25차원으로 위에서 살펴 본 25개의 다른 특징들로 구성되고, 출력 벡터는 6차원으로 입력 벡터가 속하는 클래스에 해당하는 뉴론의 목표 출력은 1이고, 나머지는 0이다. 출력 뉴론 특성 함수로는 sigmoid 함수 $\sigma(x)\equiv 1/(1+e^{x})\in(0,1)$를 사용하였다.

점진적 학습 알고리즘을 이용한 학습은 처음 2차 PSN으로부터 시작하여, 200번째 epoch에서 3차 PSN을 더하였고, 최종적으로 390번째 epoch에서 종료하였다. 이때 학습 데이터에 대한 패턴 분류 정확도는 93.8% (647/690)이었고, 테스트 데이터에 대한 패턴 분류 정확도는 88.6% (890/1005)이었다. 다음의 그림 7은 학습의 진행에 따른 학습 및 테스트 데이터에 대한 패턴 분류 정확도를 나타낸다. 여기서 우리가 알 수 있는 것은, 200번째 epoch에서 3차 PSN을 추가함으로써 학습 데이터에 대한 성능은 일시적으로 감소하나 곧바로 크게 향상됨을 알 수 있고, 그때까지 포화되었던 테스트 데이터 성능 특성도 크게 향상되었음을 알 수 있다.

다음의 표 2에서는 테스트 데이터에 대해서 RPN의 결과와 다른 신경회로망 모델의 결과를 비교하였다. 비교 모델로는 하나의 중간층을 갖고 backpropagation 학습 알고리즘을 이용한 일반적인 MLP[25], MLP를 기반으로 한 소멸형 동적 구조 방법인 optimal brain damage (OBD)[23], 그리고 역시 MLP를 기반으로 하나 점진적 동적 구조 방법인 cascade-correlation 알고리즘[9]을 사용하였다. 이 결과에서 OBD의 경우 모델 내 연결 강도의 제거를 위한 "saliency"[23]의 계산을 매 10 epoch 마다 수행하였고, cascade-correlation 알고리즘의 중간층 뉴론의 수는 8번 반복 실험의 평균이다.

표 2. 해저 수동 소나 신호의 테스트 데이터에 대한 RPN, 일반적 MLP, OBD, cascade-correlation 알고리즘의 분류 정확도 및 계산량의 비교.
Table 2. Comparison of test set classification accuracy and computational complexity for oceanic passive sonar signals using the RPN, standard MLP, OBD and cascade-correlation algorithm.

신경회로망 모델	학습 epoch 수	분류 정확도	곱셈수($\times 10^6$)
RPN (2차)	50	84.1%	31.9
RPN (2차)	100	84.8%	63.8
RPN (3차)	200	93.6%	191.7
RPN (3차)	300	87.8%	319.6
RPN (3차)	390	88.6%	434.7
MLP (10 중간층 뉴론)	300	84.0%	333.3
MLP (20 중간층 뉴론)	300	89.2%	666.6
MLP (40 중간층 뉴론)	300	90.5%	1333.1
MLP (80 중간층 뉴론)	300	91.0%	2666.2
OBD (10 중간층 뉴론)	300	86.2%	401.2
OBD (20 중간층 뉴론)	300	89.3%	802.3
OBD (40 중간층 뉴론)	300	90.1%	1604.7
OBD (80 중간층 뉴론)	300	98.7%	3209.3
Cascade-correlation			
(23.2 중간층 뉴론)	300	88.3%	221.2
(27.4 중간층 뉴론)	300	56.9%	267.6

표 2의 타 모델에 의한 결과는 반복적인 실험을 통해 얻어진 최상의 결과를 나타낸다. 타 신경회로망 모델들을 위해 sigmoid 함수가 뉴론 특성 함수로 사용되었다. 학습률은 초기 20 epoch 까지는 0.2, 그 후에는 0.1을 사용하였고, 모멘텀 지수 (momentum factor)[25]는 0.9였으며, 학습 전에 각 신경회로망 모델 내의 연결 강도들은 -1에서 1 사이의 실수값으로 랜덤하게 선택되어 초기화 되었다. 학습이 끝난 후 RPN의 경우와 유사하게 이들 모델들은 학습 데이터에 대해 90 - 100%의 좋은 패턴 분류 정확도를 보였다. 표 1에서는 또한 최종 성능을 얻는데 필요한 학습 알고리즘의 계산량 (곱셈수)도 함께 고려하므로써, RPN과 타 모델의 성능 및 학습 속도를 객관적으로 비교하였다. 일반적으로 이 계산량은 각 모델들마다 공통적으로 사용하는 비선형 뉴론 특성 함수의 계산을 고려하지 않을 경우, 최종 종료까지 소요된 학습 알고리즘의 epoch 수와 각 학습 epoch에 필요한 계산량의 곱으로 나타내어 질 수 있다. 여기서, 각 epoch은 학습 데이터 하나에 대한 (i) 모델 출력값 계산을 위한 전방향 단계 (forward phase)와 (ii) 연결 강도 개선을 위한 역방향 단계 (backward phase)를 모든 학습 데이터에 반복함으로써 구성된다. 이를 통해 각 모델들이 다른 컴퓨터에 구현되었을 경우에도 모델 자체의 계산량을 나타내어 타 모델과의 객관적 비교가 가능해진다.

표 2에서 알 수 있듯이 테스트 데이터에 대한 RPN의 패턴 분류 정확도는 타 비교 신경회로망 모델들과 비슷한 결과이나, 일반적인 MLP나 초기에 많은 뉴론을 사용하고 학습을 통해 제거해 나가는 OBD보다는 상당히 적은 계산량을 가짐을 알 수 있다 (유사한 성능에 대해 MLP에 비교해서는 1/10에서 1/1.5, OBD에 비교해서는 1/8에서 1/2 가량의 계산량만 필요함). 이러한 결과 이외에 중요한 점은, MLP나 OBD의 경우 학습 전 반드시 중간층 뉴론의 수로 표시되는 모델의 구조를 정하여 주어야 하므로 실제 학습에 있어서 만약 하나의 모델을 통해 만족할 만한 결과를 얻지 못하면 다른 구조를 갖는 새로운 모델을 학습하여야 하며, 이는 바로 수치로서 환산되지 않은 별도의 실제적인 학습 시간 연장을 의미하게 된다. 하지만 RPN의 경우에 있어서는 학습 전에 모델의 구조를 정할 필요가 없고, 최적의 모델 구조는 학습이 진행됨에 따라 자동적으로 결정된다. 이러한 MLP나 OBD의 성질을 고려하면, RPN과의 실제 계산량 차이는 더욱 커질 수 있음을 알 수 있다. MLP에 기반을 둔 점진적 학습 알고리즘의 하나인 cascade-correlation은 RPN과 마찬가지로 이런 문제를 피할

수 있으며, 표 1에서 알 수 있듯이 적은 계산량을 가짐을 알 수 있다. 하지만 모의 실험 결과 이 모델은 학습 데이터에 대해서는 우수한 성능을 보이나 (예를 들어 27.4 개의 평균 중간층 뉴론을 사용할 경우 학습 데이터에 대한 분류 정확도는 100%였다), 테스트 데이터에 대해서는 이러한 성능에 비해 상대적으로 좋지 않은 결과를 보이며, 이로부터 우리는 이 문제에 대해서는 cascade-correlation 알고리즘이 좋지 않은 일반화를 얻었다고 생각된다. 이 모델의 다른 단점으로 모델 내의 뉴론 생성을 위한 척도의 변화에 따라 성능의 기복이 크다는 점을 또한 알 수 있었다.

Ⅵ. 결 론

본 논문에서 살펴 본 ridge polynomial network (RPN)은 일반적 다변수 다항식에 대한 릿지 다항식의 표현 정리[5]와 본 필자가 제안한 pi-sigma network (PSN)[13]을 일반화한 고차 신경회로망 모델이다. RPN은 입력 변수와 연결 강도의 내적을 새로운 입력 변수로 하는 릿지 다항식을 사용함으로써, multi-layered perceptron (MLP)의 경우와 유사하게 통계학에서 사용되고 있는 projection pursuit regression[11],[17]과 형태적으로 밀접한 관련을 가짐을 알 수 있으나, 적응적, 점진적, 지도 학습 알고리즘을 사용한다는 점에서 큰 차이를 보인다.

비선형 함수에 대한 RPN의 근사 정리를 통해 이 모델이 수학적으로 우수한 근사 능력을 가지고 있음을 보였다. 또한 다변수 연속 함수의 근사와 실제 패턴 분류 문제에의 응용을 통해, 본 논문에서 살펴 본 RPN을 위한 점진적 학습 알고리즘이 좋은 일반화 능력을 가지고 있음을 실험적으로 보였다. 이는 학습 중에 점점 고차의 PSN을 더하여 나감으로써 주어진 문제 내의 고차 변동 성분을 점진적으로 표현할 수 있었기 때문이다. PSN을 기본 구조로 이용한 점진적 학습 알고리즘을 통해서 RPN은 적은 계산량과 모델 복잡도, 그리고 규칙적인 구조를 유지하면서 타 모델들과 비슷하거나 우수한 성능을 보였다.

앞으로 실제 점진적 학습 알고리즘을 통해서 얻어진 구조가 처음부터 고정된 RPN 구조 (그림 2)와 비슷하거나 같은 근사 능력을 갖는지에 대한 수학적 고찰이 큰 과제로 남아 있다. 또한, 보다 많은 응용 문제에 RPN을 적용함으로써 이 모델의 장, 단점을 분석, 종합하는 것도 중요한 향후 과제라 생각된다.

참고문헌

1. D. H. Ackley, G. E. Hinton and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, Vol. 9, pp.147-169, 1985.
2. H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Auto. Contr.*, Vol. 19, pp.716-723, 1974.
3. T. Ash, "Dynamic node creation in backpropagation networks," Tech. Rep. #ICS-8901, Institute for Cognitive Science, Univ. of California at San Diego, February 1989.
4. S. Chong, S.-Q. Li and J. Ghosh, "Predictive dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM," *IEEE Jour. Sel. Areas in Commun.*, Vol. 13, pp.12-23, Jan. 1995.
5. C. K. Chui and X. Li, "Realization of neural networks with one hidden layer," Tech. Rep. #CAT-244, Center for Approximation Theory, Department of Mathematics, Texas A&M Univ., March 1991.
6. J. M. Combes, A. Grossman and Ph. Tchamitchian (Eds.), *Wavelets: Time-Frequency Methods and Phase Space*, Springer-Verlag, 1989.
7. J. G. Daugman, "Entropy reduction and decorrelation in visual coding by oriented neural receptive fields," *IEEE Trans. Biomed. Eng.*, Vol. 36, pp.107-114, 1989.
8. R. Durbin and D. E. Rumelhart, "Product units: a computationally powerful and biologically plausible extension to backpropagation networks," *Neural Computation*, Vol. 1, pp.133-142, 1989.
9. S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances*

in Neural Info. Proc. Syst. 2, D. S. Touretzky (Ed.), Morgan-Kaufmann, pp.525-532, 1990.

10. J. A. Feldman and D. H. Ballard, "Connectionist models and their properties," *Cognitive Science*, Vol. 6, pp.205-254, 1982.
11. J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Jour. Amer. Stat. Assoc.*, Vol. 76, pp.81-823, 1981.
12. S. Geman, E. Bienenstock and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, Vol. 4, pp.1-58, 1992.
13. J. Ghosh and Y. Shin, "Efficient higher-order neural networks for classification and function approximation," *Int'l Jour. Neural Syst.*, Vol. 3, pp.323-350, 1992.
14. C. L. Giles and T. Maxwell, "Learning, invariance and generalization in a high-order neural network," Applied Optics, Vol. 26, pp.4972-4978, 1987.
15. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational capabilities," *Proc. Int'l Acad. Sci. USA*, Vol. 79, pp.2554-2558, 1982.
16. K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks, Vol. 2, pp.359-366, 1989.
17. P. J. Huber, "Projection pursuit," *The Annals of Stat.*, Vol. 12, pp.435-475, 1985.
18. A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE Trans. Syst., Man and Cyber.*, Vol. 1, pp.364-378, 1971.
19. K. Iwata, J. Ghosh and Y. Shin, "Time optimal control using pi-sigma networks," *Proc. IEEE Int'l Conf. Neural Networks*, Vol. 1, pp.545-551, San Francisco, California, March 1993.
20. S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi, "Optimization by simulated annealing," *Science*, Vol.220, pp.671-680, 1983.
21. M. J. Korenberg and L. D. Paarmann, "Orthogonal approaches to time series analysis and system identification," *IEEE Signal Proc. Mag.*, pp.29-43, July 1991.
22. A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Info. Proc. Syst.* 4, S. J. Hanson, J. E. Moody and R. P. Lippmann (Eds.), Morgan-Kaufmann, pp.950-957, 1992.
23. Y. LeCun, J. S. Denker and S. A. Solla, "Optimal brain damage," in *Advances in Neural Info. Proc. Syst.* 2, D. S. Touretzky (Ed.), Morgan-Kaufmann, pp.598-605, 1990.
24. V. J. Mattews, "Adaptive polynomial filters," IEEE Signal Proc. Mag., pp.10-26, July 1991.
25. J. McClelland and D. Rumelhart, *Parallel Distributed Processing*, Vol. 1, The MIT Press, 1987.
26. M. Minsky and S. Papert, Perceptrons: *An Introduction to Computational Geometry*, The MIT Press, 1969.
27. Y. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, 1989.
28. J. Park and I. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, Vol. 3, pp.246-257, 1991.
29. J. C. Platt, "Learning by combining memorization and gradient descent," in *Advances in Neural Info. Proc. Syst.* 3, D. S. Touretzky (Ed.), Morgan-Kaufmann, pp.714-20, 1991.
30. W. Rudin, *Principles of Mathematical Analysis*, 3rd Ed., McGraw-Hill, 1976.
31. T. D. Sanger, "A tree-structured adaptive network for function approximation in high-dimensional spaces," *IEEE Trans. Neural Networks*, Vol. 2, pp.285-293, 1991.
32. M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*, Wiley, 1980.
33. Y. Shin and J. Ghosh, "Realization of Boolean functions using binary pi-sigma networks," in *Intell. Eng. Syst. through Artificial Neural*

Networks, C. H. Dagli, R. T. Kumara and Y. C. Shin (Eds.), AMSE Press, pp.205-210, 1991.

34. Y. Shin and J. Ghosh, "Ridge polynomial networks," *IEEE Trans. Neural Networks*, Vol. 6, pp.610-622, May 1995.
35. Y. Shin, J. Ghosh and D. Samani, "Computationally efficient invariant pattern classification with higher-order pi-sigma networks," in *Intell. Eng. Syst. through Artificial Neural Networks 2*, C. H. Dagli, L. I. Burke and Y. C. Shin (Eds.), AMSE Press, pp.39-384, 1992.
36. H. B. D. Sorensen and U. Hartmann, "Pi-sigma and hidden control based self-structuring models for text-independent speaker recognition," *Proc. Int'l Conf. Acoustics, Speech and Signal Proc.*, Vol. 1, pp.537-540, Minneapolis, Minnesota, April 1994.
37. M. F. Tenerio and W. Lee, "Self-organizing network for optimum supervised learning," *IEEE Trans. Neural Networks*, Vol. 1, pp.100-110, 1990.
38. B. Widrow and M. E. Hoff, "Adaptive switching circuits," *IRE WESCON Convention Record*, pp.96-104, 1960.
39. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, 1985.
40. X. L. Zhong, J. M. Lewis and H. Rea, "Neuro-accuracy compensator for industrial robots," *Proc. IEEE Int'l Conf. Neural Networks*, Vol. 5, pp.2797-2802, Orlando, Florida, July 1994.

辛 堯 安(Yoan Shin) 정회원

1965년 1월 19일 생
1987년 2월 : 서울대학교 전자공학과(학사)
1989년 2월 : 서울대학교 대학원 전자공학과(석사)
1992년 12월 : University of Texas at Austin 전기 및 컴퓨터 공학과(박사)
1992년 12월~1994년 7월 : MCC(Microelectronics & Computer Technology Corp.)콘소시엄 연구원
1994년 9월~현재 : 숭실대학교 전자공학과 전임강사
※주관심 분야 : 디지털 통신 시스템, ATM 트래픽 제어, 통신용 등화기 설계, 신경망 이론 및 응용