조선공학에서의 뉴럴네트워크 적용사례


유 원 선* 양 영 순**

1. Neural Net Basics

신경망이 무엇인가를 한마디로 정의하기란 쉽지 않다. 그것은 이 분야가 대부분의 사람이 생각하는 것과 같이 근래에 시작된 것이 아니라 인공지능이라는 용어가 만들어지면서부터 세계각처의 여러 분야에 있는 연구자들에 의하여 제각기 연구되어 왔기 때문이다. 즉, 연구자에 따라서는 같은 개념에 대해서도 서로 자기만의 용어를 사용하여 여러 가지 다른 이름을 붙여 왔다. 하지만 인간의 신경조직을 모델로, 간단한 기능을 가지는 단위 처리기들 사이를 서로 대규모로 연결시키는 네트워크 형태의 구조라는 정의에는 대부분 동의할 수 있을 것이다.

신경망은 적당한 가중치가 매겨진 입력을 합해서 출력하는 정도의 간단한 기능을 하는 처리기 여러 개가 상호 병렬적으로 연결되어 문제를 처리한다. 그 결과 역시 순간적으로 진행되어 특정장소에 저장되기보다는 주어진 입력을 통해 전체 네트워크가 어떤 평형상태에 도달하는 방식으로 얻어진다. 또한 신경망에서의 정보는 특정 장소에 저장되지 않는다. 이것은 기억장소의 특정 번지를 참조하여 변수 x의 현재 값을 알아볼 수 없다는 것을 의미한다. 정보는 각 처리기 사이의 연결 구조와 그것이 갖고 있는 값에 의해 결정된다.

여기서 기존의 계산 기법과 신경망 기법의 문제해결 방법의 차이점을 살펴보면 다음과 같다. 어떠한 문제에 직면하면 기존에는 문제를 이해하고 그에 따른 흐름도(flow chart)를 작성하여 프로그램을 코딩하려 할 것이다. 그러나 신경망적 접근방법은 전혀 다르다. 전산화를 의뢰 받은 사람은 가능한 한 많은 실제 데이터를 수집하고 신경망에 이를 입력하여 신경망으로 하여금, 소위 학습이라는 것을 통해서 그 데이터 사이의 일정한 규칙에 의하여 새로운 데이터가 입력되었을 때 적절한 결정을 할 수 있도록 할 것이다. 이때 기존의 계산에

* 서울대학교, 조선해양공학과 박사과정
** 정회원 · 서울대학교 조선해양공학과 교수

서 프로그래밍에 해당하는 과정이 신경망에서는 입력시킨 데이터 사이의 관계에 의해서 자체적으로 수행됨을 알 수 있다. 이것은 마치 인간이 새로운 사실을 배우는 과정과도 매우 유사한 것이다. 일반적으로 계산이라는 것을 주어진 입력에 대해 적당한 출력을 만들어 내는 블랙박스라고 할 때 이 두 가지 방법 사이의 차이는 그림 1로 나타낼 수 있다.

기존의 기법은 입력에 해당하는 올바른 출력을 만들어 내기 위하여 프로그램 언어의 기본 구조인 순차구조, 선택구조, 반복구조 등을 적절히 사용하여 순서적인 절차, 즉 알고리즘을 만든다. 반면에 신경망 기법은 단순한 기능을 하는 처리기를 상호 연결시켜 놓고, 이들 사이의 연결 강도를 조정하여 입출력의 관계를 나타내도록 하는 것이다. 결과적으로 신경망 기법에서의 핵심은 주어진 문

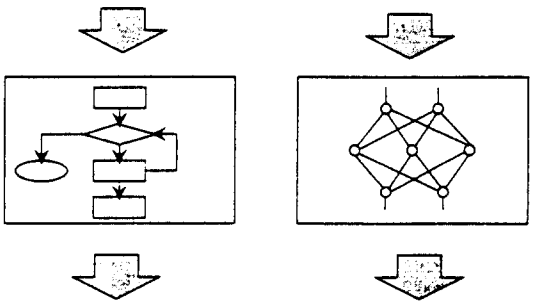

Fig. 1 Comparison of neural network & traditional algorithms

제를 풀 수 있도록 하는 노드들 사이의 연결 강도 값을 어떻게 구할 수 있느냐 하는 것인데, 이를 학습이라 한다.

2. Structure of Neural Net

신경망의 구조를 알기 위해서는 필연적으로 인간의 뇌세포가 어떻게 이루어 졌는가를 알아야 한다. 왜냐하면 신경망 연구는 기본적으로 인간의 두뇌 메카니즘을 구현하는 것을 그 목표로 하고 있기 때문이다. 그러므로 뇌세포와 그의 구성에 대하여 간단히 살펴보고, 이를 신경망에서 어떻게 구현하였는지 알아보자. 신경조직의 기본적인 구성요소는 뉴런(neuron)이다. 뉴런의 구성은 위 그림과 같은데, 인간의 두뇌 속에는 이러한 뉴런이 10^{10}에서 10^{11}개까지 존재하여 연산의 기본단위를 이룬다. 뉴런은 기본적으로 몸체에 해당하는 세포체(soma)와 다른 뉴런과 연결되는 축색돌기(axon)와 수상돌기(dendrite)라는 섬유로 되어 있다. 뉴런 사이의 연결은 몸체 부분이나 수상돌기 연장부분에서 일어난다. 이러한 연결 부분을 천에서 만개의 시냅스를 가지고 동수의 다른 뉴런과 연결되어 신호를 보낸다고 한다. 각각의 뉴런은 이렇게 보내진 신호를 입력으로 하여 그것이 적당한 수준의 임계치를 초과할 때만 출력을 생성하는 임계 처리기라고 볼 수 있다.

뉴런의 처리속도는 현재의 컴퓨터의 처리기보다 훨씬 느리다는 특성이 있다. 그럼에도 불구하고 인지적 철, 기억장소 조회, 언어처리, 추론 등의 복잡한 일을 빠른 시간 내에 해결하는 것은 매우 놀라운 일이다. 이 비밀의 열쇠는 바로 각 뉴런 사이의 고도의 병렬성에 있다.

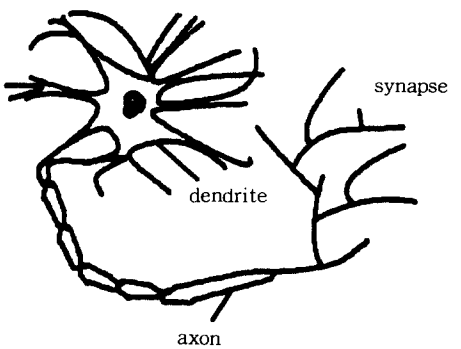

Fig. 2 Shape of neuron

뇌에서 뉴런간의 연결선의 수는 상대적으로 고정되어 있고, 연결의 생성과 제거가 없는 상태에서, 새로운 지식은 연결선의 세기를 변화시킴으로

써 습득된다고 알려지고 있다. 그리고 뇌는 잡음에 민감하지 않는 뇌의 일부, 즉 일부의 뉴런이 파괴되더라도 뇌 전체의 성능은 거의 저하되지 않는다. 또한 정보를 저장하는 방식도 하나의 사실이 특정한 뉴런에 저장되는 것이 아니고 여러 뉴런에 걸쳐서 저장되는 지식의 분산 표현방식을 사용함을 알 수 있다. 기존의 컴퓨터가 기억장소의 각 부분에 특정한 정보를 기억하고 있어서 저장 장소의 일부분이 손상되면 그에 해당하는 정보를 잃게 되어 전체 시스템이 제대로 작동하지 않는 등 고장에 약한 면을 보이는데 비해 이것은 매우 바람직한 특성이라 할 수 있다. 또 뇌는 작업 수행 전체를 제어하는 부분을 따로 가지고 있지 않아서 각각의 뉴런이 자기의 입력에 근거하여 출력을 생성하는 특성이 있다. 이는 컴퓨터가 중앙 제어장치를 사용하여 제어하는 것과는 대조적으로 분산제어를 함을 알 수 있다.

2.1 Component & Action of Neural Network

신경망에서 중요한 구성요소는 처리기(processing element)와 이들 상호간의 연결(interconnection)이다. 뇌의 신경세포에 해당하는 처리기의 구성은 옆 그림과 같다. 한 처리기에 연결된 다른 처리기들로부터의 입력을, 연결선의 가중치를 고려하여 더한 후 그 결과를 적당한 전이함수(transfer function)로 처리한 후 연결 된 다른 처리기로 출력한다. 이를 좀 더 구체적으로 설명하면 다음과 같다. 하나의 처리기가 다른 처리기의 출력을 그 입력 x_i로 받아 해당하는 연결 강도 w_{ij}에 대한 각 입력이 가중치의 합 net_j을 다음과 같이 구한다.

$$net_j = Sum_i(w_{ij}\ x_i)$$

이 합을 다음과 같은 전이함수에 적용시켜서 얻은 결과를 그 처리기의 출력으로 하여 이와 연결되어 있는 다른 처리기로 보낸다.

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

여기에서 θ_j는 j번째 처리기의 고유 바이어스이다. 이 처리기에서 중요한 구실을 하는 것은 임계치 기능으로서 이러한 전이함수는 그 특성에 따라 여러 가지가 쓰이는데 이중에서 가장 많이 사용하는 것은 다음과 같은 sigmoid 함수이다.

$$f(x) = \frac{1}{(1+e^{-x})}$$

처리기들간의 연결 또한 매우 중요한 구성요소로서 이 연결의 가중치가 어떻게 되어 있고, 어떻게 조정되느냐에 따라서 전체 신경망의 특성이 결정된다. 본래의 신경조직은 거의 임의로 연결되어 있지만, 대부분의 신경망은 처리기들을 계층적 구조로 상호 연결한다. 입력 데이터는 외부에서 입력층(input layer)을 통해서 들어오고 출력은 출력층(output layer)을 통해서 외부로 나간다. 입력층과 출력층 사이에는 중간층(hidden layer)들이 존재할 수 있다. 연결은 입력층에서 출력층 방향으로 되어 정보가 입력에서 출력으로만 흐르게 되어있는 계층적 방향 그래프로 볼 수 있다. 정보는 각 처리기 사이의 연결선의 가중치를 조정함으로써 정장되거나 학습된다.

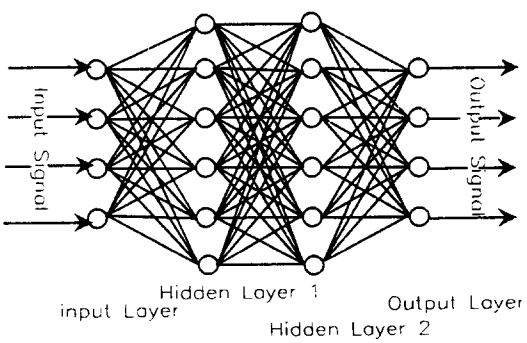

Fig. 3 Structure of feed-forward neural network

어떠한 특별한 문제를 풀기 위하여 신경망을 어떻게 구성하고 어떤 전이함수를 쓰며 어떻게 가중치를 변화시킬 것인가를 결정하는 소프트웨어를

netware라고 부른다. Netware프로그래머는 각 처리기가 수행할 알고리즘을 명시할 필요가 없다. 망의 상호 연결, 전이함수, 그리고 망의 연결 가중치 조정법칙 등만 명시하면 된다. 따라서 신경망에서는 프로그램을 수행한다 라는 말 대신에 반응한다(react), 학습한다(lean), 자체 조정한다(self organize) 등의 용어를 사용한다.

2.2 Learning in Neural Networks

신경망에서 학습의 종류는 기준에 따라서 여러 가지로 분류될 수 있다. 가장 일반적인 것으로는 지도 학습(supervised learning)과 자율학습(supervised learning)이 있다. 지도학습은 학습 중에 주어진 입력에 대하여 올바른 출력이 어떤 것이어야 하는 지를 제공해 주는 학습법이다. 이 경우에 신경망은 자신의 출력이 올바른지를 결정할 수 있기 때문에 자신의 연결강도를 조정하는데 특정 학습법칙을 어떻게 적용해야 할지를 결정하기가 쉽다. 반면에 자율학습은 주어진 입력에 대하여 올바른 출력이 무엇인지가 제공되지 않으므로 어떠한 출력을 내야 할지가 정확하지 않다. 본 보고서에는 주로 지도 학습 방법에 대하서 다룬다. 지도학습의 기본적인 학습법인 최소 평균제곱법(LMS: least mean square) 학습법을 위주로 신경망의 학습을 살펴본다.

2.2.1 LMS learning law

LMS 학습법칙은 1960년에 미국 스탠포드 대학의 Bernard Widrow와 Ted Hoff에 의해서 개발된 학습방법으로 가장 많이 사용되고 있다. 일명 Delta rule으로 불리는데, 창시자의 이름을 빌려서 Widrow-Hoff rule이라고도 한다.(참고로 Widrow는 국제 신경망 학회의 회장이다).

이 법칙의 작동과정은 다음과 같다. 먼저 각 처리기는 주어진 입력에 대하여 현재의 연결강도를 이용하여 자신의 출력값 y를 계산한다. 그 후에 이 출력값을 올바른 출력을 나타내는 입력d와 비교하여 오차 값 E(=d−y)를 계산한다. 만일 이 값이 크다면 현재의 연결강도의 값에 이 오차값만큼을 보정하여 새로운 연결강도를 구한다. 즉 현재의 연결강도를 w(n)이라 하고, 새로운 연결강도를 w(n+1)이라고 하면, 이 둘 사이에는

$$w(n+1)-w(n) = \frac{\alpha EX}{|x|^2}$$

의 관계가 있다.

따라서 새로운 연결강도 w(n+1)는 위의 식에 의하여 구해진다. 이상이 바로 유명한 LMS 학습법칙의 식이다. 이때 X는 입력벡터이고 α는 learning rate라고 불리는 상수이다.

이제 처음 입력에 대해서 학습되었다고 하자. 하지만 그 다음 입력 데이터에 대하서 올바로 작동하도록 다시 LMS 법칙으로 학습시키고 나면 연결강도의 값이 또 변경되어 이전 것을 잊어버리는 수가 있다. 따라서 이전에 학습했던 것을 다시 반복하여 학습할 필요가 있다. 그렇다면 이러한 연결강도의 변경은 언제까지 계속 반복돼야 할까? 학습이 끝나는 것은 학습데이터가 모두 올바른 출력을 내어 신경망의 오차가 매우 적어질 때이다. 이로써 우리는 각 입력의 복잡한 조건을 하나하나 나열하지 않고도 단지 학습시킬 데이터의 입출력 쌍만을 반복하여 보여줌으로써 주어진 입력과 해당 출력사이의 관계를 자동적으로 알아낼 수 있게 되었다. 하지만 신경망을 사용했을 때의 장점은 이것으로만 끝나지 않는다. 만일 어떤 입력이 학습 중에 사용한 데이터와 꼭 같니 않은 경우에도 학습 중에 배운 사실을 이용하여 어느 정도 적절한 출력을 낼 수 있다는 장점을 또한 가진다. 신경망은 학습된 것을 기본으로 하여 가장 근접한 결과를 출력할 수 있다. 이러한 기능을 신경망의 generalization capability라고 한다.

수학적으로 말하자면 LMS식은 네트워크내의 통계적 LMS 오차를 최소화하고자 하는 것이다. 이 경우에 각 처리기의 연결강도의 오차는 이상적인 연결강도 값에 기반을 두고 있다. 즉 학습할 때마다 현재의 연결강도에 의한 오차, 다시 말해서 이상적인 연결강도 값에서 얼마나 많이 떨어져 있는지를 계산하여 그 방향으로 연결강도의 값을 조정하는 것이다. 이와 같은 과정은 그래프를 사용하면 쉽게 이해될 수 있을 것이다. 네트워크에서의 오차, 즉 현재 연결강도에 의하여 출력된 값과

올바른 출력값 사이의 차이는 결국 연결강도의 함수로 나타낼 수 있는데, 2차원인 경우에는 오차가 연결강도 벡터의 이차함수로 나타나게 되므로 가능한 연결강도 벡터에 대한 평균제곱오차를 그려보면 하나의 포물선을 얻을 수 있다. 포물선의 특징중의 하나는 최소의 값을 갖는 지점이 있다는 것이다. 이점은 평균제곱오차가 최소가 되는 점을 나타내며, 우리가 구하려고 하는 연결강도의 값도 바로 이 최소 오차에 해당하는 연결강도이다. LMS법칙은 이 포물선의 임의의 지점에서부터 오차가 최소가 되는 점을 향하여 연결강도를 바꾸어 나가는 것이다. 이것은 포물선에서 미분값을 구하면 기울기를 얻을 수 있는데, 이의 반대방향(negative gradient)으로 움직여 가는 것을 의미한다. 기울기의 반대방향이 최소점에 이르는 최단 경로임은 수학적 사실로 증명되어 있다. 즉, 신경망의 학습이라는 것은 연결강도에 대하여 신경망의 오차를 나타낸 후 이 오차곡선의 임의의 초기지점에서 시작하여 경사도(gradient)를 구하고, 이의 반대방향으로 적절한 만큼씩 연결강도의 값을 조정하여 움직여 가면서 이 오차값이 최소가 되는 지점의 연결강도를 구하는 과정이라고 요약할 수 있다.

2.2.2 Back-Propagation(BP)

1) Necessity of BP

신경망의 한 노드의 기능은 입력으로 주어진 값들을 연결강도 값과 곱한 뒤 이를 모두 더해서 적절한 출력하는 것이다. 이를 수학적으로 분석해 보면 결국 입력 값들을 선형으로 분리하는 평면(이차원인 경우에는 직선)을 나타냄을 알 수 있다. 그러나 임의의 입력패턴들이 주어졌을 때 이를 평면으로 제대로 분할한다는 것은 거의 불가능하다. 이를 쉽게 해결할 수 있는 방법으로는 이러한 노드들을 여러 층으로 연결하여 나눌 수 있는 영역이 복잡할 때 이를 둘러싼 오목영역(concave region)으로 만드는 것이다. 이 경우에 문제가 되는 것은 신경망이 제대로 작동하지 않는다는 것을 알았을 때 어떤 노드와 연결된 연결강도의 값을 어떻게 조정할 것인가 이다. 즉 신경망이 단층일 경우에는 해당 출력노드와 연결되어 있는 연결강도의 값을 조정하면 되지만, 이것이 다층으로 확장되었을 경우에 출력노드와 직접 연결되지 않은 노드의 연결강도는 어떻게 조정할 것인가? 이것을 credit assignment problem 이라고 한다. BP는 이 문제에 대해서 신경망 내의 모든 노드와 연결강도가 어느 정도의 책임을 져야 한다는 식으로 문제를 해결했다. 이 때 오차에 대한 책임은 출력노드의 오차를 연결된 이전 노드로 역전파시킴으로써 할당된다. 이러한 과정은 입력노드에 이를 때까지 반복된다. 이쯤 되면 이 학습방법의 이름이 왜 Backpropagation인지 이해가 갈 것이다.

2) Basic Environment of BP

이제 BP의 기본원리를 소개하기에 앞서서 BP가 작동되는 다층신경망의 구조에 대하여 정리해 보자. 지금부터 BP학습법을 사용하여 학습되는 신경망을 BP네트워크라고 부른다. 이는 이제까지 소개한 신경망의 구조와 거의 같지만 한 두 가지 점에 있어서 짚고 넘어가야 할 필요가 있기 때문에 다시 정리해 본다.

먼저 기본 노드의 구조를 살펴 볼 때 BP네트워크를 구성하는 기본 단위라고 할 수 있는 노드를 보여준다. 일련의 입력값을 외부나 이전의 층에 있는 노드로부터 주어진다. 이 값들은 연결된 강도와 곱해져서 모두 더해진다. 이 값을 net라고 했을 때 이는 모든 노드에서 계산되어야 한다. 이 값

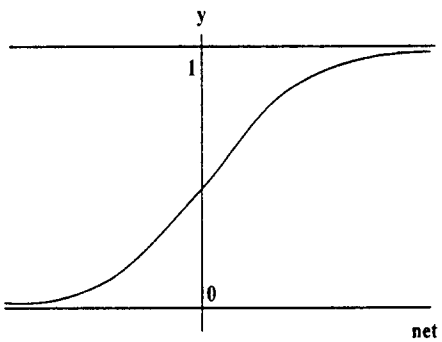

Fig. 4 Function of sigmoid

은 전이함수 f에 의하여 변경된 뒤 이 노드의 출력값을 생성하게 된다.

그림 4는 네트워크에서 사용되는 다음과 같은 전이함수를 보여준다.

$$y = \frac{1}{1+e^{-net}}$$

이 함수는 sigmoid라고 불리는데 그 출력이 단순히 0과 1사이의 값으로 사상된다는 것 이외에도 BP네트워크에 있어서 몇가지 바람직한 특성을 나타낸다.

첫째는 미분 가능하다는 것이다. 이것은 이제까지의 학습법칙에서는 따로 요구하지 않던 사항으로서 BP네트워크에는 필수 불가결하다. 또한 그 미분의 형태가 매우 간단하다는 점도 있다. 이 함수는 매우 흥미롭게도 기 미분 결과가 다음과 같다.

$$\frac{\delta y}{\delta net} = y(1-y)$$

이러한 내용이 흥미 있는 이유는 BP학습법을 사용하면 각 연결선에 대해서 전이 함수를 미분해야 하는데 그것이 복잡할 경우 매번 계산할 때 큰 노력을 필요로 하기 때문이다. 둘째로 비선형이라는 점이다. 만일 노드의 전이함수가 선형적이라면 어떻게 될까? 입력으로 들어온 값을 선형적으로 변환하여 출력한다면 다층의 신경망이 어떻게 작동되는가? 결국 다층구조가 쓸모 없게 되고 만다. 왜냐하면 선형식은 아무리 여러 개를 중복하여 사용해도 결국은 하나의 선형식으로 표현 가능한데, 이것은 마지막으로는 자동적인 gain 제어 기능이 있다는 점이다. 이것은 sigmoid함수가 입력값이 작으면 크게 변하고 클 때에는 작게 변하는 특성을 갖는다는 것을 의미하는데 매우 큰 입력값에 대해서 급격히 변화하는 것을 방지하고 미소한 입력값도 간과하지 않는 장점이 있다.

그림 5는 이러한 노드들이 연결된 간단한 BP 네트워크를 나타낸다. 이 경우에는 하나의 중간층과 출력층으로 이루어진 이층구조를 보여주고 있는데, 일반적으로는 임의의 개수로 연결되어도 상관없다. 또 연결선의 방향은 모두 입력에서 출력으로 단일 방향을 하고 있는데 BP학습법칙 자체에는 그에 대한 제한은 없다. 실제로는 보다 복잡한 응용분야, 예를 들어 음성인식 등에서는 반대방향의 연결선도 허용하는 확장된 BP 네트워크도 사용하기도 한다.

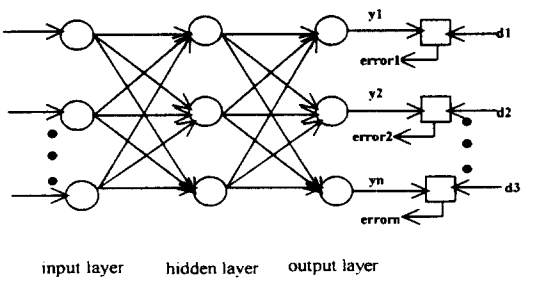

Fig. 5 Structure of BP network

3) Basic Principle of BP

신경망을 학습시킨다는 것은 주어진 문제의 입력들이 올바른 출력을 생성하도록 연결 강도의 값을 조정함을 뜻한다. BP학습법에서는 델타학습법칙과 마찬가지로 이러한 입력과 해당 출력의 쌍이 학습과정에서 주어짐을 가정하는데 이것이 지도학습을 한다는 것을 의미한다. 이때, 학습에서 사용하는 입출력 쌍을 학습패턴이라고 하고, 이 쌍들 전체를 학습패턴의 집합이라고 한다.

이제 바꾸어 말하면 학습은 모든 학습패턴의 집합에 대하여 올바른 출력을 내도록 신경망의 연결강도를 조정하는 것이라고 할 수 있다. 이때, 연결강도를 얼마나 변경시켜야 하는가를 결정하기 위해서는 각 노드의 오차값을 알아야 한다. 즉 j번째 출력노드의 출력값이 y_j라고 할 때 출력노드에서의 오차는

$$\frac{\delta E}{\delta y_j} = y_j - d_j$$

와 같이 구해질 수 있다. 여기에서 d_a는 j번째 출력노드의 올바른 출력값을 의미한다. 그렇다면 올바른 출력값을 알 수 없는 중간노드에서는 어떻게 값을 구할까?

그림 6은 i번째 중간노드의 오차값을 구하는 예를 보여준다. 이 그림에서 볼 수 있듯이 i번째 중간노드와 연결되어 있는 출력노드의 오차값을 연결된 연결강도와 곱해서 모두 곱한 값이 이 노드의 오차값이 됨을 알 수 있다. BP 학습법칙의 기본원리는 이러한 역전파 방식에 기초한다. 이 어라나 놀라운 발견인가? 이것은 마치 콜럼부스의 달걀과 필적할 수 있는 것이어서 결과만 놓고보면 그다지 새로운 것도 없지만 다층신경망의 학습법을 찾으려고 노력하던 사람들에겐 사막의 오아시스와도 같았을 것이다. BP학습법의 대략적인 구조는 다음과 같다.

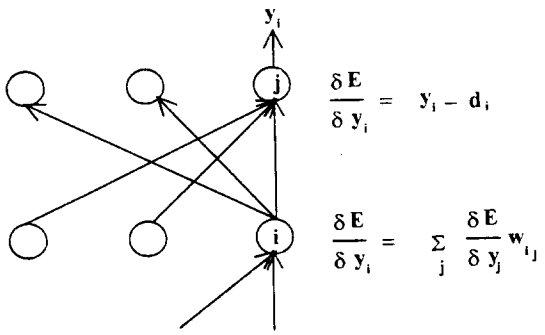

Fig. 6 Learning algorithm of BP

1. 학습패턴의 집합에서 다음에 학습될 패턴을 선택한 후, BP네트워크에 입력시킨다.
2. 이 네트워크의 출력값을 계산한다.
3. 이 값과 올바른 출력값을 계산한다.
4. 이 오차값을 최소화하도록 네트워크의 연결강도를 조정한다.
5. 전체 학습패턴의 오차가 매우 작아질 때까지 각 학습패턴에 대해서 1에서 4까지의 과정을 반복한다.

위에서 1단계와 2단계의 작동과정은 최종적으로 학습된 신경망의 작동과 유사하다. 즉 새로운 패턴이 입력되면 이를 이용하여 신경망의 출력값을 계산한다. 이때 모든 계산과정은 층별로 이루어진다. 위의 그림에서 먼저 i 번째 층의 노드들에 대해서 그 출력값을 계산하고 이를 입력으로 하여 j번째 층의 노드들이 출력값을 계산한다. 3단계에서 각 출력노드의 출력값이 올바른 출력값과 얼마나 차이가 나는지 계산한다. 이 값을 이용하여 4단계에서 연결강도의 값을 조정한다. 이러한 과정을 충분히 여러 번 반복하여 신경망의 실제 출력값과 올바른 출력값 사이의 오차가 적당한 임계치보다 작아지면 이 신경망은 학습되었다고 한다. 이제 더 이상 연결강도의 값을 변경시킬 필요 없이 곧바로 새로운 입력에 적절히 출력할 수 있게 되었다. 여기에서 대부분의 작동과정이 델타학습법에도 불구하고 매우 유사한 것을 볼 수 있다. 실제로 대부분의 신경망 학습과정은 이와 유사하다. 단지 연결강도의 조정과정에서의 자세한 사항에 있어서만 차이를 보일 뿐이다.

4) Adjustment of Weight

연결강도의 변경식은 기본 델타학습과 동일하게 다음과 같다.

$$w(n+1)=w(n)+\Delta w(n)$$
$$\Delta w(n)=\eta\delta y$$

이 식이 의미하는 것은 n+1시간의 연결강도 값은 n시간의 연결강도 값으로부터 현재 신경망의 오차 δ만큼 구성하여 구한다는 것이다. 여기까지는 델타학습법칙과 완전히 같다. 이제 δ값을 구해보자. 먼저 출력노드의 경우에는 앞서 설명한 것과 마찬가지로 이 노드가 출력해야한 올바른 값 d가 주어지기 때문에 다음과 같이 구할 수 있다.

$$\delta = y(1-y)(d-y)$$

즉 이 노드의 올바른 출력값에서 현재 신경망이 출력한 값을 빼서 오차를 구한 후에 전이함수인 sigmoid함수에 미분값 y(1-y)와 곱해서 δ값을 구한다. 이번에는 중간노드의 δ값을 구하여 보자. BP는 출력층의 오차를 층별로 역전파 시켜서 각 층의 연결강도를 조정하는 방법으로 중간층의 노드를 학습시킨다. 이를 위해서는 먼저 출력노드의 δ값을 계산하여 이 노드와 연결되어 있는 연결강도의 값을 조정한다. 그리고 나서 이 δ값을 구하는 과정에서 사용했던 연결강도의 값에 따라 이전의 노드 즉 중간노드로 이 값을 역전파 한다. 이 값들

을 이용하여 현재의 중간노드의 오차 δ값들을 다음과 같이 구한다.

$$\delta = y(1-y)(sum(\delta w))$$

이와 같은 방식으로 입력층의 노드에 이를 때까지 계속 오차를 역전파하고 이를 이용하여 연결강도의 값을 조절한다.

5) Example of BP

BP는 매우 광범위한 영역에 적용되고 있는데 여기서는 이 방법의 유용성을 설명해 줄 수 있는 정도의 몇 가지만을 설명하기로 한다. 실제로 BP 신경망은 문자인식, 지식처리, 로보틱스를 포함한 대부분의 응용 분야에서 사용되고 있다.

1. XOR Problem

Table 1 Pattern data sets of XOR problem

x(1)	x(2)	y
0	0	0
0	1	1
1	0	1
1	1	0

본 문제는 두 개의 이진 입력에 대하여 두 입력이 서로 다르면 1을, 같으면 0을 출력하는 것이다. 이러한 기능을 하는 논리회로를 XOR라 한다.

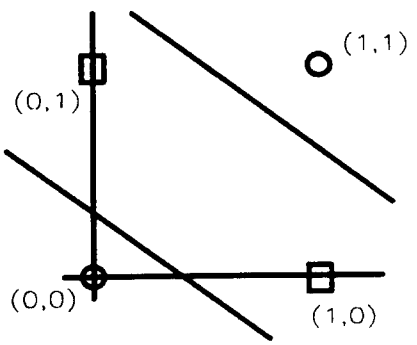

Fig. 7 Description of XOR problem

패턴인식의 입장에서 신경망의 학습을 상기해 보자. 학습은 입력패턴을 패턴영역상에서 직선 또는 평면을 사용하여 구분할 수 있도록 하는 것이다. 그럼 XOR문제를 위한 패턴영역상에 도시하고 이를 구분시킬 수 있는 직선을 그어보면 단층구조의 신경망으로는 구현할 수 없는 두 개의 직선이 필요함을 알 수 있다.

참고로 출력된 최종결과를 중간노드의 출력값과 함께 보여준다. 이것을 분석에 보면 BP학습법으로 학습된 신경망의 중간노드들의 역할이 자명해진다. 즉(0,0)이나 (1,1)에 대해서는 중간노드에서의 각각의 값에 부호만 다른 비슷한 연결강도 값을 곱하므로 0에 가까운 값을 출력할 수 있다. 또한 (1,0)이나 (0,1)에 대해서는 양의 연결강도를 갖는 중간노드가 큰 값을 출력하여 출력 노드는 1에 가까운 값을 내게 됨을 알 수 있다.

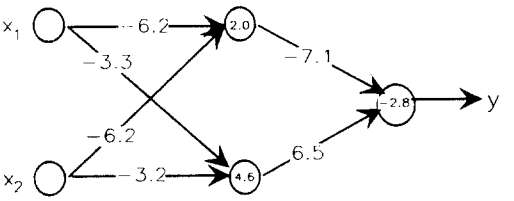

Fig. 8 A trained weight set of XOR problem

3. Numerical Implementation

numerical experiments는 다음 두 가지에 주안점을 두고 검토될 것이다. 우선 한가지 점은 구조해석과 최적화 과정에 참여되는 신경망의 위치와 역할에 관한 점이며, 또하나는 해당 신경망의 자체의 학습과 신경망 구조, 학습에 사용될 데이터 set에 관한 전반적인 정보에 대한 것이다. 우선 신경망이 구조 최적화 과정에 참여하게 되는 대략적인 묘사는 다음 그림과 같다

신경망이 구축될 수 있는 구조물 해석 응답면은 여러 문제에서 유용하게 쓸 수 있는데 그 예를 들어보면, 우선 다목적 함수 최적화 과정같이 많은 반복적인 해석이 필요한 경우와 많은 국부 최소점이 존재하여 반복적으로 초기점을 달리하면서 전체 최적점을 찾아야 하는 경우, 또 Genetic Algorithms과 같이 많은 설계 영역을 탐색하는 경우에 그 위력을 실감할 수 있을 것이다. 신경망으로 이

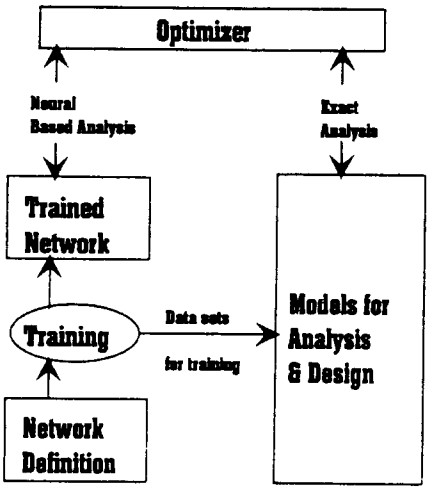

Fig. 9 Flowchart depicting the use of neural network in an optimization sequence.

루어진 응답면은 설계 영역 전체를 모사하는 응답면이라 할 수 있는데 이는 기존의 응답면 구성과 같이 관심 있는 현재점 부근의 응답면만을 우선 구성하고 진행해 나가는 것과의 극명한 차이점이 존재한다.

이러한 장점들은 일반적인 최적화 과정에 신경망을 참여시키는 motive가 된다. 그림 9를 참고하면서 그 과정을 설명하자면 다음과 같다. 우선 축적된 데이터 sets(입력과 출력 상으로 이루어진 집합)의 수와 mapping 대상체를 고려한 신경망을 구축한다. 이를 학습시킨 후 해석과 디자인을 위한 model을 대신하여 학습된 신경망을 위치시키는 것이다. 결국 학습된 인공 신경망이 실제 구조물의 모델링을 대신한다.

모든 실험은 Intel Pentium 586-60Mhz에서 이루어졌음을 밝힌다.

Example 1-Fox's Banana function

여기서 예시되는 함수는 Fox에 의해 개발된 것으로 다음과 같다.

$$y = 10x_1^4 - 20x_1^2 \cdot x_2 + 10x_2^2 - x_1^2 - 2x_1 + 5$$
$$-1.5 \le x_1 \le 1.5$$
$$-0.5 \le x_2 \le 2.0$$

아래 보여지는 등고선 그림을 통해 알 수 있듯이 최적점은 (1,1)에서 4라는 목적함수 값을 가진다.

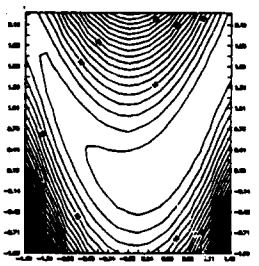

Fig. 10 Fox's banana function

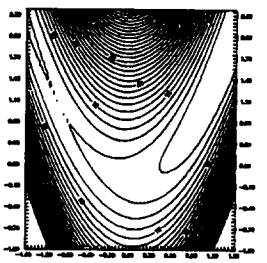

Fig. 11 Neural Net. with (2,6,1)

우선 위 식의 변수 범위 내에서 두 개의 변수를 무작위로 추출하고 상응하는 목적함수 값을 기억한다. 이렇게 2개의 입력과 1개의 출력으로 형성된 데이터 set들이 신경망의 학습 데이터로 사용된다. 본 예제에서는 각 변수당 7개씩, 모두 49개의 학습데이터를 추출하여 함수의 응답면을 만들었다. 신경망의 구조를 선택하는 문제는 경험적으로 보통 입력 (N)노드 수 더하기 출력 노드수 (N+1=3), 혹은 2N+1(=5)이지만은 본 문제의 경우, 학습 pattern 데이터의 입력, 출력 관계가 복잡하여 6개의 hidden 노드수가 필요함을 실험을 통해 알 수 있었다.

학습된 신경망을 이용하여 등고선을 나타낼 때 그 모양의 윤곽이 비슷해 보이지만, 실재 우리가 관심 있는 지역은 전체 최적점 부근의 값을 조사해 보면 그 부근의 학습 데이터의 분포에 따라 정확한 값을 주지하지는 못하였다. 위의 등고선 그림을 자세히 살펴보면 목적함수의 값이 크게는 150, 적게는 4의 값을 가지므로 이를 모두 mapping시키고자 할 때 양극 점에서 원하는 정도의 세밀한 값을 주지 못하는 것을 알 수 있었다. 그러므로 신경망을 이용한 최적점을 초기점으로 다시 기존의 최적화 방법으로 정확한 최적점과 해당하는 목적 함수의 값을 구할 수 있을 것이다. 다시 말하자면, 시간의 최적화 과정에서 대부분을 차지하는 전체 최적점까지의 접근은 신경망을 이용하

고 그 이후의 fine tuning은 기존의 최적화 방법을 사용한다는 전략이다.

역전과 학습 도중 숙지하여야할 것은 우선 weight의 range을 양수, 음수 모두 확보해야 한다는 것이다. 양수(0~1)만을 범위로 책정할 경우 학습 오차가 줄어들지 않는다. 또한 bias의 경우도 보통 1로 고정하는 경우가 많은데 이도 또한 범위를 양수, 음수 모두 확보하여야만 좋은 결과를 나타낸다.

Table 2 Optimal designs for the Fox's banana function using a neural network for analysis

Network description	x[1]	x[2]	Objective Function	Error	Computing time(sec.)
(2,2,1)	0.25	2.00	12.58	1.130 e-2	1.87
(2,4,1)	0.76	0.43	3.828	2.803 e-4	2.12
(2,5,1)	0.19	0.57	0.000	4.821 e-3	2.25
(2,6,1)	1.09	1.01	3.757	1.784 e-4	2.47
(2,10,1)	0.67	0.40	2.685	2.065 e-4	3.08

Exact Sol $x^*=(1.0, 1.0)$ $f(x^*)=4.00$

Five Bar Truss

본 문제는 그림 12의 5bar truss를 허용 변위 (D_{allow}=2.0 inch)라는 구속조건 안에서 최소중량을
구하는 최적화 문제로서 설계 변수가 각 element의 면적이다. 이를 수식으로 표현하자면 다음과 같다.

$$\text{minimize } f = \gamma \sum_{i=0}^{5} \text{Length}(i) \cdot x(i)$$

subject to

$$g_1 = \text{displ.}(1) / D_{allow} - 1 \le 0$$

$$g_2 = \text{displ.}(3) / D_{allow} - 1 \le 0$$

여기서 비중 γ는 0.1로 하였다.

신경망의 구축면을 살펴보면 앞의 예제는 Fox's function과 달리, truss 구조물의 전체 체적에 해당하는 목적함수는 비교적 계산이 간단하고 computing time이 적게 소요되므로 실제 위 계산식을 그대로 사용하고, FEM 을 통해 계산되는 변위를 학습된 신경망의 모듈로 대신한다. 학습에 사용할 데이터의 set는 입력으로 각 부재의 면적을 사용하고, 출력 데이터의 개수가 되도록 적어야 빠르고 적은 오차를 구할 수 있는 경향 때문에 truss 오른쪽 절점의 x, y 방향의 변위를 다 선택하지 않고 가장 큰 값을 주리라 예상되는 d_1, d_3만을 신경망의 출력층으로 대체하였다. 그러므로 입력층의 뉴런은 5개, 출력층의 뉴런은 2개로 신경망은 형성된다.

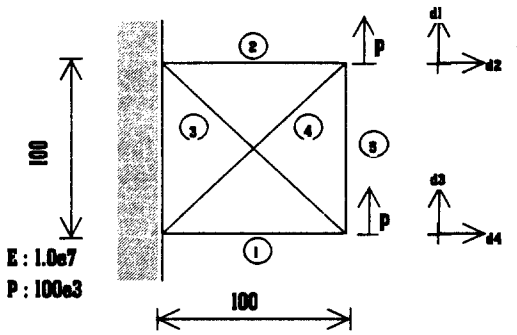

Fig. 12 Planar five bar truss

신경망 은닉층의 구성은 많은 시행착오를 거쳐 (5, 20, 2) 와 (5, 10, 10, 2) 구조를 선택하여 사용하였다. 학습에 필요한 pattern 데이터들은 1~10inch 사이에서 준 무작위(뒤에 설명)로 100개를 선택하여 이용하였고 학습 중에 결정되는 weight, bias의 선택은 −1~1사이에서 이루어졌다. 본 문제의 신경망으로 대체된 최적화 문제의 solution는 Genetic Algorithms에 의해 구했다. 본 문제의 경우 결과적으로 나온 해가 정해와 아주 비슷한 경향을 보이는 것을 알 수 있는데 그 원인을 살펴보면 신경망 target으로 정해진 값이 정규화된 데이터라는 점과 무작위로 선택되었지만 GA에서 최적화가 진행될 때 초반 5번 iteration의 best individual 의 값을 학습 데이터에 추가시킨 점을 주요 원인으로 들 수 있겠다.

비록 학습되는데 시간 ((5, 20, 2)의 경우 12 min 소요)이 필요하지만 일단 학습된 신경망은 탁월한 속도를 가짐을 알 수 있다. 또한 추가되는 데이터에 대한 학습이 기억된 weight, bias에서

Table 3 Optimal designs for the five bar truss using a neural network for analysis

Network Description	Design Variables x[1]	x[2]	x[3]	x[4]	x[5]	Objective value	Error	g[1]	g[2]	time (sec)
(5, 20, 2)	1.482	1.441	2.160	2.206	1.000	99.987	207e-4	-1.9e-4	-2.8e-4	26
(5, 10, 10, 2)	1.529	1.416	2.190	2.173	1.001	101.107	3.3e-4	-1.4e-4	-2.9e-4	54
Exact Sol with GA	1.544	1.504	2.092	2.118	1.000	100.02	-	-1.4e-5	-4.8e-5	911

부터 이루어지므로 upgrade가 용이하다.

Neuro-Optimizer

다음 예제는 신경망이 truss구조물 가해지는 하중에 따라 구조물이 스스로 환경에 적응해 나가는 과정, 즉 해당 환경에 대한 최적설계 구조물로 변화시키는 역할을 하는 경우다. 다시 말하면 입력층에 하중 데이터와 출력층에 각 설계변수(여기서는 단면적)를 매핑하는 신경망 구조로 이해하면 될 것이다.

본 문제는 2번 절점에 2~9K lb 근처의 범위에서 변화하는 하중에 대해 각 부재의 단면적이 모든 부재가 허용 응력 내에 존재하고 각 절점의 변위가 어떤 허용치 안에서 존재하도록 하면서 전체 중량을 최소화하는 값을 가지도록 설계된 신경망이라 간단히 소개할 수 있겠다. 본 시스템이 돌아가는 예를 들어보면 다음과 같다.

c:\NN>optinn

########## Five-Bar Truss Neural Net Optimizer ########

<Note> This module give Optimal 5bar-truss member's size which has Loads(Fx, Fy)(*10k lbs) by Users. Just put two loads(at No. 2 node)

(Fx)*10k lbs : 5.0

(Fy)*10k lbs : 5.5

** Result Size & Optimal Weight <in> **

x[1] = 1.40667

x[2] = 0.13401

x[3] = 1.40241

x[4] = 0.10426

x[5] = 0.09429

Optimal Weight = 5.46129e+00(lb)

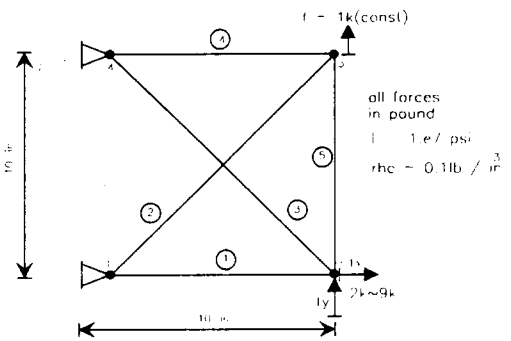

Fig. 13 Problem description

구조물이 설계될 당시에 모든 설계변수들이 정해서 나오는 것이 일반적인 상식이다. 그러나 life-cycle 동안에 변하는 환경 속에서 적응해 나갈 수 있는 일종의 intelligent structure를 위해서는 변해 가는 하중 속에서 자신에 모습을 여러 방법으로 적응해 갈 수 있을 것이다. 본 경우에는 비록 size만을 갱신하지만 더 나가 shape 도 변할 수 있는 가능성을 가진다.

4. 인공신경망에 의한 선체 중앙단면 설계

1) 선체 중앙단면 최적화

기존의 선체 중앙단면 최적화는 주로 Hooke and Jeeves 법에 의한 직접탐색법을 이용하면서 벌칙함수를 통해 구속조건을 만족시키는 SUMT 방법을 적용해 왔으나, 본 연구에서는 컴퓨터 공학 분야의 새로운 첨단기법의 하나인 인공신경망을 이용한 선체구조 최적화 방법을 개발하고 그 응용 가능성을 검토하고자 하였다.

특히 중앙단면의 최적화 시에 판 두께와 같은

설계변수가 갖는 이산적 변수의 특성과 보강재의 개수가 갖는 정수문제를 함께 고려하는 조합 최적화 문제를 효율적으로 다룰 수 있도록 뉴런의 상태를 binary 상태로 취급토록 하였으며, 최적시에 이산적 설계변수들의 특성이 중립축에서 멀리 떨어져 있는 부재 치수들의 증가를 어떻게 효율적으로 고려하는가를 살펴보았다.

2) 설계 문제의 정식화

기존의 많은 선체 중앙단면 최소 중량설계가 선저와 갑판 부재만을 변수로 두고 다른 부재는 선급규정의 값으로 고정시켜서 단면계수를 만족시키도록 선저와 갑판 부재를 줄여 가는 방법이었으나, 여기서는 중앙 단면부재의 치수와 보강재의 개수를 변수로 한 중앙단면 부재의 기하학적 면적을 목적함수로 취하였다. 대상선박은 Tanker의 중앙단면을 사용하였다.

DNV rule에 의한 부재의 최소치수와 중앙단면의 이차 단면계수 그리고 deck, bottom part의 Plate buckling을 제한 조건으로 하여 다음과 같이 정식화하였다.

Find : X_i $i=1,2,3,\cdots,19$

Such that : Minimize $F(X)=A\,X$

F : midship area

A : geometric coefficient vector

X : design variable vector

Subject to :

$G_i=x_i-t_i\geq 0$ Plate Thickness : $i=1,\cdots,13$

$G_j=Z_{cal}-Z_{req}\geq 0$

$G_k=\sigma_k-\frac{\sigma_{cal}}{\eta}\geq 0$ Buckling Constraint

outer, inner bottom : $\eta=0.9$

deck : $\eta=1.0$

예제로 사용한 선체는 아래와 같은 형상 및, Basic Parameter 부재 치수를 가지는 Tanker Type 1 이다. X[1]-X[13]은 이산 간격 0.5cm의 판 두께 부재 치수이고, X[14]-X[18]은 Stiffener의 개수를 설계변수로 잡았다.

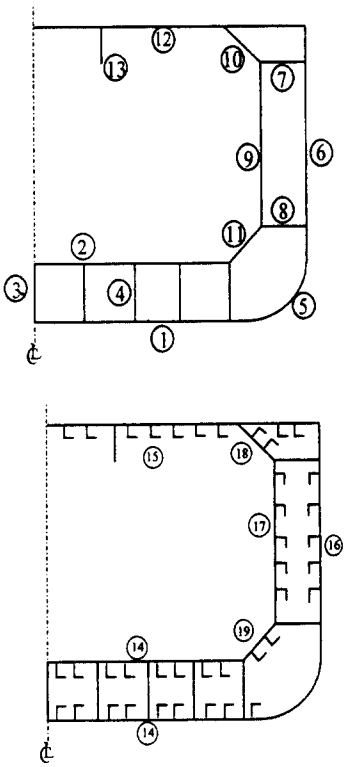

Fig. 14 Midship section

Table 4 Basic parameter of ship(m)

Part	Value
Length	171.50
Breadth	32.20
Depth	18.60
Inner Bottom Breadth	25.16
Inner Bottom Height	2.10
Hopper Tank Breadth	3.52
Hopper Tanker Height	4.60
Topside Tank Breadth	3.52
Topside Tank Height	2.80
Sideshell Breadth	2.00
Bilge Radius	1.70

Table 5 Stiffener specification(mm)

Spec.	A	B	C	D	E
Web length	150	200	250	250	300
Web thickness	8	9	10	12	11
Flange length	90	90	90	90	90
Flange thickness	13	14	15	16	16

Table 6 Design variable of midship section

X[1]	Outer Bottom Plate	X[11]	Hopper Side Bulkhead Plate
X[2]	Inner Bottom Plate	X[12]	Deck Plate
X[3]	Center Girder Plate	X[13]	Deck Girder Plate
X[4]	Side Girder Plate	X[14]	Outer & Inner
X[5]	Bilge Plate	X[15]	Deck
X[6]	Sideshell Plate	X[16]	Side Shell
X[7]	Stringer 1 Plate	X[17]	Side Bulkhead
X[8]	Stringer 2 Plate	X[18]	Topside Bulkhead
X[9]	Side Bulkhead Plate	X[19]	Hopper Blukhead
X[10]	Topside Bulkhead Plate		

3) 계산 결과 및 평가

Neuro-Optimizer 네트워크를 이용하여 최적화 과정을 수행한 계산 결과는 Table 7과 같다.

선급 Rule 에서 요구된 단면계수와 계산된 단면계수 값이 약 4% 정도로 근접해 있고 많은 설계변수가 제한조건 경계에 거의 닿아 있고 실적선과 비교할 때, 변수의 감소폭이 전체적으로 큰 편차가 없이 비슷한 경향을 보이는 것으로 보아 최적해를 구한 것을 알 수 있다.

X[10] 의 부재가 비교적 제한조건에 덜 근접해 있으나 한 단위를 내려서 즉, 12mm 값을 주면 X[10] 은 7% 정도로 제한조건에 더욱 근접해 가지만 Bucking 제한조건을 1% 정도 위반하게 되어서 해가 되지 못한다. 대부분의 부재의 값이 선급규정에 의한 제한조건의 경계에 있는 것으로 보아 갑판 부재와 선저부재를 제외한 다른 부재 값을 선급규정에 의한 값으로 고정하고 단면계수를 만족하는 갑판 부재와 선저 부재를 최적화 하는 기존의 방법이 어느 정도 타당하다고 생각된다.

GA의 결과는 주로 중립축에서 먼 부재에 큰 치수를 주고 가까운 다른 부재들을 줄임으로써 단면적을 줄여 가는데 비해, 인공 신경망은 부재의 전체적인 치수를 제한조건의 경계까지 줄여 가는 방법으로 최적화를 해 나가는 서로 다른 경향을 볼 수 있다.

실적선은 실제 건조 경험이 있는 배이므로 중량의 관점 외에 작업성, 공수, 건조기간 등이 같이 고려된 결과이므로 중앙단면 면적만으로 비교를 하기엔 다소 무리가 있지만 실적선과 실적선과 GA의 값들과 비교를 할 때, 신경망을 이용한 선체 중앙단면 최적화도 비교적 좋은 결과를 주는 것을 알 수 있다.

References

1. Hinchey, M. A Neural Network Fit to Icebreaker Resistance Data, Transactions of the ASME, Vol. 116, Nov. 1994.
2. Masson, E. and Wang, Y. J. Introduction to Computation and Learning in Artificial Neural Networks, European Journal of Operational Research 47(1990) 1-28
3. 김판영, 인공 신경망을 이용한 신뢰성 해석, 서울대학교 조선해양공학과, 1995. 2.
4. 김신형, 인공 신경망을 이용한 선체중앙단면 최적설계, 서울대학교 조선해양공학과 1996, 2.

Table 7 Optimum results obtained by neuro optimizer and GA

	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]	X[11]	X[12]	X[13]
Exist	14.5	13.5	13.5	11	14.5	13.5	11	11	11.5	14	10	14	13
GA	12	12	17	12	17	13	9.5	14	12	16.5	13.5	9.5	18
N.O.	12.5	11	12.5	10.5	13	13	10	9.5	11.5	12.5	12.5	10	9.5
	0.033	0.040	0.027	0.005	0.009	0.009	0.056	0.006	0.029	0.107	0.005	0.046	0.006
	X[14]	X[15]	X[16]	X[17]	X[18]	X[19]	Zb	Zt	Zr	F			
Exist	18E 13D	25B	22C	14C	3C	3E	10.05	7.0832	6.7331	1.610			
GA	19A	30A	24A	9C	9A	8B	9.388	7.2398	6.7331	1.4456			
N.O.	20A	29a	23a	7C	10a	4B	9.048	6.990	6.7331	1.3759			