신경 회로망을 이용한 자동차 번호판 인식 시스템의 설계 및 구현

이호현° ,최용호, 조범준
조선대학교 컴퓨터공학부

Design and Implementation of Recognition Vehicre Tag Using Neural Network System

ongHo Choi, HoHyun Lee, BeomJoon Cho
School.of Computer Engineering, Chosun University.
E-mail : newcom@ai.chosun.ac.kr, yhchoi@ai.chosun.ac.kr, bjcho@mail.chosun.ac.kr

요 약

본 논문에서는 미세 거리변와에 따른 자기자화의 변화를 구현하느데 있어 비선형적인 요소를 포함하고 있어 이를 수학적으로 모델링하여 제어기를 설계하는데는 많은 난점을 내포하고 있다. 따라서 거리변화에 따른 자기장의 비선형적인 변화 관계를 신경회로망 제어기의 학습을 통하여 제어할수 있도록 신경 회로망 제어기를 제안하려 한다.

1. 서론

주파수 응답을 이용하는 고전적 제어기 설계방법이나 상태변수 공간에서의 시스템 방정식을 기반으로 하는 현대 제어 방법에서는 제어 대상인 프랜트의 수학적 모델이 매우 중요한 역할을 한다. 그러나 매우 복잡한 프로세스는 수학적 모델을 쉽게 만들 수 없기도 하려니와, 설령 모델을 만들었다고 하더라도, 그 결과로 얻어진 비선형 시변체계와 같은 모델에 대하여는 적절한 제어기를 설계하기가 쉽지 않다.

더욱이 실제 시스템 제어시에는 외란 및 기타 파라미터 변동등에 의한 불확실한 여건으로 인하여 시스템의 특성이 순간적으로 비선형으로 변하는 수도 있고 시스템을 모델링하기 어려운 불명확한 대상도 있다. 제어변수가 대단히 많은 대구모 시스템이나 파라미터값이 시간에 따라 변화하는 시변 시스템과 같은 대상들의 경우도 수학적으로 명쾌한 제어방법을 구하기가 대단히 어렵다. 그런데 최근에 들어와 퍼지제어와 신경회로망 제어기법등과 같은 여러 지능제어기법등을 이용한 활용을 통해 새로운 관심을 모으고 있다. 지능제어의 개념은 시간이 흐름에 따라 여러 가지로 시도 변화되어 왔으나, 지능 시스템이 갖추어야 할 2가지 필수 기능 즉, 제어대상 모델에 대한 불확실성의 취급기능과 반복운전을 통해 성능향상을 달성하려는 학습기능은 궁극적으로 지능 제어기가 구현하여야 한다는 주장에는 변함이 없다. 알려진 바와 같이 신경회로 망은 불확실성의 취급 기능 및 학습기능을 가지고 있다.

본 논문에서의 미세 거리변와에 따른 자기자화의 변화는 매우 비선형적인 요소를 포함하고 있으므로 이를 수학적으로 모델링하여 제어기를 설계하는데는 많은 난점을 포함하고 있다. 따라서 거리변화에 따른 자기장의 비선형적인 변화 관계를 신경회로망 제어기의 학습을 통하여 제어할수 있도록 신경 회로망 제어기를 제안하려 한다.

2. 단층 신경 회로망

신경 회로망은 신경소자(neuron), 유닛(unit), 셀(cell) 또는 노드(node)라 불리는 단순한 기능을 수행할 수 있는 수많은 신경소자들이 병렬로 연결되어 이루어져 있다. 생물학적 신경회로망이 단순한 신경세포들의 대단위의 병렬연결로 이루어져 있듯이, 신경회로망도 단순한 기능을 수행할 수 있는 신경소자들의 수많은 병렬연결로 이루어져 있다. 기능면에서도 생물학적 신경회로망과 마찬가지로 병렬분산처리를 할 수 있을 뿐만 아니라, 학습이나 훈련을 통해서 결선강도를 조정하여 정보를 추가하거나 변경할 수 있는 적응 특성을 가지고 있다. 이와 같이 거시적인 차원에서 볼 때, 신경회로망은 생물학적 신경회로망의 구조와 기능을 닮았다고 할 수 있다. 그러나 신경소자들의 구체적인 기능, 결선을 통한 신호의 전학적 신경회로망과 많은 차이가 있거나 전혀 다를 수도 있다. 신경회로망을 구성하는 기본적인 요소는 다음과 같다.

2.1 신경소자

구성 요소들은 신경회로망의 종류에 따라 매우 다양한 형태를 갖는다. 신경소자는 신경회로망에서 이루어지는 연산을 수행하는 중심요소이다. 같은 종류의 신경소자들은 보통 층을 구성한다. Fig.2-1 은 층을 구성하고 있는 j 번째 신경소자를 보인 것이다.

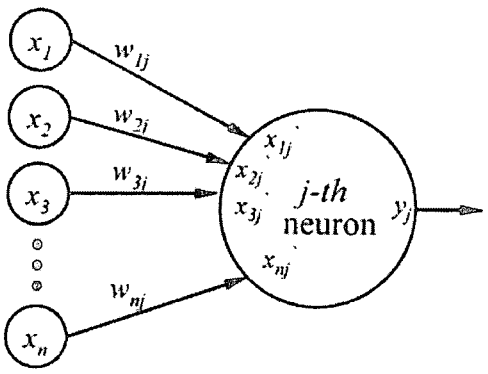


Fig 2-1. Neuron

Fig.2-1에서 x1, x2, ... xn은 신경소자에 인가된 입력이다. 이들은 외부로부터 인가된 입력일 수도 있고, 같은 층에 포함된 신경소자들의 출력이거나 다른 층에 있는 신경소자들의 출력, 또는 이들이 섞여있는 것일 수도 있다. 신경소자들 간의 결선은 신호가 전파되는 방향의 화살표로 나타낸다. 화살표 위의 wij는 I번째 입력으로부터 j번째 신경소자에 이른 결선의 강도를 표시한다. x1j' , x2j' , ...xnj' 는 각각 입력 x1,x2,.....xn 이 결선을 통하여 변조되어 j번째 신경소자에 전파된 양을 나타낸다. 이때 j번째 신경소자의 출력 yj 는 일반적으로 식 (2.1) 과 같이 계산된다.

$$y_i = f([\sum_{i=1}^{n} x_{ij}^{p}]^{1/p} - \theta_j) \qquad (2.1)$$

여기서 p 는 양의 정수로서 보통 1이나 2로 주어진다. θj 는 j 번째 신경소자의 bias 또는 threshold이고, f는 신경소자의 특성을 결정하는 활성 함수로서 대표적으로 다음과 같은 것들이 있다.

(1) 선형 함수fig 3.2.(a)

$$f(net) = \begin{pmatrix} \alpha\ net\ ; net \geq 0 \\ -\beta net; net < 0 \end{pmatrix} \qquad (2.2)$$

α, β는 양수이고 신경소자의 출력값은 실수 전체 범위에 포함된다. 일반적으로 실수 출력값을 생성하는 신경회로망의 출력층 신경소자에 사용된다.

(2)계단함수(fig2.2.(b))

$$f(net) = \begin{pmatrix} 1\ ;\ net \geq 0 \\ 0\ ;\ net < 0 \end{pmatrix} \qquad (2.3)$$

2진 출력값을 생성한다. Hopfield 회로망이나 BAM등과 같이 양극성 출력값을 필요로 하는 경우에는 다음과 같은 활성함수를 사용한다.

$$f(net) = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \qquad (2.4)$$

(3)선형 포화함수(fig2.2(c))

$$f(net) = \begin{pmatrix} 1\ ;\ 1/\alpha \leq net \\ \alpha\ net\ ;\ 0 \leq\ net\ \leq 1/\alpha \\ 0\ ;\ net < 0 \end{pmatrix}$$

선형함수와 계단 함수가 조합된 형태이다. α는 양수이고 이값에 따라 직선의 기울기를 조절할 수 있다. 특히 $\alpha \to \infty$ 이면 선형포화 함수는 식(2.3)과 같은 계단 함수가 된다. 양극성 포화값을 필요로 하는 경우에는 다음과 같은 활성함수를 사용하며, 이것은 $\alpha \to \infty$ 일 때 식(2.4)와 같은 계단 함수가 된다.

$$f(net) = \begin{pmatrix} 1\ ;\ 1/\alpha \leq net \\ \alpha\ net\ ;\ |\ net\ | < 1/\alpha \\ 0\ ;\ net < -1/\alpha \end{pmatrix} \qquad (2.6)$$

(4)sigmoid 함수(fig.2.2(d))

$$f(net) = \frac{1}{1 + \exp(-anet)} \qquad (2.7)$$

선형 포화 함수를 미분 가능한 형태로 변형한 것이다. $\alpha \to \infty$이면 sigmoid 함수는 (2.3)과 같은 계단 함수가 된다. 이 활성함수는 미분가능한 특성을 필요로 하는 신경회로망에 주로 사용된다. 한편, 양극성 포화값을 필요로 하는 경우에는 다음과 같은 활성함수를 사용하며 이것은 $\alpha \to \infty$ 일때 (2.4) 와 같은 계단 함수가된다.

$$f(net) = \tanh(-anet) \qquad (2.8)$$

(5)Gaussian 함수(fig.2.2(e))

$$f(net) = \exp(-\frac{net^2}{2}) \qquad (2.9)$$

입력패터의 특성에 따른 패턴분류를 행할 때, 출력층의 신경소자는 분류된 각 집단에 대응한다. 이 신경소자의 출력값이 특정 집단에 대한 소속도를 나타내도록 할 경우에 사용한다.

2.2 연결 강도

신경회로망에서의 연결가중치는 두 신경 세포들의 정보전달 정도를 나타내는 실수 값으로 표현된다. 다른 신경세포들로부터 전달된 활성값들은 각각 그것과 연결된 연결가중치들이 곱해져 다음 신경세포로 전달된다. 연결 가중치의

값이 크다면 그것에 연결된 신경세포의 활성값에는 큰값이 곱해져 더욱 커진 값이 전달될 것이며, 반대로 연결가중치의 값이 작다면 그것에 연결된 신경세포의 활성값에는 작은 값이 곱해져 전달되는 값은 더욱 작아질 것이다.

연결가중치는 이상과 같은 정보전달 제어 외에 또 하나의 중요한 의미를 가지고 있다. 그것은 지식의 분산 표현이 바로 이 연결가중치에 의해서 이루어진다는 것이다. 신경회로망에 있어서 그것이 가지고 있는 지식은 연결가중치에 저장된다. 여기서 중요한 것은 연결가중치 하나의 값이 하나의 지식과 1:1로 대응되는 것이 결코 아니라는 점이다. 신경회로망이 가지고 있는 모든 가중치들의 전체적인 상태가 바로 그 신경회로망이 가지고 있는 전체 지식이다.

신경회로망에 존재하는 모든 연결가중치의 값을 조절할 수 있다면 신경회로망에서의 정보전달 과정을 완벽하게 제어할 수 있을 것이다. 한편 신경회로망 스스로가 외부로부터 주어지는 정보에 대해 어떤 일정한 규칙에 의해 자신의 연결가중치를 조절할 수 있다면, 신경회로망은 주어지는 정보에 따라 자신의 정보 처리 상태를 변경시켜 갈 수 있을 것이다. 이것이 바로 학습의 원리이며, 연결가중치를 조절하는 일정한 규칙이 바로 신경회로망의 학습규칙이다.

2.3 학습 규칙

신경회로망의 학습은 원하는 정보를 회로망에 분산 저장하기 위하여 정해진 규칙에 따라 결선강도를 조정해 나가는 과정이다.

신경회로망이 학습을 통하여 결선강도를 조정하는 것에는 instar, outstar, perceptron, adaline, 역전파 신경망 등과 같이 신호를 한쪽 방향으로 전파하는 신경회로망과 미리 정해진 고정된 결선 강도를 갖는 신경회로망에는 MAXNET, Hopfield 회로망등과 같이 층내 결선 구조를 갖는 경우를 들 수 있다.

또한, 신경회로망의 학습은 지도학습과 자율학습으로 나눌 수 있다. 지도학습은 입력값과 목표출력으로 이루어진 학습 샘플들에 근거하여 입력과 출력사이의 대응관계를 가장 잘 구현하도록 결선강도가 정해진다. 그리고 자율학습은 입력패턴에 대응하는 목표출력이 주어지지 않고, 이런 경우에는 유사한 입력패턴들에 대해서는 각 신경소자들이 유사한 출력을 생성하도록 결선강도를 조정하는 것이다.

학습의 종류는 off-line 학습과 on-line 학습으로 나눌 수 있는데 off-line 학습은 일반적으로 주어진 학습샘플들을 사용하여 신경회로망의 성능이 원하는 수준이 될 때까지 결선강도를 조정하는 것이며, 신경회로망의 성능이 원하는 수준에 도달하면 학습을 종료하여 결선 강도를 고정시켜서 사용하게 된다. 이의 단점에는 학습종료 후 새로운 입력 패턴이 추가되면서 다시 처음부터 학습시켜야 한다. on-line 학습은 신경회로망의 성능이 일단 원하는 수준에 도달하더라도 학습을 종료하지 않고 인가되는 입력 패턴들에 대해서 결선강도를 조정해 나가는 것으로 단점으로는 결선강도가 최근에 인가된 입력패터에 크게 좌우되므로 이전에 분산 저장되어 있던 정보는 점점 약해진다. 다음은 다양한 신경회로망 구조에 대해서 살펴본 후 실험에 사용된 역전파 신경회로망을 설명하기 위한 기본 알고리즘을 묘사한다.

(1) Hebb의 규칙

뉴런이 흥분하면, 그 뉴런이 갖는 연결가중치 결합 도중에 자극을 전달한 것은 결합도가 증가하고 자극을 보다 전달하기 쉽게 된다고 하는 가장 단순한 형태의 학습 규칙이다. 신경소자 a가 활성화되어 이것과 연결되어 있는 신경소자 b가 활성화된다면 신경소자 a와 b사이의 연결강도는 증가한다.

$$W_{ba}^{new} = W_{ba}^{old} + \alpha y_a y_b \qquad (2.10)$$

(2)Perceptron

1950년대 말에 Rosenblatt F는 Hebb의 규칙에 기초해서 패턴을 학습 식별하는 기계, 퍼셉트론을 만들었다. 퍼셉트론은 다수의 수광소자로 되는 수용층과 다수의 모델 뉴런으로 되는 연합층과 한 개의 출력 뉴런을 갖고, 수용층에서 연합층으로는 랜덤 결선, 연합층에서 출력 입력으로는 전결선되고 있다. 이 장치의 수용층에 어떠 카테고리에 속하는 도형을 보이고, 출력 뉴런을 교사입력에 의해 흥분시켜, 또 그 카테고리에 속하지 않는 도형을 보이고, 흥분을 억제한다고 하는 것을 반복해 가면 출력 뉴런은 그 카테고리에 속하는 도형을 보였을 때만 흥분 하도록 된다. Hebb의 규칙에서는 연결가중치의 강화만 주장하고 있지만, 퍼셉트론에서는 출력 뉴런이 잘못돼 흥분했을 때 교사입력에 의해 연결 가중치는 약화 시키는 움직임도 추가 되고 있다.

(3) Wiheow-hoff 규칙

perxeptrondlsk adaline은 단일 소자 신경회로망으로 사용되기도 하지만 보통 은닉층을 이룬 단층신경회로망의 형태로 사용된다. 적을 신호처리 등의 분야에서는 활성함수가 선형함수으리 형태로 흔히 사용된다. ALC는 선형 신경회로망에 으로써 그 결선강도를 조정하는 학습규칙은 LMS에 근거한 wide-hoff 규칙이다. 그리고 이를 gradient 방법 혹은 최대경사법에 따라 순환형(recusive)알고리즘으로 구현하는 것이 델타규칙이다.

widrow-hoff 규칙에 따라 ALC와 같은 단층 신경망의 결선강도를 결정하기 위해서는 입력패턴과 그에 대응하는 목표출력으로 구성되는 하급샘플들이 필요하다. 활성함수가 선형함수인 경우 출력은

$$y_i = \sum_{i=1}^{n} w_i x_i \qquad (2.11)$$

주어진 입력패턴에 대한 신경회로망의 출력과 목표출력사이의 제곱오차를 다음 식과 같이 정의 한다.

$$\varepsilon = \frac{1}{2} \sum_{j=0}^{n} (y_{jd} - y_j)^2 \qquad (2.12)$$

또한 샘플 집합 전체의 평균제곱오차(MSE:mean square error)는

$$\varepsilon = \langle \varepsilon \rangle_s = \frac{1}{2} \langle \sum_{j=1}^{m} (y_{jd} - y_j)^2 \rangle_s \quad (2.13)$$

조정하고자 하는 결선강도 Wij에 관한 오차는 gradient는

$$\frac{\partial \varepsilon_r}{\partial w_{ij}} = \langle \frac{\partial \varepsilon_r}{\partial w_{ij}} \rangle_s = \frac{1}{2} \langle \frac{\partial}{\partial w_{ij}} \sum_{j=1}^{m} (y_{jd} - \sum_{i=0}^{n} w_{ij} x_j) x_i \rangle_s$$

$$= -\langle (y_{id} - \sum_{i=0}^{n} w_{ij} x_i) w_i \rangle_s$$

$$= -\langle (y_{id} - y_j) x_i \rangle_s \quad (2.14)$$

결선강도는 다음과 같은 순환형 gradient 알고리즘에 의해서 계산된다.

$$w_{ij}(k+1) = w_{ij}(k) - \alpha \frac{\partial \varepsilon}{\partial w_{ij}} = w_{ij(k)} + \alpha \langle (y_{jd} - y_j) x_i \rangle_s \quad (2.15)$$

3.다층 신경회로망

다층 신경회로망은 일반적으로 입력층 이외에 하나 또는 하나 이상의 중간츠으로 구성된 신경회로망을 말한다. 2장에서 말한 Hebb의 법칙과 퍼셉트론은 입력층과 출력층으로 이루어진 단층 신경망이다. 그러나 이러한 규칙은 배타적 논리합을 선형 분리 할수 없음을 알수 있었다. 따라서 퍼셉트론의 한계를 극복하기 위하여 Fig.2.3에서와 같이 3층이상의 구조를 갖는 다층 신경회로망에 많은 주목이 갔다. 다층 신경회로망은 학습 알고리즘인 역전파(backpropagation) 기법을 이용하여 오차가 감소하는 방향으로 가중치를 계속 수정하면서 학습한다. 학습후 신경회로망은 배타적 노리합 뿐만 아니라 비선형 제어부야에서도 우수한 성능을 갖는다.

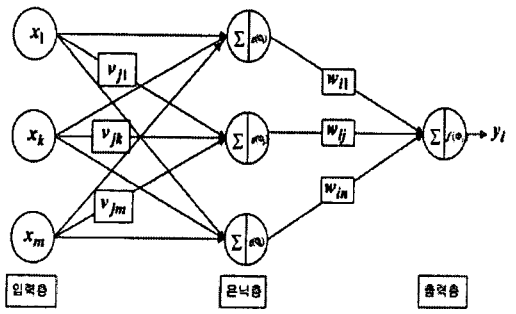


Fig-2.3 Structure of multilayer neural network.

(1)역전파 학습 규칙(Backpropagation learning rule)

역전파 회로망은 단층 perceptron을 확장한 형태로써 단층 perceptron의 한계를 극복할 수 있는 우수한 특성을 가지고 있다. 이와 같은 특성은 신경소자의 활성함수가 가지고 있는 비선형성에 기인한다. 역 전파 신경회로망과 같은 다층 perceptron을 학습하는 가장 대표적인 학습규칙은 역 전파 알고리즘이다. 미분가능한 비선형 활성함수를 가진 다층 perceprton에 LMS 방법을 확장하여 적용한 것인데 이를 일반화된 델타규칙이라고도 한다.

$$y_i = f(\Phi_i) \; ; \quad \Phi_i = \sum_{j=1}^{n} w_{ij} h_i \; ; \; j = 1,2,3,\ldots\ldots n \quad (2.16)$$

여기서 hj는 은닉층의 j 번째 신경소자릐 출력으로서 은닉층이 1개인 경우

$$h_j = g(\Phi) \; ; \quad \Phi_j = \sum_{k=1}^{m} v_{ij} x_k \; ; \; k = 1,2,3,\ldots\ldots m \quad (2.17)$$

회로망의 출력과 목표출력 사이의 제곱오차와 샘플집합전체의 평균제곱오차는 단층 perception 에서와 마찬가지로 정의된다. 먼저 출력층 Wij 에 관한 전체 오차의 gradient는 다음과 같다.

$$\frac{\partial \varepsilon_r}{\partial w_{ij}} = \langle \frac{\partial}{\partial} \frac{\varepsilon_r}{w_{ij}} \rangle_s = \frac{1}{2} \langle \frac{\partial}{\partial w_{ij}} \sum_{i=1}^{n} (y_{id} - y_i)^2 \rangle_s$$

$$= -\langle (y_{id} - y_i) \frac{\partial y_i}{\partial \Phi_i} \frac{\partial \Phi_i}{\partial w_{ij}} \rangle_s \quad (2.18)$$

여기서

$$\frac{\partial y_i}{\partial \phi_i} = \frac{\partial f(\phi_i)}{\partial \phi_i}$$

$$\frac{\partial \phi_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{j=1}^{n} w_{ijh_i} = h_j$$

따라서

$$\frac{\partial \phi_i}{\partial w_{ij}} = -\langle (y_{id} - y_i) \frac{\partial f(\phi_i)}{\partial \phi_i} h_j \rangle_s = -\langle \delta_{yi} h_j \rangle_s \quad (2\text{-}19)$$

$$\delta_{yi} = (y_{id} - y_i) \frac{\partial f(\phi_i)}{\partial \phi_i} \quad (2\text{-}20)$$

여기서, δ yi는 출력층 i번째 신경소자의 일반화된 오차이다. 마찬가지로 은닉층 결선강도 vij에 대한 전체 오차의 gradient는

$$\frac{\partial \varepsilon_r}{\partial w_{jk}} = \langle \frac{\partial \varepsilon_r}{\partial} v_{jk} \sum_{y_{id}} - y_i)^2 \rangle_s = \frac{1}{2} (\frac{\partial w_{ij}}{\partial v_{jk}} \frac{\partial}{\partial w_{ij}} \sum_{i=1}^{n} (y_{id} - y_i)^2 \rangle_s$$

$$= -\langle (y_{id} - y_i) \frac{\partial w_{ij}}{\partial v_{jk}} \frac{\partial y_i}{\partial w_{ij}} \rangle_s = -\langle (y_{id} - y_d) \frac{\partial y_i}{\partial \phi_i} \frac{\partial \phi_i}{\partial h_j} \frac{\partial h_j}{\partial \phi_j} \frac{\partial \phi_j}{\partial w_{jk}} \rangle_s$$

여 기 서 ,

$$\frac{\partial \phi_i}{\partial h_j} = \frac{\partial}{\partial h_j} \sum_{j=1}^{n} w_{ij} h_j = w_{ij}$$

$$\frac{\partial h_j}{\partial \phi_j} = \frac{\partial g(\phi_j)}{\partial (\phi_j)}$$

$$\frac{\partial \phi_j}{\partial v_{jk}} = \frac{\partial}{\partial v_{jk}} \sum_{k=1}^{m} v_{jk} x_k = x_k$$

$$\delta_{yj} = (y_{id} - y_i) \frac{\partial y_i}{\partial \phi_i} \quad (2.21)$$

따라서,

$$\frac{\partial \varepsilon_r}{\partial v_{jk}} = -\langle \sum_{j=1}^{n} \frac{\partial g(\phi_j)}{\partial \phi_j} x_k \rangle_s = \langle \delta_{hj} x_k \rangle_s \quad (2.23)$$

위식 우변의 미분항은 출력층의 일반화된 오차가 은닉층의 j번째 신경소자에 역으로 전파되어온 값이다. 결선강도의 조정을 위한 계산식은

$$w_{ij}(k+1) = w_{ij(k)} - d \frac{\partial \varepsilon_r}{\partial w_{ij}} = w_{ij}(k) + d \langle \delta_{yi} h_j \rangle_s$$

$$v_{ij}(k+1) = v_{ij(k)} - d\frac{\partial \varepsilon_r}{\partial v_k}(k) + d\langle \delta_h x_k \rangle_s \qquad (2,24)$$

이 식은 매번 샘플 집합전체를 다 사용하는 off-lone 알고리즘이다.

widrow-hoff 학습규칙과 마찬가지로 샘플 전체에 대한 오차의 gradient를 학습샘플 한 개에 대한 오차의 gradient로 근사화 하면 on-line 학습규칙을 얻을 수 있다. 이는 은닉층이 1개이지만 여러개의 은닉층을 가진 경우로 확장이 가능하다.

4. 구현 및 고찰

첫 번째는 영상파일을 신경망 입력패턴을 생성하기 위한 생성기이며, 두 번째는 역전파 알고리즘을 사용한 학습처리이다. 마지막은 이를 이용한 차량 번호판 인식 프로그램이다.

4.1 구현된 변수 설명

각 Layer의 개수는 Define하여 고정시켰다. 이때, 은닉층의 개수는 본인이 임의로 잡았다.

보통 입력층의 개수의 10분의 1이라는 설명이 있으나 이에 대한 최적화하는 방법에는 정의된 것이 없다. 약 20여개의 은닉층을 가질 경우 학습은 되지 않는다. 약 50개 이상의 은닉층을 가져야만 아래의 프로그램은 학습을 할 수 있었다. 또한 아래에서와 같이 120개의 은닉층을 가질 경우 Error값이 진동하지 않고 부드럽게 하강곡선을 그리며 줄어드는 것을 확인할 수 있었다.

```
#define    INPUT_NUM     260        //입력층의 노드수
#define HIDDEN_NUM 120//120       //히든층의노드수
#define    OUTPUT_NUM    10         //출력층의 노드수
```

아래의 변수는 학습률과 Beta값이다. Weight값을 갱신할 때 사용하는 변수로 Alpha는 아래 설명할 v와 w층에서 사용하며, Beta는 Bias값을 갱신할 때 사용한다.

```
#define alpha                    0.025//0.125
#define beta                     0.125//0.125
```

아래의 변수에서 입력층과 은닉층 사이의 Weight는 "v" 값으로 배열을 잡았다. 학습 프로그램에서 계산된 값들의 저장은 "vw.txt"에 저장하고 이때 이 파일을 사용할 때 사용한 FILE 변수의 변수명은 "v_weight"로 잡았다. 각 은닉층마다 하나씩 Bias Weight값을 갖는다. 이때 사용한 변수는 "bias_h"으로 배열을 잡고, 학습 프로그램에서 계산된 값들의 저장은 "hb.txt"에 저장하고 이때 이 파일을 사용할 때 사용한 FILE 변수의 변수명은 "h_bias"로 잡았다.

또한 은닉층의 출력값은 "output_h" 이름으로 배열을 잡았다. 이러한 형식으로 나머지 변수들도 변수명을 작성하였다. 이때 주의할 점은 v, w의 이중 배열의 순서이다. 모든 프로그램에서 일력층을 의미하는 변수는 i를 사용하였다. 은닉층은 j를 사용하였으며, 출력층은 k를 사용하였다. 아래의 설명에서처럼 2차원 배열의 순서는 신경망의 진행방향과 일치시켰다.

```
double v[HIDDEN_NUM][INPUT_NUM];
//입력층과 은닉층 사이의 Weight값
double bias_h[HIDDEN_NUM];
// v[은닉][입력] :: 배열의 순서 주의
double output_h[HIDDEN_NUM];
//은닉은:: j, 입력은:: i --> v[j][i]
double w[OUTPUT_NUM][HIDDEN_NUM];
//은닉층과 출력층 사이의 Weight값
double bias_o[OUTPUT_NUM];
// w[출력][은닉] :: 배열의 순서 주의
double output_o[OUTPUT_NUM];
//출력은:: k, 은닉은:: j --> w[k][j]
FILE *v_weight;
//계산된 Weight파일을 불러온다.
FILE *w_weight;
FILE *h_bias;
//bios값을불러온다.
FILE *o_bias;
```

아래 정의된 수는 Report0624.c에서 사용한 것이다. 받아들이는 영상파일이 40*60의 규격화된 Raw파일이다. 이를 신경망에 사용이 가능하도록 260개의 양자화된 입력값들 만들어 주었다. 이때 SUM의 의미는 3*3의 양자화 시 9개의 배열을 모두 더한다는 의미에서 사용하였다. 이는 앞에서 설명한 INPUT_NUM과 같이 사용할 수 있다.

```
#define DATA    2400
//파일의 크기가 40*60이다.
#define SUM              260
//양자화 파일의 크기는 13*20
```

아래의 target 배열은 학습할 시 입력한 그림과 하나씩 매칭이 되어 목표값으로 사용하기 위해서 사용하였다. 아래의 주석에서처럼 각각의 배열이 매칭되는 숫자는 다음과 같다.

```
int target[10][10]={{1,0,0,0,0,0,0,0,0,0}, //0
//학습목표를 주어준다.
                    {0,1,0,0,0,0,0,0,0,0},  //1
                    {0,0,1,0,0,0,0,0,0,0},  //2
                    {0,0,0,1,0,0,0,0,0,0},  //3
                    {0,0,0,0,1,0,0,0,0,0},  //4
                    {0,0,0,0,0,1,0,0,0,0},  //5
                    {0,0,0,0,0,0,1,0,0,0},  //6
                    {0,0,0,0,0,0,0,1,0,0},  //7
                    {0,0,0,0,0,0,0,0,1,0},  //8
                    {0,0,0,0,0,0,0,0,0,1}
};//9
```

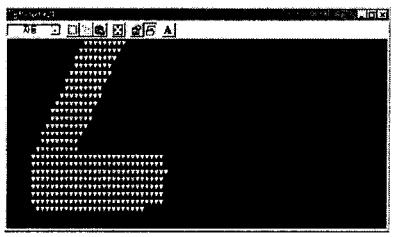
4.2 입력패턴 생성 프로그램

본 논문에서 구현에서 사용하는 영상은 순수 영상 데이터만 있는 raw파일이다. 이 영상파일의 크기는 40*60으로 그 픽셀하나를 모두 입력으로 받으면, 입력층의 개수는 2400개

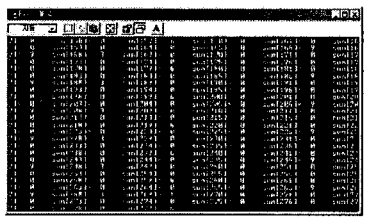
가 된다. 이를 2 Layer 신경망에 사용하는 것은 무리가 따른다. 따라서 이를 3*3의 메트릭스로 묶어 그 안의 변수를 간단히 모두 더하는 방식으로 간단히 양자화 하였다.

이렇게 처리한 파일을 "*.in"파일로 다시 저장하여 학습 프로그램과 인식프로그램에서 사용할 수 있도록 구성하였다.

```
char* fin_name[10]={"num00.raw",

"num10.raw",
"num20.raw",
"num30.raw",
"num40.raw",
"num50.raw",
"num60.raw",
"num70.raw",
"num80.raw",
"num90.raw"                    };

char* fout_name[10]={"num00.in",

"num10.in",
"num20.in",
"num30.in",
"num40.in",
"num50.in",
"num60.in",
"num70.in",
"num80.in",
"num90.in"                     };
```



a)raw파일을 텍스트 창에 출력



b)양자화된 숫자를 출력

그림1. 입력패턴 생성 실행화면

4.3 신경망 학습 프로그램

본 신경망 학습 프로그램은 2 Layer Neural Network구조를 가졌으며, 각 뉴런 함수는 Bipolar Sigmoid를 은닉층과 출력층에 모두 사용하였다. 또한 학습 알고리즘은 일반화된 델타규칙을 사용한 감독학습법 알고리즘, Back Propagation 알고리즘을 사용하였다.

```
-Bipolar SigmoidFuntion-
double sigmoid(double net)
  //bipolar sigmoid
{
  //tanh(0.5X) = (1-exp(-X)) / (1+exp(-X))
      return(tanh(0.5*net));
}
```

(수식)

$$\cosh = \frac{1+e^{-2X}}{2}$$

$$\sinh = \frac{1-e^{-2X}}{2}$$

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{1-e^{-2X}}{1+e^{-2X}}$$

∴ tanh(1/2*x) = Bipolar Sigmoid

rand()는 0 ~ 32767까지 숫자중의 임의의 수를 출력하는 함수이다.

```
      ((rand()-16383.5)/32767.)*1.0
(수식)
rand()                      ::0 ~ 32767
(rand()-16383.5)            ::-16383.5 ~ 16383.5
∴((rand()-16383.5)/32767)   ::-0.5 ~ 0.5
```

Initialize_weight()함수는 학습을 위해서 최초 임의로 Weight값을 채워주는 함수이다. 이때 -0.5 ~ 0.5 사이의 값을 사용해야 좋다. 따라서 위 수식을 사용하게 된다.

```
void Initialize_weight(void)
{
      int i, j;
//*********************************************
for(j=0; j < HIDDEN_NUM ; j++){
      for(i=0; i < INPUT_NUM; i++){
      v[j][i]=((rand()-16383.5)/32767.)*1.0;
//    printf("v[%d][%d] = %f\n", i, j, v[j][i]);
        }
      bias_h[j] = ((rand()-16383.5)/32767.)*1.
      }
//    getch();
//*********************************************
      for(j=0; j < OUTPUT_NUM ;j++){
       for(i=0; i < HIDDEN_NUM ; i++){
```

```
            w[j][i] = ((rand()-16383.5)/32767.)*1.0;
    //   printf("Wnw[%d][%d] = %f", i, j, w[j][i]);
                    }
            bias_o[j] = ((rand()-16383.5)/32767.)*1.0;
            }
    //      getch();
            return;
    }
```

forward_process() 함수는 신경망 알고리즘에서 Forward Propagation을 하는 함수이다.

```
    void forward_process(void)
    {
            int i, j, k;
            double temp;
            for(j=0; j < HIDDEN_NUM; j++){
                    temp=0.0;for( i=0;i< INPUT_NUM;i++)
temp += input_data[i] * v[j][i];
            output_h[j] = sigmoid(temp+bias_h[j]);
            }
    //**********************
            for(k=0 ; k < OUTPUT_NUM ; k++){
            temp=0.0; for(j=0 ;
    j<HIDDEN_NUM;   j++)      temp   +=   output_h[j]   *
w[k][j];
            output_o[k] = sigmoid(temp+bias_o[k]);
            }
            return;
    }
```

error()함수는 학습 시 기준 Error값 이하로 되면 학습을 멈추게 만드는 하는데 필요한 Error값 연산을 수행하는 부분이다. 현재는 은닉층이 120개 되면서 오차값들이 진동하지 않고 감소하므로 사용하지는 않는 if구문은 제외하고 화면에 에러값을 출력하여 정상적으로 학습이 이루어지고 있는지를 확인하는 목적으로 사용하고 있다.

```
    //if(error_avr < error_criterion) break;
    //에러가 기준이하이면 학습을 중지한다.
    //************<Error>************
    void error(int p)
    {
     int k=0;
     double temp=0.0;
     error_vector = 0.0;
      for(k=0; k<OUTPUT_NUM; k++){
            temp=0.5*(target[p][k]-output_o[k])
            *(target[p][k]-output_o[k]);
            error_vector += temp;
            }
            error_vector /= OUTPUT_NUM;
            return;
    }
```

아래의 두 함수는 일반화된 델타 규칙을 사용하여 Bipolar Sigmoid 함수를 적용한 경우의 역전파 알고리즘이다. 만약 적용하는 함수가 달라지면 아래 함수에서 사용하는 일반화된 오차값 (delta_h[k], delta_o[k])과 Weight 연산은 달라지게 된다.

```
    //*************< B P >***********
    void backward_process(int p)
    {
            int j, k;
            double temp;

            for(k=0; k<OUTPUT_NUM; k++)
    delta_o[k]=0.5*(target[p][k]-output_o[k])
    *(1.0- output_o[k]*output_o[k]);
    for(j=0; j < HIDDEN_NUM ; j++){
            temp = 0.0;
            for(k=0;k<OUTPUT_NUM;k++)
            temp+=w[k][j]*delta_o[k];
           delta_h[j] = 0.5 * (1.0 - output_h[j]
           *output_h[j]) * temp;
            }
            return;
    }
    //*************<updateweight>*************
    void weight_update(int p)
    {
    int i,j,k;
      for(k=0; k < OUTPUT_NUM ; k++){
            for(j=0 ; j < HIDDEN_NUM ; j++)
            w[k][j]+=alpha*delta_o[k]*output_h[j];
                    //output_o[k];
            bias_o[k] += beta*delta_o[k];
            }
            for(j=0; j < HIDDEN_NUM; j++){
             for(i=0; i < INPUT_NUM; i++)
            v[j][i]+=alpha*delta_h[j] * input_data[i];
                    //output_h[j];
            bias_h[j] += beta*delta_h[j];
           }
            return;
    }
```

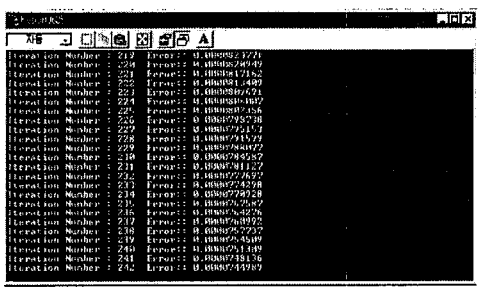
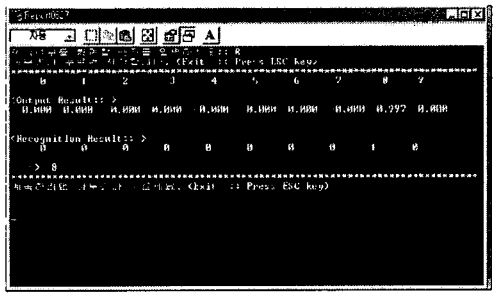


그림2. 신경망 학습 실행화면

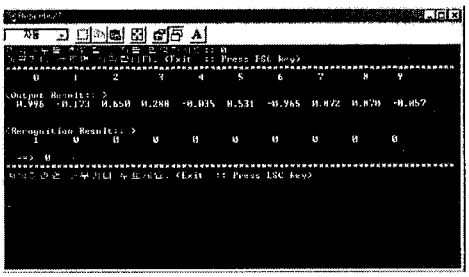
1.2.3 차량번호 인식 프로그램

프로그램에서 학습이 끝나면 앞에서 변수 설명에서 설명하였듯이 hb.txt, ob.txt, vw.txt, ww.txt이라는 4개

의 파일이 생성된다. hb.txt는 은닉층의 Bias를 저장하는 파일로 그 크기는 은닉층과 같은 크기의 120을 갖는다. ob.txt는 출력층의 Bias를 저장하는 파일로 그 크기는 출력층과 같은 크기의 10을 갖는다. vw.txt는 입력층과 은닉층의 Weight를 저장하는 파일로 그 크기는 입력층과 은닉층의 곱의 개수와 같은 크기의 31200개의 값을 갖는다. ww.txt는 은닉층과 출력층의 Weight를 저장하는 파일로 그 크기는 은닉층과 출력층의 곱의 개수와 같은 크기의 1200을 갖는다. 위 4개의 파일을 불러 각각의 해당 배열에 로딩한 후 각 파일을 모두 닫아준다. 본 논문에서 학습 알고리즘 부분에서 Initialize_weight(),backward_process(),weight_update(), error()는 사용하지 않는다. 오직 **forward_process()**을 사용한다. 인식에 대한 결과는 다음과 같다. 그림 4와 같은 선명한 그림 파일을 이용해 **20000번**의 학습이 된 Weight값들을 사용하였다. 학습에 사용한 파일을 이용하여 인식을 시켜면 그림3_a)와 같은 결과를 얻는다. Output_Result에서 알 수 있듯이 거의 정확한 결과 수치를 출력해 준다. 그림 5_c)의 그림과 같이 잡음과 기울어짐이 생기는 경우는 그림 4_b)와 같이 Output_Result가 연산됨은 알수 있다. Recognition_Result에서 알 수 있듯이 인식에는 별다른 영향을 주지 않을 정도로 목표값에 해당되는 Output값이 출력되어줌을 알 수 있다. 아래 **그림4와 그림 5는 모두 인식이 가능한 그림들을 보여주고 있다.**



a)학습용 파일을 이용한 경우



b)잡음이 들어간 파일을 사용한 경우

그림3. 실행화면

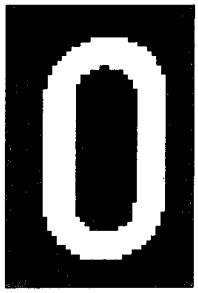


그림4. 학습용 Raw파일 (0)

a)잡음이 있는 경우 b)잡음과 뒤틀림이 있는 경우

c)잡음과기울어짐이 있는경우 d)숫자의 위치가 이동한경우

그림5. 인식 가능한 영상파일

5. 결론

본 논문에서는 신경망 프로그램 구현을 통해 잡음이나 기울어짐에는 좋은 적응력을 가지고 있음을 알 수 있었고, 이를 통해서 DSP에 사용할 Weight 값들을 산출할 수 있었다. 그러나 제한된 메모리를 사용해야 하는 마이크로 프로세스에서 사용될 경우 구현된 프로그램을 통해 얻은 Weight값이 매우 많아 상대적으로 롬의 사용량이 증가하여 연산 속도를 줄일 려고 하는 목적에 부합되지 않았다. 앞으로 가능한 적은 은닉층의 개수를 찾으면서, 고효율의 인식을 할 수 있는 Weight 값을 얻을 수 있도록 새로운 방법을 계속 연구 모색해야 한다.

[참고문헌]

[1] P.S.P. Wang, Chaacter and Handwrition Recognition, World Scientific Publishing Co. Inc,1991

[2] Marilyn Mccord Nelson & W.T IllING wort both of Texas Instruments" Rractical Guide to neural Networks", 1991.

[3] 김상우, 정윤호, 최종호, 신경 회로망을 이용한 인쇄체 한글 문자의 인식, 전자 공학회 논문지, pp 228-234, 1990

[4] 권오정 박정석 플로레터 프로그램

[5] K.S Fu and J.K.Mui, "A Survey on Image Segmentation", Patten recogition. 1991

[6] Digital Image Processing Algorithms and applications Gregory A. Baxes Wiley