사용자 특성에 적응하는 새로운 지능 제어 시스템

정 지원*, 유 석용*, 손 동설**, 엄 기환*

*동국대학교 대학원 전자공학과

** 유한대학 전자공학과

Adaptive Artificial Intelligent Illuminator for User's Characteristic

Jee-Won Jung*, Yu suk yong*, Dong-Seol Son**, Ki-Hwan Eom*

*Dept. of Electronic Eng. DongDuk Univ.

**Dept. of Electronics Eng. Yuhan college

E-mail: jungg1@cakra.dongguk.ac.kr

요 약

본 논문에서는 플랜트를 사용하는 사용자의 특성을 인공지능을 통하여 학습하여 사용자의 특성에 적응하도록 하는 새로운 지능 제어 시스템을 제안한다. 사용된 인공지능은 신경 회로망이며, 그 중에서도 LVQ(Learning Vector Quantization) 네트워크를 사용한다. 제안한 방식의 성능을 확인하기 위하여 IBM PC 상에서 Matlab을 통하여 시뮬레이션 한다.

ABSTRACT

In this paper, we propose a new intelligent control system to be adapted for the characteristic of user who use the plant. The proposed intelligent control system is composed of the artificial neural network, the learning vector quantization network. In order to verify the usefulness of the proposed system, we simulated in using Matlab.

I. 서 론

신경 회로망은 구조적으로 많은 양의 신호 및 데이터의 병렬 처리가 가능하고 특별한 기억 장치 없이 연결 강도나 활성함수를 통하여 기본적인 데이터를 기억할 수 있는 기능이 있으며 선형 대수와 수치 해석을 이용함으로서 스스로 학습하여 자체적으로 성능을 개선하는 등의 많은 장점을 갖고 있으므로 여러 분야에 폭넓게 적용되고 있다. 이러한 장점을 이용하여 제어 공학 분야에도 넓은 범위에서 신경 회로망이 적용된다.[1],[2] 본 논문에서는 제어 대상, 즉 플랜트를 사용하는 사용자가 여러 가지 변화하는 상황에 맞게 플랜트의 목표값을 지정하여야 하는 기존의 제어 방식과는 달리 각 상황에서 발생할 수 있는 모든 출력의 경우들 중에서 사용자가 가장 선호하는 출력을 만들 수 있도록 목표값을 자동으로 지정해주는 지능 알고리즘을 제안한다. 제안한 지능 알고리즘은 신경 회로망을 이용하여 구성하며 각 상황마다 사용자가 원하는 출력을 학습함으로서 학습 기간이 종료되면 이후 각 상황마다 사용자가 원하던 출력을 자동으로 만들어낼 수 있는 기능을 갖도록 한 것이다. 신경 회로망은 두 개의 학습 벡터 양자화(Learning Vector Quantization), 즉 LVQ 네트워크로 구성하며 LVQ-I은 검출된 신호의 패턴을 인식하여 LVQ-II를 학습시키고 LVQ-II는 학습된 결과를 토대로 상황을 판단하는 역할을 한다.

제안한 지능형 제어 알고리즘의 유용성을 확인하기 위하여 플랜트로서 조도가 조절 가능한 조명기에 적용하여, 조명기의 주위의 임의의 상황에 대하여 사용자가 원하는 출력을 학습한 후 같은 상황에 주어진 여러 가지 목표값들 중에서 사용자가 원하는 출력을 자동으로 발생시키는지를 시뮬레이션을 통하여 검토한다.

II. 본 론

II-1. 경쟁 네트워크(Competitive Network)

신경 회로망의 구조는 크게 전방향(Feed-Forward) 구조와 순환형(Recurrent) 구조로 구분될 수 있다. 경쟁 네트워크는 순환형 구조를 갖는 비지도 학습 신경 회로망으로 구성된다. 비지도 학습은 감독 학습과는 달리 신경 회로망의 입력을 자극으로 받아들여 그 자극에 반응하도록 동작한다. 이러한 비지도 학습은 Hebb의 규칙을 모태로 형성되었으며 Instar Rule과 Kohonen Rule을 통하여 발전하게 되었다.

경쟁 네트워크는 이러한 Instar Rule을 기초로 하여 각 뉴런의 출력을 다시 각 뉴런의 입력으로 순환시키는 뉴런(Recurrent Neuron)으로 구성된다.

경쟁 네트워크에서는 원형 패턴과 입력 패턴 사이의 벡터의 내적을 통하여 나타낸다. 그 때 이층의 뉴런들은 그들 중 승자를 결정하기 위하여 서로 경쟁한다. 경쟁 후에는 단지 하나의 뉴런만이 출력값을 0아닌 어떤 값을 갖게된다. 승리한 뉴런은 입력 벡터를 포함한 부류를 나타내게 된다. 경쟁층의 뉴런에서 사용되는 활성 함수(Activation Function)는 양의 선형 함수를 사용하며 순환형 뉴런으로 구성되어 있으므로 경쟁층의 출력은 식(1), 식(2)와 같이, 과거의 출력이 다시 뉴런의 새로운 입력이 된다.

$$a(0) = p \qquad (1)$$

$$a(t+1) = poslin(W \cdot a(t)) \qquad (2)$$

여기서 poslin은 양의 선형 함수를 나타내며 W는 순환형 연결 강도(Recurrent Weight)를 나타낸다. 또한, 경쟁 네트워크에서 순환형 연결 강도의 원소 값은 1과 음의 작은 값들로 구성되며, 이러한 행렬은 측면 억제(Lateral Inhibition)를 형성하여 그 표현은 식(3)과 같다.

$$W_{recurrent} = \begin{bmatrix} 1 & -\varepsilon & \cdots & -\varepsilon \\ -\varepsilon & 1 & \cdots & -\varepsilon \\ \vdots & \vdots & \vdots & \vdots \\ -\varepsilon & -\varepsilon & \cdots & 1 \end{bmatrix} \qquad (3)$$

식(2-3)에서 ε은 $0 < \varepsilon < \frac{1}{S-1}$의 값으로서 S는 뉴런의 개수를 나타내며 $-\varepsilon$은 억제값(Inhibition Value)이 된다. 측면 억제를 포함한 경쟁 네트워크의 구성도는 정리하면 그림1과 같다.

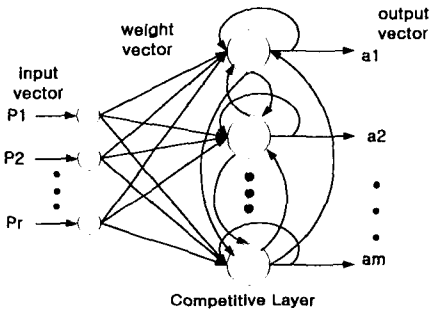


그림 1. 경쟁층의 갖는 신경 회로망의 구조
Figure 1. Structure of Neural Network with a Competitive Layer

각 뉴런에서 초기 입력값이 가장 큰 뉴런의 출력값은 다른 뉴런들의 출력값 보다 천천히 작아지게 되어 최종적으로 하나의 뉴런만이 출력을 양의 값으로서 갖게 되고 바로 이 시점에서의 상태가 안정 상태(Steady State)가 된다.

경쟁층의 연결 강도는 두 가지로 분리할 수 있는데, 하나는 원형 벡터를 표현하는 입력과 뉴런 사이에 연결된 연결 강도이고 또 다른 하나는 측면 억제를 갖으면서 뉴런 자기 자신과 다른 뉴런들 사이에 연결된 연결 강도이다. 입력과 뉴런 사이의 연결 강도는 일반적인 전방향 신경 회로망(Feed-Forward Neural Network)의 연결 강도와 같이 사용될 수 있으나 순환과 측면 억제를 위한 연결 강도는 경쟁을 위한 연결 강도이므로 별도로 정의를 해주어야하며 궤환과 측면 억제, 그리고 양의 선형함수를 포함하여 식(4)와 같이 '경쟁 함수(Competitive Function)'라고 정의한다.

$$a = compet(n) \qquad (4)$$

여기서, a는 경쟁층의 출력 벡터를 나타내고 n은 경쟁층의 입력 벡터를 나타내며 compet은 경쟁 함수를 나타낸다. 이 함수의 출력 벡터는 하나의 원소만이 1이고 나머지 원소는 0이 된다. 한편, 이러한 경쟁 함수의 정의를 해줌으로서 연결 강도의 표현은 한 개의 표현만으로도 가능하게 되어 입력과 뉴런과의 연결선만을 원형 벡터를 나타내는 연결 강도라고 정의할 수 있게 된다.

경쟁 네트워크의 학습 규칙은 원형 벡터를 알지 못할 때 사용되므로 연결 강도를 임의의 값으로 초기화 한다면 경쟁 학습을 통하여 원형 벡터를 찾을 수 있게 된다. 경쟁 학습 법칙은 인스타 학습 법칙(Instar Learning Rule)을 모태로 하여 코호넨 학습 법칙(Kohenen Learning Rule)으로 표현되며, 이 식은 식(5)와 같다.

$$W(t) = W(t-1) + \alpha(P(t) - W(t-1)) \qquad (5)$$

여기서, α는 학습률(Learning Rate)이고 P는 입력 벡터, W는 원형 벡터를 나타내는 연결 강도이다. 그러나, 연결 강도의 조정을 위한 식(5)는 승리한 뉴런에만 적용되며 다른 뉴런의 연결 강도는 과거의 값에 대하여 변함이 없게된다.

경쟁 네트워크는 패턴인식을 위한 전처리 과정에 많이 사용되어 실제 패턴을 인식하는 성능도 확인되었고 SOM(Self-Organizing Maps)의 기반 이론으로서 군집화와 패턴 인식에 널리 사용되는 LVQ(Learning Vector Quantization)의 중간층에 사용되어 가장 중요한 기능을 발휘하게 된다. [1],[2],[3],[4]

II-2. LVQ 네트워크의 구조와 동작

LVQ 네트워크는 다층 신경 회로망으로서 입력층, 은닉층(중간층), 출력층 등의 3개 층으로 구성된다. 입력층은 입력 벡터가 되고 은닉층은 경쟁층으로서 경쟁 네트워크를 사용하며 출력층은 선형층으로서 전방향 신경망으로 구성된다. 경쟁층에서 사용되는 활성 함수는 경쟁 함수(Competitive Function), 출력층에서 사용되는 활성 함수는 선형 함수(Linear Function)이다.

패턴인식을 위해서 사용되는 LVQ는 연결 강도

의 역할이 무엇보다 중요하다. 그 이유는 경쟁층의 연결 강도는 각 패턴들 군집의 중앙점(Center Point)을 표시하는 원형 벡터이고 군집들을 포함한 클래스(Class)는 선형층의 연결 강도와 함께 표현하기 때문이며 데이터 분석과 패턴 인식의 기준은 연결 강도의 값들에 의하여 결정된다.

이러한 LVQ 네트워크의 전체 구조는 그림2와 같다.

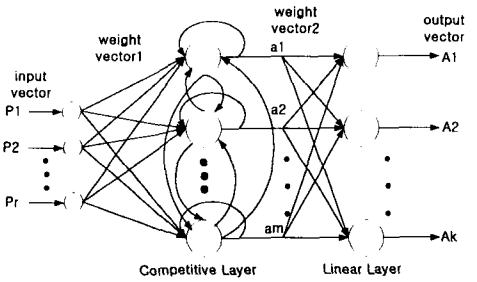


그림 2. LVQ 네트워크
Figure 2. LVQ Network

LVQ 네트워크에서 경쟁층의 입력 벡터는 경쟁 네트워크의 입력 벡터와는 다르다. 경쟁 네트워크의 입력 벡터는 입력 패턴과 원형 패턴의 내적을 계산한 결과를 입력 벡터로 사용하지만, LVQ 네트워크의 경쟁층은 입력 패턴과 원형 패턴의 거리를 직접 계산하여 그 결과를 입력 벡터로 사용한다. 따라서 경쟁 네트워크만을 구성하여 사용하면 내적 계산을 위하여 벡터의 일반화(Normalizing) 과정이 필요하지만 LVQ 네트워크에서는 이러한 과정이 필요 없이 직접 계산할 수 있는 장점이 있다. 경쟁층의 입력 벡터를 표현하면 식(6)과 같다.

$$ {}^1net = \begin{bmatrix} \| {}^1W_1 - P \| \\ \| {}^1W_2 - P \| \\ \vdots \\ \| {}^1W_m - P \| \end{bmatrix} \qquad (6) $$

여기서, ${}^1W$는 경쟁층의 연결 강도, 즉 첫 번째 연결 강도를 나타내며 ${}^1W_i$는 i번째 뉴런의 연결 강도 벡터를 나타낸다. 경쟁층의 출력 벡터는 경쟁 함수의 출력으로서 식(7)과 같다.

$$ {}^1a = compet({}^1net) \qquad (7) $$

그러므로 원형 벡터를 나타내는 경쟁층의 입력 벡터와 가장 가까운 연결 강도 벡터를 포함한 뉴런의 출력은 1이 되고, 나머지 뉴런들은 출력이 0이 된다. 이 출력 의 의미를 살펴보면, 경쟁 네트워크에서 승리한 뉴런은 입력 벡터를 포함한 클래스를 나타내지만 LVQ 네트워크에서 승리한 뉴런은 클래스에 포함된 부분 클래스를 나타낸다. 예를 들어 경쟁층의 j번째 뉴런의 출력이 1을 갖으면 j번째 뉴런에 연결된 연결 강도가 나타내는 원형 벡터에서 가까운 입력 벡터를 포함하는 부분 클래스를 인식하게 되는 것이다.

부분 클래스를 포함한 클래스는 선형층의 뉴런과 그에 연결된 연결 강도를 통하여 나타낸다. 부분 클래스를 나타내는 뉴런과 이를 포함한 클래스를 나타내는 뉴런은 서로 연결되어 있지만 클래스에 포함되지 않는 부분 클래스를 나타내는 뉴런은 연결되지 않는다. 결국 j번째 뉴런이 승리한 후 출력1을 만들면 이것이 나타내는 부분 클래스를 포함하고 있는 클래스의 뉴런이 선형층의 J번째라고 한다면 출력이 1이 되어 결국 부분 클래스와 클래스를 모두 인식하게 된다. 이러한 동작원리를 패턴 인식 그림으로 표현하면 그림3과 같다.

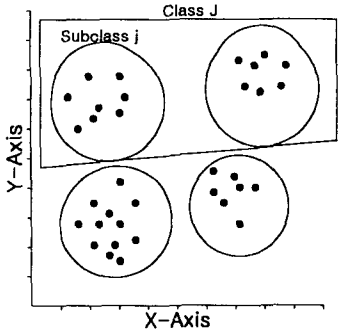
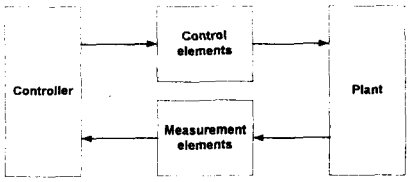


그림 3. 군집 및 클래스를 이용한 패턴인식
Figure 3. Pattern Recognition using Cluster and Class

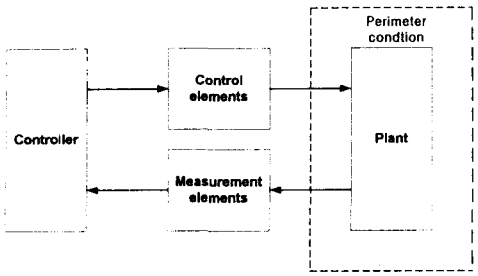
LVQ 네트워크의 이러한 동작 원리를 통하여 군집화를 할 수 있으며 군집화와 함께 패턴인식이 가능하게 된다. 기존의 군집화 및 패턴인식기의 구조보다 비교적 간단하고 부분 클래스와 클래스의 표현이 용이하여 많은 부분에서 사용되고 있다. [1],[2],[3],[4]

II-3. 제안한 지능 제어 시스템

기존의 일반적인 제어 시스템의 구성은 그림 4와 같다.



(a) 기존의 일반적인 제어 시스템



(b) 주변상황을 고려한 일반적인 제어 시스템

그림 4. 기존의 일반적인 제어 시스템의 구성도
Figure 4. Block Diagram of General Control System

일반적인 제어시스템은 제어대상(Plant), 제어를 위한 제어 요소들(Control elements), 측정 요소들(Measurement elements), 그리고 이들 신호를 이용하여 제어 대상을 제어하는 제어기 등으로 구성된다. 여기에 제어 대상의 주변상황(Perimeter condition)을 추가하면, 제어대상의 출력은 주변상황에 맞게 달라져야 한다. 주변상황을 감안한 제어 시스템의 예로서 에어콘(Air Conditioner), 광도 조절이 가능한 조명기(Illuminator), 난방 장치(Heater), 자동차, 비행기 등 여러 가지가 있다. 이러한 장치들은 주변상황이 달라지면 사용자가 원하는 상황을 만들기 위하여 출력을 만들어야 하고 주변상황이 달라지면 그때마다 사용자가 입력을 설정해 주어야 한다.

본 논문에서는 사용자가 이와 같은 시스템을 사용할 때 영구적이면서 지속적으로 변화하는 주변 상황마다 입력을 설정할 필요 없이, 일정한 학습만 시켜주면 각 상황에 적응하는 지능 제어 시스템을 제안한다. 제안한 지능형 제어 알고리즘을 구현하기 위해서는 하나의 가정이 필요하다. 이 가정은 제어대상의 출력과 그에 대응되는 제어기의 입력은 정확하게 사상(Mapping)이 이루어진 상태라고 가정한다. 이것은 제어대상의 사용자가 원하는 출력이 설정되면 그에 해당하는 제어기의 입력이 자동으로 결정됨을 의미한다. 예를 들어, 제어 시스템의 입력과 출력 중에서 원하는 출력이 선택되면 이것과 사상된 입력도 함께 선택되는 것이다. 이와 같은 가정 하에, 본 논문에서 제안한 지능형 제어 시스템의 구조는 그림5와 같다.

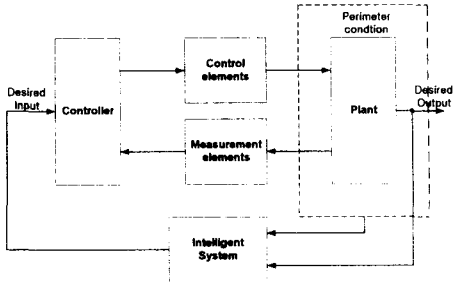


그림5. 제안한 지능 제어 시스템의 전체구조
Figure 5. Block Diagram of Proposed Intelligent Control System

제안한 지능 제어 시스템에서 제어대상의 출력과 주변상황은 센서(Sensor)에 의하여 검출되고 이 신호가 지능 시스템(Intelligent System)의 입력이 되면 주어진 주변상황과 그 때의 사용자가 원하는 출력을 인식하고 학습한다. 이 후 학습이 종료되면 제어 대상에 임의의 상황이 주어졌을 때 학습된 지능 시스템은 그 상황에 맞는 제어기의 입력값을 만들어준다.

지능 시스템은 신경 회로망으로 구성된다. 패턴 인식을 위한 신경 회로망으로서 두 개의 LVQ를 사용하고, 첫 번째 LVQ를 'LVQ-I', 두 번째 LVQ를 'LVQ-II'라 명칭하며 그 구성은 그림6과 같다.

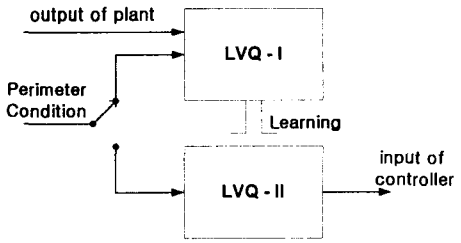
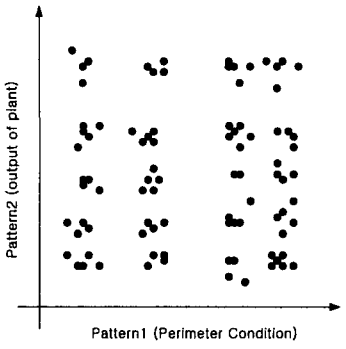


그림 6. 제안한 지능 시스템의 구조
Figure 6. Structure of the Proposed Intelligent System

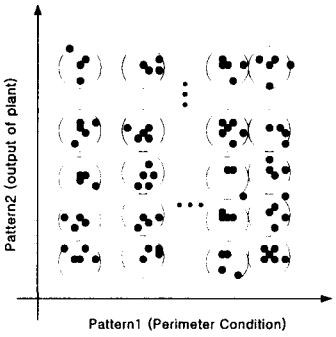
여기서, LVQ-I은 제어대상의 주변상황과 제어대상의 출력을 입력 패턴으로서 인식하여 LVQ-II를 학습시키고, 학습이 종료되면 LVQ-II에 주변상황 신호가 입력으로 들어가 주변상황에 대하여 학습된 결과를 패턴들을 기반으로 각 상황에 맞게 제어기의 입력을 만들게 된다. 주변상황을 검출하여 얻은 신호가 연결된 선은 학습기간과 관련되어 스위치 역할을 하며 학습 후에는 제어대상의 출력 신호는 검출하지 않고 주변상황만 검출하여 LVQ-II의 입력으로 들어간다.

주변상황에 대한 데이터들과 그에 대응하는 제어대상의 출력 데이터가 그림 7과 같이 주어진다면 패턴인식을 위하여 제어대상의 특성에 맞게 군집화를 할 수 있다. 또한 군집들 중에서도 같은 특성을 갖는 것끼리 서로 묶어서 하나의

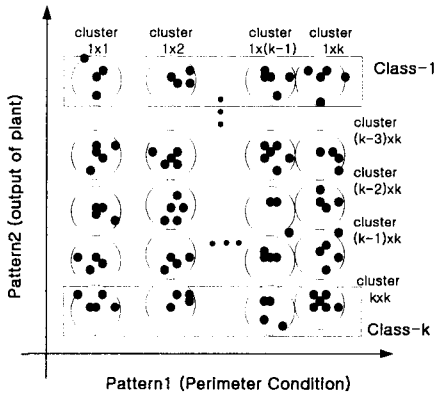
클래스를 형성할 수 있다. 군집을 만들거나 클래스를 만드는 것은 패턴 인식기를 설계하는 사람이 제어대상과 주변상황의 성질에 따라 임의의 선택할 수 있으며 제안한 지능 시스템을 적용하기 위해서도 이 과정은 반드시 선행되어야 한다. 그림7는 발생할 수 있는 기본적인 데이터들을 가정하여 나타내고 있다.



(a) 주변 상황과 제어 대상의 출력 데이터



(b) 출력 데이터의 군집화



(c) 군집화된 데이터의 패턴인식

그림 7. 주변상황과 출력데이터 및 패턴인식
Figure 7. Perimeter Condition and Output Data, Pattern Recognition

여기서, 그림 7(a)는 R차원의 주변상황 패턴벡터와 R'차원의 플랜트의 출력 패턴벡터들을 2차원 평면상에서 나타낸 그림이다. 그림 7(b)는 그림 7(a)의 데이터들에서 k ×k개의 점들을 중앙점(Center Point)으로 지정하여 중앙점에서 가까운 거리의 데이터들끼리 군집화를 한 것이다. 그림 7(c)는 그림 7(b)의 군집들을 플랜트의 출력 패턴이 유사한 것들끼리 클래스로 묶어준 것이다. 이처럼 데이터들을 특징에 따라 데이터-베이스(Data-Base)화하여 패턴 인식의 전처리 과정을 수행하면 패턴 인식기를 설계할 수 있다.

그림 7을 기반으로 패턴인식기를 설계한다. 제안한 알고리즘에서 사용된 LVQ 네트워크를 이용하여 패턴인식기를 만들고 다시 이것을 기반으로 학습용 패턴인식기를 설계한다. 그림4-4의 데이터-베이스를 기반으로 설계한 LVQ 패턴인식 네트워크는 그림 8와 같다.

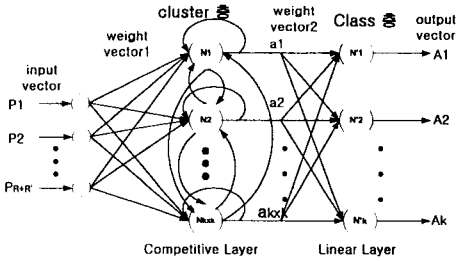


그림 8. 지능 시스템의 LVQ-I 네트워크
Figure 8. LVQ-I Network of the Intelligent System

여기서, 입력 벡터는 R+R'차원이다. 그 이유는 그림 7에서 가로축의 차원을 R차원으로 설정하였고 세로축의 차원을 R'차원으로 설정하였으므로 LVQ-I의 입력으로는 이 두 차원을 더한 R+R'차원이 되기 때문이다. 또한 이 R+R'차원은 연결 강도를 결정하는 원형 벡터의 차원도 결정해준다. 입력 벡터와 입력층과 중간층의 연결강도를 나타내는 연결강도 벡터는 식(8), 식(9)와 같다.

$$P=\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_R \\ P_{R+1} \\ \vdots \\ P_{R+R'} \end{bmatrix} \tag{8}$$

$$^1W=\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1R+R'} \\ W_{21} & W_{22} & \cdots & W_{2R+R'} \\ \vdots & \vdots & & \vdots \\ W_{k^2 1} & W_{k^2 2} & \cdots & W_{k^2 R+R'} \end{bmatrix} \tag{9}$$

3차원 이상의 고차원 벡터로서 두 종류의 벡터를 나타내기 위하여 본 논문에서는 각 벡터를 직교 좌표의 가로축과 세로축에 위치하게 함으로서 R+R'차원의 입력 벡터를 이차원적으로 해

석한다. 그 이유는 데이터의 종류를 구분하거나 군집화를 할 때 데이터 벡터들간의 거리를 계산하는데, 거리를 도시적으로 나타내기 위하여 직교 좌표를 이용하면 점들 사이의 거리를 쉽게 이해할 수 있기 때문이다. 또한 '이차원' 직교 좌표를 이용한 이유는 입력 패턴이 제어대상의 주변상황 패턴과 출력 패턴, 두 종류를 각각 가로축과 세로축으로 구분하여 나타내기 위해서이다.

중간층의 뉴런들은 그림7(c)의 각 클러스터와 사상(Mapping)되며 첫 번째 뉴런 N1은 Cluster 1×1, 두 번째 뉴런 N2는 Cluster 1×2, …, 마지막 번째 뉴런 N $k\times k$는 Cluster $k\times k$을 나타낸다. 출력층의 뉴런들은 각각 그림7(c)의 클래스를 나타낸다. 각 클래스들은 소속클래스(Subclass), 즉 클러스터를 갖고 있으며 이러한 원리를 통하여 중간층과 출력층의 연결 강도가 결정된다. 중간층의 뉴런 N1부터 Nk는 클래스-1에 소속되어있으므로 뉴런 N1부터 Nk과 뉴런 N'1과의 연결 강도만이 '1'의 값을 갖고 N'1과 중간층의 나머지 뉴런들과의 연결 강도는 '0'의 값을 갖게 된다. 결국 이와 같은 원리에 의하여 중간층과 출력층의 연결 강도가 결정되어 식(10)과 같이 연결 강도 벡터가 결정된다.

$$ {}^2W=\begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 & 1 & \cdots & 1 \end{bmatrix} \tag{10} $$

여기서, 연결 강도 벡터 ${}^2W$은 $k\times k^2$차원이 되며, LVQ-I의 동작은 다음과 같다.

먼저, 입력 벡터 P가 들어오면 원형 벡터 ${}^1W$과 의 거리를 계산하여 중간층의 뉴런들의 입력을 만들고 식(11)와 같다.

$$ {}^1net=norm(P-{}^1W)=\|P-{}^1W\| \tag{11} $$

이 중간층은 경쟁 함수를 사용하므로 중간층의 출력은 식(12)와 같다.

$$ {}^1a=compet({}^1net)=\begin{bmatrix} {}^1a_1 \\ {}^1a_2 \\ \vdots \\ {}^1a_{k^2} \end{bmatrix} \tag{12} $$

${}^1a$는 다시 ${}^2W$와 곱해져 식(13)과 같이 되어 출력층의 입력으로 들어가며 양의 선형 함수를 사용하는 출력층의 출력은 식(14)과 같다.

$$ {}^2net={}^2W\cdot{}^1a=\begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 & 1 & \cdots & 1 \end{bmatrix}\cdot\begin{bmatrix} {}^1a_1 \\ {}^1a_2 \\ \vdots \\ {}^1a_{k^2} \end{bmatrix} \tag{13} $$

$$ A=poslin({}^2net)=\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix} \tag{14} $$

위 식에 의하여 LVQ-I의 출력이 결정되며, 결국 제어대상의 주변상황 벡터와 출력 벡터를 패턴 인식하게 된다. 그림 8과 같은 LVQ-I 네트워크는 패턴을 인식하면서 또 다른 LVQ 네트워크, 즉 LVQ-II를 학습시키며 LVQ-II 네트워크는 그림9와 같이 설계된다.

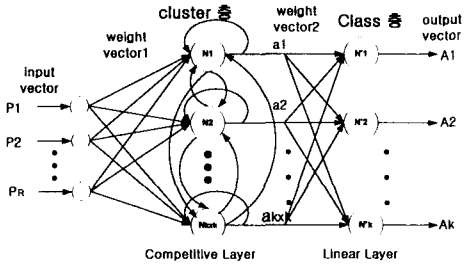


그림9. 지능 시스템의 LVQ-II 네트워크
Figure 9. LVQ-II Network of the Intelligent Network

LVQ-II에서 입력 벡터의 차원은 LVQ-I의 입력 벡터 R+R'차원과는 달리 LVQ-II R차원이다. LVQ-II는 입력으로 제어대상의 주변상황만을 입력으로 받아들이고 LVQ-I에 의하여 학습된 결과를 바탕으로 사용자가 원하는 클래스의 출력을 만들기 때문에 입력 벡터가 R차원이 된다. 그러나 LVQ-I 구조와의 차이는 입력 벡터에 의하여 입력층과 중간층 사이의 연결 강도 이외에는 존재하지 않기 때문에 중간층의 뉴런수와 출력층의 뉴런수, 그리고 중간층과 출력층 사이의 연결 강도는 같다. LVQ-I과 차이나는 LVQ-II의 입력 벡터와 연결 강도는 식(15), 식(16)와 같다.

$$ {}'P=\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_R \end{bmatrix} \tag{15} $$

$$ {}^1{}'W=\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1R} \\ W_{21} & W_{22} & \cdots & W_{2R} \\ \vdots & & & \vdots \\ W_{k^21} & W_{k^22} & \cdots & W_{k^2R} \end{bmatrix} \tag{16} $$

$'P$가 LVQ-II의 입력이 되어 원형 벡터 ${}^2W$과의 거리를 계산하는 것은 LVQ-I의 동작 원리와 같으며 그 표현은 식(17)과 같다.

$$^1n\!et = norm(P - {}^1W') = \|P - {}^1W\| \quad (17)$$

제안한 방식은 LVQ-I에서 어떤 원형 벡터와 거리가 가까운 입력 벡터가 들어왔을 경우 이 원형 벡터와 연결된 뉴런은 활성하고 LVQ-II의 뉴런을 학습시킨다. 즉, LVQ-II의 중간층의 모든 뉴런들이 LVQ-I의 모든 뉴런들과 각각 하나씩 연결되어, LVQ-I의 중간층의 뉴런중에서 j번째 뉴런이 활성하면 LVQ-II의 중간층의 j번째 뉴런은 활성도이 높아지도록 한다. 그래서, 학습을 마치면 LVQ-II의 중간층의 뉴런들은 각각 자기만의 활성도를 갖게 되어 LVQ-I의 입력 벡터들 중에서 원소 P_{R+1}부터 $P_{R+R'}$까지의 원소가 없어도 활성도가 높은 뉴런이 활성할 수 있도록 한는 것이다. 이것은 식(18)에 의하여 표현될 수 있다.

$$^1a = compet(net/\delta) \quad (18)$$

여기서 δ는 활성도를 나타내고 이것은 제안한 방식을 위하여 식(19)와 같이 정의된다.

$$\delta = 1 + \alpha \cdot L \quad (19)$$

α, 또한 제안한 방식을 위하여 새로 정의되며 학습률을 나타낸다. α는 $0<\alpha<0.01$의 범위를 갖아야 하며, 거리를 계산함에 있어서 경쟁 함수의 입력에는 영향을 미치지만 다른 클러스터를 나타내는 클러스터와의 거리에는 영향을 미치지 말아야 하게 때문에 그 값이 작아야 한다. L은 LVQ-I의 중간층의 각 뉴런에 대한 활성 횟수를 나타낸다. 결국, LVQ-I의 중간층의 j번째 뉴런이 l번 활성했다면 L은 l값이 되고 LVQ-II의 중간층의 j번째 뉴런의 활성도는 같은 클래스에 속하는 다른 뉴런들 보다 상대적으로 높아지게 된다. 이와 같은 원리를 통하여 하나의 클래스 내에 활성도가 높은 뉴런이 승리하게 됨으로서 임의의 주변상황이 주어지면 그 상황에 맞는 뉴런이 승리하여 지능 시스템의 출력을 만들게 된다.

III. 시뮬레이션

제안한 지능 제어 시스템의 성능을 확인하기 위하여 제어대상을 조절 가능한 조명기에 적용하여 시뮬레이션 한다. IBM PC상에서 MATLAB을 이용하여 검토한다.

제어대상인 조명기는 주변상황이 주위밝기가 되며 출력이 조명기의 밝기가 된다. 주위밝기를 S1이라 하고 조명기의 출력을 S2라고 하여 군집화와 클래스를 그림10과 같이 임의로 설정하여 데이터-베이스를 만든다.

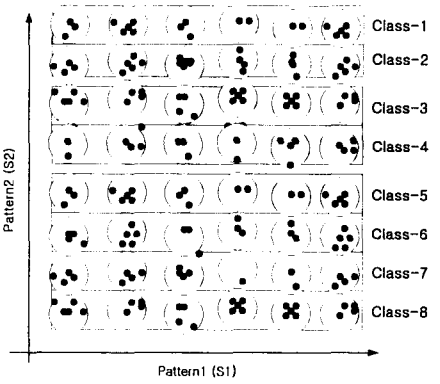


그림 10. 조명기의 주위밝기와 출력 데이터
Figure 10. Illuminance of the Illuminator and Output Data

데이터 그림에서 주위밝기는 6개의 단계로 구분하여 군집을 설정하고 조명기의 출력 조도는 8개의 클래스로 설정하였다. 총 48개의 군집과 6개의 군집을 갖는 8개의 클래스가 되도록 한 후, 제안한 지능 시스템의 LVQ-I를 설계하면 입력 벡터의 원소는 2개, 중간층의 뉴런수는 48개, 출력층의 뉴런수는 8개가 된다. 각 군집의 중심점의 좌표는 다음과 같다.

(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8)
(2,1), (2,2), (2,3), (2,4), (2,5), (2,6), (2,7), (2,8)
(3,1), (3,2), (3,3), (3,4), (3,5), (3,6), (3,7), (3,8)
(4,1), (4,2), (4,3), (4,4), (4,5), (4,6), (4,7), (4,8)
(5,1), (5,2), (5,3), (5,4), (5,5), (5,6), (5,7), (5,8)
(6,1), (6,2), (6,3), (6,4), (6,5), (6,6), (6,7), (6,8)

이 중심점들과 군집을 포함한 클래스를 바탕으로 LVQ-I의 연결 강도를 결정하여 패턴인식을 하면 그림11과 같다.

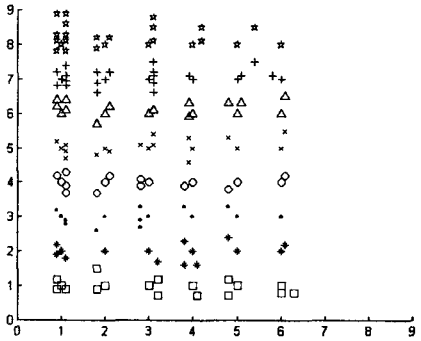


그림11. LVQ-I을 이용한 데이터-베이스와 패턴 인식
Figure 11. Data-Base and Pattern Recognition using LVQ-I

그림11에서 LVQ-I의 패턴 인식 성능을 확인할 수 있다. 많은 데이터를 입력으로 주었을 때 정확히 패턴 인식한 후 그림에서 각 클래스에 해당하는 기호를 통하여 출력을 만드는 것을 확인할 수 있다. 이러한 패턴 인식의 골격이 되는 LVQ-I의 원형 벡터만을 시뮬레이션 하면 그림12와 같다.

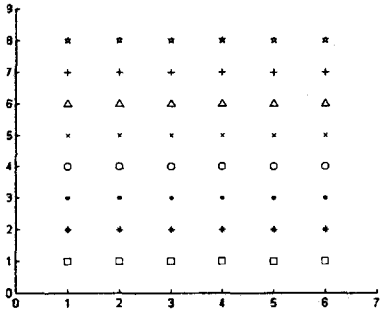


그림 12. LVQ-I의 원형 벡터
Figure 12. Prototype Vector of LVQ-I

그림12를 기반으로 지능 시스템을 구성하여 제어대상인 조명기에 접목시켜 20번의 학습을 시킨다. 여기서 LVQ-I에 의해 학습되는 LVQ-II와 관련된 식19에서 학습률 α는 0.01로 한다. 학습하는 LVQ-I의 출력은 그림13과 같으며 학습된 LVQ-II의 출력과 조명기의 출력은 그림14와 같다.

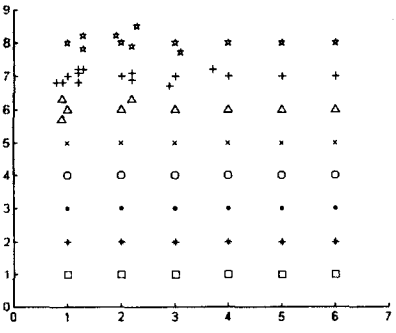


그림 13. 제안한 지능 시스템의 학습
Figure 13. Learning of the Proposed Intelligent System

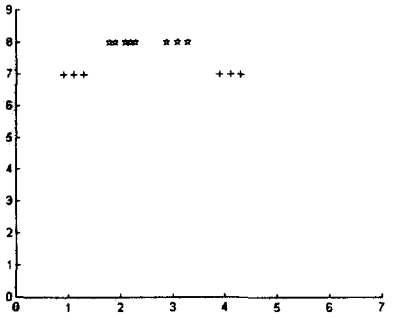


그림 14. 제안한 지능 시스템의 출력
Figure 14. Output of the Proposed Intelligent System

IV. 결론

본 논문에서는 제어 대상, 즉 플랜트를 사용하는 사용자가 여러 가지 변화하는 상황에 맞게 플랜트의 목표값을 지정하여야 하는 기존의 제어 방식과는 달리 각 상황에서 발생할 수 있는 모든 출력의 경우들 중에서 사용자가 가장 선호하는 출력을 만들 수 있도록 목표값을 자동으로 지정해주는 지능 알고리즘을 제안하였다. 제안한 지능 알고리즘은 신경 회로망을 이용하여 구성하며 각 상황마다 사용자가 원하는 출력을 학습함으로서 학습 기간이 종료되면 이후 각 상황마다 사용자가 원하던 출력을 자동으로 만들어낼 수 있는 기능을 갖도록 한 것이다. 신경 회로망은 두 개의 학습 벡터 양자화(Learning Vector Quantization), 즉 LVQ 네트워크로 구성하며 LVQ-I은 검출된 신호의 패턴을 인식하여 LVQ-II를 학습시키고 LVQ-II는 학습된 결과를 토대로 상황을 판단하는 역할을 한다.

제안한 지능형 제어 알고리즘의 유용성을 확인하기 위하여 플랜트로서 조도가 조절 가능한 조명기에 적용하여, 조명기의 주위의 임의의 상황에 대하여 사용자가 원하는 출력을 학습한 후 같은 상황에 주어진 여러 가지 목표값들 중에서 사용자가 원하는 출력을 자동으로 발생시키는지를 시뮬레이션 및 실험을 통하여 검토하였다.

검토 결과 제안한 지능 시스템은 학습한데로 주어진 상황에 적응하여 조명기의 출력을 자동으로 만들어 냄을 확인할 수 있었다.

향후 연구 방향은 조명기 뿐만 아니라, 에어컨, 자동차, 히터 등 주어지는 상황이 달라지면 새로운 입력을 설정해주어야 하는 기타 다른 장치에도 제안한 지능 시스템을 적용해 보는 것이다.

참고 문헌

[1] Hagan, Demuth, Beale, "Neural Network Design", PWS Publishing Company, 1995.

[2] Yong-Zai Lu, "Industrial Intelligent Control" Fundamentals and Applications, JOHN WILEY & SONS, 1996.

[3] B. D. Ripley, "Pattern Recognition and Neural Networks", CAMBRIDGE UNIVERSITY PRESS, 1996.

[4] Abhijit S. Pandya, Robert B. Macy, "Pattern Recognition with Neural Networks in C++", CRC PRESS, IEEE PRESS, 1995.

[5] N. K. Bose, P.Liang "Neural Network Fundamentals With Graphs, Agorithms, and Applications" ,McGraw-Hill, 1996.

[6] Madan M. Gupta, Naresh K. Sinha "Intelligent Control Systems", IEEE PRESS, 1995