論文2001-38CI-1-1

퍼셉트론 형태의 LVQ : LVQ의 일반화
(Perceptron-like LVQ : Generalization of LVQ)

宋 槿 培*, 李 幸 世*
(Geun Bae Song and Haing Sei Lee)

요 약

본 논문에서는 Hebb 학습법에 기초한 Kohonen의 LVQ 학습법을 퍼셉트론 학습에 사용되는 경도 강하(Gradient descent) 학습법에 의해 재해석한다. Kohonen의 LVQ는 학습법에 따라 두 가지로 나뉠 수 있는데 하나는 자율학습 LVQ(ULVQ)이며 다른 하나는 타율학습 LVQ(SLVQ)이다. 두 경우 모두 출력뉴런의 목표 값을 적당히 생성할 경우 타율학습 경도 강하학습법으로 표현될 수 있다. 결과적으로 LVQ학습법은 타율학습 경도 강하 학습법의 특수한 형태임을 알 수 있으며 또한 LVQ는 보다 일반화된 '퍼셉트론 형태의 LVQ(PLVQ)'알고리즘으로 표현될 수 있음을 알 수 있다. 본 논문에서는 이를 증명하고 결론을 맺는다.

Abstract

In this paper we reanalyze Kohonen's learning vector quantizing (LVQ) Learning rule which is based on Hebb's learning rule with a view to a gradient descent method. Kohonen's LVQ can be classified into two algorithms according to learning mode: unsupervised LVQ(ULVQ) and supervised LVQ(SLVQ). These two algorithms can be represented as gradient descent methods, if target values of output neurons are generated properly. As a result, we see that the LVQ learning method is a special case of a gradient descent method and also that LVQ is represented by a generalized percetron-like LVQ(PLVQ).

Ⅰ. 서 론

Kohonen의 LVQ(Learning Vector Quantizer)[1,2,3]는 신경망의 관점에서 단층구조이며 경쟁학습(Competitive learning)을 하며, Hebbian rule[4]에 기초한 Heuristic에 의해 학습한다. 이러한 LVQ는 학습법에 따라 크게 두 가지로 나뉠 수 있는데 하나는 'Unsupervised LVQ(ULVQ)'이며 다른 하나는 'Supervised LVQ(SLVQ)'이다. 두 경우 모두 입력벡터에 대한 출력은 각 뉴런의 가중치(Weight) 벡터와 입력 벡터간의 거리를 계산한 뒤 경쟁을 통해 승리한 뉴런만이 활성화된 출력(가령, 1)을 내는 형태로 정의된다. 즉, 구체적인 '출력 값'이 중요한 것이 아니라 활성화된 뉴런의 '위치'가 중요한 의미를 갖는다. 또한 가중치 벡터는 클러스터링 하려는 벡터공간에서의 Centroid 벡터로서의 의미를 가지게 된다. LVQ는 Hebbian Heuristic에 기초한 학습법으로 외부의 교사(목표 값) 없이 주어진 입력 벡터에 대해 승자(Winner) 뉴런을 강화시키고 패자(Loser) 뉴런을 약화 혹은 현상유지시키는 형태의 학습과정을 통해 주어진 공간에서의 최적의 Centroid 벡터, 즉 벡터분포의 평균점을 찾아가게 된다.

한편, 퍼셉트론[5,6,7]의 학습에 사용되는 타율학습 경도 강하(Gradient descent) 학습법은 목표 값이 주어져야 하며 이에 따라 목적함수(보통 Mean Square Error)가 정의되고 이 함수에의 최적화를 수행하는 타율학습 알고리즘이다. 따라서 퍼셉트론의 학습법과 과 LVQ 학습법은 서로 다른 이론적 배경에서 제안되었다. 그러나 본 논문에서는 퍼셉트론(혹은 퍼셉트론 형

* 正會員, 亞洲大學校 電子工學部
(Department of Electronics Engineering, Ajou University)
接受日字:2000年6月17日, 수정완료일:2000年12月13日

태의 단층 신경망)의 목표 값을 적당히 선정할 경우 퍼셉트론의 학습법은 LVQ의 학습법과 같게 되며, 그리하여 보다 일반화된 형태의 LVQ 즉, Perceptron-like LVQ(PLVQ)을 정의할 수 있음을 보여준다.

PLVQ를 정의함으로써 LVQ 설계의 융통성을 가질 수 있게 되는데 이를테면, 승자뿐만 아니라 패자의 학습에 대해서도 고려해 볼 수 있다는 자유도가 생겨나며 학습속도 개선이나 국부 최소(Local minimum) 문제를 극복하기 위해 그 동안 타율학습 경도 강하 학습법 분야에서 제안된 방법들의 사용 및 결과 예측이 가능해진다는 것이 그것이다.

2절에서는 Kohonen의 대표적인 두 가지 형태의 LVQ알고리즘과 퍼셉트론에 대해 복습하며 PLVQ알고리즘을 정의한다. 3절에서는 LVQ의 학습법을 타율학습 경도 강하 학습법에 의해 재해석하고 LVQ는 PLVQ의 특수한 형태임을 증명하며, 4절에서 결론을 맺는다.

II. Kohonen의 LVQ와 퍼셉트론에 대한 검토와 PLVQ의 정의

LVQ 혹은 단층 퍼셉트론의 신경망 구조는 그림 1과 같다. 이제 입력 벡터를 $X=\{x_1,x_2,\ldots,x_N\}^T\in R^N$ 가중치 벡터를 $W_i=\{w_{i1},w_{i2},\ldots,w_{iN}\}^T\in R^N$ 라 하고 출력 뉴런의 출력 값을 y_i라 하자.

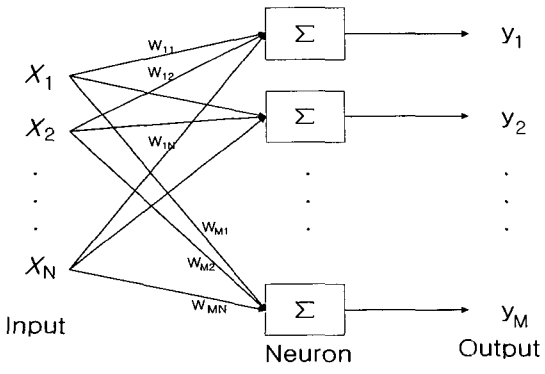

그림 1. 단층 신경망
Fig. 1. Silgle-layer Neural Network.

여기서 $i\in\{1,2,\ldots,M\}$는 출력 뉴런을 가리킨다. 그러면 대표적인 두 가지 형태의 LVQ와 퍼셉트론의 학습규칙은 다음과 같이 나타낼 수 있다.

1. Unsupervised LVQ(ULVQ, Kohonen 1981)

(1)뉴런 입출력 관계식(정의) : 원래의 ULVQ의 입출력은 아래와 같은 형태로 정의되어 있지 않다. 아래의 정의는 단지 PLVQ와의 설명의 일관성을 위해 설정한 것이다. 이는 SLVQ의 경우에 있어서도 마찬가지이다.

$$y_i(t) = \| X(t)-W_i(t) \|, \tag{1}$$

이후로 $\|\cdot\|$는 Euclidean Norm을 뜻하며 t는 학습 사이클을 가리킨다.

(2) 승자 결정(Winner decision) :

$$\|X(t)-W_c(t)\| = \min_i \{ \|X(t)-W_i(t)\| \}, \tag{2}$$

이후로 c는 승자 출력뉴런을 가리킨다.

(3) 학습 (ULVQ학습법) :

$$\begin{aligned} W_c(t+1) &= W_c(t)+\Delta^{ulvq}W_c(t) \\ &= W_c(t)+\alpha(t)[X(t)-W_c(t)] \\ &\quad (0<\alpha(t)<1) \end{aligned} \tag{3}$$

$$W_i(t+1)=W_i(t), \quad \text{그 밖의 } i\neq c\ . \tag{4}$$

2. Supervised LVQ(SLVQ, Kohonen 1990)

학습벡터들은 각각 소속된 Class의 표식 $k\in\{1,2,\ldots,C\}$를 가진다. 여기서 C는 Class의 수를 뜻한다. 그리고 출력 뉴런 역시 학습벡터와 마찬가지로 사전에 이들 Class들 중 하나로 지정된다는 것이 중요하다. 학습과정은 다음과 같다.

(1) 뉴런 입출력 관계식(정의) :

$$y_i(t) = \| X^k(t)-W_i(t) \|, \tag{5}$$

여기서 k는 $X(t)$의 Class를 나타낸다.

(2) 승자 결정(Winner decision) :

$$\|X^k(t)-W_c(t)\| = \min_i \{ \|X^k(t)-W_i(t)\| \} \tag{6}$$

(3) 학습 (SLVQ학습법) :

만약 $X(t)$의 Class k와 승자 뉴런 c의 Class가 같다면,

$$\begin{aligned} W_c(t+1) &= W_c(t)+\Delta^{slvq}W_c(t) \\ &= W_c(t)+\alpha(t)[X^k(t)-W_c(t)] \\ &\quad (0<\alpha(t)<1) \end{aligned} \tag{7}$$

만약 $X(t)$의 Class k와 승자 뉴런 c의 Class가 다르다면,

$$W_c(t+1) = W_c(t) + \Delta^{shrq} W_c(t)$$
$$= W_c(t) - \alpha(t)[X^k(t) - W_c(t)] \quad (8)$$
$$(0 < \alpha(t) < 1)$$

그 밖의 $i \neq c$ 에 대해

$$W_i(t+1) = W_i(t).$$

3. 퍼셉트론

그림 2는 퍼셉트론의 뉴런 하나의 모델을 나타낸다. 여기서 b_i는 bias항을 $f(\cdot)$는 활성함수를 나타낸다. 퍼셉트론의 입출력 관계는 입력 벡터와 가중치 벡터와의 내적을 구한 뒤 여기에 활성함수를 가한 형태로 정의된다. 퍼셉트론의 활성함수로는 계단 함수, 선형 1차 함수, 시그모이드 함수 등 다양한 종류의 함수가 사용될 수 있다.

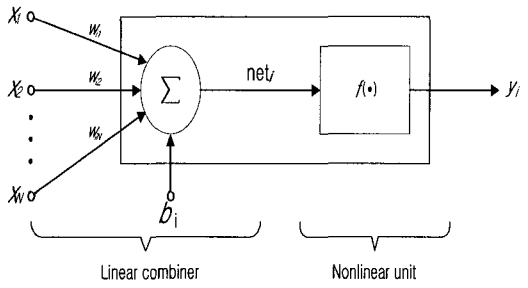

그림 2. 뉴런 하나의 모델
Fig. 2. A model of a neuron.

퍼셉트론은 각각의 입력에 대해 대응하는 목표 값이 주어지며 실제 출력이 이 목표 값에 근접하도록 출력하도록 학습 받는다. 퍼셉트론의 학습규칙인 경도 강하 학습법은 만약 해가 존재할 경우 유한한 시간 안에 해에 수렴한다는 것이 입증되었다[9]. 퍼셉트론의 학습규칙은 다음과 같이 요약할 수 있다.

(1) 뉴런 입출력 관계식 :

$$\text{net}_i(t) = \sum_{j=1}^{N} w_{ij}(t) x_j(t) + b_i$$
$$= W_i^T(t) X(t) + b_i$$
$$y_i(t) = f(\text{net}_i) \quad (9)$$

(2) 학습(경도 강하 학습법) :

$$W_i(t+1) = W_i(t) + \Delta^{\text{gradient}} W_i(t)$$
$$= W_i(t) + [-\alpha(t) \nabla \varepsilon_i]$$
$$= W_i(t) + [\alpha(t)\ (t_i(t) - y_i(t))\, f'(\text{net}_i)\, X(t)]$$
$$i \in \{1, 2, \ldots, M\},\quad (0 < \alpha(t) < 1) \quad (10)$$

여기서 t_i는 i번째 출력 뉴런의 목표 값을 나타내며 $\varepsilon_i \equiv \frac{1}{2}(t_i - y_i)^2$은 목표 값과 실제 출력 사이의 오차함수를, $\nabla \varepsilon_i$는 ε_i를 가중치 벡터 W_i 방향으로 Gradient 연산하는 것을 나타낸다. Chain rule을 사용하여 $\nabla \varepsilon_i$를 계산하면 다음과 같다.

$$\nabla \varepsilon_i = \frac{1}{2} \frac{\partial \varepsilon_i}{\partial W_i} = \frac{1}{2} \frac{\partial \varepsilon_i}{\partial y_i} \frac{\partial y_i}{\partial \text{net}_i} \frac{\partial \text{net}_i}{\partial W_i}$$
$$= -(t_i(t) - y_i(t)) f'(\text{net}_i) X(t) \quad (11)$$

여기서 $f'(\text{net}_i) \equiv \frac{\partial y_i}{\partial \text{net}_i}$을 나타낸다.

따라서 증분항은

$$\Delta^{\text{gradient}} W_i(t) = -\alpha(t) \nabla \varepsilon_i$$
$$= \alpha(t)(t_i(t) - y_i(t)) f'(\text{net}_i) X(t) \quad (12)$$

가 된다.

4. Perceptron-like LVQ

그림 3은 본 논문에서 제안하는 PLVQ의 개략도를 나타내고 있다. $\|W_i - X\|$이 표시된 블록은 그림1의 신경망 구조에 대응되는 블록이다. PLVQ는 앞 절에서 설명된 퍼셉트론과 마찬가지로 경도 강하 학습법을 사용한다. 그러나 PLVQ는 앞 절의 퍼셉트론에 대해 다음 사항이 수정이 되었다. 이 점을 4.1절에서 언급하며 4.2절에서는 PLVQ의 학습 알고리즘에 대해서 언급한다.

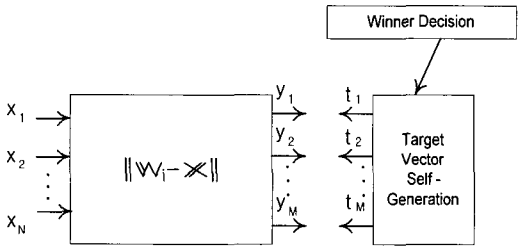

그림 3. 제안된 퍼셉트론 형태의 LVQ
Fig. 3. The proposed Perceptron-like LVQ.

(1) PLVQ와 표준 퍼셉트론의 차이점

① 뉴런 입출력 관계식

뉴런의 출력은 가중 합(Weighted sum)의 형태

$$y_i(t)=f(\sum_{j=1}^{N} w_{ij}(t)x_j(t))=f(W_i^T(t)X(t))$$

에서 Euclidean Norm의 형태로 바뀐다. 즉,

$$y_i(t) = \| X(t)-W_i(t) \|.$$

Euclidean Norm은 연속적으로 미분 가능한 함수이므로 뉴런 출력을 이와 같이 정의할 때 경도 강하 방법을 적용하기 용이하다.(이와 유사한 형태의 입출력을 정의하는 신경망으로 RBF신경망[8]이 있다. 그러나 RBF에서는 PLVQ에서 다루지 않는 활성함수가 중요한 의미를 가진다).

② 경쟁학습의 도입

LVQ와 마찬가지로 경쟁학습 방법을 도입한다. 즉, 출력 단에 승자 결정 블록을 둔다. 그리하여 목표패턴은 기존의 퍼셉트론과 달리 각 입력벡터에 대해 사전에 정해지지 않고 일단 승자 뉴런이 결정된 뒤 그에 따라 적절한 규칙에 의해 생성되게 된다.

③ 활성함수

활성함수는 사용하지 않는다(혹은 선형 함수를 사용한다). 이는 위 1번의 조건에 함축되어 있다.

④ Bias 항

퍼셉트론에서 일반적으로 사용되는 Bias 항은 사용하지 않는다. 이는 PLVQ의 가중치 벡터의 의미가 패턴공간의 경계선과 관련된 파라미터가 아니라 LVQ와 마찬가로 클러스터 Centroid 벡터의 의미를 가지므로 불필요한 항이다.

(2) PLVQ 학습 알고리즘

① 뉴런 입출력 관계식 :

$$y_i(t) = \| X(t)-W_i(t) \|.$$

② 승자 결정(Winner decision) :

$$\|X(t)-W_c(t)\| = \min_i \{ \|X(t)-W_i(t)\| \},$$

여기서 c는 승자 출력뉴런을 가리킨다.

③ 학습(PLVQ 학습법) :

아래는 경도 강하 방법에 의해 PLVQ 학습법을 유도한 결과이다. 이 학습법은 PLVQ가 ULVQ를 구현하느냐 SLVQ를 구현하느냐에 따라 단지 각 뉴런에 부여되는 목표 값 t_i만 달리하면 되므로 두 경우 모두에 적용될 수 있는 학습법이다. 아래 식의 유도는 Ⅲ절에서 하기로 한다.

$$\begin{aligned} W_i(t+1) &= W_i(t)+\Delta^{plvq}W_i(t) \\ &= W_i(t) + [-\alpha(t)\nabla\varepsilon_i] \\ &= W_i(t) + [-\alpha(t)(\frac{t_i}{y_i}-1)[X-W_i]] \\ & i\in\{1,2,\ldots,M\},\ (0<\alpha(t)<1) \end{aligned} \quad (13)$$

Ⅲ. PLVQ 와 LVQ의 관계

1. PLVQ와 ULVQ

정리 1 : PLVQ는 승자 뉴런 c의 목표 값을 '0'으로 하고 나머지 뉴런(패자)의 목표 값은 자기 자신의 출력 값 $y_i(i\neq c)$로 하였을 경우 ULVQ와 같아진다.

증명 : PLVQ와 ULVQ의 신경망 구조는 동일하므로 가중치에 대한 두 신경망의 학습법이 같음을 보이기만 하면 된다. 먼저 경도 강하 방법에 의해 PLVQ의 증분 항 $\Delta^{plvq}W_i(t)$을 유도한 뒤 정리에서 제시된 목표 값이 주어질 경우 $\Delta^{plvq}W_i(t)=\Delta^{ulvq}W_i(t)$로 귀결됨을 보인다.(식을 간단히 하기 위해 미분 식은 벡터 형태로 나타내고 학습 사이클 t는 생략하기로 한다).

위 PLVQ의 입력벡터를 X, 뉴런의 출력을 y_i라 하면 입출력 관계식은 아래와 같다.

$$y_i=\|X-W_i\|,\quad i\in\{1,2,\ldots,M\}$$

각 출력 노드 i에서의 목표 값을 t_i라하고, 오차함수 ε_i는 다음과 같이 정의하기로 한다.

$$\varepsilon_i = \frac{1}{2}(t_i-y_i)^2 = \frac{1}{2}(t_i-\|X-W_i\|)^2 \quad (14)$$

이 오차함수에 대해 벡터 W_i 방향으로의 Gradient $\nabla\varepsilon_i$를 계산하면

$$\begin{aligned} \nabla\varepsilon_i &= \frac{1}{2}\frac{\partial}{\partial W_i}(t_i-\|X-W_i\|)^2 \\ &= \frac{1}{2}\frac{\partial}{\partial W_i}(t_i^2-2t_i\|X-W_i\|+\|X-W_i\|^2) \\ &= (\frac{t_i}{\|X-W_i\|}-1)[X-W_i] \\ &= (\frac{t_i}{y_i}-1)[X-W_i]. \end{aligned} \quad (15)$$

따라서 각 뉴런의 가중치 벡터들을 학습시키기 위한 증분 항은 다음과 같이 된다.

$$\Delta^{plvq}W_i = -\alpha\nabla\varepsilon_i = -\alpha(\frac{t_i}{y_i}-1)[X-W_i]. \quad (16)$$

이때 승자 뉴런의 목표 값 t_c을 0으로 주면 승자 뉴런의 가중치 벡터의 증분 항은

$$\Delta^{plvq}W_c = -\alpha(\frac{0}{y_c}-1)[X-W_c]$$
$$= \alpha[X-W_c] = \Delta^{ulvq}W_c \text{ 가 된다.}$$

또한 이 식에서 패자들의 목표 값 $t_i(i\neq c)$를 그 뉴런의 출력 값 y_i로 할 경우 $\Delta^{plvq}W_i=0=\Delta^{ulvq}W_i$ 이 됨을 쉽게 알 수 있다. ■

<검토> ULVQ 학습법을 살펴보면 승자의 가중치 벡터를 입력벡터 쪽으로 $\Delta^{ulvq}W_c=\alpha[X-W_c]$양만큼 이동시키는 형태이다. 즉, 승자의 가중치 벡터가 입력벡터에 가까워지도록 한다. 그러면 승자의 출력 $y_c=\|X-W_c\|$가 0에 가까워질 것이다. 경도 강하 학습법 관점에서, 승자의 목표 값 t_c를 $0\leq t_c<y_c$의 어떤 값으로 줄 경우 위 학습법과 동등한 효과를 가져올 수 있다. 특별히 t_c를 0으로 줄 경우 승자 뉴런의 학습에 있어서 PLVQ와 ULVQ는 정확히 일치하게 된다. 마찬가지로, ULVQ에서 패자들의 가중치 벡터를 변화시키지 않는다는 것(즉, $\Delta^{ulvq}W_i=0,\ i\neq c$)은 경도 강하 학습법 관점에서 패자들의 목표 값 $t_i(i\neq c)$를 전의 출력 y_i로 주는 것과 동등하다. 이와 같이 각 뉴런의 목표 값을 정해준 결과 PLVQ 학습법은 ULVQ 학습법과 같아진다. 따라서 주어진 조건에서 PLVQ는 ULVQ가 됨을 알 수 있다.

2. PLVQ와 SLVQ

정리 2 : PLVQ는 승자 뉴런 c의 목표 값을 패턴분류(Classification)가 옳으면 '0', 틀리면 '$2y_c$'로 하고 나머지 뉴런의 목표 값은 자기 자신의 출력 값 $y_i(i\neq c)$로 하였을 경우 SLVQ와 같아진다.

증명 : 이 경우 역시 정리 1의 증명에서처럼 경도 강하 방법에 의해 PLVQ의 증분 항 $\Delta^{plvq}W_i(t)$을 유도한 뒤 정리에서 제시된 목표 값이 주어질 경우 $\Delta^{plvq}W_i(t)=\Delta^{slvq}W_i(t)$로 귀결됨을 보이면 된다.

입출력 관계식은 위 정리 1의 경우와 같으며 오차함수, Gradient 연산, 증분 항 또한 마찬가지다. 즉,

$$\Delta^{plvq}W_i = -\alpha\nabla\varepsilon_i$$
$$= -\alpha(\frac{t_i}{y_i}-1)[X^k-W_i] \text{ 이다.}$$

여기서 입력 벡터 X의 윗첨자 k는 벡터 X가 소속된 Class를 가리킨다.

만약 $X(t)$의 Class k와 승자 뉴런 c의 Class가 같다면, 승자 뉴런의 목표 값 t_c을 0으로 준다. 그러면 승자 뉴런의 가중치 벡터의 증분 항은

$$\Delta^{plvq}W_c = -\alpha(\frac{0}{y_c}-1)[X^k-W_c]$$
$$= \alpha[X^k-W_c] = \Delta^{slvq}W_c \text{ 가 된다.}$$

다른 한편, 만약 $X(t)$의 Class k와 승자 뉴런 c의 Class가 다르다면, 승자 뉴런의 목표 값 t_c을 $2y_c$로 준다. 그러면 승자 뉴런의 가중치 벡터의 증분 항은

$$\Delta^{plvq}W_c = -\alpha(\frac{2y_c}{y_c}-1)[X^k-W_c]$$
$$= -\alpha[X^k-W_c] = \Delta^{slvq}W_c \text{ 가 된다.}$$

마지막으로, 이 식에서 패자들의 목표 값 $t_i(i\neq c)$를 그 뉴런의 출력 값 y_i로 할 경우 $\Delta^{plvq}W_i=0=\Delta^{slvq}W_i$ 이 됨을 쉽게 알 수 있다. ■

<검토> SLVQ 학습법을 살펴보면, 패턴분류가 옳으면 승자의 가중치 벡터를 입력 벡터 쪽으로 $\Delta^{slvq}W_c=\alpha[X^k-W_c]$양만큼 이동시킨다.(이는 정리 1의 ULVQ의 경우와 같다.) 반면에 패턴분류가 틀리면 입력 벡터에서 멀어지는 쪽으로 $\Delta^{slvq}W_c=-\alpha[X^k-W_c]$양만큼 이동시킨다. 이러한 학습의 결과 패턴분류의 옳고 그름에 따라 같은 입력 X에 대해 승자의 출력 $y_c=\|X-W_c\|$는 학습 전보다 작아지거나 커지거나 하게된다. 경도 강하 학습법 관점에서, 위의 학습법은 패턴분류가 옳으면 승자 뉴런의 목표 값 t_c를 현재 출력 y_c보다 작은 값 ($0\leq t_c<y_c$)으로, 패턴분류가 틀렸으면 t_c를 현재 출력 y_c보다 큰 값 ($y_c<t_c<\infty$)으로 지시하는 경우와 동등하다. 특별히, 패턴분류가 옳으면 $t_c=0$, 그르면 $t_c=2y_c$로 정해줄 경우 승자에 대한 PLVQ 학

습법은 SLVQ의 학습법과 정확히 일치하게 된다. 패자들에 대한 목표 값의 논의는 위 정리 1의 <검토>에서의 논의와 같다. 이와 같이 각 뉴런의 목표 값을 정해준 결과 PLVQ 학습법은 SLVQ 학습법과 같아진다. 따라서 주어진 조건에서 PLVQ는 SLVQ가 됨을 알 수 있다.

Ⅳ. 결 론

본 논문에서는 Kohonen LVQ 학습법을 타율학습 경도 강하 학습법에 의해 재해석하였다. 결과적으로 이렇게 해서 유도된 새로운 PLVQ알고리즘은 Kohonen LVQ의 일반화된 알고리즘이 됨을 보였다. 즉, PLVQ의 출력뉴런에 적당한 목표 값 부여할 경우 PLVQ는 Kohonen의 LVQ로 귀결된다.

PLVQ가 LVQ의 일반화된 신경망이라는 관점을 얻음으로써 LVQ 설계의 보다 큰 자유도를 얻을 수 있다. 즉, 수렴속도 개선이나 국부 최소 문제를 극복하기 위해 다른 형태의 목표패턴을 고려를 해볼 수 있으며 또한 이들 문제들을 해결하기 위해 그 동안 타율학습 경도 강하 학습법 분야에서 제안된 방법들의 사용 및 결과예측이 가능해진다. 이는 본 논문의 향후 연구과제가 될 것이다.

참 고 문 헌

[1] T. Kohonen, Self-Organizing Maps, Springer-verlag, 1995.

[2] T. Kohonen, "Learning vector quantization for pattern recognition," Rep. TKK-F-A601, Helsinki Univ. Technol., Dept. Techn.Phys., Lab. Computer an Inform. Sci., 1986.

[3] T. Kohonen, "versions of learning vector quantization," in Proc. IEEE int. Joint Conf. on Neur. Networks, vol. 1, pp. 223-228, San Diego, CA, June 1990.

[4] D. O. Hebb, Organization of Behavior, New York : Science Editions, 1949.

[5] R. Rosenblatt, Principles of Neuro-dynamics, New York : Spartan Books, 1959.

[6] D. Rumelhart, J. McClelland and the PDP Research Group, Parallel Distributed Processing: Explorations in the Micro-structure of Cognition, MIT Press, Cambridge, MA, Vol. 1, 1986.

[7] R. P. Lippman, "An introduction to computing with neural nets," IEEE ASSP Magazine, vol. ASSP-38, vol. 4, pp. 4-22, April 1987'.

[8] X. M. Song, "A radial basis function network for empirical modeling of soil extraction process," in Proc. of EANN'95 Conf., 1995.

[9] B. Widrow, "sampled-data systems, a statistical theory of adaptation," 1959 IRE WESCON Convention Record, part 4, New York: Institute of Radio Engineers, 1959.

저 자 소 개

宋 權 培(正會員)

1992년 아주대학교 전자공학과(학사). 1995년 아주대학교 전자공학과 대학원(석사). 1995년~1996년 현대전자 소프트웨어 연구소 근무. 1996년~현재 아주대학교 전자공학과 박사과정. 주관심 분야는 음성인식, 인공지능 및 신경망

李 幸 世(正會員)

1966년 전북대학교 전기공학과(학사). 1972년 서울대학교 전자공학과 대학원(석사). 1984년 고려대학교 전자공학과(박사). 1968~1970년 해군사관학교 교관. 1982~1983년 Columbia Univ.n.y., 객원교수. 1987~1988년 INRIA PARIS 객원교수. 1992~1994년 거제전문대학장. 1973~현재 아주대학교 교수. 주관심분야는 음성인식, 문자 및 지도 인식, 인공지능 및 신경망, 퍼지 시스템 및 카오스 이론