한국지능정보시스템학회논문지 제10권 제2호 2004년 11월(pp.53-63)

순환 케스케이드 코릴레이션 알고리즘의 일반화와 새로운 활성화함수를 사용한 모스 신호 실험

이상화
서원대학교 컴퓨터정보통신공학부
(swl@seowon.ac.kr)

송해상
서원대학교 컴퓨터정보통신공학부
(hssong@seowon.ac.kr)

순환 케스케이드 코릴레이션(Recurrent Cascade Correlation(RCC))은 감독에 의하여 학습하는 알고리즘이고 네트워크의 크기와 형태는 자동으로 이루어진다. RCC는 새로운 은닉뉴런들이 한 층에 하나씩 순서대로 네트워크에 삽입되기 때문에 다층구조를 형성하고 2계(Second Order) RCC는 새로운 은닉뉴런들이 한 층에만 순서대로 생성되어 나열되므로 2층 구조를 형성한다. 따라서 이러한 은닉뉴런들끼리는 서로 연결하지 않는다. 이 논문에서는 RCC와 2계 RCC의 조합을 통한 RCC 네트워크의 일반화를 소개한다. 새로운 RCC 알고리즘은 은닉뉴런이 네트워크에 첨가될 때마다 네트워크가 수직성장 또는 수평성장을 해야 하는지를 스스로 결정한다. 또한 뉴런의 활성화를 위한 새로운 활성화함수를 소개하고 기존의 sigmoid, tanh 함수와 함께 사용하여 모스 벤치마크 문제에 관하여 실험하였다. 이러한 활성화 함수들을 사용한 RCC 네트워크의 일반화 실험에서 은닉뉴런의 숫자가 감소하였음을 알 수 있다.

논문접수일 : 2004년 4월 게재확정일 : 2004년 11월 교신저자 : 이상화

1. 서론

케스케이드 코릴레이션(Cascade Correlation (CC)) 네트워크[Fahlman, 1990]는 학습중에 스스로 네트워크를 형성하므로 문제해결을 위하여 전문가가 미리 네트워크를 설계할 필요가 없는 특징을 갖고 있다. 초기 CC 네트워크는 은닉뉴런과 은닉층 없이 다만 입력뉴런과 출력뉴런의 완전한 연결에 의하여 구성된다. CC 학습알고리즘에서 은닉뉴런들은 네트워크에 한 번에 한 개씩 추가되고 선택된 가중치의 값은 변화하지 않는다. 여기에서 추가할 뉴런의 출력과 네트워크의 잔여오차의 상호관계값(correlation value)의 극대화를 시도한다. 새로운 한 은닉뉴런의 생성을 위하여, 후보뉴런(candidate unit)이 학습할 수 있는 시그널은 네트워크의 모든 입력과 이미 존재하고 있는 모든 은닉뉴런들과의 연결들을 통하여 전달된다. 여기에서 이 후보뉴런의 출력은 아직 네트워크와는 연결되지 않은 상태이다. 네트워크는 주어진 패턴들에 대하여 한 번의 학습 후에 후보뉴런들의 입력가중치(input weight)를 수정한다. S_i의 극대화를 위한 수정은 다음과 같이 정의 한다: S_i는 한 후보뉴런 i에서 생성된 값 c와 모든 출력뉴런 o에서 측정된 잔여출력오차 E_o의 상호관계(correlation)의 합계로 나타낸다.

$$S_i = \sum_o \left| \sum_p (c_{pi} - \bar{c}_i)(E_{po} - \bar{E}_o) \right| \qquad (1)$$

식 (1)에서 E_{po}는 패턴 p에 대한 출력뉴런 o에서의 오차이고 c_{pi}는 패턴 p에 대한 후보뉴런 i의 출력이다. $\bar{c}_i$는 모든 패턴들에 대한 후보뉴런 i의 평균출력이고 $\bar{E}_o$는 모든 패턴들에 대한 출력뉴런 o에서의 평균오차이다.

후보뉴런 i의 입력을 위한 연결에서의 가중치 w_{ij}에 대한 S_i의 편미분의 유도식은 다음과 같다:

$$\frac{\partial S_i}{\partial w_{ij}} = \sum_o \frac{\partial}{\partial w_{ij}} \left| \sum_p (c_{pi} - \bar{c}_i)(E_{po} - \bar{E}_o) \right|$$

$$= \sum_o \left(sign\left[\sum_p (c_{pi} - \bar{c}_i)(E_{po} - \bar{E}_o) \right] \sum_p \frac{\partial}{\partial w_{ij}} \left[(c_{pi} - \bar{c}_i)(E_{po} - \bar{E}_o) \right] \right) \qquad (2)$$

$$= \sum_o \left(sign\left(\sum_p (c_{pi} - \bar{c}_i)(E_{po} - \bar{E}_o) \right) \sum_p \frac{\partial c_{pi}}{\partial w_{ij}} (E_{po} - \bar{E}_o) \right)$$

σ_o를 다음과 같이 정의하면

$$\sigma_o = sign\left(\sum_p (c_{pi} - \bar{c}_i)(E_{po} - \bar{E}_o) \right)$$

식 (2)는 다음과 같이 쓸 수 있다.

$$\frac{\partial S_i}{\partial w_{ij}} = \sum_o \sum_p \sigma_o \frac{\partial}{\partial net_{pi}} f_p(net_{pi}) \frac{\partial net_{pi}}{\partial w_{ij}} (E_{po} - \bar{E}_o)$$

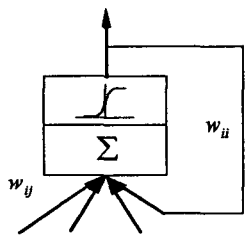
따라서 S_i의 편미분은 다음과 같다:

$$\frac{\partial S_i}{\partial w_{ij}} = \sum_o \sum_p \sigma_o f'_p \, I_{pi} (E_{po} - \bar{E}_o) \qquad (3)$$

식 (3)에서 σ_o는 후보뉴런 i에서 패턴 p에 대한 출력c_{pi}와 출력뉴런 o에서 패턴 p에 대한 오차 E_{po}와의 상호관계 값의 부호를 나타낸다. f'_p는 net_{pi}에 대한 후보뉴런 i의 활성화함수의 미분이고 I_{pi}는 패턴 p에 대한 후보뉴런 i의 입력을 의미한다.

S_i의 극대화를 위한 후보뉴런의 학습과 오차수정을 위한 출력뉴런의 학습은 quickpropagation 알고리즘[Fahlman, 1989]을 이용한다.

RCC 네트워크에서의 학습은 CC 네트워크에서의 학습과 동일하고, 다만 후보뉴런이 생성될 때마다 [그림 1]과 같이 부가적으로 순환 연결이 이루어져서 출력 값이 다시 뉴런의 입력 가중치로 사용된다.



[그림 1] 순환 연결 가중치를 갖고 있는 후보뉴런

따라서 후보뉴런 i의 출력은 다음과 같이 계산되어진다.

$$o_i^{(k)}(t) = {}_{act}\left(\sum_j o_j(t) w_{ij}^{(k)} + o_i^{(k)}(t-1) w_{ii}^{(k)} \right)$$

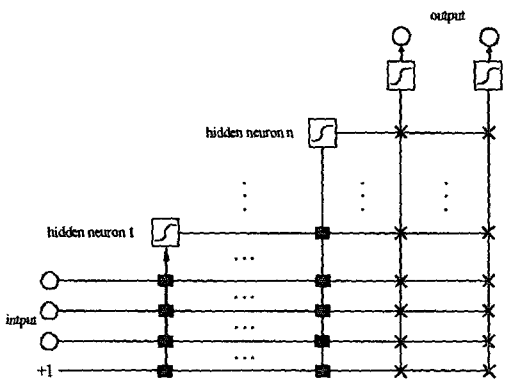
여기에서 가중치 $w_{ij}^{(k)}$와 $w_{ii}^{(k)}$에 따른 출력의 미분은 다음과 같다.

$$\frac{\partial}{\partial w_{ij}^{(k)}} o_i^{(k)}(t) = f'_{act}(net_i) \left(o_j(t) + \frac{\partial}{\partial w_{ij}^{(k)}} o_i^{(k)}(t-1) w_{ii}^{(k)} \right) \qquad (i \neq j)$$

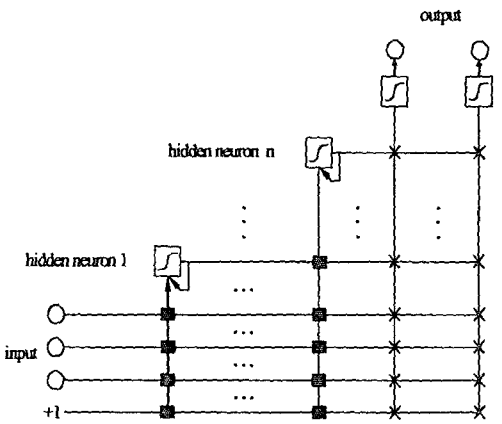
$$\frac{\partial}{\partial w_{ii}^{(k)}} o_i^{(k)}(t) = f'_{act}(net_i)\left(o_i^{(k)}(t-1) + \frac{\partial}{\partial w_{ii}^{(k)}} o_i^{(k)}(t-1) w_{ii}^{(k)}\right)$$

$\partial o_i^{(k)}(t-1)/\partial w_{ij}^{(k)}$ 의 값은 이전의 학습에서 생성된 것이 현재의 학습에서 다시 사용된다. 따라서 RCC 네트워크에서는 뉴런의 활성화 값 $o_i^{(k)}(t-1)$ 과 가중치 값들의 저장이 부가적으로 필요하다.

[그림 2] (a)는 CC 네트워크를 [그림 2] (b)는 RCC 네트워크를 보여준다.



(a)



(b)

[그림 2] (a): CC의 설계, (b) RCC의 설계

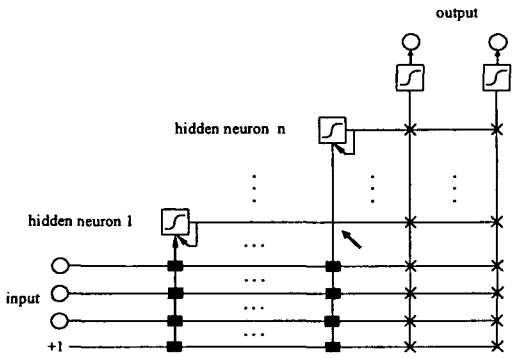
CC 및 RCC 알고리즘의 특성:

1) 알고리즘은 네트워크의 크기와 형태(즉 은닉층과 은닉뉴런의 수 그리고 그에 따른 연결)를 스스로 형성한다.
2) 학습과정에서 각 뉴런들의 연결을 통해 시그널은 전진전달(forward propagation)만 하므로 오차의 후진전달(backwards propagation)을 필요로 하지 않는다.
3) 여러 후보뉴런들이 풀(pool)을 형성하여 병렬로 학습할 수 있다.
4) 알고리즘은 오직 네트워크의 한 층과 관계된 가중치들, 즉 한 그룹에 속한 후보뉴런들과 학습한다. 네트워크의 나머지 부분은 변경되지 않는다. 그에 따른 뉴런의 활성화 값과 네트워크의 오차 값은 각 학습단계마다 저장되어 다시 사용 된다.
5) 새로운 뉴런은 각 층에 한 뉴런만 첨가되므로 학습과정 동안 매우 깊은 네트워크를 형성하고 이는 은닉뉴런의 높은 fan-in의 요인이 된다.
6) 알고리즘은 증가학습(incremental learning)에 속하므로 이미 학습한 네트워크가 새로운 패턴들과 함께 다시 학습할 수 있다.

2. 관련연구

RCC 네트워크의 수정된 모델로서 2계 RCC 알고리즘(Second order Recurrent Cascade Correlation Algorithm) (Fahlman, 1991)은 네트워크의 수직성장이 아닌 수평성장으로서 특징된다. 은닉뉴런이 생성되어 네트워크에 첨가될 때 오직 한 은닉층에만 나열되므로 이 은닉뉴런들의 입력은 네트워크의 입력뉴런들과 바이어스로부터 받게 되고 이미 생성된 다른 은닉뉴런들로 부

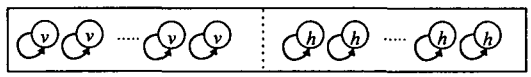
터는 아무런 영향도 받지 않는다. 이 경우에 네트워크는 2층 퍼셉트론(two-layer perceptron)과 같다. [그림 3]에서는 2계 RCC 네트워크를 묘사했다. 여기에서 굵은 화살표는 은닉뉴런들 사이에서 연결이 안 된 상태를 표시하고 또한 사각형으로 표시된 연결들은 고정되고(한번 선택된 가중치는 다음 학습단계에서도 수정 없이 계속 사용된다), x와 함께 표시한 연결들은 반복해서 학습한다(각 학습단계마다 가중치는 다시 수정된다).



[그림 3] 2계 RCC의 설계

3. 제안된 방법

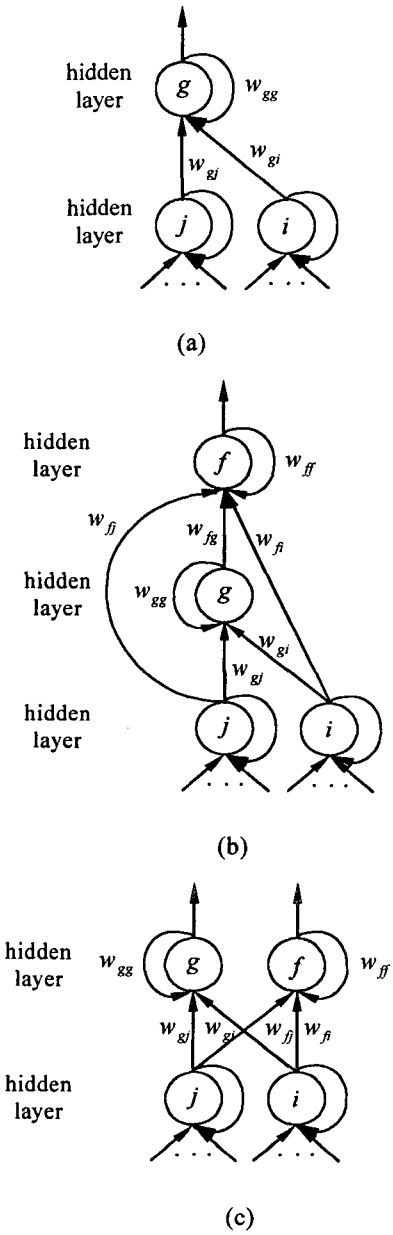
RCC 네트워크의 일반화를 위한 새로운 학습알고리즘을 소개한다. 이 알고리즘은 RCC 네트워크와 2계 RCC 네트워크의 조합을 통하여 이루어지고, 이를 위하여 풀(pool)에 있는 후보뉴런들을 두 그룹으로 나눈다. 첫 번째 그룹에 있는 후보뉴런들은 모든 입력뉴런과, 바이어스와 그리고 이미 생성된 모든 은닉뉴런과 연결한다. 두 번째 그룹에 있는 후보뉴런들은 모든 입력뉴런과, 바이어스와 그리고 같은 층에 있지 않은 모든 은닉뉴런과 연결한다. 기회균등의 원칙에 의하여 두 구룹은 같은 수의 후보뉴런들로 구성되어야 한다. [그림 4]는 한 예로서 각 그룹에 네개의 후보뉴런들이 나열된 상태를 보여준다. 여기에서 v로 표시한 후보뉴런은 수직 성장을 위하여 그리고 h로 표시한 후보뉴런은 수평성장을 위하여 학습한다.



[그림 4] 한 풀에 정렬되어 있는 후보뉴런들

같은 그룹에 있는 모든 은닉뉴런들은 같은 입력시그널을 받으며 각 패턴들로부터 원인이 된 같은 잔여오차를 갖고 있다. 그들은 서로 아무런 영향을 주고받지 않고 또한 학습하는 동안 네트워크에도 영향을 주지 않기 때문에 풀에서 병렬로 학습한다. 더 이상의 학습이 진행되지 않을 때에는, 즉 뉴런의 학습 횟수가 주어진 파라미터 값에 도달했을 때 가장 큰 상호관계 값(correlation score)을 갖고 있는 한 후보뉴런이 은닉뉴런으로 선택되어 네트워크에 첨가되고 다른 뉴런들과 연결된다. 그때마다 네트워크의 성장이 결정된다. 만약 후보뉴런이 첫 번째 그룹에서 선택되었다면 네트워크는 수직성장을 하고 두 번째 그룹에서 선택되었다면 네트워크는 수평성장을 한다.

[그림 5]는 수직성장 또는 수평성장의 학습을 위한 후보뉴런 k의 입력을 보여주기 위한 예이다. 여기에서 j, i, g는 이미 생성된 은닉뉴런을 나타낸다. 수직성장을 위해 뉴런 k는 뉴런 j, i 그리고 g로 부터 입력시그널을 받고 수평성장을 위해서는 뉴런 j와 i로부터 입력시그널을 받지만 뉴런 g로 부터는 아무런 영향도 받지 않는다.



[그림 5] (a): 이전의 학습단계에서 이미 생성된 세 은닉뉴런 *j*, *i*, *g*. 현재 학습단계에서 (b): 네트워크의 수직성장을 위한 후보뉴런 *k*의 입력 (c): 네트워크의 수평성장을 위한 후보뉴런 *k*의 입력 (이 그림에서는 은닉뉴런들과의 연결만 표시하였음)

수직성장과 수평성장을 위한 뉴런 *k*의 출력은 다음과 같다:

수직성장을 위한 후보뉴런 *k*의 출력:

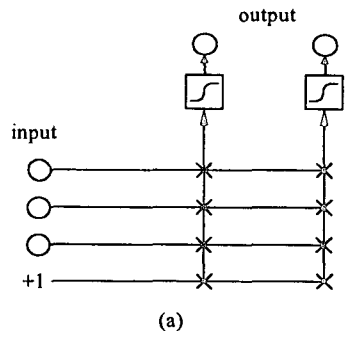
$$o_f^{(k)}(t) = f_{act}\left(\sum_{l=j}^{g} o_l^{(k)}(t) w_{fl}^{(k)} + o_f^{(k)}(t-1) w_{ff}^{(k)}\right)$$

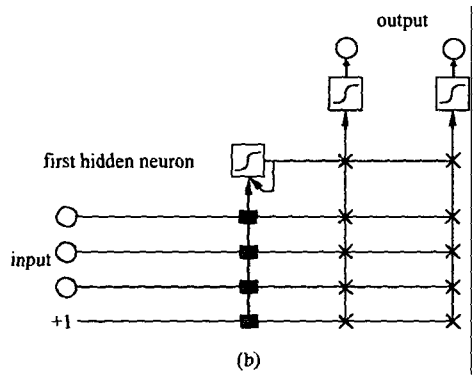
수평성장을 위한 후보뉴런 *k*의 출력:

$$o_f^{(k)}(t) = f_{act}\left(\sum_{l=j}^{i} o_l^{(k)}(t) w_{fl}^{(k)} + o_f^{(k)}(t-1) w_{ff}^{(k)}\right)$$

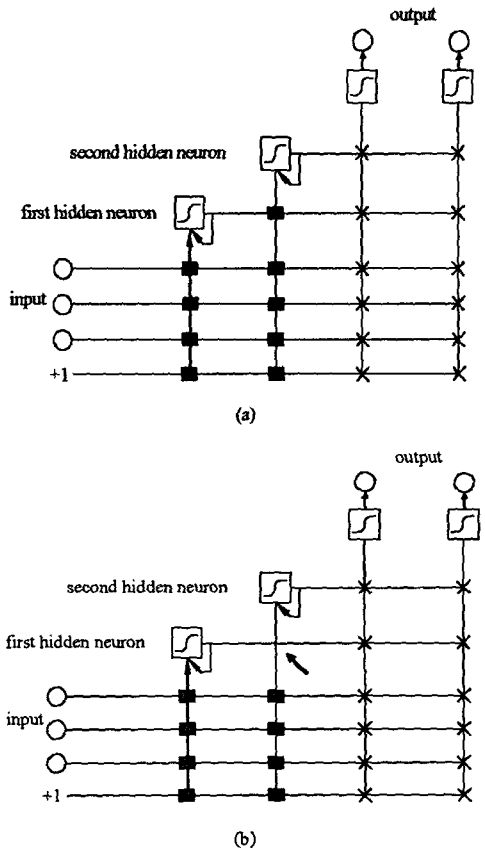
초기 네트워크로부터 첫 번째 성장에서는, 은닉층의 생성을 위하여 풀의 첫 번째 그룹에 있는 후보뉴런 중에서 한 뉴런이 은닉뉴런으로 선택되어야 한다. [그림 6] (a)는 새로운 네트워크 설계의 초기상태를 보여주고 [그림 6] (b)는 첫 번째 은닉뉴런이 첨가된 상태를 나타낸다. 두 번째 성장부터는 두 그룹에 있는 후보뉴런들 중에서 한 뉴런이 은닉뉴런으로 선택되고 첨가되어 그에 따라 네트워크의 성장이 결정된다. [그림 7]에서 (a)와 (b)는 새로운 네트워크의 수직성장과 수평성장을 보여준다.

[그림 6]과 [그림 7]에서와 같이 새로이 설계된 네트워크를 일반화된 RCC(Generalized RCC)라 하고 G-RCC라고 표기하기로 한다.





[그림 6] (a): 새로운 네트워크 설계의 초기 상태, 그리고 (b): 첫 번째 성장



[그림 7] 두 번째 성장: (a): 수직으로, 또는 (b): 수평으로

4. 실험 및 성능 분석

네트워크의 학습향상을 위하여 새로운 두 활성화함수를 소개한다. 한 활성화함수는 CosExp라 칭하고 그 함수와 미분은 아래와 같다:

$$f_{act}(x) = e^{-b|x-d|}\cos(c|x-d|).$$

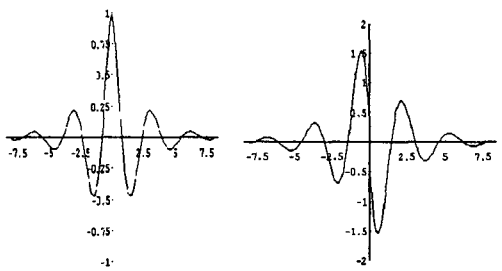
$$f'_{act}(x) = \frac{d-x}{|d-x|}e^{-b|x-d|}(b\cos(c\,|\,x-d\,|) + c\sin(c\,|\,x-d\,|)) \quad (d-x \neq 0).$$

다른 활성화함수는 CosGauss라 칭하고 그 함수와 미분은 다음과 같다:

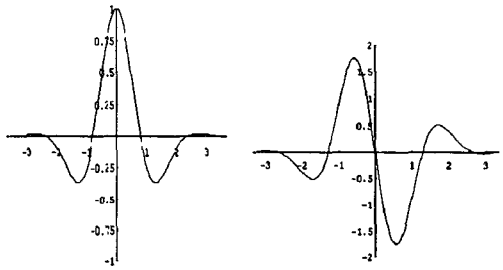
$$f_{act}(x) = e^{-b(x-d)^2}\cos(c(x-d))$$

$$f'_{act}(x) = -2b(x-d)e^{-b(x-d)^2}\cos(c(x-d)) - ce^{-b(x-d)^2}\sin(c(x-d)).$$

CosExp 함수의 상수 b는 지수함수의 기울기를 결정하고, CosGauss 함수에서는 가우스함수의 기울기를 결정한다. 두 함수의 상수 d는 hyperplane의 위치를 결정하고 상수 c는 한 일정한 구간에서 주기(periode)의 길이를 결정한다. 그 함수들의 최대값은 항상 1이다. x축의 한 일정한 구간에서 극점의 수는 c와 b의 값에 의해서 결정되며 이러한 관계를 설명하기 위한 예로 [그림 8] (a)에서는 b=0.5, c=2, d=0의 값을 갖고있는 CosExp 함수를 [그림 8] (b)는 그의 미분함수를 보여준다. [그림 9] (a)에서는 b=0.5, c=2, d=0의 값을 갖고있는 CosGauss 함수를 [그림 9] (b)는 그의 미분함수를 나타낸다.



[그림 8] (a): b=0.5, c=2, d=0의 파라미터값을 갖고 있는 CosExp 함수와 (b): 그의 미분



[그림 9] (a): b=0.5, c=2, d=0의 파라미터값을 갖고 있는 CosGauss 함수와 (b): 그의 미분

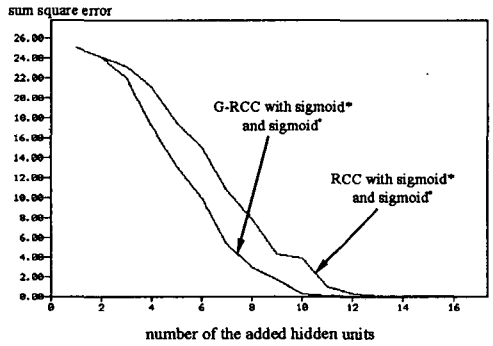
새로운 활성화함수를 사용하여 일반화된 RCC 알고리즘의 성능을 실험하기 위하여 벤치마크문제(benchmark- problem)인 모스 신호문제를 선택했다. 이 문제는 모든 알파벳에 해당하는 모스 신호를 학습하는 것이다. 네트워크의 입력 뉴런은 1개이고 출력뉴런은 27개이다. 모든 입출력은 부울 값(boolean value)으로 나타내고 한 점은 ON-OFF를 그리고 한 선은 ON-ON-OFF을 나타내며 여기에서 ON은 1의 값을 그리고 OFF는 0의 값을 의미한다. 예를 들어서 "...-"는 모스 시그널로 알파벳 "V"를 나타낸다. 이것은 입력값 "10 10 10 110 0"로 코딩된다. 마지막 0은 알파벳의 끝을 나타내기 위하여 표시한다. 이 0의 값을 읽고난 후에 그에 해당하는 뉴런, 즉 이 경우에 22번째 뉴런과 다른 알파벳의 시작을 알리기 위하여 마지막 출력 뉴런인 27번째 뉴런에 1의 값을 전달한다. 알파벳을 위한 코드길이는 3~12비트이고 정확하게 하나의 알파벳을 인식하기 위하여 입력값은 순서대로 주어지지 않는다.

모스 문제의 실험데이터는 *ftp.cs.cmu.edu*로 anonymous ftp를 통하여 Carnegie Mellon 대학의 S. E. Fahlman과 그의 연구원들에 의해서 구성된 Neural Bench Collection의 /afs/cs/project/connect/ bench에서 구할 수 있으며 또한 *ics.uci.edu*로 anonymous ftp를 통하여 UCI(University of California, Irvine)의 P. M. Murphy, D. W. Aha에 의해서 구성된 machine learning databases의 /pub/machine- learning-databases에서 얻을 수 있다. 실험을 위한 기본 시뮬레이터(Simulator)는 독일 Stuttgart대학의 Institute for Parallel and Distributed High Performance Systems (IPVR)에서 개발한 Stuttgart Neural Network Simulator (SNNS)[Stuttgart, 1998]를 이용했으며 이 시뮬레이터는 *ftp.informatik.uni-stuttgart.de* (129.69.211.1)로 anonymous ftp를 통하여 */pub/SNNS*에서 가져올 수 있다.

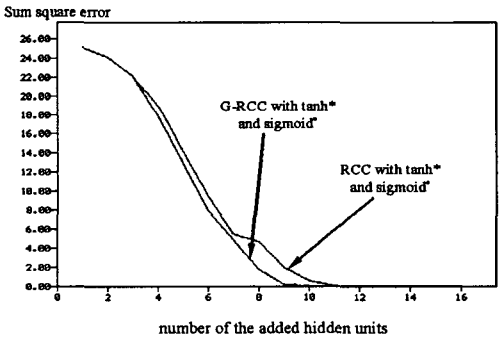
이 논문에서 모든 실험들은 동일한 조건하에서 실행하였다. 즉 G-RCC와 RCC의 학습에 필요한 파라미터값을 동일하게 주었다. 또한 네트워크의 출력뉴런을 위한 활성화함수는 항상 sigmoid함수를 사용하였다. 이 논문의 그림과 표에서 은닉뉴런과 후보뉴런을 위한 활성화함수는 '*'와 함께 그리고 네트워크의 출력뉴런을 위한 활성화함수는 '·'와 함께 표시했다. 그림들에서 원안의 숫자는 은닉뉴런들의 생성된 순서를 나타낸다.

[그림 10]에서는 sigmoid 활성화 함수를 사용

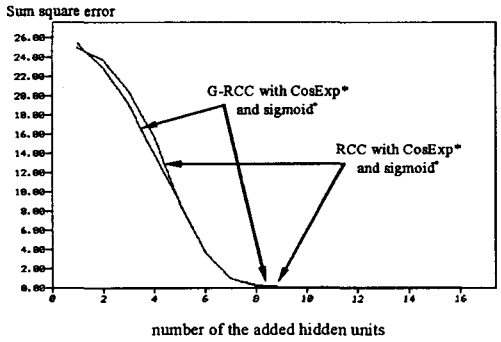
한 실험을 보여주고 RCC에 비해서 G-RCC에서 2개의 은닉 뉴런을 절약할 수 있었다. [그림 11]은 학습후에 형성된 G-RCC의 구조를 보여준다. [그림 12]에서는 tanh 활성화 함수를 사용하여 실험



[그림 10] 학습오차와 생성된 은닉뉴런 수의 비교



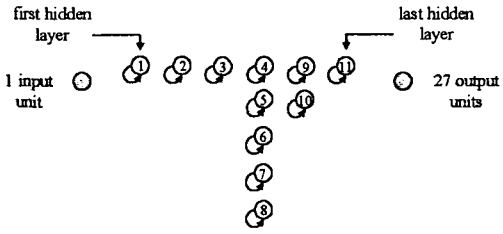
[그림 12] 학습오차와 생성된 은닉뉴런 수의 비교



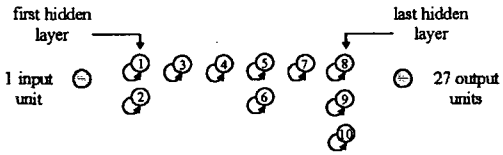
[그림 14] 학습오차와 생성된 은닉뉴런 수의 비교

하였는데 역시 RCC에 비해서 G-RCC에서 보다 빠른 학습 속도를 보여주었다.

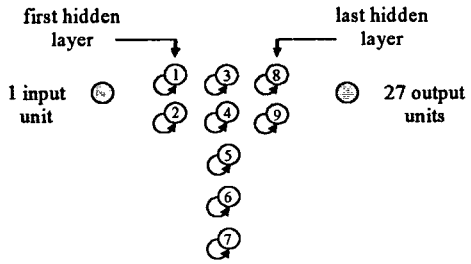
[그림 13]은 네트워크의 학습 후 은닉뉴런의 나열된 상태를 나타낸다. [그림 14]에서는 CosExp



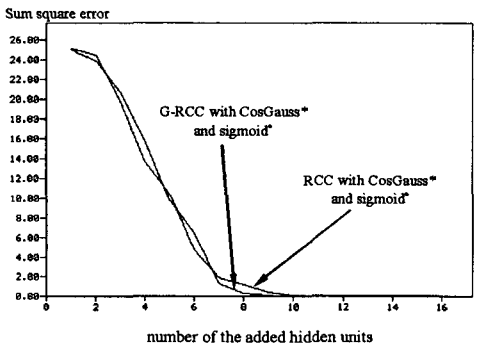
[그림 11] 'G-RCC'와 'RCC'에서 'sigmoid*'와 'sigmoid·'를 사용하여 학습을 완료한 후 각 은닉층에 뉴런들이 나열된 상태



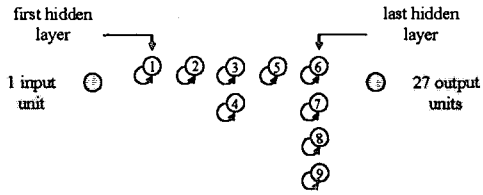
[그림 13] 'G-RCC'와 'RCC'에서 'tanh*'와 'sigmoid·'를 사용하여 학습을 완료한 후 각 은닉층에 뉴런들이 나열된 상태



[그림 15] 'G-RCC'와 'RCC'에서 'CosExp*'와 'sigmoid·'를 사용하여 학습을 완료한 후 각 은닉층에 뉴런들이 나열된 상태



[그림 16] 학습오차와 생성된 은닉뉴런 수의 비교



[그림 17] 'G-RCC'와 'RCC'에서 'CosGauss*'와 'sigmoid·'를 사용하여 학습을 완료한 후 각 은닉층에 뉴런들이 나열된 상태

의 새로운 활성화 함수, 그리고 [그림 16]에서는 CosGauss의 새로운 활성화 함수를 사용하여 실험하였는데, 특히 RCC와 G-RCC에서 모두 기존의 활성화 함수에 비해서 빠른 속도로 학습하였음을 알 수 있었다. CosExp의 경우에는 RCC와 G-RCC에서 거의 같은 성능을 보였지만, CosGauss의 경우에는 G-RCC에서 RCC보다 좋은 성능을 확인할 수 있었다. [그림 15]와 [그림 17]은 각기 새로운 활성화 함수에 따라서 학습한 후의 네트워크 은닉층에 나열된 뉴런들의 형태를 보여 준다.

5. 결론 및 연구과제

이 논문에서는 RCC 네트워크와 2계 RCC 네트워크를 조합하여 RCC 네트워크의 일반화를 시도하였다. 이 새로운 학습알고리즘은 새 은닉뉴런이 생성될 때마다 네트워크의 수직성장 또는 수평성장을 스스로 결정한다. 또한 새로운 활성화함수를 사용하여 기존의 활성화 함수보다 학습이 빠른 속도로 진행된 것을 알 수 있었다. 그리고 G-RCC는 모든 활성화 함수에서 RCC 보다 더 빠른 학습 속도를 가지고 있음을 확인할 수 있었다. 앞으로의 연구방향은 Cascade-Correlation 네트워크와 2계 순환 Cascade- Correlation 네트워크를 조합하여 일반화 시킨 Cascade-Correlation 네트워크와 G-RCC를 다시 조합한 알고리즘을 구현하고 또한 여기에서 유전자알고리즘(genetic algorithm) 또는 진화전략(evolutionary strategy)을 이용하여 이 논문에서 사용한 새로운 활성화 함수의 좋은 파라미터 값을 구하는 것도 향후 연구과제이다.

참고문헌

[1] Drago, G. P. and Ridella, S., "Cascade Correlation Convergence Theorem", Proceedings of the International Conference on Artificial Neural Networks, Volume One, Springer-Verlag, Sorrento, Italy, 26-29, May, 1994, 573~576.

[2] Fahlman, S. E., "Faster-learning variations on back-propagation: An empirical study", Proceedings of the 1988 Connectionist Models Summer School. Morgan Kaufmann, 1988.

[3] Fahlman, S. E. and Lebiere, C., "The

cascade-correlation learning architecture", S. Touretzky, Editor, Advances in Neural Information Processing Systems 2, Morgan Kaufmann, 1990.

[4] Fahlman, S. E., "The Recurrent Cascade-Correlation Architecture", Advances in Neural Information Processing Systems 3, R. P. Lippmann, J. E. Moody, D. S. Touretzky (Eds.), Morgan Kaufmann Publishers, Inc., 1991, 190~198.

[5] Grabec, I., "Cascade Neural Network Developed for Time Series Prediction", Proceedings of the International Conference on Artificial Neural Networks, Springer-Verlag, Amsterdam, Netherlands, 13-16 September, 1997, 468~471.

[6] Lee, S. W., "New non-monotonic Activation Functions for Neural Networks and Optimizing of the Cascade-Correlation-Architecture", Research Report, 633/1996, Department of Computer Science, University of Dortmund, Germany, December, 1999.

[7] Lee, S. W., "Verallgemeinerte Neuronale Netze mit kaskadierter Korrelations-Architektur", 16. Workshop Interdisziplinäre Methoden in der Informatik, Haus Nordhelle, Meinerzhagen-Valbert. Fakultät für Informatik, Universität Dortmund, Germany, 16-19. 08 2000.

[8] Prechelt, L., "PROBEN1-A Set of Neural Network Benchmark Problems and Benchmarking Rules", Technical Report 21/94, Department of Computer Science, University of Karlsruhe, September 30, 1999.

[9] Stuttgart Neural Network Simulator (SNNS), User Manual, Version 4.0, Institute for Parallel and Distributed High Performance Systems (IPVR), University of Stuttgart, 1998.

Abstract

Generalization of the Recurrent Cascade Correlation Algorithm and Morse Signal Experiments using new Activation Functions

Lee, Sang-Wha* · Song, Hae-Sang*

Recurrent-Cascade-Correlation(RCC) is a supervised learning algorithm that automatically determines the size and topology of the network. RCC adds new hidden neurons one by one and creates a multi-layer structure in which each hidden layer has only one neuron. By second order RCC, new hidden neurons are added to only one hidden layer. These created neurons are not connected to each other. We present a generalization of the RCC Architecture by combining the standard RCC Architecture and the second order RCC Architecture. Whenever a hidden neuron has to be added, the new RCC learning algorithm automatically determines whether the network topology grows vertically or horizontally. This new algorithm using sigmoid, tanh and new activation functions was tested with the morse-benchmark-problem. Therefore we recognized that the number of hidden neurons was decreased by the experiments of the RCC network generalization which used the activation functions.

Key words : RCC, Second order RCC, sigmoid, tahh, new activation function

* School of Computer, Information and Communication, Seowon University