

퍼지 전처리기를 가진 신경회로망 모델의 개발

조성원, 최경삼, 황인호
홍익대학교 공과대학 전기공학과

Development of a Neural Network with Fuzzy Preprocessor

Seongwon Cho, Kyung-Sam Choi, In Ho Hwang
Department of Electrical & Control Engineering
Hong Ik University

ABSTRACT

In this paper, we propose a neural network with fuzzy preprocessor not only for improving the classification accuracy but also for being able to classify objects whose attribute values do not have clear boundaries. The fuzzy input signal representation scheme is included as a preprocessing module. It transforms imprecise input in linguistic form and precisely stated numerical input into multidimensional numerical values. The transformed input is processed in the postprocessing module. The experimental results indicate the superiority of the backpropagation network with fuzzy preprocessor in comparison to the conventional backpropagation network.

1. 서론

본 논문에서는, 신경회로망의 분류 정확도를 향상시키고 퍼지 데이터도 분류할 수 있도록 하기 위하여 퍼지 전처리기를 갖는 신경회로망을 제안한다.

퍼지 전처리기는 퍼지집합으로 주어지는 입력이나 수치값으로 주어지는 입력을 다차원 데이터로 변환하고, 이 변환된 입력은 신경회로망의 학습 데이터로 사용된다.

본 논문은 다섯 장으로 이루어져있다. 2장은 학습 데이터의 분포에 근거하여 퍼지집합의 멤버십 함수를 자동으로 생성하는 것에 대해 설명한다. 3장은 유도된 퍼지집합들과 유사정도(Degree of Match) 계산법을 사용하여 원래의 데이터를 다차원 데이터로 변환한다. 4장에서는 퍼지 전처리기를 갖는 신경회로망과 원래의 신경회로망과의 분류성능을 비교하고, 5장은 결론부이다.

2. 멤버십함수의 자동적인 생성

일반적으로 퍼지시스템에서는 전문가의 경험을 이용해 퍼지집합의 멤버십 함수를 정의한다. 이것을 할 수 없을 때에는 실험 데이터로부터 직접 멤버십함수를 유도하게 되는데, 지금까지 여러가지 방법들이 제안되었다 [1][3][4][6].

이번 장에서는 실험 데이터로부터 퍼지집합의 멤버십함수를 자동으로 생성하는 새로운 방법을 제안한다.

2.1 기본 개념

비퍼지 시스템과는 다르게, 퍼지시스템에서는 사용되는 퍼지집합에 대한 정보를 필요로한다. 이런 시스템을 개발하려면, 퍼지집합들의 멤버십함수를 정의해 주어야 한다.

퍼지시스템에서는 퍼지집합의 멤버십함수를 어떻게 정의해야 하는가에 대한 문제가 종종 제기된다. 어떤 연구자들은 퍼지 멤버십함수는 주관적으로 정의 된다는 점을 지적한다. 실험주의자들은 전문가나 사용자들의 통계적인 고찰에 근거해 퍼지집합의 멤버십함수를 정의 해야한다고 주장한다. 다시말해 멤버십함수는 전문가나 사용자에 의해 주관적으로 정의될 수 있다는 것을 의미한다.

본 논문에서는 주관적인 견해에 근거하지 않고 실험 데이터의 분포에 근거해 퍼지집합의 멤버십함수를 객관적으로 유도하는 방법을 제안한다. 실험 데이터는 샘플의 집합으로 이루어져 있고, 각 샘플은 관찰되는 임출력 관계를 표현하며 어떤 대상의 몇가지 속성값들과 그 대상이 속하는 클래스로 이루어져 있다. 각 속성에 대하여 제한된 수의 퍼지집합과 그것들의 멤버십함수들이 유도된다.

2.2 알고리즘

샘플 데이터로부터 퍼지집합의 멤버십함수를 유도하기 위하여 각 속성에대해 수행해야할 절차는 다음과같다.

초기화: $i=1$

단계1: 클래스 i 에 속하는 샘플들에 대하여, 각 속성의 히스토그램을 만든다. 변수 n 은 가능한 클래스의 갯수이다. 히스토그램의 X축과 Y축은 각각 속성값의 전체 영역과 각 구간에 속하는 샘플들의 갯수이다.

단계2: j 번째 구간, x_j 퍼지집합 A_i 의 특성들을 만족하는 정도를 계산한다.

$$m_{A_i}(x_j) = \frac{1}{N_d} \sum_{k=1}^L \min(N_j N_k) \quad (1)$$

여기서 $m_{A_i}(x_j)$ 는 A_i 에 대한 x_j 의 멤버십값을 나타내며, N_j 와 N_k 는 각각 j 와 k 번째 구간에 속하는 샘플들의 개수이고, L 는 구간의 개수이며, N_d 는 클래스 i 에 속하는 전체 샘플의 개수이다.

단계3: $i=n$ 이면 단계 4로 가고, 그렇지 않으면 i 를 하나 증가 시키고 단계 1로 간다.

단계4: 단계 2에서 계산된 멤버십함수들을 사용해 n 개의 클래스에 대응되는 퍼지집합들 사이의 유사정도를 계산한다. 유사정도의 계산에는 Euclidian 거리법이나 Hamming 거리법을 사용한다.

단계5: 만약 유사정도가 임계값보다 작으면, 유사한 퍼지집합들을 합쳐 하나의 새로운 퍼지집합을 만든다. 새로운 퍼지집합의 멤버십함수는 식(1)에 의해 다시 계산한다.

한 속성에 대하여 가능한 퍼지집합의 최대 개수는 n 이다. 단계 1에서 단계 5의 과정이 각 속성에 대하여 반복적으로 수행된다. 그러므로, 만약 m 개의 속성이 있다면 퍼지집합의 전체 최대 개수는 $m \times n$ 이다.

3. 퍼지 입력신호 표현

입력정보를 어떻게 수치값으로 표현할 것인지가 신경회로망에서는 중요한 문제이다. 입력 신호를 십진수나 이진수로 표현하는 것이 반드시 최적인 것은 아니다. 이번 장에서는 학습 데이터로부터 생성된 퍼지집합의 멤버십함수를 이용한 새로운 표현법을 제안한다.

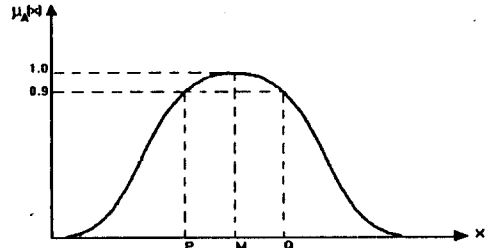
3.1 신경회로망의 학습모드에서 퍼지 입력신호 표현
신경회로망의 학습의 전처리 단계로써 주어진 입력신호를 퍼지 입력신호로 변환하게 되는데 그 절차는 다음과 같다.

단계1: 2.2장에서 제시된 알고리즘을 사용하여 원래의 학습 데이터로부터 퍼지집합의 멤버십함수를 유도한다.

단계2: 각 퍼지집합을 두개의 새로운 퍼지집합으로 나눈다. 이 단계를 수행하는 방법은 3.1.1에서 설명한다.

단계3: 퍼지집합들의 클래스 분리정도에 근거해 K 개의 퍼지집합을 선택한다. 자세한 것은 3.1.2에서 설명한다.

단계4: 단계 3에서 선택된 퍼지집합들과 유사정도 계산법을 사용하여 학습벡터들을 다차원 벡터로 변환한다. 자세한 것은 3.1.3에서 설명한다.



<그림 1> 원래의 퍼지집합 A

3.1.1 퍼지집합의 변형

일반적으로 학습 데이터로부터 생성된 퍼지집합의 멤버십함수는 비감소 함수(nondecreasing function)가 아니다. <그림 1>에서 예를 보여준다.

P와 Q점은 X축에서는 서로 다른 값이지만 퍼지집합 A에서의 멤버십값은 서로 같다. 그러므로, 만약 이러한 멤버십함수를 신경회로망의 퍼지 입력신호에 직접 사용한다면 P점에 위치한 어떤 대상물이 Q점에 위치한 다른 대상물과 동일한 것으로 간주될 가능성이 크다. 이 문제를 해결하기 위해서는 멤버십함수를 변형해 주어야 하는데, 다음에서 이를 위한 하나의 방법을 기술한다.

단계1: 퍼지집합, A,의 멤버십값들중 가장 큰 값을 갖는 X축상의 값, M,을 찾는다.

$$m_A(M) = 1 \quad (2)$$

단계2: 원래의 퍼지집합을 두개로 나눈다.

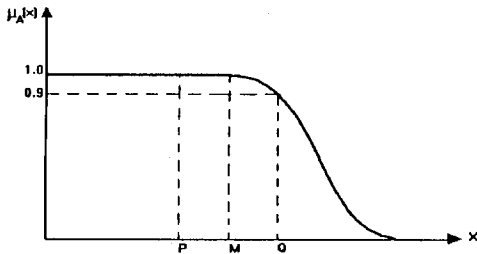
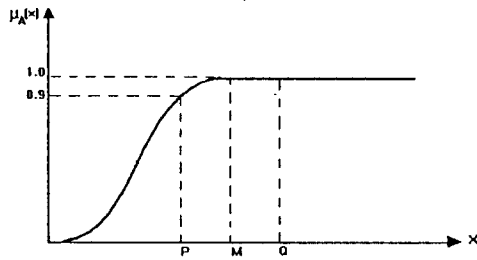
1) X축상의 값, x ,이 M보다 크거나 같으면, 첫번째 새로운 퍼지집합, A_1 ,의 멤버십값들은 1이고 두번째 퍼지집합, A_2 ,의 멤버십값들은 원래 퍼지집합의 멤버십값들과 같다.

$$m_{A_1}(x) = 1 \text{ and } m_{A_2} = m_A(x) \text{ for } x \geq M \quad (3)$$

2) 만약 X축상의, x ,이 M보다 작으면, 첫번째 새로운 퍼지집합, A_1 ,의 멤버십값은 원래 퍼지집합의 멤버십값과 같고 두번째 퍼지집합, A_2 ,의 멤버십값은 1이다.

$$m_{A_1}(x) = m_A \text{ and } m_{A_2} = 1 \text{ for } x < M \quad (4)$$

<그림 2>는 <그림 1>의 원래의 퍼지집합으로부터 위의 방법에 따라 변형된 두개의 새로운 퍼지집합을 보여주고 있다.



〈그림 2〉 A로부터 나누어진 퍼지집합 A₁과 A₂

3.1.2 퍼지집합의 선택

신경회로망에서 퍼지 입력신호 표현법을 사용할때, 입력층의 노드의 갯수는 퍼지집합의 전체 갯수와 같다. 모든 각각의 퍼지집합이 두개의 새로운 퍼지집합으로 나누어진 후에는, 전체 퍼지집합의 최대 갯수는 $2 \times m \times n$ 이다. 여기서 m 은 속성의 갯수(원래 입력 벡터의 차원)이고 n 은 클래스의 갯수이다. 그러나, 실제 하드웨어적인 구현에 있어서는, 입력층에 있는 노드의 갯수를 줄일 필요가있다.

이번 장에서는, 클래스 분리정도에 근거해서 퍼지집합들을 선택하는 방법론을 기술한다. 클래스 분리정도가 작은 퍼지집합들은 사용하지 않은 때 보다 오히려 학습능력을 저하 시킬 우려가 있으므로 제외시키고 분리정도가 큰 퍼지집합들만을 선택하여 입력신호 변환에 사용한다. 퍼지집합의 클래스 분리정도는 within-class scatter와 between-class scatter를 사용하여 계산하는데, 그 절차는 다음과 같다.

단계1: 각 클래스 i 와 클래스 i 에 속하는 입력 벡터들의 j 번째 속성값들의 멤버십값들을 갖는 퍼지집합 A_k 에 대하여 샘플들의 평균값, m_i , 을 계산한다.

$$m_i = \frac{1}{N_i} \sum_{m=1}^{N_i} m_{A_k}(x_{mj}) \quad (5)$$

단계2: 각 클래스 i 와 클래스 i 에 속하는 입력 벡터들의 j 번째 속성값의 멤버십을 갖는 퍼지집합 A_k 에 대하여 샘플의 분산, s_i^2 , 을 계산한다.

$$s_i^2 = \frac{1}{N_i} \sum_{m=1}^{N_i} [m_{A_k}(x_{mj}) - m_i]^2 \quad (6)$$

단계3: 퍼지집합 A_k 에대한 within-class scatter, S_w , 를 계산한다.

within-class scatter는 클래스내에 있는 샘플들의 평균분산을 나타내며 다음과 같이 표현된다.

$$S_w = \sum_{i=1}^n P_i s_i^2 \quad (7)$$

여기서 n 은 클래스의 전체 갯수이며, P_i 는 전체 데이터 중에서 각 속성의 데이터들이 차지하는 비율을 나타내는 확률이다.

$$P_i = \frac{N_i}{\sum_{k=1}^n N_k} \quad (8)$$

단계 4: 퍼지집합 A_k 에 대하여 m_i 의 평균값 m_o 를 계산한다.

$$m_o = \sum_{i=1}^n P_i m_i \quad (9)$$

단계 5: between-class scatter, S_b , 를 계산한다.

$$S_b = \sum_{i=1}^n P_i [m_i - m_o]^2 \quad (10)$$

단계 6: 마지막으로, within-class scatter와 between-class scatter를 이용하여 클래스 분리 정도, S , 를 계산한다.

$$S = \frac{S_b}{S_w} \quad (11)$$

단계 7: 모든 퍼지집합에 대하여 단계 1에서 단계 6 까지를 반복한다.

3.1.3 학습입력의 멤버십함수 벡터로의 변환

일단 입력 표현에 사용될 퍼지집합들을 선택한 후에는, 원래의 학습입력들이 수치값으로 주어진 경우에는 다음식을 이용하여 멤버십함수 벡터로 변환된다 [6].

$$M_k = \mu_{A_k}(x_{mj}) \quad (12)$$

여기서 x_{mj} 는 입력벡터 x_m 의 j 번째 속성값을 의미하며, A_k 는 j 번째 속성의 k 번째 퍼지집합이고, $\mu_{A_k}(x_{mj})$ 는 x_{mj} 의 A_k 에서의 멤버십값이다. M_k 는 j 번째 속성의 k 번째 퍼지집합으로부터 얻어지는 멤버십값이다. j 번째 속성값에 대하여, 선택된 퍼지집합의 수가 K_j 라면, 원래 학습벡터의 대응되는 속성 값은 K_j 차원의 멤버십값으로 변환된다.

만약 원래의 학습 벡터들이 수치값이 아닌 애매모호함을 나타내는 퍼지집합으로 주어지면, 멤버십 벡터들은 다음과 같은 식들을 사용해 얻어진다. 이를 위해 먼저 두개의 파라미터들이 정의된다 [8].

$$CERT_{+} = \max(\min(\mu_A, \mu_{A^c}))$$

$$CERT_{-} = \max(\min(1 - \mu_A, \mu_{A^c}))$$

여기서 μ_A 는 선택된 퍼지집합의 멤버십함수이고 μ_{A^c} 는 학습 입력을 나타내는 퍼지집합의 멤버십함수이다.

$CERT_{+}(x_{mj}, A_{jk}) \geq 0.5$ 이고 $CERT_{-}(x_{mj}, A_{jk}) \leq 0.5$ 인 경우에는,

$$M_{jk} = CERT_{+}(x_{mj}, A_{jk}) \quad (13)$$

$CERT_{+}(x_{mj}, A_{jk}) \geq 0.5$ 이고 $CERT_{-}(x_{mj}, A_{jk}) \geq 0.5$ 인 경우에는,

$$M_{jk} = (CERT_{+}(x_{mj}, A_{jk}) - CERT_{-}(x_{mj}, A_{jk})) + 0.5 \quad (14)$$

$CERT_{+}(x_{mj}, A_{jk}) \leq 0.5$ 이고 $CERT_{-}(x_{mj}, A_{jk}) \geq 0.5$ 인 경우에는,

$$M_{jk} = 1 - CERT_{-}(x_{mj}, A_{jk}) \quad (15)$$

멤버십벡터는 입력벡터의 모든 M_{jk} 값들을 일정한 순서로 나열함으로써 얻어진다. 선택된 퍼지집합의 멤버십 함수들과 M_{jk} 값들의 멤버십벡터에서의 순서는 테스트벡터의 퍼지 입력신호 표현에 사용할 수 있도록 저장해둔다. 멤버십벡터의 크기는 3.1.2절의 절차를 사용해 선택한 전체 퍼지집합의 갯수와 같다.

$$\text{멤버십벡터의 차원} = \sum_{j=1}^m K_j \quad (16)$$

여기서 K_j 는 j 번째 속성에 대하여 선택된 퍼지집합의 갯수이고 m 은 속성의 전체 갯수이다.

3.2 테스트 모드에서의 퍼지 입력신호 표현

테스트 모드에서의 퍼지 입력신호 표현은 간단하다. 멤버십함수를 유도하고, 퍼지집합을 변형하고 그리고 퍼지집합을 선택하는 단계들은 필요없다. 테스트 데이터들은 학습 데이터와 같은 방법의 의해 멤버십벡터로 변환된다.

정확한 속성값의 경우에는, 식(12)에 의해 멤버십벡터로 변환된다. 부정확한 속성값인 경우에는 식(13)~(15)를 사용하여 멤버십값들을 계산한다.

4. 실험 결과

퍼지 입력신호 표현을 십진이나 이진 데이터 입력 표현과 비교하기 위해 가장 널리 사용되는 신경 회로망인 Backpropagation Network(BPN)을 사용한 실험들이 수행되었다. 실험에 사용된 데이터는 원격거리 측정 데이터, FLC1, 가 사용되었다.

4.1 Flight Line C1 데이터를 사용한 실험

본 실험에서는 4개의 클래스를 갖는 데이터가 사용되었다. 퍼지 전처리기의 분류성능을 고찰하기 위해, 퍼지 전처리기를 사용한 BPN과 원래의 BPN의 분류성능들을 비교하였다.

4.1.1 데이터 셋(Data Set)

실험 데이터로는 미국 중부 지방의 평원 지역을 항공 촬영한 원격 탐사(remote sensing) 데이터를 사용하였다 [9]. 이 데이터는 8-밴드(band)를 가진다. 즉 입력벡터들은 8차원의 0에서 255사이의 정수값을 갖는다. 또한 <표1>과 같이 4개의 농작물을 나타내는 클래스로 이루어진다.

각 클래스당 학습 데이터는 200개의 샘플로 구성되며 테스트 데이터는 375개로 구성되어 있다. 그러므로 전체 학습 데이터의 갯수는 800개 이고 테스트 데이터의 갯수는 1500개이다.

<표 1> 실험 데이터의 구성

클래스	영역 설명
1	풀밭
2	옥수수
3	귀리
4	붉은 클로버

4.1.2 시뮬레이션

퍼지집합에 대한 멤버십함수가 유도된후, 유사정도 계산에는 Hamming 거리법이 사용되었다. 퍼지집합이 유사한가 아닌가를 판단하기 위한 임계값으로는 Hamming 거리 4.0이 사용되었다. 각 퍼지집합을 2개로 나눈후, 퍼지집합의 전체 갯수는 44개였다. 이 중에서 클래스 분리정도가 큰 26개의 퍼지집합만을 선택했다. 그러므로 퍼지 입력표현을 갖는 Backpropagation Network(BPN)인 경우, 입력노드의 수는 1개의 bias를 포함하여 27개이고, 은닉층의 노드수도 1개의 bias를 포함하여 27개로 하였고 출력노드는 4개로 잡았다. 이진 데이터를 사용한 실험에서는 각 속성을 8 비트로 이진화하여 사용하였다. 이 경우 입력노드의 갯수는 bias를 포함하여 65개이고, 은닉층의 노드수도 1개의 bias를 포함하여 65개로 하였고 출력노드는 4개로 잡았다. 십진 데이터를 사용한 실험에서는 모든 데이터를 각 속성에 대하여 0에서 1의 값으로 정규화(normalization)하여 사용하였다. 입력노드는 bias를 포함하여 9개이고, 은닉층의 노드수도 1개의 bias를 포함하여 9개로 하였고 출력노드는 4개로 잡았다. 학습률은 모든 실험에서 똑같이 0.01로 잡았고, 기중치(weight)는 -0.5에서 0.5사이의 값들로 초기화하여 실험을 하였다.

Classification Accuracy		
Iteration	Training data	Testing Data
200	91.88%	84.60%
300	94.00%	87.53%
400	96.25%	88.73%
500	97.25%	89.60%
600	98.25%	90.07%
700	99.00%	90.13%
800	99.13%	90.53%
900	99.63%	90.53%
1000	99.88%	90.73%
1100	99.88%	90.67%

<표 2> 이진 데이터를 사용한 Backpropagation Network(BPN)의 분류성능

Classification Accuracy		
Iteration	Training data	Testing Data
200	91.63%	87.13%
300	92.75%	89.67%
400	93.13%	90.20%
500	93.75%	90.20%
600	94.00%	90.33%
700	94.38%	90.40%
800	94.50%	90.87%
900	94.75%	91.20%
1000	95.00%	91.33%
1100	95.13%	91.13%

<표 3> 십진 데이터를 사용한 Backpropagation Network(BPN)의 분류성능

Classification Accuracy		
Iteration	Training data	Testing Data
200	94.88%	92.13%
300	95.50%	92.47%
400	95.88%	92.93%
500	96.00%	93.20%
600	96.25%	93.53%
700	96.75%	93.60%
800	96.88%	93.47%
900	97.13%	93.73%
1000	97.13%	93.87%
1100	97.63%	93.80%

<표 4> 퍼지 입력표현을 사용한 Backpropagation Network(BPN)의 분류성능

위의 표들에서 알 수 있듯이 이진 데이터를 BPN의 입력으로 사용했을 경우의 분류 성능은 학습 데이터에 대해서는 거의 완벽하지만 테스트 데이터에 대해서는 세가지 입력 표현중에서 가장 낮음을 알 수 있다. 십진 데이터를 BPN의 입력으로 사용했을 경우의 분류 성능은 학습 데이터에 대해서는 세가지 입력 표현중에서 가장 낮지만 테스트 데이터에 대해서는 이진 데이터의 경우 보다 높게 나온다. 퍼지 입력신호 표현을 사용한 경우에는 학습 데이터의 분류 성능은 이진 데이터를 입력으로 사용한 경우보다 조금 떨어지지만 테스트 데이터에 대해서는 가장 높은 결과가 나왔다. 즉 제안된 퍼지 입력표현 방법이 신경회로망의 generalization에 도움이 됨을 알 수 있다.

5. 결론

본 논문에서는, 퍼지 입력신호 표현을 갖는 신경회로망을 제안했다. 실험 데이터로부터 멤버십 함수를 자동으로 유도하는 방법이 제안되었고, 데이터들이 잘못 분류될 수 있는 가능성을 줄이기 위하여 유도된 퍼지집합들을 변형했다. 그리고 나서, 클래스 분리정도가 큰 퍼지집합들을 선택한 후 유사정도 계산법을 사용하여 원래의 데이터를 다차원 벡터 값으로 변환했다. Backpropagation Network(BPN)을 사용한 실험에서 퍼지 입력신호 표현을 사용한 신경회로망의 분류성능이 우수하다는 것을 확인할 수 있었다. 여기서 제안한 퍼지 입력신호 표현은 다른 신경회로망들과도 결합될 수 있을 것이다.

참고 문헌

- [1] B. Kosko, Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence, Eglewood Cliffs, NJ:Prentice Hall, 1992.
- [2] C.H. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I and II", IEEE Transactions on System, Man, and Cybernetics, Vol. 20, No. 2, pp. 404-435, March / April 1990.
- [3] C.T. Lin and C.S.G. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System", IEEE Transaction on Computers, Vol. 40, No. 12, pp. 1320-1336, December 1991.
- [4] H.Kang, H.G.Lee, Y.H.Kim and H.T.Jeon, "Automatic Tuning of Fuzzy Optimal Control Systems", International Fuzzy Systems Association World Congress '93, Vol. II, pp. 1195-1198.
- [5] J.M.Zurada, Introduction to Artificial Neural Systems, West Publishing Company, St. Paul, 1992
- [6] R. Krishnapuram and F.C-H Rhee, "Compact Fuzzy rulebase generation methods for computer

vision, "Proc. Second IEEE Int. Conf. Fuzzy Systems, March 1993, San Francisco, CA, Vol. II, pp. 809-814.

[7] S. Cho and M.R. Lehto, "An Algorithm to compute the Degree of Match in Fuzzy Systems", Fuzzy Sets and Systems, Vol. 49, No. 3, August 1991.

[8] S. Cho and M.R. Lehto, "A Fuzzy Scheme with Application to Expert Classification Systems for Computing the Degree of Match between a rule and an Assertion", Cybernetics and Systems, Vol. 50, August 1992.

[9] S. Cho, O.K. Erosy and M.R. Lehto, "Parallel, Self-Organizing, Hierarchical Neural Networks With Learning and Safe Rejection Schemes", IEEE Transaction on Circuit and Systems, to be printed.