Coulomb Energy Netwrok를 이용한 한글인식 Neural Network

이 경 희*, 이 원 돈**

한 국 전 자 통 신 연 구 소*, 충 남 대 학 교 전 산 학 과**

APPLICATION OF COULOMB ENERGY NETWORK TO KOREAN RECOGNITION

Kyunghee Lee*, Won Don Lee**

Electronics and Telecommunications Research Institute *
Dept. of Computer Science, ChungNam National University **

요 약

최근 Scofield는 coulomb energy network에 적용할 수있는 learning algorithm(supervised learning algorithm)을 제안 하였다. 이 learning algorithm은 multi-layer network에도 쉽게 적용이 가능하고 한 layer에서 발생한 error가 다른 layer에 영향을주지 않아서 system을 modular하게 구성할 수가 있으며 각 layer를 독립적으로 learning시킬 수 있는 특징이 있다. 본 논문에서는 coulomb energy network를 이용하여 한글인식을 위한 neural network를 구현하여 인식실험을 한 결과와 구현한 network에서 인식율을 높이기 위한 방안(2 stage learning)을 제시한다.

I. 서 론

현대의 정보처리는 거의 대부분 computer에 의해 처리 되어지고 있다. 그러나, 문자 또는 image인식, 언어습득 등과 같은 분야에서는 인간의 뇌가 현존하는 어떠한 computer system보다도 우수한 능력을 보이고 있다. 인간의 뇌가 정보를 처리하는 방식은 digital computer와 달리 병렬분산처리를 기본으로하여 학습 및 자기조직능력등을 갖고 뇌세포(neuron)간의 상호작용으로 정보를 처리하기 때문이다. 다시 말해서 processing element에 해당하는 neuron등이 synaps에 의해 서로 연결되어 정보를 주고 받음으로써 복잡한 문제를 순간적으로 해결할 수가 있다. 이런 인간의 뇌기능을 modeling하여 정보처리를 시도 하려는 연구 즉, neural network에 관한 연구가 과학자들에 의하여 이루어져 왔다.

Neural network중 pattern recognition model에서는 계층적인 구조(layered netework) 또는 상호 결합된 형태의 network을 구성하고 내부의 많은 neuron들이 상호 작용하여 parallel하게 동작함으로써 pattern recognition및 associative memory의 기능을 수행하도록 하는 방법이 많이 연구 되어지고 있다. 학습에 의한 pattern recognition machine인 perceptron[Rosenblatt, 1958]이 제안된 이래 이 perceptron의 단점을 보완하기위해 network내부에 internal layer(hidden layer)를 첨가한다든지 [Rumelhart,1986], layer의 내부에 feedback connection을 첨가하는 등 여러가지 제안이 생겨나오게 되었다.

Internal layer를 부가함으로써 neural network의 정보처리 능력은 대폭 강화 되었으나 이 internal layer를 포함한 neural network의 learning algorithm 이 명확하지 않은 것이 문제였다. 이런 문제를 해결하려는 시도로서 error back propagation(EBP) algorithm [Rumelhart,1986]및 Boltzmann machine의 learning algorithm등이 제안되었다. EBP의 기본적인 idea 자체는 새로운 것이 아니지만 multi-layer network에 적용이 가능한 algorithm을 명시 했다는 점에서 높이 평가를 받고 있다.

최근 Bachmann등은 Hopfield model[Hopfield,1982]과는 약간 다르게, 많은 수의 item이 저장 가능한 high density storge relaxation model을 제안하였으며, Scofield는 coulomb energy network에 적용할수 있는 learning algorithm(supervised learning algorithm)을 제안하였다. 이 coulomb energy network의 learning algorithm은 multi-layer로도 쉽게 응용이 가능하다는 특징이 있다. 즉, 각 layer별로 learning을 시킬 수가 있고, learning후 각 layer들이 결합하여 multi-layer network으로 구성이 가능하다. 본 논문에서는 coulomb energy network를 이용하여 '각'type의 한글 인식 network구현과 구현한 network을 이용한 한글인식 실험결과 그리고 인식율을 높이기 위한 방안(2-stage learning)을 연구 하였다.

II. COULOMB ENERGY NETWORK

2.1 개 요

Content addressing기능및 recall기능을 갖는 associative memory에 대한 연구는 오래 전부터 있었다. Hopfield는 1982년 relaxation model을 제안하였으나 그의 model 자체는 recall 기능이 약하여 network내에 존재하는 neuron 갯수의 일정한 비율(약 15%)을 넘는 item이 저장되어 있으면 recall 기능이 급격히 떨어지는 단점이 있다.

Bachmann등은 이런 Hopfield model과는 약간 다르게 많은 수의 item이 store될수 있는 relaxation model을 제안하였다. 이 relaxation model은 N-dimensional coulomb energy potential에 기초를 두고있으며, 기본 idea는 item이 저장되어야될 특정 위치에 energy minima가 놓이도록 energy function 을 define하고, 그것이 function내에서 유일한 minima임을 보인다는 것이다 [Bachmann. 1987]. Bachmann은 아래와 같은 N-dimension coulomb energy function을 정의 하였다.

$$\xi = -1 / L \sum_{j=1}^{M} Q_j |\mu - X_j|^{-L} \qquad (1)$$

위의 함수에서, 적당한 L을 선택함으로써 store된 memory (M)의 수에 상관이 없이 input item은 가장 가까운 memory에 converge된다는 것이 이미 알려져 있다. 또한 Scofield는 single 또는 multilayer N-dimensional coulomb energy network에 적용이 가능한 learning algorithm을 제안 하였다. 이 algorithm의 특징은 한 layer에서 발생한 error가 EBP의 경우와는 다르게 다

른 layer에 영향을 미치지 않게되며, 따라서 system을 modular하게 구성할 수 있고 각 layer를 독립적으로 learning시킬 수 있다는 특징이 있다. 이 algorithm의 중요한 idea는 network의 energy function 을 적절히 정의 함에 의하여 network내의 memory site(memory)들은 attractor또는 repeller로 작용할 수 있다는 것이다. 즉, memory site간의 attractive및 repulsive potential을 정의 하고 network energy를 minimize시켜감으로써 그 model이 relax된다는 것이다[Scofield,1988]. 이 방법은 각 layer별로 learning을 시킨후 결합하는 것에 의하여 multi-layer network의 learning algorithm으로도 쉽게 응용이 가능 하다.

2.2 ENERGY MINIMIZATION

M개의 memory site $x_1,\ldots,x_M$을 갖고 있는 system을 생각해 보면, memory site들은 각각 다른 class의 pattern을 나타내는 group들로 나뉘어진다고 가정할 때 앞에서 기술한 Bachmann의 energy function을 적용 시켜보면, test charge가 없는 경우에 electrostatic potentional energy는

$$\Psi = 1/(2L)\sum_{i=1}^{M}\sum_{j=1}^{M} Q_i Q_j |\mathbf{X}_i-\mathbf{X}_j|^{-L} \qquad (2)$$

여기에 동일한 class에 속하는 memory site들은 attractive potential energy를 갖도록 하기 위하여 class가 c 인 memory site i의 charge는 다음과 같이 정의 한다.

$$sign(Q_i(c)) \neq sign(Q_i(c')) \text{ for } c = c'$$
$$sign(Q_i(c)) = sign(Q_i(c')) \text{ for } c \neq c' \qquad (3)$$

위와 같이 정의 함으로써 Ψ는 같은 class에 속하는 모든 memory site들의 pair들과 서로다른 class의 memory site pair들의 electrostatic potential의 합으로 표현이 되며, 같은 class의 memory site끼리는 서로 당기게 되어 더욱 가까이 모이게 되고 서로 다른 memory site들은 서로 밀어내게 되어 더욱 멀어지게 된다.

이 coulomb energy network은 Hopfield model과 같이 network의 평형상태 즉, W에 대한 electrostatic energy의 변화가 적은 상태로 움직여 나가도록 하는 것이 중요 하다. 이를 위하여 weight Wnm에 대한 potential energy의 gradient를 계산하여 Wnm의 변화량으로 삼는다.

$$\delta\omega_{nm} = \partial\Psi/\partial\omega_{nm}$$

$$= 1/2\sum_{i=1}^{M}\sum_{j=1}^{M} Q_i Q_j |\mathbf{R}_{ij}|^{-(L+2)}\,\mathbf{R}_{ij}\cdot\partial/\partial\omega_{nm}\mathbf{R}_{ij} \qquad (4)$$
$$\text{Where : } \mathbf{R}_{ij}=\mathbf{X}_i-\mathbf{X}_j$$

또한 이식은 아래와 같이 근사적으로 표현할수있다..

$$\delta\omega_{nm} = 1/2\sum_{i=1}^{M}\sum_{j=1}^{M} Q_i Q_j |\mathbf{R}_{ij}|^{-(L+2)}\,\Delta_{nm}(\mathbf{f}_i,\mathbf{f}_j) \qquad (5)$$

$$\Delta_{nm}(\mathbf{f}_i,\mathbf{f}_j) \equiv \mathbf{R}_{ij}\cdot\partial/\partial\omega_{nm}\mathbf{R}_{ij}$$

2.3 LEARNING ALGORITHM

앞에서의 (5)번식은 memory site가 존재하지 않는 상태에서도 적용시킬 수 있다. 다시 말해서 M개의 pattern이 준비되어 있는 상태에서 그중 2개를 random하게 선택하여 input layer의 afferent line에 연결하여 dWnm의 계산을 반복 함으로써 network의 learning이 이루어 지게 된다.

f(t) pattern이 input으로 들어옴에 따른 activity는

$$\mathbf{X}(t)=\sum_{n=1}^{N}\mathbf{e}_n F_n\Big(\sum_{m=1}^{K}\omega_{nm}f_m(t)\Big)=\sum_{n=1}^{N}\mathbf{e}_n F_n(\mathbf{f}(t)) \qquad (6)$$

앞에서 W에 대한 relaxation procedure를 정의 하였으나 이것을 pattern environment에서의 activity function으로 바꾸어 생각해보면, 연속된 input pattern (f(t), f(t+1))에 대하여

$$\delta\omega_{nm} = (+/-)\,\eta\,|\mathbf{X}(t)-\mathbf{X}(t+1)|^{-(L+2)}\Delta_{nm}(\mathbf{f}(t),\mathbf{f}(t+1)) \qquad (7)$$

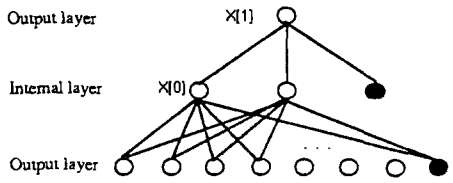
여기서 2개의 sebsequent pattern이 같은 class이라면 '-'sign을, 서로 다른 class이라면 '+'sign을 갖게한다.

Multi-layer network의 경우에도 각 layer에 (2)식의 energy function을 정의하고 (7)식에 의하여 minimize시키면 그 network system은 layer별로 서로 독립적으로 구성이 가능하며, EBP algorithm과는 다르게 dW는 다른 layer로부터의 error에 의존하지 않게된다. 또한 Hopfield model의 경우와도 달리 network에서 수용할수 있는 pattern의 갯수도 내부의 neuron수에 관계없이 크게 된다.

2.4 SIMULATION

이 learning algorithm을 symetry problem에 적용하여 보았다. symetry problem은 명확하게 선형분리가 불가능하기 때문에 2layer의 network으로는 learning이 안되고 3 layer의 구성을 필요로 하게 된다. 여기에서는 input layer에는 6개, internal layer에서는 2개, output layer에서는 1개의 neuron을 갖도록 network를 구성 하였다(그림 1).

Network에서 learning시키기 위한 input pattern들로는 20개의 6bit pattern을 준비하여 그 6bit(1또는0)의 pattern이 좌우 대칭이면 class 1에, 그렇지 않으면 class 2에 속하도록 정하였다 (그림 2).



(그림 1. Symetry problem을 위한 neural network model)

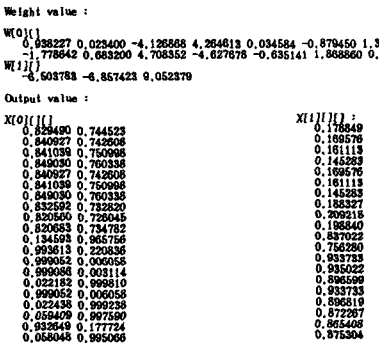
CLASS 1	CLASS 2
110011	101010
011110	010101
101101	110100
111111	110110
011110	011001
101101	110100
111111	001011
001100	111011
010010	101110
100001	101011

(그림 2. Training patterns)

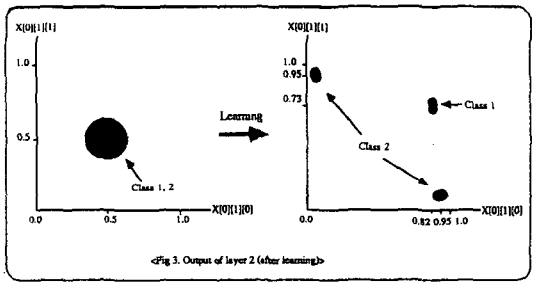
Learning시키기 위하여 미리 준비된 learning pattern중에 random하게 2개의 pattern을 선택하여 각각 f(t)와 f(t+1)로 하여 dW를 구하고 해당 W를 조정해 나감을 반복한다. initial W의 값은 -0.3 ~ 0.3 사이의 random floating 값으로 하고, L=2, eta는 0.0005, 반복횟수는 133,000, T는 1.0으로 하였다. learning시킨후의 weight value와 output을 그림 3에 나타내고 이를 2차원적으로 도식화한 것을 그림 4에 보인다.

```
Initial weight value :

W[0][][]
   0.103100 -0.050900 -0.292300 -0.202500  0.023900 -0.087700  0.070700
  -0.192900 -0.079100 -0.198900 -0.102700 -0.217700 -0.009500 -0.037300
W[1][][]
  -0.149100 -0.016100  0.103900

Output value :

X[0][][]                                        X[1][][] :
  0.514771  0.369002                              0.505301
  0.389099  0.346219                              0.510076
  0.399224  0.367865                              0.509612
  0.392766  0.301935                              0.510118
  0.389099  0.346219                              0.510076
  0.399224  0.367865                              0.509612
  0.392766  0.301935                              0.510118
  0.395536  0.416077                              0.509556
  0.510923  0.417243                              0.505251
  0.521512  0.440360                              0.504763
  0.478368  0.343711                              0.506835
  0.432609  0.443098                              0.506058
  0.480111  0.398433                              0.506475
  0.486079  0.347579                              0.506457
  0.410911  0.419506                              0.508969
  0.480111  0.398433                              0.506475
  0.429130  0.386180                              0.508424
  0.441963  0.324011                              0.508196
  0.426267  0.320930                              0.508793
  0.454551  0.341571                              0.507656
```

Learning



(그림 3. learning후의 network, output)



(그림 4. Output value의 2차원 표시)

이와 같이 learning이 되고나면 class 1과 class2의 pattern들이 확실히 분리 되어짐을 볼수 있다.

앞의 그림에서 알 수 있듯이 입력층과 중간층의 weight value들은 거의 좌우대칭을 이루고 있다. 또한 output도 EBP algorithm을 사용한 결과와 유사한 결과를 나타낸다는 것을 볼수 있다.

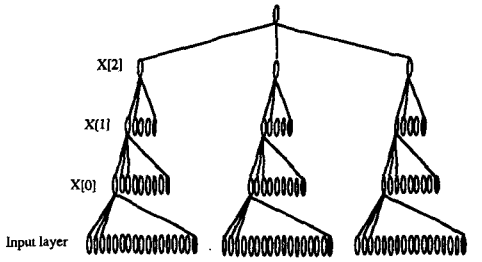
III, 한글인식을 위한 COULOMB ENERGY NETWORK의 구현

Coulomb energy network를 한글인식 문제에 적용하여 보았다. 한글은 6가지 type('가','각','고','곡','과','곽')으로 구성된 많은 글자가 있고, 한글의 자.모음에는 유사한 pattern이 많이 있다는 것이 한글인식의 어려운 점이다. 본 논문에서는 computer (VAX8800)의 computing power, learning time등의 제약때문에 초성자음.모음.종성자음의 구성을 갖는 '각' type의 한글 중에서 'ㄱ','ㅁ','ㄷ','ㄴ','ㅏ',' ','ㅓ'로 구성된 한글을 인식하는 network model을 구현하였다.

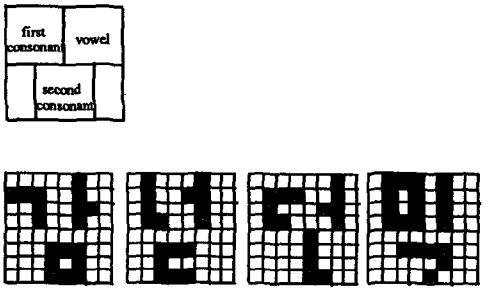
3.1 NETWORK MODEL

한글의 자.모음 인식을 위한 network model은 input layer는 input pattern의 binary value를 받아들는 input layer와 coulomb potential function unit으로 구성되는 internal layer와 output layer로 구성되며 한글을 인식하기 위해서는 초성자음을 인식하기 위한 network, 모음을 인식하기위한 network 그리고 종성자음을 인식하기 위한 network등으로 구성되어진다(그림 5).

Neural network으로 '각'type의 한글을 인식하기 위해서 우선, 한글의 자음,모음의 learning과정을 거쳐야 한다. learning에 사용되는 자음,모음의 pattern은 4x4의 크기로 하였고, 인식에 사용되는 한글 test pattern은 8x8의 크기로 하였다. 따라서 각 coulomb energy network의 input layer의 neuron은 16개로 하였고, internal layers의 neuron은 8개와 4개, output layer에는 1개로 하였다. 물론 각 layer들은 threshold unit을 하나씩 포함하고 있다. 한글 pattern에서 초성자음, 모음, 종성자음의 영역과 example을 아래에 보인다(그림 6).



(그림 5. 한글인식을 위한 neural network의 구성)

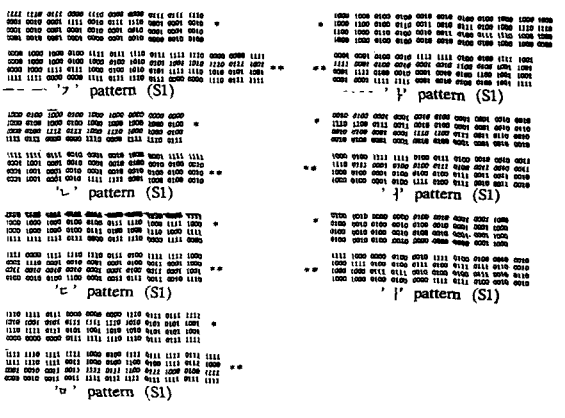


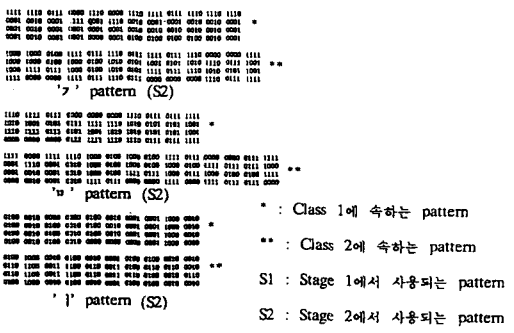
(그림 6. 한글 pattern의 영역 및 예)

3.2 LEARNING

앞에서 이미 기술한 (7)식을 이용하여 weight를 조정하며 이 network model을 learning시켰다. learning rate(eta)가 너무 크게 하면 Network가 수렴을 하지 못하고 발산을 하게 되며 너무 작으면 learning 시간이 많이 소요 된다. 이 모델에서는 eta를 <0.0005로 하였다. learning pattern은 자/모음 각각 20개정도 씩 준비하여 그중 10개정도는 class 1에 속하는 pattern으로서 인식하고자 하는 자/모음의 pattern으로, 나머지 10개정도는 class 1의 pattern으로 인식되어서는 안될 pattern(class 2)들로 구성하였다.

Learning count는 learning단계에서 사용되는 training pattern의 pair를 random하게 선택하여 learning을 시도하는 횟수로서, 한글의 자모음을 training시키기 위해서 38000번의 learning count를 사용하였다. 또한 중요한 parameter로 T (Temperature)는 0.8, L은 2, initial weight는 -0.3~0.3의 random값으로 하였다. learning에 사용된 자/모음의 pattern을 아래 그림 7에 나타낸다.





(그림 7: 한글 자/모음 pattern)

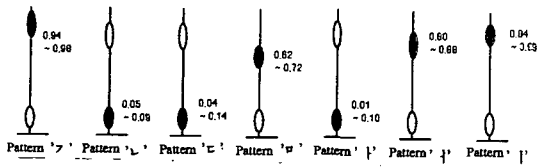
앞에서 언급한 것과 같이 'ㄱ','ㅁ','ㄷ','ㄴ','ㅏ',' ㅣ','ㅓ'등은 유사성이 크기때문에 유사한 pattern들을 정확히 구별할 수 있도록 하기 위하여 learning과정에서는 2 stage learning을 이용하였다. 서로 구별이 쉽게되는 pattern들을 input pattern으로 하여 learning(stage 1)시킨 뒤, 유사하지만 구별이 되어야될 pattern들(stage 2 pattern)을 다시 learning(stage 2)시킴으로써 유사 pattern들로 부터의 confusion을 감소시킬 수 있었다.

다시 말해서 Stage 1에서는 각 pattern(자/모음)의 major feature를 learning하게되고 stage 2에서 micro feature를 learning시킴으로써 유사성이 큰 pattern들도 인식할 수 있도록 learning시킬수 있었다.

IV. 실험 결과

'ㄱ', 'ㅁ', 'ㄷ', 'ㄴ', 'ㅏ', ' ㅣ', 'ㅓ'의 pattern을 각각 learning시킨 후 network의 output value를 그림 8에 나타낸다. 이 그림에서 output이 2개의 cluster로 확실히 구분되는 것을 볼 수있다. 특히 ' ㅣ'는 'ㅏ', 'ㅓ'등으로 confusion이 발생하는 경우가 생기게 되어 2 stage learning이 필요하였다. 이런 stage 1,2의 과정을 그림 9에 나타낸다. ' ㅣ'이외에도 'ㄱ', 'ㅁ'의 인식을 위해서도 2 stage learning을 사용하여 confusion을 감소 시켰다. 이와 같은 실험결과를 요약해 보면 아래 표와 같다(표 1).

이런 weight의 결과로 이들 한글 자/모음으로 구성된 '각'type의 한글을 random하게 100자를 test pattern으로 하여 인식실험한 결과중에서 일부를 그림 10에 나타낸다.



(그림 8. Network의 output value)

Output value :

X[1][][]	X[2][][]
0.283432 0.297017 0.294803 0.288301	0.872286
0.428192 0.397951 0.406827 0.347349	0.651349
0.278104 0.294899 0.225852 0.282852	0.876519
0.290416 0.256749 0.175555 0.264513	0.915347
0.308943 0.316531 0.263826 0.299957	0.845094
0.250938 0.272859 0.198126 0.274651	0.900466
0.249222 0.270723 0.195527 0.272338	0.902123
0.283261 0.286001 0.234930 0.290003	0.871820
0.275745 0.290824 0.225242 0.284750	0.879883
0.250938 0.272859 0.198126 0.274651	0.900466
0.535165 0.548969 0.572986 0.437928	0.213779
0.739175 0.630154 0.792321 0.489283	0.097468
0.683668 0.585120 0.790054 0.459074	0.149868
0.686900 0.586357 0.793042 0.460462	0.146965
0.698672 0.595117 0.745695 0.465616	0.134776
0.750606 0.638656 0.803666 0.495077	0.089536
0.683713 0.584991 0.790046 0.459878	0.149925
0.784775 0.671156 0.840719 0.516730	0.067581
0.701891 0.600241 0.751085 0.469742	0.190341
0.652124 0.561284 0.693508 0.444748	0.189352
0.750606 0.638656 0.803666 0.495077	0.089536
0.683713 0.584991 0.790046 0.459878	0.149925
0.784775 0.671156 0.840719 0.516730	0.067581
0.701391 0.600241 0.751085 0.469742	0.190341
0.701391 0.600241 0.751085 0.469742	0.190341

(그림 9-a. stage 1 learning의 결과)

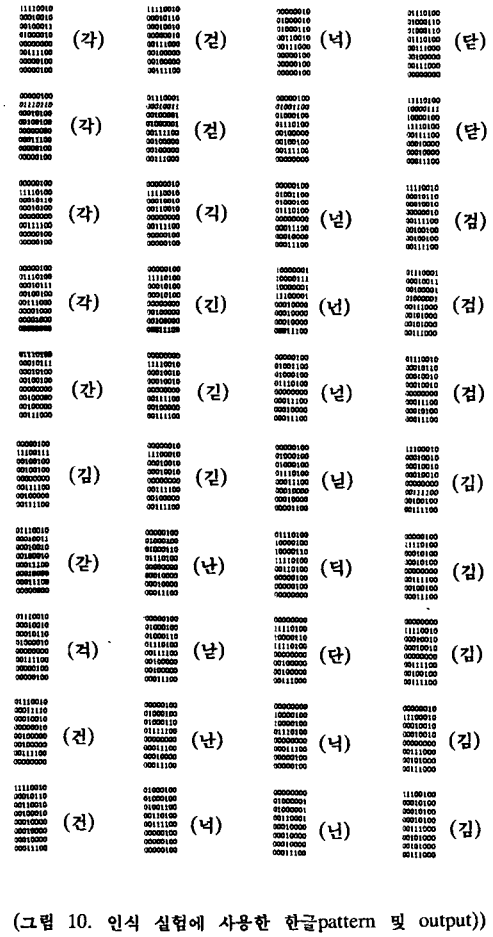
Output value :

X[1][][]	X[2][][]
0.840719 0.645797 0.410892 0.492575	0.856304
0.331154 0.648003 0.407253 0.435846	0.867403
0.213297 0.675993 0.366578 0.397395	0.954454
0.222872 0.673290 0.369607 0.405430	0.949954
0.222584 0.671595 0.374315 0.406518	0.949174
0.188003 0.682829 0.357389 0.374873	0.964714
0.121405 0.705435 0.325957 0.305964	0.983232
0.150603 0.694485 0.342062 0.338852	0.976334
0.114687 0.708993 0.321195 0.297295	0.984660
0.188003 0.682829 0.357389 0.374873	0.964714
0.938135 0.473318 0.640669 0.880426	0.016772
0.896052 0.501279 0.604002 0.840156	0.028800
0.903156 0.496433 0.607123 0.846216	0.026668
0.886163 0.507853 0.596589 0.831453	0.032529
0.939538 0.471683 0.643090 0.882053	0.016378
0.905006 0.496305 0.610018 0.848038	0.025916
0.807983 0.539253 0.554649 0.773486	0.072607
0.802283 0.541315 0.551496 0.769549	0.078747
0.841087 0.527500 0.570547 0.796644	0.052580
0.761972 0.552930 0.535854 0.743527	0.109816
0.905006 0.496305 0.610018 0.848038	0.025916
0.807983 0.539253 0.554649 0.773486	0.072607
0.802283 0.541315 0.551496 0.769549	0.078747
0.841087 0.527500 0.570547 0.796644	0.052580
0.841087 0.527500 0.570547 0.796644	0.052580

(그림 9-b. stage 2 learning의 결과)

item \ stage no.	stage 1	stage 2
learning count	38,000	19,000
recognition ratio	65%	97%
confusion ratio	32	2
error ratio	3	1

(표1. 한글인식 실험 결과)



(그림 10. 인식 실험에 사용한 한글pattern 및 output))

Test pattern으로는 noise도 있고, size가 큰것, 작은것, 약

간 shift된 것등이 포함되어 있다. 요약 표에서 볼수 있듯이 stage 1의 learnig에서는 65% 정도만 올바르게 인식하였으나 stage 2의 learning을 시킨결과 97%까지 인식율이 증가 되었다.

V. 결 론

Coulomb energy network를 한글 인식문제에 적용시켜 perfomance를 측정하여 보았다. 한글인식에 적용하여 본 결과 복잡한 pattern의 인식에는 2 stage learning이 유효하다는 것을 알았다.

Learning결과 clustering이 자연적으로 이루어 졌으며, 길지 않은 learning time(90 minutes/자음)과 20개정도의 test pattern으로 'ㄱ','ㄴ','ㄷ','ㅁ',' ㅏ',' ㅓ',' ㅣ'로 구성된 '각'type의 한글을 인식할 수 있는 neural network를 구현할 수 있었다. 이 coulomb energy network에서는 EBP과 다르게 error의 propagation이 다른 layer에 영향을 미치지 않게됨으로써 system이 modular하게 되고 따라서 각 layer별로 learning시킬수 있는 장점도 있다.

그러나 구별하여야 할 자/모음이 많을수록 confusion은 증가하게 되고 더 많은 training pattern, learning time이 요구되어 진다. 따라서 이런 문제들을 해결하기 위하여 neocognitron [Fukushima,1986]에서의 lateral inhibition기능 및 selective attention기능 등이 고려되어야 한다.

REFERENCES

[1] Scofield, C.L. : Learning internal representations in the coulomb energy network. ICNN, Vol I, 271 - 275(Auguest, 1988)

[2] Bachmann, C.M., Cooper, L.N., Dembo,A.,Zeitouni,O.: A relaxation model for memory with high density storage. Proc. Natl. Acad. Sci. USA 21, 7529 - 7531(November, 1987)

[3] Hopfield, J.J. : Neural network and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. USA 79, 2554 - 2558 (April, 1982)

[4] Hopfield, J.J. : Neurons with graded response have collective computational properities like those of two-state neurons. Proc. Natl. Acad. Sci. USA 81, 2088 - 3092(May, 1984)

[5] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. : Learning internal representations by error propagation in D.E.Rumelhart and J.L.McClelland(Eds) Parallel Distribution Processing, MIT Press, 318 - 364(1986)

[6] Fukushima, K. : A neural network model for selective attention in visual pattern recognition, Biological Cybernetics, 55(1), 5 - 15(1986)