A Basic Introduction to Artificial Neural Network (ANN)

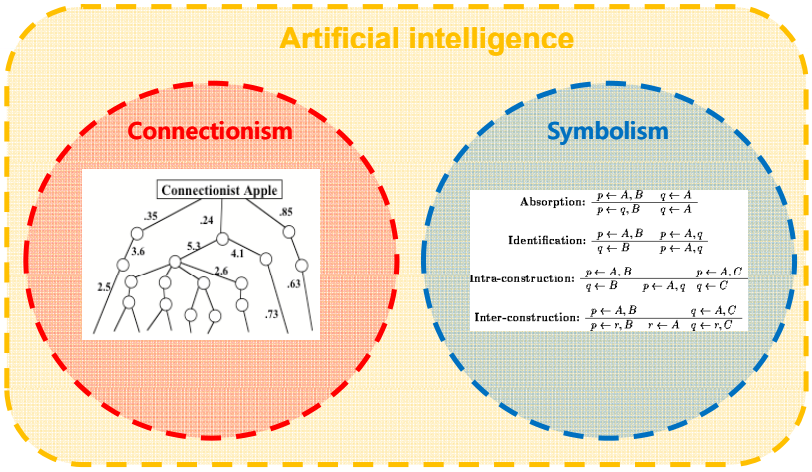
도대체 인공신경망이란 무엇인가?

INDEX

1. Introduction to ANN

Introduction to ANN

- Connectionism vs Symbolism



Introduction to ANN

• Connectionism vs Symbolism

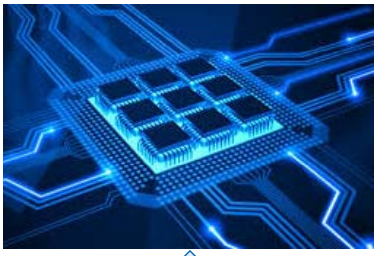
- Connectionism과 Symbolism은 인공지능 분야를 대표하는 두 가지 고전적 접근방법

- **Symbolic AI**는 지식을 symbol과 그들간의 relation 또는 Logic으로 표현.
  - 문제 해결 또는 새로운 지식 추론을 위하여 symbolic instance와 그들간의 relation에 특정 algebraic Inference을 적용한다
  - e.g. Logical Inference in Ontology, Probabilistic Inference, Fuzzy inference)
- **Connectionist AI**는 지식을 network상에 분산된 형태로 표현.
  - 생명체의 신경 구조를 모방하여 지능적 프로세스를 발현시킴으로써 인식/학습/추론 문제를 해결
  - e.g. Artificial Neural Net

Introduction to ANN

• The Brain vs. Computer

1. 10 billion neurons
2. 60 trillion synapses
3. Distributed processing
4. Nonlinear processing
5. Parallel processing



1. Faster than neuron (10^{-9}sec)
   cf. neuron: 10^{-3}sec
2. Central processing
3. Arithmetic operation (linearity)
4. Sequential processing

AI Lab, Hanyang University

Introduction to ANN

- **Biological inspiration**
  - 생명체의 경우 주변 환경에 적응적 행동 및 학습을 수행함
  - 생명체는 네트워크 형태의 "신경구조(nervous system)"를 사용

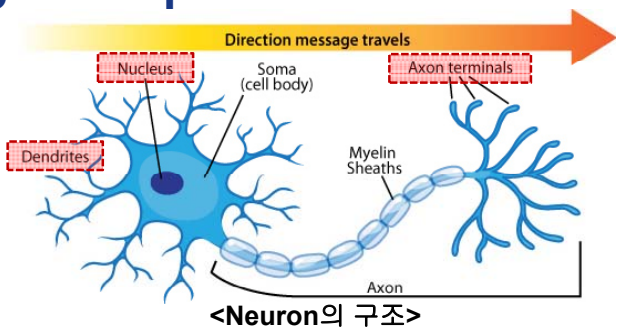
<Nervous system>

Introduction to ANN

- **Biological inspiration**
  - Artificial neural network는 이러한 동물의 신경구조(nervous system)를 모방하여 기존의 Symbolic AI 로 해결하기 어려할 수 없었던 문제를 해결하고자 하는 접근.
    - 신경구조(nervous system)는 **뉴런(neuron)**이라는 간단한 형태의 기본 unit의 연결망으로 구성
    - 뉴런의 기본 형태를 모사하여 신경구조의 기능과 행동을 발현하고자 함

Introduction to ANN

- **Biological inspiration**

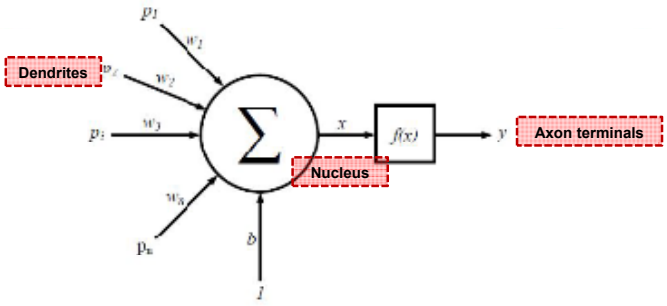


<Neuron의 구조>

- 뉴런은 Dendrites를 통해, 다수의 타 뉴런들로 부터의 "입력신호"를 전달 받음
- Nucleus(핵)을 거친 신호는 Axon terminals를 통해 다음 뉴런으로 출력 전파

Introduction to ANN

- **Biological inspiration**



- 뉴런의 입력, 처리, 출력(전파)를 각각 담당하는 Dendrites, Nucleus, Axon terminals를 모방한 구조
- ANN에서 위와 같은 구조의 unit을 **Perceptron**이라 한다

Types of simple ANNs

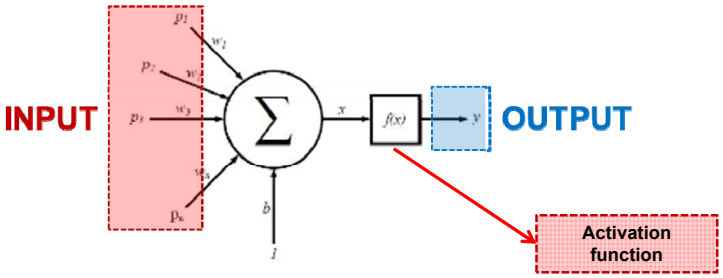
입력형식	학습방식	Artificial neural network model
이진입력	지도(supervised) 학습	Hopfield memory, BAM
실수입력	지도(supervised)학습	Perceptron, Backpropagation neural network
	비지도(unsupervised) 학습	Self-Organizing Map(SOM)

< ANN 모델 분류 예 >

2. Perceptron

Perceptron

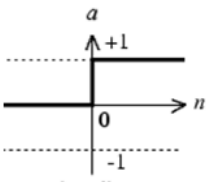
- **What is Perceptron?**
  - Perceptron은 ANN을 구성하는 단위
  - 복수개의 Input을 입력 받은 뒤 처리를 거친 후, 하나의 Output을 반환
  - 입력 신호 p_i와 가중치 w_i의 곱을 모두 합한 값에 따라 출력신호 y의 흥분여부가 결정 된다



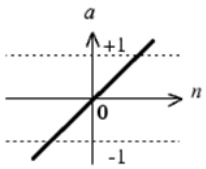
<Perceptron의 구조>

Perceptron

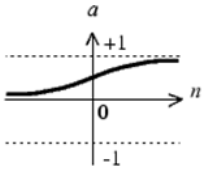
- **Activation function**
  - 자주 사용되는 Activation function f_n



<hard limiter>



<Linear>



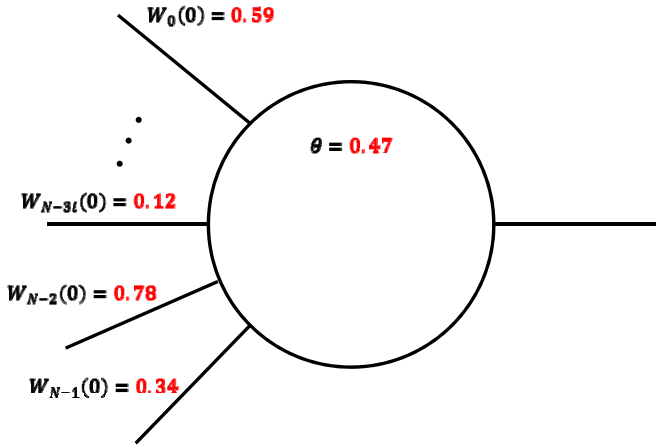
<Sigmoid>

Perceptron

- **Learning (Perceptron)**
  - 지도학습(Supervised Learning), 이진 & 아날로그 입력처리
  - 전체 출력뉴런들에 대하여 계산된 출력값과 목표값과의 차이를 최소화시킴
    → Widrow-Hoff rule(delta rule)
  - 만일 계산된 출력값과 목표값 간에 차이가 없으면 연결 가중치(w_i)는 변경되지 않으며,차이가 있으면 차이를 줄이는 방향으로 가중치를 변경.

Perceptron

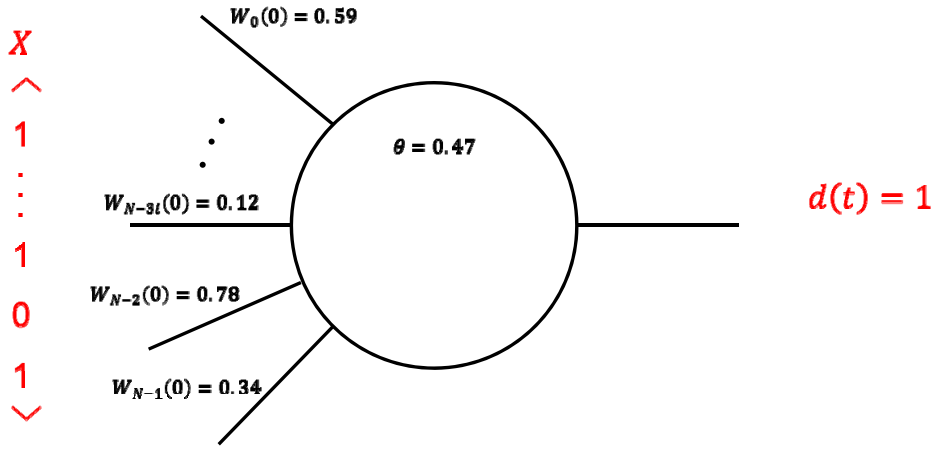
- **Widrow-Hoff rule(delta rule)**



1. 가중치($W_i(0)$)와 임계치(θ)를 임의의 작은 값으로 초기화

Perceptron

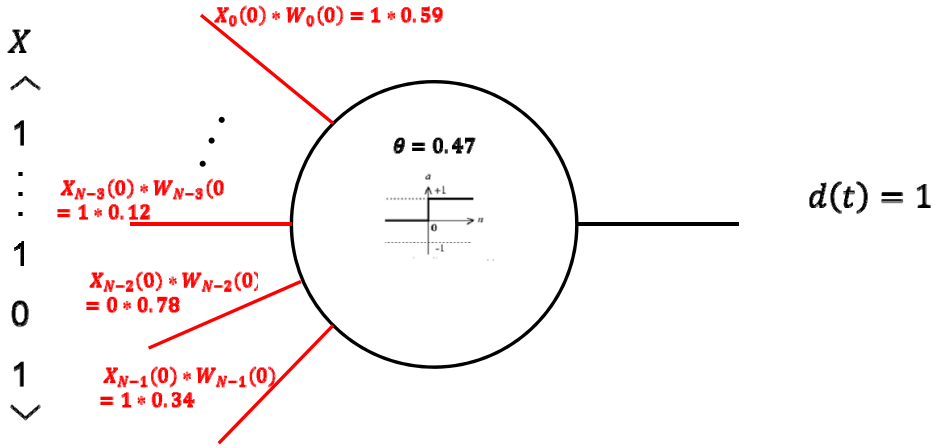
- **Widrow-Hoff rule(delta rule)**



2. 새로운 입력패턴(X_0, X_1, ...)과 목표출력 패턴(d(t))을 제시

Perceptron

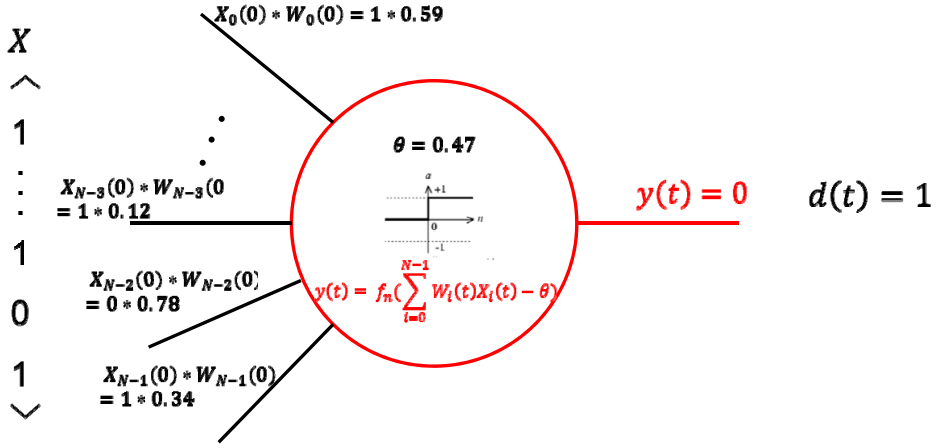
- **Widrow-Hoff rule(delta rule)**



3. Activation function(hard limiter) f_n를 사용하여 실제 출력값(y(t))을 계산

Perceptron

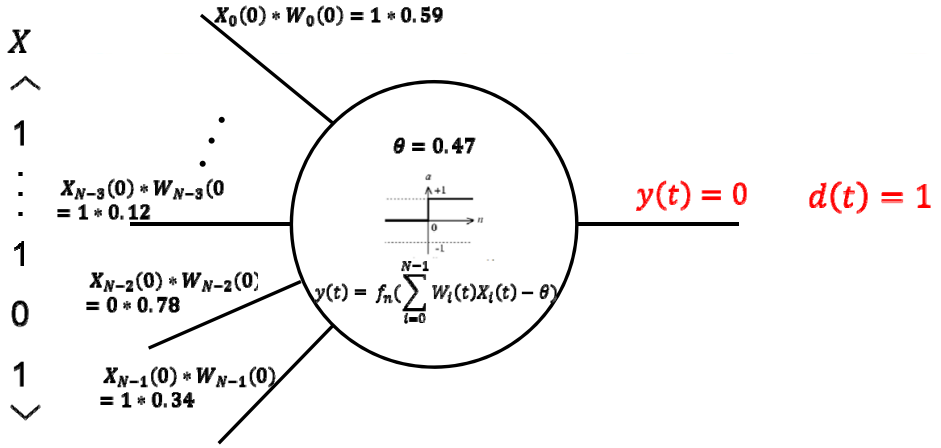
- **Widrow-Hoff rule(delta rule)**



3. Activation function(hard limiter) f_n를 사용하여 실제 출력값(y(t))을 계산

Perceptron

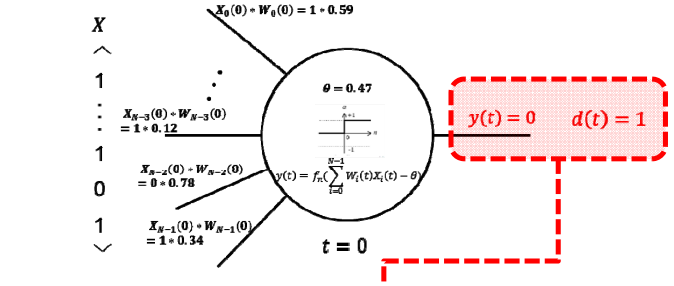
- **Widrow-Hoff rule(delta rule)**



4. 목표값($d(t)$)과 출력값($y(t)$)을 이용한 가중치를 갱신

Perceptron

- **Widrow-Hoff rule(delta rule)**

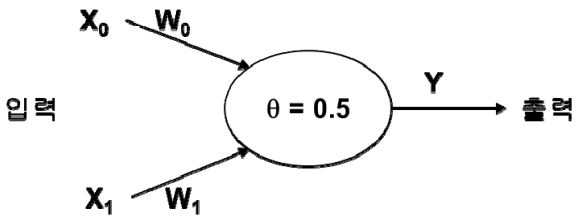


$$W_i(t+1) = W_i(t) + \alpha[d(t) - y(t)] * X_i(t), (0 \le i \le N-1)$$

4. 목표값($d(t)$)과 출력값($y(t)$)을 이용한 가중치를 갱신

Perceptron

- **Example(AND Operation using Perceptron)**



입력		출력
X_0	X_1	AND
0	0	0
0	1	0
1	0	0
1	1	1

- 뉴런에 입력되는 가중치의 합이 임계치를 초과하면 1, 아니면 0
  →Activation function → hard limiter
- AND

$$\begin{aligned} 0\times W_0 + 0\times W_1 &= 0 && < 0.5 \\ 0\times W_0 + 1\times W_1 &= W_1 && < 0.5 \\ 1\times W_0 + 0\times W_1 &= W_0 && < 0.5 \\ 1\times W_0 + 1\times W_1 &= W_0 + W_1 && > 0.5 \end{aligned}$$

- W0, W1 : 0.3 or 0.4

Perceptron

- **Example(XOR Operation using Perceptron)**

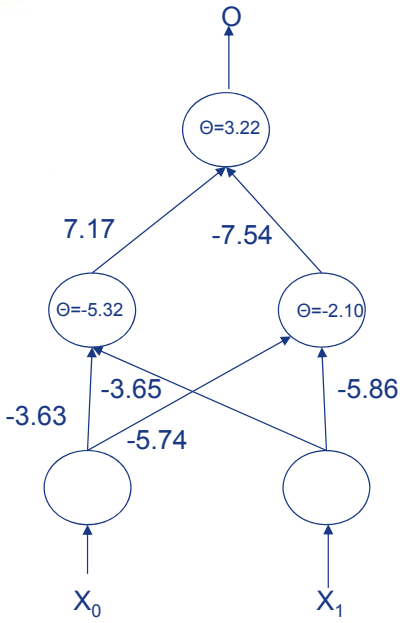
$$
\begin{aligned}
0\times W_0 + 0\times W_1 = 0 &\quad < 0.5 \\
0\times W_0 + 1\times W_1 = W_1 &\quad > 0.5 \\
1\times W_0 + 0\times W_1 = W_0 &\quad > 0.5 \\
1\times W_0 + 1\times W_1 = W_0 + W_1 &\quad < 0.5
\end{aligned}
$$

→ 만족하는 **W0, W1** 는 존재하지 않음
→ 하나의 **Perceptron**으로는
간단한 **XOR** 문제도 해결하지 못함
XOR : linearly non-separable

입력		출력
X_0	X_1	XOR
0	0	0
0	1	1
1	0	1
1	1	0

- 이러한 문제를 해결하기 위해서 2개 또는 3개의 층(layer)을 사용
- Backpropagation Neural Network (Multi-layer Perceptron)
  →**3층 Perceptron**으로 어떤 문제도 (근사적으로) 해결가능
- Perceptron은 Multi-layer Perceptron 및 Error Back propagation Algorithm 의 기반

XOR 다층 퍼셉트론 예

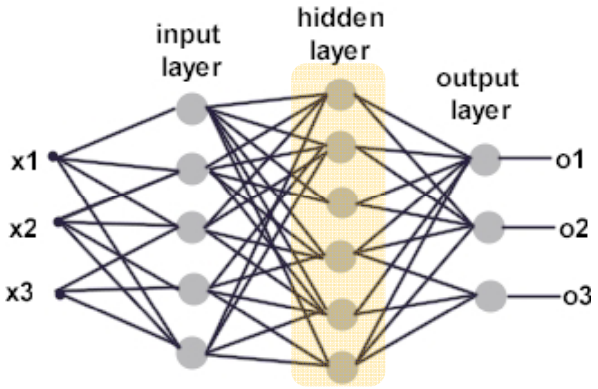


x_0	x_1	O
0	0	0
0	1	1
1	0	1
1	1	0

3. Backpropagation Neural Network

Backpropagation Neural Network

- **What is Backpropagation Neural Network?**



<Backpropagation Neural Network>

- input layer과 output layer 사이에 하나 이상의 hidden layer을 가지는 단방향 신경회로망

Backpropagation Neural Network

What is Backpropagation Neural Network?

- 단층 퍼셉트론의 선형분리(linearly non-separable) 문제점을 해결(XOR operation 등 구현가능)
- **일반적인 continuous function approximation 문제 해결을 위해 널리 사용**
- 80년대 중반 등장한 Error Back propagation Algorithm을 바탕으로 함
  → 일반화된 델타 규칙(generalized delta rule)
- Learning?
  → 원하는 목표값(d)과 실제 출력값(o) 사이의 오차제곱합으로 정의된 Error Function의 값을 최소화하는 방식으로 학습

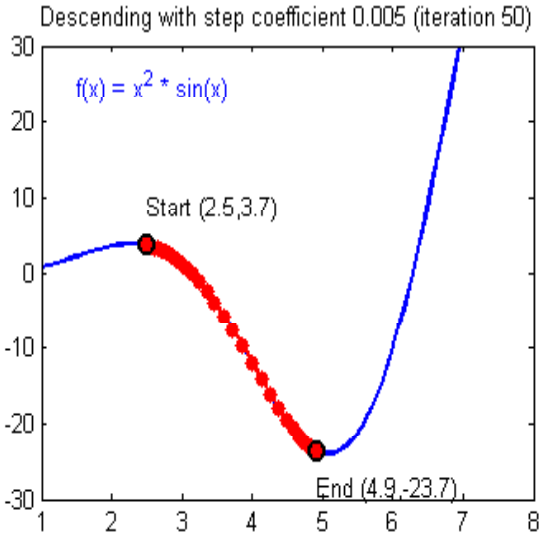
Backpropagation Neural Network

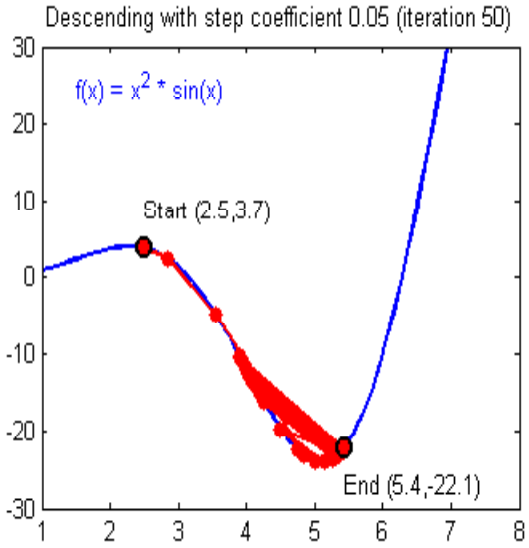
Learning? Error backpropagation!

- Error backpropagation Algorithm 개념
  - hidden layer의 학습을 위해 output layer에서 발생한 오류를 이용하여 hidden layer 가중치 재계산
  - 이 값을 다시 input layer으로 역전파(backpropagation)시켜 가중치를 재계산
  - output layer의 오류를 Gradient Descent Method 기법으로 최소화함
- 문제점
  - 상위층의 목표값과 실제 출력값 간의 오류를 하위층으로 역전파시키는 것은 생물학적 현상과 일치하지 않음
    - ✓ 하위층의 각 뉴런이 상위층의 목표값을 알지 못하는 경우가 일반적인 생물학적 현상임
- **보편적으로 많이 사용되는 인공신경망 학습 방법**

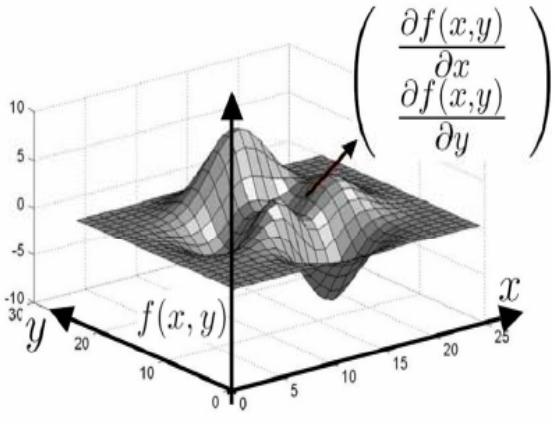
AI Lab, Hanyang University

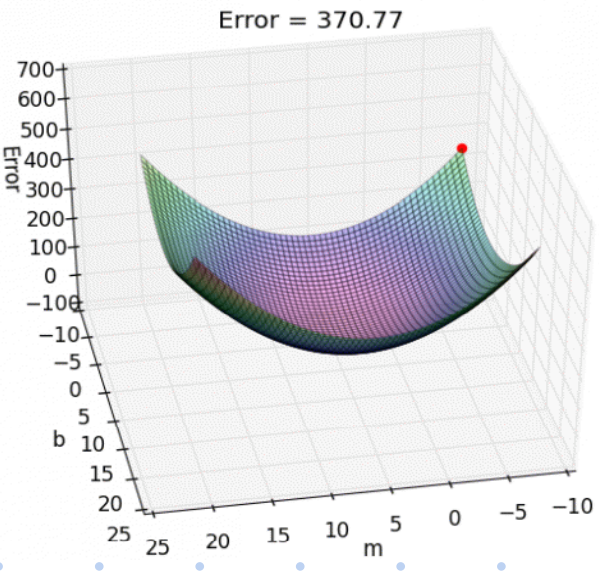
Gradient Descent Method





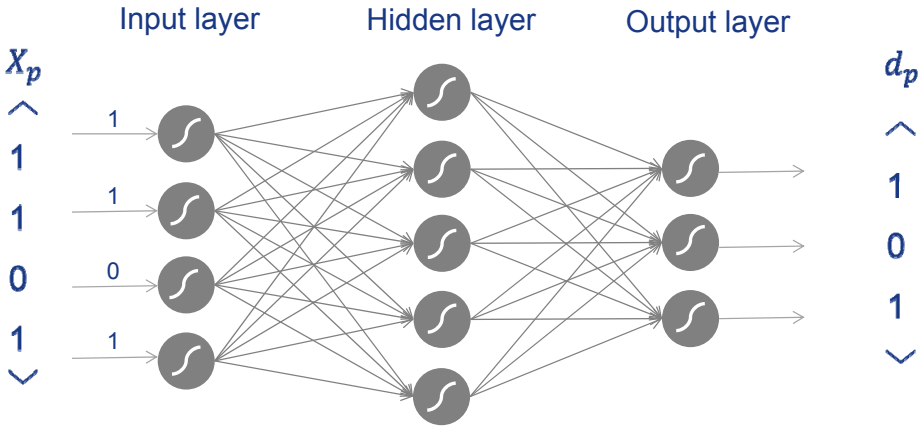
Gradient Descent Method





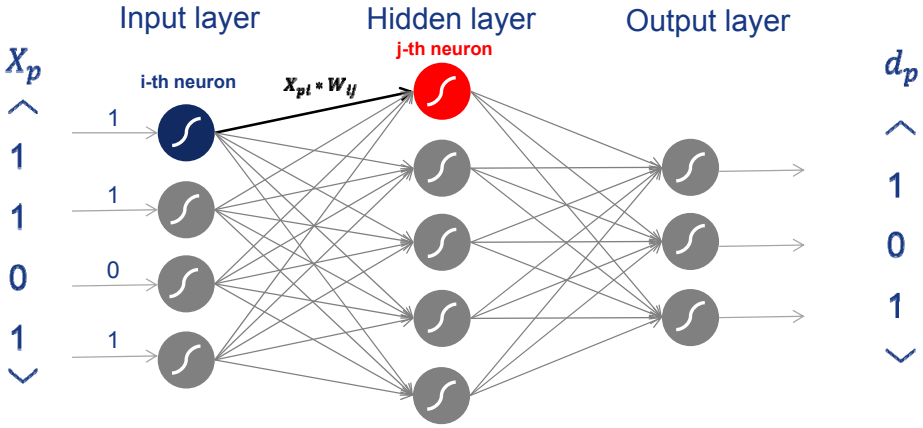
Backpropagation Neural Network

- **Learning? Error backpropagation!**



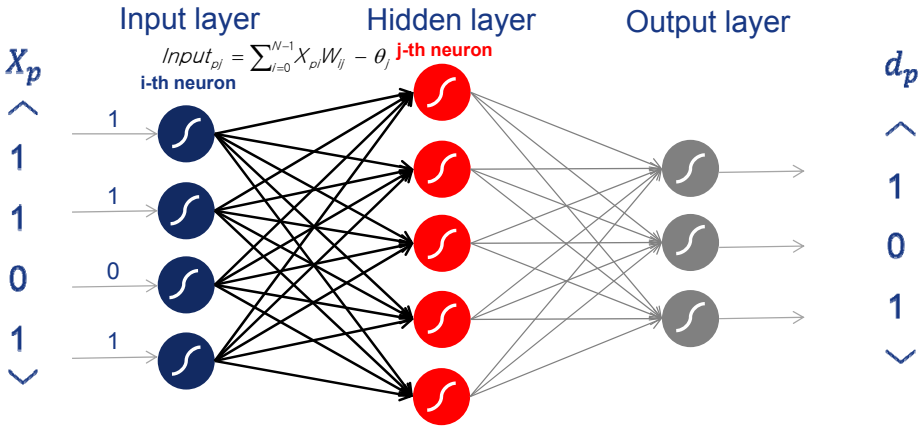
Backpropagation Neural Network

- **Learning? Error backpropagation!**



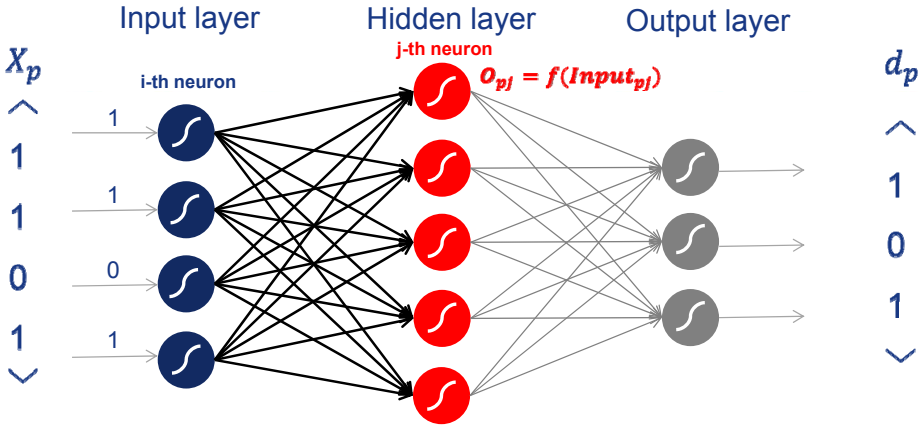
Backpropagation Neural Network

- **Learning? Error backpropagation!**



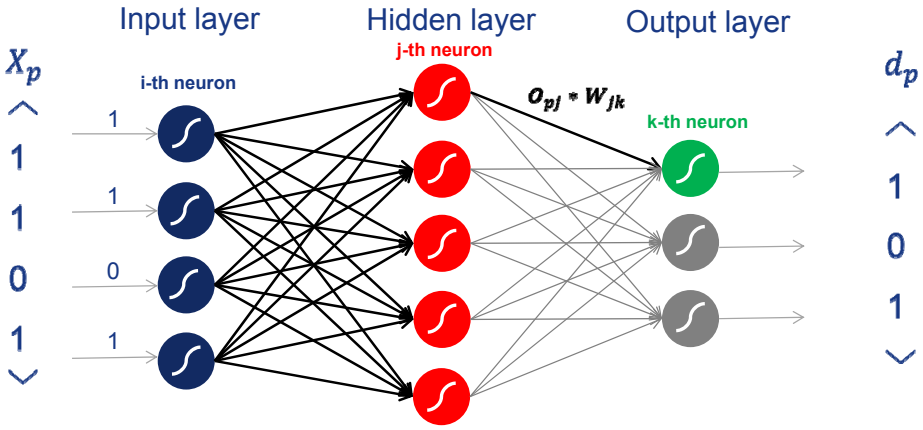
Backpropagation Neural Network

- **Learning? Error backpropagation!**



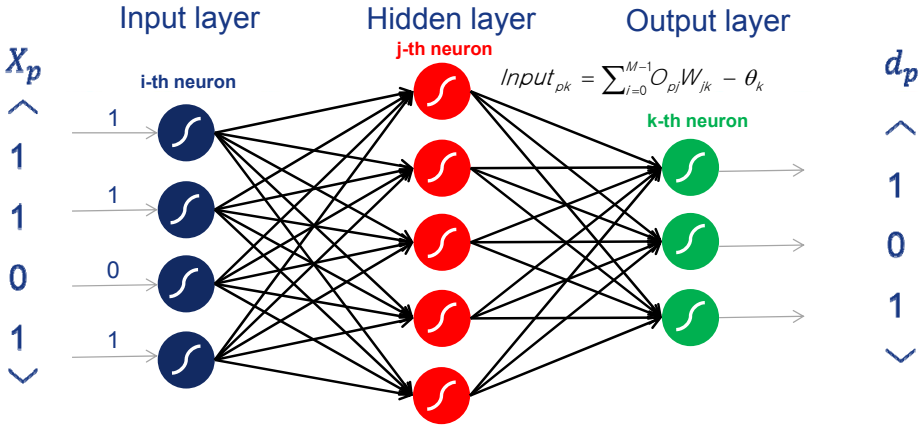
Backpropagation Neural Network

- **Learning? Error backpropagation!**



Backpropagation Neural Network

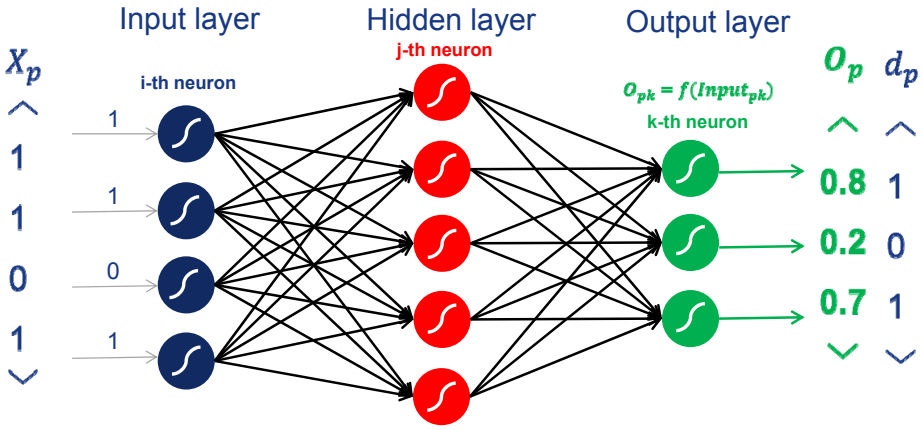
- **Learning? Error backpropagation!**



AI Lab, Hanyang University

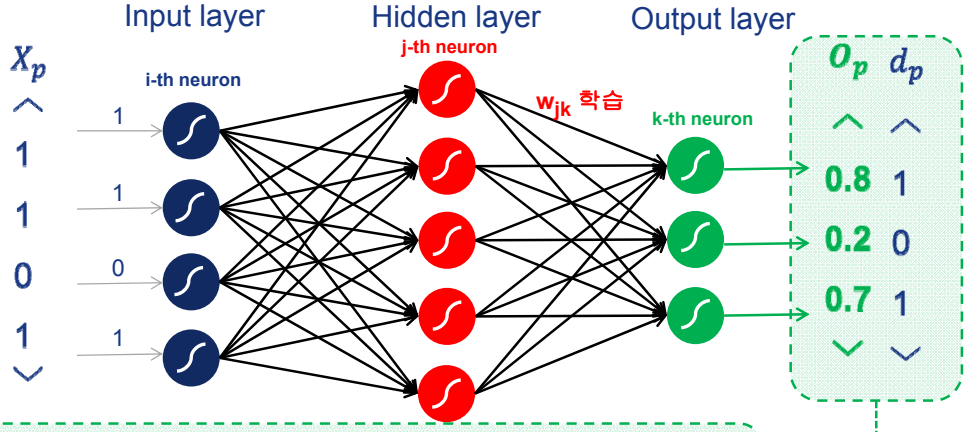
Backpropagation Neural Network

- **Learning? Error backpropagation!**



Backpropagation Neural Network

- **Learning? Error backpropagation!**



$$E_p = \frac{1}{2}\sum_k (d_{pk} - O_{pk})^2$$

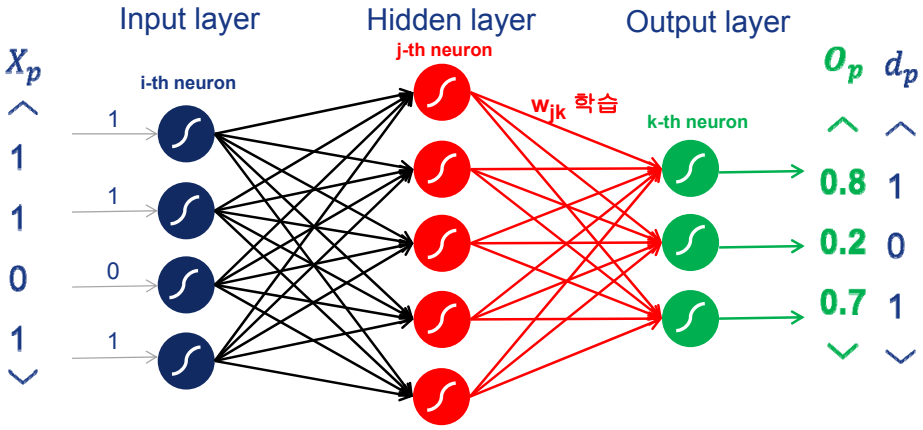
$$-\frac{\partial E_p}{\partial W_{jk}} = (d_{pk} - O_{pk})\frac{\partial (O_{pk})}{\partial (input_{pk})}\frac{\partial (input_{pk})}{\partial (W_{jk})} = \boxed{(d_{pk} - O_{pk})O_{pk}(1 - O_{pk})}O_{pj}$$

δ_{pk}

Output layer 오류 δ_{pk} 계산 및 w_{jk} 학습

Backpropagation Neural Network

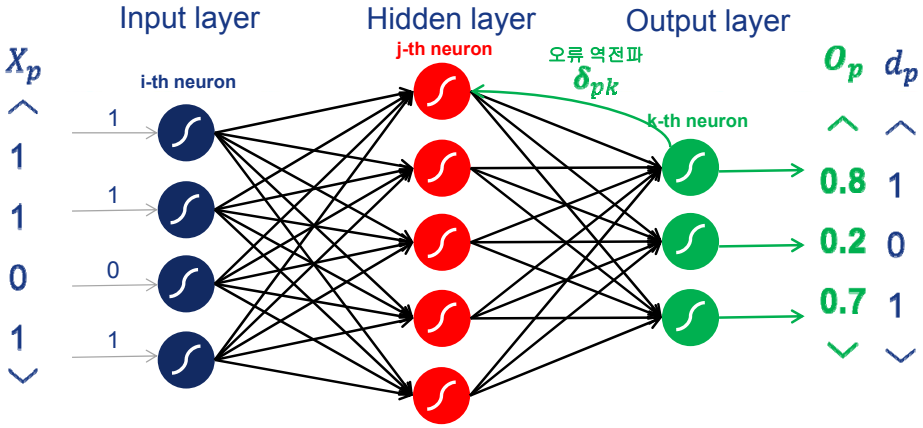
- **Learning? Error backpropagation!**



$$W_{jk}(t+1) = W_{jk}(t) + \eta * \delta_{pk} * O_{pj}$$

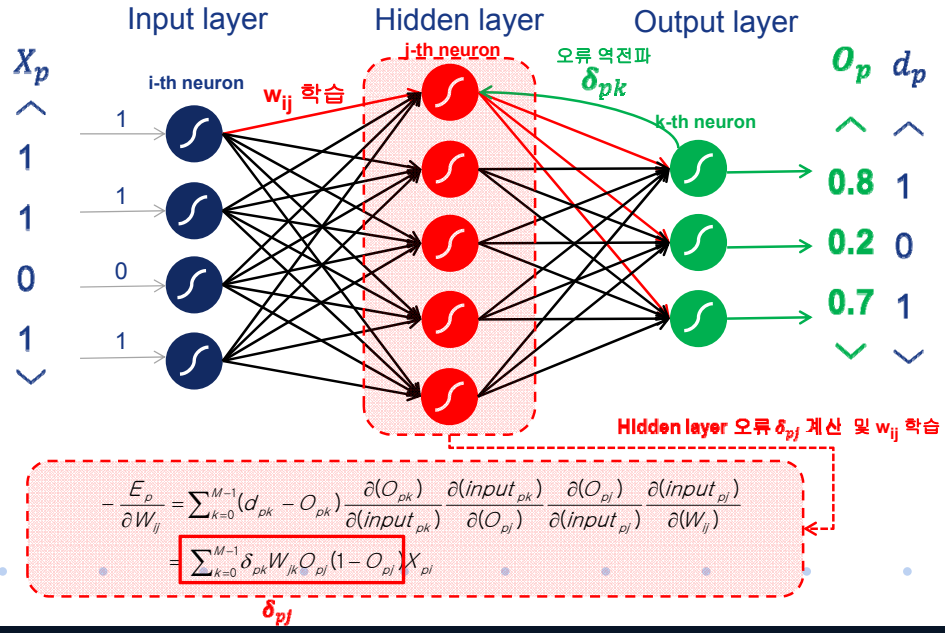
Backpropagation Neural Network

- **Learning? Error backpropagation!**



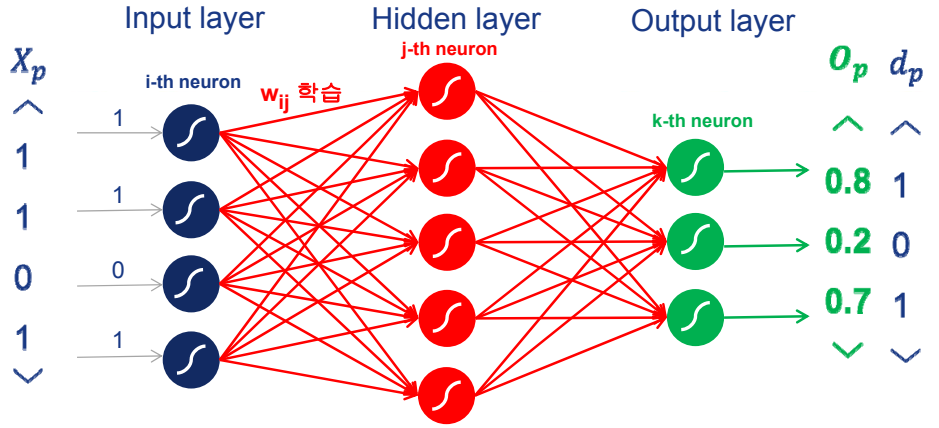
Backpropagation Neural Network

- **Learning? Error backpropagation!**



Backpropagation Neural Network

- **Learning? Error backpropagation!**



$$W_{ij}(t+1) = W_{ij}(t) + \eta * \delta_{pj} * X_{pi}$$

Backpropagation Neural Network

- **Learning? Error backpropagation!**

Activation function → Sigmoid function

1. 가중치(W)와 임계치(θ)를 초기화
2. 입력(X)과 목표 출력(d) 을 제시
3. 제시된 입력벡터를 이용하여 hidden layer j번째 뉴런으로의 입력값 계산

$$Input_{pj} = \sum_{i=0}^{N-1} X_{pi} W_{ij} - \theta_j$$

4. Sigmoid function를 사용하여 hidden layer의 출력(O_{pj})을 계산

$$O_{pj} = f(Input_{pj})$$

5. hidden layer의 출력을 이용하여 output layer 뉴런 k로의 입력값을 계산
6. Sigmoid function을 사용하여 output layer의 출력(O_{pk})을 계산

$$O_{pk} = f(Input_{pk})$$

Backpropagation Neural Network

- **Learning? Error backpropagation!**

Activation function → Sigmoid function

7. 신경망 가중치 Gradient (각 가중치 W에 대한 E_p의 변화율)를 구한다.

$$-\frac{\partial E_p}{\partial W_{jk}} = (d_{pk} - O_{pk}) \frac{\partial (O_{pk})}{\partial (input_{pk})} \frac{\partial (input_{pk})}{\partial (W_{jk})} = \underbrace{(d_{pk} - O_{pk}) O_{pk} (1 - O_{pk})}_{\delta_{pk}} O_{pj}$$

$$-\frac{\partial E_p}{\partial W_{ij}} = \sum_{k=0}^{M-1} (d_{pk} - O_{pk}) \frac{\partial (O_{pk})}{\partial (input_{pk})} \frac{\partial (input_{pk})}{\partial (O_{pj})} \frac{\partial (O_{pj})}{\partial (input_{pj})} \frac{\partial (input_{pj})}{\partial (W_{ij})}$$

$$= \sum_{k=0}^{M-1} (d_{pk} - O_{pk}) O_{pk} (1 - O_{pk}) W_{jk} O_{pj} (1 - O_{pj}) X_{pi}$$

$$= \underbrace{\sum_{k=0}^{M-1} \delta_{pk} W_{jk} O_{pj} (1 - O_{pj})}_{\delta_{pj}} X_{pi}$$

① $E_p = \frac{1}{2} \sum_k (d_{pk} - O_{pk})^2$

② $\frac{\partial (input_{pk})}{\partial W_{jk}} = O_{pj}, \quad \frac{\partial (input_{pk})}{\partial O_{pj}} = W_{jk}$

③ $\frac{\partial (input_{pj})}{\partial (W_{ij})} = X_{pi}$

④ $y = f(x) = \frac{1}{1 + e^{-x}} \quad \therefore \frac{\partial y}{\partial x} = y(1 - y)$

AI Lab, Hanyang University

Backpropagation Neural Network

- **Learning? Error backpropagation!**

8. Gradient Descent 기법으로 Hidden-Output연결 가중치를 갱신한다.

$$W_{jk}(t+1) = W_{jk}(t) + \eta * \delta_{pk} * O_{pj}$$

9. Gradient Descent 기법으로 Input-Hidden연결 가중치를 갱신한다.

$$W_{ij}(t+1) = W_{ij}(t) + \eta * \delta_{pj} * X_{pi}$$

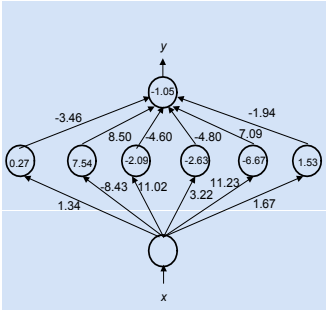
10단계. 모든 학습쌍에 대하여 전부 학습 할 때까지 2로 분기하여 반복 수행한다.

12단계. 출력층의 오차합 E가 허용값 이하이거나 최대 반복회수보다 크면 종료, 그렇지 않으면 2로 가서 다시 반복한다.

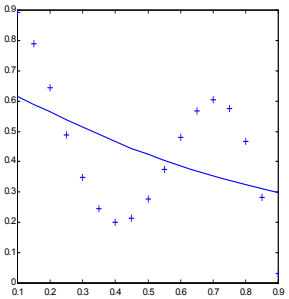
Backpropagation Neural Network

- 함수 근사화(function approximation)

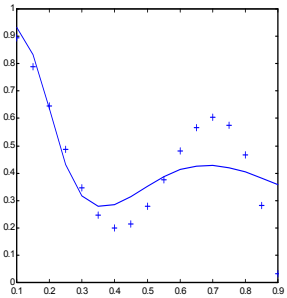
$$y = 0.5(\cos\ 8x + \sin\ 4x - x + 0.8)$$



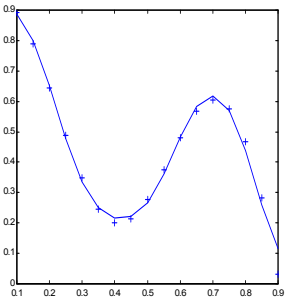
- 위 관계식으로 부터 생성된 임의의 **15**개 **<x,y>** 쌍을 학습데이터로 사용 **(***학습데이터: 아래 그림의 점**)**
- 입력층: **1**개 뉴런, 은닉층: **6**개 뉴런, 출력층: **1**개 뉴런 사용



주어진 데이터로 1,000회학습후



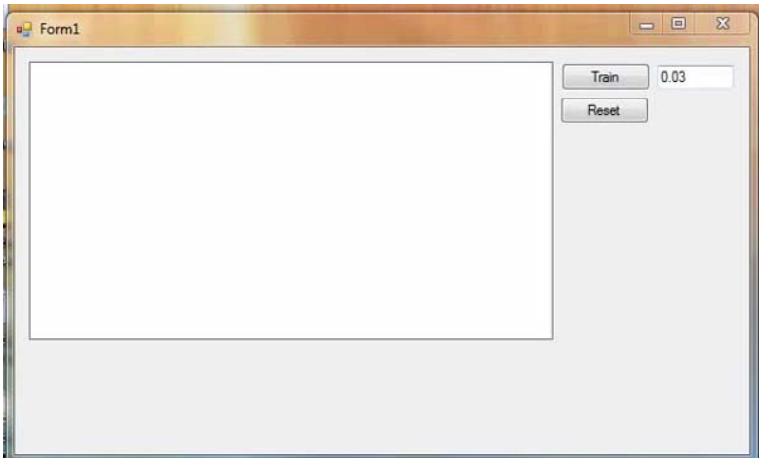
10,000회학습후



20,000회학습후

Backpropagation Neural Network

- **Example**



https://www.youtube.com/watch?v=6lgAzEomn-4

5. Hopfield memory

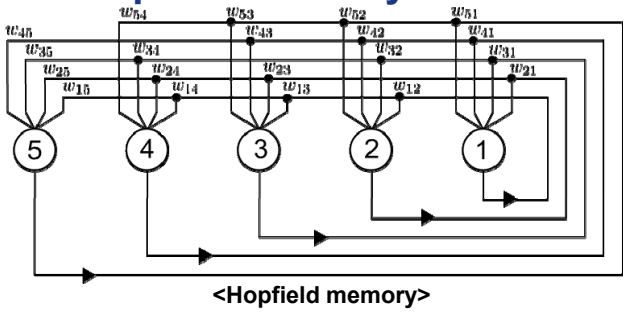
Types of simple ANNs

입력형식	학습방식	Artificial neural network model
이진입력	지도(supervised) 학습	Hopfield memory, BAM
실수입력	지도(supervised)학습	Perceptron, Backpropagation neural network
	비지도(unsupervised) 학습	Self-Organizing Map(SOM)

< ANN 모델 분류 예 >

Hopfield memory

- **What is Hopfield memory?**

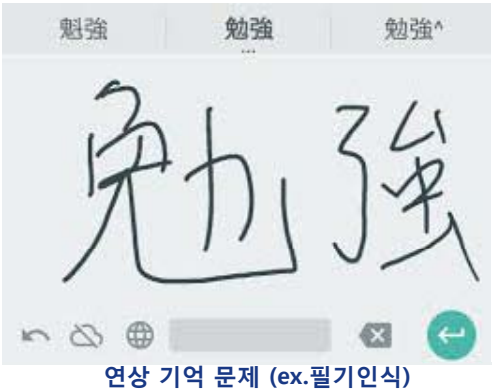


<Hopfield memory>

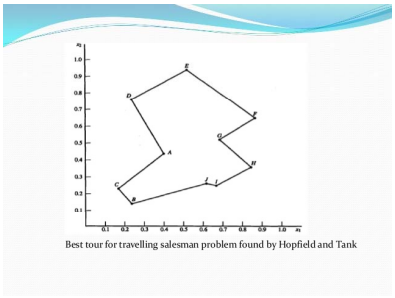
- Hopfield memory는 자신을 제외한 모든 뉴런과 양방향으로 상호 연결된 형태의 ANN
- Activation function으로 hard limiter를 사용
- 기본 모델은 bipolar 값(+1, -1)을 사용
- 연상기억 또는 최적화 문제를 푸는데 주로 사용
- 다른 종류의 ANN model과 달리 점진적 학습을 하지 않고, 초기 학습패턴의 외적합(sum of outer product)을 사용하여 연결가중치를 만듦
- 하나의 뉴런층을 사용하므로 입력벡터와 출력벡터의 차원이 동일(Auto associative Memory)

Hopfield memory

- What kind of problem can I solve with Hopfield memory?



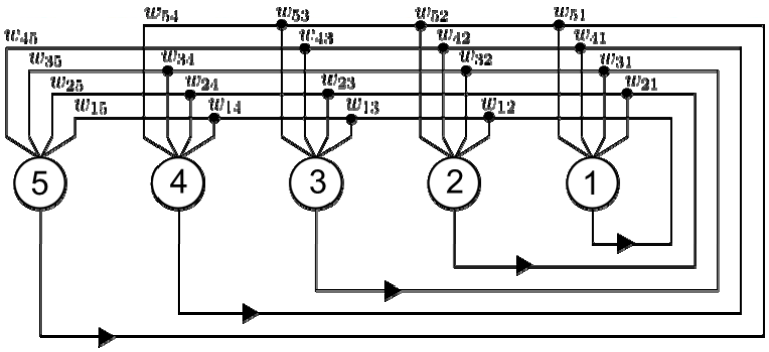
연상 기억 문제 (ex.필기인식)



최적화 문제 (ex.Traveling salesman problem)

Hopfield memory

- How does Hopfield memory work?



<Hopfield memory>

Hopfield memory

- **How does Hopfield memory work?**

Activation function → Hard limiter

1. M개의 패턴을 이용하여 N개 뉴런 사이의 연결 가중치를 지정
   (W_{ij}는 뉴런i에서 뉴런j로의 연결 가중치)

$$W_{ij} = \begin{cases} \sum_{s=0}^{M-1} X_i^s X_j^s \ (i \neq j) \\ 0 \ (i = j) \end{cases} \qquad 0 \le i, j \le N-1$$

2. 미지의 입력 패턴을 Hopfield memory에 제시

$$\mu_j(0) = x_i \qquad 0 \le i \le N-1$$

3. 뉴런들의 출력과 가중치를 곱한 값을 합하여 Activation function을 통과시킴

$$\mu_j(t+1) = f_b\left(\sum_{i=0}^{N-1} W_{ij} * \mu_i(t)\right) \qquad 0 \le i \le N-1$$

4. 수렴(뉴런의 출력 변화가 없는 상태)할 때까지 3을 반복

Hopfield memory

- **Learning(Hopfield memory)**

Example

1. 두 개의 학습 패턴 $P_0 =< 1,1,1,-1 >, P1 =< 1,-1,1,-1 >$을 Hopfield memory에 제시

2. $$W_{ij} = \begin{cases} \sum_{s=0}^{M-1} X_i^s X_j^s \ (i \neq j) \\ 0 \ (i = j) \end{cases} \qquad 0 \le i, j \le N-1$$

→ $W_{01} = X_0^0 X_1^0 + X_0^1 X_1^1 = (1 \times 1) + (1 \times (-1))$
$W_{02} = X_0^0 X_2^0 + X_2^1 X_2^1 = (1 \times 1) + (1 \times 1)$
$W_{03} = X_0^0 X_3^0 + X_3^1 X_3^1 = (1 \times (-1)) + (1 \times (-1))$
…

$$W = \begin{bmatrix} 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & -2 \\ -2 & 0 & -2 & 0 \end{bmatrix}$$

Hopfield memory

- **Learning(Hopfield memory)**

Example

3. 새로운 입력 패턴 $P_{new} =< 1,1,-1,-1 >$ 을 제시. 수렴할 때까지 W행렬과 곱해줌

(a) $\mu_0(0) = 1, \mu_1(0) = 1, \mu_2(0) = -1, \mu_3(0) = -1$

(b) $\mu_j(t+1) = f_b(\sum_{i=0}^{N-1} W_{ij} * \mu_i(t)) \quad 0 \le i \le N-1$

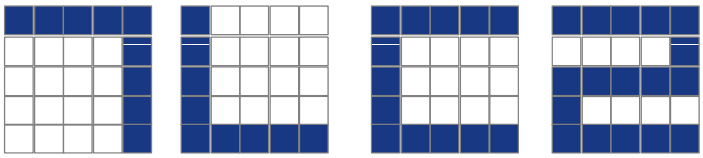
(c) $\mu(1) = [1\ 1\ -1\ -1] * \begin{bmatrix} 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & -2 \\ -2 & 0 & -2 & 0 \end{bmatrix} = f_b([0\ 0\ 4\ 0]) = [1\ 1\ 1\ 1]$

$\mu(2) = [1\ 1\ 1\ 1] * \begin{bmatrix} 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & -2 \\ -2 & 0 & -2 & 0 \end{bmatrix} = f_b([0\ 0\ 0\ -4]) = [1\ 1\ 1\ -1]$

$P_0 =< 1,1,1,-1 >$에 수렴

Hopfield memory

- 패턴이미지 크기: (5x5) 대상 : {ㄱ, ㄴ, ㄷ, ㄹ}으로 학습

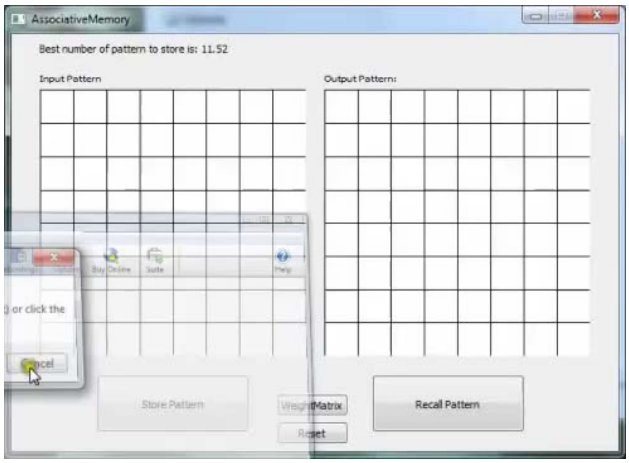


  - 회로망 크기에 대한 저장 가능한 패턴수 문제
    - Hopfield에서는 뉴런 수가 N인 경우 일반적으로 0.15N개의 패턴 기억 가능
    - 이 예제의 경우 25x0.15 = 3.75 → 4개 미만의 패턴 인식

- Hopfield 신경망의 큰 문제점은 수렴결과가 최적인지 보장 안됨
  - 잘못된 기억을 연상해 낼 수 있음

Hopfield memory

- **Example of Hopfield memory**



https://www.youtube.com/watch?v=EGazcWEJGuY

7. Self Organizing Map(SOM)

Self Organizing Map(SOM)

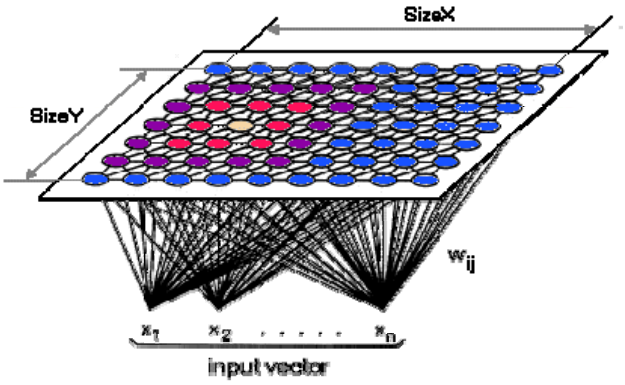
입력형식	학습방식	Artificial neural network model
이진입력	지도학습	Hopfield memory, BAM
실수입력	지도학습	Perceptron, Backpropagation neural network
	비지도학습	**Self-Organizing Map(SOM)**

<ANN 모델의 분류>

Self Organizing Map (SOM)

- **What is SOM?**

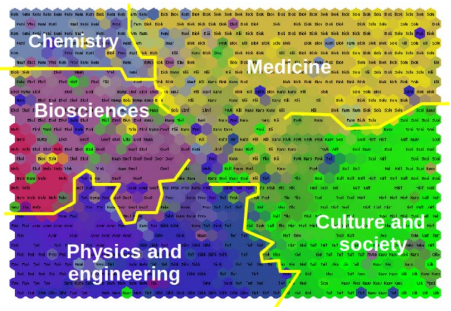
<Self Organizing Map>



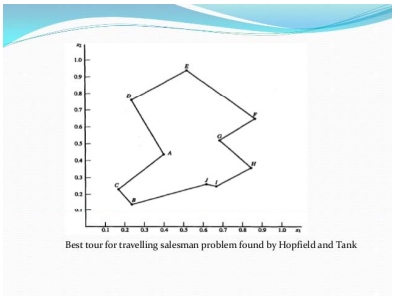
- 인접한 출력뉴런들은 비슷한 기능을 수행할 것이라는 예측 (뇌의 부분에 따라 인지 기능의 종류가 다르고, 뇌의 비슷한 부분은 비슷한 인지 기능을 수행한다는 가정)
- 입력벡터와 가장 가까운 출력뉴런(승자뉴런) 뿐만 아니라 위상적으로 이웃한 뉴런들도 함께 학습시킴

Self Organizing Map(SOM)

- **What kind of problem can I solve with SOM?**



Clustering & Classification



최적화 문제 (ex.Traveling salesman problem)

Self Organizing Map(SOM)

- **Learning (SOM)**

1. 연결가중치를 초기화
→ N개의 입력으로부터 M개의 출력 뉴런 사이의 연결강도를 임의의 값으로 초기화

2. 새로운 입력패턴을 입력뉴런에 제시한다.

3. 입력벡터와 모든 출력뉴런들과의 거리(입력벡터와 가중치벡터의 거리)를 계산

$$d_j = \sum_{i=0}^{N-1} (X_i(t) - W_{ij}(t))^2$$

4. 최소거리를 가지는 승자뉴런을 구함. d_j가 최소인 출력 뉴런 j*를 선택

Self Organizing Map(SOM)

- **Learning (SOM)**

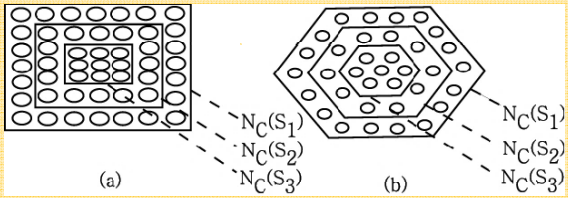
5. 뉴런 j*와 그 이웃 반경내의 뉴런들의 연결강도를 다음 식에 의해 재조정

$$W_{ij}(t+1) = W_{ij}(t) + \alpha(x_i(t) - W_{ji}(t))$$

→여기서 j는 j*의 이웃 반경내의 모든 뉴런

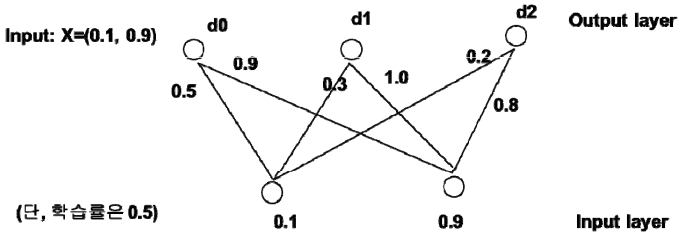
6. 2로 가서 모든 입력벡터를 처리

7. 지정된 학습회수까지 이웃 반경을 점차 감소시키면서 2부터 6의 과정을 충분한 횟수 반복



Self Organizing Map(SOM)

- **Learning (SOM, Example)**



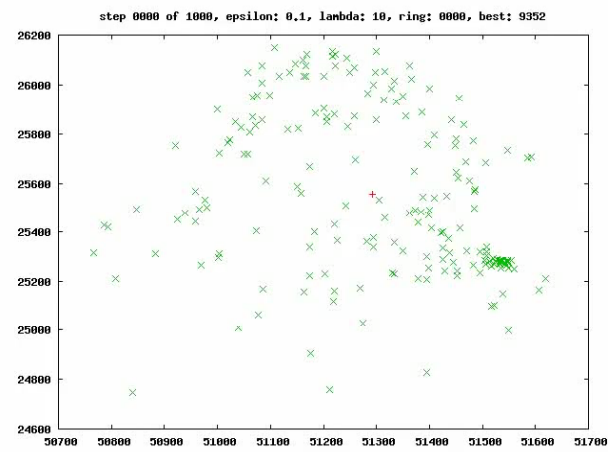
- d_0 = (0.1-0.5)^2 + (0.9-0.9)^2 = 0.16, d_1 = 0.05, d_2 = **0.02(승자뉴런)**
- d_2 의 연결 가중치 조정

$$W_{02}(t+1) = 0.2+0.5(0.1-0.2) = 0.15$$
$$W_{12}(t+1) = 0.8+0.5(0.9-0.8) = 0.85$$

→ 제시된 입력과 가장 유사한 출력뉴런의 가중치벡터가 입력을 향하여 이동
→ d_2 의 이웃 뉴런들 역시 동일하게 학습

Self Organizing Map(SOM)

- **Example of SOM(TSP problem)**



https://www.youtube.com/watch?v=8tnxgfE6gII

THANK YOU!