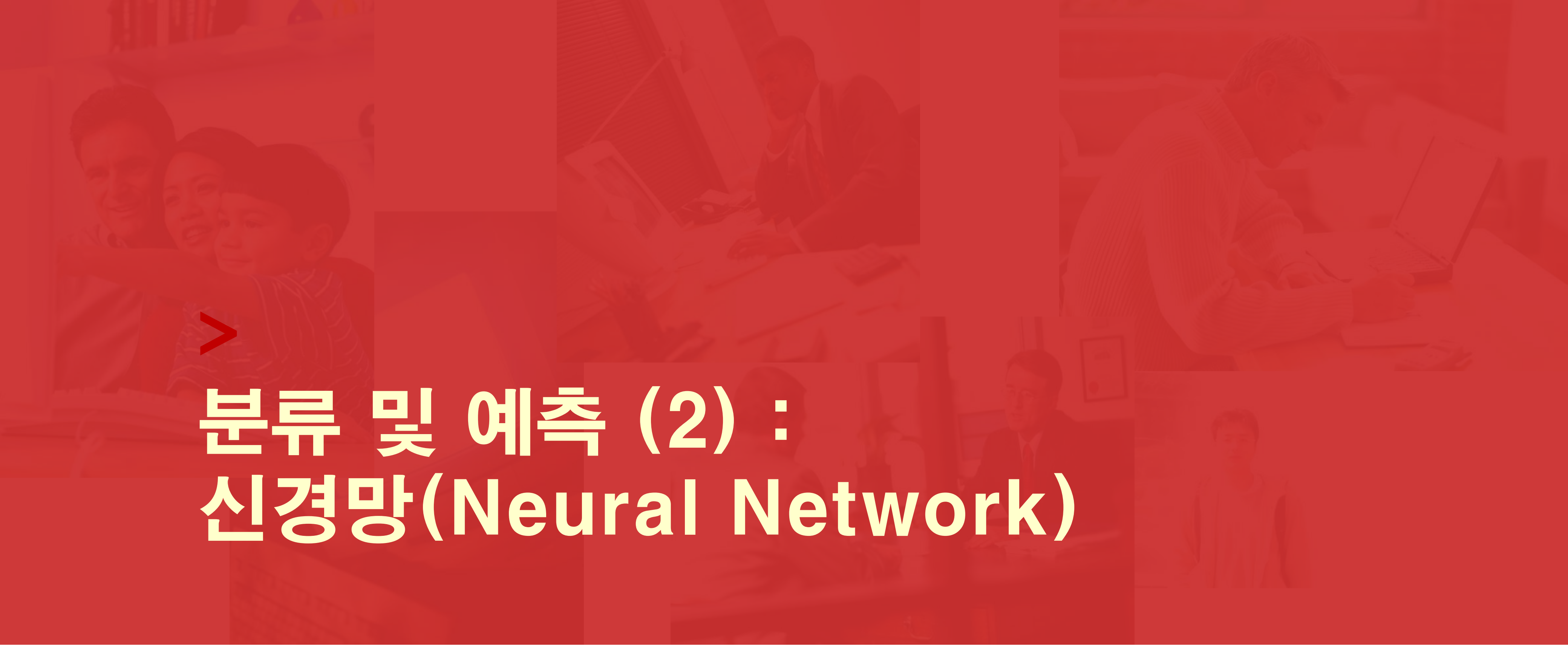


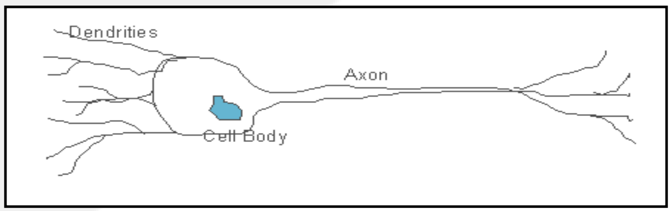
분류 및 예측 (2) : 신경망(Neural Network)

개념 및 알고리즘

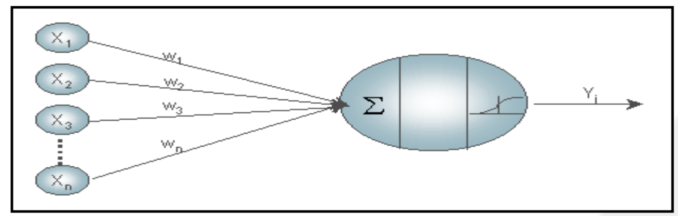
> 신경망(Neural Networks)의 개요

❖ 데이터 마이닝 알고리즘 중 가장 많이 알려진 것이 신경망 분석이며, 보통 데이터 마이닝에서 "신경망 분석 = 패턴을 찾아내는 것"이라고 연상할 만큼 잘 알려진 분석이다.

❖ 인간 두뇌의 신경망(100억 개의 뉴런과 100조 개의 시냅스로 구성)을 흉내 내어 실제 자신이 가진 데이터로부터의 반복적인 학습 과정을 거쳐 데이터에 숨어 있는 패턴을 찾아내는 모델링 기법



신경세포(neuron)



신경망(neural networks)

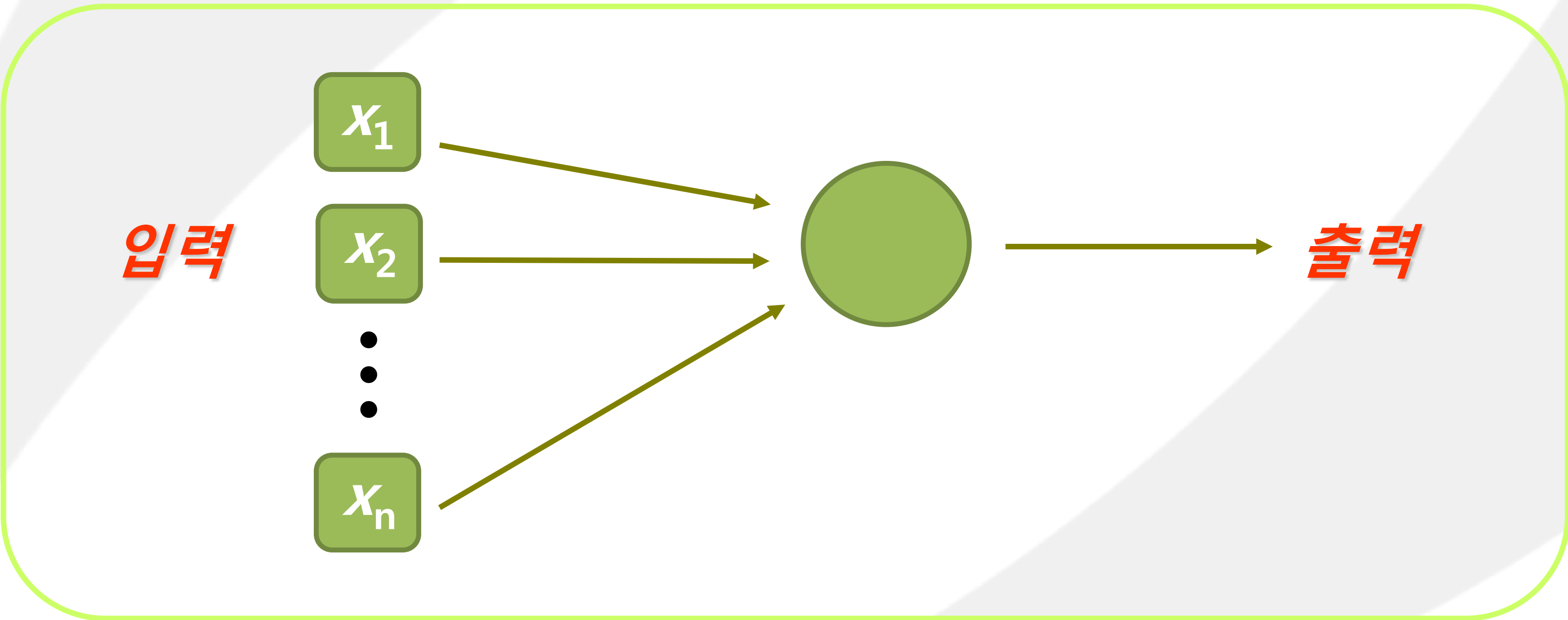
❖ 계층 구조를 갖는 수많은 프로세싱 요소로 이루어진 수학모형
  ✓ 신경망 이론의 다양한 아키텍처를 이용하여 예측모델 생성
  ✓ 자료의 패턴이 변화함에 따라 이를 학습하고, 이에 가중치를 변화 적용하여, 최적의 해를 구함

❖ 장단점
  ✓ 비선형 자료, 범주/연속형 혼합 자료 처리가 탁월하고 통계적 가정이 불필요
  ✓ 설명변수들이 목표변수에 구체적으로 어떠한 영향을 주는지 해석하기 어렵고, Over-Fitting 가능성 높음

> 신경망(Neural Networks)의 구성요소 (1/3)

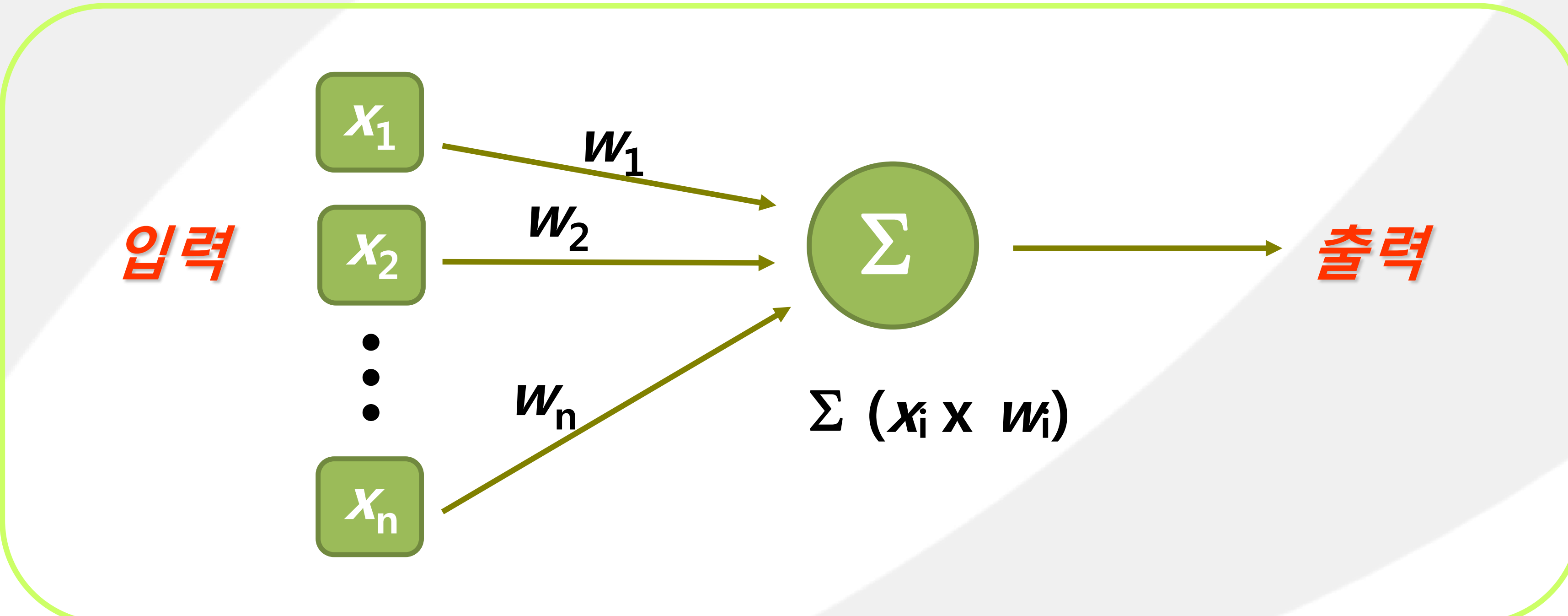
❖ 프로세싱 노드(processing unit, node)

- 입력신호를 측정
- 총 입력신호를 합산
- 출력신호를 결정



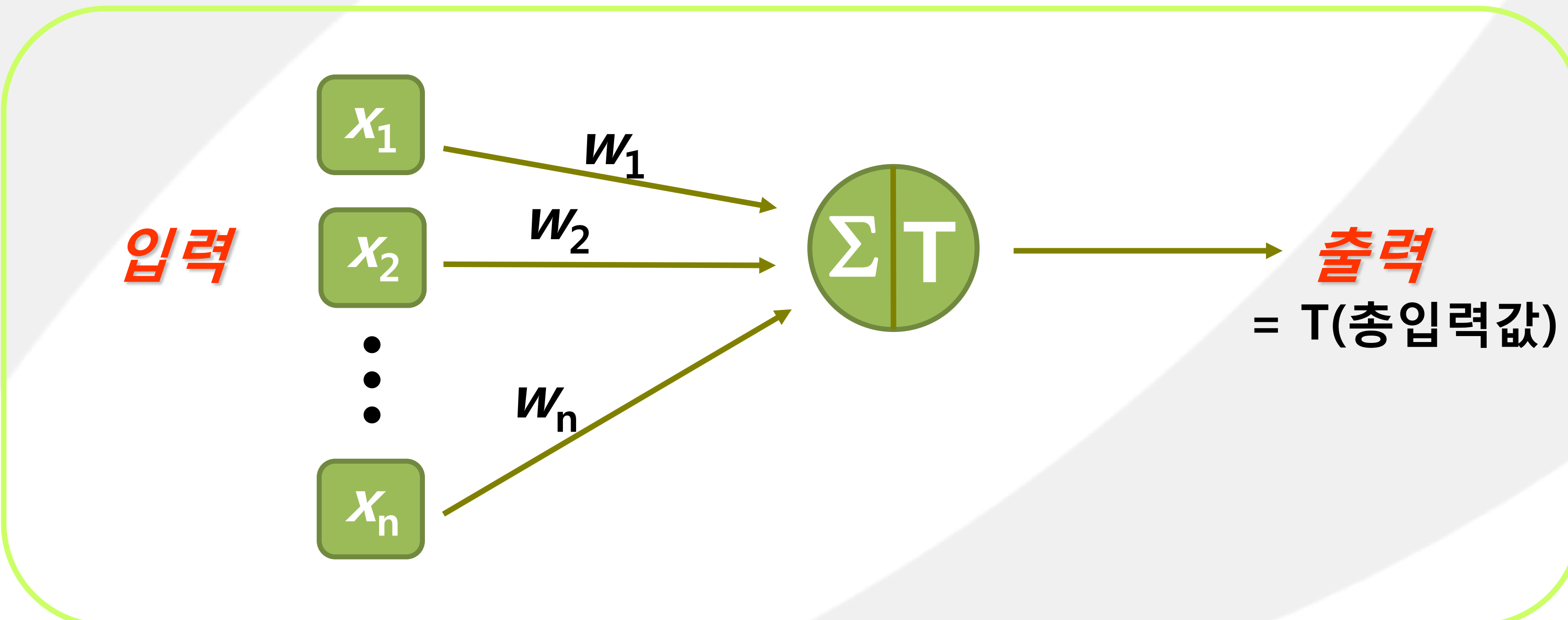
> 신경망(Neural Networks)의 구성요소 (2/3)

- 연결강도(weight)
  - 입력신호의 강도를 표현
- 총 입력값
  - 입력값의 선형결합함수
  - 총 입력값 = $x_1 \times w_1 + x_2 \times w_2 + \cdots + x_n \times w_n$



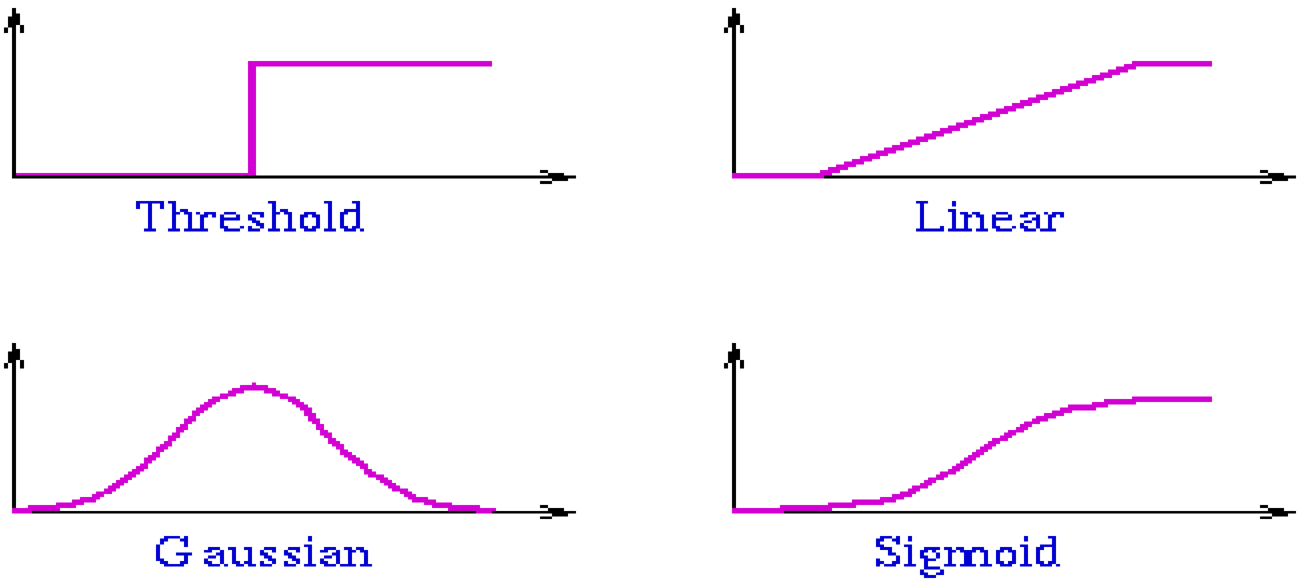
> 신경망(Neural Networks)의 구성요소 (3/3)

- 활성함수(activation/transfer function)
  - 입력정보의 합성값(결합값)을 일정 범위의 값으로 전환해주는 함수
  - 출력값을 결정
  - 비선형 함수를 사용



> Activation Functions

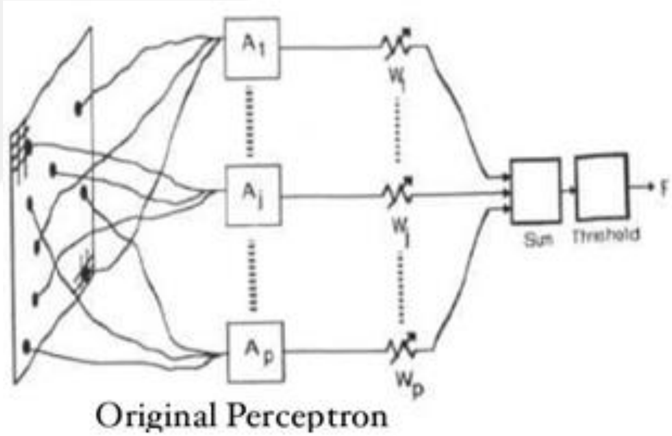
- 총입력값 (a = ∑ xi * wi)가 제한된 범위를 취하도록 하는 선형 또는 비선형 함수



logistic(x) = $1/(1+e^{-x})$

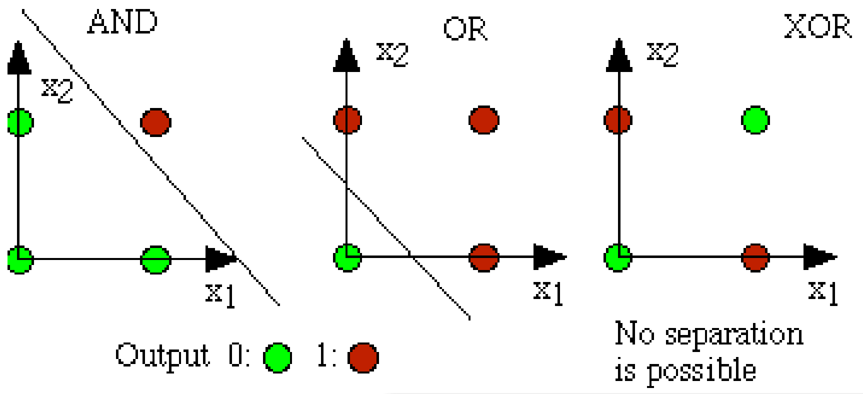
> 신경망의 배경 및 역사 (1/2)

- 1943년 McCulloch와 Pitts는 인간의 두뇌를 수많은 신경세포들로 구성된 컴퓨터라 생각하고, 최초로 신경망의 모델을 제안.
- 1951년에 Edmonds와 Minsky는 학습 기능을 갖는 최초의 신경망을 구축.
- 1957년에 Rosenblatt는 Perceptrons이라는 신경망모델을 제안하였는데,이것은 패턴을 인식하기 위하여 학습 기능을 이용.



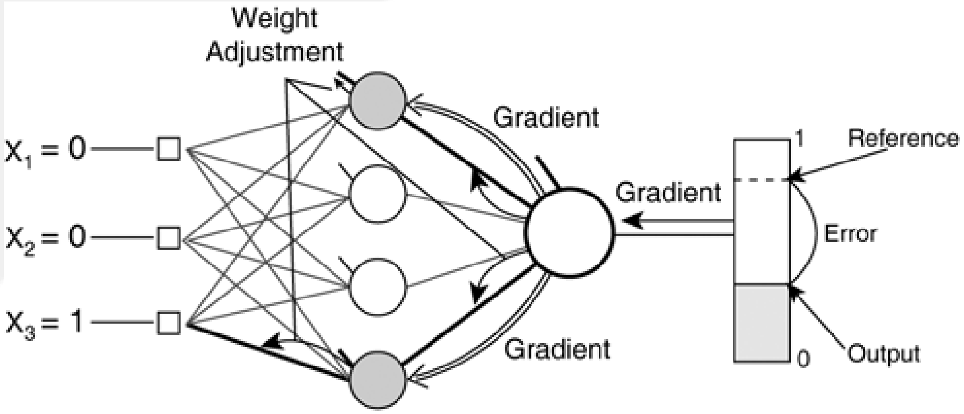
Original Perceptron

- 1969년 Minsky와 Papert가 그들의 저서 "Perceptrons"에서 퍼셉트론이 비선형 분리 문제를 풀 수 없음을 밝혀, 침체기에 들어감.

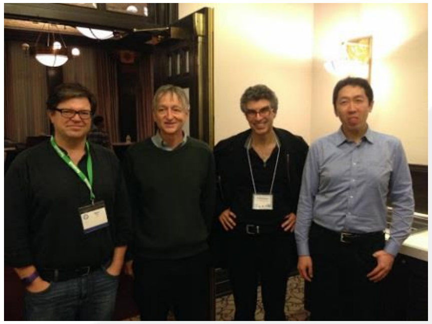


> 신경망의 배경 및 역사 (2/2)

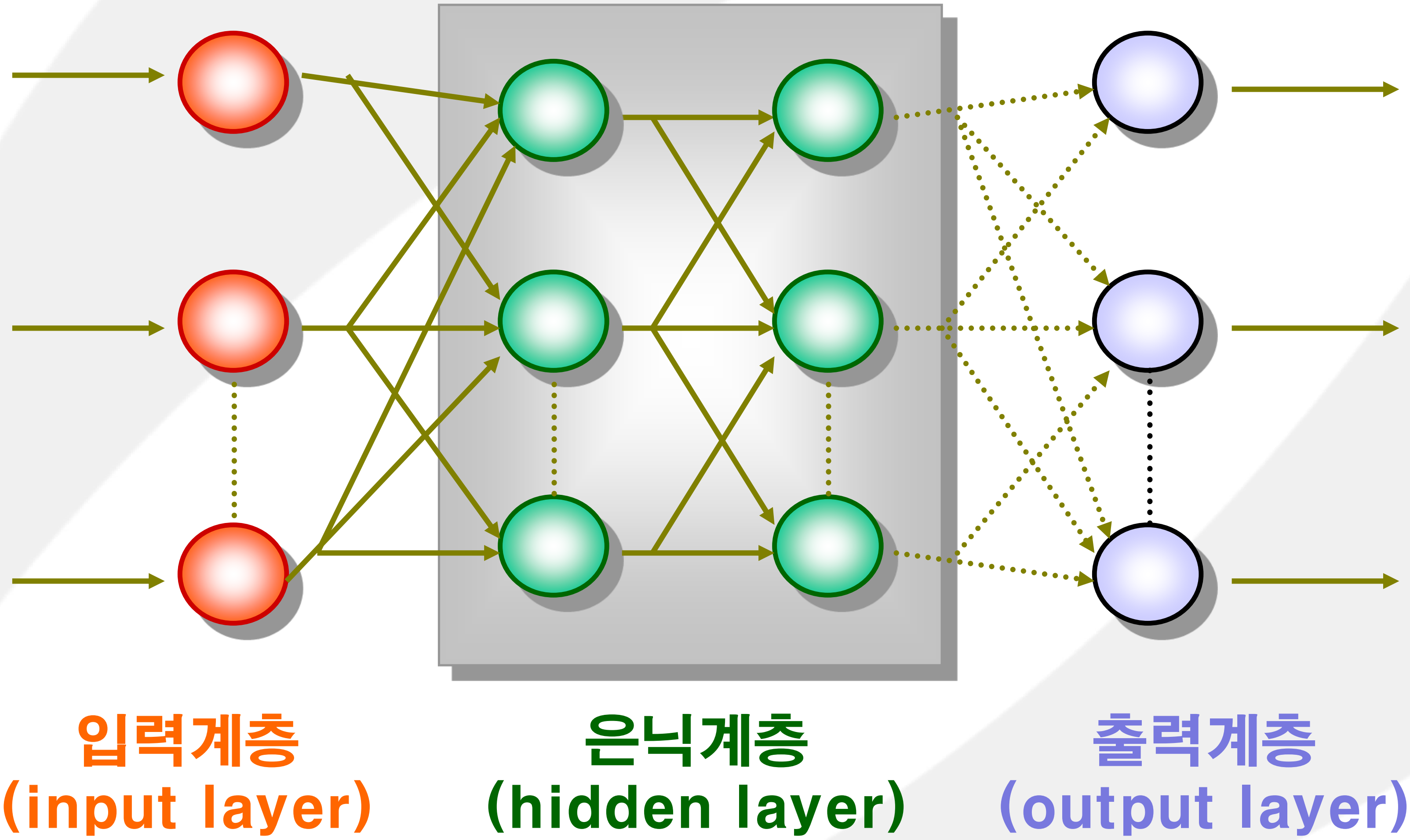
- 1980년대 초반, Hopfield의 Hopfield 네트워크, Kohonen 네트워크 등이 발표.
- 1980년대 중반 PDP그룹에 의해 Back-Propagation 알고리즘을 사용하는 MLP가 탄생되어, 신경망의 다양한 분야에 대한 연구와 응용이 이루어짐.



- Back-Propagation 알고리즘은 Vanishing gradient 문제로 인하여 은닉층이 다수인 Deep Neural Network을 학습하지 못함. 이에 따라 성능의 한계에 봉착하여, 또다시 침체기에 들어감.
- 2006년, Geoffrey Hinton에 의해 RBM 기반의 pre-training으로 Deep Neural Network의 학습이 가능해지면서 Deep Learning이라는 새로운 이름으로 다시 주목을 받기 시작함.



> 신경망의 계층구조



> MLP (Multi-Layer Perceptron)

❖ 다층 퍼셉트론 (Multi-Layer Perceptron, MLP) 구조

- 입력층 뉴런(입력변수)로부터 전달되는 신호들을 모아 선형결합
- $X_1,\cdots,X_p$ 를 설명변수(입력 노드)라고 할 때 다음 뉴런에

$$L = w_1 X_1 + \cdots + w_p X_p$$

이 전달된다. 여기서 $w_1,\cdots,w_p$는 신경선(synapse)에 붙는 가중값(weight)

- 뉴런의 활성화

  - 로지스틱(logistic) : $S = 1/(1+e^{-L}), \quad 0 \le S \le 1$

  - 쌍곡 탄젠트(hyperbolic tangent) : $S = (e^L - e^{-L})/(e^L + e^{-L}), \quad -1 \le S \le 1$

- 출력노드

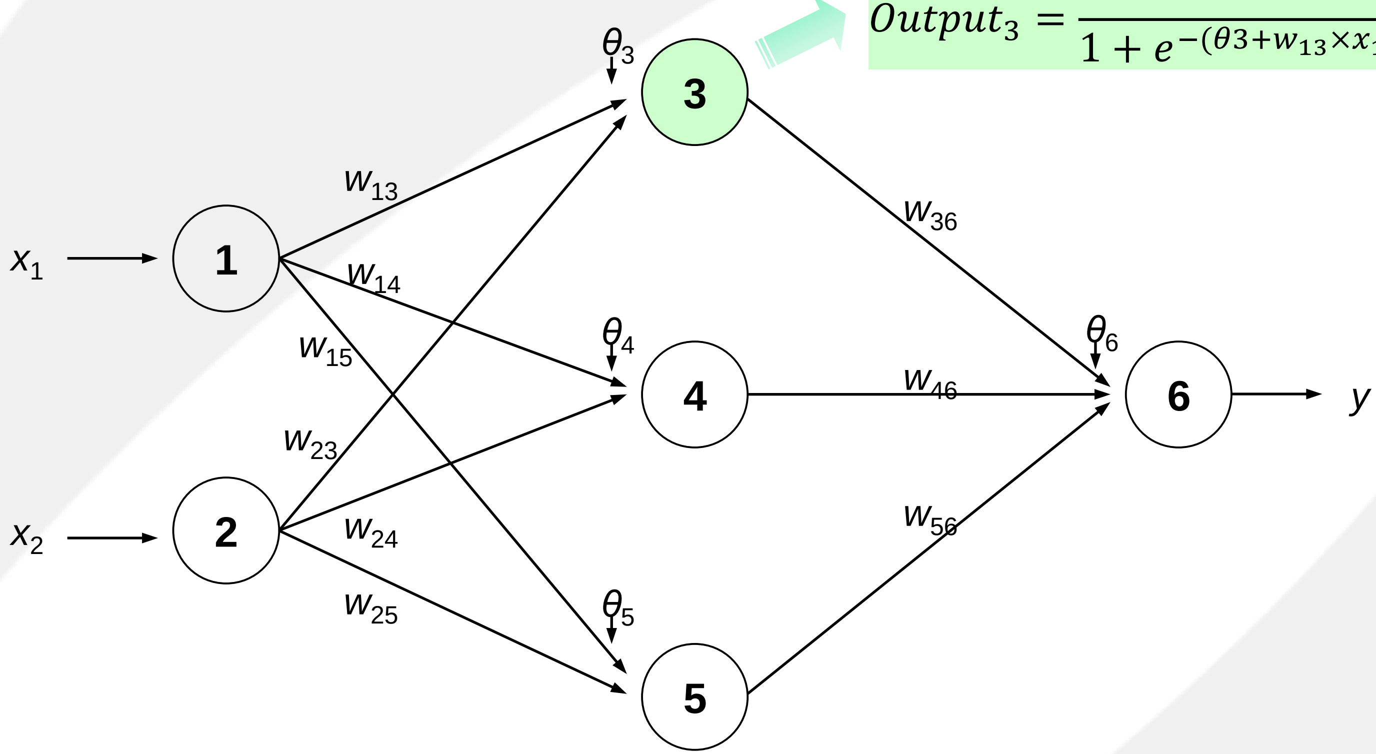
  - 연속형 : $O = L$

  - 범주형 : 소프트맥스(softmax)

$$O_k = \frac{\exp(L_k)}{\sum_{j=1}^{K} \exp(L_j)}$$ 여기서 k 는 범주.

> ANN Example (1/2)

$$Output_3 = \frac{1}{1 + e^{-(\theta 3 + w_{13} \times x_1 + w_{23} \times x_2)}}$$

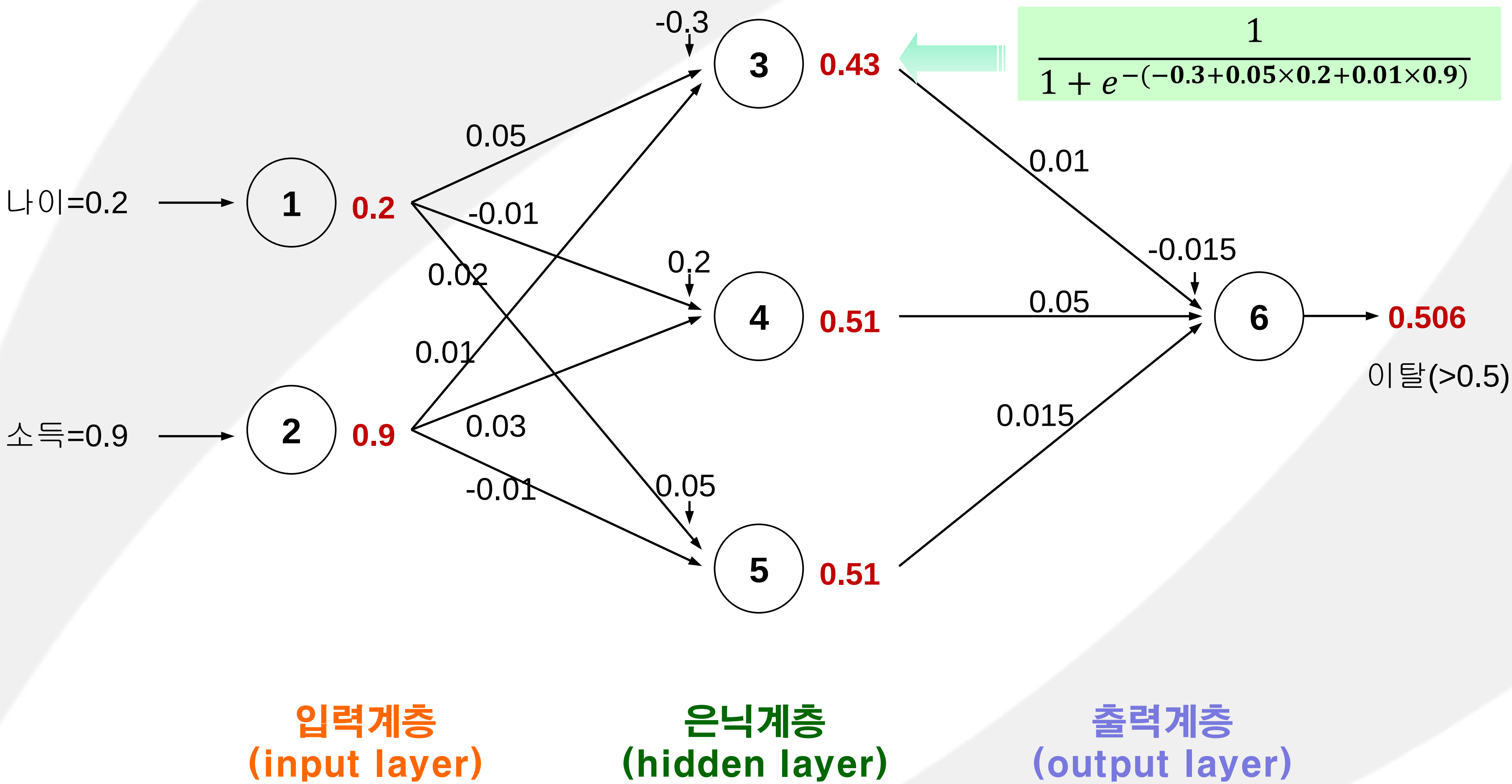


입력계층
(input layer)

은닉계층
(hidden layer)

출력계층
(output layer)

> ANN Example (2/2)



> Learning method: Back propagation (1/7)

- The error function

$$\mathrm{E}(\mathrm{W}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}(\mathbf{X},\mathbf{W}) - \mathbf{t}_n\|^2$$

- Learning NN – adjusting weights to minimize error (E)

- Iterative numerical procedure

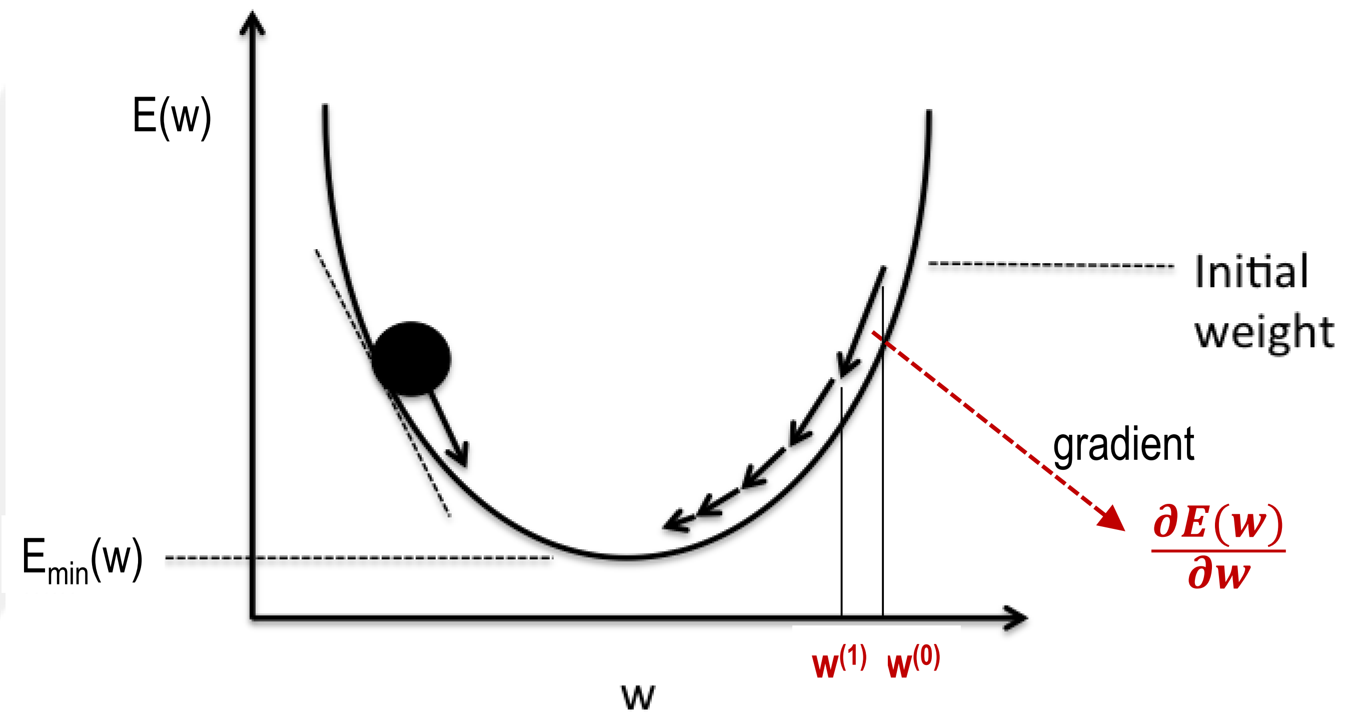
$$\mathrm{W}^{(\tau+1)} = \mathrm{W}^{(\tau)} + \Delta\mathrm{W}^{(\tau)}$$

- Gradient descent optimization

$$\mathrm{W}^{(\tau+1)} = \mathrm{W}^{(\tau)} - \eta\nabla\mathrm{E}\left(\mathrm{W}^{(\tau)}\right)$$

$\triangledown$E: changes in W to change in E
η: learning rate (0 ~ 1)

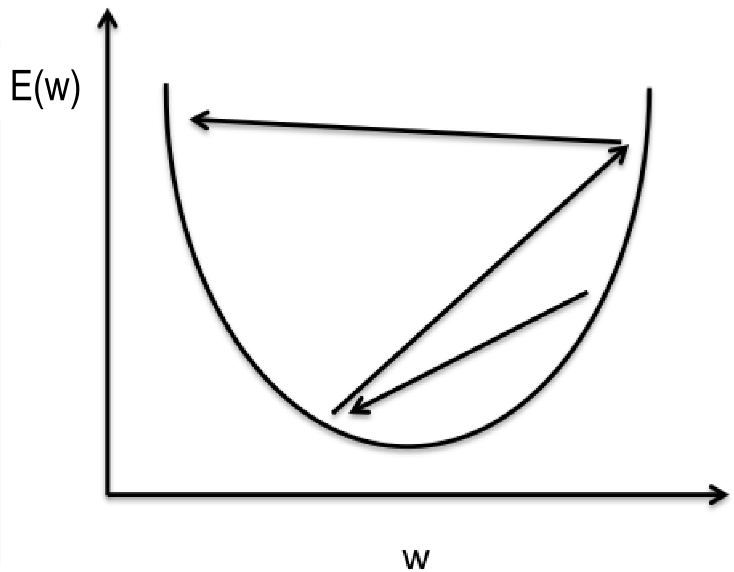
- Schematics of gradient descent



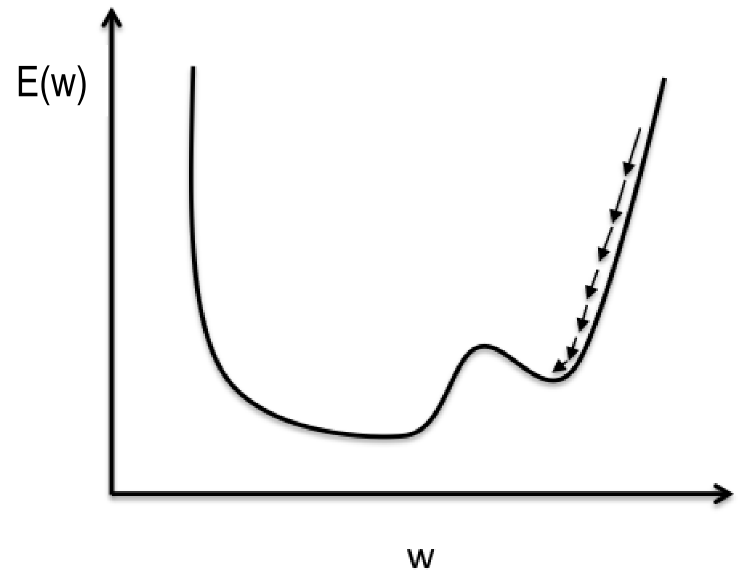
$$w^{(1)} = w^{(0)} - \eta \frac{\partial E(w)}{\partial w}$$

> Learning method: Back propagation (3/7)

- Learning rate(η)



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

> Learning method: Back propagation (4/7)

$$x \xrightarrow[g()]{w_1} h \xrightarrow[g()]{w_2} y$$

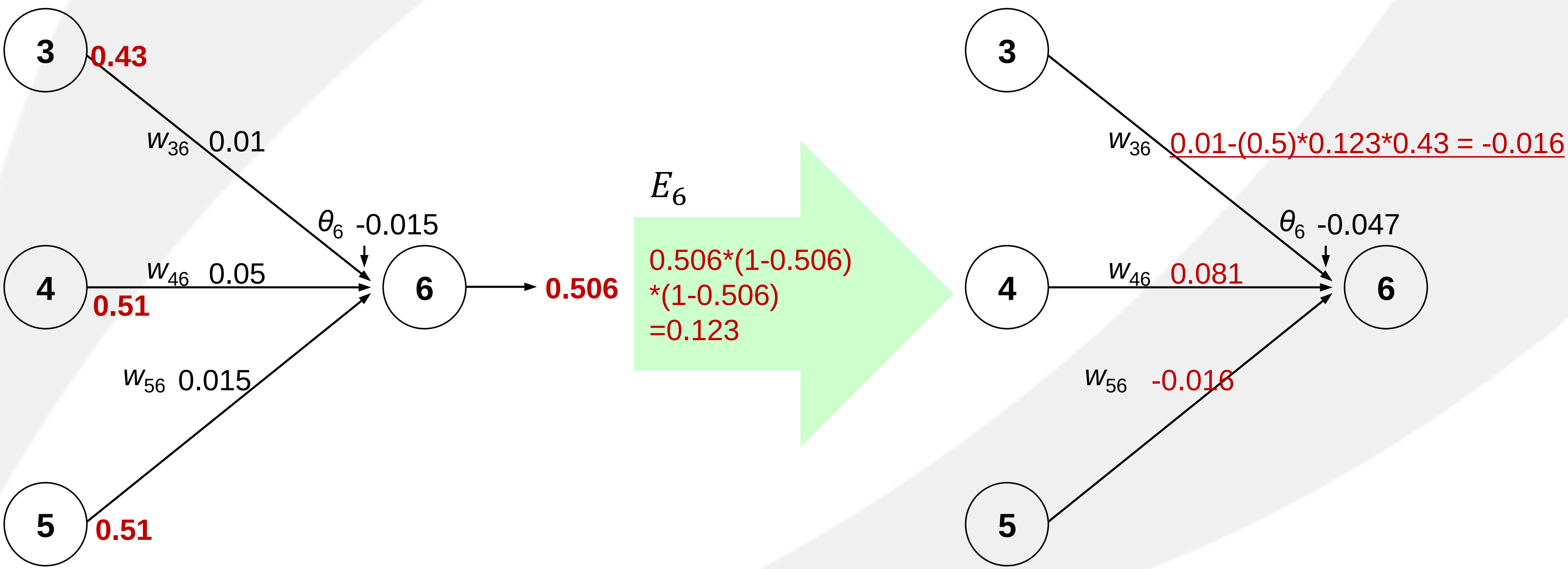
$$E(\mathbf{w}) = \frac{1}{2}(\mathrm{y} - y_i)^2, where\ y_i\ is\ a\ real\ value.$$

$$\begin{aligned}
\frac{\partial E(w)}{\partial w_2} &= (\mathrm{y} - y_i) \cdot \frac{\partial y}{\partial w_2} \\
&= (\mathrm{y} - y_i) \cdot \frac{\partial g(h \cdot w_2)}{\partial w_2} \\
&= (\mathrm{y} - y_i) \cdot g(h \cdot w_2) \cdot (1 - g(h \cdot w_2)) \times \frac{\partial (h \cdot w_2)}{\partial w_2} \\
&= (\mathbf{y} - \boldsymbol{y_i}) \cdot \boldsymbol{y} \cdot (\mathbf{1} - \boldsymbol{y}) \cdot \boldsymbol{h} = \boldsymbol{E_y} \cdot \boldsymbol{h}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E(w)}{\partial w_1} &= (\mathrm{y} - y_i) \cdot \frac{\partial y}{\partial w_1} \\
&= (\mathrm{y} - y_i) \cdot y \cdot (1 - y) \cdot \frac{\partial (h \cdot w_2)}{\partial w_1} \\
&= (\mathrm{y} - y_i) \cdot y \cdot (1 - y) \cdot w_2 \cdot \frac{\partial h}{\partial w_1} \\
&= (\mathrm{y} - y_i) \cdot y \cdot (1 - y) \cdot w_2 \cdot h \cdot (1 - h) \cdot \frac{\partial (x \cdot w_1)}{\partial w_1} \\
&= (\mathbf{y} - \boldsymbol{y_i}) \cdot \boldsymbol{y} \cdot (\mathbf{1} - \boldsymbol{y}) \cdot \boldsymbol{w_2} \cdot \boldsymbol{h} \cdot (\mathbf{1} - \boldsymbol{h}) \cdot \boldsymbol{x} = \boldsymbol{E_y} \cdot \boldsymbol{w_2} \cdot \boldsymbol{E_h} \cdot \boldsymbol{x}
\end{aligned}$$

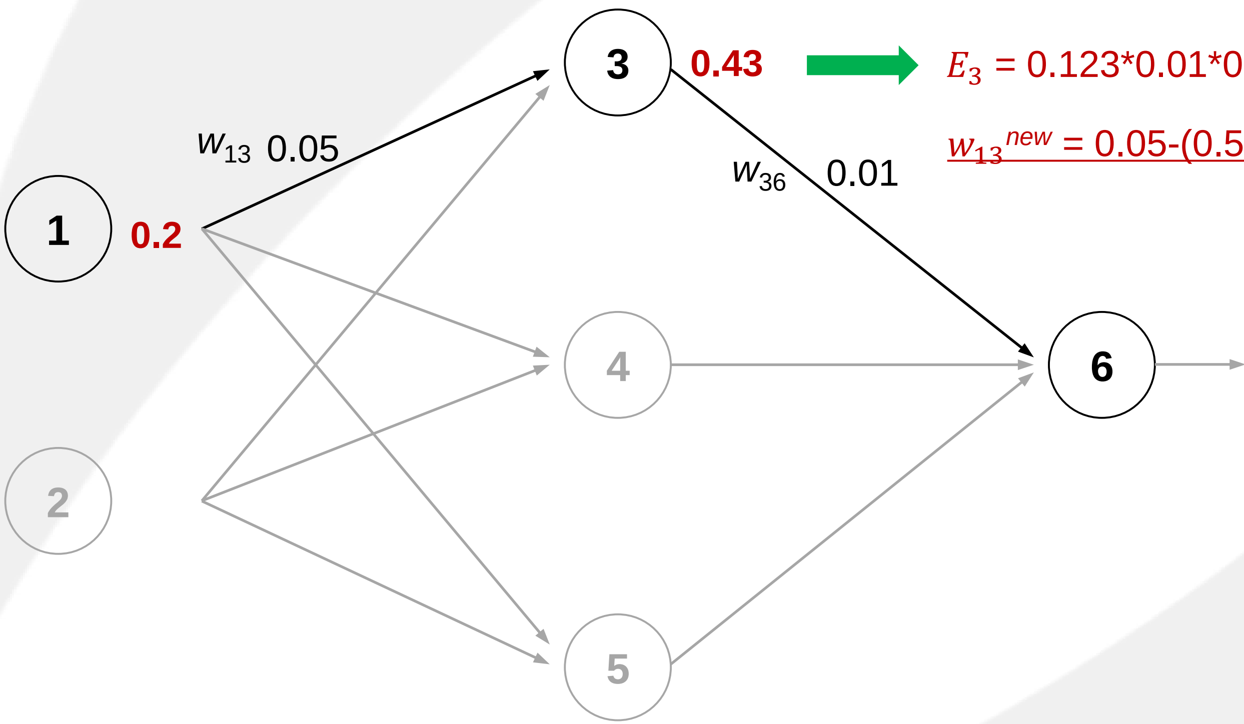
> Learning method: Back propagation (5/7)

$$w_{36}{}^{new} \leftarrow w_{36}{}^{old} - \eta \cdot E_6 \cdot O_3$$
$$where\ E_6 = \underbrace{\boxed{O_6(1-O_6)}}_{\text{오차변화율}} \cdot \underbrace{\boxed{(y-O_6)}}_{\text{추정오차}}$$



> Learning method: Back propagation (6/7)

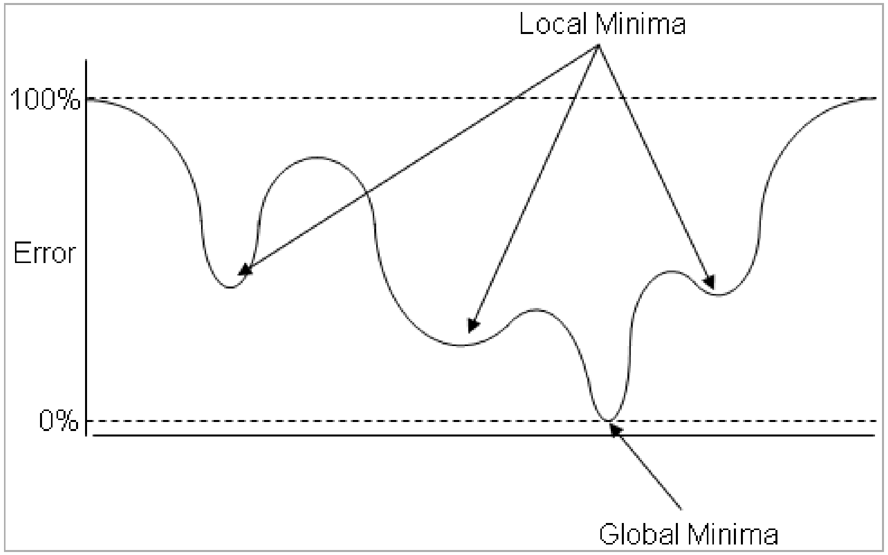
$$w_{13}^{new} \leftarrow w_{13}^{old} - \eta \cdot E_3 \cdot O_1$$
$$where\ E_3 = E_6 \cdot w_{36} \cdot O_3(1 - O_3)$$

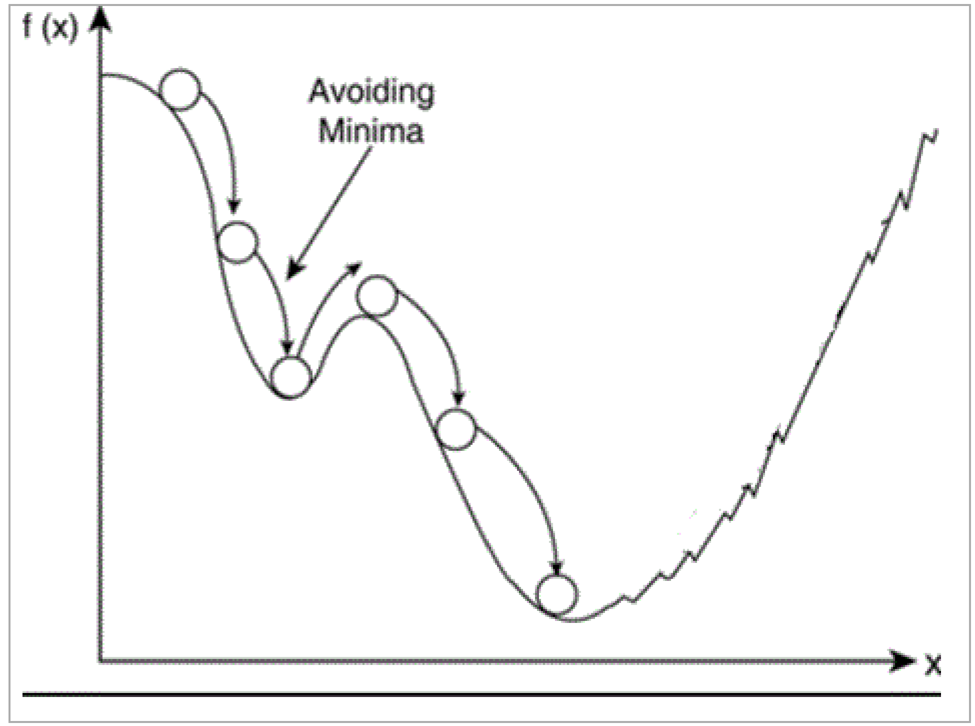


E_3 = 0.123*0.01*0.43(1-0.43) = 0.0003

w_{13}^{new} = 0.05-(0.5)*0.0003*0.2 = 0.05

- Momentum

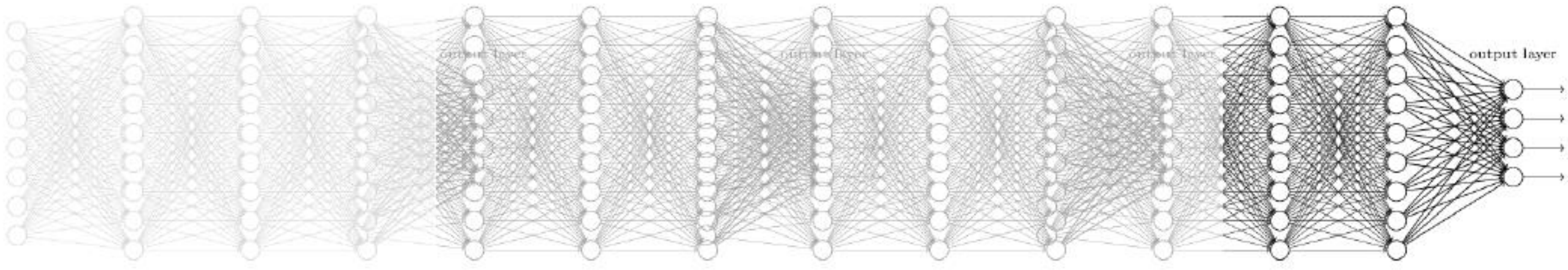




기존 업데이트에 사용했던 기울기의 일정 비율을 남겨서 현재의 기울기와 더하여 업데이트함

> Deep Learning (1/4)

Vanishing gradient (NN winter2: 1986-2006)

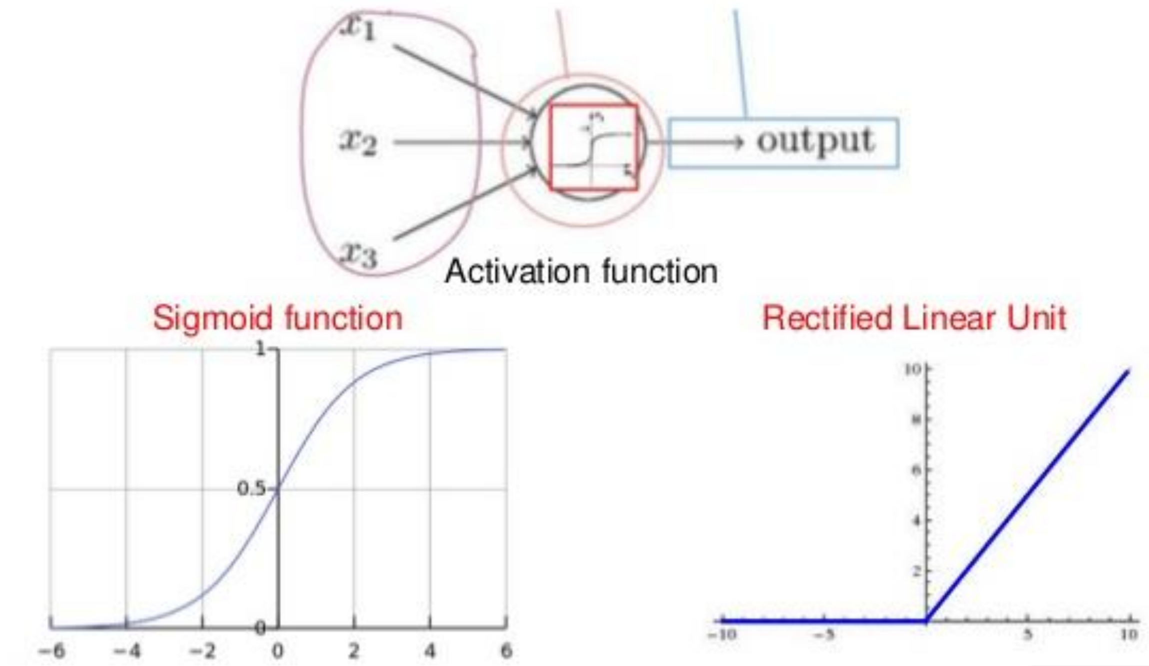


Geoffrey Hinton's findings:

- Too small data
- Too slow computer
- Initializing the weights in a stupid way
- Using wrong type of non-linearity

Source: http://hunkim.github.io/ml/

Rectified Linear Unit (ReLU)

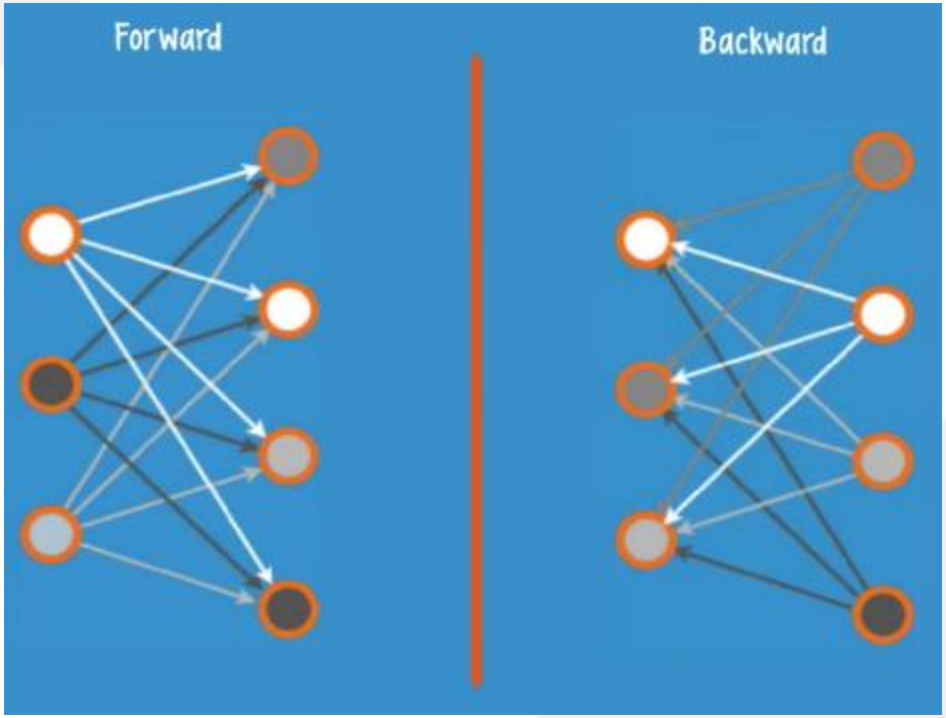


$$f(x) = max(0, x)$$

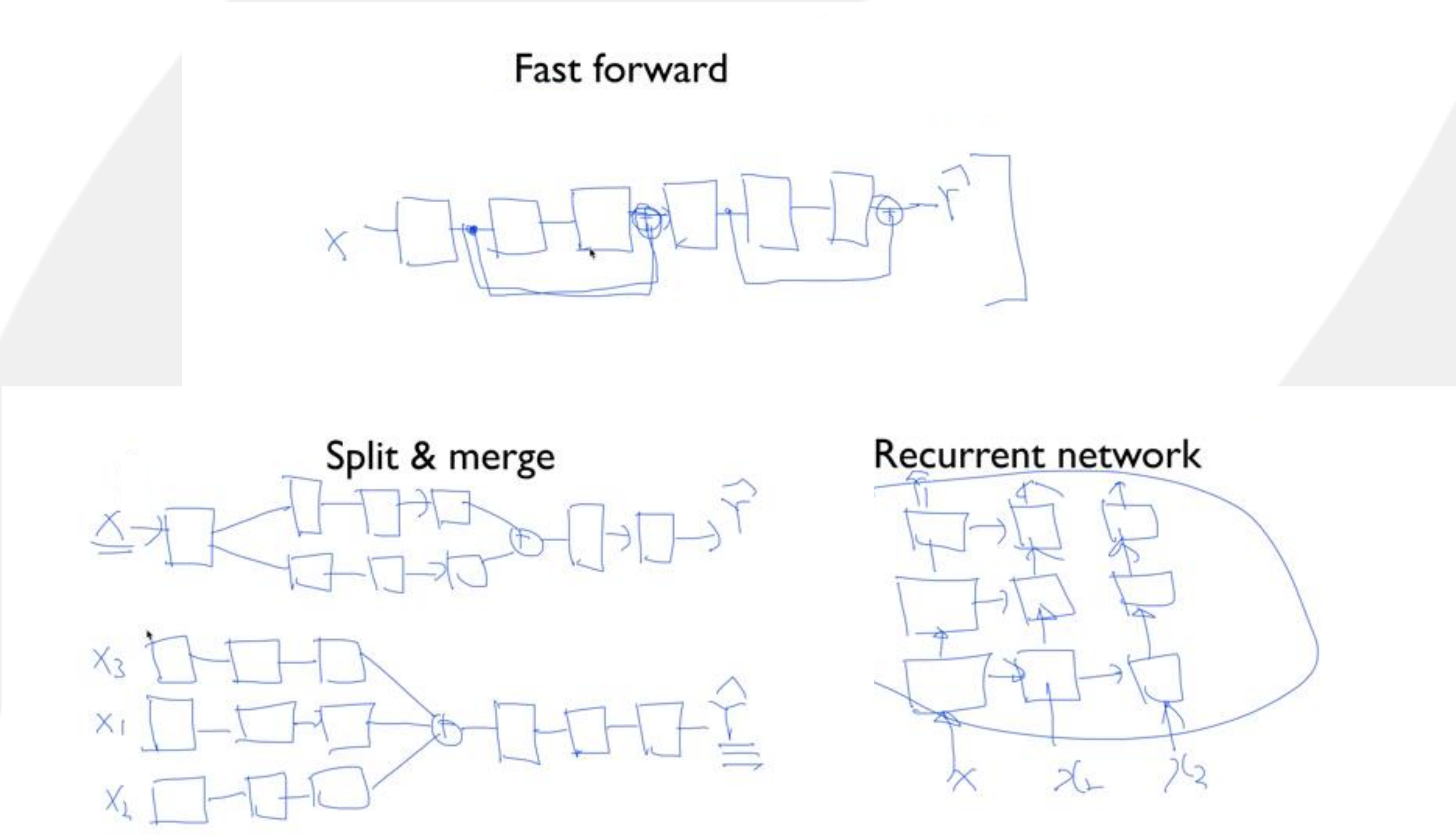
Source: http://hunkim.github.io/ml/

Need to set the initial weight values wisely

- Not all 0's
- Hinton et al. (2006) "A Fast Learning Algorithm for Deep Belief Nets"
  - Restricted Boatman Machine (RBM)



Source: http://hunkim.github.io/ml/



Source: http://hunkim.github.io/ml/

> 신경망을 사용할 때의 휴리스틱

❖ 은닉층 노드의 수는 얼마가 적당한가?

- ✓ 실제 해답은 아무도 모름
- ✓ 데이터, 감지되는 패턴들, 신경망의 유형에 따라 달라짐
- ✓ 독립변수와 종속변수 개수의 합을 n이라 할 때 n/2, n, 3n/2, 2n의 총 4가지 경우를 많이 사용

❖ 훈련 집합의 크기

- ✓ 각각의 특징에 대하여 가능한 입력의 범위를 포함할 수 있는 정도로 커야 함
- ✓ s개의 입력단위, h개의 은닉 단위, 하나의 출력을 가지면, 신경망에는 h*(s+1)+h+1개의 가중치 존재
- ✓ 만일 15개 입력 특징, 10개의 은닉 단위가 존재하면,
  - ✓ 171개의 가중치 존재
  - ✓ 각각의 가중치에 대하여 최소한 30개의 예시들은 있어야 하지만, 좀 더 나은 최소값으로는 100개가 필요. 이 예시에서는 최소한 17,100개 사례가 필요함.

❖ 학습률과 모멘텀

❖ 학습반복 횟수