7. Machine Learning

머신 러닝

- 머신 러닝
  - 기계 번역, 영상 인식과 같은 시스템을 사람이 공급하는 데이터로 학습시켜 원하는 기능을 수행하게 하는 것
  - 학습 데이터에 클래스 정보를 추가하는지에 따라 지도(supervised) 학습과 비지도(unsupervised) 학습으로 구분함
- 자연어 처리와 머신 러닝
  - 기계 번역, 음성 인식, 문서 분류, 문서 작성 등 다양한 분야에서 머신 러닝 시스템의 성능이 급속히 향상되고 있음
  - 자연어 처리 시스템의 성능은 기계 학습 시스템의 구조에 의해 향상되고 있다고 할 수 있음

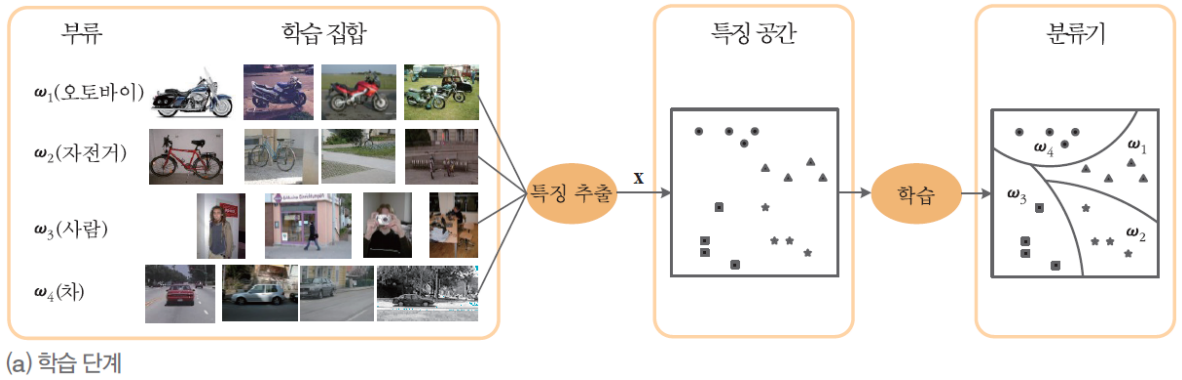
자연어 처리 참고자료

- Stanford CS224d
  - CS224n의 2017년 version 인데, 좀더 기본적인 내용들을 다루고 있음
  - CS224d: Deep Learning for Natural Language Processing
  - https://cs224d.stanford.edu

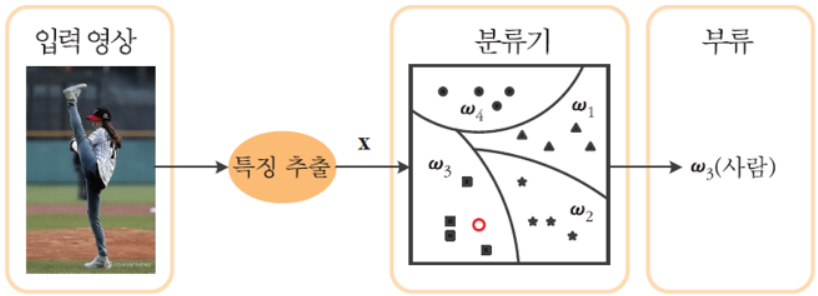
- 조경현 교수 자연어처리 강의
  - New York University의 교수로 재직중이며 자연어처리에 관한 연구 성과가 높음
  - 딥 러닝을 이용한 자연어 처리 (동영상 강의)
  - https://www.boostcourse.org/ai331

지도 학습과 비지도 학습

- **지도 학습 (Supervised learning)**
  - 클래스 정보를 가진 샘플 $(\mathbf{x}, t)$로 구성된 학습 집합 사용
    - $\mathbf{x}$는 입력신호이고 t는 $\mathbf{x}$가 속한 클래스



(a) 학습 단계



(b) 테스트(인식) 단계

비지도 학습

- 지도 학습에서 사용했던 ($\mathbf{x}$,t) 중에 클래스 정보 t가 없는 상황의 학습

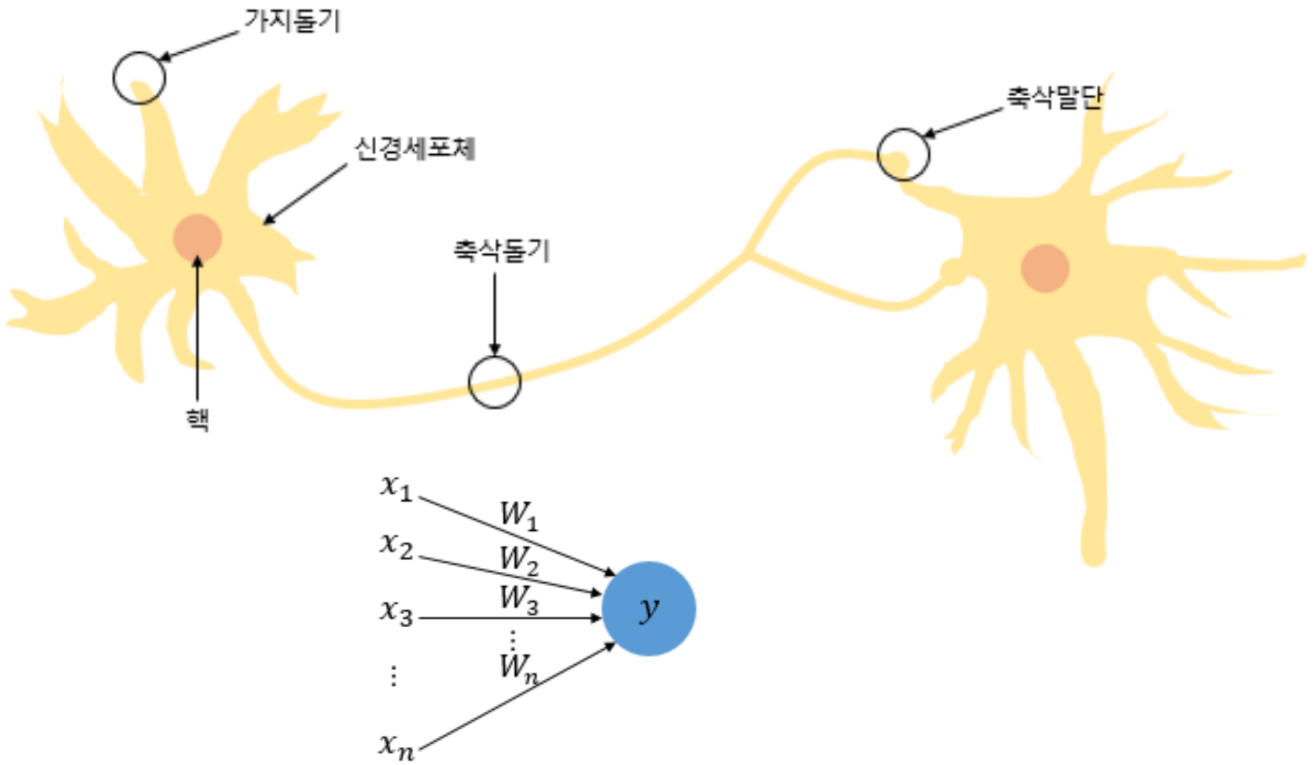
- **비지도 학습(Unsupervised learning)**
  - 유사한 특징 벡터들을 끼리끼리 모으는 군집화(clustering) 수행 (K-means, SOM 신경망, 민시프트 등의 군집화 알고리즘)
  - 군집에서 유용한 정보 추출 (데이터 마이닝, 빅데이터, 정보 검색 등의 많은 응용)

신경망(Neural network)

- 신경망은 연결주의(connectionist) 계산 모형
  - 방대하게 연결된 많은 뉴런으로 구성된 뇌 구조를 모방한 계산 모형
  - 1950년대 Rosenblatt의 퍼셉트론
  - 1980년대 퍼셉트론을 확장한 다층 퍼셉트론(MLP)
  - 일반화 능력이 뛰어남

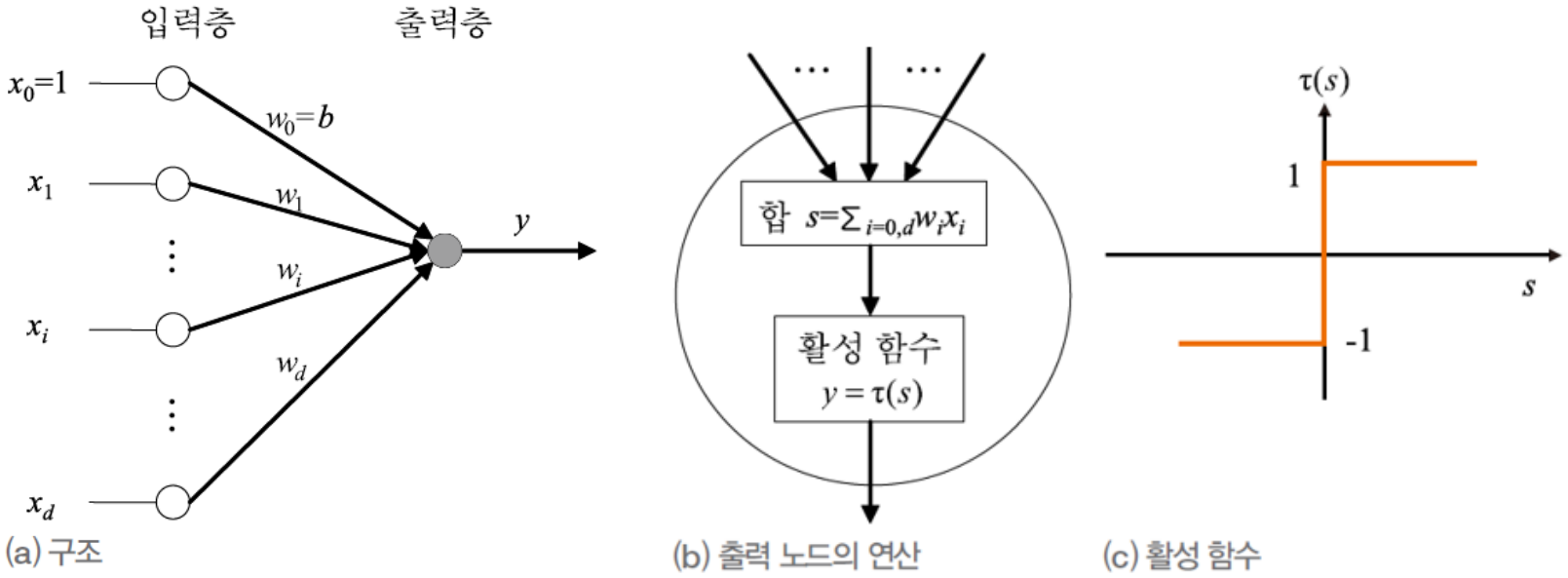
생체에서의 신경망

- 각각의 신경세포(neuron)가 독립적인 CPU와 유사하게 동작
- 신경세포간의 정보 전달이 축삭돌기(axon)를 통해 이루어짐
- 사람의 뇌에는 뉴런이 1,000억개 가량 있는 것으로 추정됨



퍼셉트론 구조와 작동 원리

- Perceptron: 1960년대에 제안되었던 신경망 모델
  - 입력은 특징 벡터 $\mathbf{x}=(x_1,x_2,\ldots,x_d)$
  - $\mathbf{x}$를 두 개의 부류 ω_1과 ω_2 중의 하나로 분류하는 이진 분류기



퍼셉트론 작동 원리

- 수식으로 쓰면,
  - 출력 노드는 가중치 합과 활성 함수 계산
    - 특징 벡터 $\mathbf{x}=(x_1,x_2,...,x_d)$, 가중치 벡터 $\mathbf{w}=(w_1,w_2,...,w_d)$로 표기하면

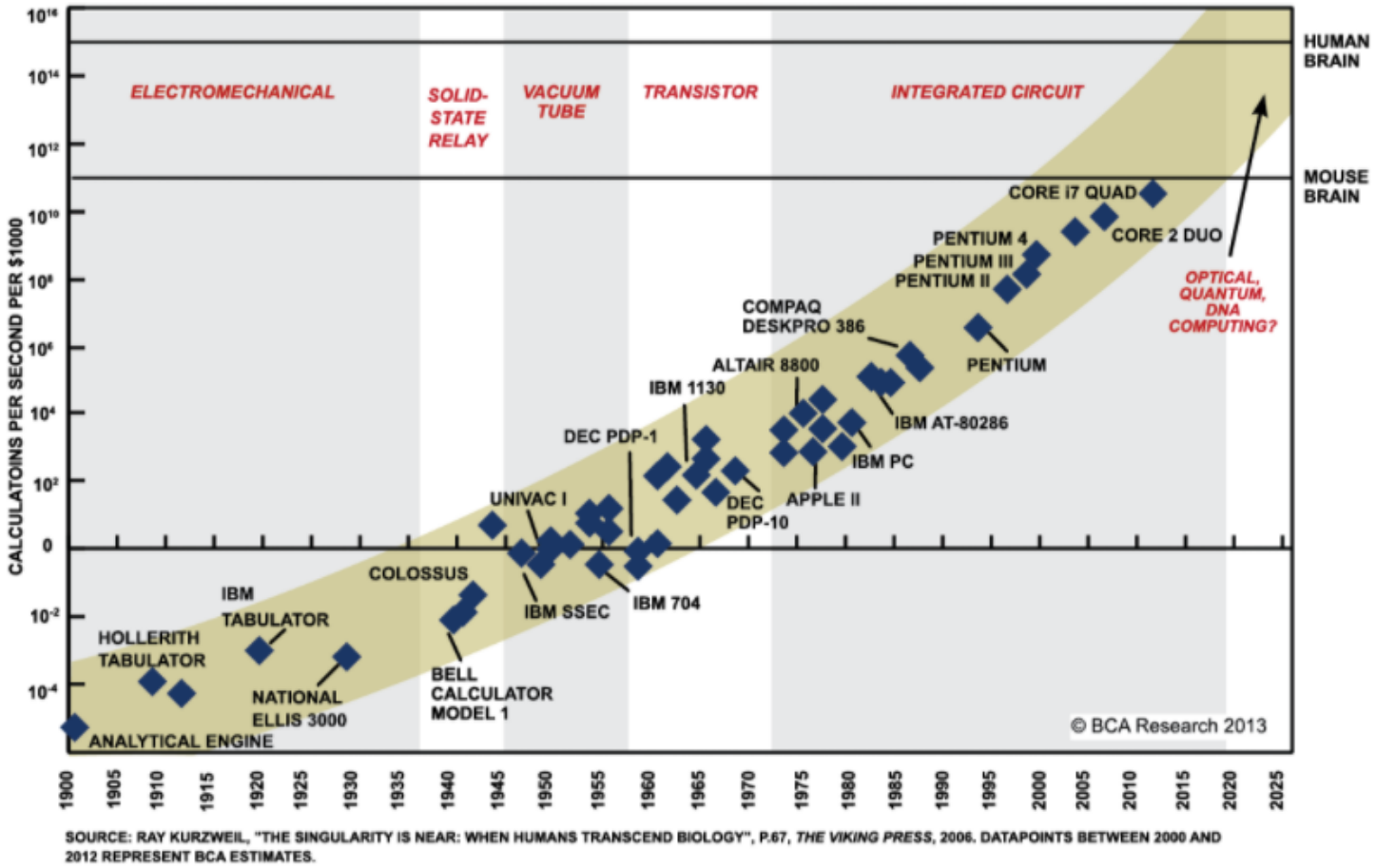
$$y = \tau(s) = \tau\left(\sum_{i=1}^{d} w_i x_i + b\right) = \tau(\mathbf{w}\mathbf{x}^{\mathrm{T}} + b)$$

$$\text{이때 } \tau(s) = \begin{cases} +1, & s \geq 0 \\ -1, & s < 0 \end{cases}$$

  - 계단 함수를 활성 함수로 사용: **이 점이 현재의 신경망 구조와 다름**
    - 출력은 +1(ω_1에 해당) 또는 -1(ω_2에 해당) → 이진 분류기
- 1단의 퍼셉트론으로 처리할 수 있는 작업은 매우 제한적이어서 1980년 무렵까지는 별 진전이 없었음

컴퓨터 속도의 발전

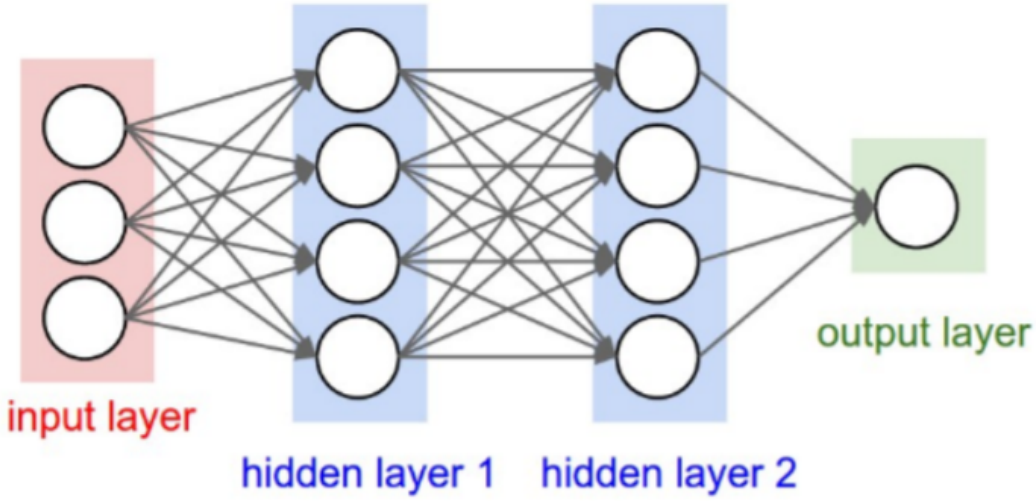
Moore's Law



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

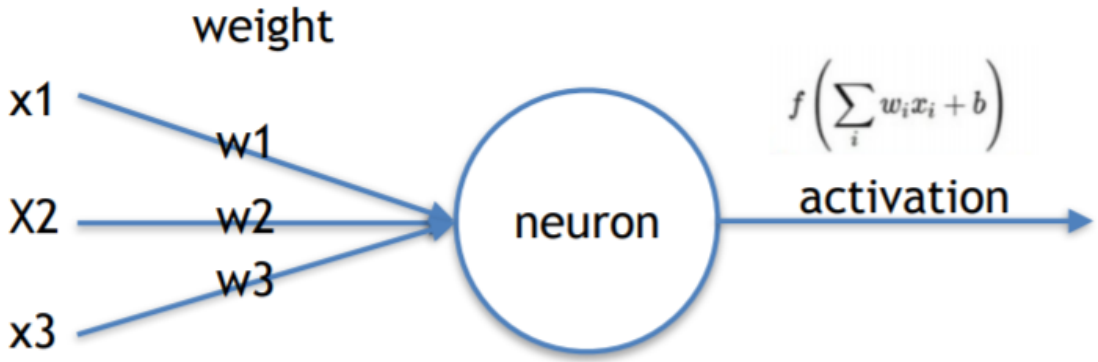
신경망(Neural network)

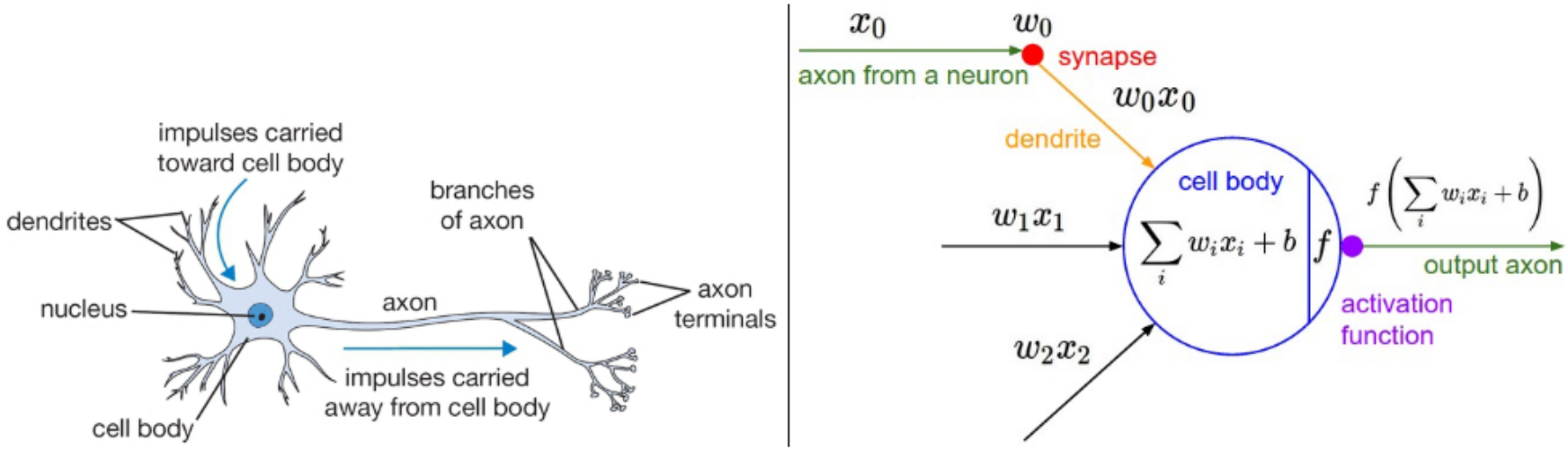
- 일반적으로 입력층(input layer), 은닉(hidden)층, 출력(output)층으로 구성됨
- 은닉층: 결과가 바로 관찰되지는 않지만 중간 계산 과정에 포함되는 연결 부분
- 각 층은 일정한 개수의 neuron(신경세포)으로 구성됨



뉴런

- 뉴런은 퍼셉트론과 유사하지만 다른 형태의 활성화함수를 사용

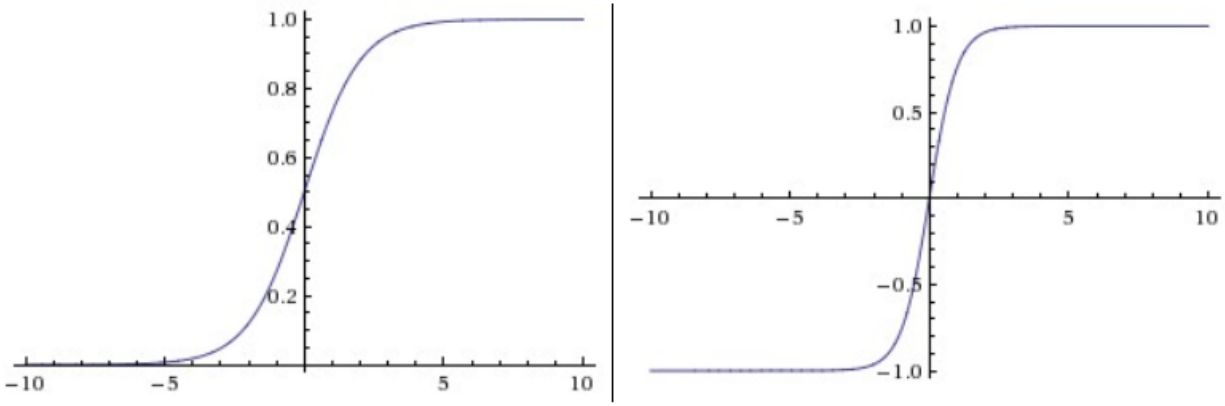




A cartoon drawing of a biological neuron (left) and its mathematical model (right).

자주 사용되는 활성화 함수(1)

- 활성화 함수가 없으면 뉴런은 선형 연산만 하게 됨
- 활성화 함수는 비선형(nonlinear) 함수이며, 여러 개의 뉴런을 동시에 사용하면 다양한 함수 형태를 근사적으로 표현할 수 있음



Left: Sigmoid non-linearity squashes real numbers to range between [0,1] **Right:** The tanh non-linearity squashes real numbers to range between [-1,1].

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\tanh(x) = 2\sigma(2x) - 1$$

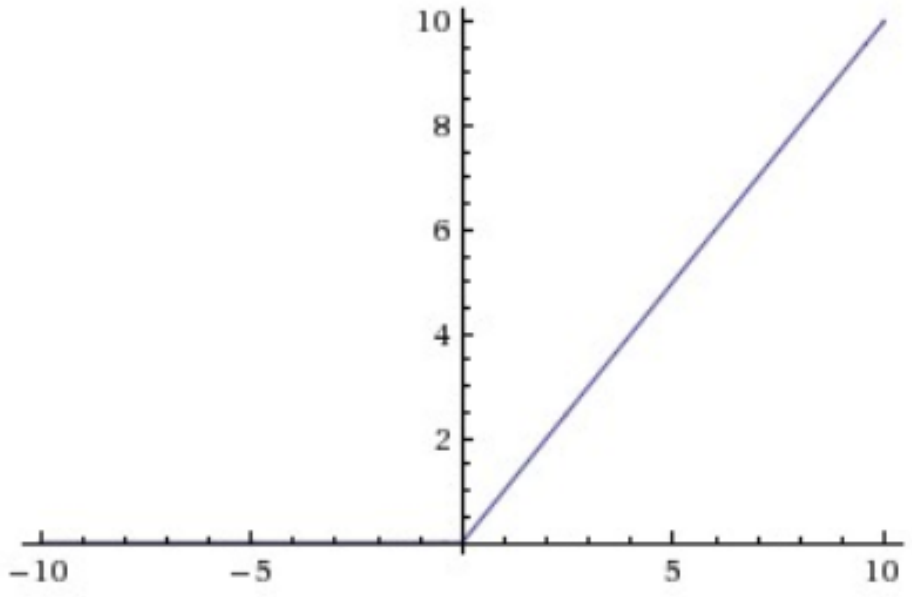
$$= \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

sigmoid 함수는 확률과 유사하게
0과 1 사이의 값을 출력함

$\sigma(\infty) = 1$

자주 사용되는 활성화 함수(2)

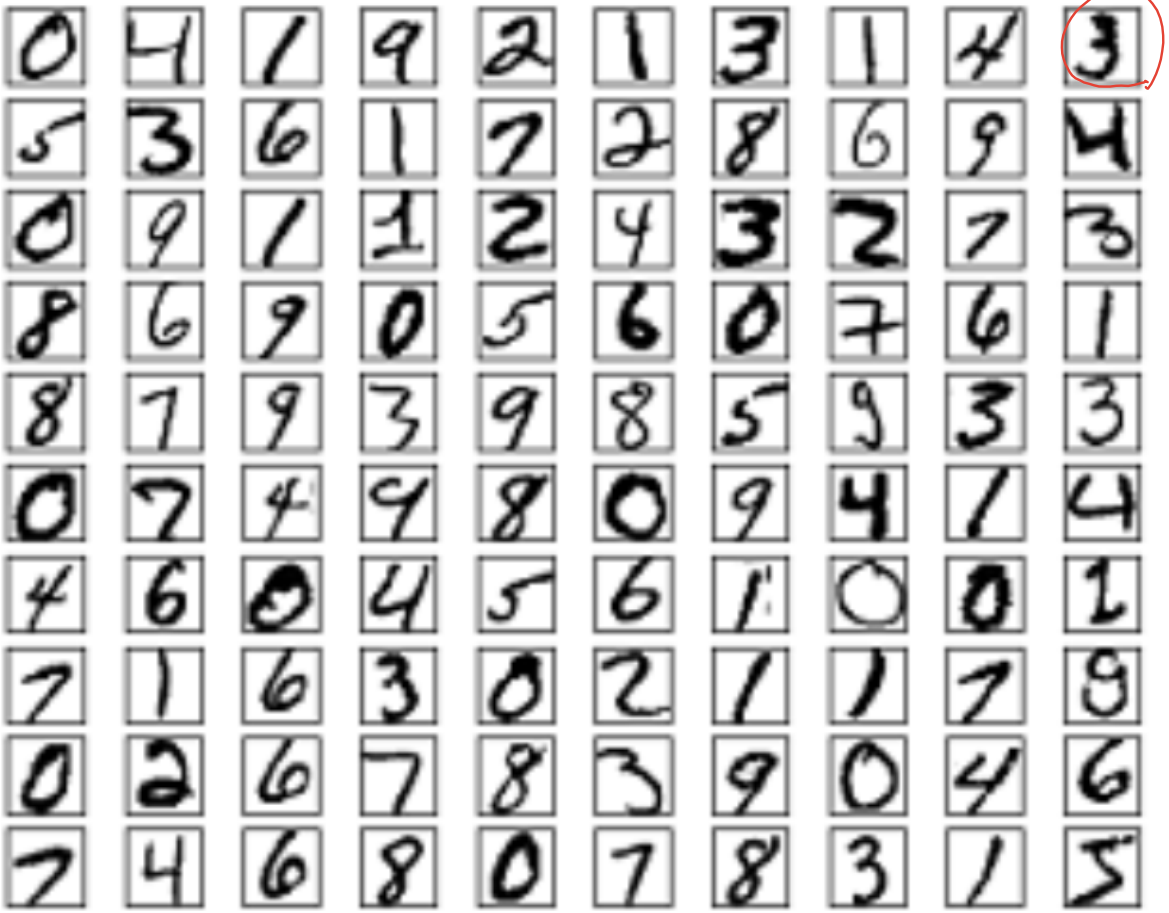
- ReLU: Rectified linear unit



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

신경망 학습

- 미리 준비된 입-출력 데이터($\mathbf{x}_i$, $\mathbf{y}_i$, $i = 1,...,N$)를 이용하여 각 뉴런의 가중치($\mathbf{w}_j$)를 결정하는 과정을 훈련(training)이라고 함



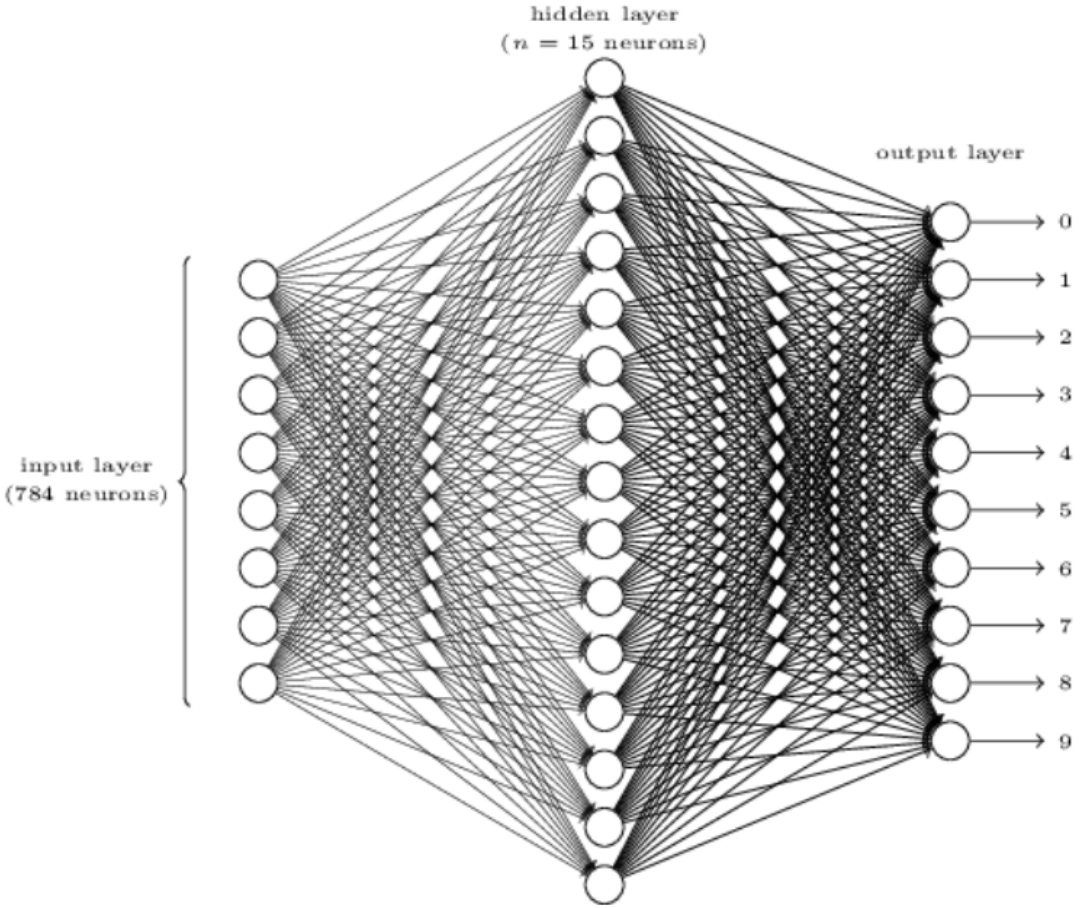
영상 X

이것을 3으로 해석해야됨→

$(X_i, y_i) \rightarrow w_j$

사례: Hand-written digit recognition

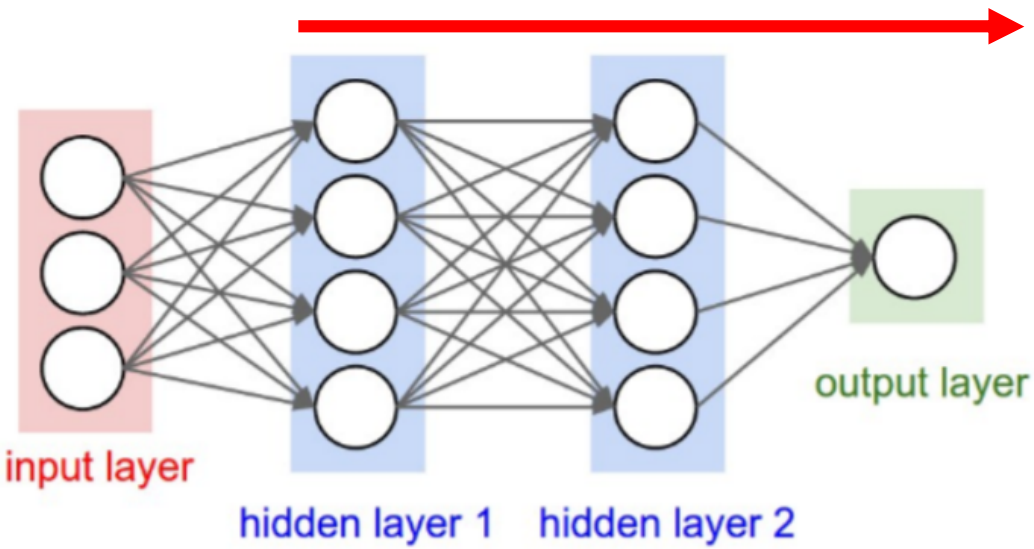
- 입력: 28x28 크기의 이진 영상. 784 크기의 벡터
- 출력: 0~9 까지의 10개의 클래스
- 은닉층: 15개의 뉴런으로 구성한다고 가정



학습시켜야 하는 계수 갯수:
784x15 + 15x10 = 11,910

Feed-forward computation

- 현재의 계수들($\mathbf{w}_j$)을 이용하여 각 은닉층 뉴런과 출력 뉴런의 값을 차례로 계산하는 과정



$$h_j(\mathbf{x}) = f\left(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}\right)$$

$$o_k(\mathbf{x}) = g\left(w_{k0} + \sum_{j=1}^{J} h_j(\mathbf{x}) w_{kj}\right)$$

D: 입력 데이터 차원
v_{ji}: 은닉층의 계수
J: 은닉층 뉴런 수
w_{kj}: 출력 뉴런의 계수

Error (Loss) function

- 계수를 학습시키기 위해 각 입력 데이터로 부터 생성된 출력값이 실제 클래스값과 얼마나 일치하는가에 대한 측정 수단이 필요
- 예: 숫자 인식의 경우 6에 해당하는 영상을 입력시켰을 때 원하는 출력값은

$$\mathbf{y} = (0,0,0,0,0,0,1,0,0,0)$$

신경망의 출력이

$$\mathbf{a} = (0,0,0.3,0.2,0,0.1,0,0,0,0.4)$$

라 하면 오차는

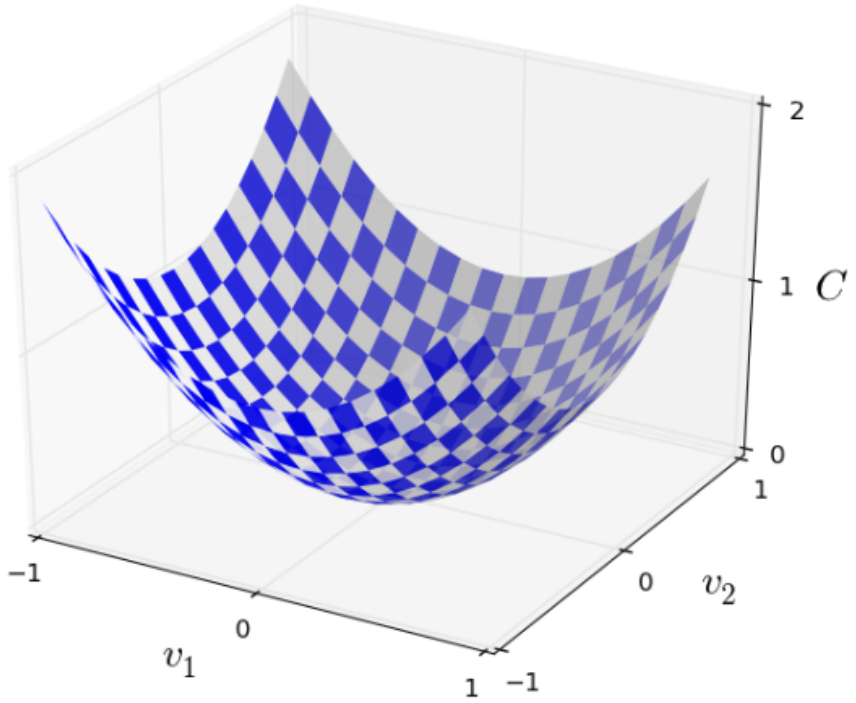
$$e = \|\mathbf{y} - \mathbf{a}\|$$

Error 함수를 다음과 같이 정의할 수 있음

$$E = \frac{1}{N}\sum_i \|\mathbf{y}_i - \mathbf{a}_i\|^2$$

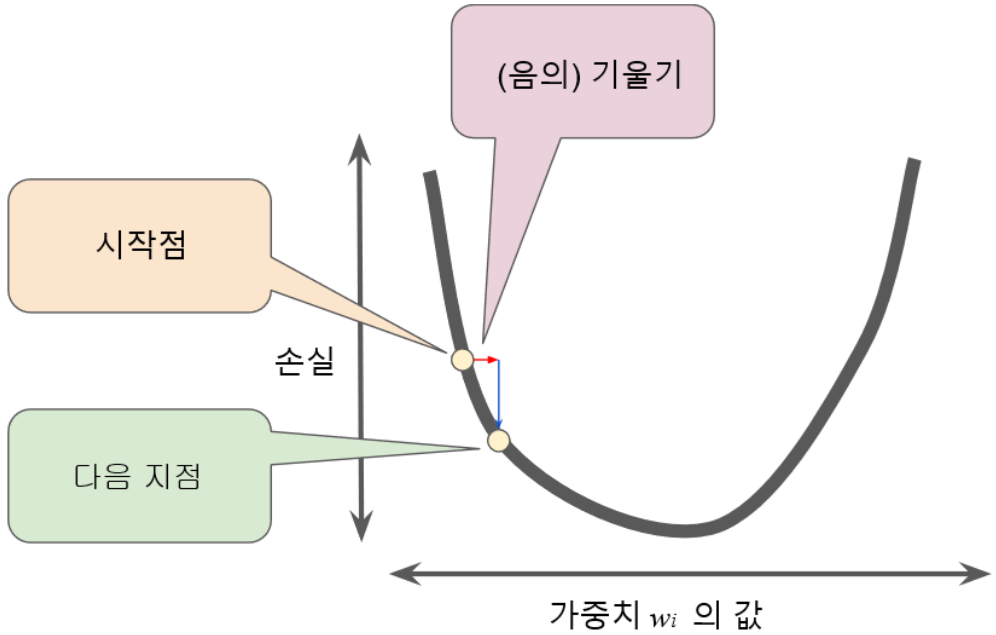
Optimization

- Loss 함수는 가중치 W의 함수이며, loss를 최소화하는 가중치들 W를 구하는 것이 문제임
- **앞 필기체 숫자 인식의 경우 11,910의 계수가 있으므로 L을 최소화하는 11,910개의 계수를 방정식으로 푸는 것은 불가능에 가까움**
- **문제:** 다음 그림에서와 같이 $E(w_1, w_2,....,w_m)$을 최소화하는 가중치들 $(w_1,....,w_m)$ 을 구함



Gradient descent (경사 하강법)

- 현재점에서의 gradient를 계산하여 그 방향으로 조금씩 이동하여 최저점에 도달하는 방식
- 최저점에서는 gradient가 0이므로 최저점에 도달하면 더 이상 움직이지 않음



Gradient descent

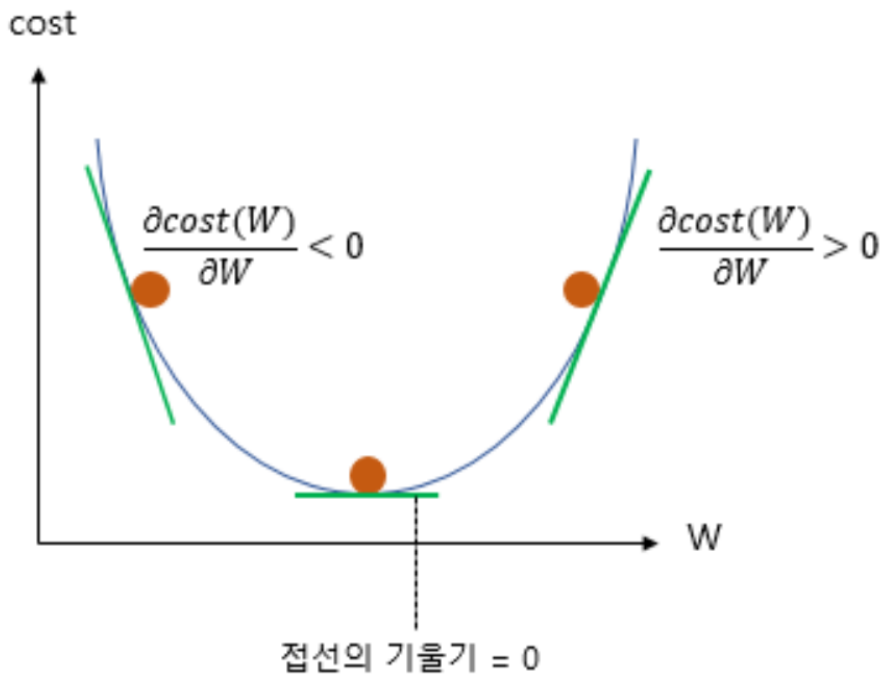
- Gradient descent를 사용하려면 각 가중치로 error 함수를 편미분한 식을 사용

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta \frac{\partial E}{\partial \boldsymbol{w}^{t}}$$

- 윗 식에서 E는 error 함수, $\boldsymbol{w}^{t}$는 t 시점에서의 가중치, $\boldsymbol{w}^{t+1}$ 은 다음 시점에서의 가중치, η는 learning rate임. η의 크기에 따라 수렴속도가 달라짐
- 가중치는 매 데이터 샘플을 읽을 때마다 새로 갱신(update)하거나, 일정 개수의 sample(batch size)을 처리한 다음 갱신하기도 함
- **Epoch:** 전체 입력 데이터로 가중치 갱신을 한 번 마치는 것. 일반적으로 신경망 학습과정은 수백번의 epoch을 반복하는 것으로 수행됨

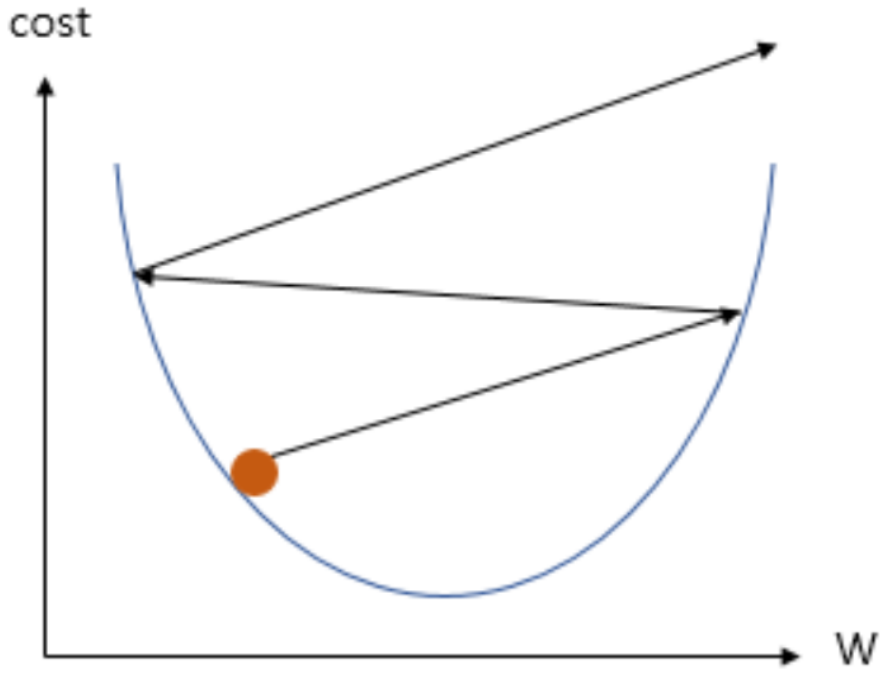
경사 하강법 개념

- 접선의 기울기에 따라 값을 조절하는 방향을 정함



경사 하강법에서의 학습률

- 학습률이 너무 크면 결과가 수렴하지 못할 수 있음



Gradient descent 사례

- 함수 $y = f(x) = x^2 + 3x + 7$ 의 최소값을 구하는 경우
- 함수를 미분하여 기울기가 0이 되는 x값을 구하면 최소값을 구할 수 있음
  - 기계 학습에서는 변수의 개수가 많으므로 이 방법을 사용할 수 없음
- 초기값을 $\boldsymbol{x_0 = 10, \eta = 0.3}$ 으로 두고 gradient descent를 수행

$$x^{t+1} = x^t - \eta \frac{\partial f}{\partial x} = x^t - 0.3 \cdot (2x^t + 3)$$

x^t	$f'(x^t)$	x^{t+1}
10	23	3.1
3.1	9.2	0.34
0.34	3.68	-0.76
-0.76	1.47	-1.2
-1.2	0.59	-1.38
-1.38	0.24	-1.45
-1.45	0.094	-1.48
-1.48	0.038	-1.49

Gradient descent 사례(계속)

- 함수 $y = x^2 + 3x + 7$ 의 최소값을 구하는 경우
- 초기값을 $\boldsymbol{x_0 = 10, \eta = 1.1}$ 로 두고 gradient descent를 수행

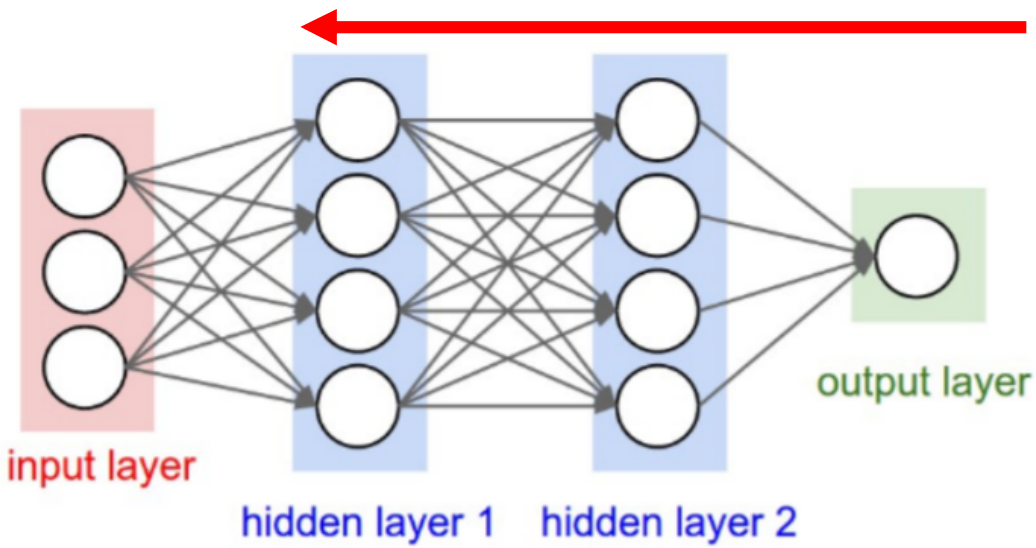
$$x^{t+1} = x^t - \eta \frac{\partial f}{\partial x} = x^t - 1.1 \cdot (2x^t + 3)$$

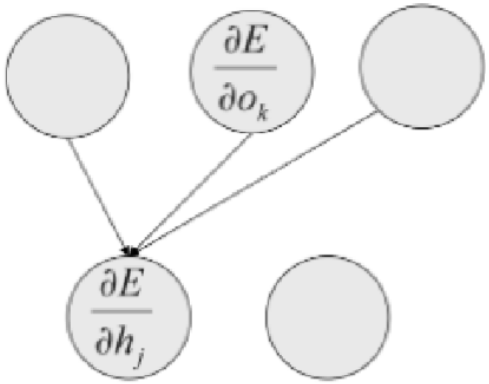
x^t	$f'(x^t)$	x^{t+1}
10	23	-15.3
-15.3	-27.6	15.1
15.1	33.1	-21.4
-21.4	-39.7	22.3
22.3	47.7	-30.1
-30.1	-57.2	32.8
32.8	68.7	-42.7

- 이 경우 η 가 너무 커서 결과가 수렴하지 않음

Backpropagation

- 가중치를 update 할 때는 출력층을 먼저 update 하고 뒷 단의 은닉층부터 앞 단으로 내려오면서 계산





비용 함수(Cost function)

- 신경망에 의해 예측된 값과 실제 클래스값의 차이를 정의하는 함수를 비용함수라고 함
- 훈련 단계(training step)에서는 테스트 데이터를 이용하여 비용 함수값을 줄이는 방향으로 신경망의 계수들을 조절함

MSE(평균제곱오차) 비용 함수

- 기본적인 비용함수로는 평균제곱오차(Mean squared error)를 사용
- 훈련 데이터의 입력이 x에 대해 예측된 값 H(x)는 다음과 같이 계산된다고 가정. W와 b는 신경망의 계수임

$$H(x) = Wx + b$$

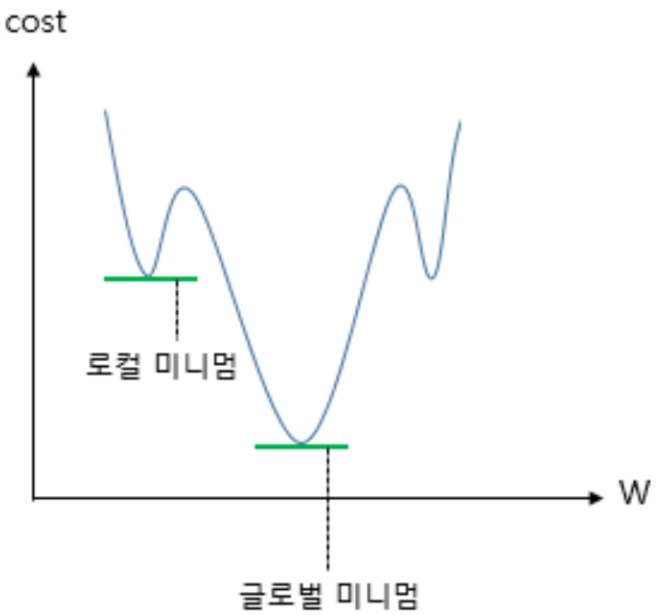
- 훈련 데이터의 클래스값이 y, 데이터의 개수가 n으로 주어졌을 때 MSE는 다음과 같이 정의됨

$$cost(W, b) = \frac{1}{n}\sum_{i=1}^{n}[y^{(i)} - H(x^{(i)})]^2$$

- 이 비용을 줄이는 방향으로 W와 b값을 조절하는 훈련이 이루어짐

Logistic Regression – 이진 분류

- 데이터 분류가 binary로 이루어지는 경우를 이진 분류라고 함(예: Spam 메일 판정)
- MSE를 비용함수로 사용하는 경우 Sigmoid 함수와 결합되면 함수 형태가 다음과 같이 될 수 있음



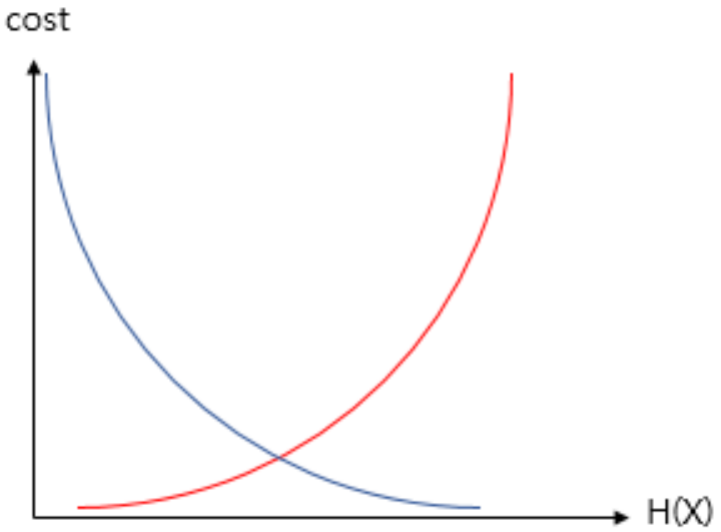
- MSE로 인해 발생하는 오류를 피하기 위해 logistic regression을 사용

Logistic Regression 원리

- 실제값이 0일 때 y값이 1에 가까와지면 오차가 커지고, 실제값이 1일 때 y값이 0에 가까와지면 오차가 커지도록 다음과 같은 함수를 사용

$$\text{if } y = 1 \rightarrow \text{cost}\,(H(x), y) = -\log(H(x))$$

$$\text{if } y = 0 \rightarrow \text{cost}\,(H(x), y) = -\log(1 - H(x))$$



$$0 \leq H(x) \leq 1$$

Logistic Regression 함수

- 비용함수를 다음과 같이 정의

$$cost(H(x), y) = -\left[ylogH(x) + (1-y)\log\left(1-H(x)\right)\right]$$

- 이 함수값은 H(x)와 y가 모두 0 또는 1이면 값이 0이고, 하나가 0, 다른 것이 1이면 값이 ∞가 됨
- 전체 비용은 다음과 같이 정의됨

$$J(W) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}logH\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-H\left(x^{(i)}\right)\right)\right]$$

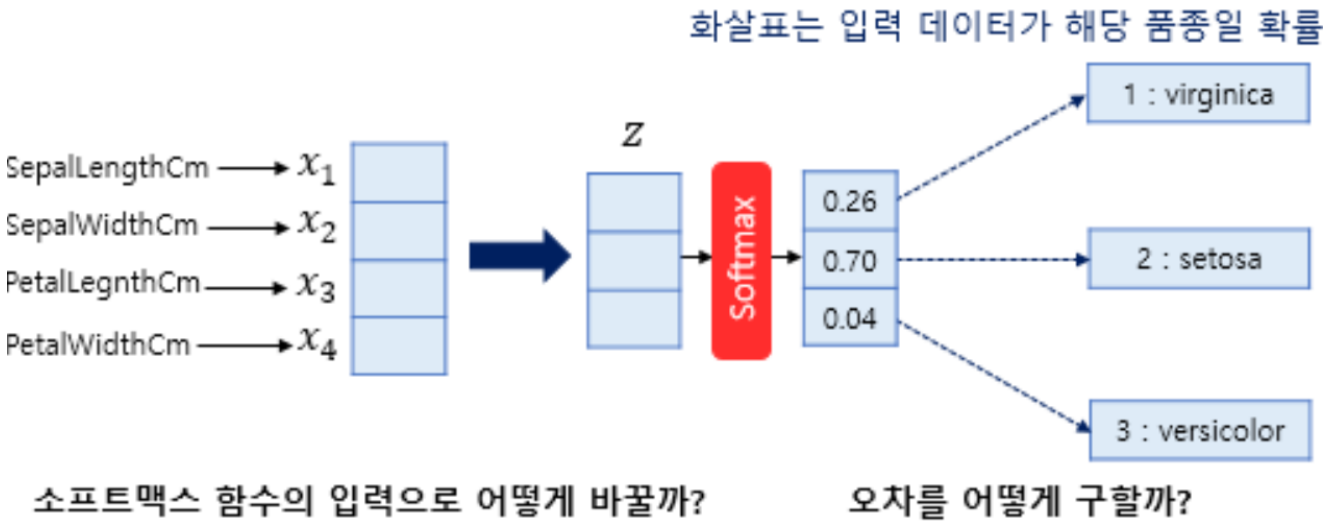
Softmax Regression – 다중 클래스 분류

- 출력의 선택값이 3개 이상일 경우 결과를 확률로 나타내는 방식으로 Softmax regression이 사용됨. 모든 출력값은 0과 1 사이값을 가지고 합이 1이 됨
- 출력의 개수가 k개 일 때 i번째 원소를 z_i, i번째 클래스가 정답일 확률이 p_i라 하면 softmax 함수는 p_i를 다음과 같이 정의함

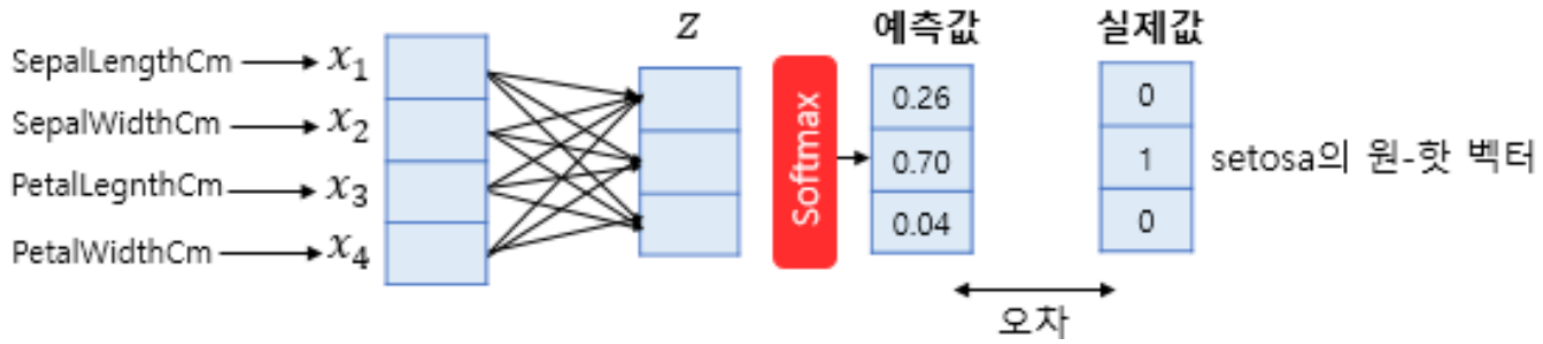
$$p_i = \frac{e^{z_j}}{\sum_{j=1}^{k} e^{z_j}}, \qquad for\ i = 1,2,\dots,k$$

Softmax regression 사례

- k = 3 일 때 softmax 출력



- 실제값과의 오차 계산



Softmax regression 비용함수

- 출력의 개수가 k개 일 때 softmax의 비용은 다음과 같이 정의됨
- Cross entropy 함수

$$cost(W) = -\sum_{j=1}^{k} y_j \log(p_j)$$

  - 확률과 y가 모두 1 일 때 비용은 0이 되고, 그렇지 않으면 비용이 커짐

- n개의 전체 데이터에 대한 평균은 다음과 같이 정의됨

$$cost(W) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_j^{(i)} \log(p_j^{(i)})$$