AI 알고리즘

손실함수와 경사하강법

수퍼스타를 키우는 곳
전주대학교

학습내용

- 손실함수
- 역전파알고리즘

학습목표

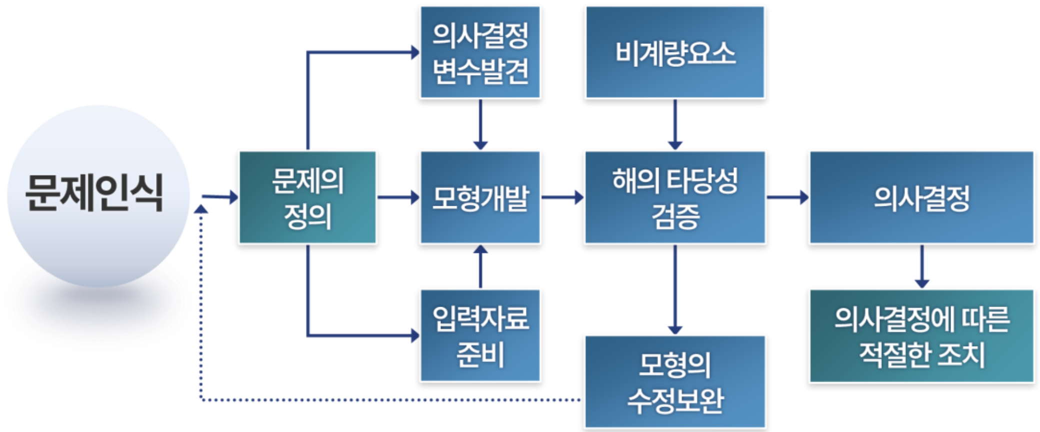
- 손실함수의 정의와 종류를 확인할 수 있다.
- 역전파알고리즘에 대해 이해할 수 있다.

손실함수와 역전파알고리즘

손실함수

손실함수

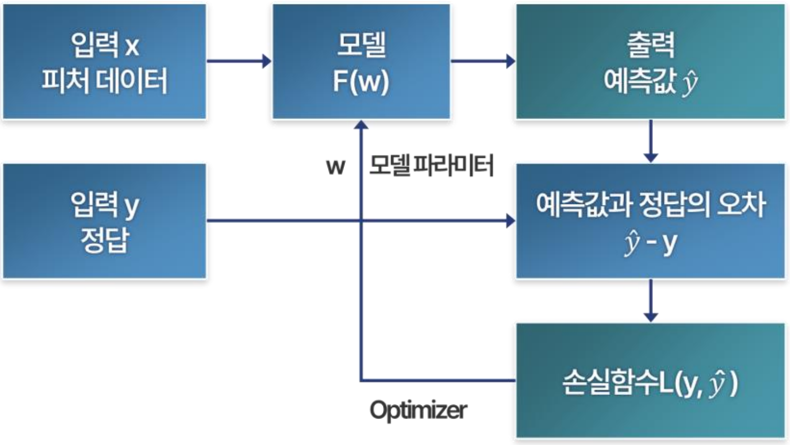
❖ 수리적 모형 개발



❖ 지도 학습의 평가 척도

손실함수

❖ 지도 학습의 평가 척도(회귀)



❖ 지도 학습의 평가 척도(회귀)

- 손실함수의 역할
  - 입력 데이터를 모델에 넣어 출력
  - 출력된 값을 타겟 데이터(정답)와 비교하여 나온 오차를 가지고 손실함수를 만듦
  - 손실함수를 최소화하기 위해 모델의 파라미터 값을 원하는 방향으로 수정

손실함수

❖ 손실함수의 종류(회귀)

평균제곱오차 (MSE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

평균제곱근오차 (RMSE)

$$L = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

많이 사용되는 형태

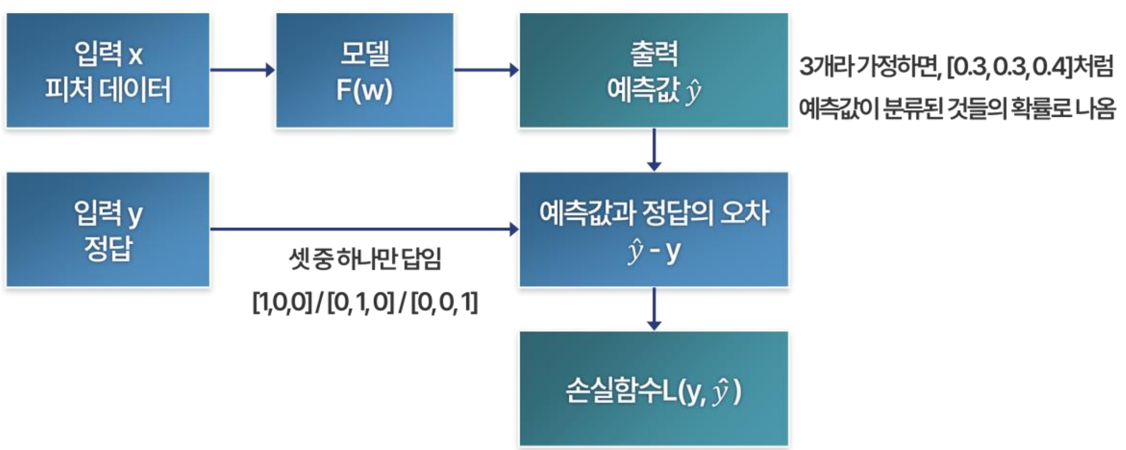
평균절대값오차 (MAE)

$$L = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

미분이 어려움

$\hat{y}_i$ = 예측값($i = 1,2,\cdots,n$)
y_i = 정답($i = 1,2,\cdots,n$)
n = 데이터의 개수
L = 손실함수

❖ 지도 학습의 평가 척도(회귀)



손실함수

❖ 엔트로피

- 1860년도 경 독일의 클라우지우스(Clausius)
  - 엔트로피(Entropy)라는 개념을 제시
    - 자연계의 모든 변화는 반드시 엔트로피가 증가하는 방향으로 일어난다.
      S=Q(열)/T(온도)
- 루트비히 볼츠만
  - 한 번 사용한 에너지는 두 번 다시 사용할 수 없다.
  - 엔트로피 $S=klogW$
  - 엔트로피는 무질서한 정도다.
- 클라우드 섀넌(Claude Shannon)
  - 1948년 벨 연구소 근무 당시 정보를 보낼 때 얼마를 받는지 과금 체계에 대한 고민
  - 통신의 수학적 이론(A Mathematical Theory of Communication)이라는 논문 발표
    - 정보량이 많을 수록 통신료를 많이 부과해야 함

손실함수

❖ 엔트로피

- 1860년도 경 독일의 클라우지우스(Clausius)

A 0, 0, 0, 0, ..., 0(1억 개)

B 0, 1, 0, 0 , ... , 1(Random 100개)

정보량
A<B

- 정보를 보낼 때 필요한 비트 공식

$$H(P) = H(x) = -\sum P(x)\log_2 P(x)$$

$P(x)$ = 각 정보들의 출현 확률
$H(x)$ = 정보를 보내는 데 필요한 비트의 수

$$H(S) = -\sum_i p_i log_2(p_i)$$

p_i = label i의 확률

손실함수

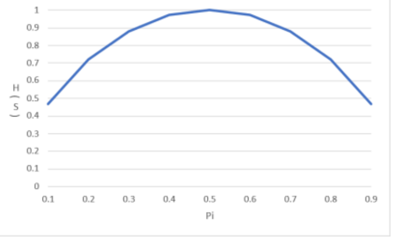
❖ 엔트로피

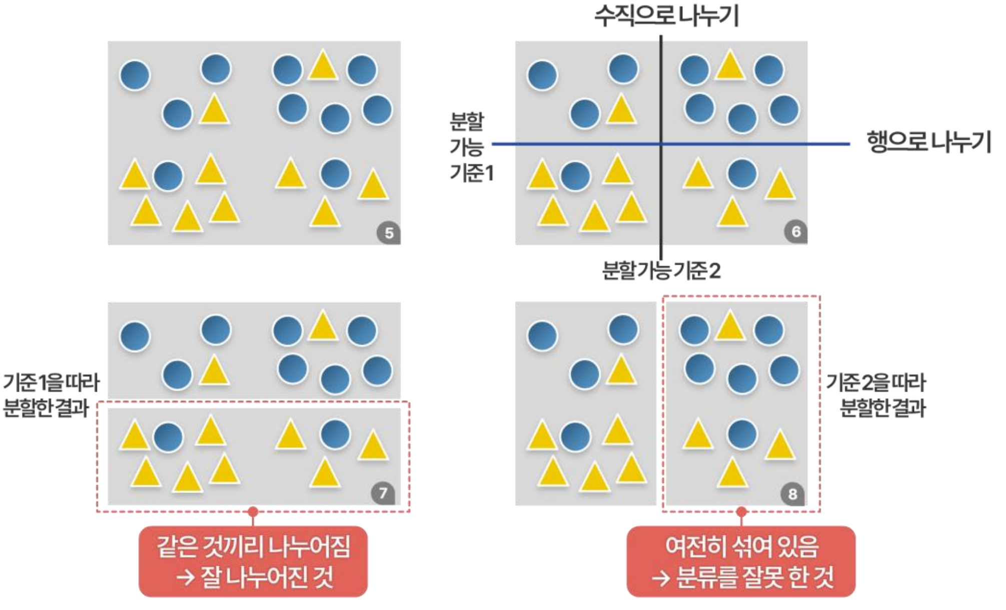
- 2가지로분류한다가정
  - 정확히한쪽으로양성 음성나눌경우

$$H(S) = -1log_2 1 - 0log_2 0 = 0$$

  - 정확하게반반있을경우

$$H(S) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$$





손실함수

❖ 교차엔트로피

- 교차엔트로피(Cross Entropy)
  - 모델에서 예측한 확률값이 실제값과 비교했을 때 틀릴 수 있는 정보량을 의미함
  - 두 확률 분포의 차이를 구하기 위해 사용
    - 모델에서 예측한 확률과 정답 확률 둘 다 사용

$$L = -\sum_{i=1}^{n} q(x_i) log p(x_i)$$

$p(x_i)$ = 모델링을 통하여 구한 분포($i = 1,2,\cdots,n$)
$q(x_i)$ = 실제 정답의 분포($i = 1,2,\cdots,n$)

  - 두 값이 완전히 다른 경우 값이 무한대
  - 두 값이 완전히 같은 경우 엔트로피 값이 0이 됨
- 교차엔트로피를 사용하는 이유
  - 차이가 날 경우 더 큰 패널티를 주는 개념

❖ 손실함수 종류

- 교차엔트로피

$$L = -\sum_{i=1}^{n} y_i log \hat{y}_i$$

- 이진 크로스엔트로피

$$L = \frac{1}{n}\sum_{i=1}^{n} (-y_i log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i))$$

- 범주형 크로스엔트로피

$$L = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{c} t_{ij} log(y_{ij})$$

손실함수

❖ Kullback-Leibler Divergence(KL Divergence)

- 서로 다른 두 분포의 차이를 측정하는데 쓰이는 Measure
- 크로스 엔트로피와 엔트로피의 차이로 계산
  - 솔로몬 쿨백
  - 리차드 라이블러
- 항상 크로스엔트로피가 크므로 KL 다이버전스는 항상 0보다 큰 값을 가짐
- 엔트로피가 고정이므로, 크로스엔트로피를 최소화 시키는 것이 KL 다이버전스를 최소화 시키는 것

$$D_{KL}(q|p) = -\sum_{i=1}^{n} q(x_i)[\log p(x_i) - \log q(x_i)]$$

`class BinaryCrossentropy` : Computes the cross-entropy loss between true labels and predicted labels.

`class BinaryFocalCrossentropy` : Computes focal cross-entropy loss between true labels and predictions.

`class CategoricalCrossentropy` : Computes the crossentropy loss between the labels and predictions.

`class CategoricalFocalCrossentropy` : Computes the alpha balanced focal crossentropy loss.

`class CategoricalHinge` : Computes the categorical hinge loss between `y_true` & `y_pred`.

`class CosineSimilarity` : Computes the cosine similarity between labels and predictions.

`class Hinge` : Computes the hinge loss between `y_true` & `y_pred`.

`class Huber` : Computes the Huber loss between `y_true` & `y_pred`.

`class KLDivergence` : Computes Kullback-Leibler divergence loss between `y_true` & `y_pred`.

`class LogCosh` : Computes the logarithm of the hyperbolic cosine of the prediction error.

`class Loss` : Loss base class.

`class MeanAbsoluteError` : Computes the mean of absolute difference between labels and predictions.

`class MeanAbsolutePercentageError` : Computes the mean absolute percentage error between `y_true` & `y_pred`.

`class MeanSquaredError` : Computes the mean of squares of errors between labels and predictions.

손실함수

❖ 손실함수 종류 및 선택

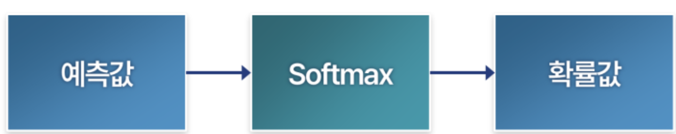
- A general and adaptive robust loss function
  - 데이터 분포에 따라 적용하여 손실함수가 변화하는 방식
  - 이상치를 거르고 중요한 데이터에 맞추어 학습할 수 있도록 함
- 강화학습을 이용하여 학습과정에서 최적의 손실함수를 찾는 방식도 진행되고 있음

❖ 소프트맥스(Softmax) 함수

- 예측값들을 해당 클래스의 확률값으로 바꿔주는 함수

y_1, y_2 $\quad$ $y_1 + y_2$ $\quad$ $\frac{y_1}{y_1+y_2}, \frac{y_2}{y_1+y_2}$

- 예측값이 0이나 음수 값이 나올 경우, 다 더하면 마이너스가 됨
  - 지수함수를 사용하여 0이나 마이너스가 나와도, 0 이상의 값을 갖게 함



y_1, y_2 $\quad$ $e^{y_1} + e^{y_2}$ $\quad$ $\frac{e^{y_1}}{e^{y_1}+e^{y_2}}, \frac{e^{y_2}}{e^{y_1}+e^{y_2}}$

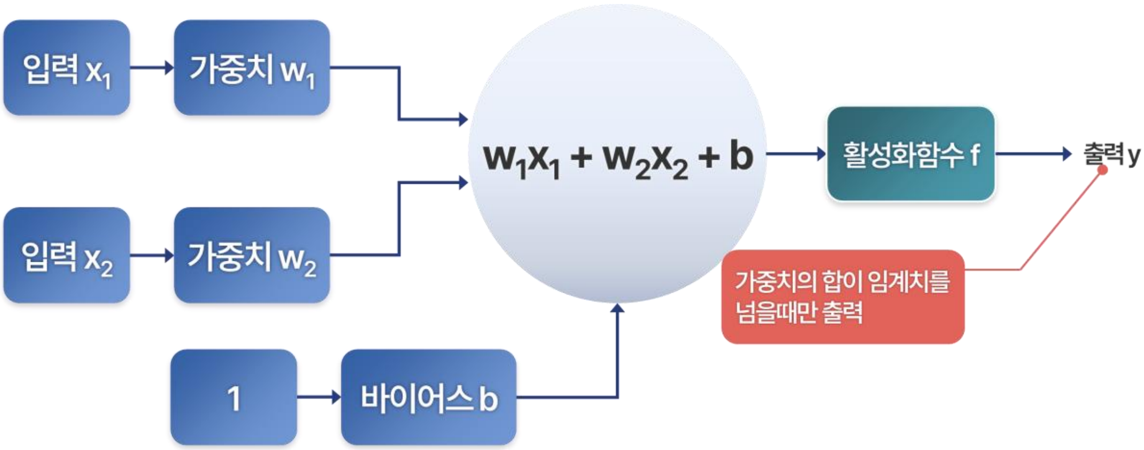
손실함수와 역전파알고리즘

역전파알고리즘

역전파 알고리즘

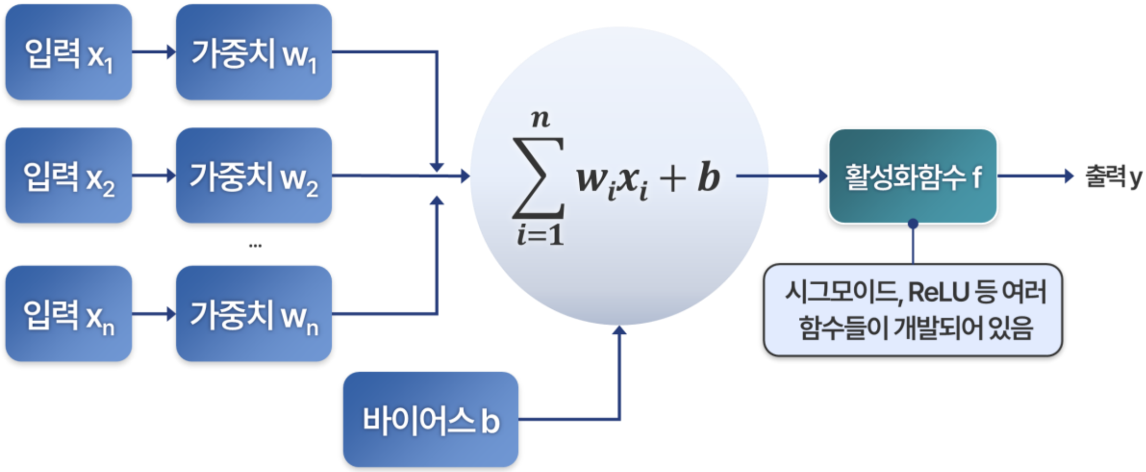
❖ 퍼셉트론(Perceptron)

- 1957년 로젠블라트가 고안한 인공 신경망
  - 로젠블라트와 퍼셉트론



❖ 다층 퍼셉트론

- 퍼셉트론의 한계
  - XOR 문제를 해결하지 못함
    - 은닉층을 두어 해결
- 다층 퍼셉트론(Multilayer Perceptron: MLP)
  - 퍼셉트론에 은닉층을 두는 개념
    - 입력층과 출력층 사이에 은닉층을 둠



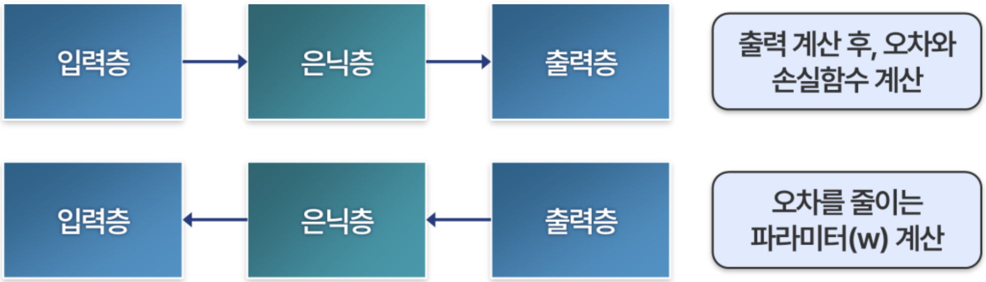
역전파알고리즘

❖ 다층 퍼셉트론

- 실제 출력과 모델 출력 사이의 오차 발생
  - 오차의 제곱 합을 평균하는 MSE를 기준
  - 다층 퍼셉트론의 손실함수들을 바꿔가며, 모델의 파라미터 값을 구함

$$E = \frac{1}{m}\sum(y_i - \hat{y}_i)^2$$

E	= 예측값과 정답 간의 평균 제곱 오차(MSE)
m	= 데이터의 개수



- w 구하기
  - 미분의 체인룰을 이용하여 유도 가능
    - 미분이 가능해야 한다는 조건이 있음

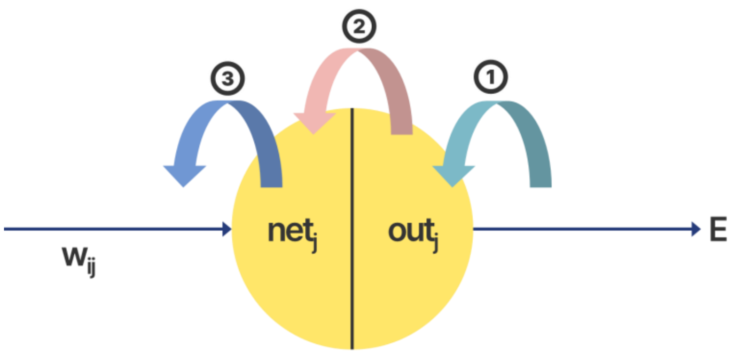
$$w(t+1) = w(t) - \eta\frac{\partial E}{\partial w}$$

$w(t+1)$	= 바뀐 후의 가중치
$w(t)$	= 바뀌기 전 가중치
η	= 학습률(경사하강법에서 사용)
$\frac{\partial E}{\partial w}$	= 현재 값에서의 기울기

역전파알고리즘

❖ 다층 퍼셉트론

- 출력층의 모습



$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial out_j}\frac{\partial out_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

w_{ij}	입력 가중치
net_j	입력 가중치로 인해 변환된 값
out_j	활성화함수에 의해 변환된 값

- 한번에 미분이 불가능해 세 단계의 곱으로 구함 → 체인룰

$$\frac{\partial E}{\partial out_j} = \frac{\partial}{\partial out_j}\sum \frac{1}{2}(target_k - out_k)^2$$

출력값 변환에 따른 오차의 변화율

$$\frac{\partial out_j}{\partial net_j} = \frac{\partial f(net_j)}{\partial net_j} = f'(net_j)$$

활성화 함수의 미분값

$$\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\left(\sum_{k=0}^{n} w_{kj}\cdot out_k\right) = \frac{\partial}{\partial w_{ij}} w_{ij}out_i = out_i$$

계수 형태의 값만 알면 구할 수 있음

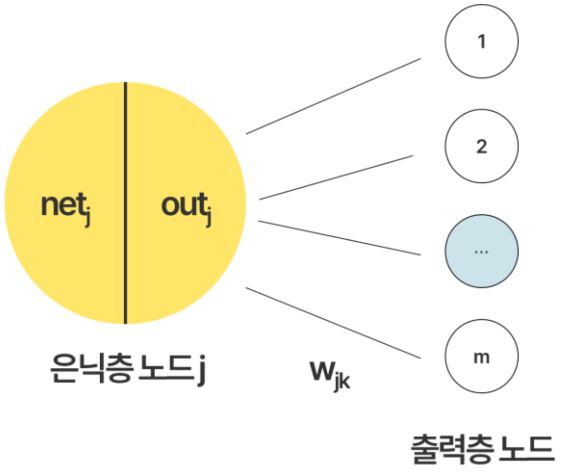
$$\frac{\partial E}{\partial w_{ij}} = (out_j - target_j)\times f'(net_j)\times out_i$$

역전파알고리즘

❖ 다층 퍼셉트론

- 은닉층
  - 미분의 체인룰을 이용하여 유도 가능
    - 활성화 함수에 나가는 것
    - 가중치 w로 들어오는 것의 합
    - w로 표현하는 것
    - 세 가지의 곱의 형태로 일어남

$$\frac{\partial E}{\partial out_j} = \sum_{k \in L}\left(\frac{\partial E}{\partial out_k}\frac{\partial out_k}{\partial net_k}\frac{\partial net_k}{\partial out_j}\right)$$
$$= \sum_{k \in L}\left(\frac{\partial E}{\partial out_k}\frac{\partial out_k}{\partial net_k}w_{jk}\right)$$



역전파알고리즘

❖ 다층 퍼셉트론

- 델타(Delta)
  - 여러 은닉층이 있어도 역방향으로 계산이 가능케 함

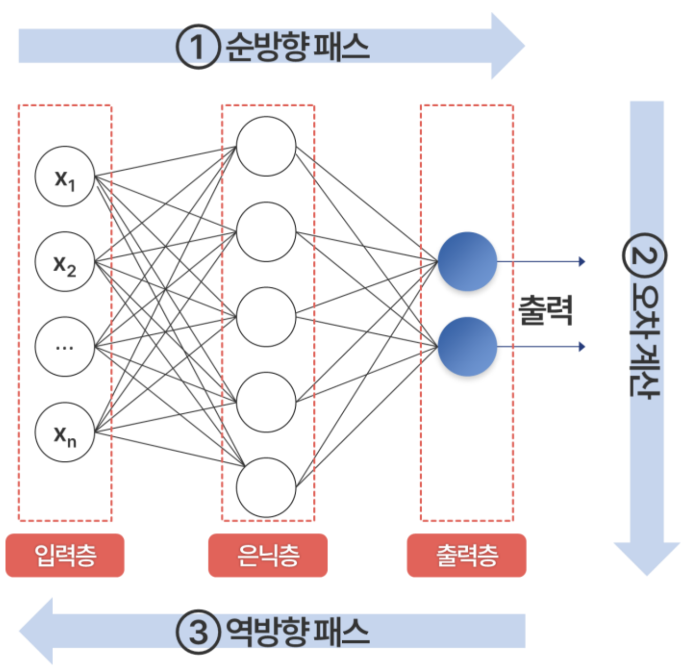
$$\frac{\partial E}{\partial w_{ij}} = \delta_j out_i$$

출력층

$$\delta_j = \left(out_j - target_j\right) f'(net_j)$$

은닉층

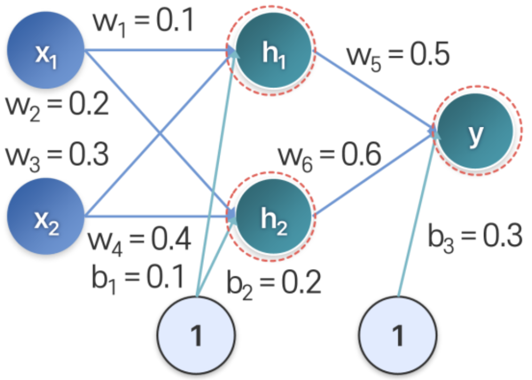
$$\delta_j = \left(\sum_k w_{jk}\delta_k\right) f'(net_j)$$



역전파알고리즘

❖ 다층 퍼셉트론

- 델타(Delta)
  - 여러 은닉층이 있어도 역방향으로 계산이 가능케 함



$$z_1 = w_1x_1 + w_3x_2 + b_1 = 0.1$$

$$a_1 = \frac{1}{1+e^{-z_1}} = \frac{1}{1+e^{-0.1}} = 0.524979$$

$$z_2 = w_2x_1 + w_4x_2 + b_2 = 0.2$$

$$a_2 = \frac{1}{1+e^{-z_2}} = \frac{1}{1+e^{-0.2}} = 0.549834$$

$$z_y = w_5a_1 + w_6a_2 + b_3 = 0.892389$$

$$a_y = \frac{1}{1+e^{-z_y}} = \frac{1}{1+e^{-0.892389}} = 0.709383$$

$$E = \frac{1}{2}\left(target_y - out_y\right)^2 = \frac{1}{2}(0 - 0.709383)^2 = 0.251612$$

$$E = \frac{1}{2}\left(target_y - out_y\right)^2 = \frac{1}{2}(0 - 0.709383)^2 = 0.251612$$

$$E = \frac{1}{2}\left(target_y - out_y\right)^2 = \frac{1}{2}(0 - 0.699553)^2 = 0.244687$$

...

$$E = \frac{1}{2}\left(target_y - out_y\right)^2 = \frac{1}{2}(0 - 0.005770)^2 = 0.000016$$

  - 1만번 시행하여 w값을 구함