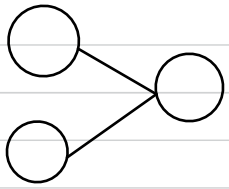
학습 : 훈련 데이터로 부터 가중치 매개변수의 최적값을 자동으로 획득. (w, b)

손실함수 : 학습할 수 있도록 해주는 지표 (loss function)

optimizer → How. 경사하강법 등등.

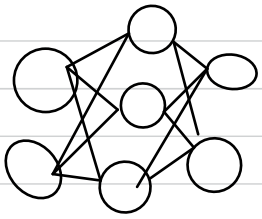
<이 손실함수의 결과 값을 가장 작게 만드는 가중치 매개변수를 찾는 것이 학습의 목표>

Perceptron



- 단층. 선형
- 직접 w, b를 선정

Neural network



- 다층. 비선형
- 자동으로 w, b 선정

<기본 코딩>

x + 알고리즘 ⇒ y

<deep learning> = end to end machine learning

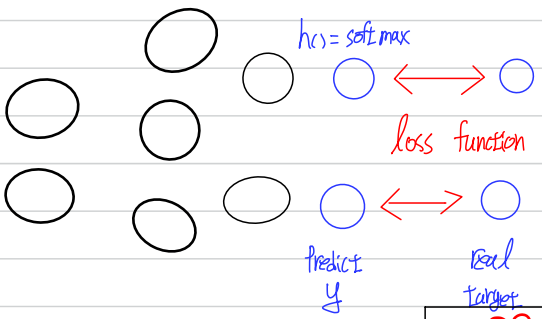
x + y ⇒ 알고리즘 (= 가중치)

= '처음부터 끝까지'

= '데이터에서 목표한 결과를 사람의 개입없이'

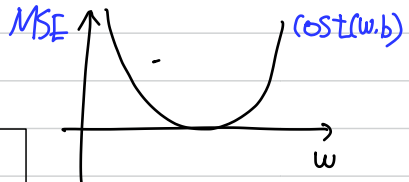
어떻게 이 지표를 가지고 자동으로 w, b를 가질수 있는 거지?

loss function



① MSE

$$cost(w,b) = \frac{1}{n}\sum_{i}^{k}(y_i - t_i)^2$$



○ y'의 활성화 함수가 소프트맥스, 시그모이드가 아닐때. (지수연산이 줄어듦)

$y_i = h(W_i x_i + b)$

$= h(w_1x_1 + w_2x_2 + \cdots b)$

$= h(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n)$

$= h(\theta^T x)$

θ_0 = bias

θ_i $(i=1,2,3\cdots)$ = weight

① ② 출력층 node가 1개일때

③ 출력층 node가 n개일때

① $cost(w,b) = J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y^i - t^i)^2$

$\frac{J(\theta)}{dw_1} = \frac{1}{2m}\sum_{i=1}^{m}(h(x^i) - t^i)^1$

$= \frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 - t^i)$ / dw_1 (∈ dθ)

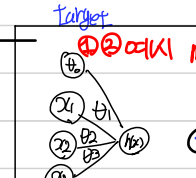
$= \frac{1}{2m}\sum_{i=1}^{m}(h(x^i) - t^i)x_1^i$

(cf) $\frac{dJ(\theta)}{d\theta_3} = \frac{1}{2m}\sum_{i=1}^{m}(y^i - t^i)b^i$ $(b^i = x_0)$

$\therefore \frac{dJ(\theta)}{dw_i} = \frac{1}{2m}\sum_{i}^{m}(y^i - t^i)x_i^i$

$\therefore \begin{bmatrix} \frac{\partial J(\theta)}{\partial w_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial w_n} \end{bmatrix} = \frac{1}{2m}X^T(y - t)$

② 예시 model



$h(x) = \theta_0 x_0(1) + x_1\theta_1 + x_2\theta_2$

출력층의 모든 node 표현

② $cost(w,b) = \frac{1}{m}\sum_{i}^{m}(y^i - t^i)^2$

$\frac{cost(w,b)}{dw} = \frac{1}{m}\sum_{i}^{m}((wx^i + b) - t^i)^2$ / dw

$\frac{cost(w,b)}{dw} = \frac{1}{m}(y - t)^2 = 2(y-t)\cdot\frac{dy}{dw}(y)$

$\frac{cost(w,b)}{dw} = \frac{2}{m}\sum_{i}^{m}((w_i x_i + b) - t^i)x_i$

$\frac{dLoss}{dw_i} = \frac{2}{m}\sum_{i}^{m}(y^i - t^i)x_i$

정답 node

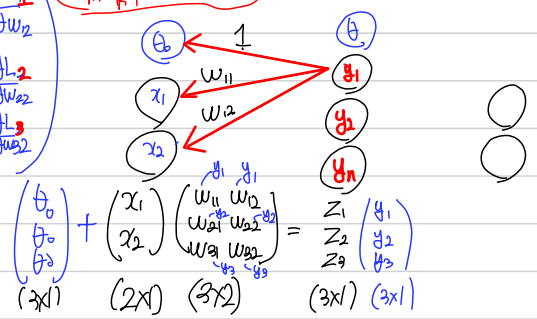
출력층 모든 node 표현

이항은 정도에 영향을 미칠뿐 방향엔 영향 X

$W_{new} = W - \eta\frac{dLoss}{dw} \rightarrow \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} - \eta\begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} \\ \frac{\partial L}{\partial w_{31}} & \frac{\partial L}{\partial w_{32}} \end{pmatrix}$

③ $\begin{pmatrix} \frac{dL_1}{dw_1} & \frac{dL_1}{dw_2} \\ \frac{dL_2}{dw_1} & \frac{dL_2}{dw_2} \\ \frac{dL_3}{dw_1} & \frac{dL_3}{dw_2} \end{pmatrix}$ $\frac{1}{m}\sum_{k}^{n}\sum_{i}^{m}(y_k^i - t_k^i)^2$

(dy) ○ 기울기 가중치 매개변수에 대한 손실 함수의 기울기. (dx)



$\begin{pmatrix} \theta_0 \\ \theta_0 \\ \theta_0 \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$

(3×1) (2×1) (3×2) (3×1) (3×1)

$b + wx = z, y$ 편미분의 행렬

무슨문제를 풀고싶냐, 즉 마지막 활성화 함수가 무엇이냐에 따라 쓰는 loss function이 다르다!

○ y_i가 선형으로 유지될때 (ex $h(x)=x$)

$y = wx+b$

$cost = \frac{1}{n}\sum_i^n (y_i - t_i)^2$

$= \frac{1}{n}\sum_i^n (wx_i+b-t_i)^2$

y_i가 비선형으로 있을때

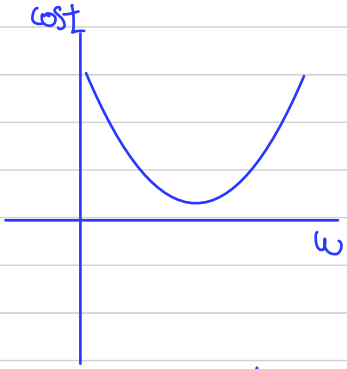
ex) $h(x) = sigmoid(x)$

$y = sigmoid(wx+b)$

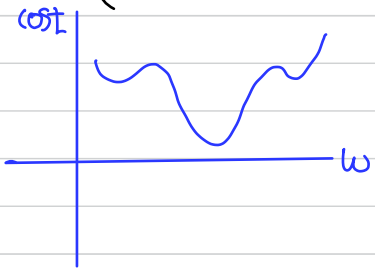
$cost = \frac{1}{m}\sum_i^m (y_i - t_i)^2$

$cost = \frac{1}{m}\sum_i^m (sigmoid(wx+b) - t_i)^2$

$= \frac{1}{m}\sum_i^m \left(\frac{1}{1+e^{-(wx+b)}} - t_i\right)^2$

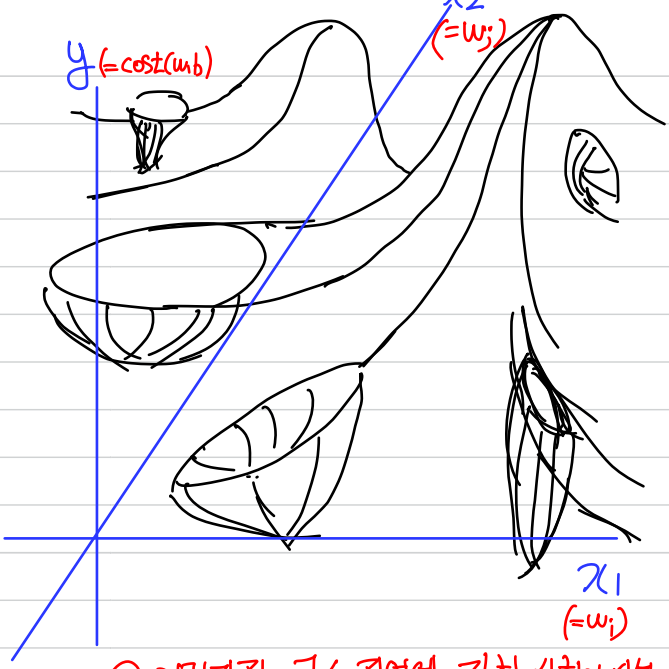


○ w에 대한 2차함수



○ 심한 비탈을 형태

○ 국소지역에 갇힐



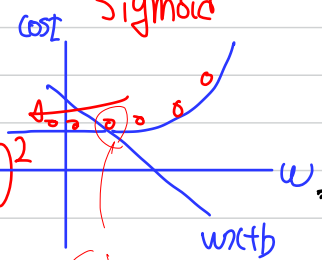
○ 여러가지 국소지역에 갇힐 상황이 생김

How solve

ReLu

$cost = \frac{1}{n}\sum_i^n (relu(wx+b)-t_i)^2$

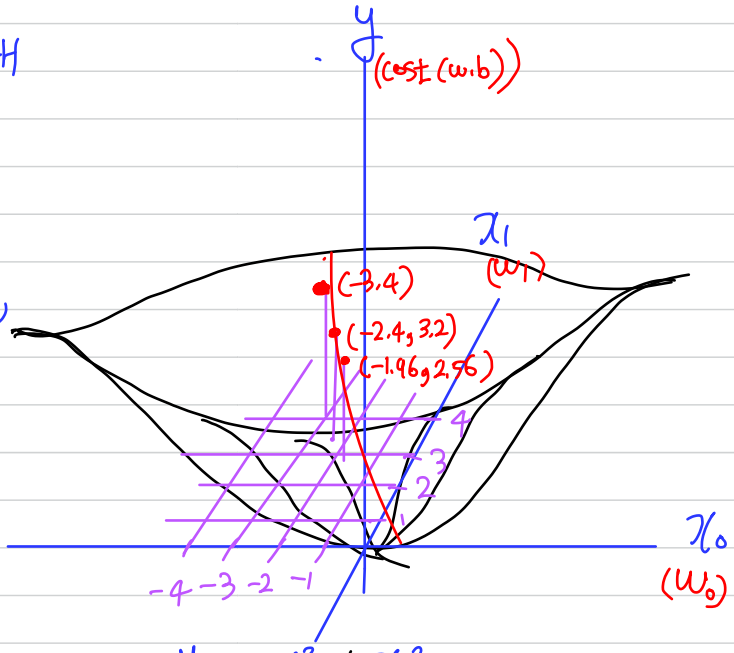
$= \frac{1}{n}\sum_i^n (\max(0, wx+b)-t_i)^2$



$cost = \begin{cases} \frac{1}{n}\sum_i^n (wx_i+b-t_i)^2 & (wx+b>0) \\ \frac{1}{n}\sum_i^n (0-t_i)^2 & (wx+b<0) \end{cases}$

Stop

$h=2$



$y = +x_0^2 + x_1^2$

$\frac{\partial y}{\partial x_0} = 2x_0$ $\frac{\partial y}{\partial x_1} = 2x_1$

(gradient descent)

경사 하강법

(otimizer의 시초)

6장에서..

$w = w - \eta \frac{dL}{dw}$

w를 조금 변화했을때 손실함수L이 얼마나 변하는지

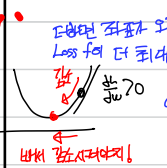
$-\frac{dL}{dw}$ 기울기. 손실함수가 최솟값이 되는 방향을 가리킴 (왼쪽, 오른쪽)

평지에 멈출수도 있어.

η 학습률. 너무 작으면 최솟값에 도달하기도 전에끝, 너무 크면 발산 해버려. (언덕을 타고 넘어)

Step: 몇번 반복했냐.

이상적으로 무한히 반복하면 최솟값는 갖는 좌표 (w_1, w_2)로 옮겨줘!



다행히 좌표 조금씩 옮겨서 Loss fn 더 최대값을 갖는쪽으로 좌표 이동시켰네. (w_1, w_2)

기울기가 양의 기울기면 더하지 말고 빼줘줘!

ex) (-3,4)에서의 기울기, $\alpha = 0.1$

solve) $\frac{\partial y}{\partial x_0}\Big|_{x_0=-3} = -6$ $\frac{\partial y}{\partial x_1}\Big|_{x_1=4} = 8$

$x_0 = x_0 - \alpha \cdot \frac{\partial y}{\partial x_0}$ $x_1 = x_1 - \alpha \cdot \frac{\partial y}{\partial x_1}$

(-3, 4)

$-2.4 = -3 - (0.1)(-6)$ $3.2 = 4 - (0.1)(8)$

(-2.4, 3.2)

$-1.96 = -2.4 - (0.1)(-4.8)$ $2.56 = 3.2 - (0.1)(6.4)$

(-1.96, 2.56)

⋮

(≒0, ≒0)

○ w에 대한 이차 함수로 표현이 가능

○ $\frac{\partial y}{\partial x}$: 기울기가 가리키는 방향은 함수의 출력값을 가장 크게 늘리는 방향

○ activation function. 즉, 회귀함수 일때는 위와같은 MSE가능!

○ 특성수 2개

Loss function을 미분할때 activation function이 영향을 주는 상황

9 $h(x)$ = sigmoid 일때

$\frac{d\,cost}{dw}$ · $\frac{d\,cost}{dx}$의 차이

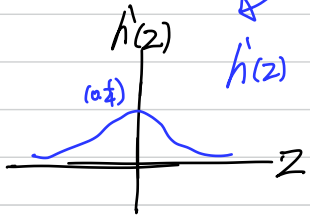
$$cost(w,b) = \frac{1}{m}\sum_{1}^{m}(y^i - t^i)^2 = \frac{1}{m}\sum_{1}^{m}(h(wx+b)-t^i)^2 \quad \text{where } h(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d\,cost(w,b)}{dw} = \frac{1}{2m}\sum_{1}^{m}(y^i - t^i)\cdot(y^i)'$$

$$= \frac{1}{2m}\sum_{1}^{m}(y^i - t^i)\cdot(h(wx+b))'$$

$$= \frac{1}{2m}\sum_{1}^{m}(y^i - t^i)\cdot(y^i)'\cdot x$$

$h'(z)$



$h'(z)$ $(z = wx+b)$

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

$h'(x) = h(x)(1-h(x))$

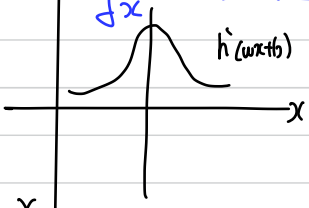
$h(wx+b) = \frac{1}{1+e^{-(wx+b)}}$

$h'(wx+b) = (h(wx+b))'\cdot(wx+b)'$

$= h(wx+b)\cdot(1-h(wx+b))\cdot x$

$h = \frac{1}{1+e^{-(wx+b)}}$

$\frac{d\,h(wx+b)}{dx} = h\cdot(1-h)\cdot w$



$\left(\frac{1}{1+e^{-3x}}\right)\left(1-\frac{1}{1+e^{-3x}}\right)\cdot 3$

$= \frac{1}{2}\cdot\left(1-\frac{1}{2}\right)\cdot 3$

$= \frac{3}{4}$ $(x=0)$

$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}$ → 다 엮이기 때문에

작은 미분계수 때문에 weight, bias가 제대로 업데이트 되지 않는다.

Z가 너무 크거나 너무 작으면 $h'(z)$의 기울기 값이 0에 가까워진다.

⇒ 작은 미분계수 때문에 weight, bias가 제대로 업데이트 되지 않는다.

⇒ 변화율은 많지만 변화율이 너무 적어 개선하기에 학습량이 길어지게 된다.

$$\frac{d}{dw_{ij}}cost(w,b;x,y) = \frac{1}{n}\sum_{x}(h(z)-t^i)\,h'(z)\cdot x_j$$

$$\frac{d}{db_j}cost(w,b;x,y) = \frac{1}{n}\sum_{x}(h(z)-t^i)\,h'(z)$$

⇒ 따라서 Sigmoid는 Cross Entropy를 사용하게된다.

global minimum이 있는데 local minimum에서 멈출수있는 MSE의 한계를 위해 Cross entropy를 사용!

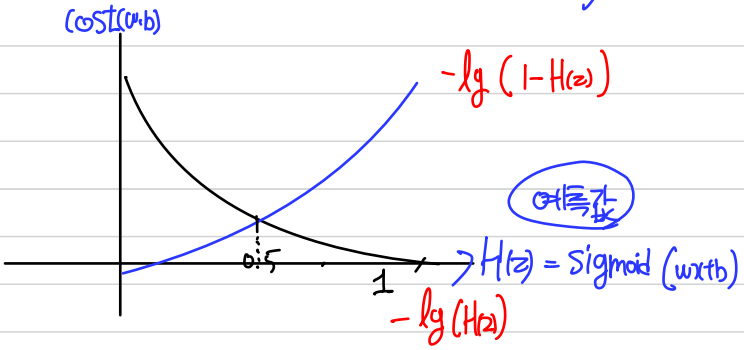
+ 다른 optimizer를 사용하면 그래도 안되는가? MSE의 한계가 아니라?

Cross Entropy

→ Sigmoid 와 같은 activation function 에 쓰는 Loss function

binary cross Entropy

- 출력값 0~1 ⟷ 실제값 0 or 1 (F. T) ⇒ log 함수가 적절
- 차이가 작은건 작게, 큰거는 크게



- 실제값이 1일때 예측값 H(z)=1 이면 오차는 0 $-\lg(H(z))$
- 실제값이 0 일때 예측값 H(z)=1 이면 오차는 ∞ $-\lg(1-H(z))$

하나로 통합하면

$$cost(H(z),y) = -y\cdot\lg(H(z)) - (1-y)\cdot\lg(1-H(z))$$

$$= -\frac{1}{n}\sum_{i}^{n}\left(y^{i}\cdot\lg(H(z^{i})) + (1-y^{i})\cdot\lg(1-H(z^{i}))\right)$$

if) 실제값이 1 ⇒ y=1 $cost(H(z),y) = -\lg(H(z))$

〃 0 ⇒ y=0 $cost(H(z),y) = -\lg(1-H(z))$

즉 실제값 y, 예측값 H(z)의 차이가 커지면 cost가 커지고.

〃 〃 차이가 작으면 cost가 낮아진다.

일반화를 한다면?

Categorical Cross Entropy

$$cost(H(z),y) = -\left(\underbrace{y_1}_{y_1}\cdot\lg\left(\underbrace{H(z)}_{P_1}\right) - \underbrace{(1-y)}_{y_2}\cdot\lg\underbrace{(1-H(z))}_{P_2}\right)$$

(P_1 = (one-hot 에서의 예측값))

$$= -\left(y_1\cdot\lg(P_1) + y_2\cdot\lg(P_2)\right)$$

$$= -\sum_{i}^{k} y_i\cdot\lg(P_i)$$

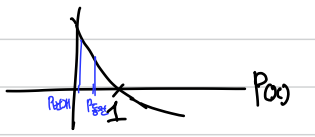
(k: softmax의 Class의 갯수; y_i: 실제값; P_i: 예측값)

$$\therefore cost(H(z),y) = -\frac{1}{n}\sum_{i}^{n}\sum_{j}^{k} y_j^{(i)}\cdot\lg(P_j^{(i)})$$

---

entropy = 정보량의 기댓값

$\log\left(\frac{1}{P(x)}\right) = -\lg(P(x))$



확률이 낮을수록 높은 정보량을 갖음

ex) $P(\text{동전}) = \frac{1}{2}$ vs $P(\text{번개}) = \frac{1}{\text{증명}}$

다시, 정보의 기댓값 (entropy)는

어떤 랜덤변수 A에 $i=1,2,\cdots n$ 의 요소가 들어있을때

$$-\sum_{i}^{n} A_i\cdot\lg(P(A_i))$$

그럼 Cross entropy는

다른 사건의 확률로 곱해서 entropy계산

= 두 개의 확률분포 P와 Q에대한 하나의 사건 X가 갖는 정보량으로 정의

= Q에 대한 정보량들 P에대해 평균

(왼쪽의 0.1 단계별 확률분포)

$$H_{P,Q} = -\sum_{i=1}^{N} P(x_i)\cdot\lg q(x_i)$$

$$H_{P,Q}(Y|X) = -\sum_{i}^{N}\sum_{y\in(0,1)} P(y_i|x_i)\cdot\lg q(y_i|x_i)$$

$$= -\sum_{i}^{N}\left(P(y_i|x_i)\cdot\lg q(y_i|x_i) + P(y_i|x_i)\cdot\lg(1-q(y_i|x_i))\right)$$

$$= -\sum_{i}^{N} P(y_i|x_i)\cdot\lg q(y_i|x_i) - (1-P(y_i|x_i))\lg(1-q(y_i|x_i))$$

$$= -\sum_{i}^{N} P(y_i)\cdot\lg q(y_i) + \{1-P(y_i)\}\lg(1-q(y_i))$$

Cross entropy

$$= -Q(x=0)\cdot\lg P(x=0) - Q(x=1)\cdot\lg P(x=1)$$