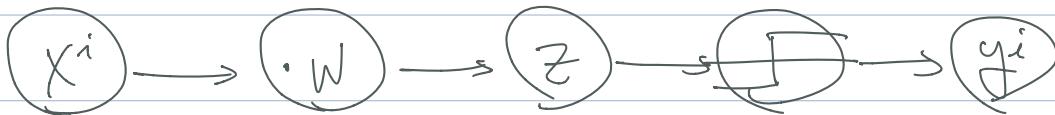


2. (인접 흐른 (parceptron) 학습)



W 를 0에 가까운 랜덤 변수로 초기화를 하고 학습을 한다면

$(W_i^{init}) \leftarrow 1$

Case 1. 본래가 정확하기로 된 때 ($y^i = \hat{y}^i$)

$$\Delta W = \eta (y^i - \hat{y}^i) x^i = 0 : \text{weight 업데이트 없음}$$

Case 2. 양의 \hat{y}^i 가 잘못 분류 됐을 때 ($y^i=1, \hat{y}^i=-1$)

$$\Delta W = \eta (y_i - \hat{y}_j^i) x^i = \eta_2 x^i$$

Case 3. 음의 셉풀이 잘못 분류되었을 때 ($y^i = -1$, $\hat{y}^i = 1$)

$$\Delta W = \eta(y^i - \hat{y}^i) x^i = -\eta x^i$$

학습 샘플들이 다음과 같이 구성된다면

$$X = [\underbrace{x^1, x^2, \dots, x^P}_{\text{positive: } y_i=1}, \underbrace{x^{P+1}, \dots, x^N}_{\text{negative: } y_i=-1}]$$

1 epoch 학습시 W 는 다음과 같이 학습된다.

$$w = w^{init} + \eta (c^1 x^1 + c^2 x^2 + \dots + c^p x^p - c^{p+1} x^{p+1} - c^{p+2} x^{p+2} - \dots - c^n x^n)$$

$$= w^{init} + \eta \sum_{i=1}^p c^i x^i - \eta \sum_{i=p+1}^n c^i x^i$$

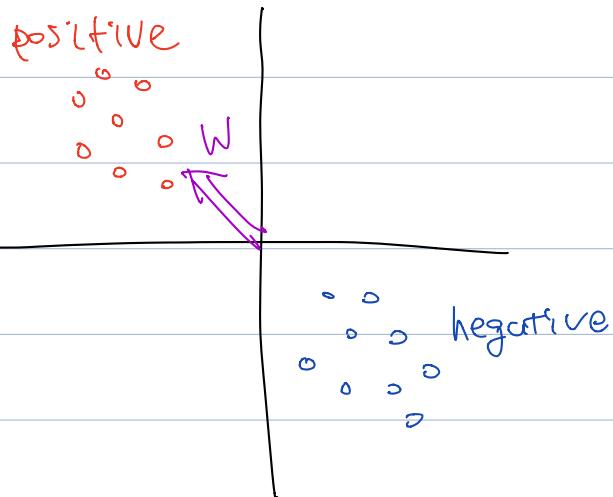
$$C^i = \begin{cases} 1 & : i\text{ 번째 샘플이 잘못 분류된 경우} \\ 0 & : \quad \quad \quad \text{“} \quad \quad \quad \text{바르게 분류된 경우} \end{cases}$$

즉 잘못 분류된 샘플 중 양의 샘플을 더하고 음의 샘플은 뺀 것이다.

$$\text{설명 } w_0 = w_0^{\text{init}} + \eta \sum_{i=1}^p c_i - \eta \sum_{i=p+1}^n c_i$$

$$= \eta (\text{잘못 분류된 양의 샘플 } A - \text{잘못 분류된 음의 샘플 } B)$$

이게 분류에 어떻게 도움이 되까? 2차원 예시로 이해



두 가지 샘플들이 다음과 같이 배치된

상황에서 모든 $C^i = 1$ 이라면

$$W = W^{init} + \eta \sum_{i=1}^p x^i - \eta \sum_{i=p+1}^n x^i$$

= 초기값 + 양의 샘플 방향 - 음의 샘플 방향

\simeq 음의 샘플에서 양의 샘플로 가는 방향

그럼 여기서 W 가 단순히 $(-1, 1)$ 이라면 net input z^i 는

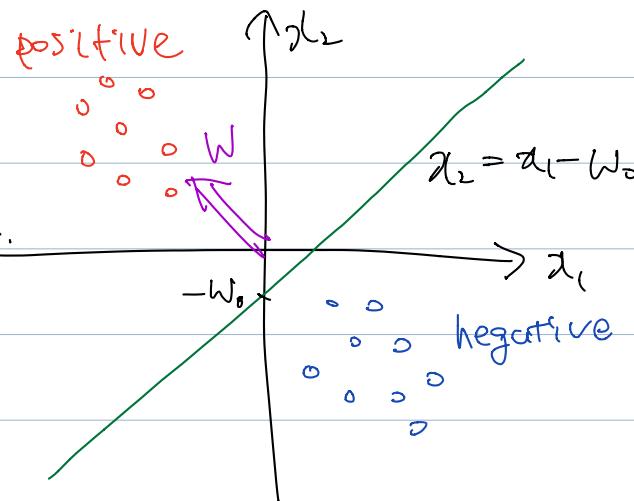
$$z^i = w_1 x_1^i + w_2 x_2^i + w_0 = -x_1^i + x_2^i + w_0.$$

$$\text{축복이 } 1 \text{ 이 높아지 위해서는 } z^i = -x_1^i + x_2^i + w_0 \geq 0$$

$$x_2 \geq x_1 - w_0.$$

결정 경계선을 그리면 \rightarrow

여기 W 와 손잡인 직선이 그려졌다.



W 는 음의 샘플에서 양의 샘플로 가는 방향으로 학습도모로

결정 경계가 둘 사이를 가로지르는 직선이 된다.



이렇게 W 에 대한 방향은 맞는데 전편이

안 맞는 경우에는 전편 업데이트 시에 의해

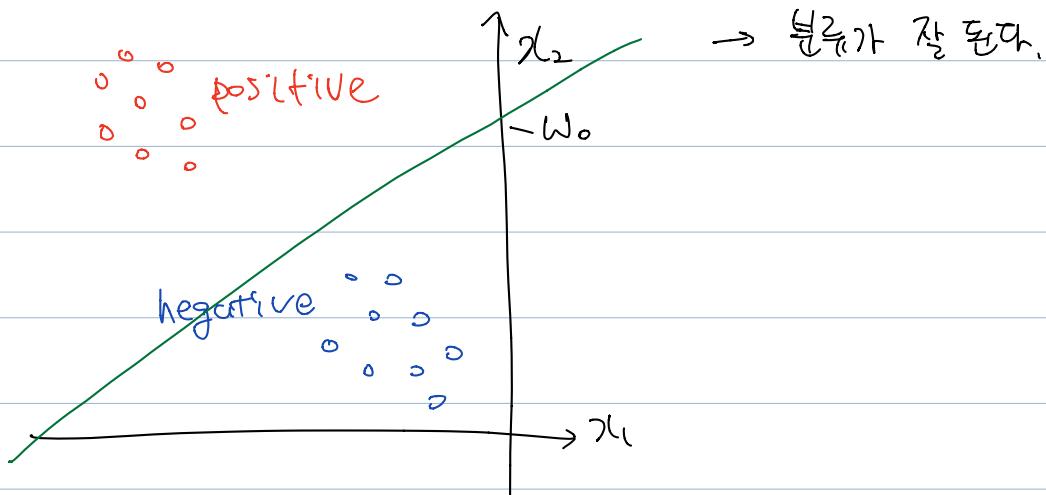
$$w_0 := w_0 + \eta \sum_{i=1}^p C^i - \eta \sum_{i=p+1}^n C^i$$

전편 $-w_0$ 가 정정이 된다.

여기서는 음의 샘플들만 잘 봤을 때로

$$w_0 := w_0 - \eta \sum_{i=1}^n c^i \rightarrow \text{음수로 값이 커진다.}$$

$\rightarrow -w_0$ 가 양으로 커진다 \rightarrow 절편이 올라간다.



2.3.1 경사하강법 (Gradient Descent)

입력 x, y 가 있을 때 입력을 통해 출력을 예측하는 모델 함수를 다음과 같이

$$\text{표현한다면: } \hat{y} = \phi(w, x)$$

w 는 모델의 파라미터로서 머신러닝에서 학습 대상이 된다.

학습의 목적은 타겟 y 를 정확하게 예측하도록 모델 파라미터 w 를 조정하는 것

머신러닝에서 이러한 문제를 해결하는 전형적인 과정은 다음과 같다.

① 목적함수 정의

목적함수 (objective function, 비용함수, cost function)는

모델의 성능을 하나의 수치로 표현할 수 있는 함수다.

즉, 학습에서 w 를 조절해서 줄여야 할 대상이 된다.

학습의 목적은 다음과 같은 \hat{J} 을 찾는 것이다.

$$\hat{J} = \arg \min_w J(w)$$

목적함수는 전형적으로 아래 형태가 많이 쓰인다.

$$J(w) = \frac{1}{2} \sum_i (y^i - \phi(w, x^i))^2$$

그 이유는 제곱을 하면서 오차의 부호가 아닌 크기만 영향을 많이 끌으며 모든 샘플의 오차를 더하여 스칼라가 되고, 제곱의 특성상 오차가 큰 샘플에 대해 더 민감하게 반응하여 학습시 모든 샘플의 오차를 평준화하려는 경향이 있기 때문이다.

② 최적화 (Optimization)

목적함수가 정의되고 손실(파라미터 w)을 고려해 목적함수를 최소화 혹은 최대화하는 것을 최적화(Optimization)라 한다. 최적화는 기법과 종류가 매우 다양하는데 크게 두 가지 분류가 있다.

a. global optimization : 미분을 통해 전역 극소값을 바로 계산하는 방식이다.

보통 1차 미분을 0으로 만드는 점을 찾는다: $\frac{\partial J(w)}{\partial w} = 0$

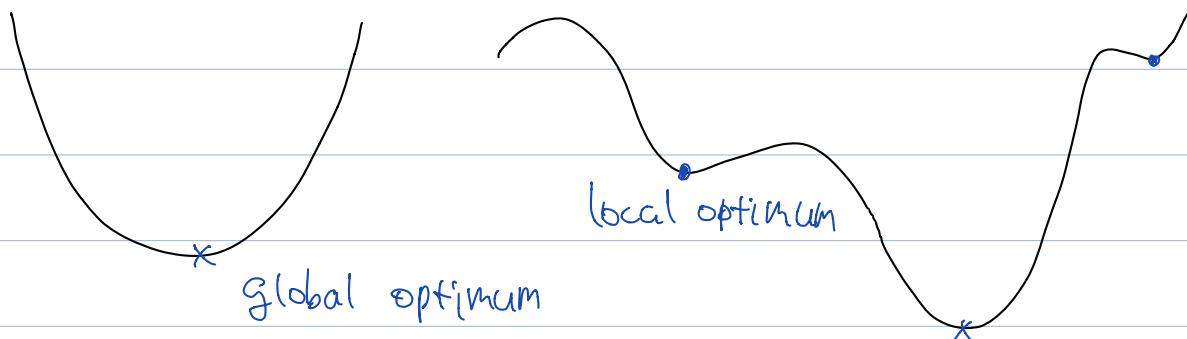
그러나 이러한 방식을 적용하려면 극소값의 개수가 유한해야하고 모두 계산할 수 있어야 한다. (해를 구할 수 없는 함수 예: $-\log x \cdot \cos x = 1$)

b. local optimization : global optimization을 할 수 없는 문제들은 일정한

초기값으로부터 출발하여 미분경사법을 따라 반복적으로 이동하면서 찾아야

한다. 마치 눈을 감고 벽을 더듬어 오목한 점을 찾는 것과 같다.

local optimization은 전역 극소값을 찾는다는 보장은 없고 단지 값을 계산해서 "경험한" 것들 중 최소값을 찾는 방식이다. local optimization 기법은 여러가지가 있는데 Gradient Descent, Newton-Rapson method, Levenberg-Marquardt method 등이 대표적이다.



경사하강법에서는 다음과 같이 파라미터를 업데이트 한다.

$$\omega := \omega - \eta \frac{\partial J(\omega)}{\partial \omega}$$

여기서 ω 는 여러 파라미터가 들어있는 벡터다. $J(\omega)$ 는 스칼라다.

스칼라를 벡터로 미분하면 무엇이 나을까? 답은 벡터다.

$$\frac{\partial J(\omega)}{\partial \omega} = \begin{bmatrix} \frac{\partial J(\omega)}{\partial \omega_1} \\ \frac{\partial J(\omega)}{\partial \omega_2} \\ \vdots \\ \frac{\partial J(\omega)}{\partial \omega_m} \end{bmatrix} : m \times 1 \text{ 벡터}$$

$J(\omega)$ 를 ω 의 각각의 원소로 미분하여 합친 것이다.
이렇게 벡터로 미분한 것을 Jacobian이라고 한다. (자전ベアン)

즉 업데이트 식은 아래와 같다.

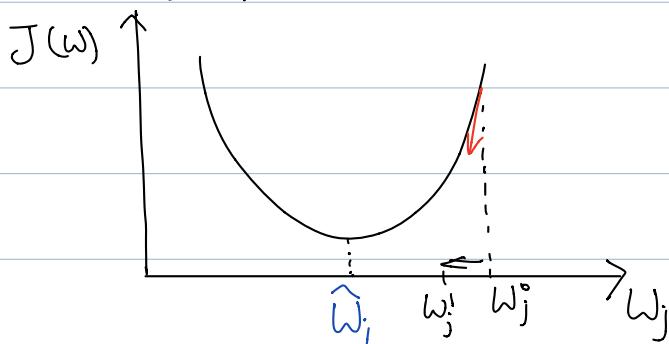
$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_m \end{bmatrix} := \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_m \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial J(\omega)}{\partial \omega_1} \\ \frac{\partial J(\omega)}{\partial \omega_2} \\ \vdots \\ \frac{\partial J(\omega)}{\partial \omega_m} \end{bmatrix}$$

저 식을 통해 어떤지 목적함수가 줄어들 수 있는지 그림을 통해 알아보자.

책에서 $\phi(\omega, x^i) = \omega^\top x^i$ 이기 때문에

$$J(\omega) = \frac{1}{2} \sum_i (y^i - \omega^\top x^i)^2 \text{이다.}$$

즉 $J(\omega)$ 는 ω 에 대해서 제곱함수이고 그раф로 나타내면 다음과 같다.



ω_j^* 는 파라미터의 초기값이다.

해상포는 초기의 미분역방향이다.

$$\leftarrow : -\eta \frac{\partial J(\omega)}{\partial \omega_j} \Big|_{\omega_j=\omega_j^*}$$

$$= -\eta \times \omega_j^* \text{에서의 기울기}$$

시작점에서의 기울기는 양수이므로 ($\frac{\partial J(w)}{\partial w_j} > 0$) 업데이트는 음의 방향으로 이루어진다.

$$w_j' = w_j - \gamma \frac{\partial J(w)}{\partial w_j}$$

그러면 w_j' 는 최적의 \hat{w}_j 에 더 가까워지고 업데이트 된 w_j' 으로 다시 $J(w)$ 를 계산하면 w_j' 로 계산한 것 보다 작은 값이 나온다.

이 과정을 반복하면 결국 최적의 \hat{w}_j 에 깊이 가까운 w_j 를 얻을 수 있다.

제곱합수에서 기울기는 극소점에 가까워질 때까지 작아지므로

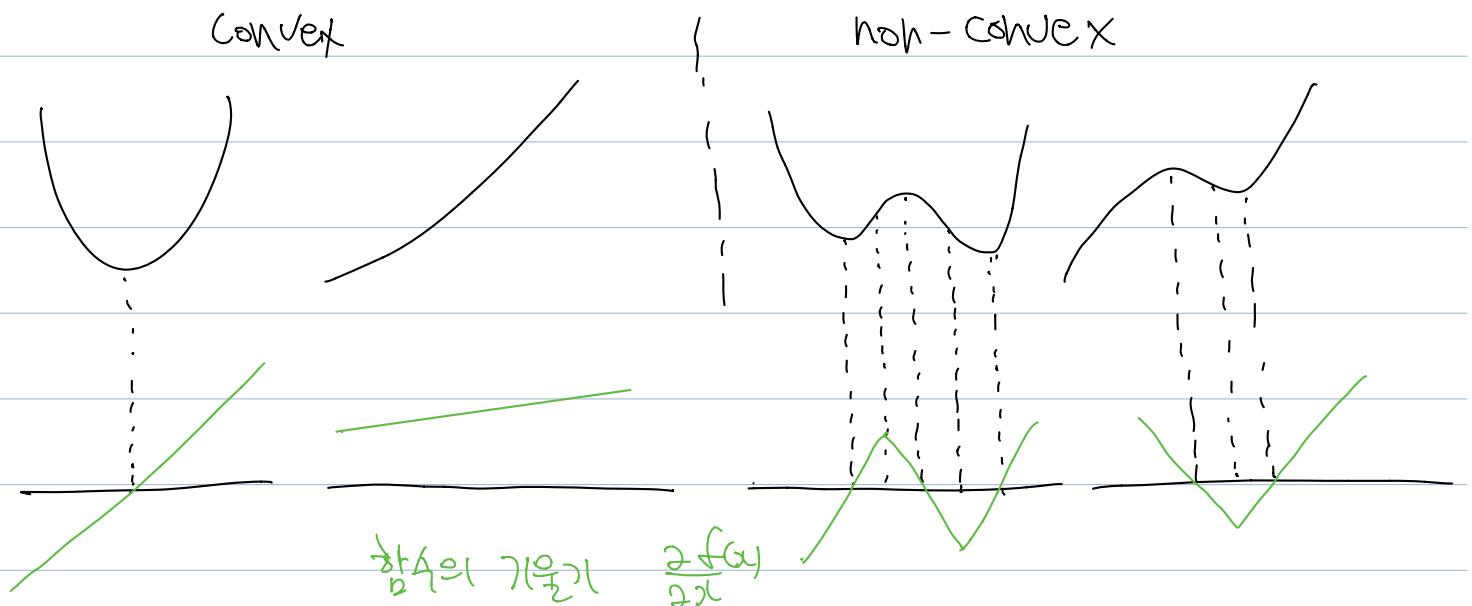
학습이 반복될 수록 업데이트 양이 줄어들지만 그로 인해 최적값에 더 정밀하게 접근할 수 있다.

X: Convex function

$J(w)$ 는 제곱합수이고 제곱합수는 "Convex"하기 때문에 경사하강법을 적용해 보면 이론적인 극소값에 도달할 수 있다.

Convex 하다는 것은 말 그대로 오목히 들이가 있다는 것인데 그림을 통해

Convex와 non-convex를 쉽게 구별할 수 있다.



Convex함수는 기울기가 항상 증가하고 있는 함수다.

왼쪽 그림과 오른쪽 그림을 비교해 보면 왼쪽은 항상 기울기가 증가하고 있는데 오른쪽은 기울기가 감소하는 영역이 있다.

Convex 함수의 조건을 수학적으로 정의하면 다음과 같다.

$$\frac{\partial^2 f(x)}{\partial x^2} > 0 \quad (\text{기울기의 변화율은 2차 미분})$$

목적함수가

- Convex 함수인 경우 경사하강법을 통해 global optimum에 도달할 수 있다.
- non-convex 함수의 경우 경사하강법으로는 local optimum까지만 갈 수 있다.

