예측 모델의 기본 원리

오차지표 정리

Mean Squared Error(MSE)

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

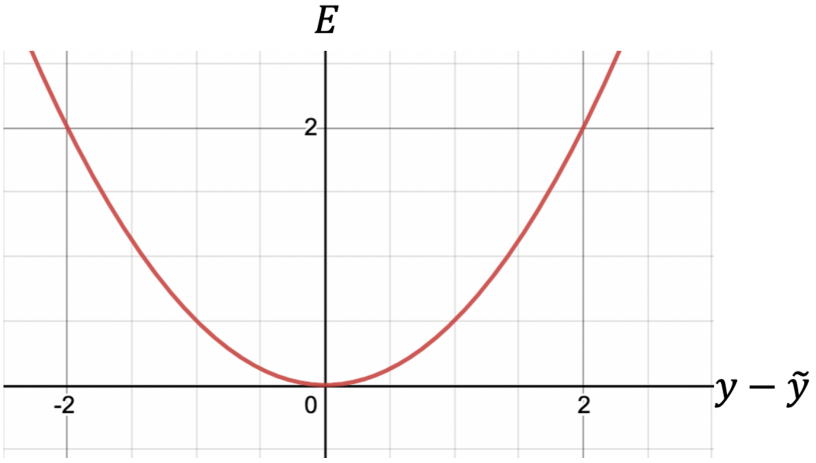
오차를 제곱한 값의 평균 오차가 커질수록 손실 함수 값이 빠르게 증가한다.

⇒ 모델이 오차에 민감하게 반응한다

회귀문제에 주로 사용함.

루트를 씌우면 RMSE(Root Mean Squared Error)

큰 값이 계산 전체에 지나친 영향을 미치지 않게 제어하는 장점이 있음



다음은 MSE를 좌표평면으로 나타낸 그림 손실함수(E)의 크기가 오차의 제곱에 비례하여 커짐

Mean Absolute Error(MAE)

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

- 직관적이고 예측변수와 같은 단위를 사용한다(제곱하지 않고 단순 빼주기 때문에)
- 절대값을 취하기 때문에 실제보다 낮은 값인지 큰 값인지 알 수 없다
- MSE보다 이상치에 덜 민감하다

예측 오차의 크기를 실제 값의 단위와 동일하게 유지하기 때문에 **직관적으로 크기를 이해하고 싶을 때 유용**

언제 사용하면 좋을까?

- 예측 값의 해석이 중요한 경우 ex) 등록차량 수 예측
- 아웃라이어가 적은 데이터(잘 정제된 데이터)

Mean Absolute percentage error(MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

- 범위가 정해짐(0~100%)
- 실제 값이 0이거나 작으면 MAPE 값이 지나치게 커짐
- 단위가 없기에 다양한 데이터 셋간 비교가 쉬움
- 극단적인 값에 민감함

경사 하강법

loss function의 값을 **최소화**하여 최적의 **parameter**를 찾는 것

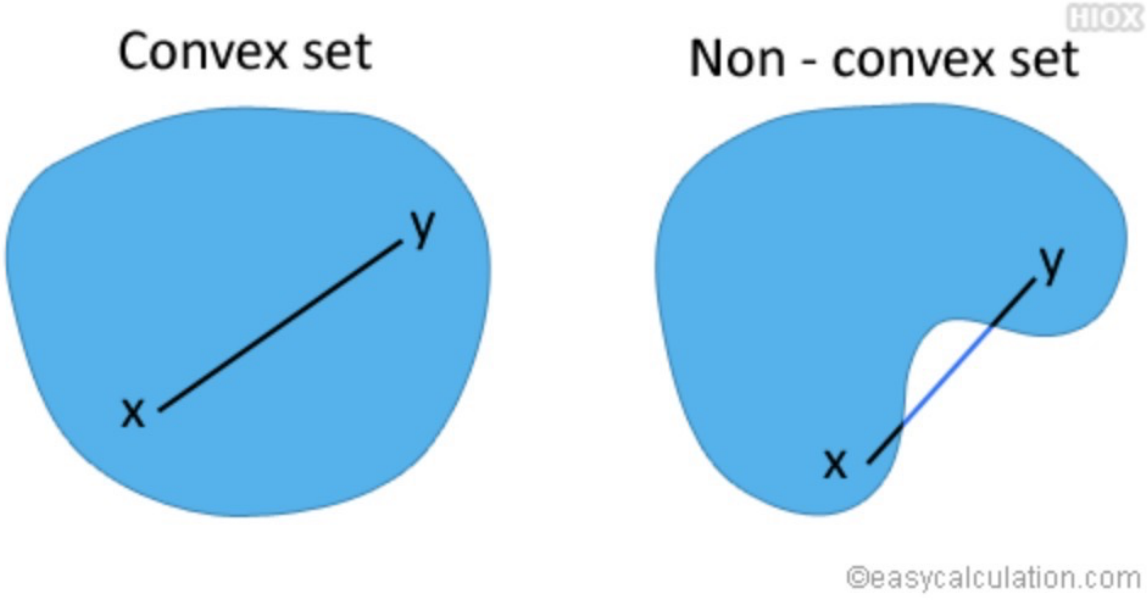
만약 loss function의 전 구간이 convex하다면 어떤 초기값에서 시작하더라도

최적화 알고리즘을 통해 수렴하는 local optimal은 global optimal이 된다

convex function(볼록함수)

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \quad \text{for all } x, y \text{ and } \lambda \in [0,1]$$

함수가 정의된 영역 내에서 두 점을 연결하는 선이 함수 그래프 위에 항상 위치한다면 그 함수는 convex하다고 수학적으로 정의한다.



convex optimazation

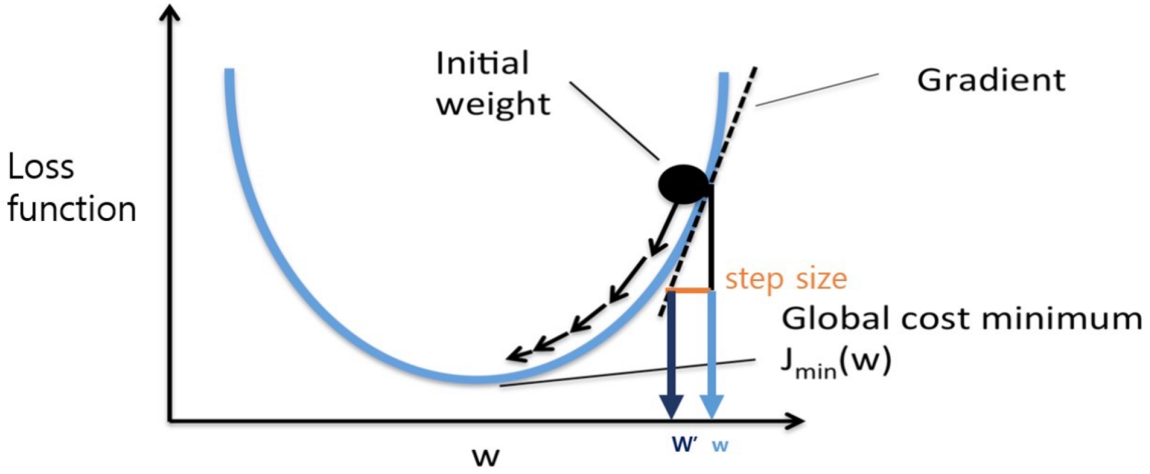
$$\min_{x \in \mathbb{R}^n} \ f(x)$$
$$\text{subject to}$$
$$g_i(x) \leq 0, \ i = 1, \ldots, m$$
$$h_j(x) = 0, \ j = 1, \ldots, p$$

$f(x)$: 목적함수이며 볼록함수

$g_i(x)$: 제약조건이며 볼록함수

$h_j(x)$: 등식 제약 조건이며 선형 함수

아쉽게도 대부분의 식은 convex하지 않다. 그렇다고 파라미터들의 최적 값을 구하기 위해 편미분을 때려버린다면 연산량이 계산 불가능에 가까워진다.(신경망의 경우 모델에 따라 파라미터가 수천, 수억 개...) 그렇기 때문에 차선의 방책으로 그래프 상의 점을 조금씩 움직여 함수의 최솟 값을 찾는다.



$$(\Delta x_1, \Delta x_2, \ldots, \Delta x_n) = -\eta \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

미분 가능한 함수 $f(x_1, x_2, ..., x_n)$에서 변수를 차례대로 $x_1 + \Delta x_1, x_2 + \Delta x_2, ..., x_n + \Delta x_n$

작은 값으로 변화 해준다. 이 때 이동시킬 때 얼만큼 이동할지 정하는게 $-\eta$이다.

💡 손실함수는 인공지능 모델의 파라미터를 통하여 나온 예측 값과 실제 값의 차이
손실 함수 값을 가장 낮게 나오게 하는 파라미터가 바로 최적의 파라미터
문제는 우리가 마주하는 함수들은
복잡하고 비선형(계산이 힘듬)
그래서 경사하강법을 이용해서 최소값을 구함

문제점

적절한 Learning rate 찾기

학습률과 기울기 정보를 혼합하여 방향을 결정한다

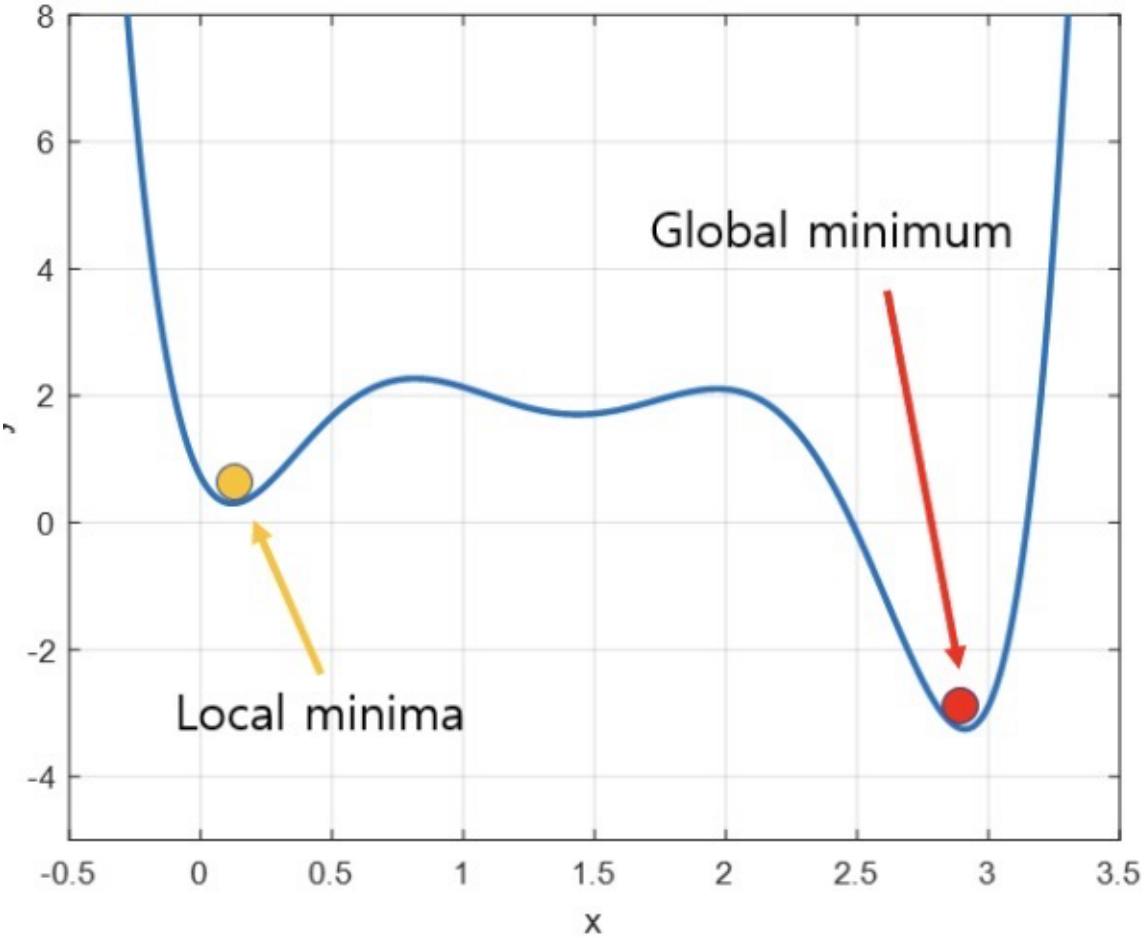
학습률이 높으면 더 많이 나아간다 = 운이 좋으면 최적 값에 빨리 수렴할 수 있다

그러나 다른 곳에 발산하는 가능성이 커진다. 그렇다고 학습률이 낮으면? 시간이 오래 걸린다.

⇒ scheduling 기법 : 학습 시간 혹은 검증 세트의 손실 값에 따라 학습률 조절

Local Minimum 문제

최초의 x위치에 따라 해가 local이나 global이냐 갈리게 되며, 또 적당히 좋은 위치에 x를 위치하더라도 학습률에 따라 지나칠 가능성이 커진다.



이러한 문제를 위해 다양한 경사 하강법이 나왔다

- adaptive GD
- momentum GD

▼ 참고

https://taek-guen.tistory.com/38

https://www.youtube.com/watch?v=E-INcQ4rRVk&list=PL_iJu012NOxeMJ5TPPW1JZKec7rhjKXUy

https://convex-optimization-for-all.github.io/

https://www.youtube.com/watch?v=IHZwWFHWa-w&t=595s

로지스틱 회귀

회귀 문제는 보통 연속형 분류문제를 다루지만 로지스틱 회귀는 참 거짓과 같은 범주형 분류 문제를 해결하기 위해 사용 된다. 선형 회귀문제와 마찬가지로 독립 변수와 종속 변수 간의

관계를 모델링하는데 독립 변수의 결합을 로지스틱 함수(시그모이드 함수)를 통해 변환하여 종속 변수의 확률을 예측한다.

시그모이드 함수

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

이때 z는 $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ 독립 변수들의 선형 결합이다.

출력 값은 시그모이드 함수의 특성으로 인해 0과 1사이의 값으로 변환된 확률 값이며 특정 임계값을 넘으면 클래스 1로 분류하고 아니면 클래스 0으로 분류한다.

손실함수

log-loss와 cross entropy를 사용한다

log-loss

$$\text{logloss}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{if } y = 1 \\ -\log(1 - h_\theta(x)), & \text{if } y = 0 \end{cases}$$

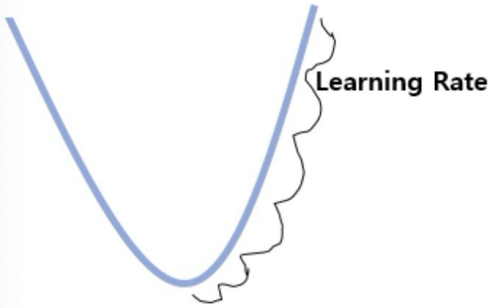
근데 if문을 고려해서 코딩하면 복잡해지니 한줄로 표현해서

$$\text{logloss}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

왜 최소 제곱법 안쓰는가?

$$Cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^i) - (y^i))^2$$

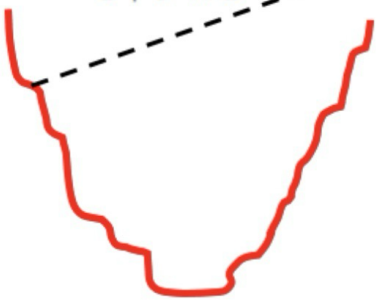
$H(X) = Wx + b$



Linear Regression

$H(X) = \frac{1}{1 + e^{-Wx}}$

평평한 부분을 가장 오차가 적은 부분으로 생각할 수 있음



Logistic Regression