3-2 경사하강법

회귀 문제를 풀기 위해 경사 하강법 외에도 많은 알고리즘 존재

산점도를 잘 나타내는 직선의 방정식을 경사하강법을 이용해 찾기

미리 준비되어 있는 정답 ← 타깃 → $y = ax + b$ ← 기울기, 절편

모델을 이용해 계산한 타깃 ← 예측값 → $\hat{y} = wx + b$ ← 가중치

∴ 준비된 입력 data (x) 와 타깃 data (y) 를 이용해 최적의 w와 b 찾기

< 훈련 데이터에 맞는 w와 b 찾는 방법 >

① 무작위 선정

② 무작위로 모델 예측 ($\hat{y}$ 계산)

③ $\hat{y}$과 y 비교 → 당연히 틀림

④ w, b 조정

→ w의 변화율 (w-rate)은 샘플값 자체

→ b의 변화율은 항상 1

② 번복

1) w값 조정하기
- w-new = w + 0.1
- w-new = w + w-rate

2) b값 조정하기
- b-inc = b + 0.1
- b-new = b + 1 (1 = b-rate)

but 이 방식은 w와 b를 큰 폭으로 수정불가 , $\hat{y}$을 감소시킬 수 x

오차 역전파로 가중치와 절편 적절하게 업데이트

$\hat{y}$이 증가 or 감소 상황에 따라 변경가능 (부호)

오차와 변화율을 곱하여 가중치 업데이트 → 오차의 크기에 따라 많은 폭으로 변경 가능

w_new = w + w_rate x err

err → y[0] - y_hat

b_new = b + 1 x err

↓

모든 샘플 이용해 w와 b 업데이트 하기 (1epoch 이용)

↓

더 좋은 모델을 찾기 위해 여러 에포크 반복 (이중 루프)

↓

이렇게 찾은 모델을 이용해 새로운 data 예측 가능

수학적으로 경사하강법 유도

3-3 손실함수와 경사 하강법의 관계

손실함수 : 예측한 값과 실제 타깃 값의 차이 측정

↳ 줄이는 방법으로 경사하강법 사용

↳ 직접 정의 할 수도 있지만 회귀, 분류 등에 널리 사용되는 손실함수도 존재

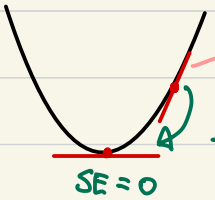
제곱오차 : 회귀에서 사용하는 대표적인 손실함수 중 하나

↳ $SE = (y - \hat{y})^2$ → 차이가 많이 나는 오차일 경우 더 크게 벌칙을 부과하는 효과

y: 타깃, $\hat{y}$: 예측 → 임의의 w로 정한 영터리 예측 → $wx+b$

↓

작을수록 가장 최적의 solution



입력 data가 양수

무조건 기울기가 양수이므로 SE가 최적인 0으로 감소하기 위해 낮아지는 쪽으로 이동해야 함 → 미분해서 기울기 알기

〈가중치〉 $\frac{\partial SE}{\partial w} = -(y-\hat{y})x$

∴ 미분 값을 가중치에서 빼면 손실함수의 낮은 쪽으로 이동

→ $w = w - \frac{\partial SE}{\partial w} = w + (y-\hat{y})x$

$= w + w\text{-rate} \times err$

〈절편〉 $\frac{\partial SE}{\partial b} = -(y-\hat{y})$

→ $b = b - \frac{\partial SE}{\partial b} = b + (y-\hat{y})$

err

3-4 선형 회귀를 위한 뉴런 만들기

Neuron 이라는 파이썬 클래스로 경사 하강법 알고리즘 구현

① __init__ 메서드 작성

② 정방향 계산 만들기 (forpass) : $\hat{y} = wx + b$ → $\hat{y}$을 계산해야 오차 계산 가능
- $\hat{y}$ 예측값 계산

③ 역방향 계산 만들기 (backprop) : $\hat{y}$을 이용해 오차 구하기 → 오차가 (b 에 적용될 때 1 곱하기, w에 적용될 때 x 곱하기)
- 그레이디언트 계산 $(-(y-\hat{y}))$
- $-(y-\hat{y})x$

④ fit 메서드 구현
1) forpass 를 호출하여 $\hat{y}$ 구하기
2) 오차 계산
3) backprop을 호출하여 그레이디언트 구하기
4) 가중치, 절편 업데이트

→ 에포크 만큼 반복하기