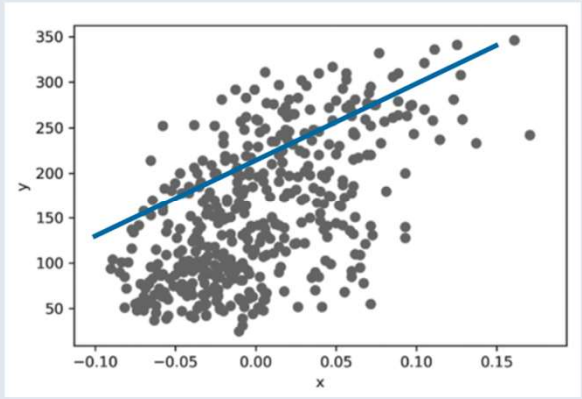
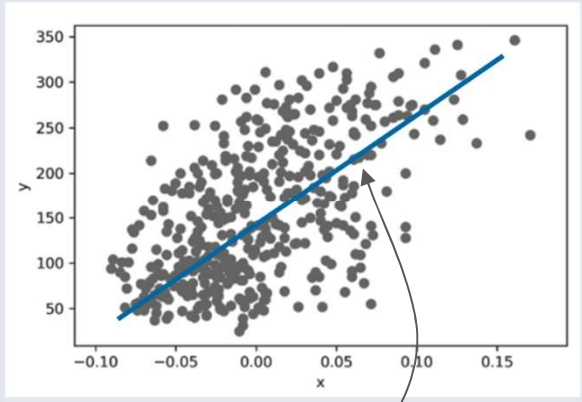
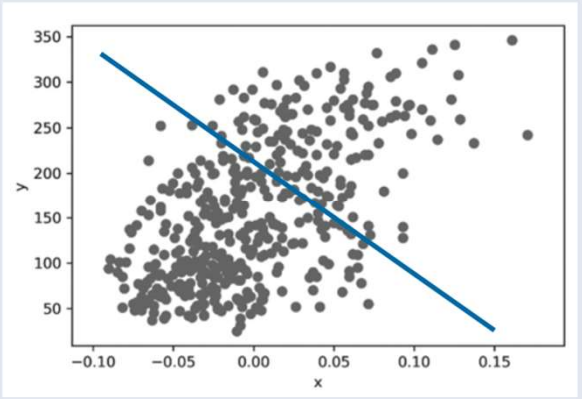
03 머신러닝의 기초를 다집니다
- 수치 예측

03-2 경사 하강법으로 학습하는 방법을 알아봅니다

어떤 직선이 가장 잘 표현하고 있나요?



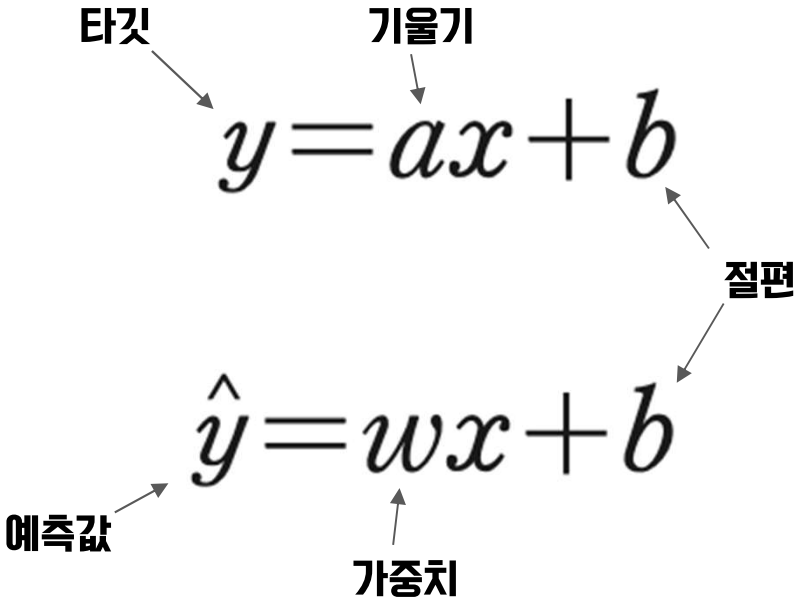




경사 하강법으로 찾기!

회귀 문제를 풀기 위해 경사 하강법 외에 다른 방법은 없나요?

타깃과 예측값



훈련 데이터에 잘 맞는 w와 b를 찾는 방법

① 무작위로 w와 b를 정합니다(무작위로 모델 만들기).

② x에서 샘플 하나를 선택하여 $\hat{y}$을 계산합니다(무작위로 모델 예측하기).

③ $\hat{y}$과 선택한 샘플의 진짜 y를 비교합니다(예측한 값과 진짜 정답 비교하기, 틀릴 확률 99%).

④ $\hat{y}$이 y와 더 가까워지도록 w, b를 조정합니다(모델 조정하기).

⑤ 모든 샘플을 처리할 때까지 다시 ②~④ 항목을 반복합니다.

실제로 훈련 데이터에 맞는 w와 b 찾아보기

```
w = 1.0
b = 1.0
```

임의의 값으로 시작

```
y_hat = x[0] * w + b
print(y_hat)
1.0616962065186886
```

첫 번째 샘플에 대한 예측 만들기

어떻게 이 차이를 좁힐 수 있을까요?

```
print(y[0])
151.0
```

첫 번째 샘플의 실제 타깃

w 값을 조절해 예측값을 바꾸어 보죠

w를 0.1만큼 증가시켜 봅니다

```
w_inc = w + 0.1
y_hat_inc = x[0] * w_inc + b
print(y_hat_inc)
1.0678658271705574
```

이전보다 증가했습니다.

```
y_hat = x[0] * w + b
print(y_hat)
1.0616962065186886
```

타깃에 조금 더 가까워졌습니다

```
print(y[0])
151.0
```

w를 0.1만큼 증가시킨 것은 올바른 결정이었네요!

그럼 얼마만큼 증가했나요?

```
w_rate = (y_hat_inc - y_hat) / (w_inc - w)
print(w_rate)
0.061696206518688734 ←——— 변화율
```

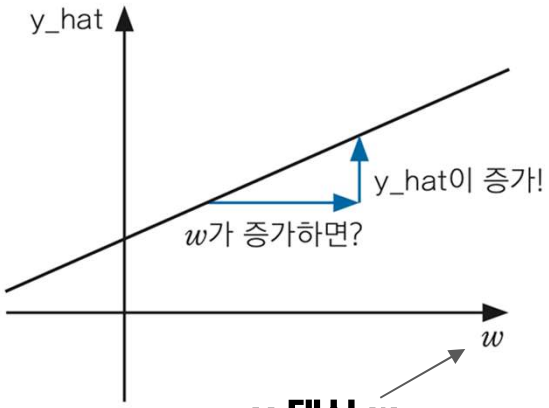
$$w\_rate = \frac{y\_hat\_inc - y\_hat}{w\_inc - w} = \frac{(x[0] * w\_inc + b) - (x[0] * w + b)}{w\_inc - w}$$

$$= \frac{(x[0] * (w + 0.1) - w)}{(w + 0.1) - w} = x[0]$$ ←——— 변화율은 x[0] 자체입니다

변화율을 보고 w를 어떻게 바꾸어야 할지 알 수 있나요?

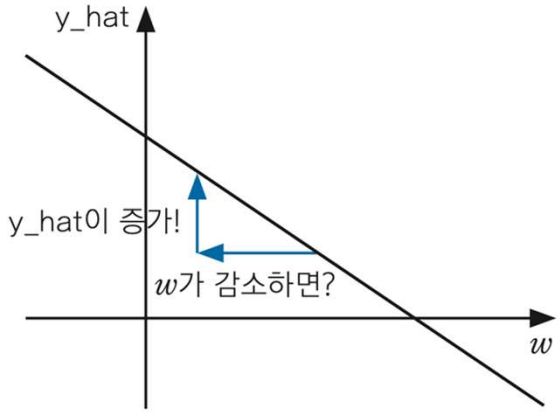
변화율 부호에 따라 가중치를 업데이트하는 방법

변화율이 양수일 때



x 대신 w

변화율이 음수일 때




```
w_new = w + w_rate
print(w_new)
1.0616962065186888
```

변화율을 더하면 예측값이 증가!

변화율로 절편 업데이트하기

```
b_inc = b + 0.1
y_hat_inc = x[0] * w + b_inc
print(y_hat_inc)
1.1616962065186887

b_rate = (y_hat_inc - y_hat) / (b_inc - b)
print(b_rate)
1.0
```

$$b\_rate = \frac{y\_hat\_inc - y\_hat}{b\_inc - b} = \frac{(x[0] * w + b\_inc) - (x[0] * w + b)}{b\_inc - b}$$

$$= \frac{(b + 0.1) - b}{(b + 0.1) - b} = 1$$

```
b_new = b + 1
print(b_new)
2.0
```

이 방식에 어떤 문제점이 있을까요?

- y_hat이 y에 한참 미치지 못 하는 값인 경우, w와 b를 더 큰 폭으로 수정할 수 없습니다(앞에서 변화율 만큼 수정을 했지만 특별한 기준을 정하기가 어렵습니다).
- y_hat이 y보다 커지면 y_hat을 감소시키지 못 합니다.

y_hat 과 y 의 차이가 크면 w와 b를 그에 비례해서 바꿔야 됩니다. --> 빠르게 솔루션에 수렴

y_hat이 y 보다 크면 w와 b를 감소시켜야 합니다. --> y_hat과 y 값에 능동적으로 대처

오차 역전파로 가중치와 절편을 업데이트합니다

오차와 변화율을 곱하여 가중치를 업데이트합니다

```
err = y[0] - y_hat
w_new = w + w_rate * err
b_new = b + 1 * err
print(w_new, b_new)
10.250624555904514 150.9383037934813
```

가중치와 절편이 큰 폭으로 바뀌었습니다

```
w_new = w + w_rate
print(w_new)
1.0616962065186888
```

```
b_new = b + 1
print(b_new)
2.0
```

두 번째 샘플을 사용하여 w와 b를 계산합니다

```
y_hat = x[1] * w_new + b_new
err = y[1] - y_hat
w_rate = x[1]
w_new = w_new + w_rate * err
b_new = b_new + 1 * err
print(w_new, b_new)
14.132317616381767 75.52764127612664
```

두 번째 샘플의 변화율은 샘플 값 그 자체입니다

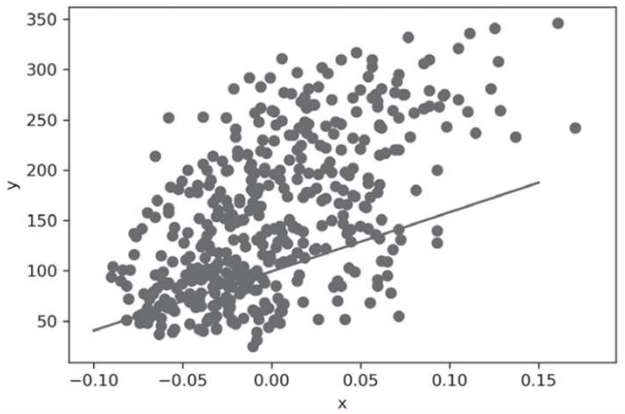
이런 식으로 모든 샘플에 대해 처리해 볼까요?

전체 샘플을 반복하여 가중치와 절편을 조정하기

```
for x_i, y_i in zip(x, y):
    y_hat = x_i * w + b
    err = y_i - y_hat
    w_rate = x_i
    w = w + w_rate * err
    b = b + 1 * err
print(w, b)
```

```
587.8654539985689 99.40935564531424
```

```
plt.scatter(x, y)
pt1 = (-0.1, -0.1 * w + b)
pt2 = (0.15, 0.15 * w + b)
plt.plot([pt1[0], pt2[0]], [pt1[1], pt2[1]])
plt.xlabel('x')
plt.ylabel('y')
plt.show( )
```



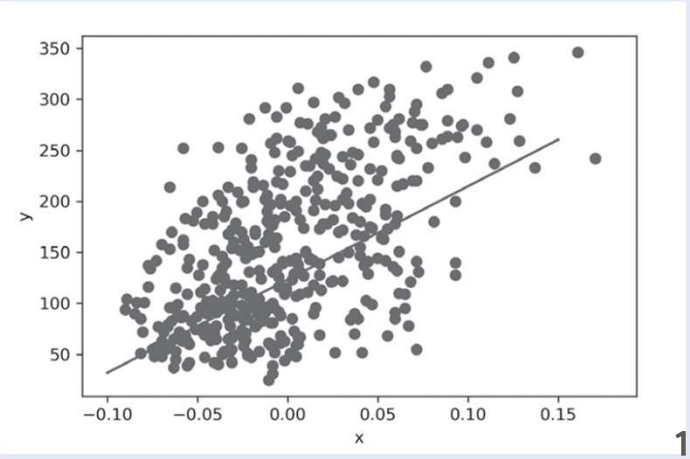
여러 에포크를 반복하기

```
for i in range(1, 100):
    for x_i, y_i in zip(x, y):
        y_hat = x_i * w + b
        err = y_i - y_hat
        w_rate = x_i
        w = w + w_rate * err
        b = b + 1 * err
print(w, b)
913.5973364345905 123.39414383177204
```

경사 하강법으로 찾은 선형 회귀 모델

$$\hat{y} = 913.6x + 123.4$$

```
plt.scatter(x, y)
pt1 = (-0.1, -0.1 * w + b)
pt2 = (0.15, 0.15 * w + b)
plt.plot([pt1[0], pt2[0]], [pt1[1], pt2[1]])
plt.xlabel('x')
plt.ylabel('y')
plt.show( )
```

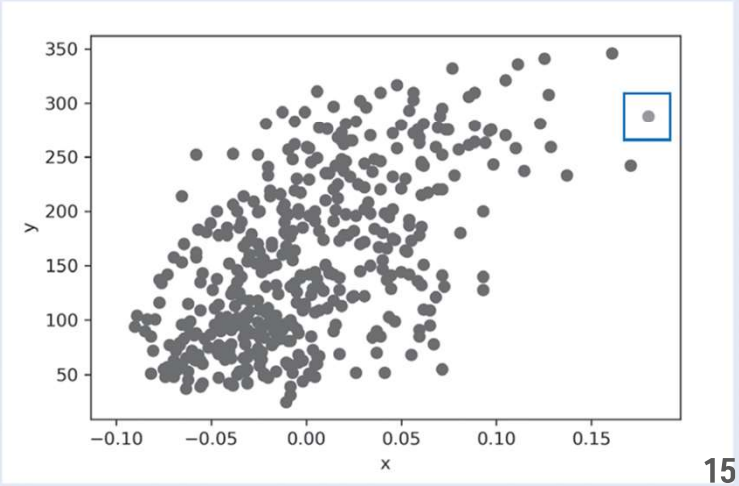


모델로 예측하기

$$\hat{y} = 913.6x + 123.4$$

```
x_new = 0.18
y_pred = x_new * w + b
print(y_pred)
287.8416643899983
```

```
plt.scatter(x, y)
plt.scatter(x_new, y_pred)
plt.xlabel('x')
plt.ylabel('y')
plt.show( )
```



지금까지 실습 내용을 정리해 보죠

지금까지는 모델을 이렇게 만들었습니다

1. w와 b를 임의의 값$(1.0, 1.0)$으로 초기화하고 훈련 데이터의 샘플을 하나씩 대입하여 y와 $\hat{y}$의 오차를 구합니다.
2. 1에서 구한 오차를 w와 b의 변화율에 곱하고 이 값을 이용하여 w와 b를 업데이트합니다.
3. 만약 $\hat{y}$이 y보다 커지면 오차는 음수가 되어 자동으로 w와 b가 줄어드는 방향으로 업데이트됩니다.
4. 반대로 $\hat{y}$이 y보다 작으면 오차는 양수가 되고 w와 b는 더 커지도록 업데이트됩니다.