03 머신러닝의 기초를 다집니다

- 수치 예측

03-4 선형 회귀를 위한 뉴런을 만듭니다

Neuron 클래스 만들기

```
class Neuron:

    def __init__(self):
    # 초기화 작업을 수행합니다.
    ...

    # 필요한 메서드를 추가합니다.
    ...
```

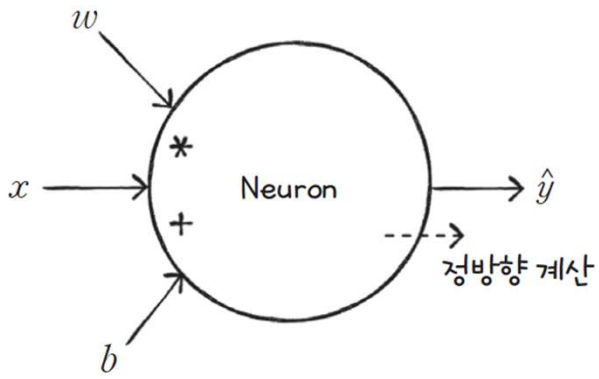
__init__() 메서드 작성하기

```
def __init__(self):
    self.w = 1.0
    self.b = 1.0
```

정방향 계산 만들기

오차를 계산하기 위해 $\hat{y} = w \times x + b$ 을 먼저 구해야 합니다

```
def forpass(self, x):
    y_hat = x * self.w + self.b        # 직선 방정식을 계산합니다.
    return y_hat
```



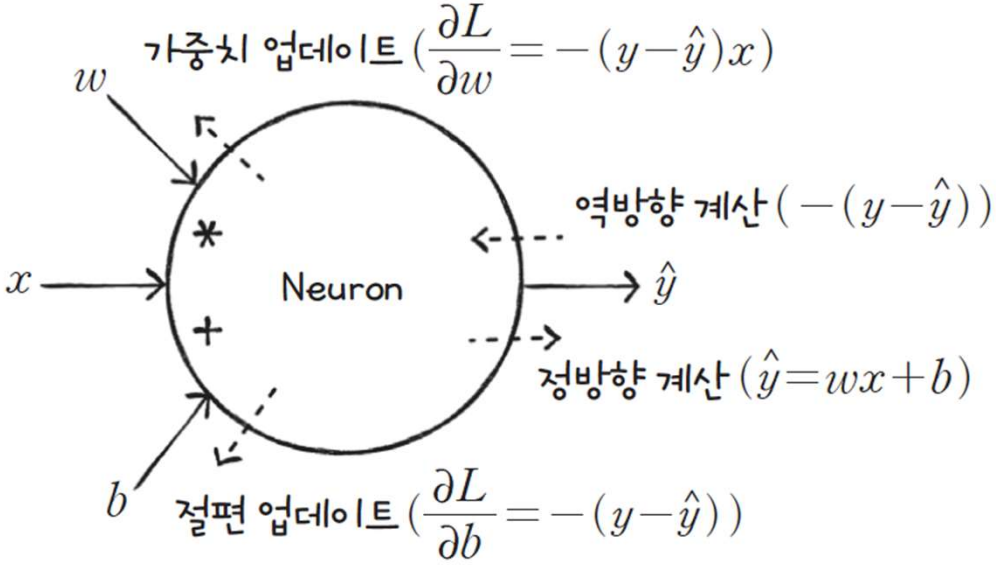
역방향 계산 만들기

손실 함수(제곱 오차 함수)

$$\frac{\partial L}{\partial w} = -(y - \hat{y})x$$

$$\frac{\partial L}{\partial b} = -(y - \hat{y})$$

```
def backprop(self, x, err):
    w_grad = x * err
    b_grad = 1 * err
    return w_grad, b_grad
```



훈련을 위한 fit() 메서드 구현

```
def fit(self, x, y, epochs=100):
    for i in range(epochs):                  # 에포크만큼 반복합니다.
        for x_i, y_i in zip(x, y):           # 모든 샘플에 대해 반복합니다.
            y_hat = self.forpass(x_i) # 정방향 계산
            err = -(y_i - y_hat)             # 오차 계산
            w_grad, b_grad = self.backprop(x_i, err) # 역방향 계산
            self.w -= w_grad                 # 가중치 업데이트
            self.b -= b_grad                 # 절편 업데이트
```

Neuron 클래스

```
class Neuron:

    def __init__(self):
        self.w = 1.0                        # 가중치를 초기화합니다.
        self.b = 1.0                        # 절편을 초기화합니다.

    def forpass(self, x):
        y_hat = x * self.w + self.b         # 직선 방정식을 계산합니다.
        return y_hat

    def backprop(self, x, err):
        w_grad = x * err                    # 가중치에 대한 그레이디언트를 계산합니다.
        b_grad = 1 * err                    # 절편에 대한 그레이디언트를 계산합니다.
        return w_grad, b_grad

    def fit(self, x, y, epochs=100):
        for i in range(epochs):             # 에포크만큼 반복합니다.
            for x_i, y_i in zip(x, y):      # 모든 샘플에 대해 반복합니다.
                y_hat = self.forpass(x_i) # 정방향 계산
                err = -(y_i - y_hat)        # 오차 계산
                w_grad, b_grad = self.backprop(x_i, err) # 역방향 계산
                self.w -= w_grad            # 가중치 업데이트
                self.b -= b_grad            # 절편 업데이트
```

뉴런 훈련

```
neuron = Neuron( )
neuron.fit(x, y)
```

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