Linear Regression
With
PyTorch

2017010698 수학과 오서영

1. Data Definition

훈련 데이터셋 (training dataset)
: 예측을 위해 사용하는 데이터

테스트 데이터셋 (test dataset)
: 모델이 얼마나 잘 작동하는지 판별하는 데이터

Hours (x)	Points (y)
1	2
2	4
3	6
4	?

Training dataset

Test dataset

> 내가 4시간을 공부한다면 몇 점을 맞을 수 있을까요?

1. Data Definition

훈련 데이터셋
파이토치의 텐서의 형태 (torch.tensor)
입력과 출력을 각기 다른 텐서에 저장

$$X_{\text{train}} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad Y_{\text{train}} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$$

x_train : 공부한 시간, y_train : 점수

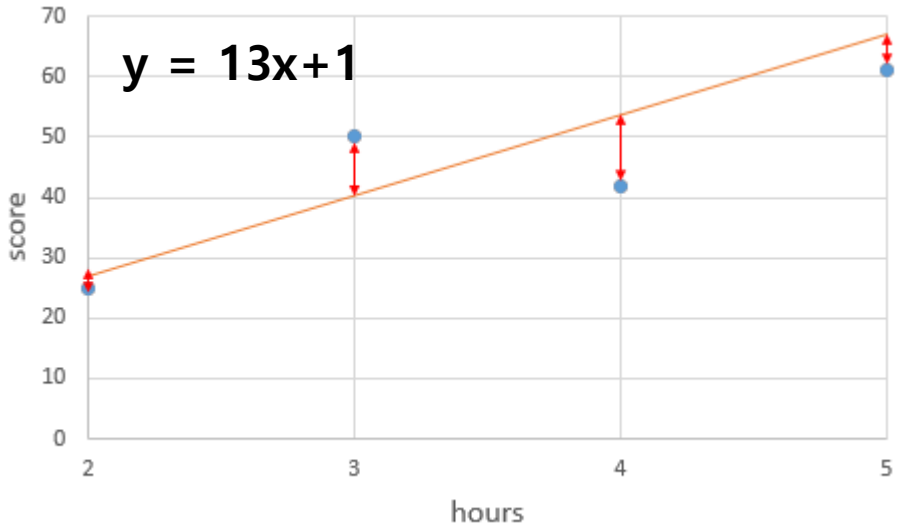
```
x_train = torch.FloatTensor([[1], [2], [3]])
y_train = torch.FloatTensor([[2], [4], [6]])
```

2. Hypothesis and Cost function

선형회귀의 가설

$y = Wx+b$
$H(x) = Wx+b$

W : Weight, **b** : bias



어떤 직선이 가장 적절한 직선? -> **오차(error)** 개념을 도입

hours(x)	2	3	4	5
실제값	25	50	42	61
예측값	27	40	53	66
오차	-2	10	-7	-5

$$\sum_{i=1}^{n} \left[y^{(i)} - H(x^{(i)})\right]^2 = (-2)^2 + 10^2 + (-7)^2 + (-5)^2 = 178$$

$$\frac{1}{n}\sum_{i=1}^{n} \left[y^{(i)} - H(x^{(i)})\right]^2 = 178/4 = 44.5$$

평균 제곱 오차(Mean Squared Error, MSE)

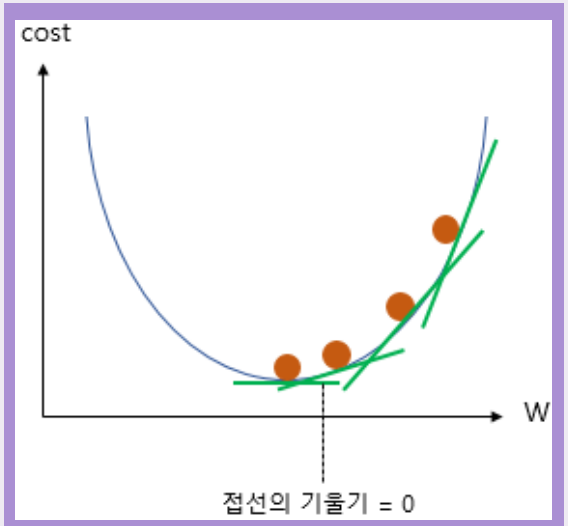
3. Gradient Descent

Cost Function

$$cost(W,b) = \frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} - H(x^{(i)})\right]^2$$

Cost(W, b)를 최소가 되게 만드는 W와 b를 구하기 -> how?

-> **Optimizer – Gradient Descent**



$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

$$\text{기울기} = \frac{\partial cost(W)}{\partial W}$$

- 기울기가 음수일 때 : W의 값이 증가

  $W := W - \alpha \times (\text{음수기울기}) = W + \alpha \times (\text{양수기울기})$

- 기울기가 양수일 때 : W의 값이 감소

  $W := W - \alpha \times (\text{양수기울기})$

접선의 기울기가 음수거나, 양수일 때
모두 접선의 기울기가 0인 방향으로 W의 값을 조정

4. Training

```
# data
x_train = torch.FloatTensor([[1], [2], [3]])
y_train = torch.FloatTensor([[2], [4], [6]])

# initialize W and b
W = torch.zeros(1, requires_grad=True)
b = torch.zeros(1, requires_grad=True)

# optimizer
optimizer = optim.SGD([W, b], lr=0.01)

nb_epochs = 2000
for epoch in range(nb_epochs + 1):

    # Hypothesis
    hypothesis = x_train * W + b

    # cost
    cost = torch.mean((hypothesis - y_train) ** 2)

    # update
    optimizer.zero_grad()
    cost.backward()
    optimizer.step()

    # print
    if epoch % 100 == 0:
        print('Epoch {:4d}/{} W: {:.3f}, b: {:.3f} Cost: {:.6f}'.format(
            epoch, nb_epochs, W.item(), b.item(), cost.item()
        ))
```

5. Autograd

```
w = torch.tensor(2.0, requires_grad=True)

y = w**2
z = 2*y + 5

z.backward()
print('수식을 w로 미분한 값 : {}'.format(w.grad))
```

```
수식을 w로 미분한 값 : 8.0
```

6. Multivariable Linear regression

$$H(X) = w_1x_1 + w_2x_2 + w_3x_3$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1w_1 + x_2w_2 + x_3w_3 \end{pmatrix}$$

$$H(X) = XW$$

Quiz 1 (x1)	Quiz 2 (x2)	Quiz 3 (x3)	Final (y)
73	80	75	152
93	88	93	185
89	91	80	180
96	98	100	196
73	66	70	142

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} + \begin{pmatrix} b \\ b \\ b \\ b \\ b \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 + b \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 + b \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 + b \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 + b \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 + b \end{pmatrix}$$

$$H(X) = XW + B$$

6. Multivariable Linear regression

```
# initialize W and b
W = torch.zeros((3, 1), requires_grad=True)
b = torch.zeros(1, requires_grad=True)

# optimizer
optimizer = optim.SGD([W, b], lr=1e-5)

nb_epochs = 20
for epoch in range(nb_epochs + 1):

    # H(x)
    # b : broadcasting
    hypothesis = x_train.matmul(W) + b

    # cost
    cost = torch.mean((hypothesis - y_train) ** 2)

    # update
    optimizer.zero_grad()
    cost.backward()
    optimizer.step()

    # print
    print('Epoch {:4d}/{} hypothesis: {} Cost: {:.6f}'.format(
        epoch, nb_epochs, hypothesis.squeeze().detach(), cost.item()
    ))
```