

# 행렬곱의 Backpropagation

조희철

2020년 12월 10일

$m \times n$  행렬  $X$ ,  $n \times k$  행렬  $W$ 와 함수  $f: \mathbb{R}^{mk} \rightarrow \mathbb{R}$ 에 대하여,  $X, W$ 의 각 원소에 대한 미분을 구해보자.

$$\begin{array}{c} \boxed{X} \\ \boxed{W} \end{array} \begin{array}{c} \longrightarrow Y = XW \xrightarrow{f} f(Y) = f(XW) \in \mathbb{R} \end{array}$$

행렬  $Y$ 는 함수  $f$ 를 거쳐 최종적으로 loss(cost) 값인 scalar가 된다.

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}, W = \begin{pmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nk} \end{pmatrix}, Y = XW = \begin{pmatrix} y_{11} & \cdots & y_{1k} \\ \vdots & \ddots & \vdots \\ y_{m1} & \cdots & y_{mk} \end{pmatrix}.$$

$\frac{\partial f}{\partial Y}$  (Jacobian)를 다음과 같이 정의하면 (또는 주어졌다고 하면),

$$\frac{\partial f}{\partial Y} := \begin{pmatrix} \frac{\partial f}{\partial y_{11}} & \cdots & \frac{\partial f}{\partial y_{1k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial y_{m1}} & \cdots & \frac{\partial f}{\partial y_{mk}} \end{pmatrix} \leftarrow m \times k$$

$X, W$ 의 각 원소별 미분 (Jacobian)은 다음과 같다.

$$\begin{pmatrix} \frac{\partial f}{\partial x_{11}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_{m1}} & \cdots & \frac{\partial f}{\partial x_{mn}} \end{pmatrix} = \underbrace{\frac{\partial f}{\partial Y} W^T}_{(m \times k)(k \times n)} \leftarrow m \times n \quad (1)$$

$$\begin{pmatrix} \frac{\partial f}{\partial w_{11}} & \cdots & \frac{\partial f}{\partial w_{1k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial w_{n1}} & \cdots & \frac{\partial f}{\partial w_{nk}} \end{pmatrix} = \underbrace{X^T \frac{\partial f}{\partial Y}}_{(n \times m)(m \times k)} \leftarrow n \times k \quad (2)$$

증명을 하기에 앞서 문제를 다시 한번 정리해 보자.  $X, W$ 가 주어져 있고,  $Y = XW, f(Y) \in \mathbb{R}$ 로 정의된다고 하자. 이 때  $f$ 를  $Y$ 의 각 원소별로 미분한 Jacobian  $\frac{\partial f}{\partial Y}$ 가 주어진다고 하면,  $f$ 를  $X, W$ 의 각 원소로 미분한 Jacobian은 각각 식(1), (2)가 된다는 것이다.

식(1), (2) 중에 하나만 증명하면, 나머지 하나는 대칭성에 의해 증명된다. 그래서 우리는 식(1)만 증명하고자 한다. 식(1)을 3단계로 나누어서 증명해보자.

- $X, W$ 가 각각  $1 \times n, n \times 1$ 인 경우:

$$XW = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = y = Y$$

$$\begin{aligned}
\frac{\partial f}{\partial x_i} &= \frac{\partial f}{\partial y} \frac{\partial y}{\partial x_i} = \frac{\partial f}{\partial y} w_i \\
\frac{\partial f}{\partial X} &= \left( \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right) \\
&= \left( \frac{\partial f}{\partial y} w_1 \quad \frac{\partial f}{\partial y} w_2 \quad \cdots \quad \frac{\partial f}{\partial y} w_n \right) \\
&= \frac{\partial f}{\partial y} \begin{pmatrix} w_1 & w_2 & \cdots & w_n \end{pmatrix} \\
&= \frac{\partial f}{\partial y} \begin{pmatrix} w_1 & w_2 & \cdots & w_n \end{pmatrix} \\
&= \underbrace{\frac{\partial f}{\partial Y}}_{(1 \times 1)(1 \times n)} W^T \quad \leftarrow 1 \times n
\end{aligned} \tag{3}$$

- $X, W$  가 각각  $1 \times n, n \times k$  인 경우:

$$\begin{aligned}
XW &= \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nk} \end{pmatrix} \\
&= \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \left( \begin{array}{c|c|c} \boxed{W_1} & \boxed{W_2} & \cdots & \boxed{W_k} \end{array} \right) \quad \leftarrow W_i \text{ 는 } W \text{ 의 열벡터} \\
&= \begin{pmatrix} y_1 & y_2 & \cdots & y_k \end{pmatrix} = Y
\end{aligned}$$

먼저, chain rule에 의해,

$$\frac{\partial f}{\partial x_i} = \frac{\partial f}{\partial y_1} \frac{\partial y_1}{\partial x_i} + \frac{\partial f}{\partial y_2} \frac{\partial y_2}{\partial x_i} + \cdots + \frac{\partial f}{\partial y_k} \frac{\partial y_k}{\partial x_i} \quad \text{by chain rule}$$

가 되고,  $\frac{\partial f}{\partial X} = \left( \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right)$ ,  $\frac{\partial y_i}{\partial X} = \left( \frac{\partial y_i}{\partial x_1} \quad \frac{\partial y_i}{\partial x_2} \quad \cdots \quad \frac{\partial y_i}{\partial x_n} \right)$  이므로,

$$\begin{aligned}
\frac{\partial f}{\partial X} &= \frac{\partial f}{\partial y_1} \frac{\partial y_1}{\partial X} + \frac{\partial f}{\partial y_2} \frac{\partial y_2}{\partial X} + \cdots + \frac{\partial f}{\partial y_k} \frac{\partial y_k}{\partial X} \quad \text{by chain rule} \\
&= \frac{\partial f}{\partial y_1} W_1^T + \frac{\partial f}{\partial y_2} W_2^T + \cdots + \frac{\partial f}{\partial y_k} W_k^T \quad \text{by (3)} \\
&= \left( \frac{\partial f}{\partial y_1} \quad \frac{\partial f}{\partial y_2} \quad \cdots \quad \frac{\partial f}{\partial y_k} \right) \left( \begin{array}{c} \boxed{W_1^T} \\ \boxed{W_2^T} \\ \vdots \\ \boxed{W_k^T} \end{array} \right) \quad \leftarrow (\text{행벡터의 일차결합}) \\
&= \left( \frac{\partial f}{\partial y_1} \quad \frac{\partial f}{\partial y_2} \quad \cdots \quad \frac{\partial f}{\partial y_k} \right) W^T \\
&= \underbrace{\frac{\partial f}{\partial Y}}_{(1 \times k)(k \times n)} W^T \quad \leftarrow 1 \times n
\end{aligned}$$

- $X, W$  가 각각  $m \times n, n \times k$  인 경우:  $X$  의 각 행을 서로 독립적이다. 그렇기 때문에,  $X$  의 각 행은  $Y$  의 각 행에 대응된다. 따라서 행으로의 확장은  $X, \frac{\partial f}{\partial Y}$  를 행으로 확장하면 된다.

$$\frac{\partial f}{\partial X} = \underbrace{\frac{\partial f}{\partial Y}}_{(m \times k)(k \times n)} W^T \quad \leftarrow m \times n$$

## ♠ Example 1 <sup>1</sup>

```
import numpy as np
# N 은배치크기이며 , D_in 은입력의차원입니다 .
# H 는은닉층의차원이며 , D_out 은출력차원입니다 .
N, D_in, H, D_out =64, 1000, 100, 10

# 무작위의입력과출력데이터를생성합니다 .
x =np.random.randn(N, D_in)
y =np.random.randn(N, D_out)

# 무작위로가중치를초기화합니다 .
w1 =np.random.randn(D_in, H)
w2 =np.random.randn(H, D_out)

learning_rate =1e-6
for t in range(500):
    # 순전파단계 : 예측값 y 를계산합니다 .
    h =x.dot(w1) # (N,H)
    h_relu =np.maximum(h, 0)
    y_pred =h_relu.dot(w2) # (N,D_out)

    # 손실 (loss) 을계산하고출력합니다 .
    loss =np.square(y_pred -y).sum()
    print(t, loss)

    # 손실에따른 w1, w2 의변화도를계산하고역전파합니다 .
    grad_y_pred =2.0 *(y_pred -y) # shape (N,D_out)
    grad_w2 =h_relu.T.dot(grad_y_pred) # shape w2 = (H,D_out)
    grad_h_relu =grad_y_pred.dot(w2.T)
    grad_h =grad_h_relu.copy()
    grad_h[h <0] =0
    grad_w1 =x.T.dot(grad_h) # shape w1 =(D_in,H)

    # 가중치를갱신합니다 .
    w1 -=learning_rate *grad_w1
    w2 -=learning_rate *grad_w2
```

## ♠ Example 2 <sup>2</sup>

$n \times 1$  행렬  $X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}^T$ ,  $Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}^T$  에 대하여

$$\hat{Y} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = XW$$

<sup>1</sup>[https://tutorials.pytorch.kr/beginner/examples\\_tensor/two\\_layer\\_net\\_numpy.html#sphx-glr-beginner-examples-tensor-two-layer-net-numpy](https://tutorials.pytorch.kr/beginner/examples_tensor/two_layer_net_numpy.html#sphx-glr-beginner-examples-tensor-two-layer-net-numpy)

<sup>2</sup>[https://pytorch.org/tutorials/beginner/examples\\_tensor/polynomial\\_numpy.html#sphx-glr-beginner-examples-tensor-polynomial-numpy](https://pytorch.org/tutorials/beginner/examples_tensor/polynomial_numpy.html#sphx-glr-beginner-examples-tensor-polynomial-numpy)

$$\frac{\partial L}{\partial W} = X^T \begin{pmatrix} 2(\hat{y}_1 - y_1) \\ 2(\hat{y}_2 - y_2) \\ \vdots \\ 2(\hat{y}_n - y_n) \end{pmatrix} = \begin{pmatrix} 2 \sum_{i=1}^n (\hat{y}_i - y_i) \\ 2 \sum_{i=1}^n x_i (\hat{y}_i - y_i) \\ 2 \sum_{i=1}^n x_i^2 (\hat{y}_i - y_i) \\ 2 \sum_{i=1}^n x_i^3 (\hat{y}_i - y_i) \end{pmatrix}$$

```
import numpy as np
import math
X = np.linspace(-math.pi, math.pi, 2000) #shape (2000,)
Y = np.sin(X)

# Randomly initialize weights
a = np.random.randn()
b = np.random.randn()
c = np.random.randn()
d = np.random.randn()

learning_rate = 1e-6
for t in range(2000):
    # Forward pass: compute predicted y
    # y = a + b x + c x^2 + d x^3
    y_pred = a + b * x + c * x ** 2 + d * x ** 3

    # Compute and print loss
    loss = np.square(y_pred - y).sum()
    if t % 100 == 99:
        print(t, loss)

    # Backprop to compute gradients of a, b, c, d with respect to loss
    grad_y_pred = 2.0 * (y_pred - y)
    grad_a = grad_y_pred.sum()
    grad_b = (grad_y_pred * x).sum()
    grad_c = (grad_y_pred * x ** 2).sum()
    grad_d = (grad_y_pred * x ** 3).sum()

    # Update weights
    a -= learning_rate * grad_a
    b -= learning_rate * grad_b
    c -= learning_rate * grad_c
    d -= learning_rate * grad_d

print(f'Result: y = {a} + {b} x + {c} x^2 + {d} x^3')
```