Khyeyoon

Computing derivatives

$$\begin{tikzcd}
da=3 \quad a=5 \arrow[rr] & & \overset{11}{\boxed{v=a+u}} \arrow[r] & \overset{33}{\boxed{J=3v}} \\
db=6 \quad b=3 \arrow[r] & \overset{6}{\boxed{u=bc}} \arrow[ur] & & \\
dc=9 \quad c=2 \arrow[ur] & & &
\end{tikzcd}$$

① $\frac{dJ}{dv} = 3$

② $\frac{dv}{du} = 1$

③ $\frac{dv}{da} = 1$

④ $\frac{du}{db} = c$

⑤ $\frac{du}{dc} = b$

"chain rule"

$$\frac{dJ}{da} = \frac{dJ}{\cancel{dv}} \cdot \frac{\cancel{dv}}{da} = 3 \cdot 1 = 3$$

$$\frac{dJ}{db} = \frac{dJ}{dv} \cdot \frac{dv}{du} \cdot \frac{du}{db} = 3 \cdot 1 \cdot c = 3c \rightarrow 6$$

$$\frac{dJ}{dc} = \frac{dJ}{dv} \cdot \frac{dv}{du} \cdot \frac{du}{dc} = 3 \cdot 1 \cdot b = 3b \rightarrow 9$$

Khyeyoon

Logistic regression derivatives

$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

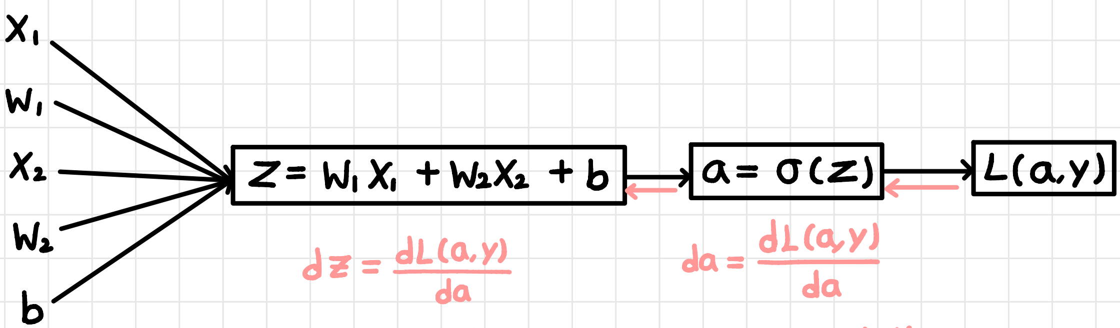
$$L(a, y) = -(y \log(a) + (1-y)\log(1-a))$$

경사하강법

$$w_1 := w_1 - \alpha dw_1$$

$$w_2 := w_2 - \alpha dw_2$$

$$b := b - \alpha db$$



$$dz = \frac{dL(a,y)}{da}$$

$$= \frac{dL}{da} \cdot \frac{da}{dz}$$

$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) \cdot a(1-a)$$

$$= a - y$$

$$da = \frac{dL(a,y)}{da}$$

$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

$$dw_1 = x_1 \cdot dz$$

$$dw_2 = x_2 \cdot dz$$

$$db = dz$$

Logistic regression on m examples

Khyeyoon

$J=0$; $dw_1=0$; $dw_2=0$; $db=0$

① For $i=1$ to m

$z^i = w^T x^i + b$

$a^i = \sigma(z^i)$

$J += -[y^i \log a^i + (1-y^i)\log(1-a^i)]$

$dz^i = a^i - y^i$

$dw_1 += x_1^i dz^i$

$dw_2 += x_2^i dz^i$

n=2 ②

$db += dz^i$

$J /= m$

$dw_1 /= m$

$dw_2 /= m$

$db /= m$

1 step

$$\boxed{\begin{aligned} w_1 &:= w_1 - \alpha dw_1 \\ w_2 &:= w_2 - \alpha dw_2 \\ b &:= b - \alpha db \end{aligned}}$$

2개의 for문이 사용됨

→ 비효율적인 코드!!

어떤식으로 구현해야 할까??

Vectorization

Khyeyoon

What is vectorization?

$$z = W^T X + b \qquad W = \begin{bmatrix} \vdots \\ \cdot \\ \vdots \end{bmatrix} \qquad X = \begin{bmatrix} \vdots \\ \cdot \\ \vdots \end{bmatrix}$$

Non-vectorization

```
z = 0
for i in range(nx):
    z += W[i] * X[i]
z += b
```

numpy 내장함수 이용하여 for문 제거

- np.exp
- np.log
- np.abs
- np.maximum

Vectorization

```
z = np.dot(w, x) + b
```

Implementing Logistic Regression

Khyeyoon

$J=0$; $dw_1=0$; $dw_2=0$; $db=0$

① For $i=1$ to m

$z^i = w^T x^i + b$ → np.dot(W^T, X) + b

$a^i = \sigma(z^i)$ → $A = \sigma(Z)$

$J += -[y^i \log a^i + (1-y^i)\log(1-a^i)]$

$dz^i = a^i - y^i$ → $dZ = A - Y$

$dw_1 += x_1^i dz^i$

$dw_2 += x_2^i dz^i$

n=2 ② → $dw = \frac{1}{m} X dz^T$

$db += dz^i$ → $db = \frac{1}{m}$ np.sum(dz)

$J /= m$

$dw_1 /= m$

$dw_2 /= m$

$db /= m$

1 step

$$\boxed{\begin{aligned} w_1 &:= w_1 - \alpha dw_1 \\ w_2 &:= w_2 - \alpha dw_2 \\ b &:= b - \alpha db \end{aligned}}$$

1 step

$$\boxed{\begin{aligned} W &:= W - \alpha dW \\ b &:= b - \alpha db \end{aligned}}$$

경사하강법을 여러번 하려면
for 문을 사용해야 함