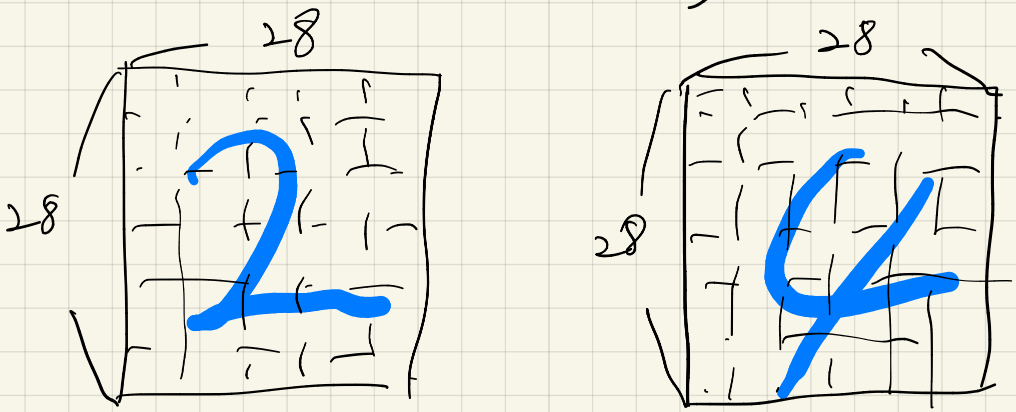
cosmax AI class

- numpy를 이용한

Logistic Regression의 구현

Problem : MNIST image binary classification. (이진분류)



MNIST Dataset

0 ~ 9 까지의

'손글씨' Data

$$\underbrace{\left.\begin{bmatrix} [0,0,0,\dots,0], \\ [0,0,1,\dots,0] \\ \vdots \\ [0,0,1,\dots,0] \end{bmatrix}\right\}28}_{28}$$

2 × 2 Array 로 구성되어 있음.

MNIST Dataset 의 구조

x_train.shape = (60000, 28, 28)

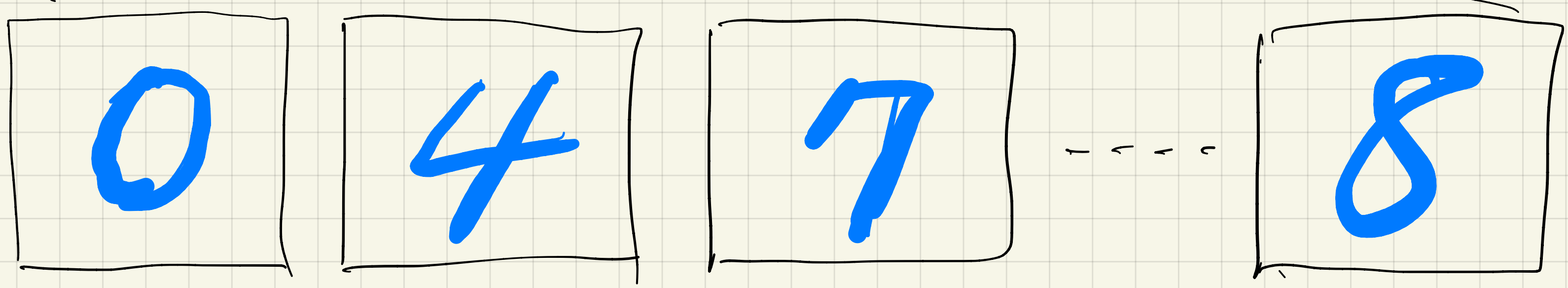
28×28
60,000장

y_train.shape = (60000, 1)

0~9 숫자를
나타내는 label
60,000

60,000개

x_train:

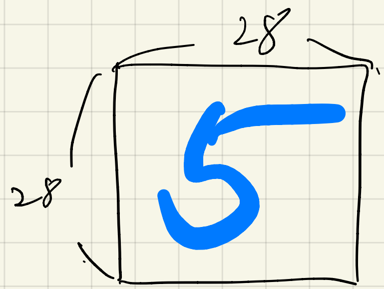


y_train: 0 4 7 8

x_test.shape (?)

y_test.shape (?)

Training Set Prep



$$\Rightarrow \left[\;\begin{array}{l}[0,0,1,0\cdots],\\ [0,1,0,1\cdots]\\ \quad\vdots\\ [0,1\cdots\cdots,0,1]\end{array}\;\right] 28 \;\Rightarrow\; \overbrace{[0,1,0,0,1,\;\cdots\cdots,\;0,1]}^{784\,=\,28*28}$$

2D-Array

1D-Array flatten.

Flattened

X_train (?) : 784

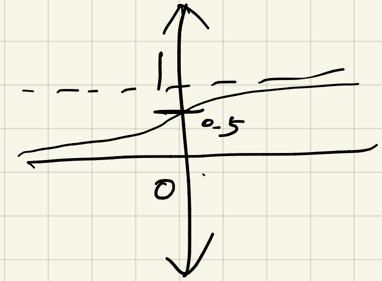
$$\left.\begin{array}{l}[0,1,0,\;\cdots\cdots\cdots,\;0,1]\\ [0,1,0\;\cdots\cdots\cdots\;0,1]\\ \quad\vdots\\ [0,1,0,1,1\;\cdots\cdots\;0,1]\end{array}\right\}\; 60000개$$

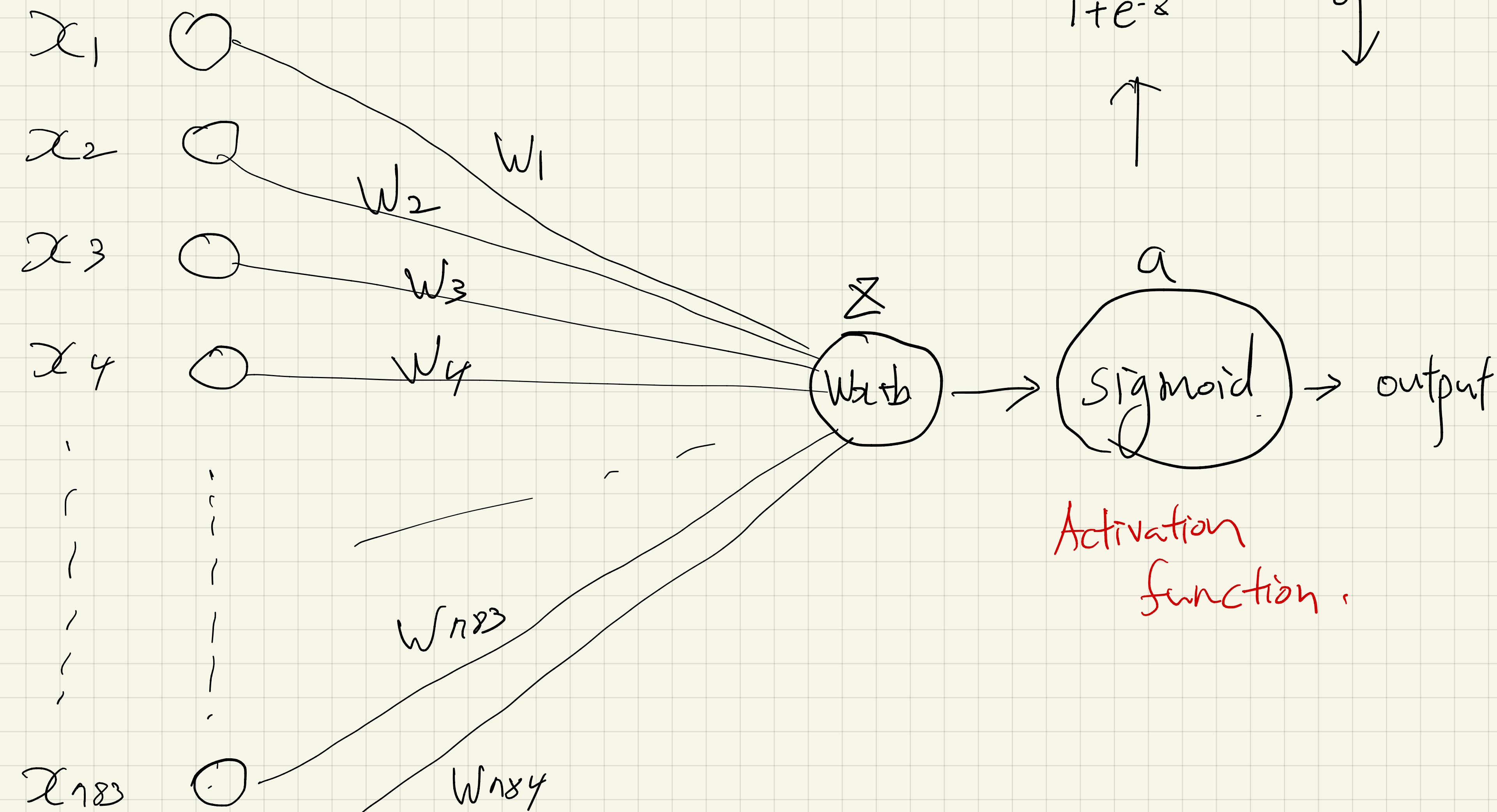
$x_1,\; x_2,\; x_3,\; x_4\; \cdots\cdots\; x_{783}\; x_{784}$

(training example)

Logistic Regression Model

$$\text{sigmoid}(z) = \frac{1}{1+e^{-z}}$$



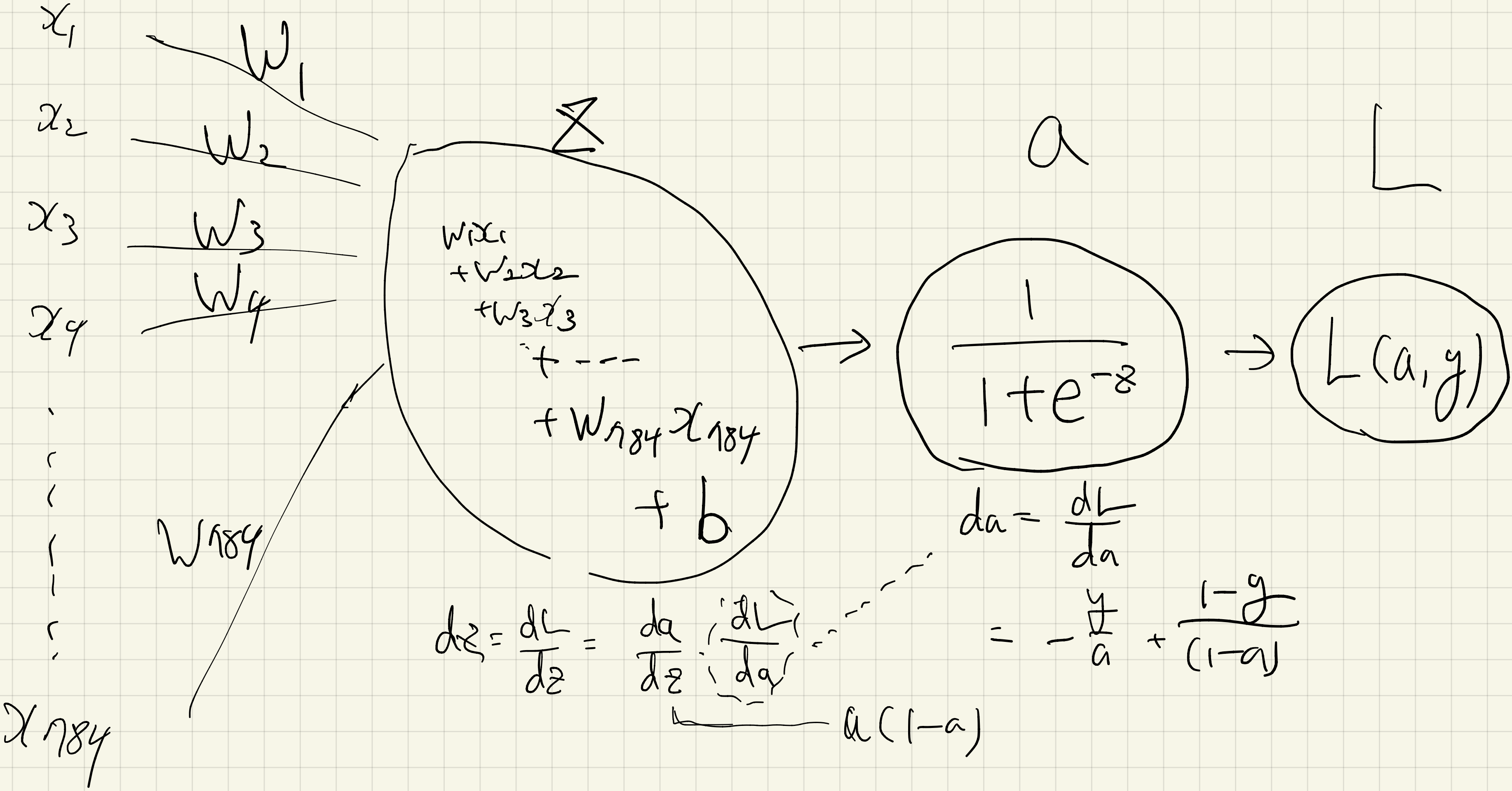


$\boxed{n\_x = 784}$

$\boxed{W: 784개 \quad b: 1개}$

trainable parameter.

Logistic Regression Basic Algorithm



$$da = \frac{dL}{da} = -\frac{y}{a} + \frac{1-y}{(1-a)}$$

$$dz = \frac{dL}{dz} = \underbrace{\frac{da}{dz}}_{a(1-a)} \cdot \frac{dL}{da}$$

$$= a - y$$

training set 1개 기준

하나의 traing set 에 대해서

$z = \boxed{w}\underset{\text{data}}{x} + \boxed{b}$ 계산

trainable parameter.

$a = \sigma(z)$ 계산

$L = L(a, \underbrace{y}_{\text{true label}})$ 계산

$\frac{dL}{dw_1} = dw_1 = x_1 \cdot dz \qquad \frac{dL}{dw_2} = dw_2 = x_2 \cdot dz$

$\cdots \frac{dL}{dw_{784}} = dw_{784} = x_{784} \cdot dz$

$\frac{dL}{db} = db = dz$

$w_1 := w_1 - \alpha dw_1 \qquad w_2 := w_2 - \alpha dw_2 \cdots$

$w_{784} := w_{784} - \alpha dw_{784}$

$b := b - \alpha db$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(a^{(i)}, y^{(i)})$$: cost function.

$$a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

$$(x^{(i)}, y^{(i)}) = \underbrace{\boxed{dw_1^{(i)},\ dw_2^{(i)},\ \cdots,\ dw_{784}^{(i)}}}_{784\text{개}},\ \underbrace{\boxed{db^{(i)}}}_{1\text{개}}$$

$$\frac{d}{dw_1} J(w,b) = \frac{\partial}{\partial w_1} J(w,b)$$

$$= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial w_1} L(a^{(i)}, y^{(i)})$$

— 다 합해서 평균을 낸다 —

Algorithm $J=0$, $dw_1=0 \sim dw_{784}=0$, $db=0$ ----

For $i=1$ to m ; One Epoch

$$z^{(i)} = w^T x^{(i)} + b$$

$$a^{(i)} = \text{sigmoid}(z^{(i)})$$

$$J \mathrel{+}= -[L(a^{(i)}, y^{(i)})]$$

$$= -(y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)}))$$

$$dz^{(i)} = a^{(i)} - y^{(i)}$$

$$dw_1 \mathrel{+}= x_1^{(i)} \cdot dz^{(i)}$$

$$\vdots$$

$$dw_{784} \mathrel{+}= x_{784}^{(i)} \cdot dz^{(i)}$$

$$db \mathrel{+}= dz^{(i)}$$

— Gradient 계산

$J /= m$, $dw_1 /= m$, ---, $dw_{784} /= m$, $db /= m$

$$w_1 = w_1 - \alpha\, dw_1$$

$$w_2 = w_2 - \alpha\, dw_2$$

$$\vdots$$

$$b = b - \alpha \cdot db$$

— Gradiant update

Training Set Vectorized Representation.

$$
\left.
\begin{aligned}
X_1 &= \overbrace{[0, 0, 0, 1, 0 \; \cdots\cdots \; 0, 1]}^{784} \\
X_2 &= [0, 0, 0, 1 \; \cdots\cdots \; 0, 1] \\
&\;\vdots \\
X_{60000} &= [0, 0, 1, 0 \; \cdots\cdots \; 0, 1]
\end{aligned}
\right\} 60000
$$

$n\_x = 784$: number of features

m = number of traing set

60,000개의 training image data를 하나의
matrix로 표시하면 (?)

$$X = \underbrace{\begin{bmatrix} | & | & | & & | \\ X_1 & X_2 & X_3 & \cdots\cdots & X_m \\ | & | & | & & | \end{bmatrix}}_{m} \left.\vphantom{\begin{bmatrix} | \\ X \\ | \end{bmatrix}}\right\} n_x = 784$$

$$X.\text{shape} = (n_x, m) = (784, 60000)$$

Label Vectorized Form

$$Y = [\underbrace{0,\ 2,\ 4,\ 8,\ \cdots,\ 7,\ 9}_{\substack{60{,}000\text{개} \\ m}}] \Big\}\ 1\text{개}$$

$$Y.\text{shape} = (1,\ m)$$

$$\begin{bmatrix} X \in \mathbb{R}^{n_x \times m} \\ Y \in \mathbb{R}^{1 \times m} \end{bmatrix}$$

$$X = \underbrace{\begin{bmatrix} | & | & & | \\ X_1 & X_2 & \cdots & X_m \\ | & | & & | \end{bmatrix}}_{m} \Big\} n_x \qquad Y = \underbrace{[0, 4, 1, \cdots 8]}_{m}$$

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_{n_x} \end{bmatrix} \qquad b = b$$

"real number"

cf. numpy broadcast

$$Z = W^T X + b \rightarrow Z = [Z^{(1)}, Z^{(2)}, \cdots, Z^{(m)}]$$

$$A = \text{sigmoid}(Z) \quad A = [A^{(1)}, A^{(2)}, \cdots, A^{(m)}]$$

$$L = L(A, Y) \quad L = [L^{(1)}, L^{(2)}, \cdots, L^{(3)}]$$

Vectorized Implementation

: m 개의 training set (1 - epoch)
을 단 "한번" 에 계산 !

$$Z = W^T X + b$$

$$A = \text{sigmoid}(Z)$$

$$L = \text{loss}(A, Y)$$

$$dZ = A - Y$$

$$dW = \frac{1}{m} \cdot X \cdot dZ^T = \frac{1}{m} * \text{np.dot}(X, dZ^T)$$

$$db = \frac{1}{m} \cdot \text{np.sum}(dZ)$$

$W = W - \alpha \cdot dW$ ------ vector

$b = b - \alpha \cdot db$ ------ scalar