2026/1/

2. Neuron Modeling

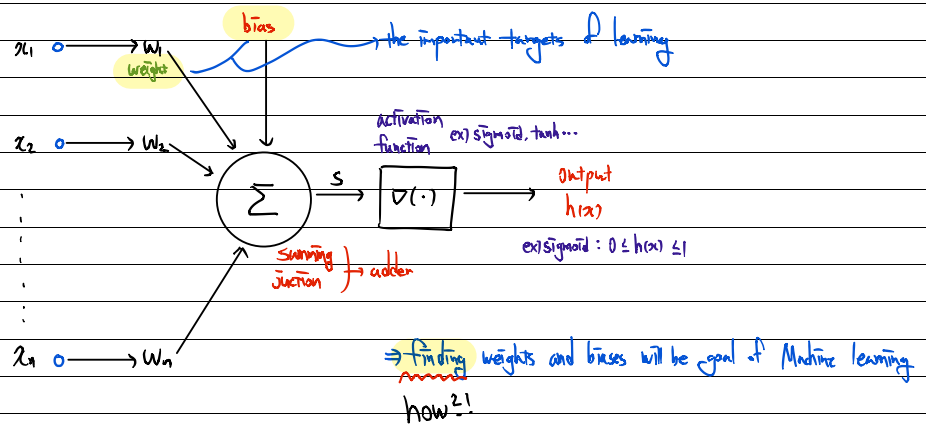
2-1. Modeling a Neuron

- three basic elements

1. synapses (weights)

2. adder (input vector → scalar : $\underset{\sim}{x} \longrightarrow w^T x$)
   → 우리가 signal을 구하려면 weighted sum ($w^T x$)를 해야함!

3. activation function (nonlinear)



⇒ finding weights and biases will be goal of Machine learning

how?!

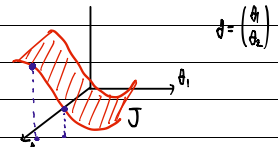
2.2 gradient descent

- a general technique for minimizing differentiable function

⇒ $\min_{\theta} J(\theta)$: parameter θ를 변화시켰을 때 어떤 값 $J(\theta)$를 가장 작게 만드는 방법 (∵ $J(\theta)$: cost function)

(Objective function : 학습과정에서 optimize 하고자 하는 function
Cost function : 전체 dataset에 대해 계산된 loss들의 Avg or sum
모델 학습 시 일반적으로 minimizing 하는 function
Loss function : 하나의 sample에 대해 모델의 $f(x) - \hat{f}(x)$: 오차를 측정하는 function)

- Idea : $J(\theta)$는 parameter space 안의 surface



1. start from somewhere on J
2. roll down the surface, decreasing J step by step

- two things to decide at each step!!

2.2 Gradient descent

- two things to decide at each step!!

① which direction ?!

⇒ 우리의 cost function이 decreasing 하는 방향!

방향? ⟶ $\nabla J(\theta)$: gradient는 함수가 가장 빠르게 증가하는 방향

→ 즉! $-\nabla$로 direction 결정하자!

∴ $-\nabla J(\theta) := \Delta(\theta)$ (현재 위치 θ에서 $\Delta(\theta)$ 만큼 가자!)

② how much ?

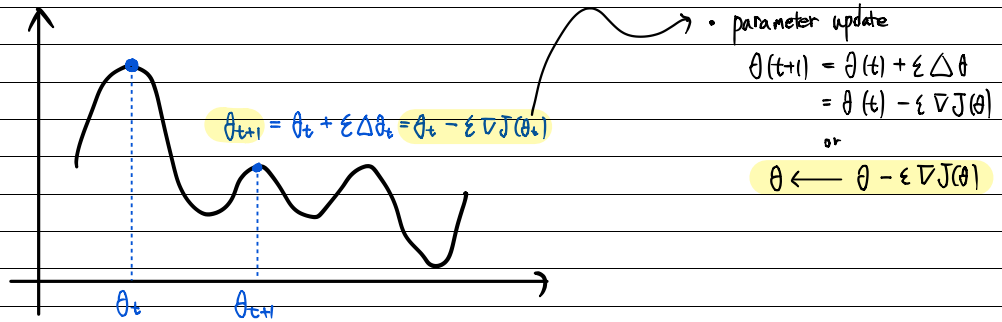
⇒ N·M ⟶ f''(곡률 정보)

∴ $1/f''$ ⇒ 곡률 大, 경사진 곳 step short

곡률 小, 평평한 곳 step long

⇒ but in our case.... f''를 구하는 건 불가능..

∴ ε (learning rate) (hyperparameter)



- parameter update

$\theta(t+1) = \theta(t) + \varepsilon\Delta\theta$

$= \theta(t) - \varepsilon\nabla J(\theta)$

or

$\theta \longleftarrow \theta - \varepsilon\nabla J(\theta)$

2.3 Gradient Descent Algorithm

algorithm 1 gradient descent

1: initialize θ (초기화)

2: **while** stopping criterion not met **do**

3: sample m examples: $\mathbb{X}_m = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(m)}, y^{(m)})\}$ (minibatch; data, label)

4: compute gradient estimate: $\hat{g} \leftarrow \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)})$ ▷ m forward props

5: apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\hat{g}$ ▷ ϵ: learning rate

6: **end while**

$\hat{g} \leftarrow \frac{1}{m}\sum \nabla_\theta L$

ex) simple linear regression

$S = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$

$= \frac{1}{n}\sum_{i=1}^{n}\varepsilon^2$

$\hat{g} = \frac{d\varepsilon}{dw}$

three variants

① m=1 (random 하게 하나씩 sample 뽑!)

⇒ stochastic GD

② 1 < m < N : minibatch sgd

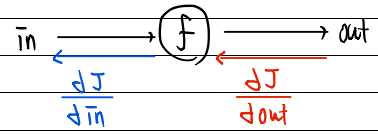
③ m = N : batch gradient descent.

2.4 Backprop Demystified

1. Single gate backprop

output → input (gradient)

- out = f(in)

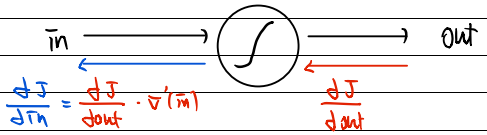


$$\frac{dJ}{d\bar{in}} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{out}}{d\bar{in}} = \frac{dJ}{d\bar{out}} \cdot \frac{df(\bar{in})}{d\bar{in}} = \frac{dJ}{d\bar{out}} \cdot f'(\bar{in})$$

output gradient · local gradient

2. Sigmoid

- $out = \sigma(\bar{in}) = \frac{1}{1+e^{-\bar{in}}}$



$$\sigma(x) = \frac{1}{1+e^{-x}} = (1+e^{-x})^{-1}$$

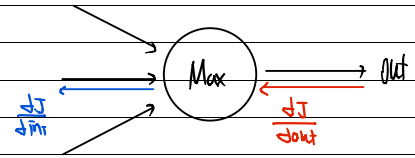
$$\frac{d\sigma}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} \quad \because \sigma(x)(1-\sigma(x)) = \frac{d\sigma}{dx} = \sigma'(x)$$

$$1-\sigma(x) = \frac{e^{-x}}{1+e^{-x}}$$

$$\frac{dJ}{d\bar{in}} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{out}}{d\bar{in}} = \frac{dJ}{d\bar{out}} \cdot \frac{d\sigma(\bar{in})}{d\bar{in}} = \frac{dJ}{d\bar{out}} \cdot \sigma'(\bar{in}) = \frac{dJ}{d\bar{out}} \cdot \sigma(1-\sigma)$$

5. Max (CNN: maxpooling)

- $out = \max_i \{\bar{in}_i\}$



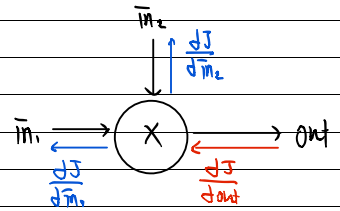
$$\frac{dJ}{d\bar{in}_i} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{out}}{d\bar{in}_i} = \frac{dJ}{d\bar{out}} \cdot \frac{d\max(\bar{in}_i)}{d\bar{in}_i} = \frac{dJ}{d\bar{out}} \text{ or } 0$$

$$\Rightarrow \begin{cases} \frac{d\bar{in}_i}{d\bar{in}_i} = 1 & \text{if } \bar{in}_i \text{ is max} \\ 0 & \text{o.w} \end{cases}$$

⇒ Max (forward) ⟺ Max (backprop)

3. Multiplication

- $out = \bar{in}_1 \cdot \bar{in}_2$

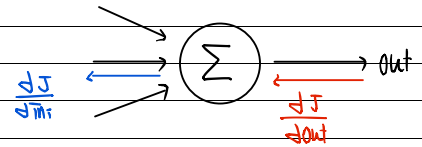


$$\frac{dJ}{d\bar{in}_1} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{out}}{d\bar{in}_1} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{in}_1\bar{in}_2}{d\bar{in}_1} = \frac{dJ}{d\bar{out}} \cdot \bar{in}_2$$

$$\frac{dJ}{d\bar{in}_2} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{out}}{d\bar{in}_2} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{in}_1\bar{in}_2}{d\bar{in}_2} = \frac{dJ}{d\bar{out}} \cdot \bar{in}_1$$

4. Summation

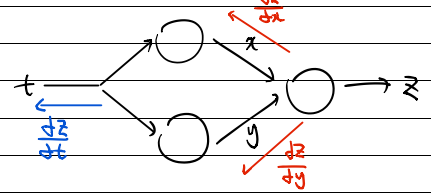
- $out = \sum_i \bar{in}_i = \bar{in}_1 + \bar{in}_2 + \cdots + \bar{in}_i$



$$\frac{dJ}{d\bar{in}_i} = \frac{dJ}{d\bar{out}} \cdot \frac{d\bar{out}}{d\bar{in}_i} = \frac{dJ}{d\bar{out}} \cdot \frac{d(\bar{in}_1 + \bar{in}_2 \cdots \bar{in}_i)}{d\bar{in}_i} = \frac{dJ}{d\bar{out}}$$

⇒ Sum (forward) ⟺ Fanout (backprop)

6. Backprop through fanout

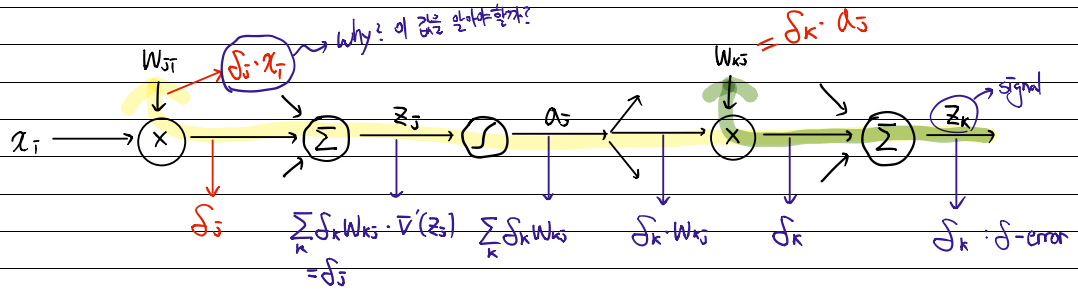


$$\frac{dz}{dt} = \frac{dz}{dx} \cdot \frac{dx}{dt} + \frac{dz}{dy} \cdot \frac{dy}{dt}$$

⇒ Fanout (forward) ⟺ Sum (backprop)

example !!

· compute $\frac{\partial J}{\partial w_{ji}}$



· parameter update

$$w_{ji} \leftarrow w_{ji} - \varepsilon \nabla \bigcirc$$

을 알아야하기 때문!

$$\Rightarrow w_{ji}^{(t+1)} \leftarrow w_{ji}^{(t)} - \varepsilon \cdot \frac{\partial J}{\partial w_{ji}}$$

$$= w_{ji}^{(t)} - \varepsilon \cdot (\delta_j \cdot x_i)$$

$$= w_{ji}^{(t)} - \varepsilon \cdot \left(\sum_k \delta_k \cdot w_{kj} \right) \cdot \sigma'(z_j) \cdot x_i$$

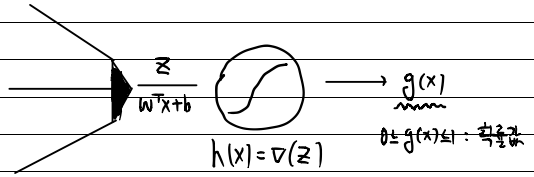
2.5 logistic regression

· logistic regression (in early neural nets : logistic regression unit = neuron)

: outputs probability of a binary response. ex) 내일 vs 그외날 / dead or alive

return soft labels (probability) ⇒ allow uncertainty

→ output : real (like regression) but bounded (like classification. probability)



· problem

given : $x \in \mathbb{R}^n$

want : $\hat{y} = p(y=1 \mid x)$

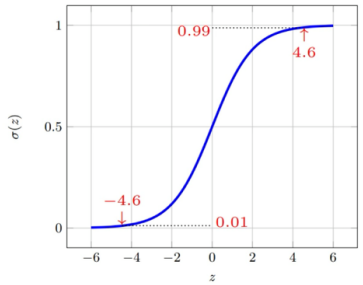
· model

$\hat{y} = \sigma(w^T x + b)$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$ (parameter)

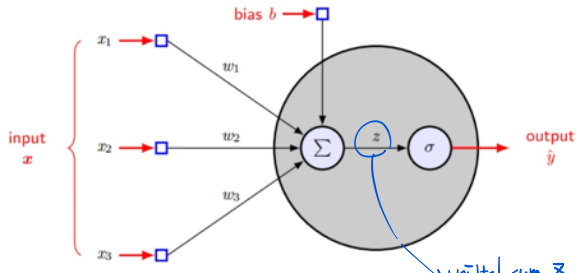
· sigmoid

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

$$\sigma'(z) = \sigma(z)(1-\sigma(z))$$



• computational graph



$$\hat{y} = \sigma(\underbrace{w^\top x + b}_{\triangleq z})$$

↘ weighted sum z

$$z = \sum_{i=1}^{n} w_i x_i + b$$

w_i: goal of ML

· Probabilistic interpretation

· given training set drawn from P_{data}

$$X = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\}$$

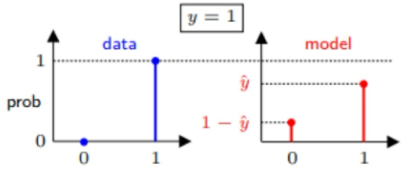
ex) 4-9 손글씨

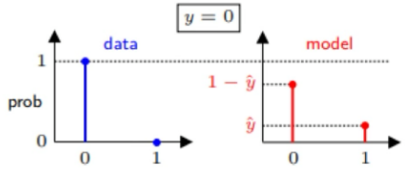
→ consider label $y^{(i)} \in \{0, 1\}$ as hard probability

$$y \equiv P(y=1|x) \Rightarrow P(y=0|x) = 1 - P(y=1|x) = 1-y$$

then! observed pmf $\hat{p}_{data} \in \{y \equiv P(y=1|x),\ 1-y \equiv P(y=0|x)\}$

fitted pmf $p_{model} \in \{\hat{y},\ 1-\hat{y}\}$





· Loss (pointwise)

· we use log loss

$$L(y, \hat{y}) = -y \log \hat{y} - (1-y) \log(1-\hat{y})$$

if $y=1 \Rightarrow L(y,\hat{y}) = -\log \hat{y}$, $\hat{y} \to 1$, $L(y,\hat{y}) \to 0$

if $y=0 \Rightarrow L(y,\hat{y}) = -\log(1-\hat{y})$, $\hat{y} \to 0$, $L(y,\hat{y}) \to 0$

★ $y = P(y=1|x)$ ★

· Cost function

$$J(w,b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)}) \right]$$

for update parameter w & b, we need to compute $\frac{dJ}{dw}$, $\frac{dJ}{db}$

($\because w_j \leftarrow w_j - \varepsilon \nabla \frac{dJ}{dw}$)

$$\frac{dJ}{dw} = \frac{dJ}{dz} \cdot \frac{dz}{dw} = \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot \left(x_j^{(i)}\right) = \frac{1}{m} \sum \left(\hat{y}^{(i)} - y^{(i)}\right) x_j^{(i)} = \frac{1}{m} \sum \left(\sigma(w^T x^{(i)} + b) - y^{(i)}\right) x_j^{(i)}$$

$$\frac{dJ}{db} = \frac{1}{m} \sum_{i=1}^{m} \left(\sigma(w^T x^{(i)} + b) - y^{(i)}\right) \cdot 1$$

2.6 Training by Backprop

Training a neuron

- Find w and b that minimize cost function $J(w,b)$
  ⇒ use iterative optimization i.e gradient descent

⇒ forward ⟶ output ⟶ output vs label ⟶ error ⟶ derror ⟶ gradient ⟶ backward

- Mini batch!

$i=1 \rightarrow z^{(1)} = w^T x^{(1)} + b$

$\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = w_1 x_1 + w_2 x_2 + b$

$\nabla(w_1 x_1 + w_2 x_2 + b) = a - \hat{y}$

Training logistic Regression by backpop

- minimize cost function by gradient descent.

$$J(w,b) = -\frac{1}{m}\sum \left(y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)}) \right)$$

- repeat over training examples

ε : learning rate (hyperparameter)

$$w \leftarrow w - \varepsilon \nabla J_w$$
$$b \leftarrow b - \varepsilon \nabla J_b$$

$\frac{1}{3}\sum$ $x_1^{(1)}\ \delta^{(1)}$, $x_1^{(2)}\ \delta^{(2)}$, $x_1^{(3)}\ \delta^{(3)}$

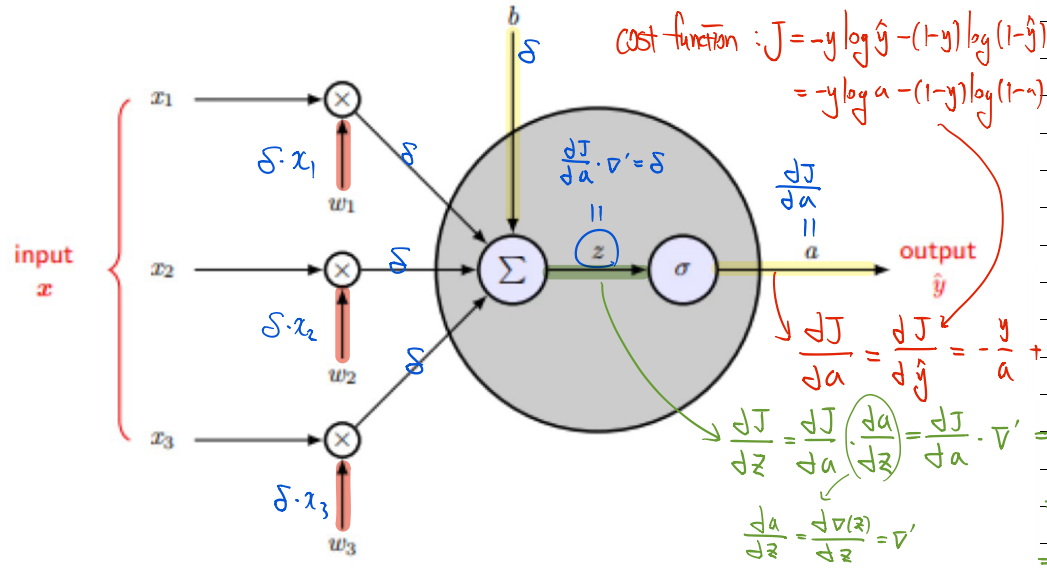
	x_1	x_2	y
1		D	△
2	☆		○
3			
4			
5			

$$\frac{dJ}{dw_1} = \frac{1}{m}\sum x_1^{(i)} \delta^{(i)} = \frac{1}{3}\sum_{i=1}^{3} x_1^{(i)} \delta^{(i)}$$

$$\frac{dJ}{dw_2} = \frac{1}{m}\sum x_2^{(i)} \delta^{(i)}$$

forward prop:

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



cost function : $J = -y \log \hat{y} - (1-y)\log(1-\hat{y})$
$= -y \log a - (1-y)\log(1-a)$

$\frac{dJ}{da} = \frac{dJ}{d\hat{y}} = -\frac{y}{a} + \frac{(1-y)}{1-a}$

$\frac{dJ}{dz} = \frac{dJ}{da}\left(\frac{da}{dz}\right) = \frac{dJ}{da} \cdot \nabla' = \left(-\frac{a}{y} + \frac{(1-y)}{1-a}\right) \cdot a(1-a)$

$= a - y = \delta$ (delta error)

$= \hat{y} - y$

$\frac{da}{dz} = \frac{d\nabla(z)}{dz} = \nabla'$

- SGD equation (1 example)

⇒ $J(w,b) = L(y,a) = -y \log a - (1-y)\log(1-a)$

$$\frac{dJ}{dw_1} = x_1 \cdot \delta = x_1 \cdot \frac{dJ}{dz} = x_1(a-y)$$

$$\frac{dJ}{db} = \delta = \frac{dJ}{dz} = (a-y)$$

⇒ apply update $w_1 \leftarrow w_1 - \varepsilon \frac{dJ}{dw_1} = w_1 - \varepsilon \cdot x_1 \cdot \delta$
$= w_1 - \varepsilon x_1 (a-y)$

$$b \leftarrow b - \varepsilon \frac{dJ}{db} = b - \varepsilon(a-y)$$

· Arranging minibatch

· two options to arrange m examples

▶ in columns

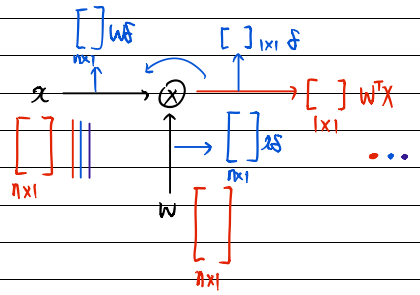
$$X_{(n,m)} = \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & \cdots & x_2^{(m)} \\ x_3^{(1)} & \cdots & x_3^{(m)} \\ x_n^{(1)} & \cdots & x_n^{(m)} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ X^{(1)} & X^{(2)} & \cdots & X^{(m)} \\ | & | & & | \end{bmatrix}$$

▶ in rows (design matrix)

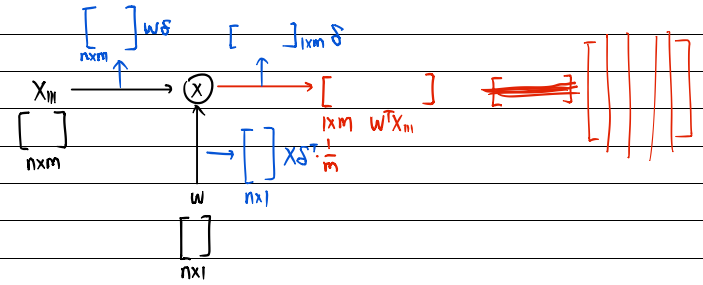
$$X = \begin{bmatrix} — & X^{(1)T} & — \\ & \vdots & \\ — & X^{(m)T} & — \end{bmatrix}$$

· Weighted Sum

· 1 example (column)

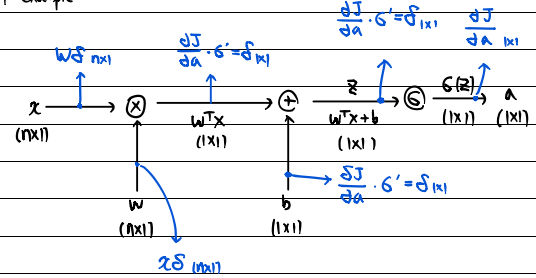


· size-m minibatch (column)



· Forward / backward prop

· 1 example



· size m minibatch