Class Plan

Reminders

Recap

Neural Nets

Architectures

Gradient Descent

Back propagation

Grading 100 points

93 - A	86 - B+
90 - A-	83 - B

Google Form

- go/cs1051/
- due midnight every class day
- worth 1 point per submission

Homework

- 4 points (3 soln., 1 self grade)
- use LaTex for theory
- python notebook for code

24 hours no questions asked
total lateness

Honor code

Attribute ideas and code

My expectation:

- do reading before class (max 15 min)
- do problem day it's assigned (mostly)
- complete google form

Recap

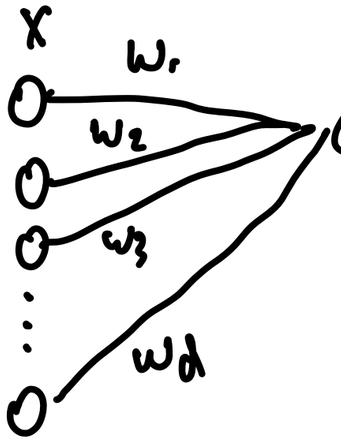
3 step recipe

1) architecture / model
2) loss $\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{y}_i)$
3) optimizer

Linear Regression

(x, y) $\quad f_w(x) \approx y$

1)



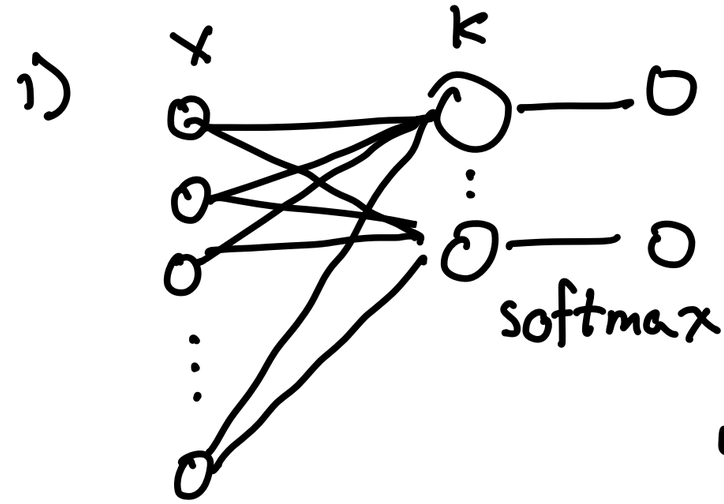
$$\langle x, w \rangle = \sum_{i=1}^{d} x_i \cdot w_i$$

$y \in \mathbb{R}$

2) $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$

Logistic Regression

1)

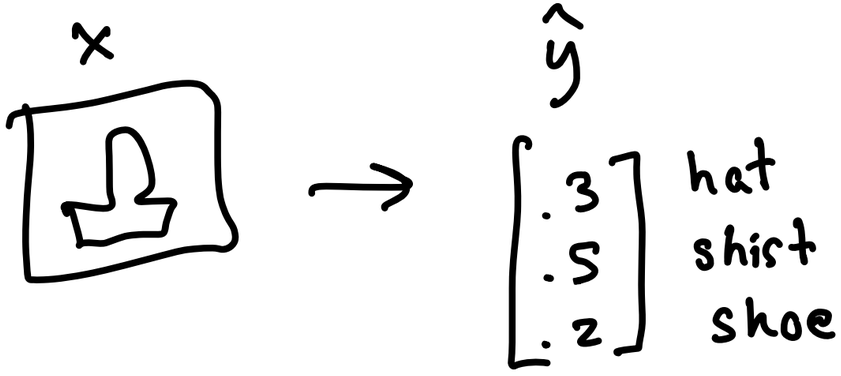


$y \in \mathbb{R}^K$

2)

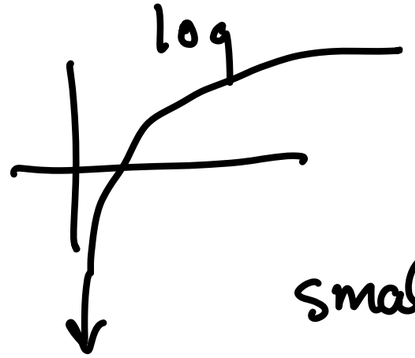
$$\ell(y, \hat{y}) = \sum_{j=1}^{K} \mathbb{1}[y = j] \cdot -\log(\hat{y}_j)$$

$\mathbb{1}[y = j]$: 1 if $y = j$, 0 else



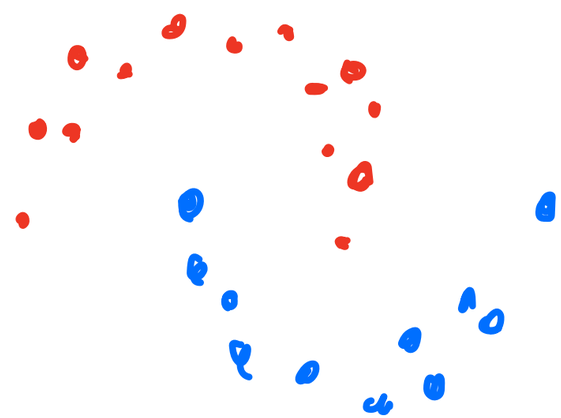
$$\ell(y, \hat{y}) = \sum_{j=1}^{k} \mathbb{1}[y = j] \cdot -\log(\hat{y}_j)$$

↑ 1 if $y = j$, 0 else

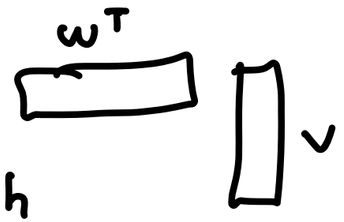


small $\hat{y}_j$ → large loss

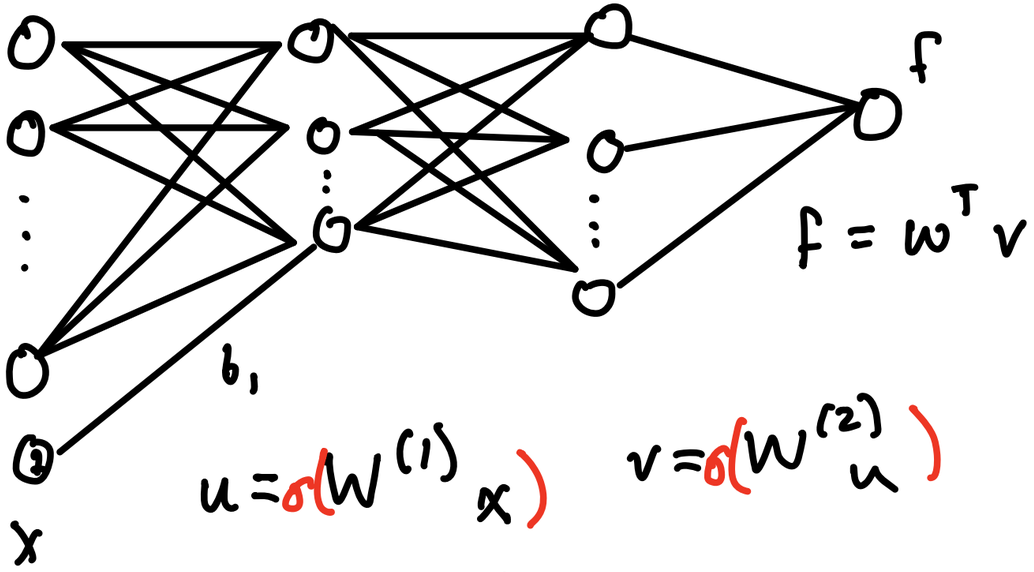
large $\hat{y}_j$ → small loss



Neural Networks



$x \in \mathbb{R}^d \quad u \in \mathbb{R}^h \quad v \in \mathbb{R}^h$



$u = \sigma(W^{(1)} x) \qquad v = \sigma(W^{(2)} u)$

$h \quad h \times d \quad d \qquad h \quad h \times h \; h$

$$u_i = \langle W^{(1)}_{(i)}, x \rangle + b_i = \sum_{j=1}^{d} x_j \cdot W^{(1)}_{ij} + b_i$$

$$\sigma(v) = \begin{bmatrix} \sigma(v_1) \\ \sigma(v_2) \\ \vdots \end{bmatrix}$$

$$f = w^T \sigma(W^{(2)} \sigma(W^{(1)} x))$$

$1 \times h \quad h \times h \quad h \times d$

Nonlinear Activations!

$\sigma(z) = z$ linear

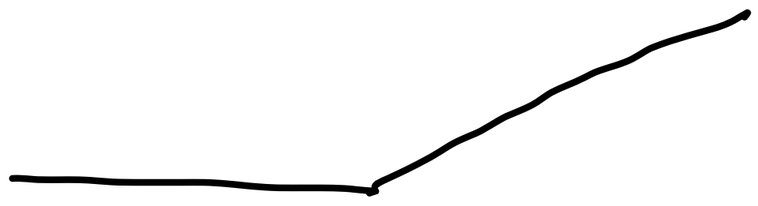
$$\sigma(z) = \frac{\exp(z)}{\sum_{z'} \exp(z')}$$ softmax

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

$\sigma(z) = \max(0, z)$ ReLU

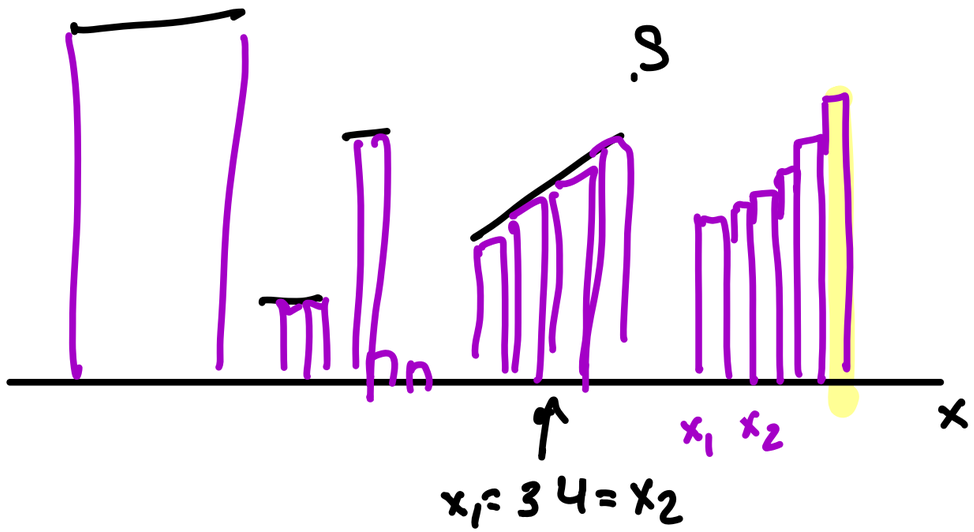
$$\sigma(z) = \text{step}(z) = \begin{cases} 1 & \text{if } z > 0 \\ -1 & \text{else} \end{cases}$$

$\sigma : \mathbb{R} \to \mathbb{R}$

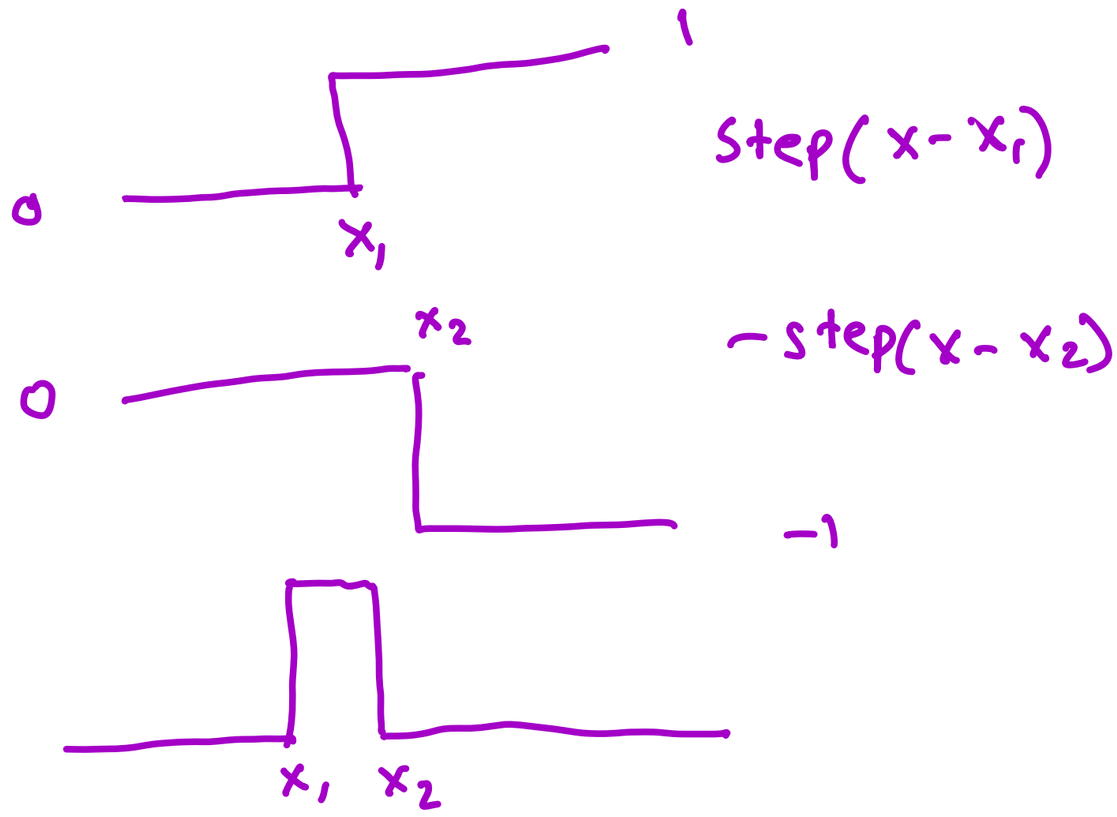


Do neural networks fit everything?

How about this function?

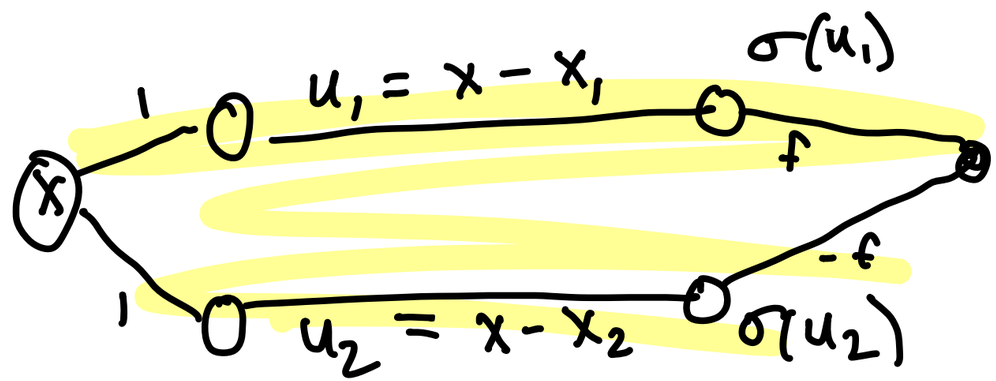


$$step(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{else} \end{cases}$$



$step(x - x_1) - step(x - x_2)$

$\sigma = step$



$f(\sigma(u_1) - \sigma(u_2))$

Good models exist.

How do we find good models?

Gradient Descent!

$\mathcal{L}(w)$

want Δ so that

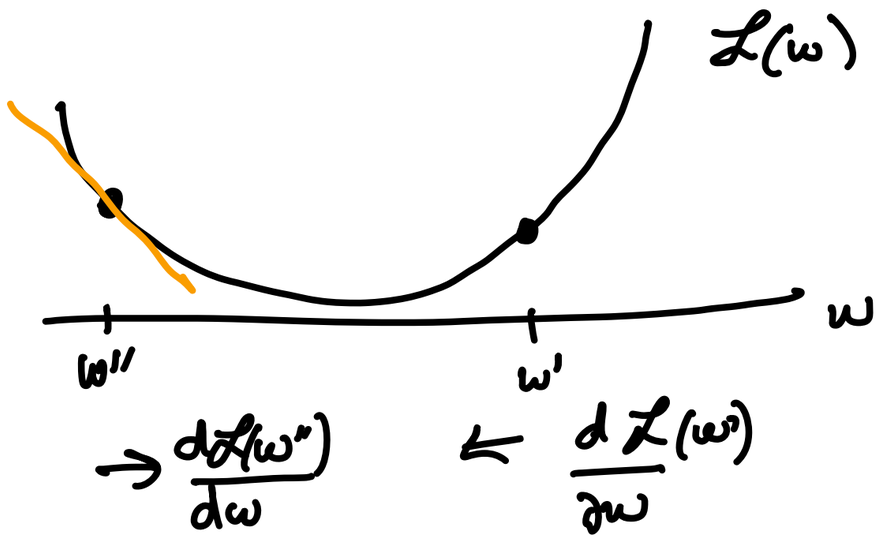
repeat {
$\mathcal{L}(w+\Delta) < \mathcal{L}(w)$

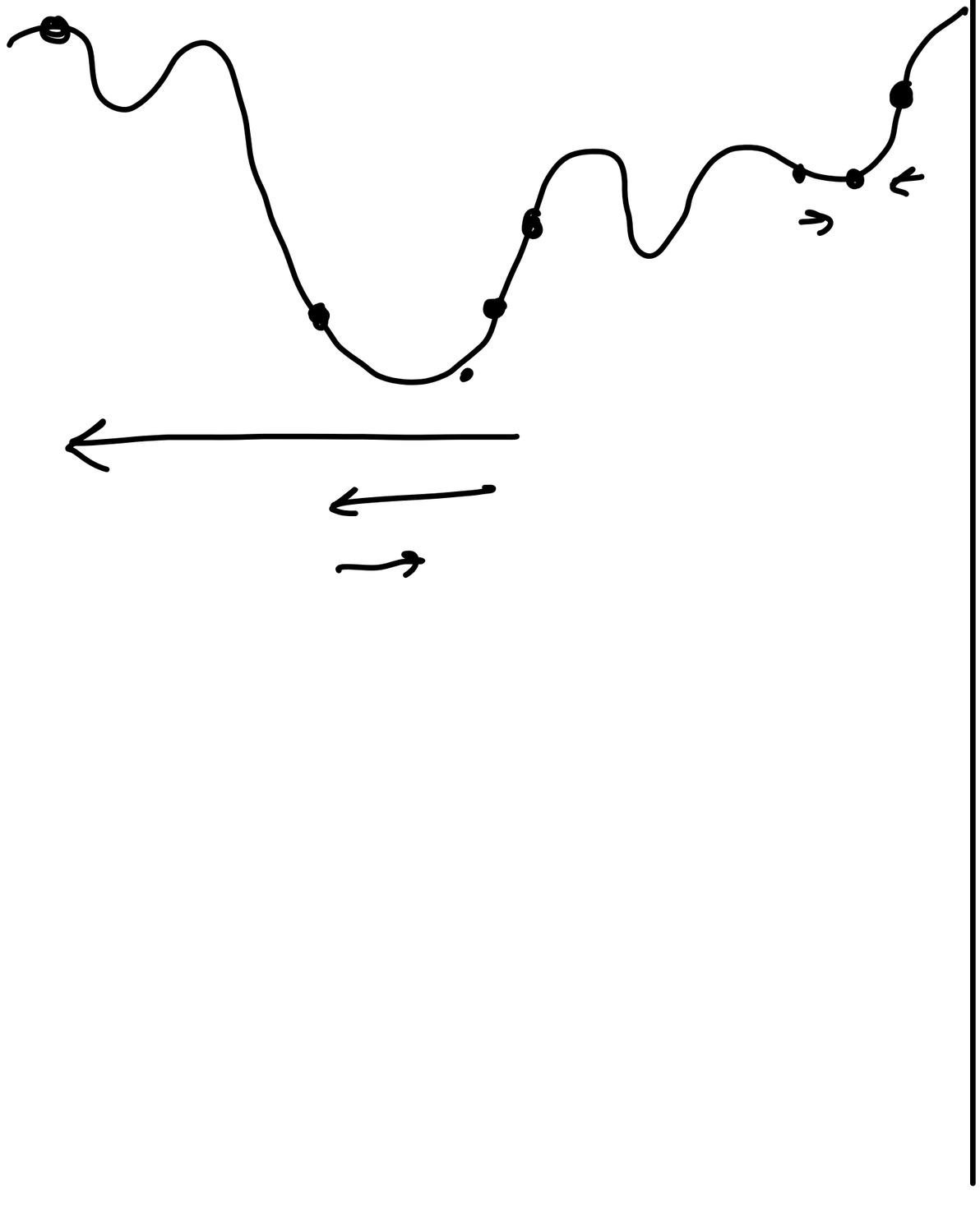
$w \leftarrow w + \Delta$
}

$$\nabla \mathcal{L}(w) = \begin{bmatrix} \frac{d\mathcal{L}(w)}{dw_1} \\ \vdots \end{bmatrix} \quad w \in \mathbb{R}^d$$

$$w \leftarrow w - \alpha \nabla \mathcal{L}(w)$$

α: learning rate



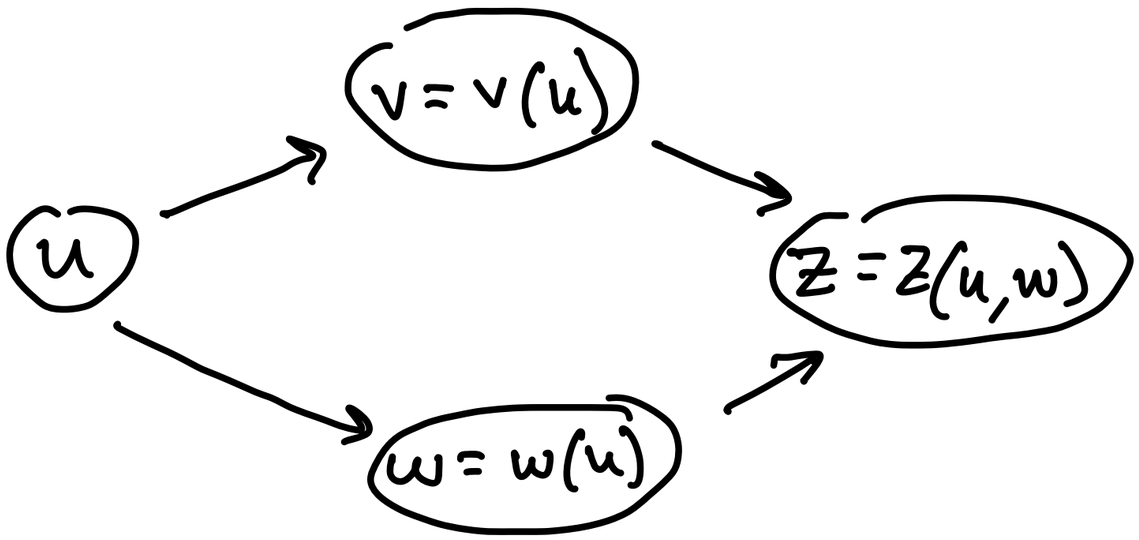


$$z = wx+b \qquad f(z) = \sigma(z)$$

$$\mathcal{L}(w,b) = \frac{1}{2}\left(y - \sigma(wx+b) \right)^2 + \lambda w^2$$

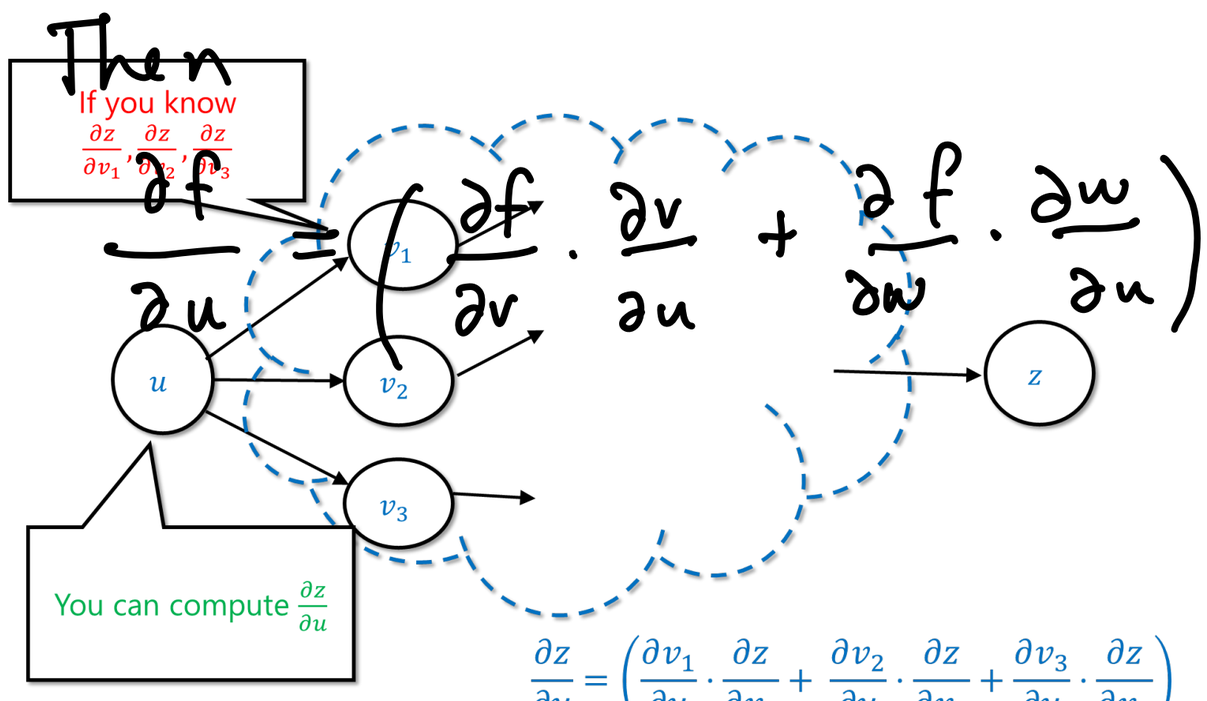
$$\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{2}\cdot 2 \left(y - \sigma(wx+b) \right) \cdot \sigma'(wx+b) \, x + 2\lambda w$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{2}\cdot 2 \left(y - \sigma(w+b) \right) \sigma'(wx+b) \cdot$$



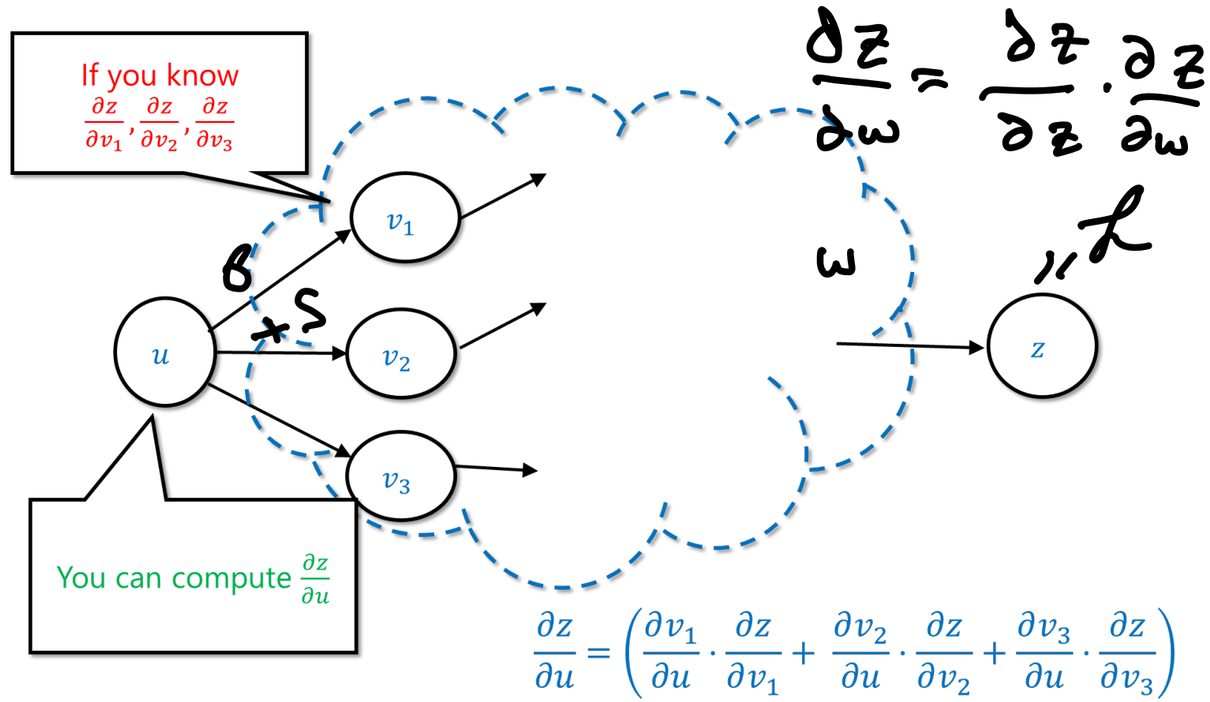
$$f(u) = z(v(u), w(u))$$

Then



$$\frac{\partial f}{\partial u} = \left(\frac{\partial f}{\partial v} \cdot \frac{\partial v}{\partial u} + \frac{\partial f}{\partial w} \cdot \frac{\partial w}{\partial u}\right)$$

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v_1}{\partial u} \cdot \frac{\partial z}{\partial v_1} + \frac{\partial v_2}{\partial u} \cdot \frac{\partial z}{\partial v_2} + \frac{\partial v_3}{\partial u} \cdot \frac{\partial z}{\partial v_3}\right)$$



$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial z} \cdot \frac{\partial z}{\partial w}$$

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v_1}{\partial u} \cdot \frac{\partial z}{\partial v_1} + \frac{\partial v_2}{\partial u} \cdot \frac{\partial z}{\partial v_2} + \frac{\partial v_3}{\partial u} \cdot \frac{\partial z}{\partial v_3}\right)$$

Forward pass

for each node:

compute v_i using Parents(v_i)

Backward

for each node u:

$$\frac{\partial \mathcal{L}}{\partial u} = \sum_{v \in \text{children}(u)} \frac{\partial \mathcal{L}}{\partial v} \cdot \frac{\partial v}{\partial u}$$

Forward pass

for each node :

compute v_i using Parents(v_i

Backward

for each node u:

$$\frac{\partial \mathcal{L}}{\partial u} = \sum_{v \in \text{children}(u)} \frac{\partial \mathcal{L}}{\partial v} \cdot \frac{\partial v}{\partial u}$$

$$v = u^2$$

$$\frac{dv}{du} = 2u$$

assuming $\frac{d\mathcal{L}}{\partial v}$ is known

then $\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial \mathcal{L}}{\partial v} \cdot \frac{\partial v}{\partial u}$