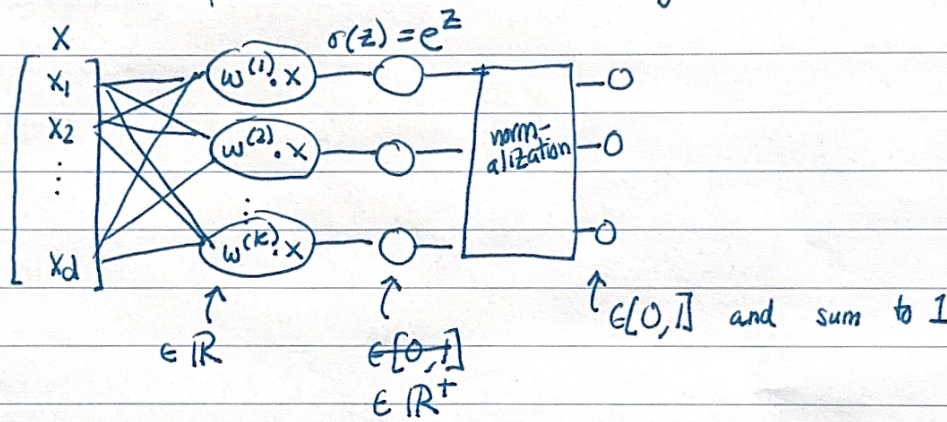
Neural Networks & Gradient Descent

Plan:
- Review
- Neural Networks
- Gradient Descent
- Back propagation

Logistics:
- zoom :)
- scribed notes (moved by 2)

Review

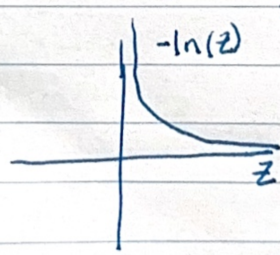
① Model $f : \mathbb{R}^d \to [0,1]^k$ $y \in \{0,1\}^k$



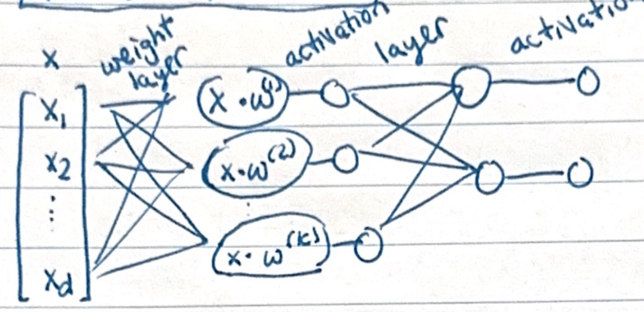
② Loss : Cross Entropy

$$H(y, f(x)) = -\sum_{i=1}^{k} y_i \log f(x)_i = -\log f(x)_{i^*}$$

+ efficient + big gradient if $f(x)_{i^*} \approx 0$



Neural Networks:



Activation functions

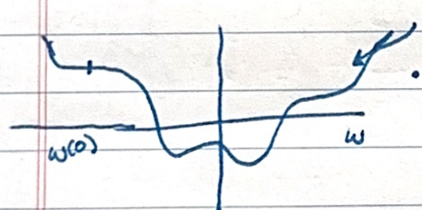
sigmoid	$\frac{1}{1+e^{-z}}$
ReLU	$\max(0, z)$
step	$\mathbb{1}[z>0]$
sign	$\mathbb{1}[z>0] - \mathbb{1}[z<0]$

$$\underset{k \times d}{\overset{W}{\begin{bmatrix} & & \\ & & \end{bmatrix}}} \underset{d \times 1}{\overset{x}{\begin{bmatrix} \\ \end{bmatrix}}} = \begin{bmatrix} w^{(1)} x \\ w^{(2)} x \\ \vdots \\ w^{(k)} x \end{bmatrix}$$

Layers

- ↳ Dense / fully connected
- ↳ Convolutional : images or spatial data
- ↳ Recurrent : sequential data
- ↳ Residual : processing technique
- ↳ Attention : sequential data with short + long dependencies

Gradient Descent:



- Initialize $w^{(0)}$
- Iteratively improve loss

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \nabla_w \mathcal{L}(w)$$

- small α "guarantees" improvement but slowly
- big α moves quickly but can overshoot

$\boxed{\text{Activity:}}$

Gradient descent on linear regression

$w^{(t+1)} \leftarrow$ ______

Time complexity of optimal solution:

Time complexity of one update:

$\boxed{\text{Stochastic Gradient Descent:}}$

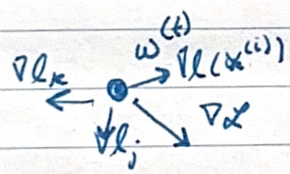
expensive! fit in memory?

$$\mathcal{L}(w) = \frac{1}{n} \sum_{i=1}^{n} l_i(x^{(i)}, y^{(i)}, w)$$

$$S \subseteq [n]$$

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \frac{1}{|S|} \sum_{i \in S} \nabla_w l_i(x^{(i)}, y^{(i)}, w)$$

Trade off: speed for "representativeness"



Back propagation

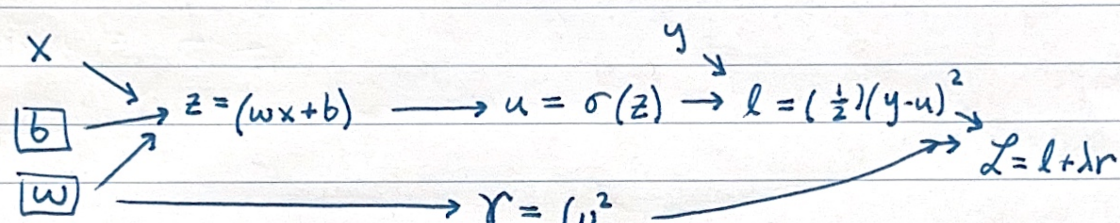
regularization on size of w ≈ "complexity"

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

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

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

Redundancy! Complicated!



Forward

$\Longrightarrow$

for $i \in \{1, \dots, N\}$:
compute v_i as function of parents

Backward

$\Longleftarrow$

$$\frac{\partial \mathcal{L}}{\partial \mathcal{L}} = 1$$

$$\frac{\partial \mathcal{L}}{\partial l} = 1$$

$$\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial \mathcal{L}}{\partial l} \cdot \frac{\partial l}{\partial u} = 1 \cdot \frac{2}{2}(-y+u)$$

$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial u} \cdot \frac{\partial u}{\partial z} = (u-y)\,\sigma'(z)$$

$$\frac{\partial \mathcal{L}}{\partial b} = (u-y)\,\sigma'(z)$$

$$\frac{\partial \mathcal{L}}{\partial r} = \lambda$$

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial r}\frac{\partial r}{\partial w} + \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w} = (u-y)\,\sigma'(wx+b)\,x + \lambda 2w$$

+ computed once
+ structured
+ modular

for $i \in \{N, \dots, 1\}$:
compute $\displaystyle \frac{\partial \mathcal{L}}{\partial v_i} = \sum_{j \in \text{children}(v_i)} \frac{\partial \mathcal{L}}{\partial v_j} \frac{\partial v_j}{\partial v_i}$