# Class Plan

Recap

Reminders

GPU

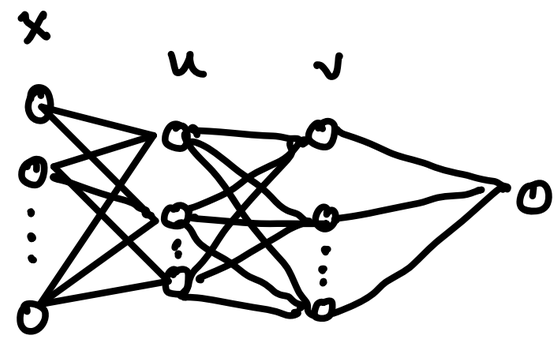
Variants of Gradient Descent

Initialization

Generalization

## Neural Network

1)




$f = w^T v$

$u = \sigma(Wx) \quad v = \sigma(W'u)$

2) $\mathcal{L}(w) =$ how bad our predictions are

3)

$$\nabla \mathcal{L}(w) = \begin{bmatrix} \frac{\partial \mathcal{L}(w)}{\partial w_i} \\ \\ \end{bmatrix}$$

$w \in \mathbb{R}^d$

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

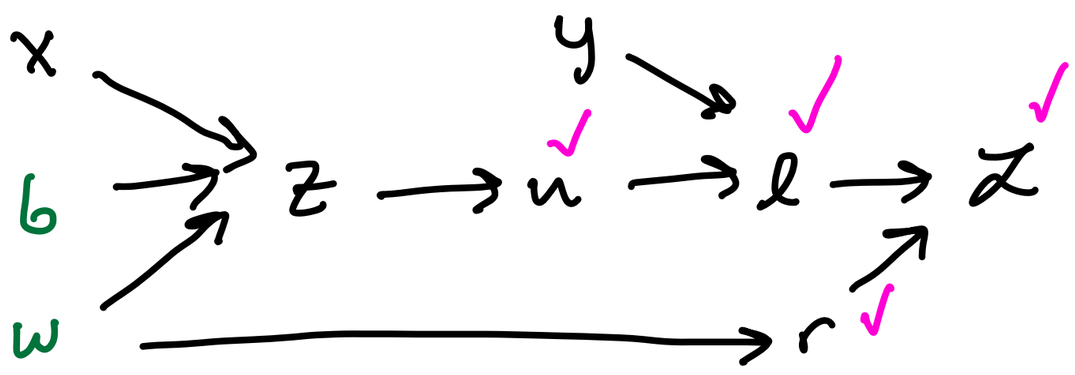
eta

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

$z = wx+b$ $\qquad r = w^2$

$u = \sigma(z)$ $\qquad \mathcal{L} = l + \lambda r$

$l = \frac{1}{2}(y-u)^2$




$\frac{\partial \mathcal{L}}{\partial b}$

$\frac{\partial \mathcal{L}}{\partial w}$

Forward pass

plug it in!

Backward pass

$\frac{\partial \mathcal{L}}{\partial \mathcal{L}} = 1 \qquad \frac{\partial \mathcal{L}}{\partial r} = \lambda \qquad \frac{\partial \mathcal{L}}{\partial l} = 1$

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

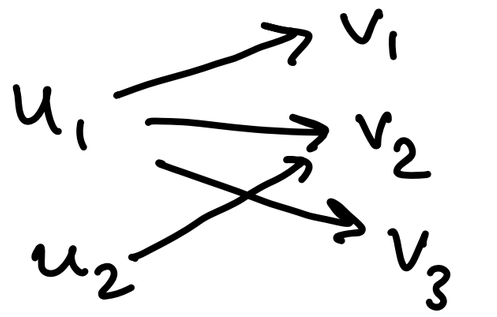
$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial u} \cdot \frac{\partial u}{\partial z} = \frac{\partial \mathcal{L}}{\partial u} \cdot \sigma'(z)$$

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial z}{\partial w} + \frac{\partial \mathcal{L}}{\partial r} \cdot \frac{\partial r}{\partial w}$$

$\frac{\partial z}{\partial w}$: $x$ $\qquad$ $\frac{\partial r}{\partial w}$: $2w$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial z}{\partial b}$$

$\frac{\partial z}{\partial b}$: $1$




$\rightarrow \mathcal{L}$

Forward pass →

$v_i = v_i(u_1, u_2)$

← Backward pass

$\frac{d\mathcal{L}}{dout} = out.grad$

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

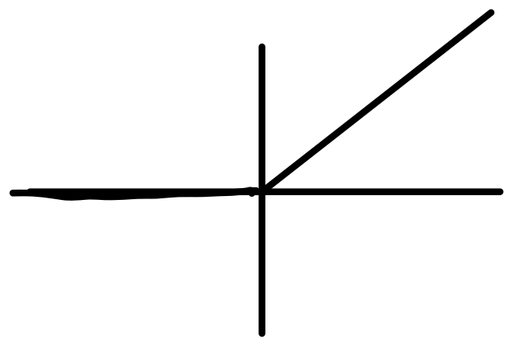
$\frac{\partial v}{\partial u}$ ← $\frac{d\,out}{d\,self}$

other →

u → v

self — out

| | out | $\frac{dout}{d\,self}$ |
|---|---|---|
| | forward | backward |
| add | self+other | 1 |
| mul | self*other | other |
| pow | self**other | other * self**(other-1) |
| relu | max(0, self) | 1 if self>0<br>0 else |
| sigmoid | | |



# Reminders

- Form: 23 / 26
  - ↳ timestamp
  - ↳ I will not respond individ.

- Homework due Friday 5
  - ↳ free questions
  - ↳ try LaTeX
    - ↳ solve before
  - ↳ update source

- Canvas discussion!

# GPU

video games

↳ parallel cores

↳ large peak memory

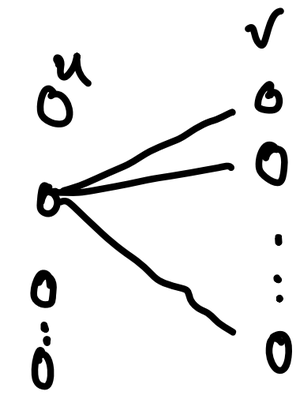
Forward "embarassingly parallel"

$$\sigma(Wu) = \begin{bmatrix} \sigma(\langle w^{(1)}, u \rangle) \\ \\ \end{bmatrix}$$

Backward "embarassingly parallel"

$$\frac{\partial \mathcal{L}}{\partial u} = \left[\frac{\partial \mathcal{L}}{\partial u_j}\right] = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \frac{\partial v_i}{\partial u_j} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \left[\frac{\partial \mathcal{L}}{\partial v_i}\right]$$

↑ vector




1 hr → 20 seconds

$S \sim$ sample

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{y}_i) \quad \text{vs.} \quad \frac{1}{|S|}\sum_{i \in S} \ell(y_i, \hat{y}_i)$$

$\mathcal{L}(w)$ $\quad$ $\widetilde{\mathcal{L}}(w)$

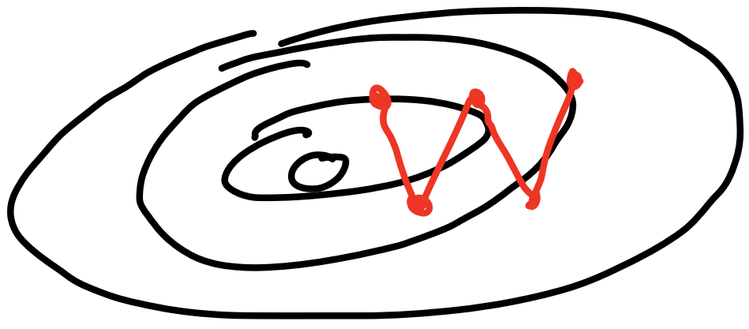
$\nabla \widetilde{\mathcal{L}}(w)$

# Variants of SGD

$$w_{t+1} \leftarrow w_t - \eta \nabla \mathcal{L}(w_t)$$

Problems:

- oscillate a lot
- get stuck



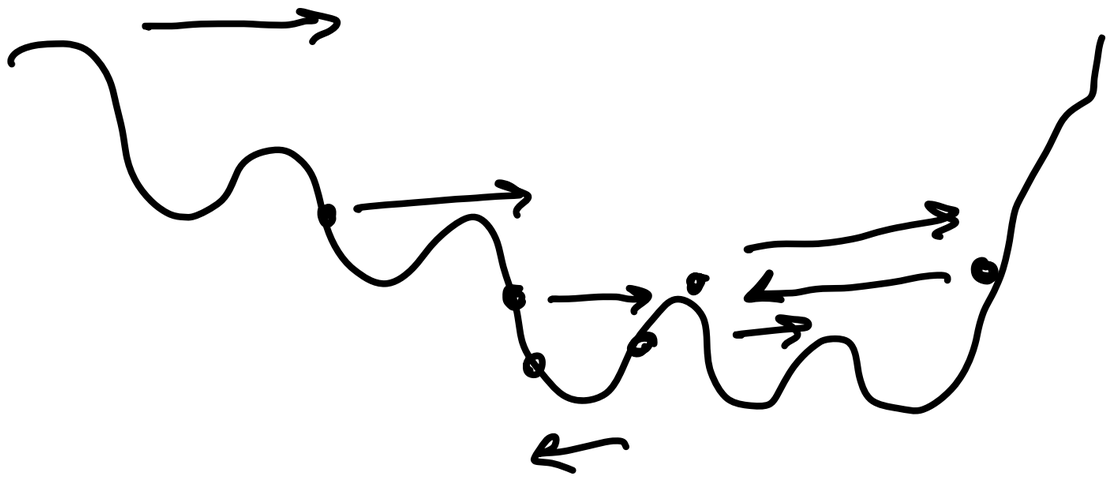
## Momentum

keeps memory

$$v_t \leftarrow \mu v_{t-1} + (1-\mu) \nabla \mathcal{L}(w_t)$$

"mu"

$$w_{t+1} \leftarrow w_t - \eta v_t$$



Adaptive Learning Rate

$$g_t \leftarrow \nabla \mathcal{L}(\omega)$$

$$s_t \leftarrow s_{t-1} + \| g_t - \text{mean}(g_t) \|_2^2$$

$$\omega_{t+1} \leftarrow \omega_t - \eta \frac{1}{\sqrt{s_t} + \epsilon} g_t$$

Adam (Adaptive Moment Estimate)

$$v_t \leftarrow \frac{\mu v_{t-1} + (1-\mu) g_t}{1-\mu^t} \leftarrow \text{Idk}$$

$$s_t \leftarrow \frac{\beta s_{t-1} + (1-\beta) \| g_i - \text{mean}(g_i) \|_2^2}{1-\beta^t}$$

($\beta$: "beta")

$$\omega_{t+1} \leftarrow \omega_t - \frac{\eta}{\sqrt{s_t} + \epsilon} v_t$$

Both!!

# Initialization

- vanishing/exploding gradient

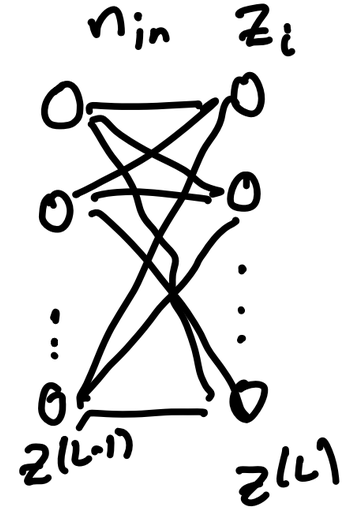
$$f_w(x) = W_L W_{L-1} \cdots W_1 x$$

ignoring activations

$$\mathcal{L}(w) = \frac{1}{2} \|f_w(x) - y\|_2^2$$

$$\frac{\partial \mathcal{L}(w)}{\partial w_n} = \left(f(w(x)-y) W_L W_{L-1} \cdots W_{r+1} W_{r-1} \cdots W_1 x\right)$$

✓ • symmetry




✓ randomly initialize

from distribution with "good" norm

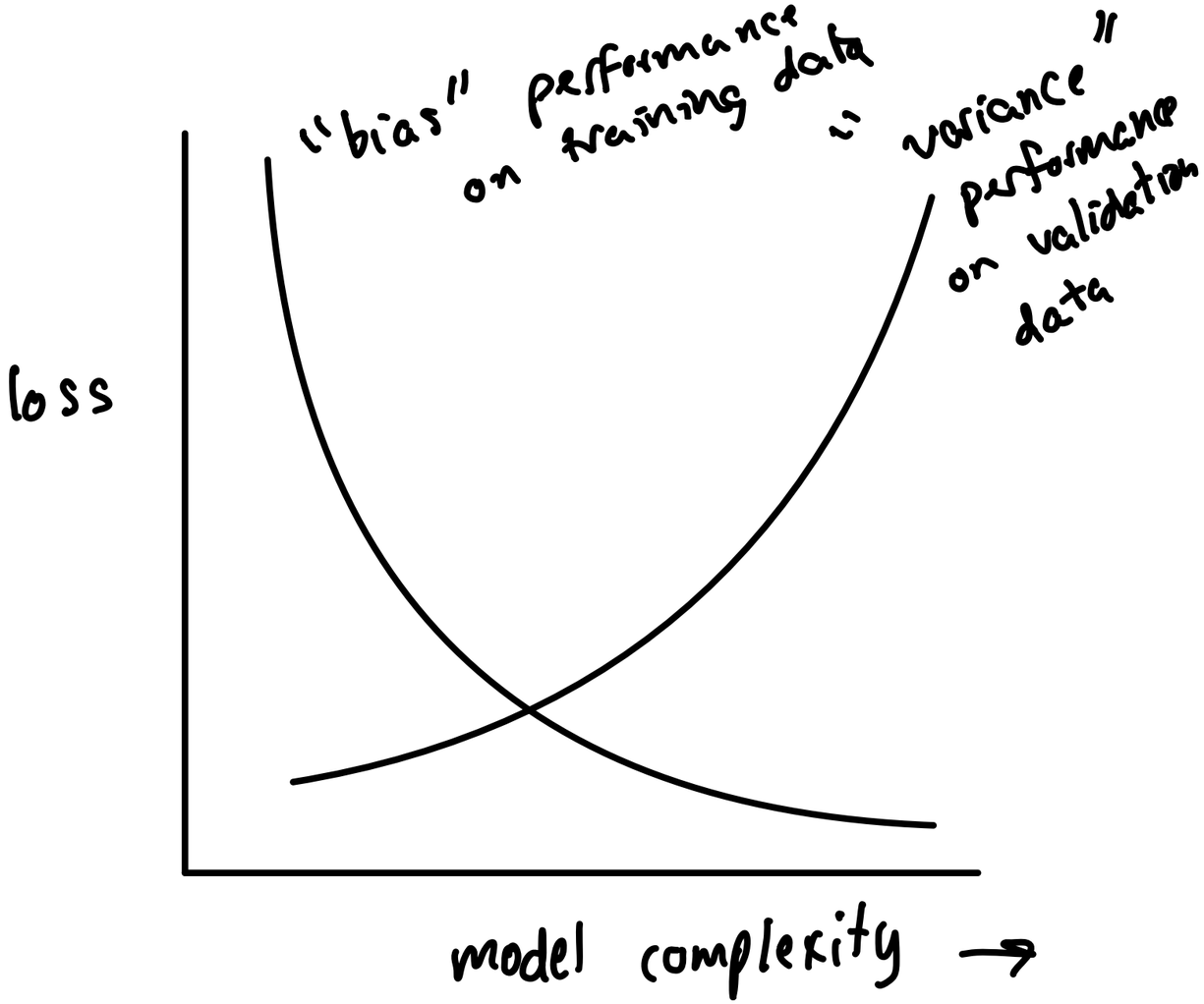
$$z_i^{(l)} = \sum_{j=1}^{n_{in}} w_{ij}^{(l)} \cdot z_j^{(l-1)}$$

$$\mathrm{var}(z_i^{(l)}) = \underbrace{n_{in} \cdot \mathrm{var}(w_{ij})}_{\approx 1} \cdot \mathrm{var}(z_i^{(l-1)})$$

$$\mathrm{var}(w_{ij}) = \frac{2}{n_{in} + n_{out}} \quad w_{ij} \sim N\left(0, \frac{2}{n_{in} + n_{out}}\right)$$

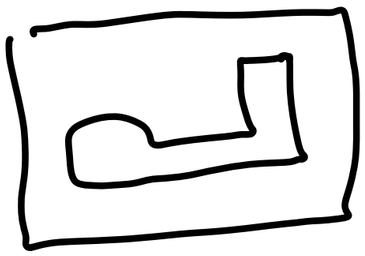
# Generalization

↳ want good performance on unseen data

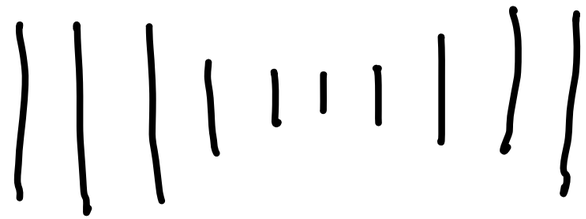



"Regularize"

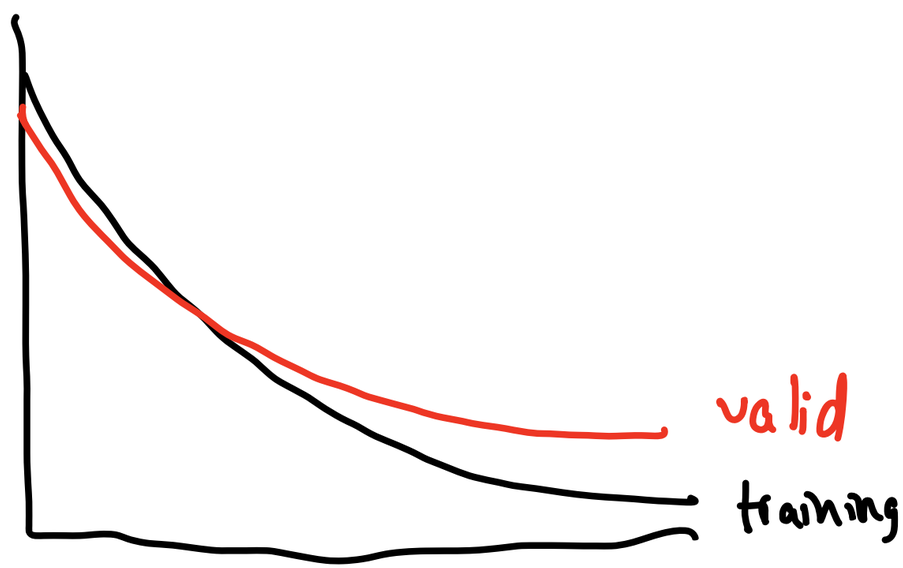
↑ reduce complexity



- bottleneck

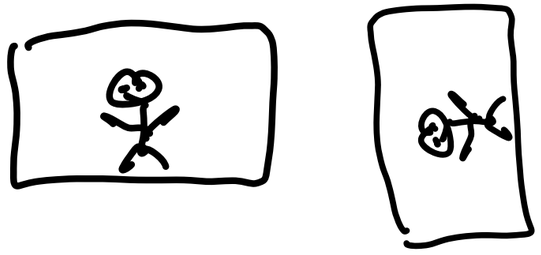


- Stop early

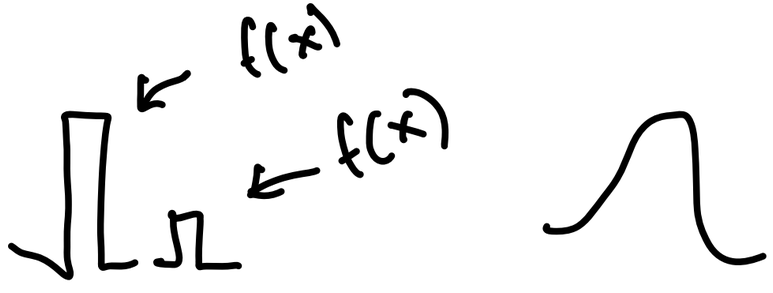


key: move from "memorization" to "learning"

- augment data



- weight decay $\mathcal{L}(w) + \lambda ||w||_2^2$




- dropout $\{0,1\}$

$v = m_i \sigma(u)$

- transfer learning

Hint

import torch.nn as nn

nn.Sequential

nn.Linear

nn.ReLU

nn.Softmax