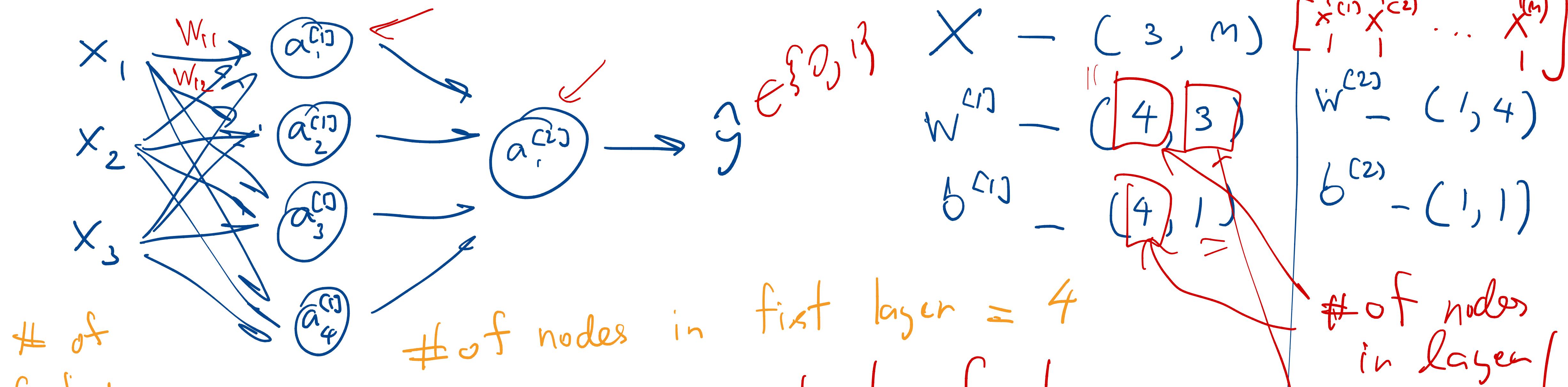


ICCS482 Deep Learning

Lecture 4: Neural Networks

Sunsern Cheamanunkul, Sep 16, 2020.



of feet
feel fine
= 3

The diagram illustrates a neural network layer with the following components:

- Input:** A blue arrow labeled x enters the layer.
- Forward Pass:** The input x is multiplied by a weight matrix $W^{[1]}$ and added to a bias vector $b^{[1]}$ to produce the pre-activations $z^{[1]}$. The formula is
$$z^{[1]} = W^{[1]} \cdot x + b^{[1]}$$
- Activation:** The pre-activations $z^{[1]}$ are passed through an activation function g to produce the activations $a^{[1]}$. The formula is
$$a^{[1]} = g(z^{[1]})$$
- Output:** The final output a is produced from the activations $a^{[1]}$.
- Backpropagation:** Red arrows indicate the flow of error gradients from the output layer back through the layer. One red arrow points from the output a towards the layer, and another points from the activation $a^{[1]}$ towards the pre-activations $z^{[1]}$.
- Dimensions:** The dimension of the output a is given as $(4, 3)$.
- Pre-activations:** The dimension of the pre-activations $z^{[1]}$ is given as $(4, 3)$.

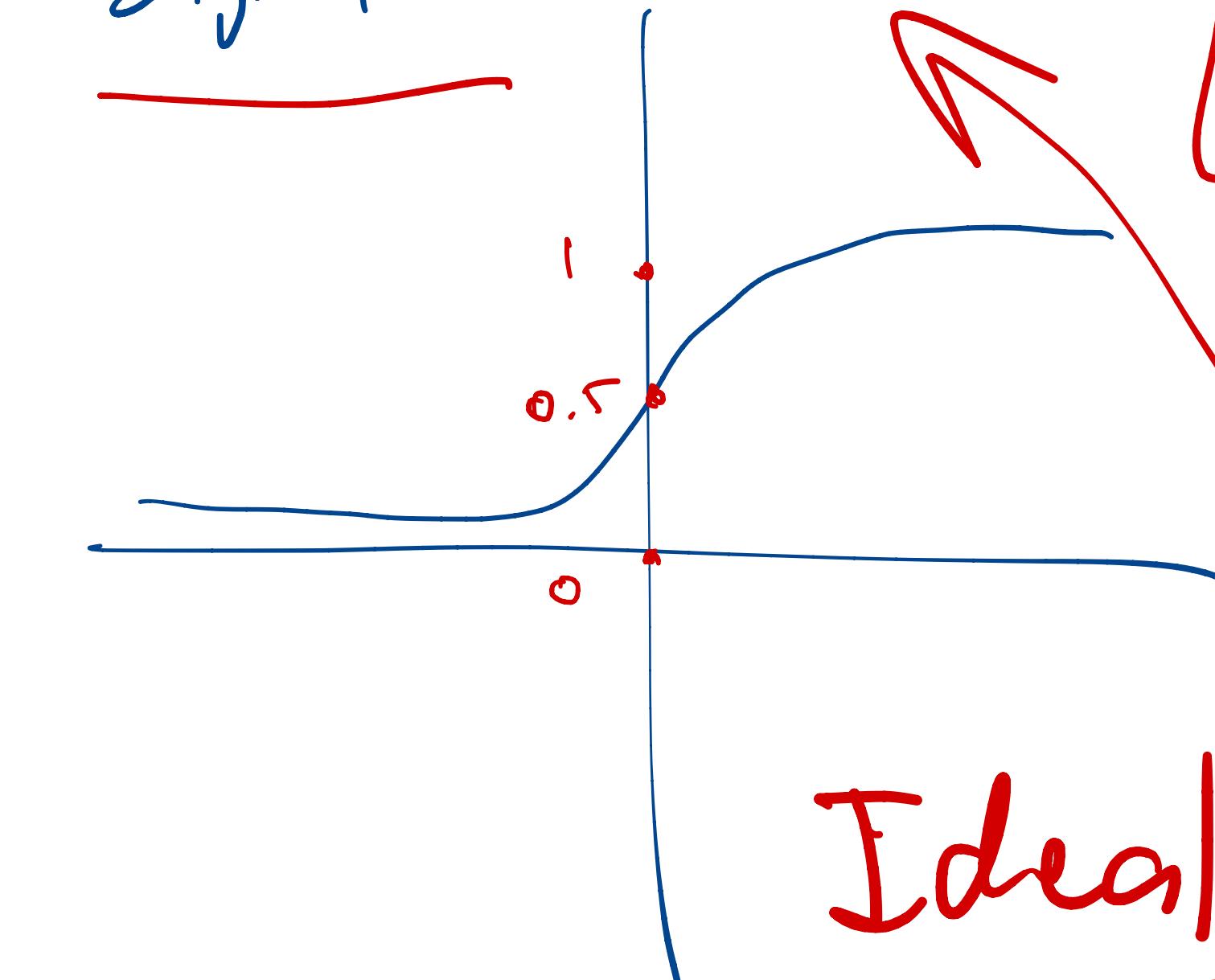
$$\begin{matrix} Z^{(1)} \\ \downarrow \\ Z^{(2)} \\ \downarrow \\ Z^{(3)} \end{matrix} - \begin{matrix} (4, M) \\ \downarrow \\ (4, N) \end{matrix} \left\{ \begin{array}{l} \text{Same} \\ \text{differ} \end{array} \right.$$

$$z^{(2)} = (1, \gamma)$$

$$a^{(2)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

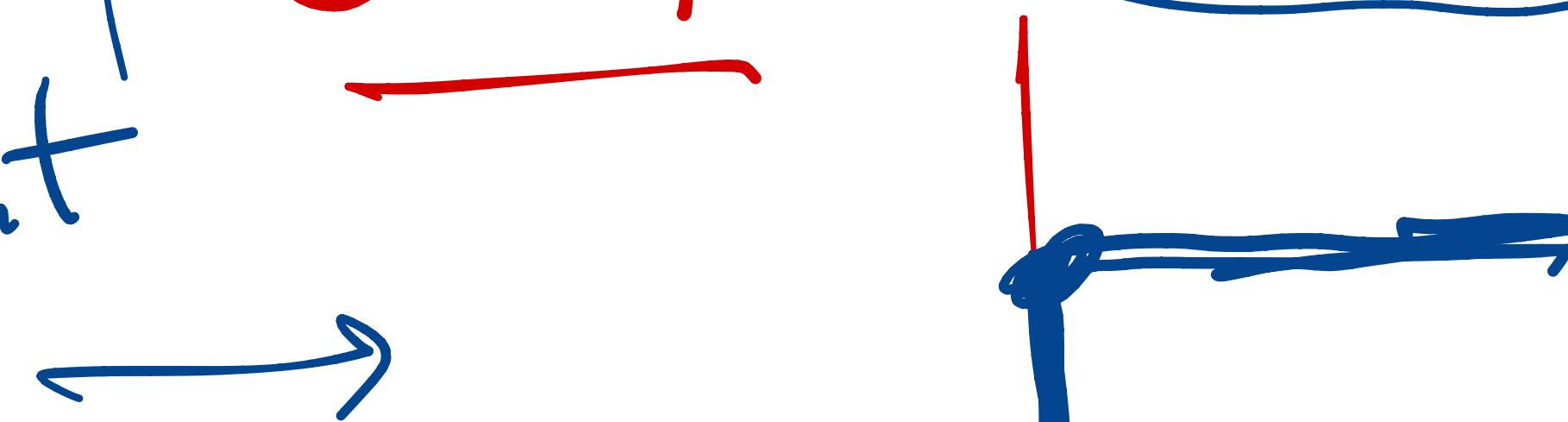
Activation Function

Sigmoid



Ideal

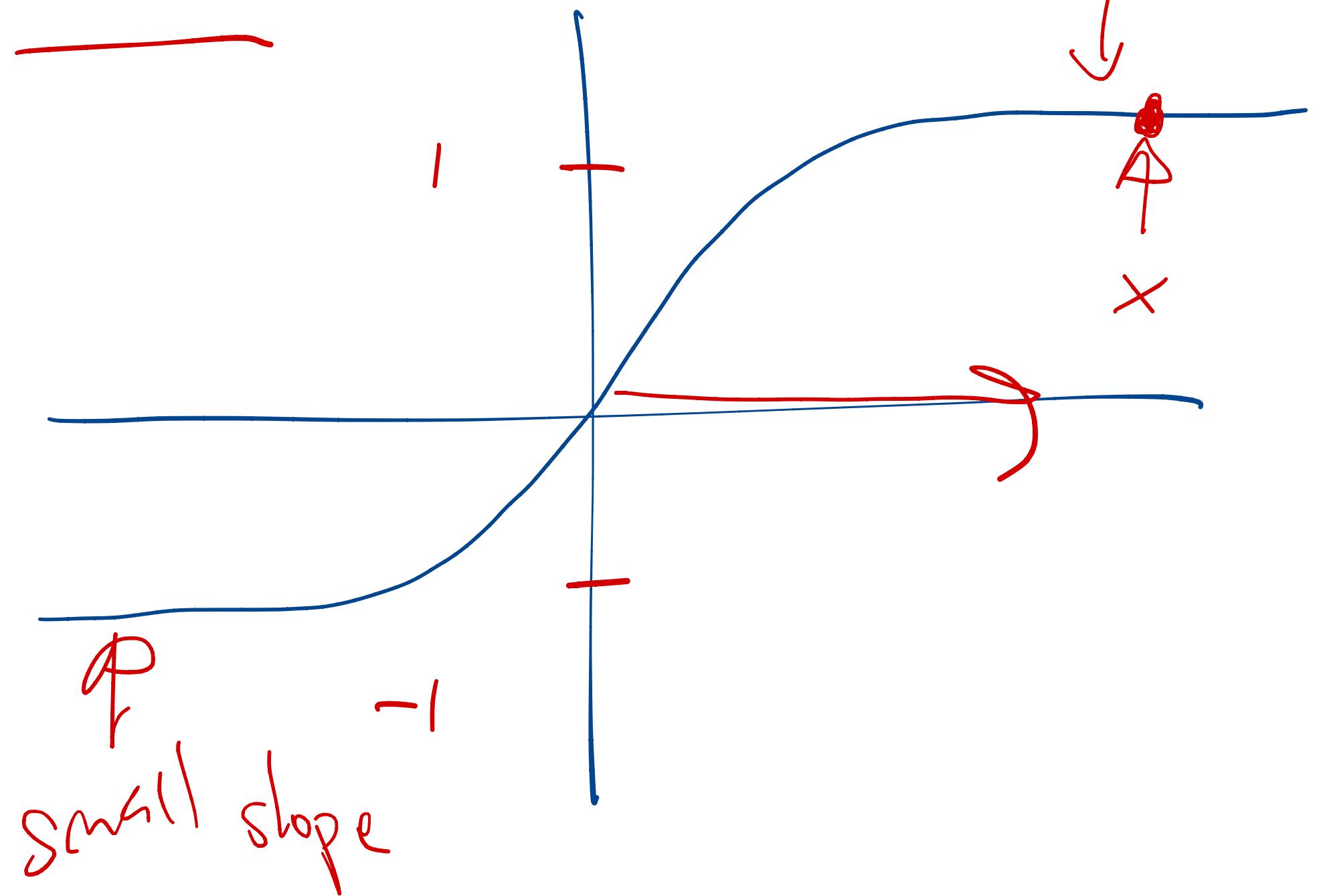
no gradient



not differentiable!

$$\begin{aligned}\sigma'(x) &= \frac{e^{-x}}{(1+e^{-x})^2} \\ &= \left(\frac{1}{1+e^{-x}}\right) \frac{e^{-x}}{1+e^{-x}} \\ &= \frac{1}{1+e^{-x}} \left(\frac{1-1+e^{-x}}{1+e^{-x}}\right) \\ &= \frac{1}{1+e^{-x}} \left(\frac{e^{-x}-1}{1+e^{-x}}\right) \\ &= \frac{1}{1+e^{-x}} \left(1 - \frac{1}{1+e^{-x}}\right) \\ &\approx \sigma(x)(1-\sigma(x))\end{aligned}$$

tanh



Small slope!

P
small slope

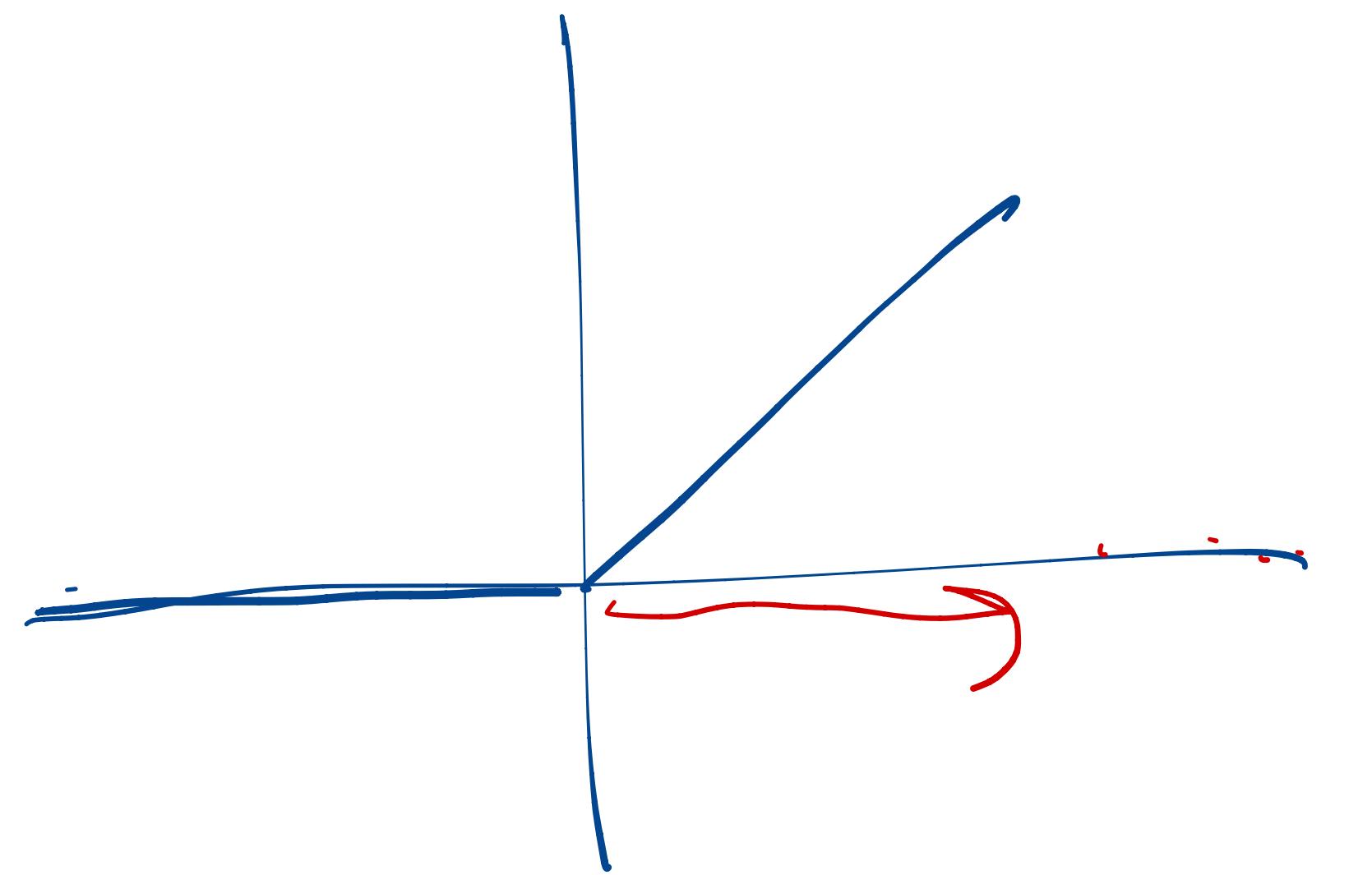
$$\begin{aligned} \tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ \frac{d \tanh(x)}{dx} &= \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} \\ &= \frac{1}{(e^x + e^{-x})^2} \\ &= 1 - (\tanh(x))^2 \end{aligned}$$

- ① centered around 0
- ② Symmetric about 0

ReLU - Rectified Linear Unit

$$w = w - \alpha \cdot \frac{dw}{\|dw\|}$$

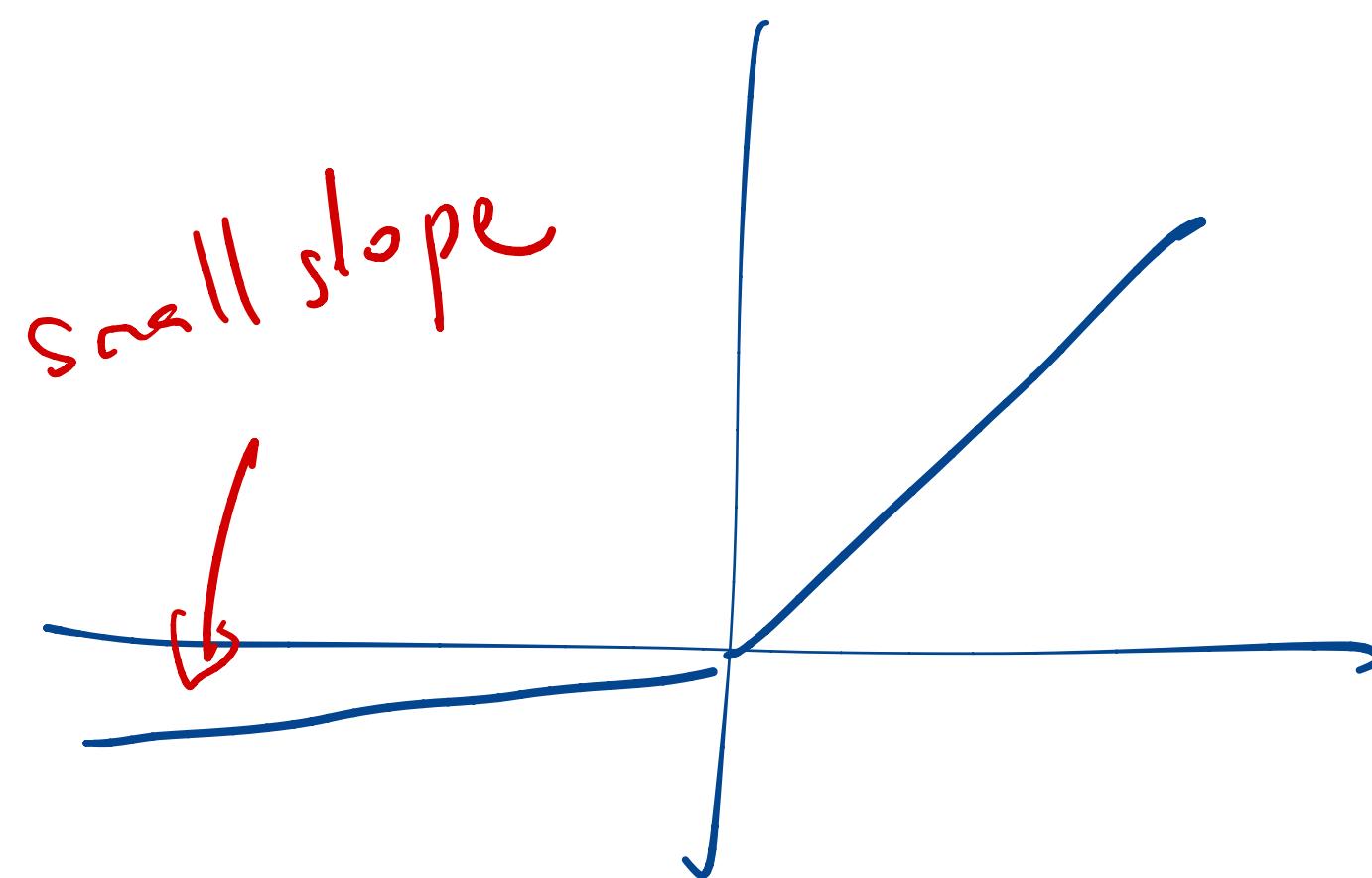
non-zero



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Leaky ReLU



$$g(z) = \max(0.01z, z)$$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Why do we need activation functions?

$$z^{[1]} = w^{[1]} \cdot x + b^{[1]}$$

$$\underline{a}^{[1]} = \underline{g}(z^{[1]})$$

$$z^{[2]} = w^{[2]} \cdot \underline{a}^{[1]} + b^{[2]} = (w^{[2]} \cdot w^{[1]}) \cdot x + (b^{[2]} + b^{[1]})$$

$$\underline{a}^{[2]} = \underline{g}(z^{[2]})$$

NN

algorithms

① Define model structure

model complexity
bias vs. variant

② Initialize model params

③ Loop:

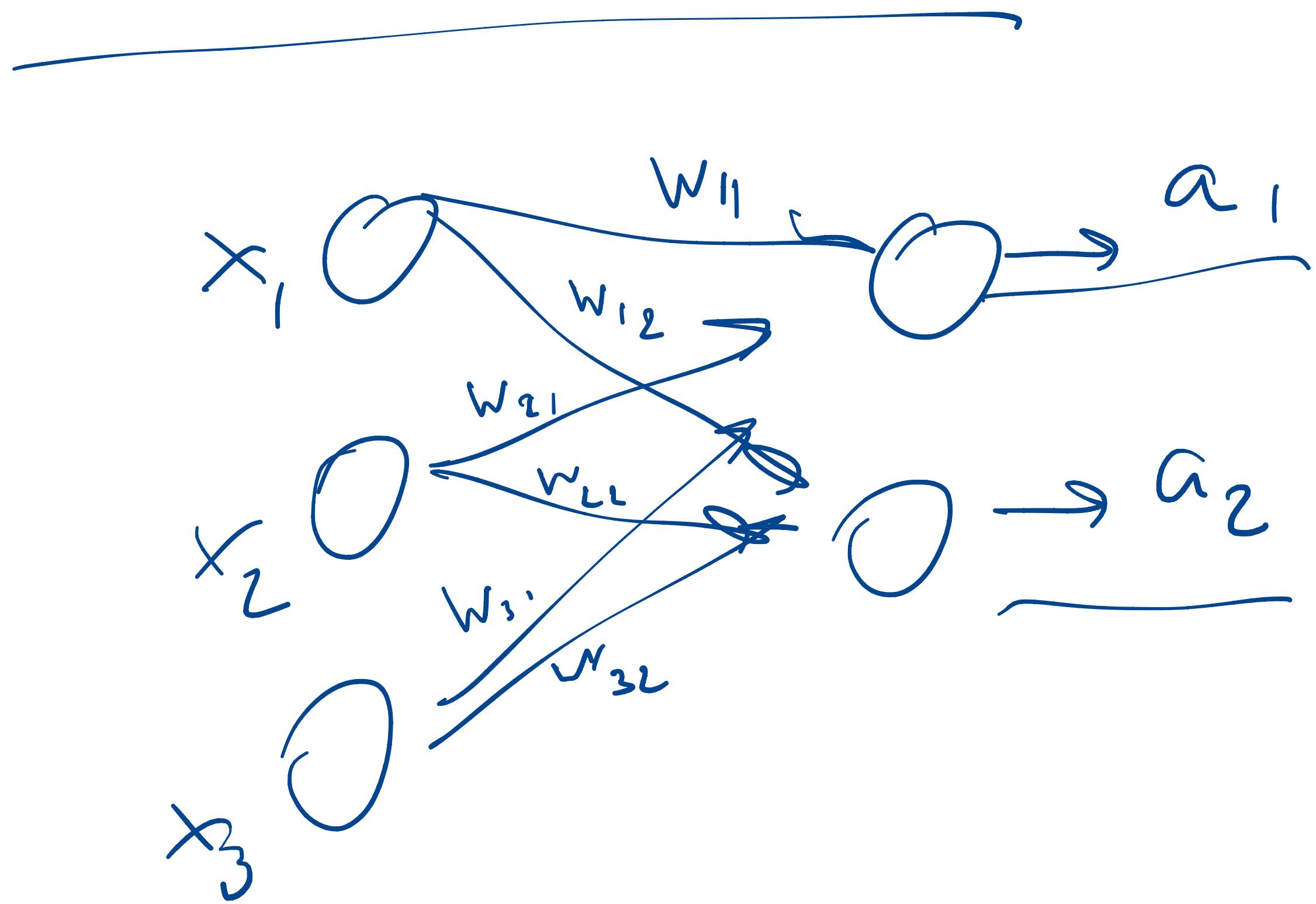
3.1) Calc loss (forward prop)

3.2) Calc derivatives (backward prop)

3.3) Apply update rule (GD)

④ Win!

Model Initialization



Idea ①

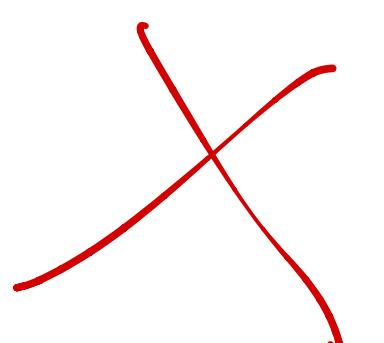
$$w_{11} = w_{12} = w_{21} = w_{22} + \epsilon > 0$$

update

$$w_{11} = w_{21}$$

$$w_{12} = w_{22}$$

⋮

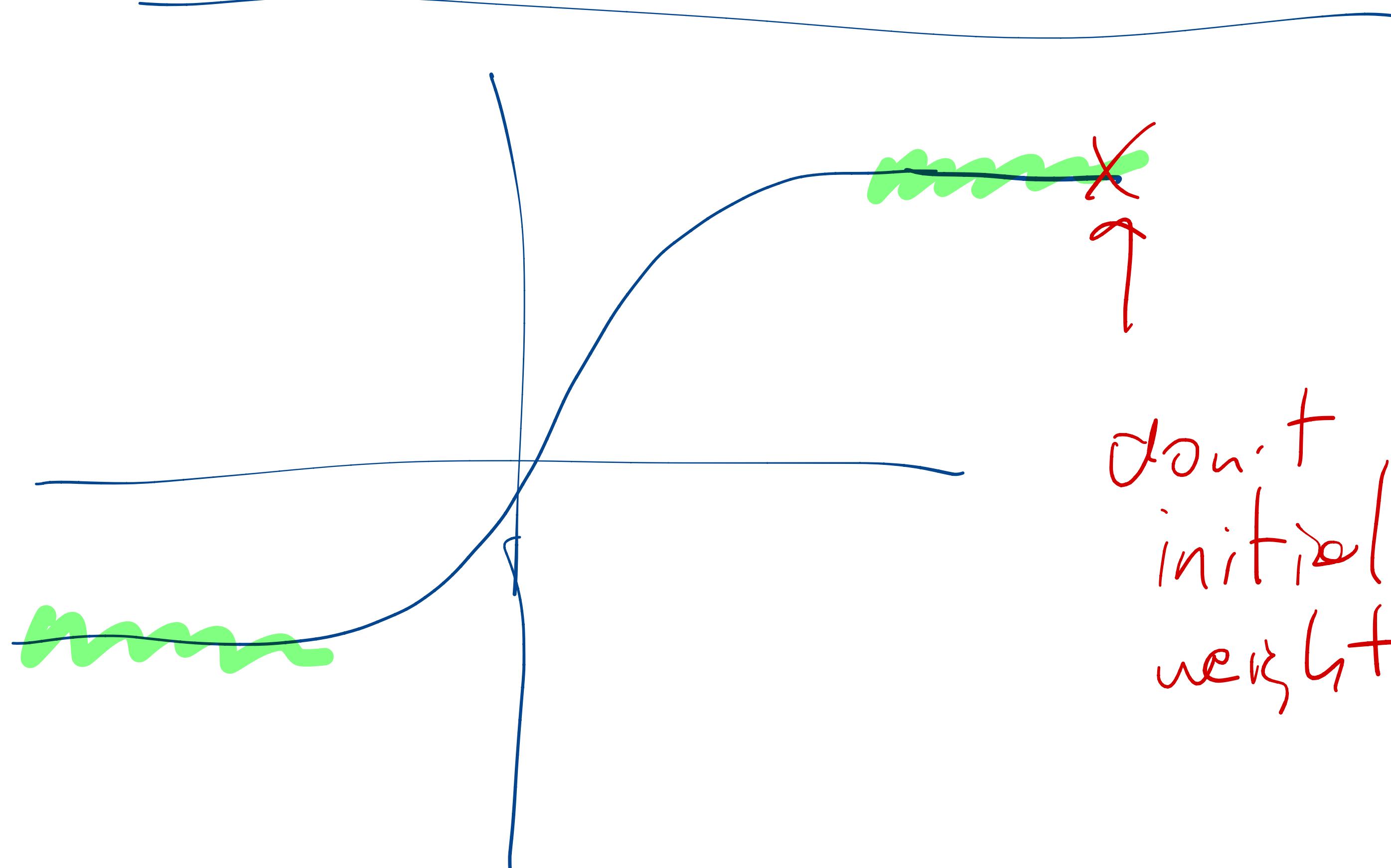


No!

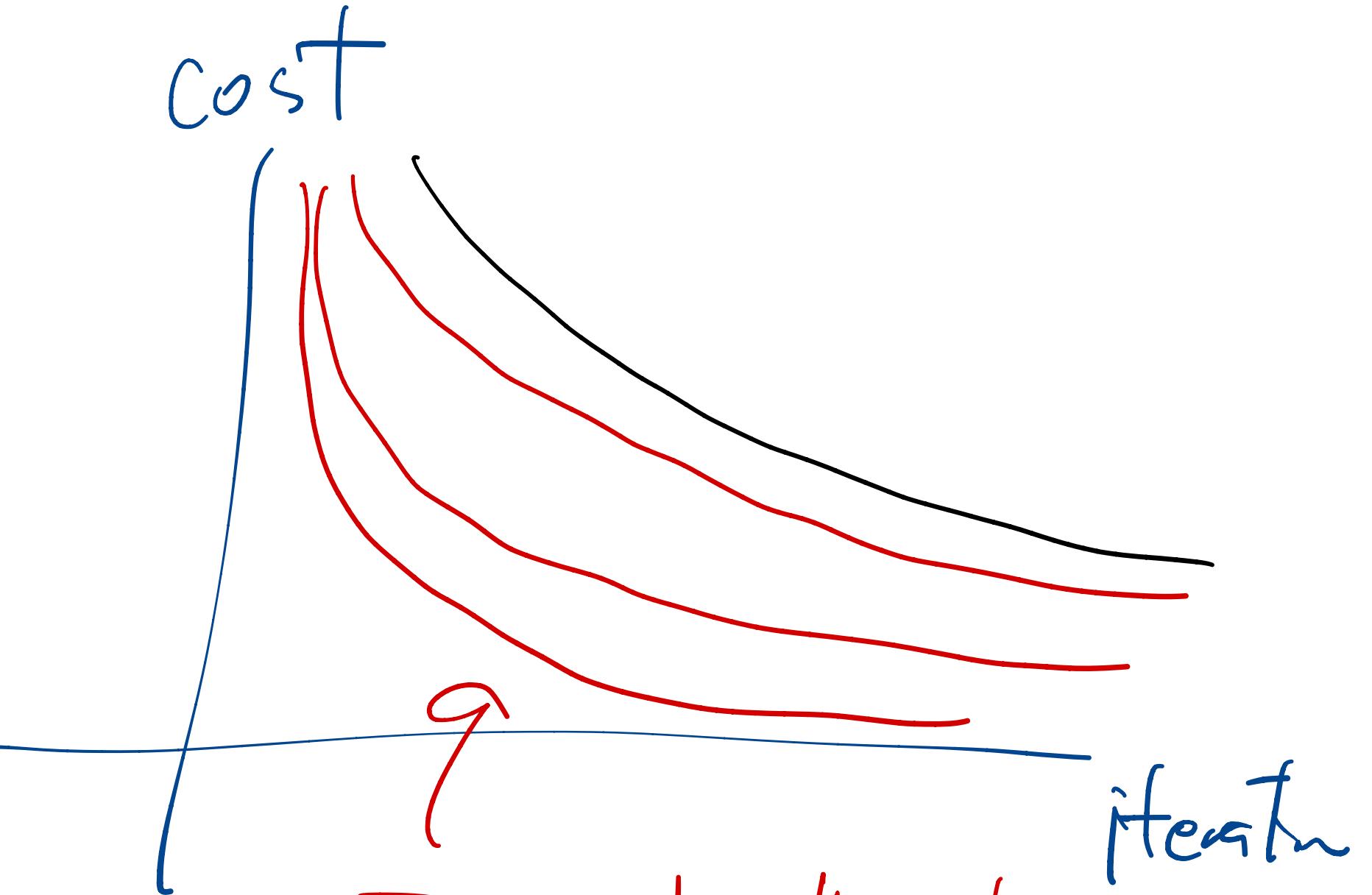
Idea ②

initialize weights
randomly!

Can we use large w?



Don't initialize weight here.



No!, you need to initialize with relatively small weight

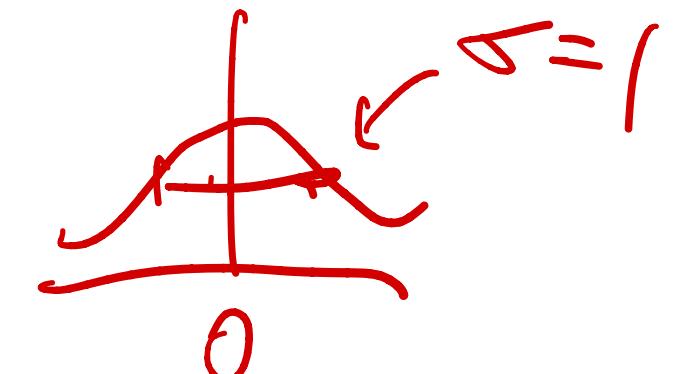
$w = np.random.rand(n)$ (shape) $\star 0.01$ ~~σ~~

← standard normal distributions

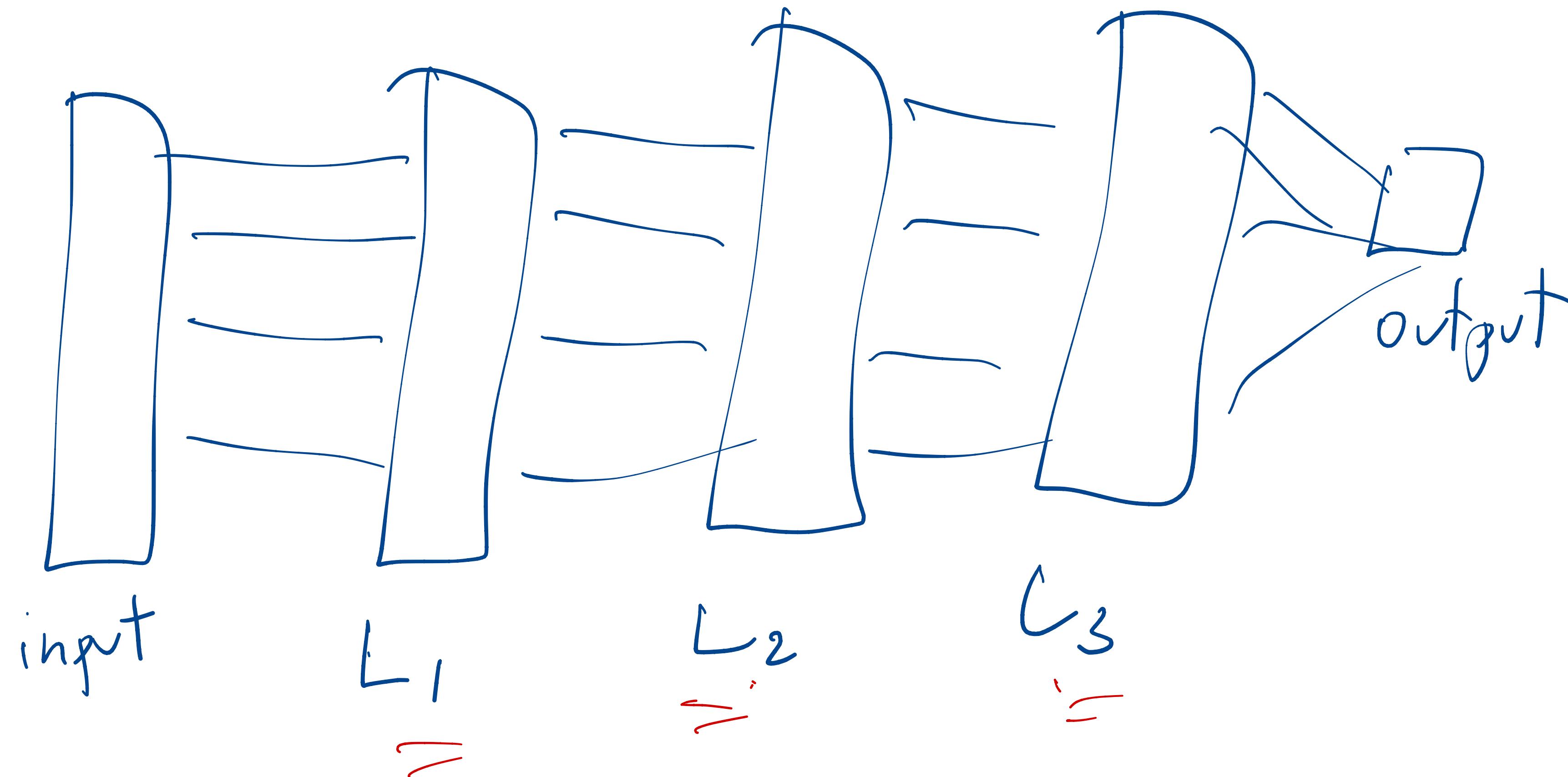
$w_z = np.random.rand(n)$ (shape) $\star 0.01$

$\sim \text{Uniform}(0, 1)$

~~σ~~



Deep Network



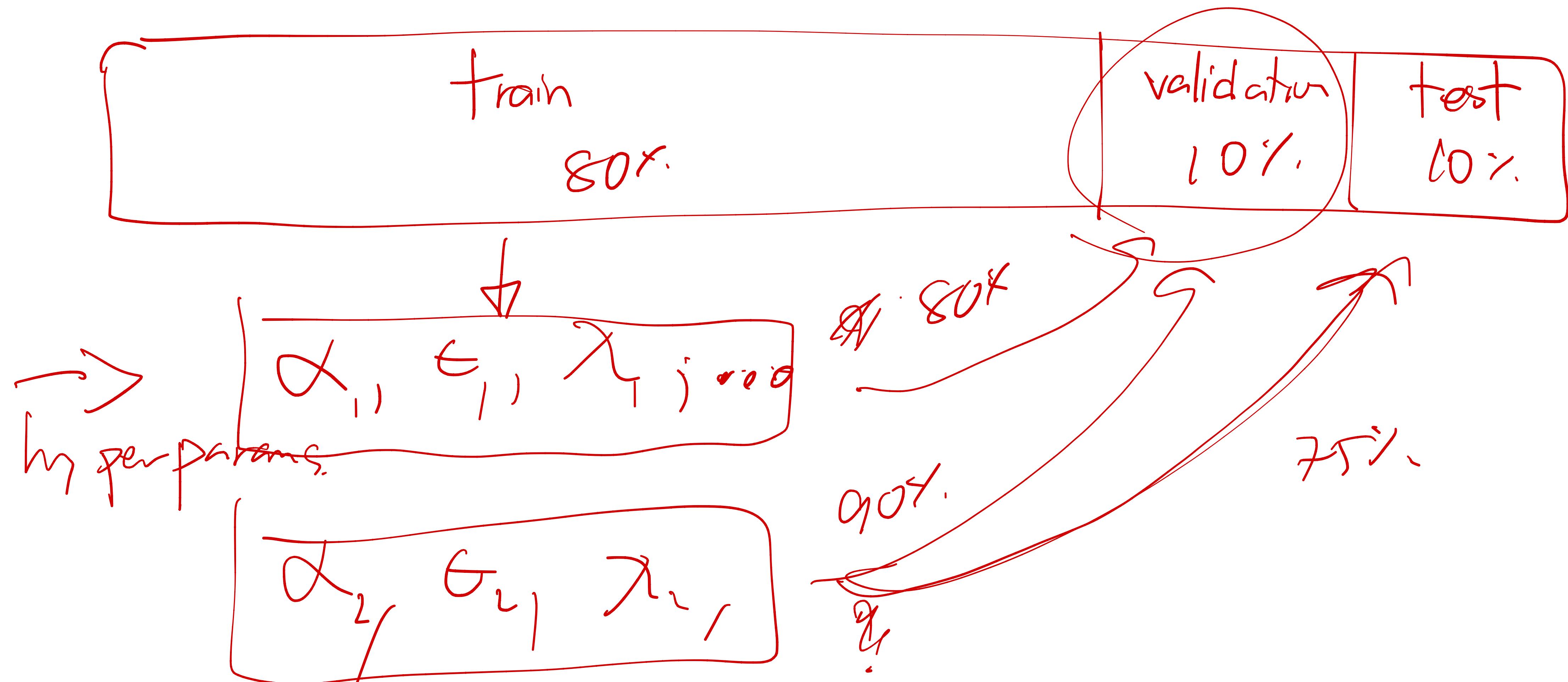
NN params: W's and b's
figured out by GD

Hyperparams:

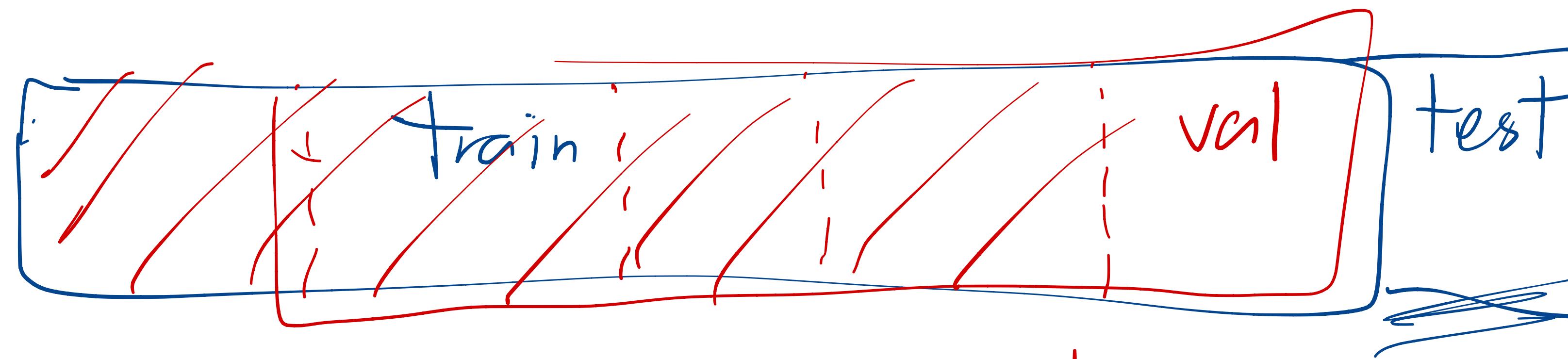
- learning rate
- # of hidden nodes
- # of hidden layers
- # of training iterations
- choice of activation functions
- etc.

} need experiment
using validation
set

entire dataset



Cross-validation

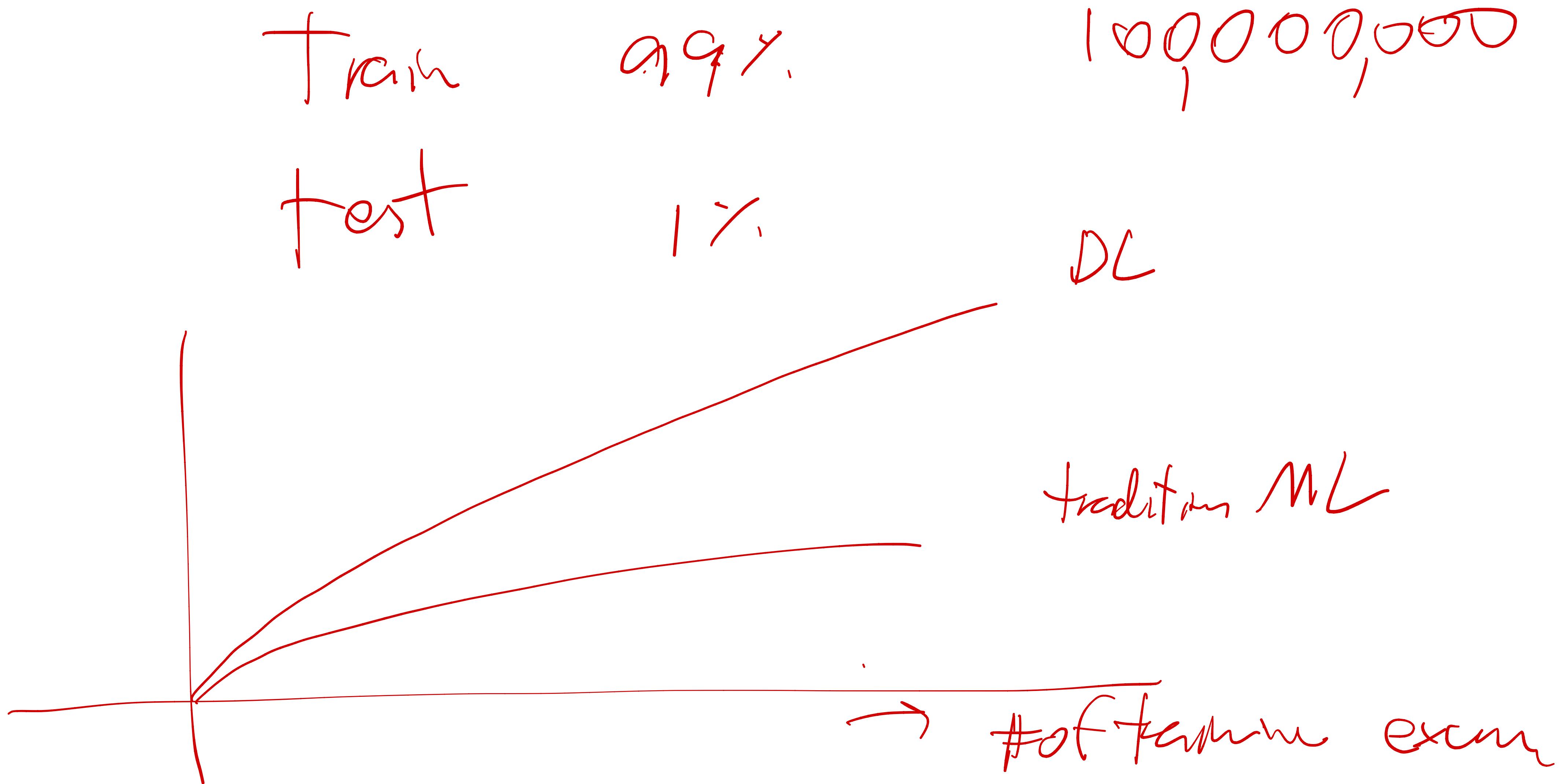


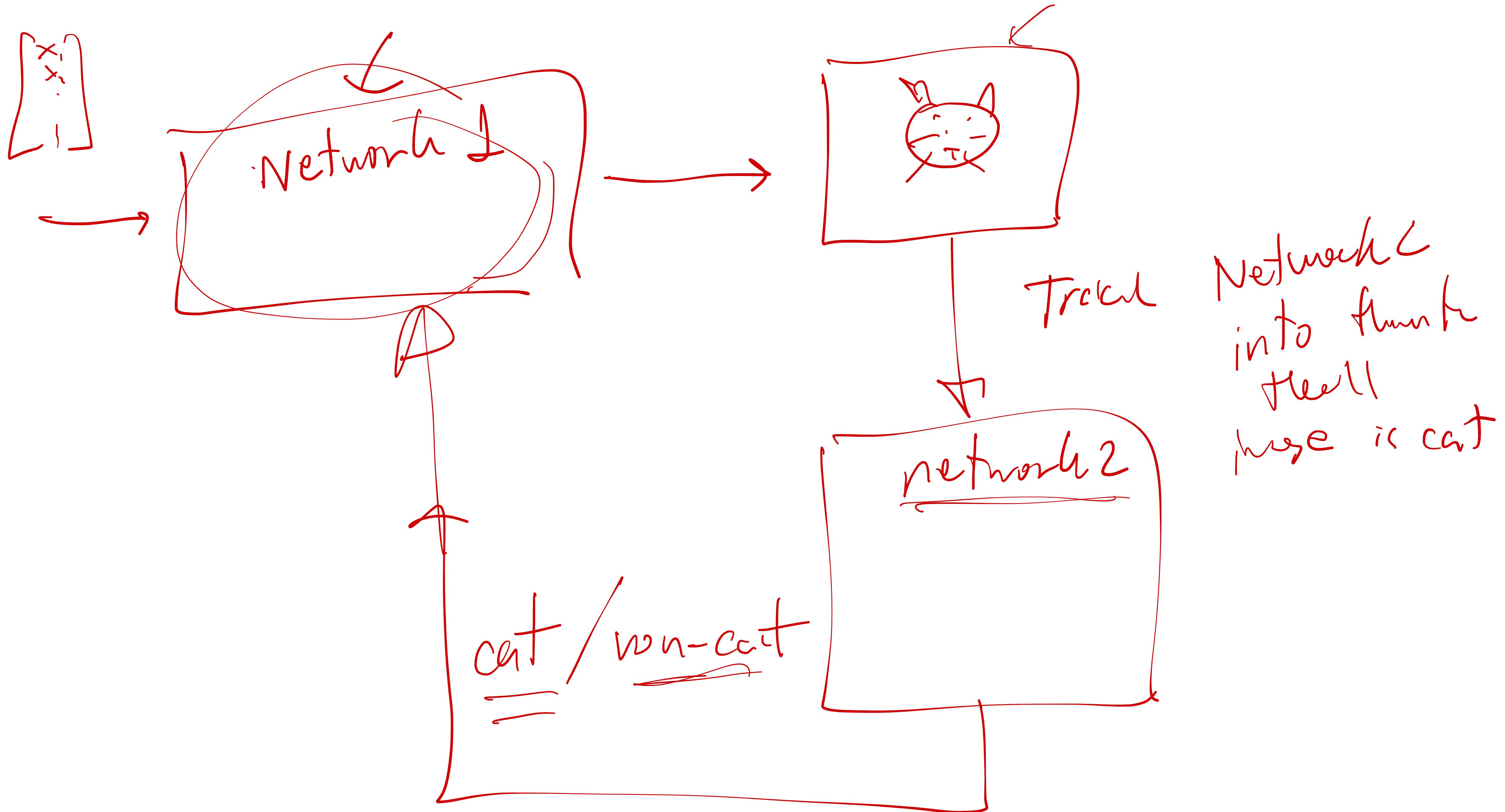
k-fold CV (cross validation)

Advantages

- you get to see the variance of your model

Practice Train-test split





Network 1
into threshold
the 11
here is cat

policy network

action network