$y = f(x)$

↳ features

↳ class

$f \longrightarrow$ KNN

$\longrightarrow$ Softmax

$\longrightarrow$ Neural Network

LINEAR REGRESSION:

$$y = ax + b$$

$a, b \rightarrow$ parameters

ERROR:

$$\hat{y}^{(i)} = f(x^{(i)})$$

$$\varepsilon_i = \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

LOSS / COST FUNCTION:

$$\sum_i \left(y^{(i)} - f(x^{(i)})\right)^2$$

LEAST SQUARES SOLUTION:

$$\hat{y} = ax + b = \theta^T \hat{x} \qquad \theta = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\hat{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

$$L(\theta) = \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \hat{y}^{(i)} \right)^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \theta^T \hat{x}^{(i)} \right)^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \theta^T x^{(i)} \right)^T \left(y^{(i)} - \theta^T x^{(i)} \right)$$

($\theta^T x^{(i)}$: scalar)

$$= \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - x^{(i)T} \theta \right)^T \left(y^{(i)} - x^{(i)T} \theta \right)$$

$$= \frac{1}{2} \left(\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} - \begin{bmatrix} \hat{x}^{(1)T} \\ \hat{x}^{(2)T} \\ \vdots \\ \hat{x}^{(N)T} \end{bmatrix} \theta \right)^T \times$$

$\mathbb{R}^N$ $\quad \mathbb{R}^{N \times 2}$ $\quad \mathbb{R}^{2 \times 1}$

$\mathbb{R}^{1 \times N}$

$$\left(\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} - \begin{bmatrix} (\hat{x}^{(1)})^T \\ (\hat{x}^{(2)})^T \\ \vdots \\ (\hat{x}^{(n)})^T \end{bmatrix} \theta \right)$$

$\mathbb{R}^{n \times 1}$

$$= \frac{1}{2} (y - x\theta)^T (y - x\theta)$$

$$= \frac{1}{2} (y^T - \theta^T x^T)(y - x\theta)$$

$$y^T x\theta = (y^T x\theta)^T = \theta^T x^T y$$

$$L(\theta) = \frac{1}{2} \left(y^T y - \underbrace{y^T x\theta}_{1\times 1} - \theta^T x^T y + \theta^T x^T x\theta \right)$$

$\theta^T x \quad x^T A \quad x$

$$= \frac{1}{2} \left(y^T y - \underbrace{2 y^T x\theta}_{(1\times n)(n\times 2)(2\times 1)} + \theta^T \underbrace{x^T x}_{} \theta \right)$$

$$\frac{\partial L}{\partial \theta} = \frac{1}{2} \left(-(2y^T x)^T + \left[(x^T x) + (x^T x)^T \right] \theta \right)$$

$$= \frac{1}{2} \left(-2x^T y + x^T x\theta + x^T x\theta \right)$$

note: $f = \theta^T x$

$$\frac{\partial f}{\partial x} = \theta$$

$$x^T A x = f$$

$$\frac{\partial f}{\partial x} = x(A + A^T)$$

$$\frac{\partial L}{\partial \theta} = (-X^T y + X^T X \theta)$$

$$\frac{\partial L}{\partial \theta} = 0$$

$$X^T y = X^T X \theta$$

$$\theta = (X^T X)^{-1} X^T y \;//$$

VECTOR AND MATRIX DERIVATIVES:

* $\frac{\partial y}{\partial \vec{x}} = \begin{bmatrix} \partial y/\partial x_1 \\ \partial y/\partial x_2 \\ \vdots \\ \partial y/\partial x_n \end{bmatrix}$ $\quad x \in \mathbb{R}^n$

$(\nabla_{\vec{x}} y)$ $\quad \frac{\partial y}{\partial x} \in \mathbb{R}^n$

$$\Delta y \approx \partial y/\partial x_1 \cdot \Delta x_1 + \partial y/\partial x_2 \cdot \Delta x_1 + \cdots + \partial y/\partial x_n \cdot \Delta x_n$$

$$\Delta y \approx (\nabla_{\vec{x}} y^T)\, \Delta \vec{x}$$

* $y = \vec{\theta}^T \vec{x}$ $\quad \theta \in \mathbb{R}^n \quad x \in \mathbb{R}^n$

$\frac{\partial y}{\partial x} \in \mathbb{R}^n$

$\rightsquigarrow (n \times 1) \times (1 \times 1)$

$$y = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$\frac{\partial y}{\partial x_1} = \theta_1 \qquad \frac{\partial y}{\partial x_2} = \theta_2$$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \partial y/\partial x_1 \\ \partial y/\partial x_2 \\ \vdots \\ \partial y/\partial x_n \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} = \theta$$

* $y = \vec{x}^T A \vec{x}$

$(1\times n)\quad (n\times n)\quad (n\times 1)$

$\partial y / \partial x$

$(n\times 1)\times(1\times 1) = (n\times 1)$

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & & & \\ \vdots & & \ddots & \\ A_{n1} & & & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{n} A_{i1} x_i & \sum_{i=1}^{n} A_{i2} x_i & \cdots & \sum_{i=1}^{n} A_{in} x_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \sum_{j=1}^{n} \sum_{i=1}^{n} A_{ij} x_i x_j$$

$$\frac{\partial f}{\partial x_1} = \underbrace{2A_{11}x_1}_{(i=j=1)} + \underbrace{\sum_{j=2}^{n} A_{1j} x_j}_{(i=1,\, j\neq 1)} + \underbrace{\sum_{i=2}^{n} A_{i1} x_i}_{(i\neq 1,\, j=1)}$$

$$= \sum_{i=1}^{n} A_{i1} x_i + \sum_{j=1}^{n} A_{1j} x_j$$

$$\begin{bmatrix} (A_{11}x_1 + A_{21}x_2 + \dots + A_{n1}x_n) \\ (A_{12}x_1 + A_{22}x_2 + \dots + A_{n2}x_n) \\ \vdots \\ (A_{1n}x_1 + A_{2n}x_2 + \dots A_{nn}x_n) \end{bmatrix} \quad \begin{bmatrix} (A_{11}x_1 + A_{12}x_2 + \dots + A_{1n}x_n) \\ (A_{21}x_1 + A_{22}x_2 + \dots + A_{2n}x_n) \\ \vdots \\ (A_{n1}x_1 + A_{n2}x_2 + \dots A_{nn}x_n) \end{bmatrix}$$

$$= \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & & & \vdots \\ \vdots & & & \\ A_{1n} & \cdots & & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & & & \vdots \\ \vdots & & & \\ A_{n1} & \cdots & & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$= A^T x$ $= A x$

$$\frac{\partial y}{\partial x} = (A + A^T)x$$

* $y = \vec{z}^T A \vec{x}$ $\partial y / \partial A$

$(1 \times m)$ $(m \times n)$ $(n \times 1)$

$$\begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ \vdots & \ddots & & \vdots \\ A_{m1} & \cdots & & A_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} z_i x_j$$

$$\frac{\partial y}{\partial A_{11}} = z_1 x_1$$

$$\frac{\partial y}{\partial A} = \begin{bmatrix} z_1 x_1 & z_1 x_2 & \cdots & z_1 x_n \\ \vdots & & \ddots & \vdots \\ z_m x_1 & & \cdots & z_m x_n \end{bmatrix}$$

$$= \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m \times 1} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$$

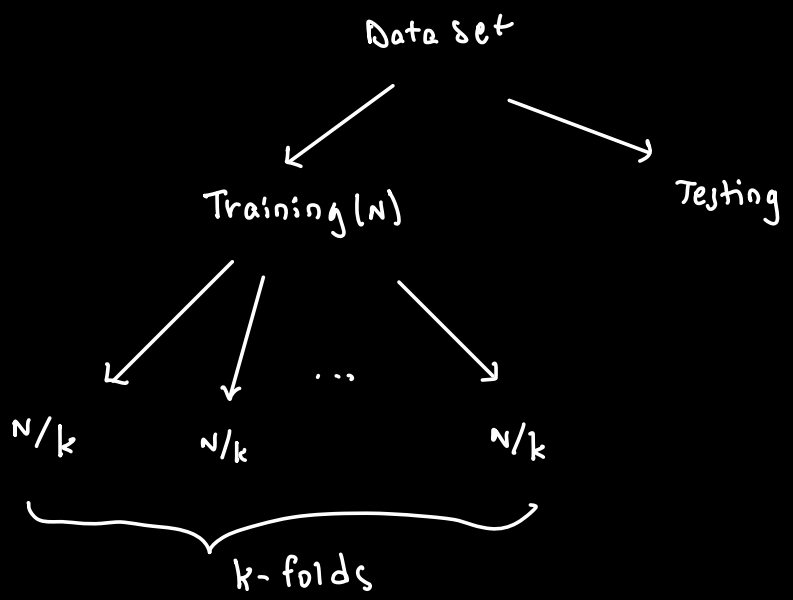
$$= \underset{(m \times 1)}{y} \underset{(1 \times n)}{x^T}$$

Reduce Overfitting:

→ Regularization

→ More data

k-Fold Cross Validation:



→ Train on (k-1) folds

→ Validation on 1 fold (Validation fold)

MAXIMUM LIKELIHOOD ESTIMATION :

Data : H T HH T T H T

Model : $x^{(i)} = \begin{cases} T, & \text{with } P = (1-\theta) \\ H, & \text{with } P = \theta \end{cases}$

MODELS : Model 1 : $\theta = 1$

Model 2 : $\theta = 0.75$

Model 3 : $\theta = 0.5$

MLE = P (Observing the data | Model) = $\theta^4 (1-\theta)^4$

Model 1 :

MLE : $1 \times 0 \times 1 \times 1 \times 0 \times 0 \times 1 \times 0 = 0$

Model 2 :

MLE = $0.75^4 \times 0.25^4$

Model 3 :

MLE = $0.5^4 \times 0.5^4$ ✓ highest

We can find this also by

$$\frac{\partial (MLE)}{\partial \theta} = 0$$

$$\log \text{MLE} = 4\log\theta + 4\log(1-\theta)$$

$$\downarrow$$

$$\partial/\partial\theta = 0$$

$$\frac{4}{\theta} + \frac{4}{1-\theta} = 0$$

$$\frac{1}{\theta} = \frac{1}{\theta - 1}$$

$$\theta = 0.5 \;//$$

MLE - CLASSIFICATION:

We will try to model each class as Gaussian and optimize the parameters for these Gaussian models.

1. Assume each class - Gaussian
2. Find best possible Gaussian model

CLASS i:

$$x^{(i)} \mid y^{(i)} = i \sim N(\mu_i, \Sigma_i)$$

N	examples
n	classes

$\mu_i, \Sigma_i \rightarrow$ PARAMETERS

n classes $\Rightarrow$ 2n parameters.

$$\Sigma_i \rightarrow \begin{bmatrix} \text{var}(\text{dim}_1) & \text{cov} \\ \text{cov}(\text{dim}_1, \text{dim}_2) & \text{var}(\text{dim}_2) \end{bmatrix}$$

Assume:

$$P(y^{(i)}) = 1/n \quad \left.\right\} \text{ equally likely classes.}$$

$$\text{MLE}: \quad L = \prod_{i=1}^{N} p(x^{(i)}, y^{(i)} \mid \theta)$$

$$\theta = (\mu_1, \Sigma_1, \mu_2, \Sigma_2 \cdots \mu_n, \Sigma_n)$$

$$\log L = \sum_{i=1}^{N} \log p(x^{(i)}, y^{(i)} \mid \theta)$$

$$= \sum_{i=1}^{N} \log \left(\underbrace{p(y^{(i)} \mid \theta)}_{1/n} \; p(x^{(i)} \mid y^{(i)}, \theta) \right)$$

$$= \sum_{i=1}^{N} \log p(x^{(i)} \mid y^{(i)}, \theta) + \underbrace{\sum_{i=1}^{N} \log \left[p(y^{(i)} \mid \theta) \right]}_{K_1}$$

$$\log p(x^{(i)} | y^{(i)} = j) = \log \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)}$$

$$= \underbrace{-\frac{n}{2} \log 2\pi}_{k_2} - \frac{1}{2} \log |\Sigma_j| - \frac{1}{2} (x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)$$

$$\log L = \sum_{i=1}^{N} -\frac{1}{2} \log |\Sigma_{y^{(i)}}| - \frac{1}{2} (x^{(i)} - \mu_{y^{(i)}})^T \Sigma_{y^{(i)}}^{-1} (x^{(i)} - \mu_{y^{(i)}}) + k$$

$$\mu_j = \frac{1}{n} \sum_{i : y^{(i)} = j} x^{(i)}$$

Find μ_i, Σ_i for all $i = 1$ to n //

TESTING:

Find class of x

Find j such that $Pr(y=j|x)$ is maximum.

$$\underset{j}{\operatorname{argmax}}\ Pr(y=j|x) = \underset{j}{\operatorname{argmax}}\ \frac{Pr(y=j, x)}{Pr(x)}$$

$$= \underset{j}{\operatorname{argmax}}\ \frac{P(j)\ P(x|j)}{P(x)}$$

($P(j)$: $1/n$; $P(x)$: independent of j)

$$= \underset{j}{\operatorname{argmax}}\ P(x|j)$$

k- NEAREST NEIGHBORS:

x : Image $32 \times 32 \times 3$ $\mathbb{R}^{3072}$

y : classes 1 - 10

TRAINING: Store all $(x^{(i)}, y^{(i)})$

PRO:
* simple, fast

CONS:
* Memory-intensive

TESTING: → Find k nearest neighbor of x

→ Choose most common y.

PRO:
* simple

CONS:
* Scale with amount of training data

HYPERPARAMETERS:
→ k
→ distance metric.

OTHER CONS:
→ distance (say shift image) → not accurate
→ Curse of dimensionality → Distance doesn't make sense.

LINEAR CLASSIFICATION:

x : Image $32 \times 32 \times 3$ $\mathbb{R}^{3072}$ $n = 3072$

y : score to be in each class (1 to 10)

$c = 10$

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

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

$$y = Wx + b$$

$$W \in \mathbb{R}^{c \times n}$$

$$b \in \mathbb{R}^c$$

$$y = \underset{c \times 1}{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_c \end{bmatrix}} = \begin{bmatrix} w_1^T x + b_1 \\ w_2^T x + b_2 \\ \vdots \\ w_c^T x + b_c \end{bmatrix} = \underset{c \times n}{\begin{bmatrix} — & w_1^T & — \\ — & w_2^T & — \\ & \vdots & \\ — & w_c^T & — \end{bmatrix}} \underset{n \times 1}{\begin{bmatrix} x \end{bmatrix}} + \underset{c \times 1}{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix}}$$

So each class i has

weight $\rightarrow w_i$

bias $\rightarrow b_i$

Answer : $\underset{i}{\text{argmax}}(y_i)$

Why not use loss function (least squares) like regression?

if $y = 7$ but $\hat{y} = 6$

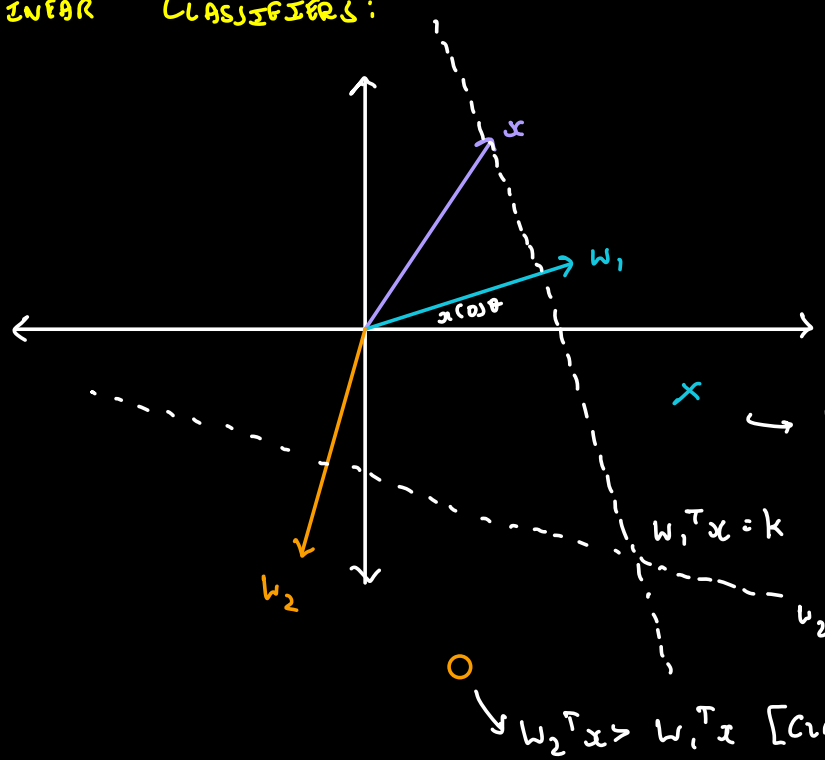
error: $(6-7)^2 = 1$

But if $\hat{y} = 1$

error: $(1-7)^2 = 36$

} but both are equally wrong!

LINEAR CLASSIFIERS:



assum = 1

$w_1^T x = \|w_1\| \|x\| \cos\theta$

$= \|x\| \cos\theta$

$x \hookrightarrow w_1^T x > w_2^T x$ [CLASS 1]

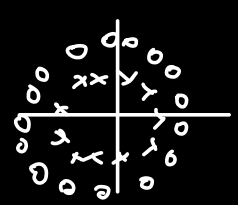
$o \hookrightarrow w_2^T x > w_1^T x$ [CLASS 2]

CONS:

→ Fails on not linearly separable data.

o	x
x	o

or



SOFTMAX CLASSIFIER:

We have $y = [y_1, y_2 \cdots y_c]^T$

y_i is a score

But to make more intuitive sense, we can convert score to probability. [It is normalized now too].

$$y_i = a_i(x) = w_i^T x + b_i$$

$$\text{Softmax}_i(x) = \frac{e^{a_i(x)}}{\sum_{j=1}^{c} e^{a_j(x)}}$$

$$\text{Prob}(y^{(i)} = i \mid x^{(i)}, \theta) = \text{softmax}_i(x^{(i)})$$

$$P(x, y|\theta) = \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}|\theta)$$

$$= \prod_{i=1}^{m} P(x^{(i)}|\theta)\, P(y^{(i)}|x^{(i)}, \theta)$$

$m \rightarrow$ number of examples

$c \rightarrow$ classes

$n \rightarrow$ number of features

$$\theta = \{w, b\}$$

MLE on this:

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{m} \underbrace{p(x^{(i)} \mid \theta)}_{\text{independent of } \theta} p(y^{(i)} \mid x^{(i)}, \theta)$$

$$= \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}, \theta)$$

$$= \arg\max_{\theta} \sum_{i=1}^{m} \log p(y^{(i)} \mid x^{(i)}, \theta)$$

$$= \arg\max_{\theta} \sum_{i=1}^{m} \log \underset{y^{(i)}}{\text{softmax}}(x^{(i)})$$

$$= \arg\max_{\theta} \sum_{i=1}^{m} \log \left[\frac{e^{a_{y^{(i)}}(x^{(i)})}}{\sum_{j=1}^{c} e^{a_j(x^{(i)})}} \right]$$

$$= \arg\max_{\theta} \frac{1}{m} \sum_{i=1}^{m} \left[a_{y^{(i)}}(x^{(i)}) - \log \sum_{j=1}^{c} e^{a_j(x^{(i)})} \right]$$

$$= \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \left[\log \sum_{j} e^{a_j(x^{(i)})} - a_{y^{(i)}}(x^{(i)}) \right]$$

Loss Function:

$$L = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \left[\log \sum_{j} e^{a_j(x^{(i)})} - a_{y^{(i)}}(x^{(i)}) \right]$$

For one example $x^{(i)} - y^{(i)}$

$$\text{MLE} : \log P_\theta(y^{(i)} | x^{(i)}) = a_{y^{(i)}}(x^{(i)}) - \log \sum_{j} e^{a_j(x^{(i)})}$$

→ If $a_{y^{(i)}}(x^{(i)})$ is the largest score

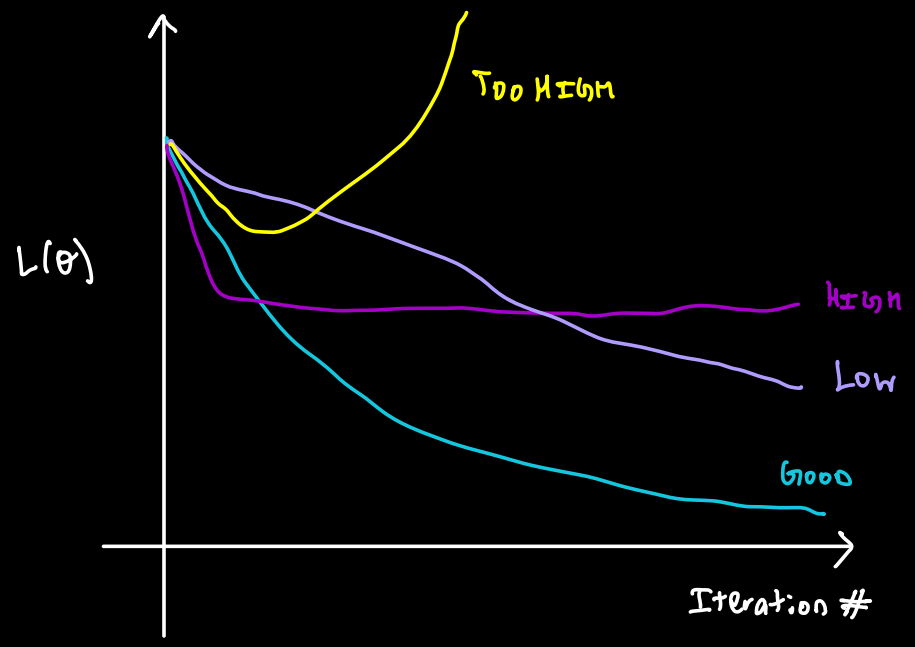
then MLE ≈ 0 (optimum)

Loss ≈ 0

→ Else MLE < 0, loss > 0.

Gradient Descent:

Interpreting cost function to find learning rate



Why use analytical gradient over numerical?

↳ $\frac{\partial f}{\partial x}$

↳ $\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

- # of parameters high
- f may be expensive.

BATCH GRADIENT DESCENT:

Each data provides a noisy estimate of the gradient at that point.

So, Take all m examples, average the $\partial f / \partial w$ and apply to w, for each iteration. (Least noisy estimate).

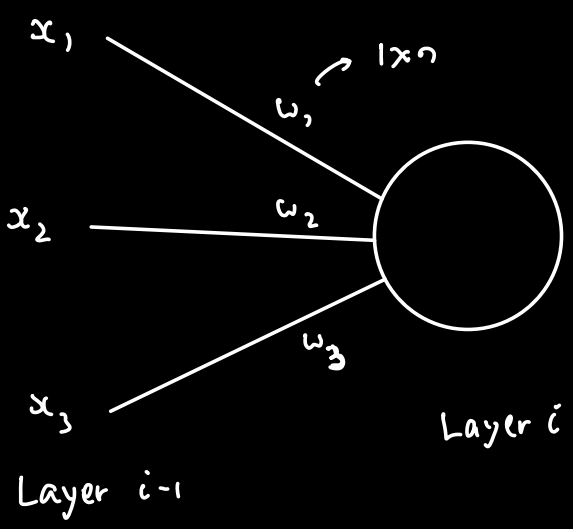
STOCHASTIC / MINI-BATCH GRADIENT DESCENT:

Noisier estimate of gradient with fewer examples.

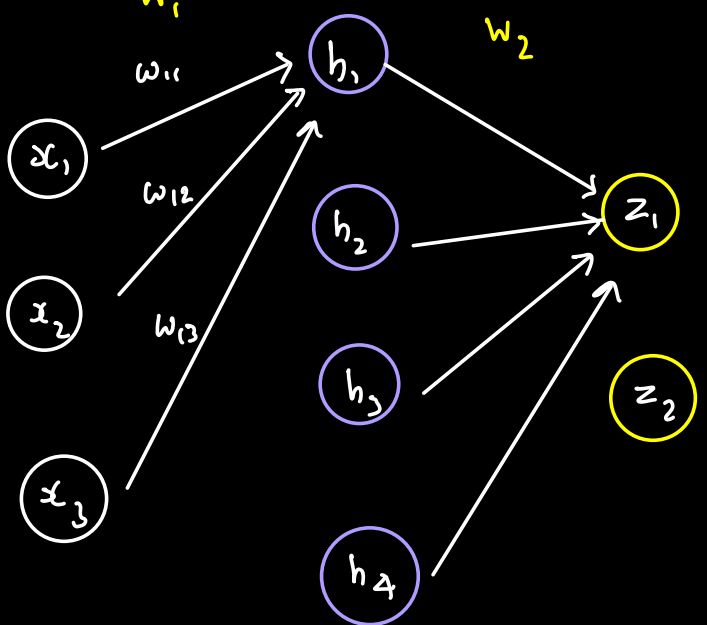
=> Regularizes

=> Faster

NEURAL NETWORK:



$$f(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$$



LAYERS = 2

NEURONS = 4 + 2 = 6

WEIGHTS = 3 × 4 + 4 × 2 = 20

BIASES = 4 + 2 = 6

PARAMETERS = 20 + 6 = 26

w_{13} → weight from 3rd neuron in $(i-1)^{th}$ layer to 1st neuron in i^{th} layer

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = f\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ \vdots & \ddots & \vdots \\ w_{41} & \cdots & w_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_4 \end{bmatrix} \right)$$

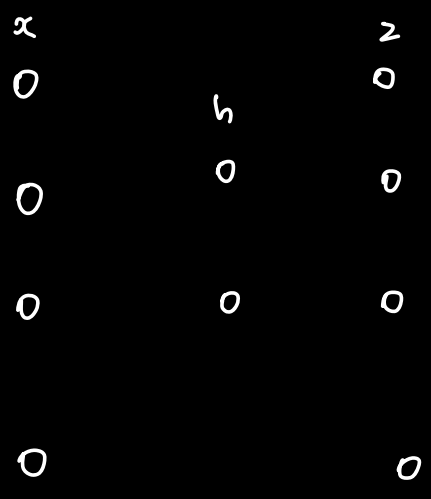
$$h = f\left(\underbrace{W_1}_{(n_h \times n_x)} x + b\right)$$

h: $(n_h \times 1)$ — W_1: $(n_h \times n_x)$ — x: $(n_x \times 1)$ — b: $(n_h \times 1)$

$$z = \underbrace{W_2}_{(n_o \times n_h)} h + b_2$$

z: $(n_o \times 1)$ — h: $(n_h \times 1)$ — b_2: $(n_o \times 1)$

AUTO ENCODER:



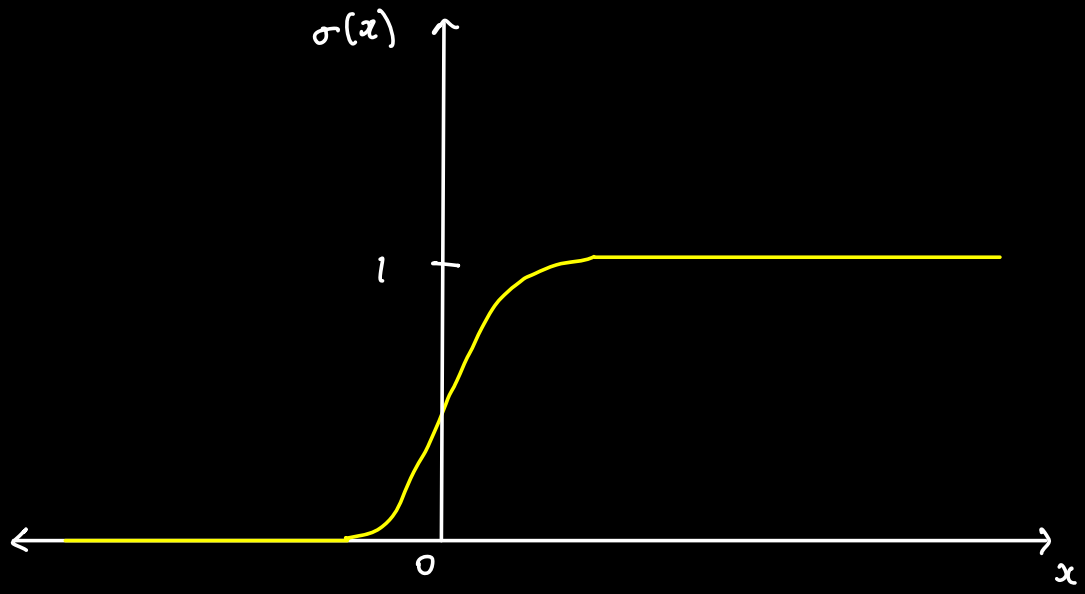
$L : \|z - x\|^2$

Dimensionality Reduction.

Activation Functions:

→ Sigmoid Activation:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$ [Softmax for 2 classes]



$$\frac{\partial}{\partial x}\sigma(x) = \frac{-1}{(1+e^{-x})^2}(-e^{-x})$$

$$= \frac{1}{1+e^{-x}} \times \left(\frac{e^{-x}}{1+e^{-x}}\right)$$

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

$$= \sigma(x)\left(1 - \sigma(x)\right)$$

DISADVANTAGE:

$h_n = \sigma(w) = \sigma(w^T x + b)$ $\qquad a = w^T x + b$

$$\frac{\partial L(w)}{\partial w} = \underbrace{\frac{\partial \sigma(w)}{\partial w}} \cdot \frac{\partial L(w)}{\partial \sigma(w)}$$

$$\underbrace{\frac{\partial \sigma(w)}{\partial a}} \cdot \frac{\partial a}{\partial w} \quad (\partial w \hookrightarrow x)$$

$$\sigma(a)(1 - \sigma(a))$$

If $\sigma(w^T x + b)$ is close to 0 or 1,

then $\frac{\partial \sigma(w)}{\partial a} \approx 0$ and hence

$$\frac{\partial L(w)}{\partial w} \rightarrow \text{very small number}$$

$$w \leftarrow w - \varepsilon \underbrace{\frac{\partial L(w)}{\partial w}}_{\approx 0}$$

w won't update

$\Rightarrow$ no learning

Pros:

→ At $x = 0$, linear $\partial\sigma/\partial x \approx 1$ (large).

→ Differentiable everywhere

Cons:

→ At extremes, $\partial\sigma/\partial x \approx 0$ No learning

→ Output not zero-centered (0.5)

↳ activation function ≥ 0

"ZERO ZAGGING GRADIENT DESCENT"

$$x \xrightarrow{w,\sigma} h_1 \xrightarrow{w,\sigma} h_2 \xrightarrow{w} \text{OUTPUT}$$

$$h_2 = f(w) = \sigma\left(w^T h_1 + b\right) = \sigma(z)$$

where $z = w^T h_1 + b$

$$\frac{\partial h_2}{\partial w} = \frac{\partial z}{\partial w} \cdot \frac{\partial h_2}{\partial z}$$

$\frac{\partial z}{\partial w}$: h_1, ≥ 0

$\frac{\partial h_2}{\partial z}$: $\sigma(z)\left(1 - \sigma(z)\right)$, ≥ 0 ≥ 0

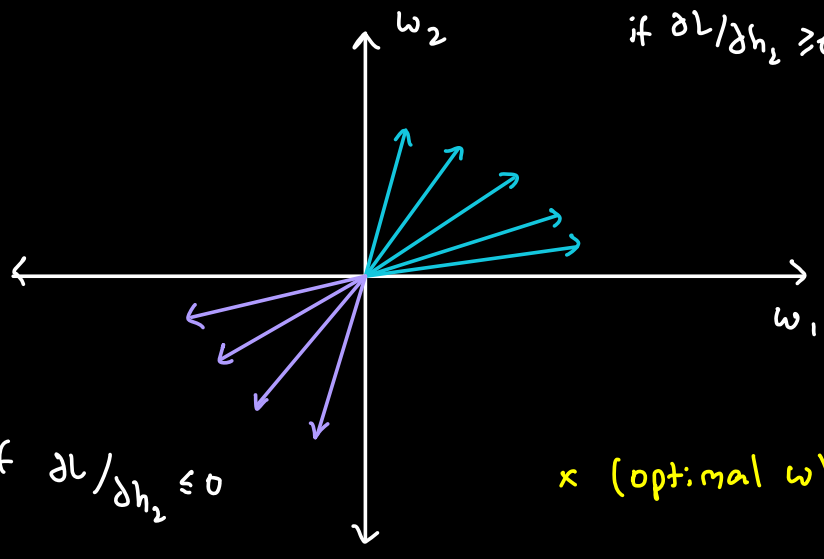
$$\frac{\partial h_2}{\partial w} \gtrless 0$$

$$\underbrace{\frac{\partial L}{\partial w}}_{} = \underbrace{\frac{\partial h_2}{\partial w}}_{\gtrless 0} \cdot \underbrace{\frac{\partial L}{\partial h_2}}_{\text{Scalar}}$$

so if $\partial L/\partial h_2 \geq 0$ all $\partial L/\partial w \geq 0$

≤ 0 all $\partial L/\partial w \leq 0$

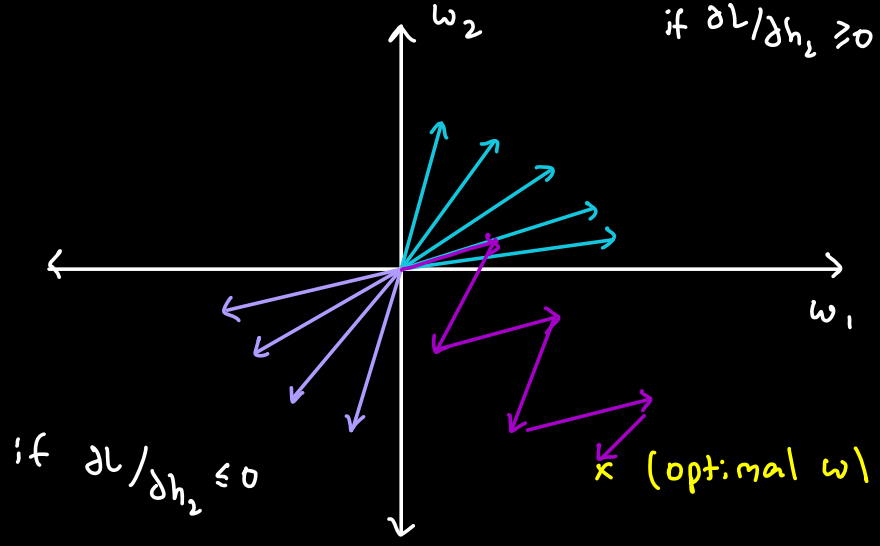
Assume 2 w's



if $\partial L/\partial h_2 \geq 0$

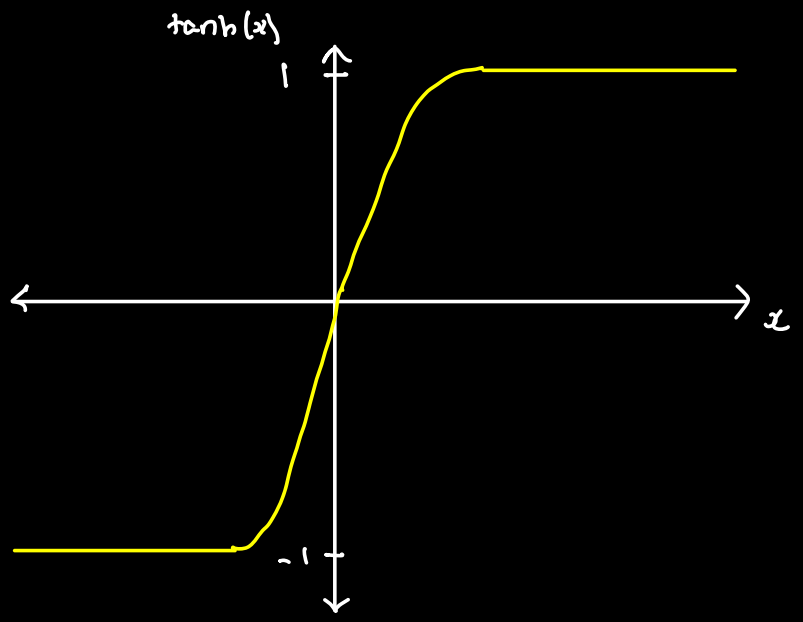
if $\partial L/\partial h_2 \leq 0$

x (optimal w)


w_2
if $\partial L / \partial h_2 \geq 0$
w_1
if $\partial L / \partial h_2 \leq 0$
x^* (optimal w)

⟶ HYPERBOLIC TANGENT

$$\tanh(x) = 2\sigma(x) - 1$$



$$\tanh(x) = \frac{2}{1+e^{-x}} - 1$$

$$= \frac{2 - 1 - e^{-x}}{1+e^{-x}}$$

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

$$\frac{\partial \tanh(x)}{\partial x} = \frac{(1+e^{-x})(e^{-x}) - (1-e^{-x})(-e^{-x})}{(1+e^{-x})^2}$$

$$= \frac{e^{-x} + e^{-2x} + e^{-x} - e^{-2x}}{(1+e^{-x})^2}$$

$$= \frac{1+e^{-x} - (1 - e^{-x})}{(1+e^{-x})^2}$$

$$= 1 - \tanh^2(x)$$

Pros:

→ $x = 0$ Linear

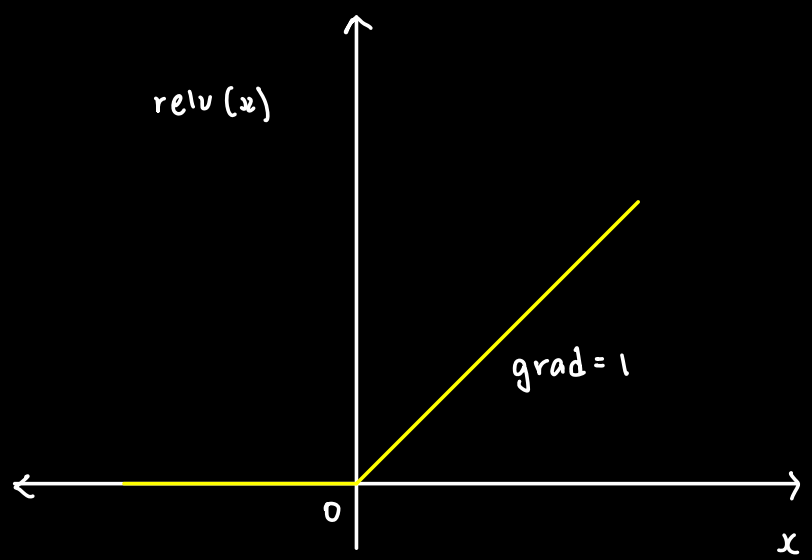
→ Differentiable

→ Zero-centered

Cons:

→ saturates at large / small values → no learning.

→ ReLU Unit:



$$\text{relu}(x) = \max(0, x)$$

$$\frac{d\,\text{relu}(x)}{dx} = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$$

Pros:

- → Converges Faster
- → Linear when active
  - ↳ derivative large
- → No saturation if $x > 0$

Cons :

→ not zero-centric (zig-zag)

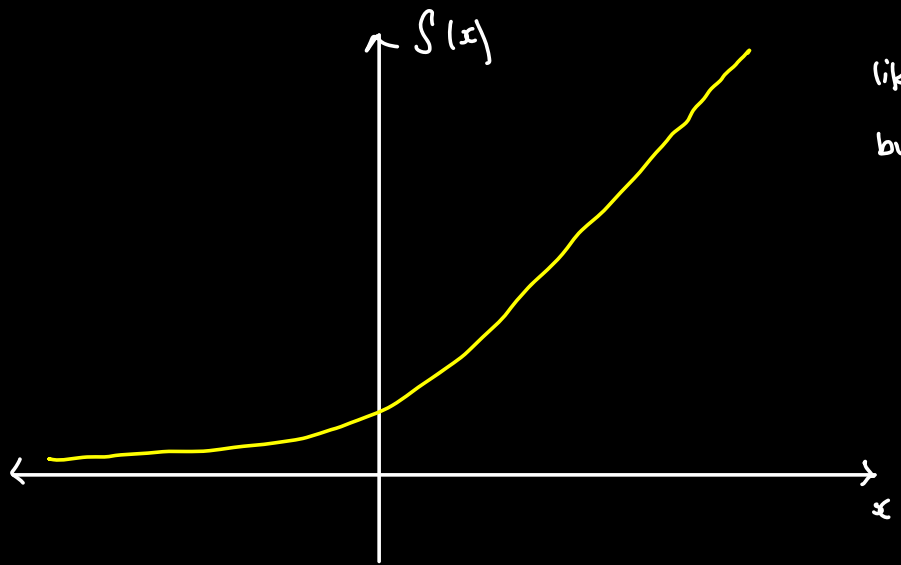
→ not differentiable at $x = 0$

→ zero activation → no learning.

→ SOFTPLUS UNIT:

$$S(x) = \log(1 + e^x)$$

$$\frac{\partial}{\partial x} S(x) = \sigma(x)$$

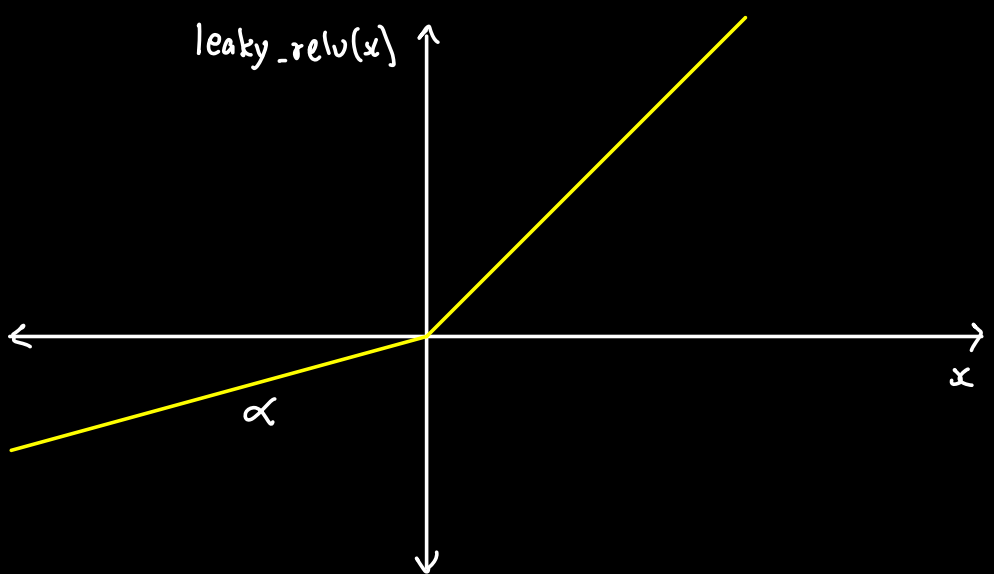


like relu

but differentiable

→ LEAKY RELU / PRELU

$$f(x) = \max(\alpha x, x)$$

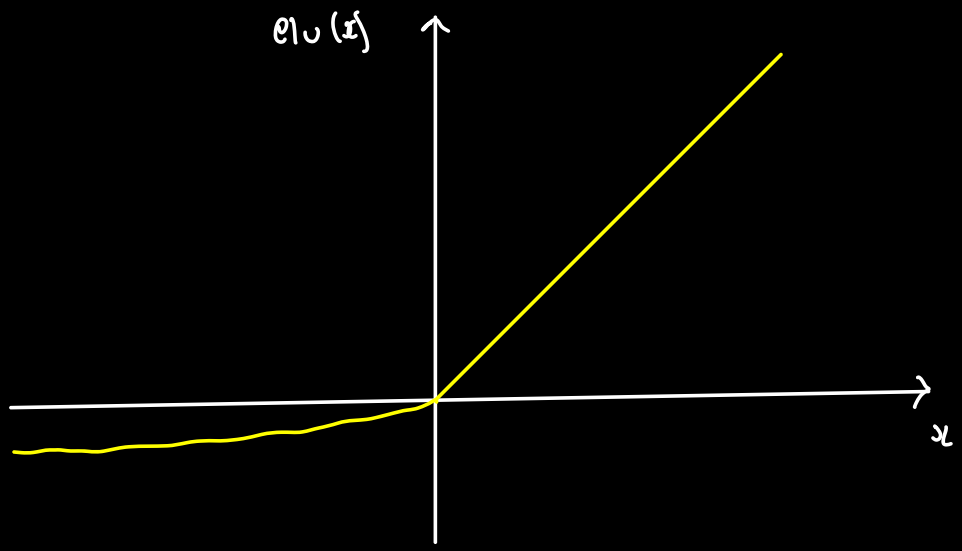


if α → parameter PReLU

→ hyperparameter Leaky ReLU

→ ELU

$$f(x) = \max(\alpha(e^x - 1), x)$$



Con: compute exponential (expensive).

Output Activations:

Assume 2 class classification

→ Apply Sigmoid

$$\hat{y} = \sigma(z)$$

Which error →

$$MSE = \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \sigma(z^{(i)})\right)^2$$

$$CE = -\sum_{i=1}^{n} \left[y^{(i)} \log \sigma(z^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(z^{(i)})) \right]$$

(Cross-Entropy)

↓

MLE - softmax - 2 classes

↓

$$P(x^{(i)}, y^{(i)} \mid \theta) \prod_{i=1}^{n} \underbrace{P(x^{(i)} \mid \theta)} \cdot P(y^{(i)} \mid x^{(i)}, \theta)$$

$$P(\text{class } 1 \mid x^{(i)}) = \frac{1}{1 + e^{-x^{(i)}}} = \sigma(x^{(i)})$$

$$P(\text{class } 2 \mid x^{(i)}) = 1 - \sigma(x^{(i)})$$

$$P(y^{(i)} \mid x^{(i)}, \theta) = \sigma(x^{(i)})^{y^{(i)}} \cdot (1 - \sigma(x^{(i)}))^{(1-y^{(i)})}$$

MLE

$$\log P = \log \sum_{i=1}^{n} P(y^{(i)} \mid x^{(i)}, \theta)$$

$$= \sum_{i=1}^{n} y^{(i)} \cdot \sigma(x^{(i)}) + (1 - y^{(i)}) (1 - \sigma(x^{(i)}))$$

Which error $\rightarrow$

$$MSE = \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \sigma(z^{(i)})\right)^2$$

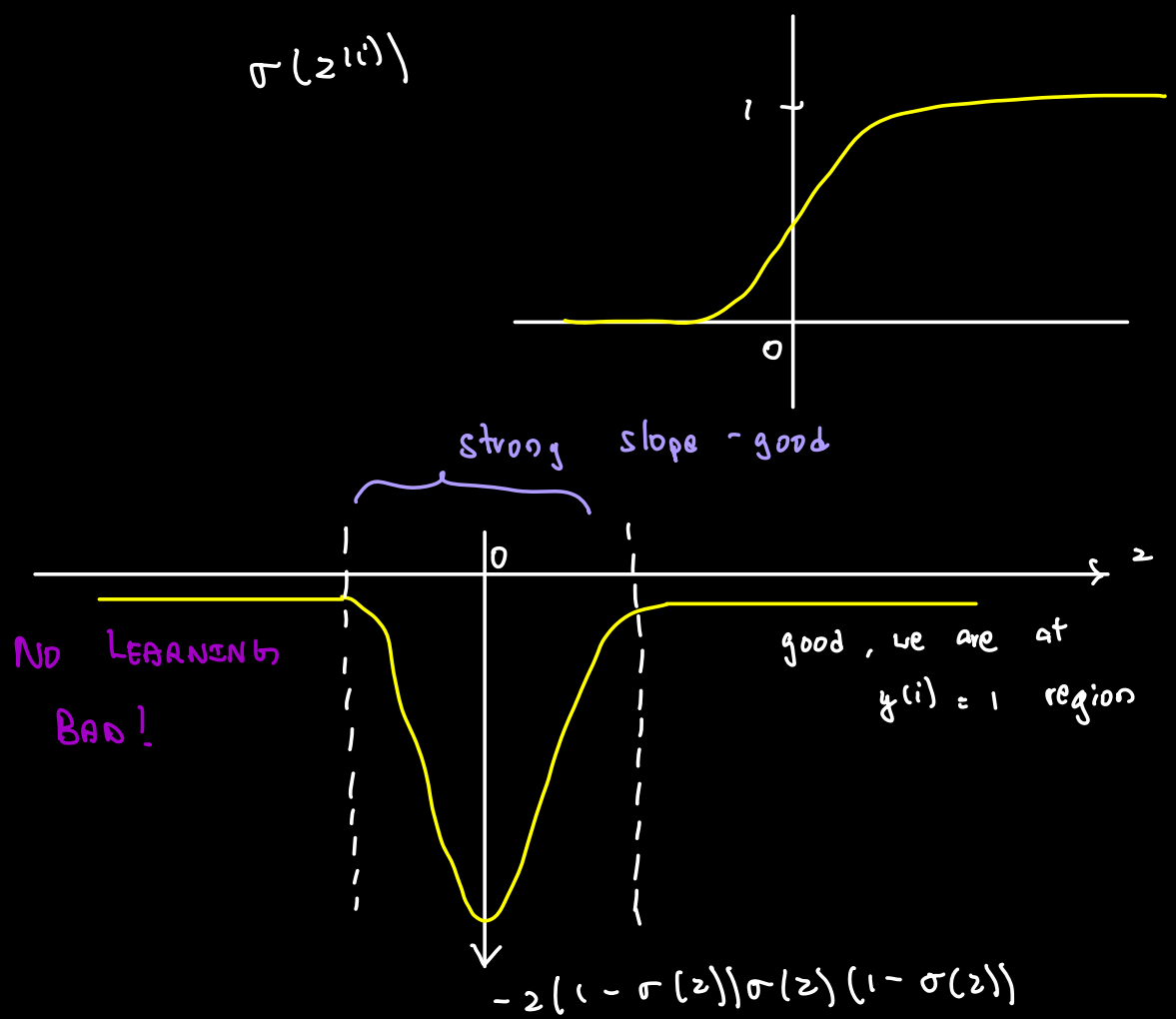
$$CE = -\sum_{i=1}^{n} \left[y^{(i)} \log \sigma(z^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(z^{(i)})) \right]$$

(CROSS-ENTROPY)

MSE :

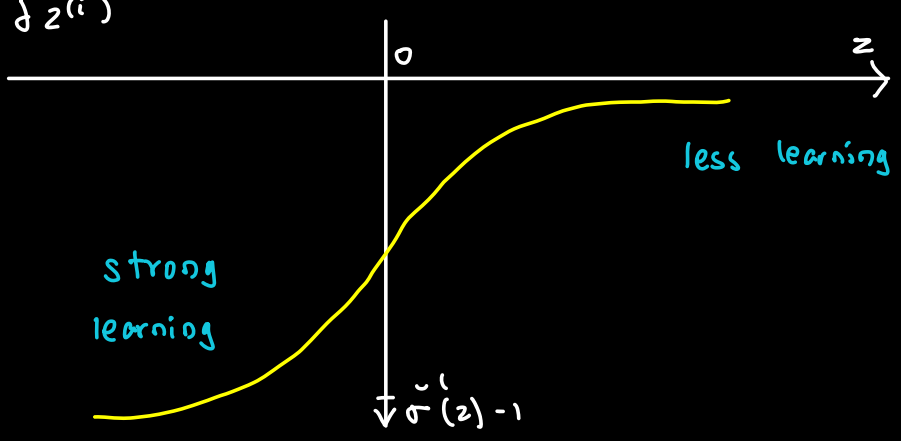
$$\frac{\partial MSE}{\partial z^{(i)}} = \underbrace{-2 \left(y^{(i)} - \sigma(z^{(i)})\right)}_{\partial MSE / \partial \sigma} \underbrace{\left(\sigma(z^{(i)}) (1 - \sigma(z^{(i)}))\right)}_{\partial \sigma / \partial z}$$

if $y^{(i)} = 1$



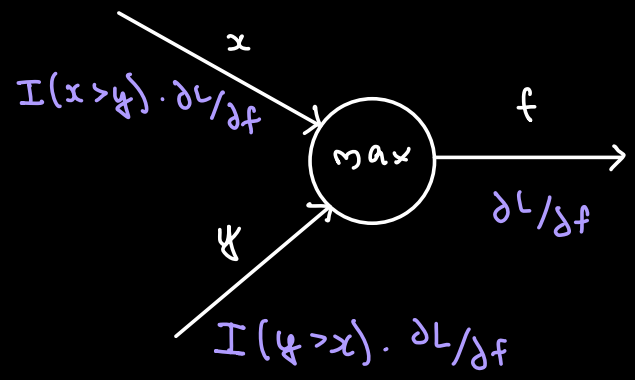
CE:

$$\frac{\partial CE}{\partial z^{(i)}} = \sigma(z^{(i)}) - 1$$



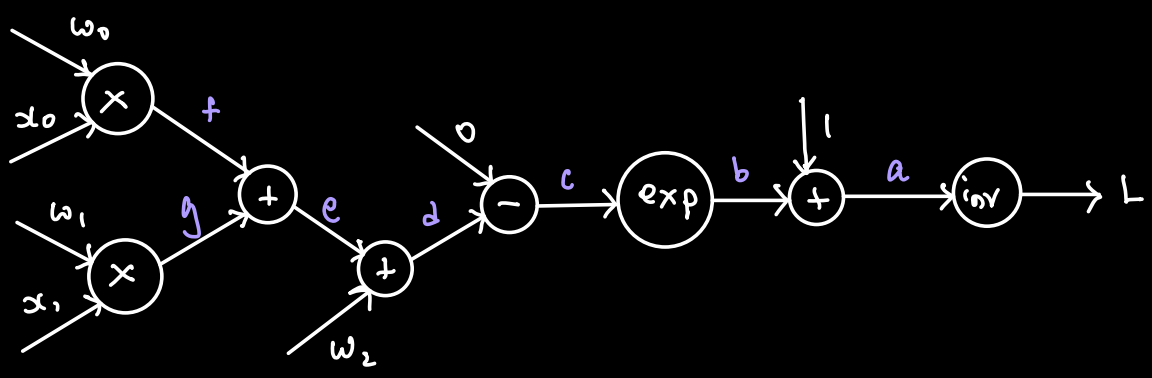
BACK PROPAGATION:

$\max(x, y)$



$$I(x>y) = \begin{cases} 1, & x>y \\ 0, & y>x \end{cases}$$

$$L = \frac{1}{1+e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



1. $L = 1/a$

   $\partial L/\partial a = -1/a^2$

2. $a = 1+b$

   $\partial L/\partial b = -1/a^2$

3. $b = e^c$

   $\partial b/\partial c = e^c$

   $\partial L/\partial c = e^c\left(-1/a^2\right)$

4. $c = -d$

$\partial L/\partial d = e^c/a^2$

5. $d = e + w_2$

$\partial L/\partial e = e^c/a^2$

$\partial L/\partial w_2 = e^c/a^2$

6. $e = f + g$

$\partial L/\partial f = \partial L/\partial g = e^c/a^2$

7. $f = w_0 x_0$

$$\frac{\partial f}{\partial w_0} = x_0$$

$$\frac{\partial L}{\partial w_0} = \frac{\partial f}{\partial w_0} \cdot \frac{\partial L}{\partial f} = \frac{e^c x_0}{a^2}$$

$$\frac{\partial L}{\partial x_0} = \frac{e^c w_0}{a^2}$$

8. $g = w_1 x_1$

$$\frac{\partial L}{\partial w_1} = \frac{e^c}{a^2} \cdot x_1$$

$$\frac{\partial L}{\partial x_1} = \frac{e^c \cdot w_1}{a^2}$$

$c = -(w_0 x_0 + w_1 x_1 + w_2)$

$a = 1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}$

DERIVATIVES OF VECTOR /MATRIX:

$$\vec{y} = W\vec{x}$$

$m \times 1$

$n \times 1$

$n \times m$

$$\frac{\partial y}{\partial x} = \nabla_x y = \begin{bmatrix} | & | & & | \\ \partial y_1/\partial x & \partial y_2/\partial x & \cdots & \partial y_n/\partial x \\ | & | & & | \end{bmatrix}$$

$$= \begin{bmatrix} \partial y_1/\partial x_1 & \partial y_2/\partial x_1 & \cdots & \partial y_n/\partial x_1 \\ \partial y_1/\partial x_2 & \partial y_2/\partial x_2 & \cdots & \partial y_n/\partial x_2 \\ \vdots & & & \\ \partial y_1/\partial x_m & \partial y_2/\partial x_m & \cdots & \partial y_n/\partial x_m \end{bmatrix}$$

$y = Wx$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ \vdots & & \ddots & \vdots \\ w_{n1} & \cdots & & w_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$$y_1 = w_{11}x_1 + w_{12}x_2 + \cdots + w_{1m}x_m = \sum w_{1i}x_i$$

$$y_2 = \sum w_{2i}x_i$$

$$\vdots$$

$$y_n = \sum w_{ni}x_i$$

$$\partial y_1/\partial x_1 = w_{11}$$

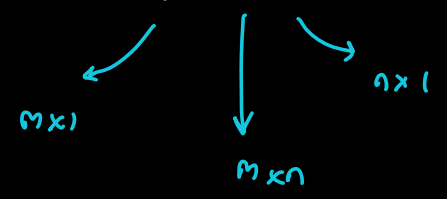
$$\partial y_1/\partial x_2 = w_{12}$$

$$\partial y/\partial x = \begin{bmatrix} \partial y_1/\partial x_1 & \partial y_2/\partial x_1 & & \partial y_n/\partial x_1 \\ \partial y_1/\partial x_2 & \partial y_2/\partial x_2 & \cdots & \partial y_n/\partial x_2 \\ \vdots & \vdots & & \vdots \\ \partial y_1/\partial x_m & \partial y_2/\partial x_m & & \partial y_n/\partial x_m \end{bmatrix}$$

$$= \begin{bmatrix} w_{11} & w_{21} & & w_{n1} \\ w_{12} & w_{22} & \cdots & w_{n2} \\ \vdots & \vdots & & \vdots \\ w_{1m} & w_{2m} & & w_{nm} \end{bmatrix} = W^T$$

Tensor Derivatives:

$y = Wx$



$$\partial y / \partial W \in \mathbb{R}^{m \times n \times m}$$

$$L = \frac{1}{2} \sum_{i=1}^{N} \| y^{(i)} - Wx^{(i)} \|^2$$

Find $\partial L / \partial W$

$$L = \frac{1}{2} \sum_{i=1}^{N} (y^{(i)} - Wx^{(i)})^T (y^{(i)} - Wx^{(i)})$$

Let $z^{(i)} = y^{(i)} - Wx^{(i)}$

$$L = \frac{1}{2} \sum_{i=1}^{N} z^{(i)T} z^{(i)}$$

$$\frac{\partial L}{\partial W} = \frac{\partial z}{\partial W} \cdot \frac{\partial L}{\partial z}$$

$m \times n$ $\quad$ $m \times n \times m$ $\quad$ $m \times 1$

$m \times 1 \rightarrow z$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z} x^T$$

$$\partial/\partial x\,(x^T y) = y$$

$$\partial/\partial x\,(Wx) = W^T$$

$$y = Wx$$

$1 \times W$ $\quad$ $W \times U$ $\quad$ $1 \times U$

$$\frac{\partial L}{\partial x} = W^T \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y} \cdot x^T$$

$$y = Wx$$

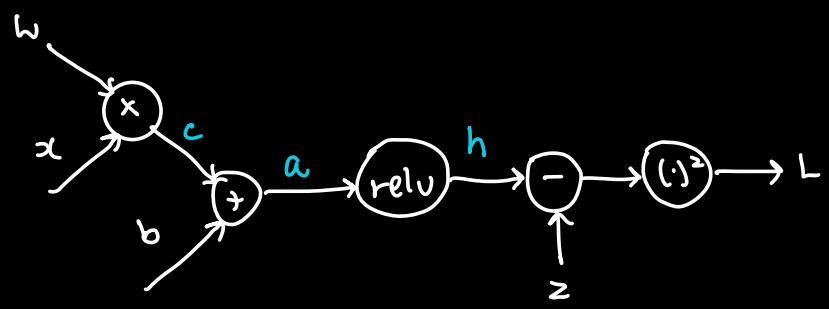
$dx \times W$ $\quad$ $W \times U$ $\quad$ $dx \times U$

$$\frac{\partial L}{\partial x} = W^T \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y} x^T$$

$$L = (h - z)^2$$

$$h = \text{ReLU}(Wx + b)$$



$$\frac{\partial L}{\partial h} = 2(h - z)$$

$$h = \text{relu}(a)$$

$$\frac{\partial h}{\partial a} = \mathbb{I}(a > 0)$$

$$\frac{\partial L}{\partial a} = \frac{\partial h}{\partial a}\frac{\partial L}{\partial h} = \mathbb{I}(a > 0) \cdot 2(h - z)$$

$$a = b + c$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial c} = \mathbb{I}(a > 0) \cdot 2(h - z)$$

$$c = Wx$$

$$\frac{\partial L}{\partial x} = W^T \frac{\partial L}{\partial c} = W^T \mathbb{I}(a > 0) \cdot 2(h - z)$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial c} \cdot x^T = \mathbb{I}(a > 0) \cdot 2(h - z)\, x^T$$

How to train Neural Networks better?

- → Weight Initialization
- → Batch Normalization
- → Regularizations
  - → L1 + L2 Norm
  - → Dataset Augmentation
  - → Model Ensembles
  - → Dropout.

Weight Initialization:

- → All zeroes

  All the neurons in hidden unit will be Symmetric

- → Random small values

$h_1 = relu(w_1 x)$

$h_2 = relu(w_2 h_1)$

⋮

As w_i are small, activations decay to zero.

Assume $z = w_{10} h_9$

$$\frac{\partial L}{\partial w_{10}} = \frac{\partial L}{\partial z} \cdot h_9^T$$

h_9 ← 0's

$\frac{\partial L}{\partial w_{10}} = 0$ No learning.

→ Random large values

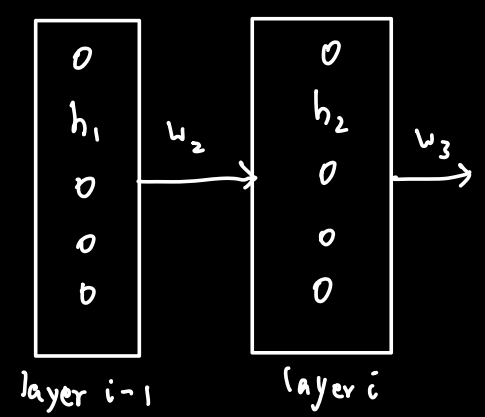
Activations explode.

$$\frac{\partial L}{\partial W_{10}} = \frac{\partial L}{\partial z} \cdot h_a^T$$

↑ huge

↑ huge, outside the space.

XAVIER INITIALIZATION:



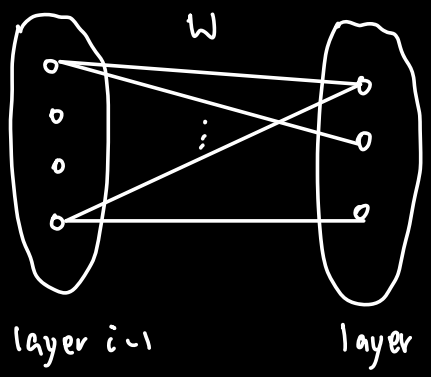
CONDITIONS:

$$\Rightarrow \text{var}(h_1) \approx \text{var}(h_2) \approx \dots \approx \text{var}(h_i)$$

$$\Rightarrow \text{var}\left(\frac{\partial L}{\partial W_i}\right) \approx \text{var}\left(\frac{\partial L}{\partial W_j}\right)$$

} only for the first step.

$$\boxed{\text{var}(W_{ij}) = \frac{1}{n_{in}}} \quad \text{or} \quad \frac{1}{n_{out}} \quad \text{or} \quad \frac{2}{n_{in} + n_{out}}$$



$n_{in} = 4$

$n_{out} = 3$

$var(w_{ij}) = 1/4$ or $1/3$ or $2/7$

Weights in layer i :

$$N\left(0, \frac{2}{n_{in} + n_{out}}\right)$$

Pros:

* Weights not 0 for few layer networks

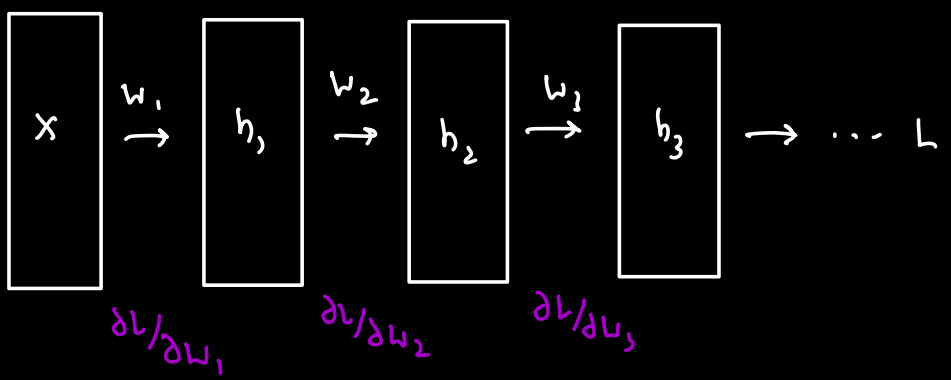
Cons:

* Large number of layers → weights die down
* Doesn't work for ReLU
  - ↳ ReLU sets all negative to 0, so it kills half the units. So "He" initialization says use $var(w_{ij}) = 2/n_{in}$ //

[MSRA]

BATCH NORMALIZATION:

What are we solving? [Internal Covariate Shift, Loss Surface Smoother]



We find $\partial L/\partial w_2$ that best decreases final loss, given the current h_1. But in gradient descent, we update w_1 and w_2 simultaneously. So the new value of w_2 which was best for the old h_1 may not be best for the new h_1 that will be calculated with the updated w_1.

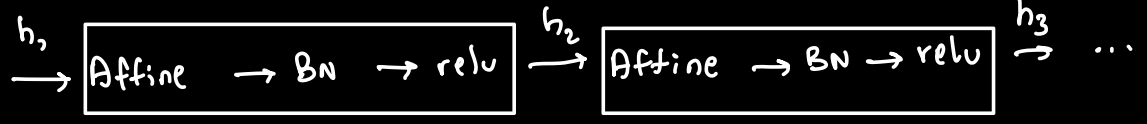
eg: h_1 had mean 100. w_2 felt that $w_2 h_1$ was low and to decrease loss, it learns to increase w_2. But may be at the same time w_1 learns and increases too. Now h_1 is large and so is w_2. This might

have increased the overall loss!

So, we can ensure the statistics of hidden layers are same by normalizing them!

$$E(h_i) = 0$$

$$var(h_i) = 1$$



PROBLEM:

→ All units have same statistics. But some features might need a different mean and variance.

=> Scale and Shift

METHOD:

1. $$\hat{x}_i = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

$$\mu_i = \frac{1}{m} \sum_{j=1}^{m} x_i^{(j)}$$

$$\sigma_i^2 = \frac{1}{m} \sum_{j=1}^{m} (x_i^{(j)} - \mu_i)^2$$

2. $$y_i = \gamma_i \hat{x}_i + \beta_i$$

$$E(y_i) = \beta_i$$

$$var(y_i) = \gamma_i^2$$

m - training examples in mini-batch.

So, → Bias the solution to make features with mean = 0, variance = 1

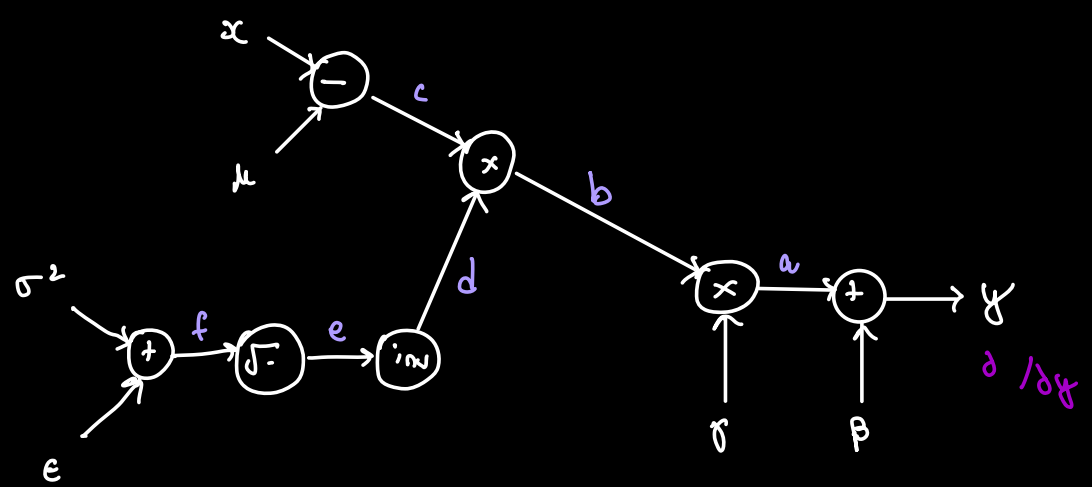
→ But also allow it to overcome this if the constraint is too much.

β_i, γ_i → learnable parameters.

→ less sensitive to weight initialization.

→ Applied separately to EACH NEURON.

BACKPROPAGATION:



$$y = a + \beta$$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial y}$$

$$a = \gamma b$$

$$\frac{\partial a}{\partial \gamma} = b \qquad \frac{\partial a}{\partial b} = \gamma$$

$$\frac{\partial L}{\partial b} = \frac{\partial a}{\partial b}\frac{\partial L}{\partial a} = \gamma \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial \gamma} = b \frac{\partial L}{\partial y}$$

$$d = \frac{1}{e}$$

$$\frac{\partial d}{\partial e} = \frac{-1}{e^2}$$

$$\frac{\partial L}{\partial c} = \frac{-1}{e^2} \cdot c\gamma \frac{\partial L}{\partial y}$$

$$e = \sqrt{f}$$

$$\frac{\partial e}{\partial f} = \frac{1}{2\sqrt{f}}$$

$$\frac{\partial L}{\partial f} = \frac{1}{2\sqrt{f}}\left(\frac{-1}{e^2}\right) c\gamma \frac{\partial L}{\partial y}$$

$$b = cd$$

$$\frac{\partial L}{\partial c} = d\gamma \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial d} = c\gamma \frac{\partial L}{\partial y}$$

$$c = x - \mu$$

$$\frac{\partial L}{\partial x} = d\gamma \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial \mu} = -d\gamma \frac{\partial L}{\partial y}$$

$$f = \sigma^2 + \epsilon$$

$$\frac{\partial L}{\partial \sigma^2} = \frac{1}{2\sqrt{f}}\left(\frac{-1}{e^2}\right) c\gamma \frac{\partial L}{\partial y}$$

$$= \frac{\partial L}{\partial \epsilon}$$

Each training data returns this gradient. Total is sum across them (Law of total gradient).

$$\frac{\partial L}{\partial \beta} = \sum_{i=1}^{m} \partial L / \partial y^{(i)}$$

μ contributes to σ^2

$$\frac{\partial L}{\partial \mu} = -\partial \gamma \sum_{j=1}^{m} \frac{\partial L}{\partial y^{(j)}} + \frac{\partial L}{\partial \sigma^2} \cdot \frac{\partial \sigma^2}{\partial \mu}$$

$$\sigma^2 = \frac{1}{m} \sum_{j=1}^{m} (x^{(j)} - \mu)^2$$

$$\frac{\partial \sigma^2}{\partial \mu} = \frac{-2}{m} \sum_{j=1}^{m} (x^{(j)} - \mu)$$

$$\frac{\partial L}{\partial \mu} = \frac{-1}{\sqrt{\sigma^2 + \epsilon}} \gamma \sum_{j=1}^{m} \frac{\partial L}{\partial y^{(j)}} + \frac{1}{m (\sigma^2 + \epsilon)^{3/2}} \sum_{j=1}^{m} (x - \mu)^2 \gamma \frac{\partial L}{\partial y^{(j)}}$$

$$\frac{\partial L}{\partial x^{(j)}} = \partial \gamma \sum_{j=1}^{m} \frac{\partial L}{\partial y^{(j)}} + \frac{\partial \sigma^2}{\partial x^{(j)}} \cdot \frac{\partial L}{\partial \sigma^2} + \frac{\partial \mu}{\partial x^{(j)}} \cdot \frac{\partial L}{\partial \mu}$$

$\downarrow 1/m$

$$2(x^{(j)} - \mu)/m$$

PROS OF BATCH NORMALIZATION:

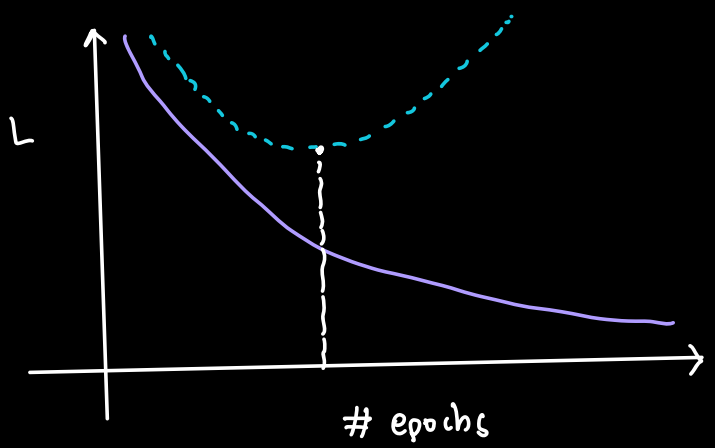
→ Allows higher learning rates (not sure update of other weights would impact, so smaller steps were taken)

→ Reduce strong dependencies on initialization.

REGULARIZATION:

Modification to reduce generalization error but not training error.

EARLY STOPPING:



PARAMETER NORM PENALTIES:

$$J(\theta; X, y) + \alpha \Omega(\theta)$$

→ L_2 REGULARIZATION:

$$\Omega(\theta) = \frac{1}{2} w^T w \qquad \frac{1}{2} \|w\|_F^2$$

Ridge Regression / Tikhonov Regularization

$$\frac{\partial \Omega(\theta)}{\partial w} = w$$

$$\text{Cost}: \quad \tilde{J}(w; X, y) = J(w; X, y) + \frac{\alpha}{2} w^T w$$

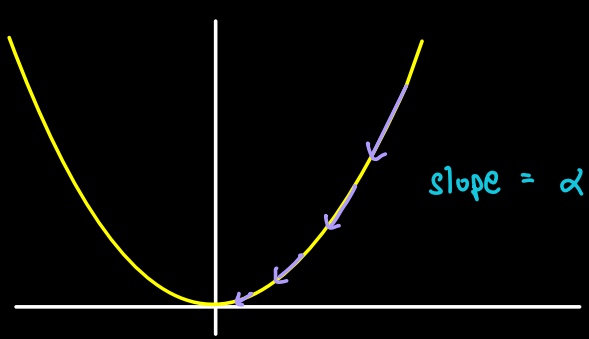
$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y)$$

$$w \leftarrow \underbrace{(1 - \epsilon\alpha) w}_{\text{Decreases weights}} - \epsilon \nabla_w J(w; X, y)$$

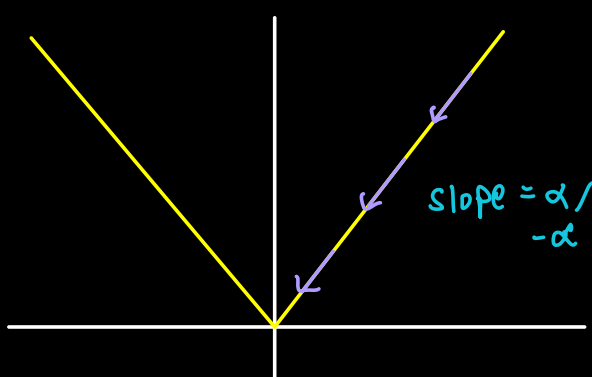
→ L1 REGULARIZATION:

$$\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$$

→ Tends to push more weights to 0.



2-NORM



1- NORM

Slope is constant, Doesn't decrease with weight.

So,

↓

$$w \leftarrow w - \epsilon\alpha w - \epsilon \nabla_w J$$

$$w \leftarrow w - \underbrace{\epsilon\alpha}_{\sim} - \epsilon \nabla_w J$$

→ Used for feature selection.

DATA AUGMENTATION:

→ Add cropped / flipped ... images

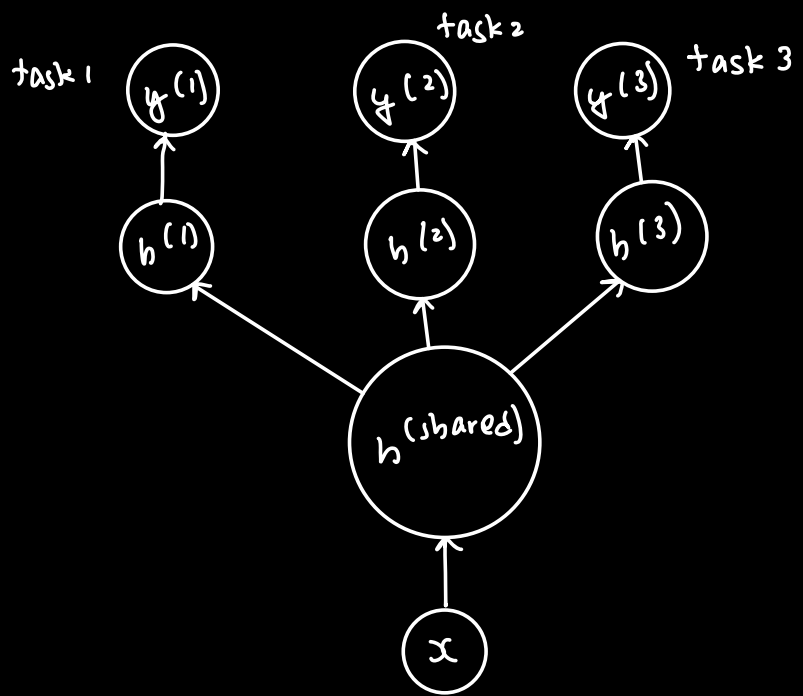
→ Inject noise

→ Label Smoothing

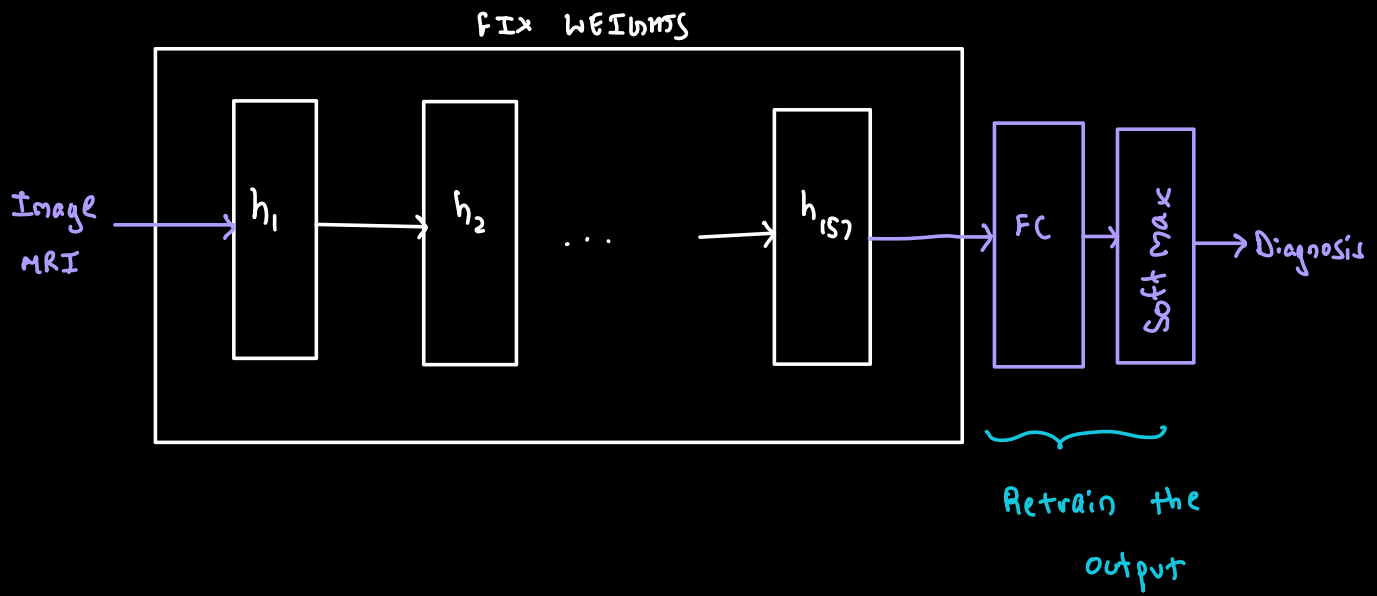
Instead of [0 0 ... 1 0 0 ... 0] (1 ← dog)

[0.011 0.011 ... 0.9 0.011 0.011 ... 0.01]

MULTITASK LEARNING:



TRANSFER LEARNING:



ENSEMBLE METHODS:

→ Train multiple different models

→ Average their results together.

k independent models

↳ Best case: $1/k$ decrease in error

↳ Worst case: Same error.

Pro:

→ Neural networks when repeatedly trained → independent errors

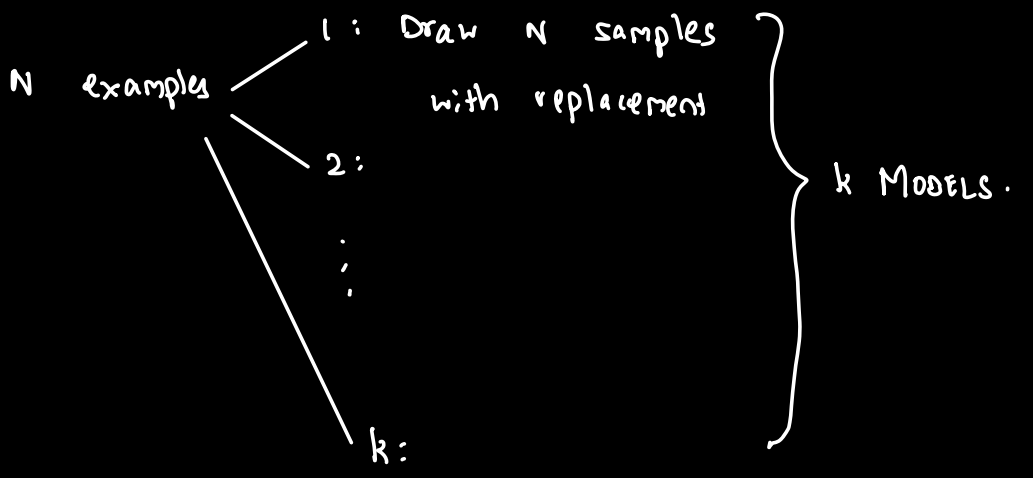
→ Almost always better accuracy.

Con:

→ Too much time to train.

Solution: Take suboptimal models. Set learning rate as cosine function and take each of the dip as a model.

→ Bagging (Bootstrap Aggregating):



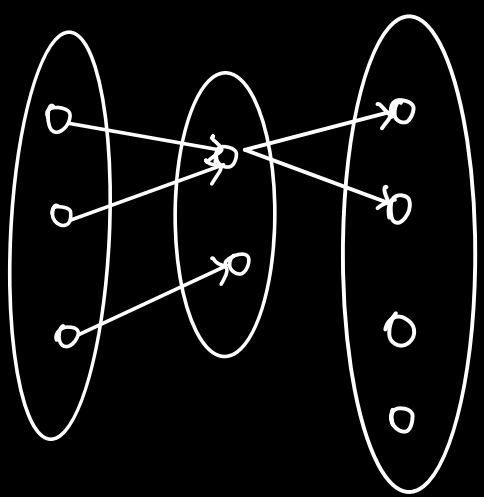
DROPOUT:

→ Approximation of Bagging

Binary Mask:

p → probability of seeing a 0.

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$ # of units / neurons.



mask → (9 x 1)

2^9 possible configurations.

Affine → relu → batchnorm → dropout

↓

this could affect batchnorm statistics.

→ What about test time?

Assume $p = 0.5$

What if we use all units?

→ over activate

Over many iterations, contributions of $w_i h_i$ to h_{out} is actually $(1-p) w_i h_i$.

$h_{out} = \text{relu}(w_i h_i) * (1-p)$

Pass p to test!

BETTER: INVERTED DROPOUT:

Scale at training:

$h_1 = (h_1 < (1-p)) / (1-p)$

PROS:

→ Estimates bagging

→ Regularizes each hidden unit to work well in different contexts.

→ Encode redundant features.

Stochastic Gradient Descent:

Loop

→ Sample m examples

→ $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ $\theta \leftarrow \theta - \varepsilon g$

Until stopping criterion is met.

Momentum:

→ Maintain running mean of the gradients.

$v = 0$

$\alpha \in [0, 1]$

Loop

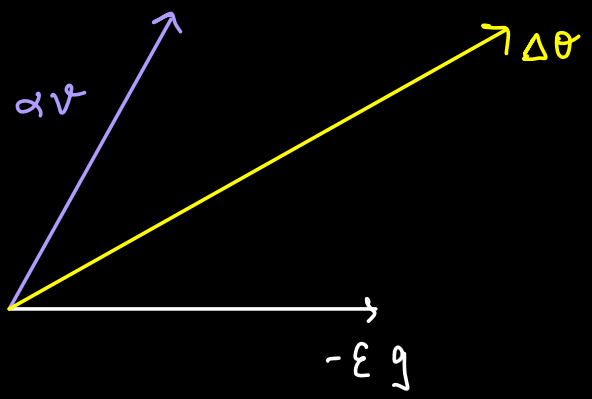
→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ $v \leftarrow \alpha v - \varepsilon g$

→ $\theta \leftarrow \theta + v$

Until stopping criterion is met.

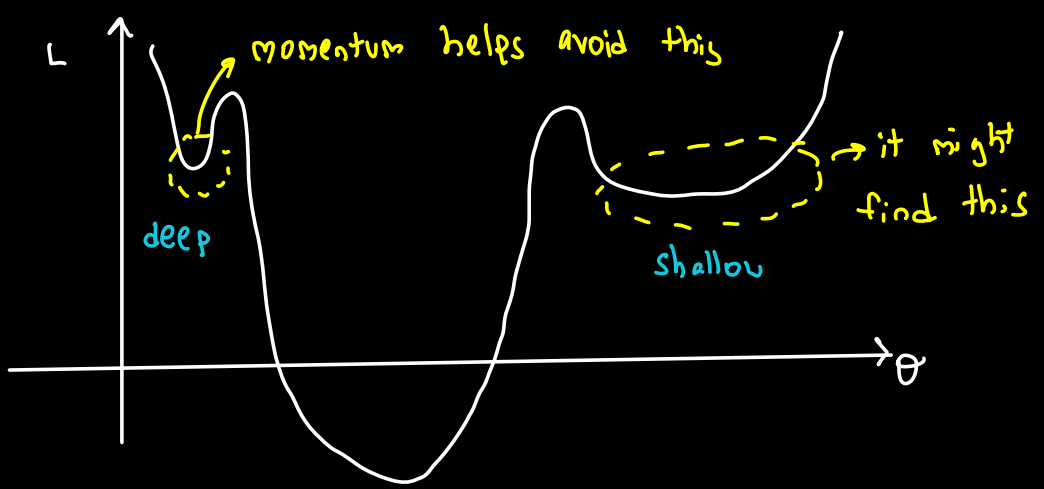
So, $v_4 = -(\alpha^3 g_1 + \alpha^2 g_2 + \alpha g_3 + g_4)$



PRO:

→ Solves zig-zagging

→ Helps to avoid deep local minima. Momentum pushes it



Shallow local minima: Small change in θ, doesn't change L much. Test data might by slightly different. So small changes in θ shouldn't affect much. This is Robust!

NESTEROV MOMENTUM:

$v = 0$

$\alpha \in [0, 1]$

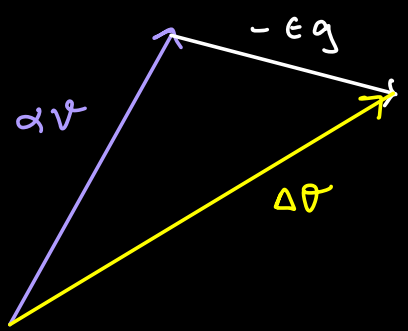
Loop

→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta + \alpha v)$

→ $v \leftarrow \alpha v - \varepsilon g$

→ $\theta \leftarrow \theta + v$

Until stopping criterion is met.



ADAPTIVE GRADIENT (ADAGRAD):

It we took larger steps in a particular direction, start taking smaller steps along that direction now.

$a = 0$

$\nu = 10^{-7}$

Loop

→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ $a \leftarrow a + g \odot g$ $\quad \odot \rightarrow$ element wise product

→ $\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{a} + \nu} \odot g$

Until stopping criterion is met.

n neurons / unit:

$g \in \mathbb{R}^m$

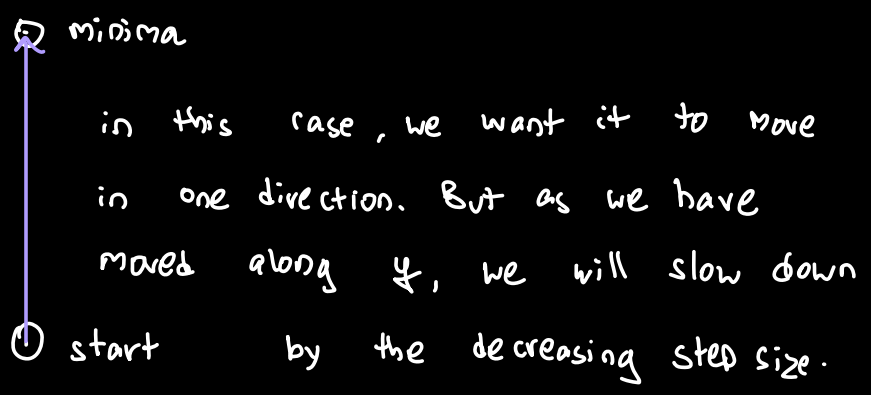
$$g \odot g = \begin{bmatrix} g_1^2 \\ g_2^2 \\ \vdots \\ g_n^2 \end{bmatrix}$$

$$\frac{\varepsilon}{\sqrt{a}+\nu} \odot g = \begin{bmatrix} \frac{\varepsilon}{\sqrt{a_1}+\nu} \cdot g_1 \\ \frac{\varepsilon}{\sqrt{a_2}+\nu} \cdot g_2 \\ \vdots \\ \frac{\varepsilon}{\sqrt{a_n}+\nu} \cdot g_n \end{bmatrix}$$

Con:

→ a always increases → step size always decreases.

No way for it to increase. Slower.



in this case, we want it to move in one direction. But as we have moved along y, we will slow down by the decreasing step size.

RMSProp:

Allows step size to increase.

$a = 0$

$\nu = 10^{-7}$

$\beta = [0, 1]$

Loop

→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ $a \leftarrow \boxed{\beta a + (1-\beta)\, g \odot g}$ ⟶ a can decrease.

→ $\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{a} + \nu} \odot g$

Until stopping criterion is met.

Pro:

Faster than Adagrad.

RMSProp + Momentum:

$v = 0$

$a = 0$

$\varepsilon = 10^{-7}$

$\beta = [0, 1]$

$\alpha = [0, 1]$

Loop

→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ $a \leftarrow \beta a + (1-\beta)\, g \odot g$

→ $v \leftarrow \alpha v - \frac{\varepsilon}{\sqrt{a} + v} \odot g$

→ $\theta \leftarrow \theta + v$

Until stopping criterion is met.

ADAM:

Momentum → shows the right direction

Adagrad → take more judicious steps in that direction.

Without bias correction:

$v = 0$ "First moment"

$a = 0$ "Second moment"

$\nu = 10^{-7}$

$\beta_1 = [0, 1]$

$\beta_2 = [0, 1]$

Loop

→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ $v \leftarrow \beta_1 v + (1 - \beta_1) g$

→ $a \leftarrow \beta_2 a + (1 - \beta_2) g \odot g$

→ $\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{a} + \nu} \odot v$

Until stopping criterion is met.

With bias correction:

To avoid taking larger steps at the starting.

$v = 0$ "First moment"

$a = 0$ "Seiond moment"

$\nu = 10^{-7}$

$\beta_1 = [0, 1]$

$\beta_2 = [0, 1]$

$t = 0$

Loop

→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ $t \leftarrow t + 1$

→ $v \leftarrow \beta_1 v + (1 - \beta_1) g$

→ $a \leftarrow \beta_2 a + (1 - \beta_2) g \odot g$

→ $\tilde{v} = \frac{1}{(1 - \beta_1^t)} v$

→ $\tilde{a} = \frac{1}{(1 - \beta_2^t)} a$

→ $\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{\tilde{a}} + \nu} \odot \tilde{v}$

Until stopping criterion is met.

As $t \to \infty$, $\tilde{v} \to v$

PRO:

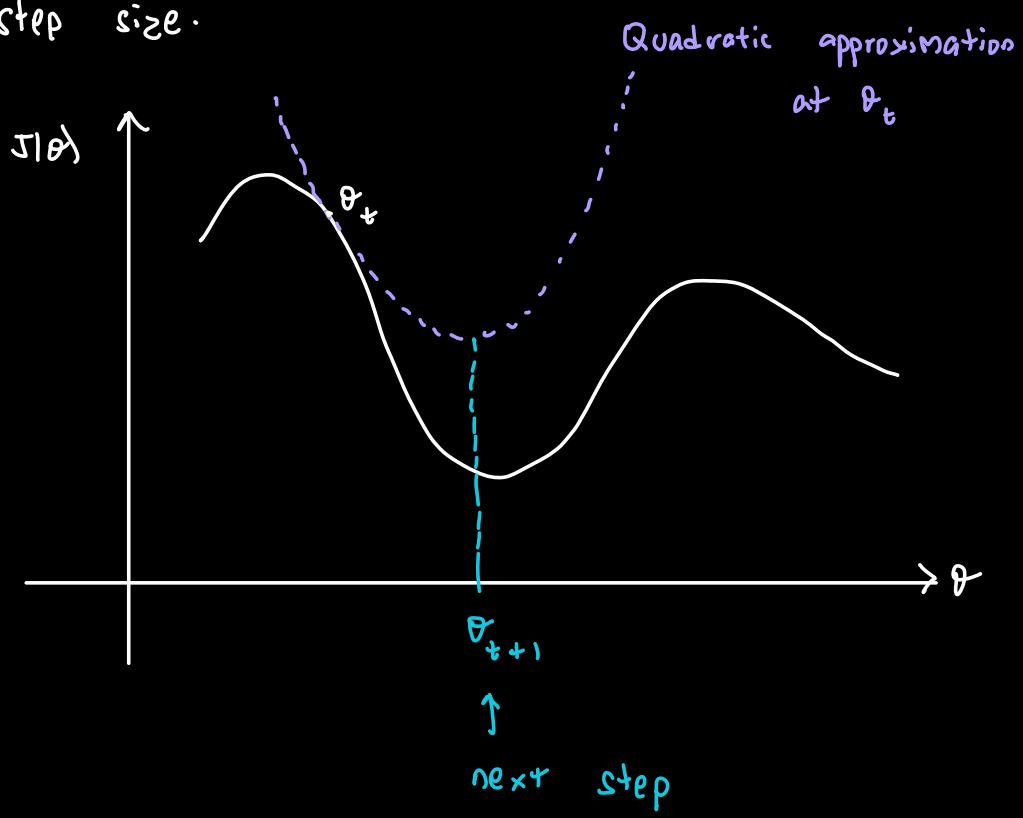
→ most used, best.

CON:

→ Might find steep local minima.

SECOND ORDER METHODS:

Use curvature of cost function to decide step size.



So deep $J(\theta)$ → small steps

shallow $J(\theta)$ → large steps.

Newton's Method:

Loop

→ Compute $g = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta)$

→ Compute Hessian, H

→ $\theta \leftarrow \theta - H^{-1} g$

Until stopping criterion is met.

Cons:

→ Hard to calculate H^{-1} (Computationally intensive)

→ Memory to store Hessian.

$n \approx 1 \times 10^6$

4 bytes

totally 3.8TB

→ H requires large batch size.

Challenges:

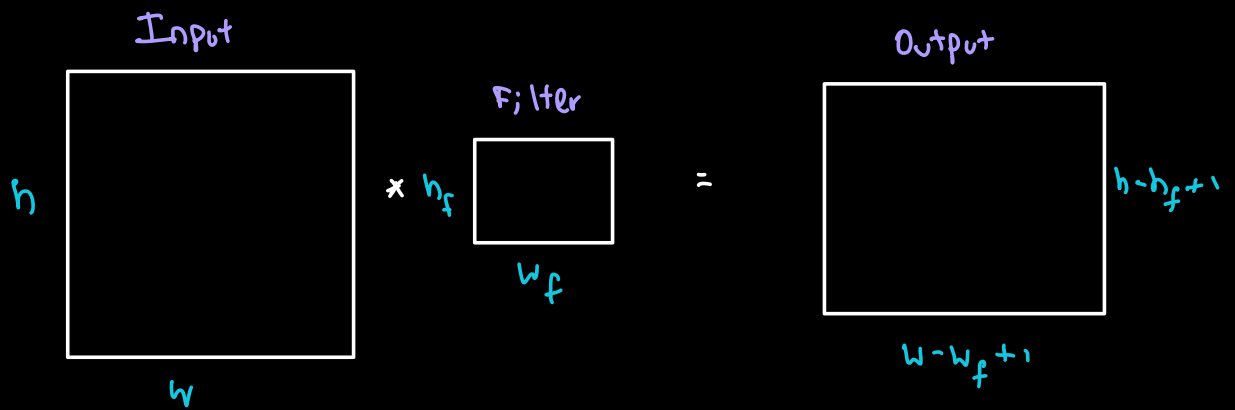
→ Exploding Gradients:

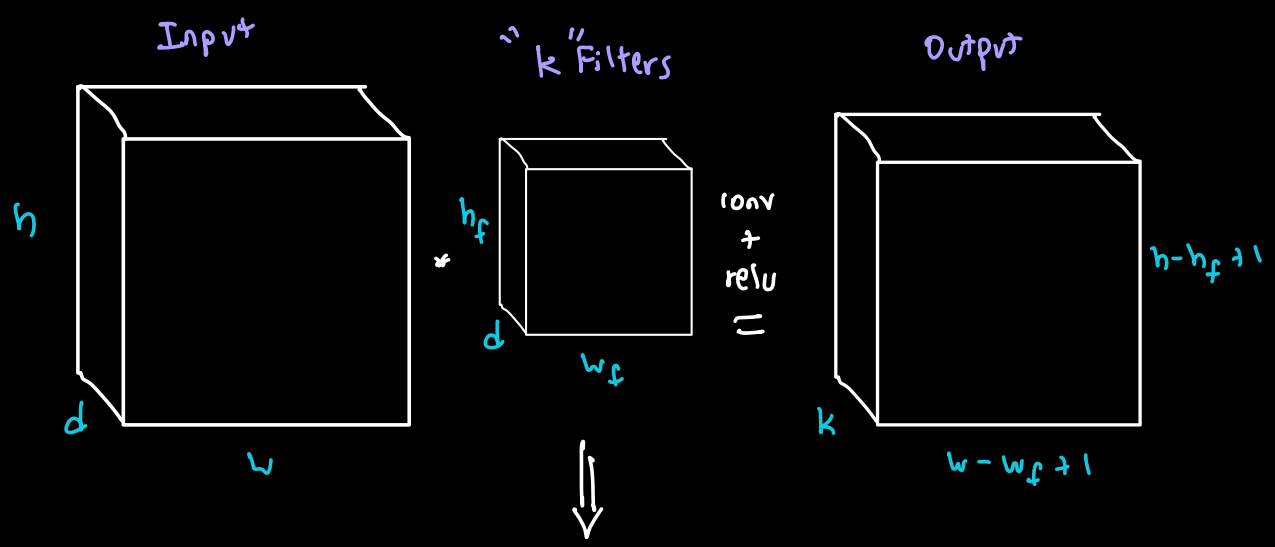
$$\|g\| \geq \text{clip}$$

$$g \leftarrow \frac{g}{\|g\|} \cdot \text{clip}.$$

CONVOLUTIONAL NEURAL NETWORK :

→ Fully Connected Network has a lot of parameters.





common for all the units of input

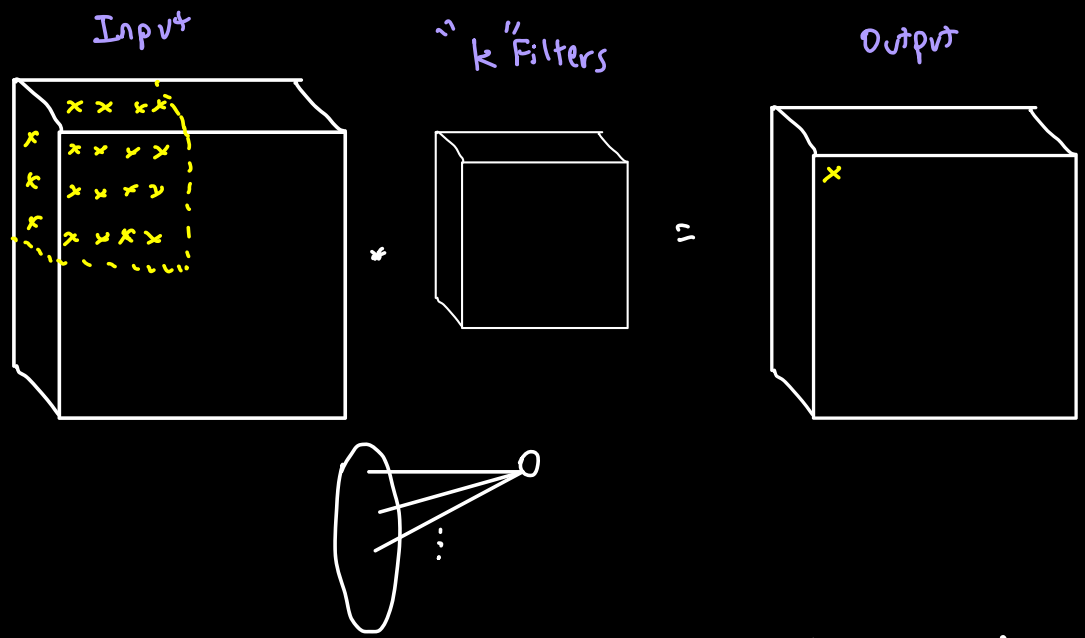
"Shared parameters"

Sparsely Connected Layers :

Unlike Fully Connected Network, the top left of

output depends only on top left of input.

Number of neurons it depends on is decided by the filter.



Fully connected layer : $w \times h \times d$ weights per neuron.

Conv : $w_f \times h_f \times d$ weights per neuron.

PRO :

→ Less Computational memory

→ Less Computational time.

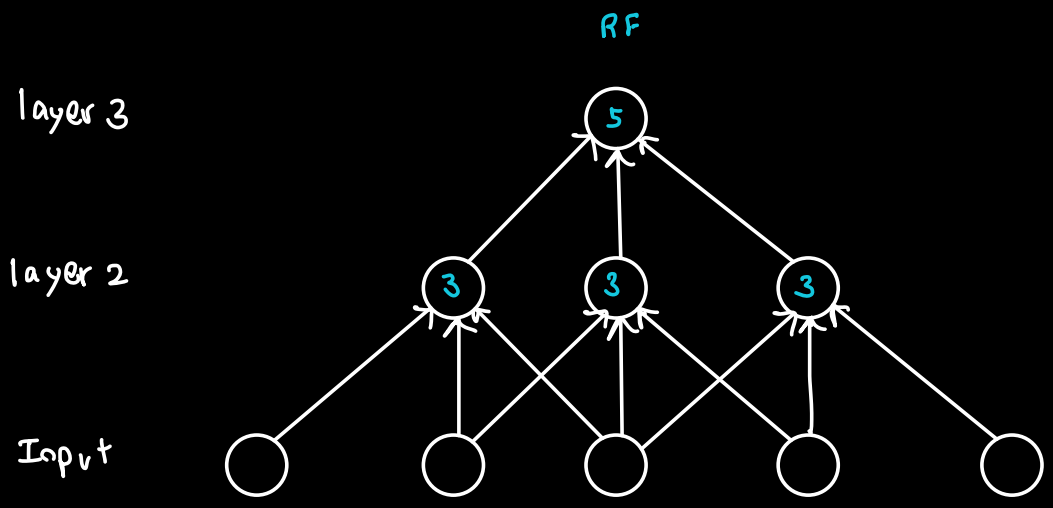
CON :

→ No neuron may have seen all of the input.

Solution : More layers.

RECEPTIVE FIELD:

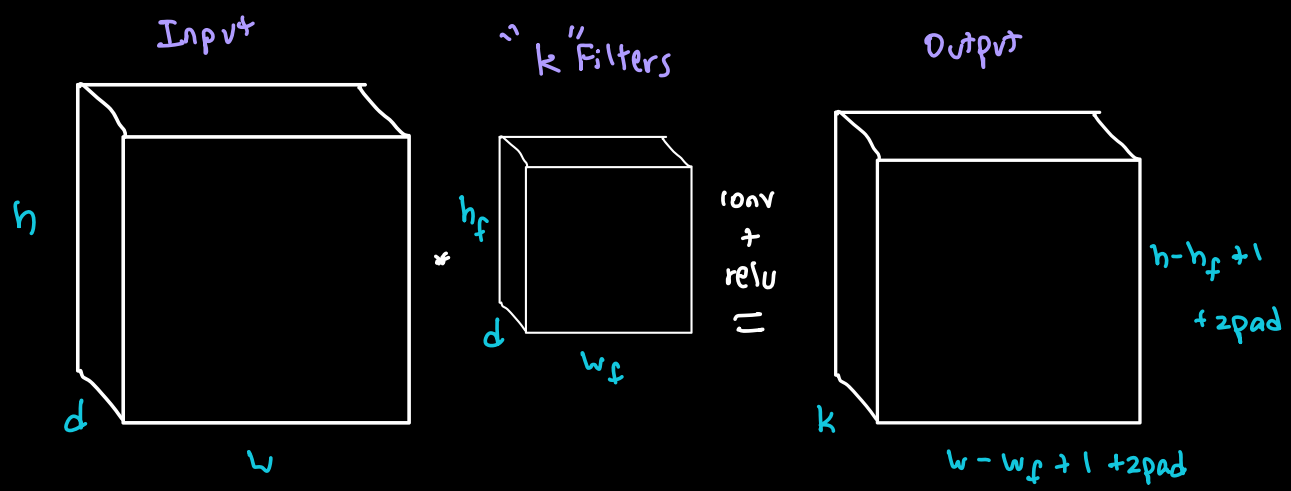
Amount of input the neuron sees.



2 ARCHITECTURES

	FN	CNN
Input:	$32 \times 32 \times 3 = 3072$	$32 \times 32 \times 3$
	↓	↓ 100 filters $4 \times 4 \times 3$
Output / feature maps	500 output	$29 \times 29 \times 100 = 84,100$
Parameters:	$3072 \times 500 = 1.5$ million $+1$ ↑ bias	$100 \times (4 \times 4 \times 3 + 1)$ ↑ bias $= 4900$

PADDING:



+ pad

if $w_f = h_f = 3$, set pad = 1

output and input → same h, w.

STRIDE:

In 7×7 → we can do stride = 1, 2 or 4.

not 3.

$$\left(\frac{w - w_f + 2pad}{stride} + 1 , \frac{h - h_f + 2pad}{stride} + 1 \right)$$

Pooling Layer:

(w_p, h_p) filter with a stride.

Operation: max() / avg().

$$\left(\frac{w - w_p}{stride} + 1, \frac{h - h_p}{stride} + 1, d \right)$$

Pad:

→ Introduce spacial invariances.

If data is say shifted one pixel, Pooling layer's output for most cases will remain the same.

LeNet-5:

Handwriting Recognition.

$32 \times 32 \xrightarrow{6 \ (5 \times 5) \text{ filters}} 28 \times 28 \times 6$

Number of connections = each output has $(5 \times 5 + 1)$ connections to input

$= 26 \times 28 \times 28 \times 6$

$= 122,304$ //

AlexNet:

$227 \times 227 \times 3$

ZF Net:

→ Smaller filters, smaller strides - earlier layers

→ more filters in deeper layers.

VGGNet:

Use small filters.

→ Lesser weights

→ More can be stacked to get same receptive field → more activations.

1. $\frac{\partial}{\partial W} tr(WA) = A^T$

$\frac{\partial}{\partial W} tr(WAW^T) = WA^T + WA$

$\frac{\partial}{\partial W} tr(ABC) = \frac{\partial}{\partial W} tr(CBA) = \frac{\partial}{\partial W} tr(BAC) = \frac{\partial}{\partial W} tr(ABC)^T$

2. $XY = Z \qquad X = ZY^{-1}$ not $Y^{-1}Z$

$\hookrightarrow XYY^{-1} = ZY^{-1}$

3. $\partial/\partial x (x^T y) = y$

$\partial/\partial x (Wx) = W^T$

4. $y = Wx$

$\frac{\partial \ell}{\partial x} = W^T \frac{\partial \ell}{\partial y}$

$\frac{\partial \ell}{\partial W} = \frac{\partial \ell}{\partial y} \cdot x^T$

5. $Y = WX$

$d \times N \quad W \times D \quad d \times D$

$$\frac{\partial \ell}{\partial X} = W^T \frac{\partial \ell}{\partial Y}$$

$$\frac{\partial \ell}{\partial W} = \frac{\partial \ell}{\partial Y} X^T$$

6. $y = x^T A x$

$$\frac{\partial y}{\partial x} = x^T (A + A^T)$$

$y = x^T a$

$\partial y / \partial x = a$

7. $z = x^T A y$

$$\frac{\partial z}{\partial x} = Ay$$

$$\frac{\partial z}{\partial y} = x^T A$$

$$\frac{\partial z}{\partial A} = x y^T$$

8. $\frac{\partial y}{\partial x^T} = \left(\frac{\partial y}{\partial x}\right)^T$

9. $y = |X|$

$\partial y / \partial X = |X| (X^{-1})^T$

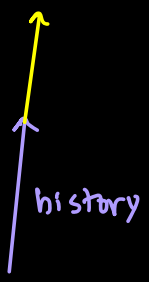
10. $\frac{\partial (x x^T)}{\partial x} = \frac{\partial (x^T x)}{\partial x} = 2x$

$\frac{\partial L}{\partial x} = 2 \left(\frac{\partial L}{\partial x x^T}\right) x$

11. $\frac{\partial}{\partial x} tr(A) = tr\left(\frac{\partial A}{\partial x}\right)$

12. $\frac{\partial}{\partial p} \|p\|_2 = \frac{p}{\|p\|_2}$

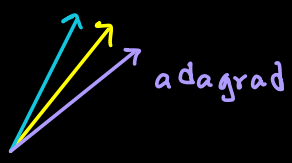
SGD



Momentum

RMSProp

Adam



RMSProp

↳ bias towards 0 in the starting.

Hw1:

$$\hat{y} = Wx$$

$$L = \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - Wx^{(i)}\right)^T \left(y^{(i)} - Wx^{(i)}\right)$$

$$Y - WX = \begin{bmatrix} \vdots & \vdots & & \vdots \\ y^{(1)} & y^{(2)} & \cdots & y^{(n)} \\ \vdots & \vdots & & \vdots \end{bmatrix} - W \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \cdots & x^{(n)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$= \begin{bmatrix} \vdots & \vdots & & \vdots \\ (y^{(1)} - Wx^{(1)}) & (y^{(2)} - Wx^{(2)}) & \cdots & (y^{(n)} - Wx^{(n)}) \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$(Y - WX)^T (Y - WX) = \begin{bmatrix} (y_1 - Wx_1)^T (y_1 - Wx_1) & (y_1 - Wx_1)^T (y_2 - Wx_2) & \cdots \\ & (y_2 - Wx_2)^T (y_2 - Wx_2) & \\ & & \ddots \end{bmatrix}$$

$$L = \frac{1}{2} \operatorname{tr} (Y - WX)^T (Y - WX) = \frac{1}{2} \operatorname{tr} \left(Y^T - X^T W^T\right)(Y - WX)$$

$$= \frac{1}{2} tr\left(y^T y - y^T W x - x^T W^T y + x^T W^T W x\right)$$

$$\frac{\partial L}{\partial W} = 0$$

$$\frac{\partial}{\partial W} tr\left(y^T W x\right) = \frac{\partial}{\partial W} tr\left(W x y^T\right)$$

$$y x^T$$

$$\frac{\partial}{\partial W} tr\left(x^T W^T y\right) = \frac{\partial}{\partial W} tr\left(y^T W x\right)$$

$$y x^T$$

$$\frac{\partial}{\partial W} tr\left(x^T W^T W x\right)$$

$$W x x^T + W x x^T$$

$$2 W x x^T$$

$$\Rightarrow \quad W x x^T - y x^T = 0$$

$$yx^T = Wxx^T$$

$$W = (yx^T)(xx^T)^{-1}$$

HW2:

$$Pr(y^{(j)} = i \mid x^{(j)}, \theta) = \text{softmax}_i(x^{(j)})$$

$$MLE: P(x^{(j)}, y^{(j)} \mid \theta)$$

$$= P(x^{(j)} \mid \theta) \cdot P(y^{(j)} \mid x^{(j)} \mid \theta)$$

$$\log L = \log P(y^{(j)} \mid x^{(j)} \mid \theta)$$

$$= (W_{y^{(j)}}^T x + b_{y^{(j)}}) - \log \sum_{k=1}^{C} (e^{W_k^T x + b_k})$$

if $i = y^{(i)}$

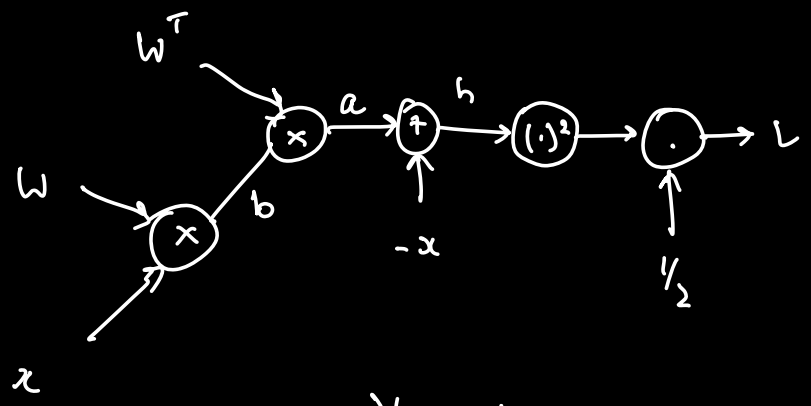
$$\frac{\partial L}{\partial W_i} = x - \frac{1}{\sum_{k=1}^{C} e^{W_k^T x}} \times e^{W_i^T x} \cdot x$$

else

$$\frac{\partial L}{\partial W_i} = \frac{-1}{\sum_{k=1}^{C} e^{W_k^T x}} \times e^{W_i^T x} \cdot x$$

HW3:

1. $L = \frac{1}{2} \| W^T W x - x \|^2$



$$\frac{\partial L}{\partial h} = h$$

$$\frac{\partial L}{\partial x} = -\left(\frac{\partial L}{\partial h}\right) = -h$$

$$\frac{\partial L}{\partial a} = h \qquad a = W^T b$$

$$\frac{\partial L}{\partial b} = W \frac{\partial L}{\partial a} \qquad \frac{\partial L}{\partial W^T} = \frac{\partial L}{\partial a} \cdot b^T$$

$$= Wh \qquad = hb^T$$

$$\frac{\partial L}{\partial W} = bh^T$$

$$y = Wx$$

$$\frac{\partial L}{\partial x} = W^T \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y} \cdot x^T$$

$$b = Wx$$

$$\frac{\partial L}{\partial x} = W^T W b \qquad \frac{\partial L}{\partial W} = W b x^T$$

$$\partial L / \partial W = W(W^T W x - x)x^T + Wx(W^T W x - x)^T$$

2)

$$\frac{\partial L_1}{\partial a} = \frac{-D}{2}$$

$$a = \log b$$

$$\frac{\partial a}{\partial b} = \frac{1}{b}$$

$$\frac{\partial L_1}{\partial b} = \frac{\partial a}{\partial b} \frac{\partial L_1}{\partial a} = \frac{1}{b} \cdot \left(\frac{-D}{2}\right)$$

$$b = |k|$$

$$\partial b / \partial k = |k| \, (k^{-1})^T$$

$$\partial L / \partial k = \partial L / \partial b \; \partial b / \partial k$$

$$= |k| \, (k^{-1})^T \left(\frac{-D}{2}\right)$$

$$\partial L / \partial \theta = \partial L / \partial \eta \; \partial \eta / \partial \theta$$

$$\partial L / \partial \eta = \alpha \, \partial L / \partial k$$

$$f = x x^T$$

$$\eta = M x$$

$$\frac{\partial L}{\partial x} = M^T \frac{\partial L}{\partial \eta}$$

$$x^T \cdot \frac{\partial L}{\partial \eta} = \frac{\partial L}{\partial M}$$

$$\frac{\partial L}{\partial x^T} = x^T \frac{\partial L}{\partial f}$$

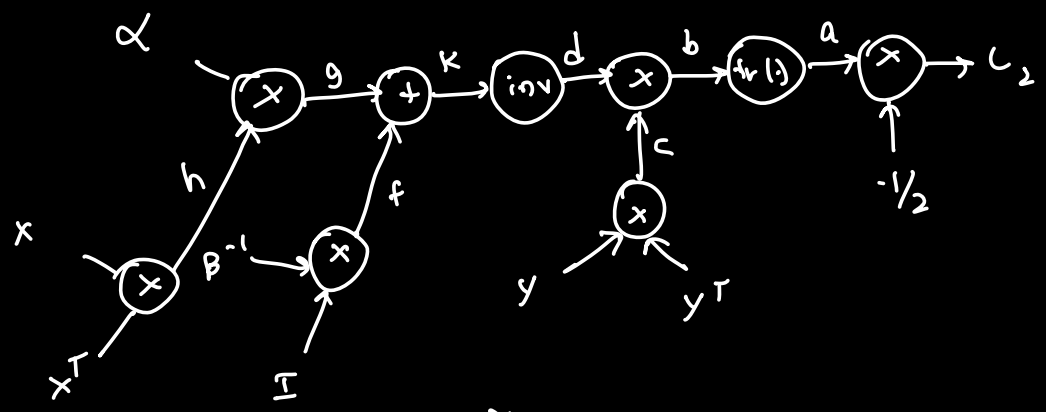
$$x \cdot \frac{\partial L}{\partial f} = \frac{\partial L}{\partial x}$$

$$\frac{\partial L}{\partial x^T} = x^T \alpha |k| (k^{-1})^T \left(\frac{-D}{2}\right)$$

$$= \left(\frac{-\alpha D}{2}\right) |k| \, x^T (k^{-1})^T$$

$$\frac{\partial L}{\partial x} = \left(\frac{-\alpha D}{2b}\right)|k|\,(k^{-1})^T x$$

$$+\left(\frac{-\alpha D}{2b}\right)|k|(k^{-1})\,x$$



$$\frac{\partial L_2}{\partial a} = -1/2 \qquad d = e^{-1}$$

$$a = tr(b)$$

$$\frac{\partial a}{\partial b} = tr\left(\frac{\partial b}{\partial b}\right) = I$$

$$\frac{\partial L}{\partial k} = -k^{-T}\frac{\partial L}{\partial d}k^{-T}$$

$$= -k^{-T}yy^T\left(\frac{-I}{2}\right)k^{-T}$$

$$\frac{\partial L_2}{\partial b} = -I/2$$

$$\frac{\partial L}{\partial g} = \frac{\partial L}{\partial k}$$

$$\frac{\partial L_2}{\partial d} = yy^T\left(\frac{-I}{2}\right)$$

$$\frac{\partial L}{\partial h} = \alpha\frac{\partial L}{\partial k}$$

$$\frac{\partial L}{\partial x} = 2x$$

$$\frac{\partial L}{\partial x} = \frac{\partial h}{\partial x}\frac{\partial L}{\partial h} = 2\alpha\left(+K^{-T}xy^T\frac{I}{2}K^{-T}\right)$$

$$= \alpha \quad K^{-T}xy^TK^{-T}x$$

$L_1 \to 1\times 1$ $\quad$ $m\times n$

$$h = Ax + b$$

$m\times 1$ $\quad$ $n\times 1$ $\quad$ $m\times 1$

hh^T $\quad$ $m\times m$

$m\times 1$ $\quad$ $1\times m$ $\quad$ I

$$\frac{\partial L_i}{\partial a} = 1$$

$$a = \log b$$

$$\frac{\partial L_i}{\partial b} = \frac{\partial a}{\partial b} \frac{\partial L_i}{\partial a} = (b^T)^{-1}$$

$$\frac{\partial L_i}{\partial c} = (b^T)^{-1}$$

$$c = hh^T$$

$$\frac{\partial L_i}{\partial h} = 2 \frac{\partial L_i}{\partial c} h = 2 (b^T)^{-1} h$$

$$h = \mathrm{ReLU}(\underbrace{Wx+b}_{a})$$

$$h = \mathrm{ReLU}(a)$$

$$\frac{\partial h}{\partial a} = 1 \quad (a>0)$$

$$\mathbb{I}(a>0) \qquad \frac{\partial L_i}{\partial a} = \mathbb{I}(a>0) \cdot 2 (b^T)^{-1} h \times 2g$$

$$a = Wx+b$$

$$\frac{\partial L_i}{\partial x} = W^T \mathbb{I}(a>0) \cdot 2 (b^T)^{-1} h \times 2g$$

$$\frac{\partial l_i}{\partial W} = \mathbb{I}(a > 0) \cdot 2(b^T)^{-1} h x^T \times 2g$$

6) a) $27 \times 27 \times 96$

$3 \times 3 \times 96$

$3 \times 3 \times 96 \times 384$

$3 \times 3 \times 384 \times 384$

$3 \times 3 \times 384 \times 256$

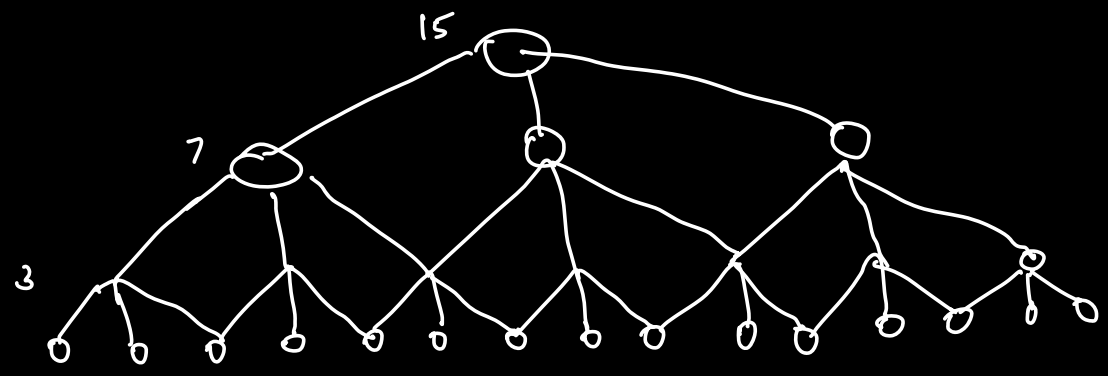
$27 \times 27 \times 256$

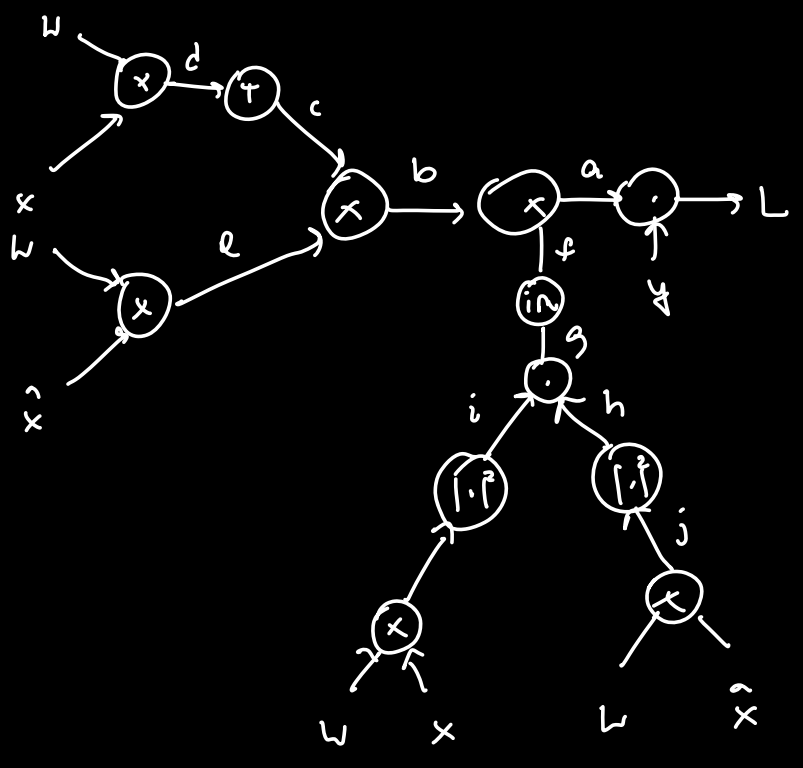
$6 \times 6 \times 256 \quad \times \quad 4096$

$27 \times 27 \times 256 \times 4$

$6 \times 6 \times 256 \times 4$

4096×4





$$g = \frac{1}{f}$$

$$\frac{\partial L}{\partial f} = by$$

$$\frac{\partial L}{\partial g} = \left(\frac{-1}{f^2}\right) by$$

$$\frac{\partial L}{\partial h} = \left(\frac{-1}{f^2}\right) by$$

$$h = |j|^2$$

$$\frac{\partial h}{\partial j} = 2j$$

$$\frac{\partial L}{\partial a} = y$$

$$a = bf$$

$$\frac{\partial a}{\partial b} = f \qquad \frac{\partial L}{\partial b} = \frac{\partial a}{\partial b}\frac{\partial L}{\partial a} = fy$$

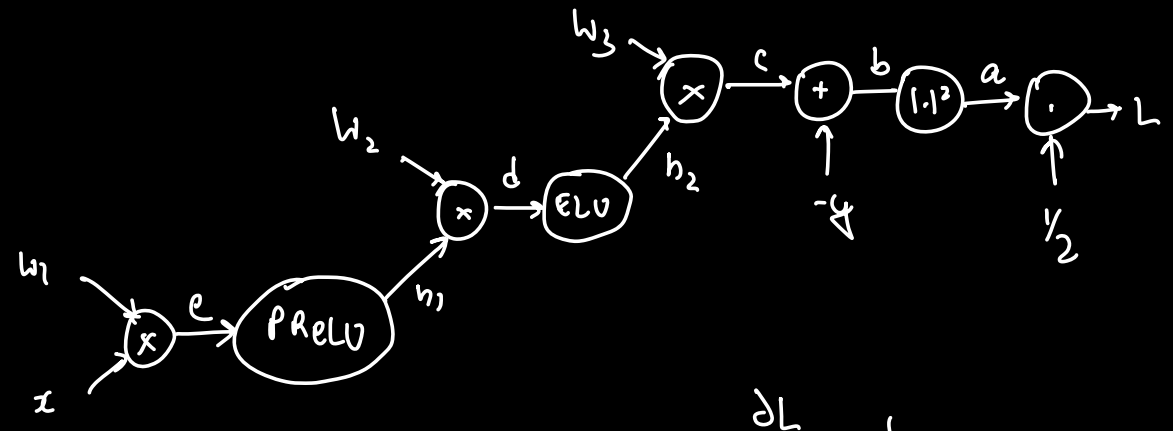
$$b = cq$$

$$\frac{\partial L}{\partial c} = \frac{\partial L}{\partial q} \cdot e^T \qquad \frac{\partial L}{\partial e} = c^T \frac{\partial L}{\partial q}$$

$$= fye^T \qquad = c^T fy$$

$$y = Wx$$

$$\frac{\partial L}{\partial x} = W^T \frac{\partial L}{\partial y} \qquad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial y} \cdot x^T$$



$$\frac{\partial L}{\partial a} = \frac{1}{2}$$

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

$$\frac{\partial L}{\partial c} = b$$

$$c = w_3 h_2$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial c} \cdot h_2^T$$

$$= b h_2^T$$

$$\frac{\partial L}{\partial h_2} = w_3^T \frac{\partial L}{\partial c}$$

$$= w_3^T b$$

$$\frac{\partial L}{\partial d} = \frac{\partial h_2}{\partial d} \frac{\partial L}{\partial h_2}$$

$h_2 = ELU(d)$

$$\frac{\partial h_2}{\partial d} = \begin{cases} \alpha e^d & , d < 0 \\ 1 & , d \geq 0 \end{cases}$$

$$\frac{\partial h_2}{\partial d} = I(d \geq 0) + I(d < 0)\alpha e^d$$

$$\frac{\partial L}{\partial d} = \frac{\partial h_2}{\partial d} \cdot W_3^T b$$

$d = W_2 h_1$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial d} \cdot h_1^T \qquad \frac{\partial L}{\partial h_1} = W_2^T \frac{\partial L}{\partial d} = W_2^T \frac{\partial h_2}{\partial d} W_3^T b$$

$h_1 = PReLU(e)$

$$\frac{\partial h_1}{\partial e} = \begin{cases} \alpha, & e < 0 \\ 1, & e \geq 0 \end{cases}$$

$$\frac{\partial L}{\partial e} = \left(I(e<0)\alpha + I(e \geq 0)\right) W_2^T \left(I(d \geq 0) + I(d<0)\alpha e^d\right) W_3^T b$$

$\rightarrow (c - y)$

$$e = W_1 x$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial e} \cdot x^T \qquad \frac{\partial L}{\partial x} = W_1^T \frac{\partial L}{\partial e}$$