Machine Learning

Lecture 13

Intro to Neural Networks

Anna Kuzina

Learning Outcomes

After this lecture you should know:

- What is an artificial neuron, perceptron and artificial neural network
- What is a dense layer
- Which activation function are used
- How to train neural networks

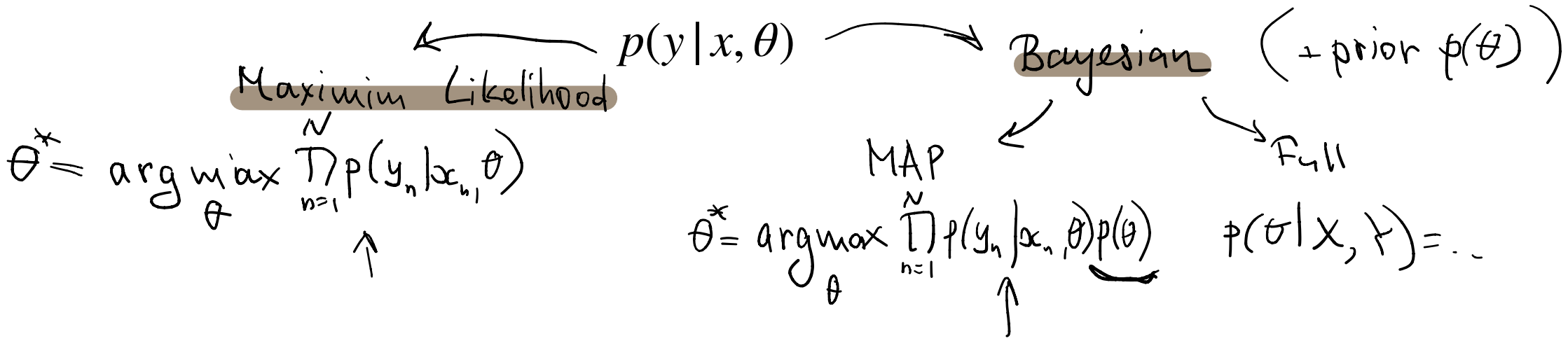
Recap Discriminative models

Data:

$\{x_1, \ldots, x_n\}$ - observed variables

$\{y_1, \ldots, y_N\}$ - unobserved / target variables

Model with unknown parameters θ:



Recap Linear Regression

Data:

$\{x_1, \ldots, x_n\}$ - observed variables

$\{y_1, \ldots, y_N\}$ - unobserved / target variables

Model with unknown parameters θ:

$$p(y \mid x, \theta) = \mathcal{N}(y \mid w^T x, \sigma^2)$$

ML: $w^* = \arg\min_w \|Xw - y\|^2$

Recap Logistic Regression

Data:

$\{x_1, \ldots, x_n\}$ - observed variables

$\{y_1, \ldots, y_N\}$ - unobserved / target variables $\in \{0, 1\}$

Model with unknown parameters θ: $\{w\}$

- $p(y \mid x, \theta) = \mathrm{Be}\left(y \mid \mathrm{sigm}(w^T x)\right)$

$\mathrm{sigm}(x) = \frac{1}{1 + \exp(-x)}$

$$p(y=1 \mid x, w) = \mathrm{sigm}(w^T x)$$

$$w^* = \arg\max_w \sum_{n=1}^{N} \log p(y_n \mid x_n, w)$$

Recap Discriminative models

Linear Regression:

$$p(y \mid x, \theta) = \mathcal{N}(y \mid w^T x, \sigma^2)$$

Logistic Regression:

$$p(y \mid x, \theta) = \mathcal{B}e(y \mid \text{sigm}(w^T x))$$

Can we parametrise distributions in other ways?

Recap Discriminative models

Linear Regression:

$$p(y \mid x, \theta) = \mathcal{N}(y \mid w^T x, \sigma^2)$$

Logistic Regression:

$$p(y \mid x, \theta) = \mathcal{B}e(y \mid \text{sigm}(w^T x))$$

Can we parametrise distributions in other ways?

Regression:

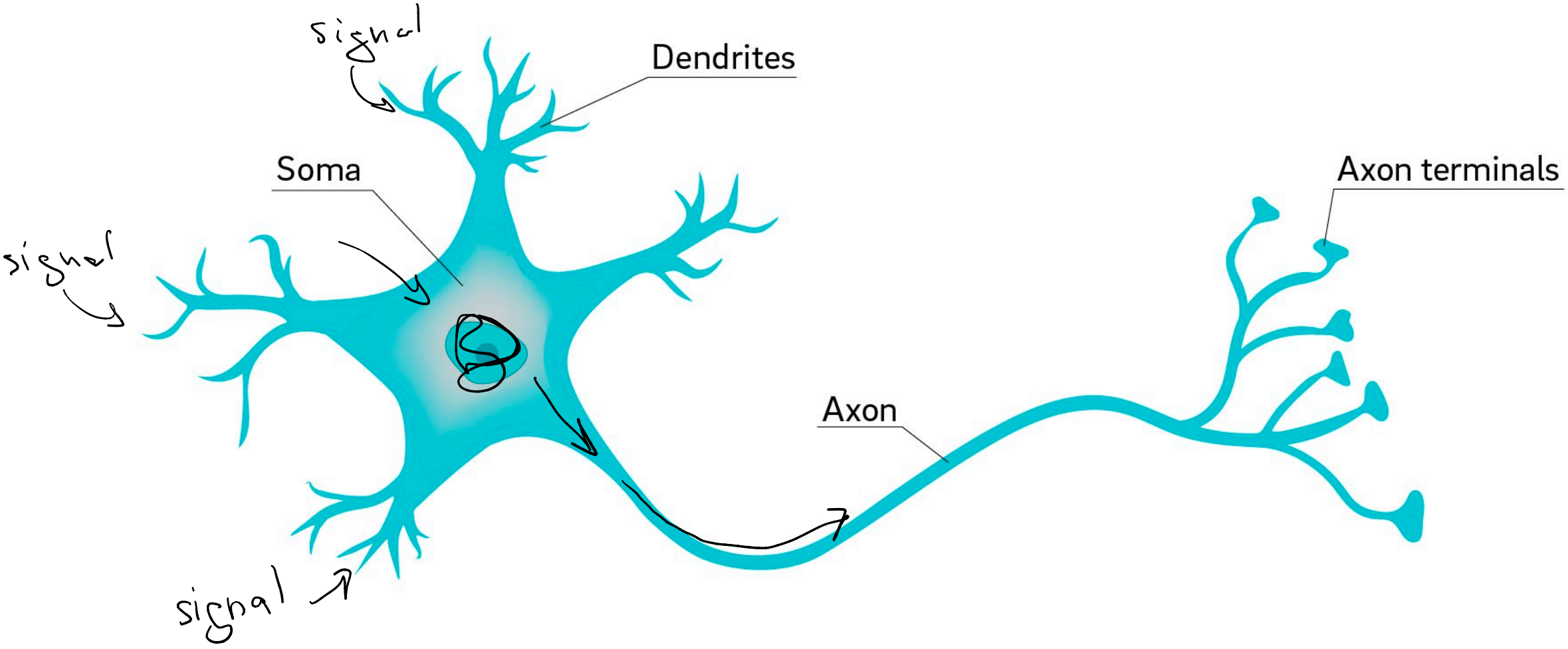
$$p(y \mid x, \theta) = \mathcal{N}(y \mid f(x), \sigma^2)$$

Binary Classification:

$$p(y \mid x, \theta) = \mathcal{B}e(y \mid f(x))$$

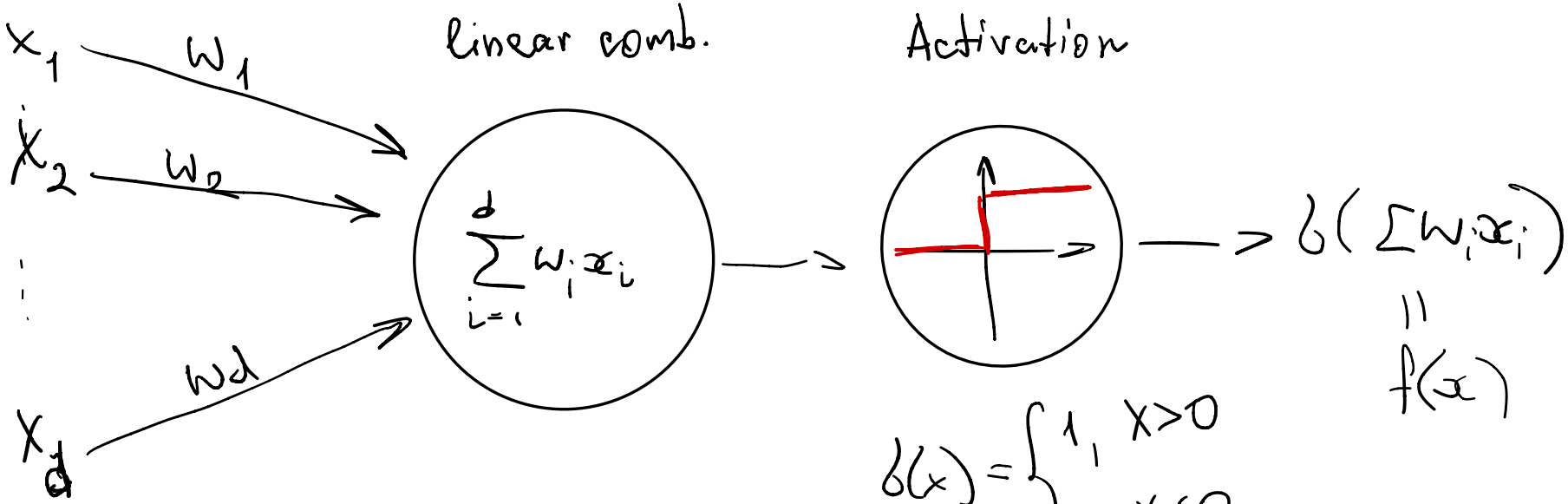
But we need to pay attention to the constraints!

Artificial Neuron: Inspiration



Artificial Neuron: Perceptron

$X = (x_1, \ldots, x_d)$



$\sigma(x) = x \Rightarrow$ Linear Regr

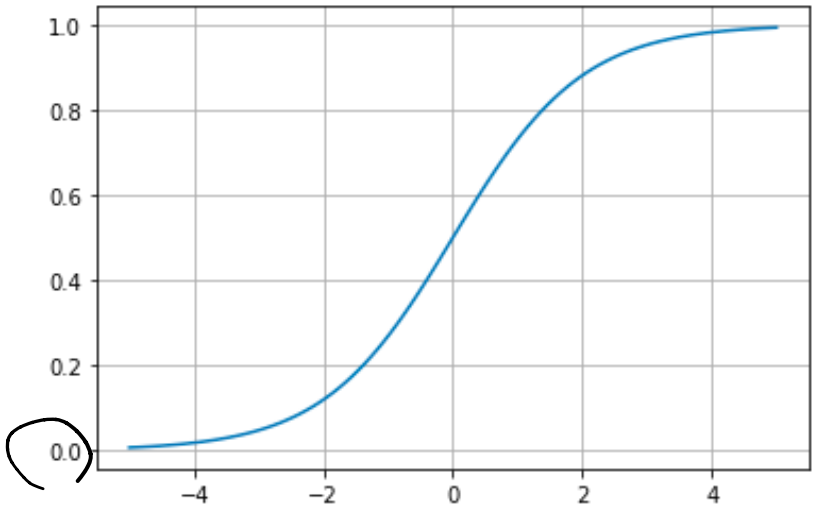
$\sigma(x) = \text{sigm}(x) \Rightarrow$ Log. regr

Artificial Neuron: Logistic Regression

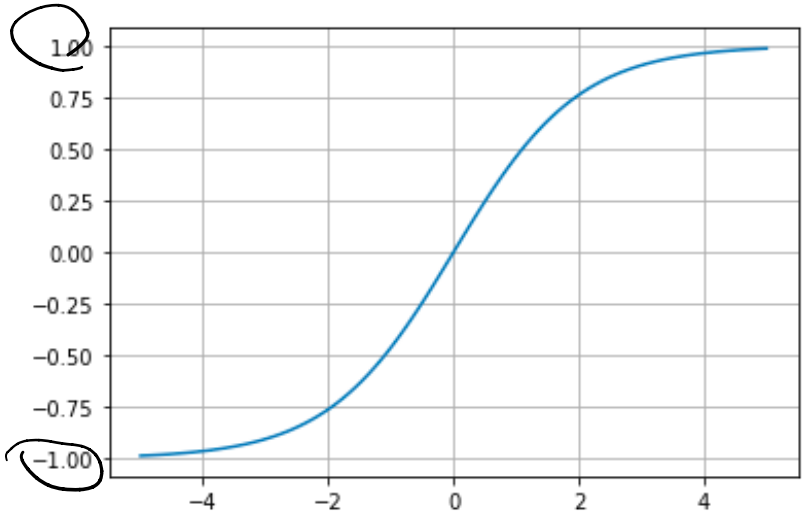
$$f(x) = sigm(w^T x) \Rightarrow \text{Artifical Neuron} \equiv \text{Log. Reg}$$

Artificial Neuron: Activation Functions

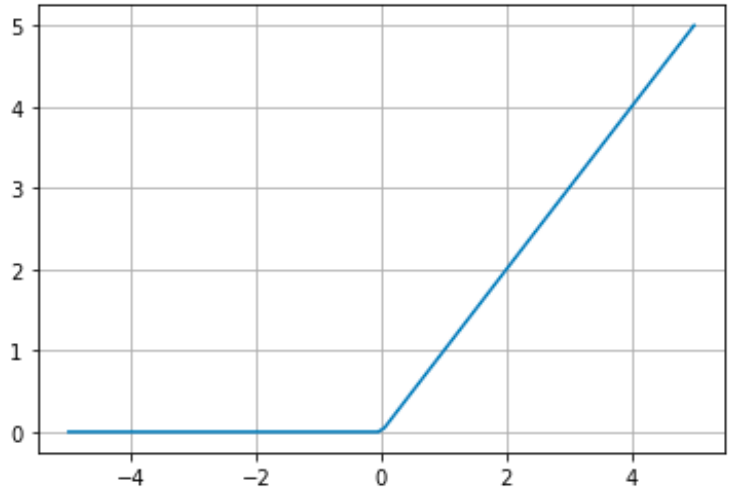
sigmoid



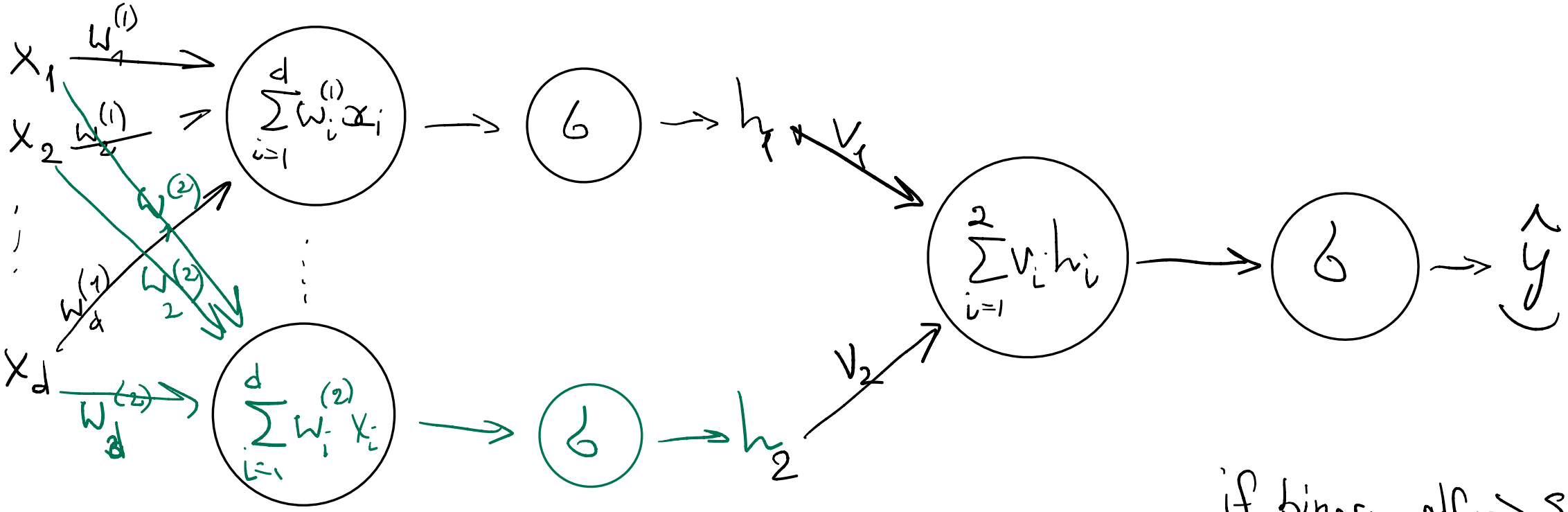
tanh



ReLU



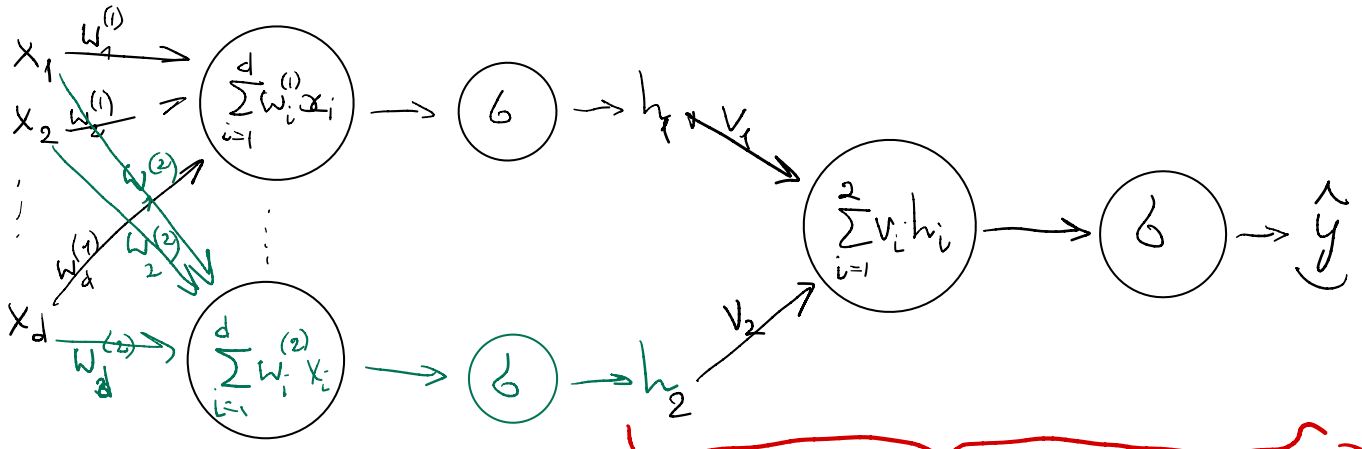
Artificial Neural Network (sceme)



if binary clf => sigmoid

if Regression => $\beta(x) = x$

Artificial Neural Network (formula)



First (hidden) layer

Second (Output) Layer

Input: $h \in \mathbb{R}^2$

Weight $v \in \mathbb{R}^2$

Act. fun.: σ

$$\hat{y} = \sigma(v^T h)$$

Input: $x \in \mathbb{R}^d$

Weight: $w^{(1)} \in \mathbb{R}^d, w^{(2)} \in \mathbb{R}^d$

Act. func: σ

Output:

$$h_1 = \sigma\left(\sum w_i^{(1)} x_i\right) = \sigma(w^{(1)T} x)$$
$$h_2 = \sigma(w^{(2)T} x)$$

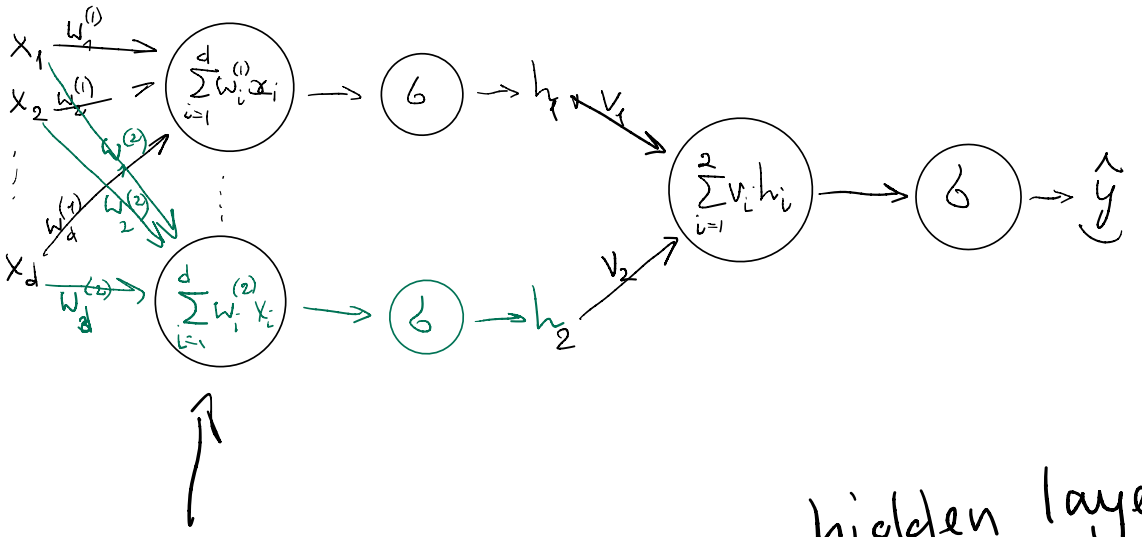
$$h = \begin{bmatrix} \sigma(w^{(1)T} x) \\ \sigma(w^{(2)T} x) \end{bmatrix} = \sigma\begin{pmatrix} w^{(1)T} x \\ w^{(2)T} x \end{pmatrix}$$

$W \in \mathbb{R}^{d \times 2}$

$W = (w^{(1)}, w^{(2)})$

$$\Rightarrow h = \sigma(W^T x)$$

Artificial Neural Network



Layer 1: $f_1(x) = \sigma(W^T x) \in \mathbb{R}^2$

Layer 2: $f_2(x) = \sigma(V^T x) \in \mathbb{R}^1$

$\hat{y} = f_2(f_1(x)) = f_2 \circ f_1(x)$

composition of functions

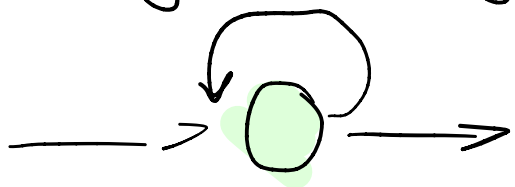
hidden layers

$f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1(x)$

last "layer"

Layers in Artificial Neural Network

- Dense $f(x) = W^T x$, $W \in \mathbb{R}^{d \times h}$ ← hidden dimention
- Non-linearty $f(x) = \begin{bmatrix} \sigma(x_1) \\ \vdots \\ \sigma(x_d) \end{bmatrix}$
- Convolutional $f(x) = W^T x$, but W should have special form
  ↖ Usually for images
- Recurrent



How to train?

$$p(y|x,\theta) = p(y|f_\theta(x))$$

model param of some distr

$$\theta^* = \arg\max_\theta \prod_{n=1}^{N} p(y_n|f_\theta(x_n))$$

or

$$\theta^* = \arg\min_\theta \mathcal{L}(y,x,\theta)$$

1.) $X \in \{x_1,\dots,x_N\}$, $Y = \{y_1,\dots,y_N\}$ are large $N \gg$ ← Stochastic g.d.

2) f_θ is a neural network ← Chain Rule Backprop

SGD recap

X, Y – dataset with N observations

$L(X, Y, \theta)$ ← loss

Initialize $\theta^{(0)}$

For i in max_iter:

- Sample mini-batch $\tilde{X}, \tilde{Y}$ (m points, $m \ll N$)
- Compute $\nabla_\theta L(\tilde{X}, \tilde{Y}, \theta)$
- Update $\theta^{(i)} = \theta^{(i-1)} - \alpha \cdot \nabla_\theta L(\tilde{X}, \tilde{Y}, \theta)$

$$\Uparrow$$

$$\left[\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}, \ldots, \frac{\partial L}{\partial \theta_{10000}}\right]$$

Chain Rule Recap

$$F(x) = f \circ g(x) = f(g(x)), \qquad ? \quad \frac{\partial F(x)}{\partial x}$$

$$\frac{\partial F(x)}{\partial x} = \frac{\partial f(g(x))}{\partial g(x)} \cdot \frac{\partial g(x)}{\partial x}$$

$$\left.\begin{array}{l} h = g(x) \\ F(x) = f(h) \end{array}\right\} \Rightarrow \frac{\partial F(x)}{\partial x} = \frac{\partial f}{\partial h} \cdot \frac{\partial g(x)}{\partial x}$$

Chain Rule in NN

$$y_L = f_L \circ f_{L-1} \circ \dots \circ f_1(x)$$

$\theta^L \quad \theta^{L-1} \quad \theta^1$

$$\nabla_\theta \mathcal{L}(y_L, x)$$

- θ^L: $\dfrac{\partial \mathcal{L}(y_L, x)}{\partial \theta^L} = \dfrac{\partial \mathcal{L}}{\partial y_L} \cdot \dfrac{\partial y_L}{\partial \theta_L}$

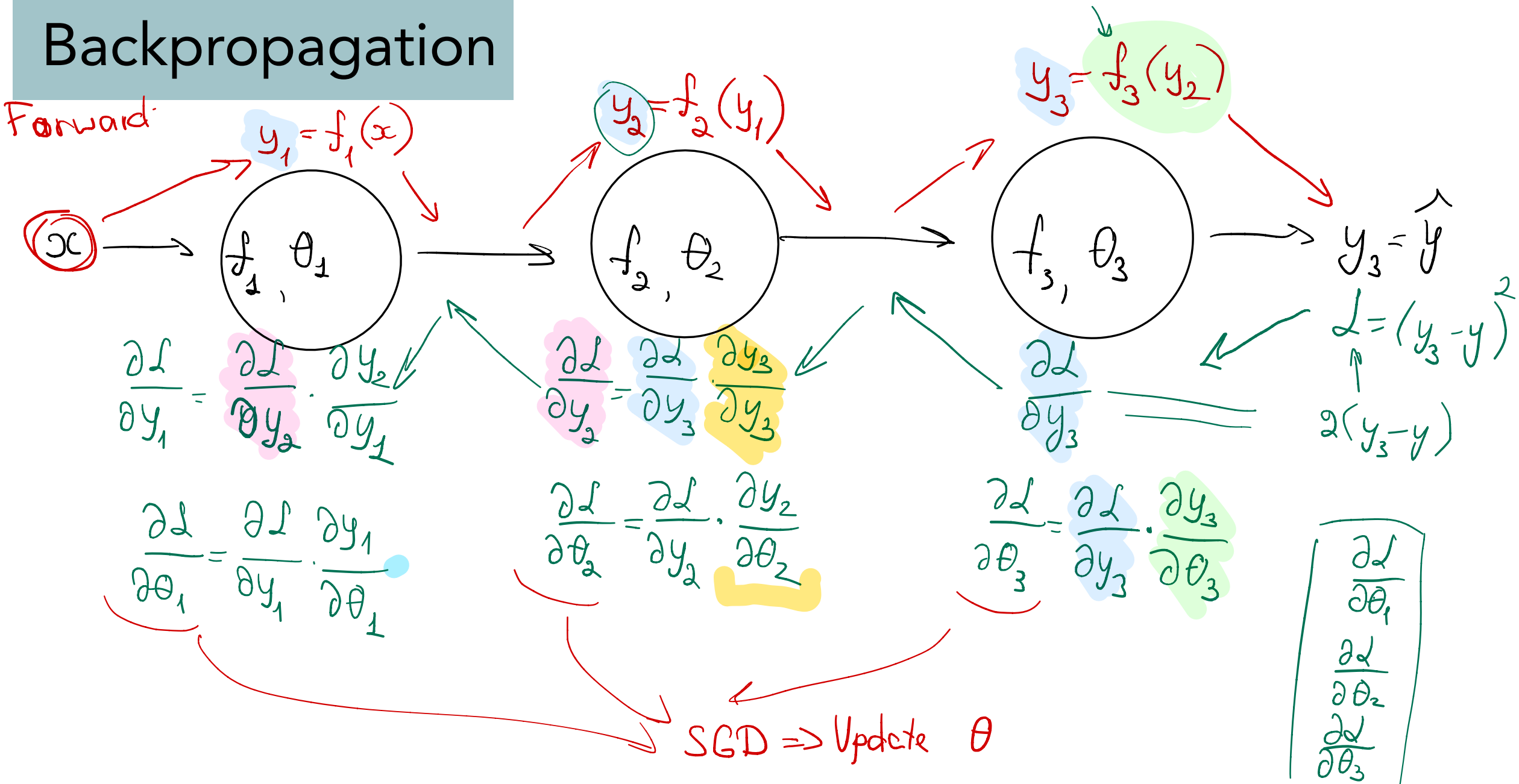
- θ^{L-1}: $\dfrac{\partial \mathcal{L}}{\partial \theta^{L-1}} = \dfrac{\partial \mathcal{L}}{\partial y_L} \cdot \dfrac{\partial y_L}{\partial \theta_{L-1}} = \dfrac{\partial \mathcal{L}}{\partial y_L} \cdot \dfrac{\partial y_L}{\partial y_{L-1}} \cdot \dfrac{\partial y_{L-1}}{\partial \theta_{L-1}}$

- θ^{L-2}: $\dfrac{\partial \mathcal{L}}{\partial \theta^{L-1}} = \dfrac{\partial \mathcal{L}}{\partial y_L} \cdot \dfrac{\partial y_L}{\partial y_{L-1}} \cdot \dfrac{\partial y_{L-1}}{\partial y_{L-2}} \cdot \dfrac{\partial y_{L-2}}{\partial \theta_{L-2}}$

① $y_{L-1} = f_{L-1}(\_, \theta_{L-1})$

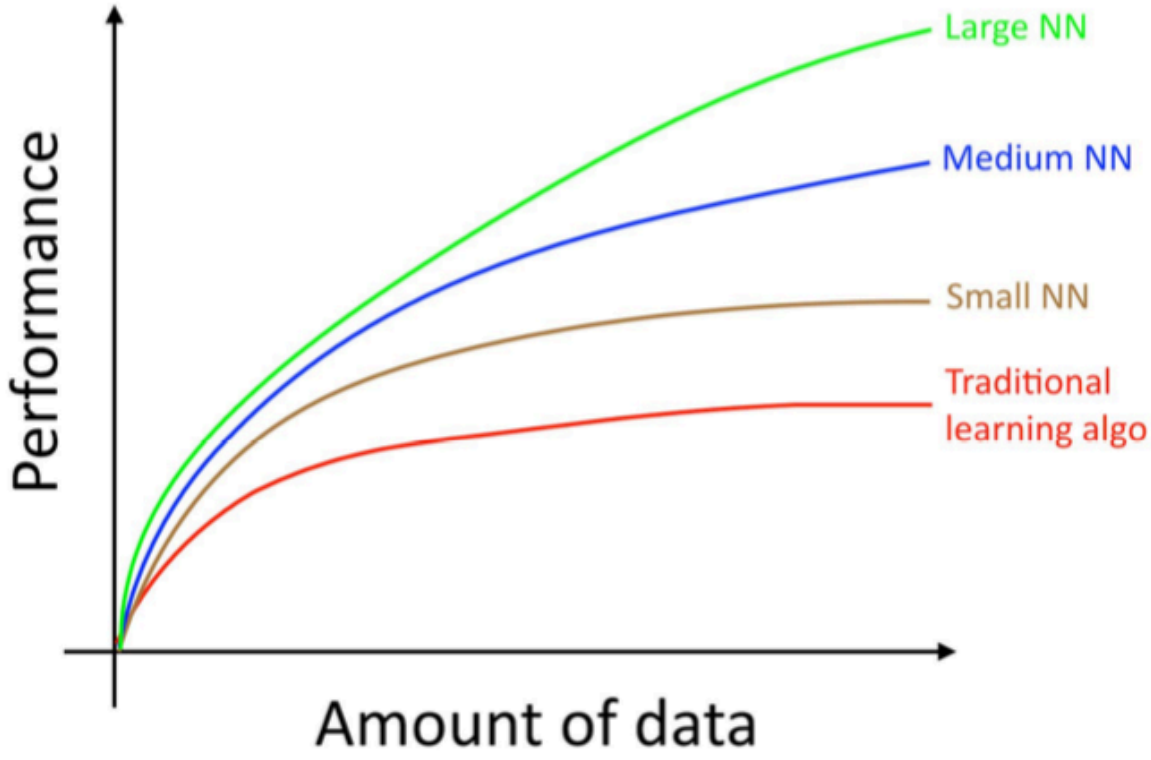
$y_L = f_L(y_{L-1})$

Backpropagation



Deep Neural Networks

- In theory can approximate any arbitrary function
- Benefit from large datasets
- Can be implemented fast on GPU



Tasks

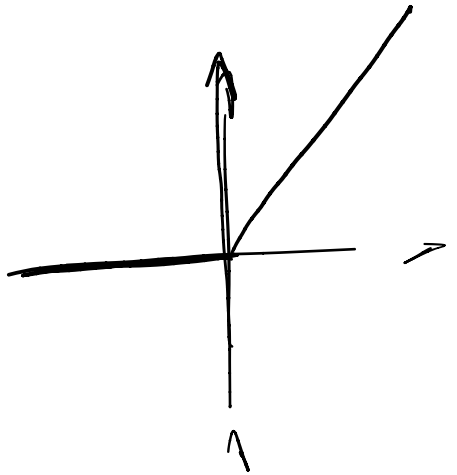
- Supervised: regression and classification
- Representation Learning
- Image / Text generation (next lecture)

$p(y|x)$

$p(x) \Rightarrow \tilde{x} \sim p(x)$

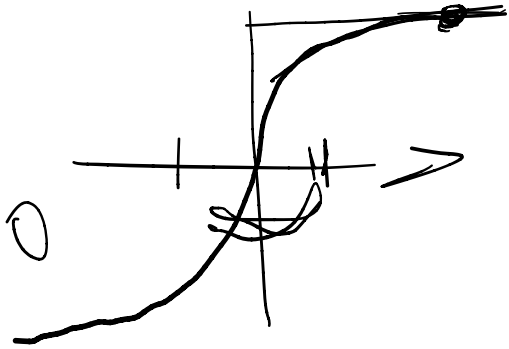
Limitations and Challenges

- Vanishing Gradient
  Other activation functions, BatchNorm, ResNet
- Overfitting
  Regularization, Data Augmentation, Dropout
- Requires Large Dataset
  Transfer Learning



$$\frac{\partial L}{\partial \theta_l} = \frac{\partial L}{\partial y_L} \cdot \frac{\partial y_L}{\partial y_{L-1}} \cdot \frac{\partial y_{L-1}}{\partial y_{L-2}} \cdots \frac{\partial y_l}{\partial \theta_l}$$

≈ 0



D - # of params

$D \gg N \gg 0$

$L + \|\theta\|^2$

↑ weight decay

Limitations and Challenges

- Vanishing Gradient
  Other activation functions, BatchNorm, ResNet

- Overfitting
  Regularization, Data Augmentation, Dropout

- Requires Large Dataset
  Transfer Learning