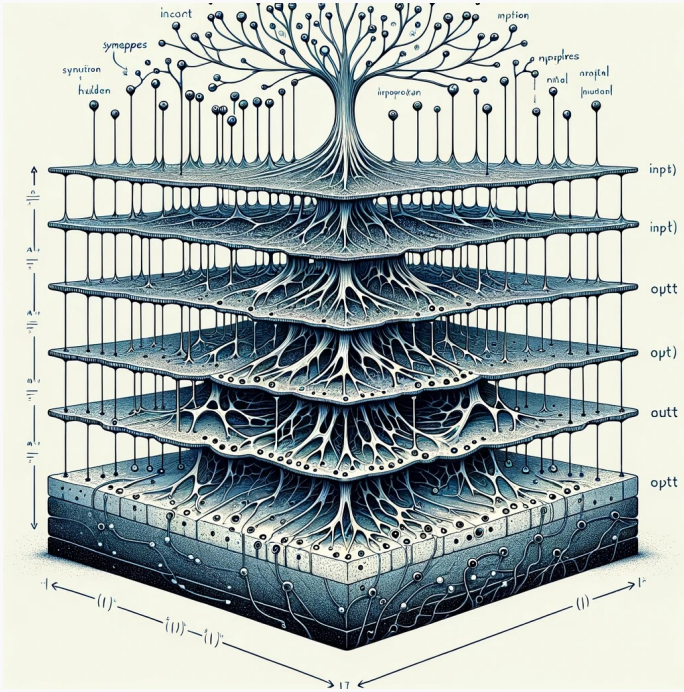
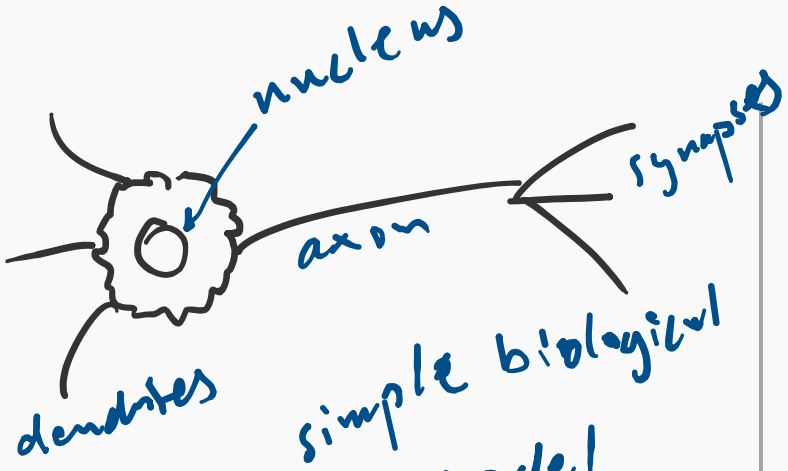
kokchun giang

artificial neural networks (ANN) mathematical models with inspiration from biological neurons



an inspiration from biological neuron

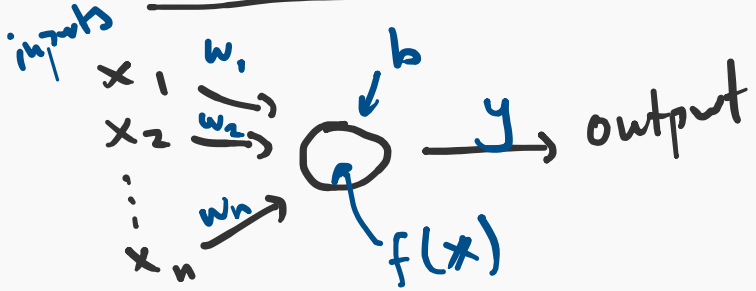


simple biological model

↓

mathematical model

Perceptron



→ weighting of evidence (inputs)

$$y = \begin{cases} 1 & w_1^T x + b > 0 \\ 0 & w_1^T x + b \le 0 \end{cases}$$

fire (1) · threshold · inhibited (0)

$$w_1^T x + b = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$$

Ex $y = \begin{cases} 1 & \text{go school} \\ 0 & \text{stay home} \end{cases}$

inputs

$$\begin{cases} x_1 \text{ - rain?} & \{0,1\} \\ x_2 \text{ - monday?} & \{0,1\} \\ x_3 \text{ - start late?} & \{0,1\} \end{cases}$$

$$w_1 = \begin{pmatrix} -2 \\ 1 \\ 4 \end{pmatrix}, \quad b = -1$$

$$x = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \text{ one sample}$$

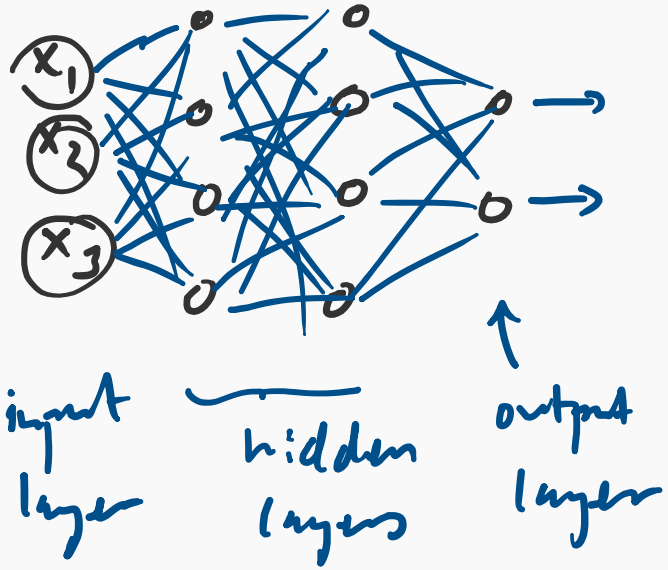
$$\hat{y} = w_1^T x + b = (-2 \mid 1 \mid 4)\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} - 1$$

$$= -2 + 0 + 4 - 1 = 1 > 0$$

$\Rightarrow y = 1 \rightarrow$ go to school

many neurons in a network of layers form a **neural network**

MLP - Multilayered perceptron



- feedforward network
- fully connected layers
- hidden layers ≥ 2
  - -> deep neural network

Activation functions

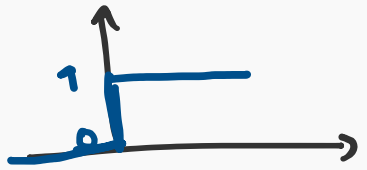
$$g(w^T x + b)$$

↑ activation fcn

the output is an activation

of a neuron.

In perceptron



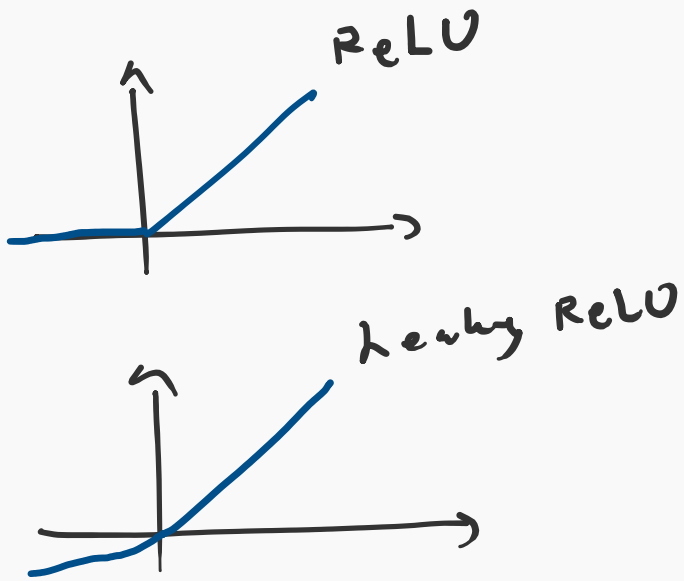
want to adjust weights & biases s.t. $\hat{y}$ gets closer to y

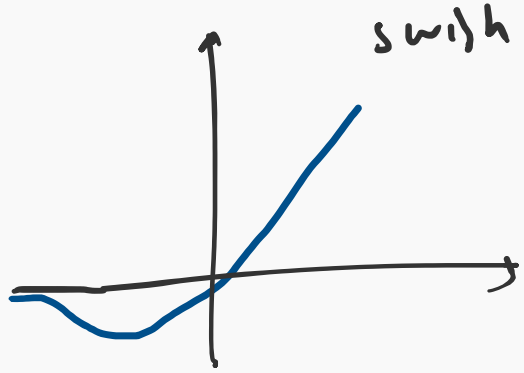
=> small change of Δw, Δb => small change in $\hat{y}$

but perceptron $0 \to 1$, $1 \to 0$

training the weights and biases

popular activation fcns that work well





Cost fcn & gradient descent

$C(w, b)$ cost fcn that we want to minimize w.r.t w, b

→ want to find weights and biases to get loss as small as possible

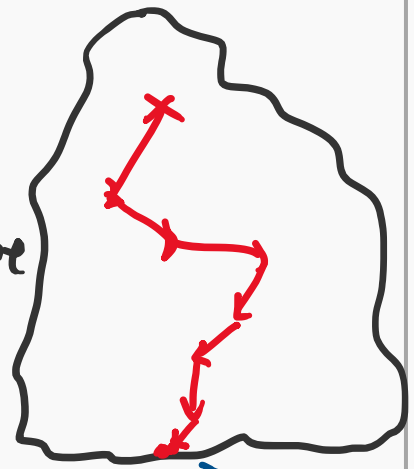
need to approximate the solution by taking small steps in direction where loss decreases

→ gradient descent

gradient descent and backpropagation

Analogy

Choose steepest slope & take a short step in that direction repeat this until reaching bottom



2 dimensional case

learning rate determines how long the step is

stochastic gradient descent (SGD) & minibatch GD are popular variants.

---

for neural networks we have many layers of neurons → propagate backwards layer by layer & compute GD to update weights & biases.

=> backpropagation