A decorative network graph pattern in the top-left corner, featuring a complex web of interconnected nodes and edges. Some nodes are highlighted with blue circles, and others with blue dots. The pattern is composed of light gray lines and dots.

Neural Networks

Jorge Guerra, PhD

A decorative network graph pattern in the bottom-right corner, similar to the one in the top-left, featuring a complex web of interconnected nodes and edges. Some nodes are highlighted with blue circles, and others with blue dots. The pattern is composed of light gray lines and dots.

Hello!

I am Jorge Guerra, PhD

You can find me at

jorge.guerra881215@gmail.com





Summary

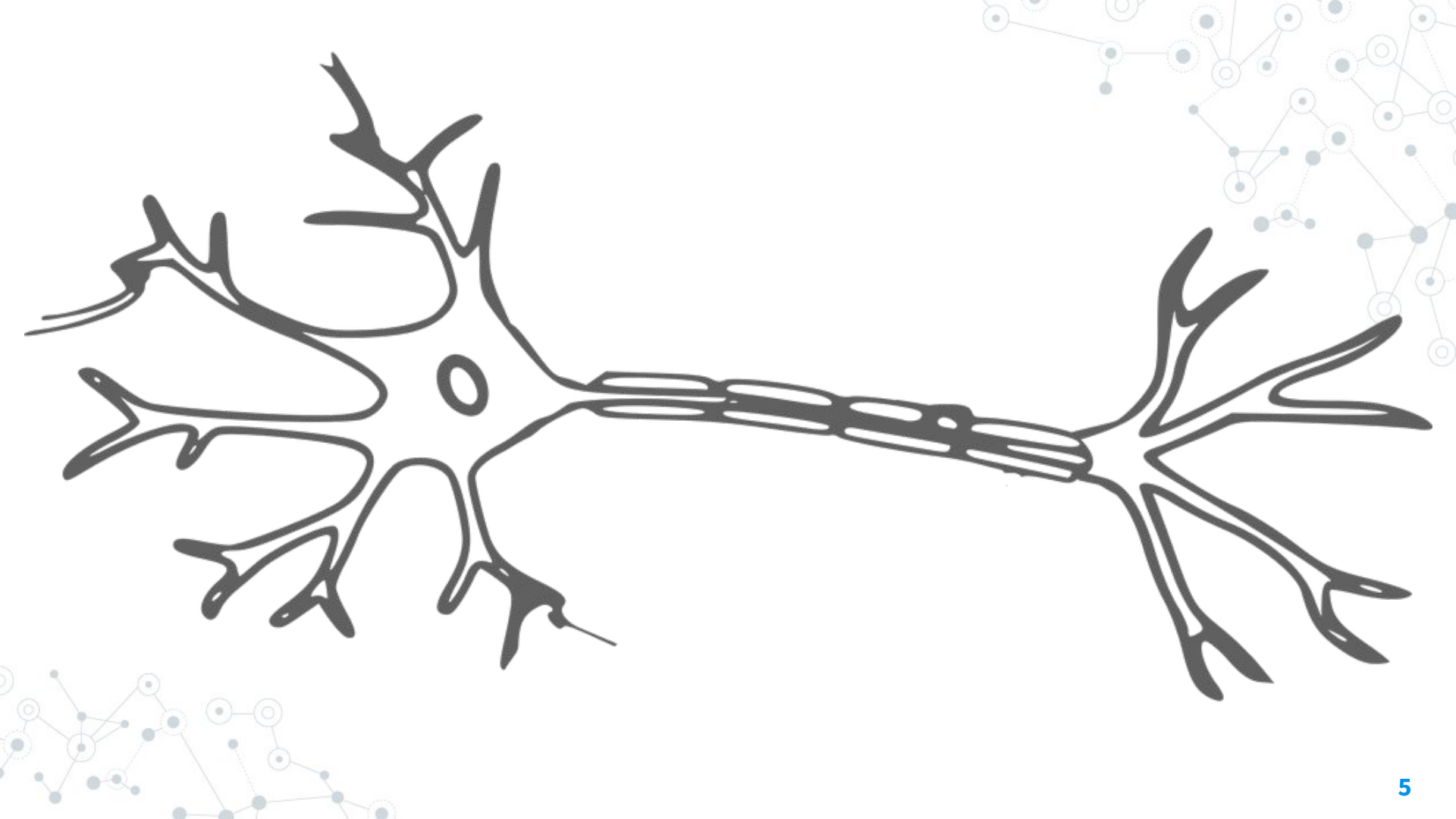
1. A brief history of neural networks
2. Neural networks architecture: Perceptron
3. Activation functions
4. Universal Approximation Theorem
5. Gradient Descent
6. Forward and backward propagation
7. Tensors
8. Cross Entropy

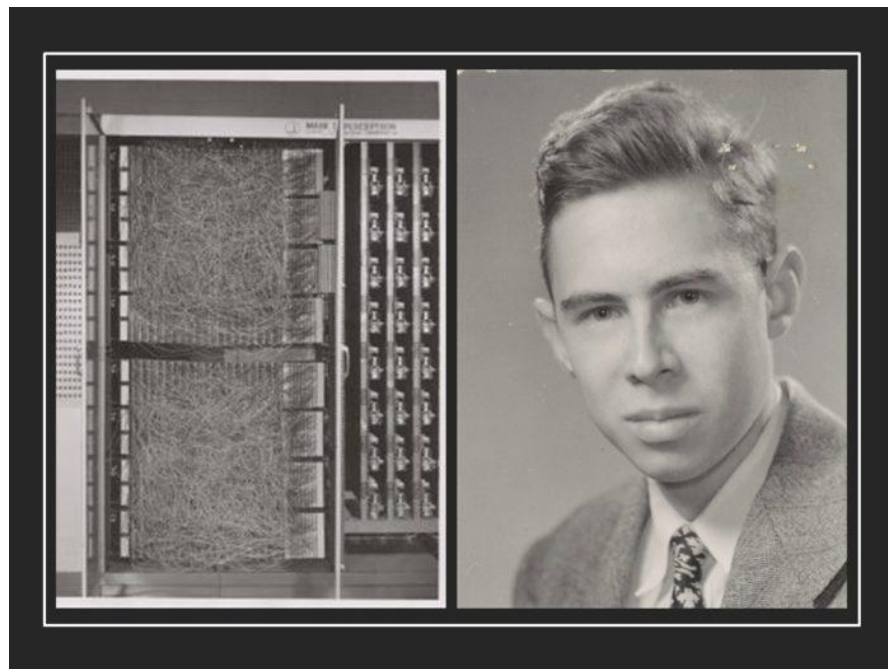
A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, with some nodes highlighted in blue.

1.

A brief history of neural networks

A brief history of misconceptions
and preconceptions





Perceptron

Frank Rosenblatt

1958

BRACE YOURSELF

**El invierno de la
Inteligencia
Artificial**

WINTER IS COMING



A brief history of neural network



Warren McCulloch & Walter Pitts, wrote a paper on how neurons might work; they modeled a simple neural network with electrical circuits.

Nathaniel Rochester from the IBM research laboratories led the first effort to simulate a neural network.

John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes.

STORY BY DATA

1943

1949

1950s

1956

1957

HISTORY OF NEURAL NETWORKS

1943-2019

1958

Donald Hebb reinforced the concept of neurons in his book, *The Organization of Behavior*. It pointed out that neural pathways are strengthened each time they are used.

The **Dartmouth Summer Research Project** on Artificial Intelligence provided a boost to both artificial intelligence and neural networks.

Frank Rosenblatt began work on the Perceptron; the oldest neural network still in use today.

1982

1981

1969

1959

1982

John Hopfield presented a paper to the national Academy of Sciences. His approach to create useful devices; he was likeable, articulate, and charismatic.

Progress on neural network research halted due fear, unfulfilled claims, etc.

Marvin Minsky & Seymour Papert proved the Perceptron to be limited in their book, *Perceptrons*.

Bernard Widrow & Marcian Hoff of Stanford developed models they called ADALINE and MADALINE; the first neural network to be applied to a real world problem.

1982

1985

1997

1998

NOW

US-Japan Joint Conference on Cooperative/Competitive Neural Networks; Japan announced their Fifth-Generation effort resulted in US worrying about being left behind and restarted the funding in US.

American Institute of Physics began what has become an annual meeting - **Neural Networks for Computing**.

A recurrent neural network framework, LSTM was proposed by **Schmidhuber & Hochreiter**.

Yann LeCun published *Gradient-Based Learning Applied to Document Recognition*.

Neural networks discussions are prevalent; the future is here!

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, rendered in a light gray color.

2.

Neural Networks

Architecture

Neural Network in the context of AI

Artificial Intelligence



Machine Learning



Neural Networks



Deep Learning

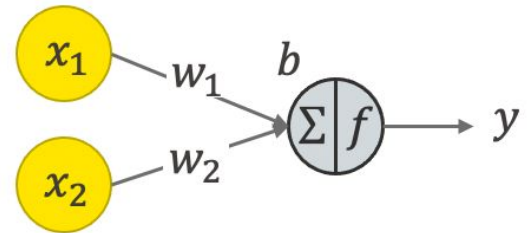


Neural Networks main characteristics

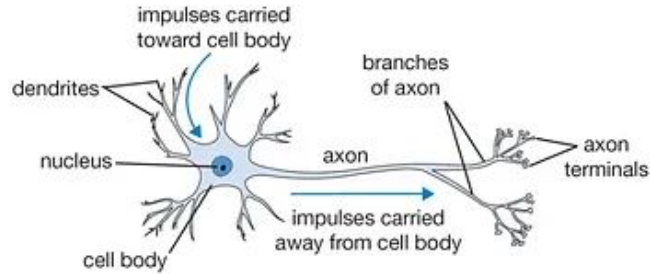
1. (Artificial) Neural networks are set of algorithms inspired by the functioning of human brain.
2. Neural networks (NN) are **universal function approximators** so that means neural networks can learn an approximation of any function $f()$ such that,

$$y = f(x)$$

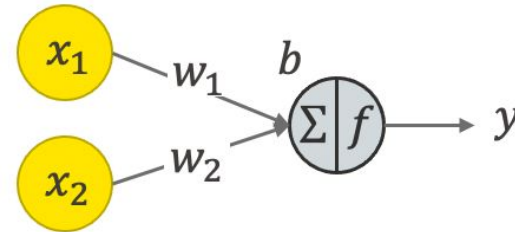
3. NN learn by example (supervised learning).
4. Used for regression and classification problems.



A single neuron



- Input nodes: x_1, x_2
- Weights: w_1, w_2 .
- Bias: b
- Sum: Σ
- Activation function: $f()$
- Output: y
- Loss function
- Optimizer



Step 1: Each input is multiplied by the associated weight.

$$a = x_1 * w_1 + x_2 * w_2 + b$$

Step 2: An activation function $f()$ converts the result into the neuron output.

$$y = f(a)$$



3.

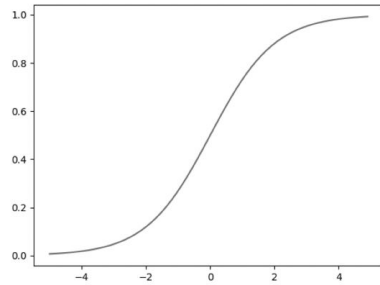
Activation functions

Activation functions

Activation function provides the possibility to learn non-linear functions

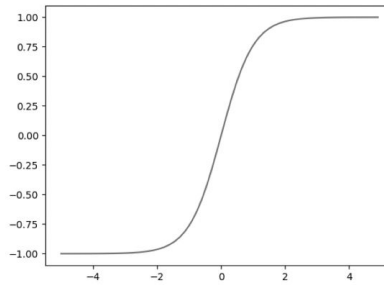
<https://www.desmos.com/calculator/plevozbz1o>

Sigmoid



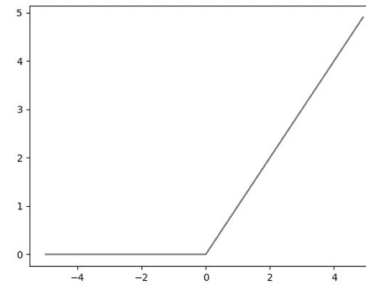
$$f(a) = \frac{1}{1 + e^{-ha}}$$

Tanh



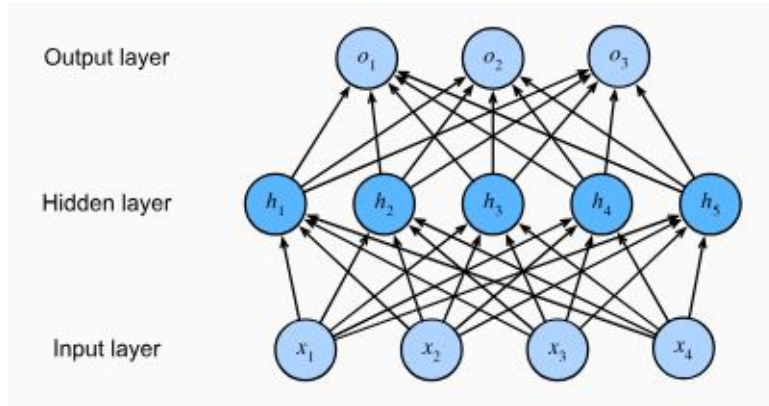
$$f(a) = \frac{e^{2ha} - 1}{e^{2ha} + 1}$$

Rectified Linear Unit (ReLU)



$$f(a) = \max\{0, ha\}$$

Network layers



$$\mathbf{H} = \sigma(\mathbf{X} \mathbf{W}^{(1)} + \mathbf{b}^{(1)})$$

$$\mathbf{O} = \mathbf{H} \mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

Input units: The activity of the input units represents the raw information that is fed into the network.

Hidden units: The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

Output units: The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.



4. **Loss functions**

Loss function names

- ◎ **Objective function:** In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights).
- ◎ **Cost function.**
- ◎ **Loss function...** or just **loss**.

$$\mathbf{w}^*, b^* = \operatorname{argmin}_{\mathbf{w}, b} L(\mathbf{w}, b).$$

Types of Loss Functions

Regression problems: given an input value, the model predicts a corresponding output value .

MSE (Mean Squared Error):

$$L(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n l^{(i)}(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)} \right)^2.$$

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}} L(\mathbf{w}, b).$$

Binary classification problems: given an input, the neural network produces a vector of probabilities of the input belonging to two pre-set categories

Logarithmic loss or Cross-Entropy:

$$CE\ Loss = \frac{1}{n} \sum_{i=1}^N - (y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i))$$

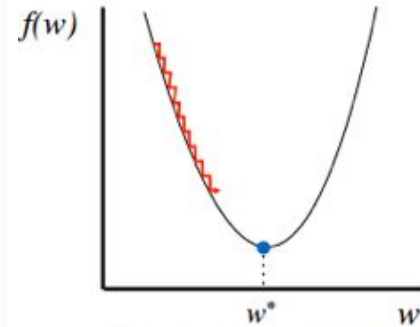
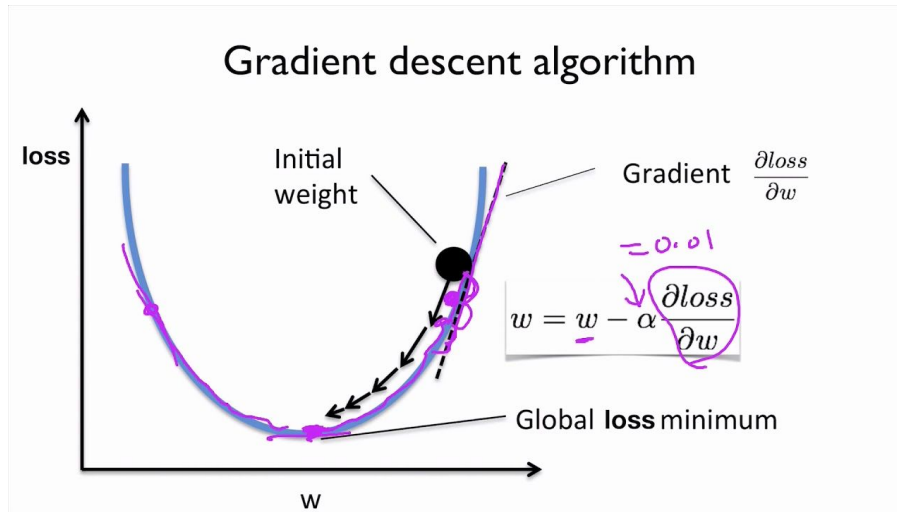
Multi-class classification problems: given an input, the neural network produces a vector of probabilities of the input belonging to various pre-set categories

Softmax

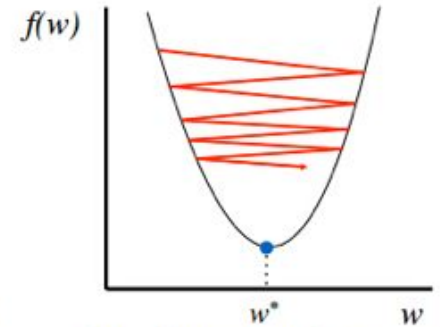


5. Gradient Descent

Gradient descent



Too small: converge very slowly



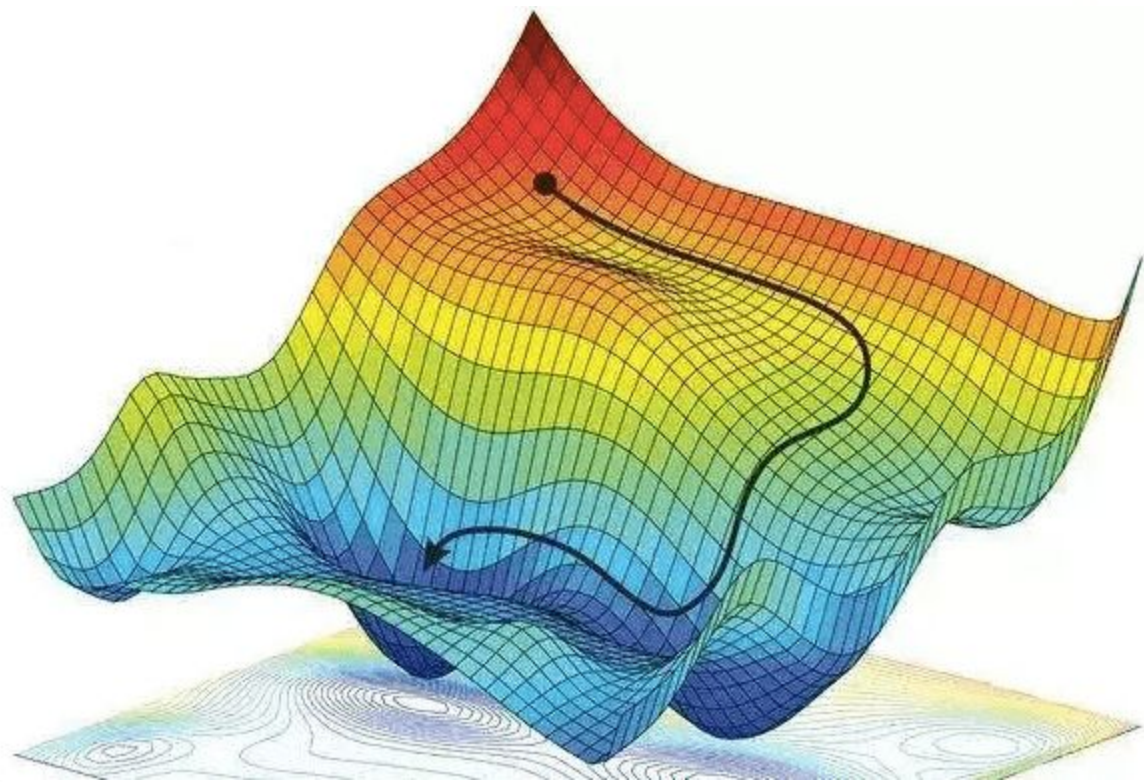
Too big: overshoot and even diverge

Most common optimizers:

Stochastic gradient descent

Adam

RMSProp

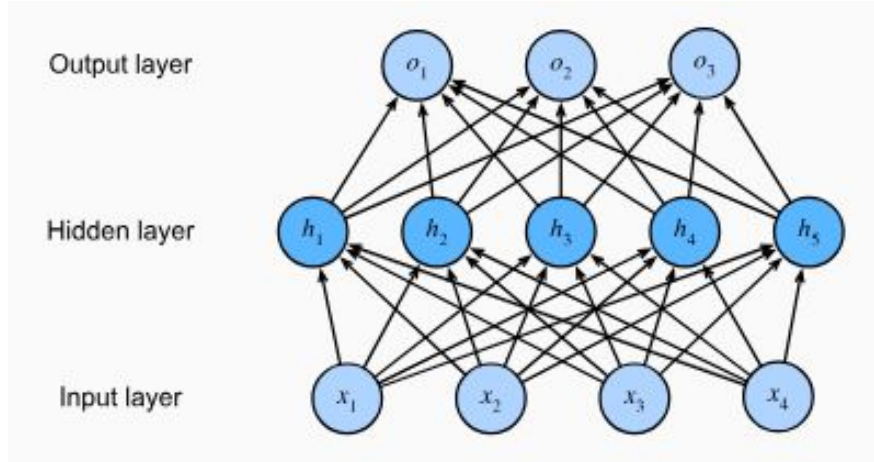




6.

Forward and backward propagation

Forward propagation



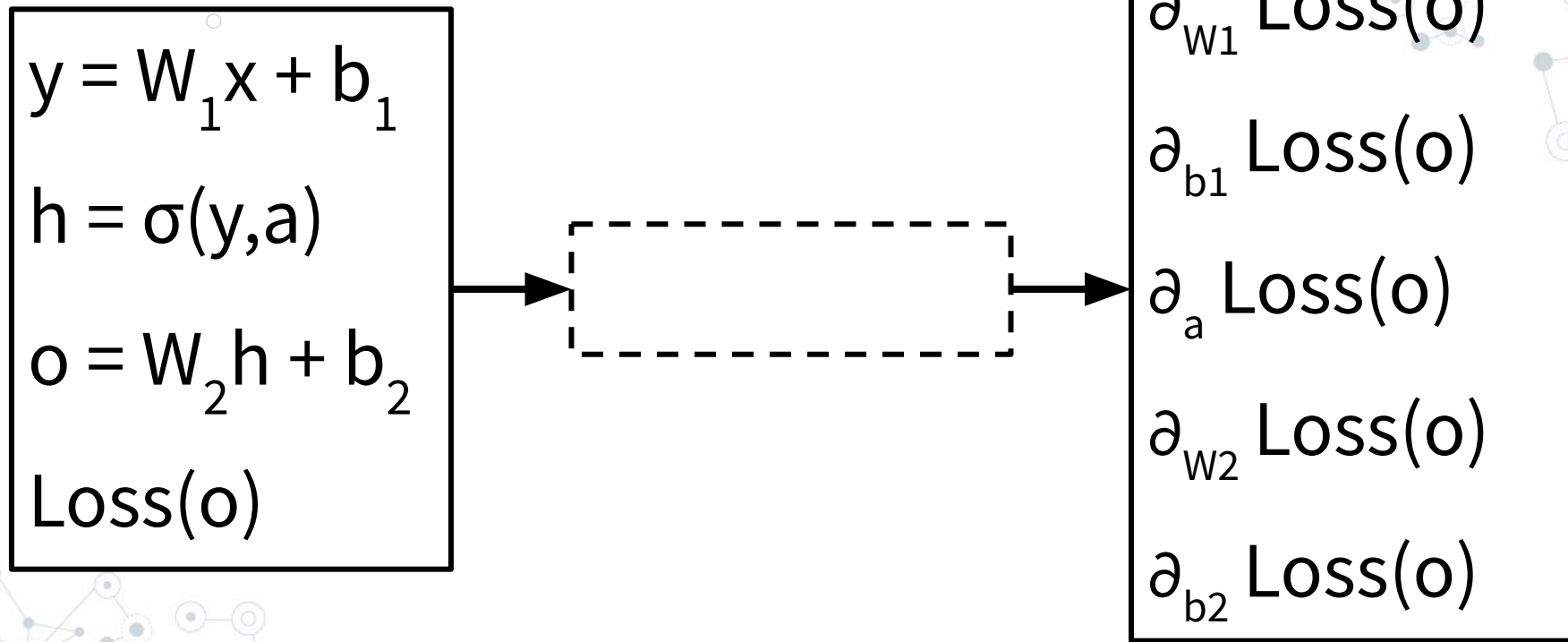
$$y = W_1 X + b_1$$


$$h = \sigma(y, a)$$

$$o = W_2 h + b_2$$

$$\text{Loss}(o)$$

Backward propagation




$$y = W_1 x + b_1$$

$$h = \sigma(y, a)$$

$$o = W_2 h + b_2$$

$$\text{Loss}(o)$$

$$\partial_{W_1} y$$

$$\partial_{b_1} y$$

$$\partial_y h$$

$$\partial_a h$$

$$\partial_h o$$

$$\partial_{W_2} o$$

$$\partial_{b_2} o$$

$$\partial_o \text{Loss}$$


$$\partial_{W_1} \text{Loss}$$

$$\partial_{b_1} \text{Loss}$$

$$\partial_a \text{Loss}$$

$$\partial_{W_2} \text{Loss}$$

$$\partial_{b_2} \text{Loss}$$

$$y = W_1 x + b_1$$

$$h = \sigma(y, a)$$

$$o = W_2 h + b_2$$

$$\text{Loss}(o)$$

$$\partial_{W_1} y$$

$$\partial_{b_1} y$$

$$\partial_y h$$

$$\partial_a h$$

$$\partial_h o$$

$$\partial_{W_2} o$$

$$\partial_{b_2} o$$

$$\partial_o \text{Loss}$$

$$\partial_{W_1} \text{Loss}$$

$$\partial_{b_1} \text{Loss}$$

$$\partial_a \text{Loss}$$

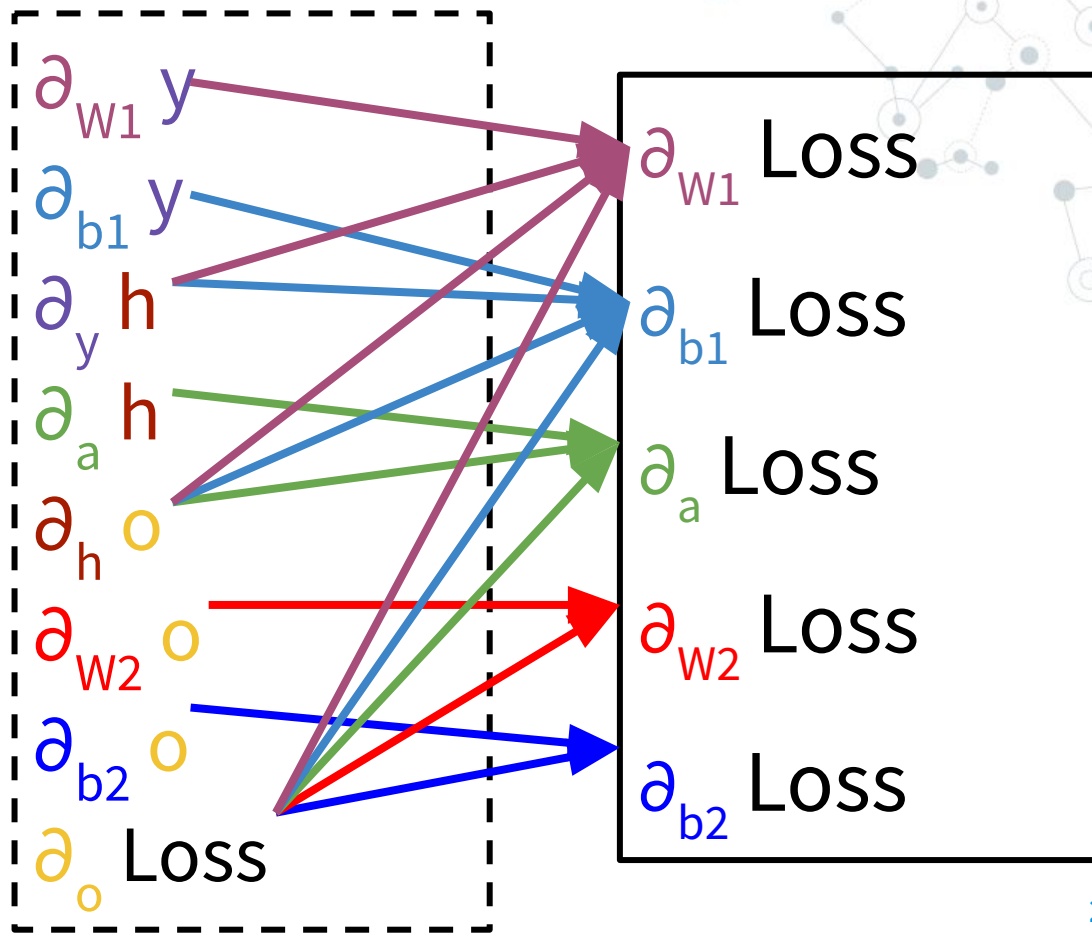
$$\partial_{W_2} \text{Loss}$$

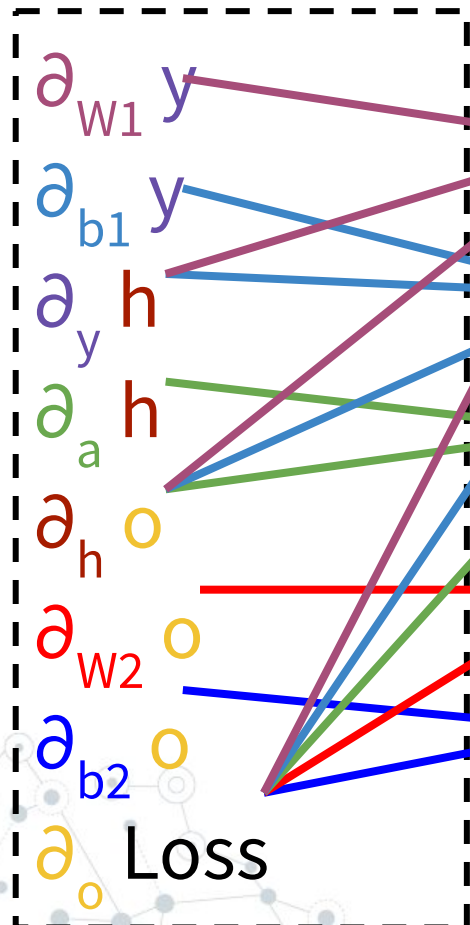
$$\partial_{b_2} \text{Loss}$$

Chain rule

$$g[f(\mathbf{x})]$$
$$\partial_{\mathbf{x}} g = \partial_f g \cdot \partial_{\mathbf{x}} f$$

$$\begin{aligned}
 y &= W_1 x + b_1 \\
 h &= \sigma(y, a) \\
 o &= W_2 h + b_2 \\
 \text{Loss}(o)
 \end{aligned}$$





$$\partial_{w1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{w1} y$$

$$\partial_{b1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{b1} y$$

$$\partial_a \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_a h$$

$$\partial_{w2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{w2} o$$

$$\partial_{b2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{b2} o$$

Chain rule

$$y = W_1 x + b_1$$

$$h = \sigma(y, a)$$

$$o = W_2 h + b_2$$

$$\text{Loss}(o)$$

$$\partial_{W_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{W_1} y$$

$$\partial_{b_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{b_1} y$$

$$\partial_a \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_a h$$

$$\partial_{W_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{W_2} o$$

$$\partial_{b_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{b_2} o$$

Chain rule

$$y = W_1 x + b_1$$

$$h = \sigma(y, a)$$

$$o = W_2 h + b_2$$

$$\text{Loss}(o)$$

$$\partial_{W_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{W_1} y$$

$$\partial_{b_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{b_1} y$$

$$\partial_a \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_a h$$

$$\partial_{W_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{W_2} o$$

$$\partial_{b_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{b_2} o$$

Chain rule

$$y = W_1 x + b_1$$

$$h = \sigma(y, a)$$

$$o = W_2 h + b_2$$

$$\text{Loss}(o)$$

$$\partial_{W_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{W_1} y$$

$$\partial_{b_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{b_1} y$$

$$\partial_a \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_a h$$

$$\partial_{W_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{W_2} o$$

$$\partial_{b_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{b_2} o$$

Chain-rule Backpropagation

$$y = W_1 x + b_1$$

$$h = \sigma(y, a)$$

$$o = W_2 h + b_2$$

$$\text{Loss}(o)$$

$$\partial_{W_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{W_1} y$$

$$\partial_{b_1} \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_y h \cdot \partial_{b_1} y$$

$$\partial_a \text{Loss} = \partial_o \text{Loss} \cdot \partial_h o \cdot \partial_a h$$

$$\partial_{W_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{W_2} o$$

$$\partial_{b_2} \text{Loss} = \partial_o \text{Loss} \cdot \partial_{b_2} o$$

		Scalar	Vector	Matrix
		x (1,)	\mathbf{x} (n,1)	\mathbf{X} (n, k)
Scalar	y (1,)	$\frac{\partial y}{\partial x}$ (1,)	$\frac{\partial y}{\partial \mathbf{x}}$ (1,n)	$\frac{\partial y}{\partial \mathbf{X}}$ (k, n)
Vector	\mathbf{y} (m,1)	$\frac{\partial \mathbf{y}}{\partial x}$ (m,1)	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ (m, n)	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}$ (m, k, n)
Matrix	\mathbf{Y} (m, l)	$\frac{\partial \mathbf{Y}}{\partial x}$ (m, l)	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}$ (m, l, n)	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$ (m, l, k, n)



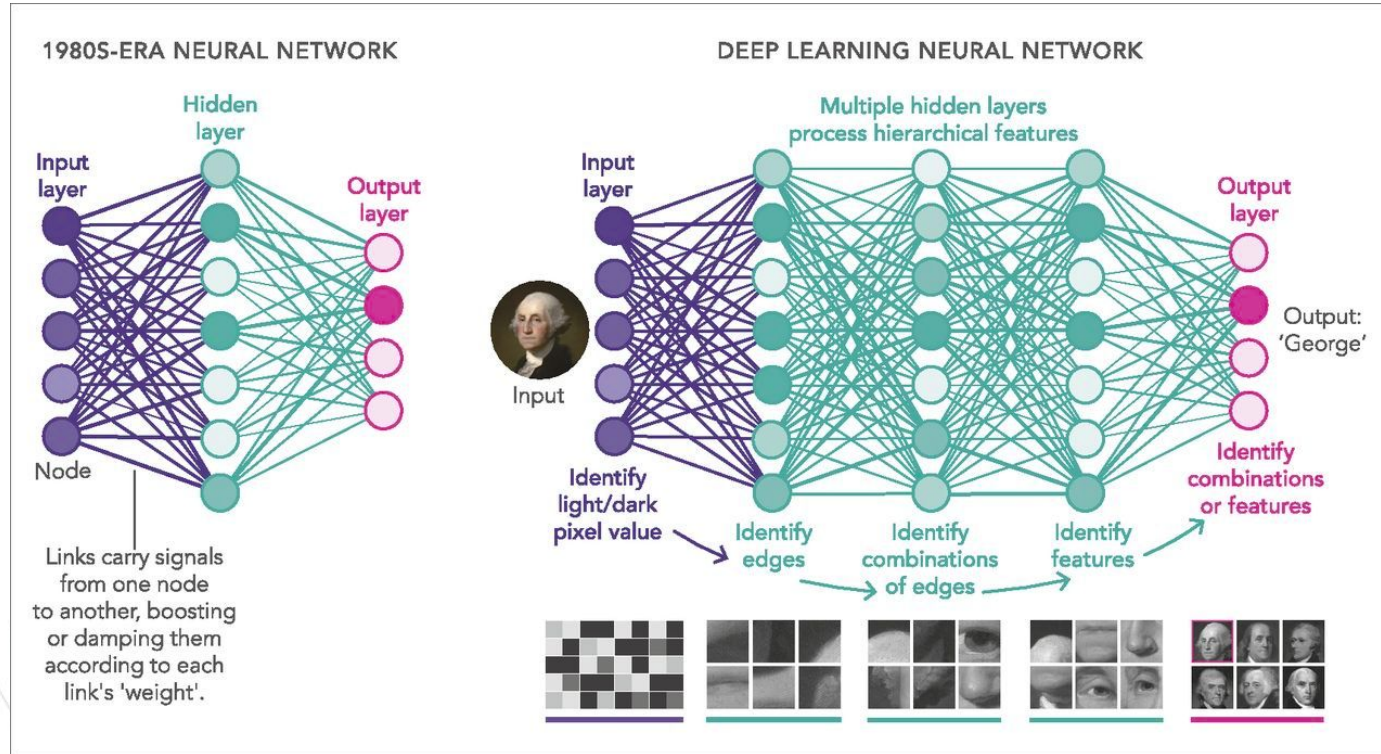
7.

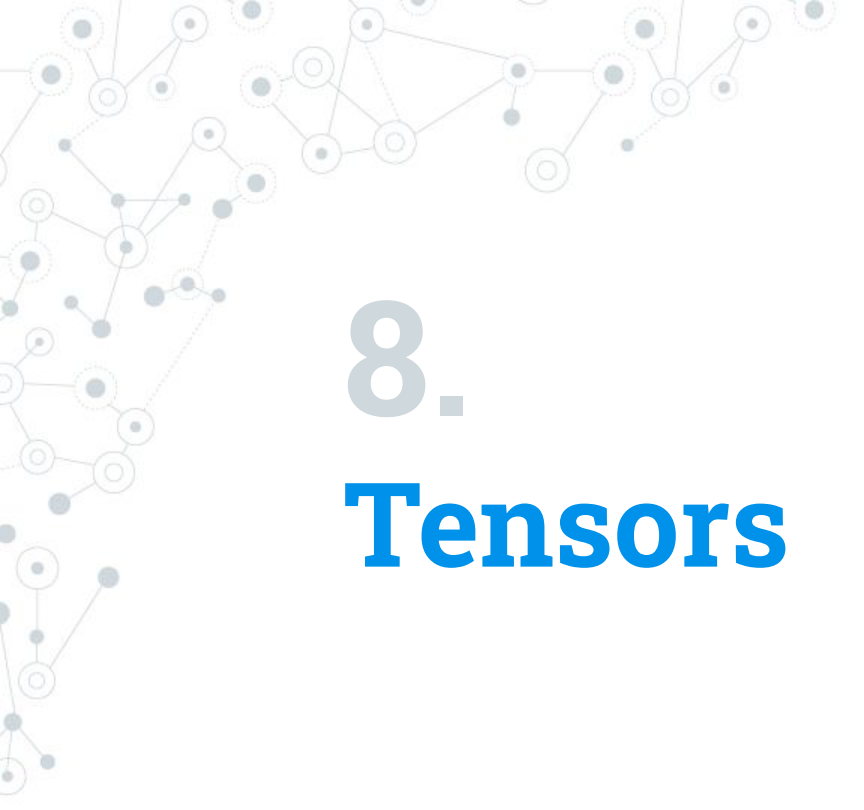
Deep learning

Not enough layers!

Deep learning is just a good name

Deep learning has become a powerful tool in various domains, including computer vision, natural language processing, and speech recognition.



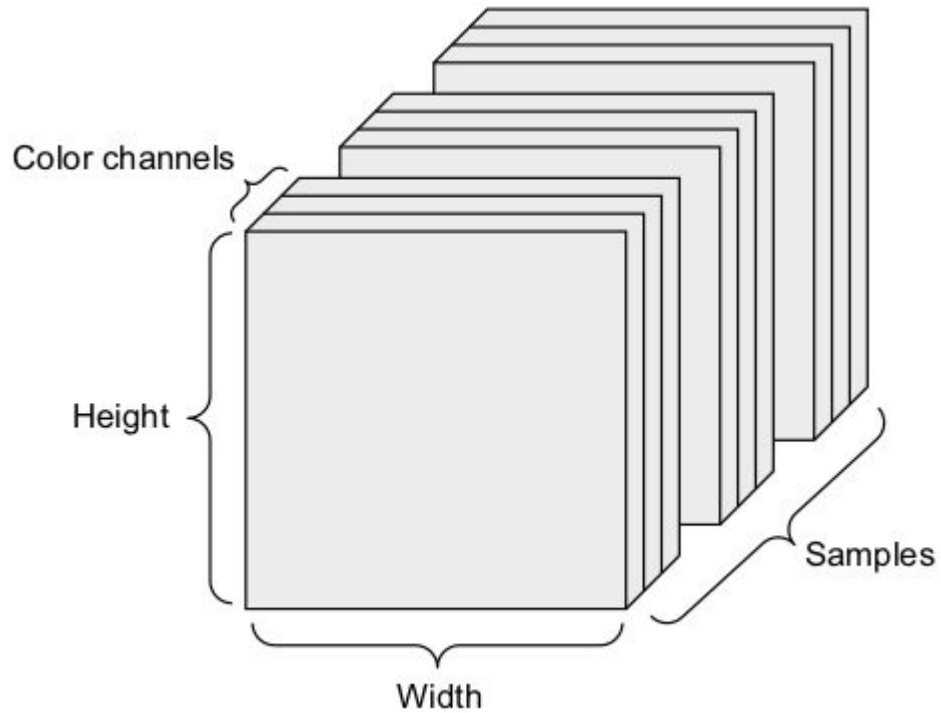


8. Tensors

Tensor: Data representations for neural networks

- ⊙ Scalars (rank-0 tensors)
- ⊙ Vectors (rank-1 tensors)
- ⊙ Matrices (rank-2 tensors)
- ⊙ Rank-3 and higher-rank tensors

Image data



A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting a hierarchical or central structure. The lines are thin and gray, connecting the nodes in a non-linear fashion.

8. Books

Book

DEEP LEARNING with Python

SECOND EDITION

François Chollet

 MANNING





Thanks!

Any questions?

You can find me at:

jorge.guerra881215@gmail.com