Neural Networks

Jorge Guerra, PhD

Hello!

I am Jorge Guerra, PhD

You can find me at

jorge.guerra881215@gmail.com

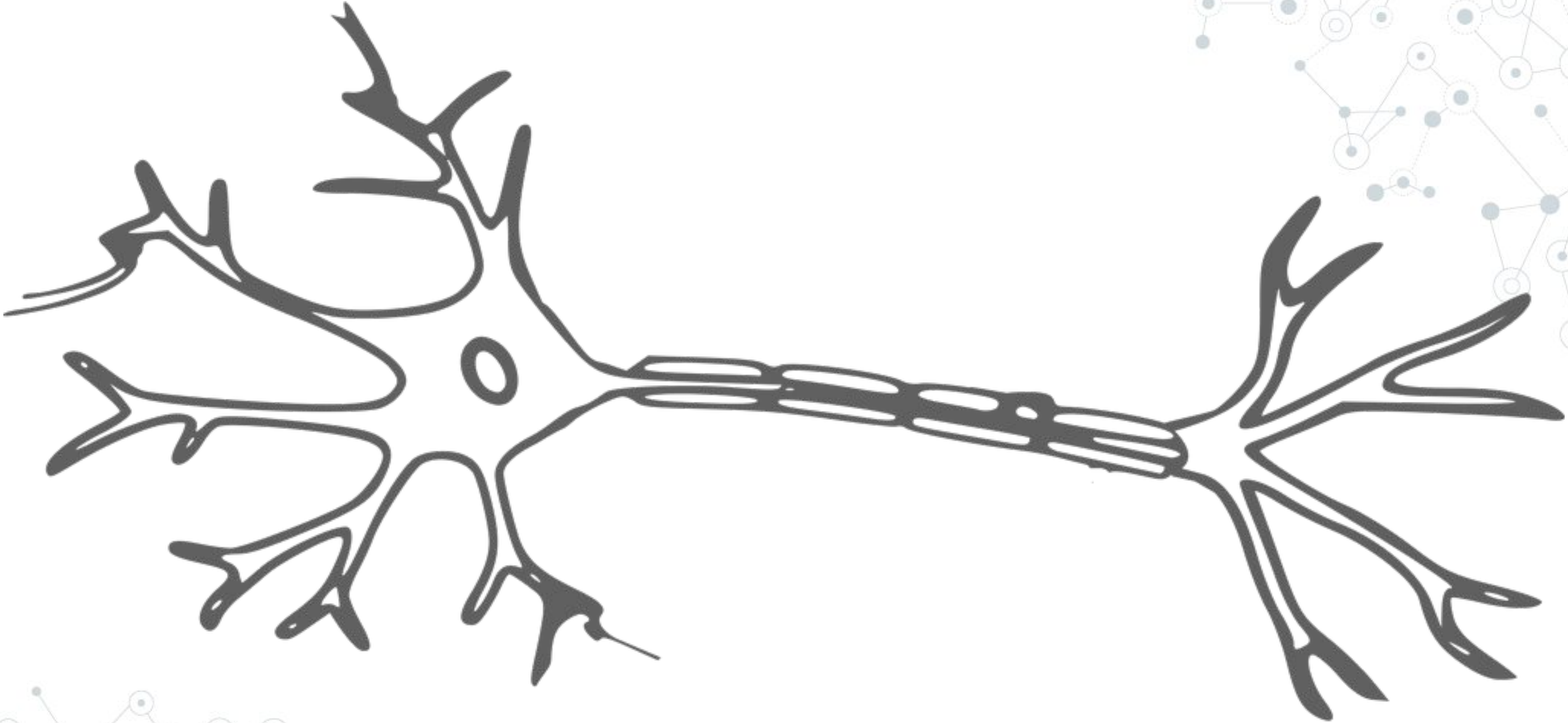
Summary

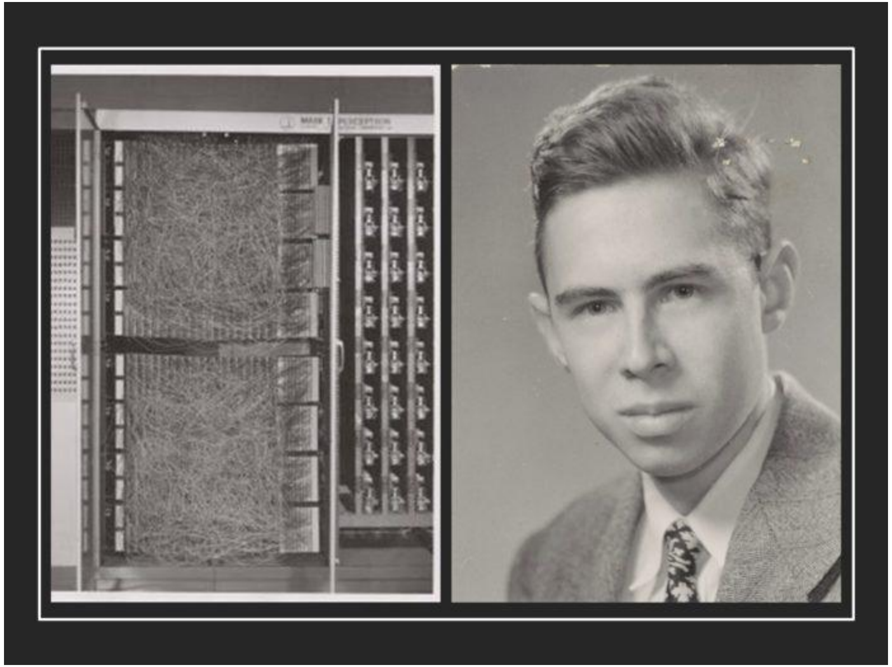
1. A brief history of neural networks
2. Neural networks architecture: Perceptron
3. Activation functions
4. Universal Approximation Theorem
5. Gradient Descent
6. Forward and backward propagation
7. Tensors
8. Cross Entropy

1.

A brief history of neural networks

A brief history of misconceptions and preconceptions

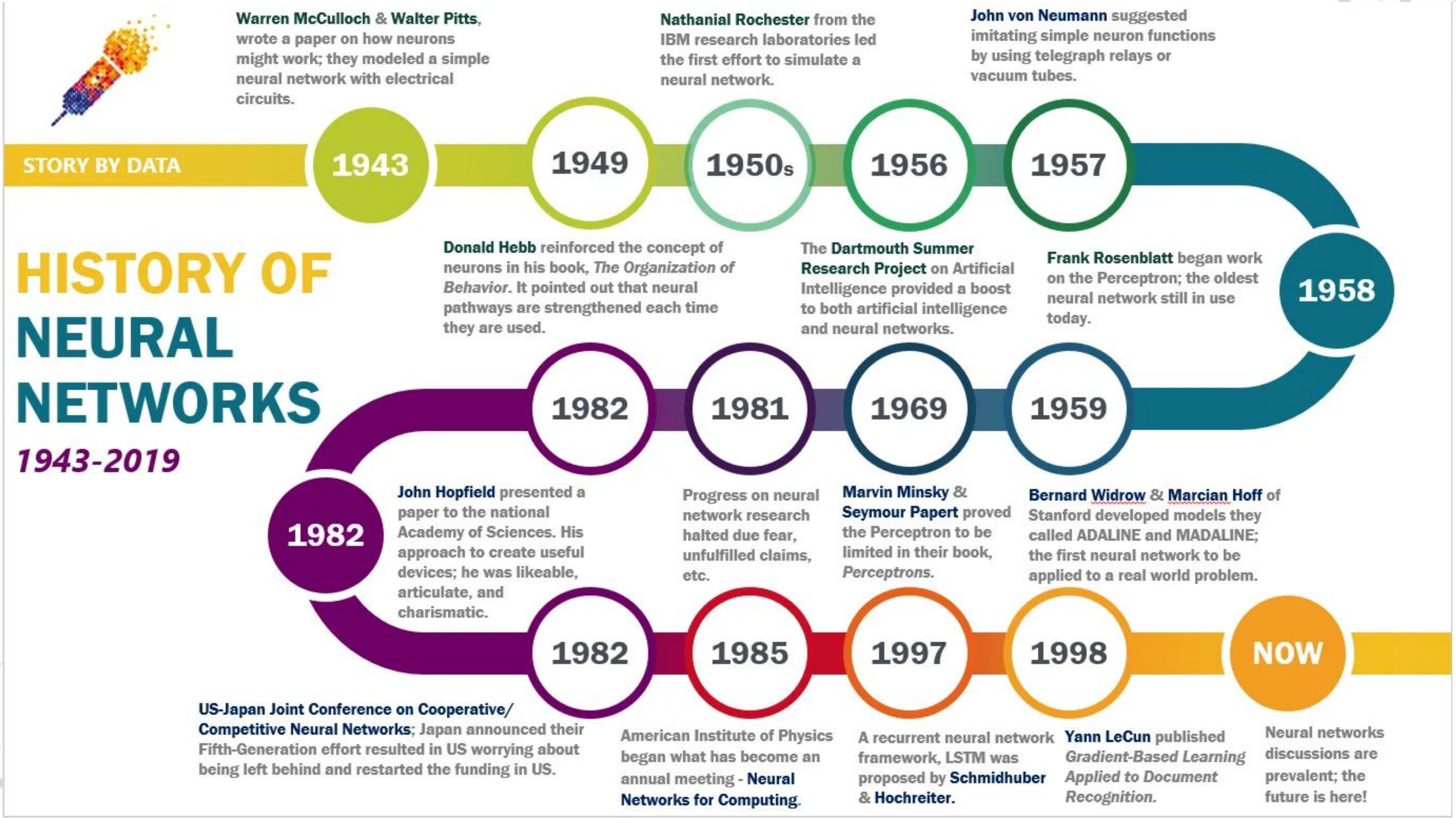




Perceptron
Frank Rosenblatt
1958

El invierno de la Inteligencia Artificial

A brief history of neural network

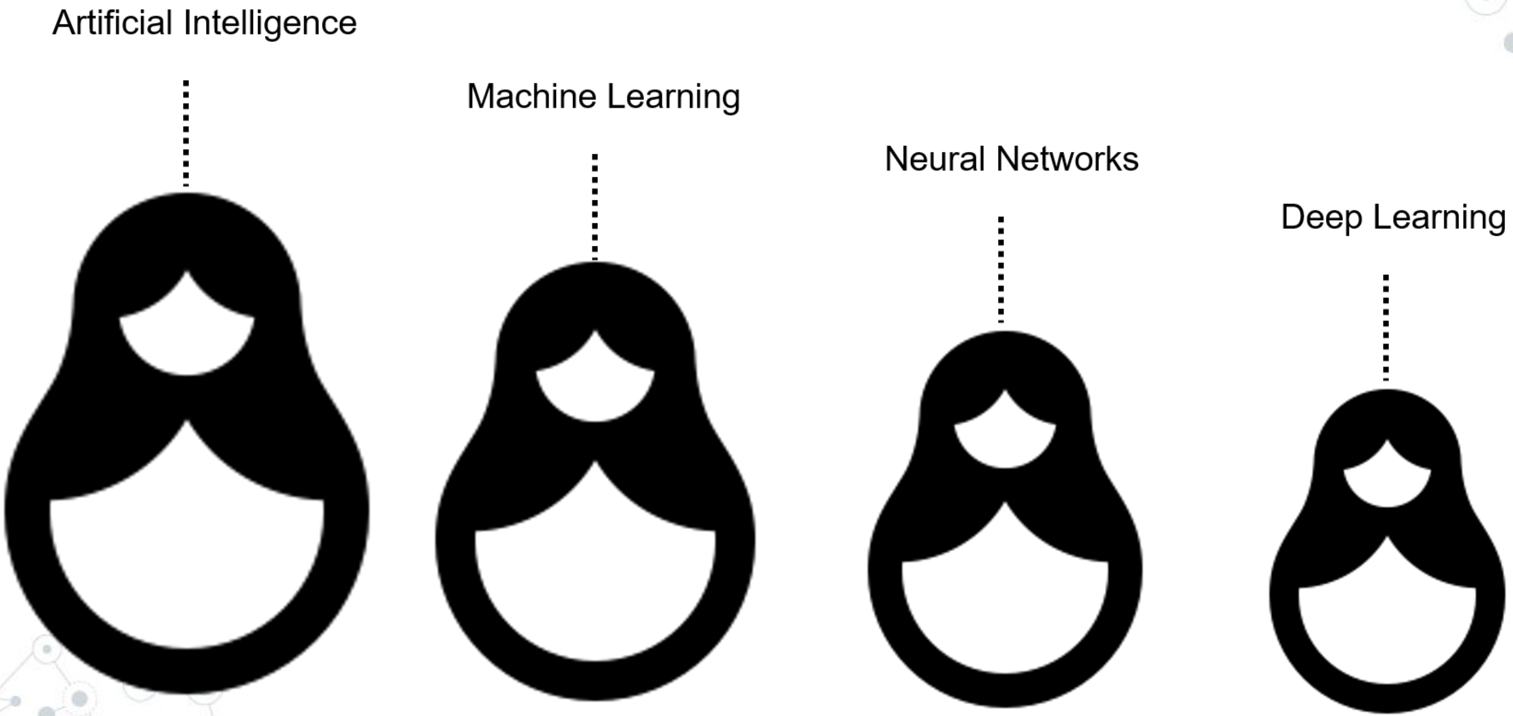


2.

Neural Networks

Architecture

Neural Network in the context of AI

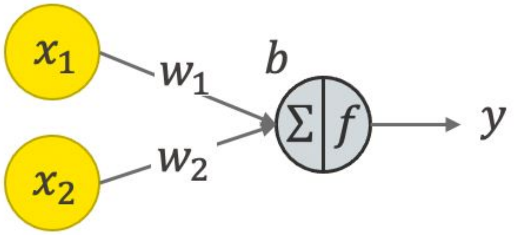


Neural Networks main characteristics

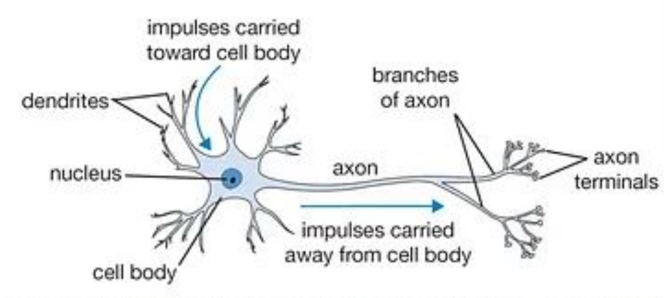
1. (Artificial) Neural networks are set of algorithms inspired by the functioning of human brian.
2. Neural networks (NN) are **universal function approximators** so that means neural networks can learn an approximation of any function f() such that,

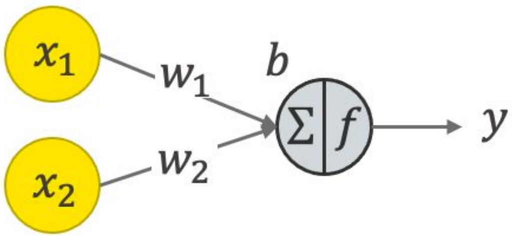
$$y = f(x)$$

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



A single neuron





- Input nodes: x1, x2
- Weights: w1, w2.
- Bias: b
- Sum: ∑
- Activation function: f()
- Output: y
- Loss function
- Optimizer

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

$$a = x1^* \, w1 + x2^* w2 + b$$

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

$$y = f(a)$$

3.

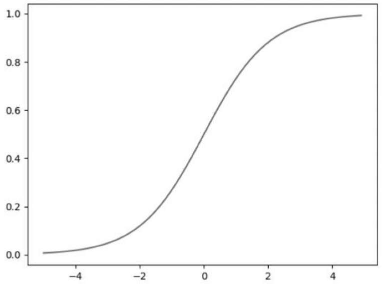
Activation functions

Activation functions

Activation function provides the possibility to learn non-linear functions

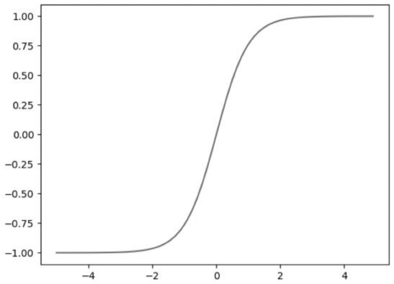
https://www.desmos.com/calculator/plevozbz1o

Sigmoid



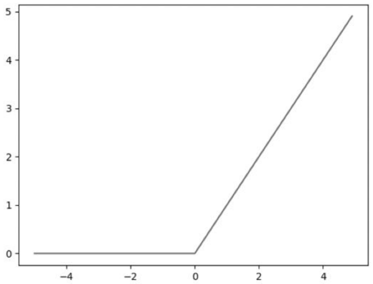
$$f(a) = \frac{1}{1 + e^{-ha}}$$

Tanh



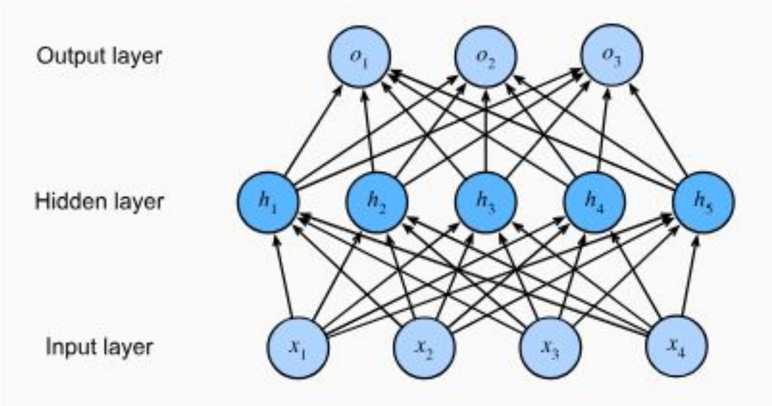
$$f(a) = \frac{e^{2ha} - 1}{e^{2ha} + 1}$$

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**.

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

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:

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

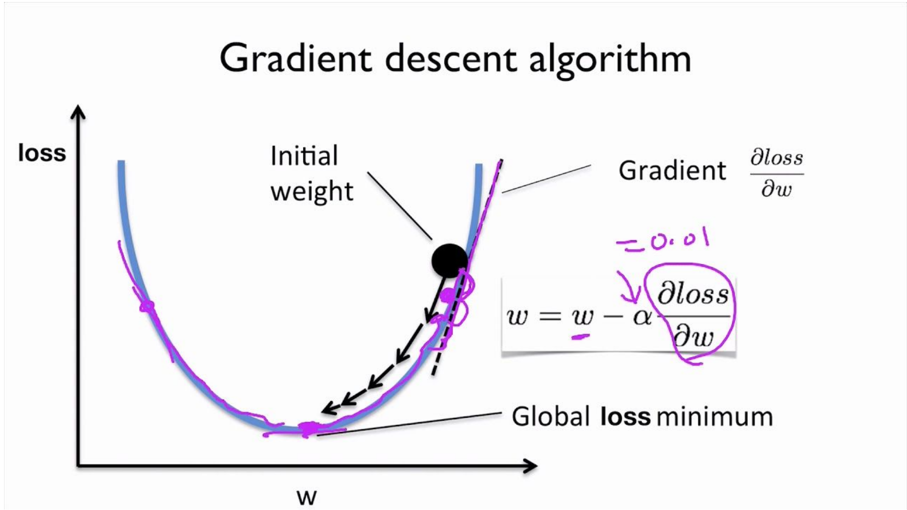
Multi-class classification problems: given an input, the neural network produces a vector of probabilities of the input belonging to various pre-set categories

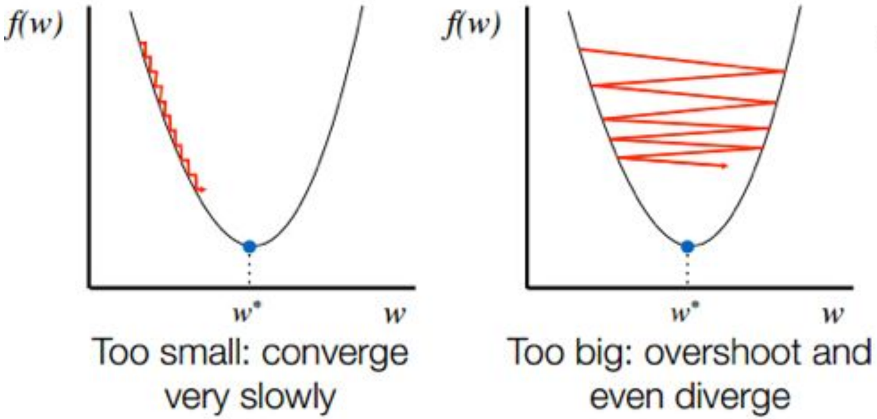
Softmax

5.

Gradient Descent

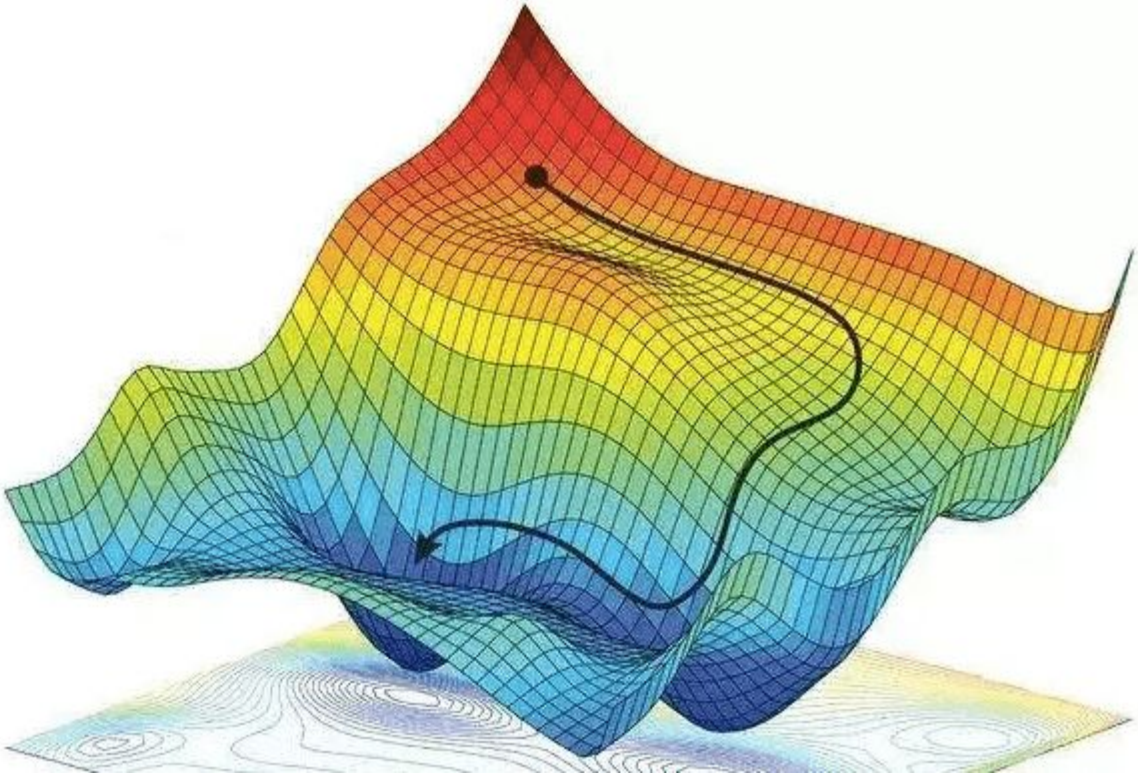
Gradient descent





Most common optimizers:

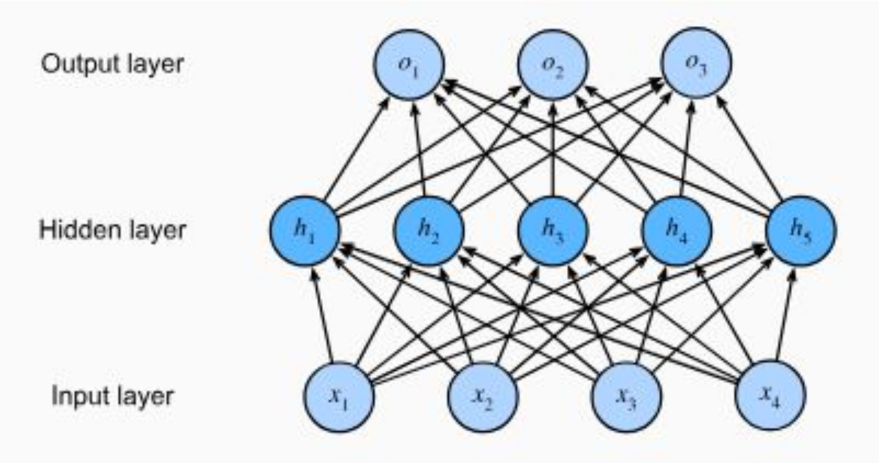
- Stochastic gradient descent
- Adam
- RMSProp



6.

Forward and backward propagation

Forward propagation



$$y = W_1 X + b_1$$

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

$$o = W_2 h + b_2$$

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

Backward propagation

$$y = W_1 x + b_1$$

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

$$o = W_2 h + b_2$$

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

$$\partial_{W1}\, \mathrm{Loss}(o)$$

$$\partial_{b1}\, \mathrm{Loss}(o)$$

$$\partial_{a}\, \mathrm{Loss}(o)$$

$$\partial_{W2}\, \mathrm{Loss}(o)$$

$$\partial_{b2}\, \mathrm{Loss}(o)$$

$$y = W_1 x + b_1$$

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

$$o = W_2 h + b_2$$

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

$\partial_{W1}\, y$

$\partial_{b1}\, y$

$\partial_{y}\, h$

$\partial_{a}\, h$

$\partial_{h}\, o$

$\partial_{W2}\, o$

$\partial_{b2}\, o$

$\partial_{o}\, \mathrm{Loss}$

$\partial_{W1}\, \mathrm{Loss}$

$\partial_{b1}\, \mathrm{Loss}$

$\partial_{a}\, \mathrm{Loss}$

$\partial_{W2}\, \mathrm{Loss}$

$\partial_{b2}\, \mathrm{Loss}$

$$y = W_1 x + b_1$$

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

$$o = W_2 h + b_2$$

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

$\partial_{W1}\, y$

$\partial_{b1}\, y$

$\partial_{y}\, h$

$\partial_{a}\, h$

$\partial_{h}\, o$

$\partial_{W2}\, o$

$\partial_{b2}\, o$

$\partial_{o}\, \text{Loss}$

$\partial_{W1}\, \text{Loss}$

$\partial_{b1}\, \text{Loss}$

$\partial_{a}\, \text{Loss}$

$\partial_{W2}\, \text{Loss}$

$\partial_{b2}\, \text{Loss}$

Chain rule

$$g[f(x)]$$

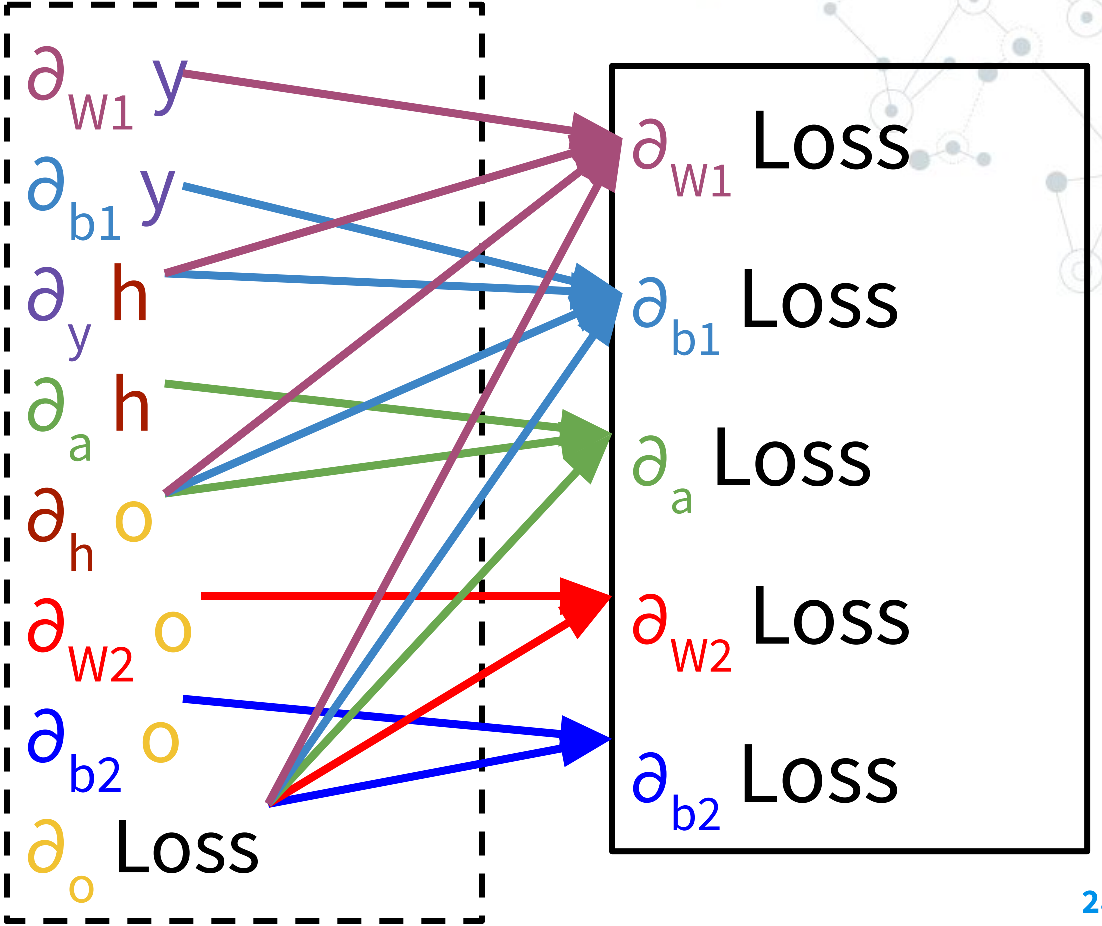
$$\partial_x g = \partial_f g \cdot \partial_x f$$

$$y = W_1 x + b_1$$

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

$$o = W_2 h + b_2$$

$$Loss(o)$$



$\partial_{W1}\, y$

$\partial_{b1}\, y$

$\partial_{y}\, h$

$\partial_{a}\, h$

$\partial_{h}\, o$

$\partial_{W2}\, o$

$\partial_{b2}\, o$

$\partial_{o}\, \text{Loss}$

$$\partial_{W1}\, \text{Loss} = \partial_{o}\, \text{Loss} \cdot \partial_{h}\, o \cdot \partial_{y}\, h \cdot \partial_{W1}\, y$$

$$\partial_{b1}\, \text{Loss} = \partial_{o}\, \text{Loss} \cdot \partial_{h}\, o \cdot \partial_{y}\, h \cdot \partial_{b1}\, y$$

$$\partial_{a}\, \text{Loss} = \partial_{o}\, \text{Loss} \cdot \partial_{h}\, o \cdot \partial_{a}\, h$$

$$\partial_{W2}\, \text{Loss} = \partial_{o}\, \text{Loss} \cdot \partial_{W2}\, o$$

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

Chain rule

$$y = W_1 x + b_1$$

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

$$o = W_2 h + b_2$$

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

$$\partial_{W1}\,\mathrm{Loss} = \partial_o\,\mathrm{Loss} \cdot \partial_h\,o \cdot \partial_y\,h \cdot \partial_{W1}\,y$$

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

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

$$\partial_{W2}\,\mathrm{Loss} = \partial_o\,\mathrm{Loss} \cdot \partial_{W2}\,o$$

$$\partial_{b2}\,\mathrm{Loss} = \partial_o\,\mathrm{Loss} \cdot \partial_{b2}\,o$$

Chain rule

$$y = W_1 x + b_1$$
$$h = \sigma(y, a)$$
$$o = W_2 h + b_2$$
$$Loss(o)$$

$$\partial_{W1} Loss = \partial_o Loss \cdot \partial_h o \cdot \partial_y h \cdot \partial_{W1} y$$
$$\partial_{b1} Loss = \partial_o Loss \cdot \partial_h o \cdot \partial_y h \cdot \partial_{b1} y$$
$$\partial_a Loss = \partial_o Loss \cdot \partial_h o \cdot \partial_a h$$
$$\partial_{W2} Loss = \partial_o Loss \cdot \partial_{W2} o$$
$$\partial_{b2} Loss = \partial_o Loss \cdot \partial_{b2} o$$

Chain rule

$$y = W_1 x + b_1$$

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

$$o = W_2 h + b_2$$

$$Loss(o)$$

$$\partial_{W1} Loss = \partial_o Loss \cdot \partial_h o \cdot \partial_y h \cdot \partial_{W1} y$$

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

$$\partial_a Loss = \partial_o Loss \cdot \partial_h o \cdot \partial_a h$$

$$\partial_{W2} Loss = \partial_o Loss \cdot \partial_{W2} o$$

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

~~Chain rule~~ Backpropagation

$$y = W_1 x + b_1$$

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

$$o = W_2 h + b_2$$

$$Loss(o)$$

$$\partial_{W1} Loss = \partial_o Loss \cdot \partial_h o \cdot \partial_y h \cdot \partial_{W1} y$$

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

$$\partial_a Loss = \partial_o Loss \cdot \partial_h o \cdot \partial_a h$$

$$\partial_{W2} Loss = \partial_o Loss \cdot \partial_{W2} o$$

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

		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)

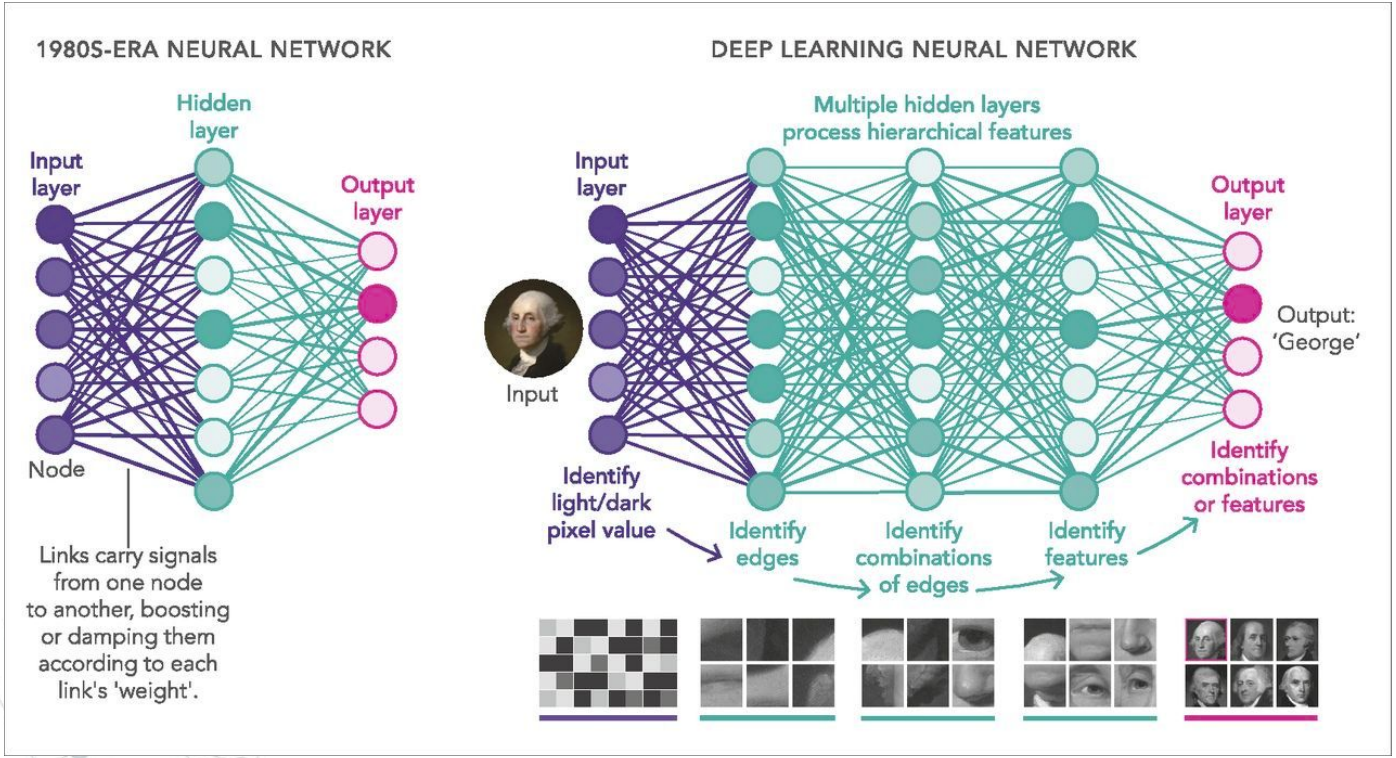
7.

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.



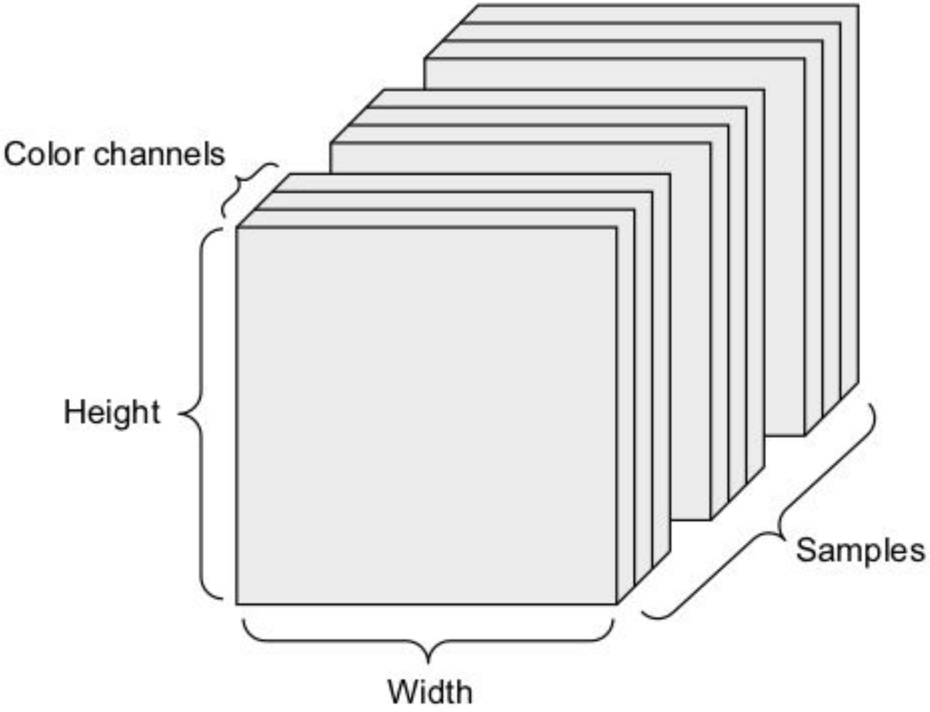
8.

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

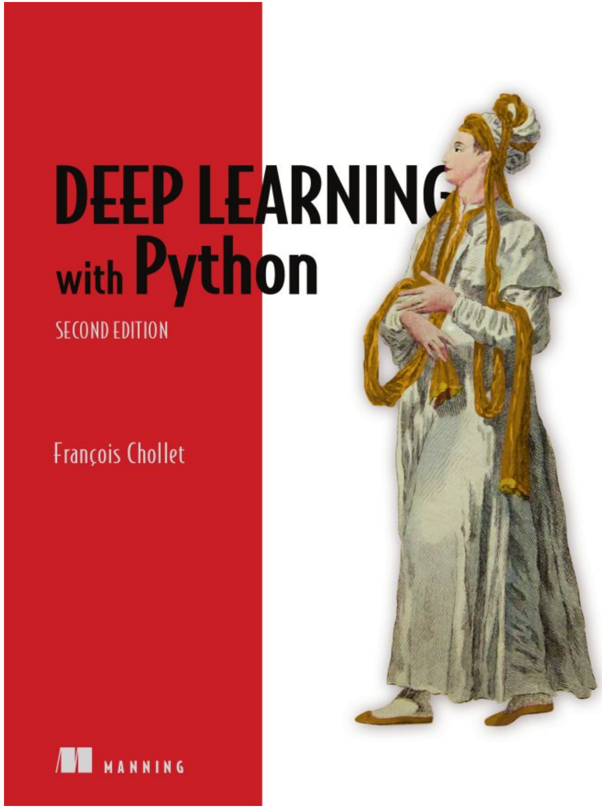
Image data



8.

Books

Book


DEEP LEARNING
with Python
SECOND EDITION
François Chollet
MANNING

Thanks!

Any questions?

You can find me at:

jorge.guerra881215@gmail.com