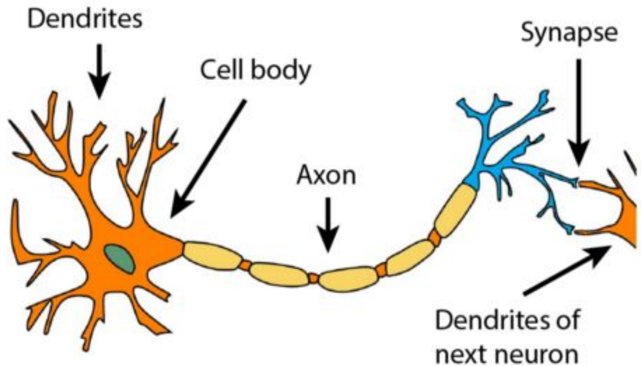
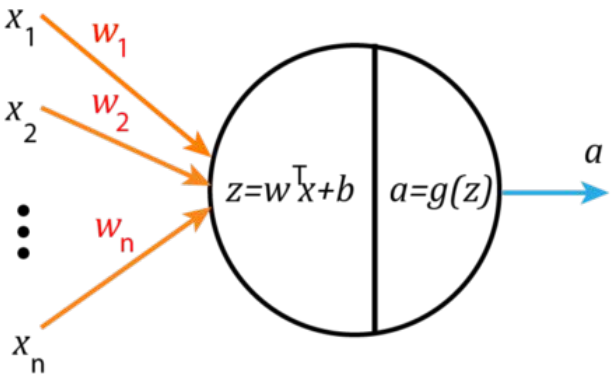
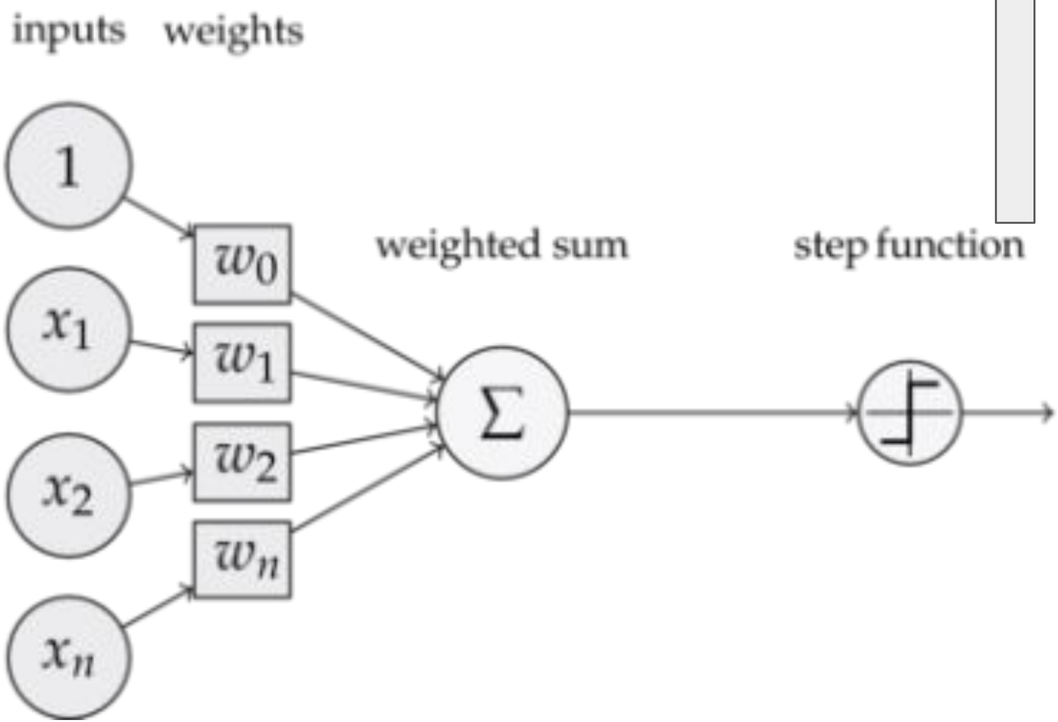
Neural Networks

ANN


Dendrites
Cell body
Axon
Synapse
Dendrites of
next neuron


x_1
x_2
x_n
w_1
w_2
w_n
$z=w^Tx+b$
$a=g(z)$
a

Perceptron



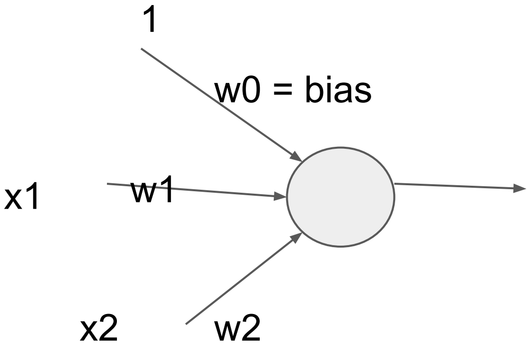
inputs
weights
1
x_1
x_2
x_n
w_0
w_1
w_2
w_n
weighted sum
Σ
step function

Perceptron - Pseudo Código

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w}$ randomly;
while !*convergence* **do**
    Pick random $\mathbf{x} \in P \cup N$;
    if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
    end
    if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
    end
end
//the algorithm converges when all the inputs are classified correctly

---



w0(bias) = w0 + etha(t-o)*1
w1 = w1 + etha(t-o)*x1
w2 = w2 + etha(t-o)*x2

Perceptron - Pseudo Código

$$w_i \leftarrow w_i + \Delta w_i$$

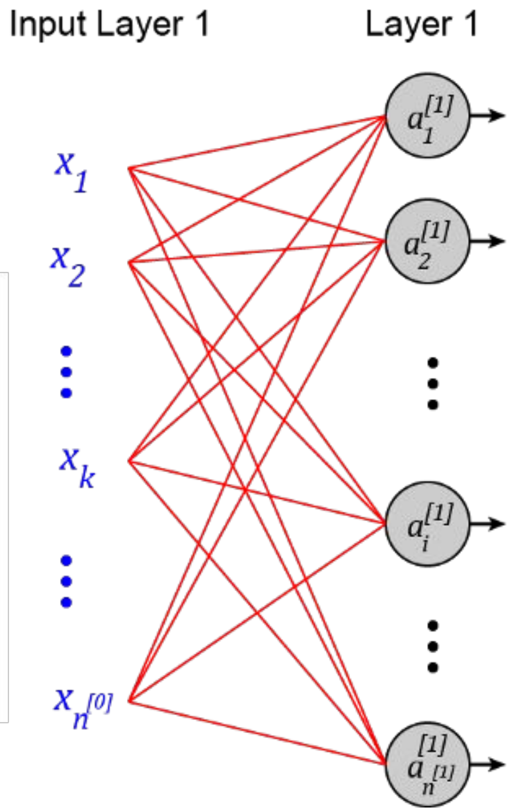
where

$$\Delta w_i = \eta(t - o)x_i$$

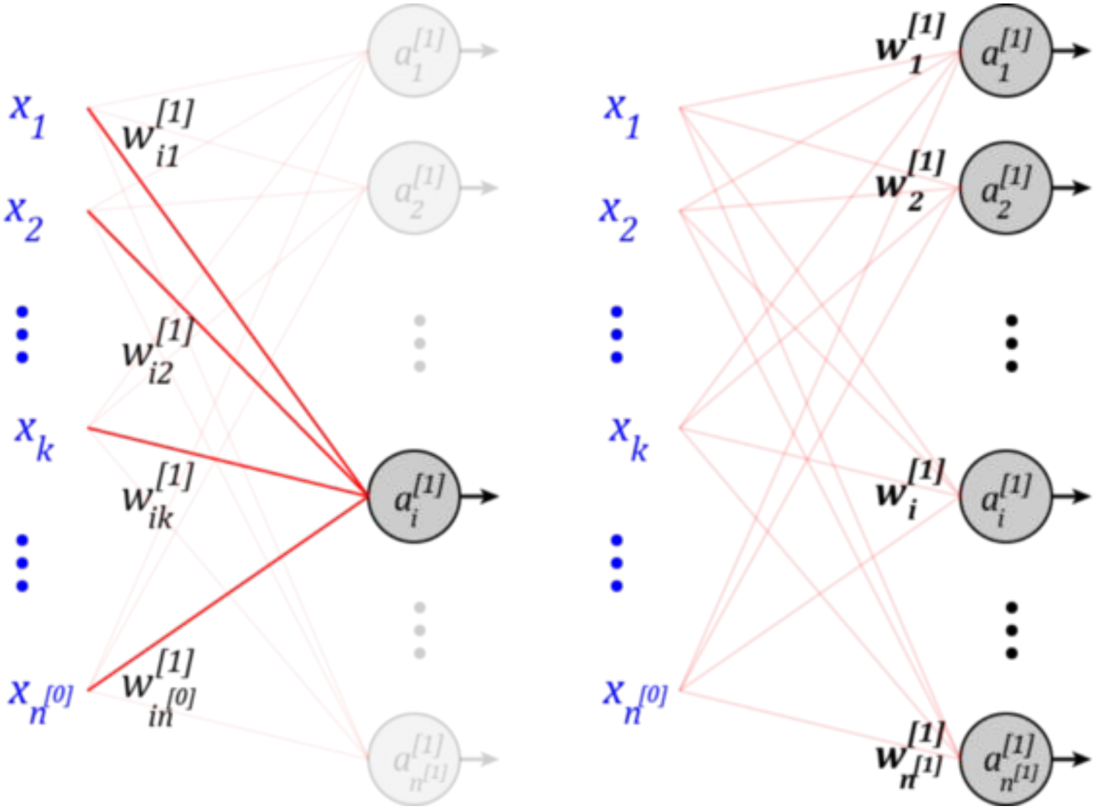
Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., 0.1) called *learning rate*

ANN



ANN

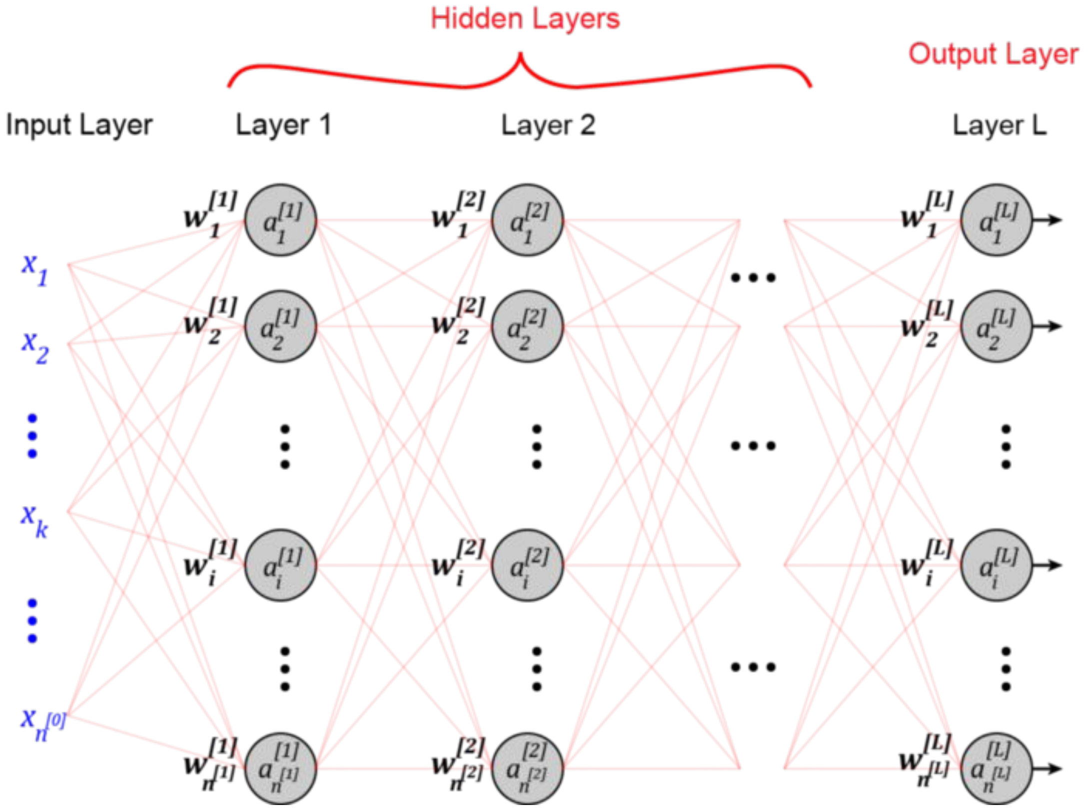


x1
x2
xk
xn[0]
w[1] i1
w[1] i2
w[1] ik
w[1] in[0]
a[1] 1
a[1] 2
a[1] i
a[1] n[1]
w[1] 1
w[1] 2
w[1] i
w[1] n[1]

ANN

$$a_i^{[1]} = g^{[1]}\left(z_i^{[1]}\right) = g^{[1]}\left(\sum_k w_{ik}^{[1]} x_k + b_i^{[1]}\right) = g^{[1]}\left(\boldsymbol{w_i^{[1]}}^T \boldsymbol{x} + b_i^{[1]}\right) \qquad (55)$$

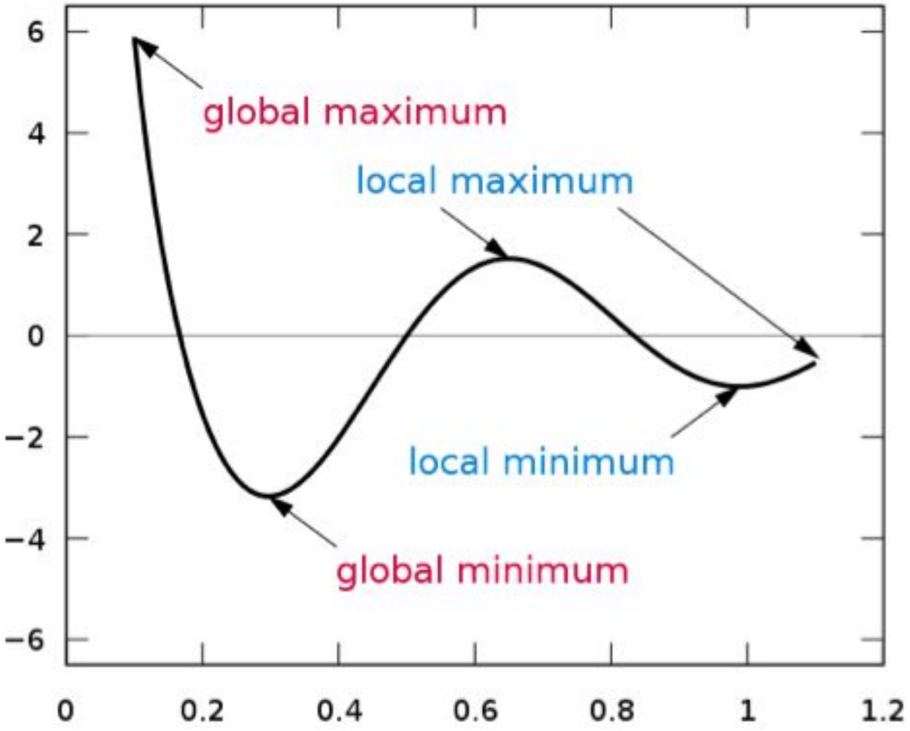
ANN



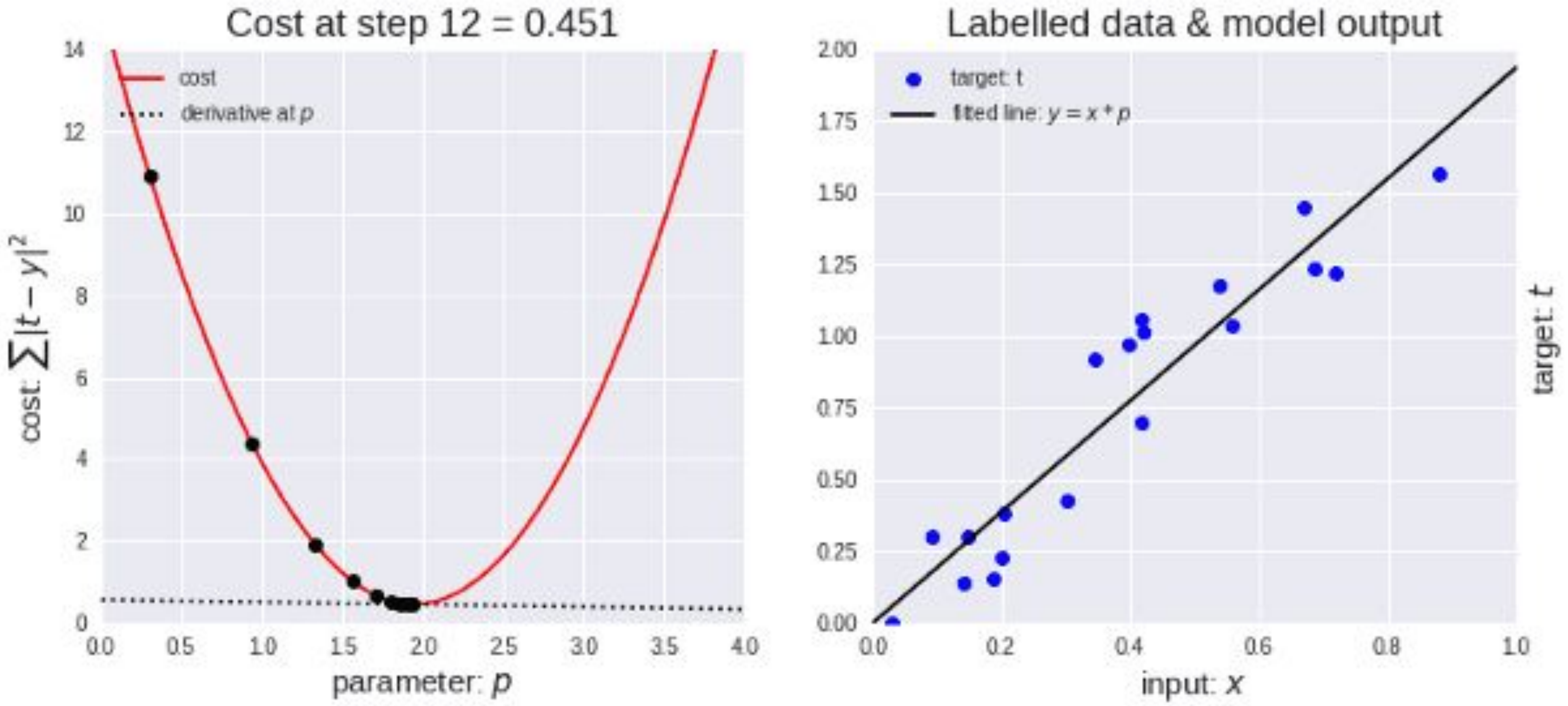
Hidden Layers
Output Layer
Input Layer
Layer 1
Layer 2
Layer L

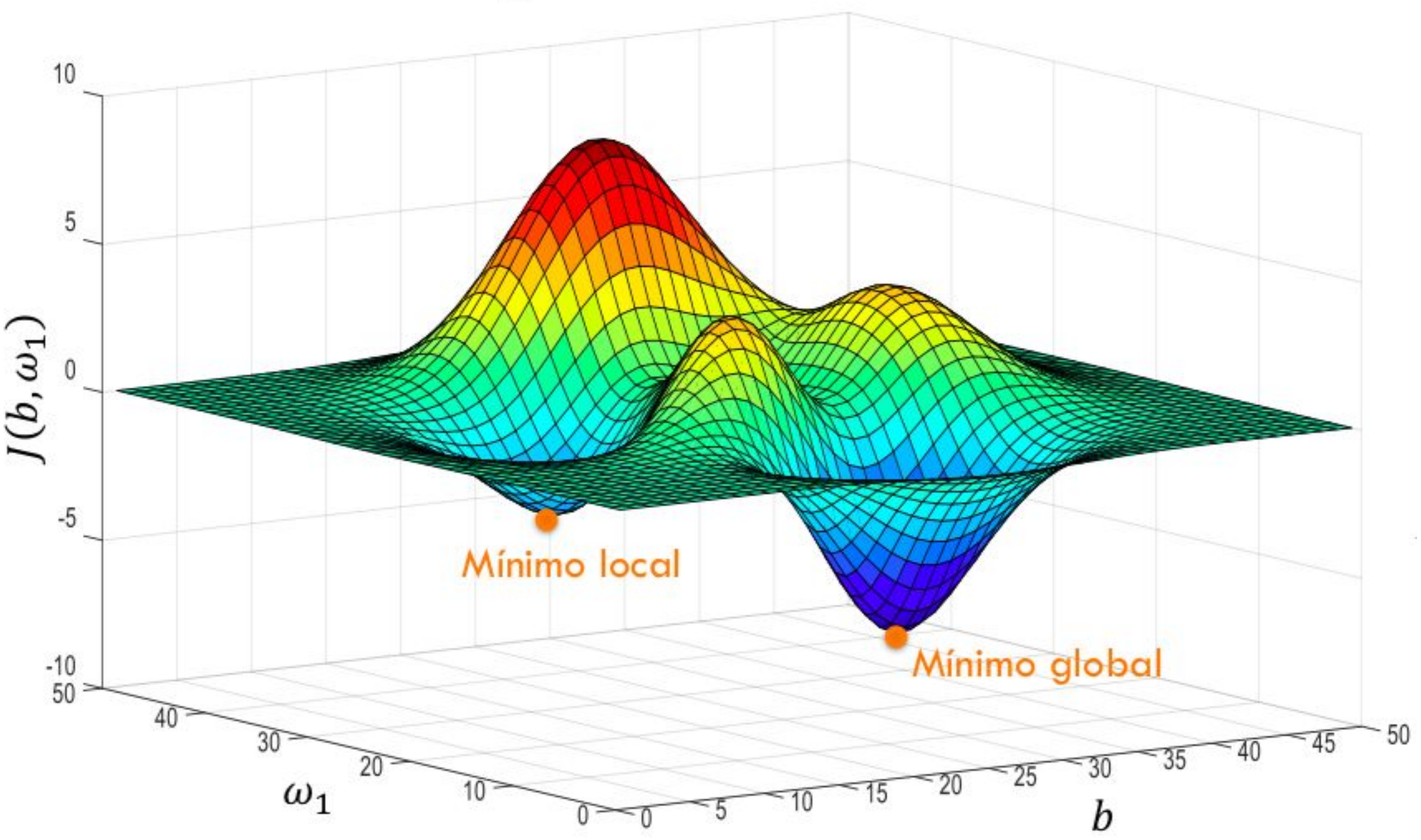
ANN - Gradient Descent

Método usado para achar os parâmetros de minimização da função custo (J)



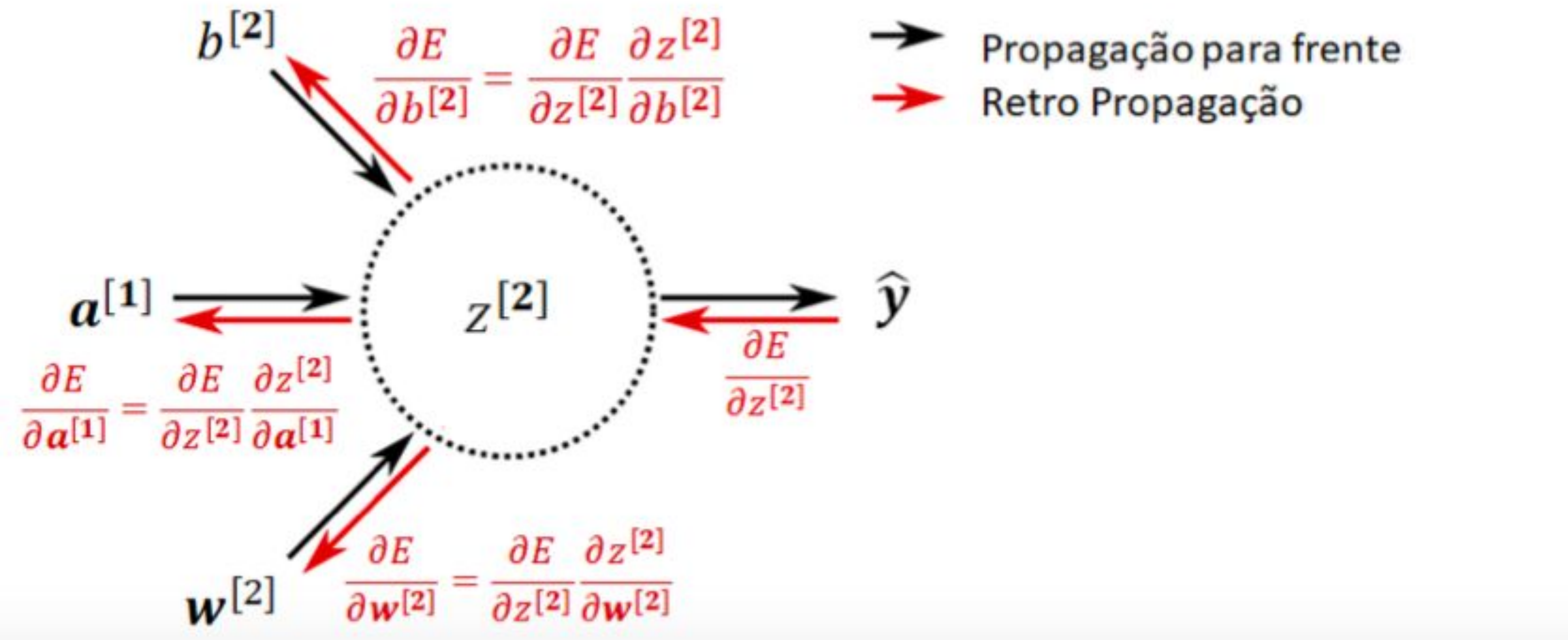
Gradient Descent


Cost at step 12 = 0.451
cost
derivative at p
cost: $\sum|t-y|^2$
parameter: p
14
12
10
8
6
4
2
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Labelled data & model output
target: t
fitted line: y = x * p
target: t
input: x
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0

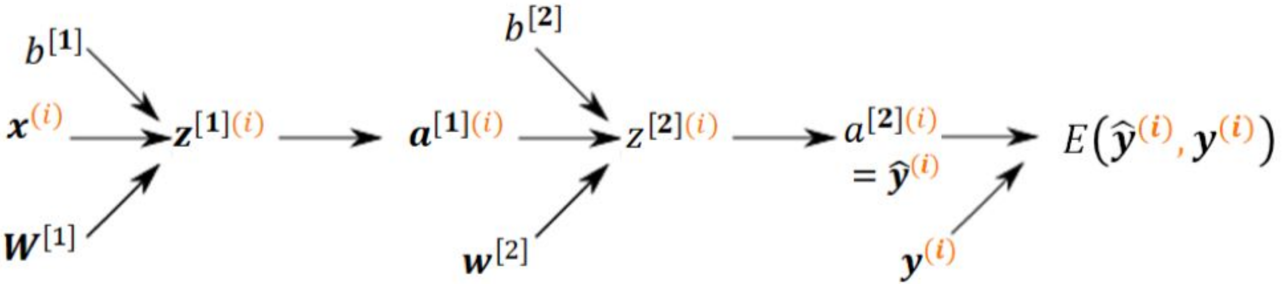


$J(b, \omega_1)$
10
5
0
-5
-10
Mínimo local
Mínimo global
ω_1
50
40
30
20
10
0
0
5
10
15
20
25
30
35
40
45
50
b

Backpropagation



Backpropagation



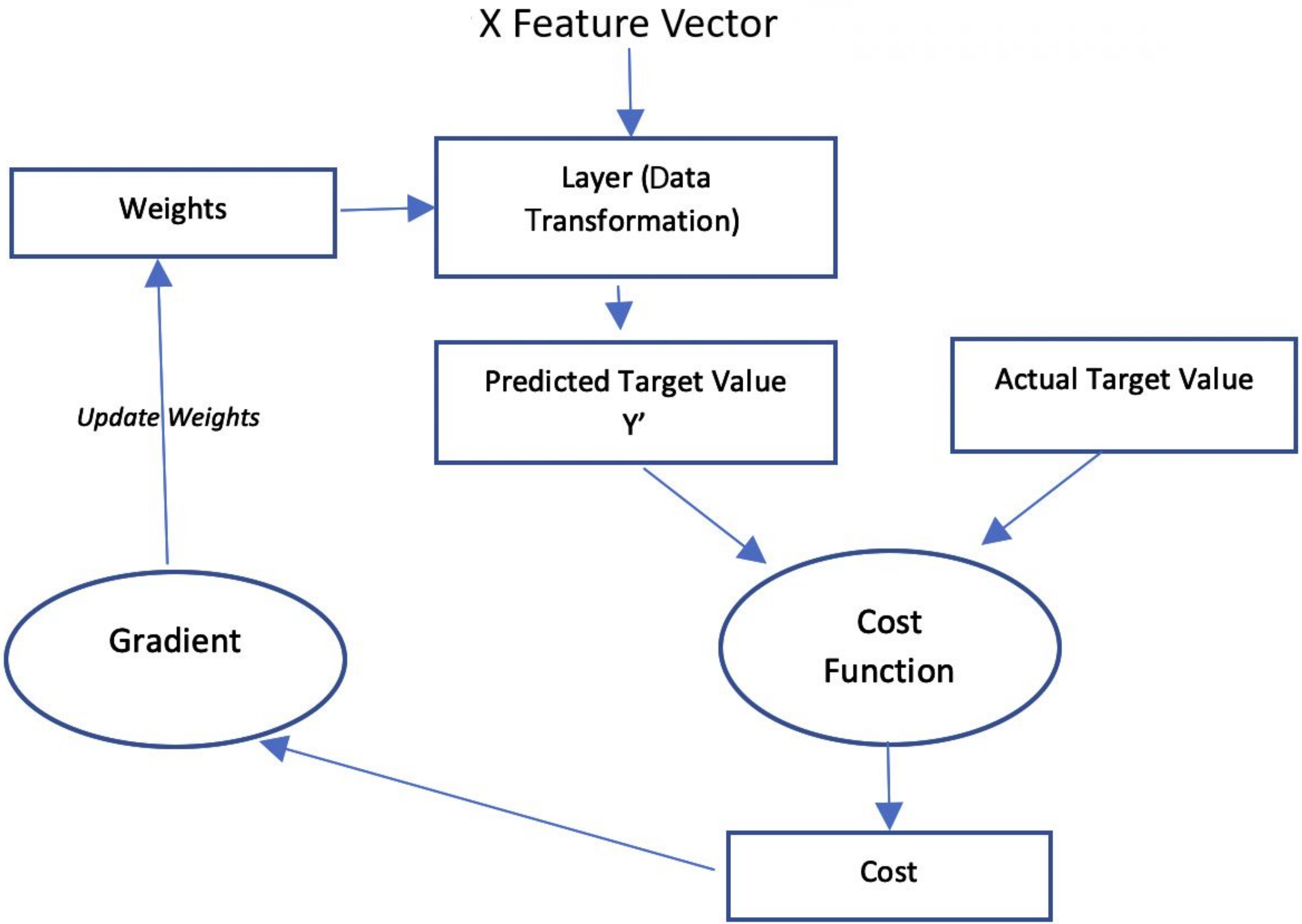
$$J(\mathbf{W}, \mathbf{b}) = \frac{1}{m} \sum_{i=1}^{m} E\left(\hat{y}^{(i)}, y^{(i)}\right)$$

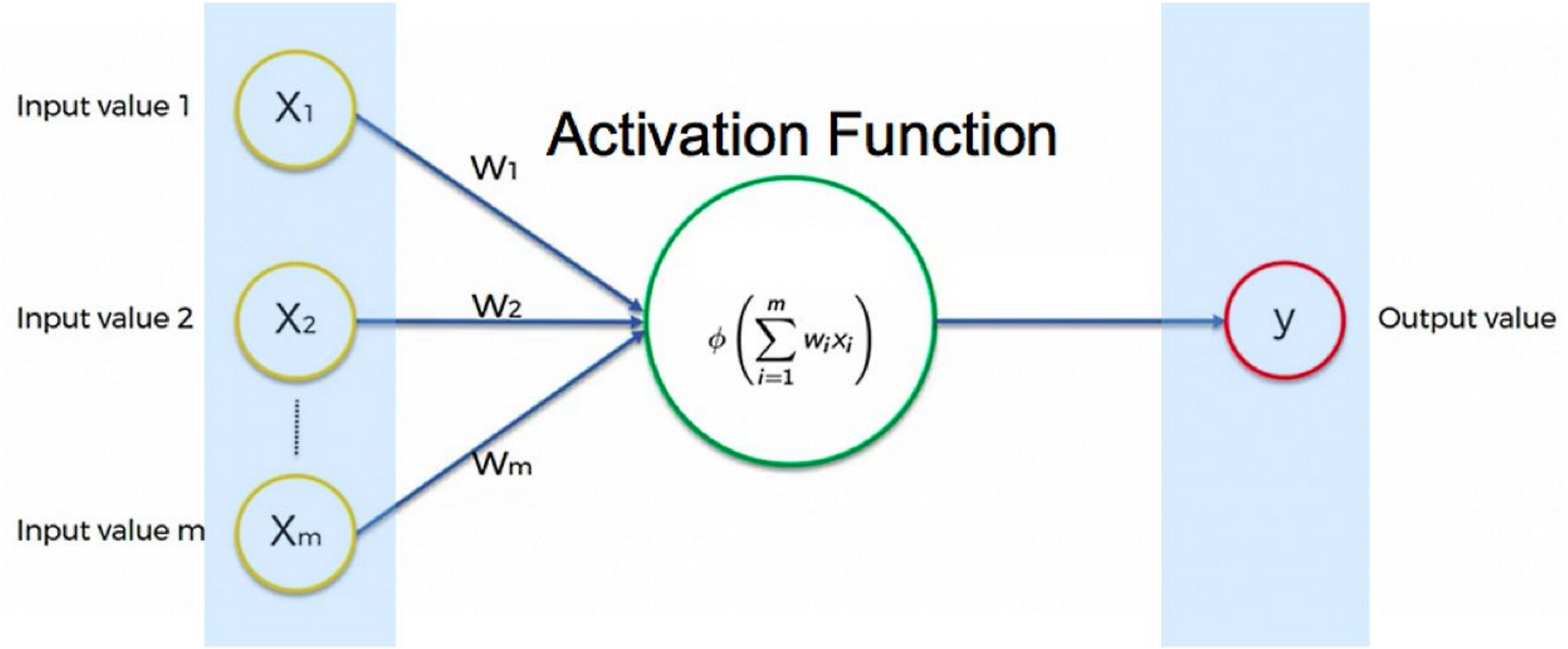
$$\frac{\partial E\left(\hat{y}^{(i)}, y^{(i)}\right)}{\partial w_{k,j}^{[l]}} = \frac{\partial E\left(\hat{y}^{(i)}, y^{(i)}\right)}{\partial a_k^{[l](i)}} \frac{\partial a_k^{[l](i)}}{\partial z_k^{[l](i)}} \frac{\partial z_k^{[l](i)}}{\partial w_{k,j}^{[l]}}$$

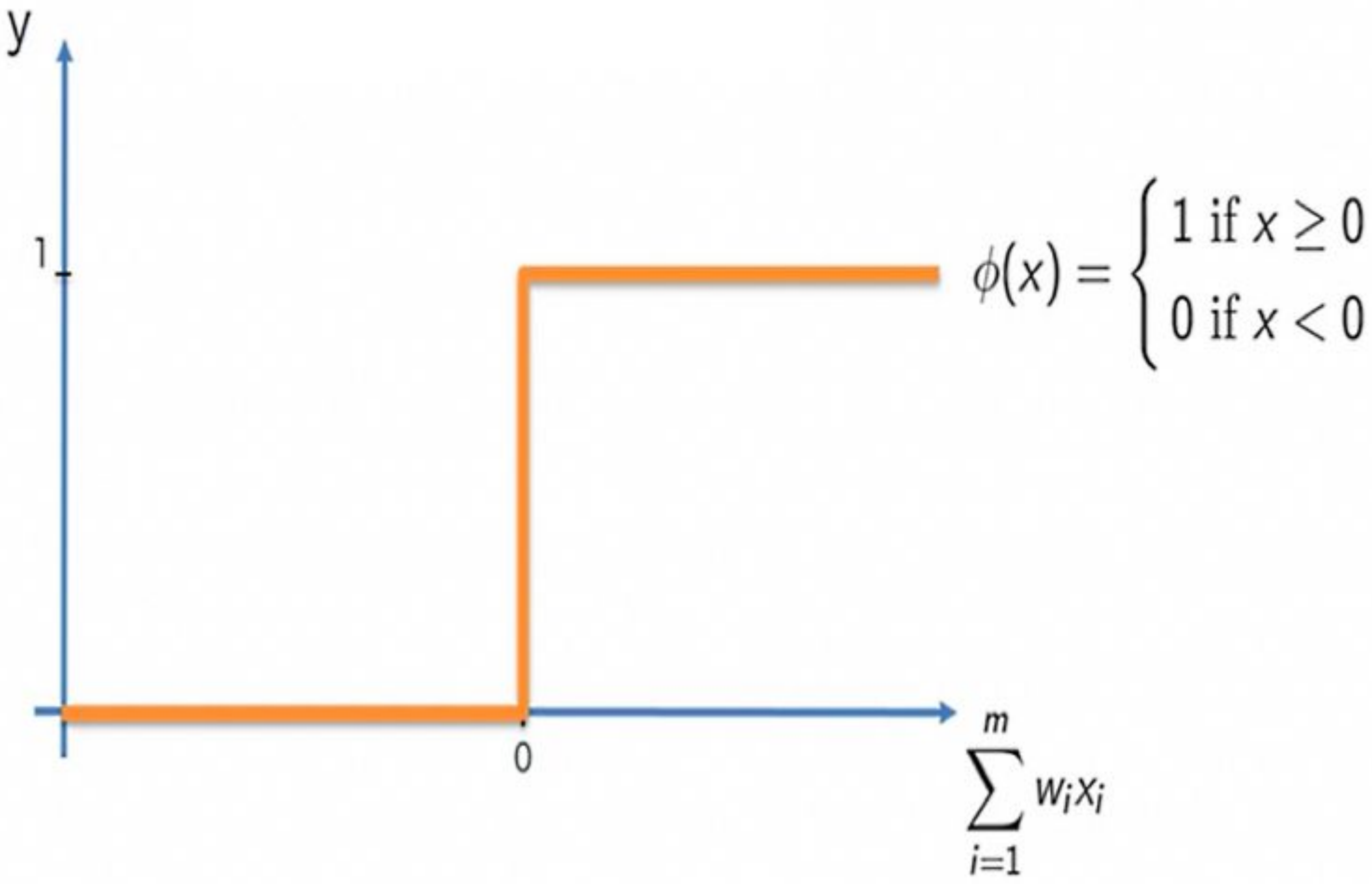
$$\frac{\partial E\left(\hat{y}^{(i)}, y^{(i)}\right)}{\partial b_k^{[l]}} = \frac{\partial L\left(\hat{y}^{(i)}, y^{(i)}\right)}{\partial a_k^{[l](i)}} \frac{\partial a_k^{[l](i)}}{\partial z_k^{[l](i)}} \frac{\partial z_k^{[l](i)}}{\partial b_k^{[l]}}$$

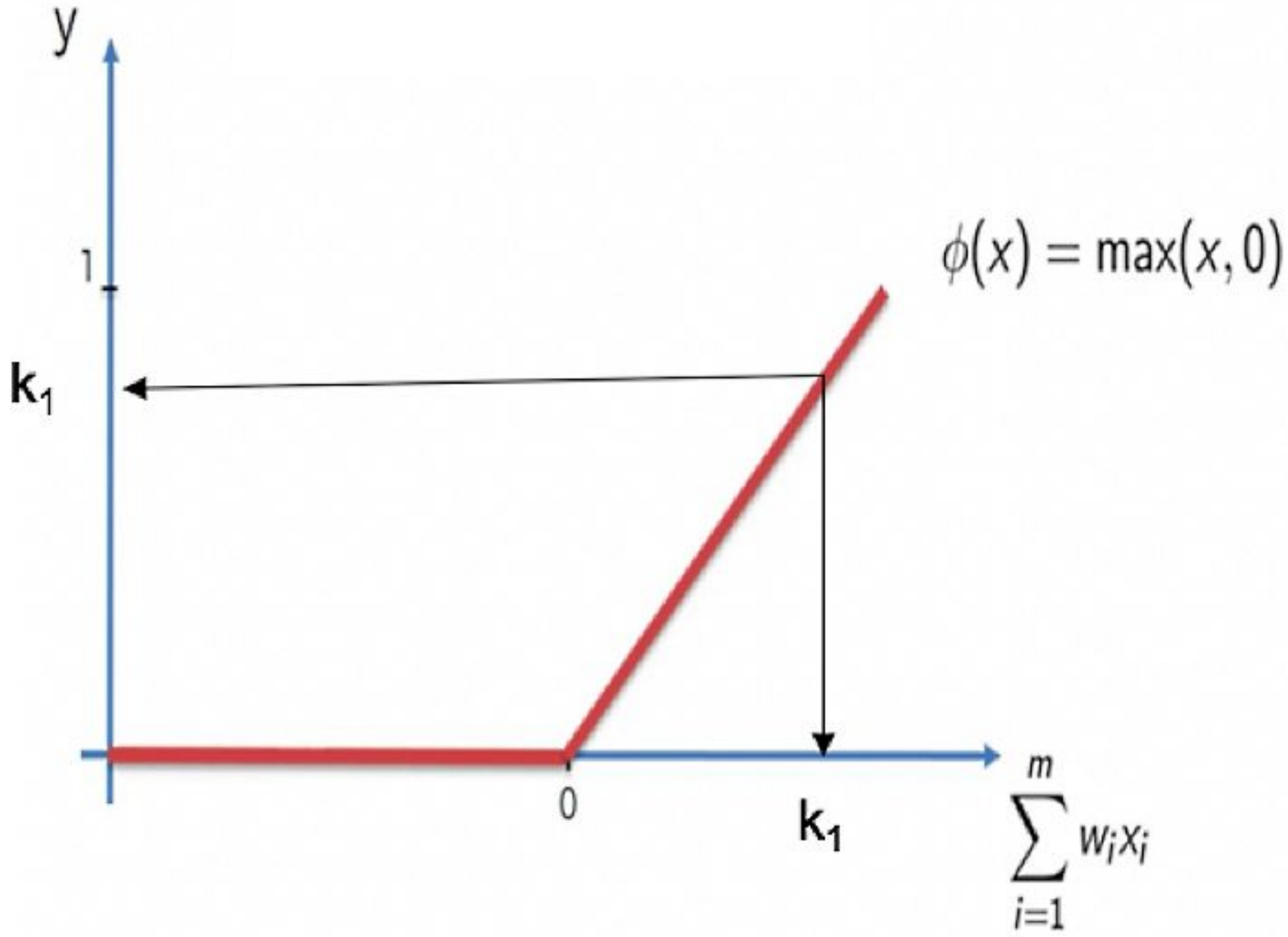
$$z_k^{[L](i)} = w_{k,j}^{[L]} a_k^{[L-1](i)} + b_k^{[L]}$$

$$a_k^{[L](i)} = g\left(z_k^{[L](i)}\right) = \hat{y}^{(i)}$$

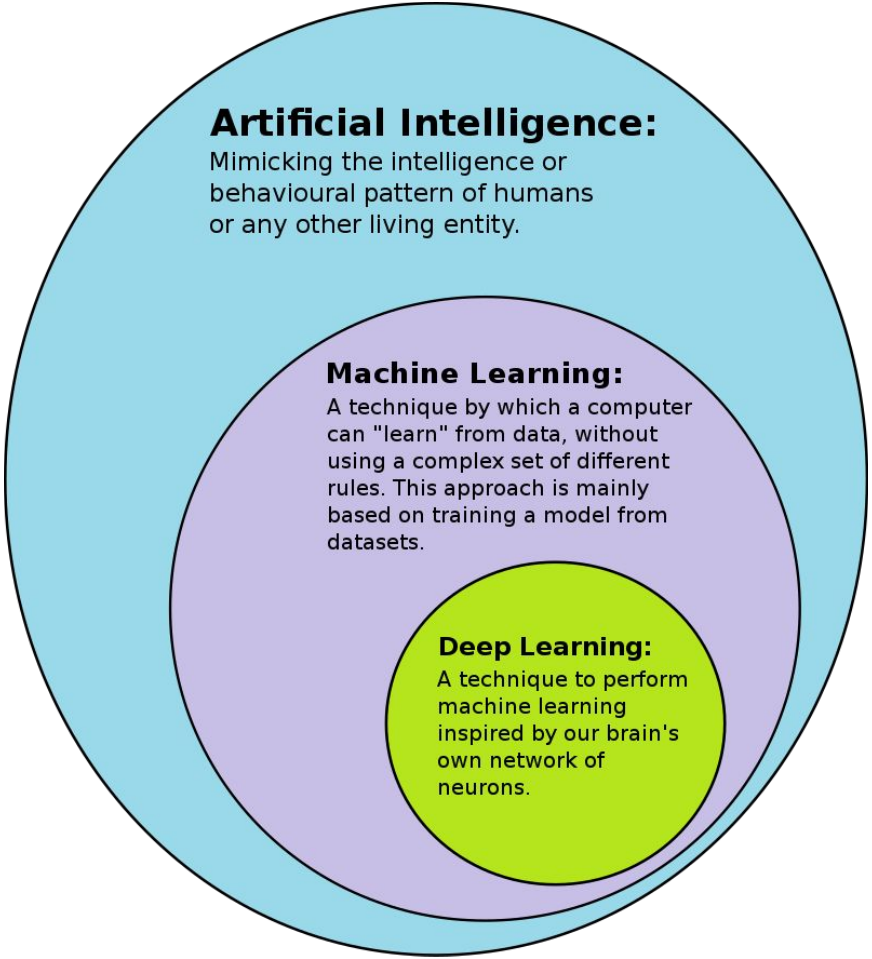

X Feature Vector
Weights
Layer (Data Transformation)
Predicted Target Value Y'
Actual Target Value
Update Weights
Gradient
Cost Function
Cost


Input value 1
Input value 2
Input value m
X1
X2
Xm
W1
W2
Wm
Activation Function
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
y
Output value

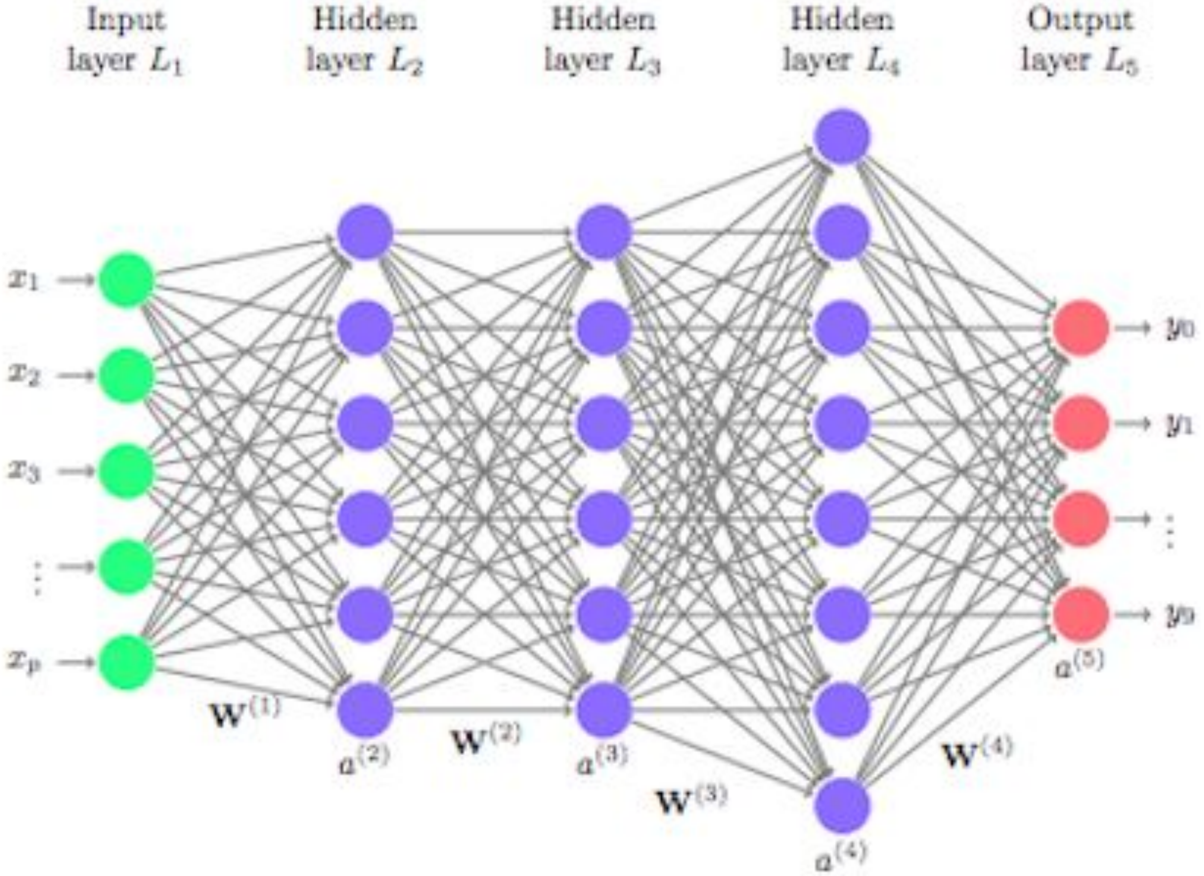

y
1
0
$\phi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$
$\sum_{i=1}^{m} w_i x_i$


y
1
$\phi(x) = \max(x, 0)$
$\mathbf{k_1}$
0
$\mathbf{k_1}$
$\sum_{i=1}^{m} w_i x_i$

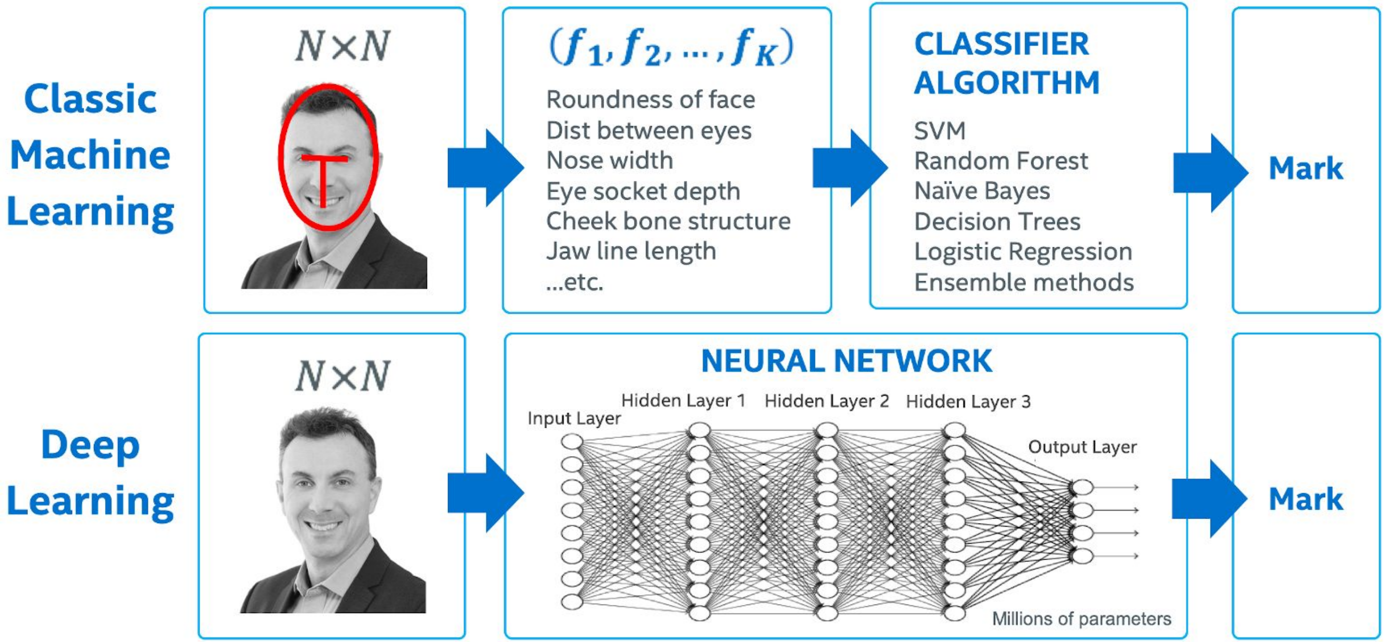
Deep-Learning



Deep-Learning



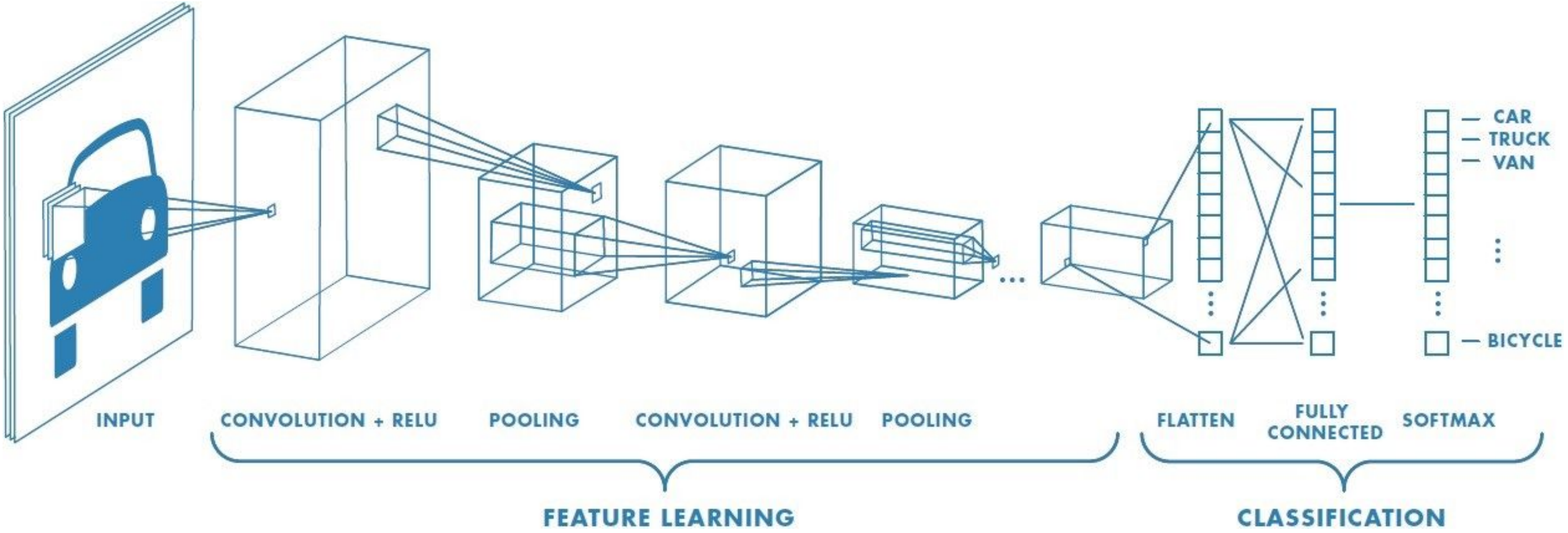
Deep-Learning

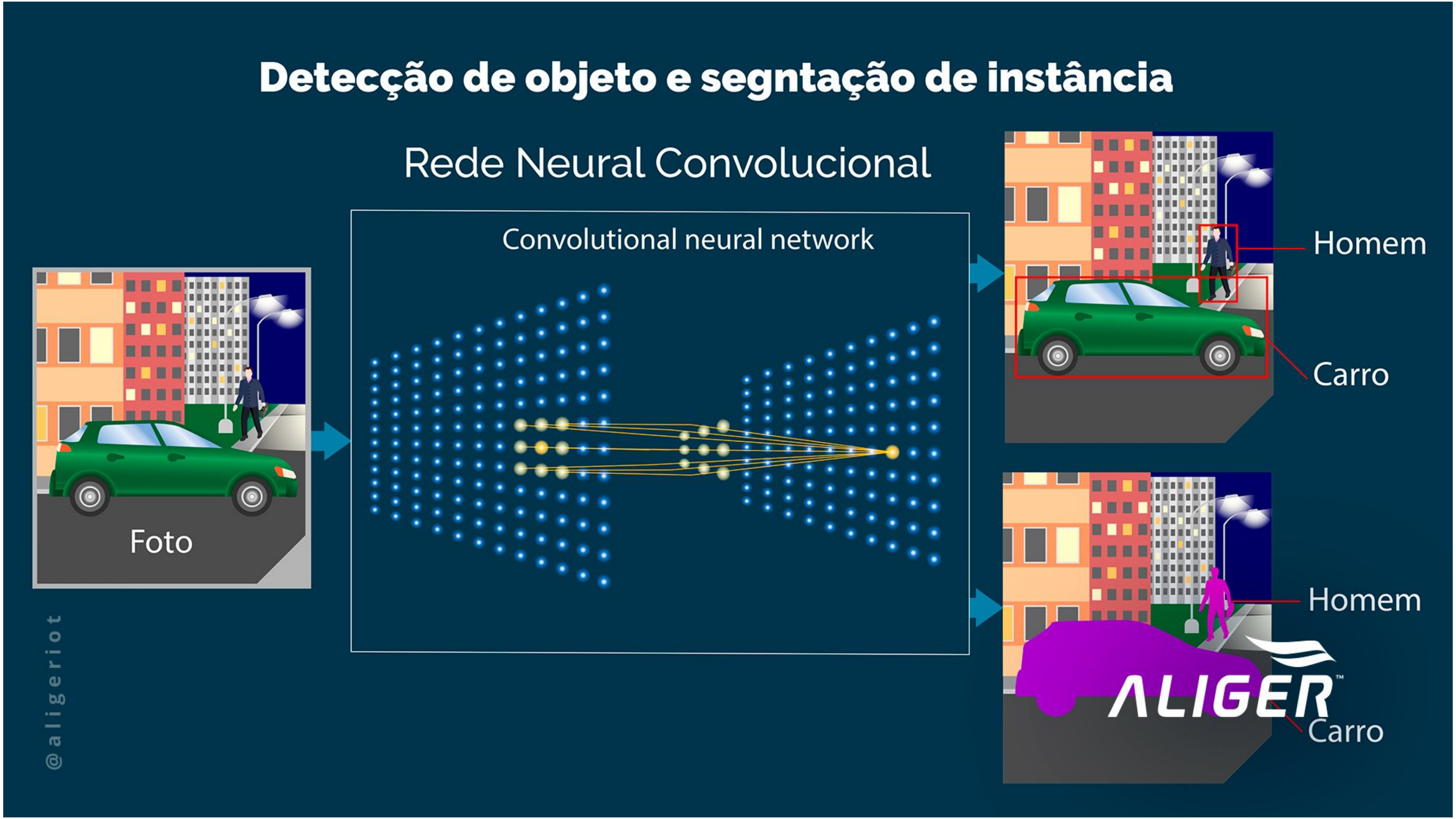


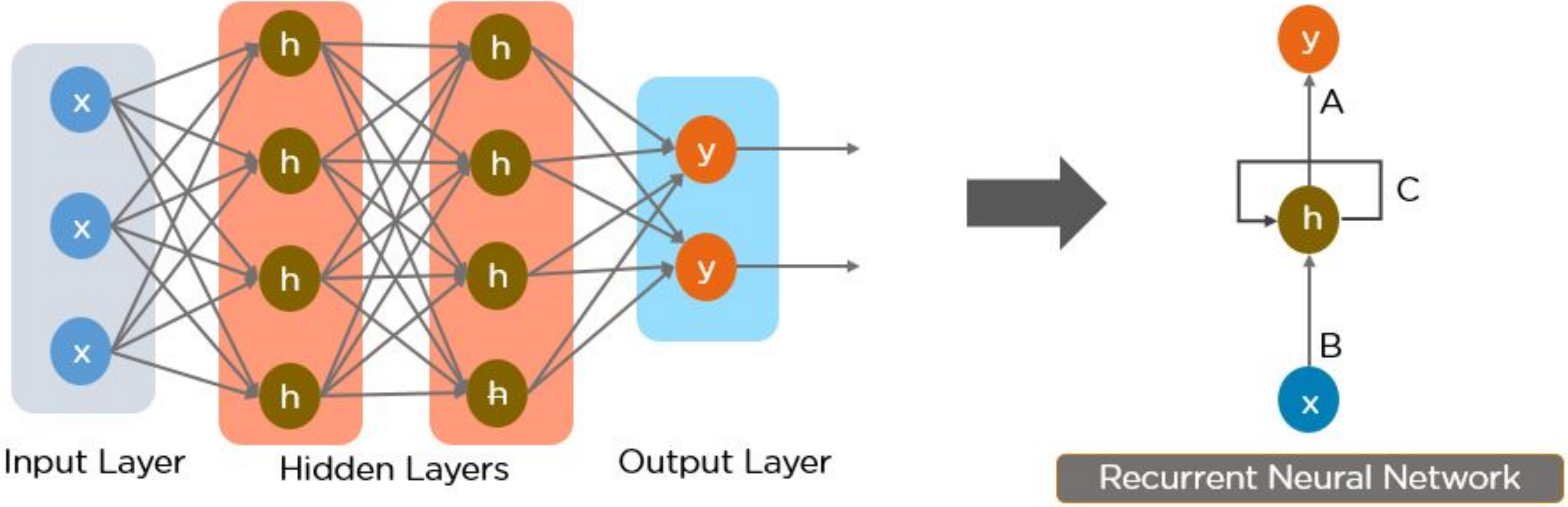
Deep-Learning

Redes Neurais Recorrentes

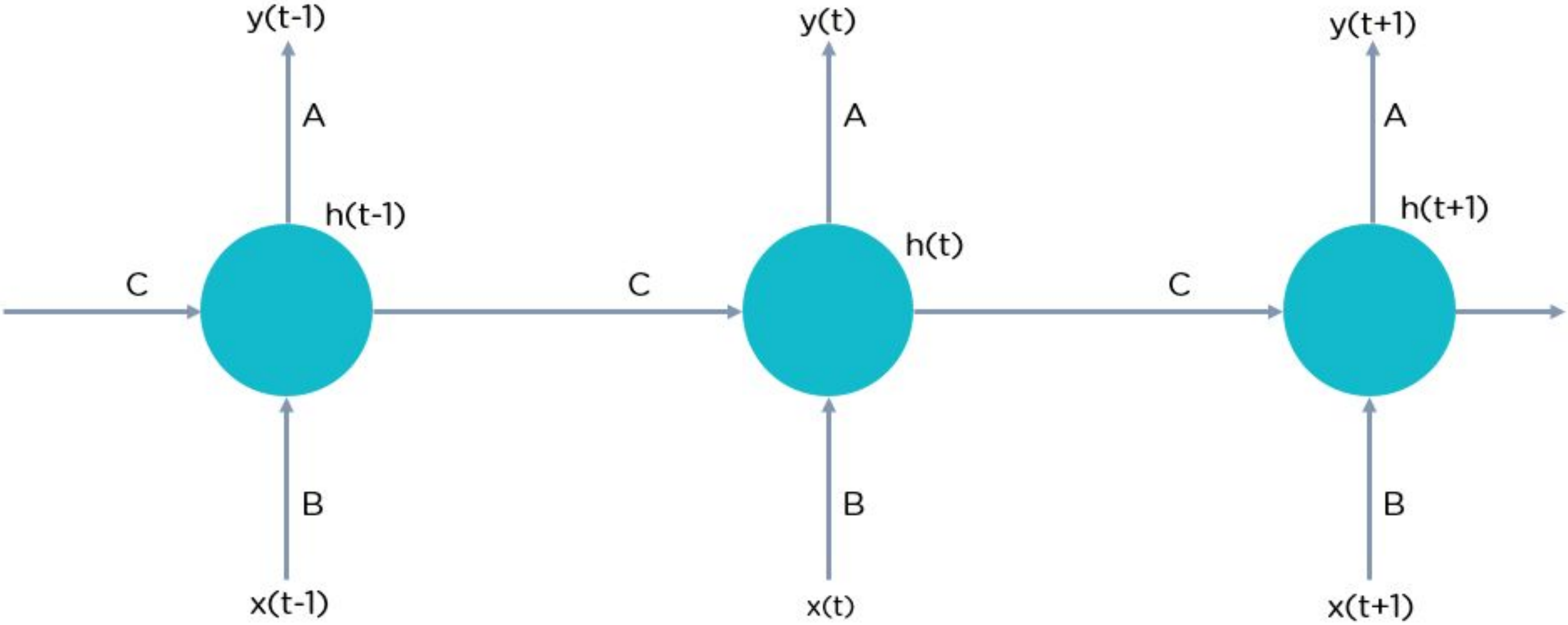
Redes Convolucionais


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


Detecção de objeto e segntação de instância
Rede Neural Convolucional
Convolutional neural network
Foto
Homem
Carro
Homem
Carro
ALIGER
@aligeriot



RNN -Redes Neurais Recorrentes



$$h(t) = f_c\,(h(t-1), x(t))$$

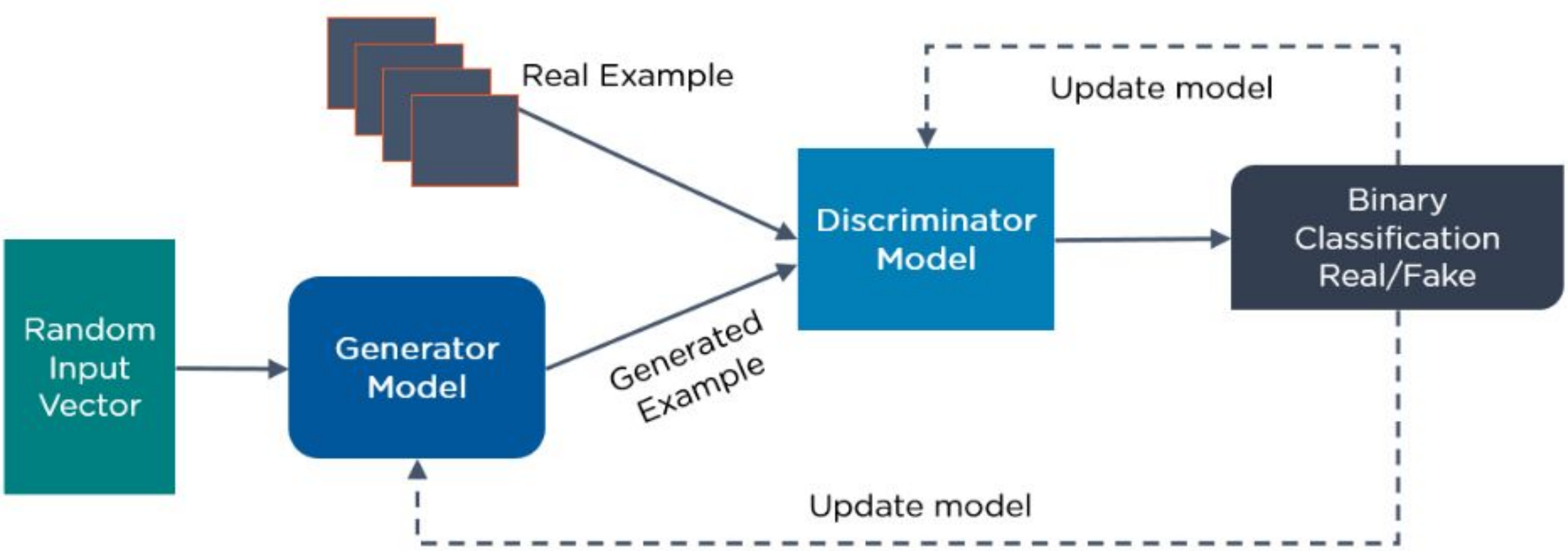
$h(t)$ = new state

f_c = function with parameter c

$h(t-1)$ = old state

$x(t)$ = input vector at time step t

Generative Adversarial Networks (GANs)




source
destination
Coarse styles copied

1 ; 2 ; 3; 4 ; 5; 6; 7

0.1	0.3	….
0.2	0.4	...
0.3	0.7	
0.4	0.8	
0.5	0.1	
0.6	0.2	
0.7	0.6	….

N x M * M x D =

1*0.1 + 2*0.2 + …. + 7*0.7 =

1*0.3 + 2*0.4 + 3*0.7 + …. + 7*06 =