FACULTAD DE
INGENIERIA ECONOMICA,
ESTADISTICA Y
CIENCIAS SOCIALES

INTRODUCCIÓN AL MACHINE LEARNING Y DESARROLLO DE REDES NEURONALES EN C++

LIN CHIU CHEN YANG

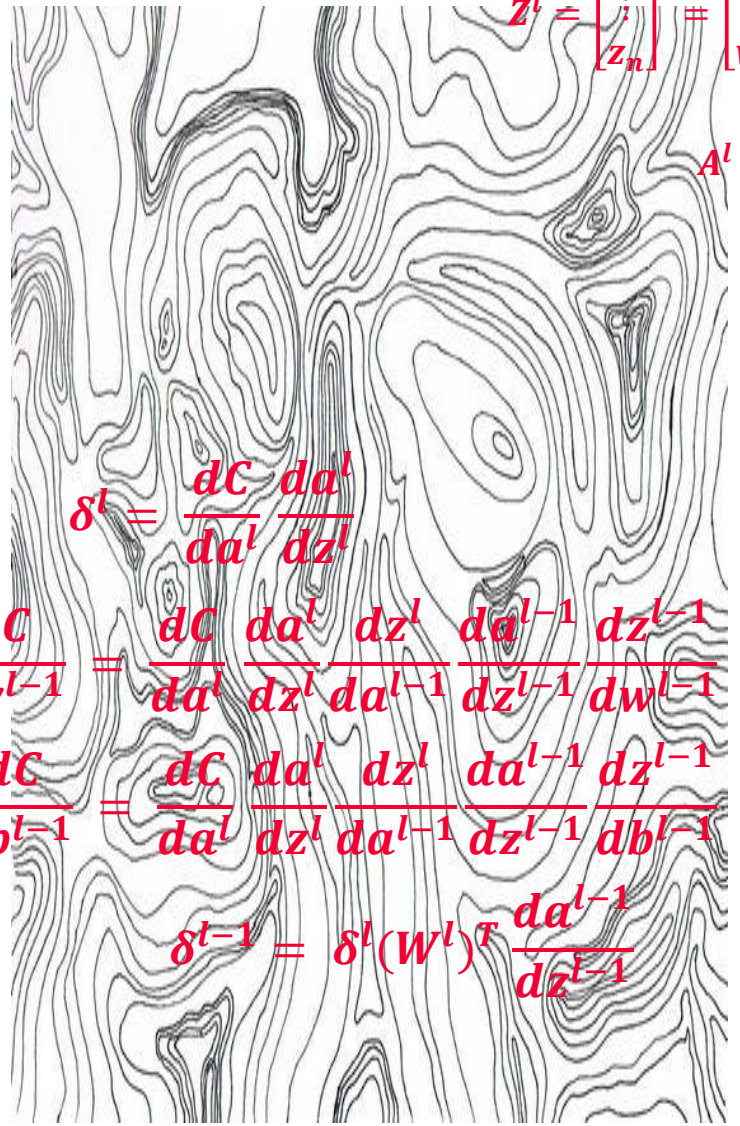
LLACSA MACAVILCA JHONATAN

LLANA ARROYO ALDAIR

$$Z^l = W^l A^{l-1} + b^l$$

$$Z^l = \begin{bmatrix} z_0 \\ \vdots \\ z_n \end{bmatrix}^l = \begin{bmatrix} w_{00} & \cdots & w_{0m} \\ \vdots & \ddots & \vdots \\ w_{n0} & \cdots & w_{nm} \end{bmatrix}^l \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix}^{l-1} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}^l$$

$$A^l = \begin{bmatrix} A_0 \\ \vdots \\ A_n \end{bmatrix}^l = \sigma\left(\begin{bmatrix} z_0 \\ \vdots \\ z_n \end{bmatrix}^l \right)$$

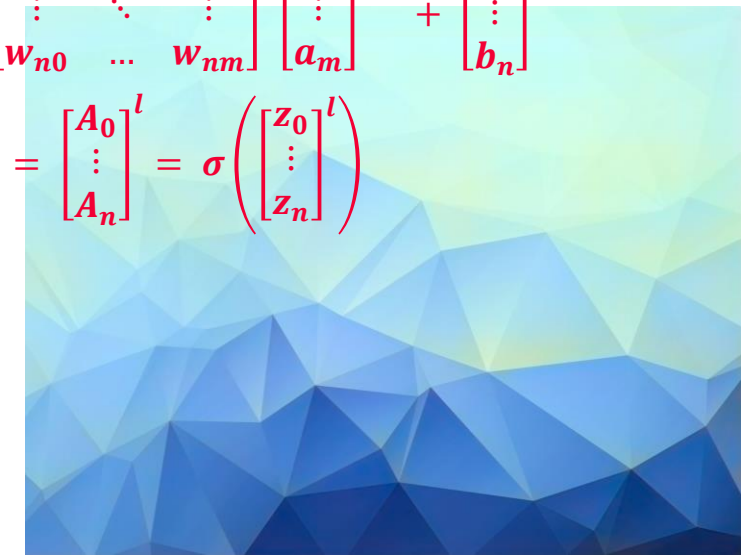


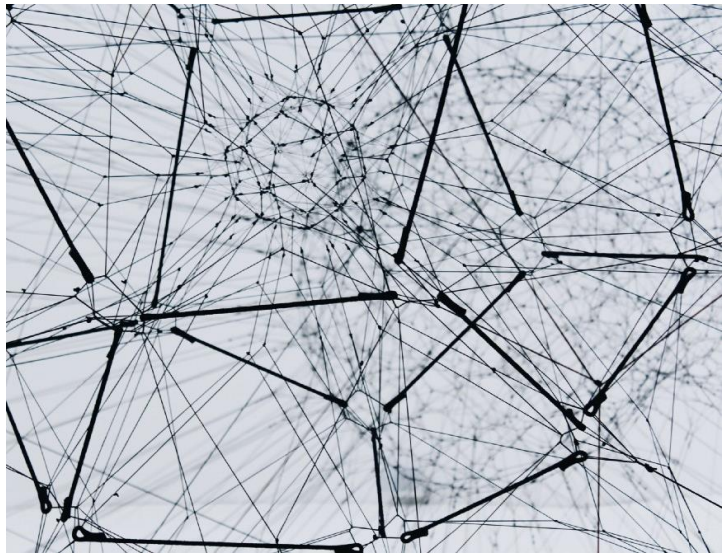
$$\delta^l = \frac{dC}{da^l}\frac{da^l}{dz^l}$$

$$\frac{dC}{dw^{l-1}} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{da^{l-1}}\frac{da^{l-1}}{dz^{l-1}}\frac{dz^{l-1}}{dw^{l-1}}$$

$$\frac{dC}{db^{l-1}} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{da^{l-1}}\frac{da^{l-1}}{dz^{l-1}}\frac{dz^{l-1}}{db^{l-1}}$$

$$\delta^{l-1} = \delta^l (W^l)^T \frac{da^{l-1}}{dz^{l-1}}$$





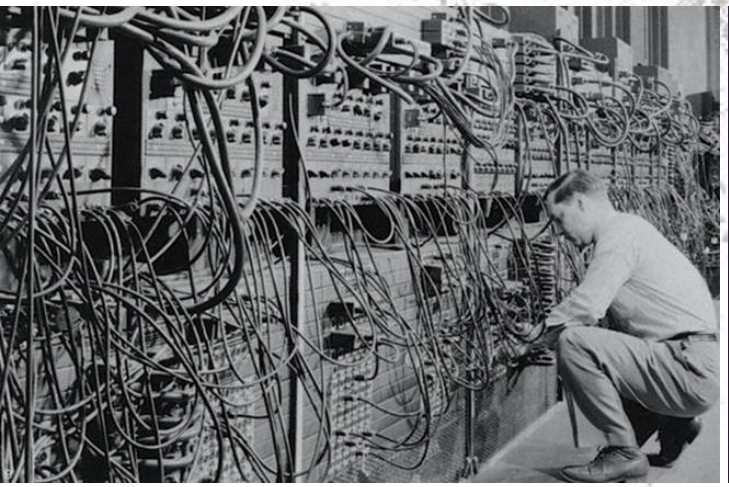
A

PRESENTACIÓN

- INTRODUCCIÓN
- OBJETIVOS

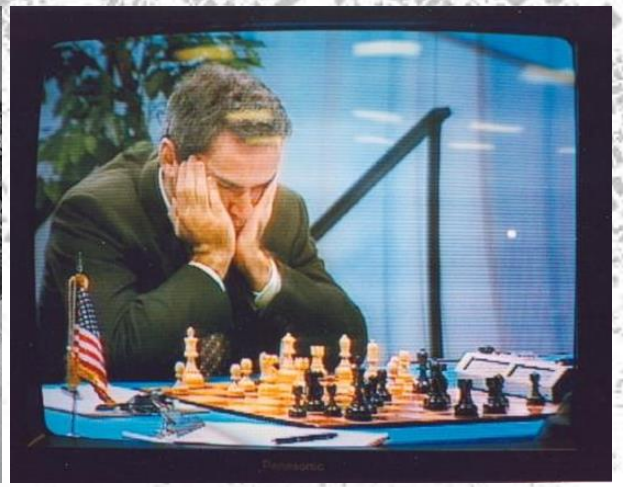
ARTIFICIAL INTELLIGENCE

1950



Alan Turing:
“Computing Machinery and Intelligence”

1997



IBM Deep Blue vs. Garri Kaspárov

2011

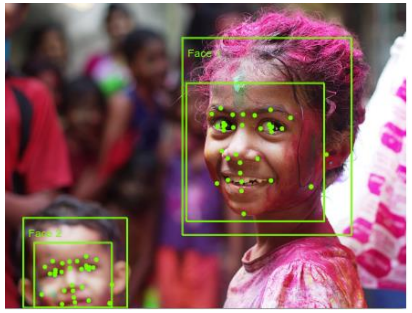
IBM Watson gana Jeopardy!

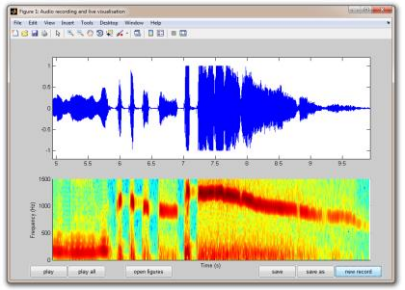
2016

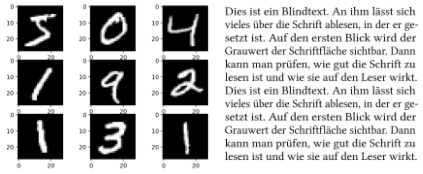
Google AlphaGo vence al campeón mundial de Go

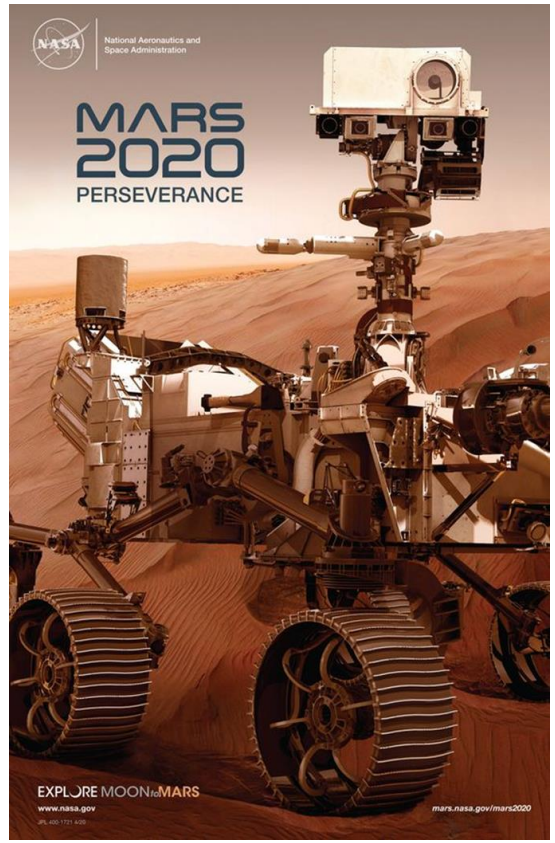
MACHINE LEARNING

- RECON. DE IMÁGENES
- RECON. FACIAL
- RECON. DE CARACTERES
- RECON. DE VOZ
- GENERACIÓN DE TEXTO
- PROS. DE LENGUAJE NATURAL
- GENERACIÓN DE IMÁGENES
- TRADUCCIÓN DE IDIOMAS
- CONDUCCIÓN AUTÓNOMA
- ROBÓTICA
- ANÁLICIS GENÉTICO
- PRONÓSTICO DE ENFERMEDAD
- PREDICCIÓN BURSÁTIL
- PREVENCIÓN DE FRAUDE
- AUTOMATIZACIÓN DE PROCESOS
- MOTORES DE BUSQUEDA









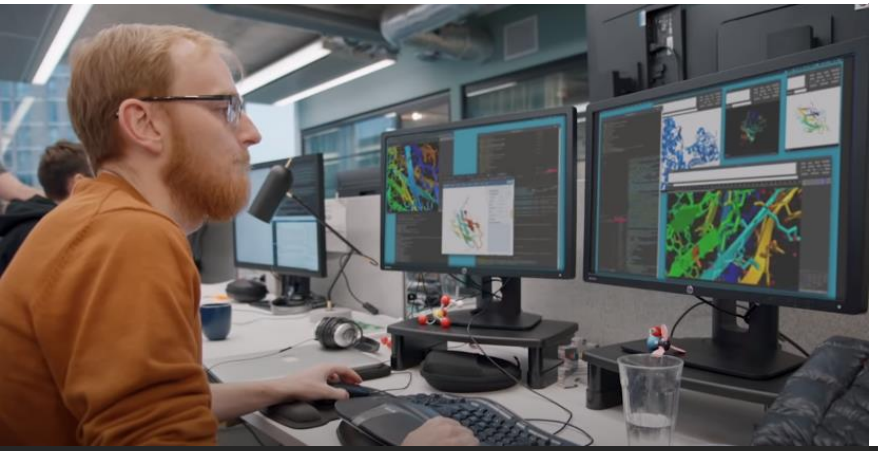
endeavour. Crucially, CASP chooses protein
that have only very recently been experime
de
the
tes
no
pre
pre
tru

BLOG POST
RESEARCH

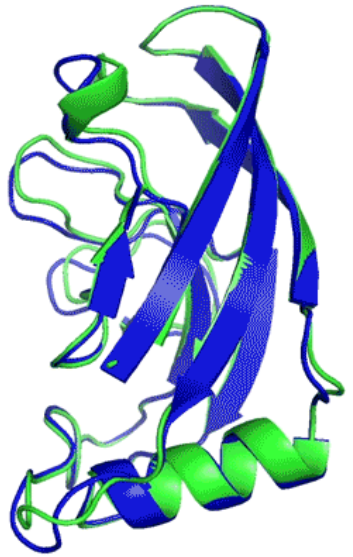
30 NOV 2020

AlphaFold: a solution to a 50-year-old grand challenge in biology

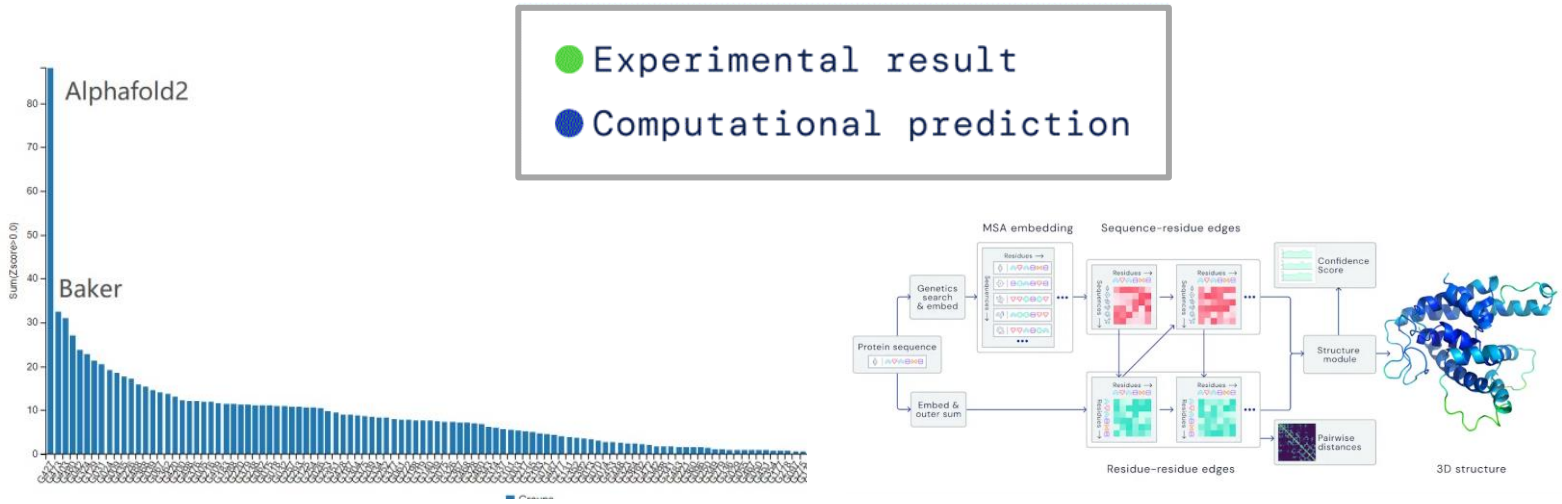
We're indebted to CASP's organisers and th
community, not least the experimentalists
structures enable this kind of rigorous asse



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)



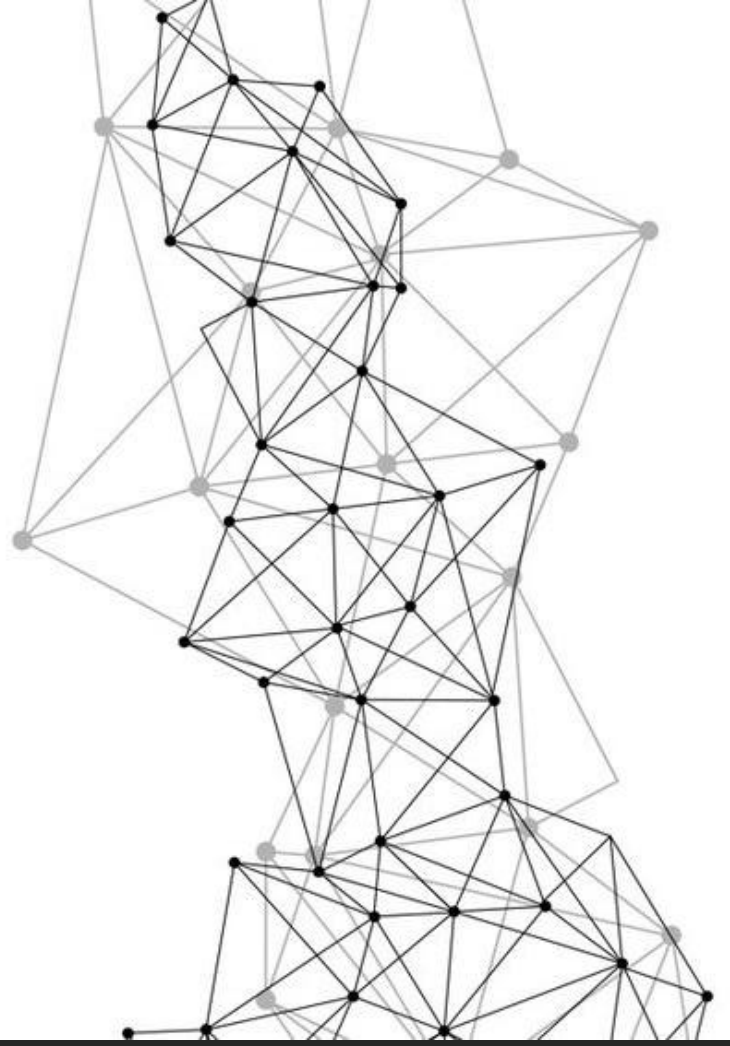
T1049 / 6y4f
93.3 GDT
(adhesin tip)



En Qué consiste ?

Porque este proyecto?

DESARROLLO DE REDES NEURONALES EN C++



- PONER EN PRÁCTICA LO APRENDIDO
- COMPARTIR LO APRENDIDO
- CONOCER EL CAMPO DE APRENDIZAJE AUTOMÁTICO
- DIVERSIFICAR CONOCIMIENTO

- ENTENDER LOS FUNDAMENTOS TEÓRICOS
- EMPLEAR LO APRENDIDO HACIA UN ENTORNO REAL
- INVESTIGACION
- BUSCAR SOLUCIONES

```
package multiLayer;

public class Neural_network {

    private double n;

    public Neural_network(int input0_Num, int layersNum, int[] neuronsNum, double n){
        initNeural_network();
    }
```

OBJETIVOS

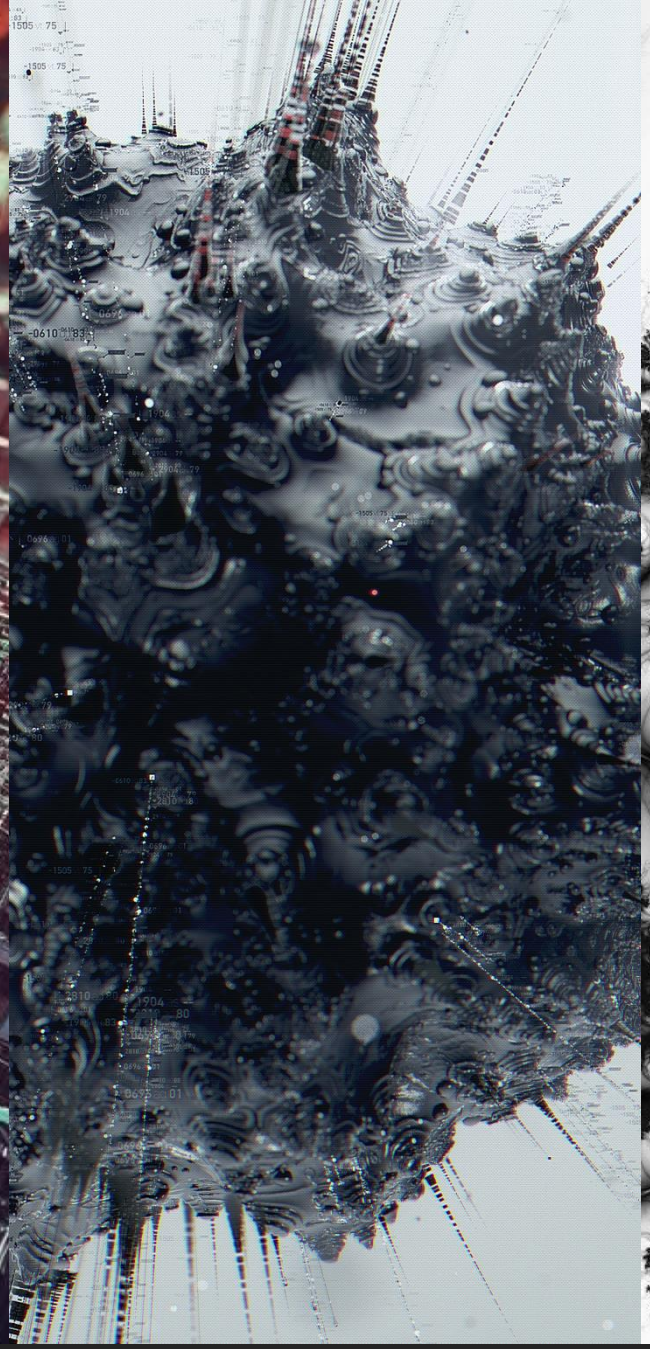
SER INGENIERO ES PROPONER SOLUCIONES INNOVADORAS
PARA UN MUNDO MEJOR

@hannakdraws

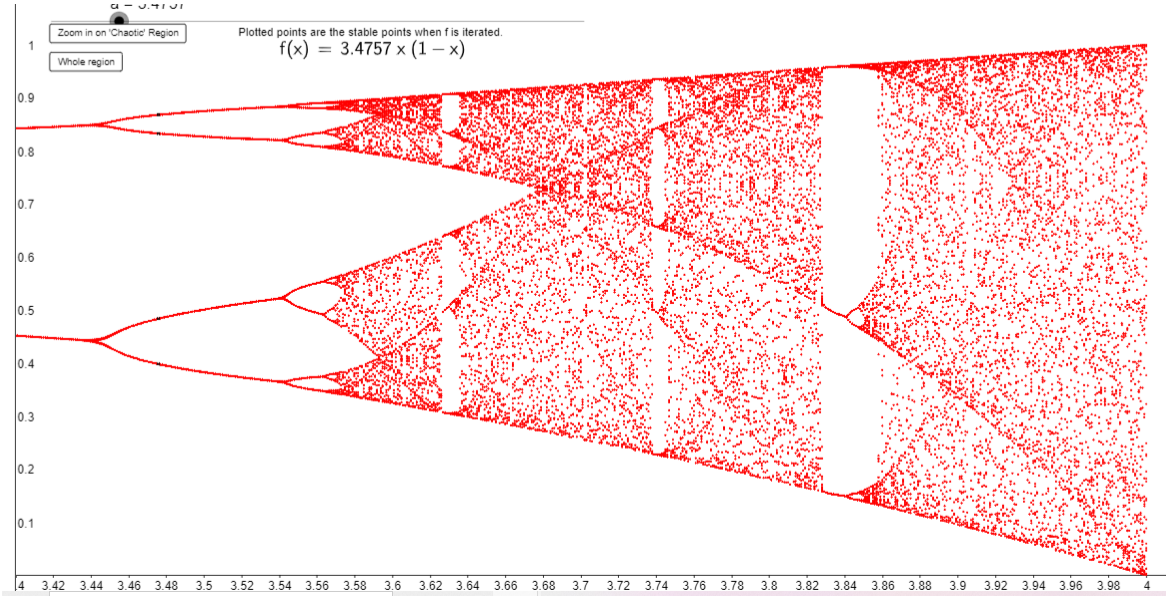
B

DESAROLLO

- FUNDAMENTOS
- DISEÑO



```
#include <iostream>
#include <math.h>
#include "And_Perceptron.h"

using namespace std;

And_Perceptron::And_Perceptron(do
this->w1=w1;
this->w2=w2;
this->b=b;
this->n=n;

cout<<"   Successful creation of the n
}

// ---- Natural Functions ----------
double And_Perceptron::trainNeuron
double error = 0;
0; i<4; i++){
 a = runNeuron(data[i][0],data[i][1]);
-= 50*pow(data[i][2]-a, 2);
endl<<"-------------------------------";
em.out.println("  w1: "+w1);
em.out.println("  w2: "+w2);
em.out.println("bias: "+b);
endl<<"case "<<i<<": "<<a;
 cout<<endl<<a;
<<endl<<"  w1: "<<w1,
<<endl<<"  w2: "<<w2,
<<endl<<"  b: "<<b,
em.out.println("error : "+error);
Error(a, data[i][2], data[i][0], data[i][1]);

 return error;
```



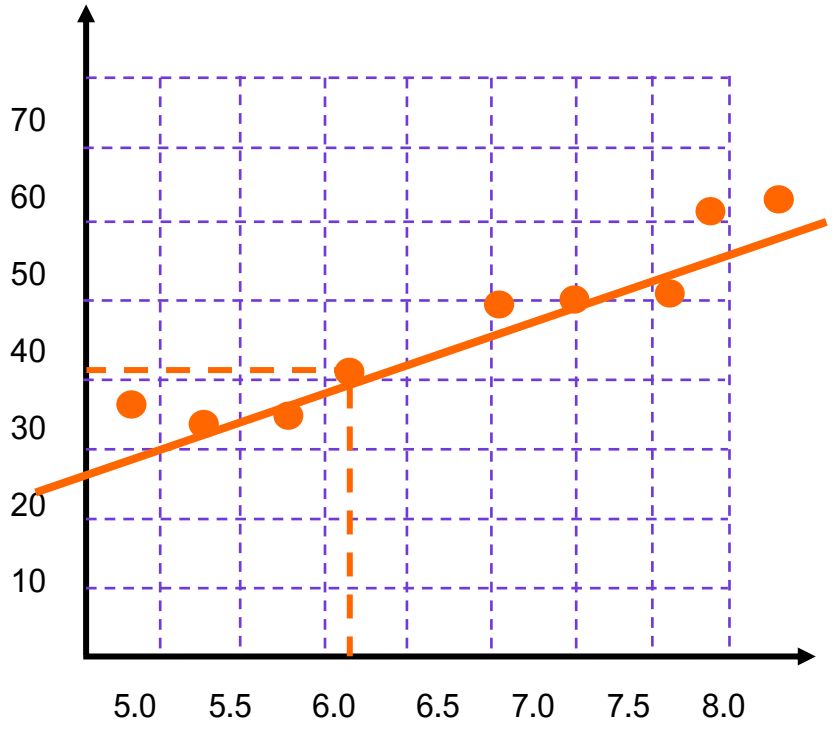
$$x_{n+1} = r x_n (1 - x_n)$$

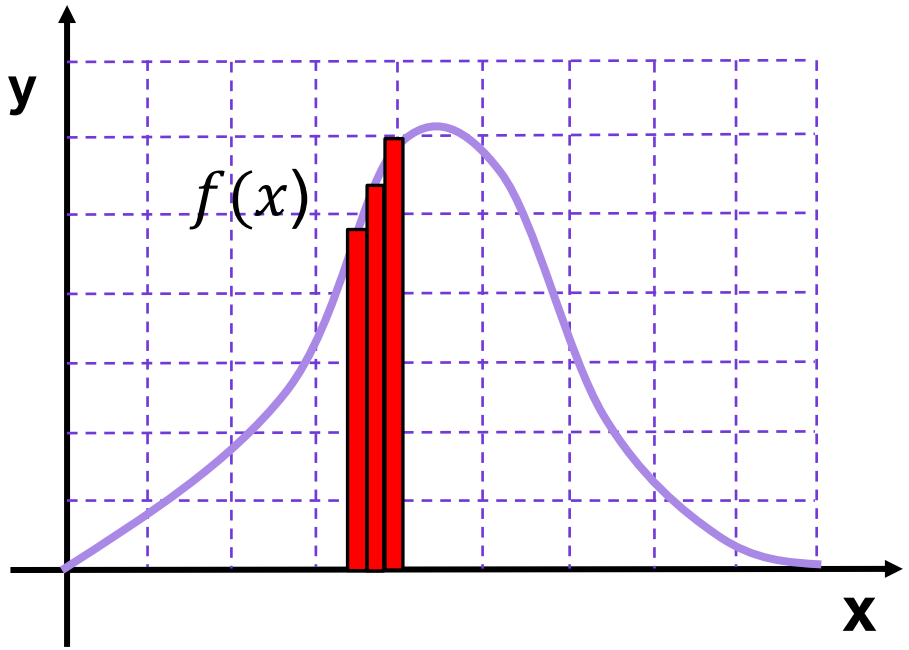
</ realidad >

- Caótico
- Indeterminado
- Complejo

</ modelo >

- Definido
- Intenta acercarse a la realidad
- Comprensible





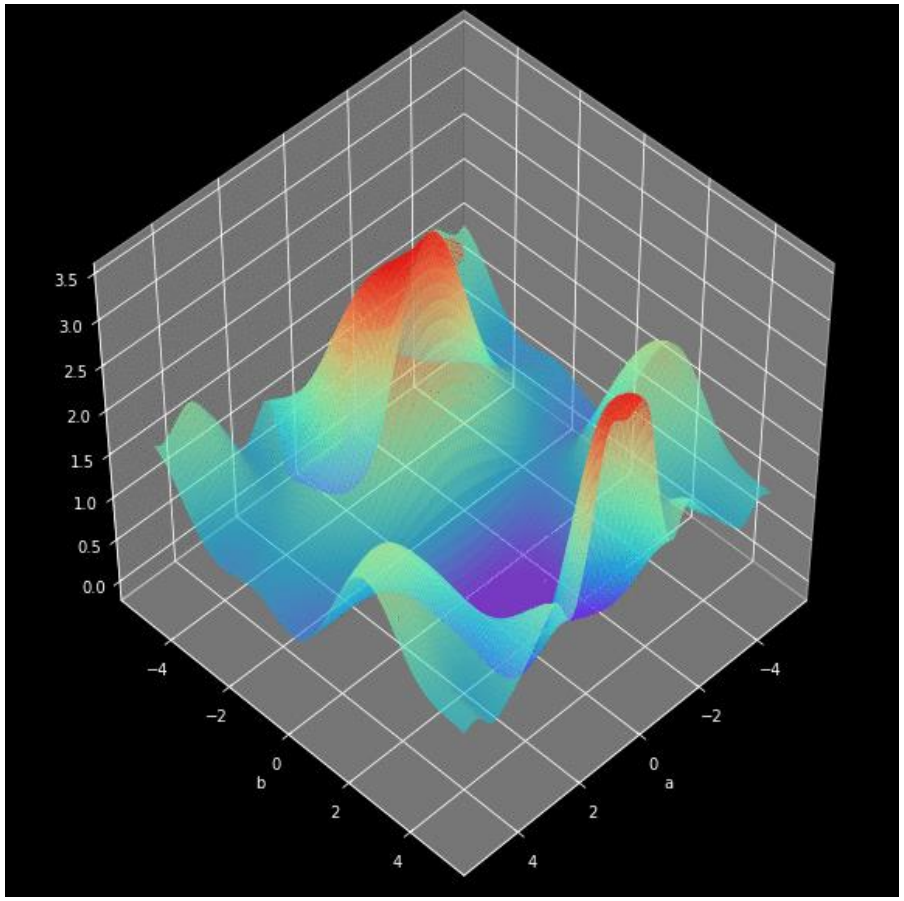
</ regresión >

</ probabilidad >

$$y = w_0 + w_1 x$$

Probabilidad de que un ave vuele

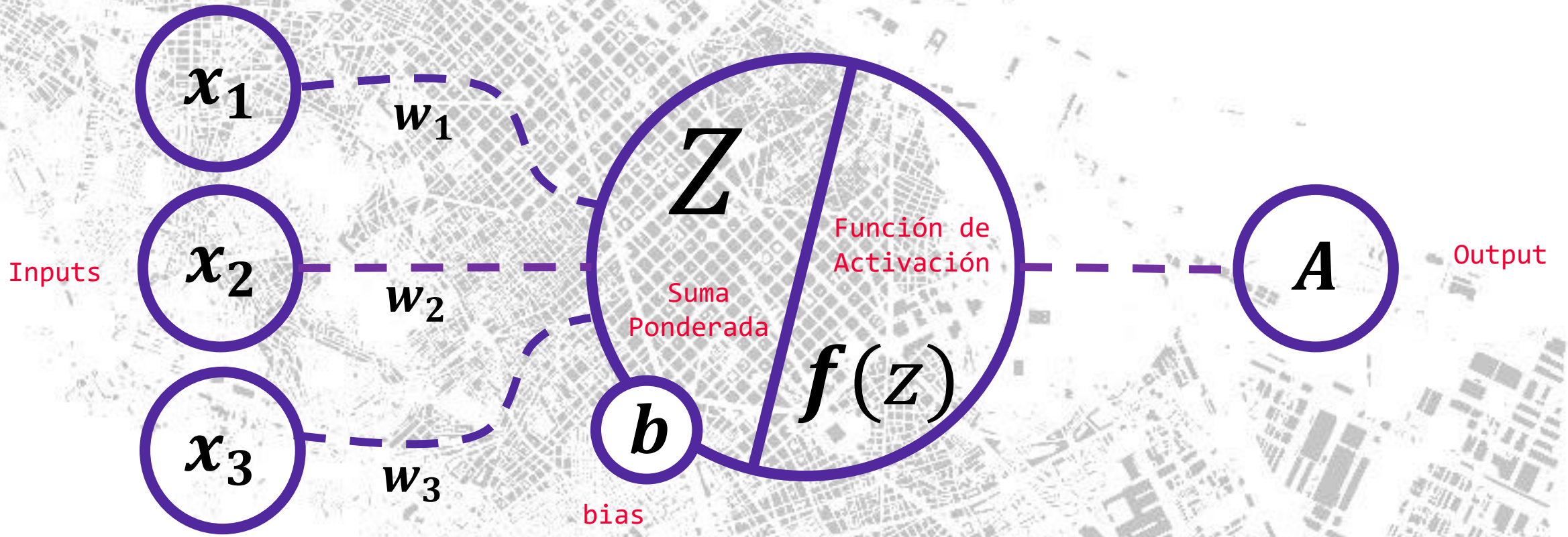
❖ generalización



</ regresión múltiple >

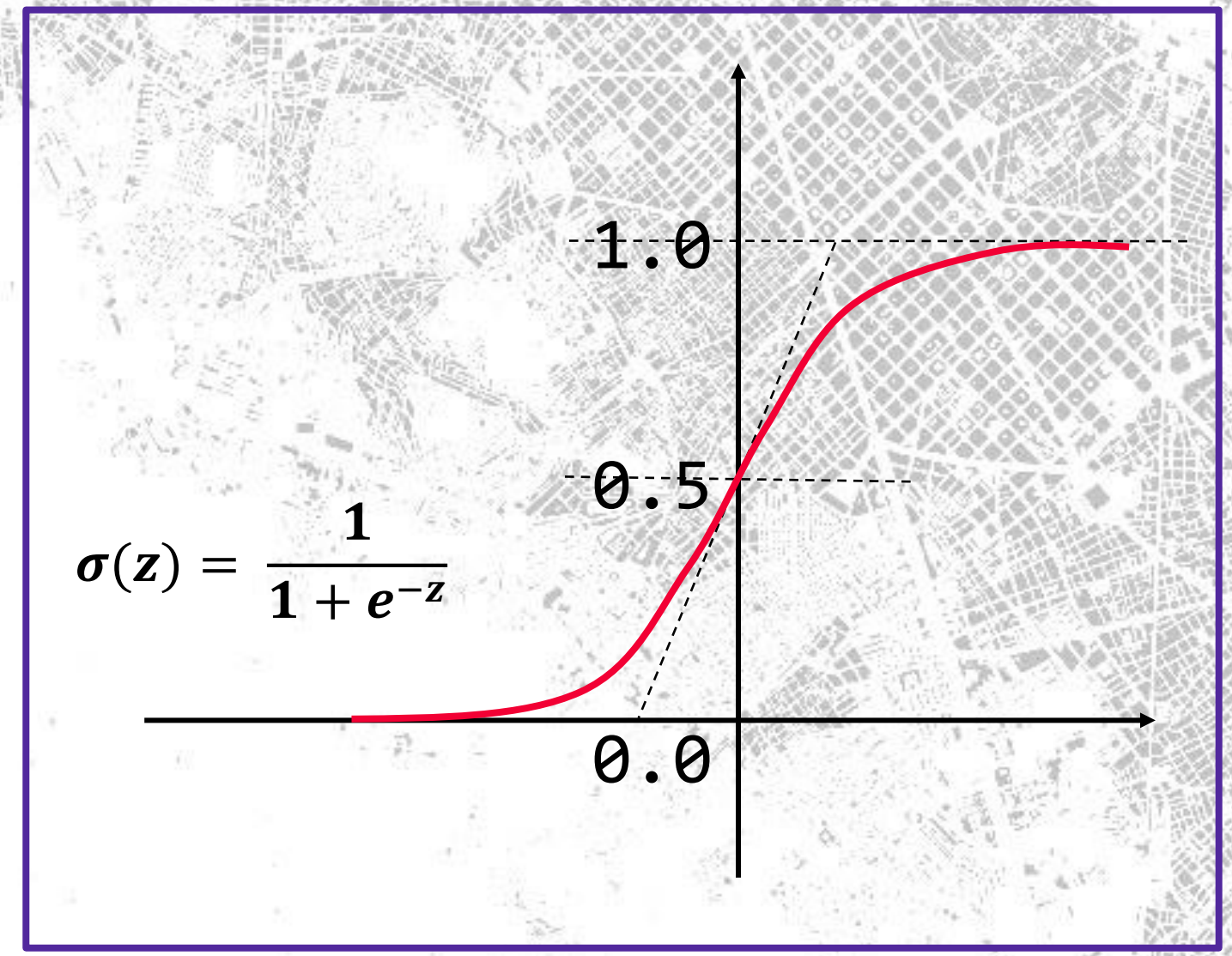
Red neuronal, un modelo matemático

$$\boldsymbol{\theta} = \{\boldsymbol{w_1}; \boldsymbol{w_2}; \ldots; \boldsymbol{w_n}; \boldsymbol{b_1}; \boldsymbol{b_2}; \ldots; \boldsymbol{b_n}\}$$

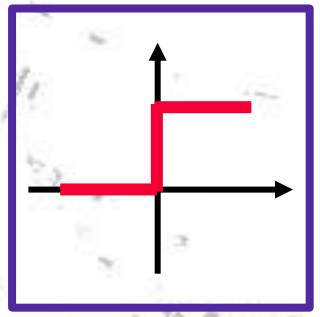


$$z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

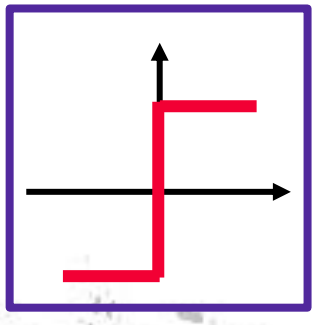
$$A = f(z)$$



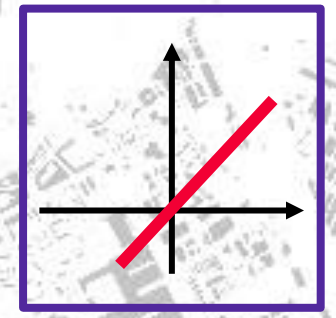
Sigmoide



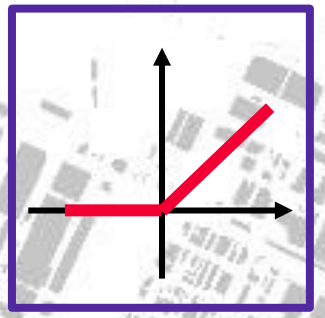
Escalón unitario



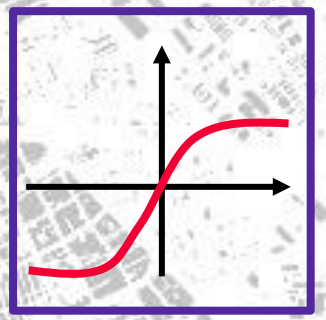
signo



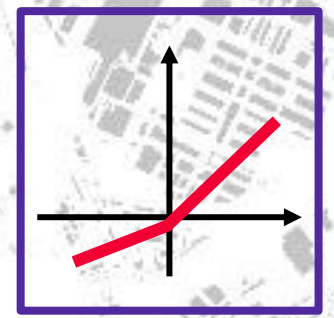
lineal



ReLu

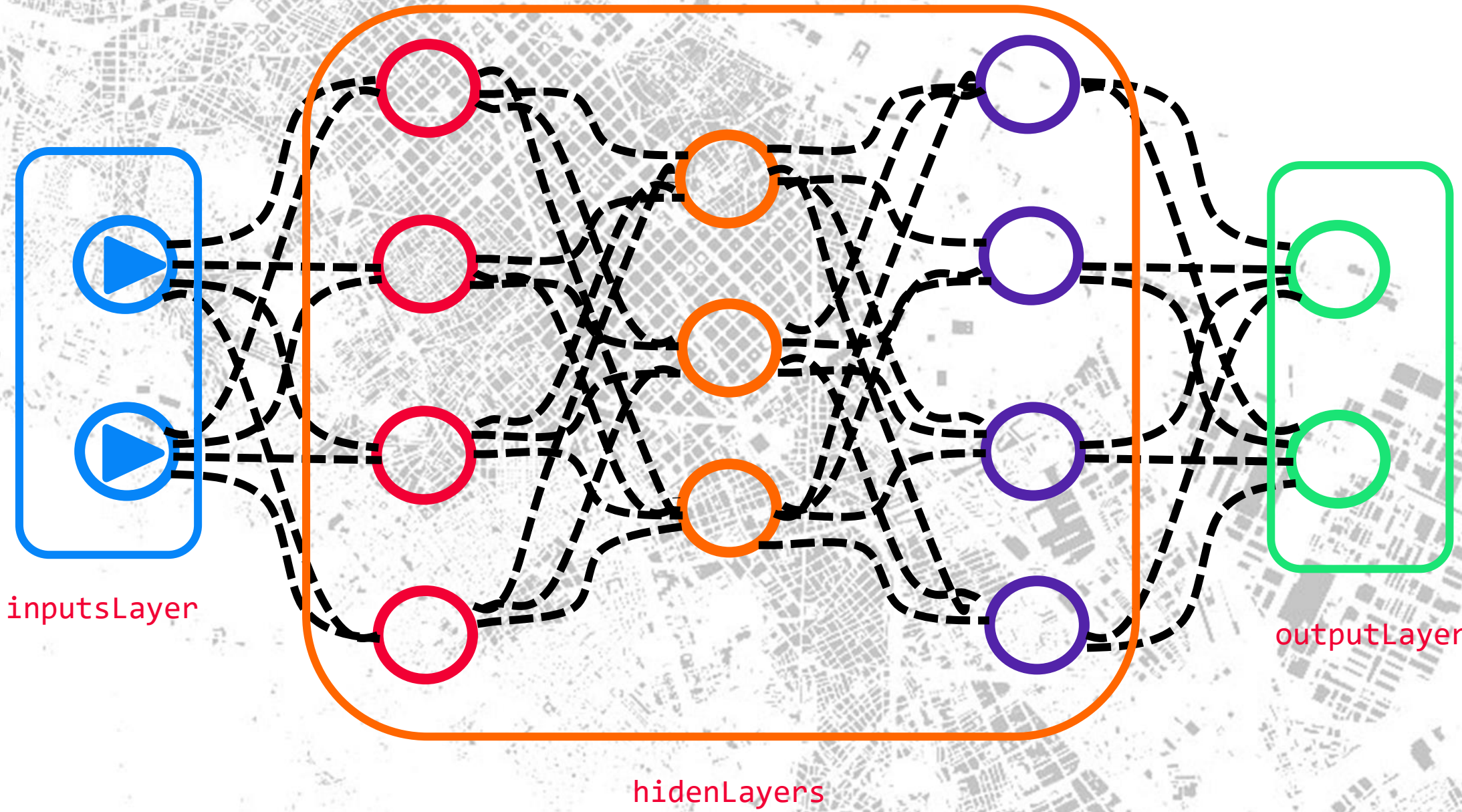


Tangente Hyperbolica

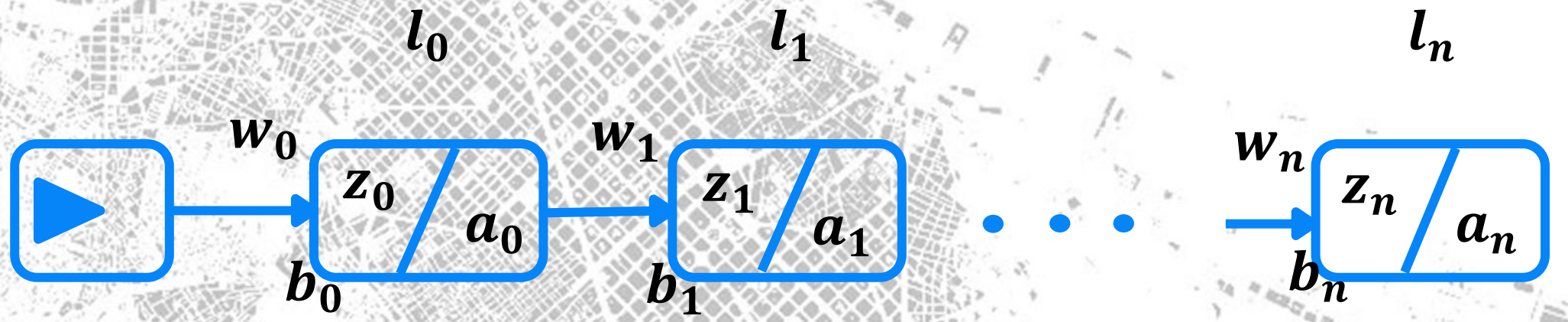


PReLu

</ función de activación >



</ red neuronal >

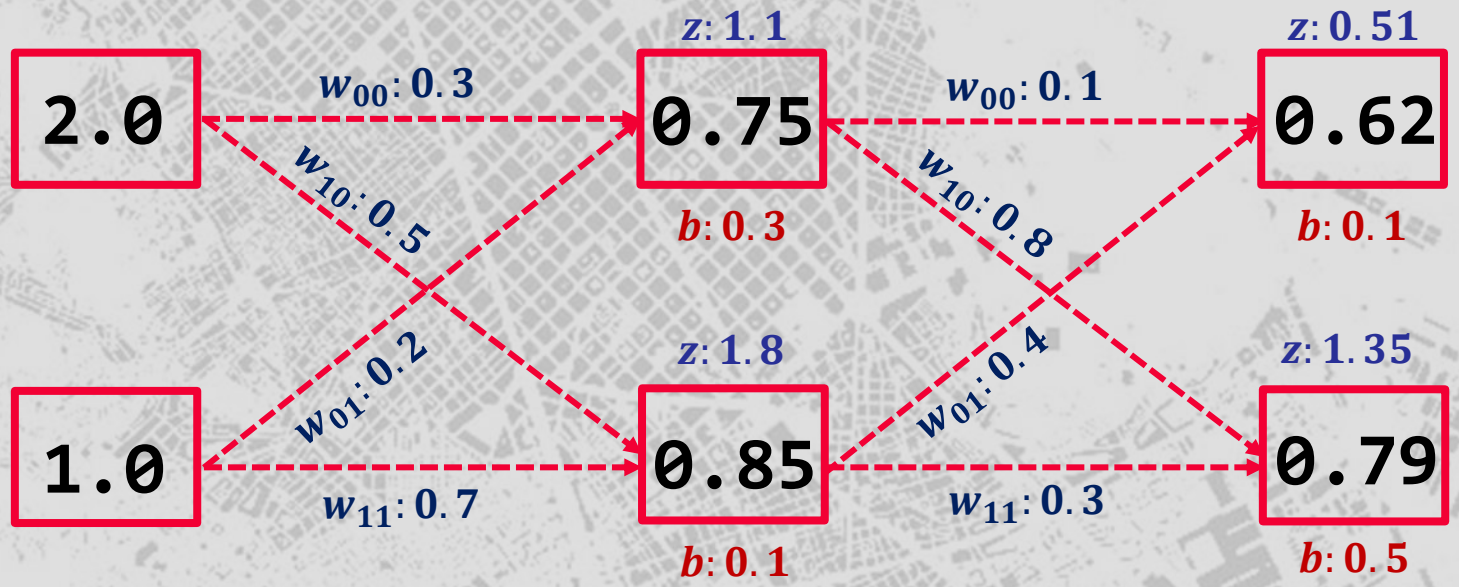


$$Z^l = \begin{bmatrix} z_0 \\ \vdots \\ z_n \end{bmatrix}^l = \begin{bmatrix} w_{00} & \cdots & w_{0m} \\ \vdots & \ddots & \vdots \\ w_{n0} & \cdots & w_{nm} \end{bmatrix}^l \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix}^{l-1} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}^l$$

$$Z^l = W^l A^{l-1} + b^l$$

$$A^l = \begin{bmatrix} A_0 \\ \vdots \\ A_n \end{bmatrix}^l = \sigma\left(\begin{bmatrix} z_0 \\ \vdots \\ z_n \end{bmatrix}^l \right)$$

$$A^l = \sigma(Z^l)$$



$$Z^1 = W^1 A^0 + b^1$$

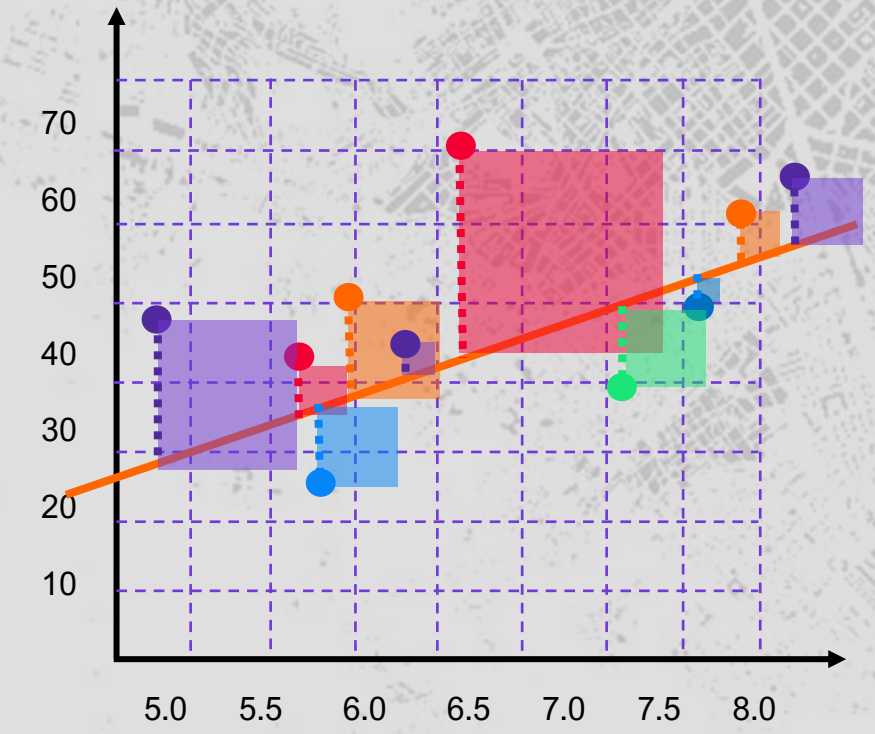
$$A^l = \sigma(Z^l)$$

$$Z^1 = \begin{bmatrix} 0.1 & 0.4 \\ 0.8 & 0.3 \end{bmatrix} \begin{bmatrix} 0.75 \\ 0.85 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.51 \\ 1.35 \end{bmatrix}$$

$$A^1 = \begin{bmatrix} \sigma(0.51) \\ \sigma(1.35) \end{bmatrix} = \begin{bmatrix} 0.62 \\ 0.79 \end{bmatrix}$$

Error cuadrático medio

</ mean square error >



$$\theta = \{w_1; w_2; \ldots; w_n; b_1; b_2; \ldots; b_n\}$$

$$\hat{y} = f(\theta)$$

$$\varepsilon = (y_i - \hat{y}_i)^2$$

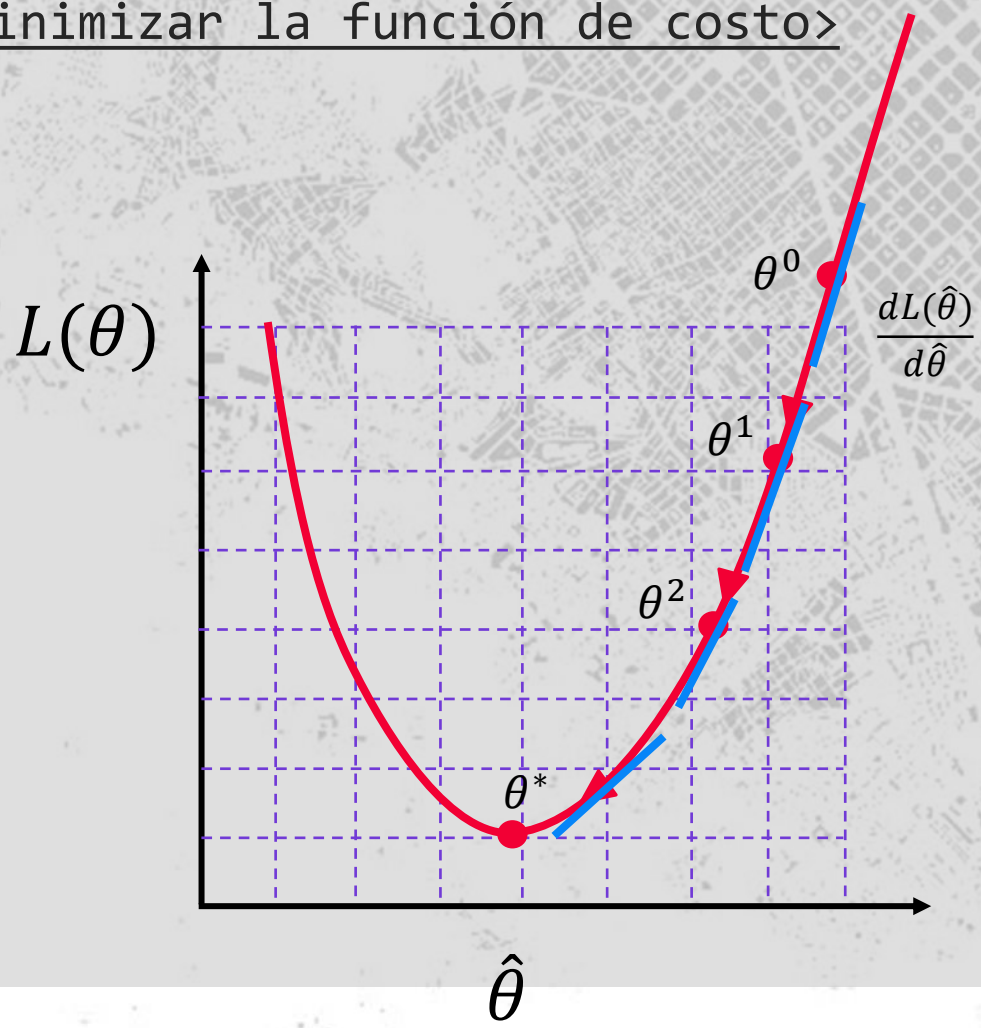
$$MSE(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

loss function

</ función de costo >

Descenso de Gradiente

</ minimizar la función de costo>



$$\theta = \{w_1; w_2; \ldots; w_n; b_1; b_2; \ldots; b_n\}$$

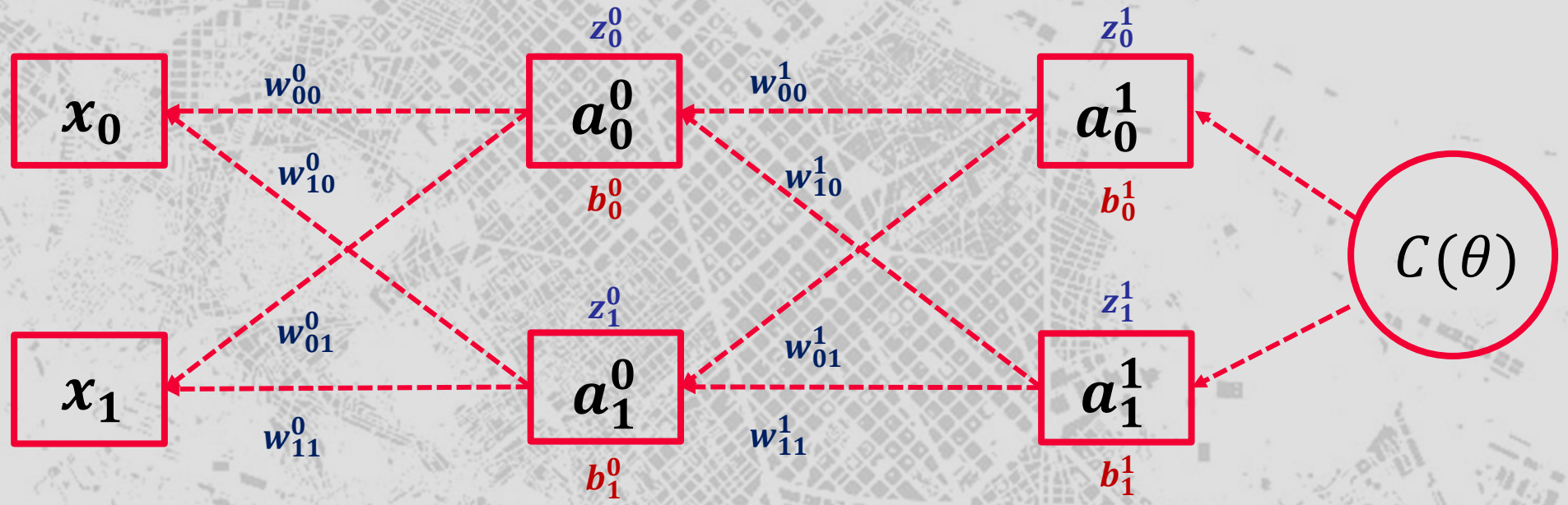
$$\boldsymbol{L(\theta) = \sum_{i=1}^{n} C^n(\theta) \quad \Rightarrow \quad \frac{dL(\theta)}{dw} = \sum_{i=1}^{n} \frac{dC^n(\theta)}{dw}}$$

$$\boldsymbol{\nabla L(\theta) = \begin{bmatrix} \frac{dC(\theta_0)}{dw_0} \\ \frac{dC(\theta_1)}{dw_1} \\ \vdots \\ \frac{dC(\theta_0)}{db_0} \\ \frac{dC(\theta_1)}{db_1} \\ \vdots \end{bmatrix}}$$

$$\boldsymbol{\theta_0 = \theta_0 - n\nabla L(\theta_0)}$$
$$\boldsymbol{\theta_1 = \theta_1 - n\nabla L(\theta_1)}$$
$$\vdots$$

Update parameters

</ gradient descent>



$$\frac{dC}{dw^{l-1}} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{da^{l-1}}\frac{da^{l-1}}{dz^{l-1}}\frac{dz^{l-1}}{dw^{l-1}}$$

$$\frac{dC}{db^{l-1}} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{da^{l-1}}\frac{da^{l-1}}{dz^{l-1}}\frac{dz^{l-1}}{db^{l-1}}$$

$$\delta^{l-1} = \delta^l (W^l)^T \frac{da^{l-1}}{dz^{l-1}}$$

$$\frac{dC}{dw^{l-1}} = \delta^{l-1} a^{l-1} \qquad \frac{dC}{db^{l-1}} = \delta^{l-1}$$

$$\frac{dC}{dw^l} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{dw^l}$$

$$\frac{dC}{db^l} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{db^l}$$

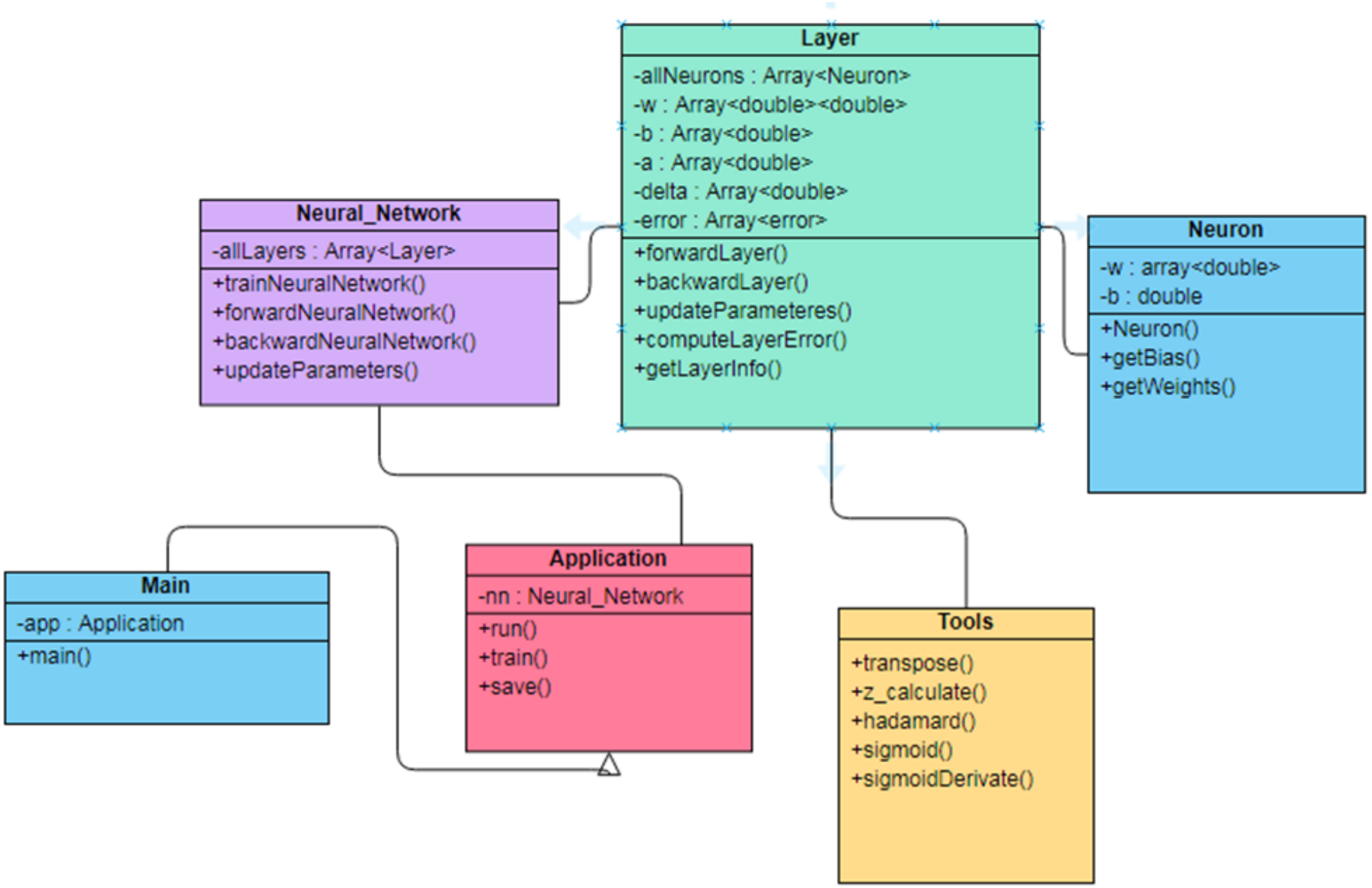
$$\delta^l = \frac{dC}{da^l}\frac{da^l}{dz^l}$$

$$w^n = \widehat{w^n} - n\frac{dC}{dw^n}$$

$$\Delta w^n = -n\frac{dC}{dw^n}$$

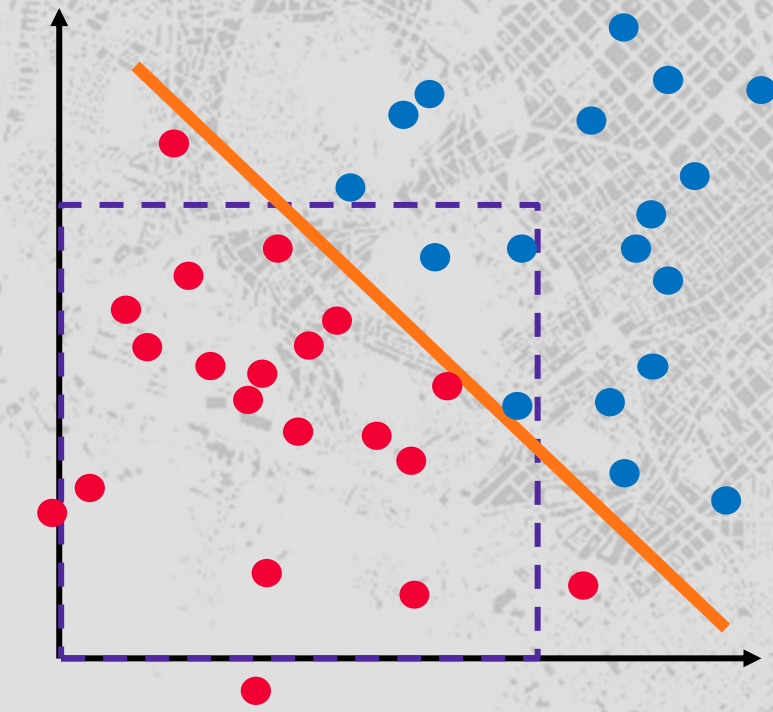
$$\Delta\theta_n = -n\frac{dC}{d\theta_n}$$

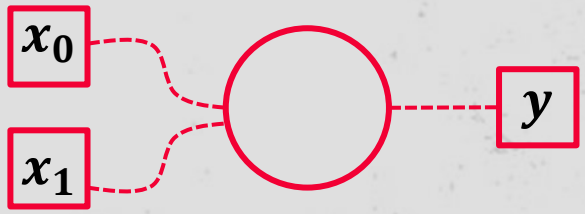
n : *learning rate*


Layer
-allNeurons : Array<Neuron>
-w : Array<double><double>
-b : Array<double>
-a : Array<double>
-delta : Array<double>
-error : Array<error>
+forwardLayer()
+backwardLayer()
+updateParameteres()
+computeLayerError()
+getLayerInfo()
Neural_Network
-allLayers : Array<Layer>
+trainNeuralNetwork()
+forwardNeuralNetwork()
+backwardNeuralNetwork()
+updateParameters()
Neuron
-w : array<double>
-b : double
+Neuron()
+getBias()
+getWeights()
Main
-app : Application
+main()
Application
-nn : Neural_Network
+run()
+train()
+save()
Tools
+transpose()
+z_calculate()
+hadamard()
+sigmoid()
+sigmoidDerivate()

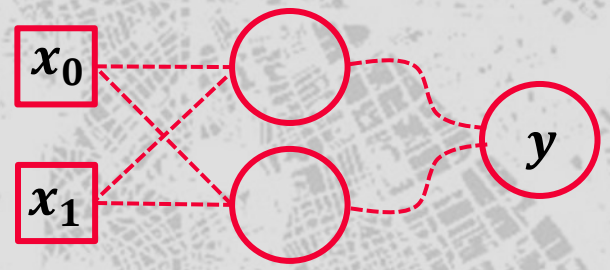
</ diseño >

</ and >





</ xor >

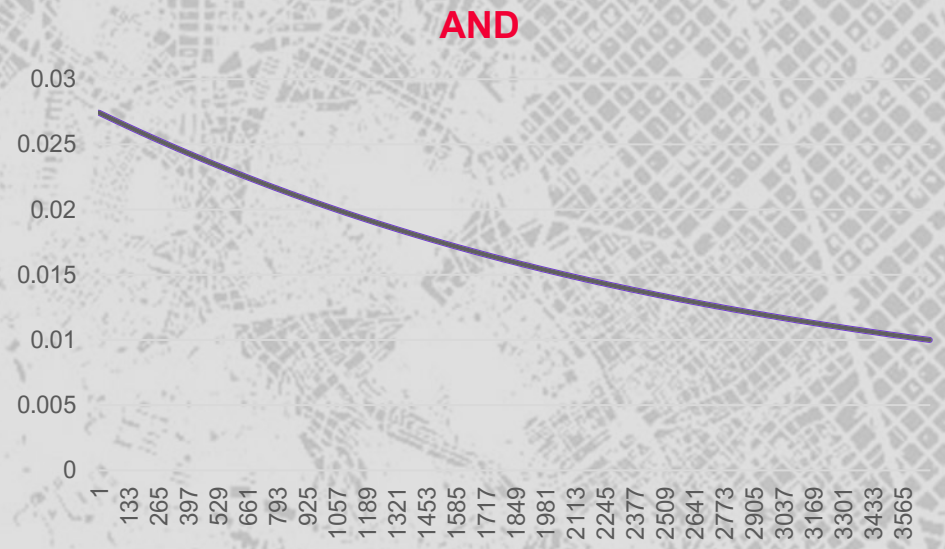


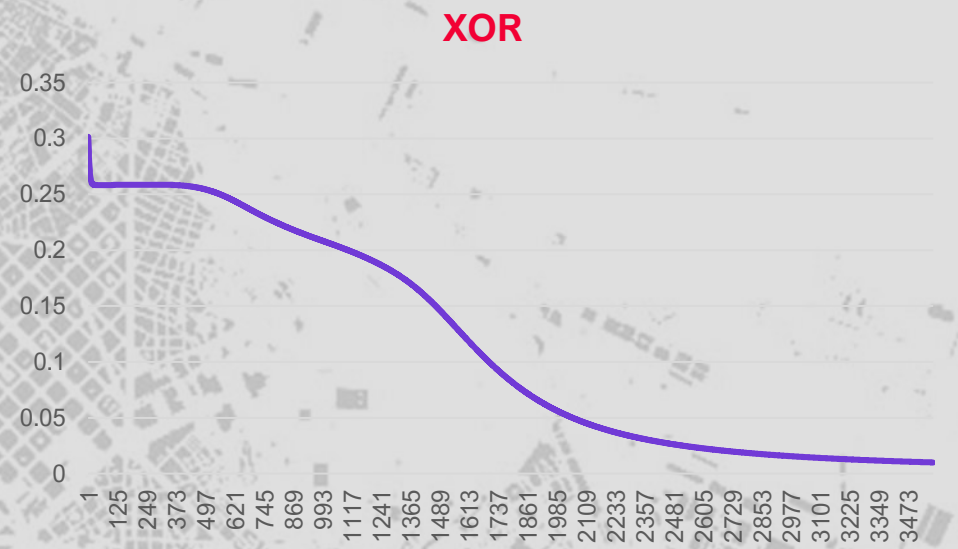
</ test >

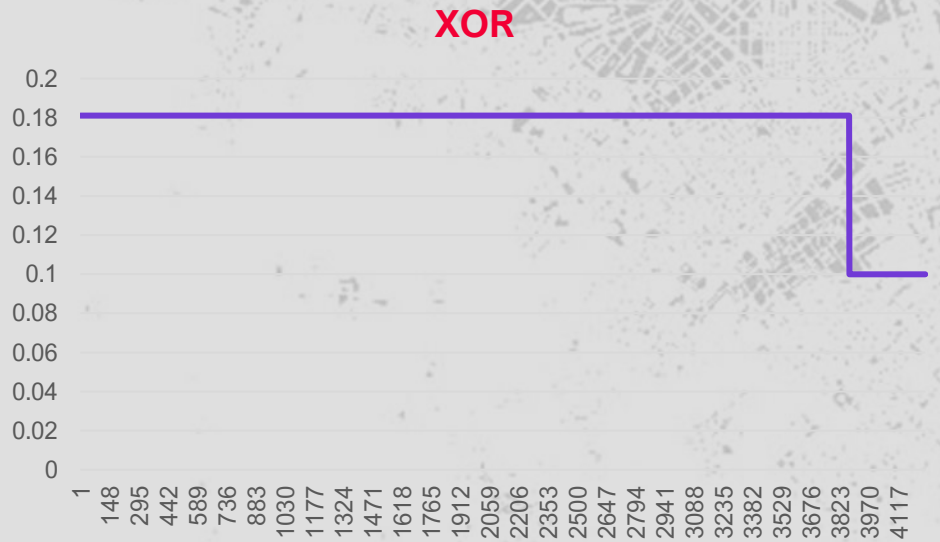
C

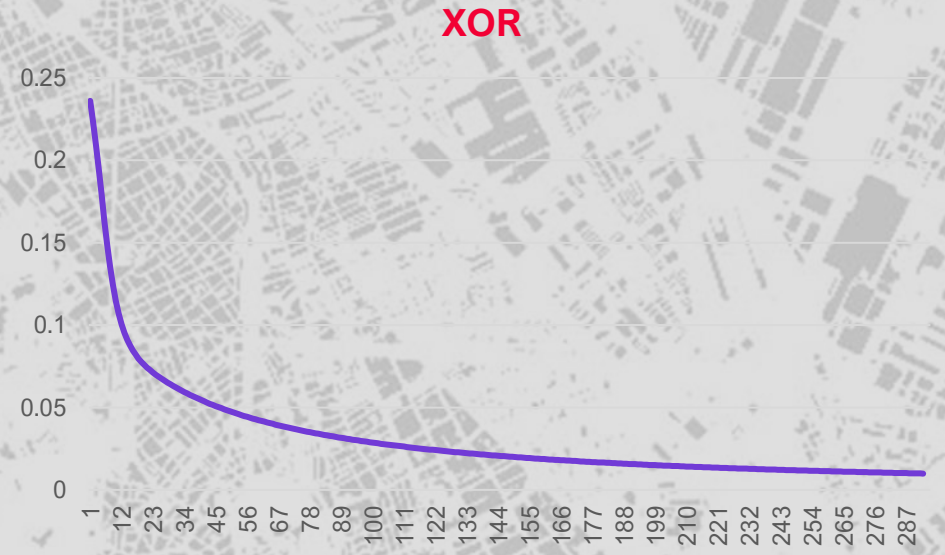
RESULTADOS

- RESULTADOS
- CONCLUSIONES


AND
0.03
0.025
0.02
0.015
0.01
0.005
0
1
133
265
397
529
661
793
925
1057
1189
1321
1453
1585
1717
1849
1981
2113
2245
2377
2509
2641
2773
2905
3037
3169
3301
3433
3565

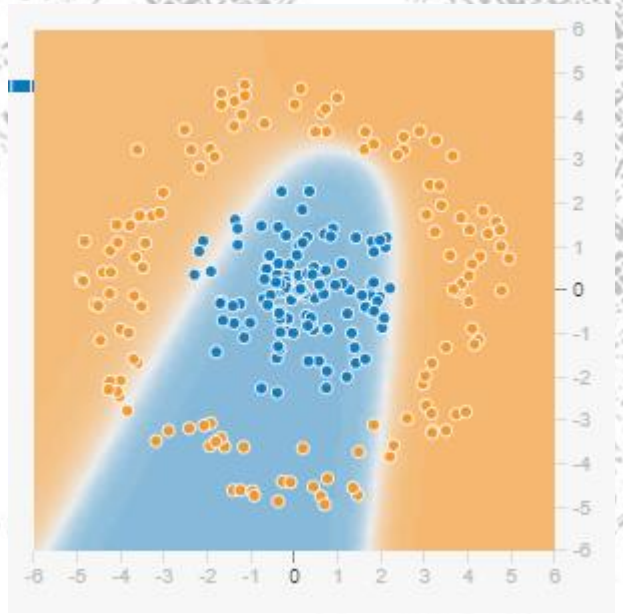

XOR
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
125
249
373
497
621
745
869
993
1117
1241
1365
1489
1613
1737
1861
1985
2109
2233
2357
2481
2605
2729
2853
2977
3101
3225
3349
3473


XOR
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1
148
295
442
589
736
883
1030
1177
1324
1471
1618
1765
1912
2059
2206
2353
2500
2647
2794
2941
3088
3235
3382
3529
3676
3823
3970
4117

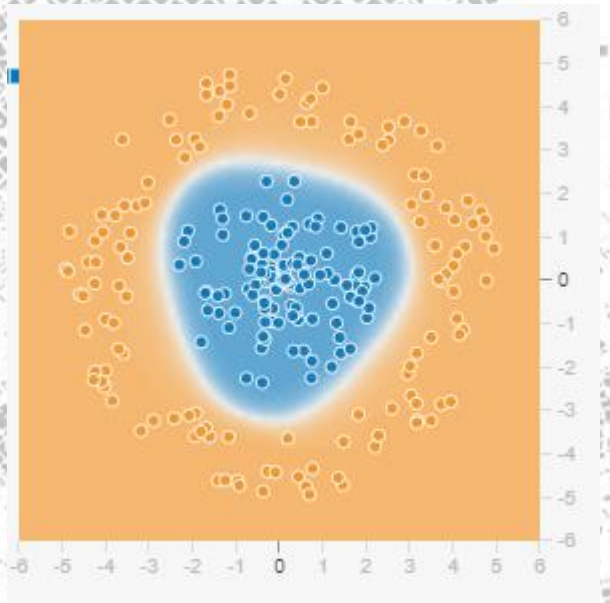

XOR
0.25
0.2
0.15
0.1
0.05
0
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188
199
210
221
232
243
254
265
276
287

</ resultados >

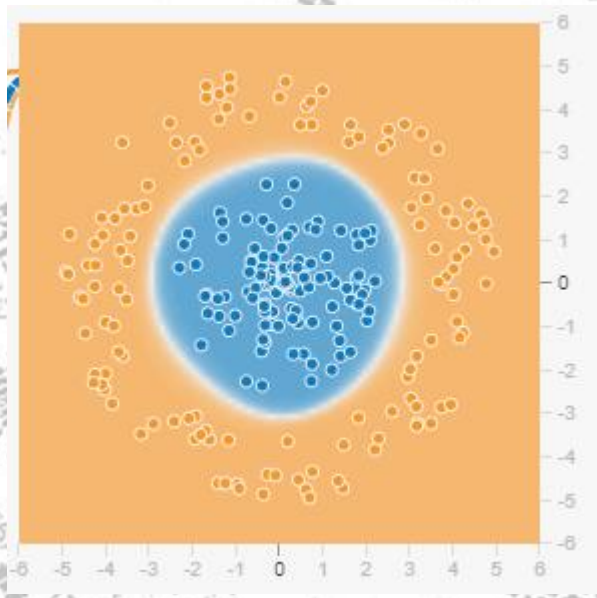
</ conclusiones>



<2 nodos>



<3 nodos-
2 capas>



<3 nodos-
6 capas>