Perceptrón simple y multicapa

72.27 - Sistemas de Inteligencia Artificial

Grupo 7

Luque Meijide, Manuel - 57386

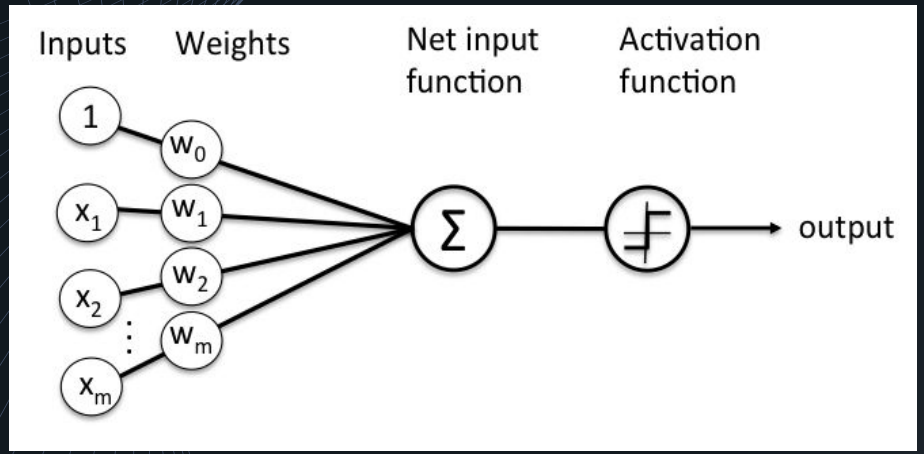
Karpovich, Lucía - 58131

Tarradellas del Campo, Manuel - 58091

1.

Perceptrón simple con función de activación escalón

Perceptrón simple



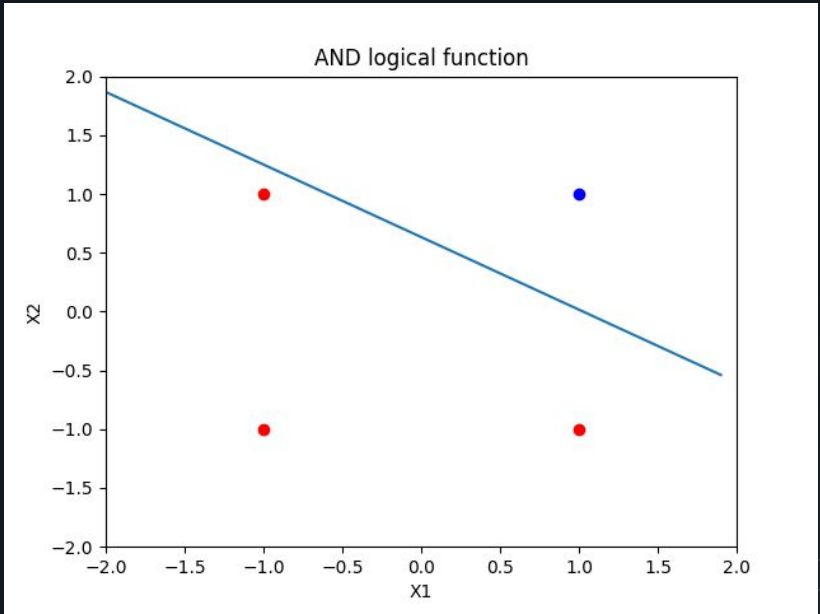
- Calcula la suma ponderada del input
- Transforma la salida en 0 ó 1

Geométricamente, esto quiere decir que el perceptrón puede separar el espacio del conjunto de entrada a través de un hiperplano.

¿Todas las funciones lógicas son separables?

a. Función lógica AND

Output



Entrada: { {−1, 1}, {1, −1}, {−1, −1}, {1, 1} }

Salida esperada: y = {−1, −1, −1, 1}

Learning rate: 0.2

a. Función lógica AND

Output

```
LOGICAL AND
Initial Perceptron:
{
 layers: [
  {
   weights: [[-0.59793564  0.02732348 -0.92731049]]
   inputs: None
   activations: None
   errors: None
  }

 ]
 rate: 0.2
}
Initial predictions:
-1  1 : -1.0
 1 -1 : 1.0
-1 -1 : 1.0
 1  1 : -1.0
```

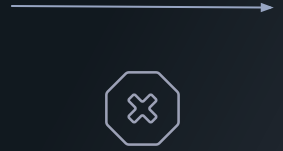


```
Final Perceptron:
{
 layers: [
  {
   weights: [[-0.19793564  0.42732348  0.27268951]]
   inputs: [1. 1. 1.]
   activations: 1.0
   errors: [0.]
  }

 ]
 rate: 0.2
}
Final predictions:
-1  1 : -1.0
 1 -1 : -1.0
-1 -1 : -1.0
 1  1 : 1.0
```

b. Función lógica XOR

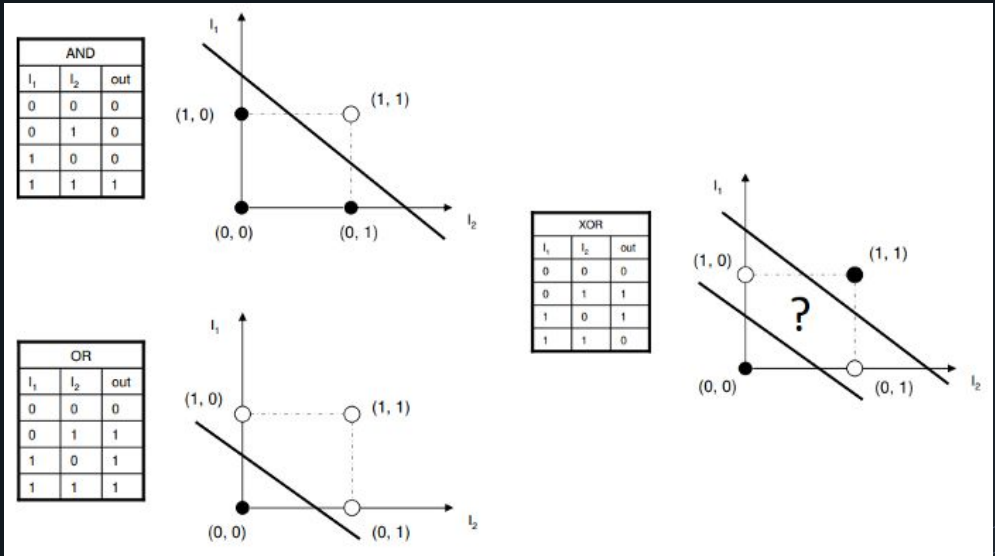
Output

```
EXERCISE 1 B:

LOGICAL XOR:
Initial Perceptron:
{
 layers: [
  {
   weights: [[-0.5141077  -0.69685676  0.38733073]]
   inputs: None
   activations: None
   errors: None
  }

 ]
 rate: 0.2
}
Initial predictions:
-1  1 : 1.0
 1 -1 : -1.0
-1 -1 : -1.0
 1  1 : -1.0
```



```
Final predictions:
Perceptron:
{
 layers: [
  {
   weights: [[-0.5141077   0.10314324 -0.41266927]]
   inputs: [1. 1. 1.]
   activations: 1.0
   errors: [-2.]
  }

 ]
 rate: 0.2
}
-1  1 : -1.0
 1 -1 : 1.0
-1 -1 : -1.0
 1  1 : -1.0
```

AND, OR y XOR

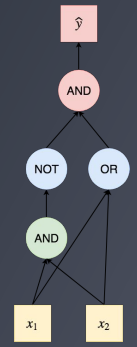


¿Cómo se puede modelar XOR?

2 opciones

Aprendizaje perceptrón simple

Perceptrón multicapa



2.
Perceptrón simple lineal y no lineal

Resultados

Perceptrón simple lineal

Training Error	Testing Error
0.1217	**0.1305**

Learning rate: 0.01
Iteraciones (entrenamiento) = 1000

Perceptrón simple no lineal

Training Error	Testing Error
0.1820	**0.2141**

Learning rate: 0.1
Función (no lineal): tanh(x)
Iteraciones (entrenamiento) = 10000

Training 150 - Testing 50

Problemas

Los gráficos anteriores nos brindan una forma de medición para los distintos algoritmos, sin embargo no dan respuestas a las siguientes cuestiones:

- ¿Cómo podemos medir la eficiencia de la red ante otro conjunto de datos distinto del de entrenamiento?

- ¿Cómo podemos elegir el mejor conjunto de datos para el entrenamiento?

Cross Validation

Se implementó Cross Validation para abordar los anteriores problemas.

- Se separa el conjunto de datos en k grupos.
- Se toman k-1 grupos como datos de entrenamiento.
- El grupo restante se utilizará para testear la red.

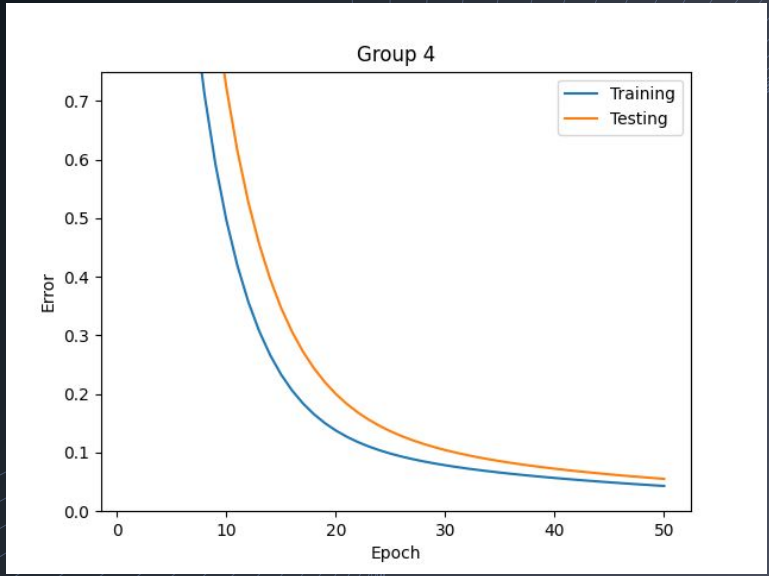
Cross Validation

¿Cómo podemos escoger el mejor conjunto de entrenamiento? → *Tomamos el conjunto que menor error produce en el testing.*

- k = 4
- Epocas = 50

Output

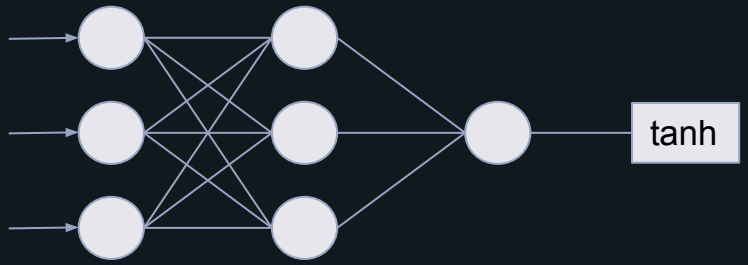
```
~~ Cross validation:

Group # 1  Epoch # 50  --> error = 0.10138165164863702
Group # 2  Epoch # 50  --> error = 0.05516562928336181
Group # 3  Epoch # 50  --> error = 0.2358440761037263
Group # 4  Epoch # 50  --> error = 0.03557168843697372
Lowest error found in: Group: 4, Epoch: 50
```



3.
Perceptrón multicapa

a. Función lógica XOR

Esquema



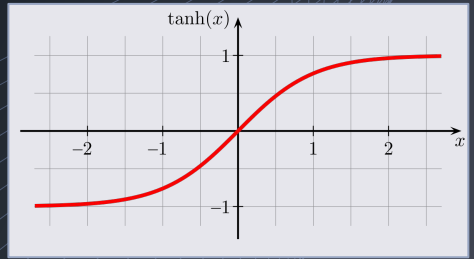
Output

```
 1  1 : [-0.99706026]
-1 -1 : [-0.99641481]
-1  1 : [0.99901092]
 1 -1 : [0.99700656]
```

Entrada: { {−1, 1}, {1, −1}, {−1, −1}, {1, 1} }

Salida esperada: y = {1, 1, −1,- 1}

Learning rate: 0.1
Función: tanh(x)
N° de capas: 3
Iteraciones (entrenamiento) = 100.000



a. Función lógica XOR

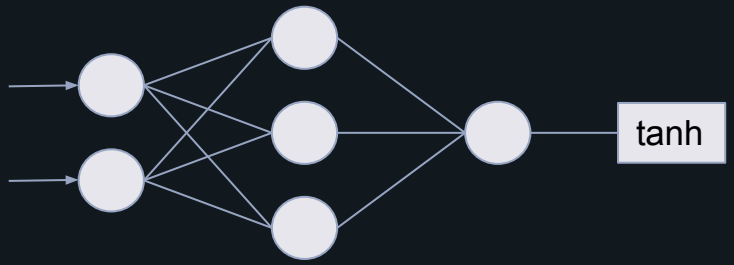
Variando algunos parámetros...

Tasa de aprendizaje	[-1, -1]	[-1, 1]	[1, -1]	[1, 1]
0.1	**-0.9928**	**0.9930**	**0.9954**	**-0.9942**
0.3	**-0.9993**	**0.9985**	**0.9985**	**-0.9986**
0.7	**-0.998**	**0.998**	**0.998**	**-0.998**

Iteraciones: 100.000
N° capas: 3

b. Discriminar n° pares

Esquema



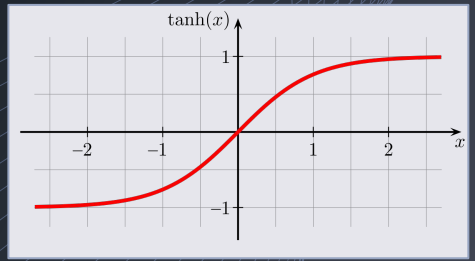
Output

```
0 : [-0.997418]
1 : [0.99824698]
2 : [-0.99745099]
3 : [0.99794024]
4 : [-0.99761232]
5 : [0.99838914]
6 : [-0.99755307]
7 : [0.99845554]
8 : [-0.99736247]
9 : [0.99836882]
```

Entrada: { [0 - 9] }

Salida esperada: y = {-1, 1, −1, 1, -1, 1, -1, 1, -1, 1}

Learning rate: 0.1
Función: tanh(x)
N° de capas: 3



Generalización

Teniendo en cuenta que:

- Los datos de entrada son la forma de los números
- Estas formas no guardan relación numérica con el próximo valor
- Es un data set muy pequeño

¿Tiene sentido hablar de generalización en este caso?

Análisis de generalización

Experimento: Se divide el set de datos en conjunto de entrenamiento y conjunto para testing.

Learning rate = 0.1

Entrenamiento	7	8	9
[0 - 7]	**0.997**	**-0.943**	**-0.984**
[0 - 8]	**0.997**	**-0.997**	**-0.988**

?

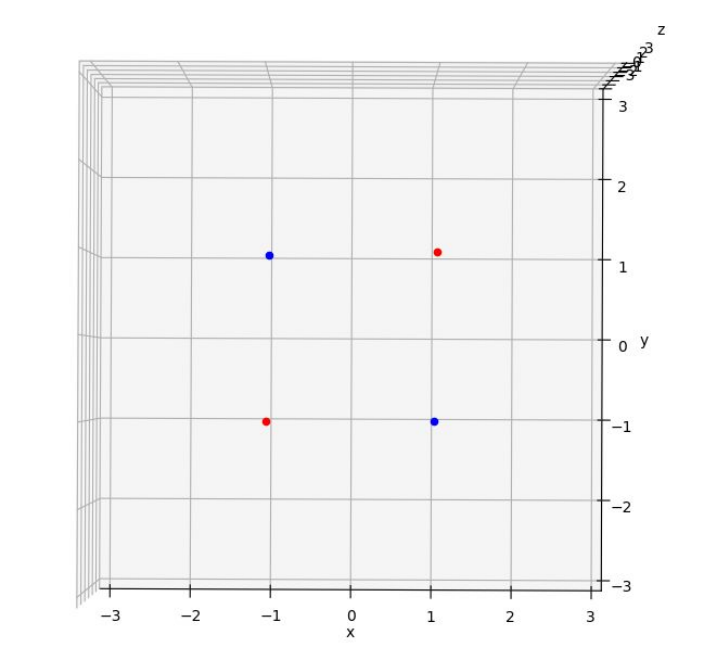
Learning rate = 0.4

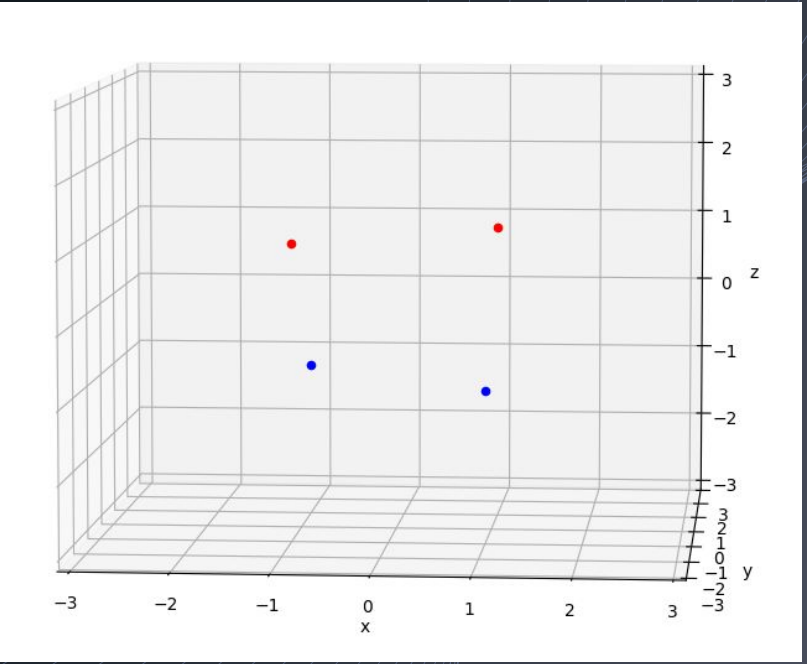
Entrenamiento	7	8	9
[0 - 7]	**0.998**	**-0.989**	**-0.997**
[0 - 8]	**0.998**	**-0.997**	**-0.99**

**Debido a la poca cantidad de datos, no se utilizaron métricas*

4.
Extensión Perceptrón simple

Extender dimensión del problema





Extender dimensión del problema