TP3

“Perceptron simple y multicapa”

Juan Pablo Oriana - 60621
Tomas Cerdeira - 60051
Santiago Garcia Montagner - 60352

EJ.1

PERCEPTRON SIMPLE ESCALONADO

DATOS DE ENTRADA

- Funcion logica **"Y" (AND)**
  - Input: [{ -1, 1 }, { 1, -1 }, { 1, 1 }, { -1, -1 }]
  - Expected: [-1, -1, 1, -1]

AND

A	B	Expected
-1	1	-1
1	-1	-1
1	1	1
-1	-1	-1

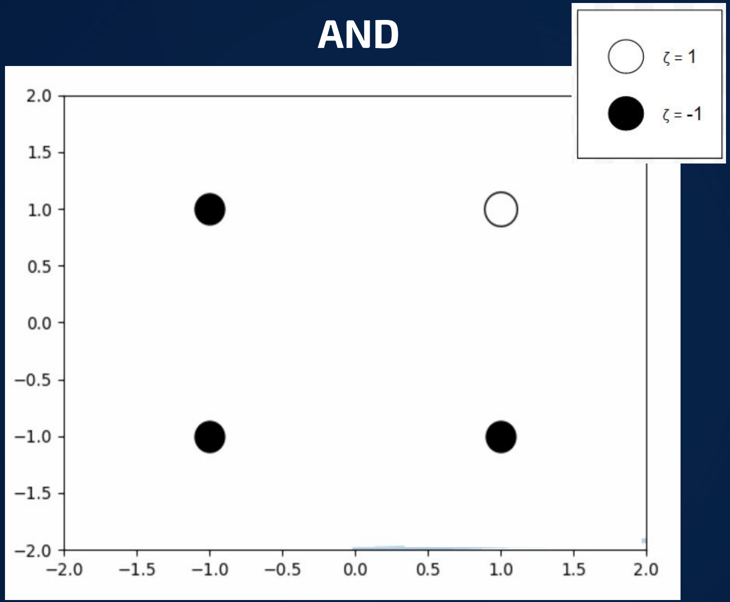
XOR

A	B	Expected
-1	1	1
1	-1	1
1	1	-1
-1	-1	-1

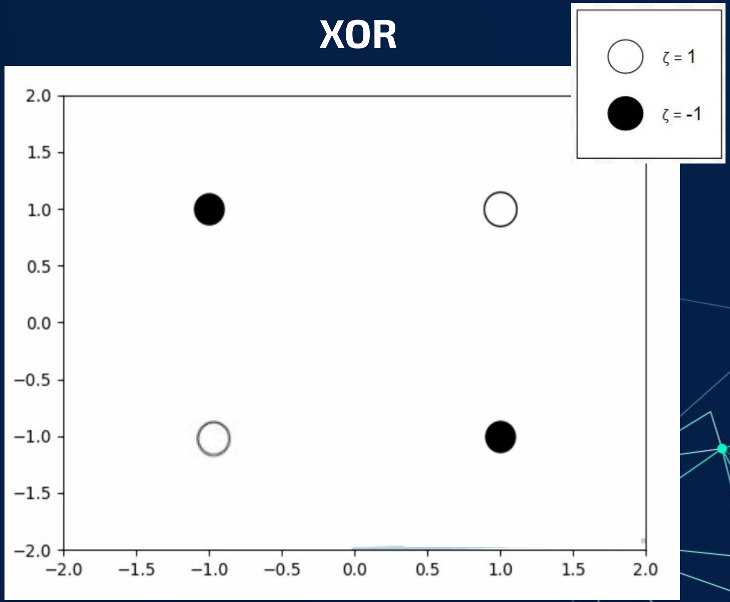
- Funcion logica **"O exclusivo" (XOR)**
  - Input: [{ -1, 1 }, { 1, -1 }, { 1, 1 }, { -1, -1 }]
  - Expected: [1, 1, -1, -1]

DATOS DE ENTRADA

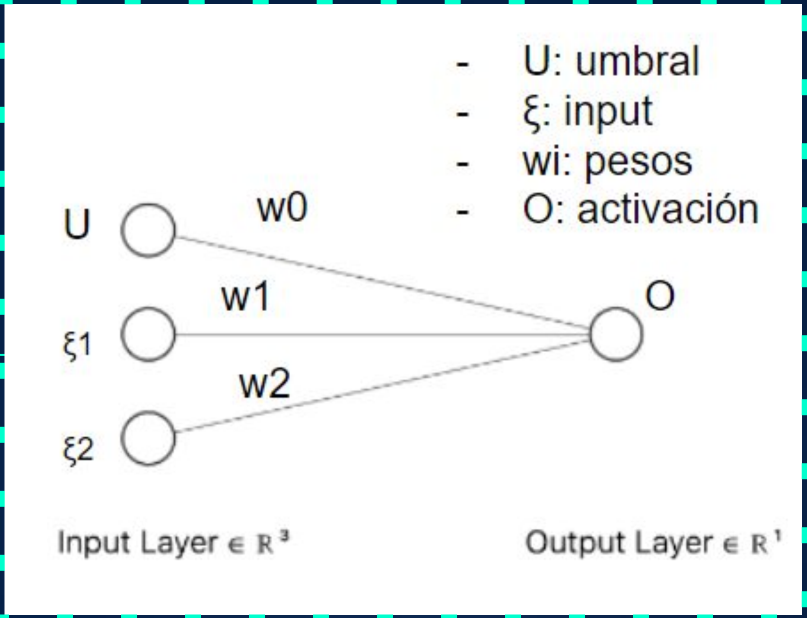
AND



XOR



PERCEPTRON SIMPLE

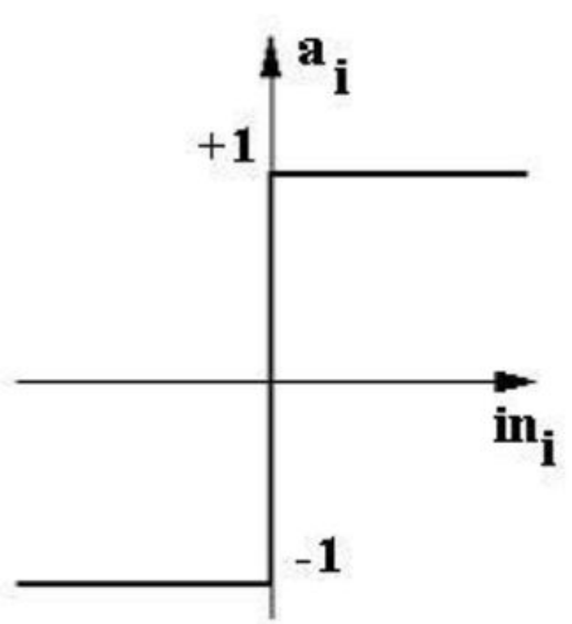


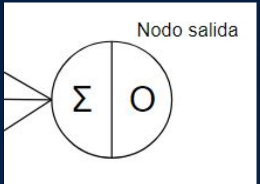
ESTRUCTURA

Trabajamos con **3 nodos en la capa de entrada**, 2 de input del **AND/XOR** y un tercero que representa **el bias.** Todos **se conectan mediante un peso a una salida activada** con una función escalonada

FUNCION DE ACTIVACION: ESCALON

$$\theta(x) = \begin{cases} 1 & x \geq 0 \\ -1 & \text{en otro caso} \end{cases}$$





$$O = \theta\left(\sum_{i=1}^{n} w_i \xi_i - umbral\right)$$

NOSOTROS QUEREMOS

$$O^{\mu} \text{ sea igual a } \zeta^{\mu}$$

DONDE ζ ES EL VALOR ESPERADO PARA ξ

OBS: esta función se usa en los problemas que **se busca separar entre 2 clases**

APRENDIZAJE DEL PERCEPTRÓN

$$w_i^{nuevo} = w_i^{viejo} + \Delta w_i$$

$$\Delta w_i = \eta(\zeta^\mu - O^\mu)\xi_i^\mu$$

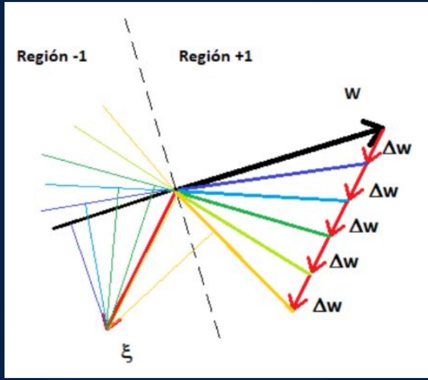
NOSOTROS QUEREMOS ENCONTRAR LOS W_i TALES QUE MINIMICEN

$$E(w) = \frac{1}{p}\sum_{\mu=1}^{p}(\zeta^\mu - O^\mu)^2$$

LA FUNCIÓN DE ERROR DEL PERCEPTRÓN

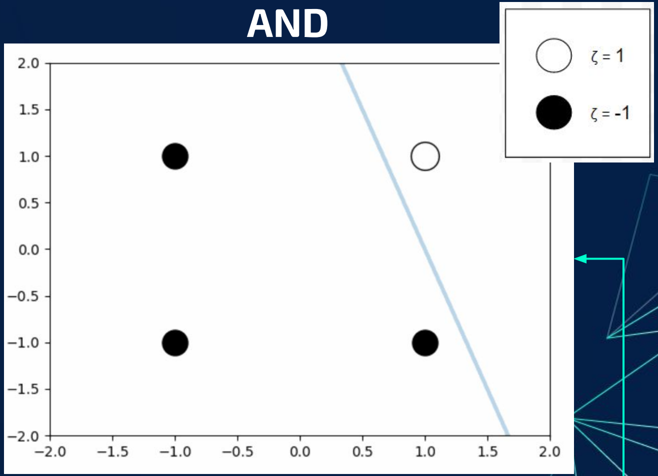
PESOS W

- El **vector W** resultante, luego de ser actualizado con el set de entrenamiento, es una **representación de la normal al hiperplano**, en R^2 a la recta, **que separa a los puntos**.
    - Logramos separar los puntos entre sus dos clases



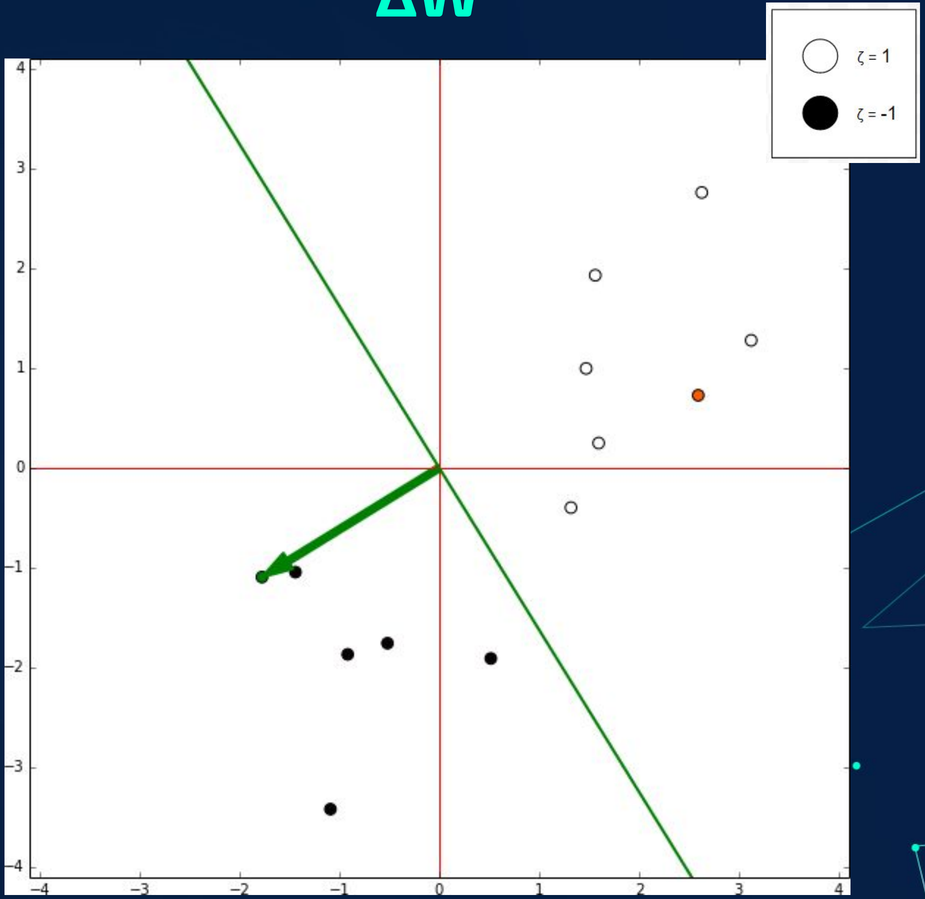

```
y = (w[0] * x + w[2]) / -w[1]
```

AND



ΔW

e.j.

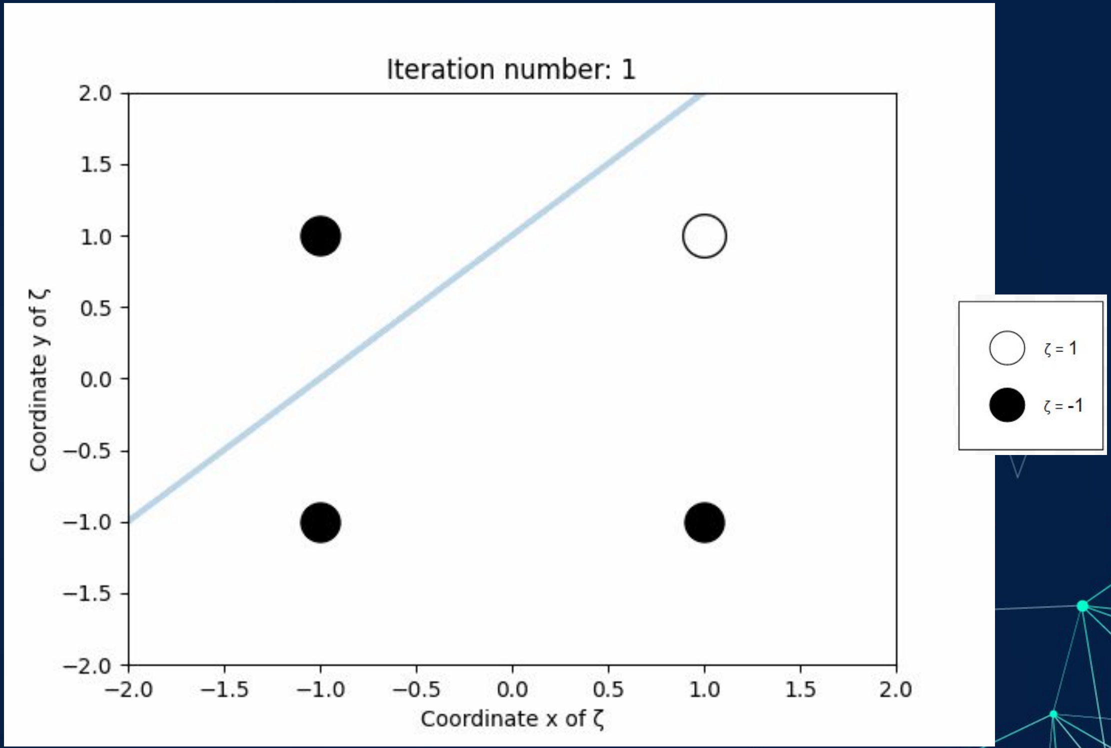


OBS: nuestro vector rojo está normalizado en base al factor de aprendizaje

AND

Θ: Funcion de activacion

$$\theta(x) = \begin{cases} 1 & x \geq 0 \\ -1 & \text{en otro caso} \end{cases}$$

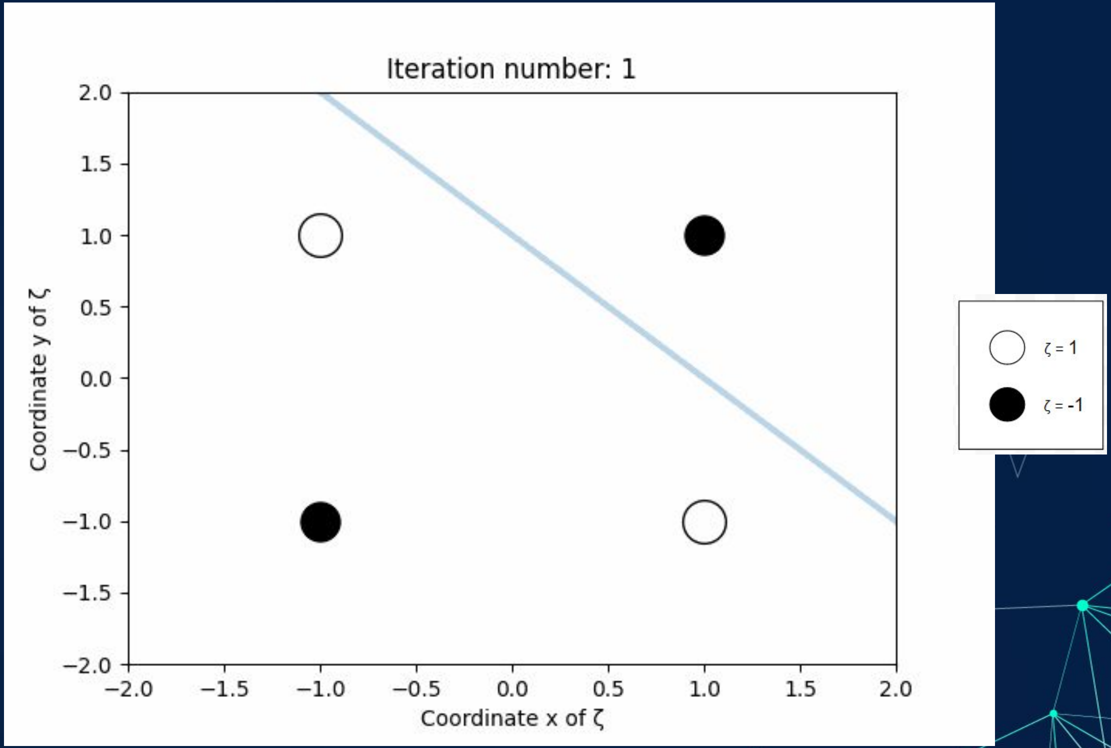


Logra llegar a un error mínimo = 0

XOR

Θ: Funcion de activacion

$$\theta(x) = \begin{cases} 1 & x \geq 0 \\ -1 & \text{en otro caso} \end{cases}$$



El error mínimo que se puede conseguir con esta configuración es 1

¿Qué puede decir acerca de los problemas que puede resolver el perceptrón simple escalón en relación a la resolución de los problemas que se le pidió que haga que el perceptrón aprenda?

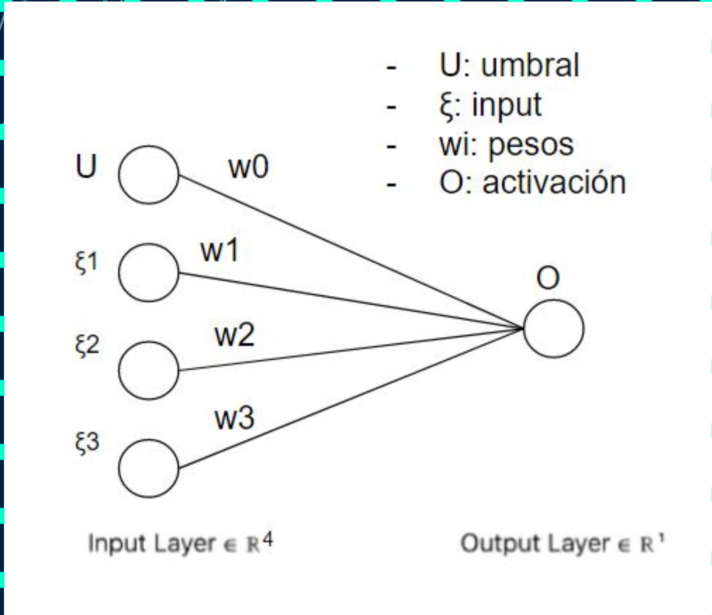
LINEALMENTE SEPARABLES

EJ.2

OTROS METODOS DE ACTIVACION

Evalue la *capacidad* **del** *perceptron simple lineal* **y** *perceptron simple no lineal* **para** *aprender* **la función cuyas muestras están presentes en los archivos indicados.**

PERCEPTRÓN SIMPLE



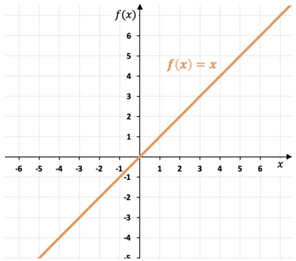
FUNCION DE ACTIVACION

$$O = \theta\left(\sum_{i=1}^{n} w_i \xi_i - umbral\right)$$

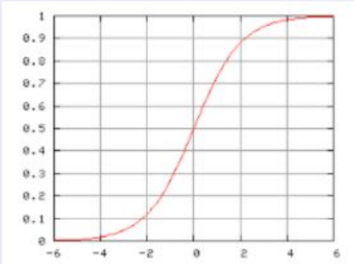
¿Como cambia el aprendizaje de un perceptrón según el tipo de función de activación usada?

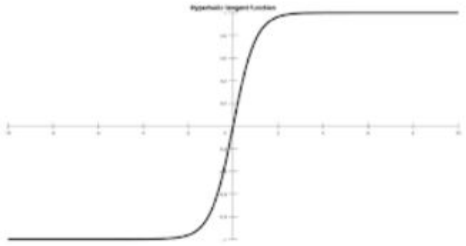
PERCEPTRÓN *LINEAL* **y** *NO LINEAL*

- Funciones de activación:
  - Lineales
    - Identidad
  - No Lineales
    - Sigmoides
      - Logistica
      - Tanh



$$f(x) = x$$





$$P(t) = \frac{1}{1 + e^{-t}}$$

$$\tanh x = \frac{\sinh x}{\cosh x}$$

Datos de entrada y su salida esperada

$\xi = (\xi_1, \xi_2, \xi_3) =$

4.4793	-4.0765	4.4558
-4.1793	-4.9218	1.7664
-3.9429	-0.7689	4.8830
-3.5796	1.5557	2.6683
-3.3354	2.2292	-1.6330
1.2096	0.3121	1.6238
0.7371	-3.9118	-2.5583
-4.4792	1.3177	-2.0449
4.3120	-3.7350	1.8018
2.2866	-3.6570	0.2785
2.3784	-4.0141	-0.8841

TP3-ej2-Conjunto-entrenamiento.txt

87.3174
1.5257
39.7859
45.5674
13.3589
74.5119
3.3358
4.2974
66.5833
26.0005
14.6809

$= \zeta$

TP3-ej2-Salida-deseada.txt

Problema de estandarización

Tenemos salidas esperadas en el mundo de los reales, sin embargo algunas de nuestras funciones de activación trabajan en un rango acotado, ¿Como podemos solucionar ese problema?

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

OBS: El cálculo del error final siempre se calcula con los valores desestandarizados, osea que se le aplica la transformación inversa

Capacidad de generalización de perceptrón no lineal

- Capacidad de generalización
  - Capacidad de obtener buenos resultados con datos que no formaron parte del conjunto de entrenamiento

- Dividimos a los datos de entrada en dos grupos
  - **Subconjunto de prueba** con 40 datos
  - **Subconjunto de entrenamiento** con 160 datos

- Fuimos variando el **factor de aprendizaje** para evaluar cómo afecta éste al "accuracy" del perceptron para ambos conjuntos

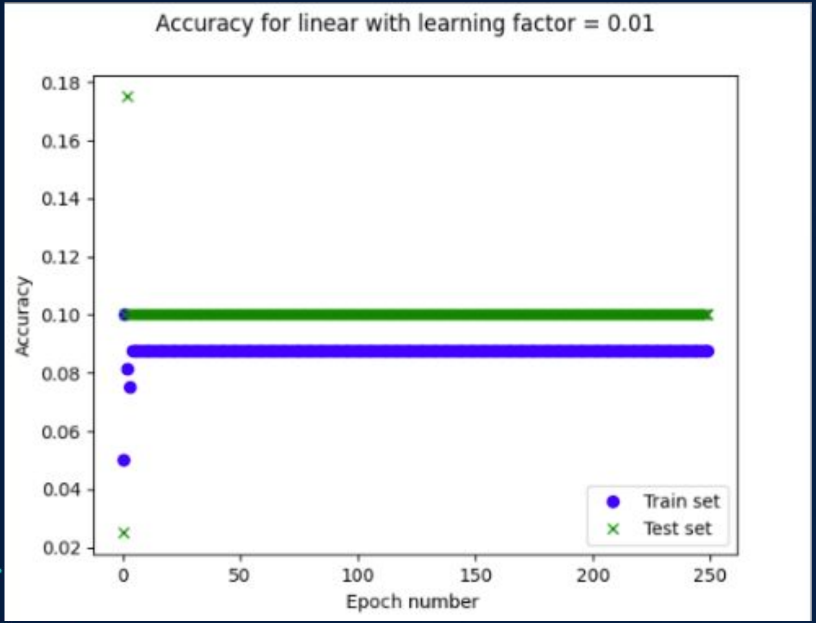
¿Como medimos el "accuracy"?

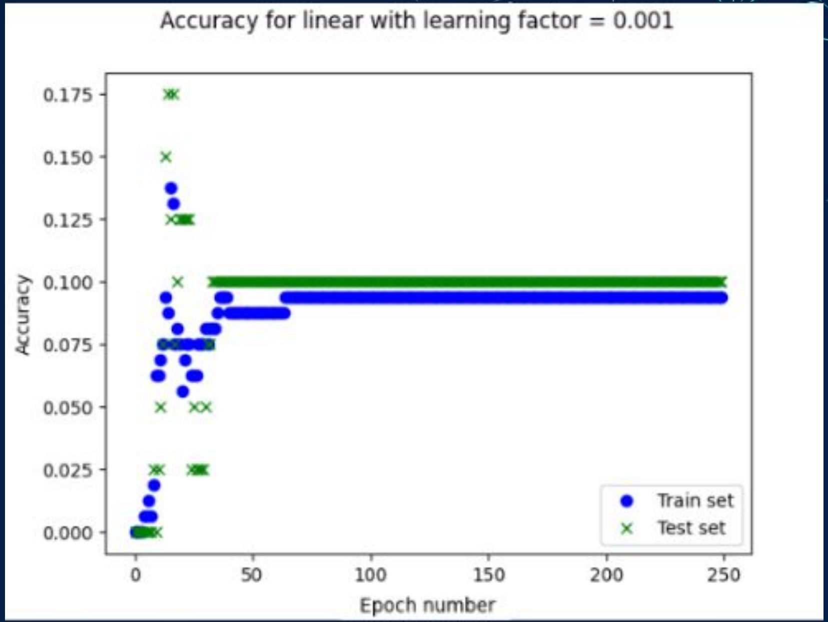
- Dado un dato de entrenamiento, si el resultado obtenido por el perceptrón está dentro de un rango definido, se toma que acerto, caso contrario que no

Evalue la *capacidad* **del** *perceptron simple lineal* **y** *perceptron simple no lineal* **para** *aprender* **la función cuyas muestras están presentes en los archivos indicados.**

PERCEPTRÓN *lineal*

Θ: Funcion de activacion



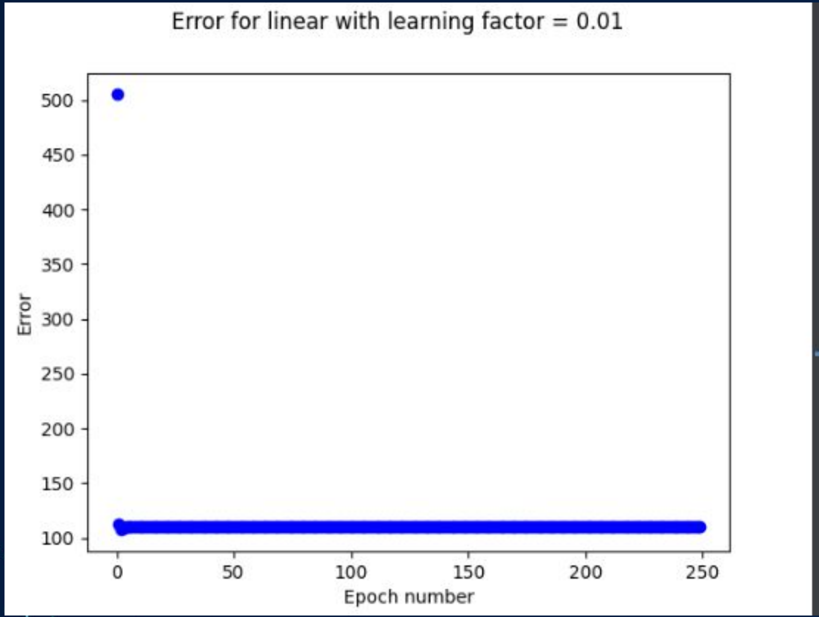


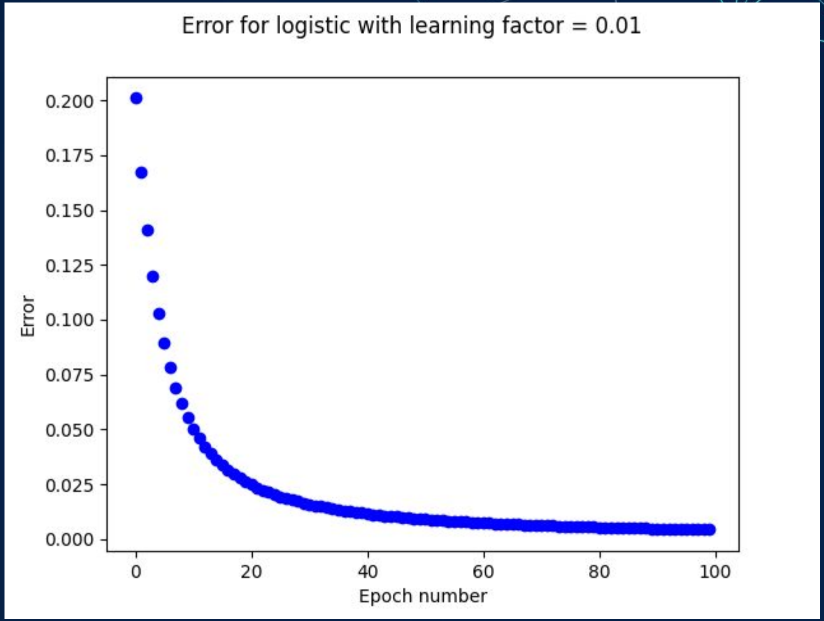
- Train set contiene 160 datos
- Test set contiene 40 datos
- Accuracy tolerance = 1.5

$$\Theta = x$$

PERCEPTRÓN *lineal* **VS** *no lineal*

Θ: Funcion de activacion





$$\Theta = x$$

Error final: 98.32

$$\Theta = \frac{1}{1 + e^{-t}}$$

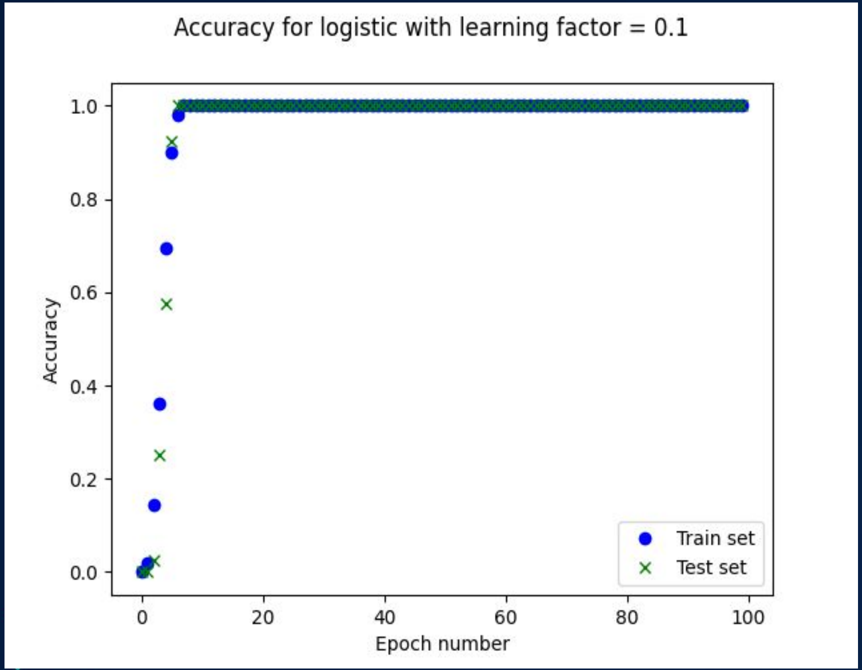
Error final: 8.9x10^-3

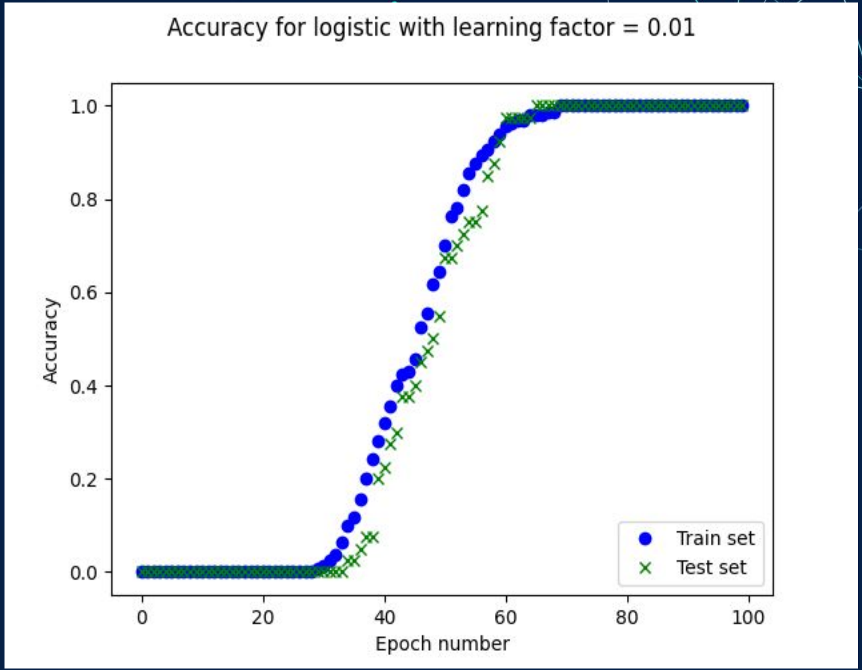
- Train set contiene 160 datos
- Test set contiene 40 datos

Evalúe la *capacidad de generalización* **del** *perceptrón simple no lineal* **utilizando, de los datos provistos, un** *subconjunto* **de ellos para** *entrenar* **y otro subconjunto para** *testear*.

Capacidad de generalización de *perceptrón no lineal*

Θ: Funcion de activacion



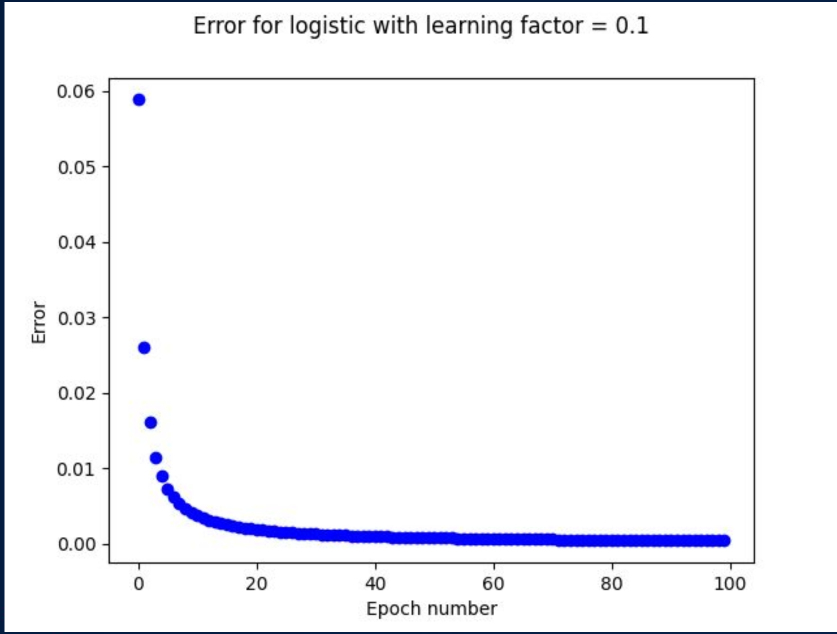


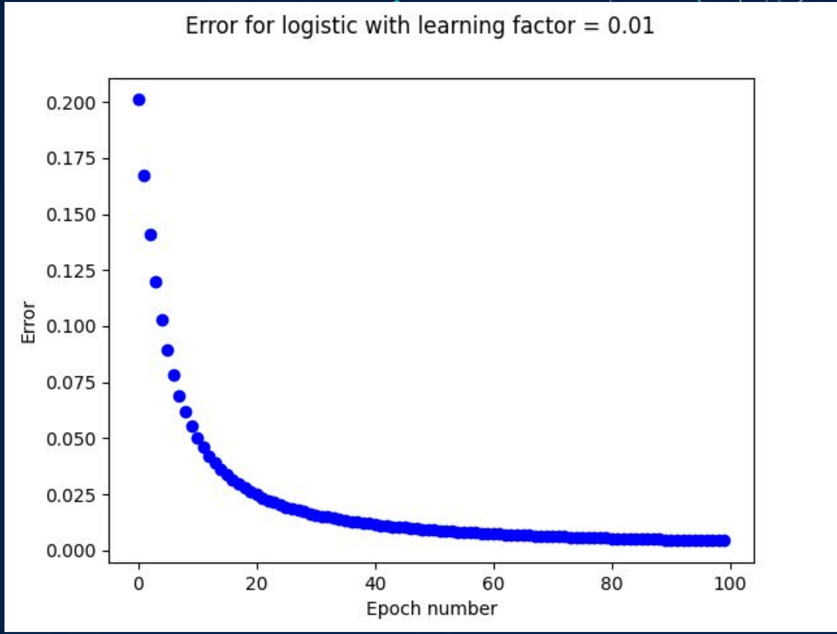
- Train set contiene 160 datos
- Test set contiene 40 datos
- Accuracy tolerance = 0.008

$$\Theta = \frac{1}{1+e^{-t}}$$

Capacidad de generalización de *perceptrón no lineal*

Θ: Funcion de activacion





Error final: 4.1x10^-3

$$\Theta = \frac{1}{1 + e^{-t}}$$

Error final: 8.9x10^-3

- Train set contiene 160 datos
- Test set contiene 40 datos

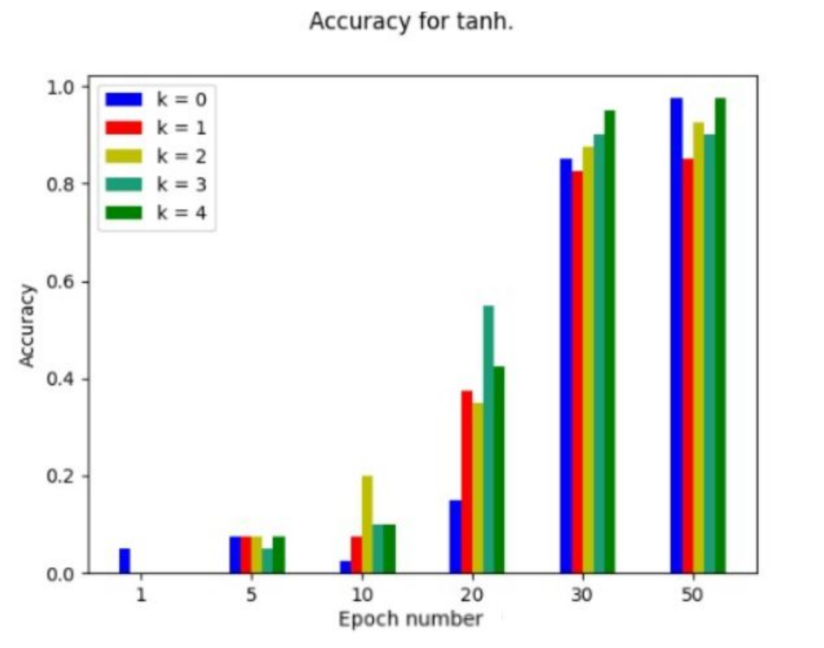
¿Como podría escoger el mejor conjunto de entrenamiento?

y

¿Cómo podría evaluar la máxima capacidad de generalización del perceptrón para este conjunto de datos?

Validación cruzada

Θ: Funcion de activacion



$$\Theta = \tanh x$$

- Se **divide el set** de entrada en 5 subconjuntos con 40 datos cada uno
- Se hacen **5 entrenamientos** (5 redes distintas)
  - Se usa como set de prueba uno de los 5 subconjuntos
  - Se usa como set de entrenamiento el resto de los datos
- Se elige al que obtenga el mejor "accuracy" en el **set de pruebas**

- Accuracy tolerance = 0.008

Obs: el par entrenamiento-prueba usado para los gráficos anteriores es, de hecho k=4

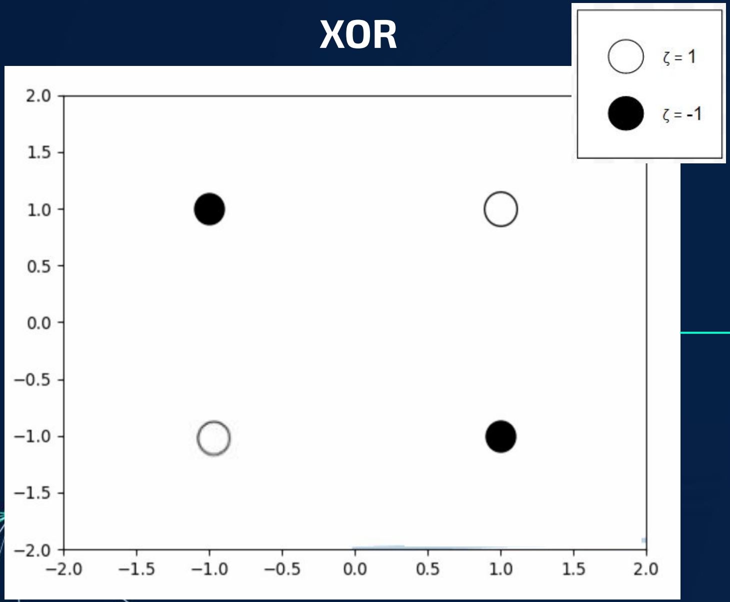
EJ.3
PERCEPTRON MULTICAPA

EJ.3.1

‘O exclusivo’

Problema con Perceptron Simple

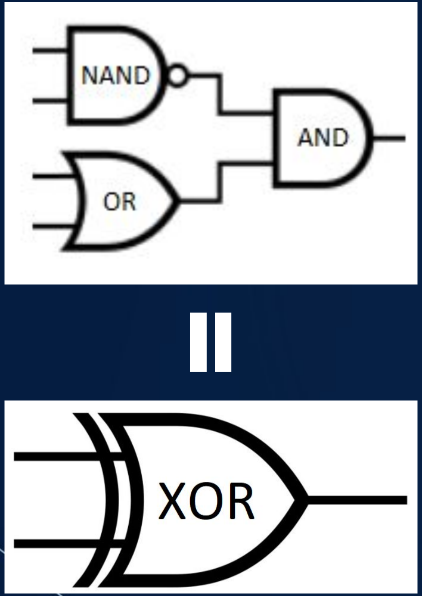
XOR

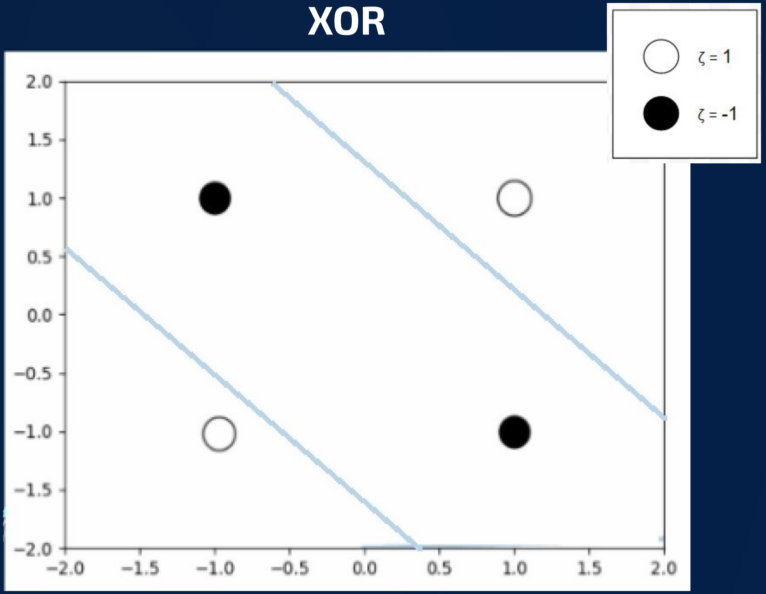


Problema NO linealmente separable

- Una única recta no sirve para separar los puntos

Composición de "funciones"





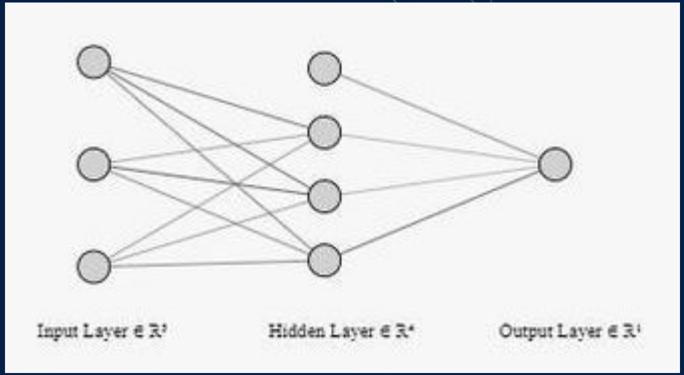
OBS: non-linear decision boundary

PERCEPTRON MULTICAPA

- Funcion logica “O exclusivo” (XOR)
  - Input: [{ -1, 1 }, { 1, -1 }, { 1, 1 }, { -1, -1 }]
  - Expected: [1, 1, -1, -1]

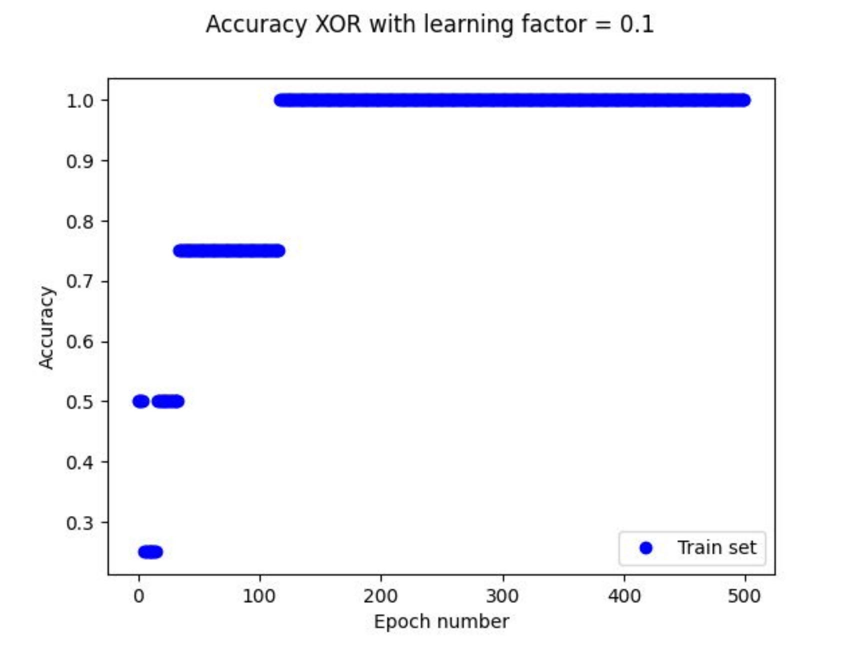
XOR

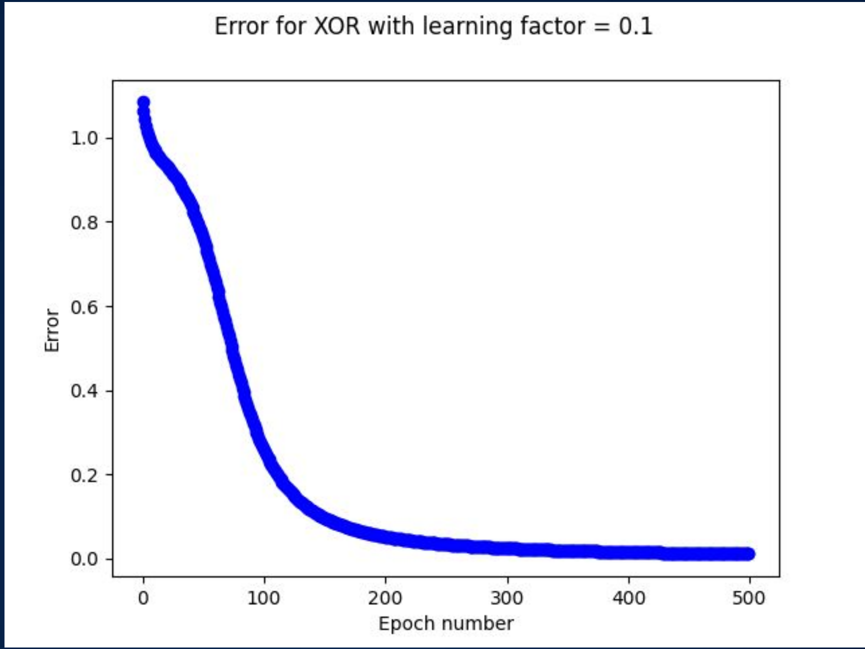
A	B	Expected
-1	1	1
1	-1	1
1	1	-1
-1	-1	-1



Resultados obtenidos

Θ: Funcion de activacion





$$\Theta = \tanh x$$

EJ.3.2

¿par o impar?

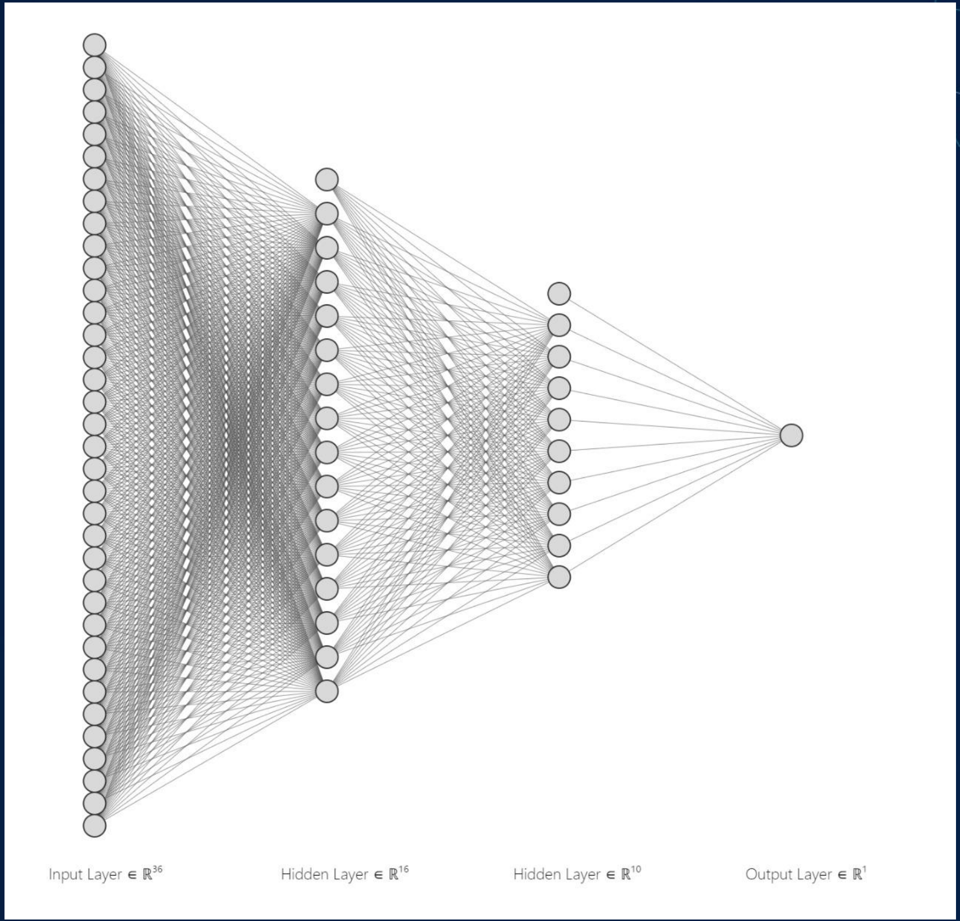
Problema de paridad

Queremos, dada una entrada de números representados por 5x7 pixeles, entrenar nuestra red para que adivine la paridad de cada número. Podemos aplanar la matriz de píxeles para tener un vector de 35 entradas. Proponemos la siguiente arquitectura para la solución.

- Input layer de 35 nodos + un bias
- Hidden layer de 16 nodos + un bias
- Hidden layer de 10 nodos + un bias
- Unico output node

Regla de los ⅔: La cantidad de nodos en hidden layers para un problema simple debería ser aproximadamente ⅔ del tamaño del input + el tamaño de la salida

Arquitectura

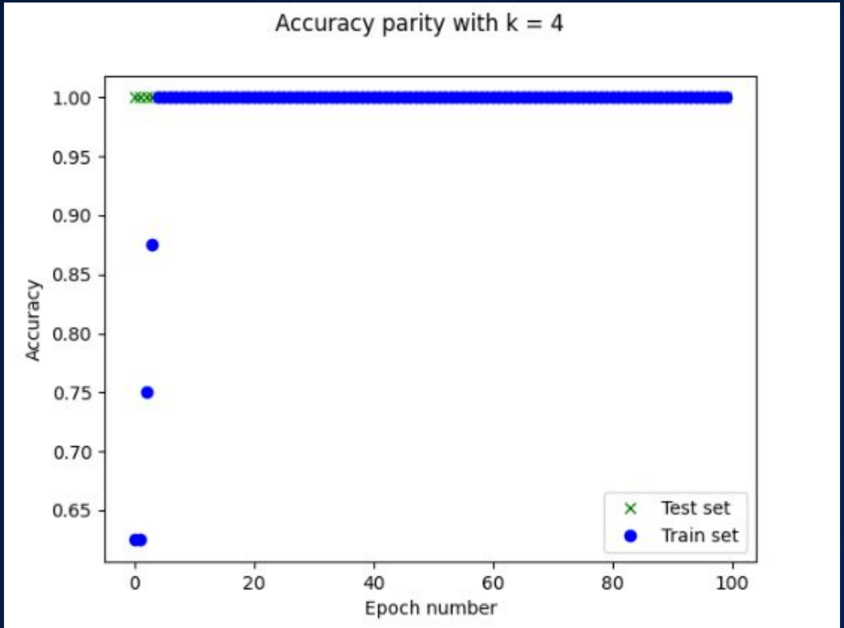


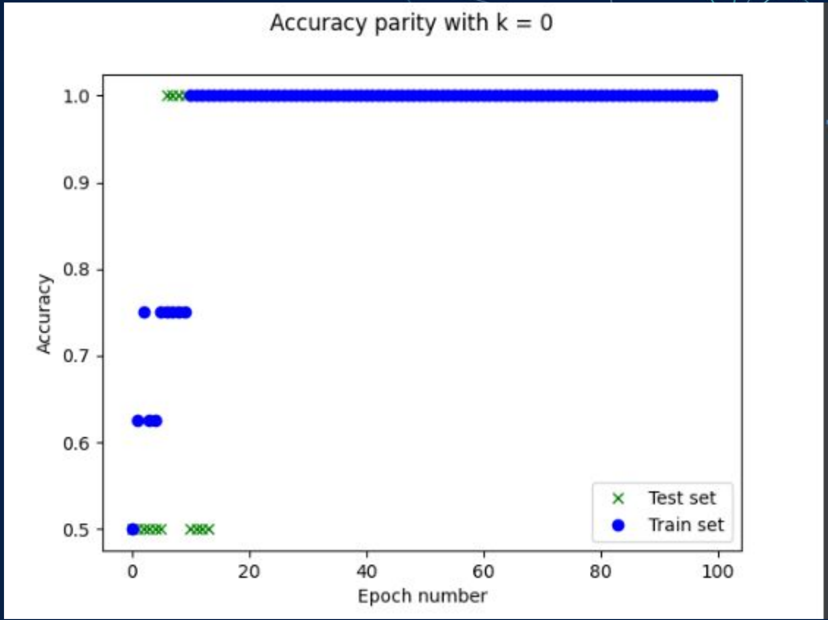
Cómo elegimos nuestro test set

- Se seleccionar pares al azar y se los separamos del train set
- Se entrena cada red con un train set distinto y luego se mide el accuracy para cada test set
- Test set obtenidos
  - K^0 : (7, 0)
  - K^1 : (6, 8)
  - K^2 : (9,2)
  - K^3 : (3,1)
  - K^4 : (4,5)

Capacidad de generalización adecuada

Θ: Funcion de activacion





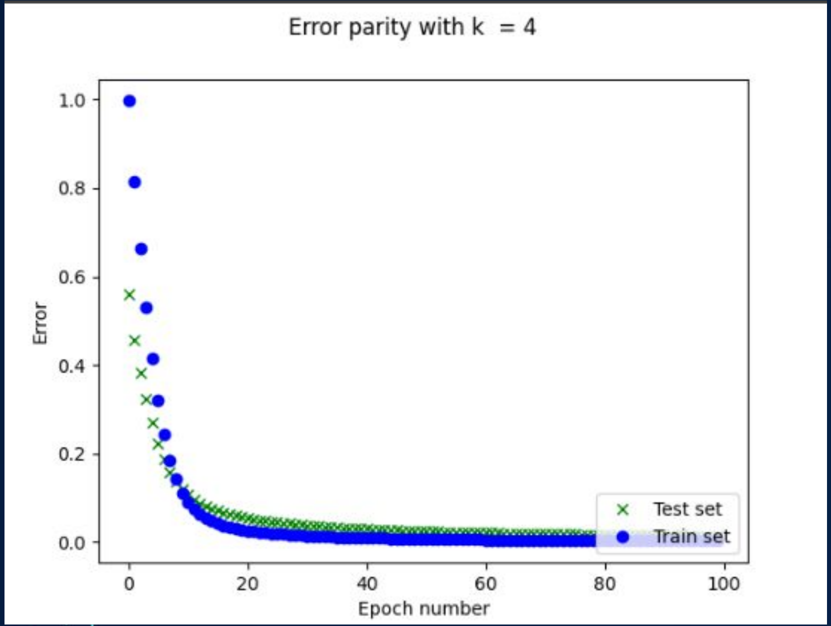
$$\Theta = \tanh x$$

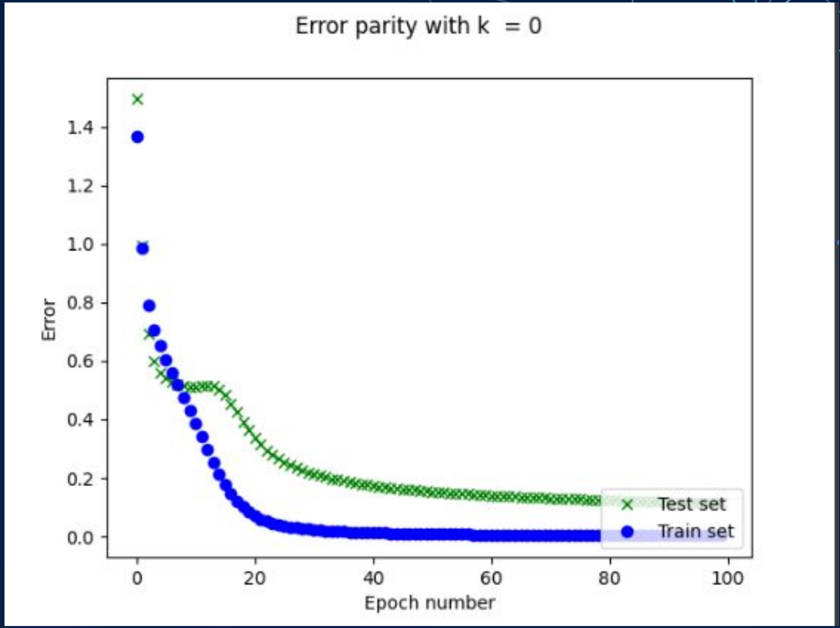
- Train set contiene 8 datos
- Test set contiene 2 datos

- $K^4 = (4,5)$
- $K^0 = (7,0)$

Capacidad de generalización adecuada

Θ: Funcion de activacion





Error final: 2.8x10^-2

Error final: 3.9x10^-3

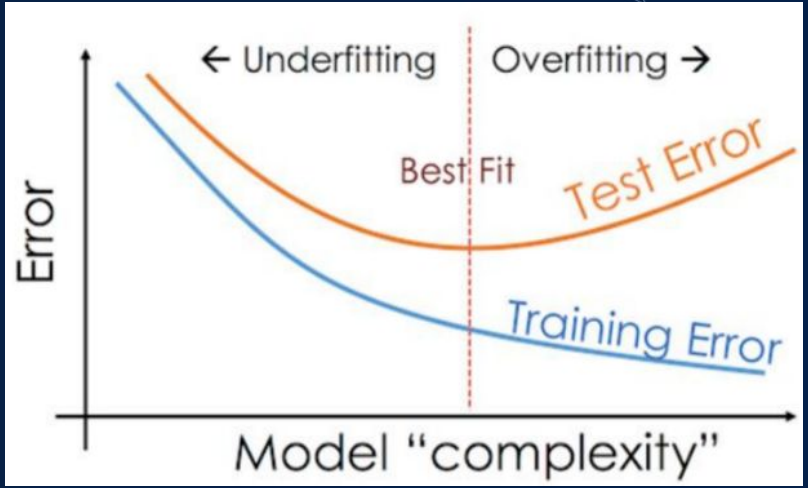
$$\Theta = \tanh x$$

- Train set contiene 8 datos
- Test set contiene 2 datos

- $K^4 = (4,5)$
- $K^0 = (7,0)$

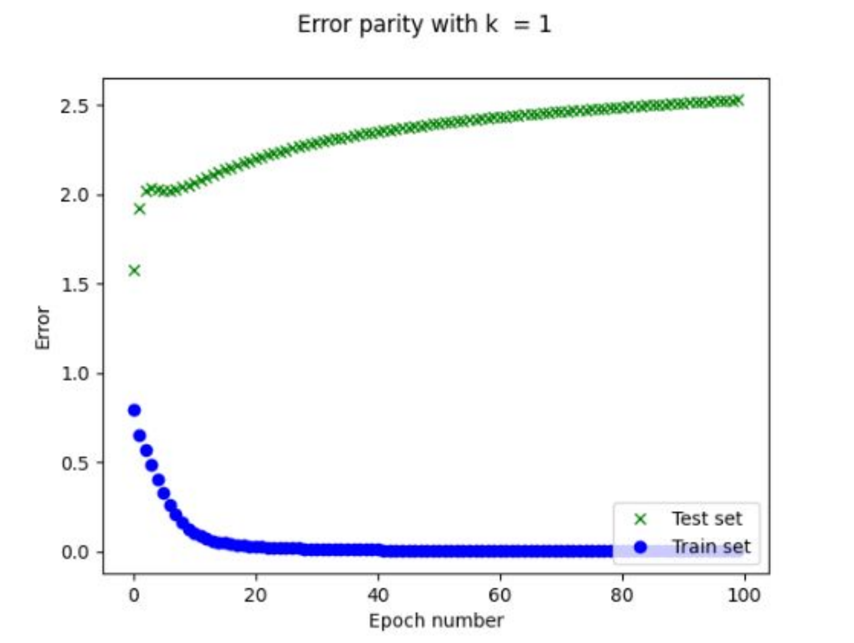
Sobre Ajuste

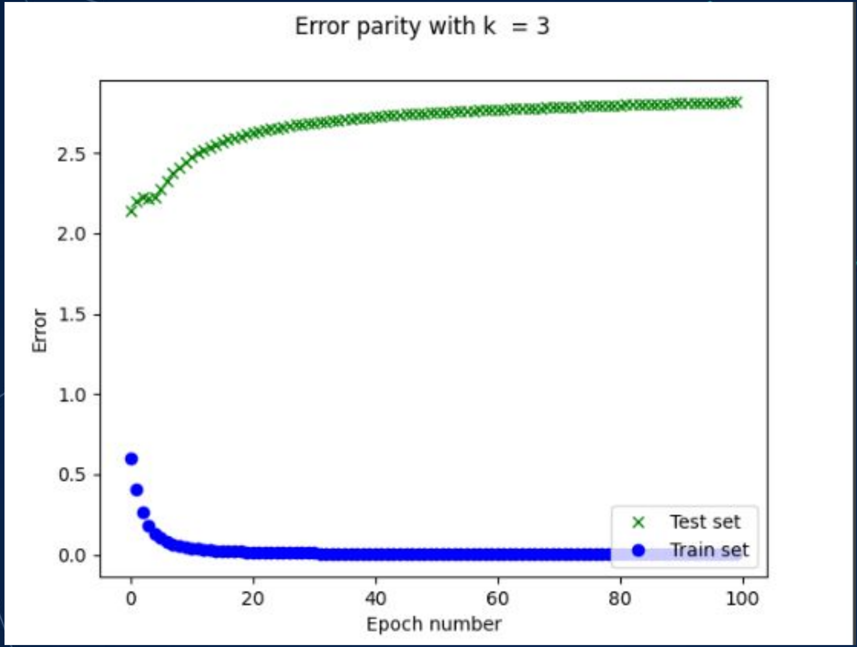
Si se sobre extiende el entrenamiento, la red aprende muy bien a resolver los numeros del train set pero pierde la capacidad de resolver nuevos problemas. A mayor learning rate, más rápidamente se observa este comportamiento.



¿Siempre es el caso?

Θ: Funcion de activacion





Error final: 7.1x10^-2

Error final: 4.8x10^-4

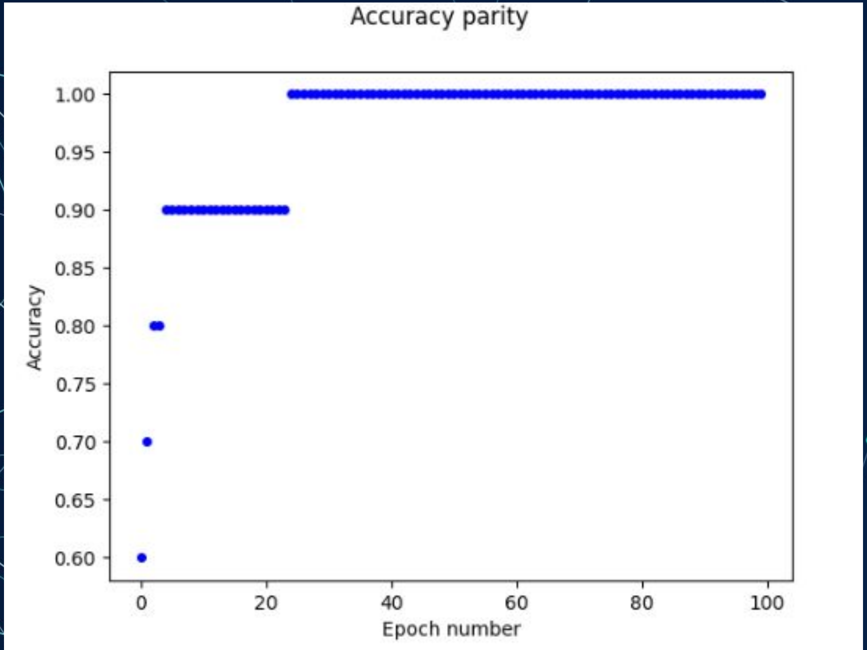
$$\Theta = \tanh x$$

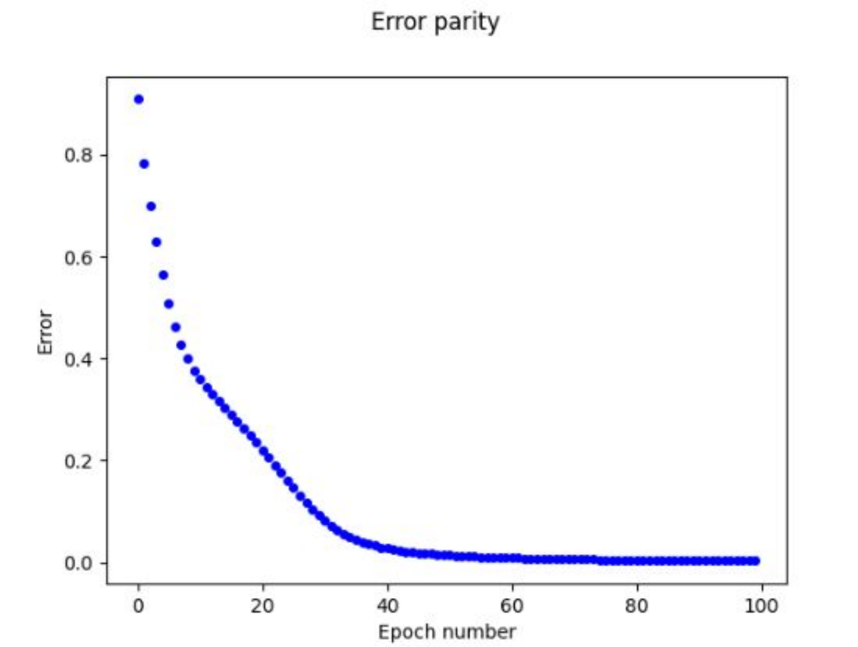
- Train set contiene 8 datos
- Test set contiene 2 datos

- K^1 = (6,8)
- K^3 = (3,1)

Resultados obtenidos entrenando con todo el set

Θ: Funcion de activacion

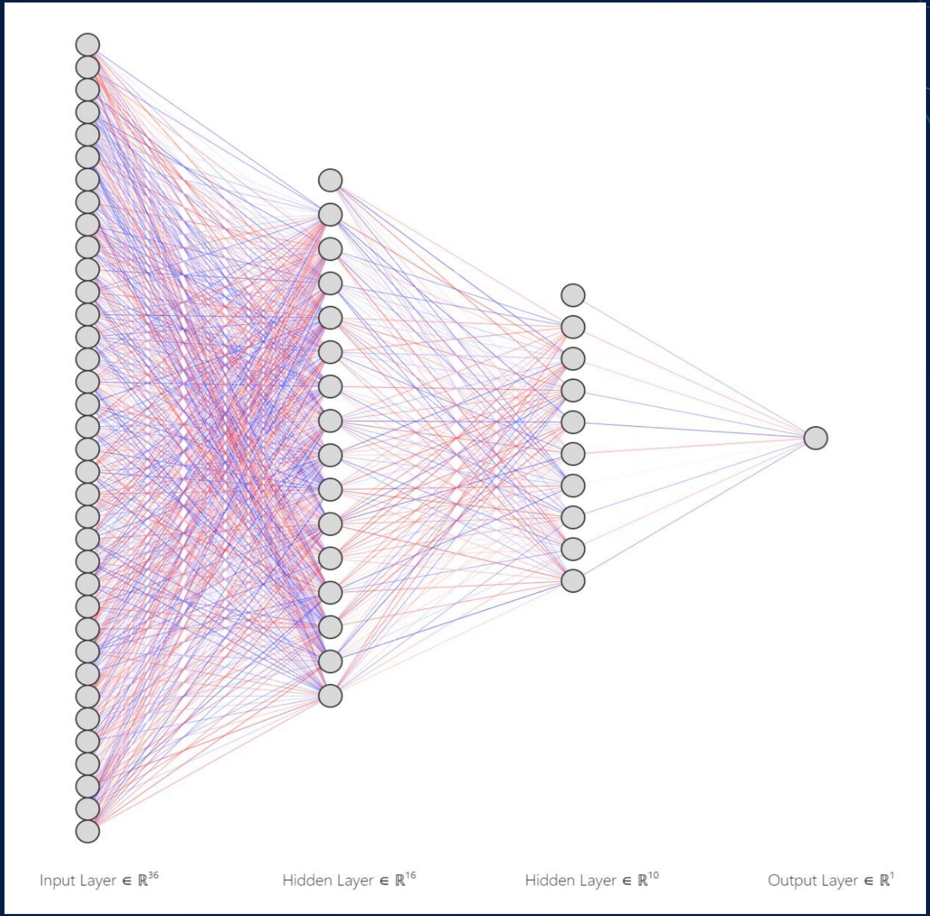




Error final: 8.9x10^-5

$$\Theta = \tanh x$$

Estado final



Rojo: pesos negativos
Azul: pesos positivos
Opacidad: modulo del peso

EJ.3.3

¿que número es?

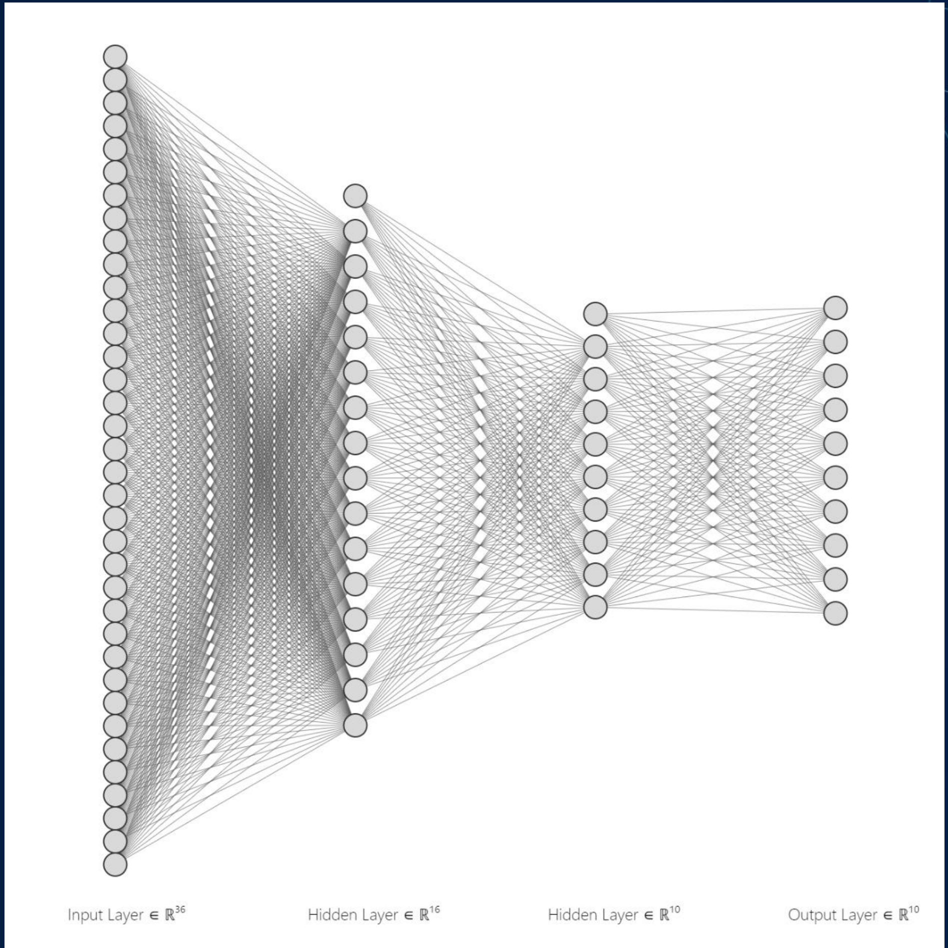
Problema de clasificación numérica

Queremos, dada una entrada de números representados por 5x7 pixeles, entrenar nuestra red para que adivine el valor real de cada número. Podemos aplanar la matriz de píxeles para tener un vector de 35 entradas. Proponemos la siguiente arquitectura para la solución.

- Input layer de 35 nodos + un bias
- Hidden layer de 16 nodos + un bias
- Hidden layer de 10 nodos + un bias
- 10 output node

Regla de los ⅔: La cantidad de nodos en hidden layers para un problema simple debería ser aproximadamente ⅔ del tamaño del input + el tamaño de la salida

Arquitectura



Evaluación de generalización

Para ver que tan bien generaliza la red, se mutaron los números originales con una probabilidad P. Se mutó el conjunto original 250 veces -> **2500 datos de testeo**. Esta mutación puede llegara a entenderse como dos cosas:

- La evaluación de imágenes que se comprimieron - perdieron información en la comunicación
- Los números se dibujan de forma ligeramente distinta

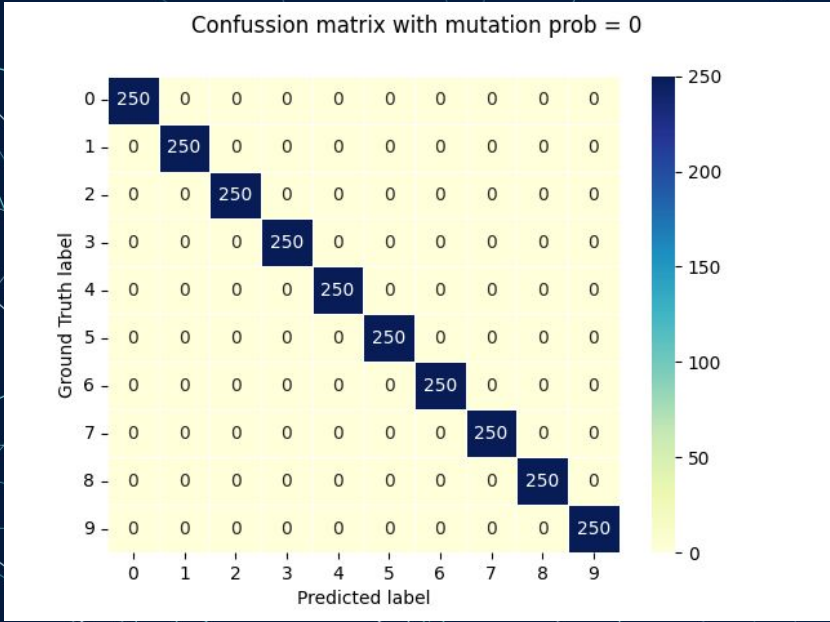
Tomemos la interpretacion que tomemos, estamos evaluando que tan bien nuestra red clasifica datos diferentes a los que usamos para entrenar.

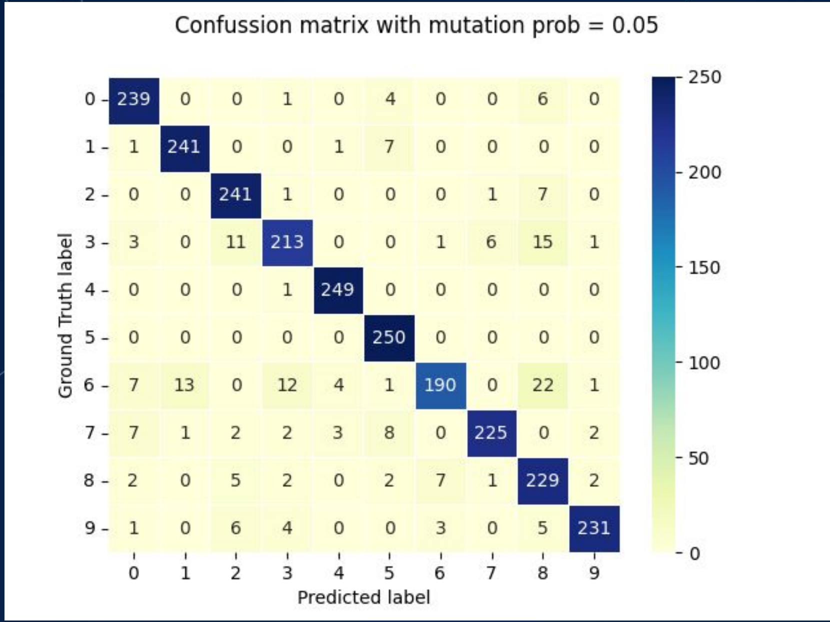
Entrenamiento de la red

- Parametros utilizados
    - Learning rate = 0.1
    - Función de activación = Tangente hiperbólica
    - Cantidad de iteraciones = 2500

Capacidad de generalización

Θ: Funcion de activacion





Accuracy : 1.00

Precision: 1.00

Recall : 1.00

$$\Theta = \tanh x$$

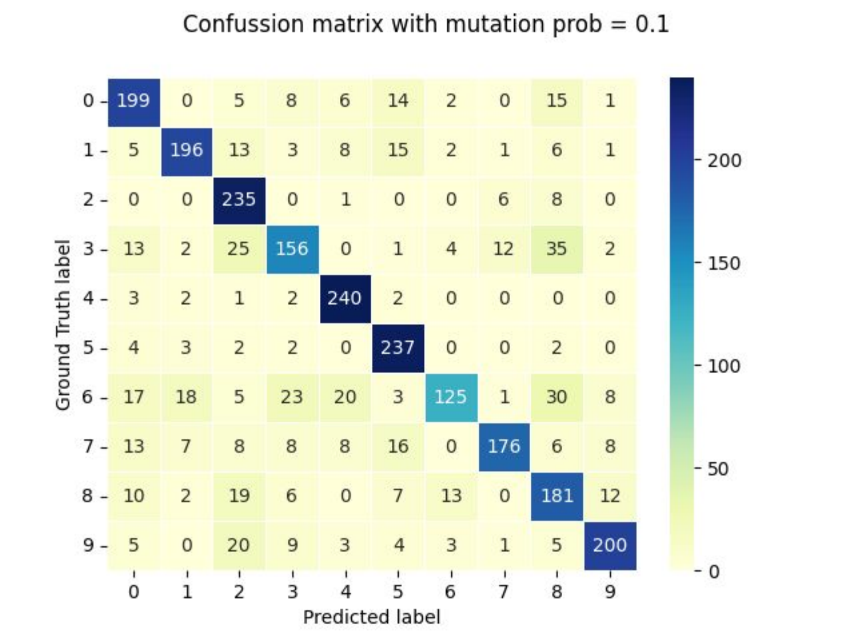
Accuracy : 0.984

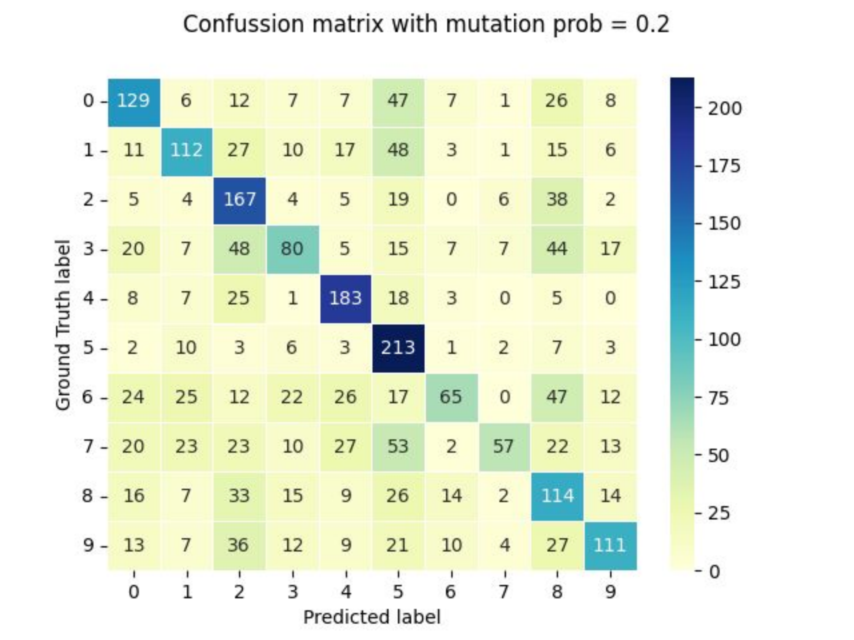
Precision: 0.9232

Recall : 0.9228

Capacidad de generalización

Θ: Funcion de activacion





Accuracy : 0.9556

Precision: 0.778

Recall : 0.777

$$\Theta = \tanh x$$

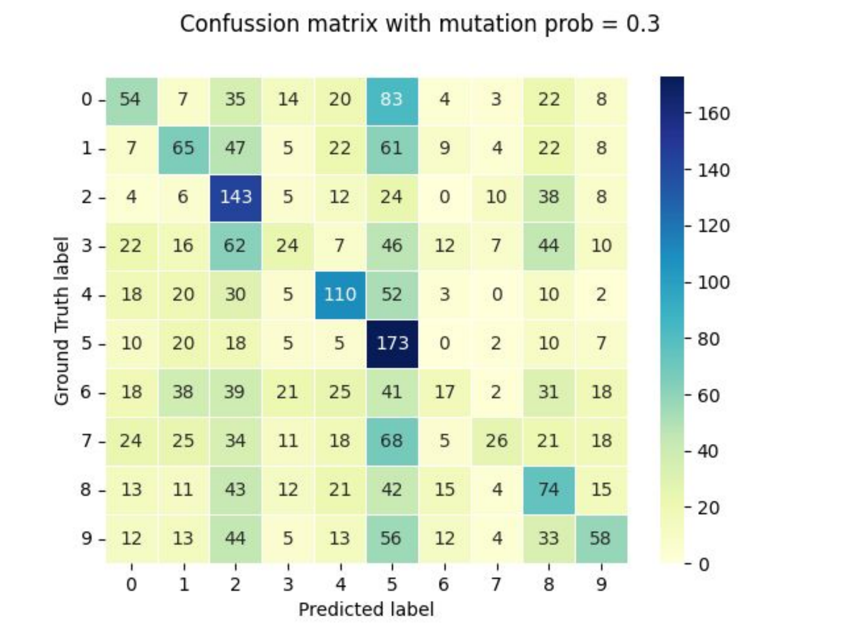
Accuracy : 0.898

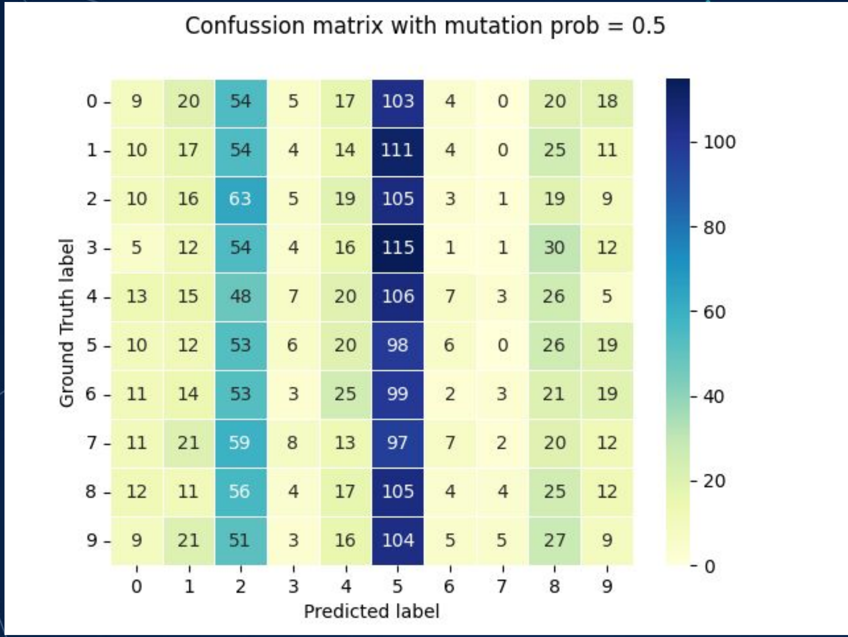
Precision: 0.69

Recall : 0.689

Capacidad de generalización

Θ: Funcion de activacion





Accuracy : 0.859

Precision: 0.2976

Recall : 0.2976

$$\Theta = \tanh x$$

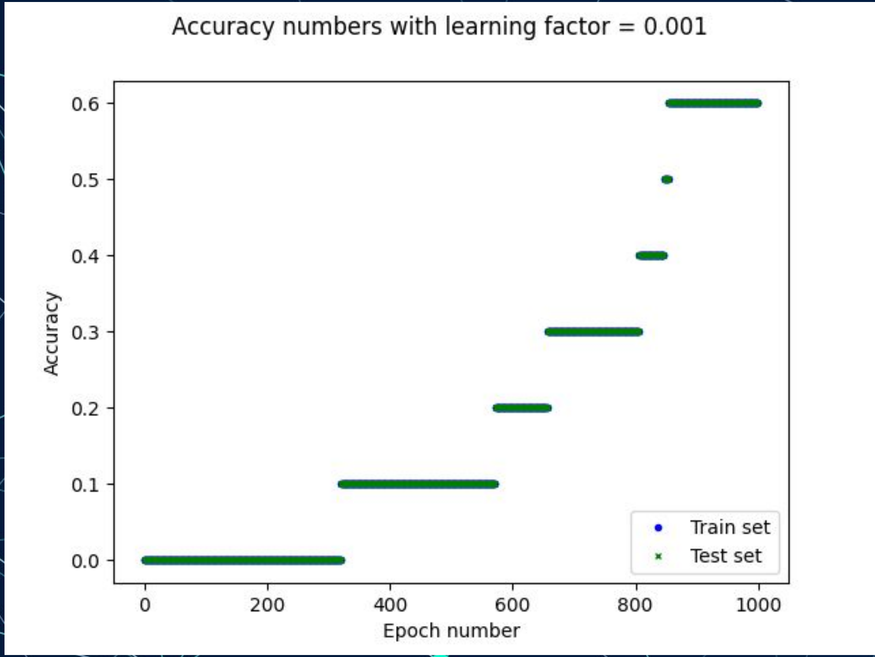
Accuracy : 0.819

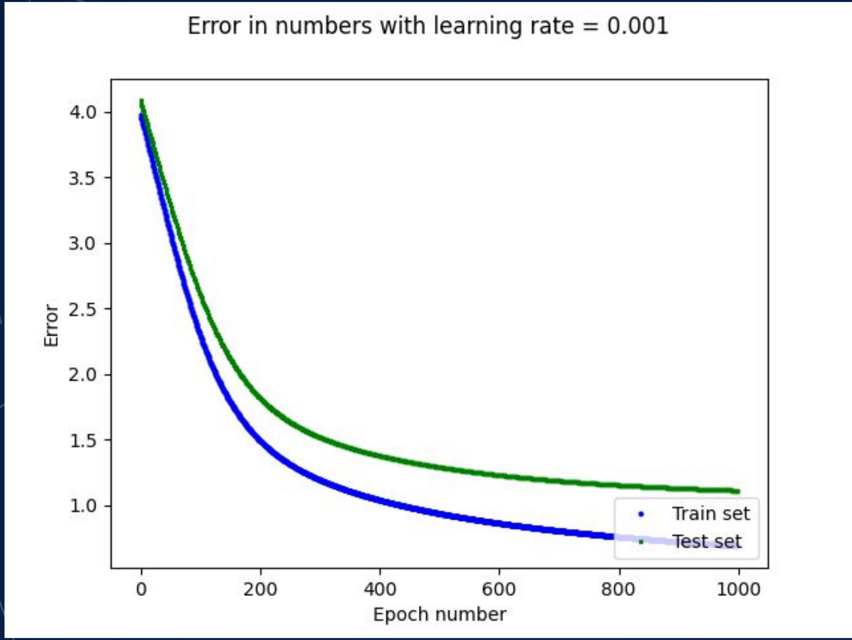
Precision: 0.0996

Recall : 0.109

Aprendizaje con el tiempo

Θ: Funcion de activacion

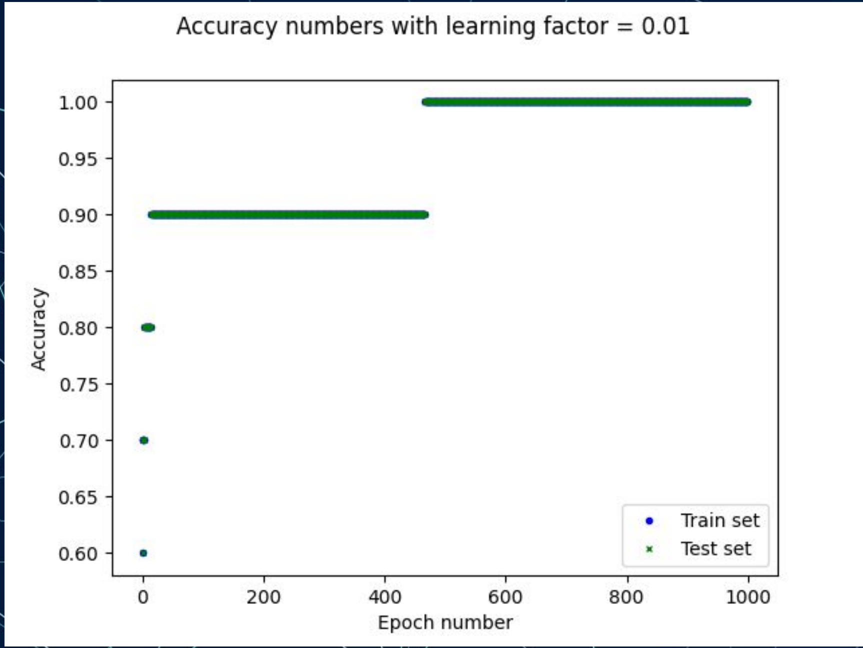


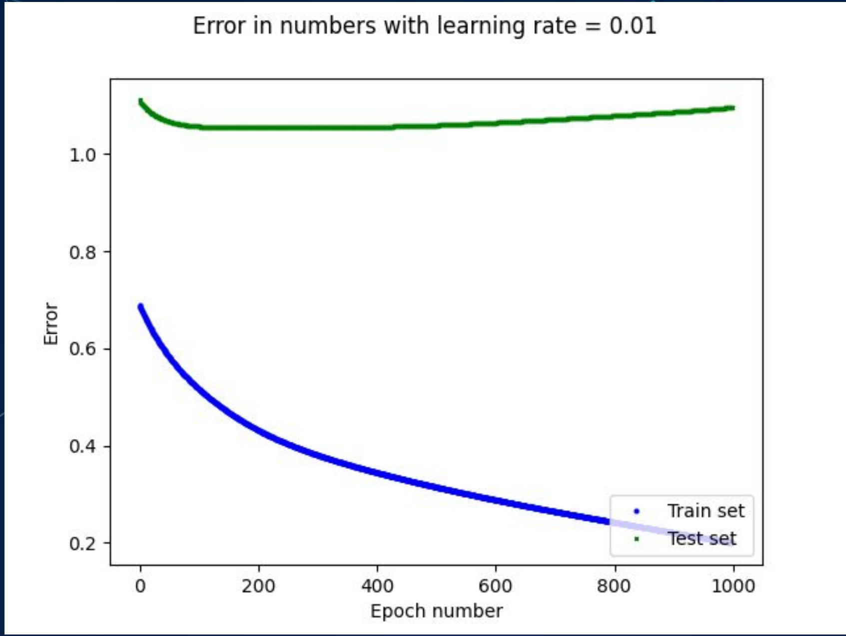


$$\Theta = \tanh x$$

Aprendizaje con el tiempo

Θ: Funcion de activacion

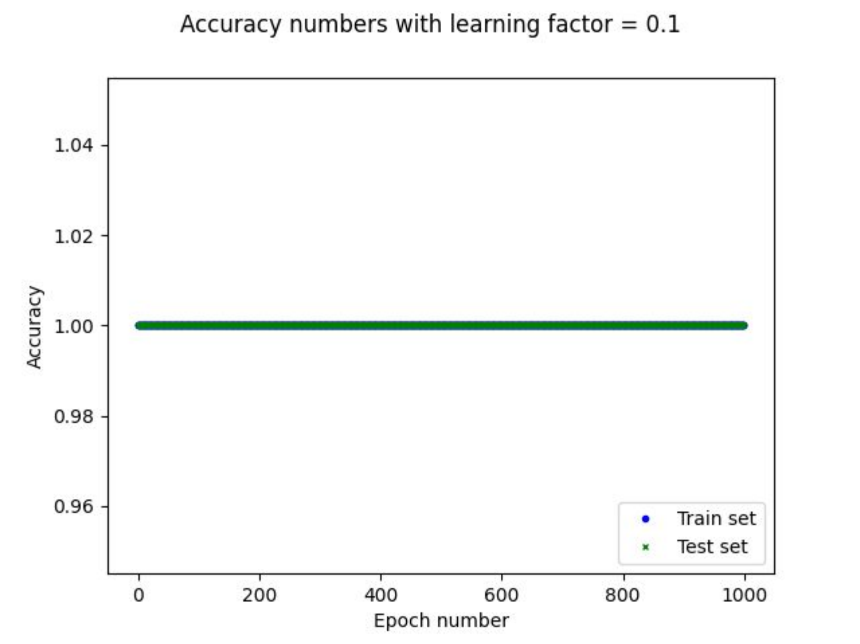


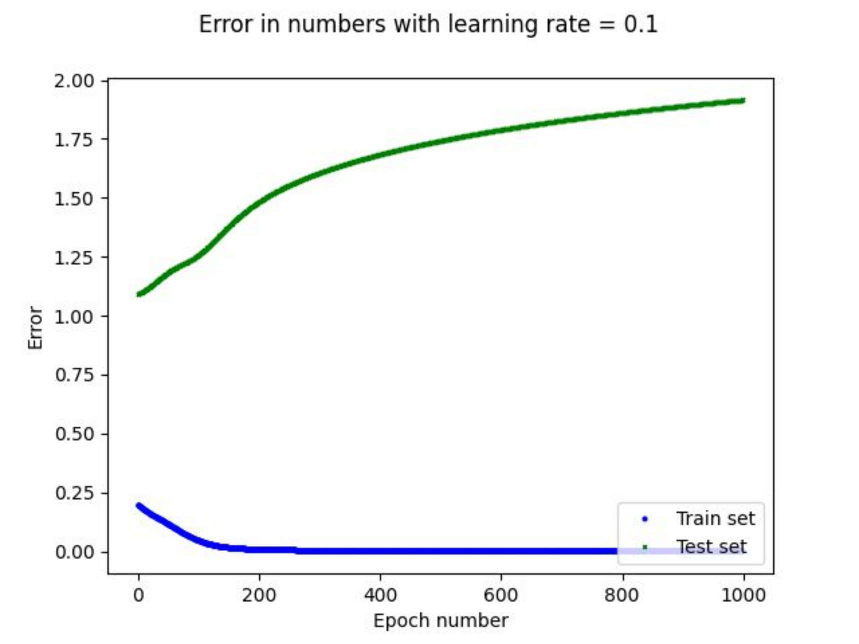


$$\Theta = \tanh x$$

Aprendizaje con el tiempo

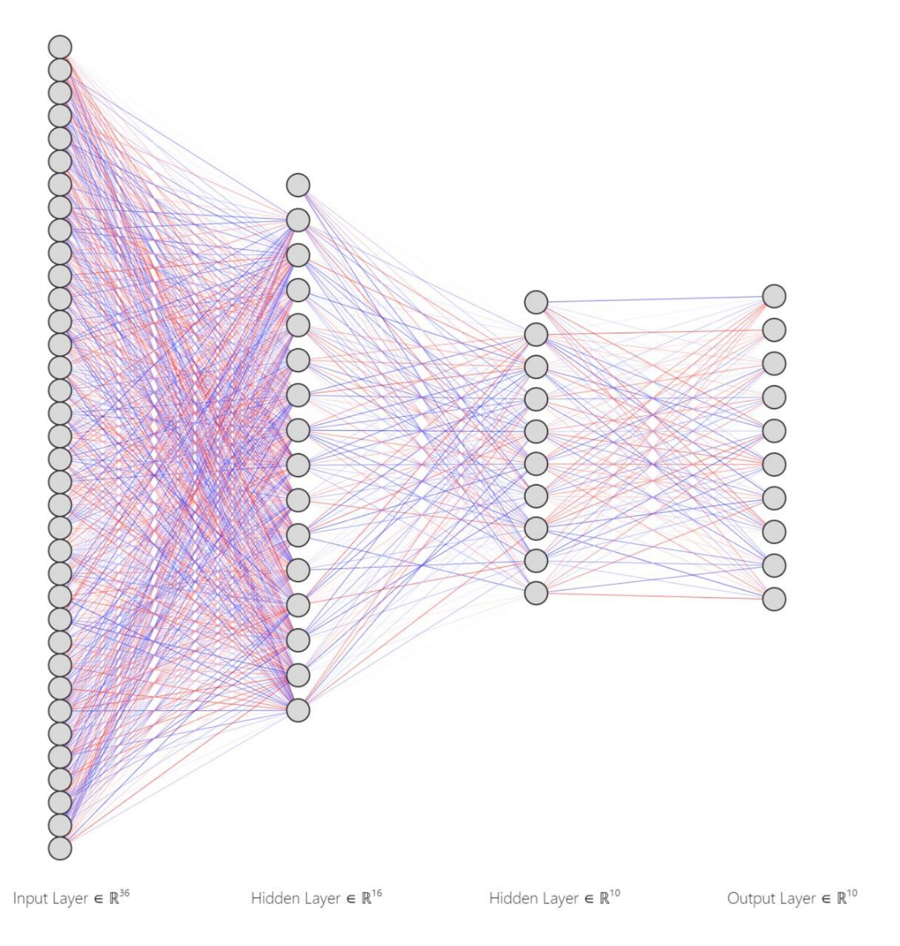
Θ: Funcion de activacion





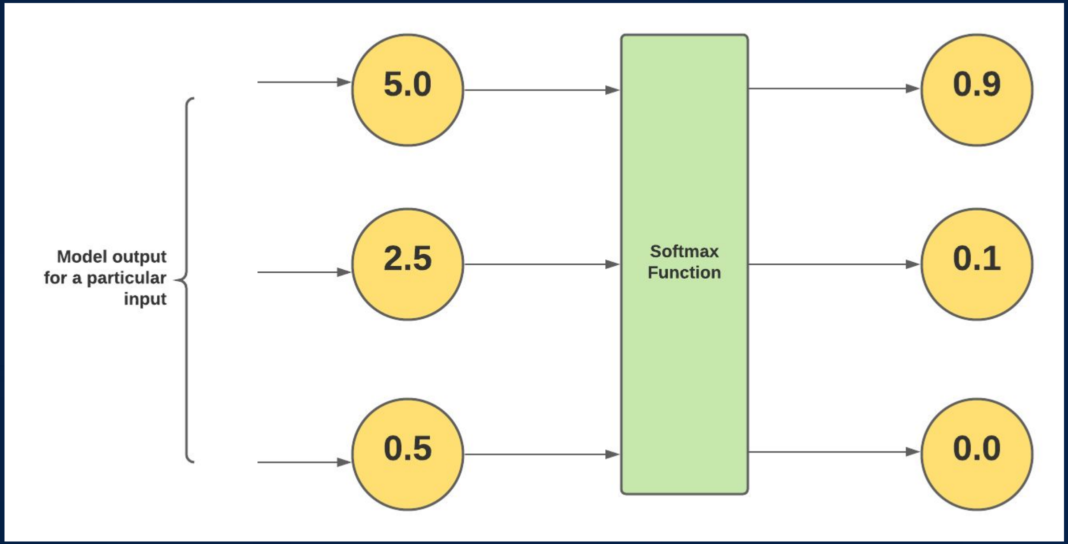
$$\Theta = \tanh x$$

Estado final



Soft-max

En problemas de clasificación con N salidas como el que acabamos de ver, se puede usar una capa de salida soft-max para transformar este conjunto de valores acotados en una distribución de probabilidad sobre las N posibles salidas.



Soft-max

Dada una mutación con 0.1 de un número 8 y redondeando las probabilidades a 2 decimales

	0	1	2	3	4	5	6	7	8	9
P(x)	0.0192	0.0	0.0	0.003	0.0	0.0	0.006	0.0	**0.971**	0.0

Gracias por su atencion.

¿Preguntas?