

# TP3

## Perceptrón Simple y Multicapa

Francisco Dominguez - 61068

Juan Pablo Arias - 60299

Gonzalo Elewaut - 61212

Facundo Anselmi - 60128

Enzo Ruedas - 65966



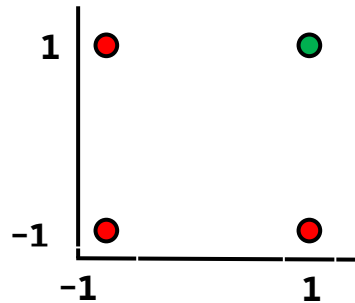
# Perceptrón Simple - Ejercicio 1

# Perceptrón Simple Escalón

AND

$\&$

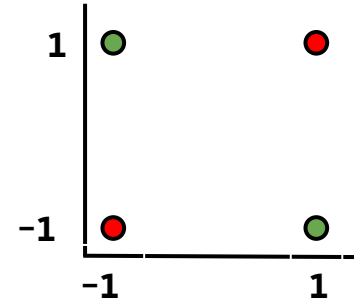
	1	-1
1	1	-1
-1	-1	-1



XOR

$\oplus$

	1	-1
1	-1	1
-1	1	-1

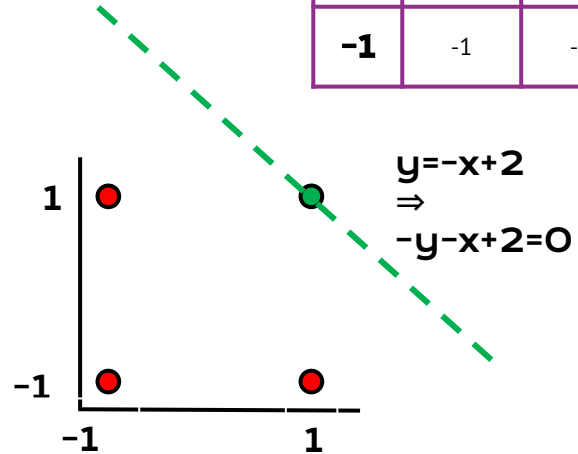


# Perceptrón Simple Escalón

**AND**

**&**

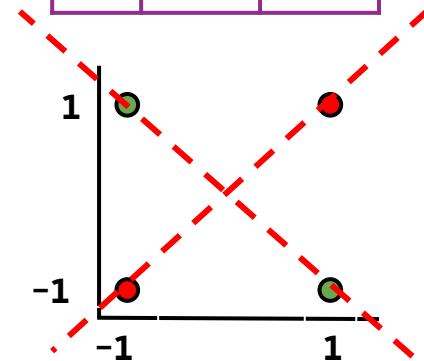
	1	-1
1	1	-1
-1	-1	-1



**XOR**

**$\oplus$**

	1	-1
1	-1	1
-1	1	-1



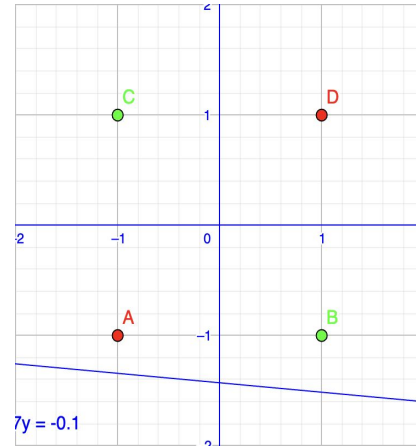
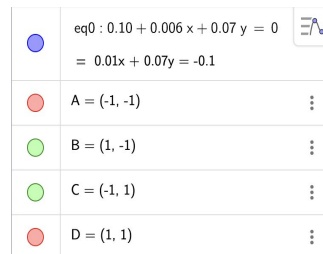
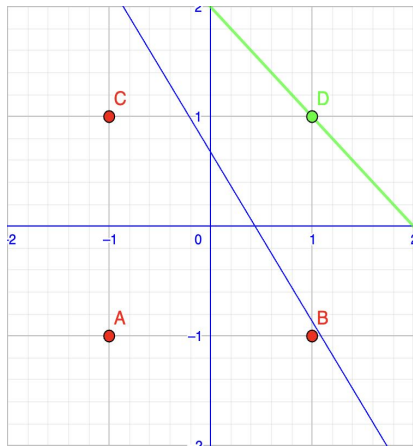
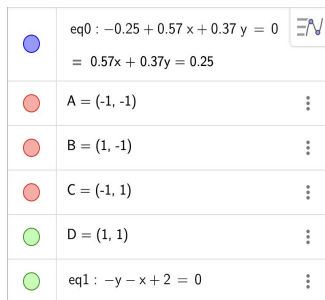
# Perceptrón Simple Escalón

For AND operator:

Input: [1 1], Predicted Class: 1  
 Input: [-1 -1], Predicted Class: -1  
 Input: [-1 1], Predicted Class: -1  
 Input: [ 1 -1], Predicted Class: -1  
 [-0.24657766 0.56744802 0.37103113]

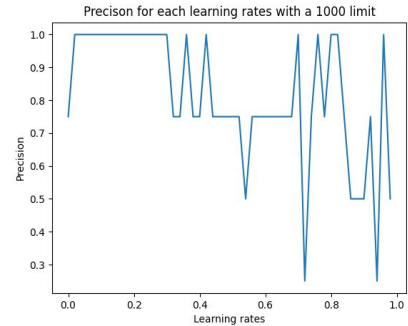
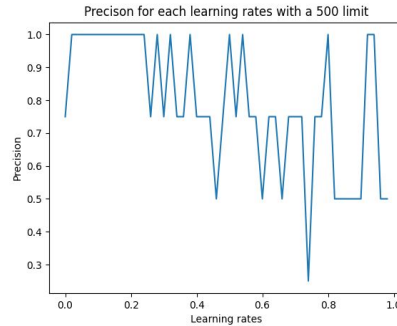
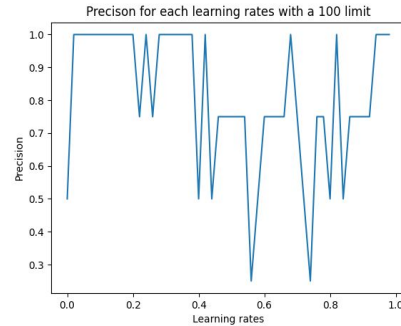
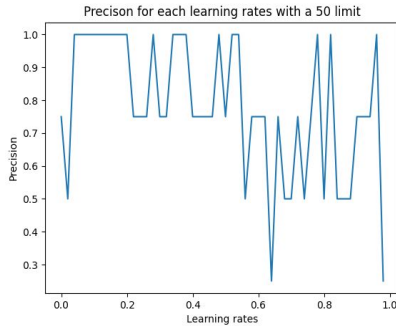
For XOR operator:

Input: [1 1], Predicted Class: 1  
 Input: [-1 -1], Predicted Class: 1  
 Input: [-1 1], Predicted Class: 1  
 Input: [ 1 -1], Predicted Class: 1  
 [0.10315568 0.00617233 0.06951592]



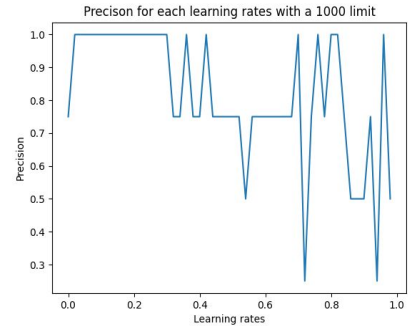
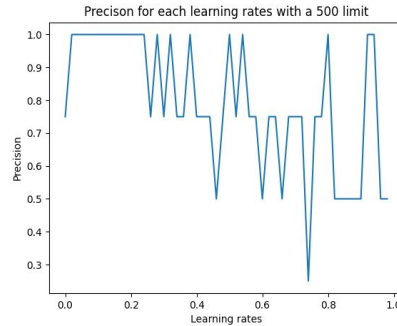
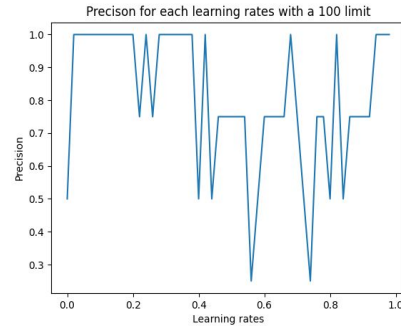
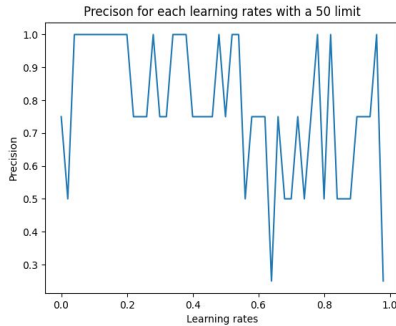
# Perceptrón Simple Escalón

learning rate = ?  
limit = ?  
epsilon = ?



# Perceptrón Simple Escalón

**learning rate = 0.1**  
**limit = 100**  
**epsilon = 0.01**





# Perceptron Lineal y No lineal



# Objetivo

En este ejercicio se buscó implementar en código la funcionalidad de un perceptrón lineal tanto como uno no lineal, para luego analizar los resultados obtenidos gracias a los datos ingresados.

Primero, se busco entrenar a los perceptrones con todos los datos, pero luego los primeros se dividieron en partes para el perceptrón no lineal y se graficaron algunos resultados.

Se debe considerar que para el perceptrón no lineal se utilizó la función tanh (tangente hiperbólica) por lo que se tuvieron que normalizar los datos ingresados de la imagen (y) al rango [-1;1] mediante la siguiente función:  $(2 * (\text{valor} - \text{min}) / (\text{max} - \text{min})) - 1$

# Funcionamiento

Estos dos tipos de redes neuronales aprenden a clasificar conjuntos de datos no necesariamente separables.

Por lo tanto, se necesita de una tarea de estimación, utilizando una función lineal o no lineal que se acerque lo más posible a los datos suministrados.

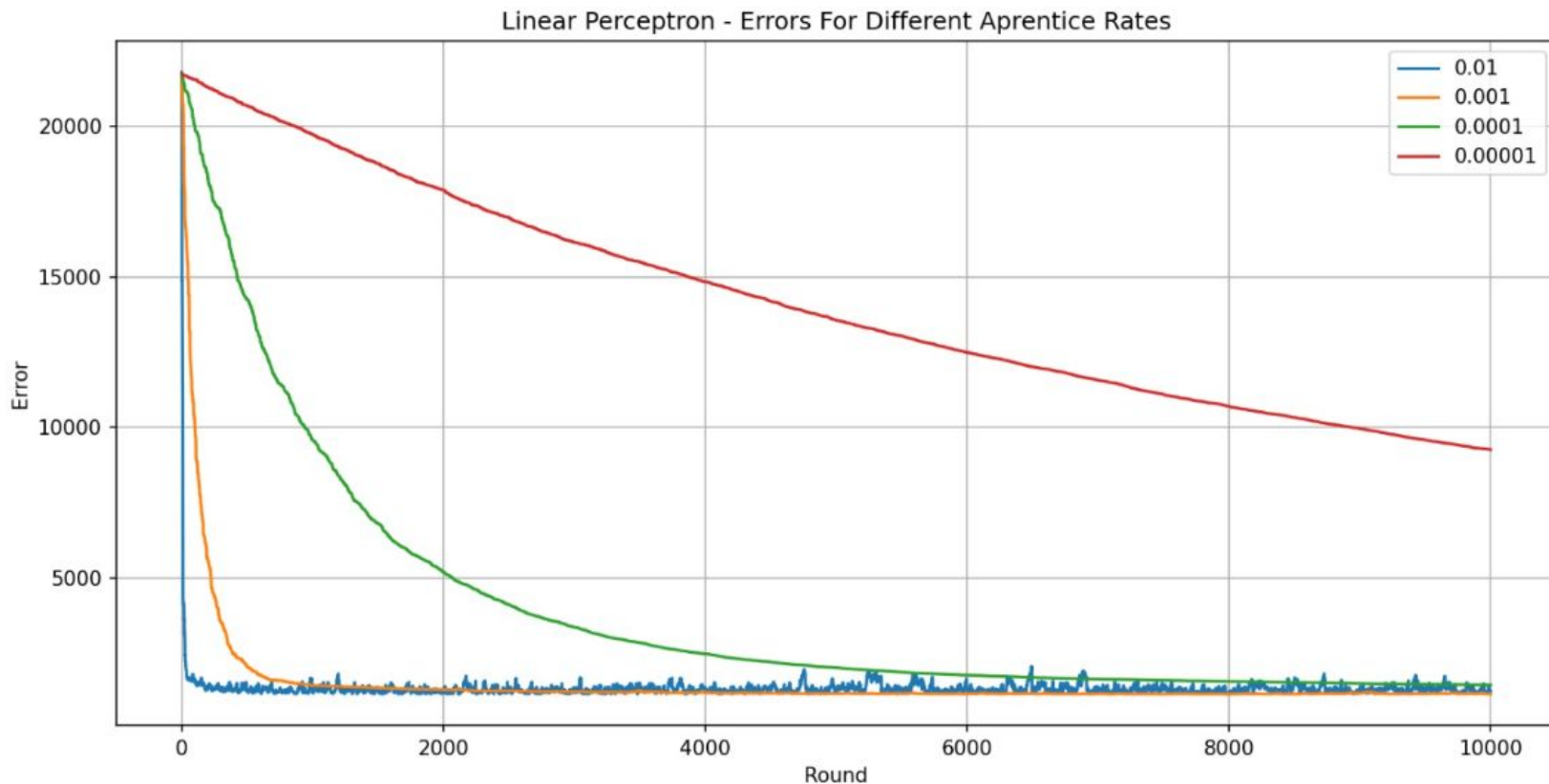
Para el funcionamiento de estos perceptrones se deben actualizar los pesos cuando sea necesario utilizando la siguiente función:

$$\eta(\zeta^{\mu} - o^{\mu})\theta'(h)x^{\mu}$$

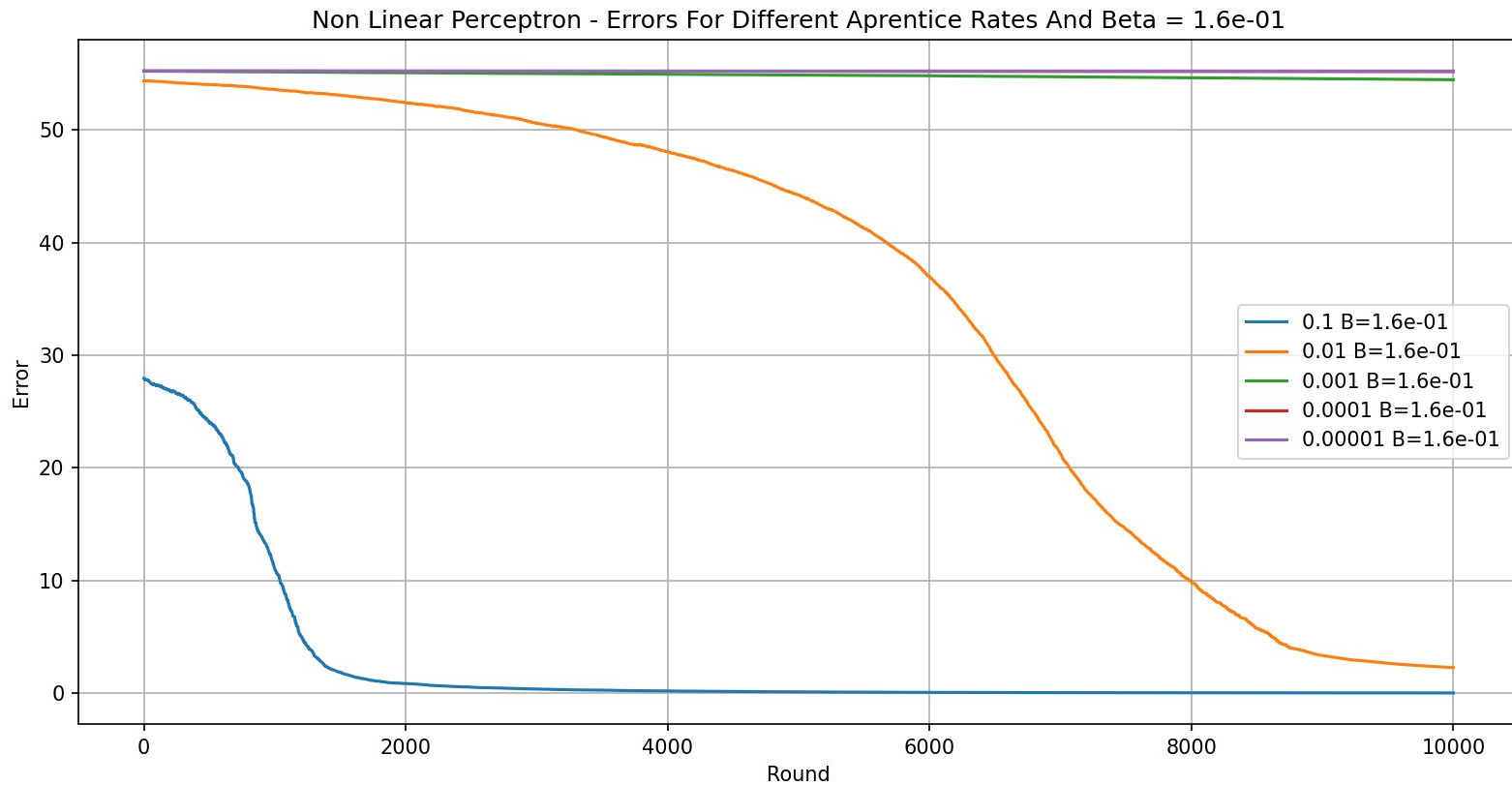


## Ejercicio 2 - A

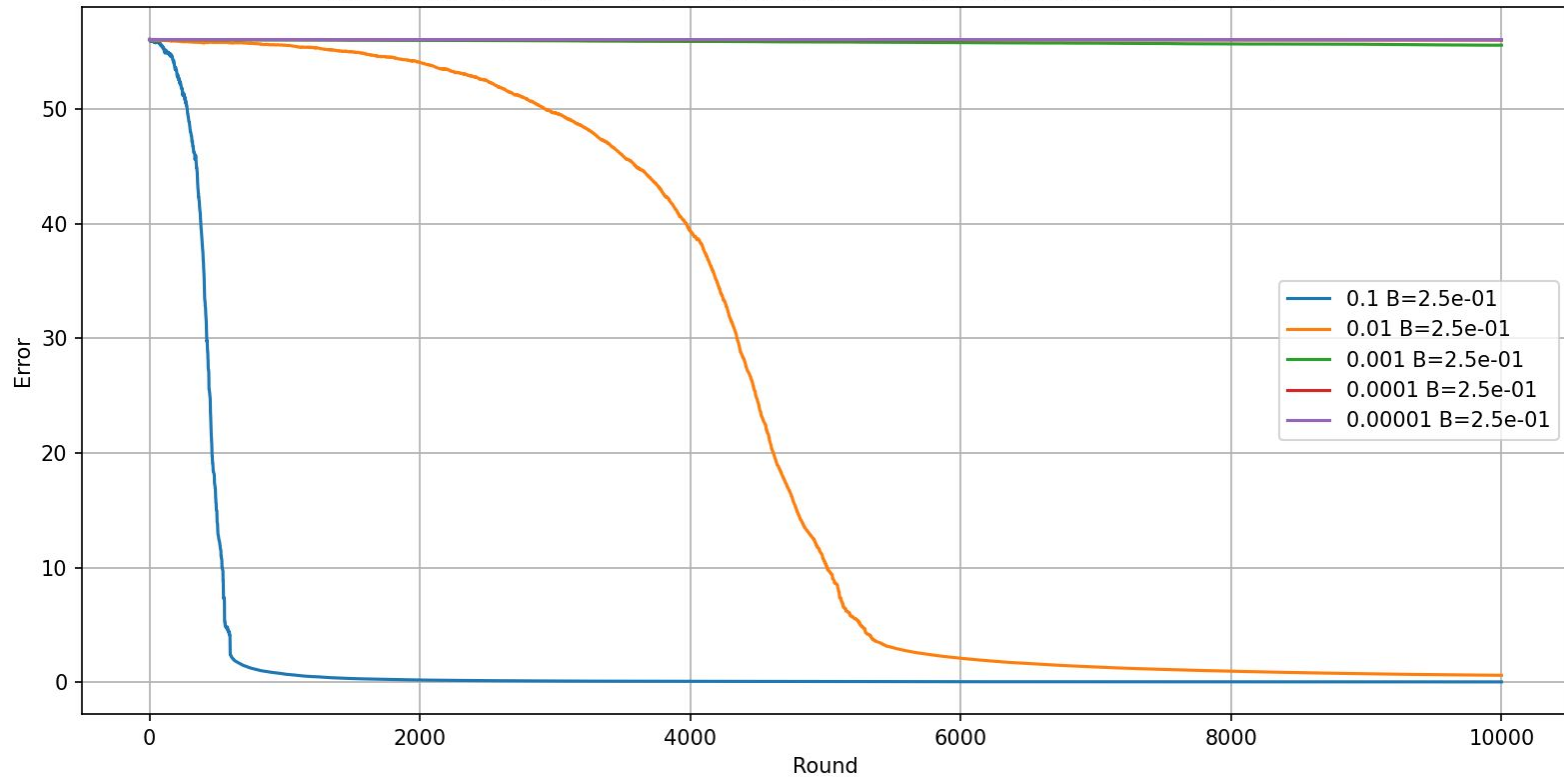
# Perceptrón Simple Lineal



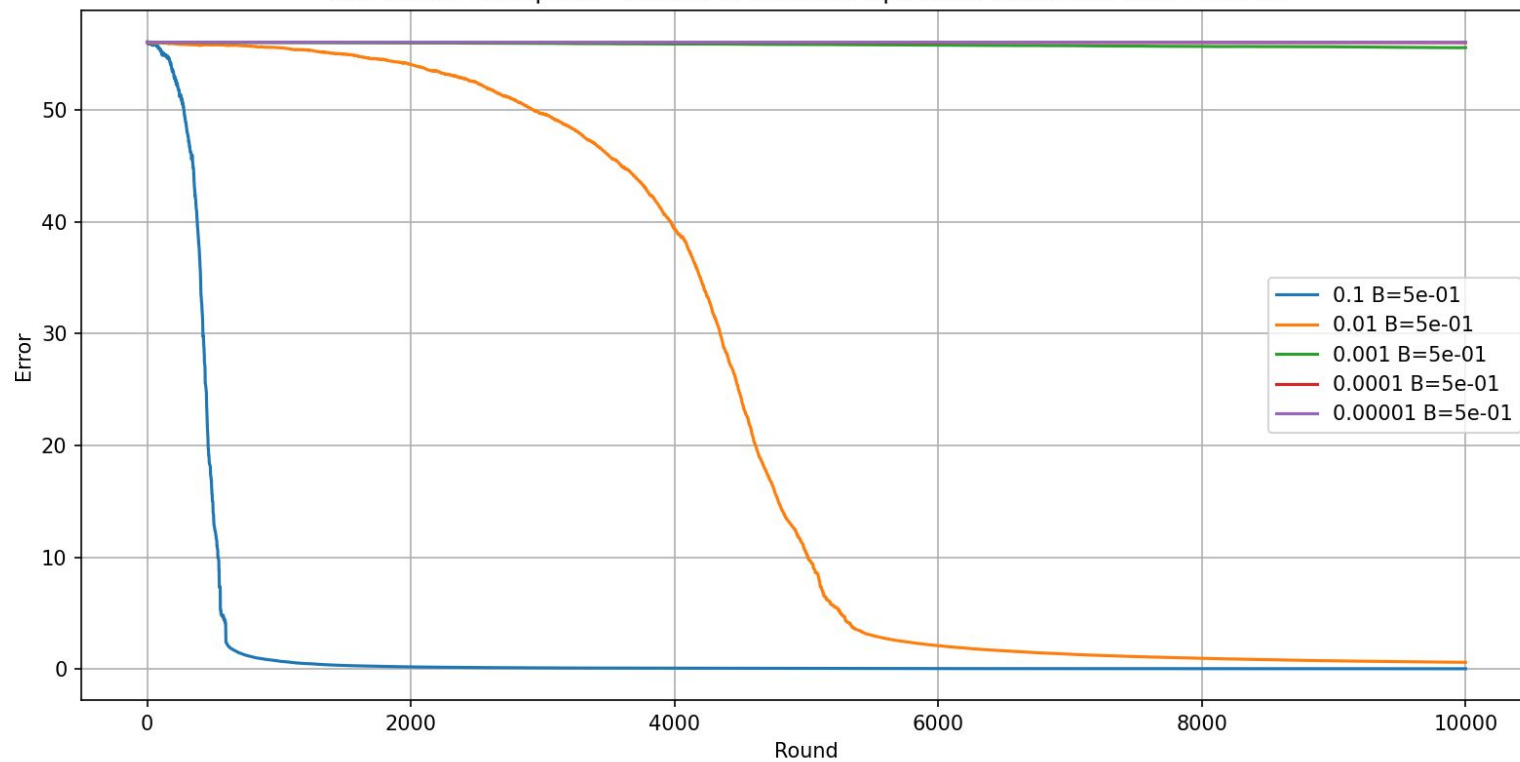
# Perceptrón Simple No Lineal - Error Distintos Beta



Non Linear Perceptron - Errors For Different Aprentice Rates And Beta = 2.5e-01



Non Linear Perceptron - Errors For Different Aprentice Rates And Beta = 5e-01



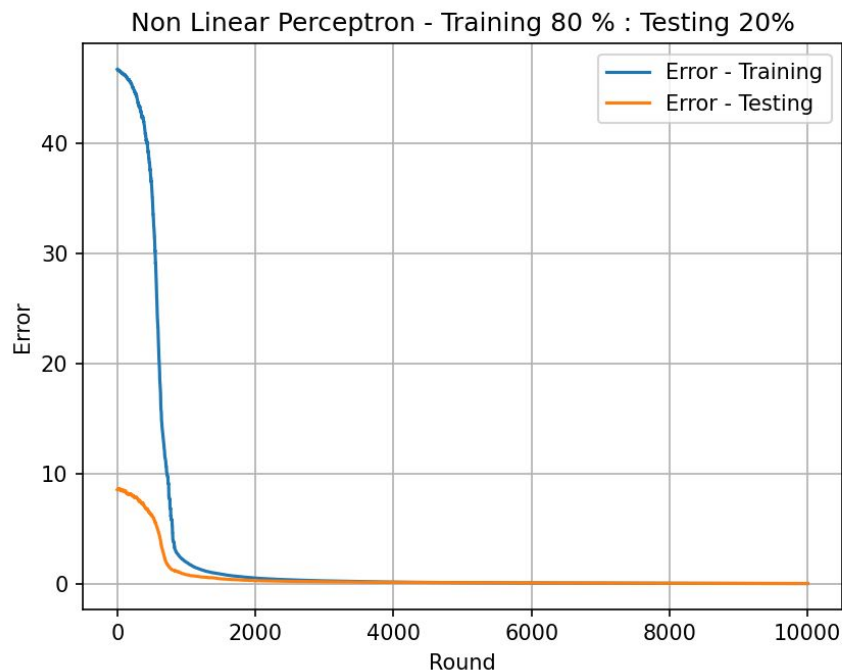
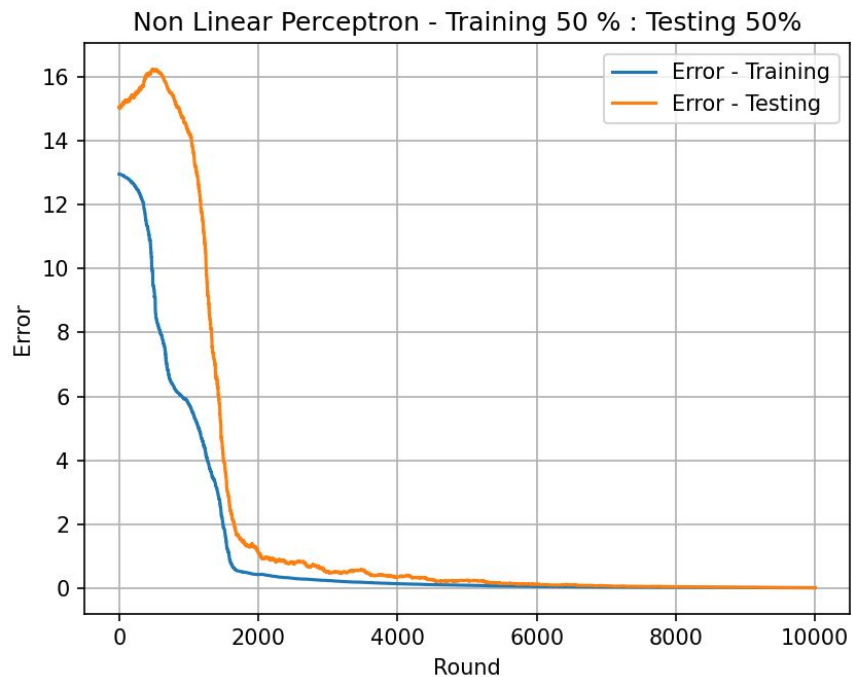


## Ejercicio 2 - B

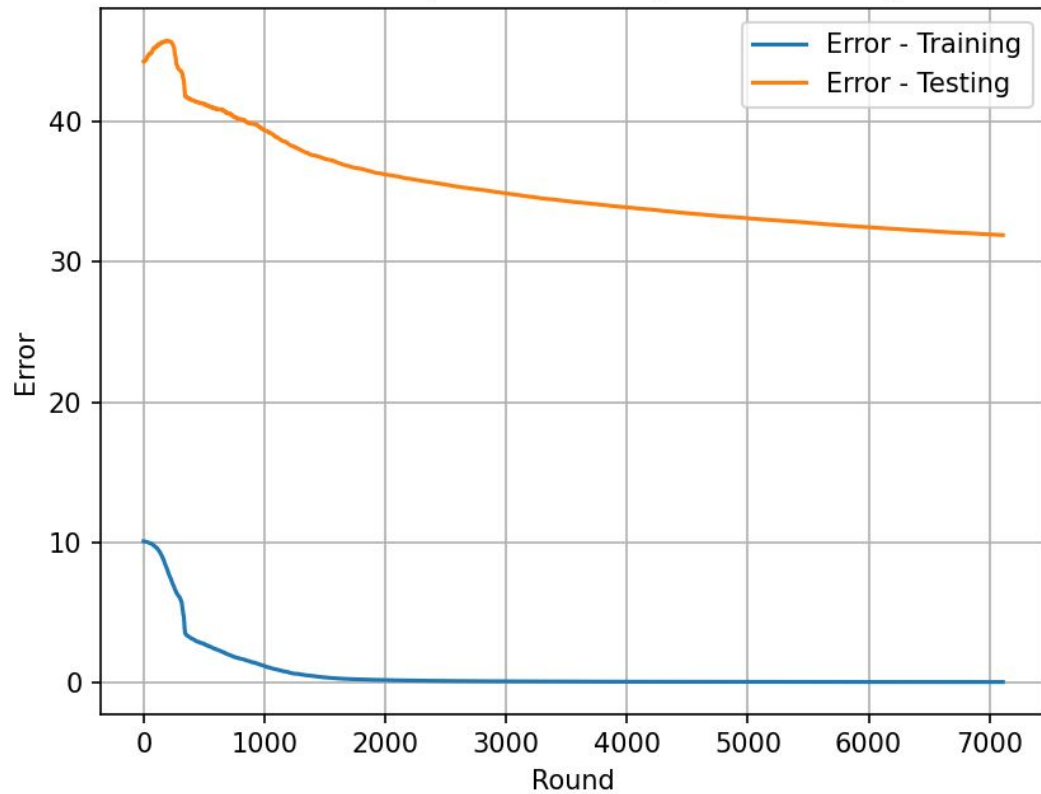




# Perceptrón Simple No Lineal - Distintos porcentajes de entrenamiento y testing




Non Linear Perceptron - Training 20 % : Testing 80%



# Conclusiones

- La tasa de aprendizaje ideal para todo tipo de perceptrones se ubica entre 0.1 y 0.01
- Para la mejor separación de datos se necesita una muestra variada y una distribución equivalente entre entrenamiento y testeo, es decir de 50% y 50% respectivamente.
- Con un entrenamiento muy grande, es decir, con un porcentaje alto, sin duda que la generalización va a ser mejor y mas especifico, sin embargo, parece haber un costo computacional muy grande para el poco testeo que se va a realizar

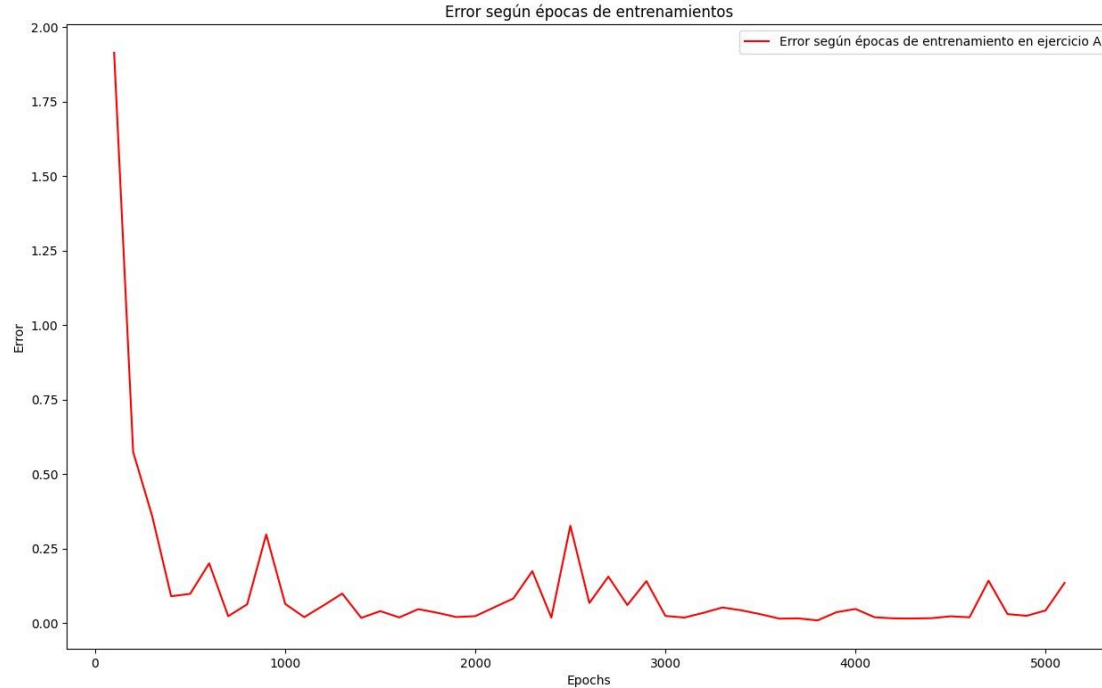


# Perceptrón Multicapa

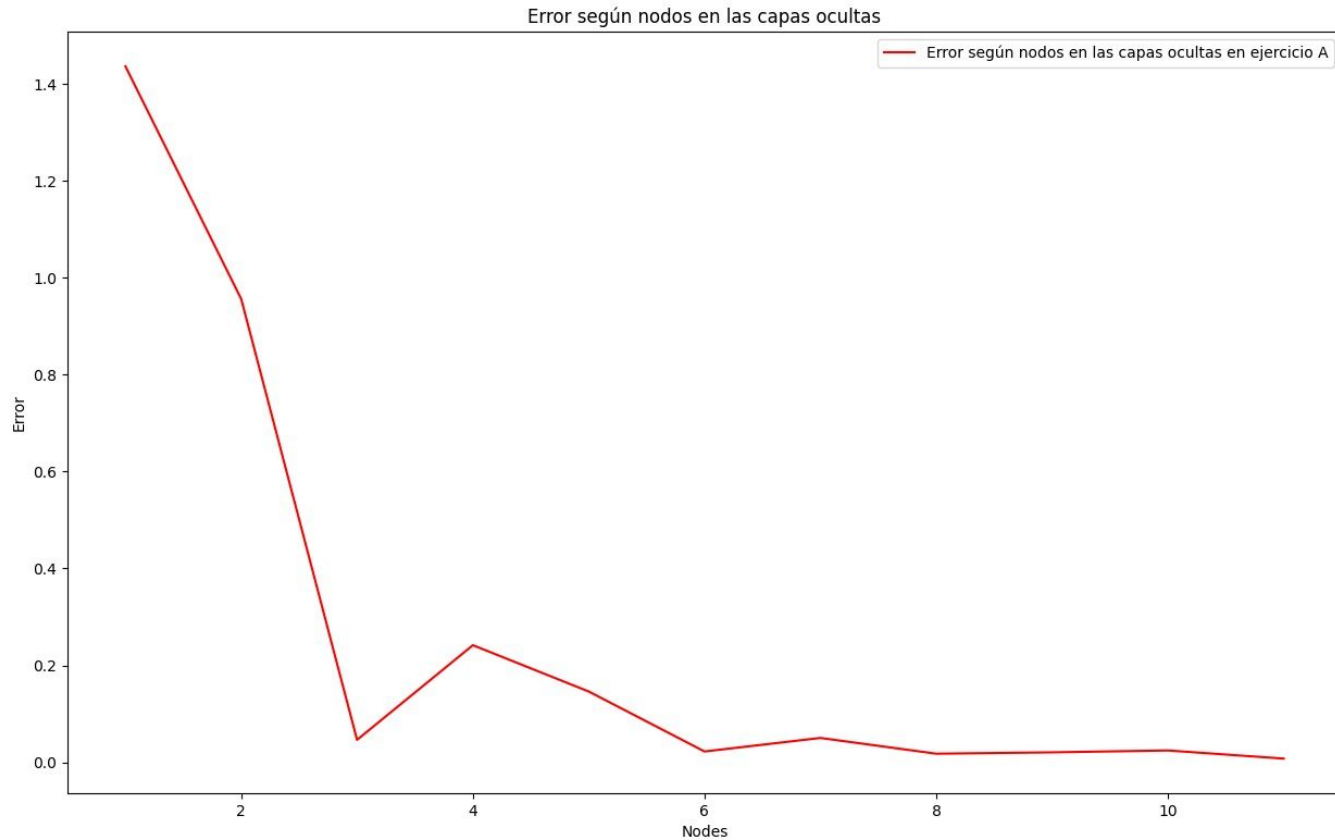
# XOR - Variables

- Nodos de salida: 1
- Nodos de capas intermedias: 6
- Capas intermedias: 6
- Función de activación: tanh
- Beta: 1
- Épocas: 5000
- Taza de aprendizaje: 0.1
- Error deseado:  $1e-5$

# XOR - Épocas vs Error



# XOR - Nodos vs Error

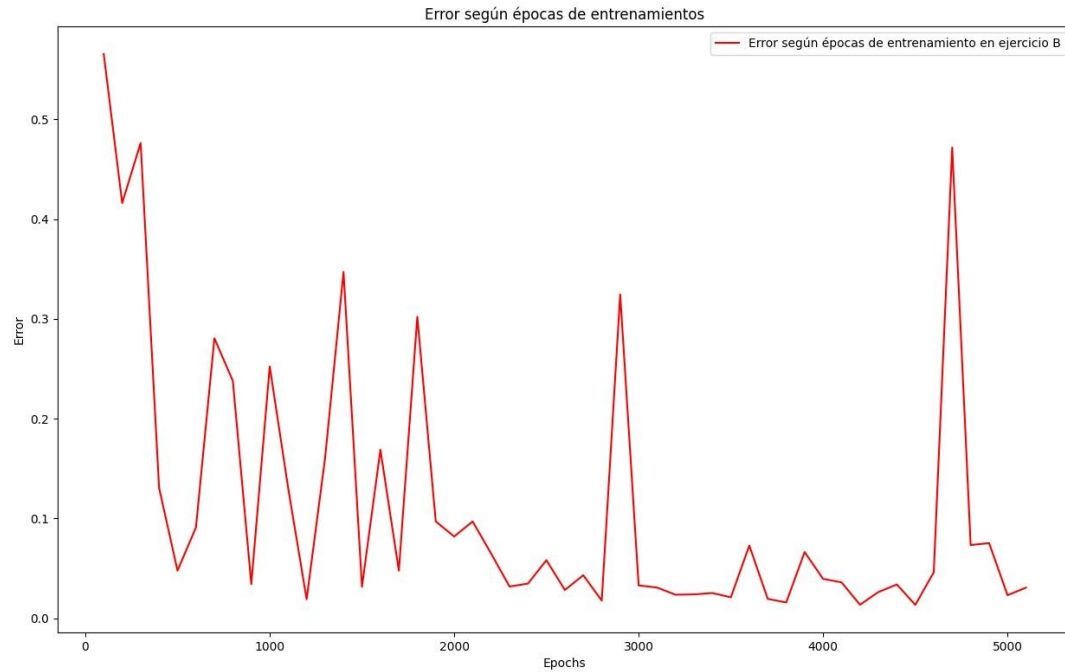


# Par - Variables

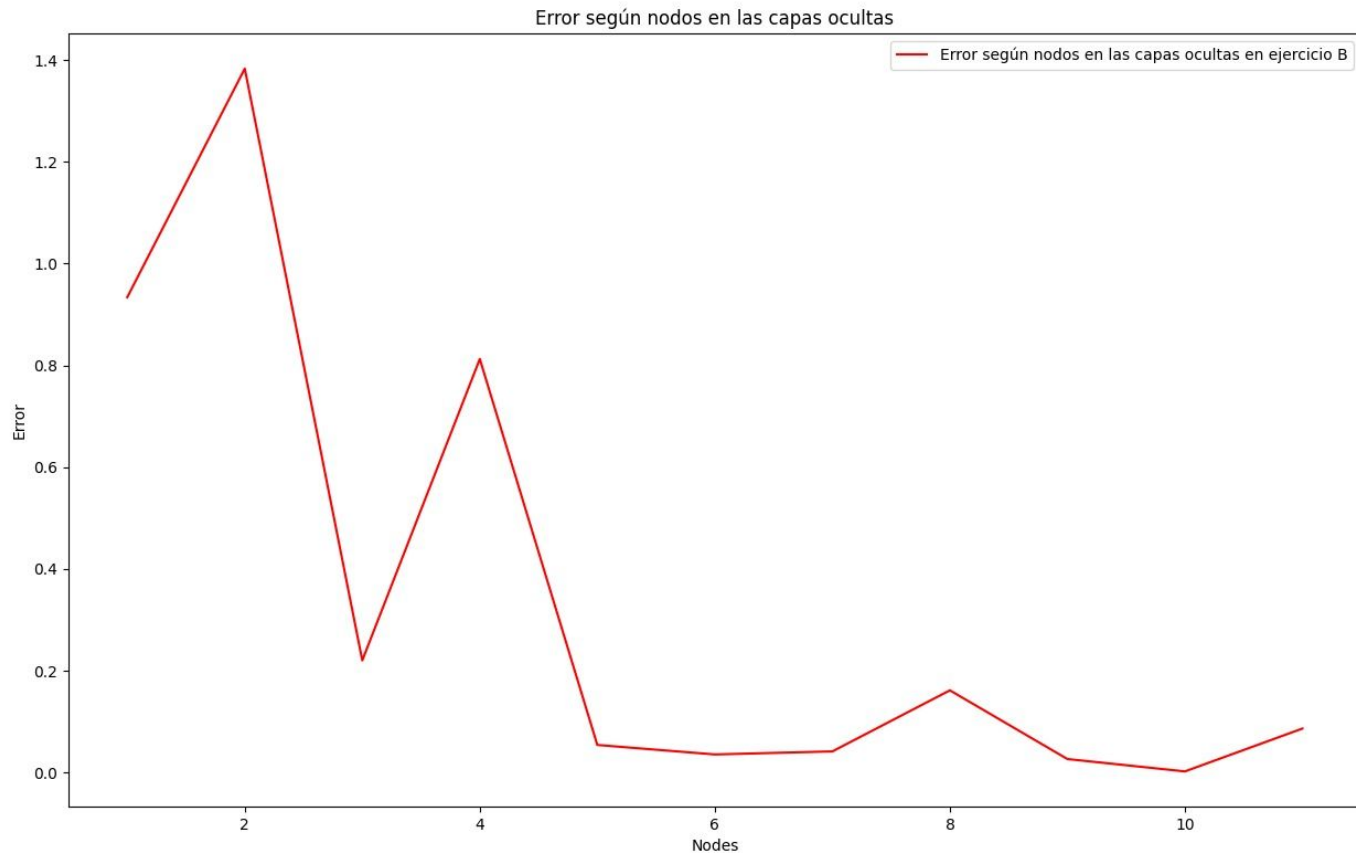
- Nodos de salida: 2
- Nodos de capas intermedias: 6
- Capas intermedias: 6
- Función de activación: tanh
- Beta: 1
- Épocas: 5000
- Taza de aprendizaje: 0.1
- Error deseado:  $1e-5$



# Par - Épocas vs Error



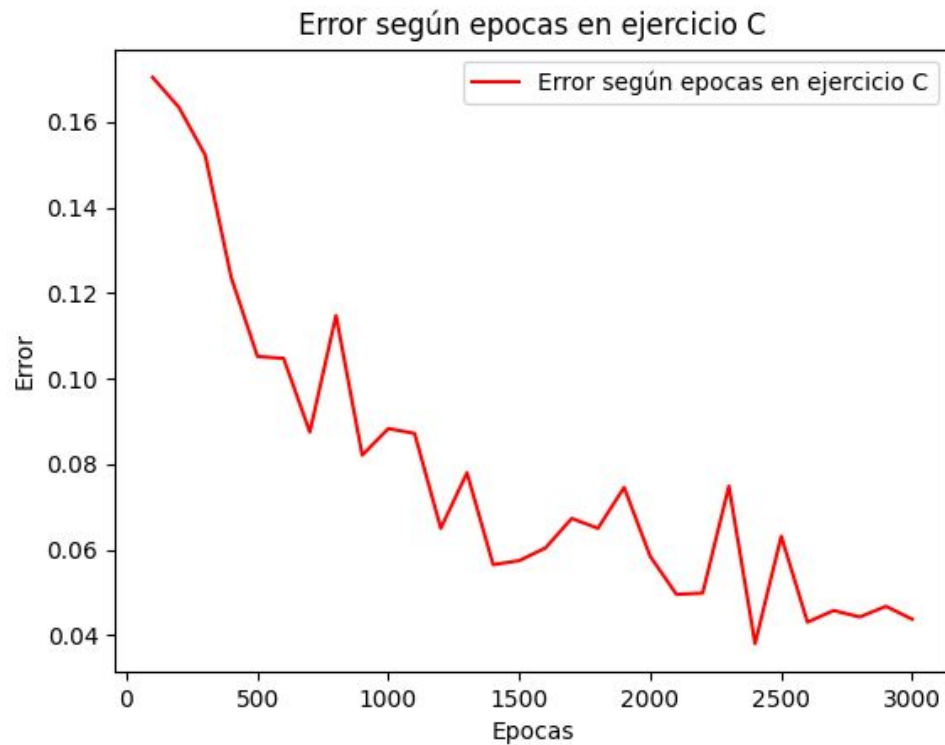
# Par - Nodos vs Error



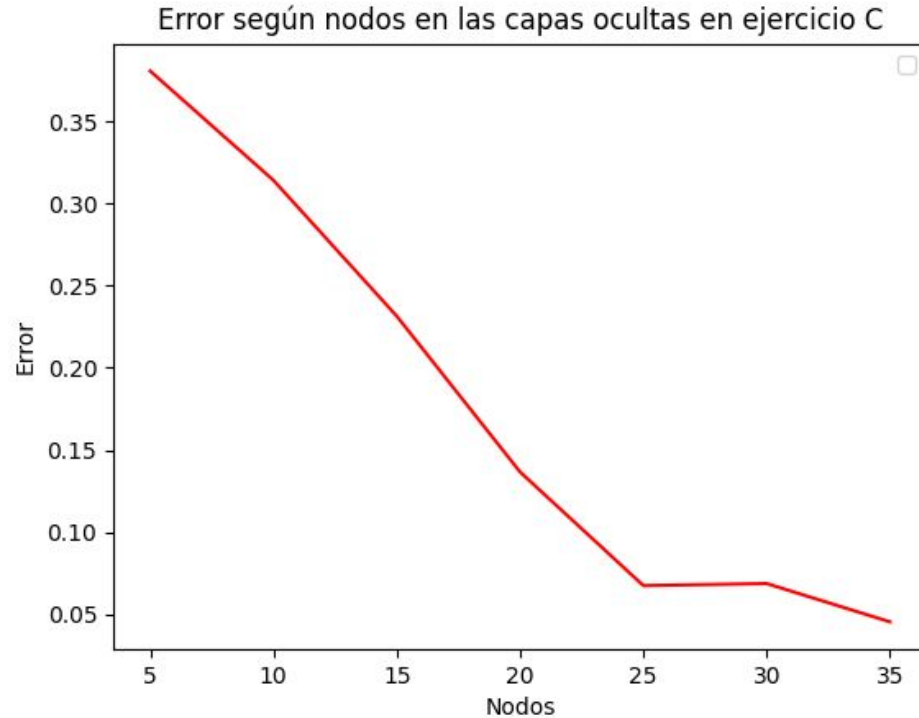
# Número - Variables

- Nodos de salida: 10
- Nodos de capas intermedias: 35
- Capas intermedias: 2
- Función de activación: tanh
- Beta:  $2,5e-1$
- Épocas: 5000
- Taza de aprendizaje: 0.1
- Error deseado:  $1e-5$

# Número - Épocas vs Error



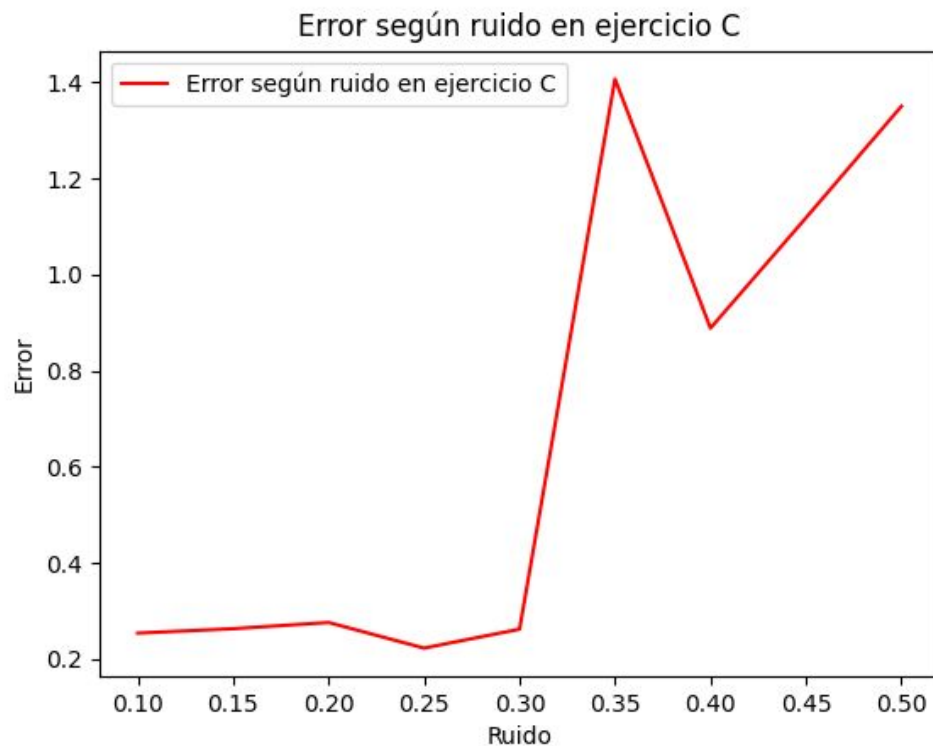
# Número - Nodes vs Error



# Número - Tasa vs Error



# Número - Ruido vs Error



# Conclusiones

- Los perceptrones multicapa pueden resolver problemas no linealmente separables.
- La cantidad de épocas reducen considerablemente el error obtenido.
- La cantidad de nodos en las capas intermedias reducen considerablemente el error obtenido.
- A mayor tasa de aprendizaje, menor error.
- A mayor ruido, mayor error.





FIN.