TP N°2

Perceptrón Simple y Multicapa

Grupo N°6

Luciana Diaz Kralj
Gonzalo Nicolás Rossin
João Nuno Diegues Vasconcelos
Mafalda Colaço Parente Morais Da Costa

E1

Perceptrón simple

Activación de escalón

Problemas a solucionar:

Clasificación binaria.

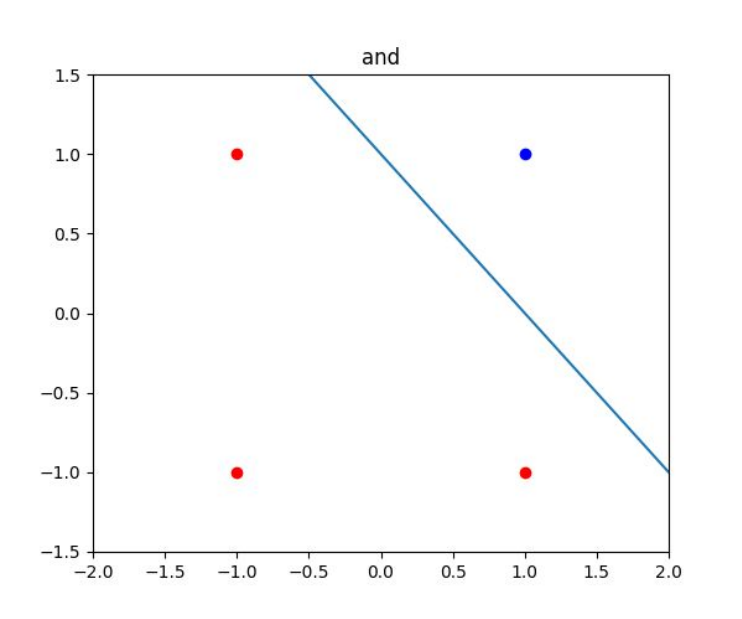
AND:

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

XOR:

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

AND

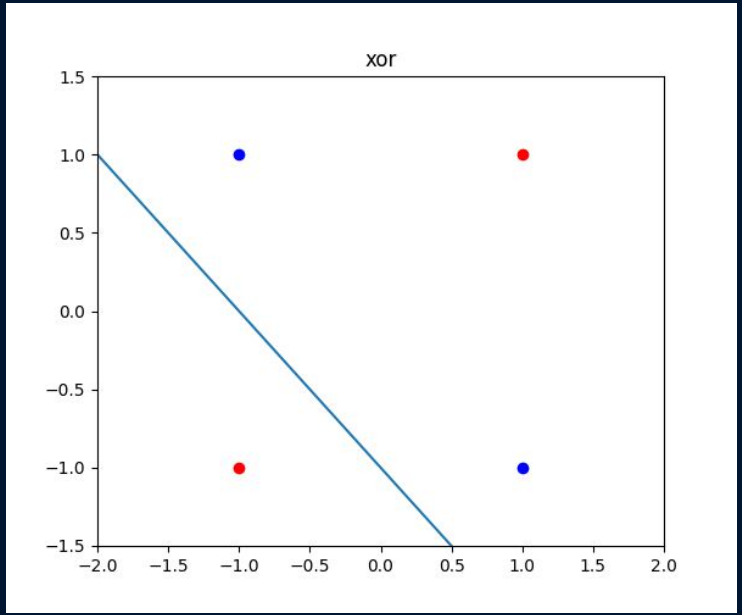


100% training set

η = 0.01

100 épocas

XOR



100% training set

η = 0.001

550 épocas

Conclusión

Se concluye que no es posible utilizar el perceptrón simple para problemas no separables linealmente, como es el caso del operador XOR.

E2

Perceptrón simple

Lineal y no lineal

Separación de conjuntos de entrenamiento y testeo

Algunas consideraciones tenidas en cuenta:

- Fuentes externas sugieren las siguientes proporciones:
  - 70% entrenamiento - 30% testeo
  - 80% entrenamiento - 20% testeo
- Importante evitar overfitting

Separación entrenamiento/testeo

Algunos problemas encontrados:

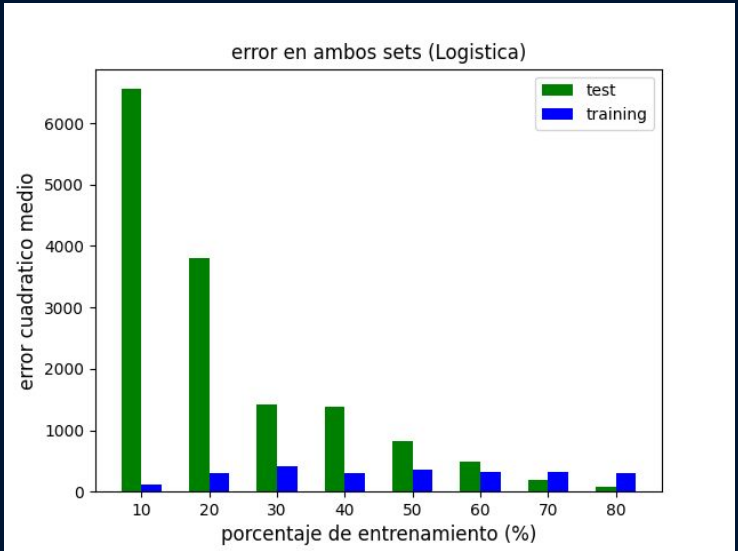
- Debido a la cantidad total de líneas del csv, la separación con porcentajes grandes para entrenamiento deja un conjunto de testeo chico. Por ejemplo:

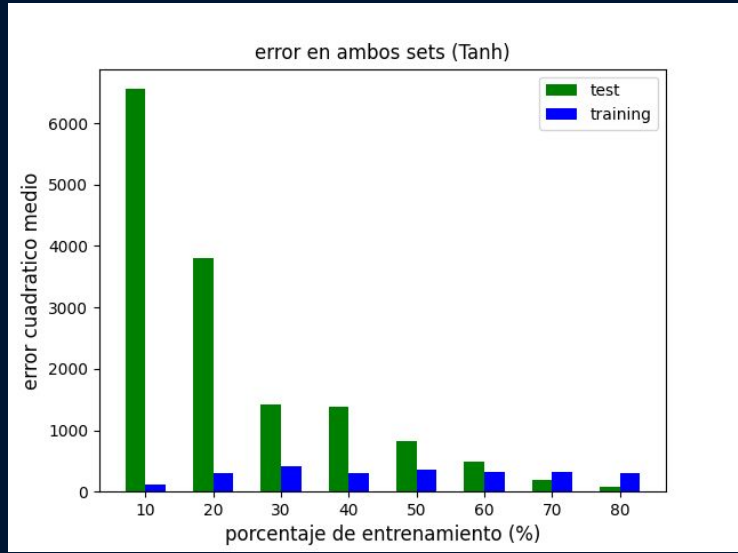
  80% entrenamiento = 23 líneas y 20% testeo = 5 líneas

- Para evitar sesgos, se decidió hacer un shuffle del input previamente a la separación entre entrenamiento y testeo.

Separación de conjuntos de entrenamiento y testeo

η = 0.0001	3.000 épocas	β = 1



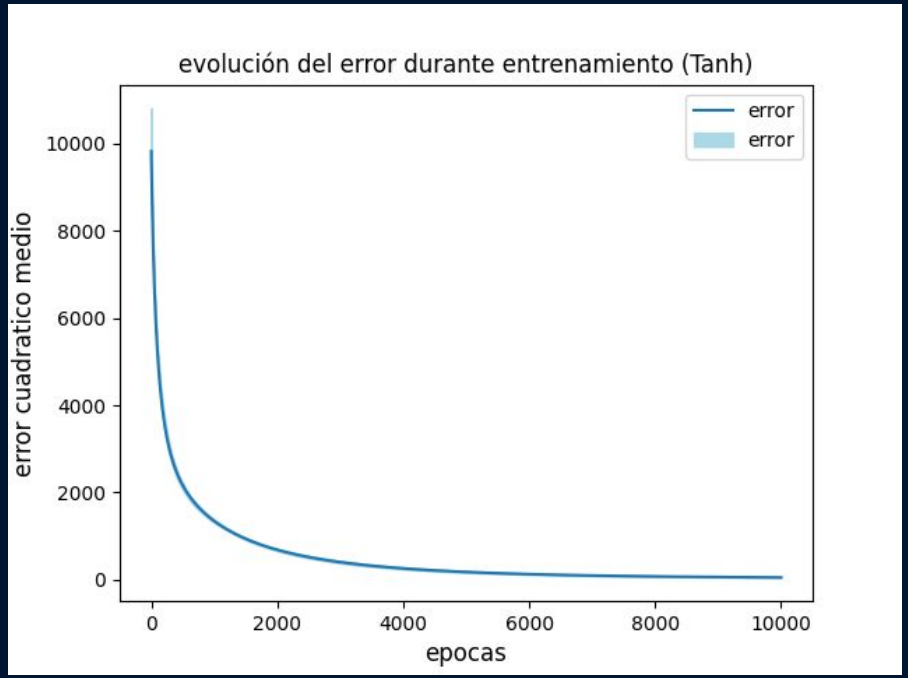


Separación de conjuntos de entrenamiento y testeo

- Como conclusión, podemos decir:
  - La **capacidad de generalización** del modelo inferido es **mayor con cuando el porcentaje de entrenamiento es del 80%** ya que el error cuadrático medio en los dos conjuntos es mínimo para ese porcentaje.
  - Si bien **para un 10% de entrenamiento, el error en el conjunto de entrenamiento es menor** que para 80%, esto no se traduce en una buena generalización ya que **el error en el conjunto de testeo es mucho mayor**.
  - Utilizar un **porcentaje de entrenamiento alto** permite que el **modelo inferido sea menos susceptible al ruido** ya que la capacidad de generalización es más grande siempre y cuando se entrene el perceptrón con valores adecuados de etha, beta e iteraciones.

Error cuadrático medio durante entrenamiento (Tanh)

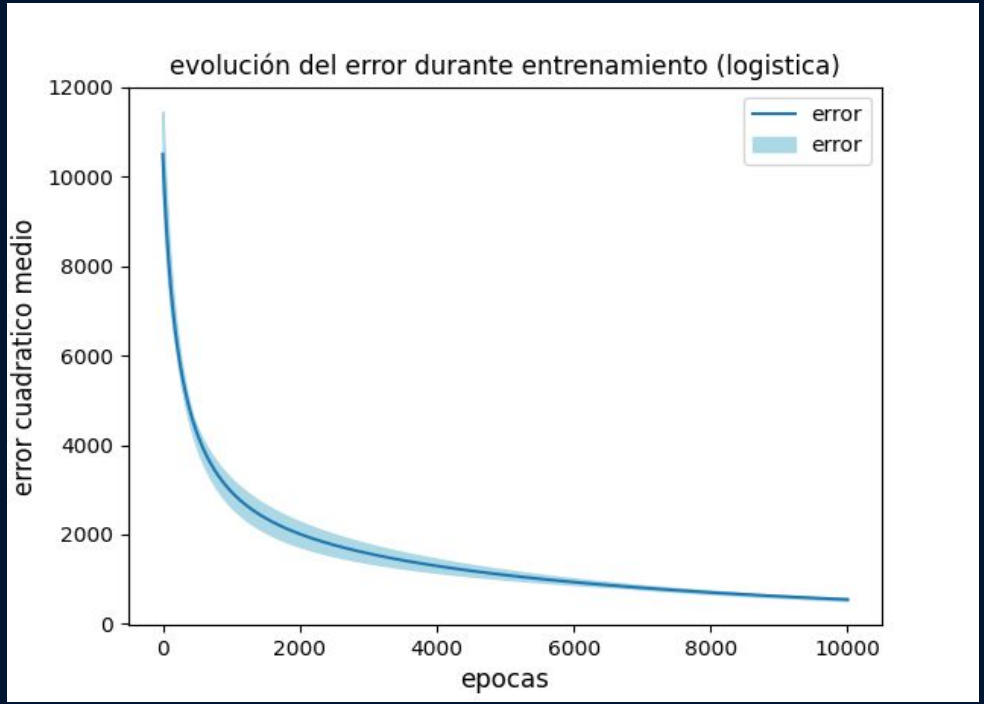
η = 0.0001	10.000 épocas	Training set 80%	β = 1



Error mínimo promedio = 56.2077

ECM durante entrenamiento (Logística)

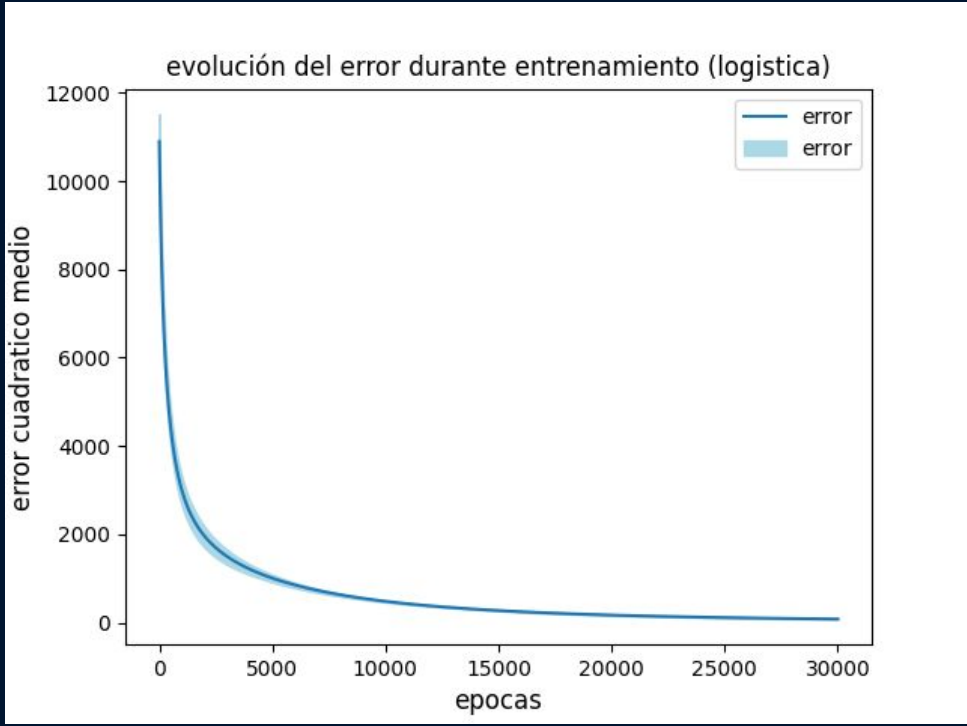
η = 0.0001	10.000 épocas	Training set 80%	β = 1



Error mínimo promedio = 543.3495

ECM durante entrenamiento (Logística)

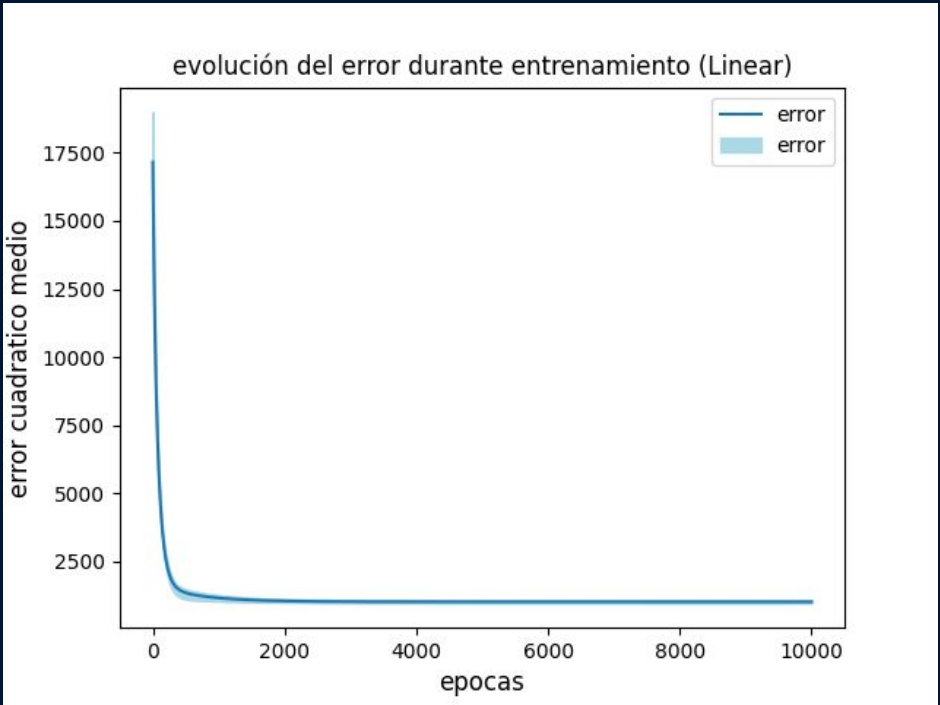
η = 0.0001	30.000 épocas	Training set 80%	β = 1



Error mínimo promedio = 81.8543

Error cuadrático medio durante entrenamiento (Linear)

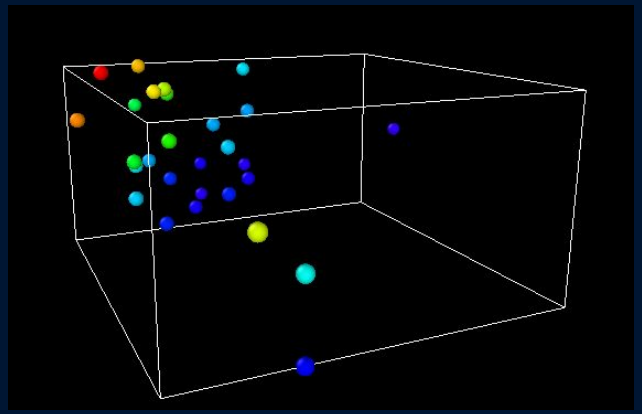
η = 0.0001	10.000 épocas	Training set 80%



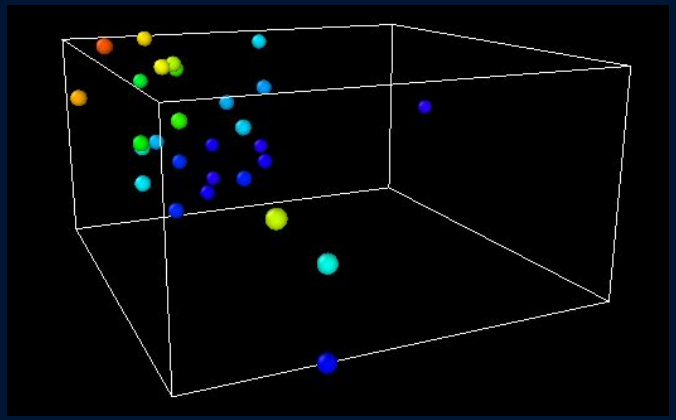
Error mínimo promedio = 1003.7464

Evolución del vector W durante el entrenamiento por épocas e incremental(aleatorio)

- Resultados esperados:



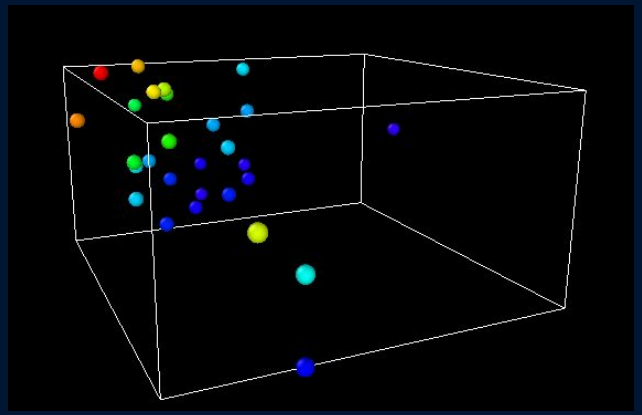
- Resultados obtenidos:



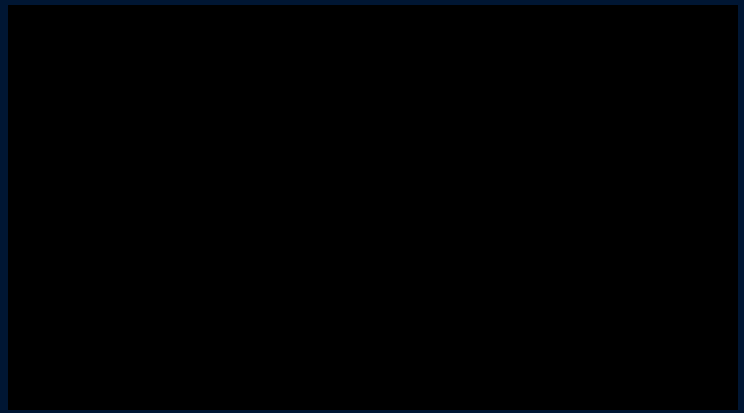
η = 0.0001	15.000 épocas	Training set 80%	Tanh	β = 1

Evolución del vector W durante el entrenamiento por épocas e incremental(aleatorio)

- Resultados esperados:



- Evolución:



η = 0.0001	20.000 épocas	Training set 80%	Tanh	β = 1

Conclusiones

Como conclusión podemos decir que:

- Para los inputs utilizados, **si bien la función de activación lineal converge más rápido** que las sigmoideas (tanh y logística), **no necesariamente eso implica que obtenga un buen resultado**. De hecho, para los mismos parámetros, la activación lineal fue la que obtuvo el error mínimo más grande entre las tres.
- Entre las funciones sigmoideas, **tanh fue la que obtuvo mejores resultados con menor cantidad de épocas** ya que para obtener un error mínimo promedio similar a tanh, la activación logística requirió tres veces la cantidad de épocas que tanh.
- El **entrenamiento incremental** (batch de 1 seleccionado aleatoriamente) **puede obtener resultados similares a entrenamiento por épocas** pero se **requiere de una cantidad mucho mayor de iteraciones** para lograrlo.

E3 | Perceptrón multicapa

Gradientes y optimizaciones

Activaciones utilizadas

- **Activación Lineal:**
  Rectified Linear Unit (ReLU).

$$\theta(x) = \begin{cases} x & si\ x > 0 \\ 0 & si\ x \leq 0 \end{cases}$$

- **Activación No Lineal:**
  Sigmoidea Logística.

$$\theta(x) = \frac{1}{1 + e^{-x}}$$

3A: Problema XOR

No es linealmente separable:

- Activación ReLU en las capas ocultas.
- Activación sigmoidea logística en la capa de salida.

Al tener un dataset pequeño, entrenamos y predecimos sobre el 100% de los datos.

3A: Resultados Problema XOR

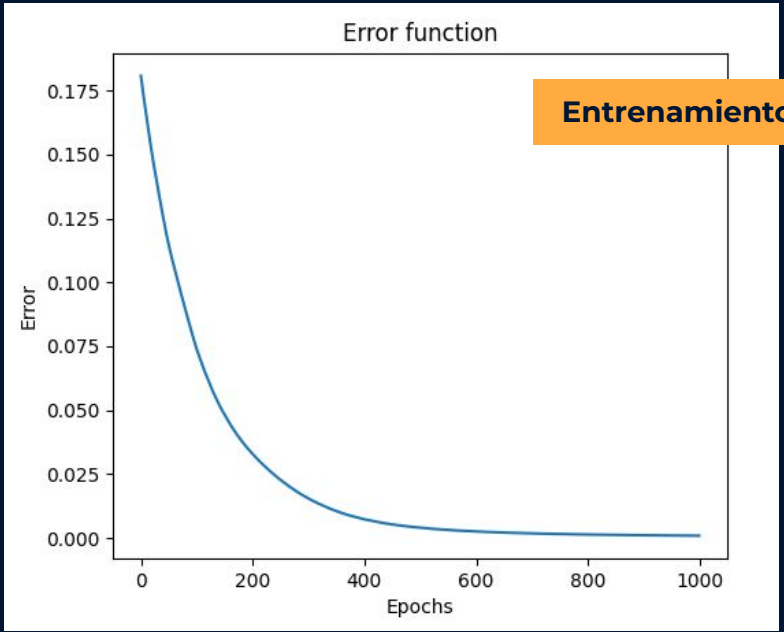
$\eta = 0.1$	1.000 épocas	Full batch	Capas = [2, 10, 10, 1]

ξ_1	ξ_2	Valor Esperado ζ	Valor Obtenido O
-1	1	1	0.9683
1	-1	1	0.9735
-1	-1	-1	-0.9699
1	1	-1	-0.9850

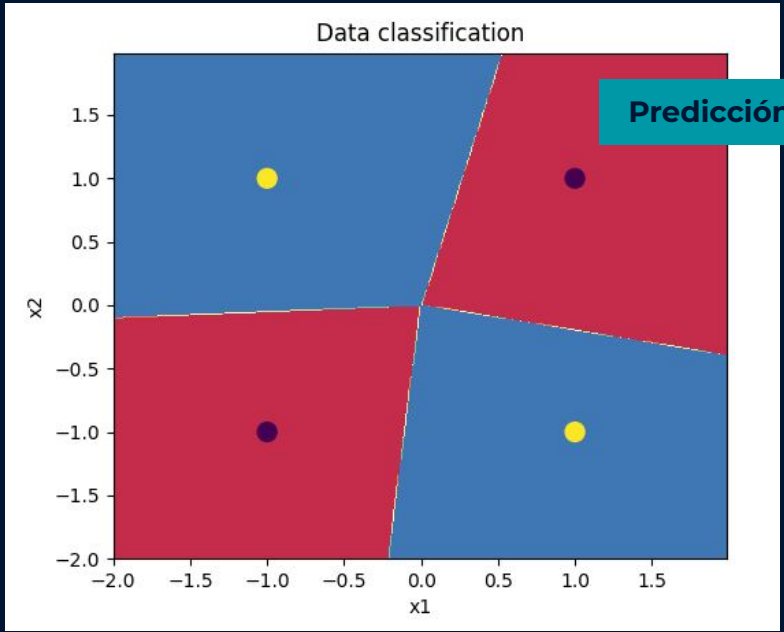
$$(\zeta - O)_{promedio} \triangleq 0.0033$$

3A: Resultados Problema XOR

η = 0.1	1.000 épocas	Full batch	Capas = [2, 10, 10, 1]



Entrenamiento



Predicción

3B: Paridad de Mapa de Bits

- **Input:** Mapas de bits de 7 filas, 5 columnas.
  ⇒ Aplanamos a **vectores columna de 35 elementos**.
- **Output:** Asignamos 1 si es par, 0 si es impar.
- Activación **ReLU** en las **capas ocultas**.
- Activación **Sigmoidea Logística** en la **capa de salida**.

3B: Resultados Paridad Mapa de Bits

100% training set

η = 0.1

1000 épocas

Full batch

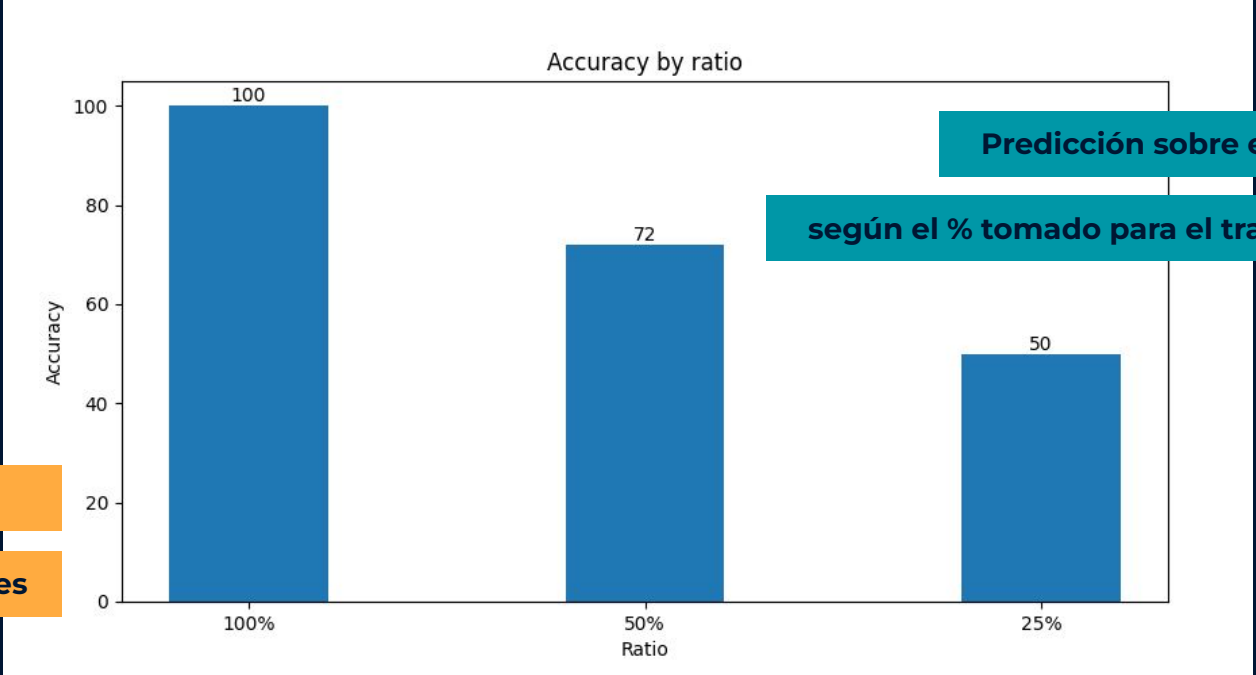
Capas = [35, 10, 10, 1]

$(\zeta - O)_{promedio} \cong 0{,}0136$

ξ	Valor Esperado ζ	Valor Obtenido O
0	1	0.9127
1	0	0.3809
2	1	0.8330
3	0	0.1326
4	1	0.9794
5	0	0.0388
6	1	0.9507
7	0	0.0196
8	1	0.7576
9	0	0.1297

3B: Precisión Paridad Mapa de Bits

η = 0.1	1.000 épocas	Full batch	Capas = [35, 10, 10, 1]



Predicción sobre el test set

según el % tomado para el training set

Promedio

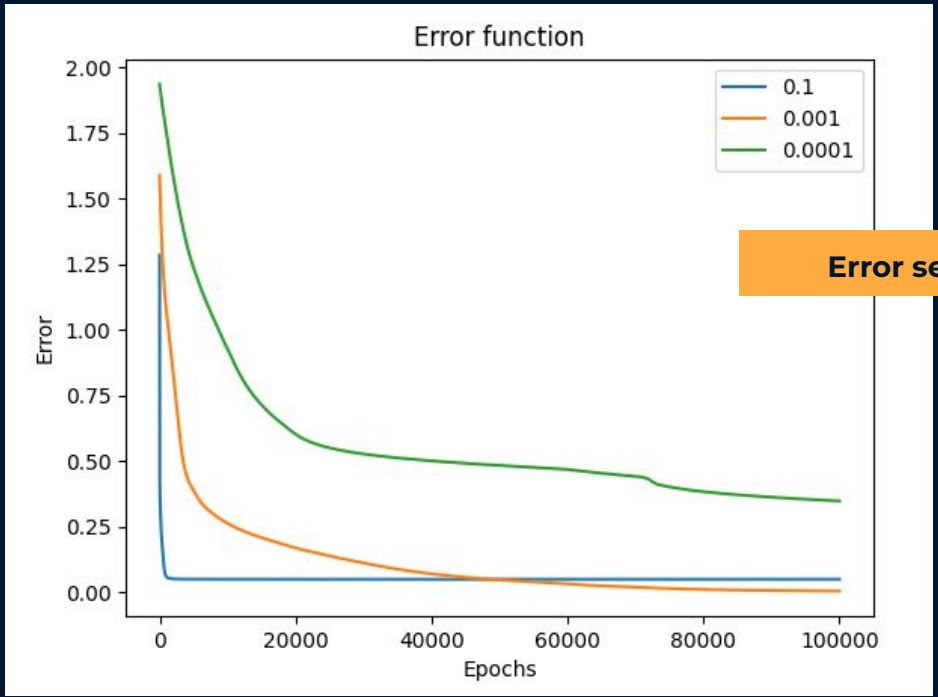
5 ejecuciones

3C: Identificación de Números

- **Input:** Mapas de bits de 7 filas, 5 columnas.
  ⇒ Aplanamos a **vectores columna de 35 elementos**.
- **Output:** Son **10 clases**, una por cada número del 0 al 9.
- Activación **ReLU** en las **capas ocultas**.
- Activación **Sigmoidea Logística** en la **capa de salida**.

3C: Resultados Identificación de Números

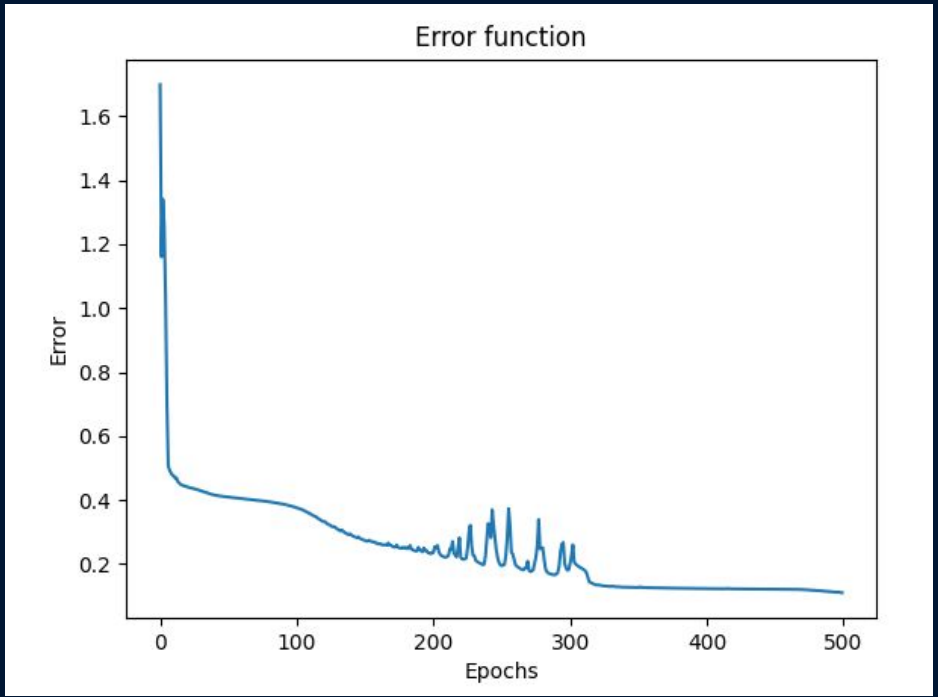
100.000 épocas	Full batch	Capas = [35, 10, 10, 10]



Error según η

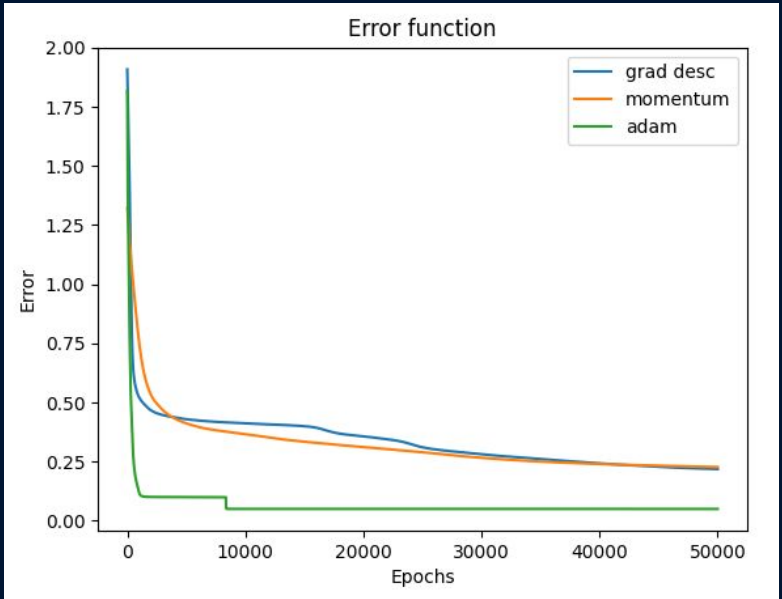
3C: Efecto de un mal η

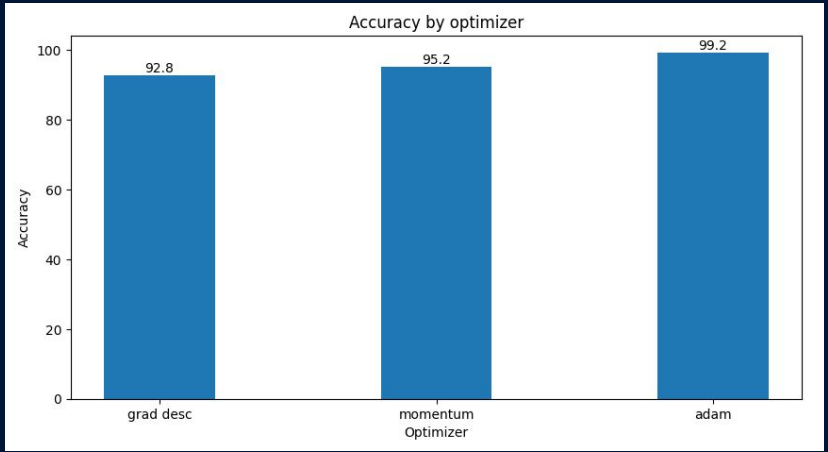
η = 0.9	500 épocas	Full batch	Capas = [35, 10, 10, 10]



3C: Efecto de Optimizadores (1)

η = 0.001	50.000 épocas	Full batch	Capas = [35, 10, 10, 10]

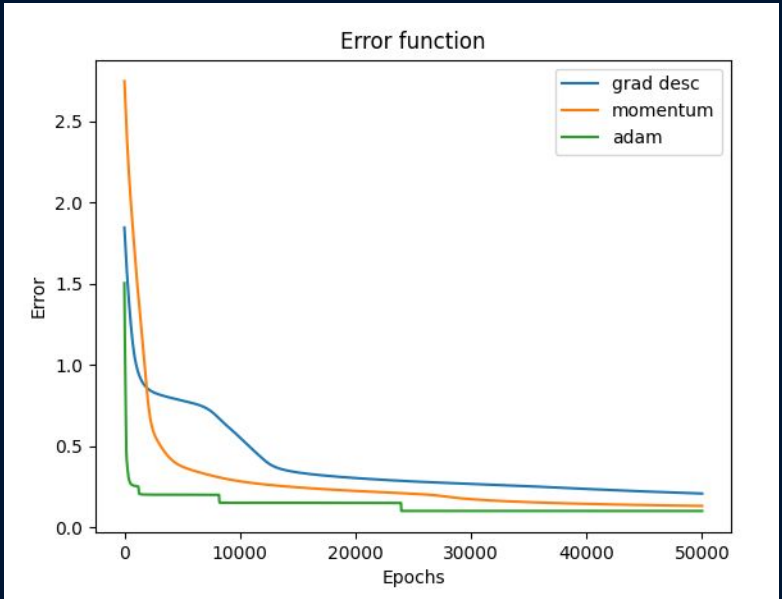


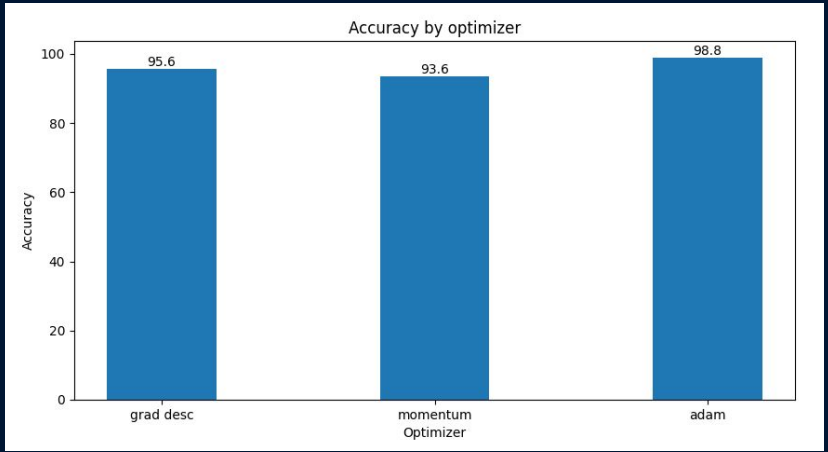


Promedio
5 ejecuciones

3C: Efecto de Optimizadores (2)

η = 0.001	50.000 épocas	Full batch	Capas = [35, 10, 10]

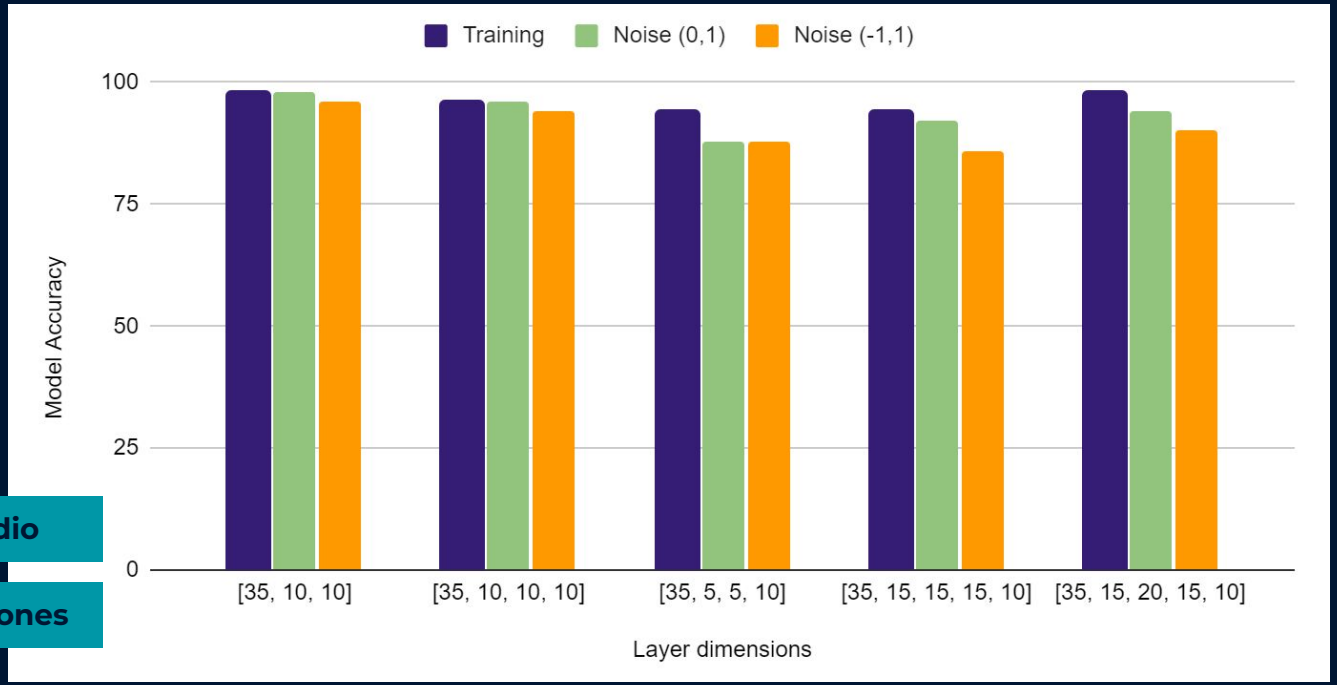




Promedio
5 ejecuciones

3C: Arquitectura y Generalización

η = 0.001	50.000 épocas	Full batch	Optimizador Adam



Promedio

5 ejecuciones

Conclusiones

Podemos decir que:

- Existen **trade-offs** con η:
  - Con una **tasa de aprendizaje alta se converge más rápido** en el mejor de los casos, **o se diverge**, en el peor.
  - Una **tasa de aprendizaje baja tarda más en converger**, pero **puede alcanzar mejores resultados**.
- Los **métodos de optimización** ayudan a **escapar de los malos mínimos locales** o ensilladuras.
- Entrenar **más neuronas y capas ocultas implica tener un mayor poder computacional**, y los resultados no siempre son mejores.

¡GRACIAS POR ESCUCHAR!

GRUPO N°6

Luciana Diaz Kralj
Gonzalo Nicolás Rossin
João Nuno Diegues Vasconcelos
Mafalda Colaço Parente Morais Da Costa