Trabajo práctico 1
Redes Neuronales y Aprendizaje Profundo

Ignacio Ezequiel Cavicchioli
Padrón 109428
icavicchioli@fi.uba.ar

10/9/2025

Índice

1. Introducción

Este documento presenta el desarrollo de las consignas del trabajo práctico N°2 de la materia de **Redes Neuronales y Aprendizaje Profundo**. El código correspondiente fue realizado en *Jupyter notebooks*, *Python*, adjuntados a la entrega en formato PDF. Toda imagen o implementación requeridas para el análisis se explicitarán en el presente archivo, por lo que la lectura del código en sí queda a discreción del lector. La teoría relevante será presentada y discutida en la sección pertinente.

2. Ejercicio 1

2.1. Consignas

Implemente un perceptrón simple que aprenda la función lógica AND y la función lógica OR, de 2 y de 4 entradas. Muestre la evolución del error durante el entrenamiento. Para el caso de 2 dimensiones, grafique la recta discriminadora y todos los vectores de entrada de la red.

2.2. Desarrollo

Lo primero que se hizo fue generar el dataset para entrenar los perceptrones, que no es más que la tabla de verdad de la función lógica que se quiere aprender.

Las figuras 1, 2,3 y 4 muestran el error en función de la iteración (de entrenamiento) para los perceptrones que emulan las funciones AND y OR de 2 y 4 entradas. Como se discutió en la teórica, el perceptrón es capaz de aprender estas funciones lógicas simples, por lo que el error final del entrenamiento es cero.

La figura 6 muestra la frontera de decisión del perceptrón que aprendió la función OR, y la 5 muestra la frontera para la compuerta AND. Los puntos del mismo color son de la misma clase (0 o 1), y son adecuadamente segregados por la frontera. Contrario a SVM, el perceptrón simple no tiene la distancia desde la frontera hasta las muestras en su función de costo, por lo que la recta que separa las clases no es única.

Para encontrar la fórmula de la recta que hace de frontera de decisión se puede partir de la expresión de la sumatoria de los X con sus respectivos pesos y el bias, y suponer que $Y = 0$, lo que significa que estás parado sobre la frontera, tu muestra no es ni de una clase ni de la otra.

$$X1 \cdot w1 + X2 \cdot w2 + b = 0$$

Si suponemos que $X1$ es nuestra abscisa que vamos a barrer y $X2$ es la ordenada, la recta toma la forma de:

$$X2 = \frac{w1 \cdot X1 + b}{-w2}$$

Y entonces $m = -w1/w2$ y b $= -b/w2$

2.3. Análisis

Los perceptrones lograron aprender sus funciones lógicas designadas sin error y en una cantidad finita de iteraciones del algoritmo. Esto es el resultado esperado, se sabe que puede aprender este tipo particular de problemas.

Una parte interesante del ejercicio fue el armado del algoritmo de gradiente descendiente, regla por la cual el perceptrón va moviendo su frontera de decisión. Los gráficos muestran un error decreciente en el tiempo, que a veces aumenta espuriamente para luego ser corregido, porque el gradiente descendiente va actualizando los pesos en función del error, intentando moverse hacia fronteras que predigan correctamente.

La obtención de la recta que hace de frontera en 2D solo requirió del desarrollo de fórmulas hecho en la sección previa.

En líneas generales, el ejercicio sirvió como una buena introducción a los perceptrones y el concepto de gradiente descendiente.

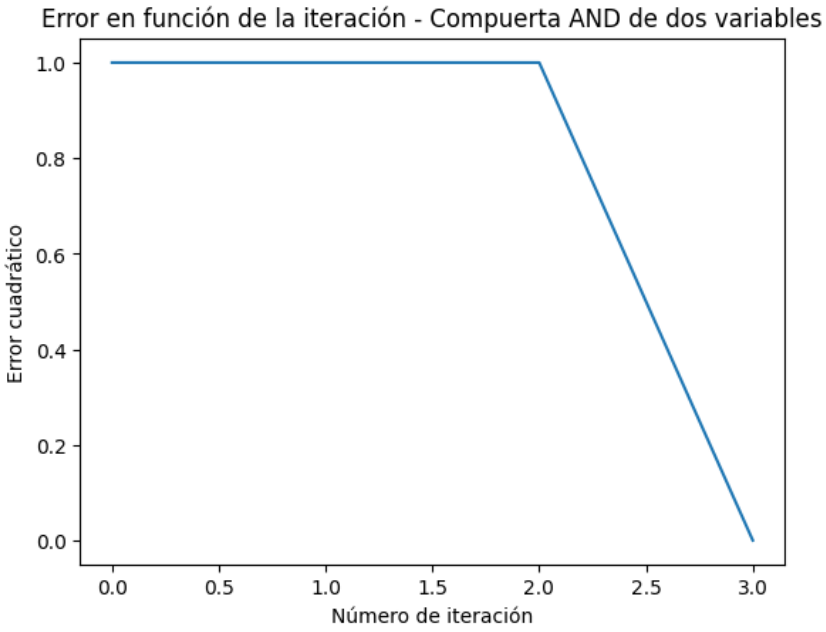

Figura 1: Error de entrenamiento en el tiempo para compuerta AND de 2 entradas

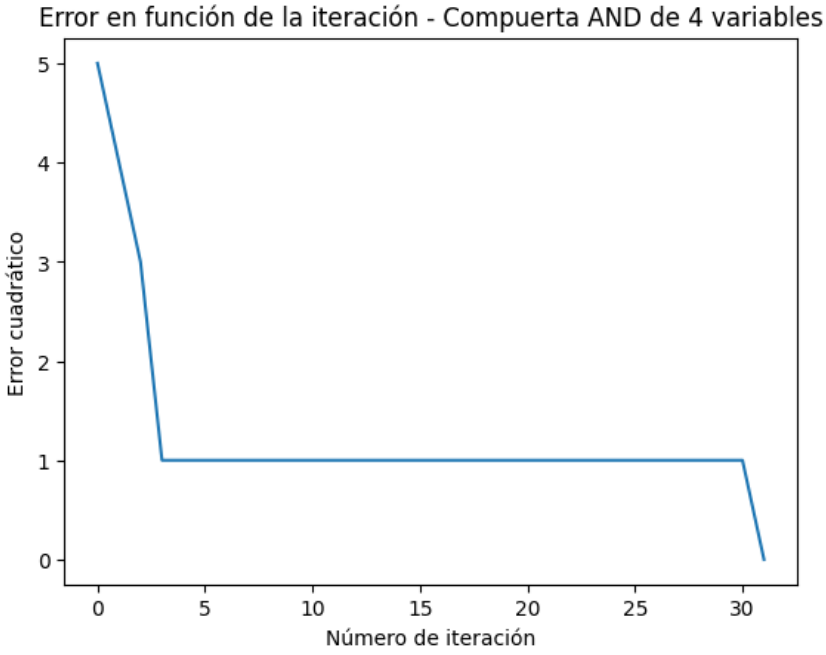

Figura 2: Error de entrenamiento en el tiempo para compuerta AND de 4 entradas

Figura 3: Error de entrenamiento en el tiempo para compuerta OR de 2 entradas

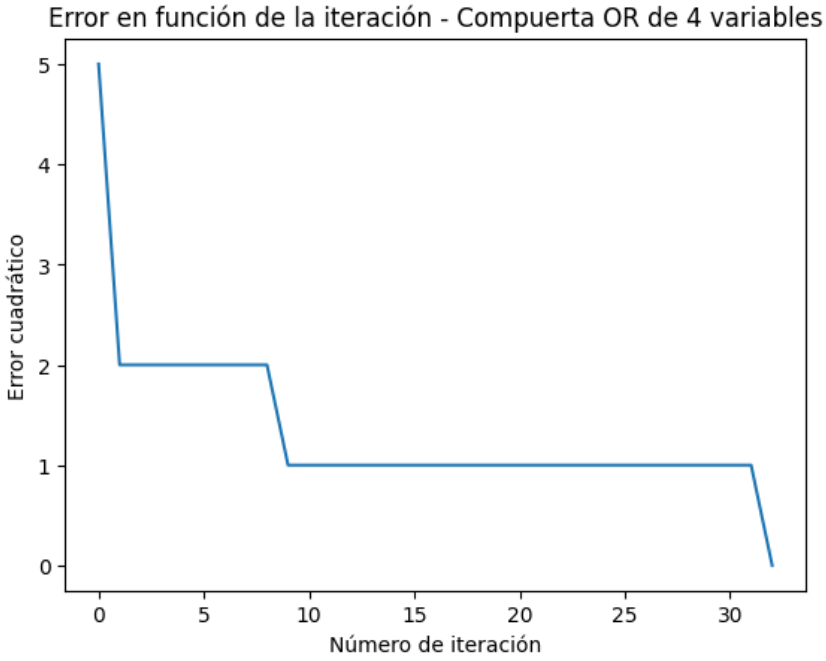

Figura 4: Error de entrenamiento en el tiempo para compuerta OR de 4 entradas

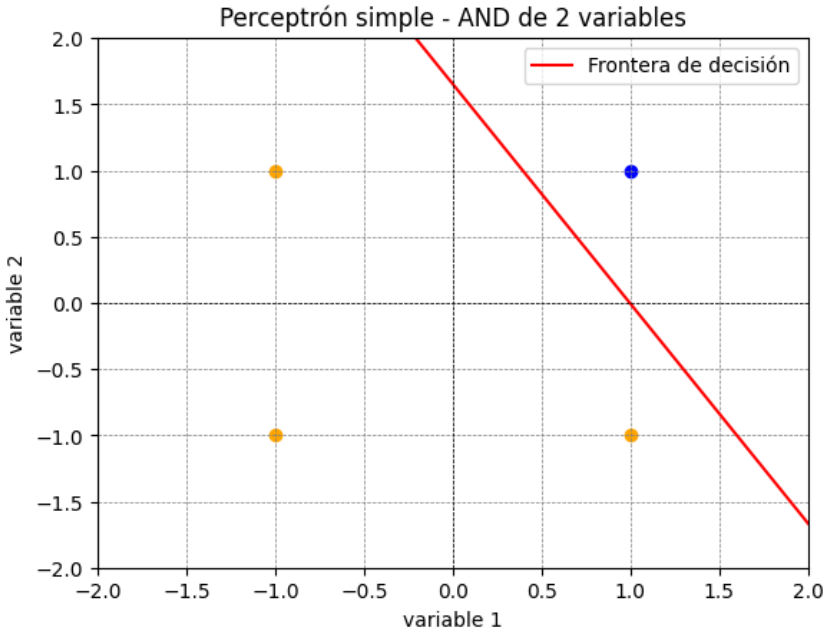

Figura 5: Frontera encontrada para compuerta AND de 2 entradas

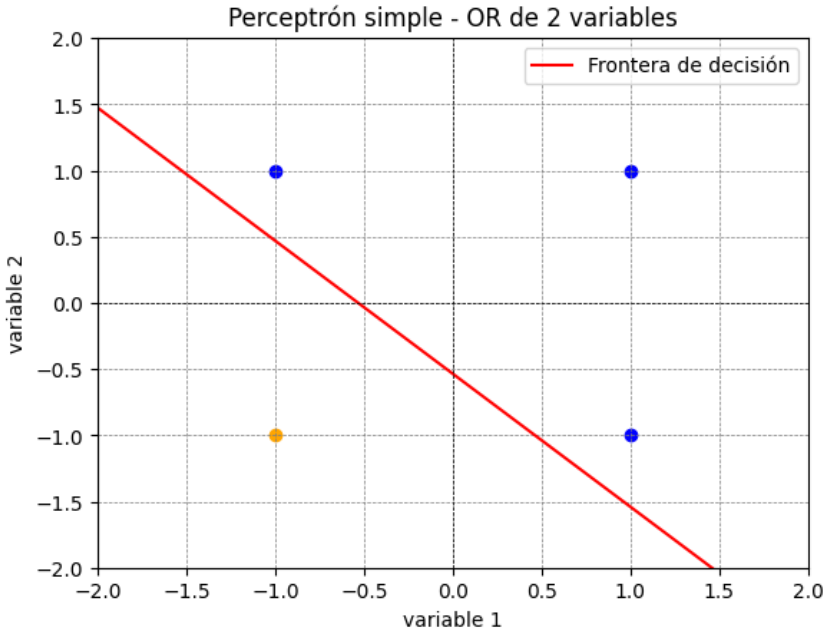

Figura 6: Frontera encontrada para compuerta OR de 2 entradas

3. Ejercicio 2

3.1. Consignas

Determine numéricamente cómo varía la capacidad del perceptrón simple en función del número de patrones enseñados.

3.2. Desarrollo

La capacidad de un perceptrón está definida en la p.111 (expresión 5.63) del libro "Introduction to the theory of neural computation". La idea es básicamente determinar la cantidad de patrones aleatorios (con etiqueta aleatoria) para el cual existe una frontera. En el libro se demuestra que la capacidad de almacenamiento máxima del perceptrón converge a $p_{max} = 2N$, con p la cantidad de patrones y N, el número de pesos sinápticos que tiene el perceptrón (o cantidad de unidades de entrada), con $N \to$ ínf.

Para corroborar experimentalmente esta aserción, se entrenó un perceptrón de un N particular con cada vez más patrones y se registró el resultado, para luego graficarlo.

En pocas palabras, el código genera conjuntos de patrones aleatorios y entrena un perceptrón simple para clasificarlos. Repite el experimento múltiples veces[1] y calcula con qué frecuencia el perceptrón logra aprender sin errores (es decir, converge). Luego grafica la probabilidad de éxito en función de $\alpha = \frac{p}{N}$, donde p es la cantidad de patrones y N la dimensión de entrada, mostrando así la capacidad del perceptrón. Se espera que la recta vertical $p/N = 2$ interseque la gráfica en $p = 1/2$ (como en la literatura citada).

los resultados para $N = 30$ se pueden ver en la figura 7, y para $N = 10$, en la 8. No se hizo con más unidades de entrada porque condiciona mucho el tiempo de ejecución y la cantidad de epochs necesarias para lograr un buen resultado.

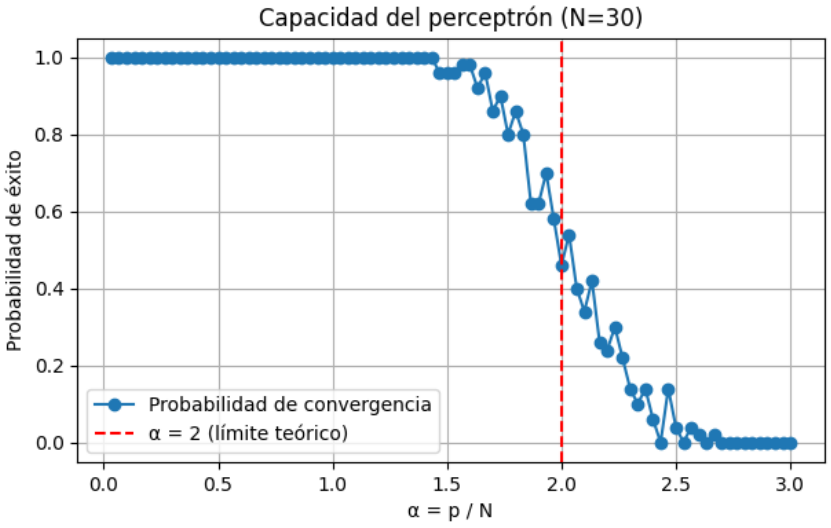

Figura 7: Ensayo de capacidad $N = 30$

[1] Para un cierto p se hacen varios experimentos y se promedia, por eso hay una cantidad de unidades de entrada y de repeticiones

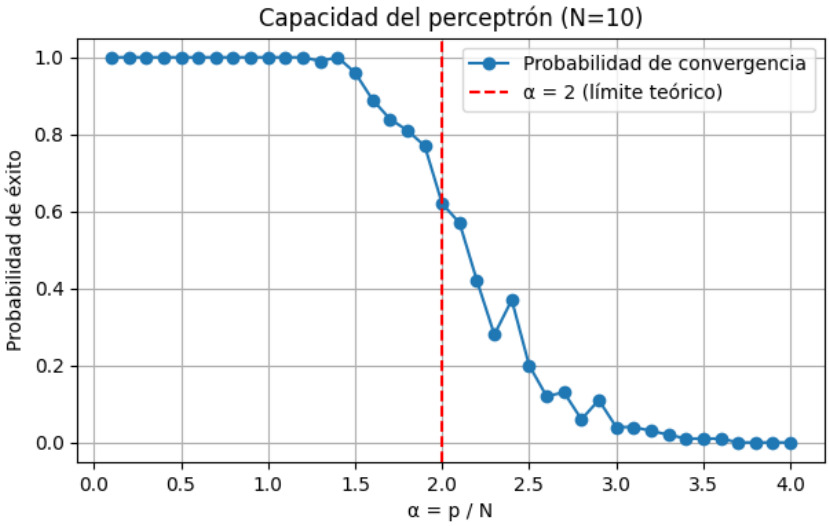

Figura 8: Ensayo de capacidad $N = 10$

3.3. Análisis

La figura 7 muestra que la capacidad de memorizar (el eje Y) del perceptrón es perfecta (vale 1) y, a medida que aumenta la cantidad de patrones, decrece hasta hacerse cero. El cruce por el punto $p/N = 2$ se da aproximadamente en $p \simeq 0{,}5$, resultado coincidente con la teoría del libro mencionado.

La transición, o flanco, no es perfectamente vertical, sino que decrece casi como una función sigmoidea. Esto es debido al uso de valores de N relativamente pequeños. A forma comparativa se incluye la imagen 8, de $N = 10$. En esta se ve que la capacidad tarda más en converge a cero (en $N = 30$ se hace cero en 2,5, y en $N = 10$, en 3,5), y la intersección del límite teórico es más cercana a 0,6 y no 0,5.

Resumiendo, aún con las limitaciones en la cantidad de unidades de entrada, se empieza a divisar una tendencia que parece coincidir con la teoría.

4. Ejercicio 3

4.1. Consignas

Implemente un perceptrón multicapa que aprenda la función lógica XOR de 2 y de 4 entradas (utilizando el algoritmo Backpropagation y actualizando en batch). Muestre cómo evoluciona el error durante el entrenamiento.

4.2. Desarrollo

En este inciso se entrenó un MLP para que aprendan la XOR, pero se hizo con 2, 4 y 10 neuronas en la capa oculta. La idea general fue crear las tablas de verdad de las funciones XOR de 2 y 4 entradas y entrenar MLP´s, registrando el error en función de las epochs.

4.2.1. 2 Neuronas en capa oculta

En este primer caso se utilizó un perceptrón multicapa con una capa oculta de dos neuronas, con el objetivo de que aprenda la función XOR. Se trabajó tanto con la XOR de dos entradas como con la de cuatro entradas, manteniendo la misma arquitectura a fin de observar las diferencias en la capacidad de representación del modelo.

Para la XOR de dos entradas, la red logró converger correctamente, alcanzando un error cercano a cero y reproduciendo exactamente la tabla de verdad de la función. En las Figuras 9 y 10 se muestra la evolución del error por época y la frontera de decisión obtenida, respectivamente, donde puede verse una separación clara entre las clases.

En contraste, al aplicar la misma arquitectura a la XOR de cuatro entradas, la red no consiguió reducir el error a valores cercanos a cero, incluso habiendo hasta triplicado la cantidad de epochs. (Figura 11).

Esto apunta a que dos neuronas ocultas no son suficientes para modelar la complejidad de la XOR de cuatro entradas, y se supone que se debe principalmente al incremento exponencial en cantidad de combinaciones posibles. La red ya no dispone de capacidad suficiente como para memorizar los patrones, siendo esto una manifestación práctica de la capacidad teórica analizada en el ejercicio anterior.

Entradas	**Target**	**Salida sigmoide**	**Predicción (umbral 0.5)**
[0 0]	0	0.082	0
[0 1]	1	0.927	1
[1 0]	1	0.927	1
[1 1]	0	0.077	0

Cuadro 1: Resultados del perceptrón con dos entradas - MLP 2 neuronas en capa oculta

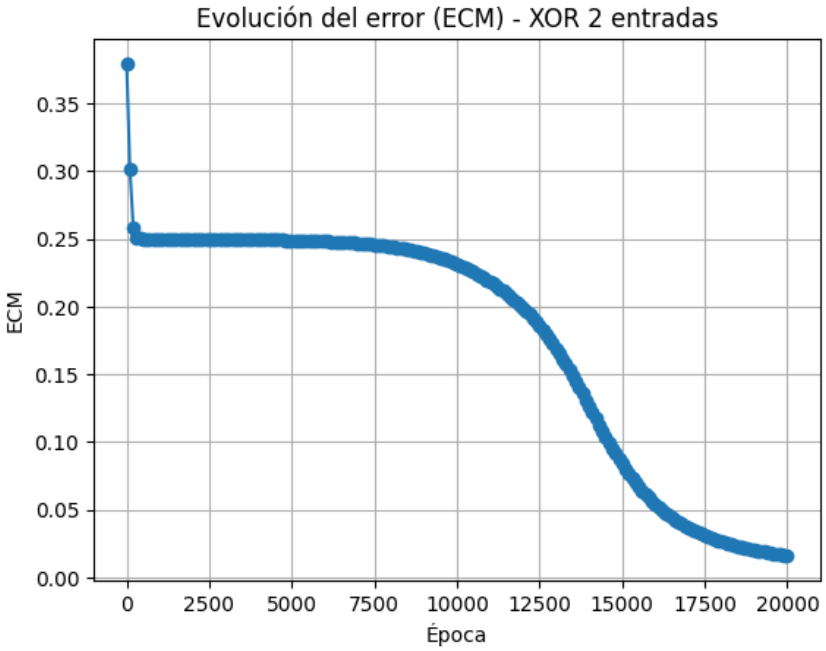

Figura 9: Error por época - XOR de 2 entradas - MLP 2 neuronas en capa oculta

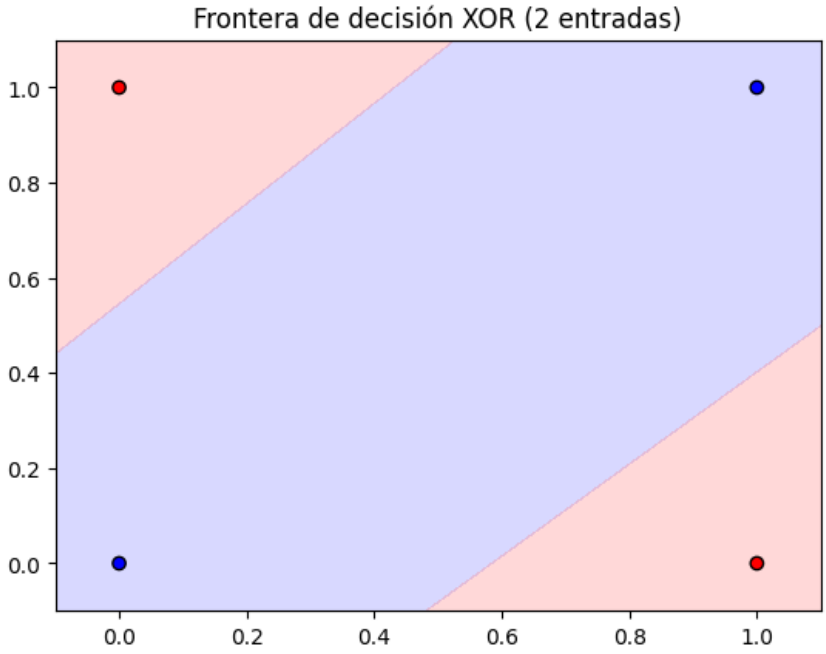

Figura 10: Frontera de decisión - XOR de 2 entradas - MLP 2 neuronas en capa oculta

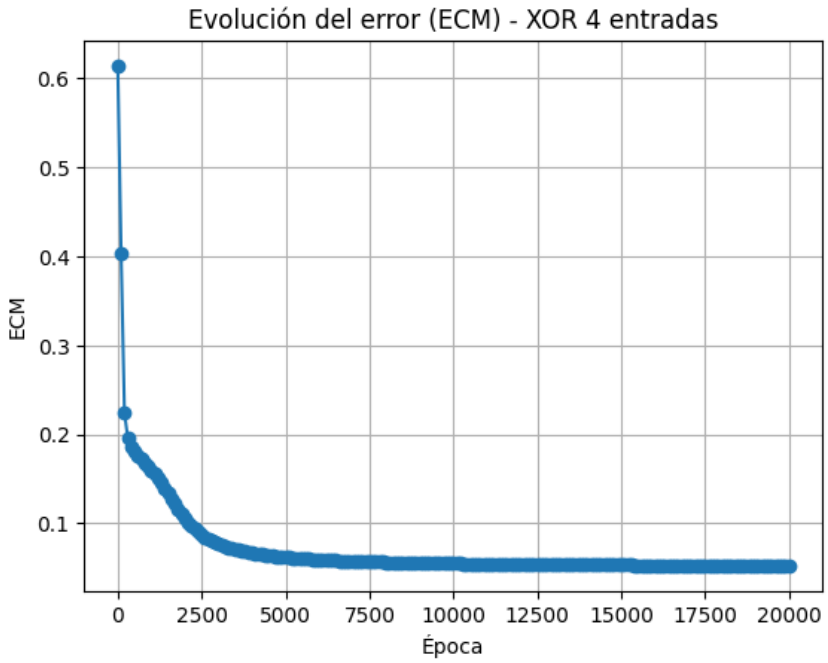

Figura 11: Error por época - XOR de 4 entradas - MLP 2 neuronas en capa oculta

Entradas	**Target**	**Salida sigmoide**	**Predicción (umbral 0.5)**
$[-1\ -1\ -1\ -1]$	0	0.725	1
$[1\ -1\ -1\ -1]$	1	0.837	1
$[-1\ 1\ -1\ -1]$	1	0.701	1
$[1\ 1\ -1\ -1]$	0	0.036	0
$[-1\ -1\ 1\ -1]$	1	0.708	1
$[1\ -1\ 1\ -1]$	0	0.048	0
$[-1\ 1\ 1\ -1]$	0	0.045	0
$[1\ 1\ 1\ -1]$	0	0.009	0
$[-1\ -1\ -1\ 1]$	1	0.879	1
$[1\ -1\ -1\ 1]$	0	0.063	0
$[-1\ 1\ -1\ 1]$	0	0.030	0
$[1\ 1\ -1\ 1]$	0	0.029	0
$[-1\ -1\ 1\ 1]$	0	0.055	0
$[1\ -1\ 1\ 1]$	0	0.036	0
$[-1\ 1\ 1\ 1]$	0	0.011	0
$[1\ 1\ 1\ 1]$	0	0.027	0

Cuadro 2: Resultados del perceptrón con cuatro entradas - MLP 2 neuronas en capa oculta

4.2.2. 4 Neuronas en capa oculta

En este caso se incrementó el número de neuronas en la capa oculta a cuatro, con el objetivo de mejorar la capacidad de representación del MLP. Para la XOR de dos entradas, la red continúa convergiendo correctamente, alcanzando un error muy bajo y generando fronteras de decisión suaves y bien definidas (Figuras 12 y 13). Esto se refleja también en la tabla de resultados, donde todas las predicciones coinciden con los valores esperados.

Para la XOR de cuatro entradas, aunque el error global se reduce ligeramente en comparación con la red de dos neuronas, la red aún no logra converger a un error nulo (Figura 14). La complejidad de la función sigue superando la capacidad de la red, aunque se observa que las fronteras de decisión empiezan a tomar formas curvas que intentan separar mejor las clases. Esto evidencia que un aumento en el número de neuronas mejora la capacidad de representación, pero todavía no es suficiente para clasificar correctamente todas las combinaciones de la XOR de cuatro entradas. En la tabla correspondiente se puede ver que varias predicciones todavía difieren del target esperado.

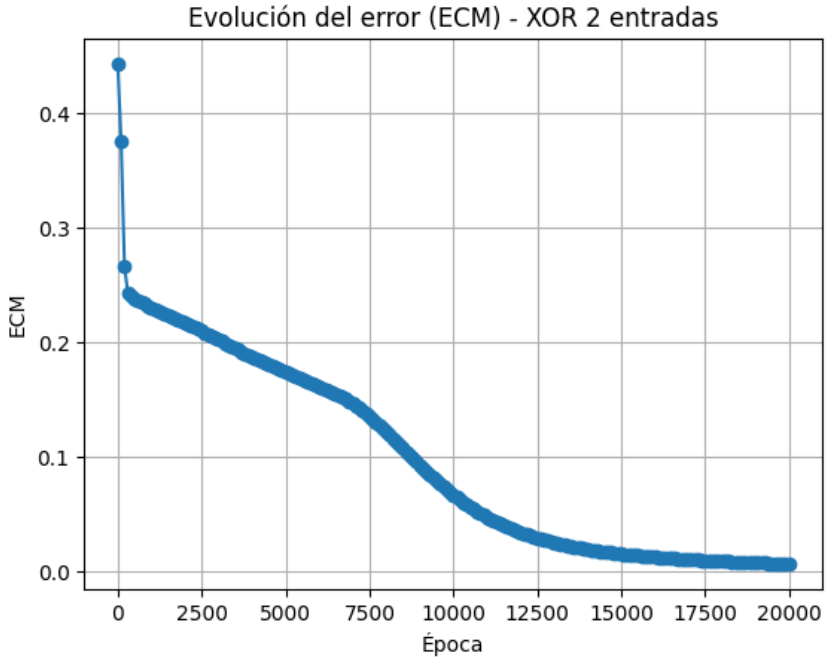

Figura 12: Error por época - XOR de 2 entradas - MLP 4 neuronas en capa oculta

Entradas	**Target**	**Salida sigmoide**	**Predicción (umbral 0.5)**
[0 0]	0	0.059	0
[0 1]	1	0.902	1
[1 0]	1	0.946	1
[1 1]	0	0.102	0

Cuadro 3: Resultados del perceptrón con dos entradas - MLP 4 neuronas en capa oculta

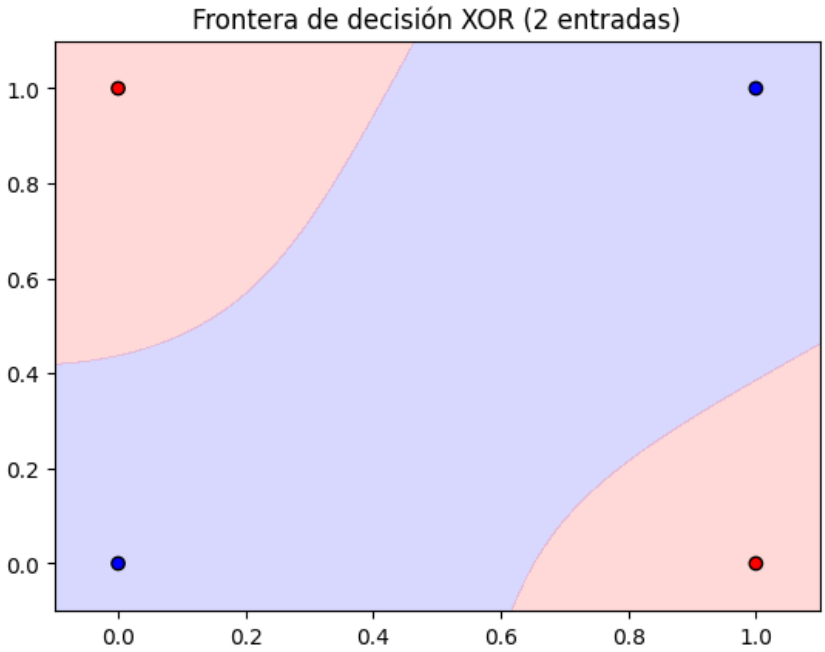

Figura 13: Frontera de decisión - XOR de 2 entradas - MLP 4 neuronas en capa oculta

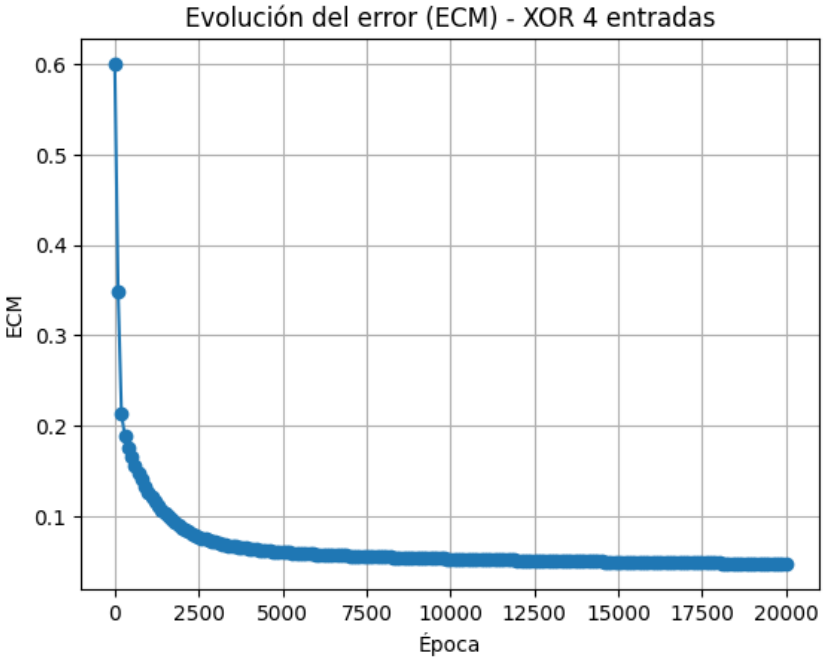

Figura 14: Error por época - XOR de 4 entradas - MLP 4 neuronas en capa oculta

Entradas	**Target**	**Salida sigmoide**	**Predicción (umbral 0.5)**
$[-1\ -1\ -1\ -1]$	0	0.727	1
$[1\ -1\ -1\ -1]$	1	0.879	1
$[-1\ 1\ -1\ -1]$	1	0.697	1
$[1\ 1\ -1\ -1]$	0	0.033	0
$[-1\ -1\ 1\ -1]$	1	0.863	1
$[1\ -1\ 1\ -1]$	0	0.054	0
$[-1\ 1\ 1\ -1]$	0	0.039	0
$[1\ 1\ 1\ -1]$	0	0.010	0
$[-1\ -1\ -1\ 1]$	1	0.696	1
$[1\ -1\ -1\ 1]$	0	0.028	0
$[-1\ 1\ -1\ 1]$	0	0.058	0
$[1\ 1\ -1\ 1]$	0	0.003	0
$[-1\ -1\ 1\ 1]$	0	0.034	0
$[1\ -1\ 1\ 1]$	0	0.007	0
$[-1\ 1\ 1\ 1]$	0	0.003	0
$[1\ 1\ 1\ 1]$	0	0.003	0

Cuadro 4: Resultados del perceptrón con cuatro entradas - MLP 4 neuronas en capa oculta

4.2.3. 10 Neuronas en capa oculta

Al aumentar el número de neuronas en la capa oculta a diez, se observa un cambio significativo en el comportamiento del MLP.

Para la XOR de dos entradas, la red sigue convergiendo sin problemas y mantiene fronteras de decisión suaves y bien definidas (Figura 15 y 16). Las fronteras de decisión siguen siendo curvas, lo que indica que la función sigue siendo no lineal, pero la gran cantidad de neuronas le da mucha variabilidad a las soluciones; en cada ejecución la frontera cambia mucho, puede ser más recta o muy curva.

Para la XOR de cuatro entradas, a diferencia de los casos anteriores con 2 o 4 neuronas, ahora la red logra aprender correctamente todos los patrones, alcanzando un error muy bajo (Figura 17). Esto confirma la sospecha de que las dificultades de aprendizaje de la XOR de 4 entradas en los experimentos anteriores estaba relacionada con la capacidad de la red, que se encontraba limitada por la cantidad de neuronas. El cuadro 6 muestra la tabla de verdad enseñada y aprendida. Nótese que, contrario a los casos para menos neuronas, la red aprendió bien la lógica, y esto se nota en que las salidas de la sigmoidea son o muy cercanas a 0, o muy cercanas a 1, indicando poca confusión.

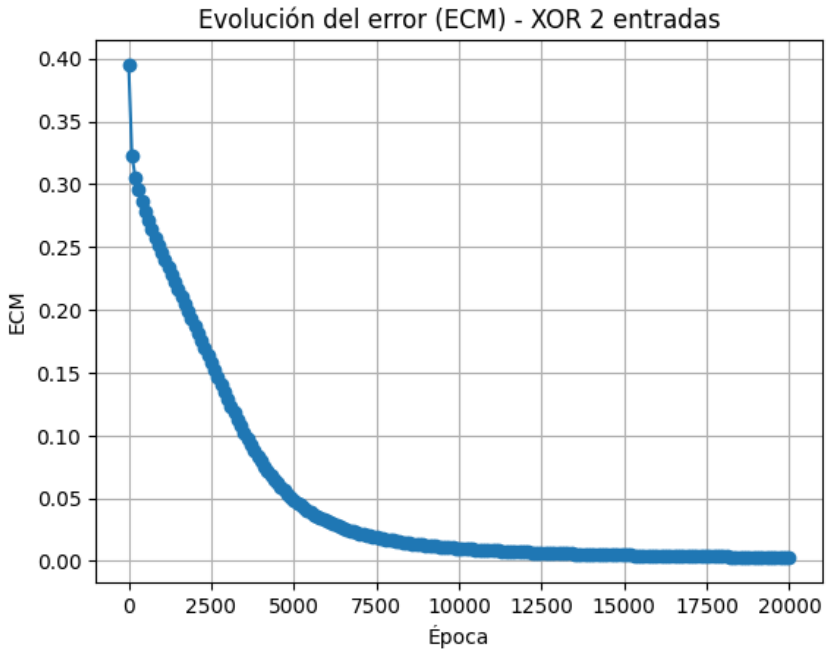

Figura 15: Error por época - XOR de 2 entradas - MLP 10 neuronas en capa oculta

Entradas	**Target**	**Salida sigmoide**	**Predicción (umbral 0.5)**
[0 0]	0	0.051	0
[0 1]	1	0.948	1
[1 0]	1	0.942	1
[1 1]	0	0.057	0

Cuadro 5: Resultados del perceptrón con dos entradas y 10 neuronas ocultas.

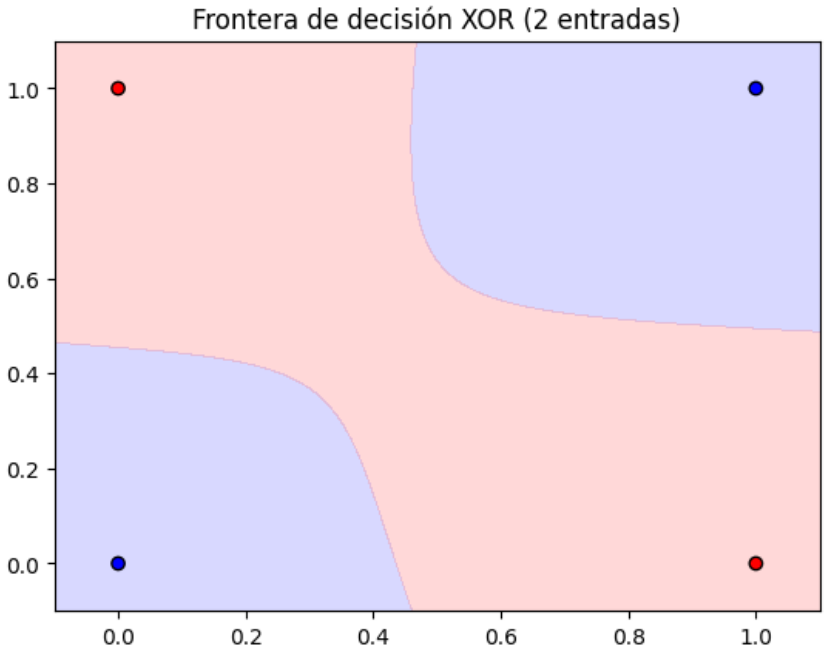

Figura 16: Frontera de decisión - XOR de 2 entradas - MLP 10 neuronas en capa oculta

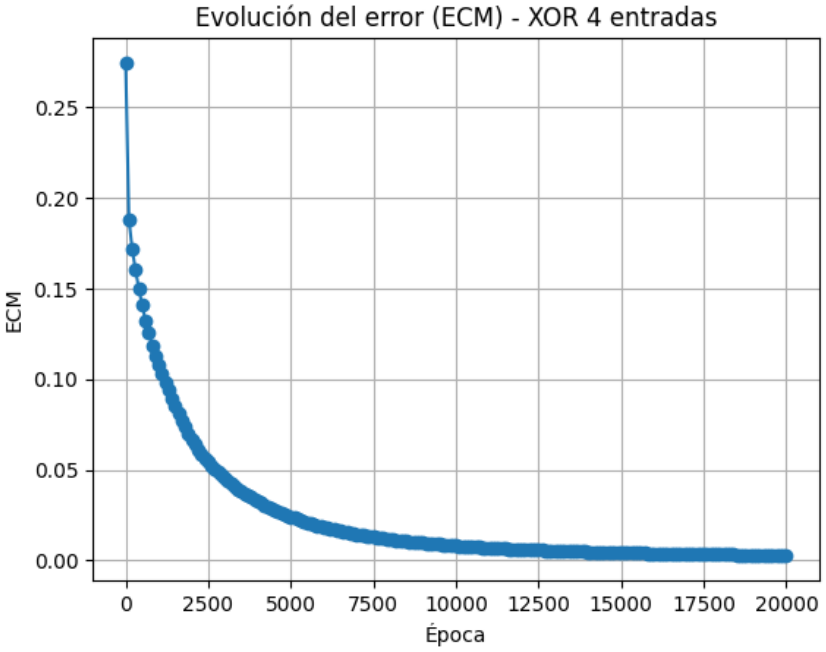

Figura 17: Error por época - XOR de 4 entradas - MLP 10 neuronas en capa oculta

Entradas	**Target**	**Salida sigmoide**	**Predicción (umbral 0.5)**
$[-1\ -1\ -1\ -1]$	0	0.113	0
$[1\ -1\ -1\ -1]$	1	0.950	1
$[-1\ 1\ -1\ -1]$	1	0.931	1
$[1\ 1\ -1\ -1]$	0	0.028	0
$[-1\ -1\ 1\ -1]$	1	0.912	1
$[1\ -1\ 1\ -1]$	0	0.018	0
$[-1\ 1\ 1\ -1]$	0	0.062	0
$[1\ 1\ 1\ -1]$	0	0.003	0
$[-1\ -1\ -1\ 1]$	1	0.921	1
$[1\ -1\ -1\ 1]$	0	0.042	0
$[-1\ 1\ -1\ 1]$	0	0.017	0
$[1\ 1\ -1\ 1]$	0	0.017	0
$[-1\ -1\ 1\ 1]$	0	0.012	0
$[1\ -1\ 1\ 1]$	0	0.010	0
$[-1\ 1\ 1\ 1]$	0	0.006	0
$[1\ 1\ 1\ 1]$	0	0.052	0

Cuadro 6: Resultados del perceptrón con cuatro entradas y 10 neuronas ocultas.

4.3. Análisis

El análisis ya se desarrolló casi completamente en cada subsección. Lo que se puede afirmar es que la función XOR de 2 entradas es relativamente simple de aprender para un MLP (3 perceptrones mínimo; 2 en la capa oculta), pero no tanto la XOR de 4 entradas, que requirió de una capa oculta bastante más grande (10 neuronas). Esto demuestra que siguen existiendo limitaciones a la cantidad de patrones que puede aprender una cierta red de perceptrones, y que parecería aumentar con la cantidad de neuronas agregadas.

Algo que no se mencionó es que los errores tienen forma de exponenciales decrecientes (aunque a veces deformes), que parecería normal para estos casos. Algunos entrenamientos presentaron como puntos silla, lo que indicaría que el algoritmo se encontró con una región de gradiente muy pequeño, como la figura 9, que se quedó varias epochs en 0.25 de ECM.

otro tema interesante para mencionar es que, como el método de gradiente usado no es estocástico, el resultado final del aprendizaje es determinístico. Esto trae como consecuencia que, si el espacio de soluciones de error mínimo es pequeño, sería difícil encontrar justo la condición inicial que caiga en el (más bien, no se encontró para menos de 10 neuronas). Claramente, el uso de más neuronas da más grados de libertad a la red, y más capacidad, y todo esto parecería ser acompañado por un incremento en accesibilidad para las regiones que son solución al problema. Volviendo al punto anterior, es posible que haya soluciones en espacios de menor dimensión que el de la red de 10 neuronas en la capa oculta, pero no fueron accedidos por la regla de aprendizaje. Opino que existe la posibilidad de que la función XOR de 4 entradas se pueda aprender con menos neuronas pero con el uso de alguna regla que meta estocacidad en el sistema, como el uso de un gradiente estocástico. Este punto se retoma en el inciso de *simulated annealing*, donde se comprueba esta hipótesis.

5. Ejercicio 4

5.1. Consignas

a - Implemente una red con aprendizaje Backpropagation que aprenda la siguiente función:

$$f(x, y, z) = \sin(x) + \cos(y) + z$$

Donde: x e $y \in [0, 2\pi]$ y $z \in [-1, 1]$.

Para ello construya un conjunto de datos de entrenamiento y un conjunto de evaluación. Muestre la evolución del error de entrenamiento y de evaluación en función de las épocas de entrenamiento.

b - Estudie la evolución de los errores durante el entrenamiento de una red con una capa oculta de 30 neuronas cuando el conjunto de entrenamiento contiene 40 muestras. ¿Que ocurre si el minibatch tiene tamaño 40? ¿Y si tiene tamaño 1?

5.2. Desarrollo

El conjunto de datos se generó sin mayores problemas, fue solo evaluar la función $f(x, y, z)$ en muchos puntos.

Los datasets de entrnamiento y testeo se separaron 70/30 (56000 muestras de entrenamiento) y se entrenó variando el tamaño de minibatch.

5.3. Análisis

6. Ejercicio 5

6.1. Consignas

Siguiendo el trabajo de Hinton y Salakhutdinov (2006), entrene una máquina restringida de Boltzmann con imágenes de la base de datos MNIST. Muestre el error de reconstrucción durante el entrenamiento, y ejemplos de cada uno de los dígitos reconstruidos.

6.2. Desarrollo

6.3. Análisis

7. Ejercicio 6

7.1. Consignas

Entrene una red convolucional para clasificar las imágenes de la base de datos MNIST. ¿Cuál es la red convolucional más pequeña que puede conseguir con una exactitud de al menos 90 % en el conjunto de evaluación? ¿Cuál es el perceptrón multicapa más pequeño que puede conseguir con la misma exactitud?

7.2. Desarrollo

7.3. Análisis

8. Ejercicio 7

8.1. Consignas

Entrene un autoencoder para obtener una representación de baja dimensionalidad de las imágenes de MNIST. Use dichas representaciones para entrenar un perceptrón multicapa como clasificador. ¿Cuál es el tiempo de entrenamiento y la exactitud del clasificador obtenido cuando parte de la representación del autoencoder, en comparación con lo obtenido usando las imágenes originales?

8.2. Desarrollo

entrenamiento final: Epoch 1/30 — Loss=0.6624 — Acc=81.28Epoch 2/30 — Loss=0.3537 — Acc=89.86Epoch 3/30 — Loss=0.3200 — Acc=90.68Epoch 4/30 — Loss=0.3073 — Acc=90.92Epoch 5/30 — Loss=0.2996 — Acc=91.19Epoch 6/30 — Loss=0.2962 — Acc=91.22Epoch 7/30 — Loss=0.2921 — Acc=91.29Epoch 8/30 — Loss=0.2884 — Acc=91.44Epoch 9/30 — Loss=0.2884 — Acc=91.51Epoch 10/30 — Loss=0.2855 — Acc=91.53Epoch 11/30 — Loss=0.2832 — Acc=91.62Epoch 12/30 — Loss=0.2837 — Acc=91.56Epoch 13/30 — Loss=0.2823 — Acc=91.63Epoch 14/30 — Loss=0.2813 — Acc=91.79Epoch 15/30 — Loss=0.2810 — Acc=91.69Epoch 16/30 — Loss=0.2799 — Acc=91.83Epoch 17/30 — Loss=0.2785 — Acc=91.79Epoch 18/30 — Loss=0.2773 — Acc=91.82Epoch 19/30 — Loss=0.2772 — Acc=91.85Epoch 20/30 — Loss=0.2775 — Acc=91.76Epoch 21/30 — Loss=0.2751 — Acc=91.91Epoch 22/30 — Loss=0.2738 — Acc=91.90Epoch 23/30 — Loss=0.2745 — Acc=91.89Epoch 24/30 — Loss=0.2738 — Acc=91.94Epoch 25/30 — Loss=0.2735 — Acc=91.97... Epoch 28/30 — Loss=0.2736 — Acc=91.98Epoch 29/30 — Loss=0.2726 — Acc=91.98Epoch 30/30 — Loss=0.2720 — Acc=92.05Test Accuracy: 92.33

8.3. Análisis

9. Ejercicio 8

9.1. Consignas

Encontrar un perceptrón multicapa que resuelva una XOR de 2 entradas mediante *simulated annealing*. Graficar el error a lo largo del proceso de aprendizaje.

9.2. Desarrollo

Para este ejercicio se reusó el código del ejercicio 3, cambiando la regla de aprendizaje a aquella del *simulated annealing*.

El algoritmo inicia con un set de pesos w, actualizados como $w(n+1) = w(n) + \mathcal{N}(0, \sigma^2)$. Es decir, el paso, o δw es aleatorio, y la varianza de la gaussiana determina que tan grande es ese paso.

Luego de obtener un set de parámetros $w(n+1)$, se calcula el $\delta E = E(w(n+1)) - E(w(n))$, que es lo mismo que decir que se calcula el cambio del error después del cambio de parámetros.

Ahora viene lo interesante: si el error decrece ($\delta E < 0$) entonces se admite el cambio de parámetros y $w(n) = w(n+1)$. En cambio, si el error empeora ($\delta E > 0$), el cambio se admite con una probabilidad $e^{\frac{-\delta E}{T}}$, con T la temperatura del sistema en ese instante.

Emula el proceso físico de templar y revenir una pieza de metal.

Como detalle final, la temperatura se debe decrecer para que el algoritmo vaya convergiendo (si es muy alta tal vez pasea por el espacio de estados sin límite). Esto generalmente se hace de forma exponencial como $T(n+1) = \alpha T(n)$.

En el desarrollo hecho se usó un decaimiento de temperatura del tiempo exponencial y un σ constante. A continuación se detallan los varios casos simulados.

9.2.1. XOR de 2 entradas - 2 neuronas en capa oculta - $\alpha = 0{,}995$

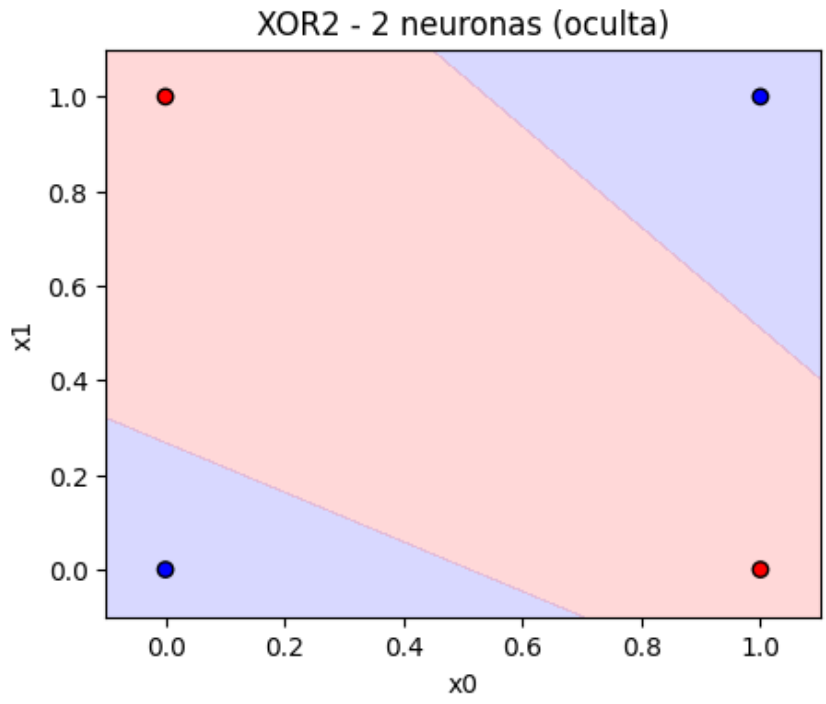

Figura 18: Frontera de decisión de MLP de 2 neuronas en capa oculta

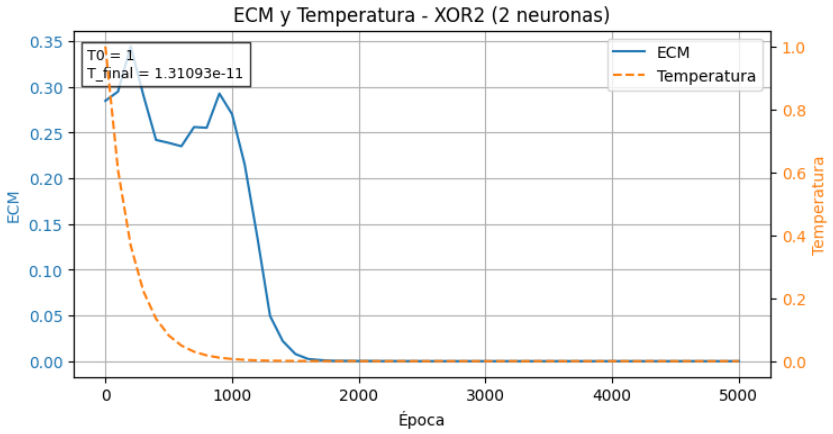

Figura 19: ECM y temperatura por interación de MLP de 2 neuronas en capa oculta

La XOR de 2 entradas fue aprendida perfectamente por una red de 2 neuronas en la capa oculta, igual que en la consigna 3. La imagen **??**uestra la frontera de decisiónfig:screenshot001, también recta como en el caso de gradiente descendiente. En el gráfico 19 se puede ver el ECM y temperatura. hay cierta estocacidad con el ECM (como es de esperar por la temperatura), pero termina disminuyendo hasta converger en 0.

9.2.2. XOR de 2 entradas - 4 neuronas en capa oculta - $\alpha = 0{,}995$

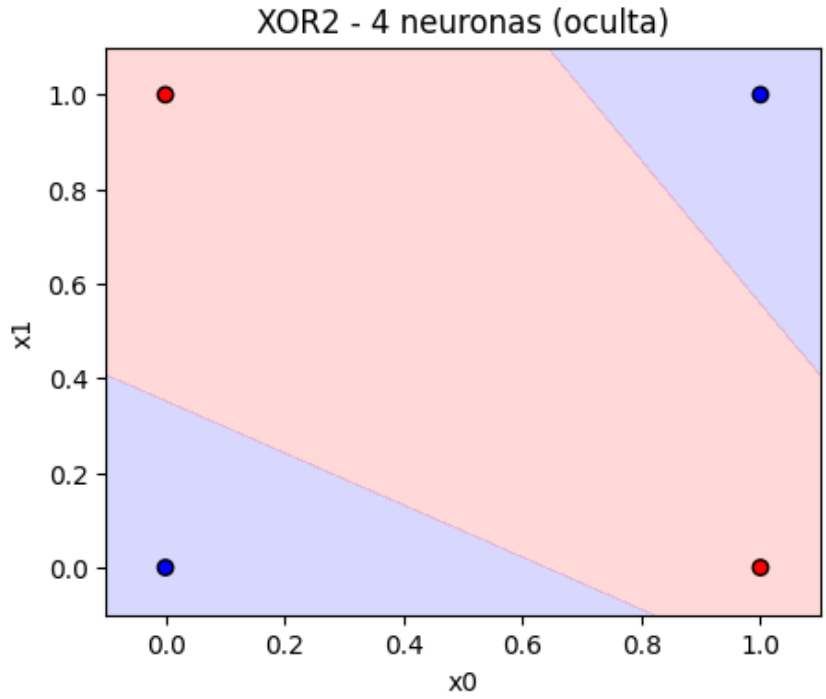

Figura 20: Frontera de decisión de MLP de 4 neuronas en capa oculta

La figura 20 muestra la frontera de decisión, y la 21 contiene el ECM y temperatura. La red aprendió el problema a la perfección. Su ECM es decreciente y la forntera es congruente con lo esperado y ya visto.

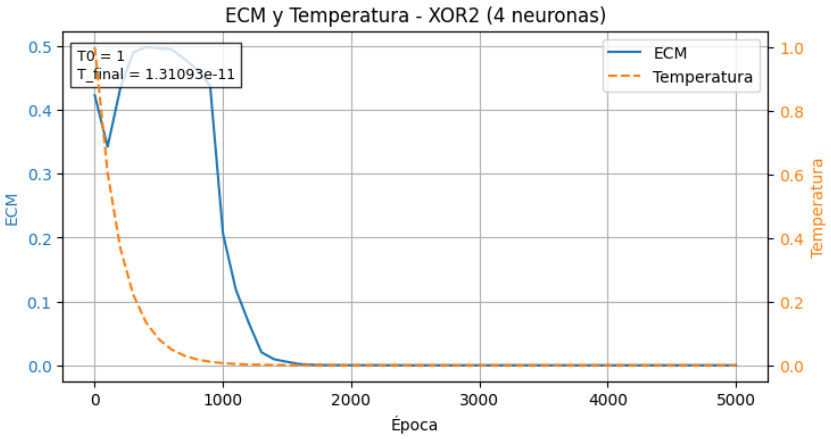

Figura 21: ECM y temperatura por interación de MLP de 4 neuronas en capa oculta

9.2.3. XOR de 2 entradas - 10 neuronas en capa oculta - $\alpha = 0{,}995$

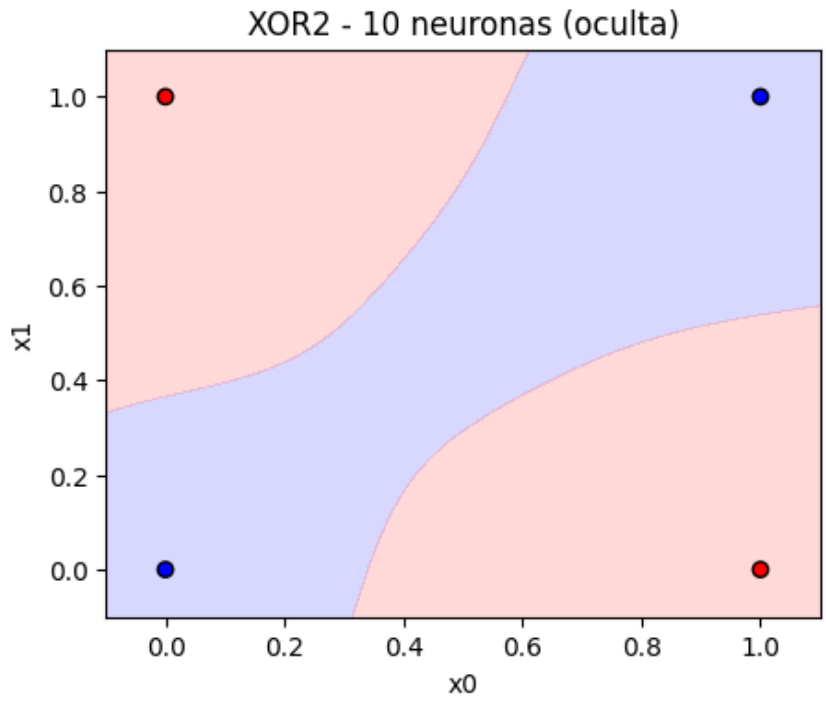

Figura 22: Frontera de decisión de MLP de 10 neuronas en capa oculta

La figura 22 muestra la frontera de decisión, y la 23 contiene el ECM y temperatura. la frontera de decisión es curva como en el caso de 10 neuronas con gradiente descendiente y el ECM disminuye como se esperaba.

9.2.4. XOR de 2 entradas - 2 neuronas en capa oculta - $\alpha = 0{,}98$

Para este caso (figuras 24 y 25), em uno la red aprendió el problema aún cuando la regla de la actualización de la temperatura era la de una exponencial rápida. En el otro no lo aprendió. Esto claramente es algo estocástico.

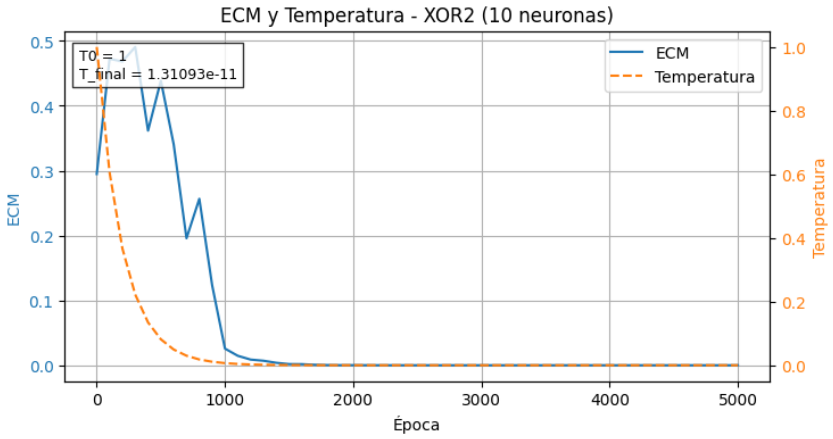

Figura 23: ECM y temperatura por interación de MLP de 10 neuronas en capa oculta

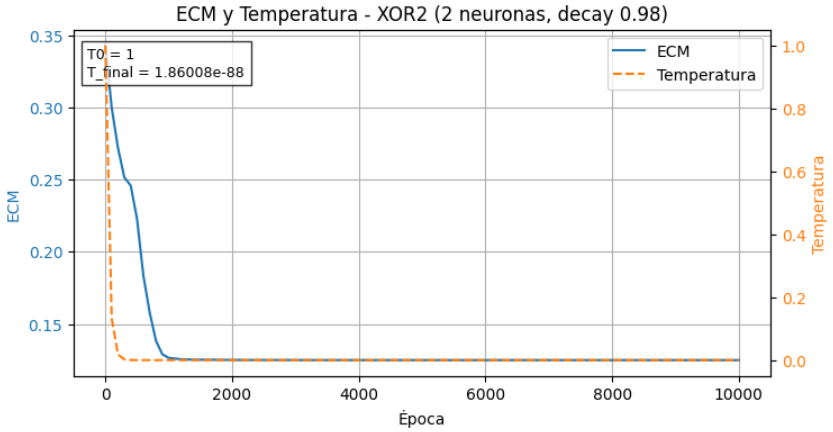

Figura 24: ECM y temperatura -Decay rate de 0.98 - caso 1

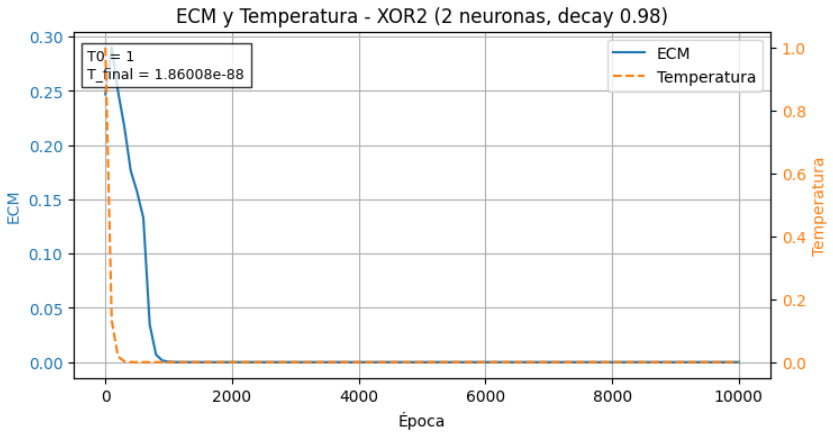

Figura 25: ECM y temperatura - Decay rate de 0.98 - caso 2

9.2.5. XOR de 2 entradas - 2 neuronas en capa oculta - $\alpha = 0{,}999$

Para este caso (figuras 26 y 27), la red claramente pasó demasiado tiempo con alta temperatura, cayendo en un mínimo no global. esto se ejecutó varias veces con el mismo resultado, aunque no es definitivo.

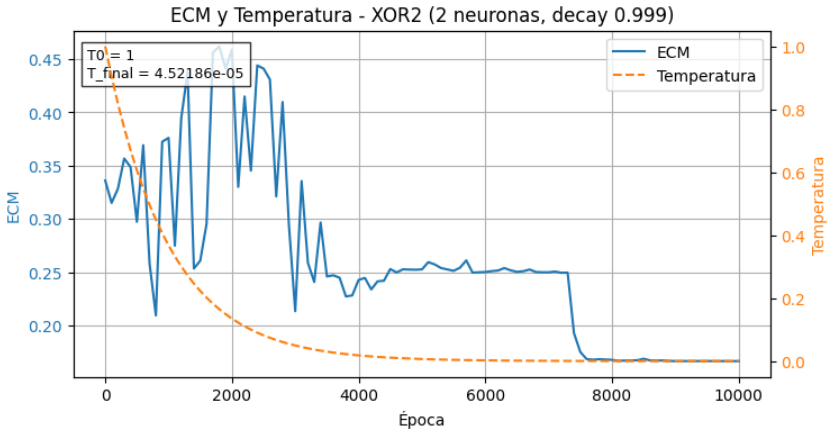

Figura 26: ECM y temperatura - Decay rate de 0.999

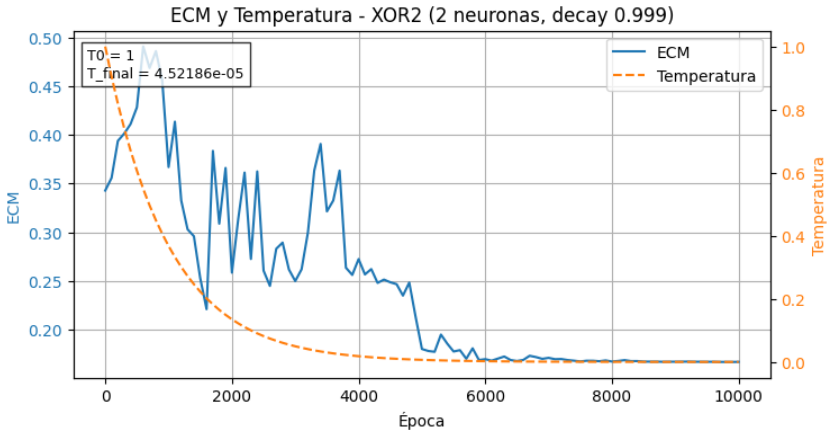

Figura 27: ECM y temperatura - Decay rate de 0.999

9.2.6. XOR de 4 entradas - 4 neuronas en capa oculta

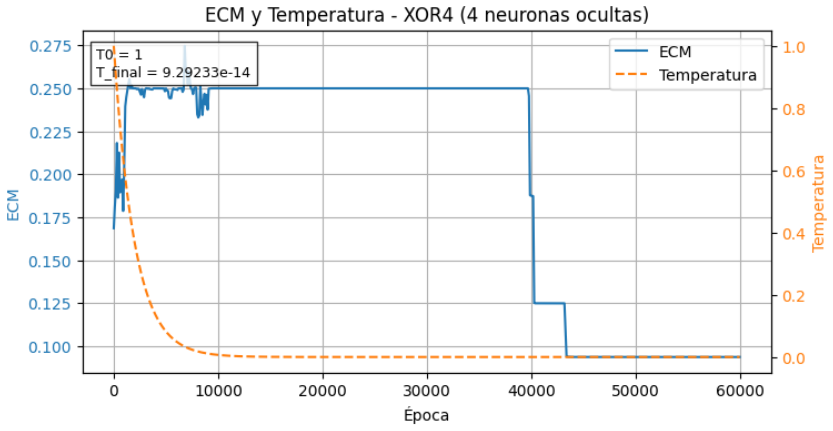

Figura 28

No aprendió

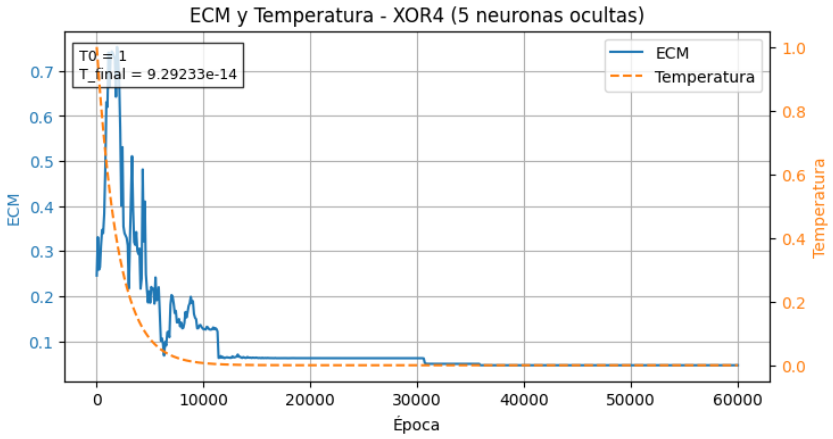

Figura 29

9.2.7. XOR de 4 entradas - 5 neuronas en capa oculta

aprendió

9.2.8. XOR de 4 entradas - 6 neuronas en capa oculta

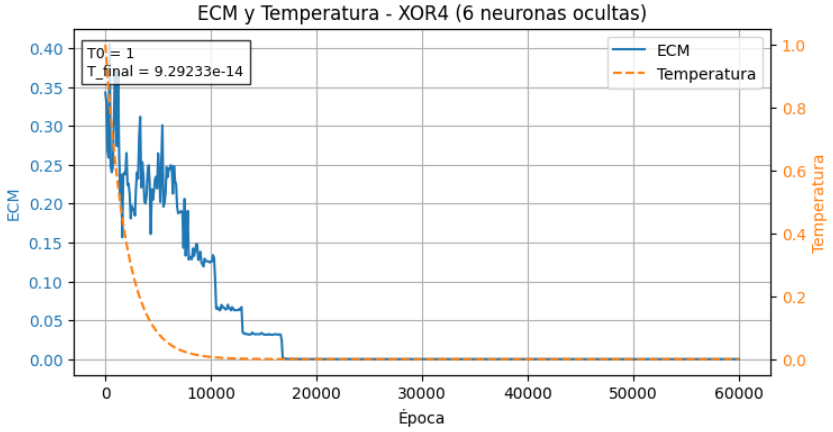

Figura 30

sobre estas últimas, puede ser que le sea más facil encontrar la solución pq el esacio de soluciones es más chico y este algoritmo se lo permite encontrar.

9.3. Análisis

10. Conclusiones