Aprendizaje supervisado en redes multicapa

Pablo Chehade
pablo.chehade@ib.edu.ar
Redes Neuronales, Instituto Balseiro, CNEA-UNCuyo, Bariloche, Argentina, 2023

EJERCICIO 1

Se implementaron dos arquitecturas para el aprendizaje de la regla XOR, las cuales se ilustran en la figura 1, considerando, en cada caso, una entrada adicional para simular el bias.

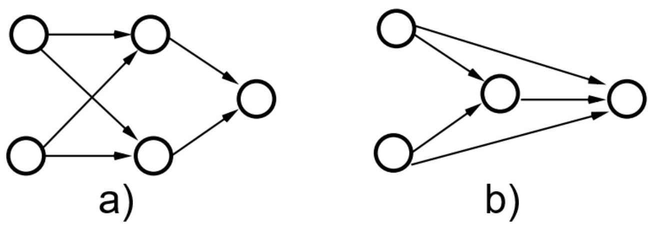

Figura 1: Arquitecturas utilizadas para el aprendizaje de la regla XOR, denominadas como arquitecturas a) A y b) B.

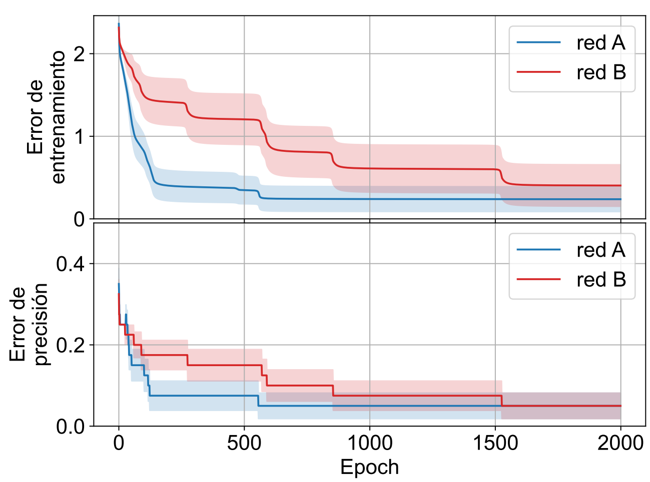

Figura 2: Valor medio del a) error de entrenamiento y b) error de precisión en función de las epochs de entrenamiento para las arquitecturas A y B. El sombreado indica la desviación estándar del promedio.

El aprendizaje fue ejecutado mediante el algoritmo de retropropagación de errores (back-propagation), con pesos inicializados aleatoriamente con un valor máximo de 0.1 y un learning rate establecido en 0.1. La función de costo empleada fue el error cuadrático medio (MSE) y se utilizó $f(x) = tanh(x)$ como función de transferencia. Los datos de entrenamiento engloban todas las posibles combinaciones de entradas y salidas. Mientras que los datos de test corresponden al mismo conjunto de datos de entrenamiento.

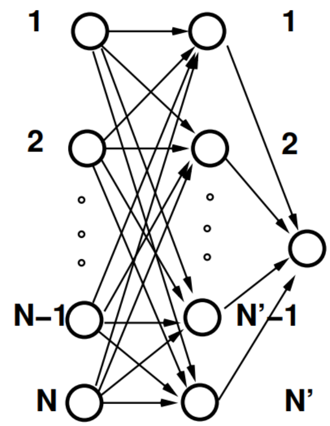

Figura 3: Arquitectura utilizada para abordar el problema de paridad.

En la Figura 2 se grafican los valores medios del error de entrenamiento y de precisión en función de las epochs para ambas arquitecturas, promediados sobre 10 condiciones iniciales de los pesos. En ambas arquitecturas, se evidencia una disminución de los errores a lo largo del tiempo, sin alcanzar un error nulo, posiblemente debido a que algunas redes se estabilizan en mínimos locales. De forma comparativa, la arquitectura A demuestra una velocidad de aprendizaje superior a la B.

Además, se observó cualitativamente que la velocidad de convergencia es influenciada por el valor máximo posible en la inicialización de los pesos y por el learning rate, existiendo configuraciones de ambos parámetros en las cuales el error no converge.

EJERCICIO 2

Se abordó la resolución del problema de paridad, extendiendo la lógica del XOR a N entradas. La arquitectura utilizada se muestra en la figura 3, habiendo N' neuronas en la capa oculta y añadiendo una entrada adicional para simular el bias. El entrenamiento se llevó a cabo a través del algoritmo de retropropagación de errores, manteniendo la función de transferencia y la inicialización de los pesos idénticas al ejercicio previo y un learning rate de 0.05. Se establecieron N = 5 y N' = 1, 3, 5, 7, 9 y 11. Al igual que antes, los datos de entrenamiento engloban todas las posibles combinaciones de entradas y salidas. Mientras que los datos de test corresponden al mismo conjunto de datos de entrenamiento.

figure[h]

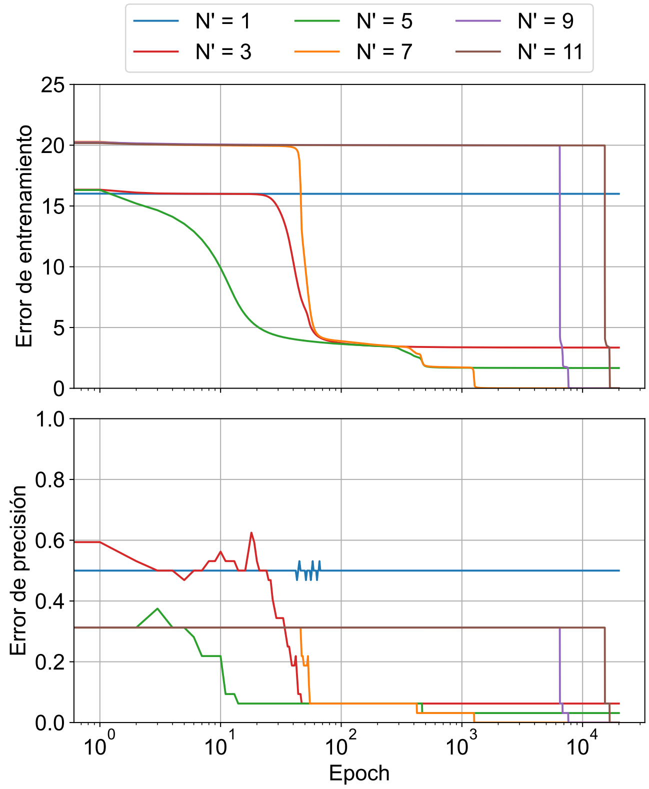

Figura 4: a) Error de entrenamiento y b) error de precisión en función de las epochs de entrenamiento, variando el número de neuronas N' en la capa oculta.

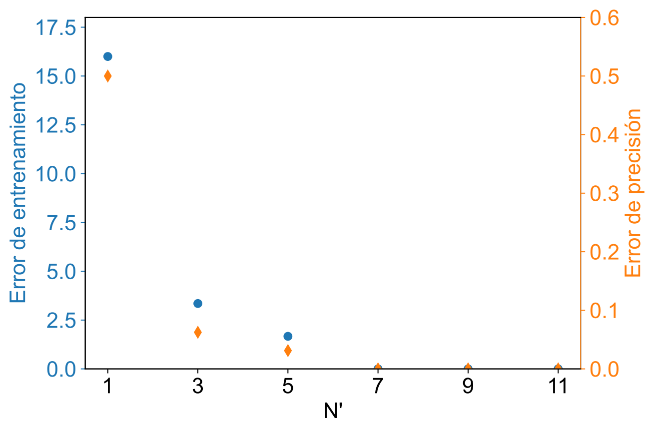

Figura 5: Error de entrenamiento y error de precisión final en función del número de neuronas N' en la capa oculta.

En la figura 4 se grafica el error de entrenamiento y de precisión en función de las epochs, explorando los diversos valores de N'.

Se observa que para N' = 1 el método converge pero con un gran error. Para N' = 3, la convergencia es más gradual hacia un error menor que el anterior pero distinto de cero. Con incrementos en N', la tendencia persiste: la convergencia es más lenta pero converge hacia errores progresivamente menores. Cuando N' ¿5, el error se anula pasado cierto número de epochs. Este fenómeno es evidenciado de forma más clara en la figura 5, donde se grafica el error final en función de N'. Se observa que el error decae con el aumento de N', directamente relacionado con la complejidad de la red.

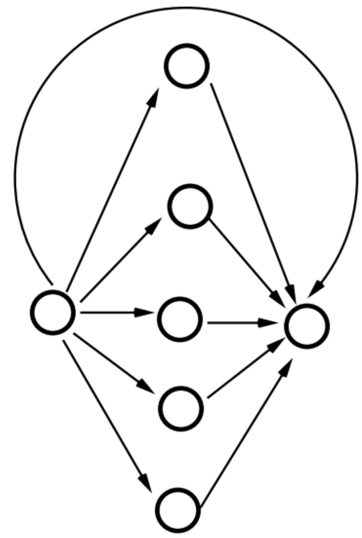

Figura 6: Arquitectura adoptada para el aprendizaje del mapeo logístico.

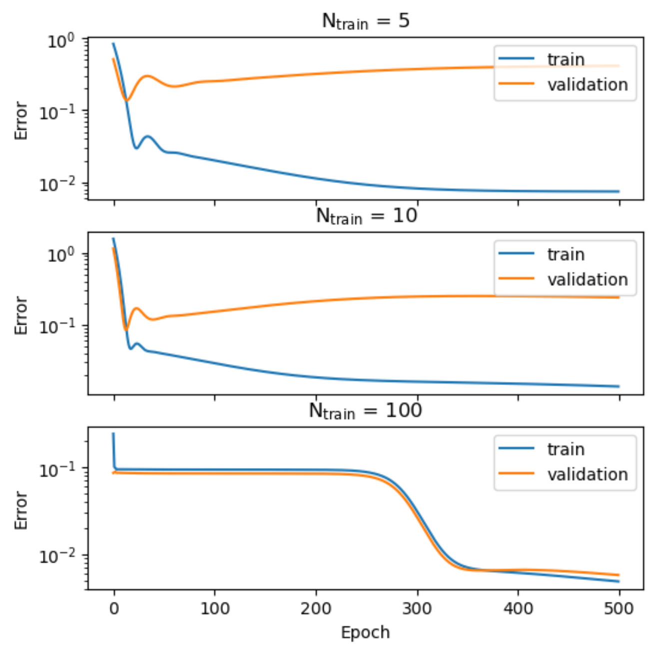

Figura 7: Error de entrenamiento y error de precisión en función de las epochs de entrenamiento, para distintas cantidades de ejemplos N_{train} en los datos de entrenamiento.

EJERCICIO 3

Se procedió al aprendizaje del mapeo logístico utilizando el método de retropropagación de errores, con la arquitectura representada en la figura 6 y añadiendo una entrada adicional para simular el bias. Se estableció un learning rate de 0.01 y, para las capas ocultas, se implementó la función de transferencia g(x) = 1/(1 + exp(-x)), mientras que la neurona de salida adoptó una función de activación lineal.

Se generaron los N_{train} datos de entrenamiento a través de la iteración del mapeo $x(t+1) = 4x(t)(1-x(t))$. De este modo, los datos corresponden a los pares x(t), x(t+1). Además, se utilizaron 100 datos generados de manera análoga como ejemplos de prueba.

En la figura 7 se grafica el error de entrenamiento y el error de generalización, calculado sobre los datos de prueba, en función de las epochs. Para $N_{train} = 5$ y 10, el comportamiento observado indica que, ante un número bajo de epochs, se está en condiciones de underfitting; luego, el error de generalización alcanza un mínimo y, posteriormente, aumenta, indicando una condición de overfitting. En cambio, para $N_{train} = 100$, la gran cantidad de datos permite que ambos errores sean muy similares, sin llegar a presentar overfitting. Este comportamiento con variaciones en N_{train} se justifica en que el error de generalización tiende a disminuir con la cantidad de ejemplos.

I. APÉNDICE

A continuación se desarrolla el código empleado durante este trabajo implementado en Python.

```
#Import libraries
import matplotlib.pyplot as plt
import numpy as np
```