Aprendizaje supervisado en redes neuronales multicapa

- **Autor:** Alexander Leano
- **Fecha:** 13 de Noviembre del 2024

Introducción

En este trabajo se busca realizar un modelo de aprendizaje supervisado utilizando redes neuronales multicapas para aprender la función XOR, problema de paridad (generalización del XOR) y el mapeo logistico.

El codigo fuente del trabajo se puede encontrar en el siguiente notebook-github.

1. Función XOR

La regla XOR tiene dos entradas (+1 o -1) y la salida es -1 si ambas son diferentes y +1 si ambas son iguales. Se busca utilizar el algoritmo de retropropagacion de errores para aprender el XOR en las siguientes arquitecturas.

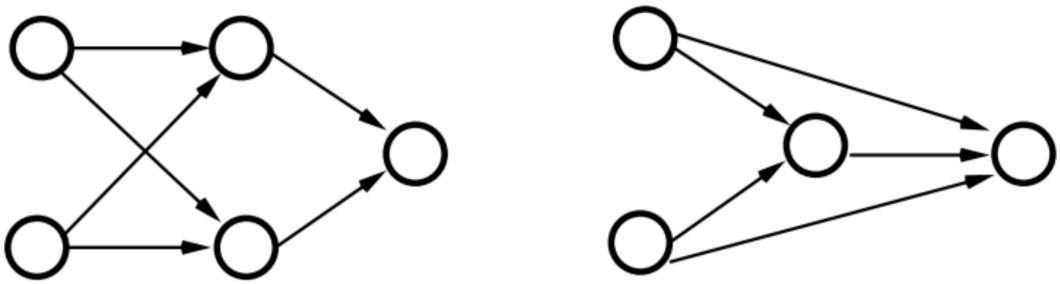

Figure 1: Dos arquitecturas posibles, izquierda Arquitectura o Red 1 derecha Arquitectura o Red 2.

En ambos casos se incluyen las unidades de entrada adicional para simular los umbrales. Se utiliza $\tanh(x)$ como función de transferencia. Se repite el proceso para 10 condiciones inciales de las conexiones.

Se comparar el tiempo medio de convergencia de ambas arquitecturas.

Los resultados obtenidos se representan en la *Figura 2* como el error cuadrático medio en función del número de iteraciones para ambas arquitecturas y 10 condiciones iniciales de las conexiones. Ademas se muestra el tiempo medio de convergencia para ambas arquitecturas.

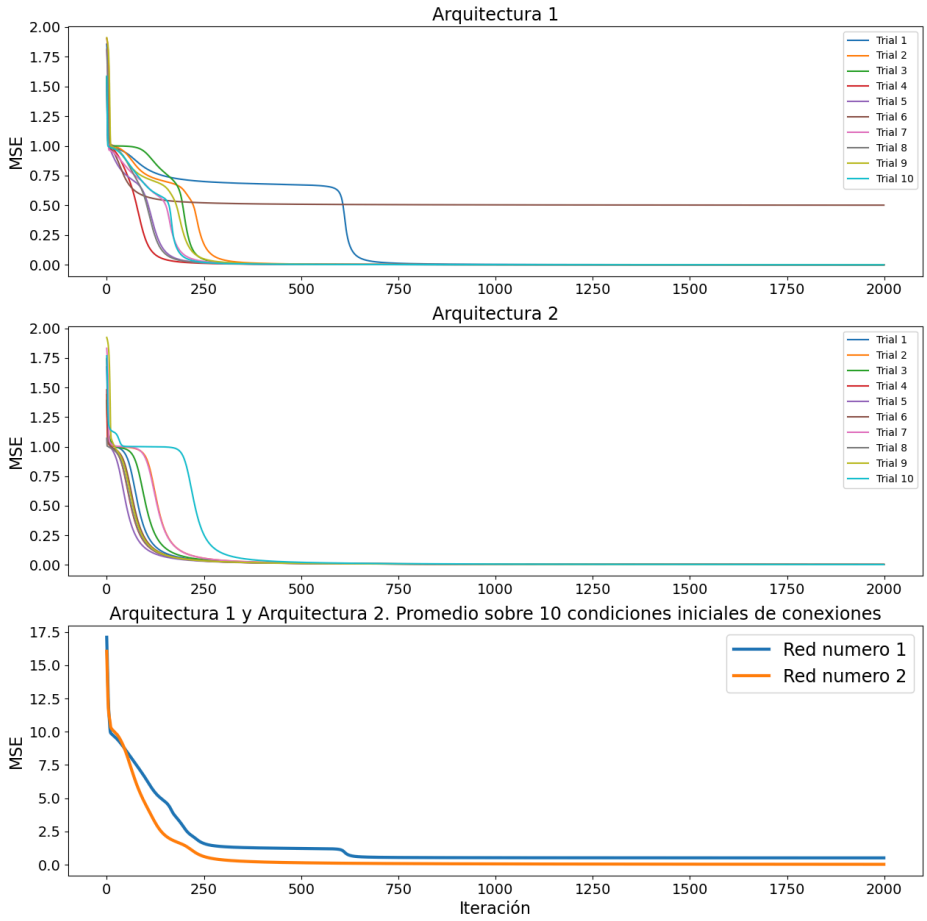

Figura 2: Error cuadrático medio en función de numero de iteración para arquitectura 1, 2 y una comparación entre los promedios promedios.

De este grafico se puede observar que la arquitectura 2 converge más rapido que la arquitectura 1 para los 10 casos iniciales de las conexiones analizados. Se puede observar tambien que la arquitectura 1 tiene minimos locales que tardan en salir de ellos. Se puede obervar que el la arquitectura 1 aun mantiene un error cuadratico medio mayor que la arquitectura 2 despues de 2000 iteraciones producto de un minimo local que aun no convergio.

Los tiempos de convergencia medios parecen ser similares, sin embargo, la arquitectura 2 muestra un mejor tiempo de convergencia para algunos casos lo que se refleja en el promedio.

2. Función de paridad

El problema de paridad es una generalización del XOR para N entradas. La salida es `+1` si el producto de las `N` entradas es `+1` y `−1` si el producto de las entradas es

-1. Se busca utilizar retropropagación para aprender el problema en la siguiente arquitectura que se muestra en la **Figura 3**.

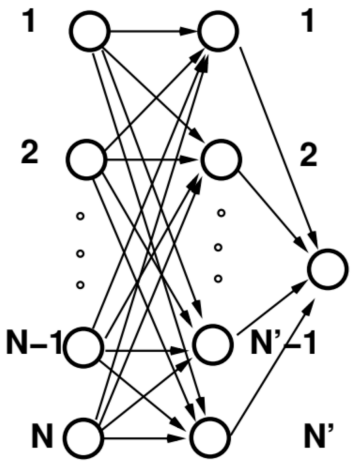

Figure 3: Arquitecturas XOR generalizado. Se resuelve para un número de entradas N=5 y un numero de neuronas ocultas de N'= 1, 3, 5, 7, 9, 11.

En este caso se buscar aprender el problema para un numero de entradas `N` fijo de 5 y un numero de neuronas `N'` en la capa oculta de 1, 3, 5, 7, 9, 11 respectivamente. De esta manera se busca analizar que sucede cuando `N'` es menor, igual o mayor a `N`.

Para cada caso con diferente numero de neuronas en la capa oculta se realizó un promedio del error entre 10 procesos de entrenamiento utilizando condiciones iniciales diferentes. Los resultados obtenidos se muestran en la **Figura 4**.

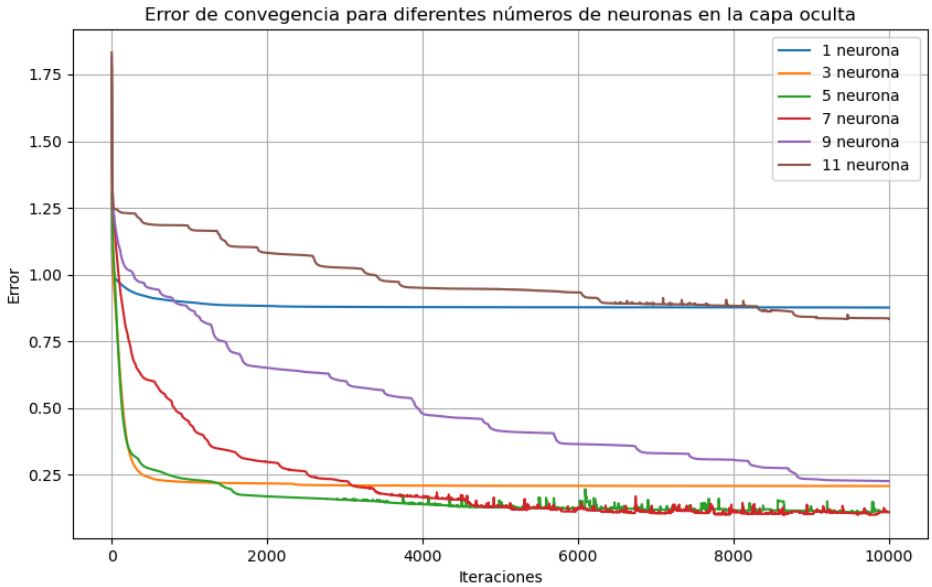

Figure 4: Error cuadrático medio en función de numero de iteración para arquitectura XOR generalizado. Se resuelve para un número de entradas N=5 y un numero de neuronas ocultas de N'= 1, 3, 5, 7, 9, 11.

Se puede observar que cuando el numero de neuronas en la capa oculta iguala al numero de entradas la red aprende la tarea de forma satisfactoria. En contraste, un numero menor de neuronas en la capa oculta que neuronas de entrada, no permite aprender la tarea ya que no alcanzan los grados de libertad para ajustar la curva de separación. Vemos como en el caso de 1 y 3 neuronas en la capa oculta mantienen un error cuadratico medio alto luego de haber llegado a un mínimo despues de aproximadamente 1.000 iteraciones y el mismo se mantiene constante.

Observamos, tambien, que en los casos donde el numero de neuronas en la capa oculta es mayor al número de entradas la red aprende la tarea pero esto requiere significativamente mas iteraciones.

3. Mapeo logístico

En esta sección se busca aprender utilizando retropropagación el mapeo logístico $x(t+1) = 4x(t)(1-x(t))$ en la arquitectura que se muestra en la **Figura 5**.

Para ello se tiene en cuenta las siguientes consideraciones:

- La función de activación de las neuronas en la capa oculta es $g(x) = 1/(1+exp(-x))$.
- Las neuronas incluyen umbrales.
- La función de activación de la neurona de salida es lineal.
- Se busca generar una base de datos $x(t), x(t=1)$ iterando el mapeo y presentar 5, 10 y 100 ejemplos.
- Se toma un 80% de los ejemplos presentados y se testea con el 20% de los ejemplos no presentados.

Como objetivo, se busca comparar el error de entrenamiento con el error de generalización.

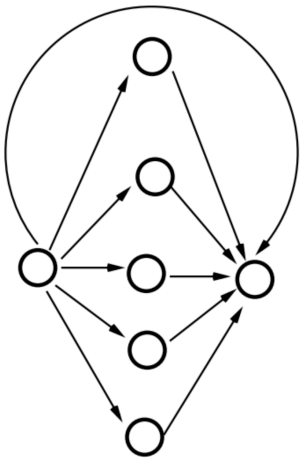

Figure 5: Arquitectura propuesta para aprender el mapeo logistico mediante retropropagación.

El modelo se implementó utilizando la libreria de python `tensorflow` y la arquitectura del modelo se muestra en la **Figura 6**.

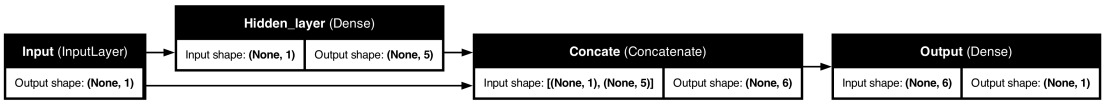

Figure 6: Arquitectura de la red propuesta utilizando la libreria de python *tensorflow*.

Para entrenar el modelo se utilizó el 80% de los datos generados y se testeo con el 20% restante. Se entreno por 1500 epocas y se utilizó un tamaño de batch de 4. Los resultados obtenidos se muestran en la **Figura 7**.

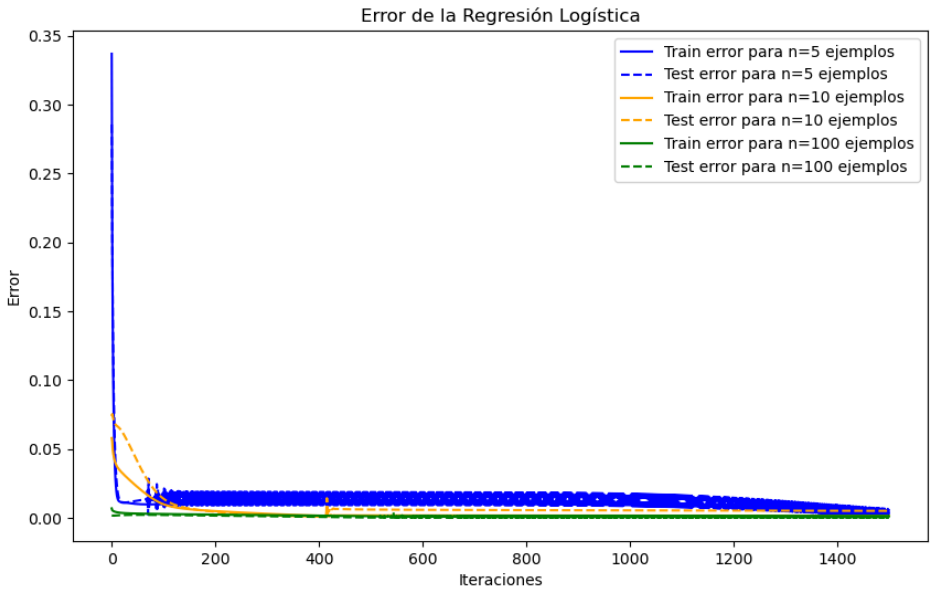

Figure 7: Error cuadrático medio en función del numero de iteraciones para 5, 10 y 100 ejemplos utilizando el 80% para entrenamiento y el 20% para testeo.

De la **Figura 7** se puede observar que en todos los casos pareciera aprender. Tambien vemos que el error disminuye rapidamente en las primeras iteraciones y luego se mantiene practicamente constante. El caso de 10 y 100 ejemplos tienen un comportamiento similar pero el error es menor en el caso de 100 ejemplos. Con respecto al error de generalización el error es menor para el caso de 100 ejemplos indicando que el modelo generaliza mejor con mas datos. El caso de 5 ejemplos es un caso especial ya que debido a la poca cantidad de datos el error de generalización es mayor que el error de entrenamiento.

Una vez entrenado el modelo se puede comparar los valores predichos con los valores reales. En la **Figura 8** se muestra la comparación entre los valores reales y los valores predichos para los distintos casos. Se muestra ademas, los datos de entrenamiento y testeo.

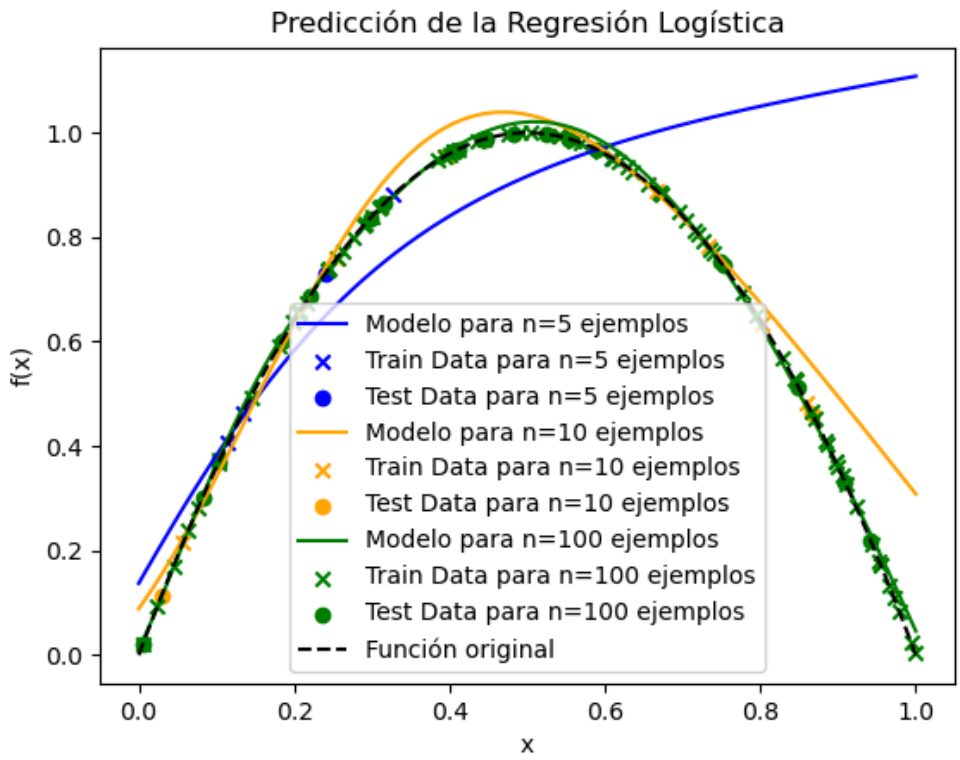

Figure 8: Comparación de la prediccion de la regresión logistica para los distintos casos de 5, 10 y 100 ejemplos con la curva real del mapeo logistico.

Al comparar los distintos modelos con la curva real del mapeo logistico vemos que el error tiende a cero por que se logra ajustar la curva con los ejemplos de entrenamiento. Sin embargo al generalizar el error aumenta. El error de generalización disminuye a medida que aumenta el numero de ejemplos utilizados para entrenar el modelo. Como caso extreme vemos el modelo para 5 ejemplos que al utilizar el 80% para entrenamiento significo solo utilizar 4 ejemples para entrenar. Dependiendo de que tan distribuidos esten los ejemplos en el espacio de entrada el modelo puede generalizar mejor o peor. Al tener mas ejemplos, hay una mayor probabilidad de que se hayen mejor distribuidos y el modelo generalice mejor.