Redes Neuronales
Práctica 4 - Aprendizaje supervisado en redes multicapa

F. M. Cabrera

27 de abril de 2021

Ejercicio 1

Para este ejercicio se busco resolver el problema de XOR a partir de dos arquitecturas distintas, las cuales pueden observarse en la Figura 1 y cuya principal diferencia es que una presenta una reinyección de la entrada mientras que otra no. En ambos casos, la función de activación de la capa oculta fue tanh y como función de costo se utilizo MSE. Debido a que la función de activación utilizada nunca retorna 1 o -1, se determino un umbral de 0.9 en la función de costo, de manera que si el resultado supera dicho umbral, se considera la salida como 1. De manera análoga, se determino un segundo umbral de -0.9 para determinar cuando la salida se considera un -1.

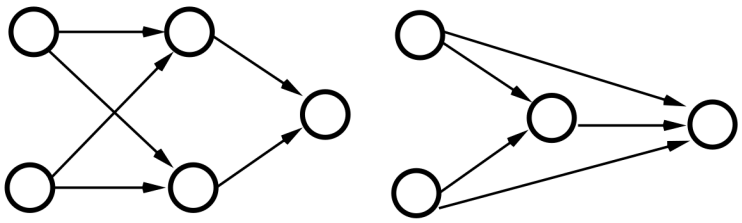

Figura 1: Esquema de las arquitecturas utilizadas para resolver el problema del XOR.

En ambos casos se utilizo la totalidad de los datos como datos de entrenamiento. Los resultados para la función de costo y el *accuracy* para la primer arquitectura en 10 procesos de entrenamiento independientes de la red pueden observarse en la Fig. 2, mientras que en la Fig. 3 se observan los resultados análogos para la segunda arquitectura. En ambos casos se observa que algunos procesos de entrenamiento no convergen y no alcanzan un valor de *accuracy* optimo del 100 %.

Se definió el tiempo de convergencia como aquella época en donde la función de costo alcanza un valor de 0,1, pero solo para los modelos que convergieron a una solución optima dentro de las 2000 épocas. Para comparar los tiempos de convergencia entre ambas arquitecturas, se entrenaron 1000 modelos independientes que lograron converger y se obtuvo su tiempo de convergencia, para ambas arquitecturas. Para la primer arquitectura se obtuvo un tiempo de convergencia promedio de 157,5 épocas, mientras que para la segunda se obtuvo un valor de 148,7. En principio, esta diferencia, aunque sea pequeña, podría indicar que la segunda arquitectura es mejor para la resolución del problema XOR. Sin embargo, para obtener 1000 modelos entrenados que hayan logrado converger a una solución optima, fue necesario entrenar 1288 modelos para la primer arquitectura, mientras que para la segunda fueron necesarios 1983. Es decir, que el 77,6 % de los modelos creados con la primer arquitectura lograron converger, mientras que para la segunda solo el 50,4 %, lo cual apunta a que la primer arquitectura es mas satisfactoria para resolver el problema.

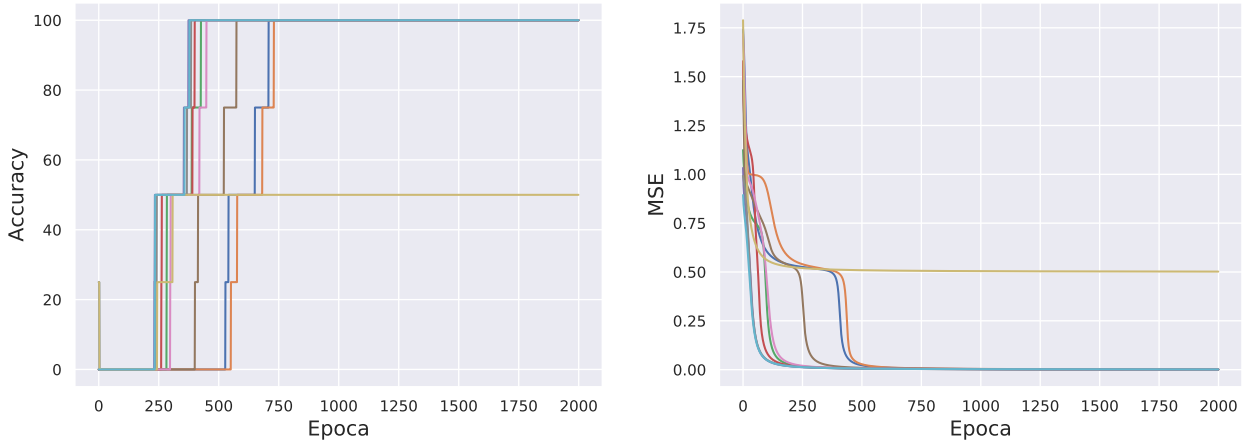

Figura 2: Función de costo MSE y *accuracy* para el entrenamiento de 10 modelos independientes, utilizando la primer arquitectura propuesta en Fig. 1.

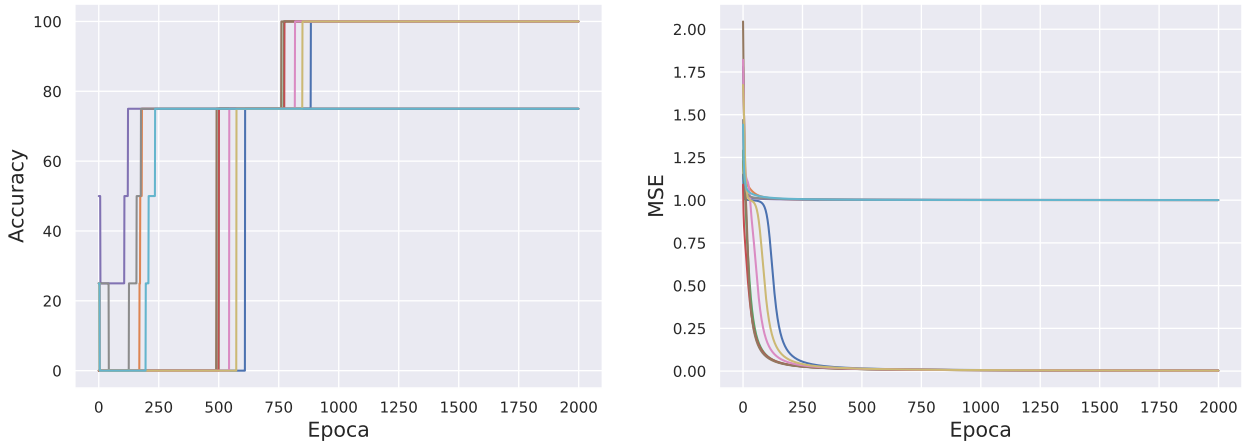

Figura 3: Función de costo MSE y *accuracy* para el entrenamiento de 10 modelos independientes, utilizando la segunda arquitectura propuesta en Fig. 1.

Ejercicio 2

Se busco estudiar una generalización del problemas de XOR para N entradas, en donde la salida esperada es el producto de todas las componentes de la entrada. La arquitectura utilizada consistió en una capa oculta de N' neuronas con una función de activación `tanh`, seguida de una capa de salida de 1 neurona tambien con activacion `tanh`. Debido a que la cantidad de datos de entrada es igual a 2^N y era de interés estudiar los casos en donde $N > N'$ y $N < N'$, se fijo N a un valor de 5 y se entreno la red para distintos valores de N': 1, 3, 5, 7, 9 y 11, en todos ellos durante 5000 épocas.

En la Figura 5 se observan los resultados obtenidos para la función de costo y *accuracy* en cada uno de los casos. En caso de que $N' > N$, los resultados son similares a los del ejercicio anterior, ya que en una pequeña cantidad de épocas se obtiene un 100 % de precisión. La situación se vuelve mas problemática al disminuir N', en donde se observa que para $N' = N = 5$ el aprendizaje de la red nunca alcanza una precisión total e incluso no mejora de manera monótona conforme avanzan las épocas. Los resultados son incluso peores cuando $N' < N$, en donde se requieren muchas mas épocas para que la red aprenda e incluso para $N' = 1$ nunca se logra alcanzar un valor de *accuracy* no nulo. Este problema es esperable, ya que el valor de N' determina las dimensiones de las matrices de pesos de cada una de las capas. Al disminuir N', menor sera la cantidad de componentes de estas matrices, con lo cual deberá ajustar una gran cantidad de datos de entrada (2^N) con menos grados de libertad, que, en caso de ser demasiado pocos, se presentara un problema de underfitting y la red no mejorara

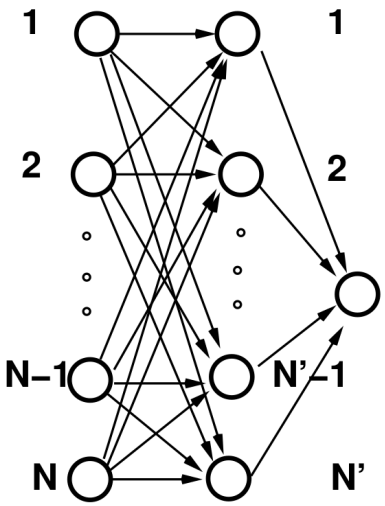

Figura 4: Esquema de la arquitectura utilizada para resolver el problema del XOR generalizado.

sus resultados.

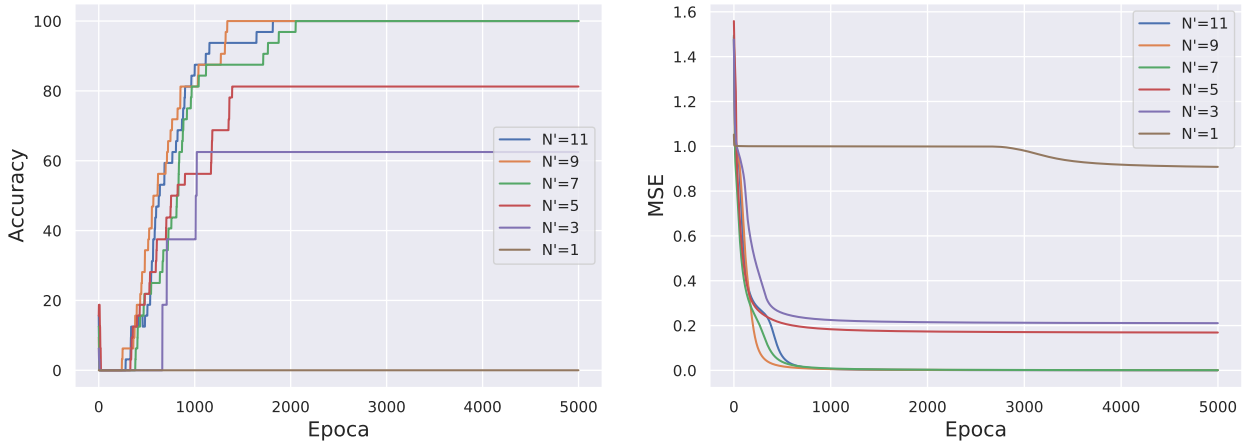

Figura 5: Función de costo MSE y *accuracy* para el entrenamiento de 6 modelos independientes variando la cantidad de neuronas de la capa oculta, utilizando la arquitectura propuesta en Fig. 1.

Ejercicio 3

En este caso se busco implementar una RN para resolver el mapeo logístico dado por la ecuación $x(t+1) = 4x(t)(1-x(t))$. En este caso la arquitectura de la red puede observarse en la Fig. 6, consistiendo en en una capa densa oculta con activación `sigmoide` y una capa de salida de una neurona con activación lineal. Como función de costo se utilizo MSE.

Se utilizaron tres valores distintos para la cantidad de datos de entrenamiento, siendo estos 5, 10 y 100. En los tres casos, los datos fueron tomados aleatoriamente con una distribución uniforme entre 0 y 1. La red se entreno de manera que realizar una predicción de un dado dato sea equivalente a realizar una iteración en el mapeo logístico. El entrenamiento se realizo durante 1000 épocas. Como conjunto de validación y de test se usaron dos conjuntos de 100 datos con la misma distribución que los datos de entrenamiento. El conjunto de validación se utilizo para seleccionar los hiperparametros, mientras que el de test se utilizo para evaluar la *performance* final de la red luego de concluir el entrenamiento.

En la Fig. 7 se observan los resultados obtenidos para la función de costo tanto para los datos de entrenamiento como los de validación, variando la cantidad de ejemplos con los cuales la red es entrenada. Como es de esperarse, la función de costo es menor cuando mayor cantidad de datos de

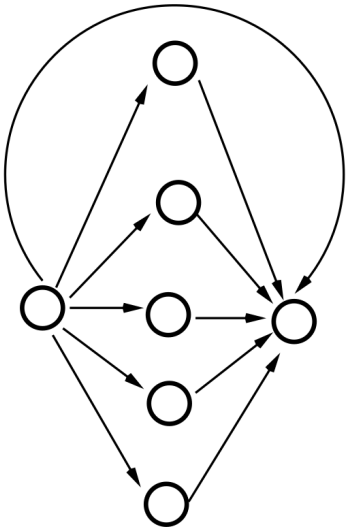

Figura 6: Esquema de la arquitectura utilizada para resolver el problema del mapeo logístico.

entrenamiento son proporcionados a la red. En todos los casos, el error con los datos de test luego de entrenar fueron cercanos a los valores finales del error con los datos de validación. Cabe destacar que, para la red entrenada con 5 ejemplos, el error con los datos de validación es menor que los obtenidos con los datos de entrenamiento, lo cual no es comportamiento típico, ya que lo usual es que el error de entrenamiento sea menor, ya que es este error el que la red buscara minimizar, independientemente lo que suceda con el error de validación. Sin embargo, la situación no es imposible y puede deberse solo a la poca cantidad de datos presentados para el entrenamiento.

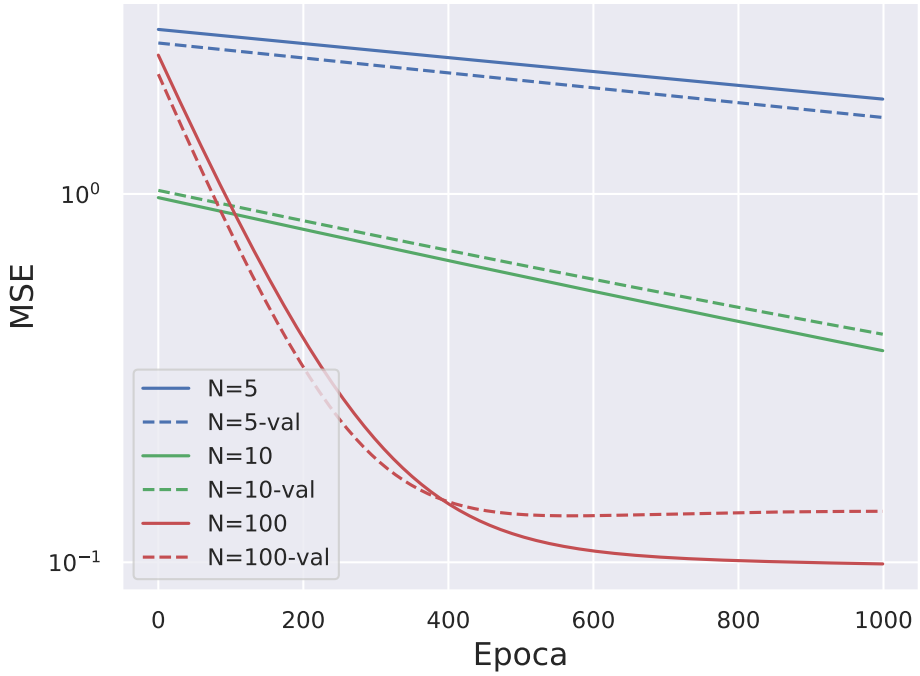

Figura 7: Evolución de la función de costo (presentada en escala logarítmica) tanto para los datos de entrenamiento como los de validación, para los tres modelos entrenados variando la cantidad de ejemplos utilizados para el entrenamiento.

Luego, solo para estudiar la utilidad de la red neuronal para estudiar la evolución de un sistema dado por el mapeo logístico, se entreno un nuevo modelo con la misma arquitectura, pero esta vez

utilizando 1000000 datos de entrenamiento, equispaciados entre 0 y 1. Como conjuntos de validación y de test se utilizaron dos conjuntos de 500 datos aleatorios entre 0 y 1. A la izquierda en la Fig. 8 se observa la evolución de la función de costo para los datos de validación y de entrenamiento a lo largo de 200 épocas.

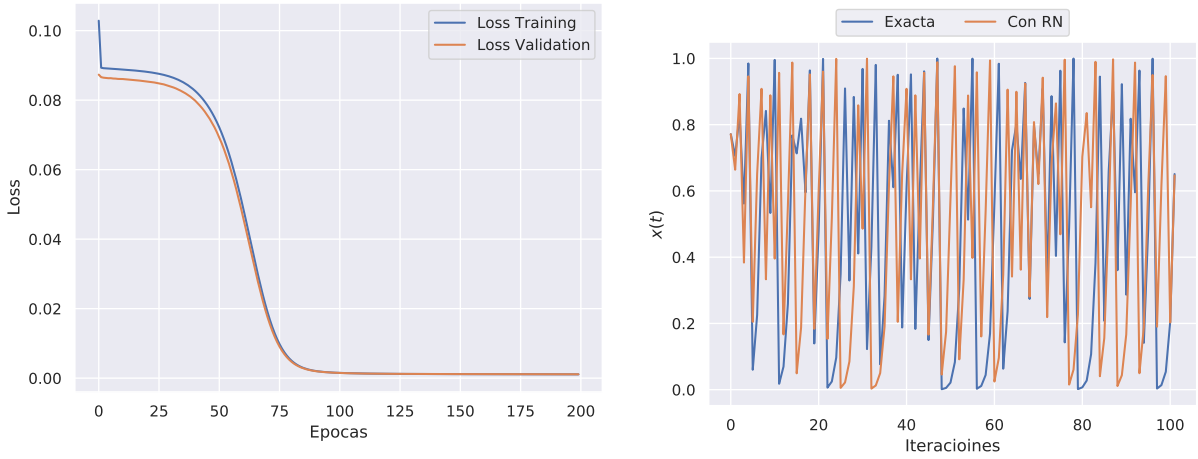

Figura 8: A la izquierda, se observa la función de costo durante el entrenamiento de la red. Por otro lado, a la derecha, se observa una comparación entre la evolución del mapeo logístico utilizando la red entrenada y utilizando la red exacta. Se observa que la red entrenada no logra reproducir la evolución del sistema.

Un vez finalizado el entrenamiento de la red, se busco comparar su rendimiento respecto a la evolución exacta. Para esto, se realizaron 100 iteraciones utilizando tanto la red entrenada como la ecuación exacta, con una misma semilla (es decir, un valor de $x(t = 0)$) inicializada de manera aleatoria entre 0 y 1. A la derecha de la Figura 8 se observa una comparación de la evolución del problema a partir de ambos métodos. Se observa que el resultado obtenido a partir de la red neuronal no se corresponde con el obtenido de manera exacta, incluso para las primeras iteraciones. Esto se debe al carácter caótico de la del mapeo, en donde una pequeña perturbación de las condiciones iniciales produce un comportamiento notablemente distinto a medida que evoluciona el sistema. Dado que la red entrenada no predice los valores con una precisión absoluta, este error cometido en cada iteración conlleva a evolución significativamente distintas. De este análisis se desprende que el uso de redes neuronales para el estudio de sistemas caóticos puede conllevar a resultados erróneos.