REDES NEURONALES - PRÁCTICA 4 - 2024

Aprendizaje supervisado en redes multicapas

Maximiliano Gatto

Instituto Balseiro (UNCuyo - CNEA) - Bariloche, Río Negro, Argentina

maximiliano.gatto@ib.edu.ar

15 de octubre de 2024

1. Introducción

En esta práctica se analizaron distintas arquitecturas de redes neuronales para resolver diversos problemas algorítmicos. Para el entrenamiento de las redes, se implementó el algoritmo de retropropagación de errores (*backpropagation*). Se establecieron métricas clave para evaluar el rendimiento de las redes, y se emplearon para compararlas entre sí. Todos los ejercicios fueron implementados mediante un script en `Python`, cuyo código está disponible en este enlace.

2. Resultados

Ejercicio 1

En este ejercicio, se implementaron dos arquitecturas de redes neuronales (ver Figura 1) para resolver el problema de una compuerta XOR de 2 entradas, donde cada entrada puede tomar valores ± 1. La salida es -1 si ambas entradas son iguales y 1 si son diferentes. Se utilizó la función de activación $\tanh(x)$ con umbrales (*bias*). El entrenamiento de cada red se realizó con el algoritmo de retropropagación de errores (*backpropagation*) durante 2000 épocas, con una tasa de aprendizaje (*learning rate*) de $lr = 0.1$.

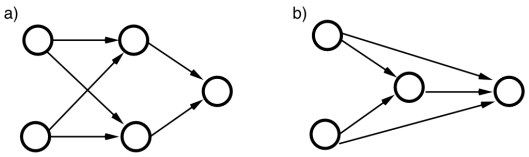


Figura 1: *(a) arquitectura 1, (b) arquitectura 2*

Para analizar la convergencia del algoritmo, se implementaron dos métricas. La primera es la *accuracy* o precisión, que mide cuán bien la red clasifica las instancias de un conjunto de datos. Se calcula como:

$$\text{accuracy} = \frac{N°\text{pred. correctas}}{N°\text{pred. totales}} * 100, \quad (1)$$

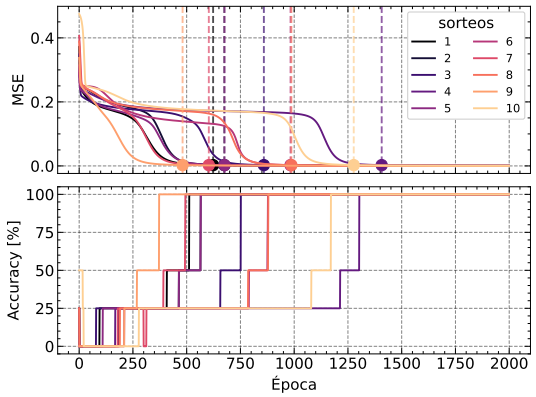
es decir, el porcentaje de instancias clasificadas correctamente sobre el total. Esta definición (Ecuación 1) se aplica a redes con salidas discretas. En nuestro caso, como la salida es continua, consideramos una predicción correcta si $|y - o| < \epsilon$, donde y es la salida deseada, o es la salida de la red, y ϵ es una constante pequeña, con un valor de $\epsilon = 0.1$ en este trabajo.

La segunda métrica utilizada es el error cuadrático medio (MSE, *mean squared error*), definido como:

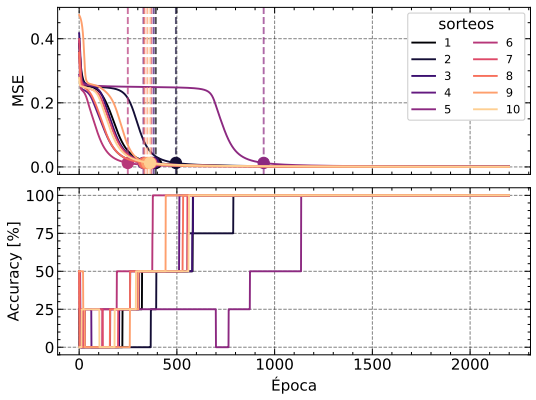
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - o_i)^2, \qquad (2)$$

donde N es el número de ejemplos de entrenamiento. Un MSE (Ecuación 2) más bajo indica que las predicciones son más cercanas a los valores reales.

Se analizó el tiempo medio de convergencia de ambas arquitecturas tras 10 condiciones iniciales aleatorias, considerando como convergencia la época en que MSE < 0.01. El tiempo medio es el promedio de cada sorteo. La Figura 2 muestra la evolución del accuracy y el MSE para cada arquitectura.



(a)



(b)

Figura 2: *métricas obtenidas durante el entrenamiento para diferentes sorteos de condiciones iniciales. El punto y la línea de trazos representa la época en la cuál se considera la convergencia. (a) arquitectura 1 y (b) arquitectura 2.*

Como resultado, se obtuvo que la arquitectura 1 tiene un tiempo medio de convergencia de 860(30) épocas, mientras que la arquitectura 2 converge en promedio en 420(20) épocas. Por lo tanto, el resultado que se obtuvo es que la arquitectura 2 converge en promedio más rápido. Sin embargo, este resultado depende de la semilla inicial que se establece en el sorteo.

Ejercicio 2

En este ejercicio se implementó una red, como la de la Figura 3, para resolver el problema de la paridad, una generalización del XOR para N entradas. La salida es 1 si el producto de las entradas es 1 y -1 si es -1.

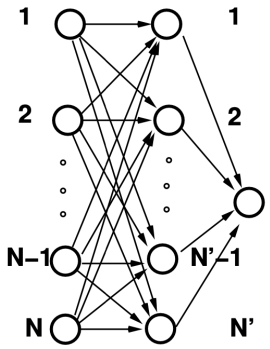


Figura 3: *arquitectura del ejercicio 2 para resolver el problema de la paridad.*

Las redes fueron entrenadas con el algoritmo de retropropagación durante 40000 épocas, usando una tasa de aprendizaje ($lr = 0.1$) con $N = 5$ entradas y variando el número de neuronas en la capa oculta ($N' = 1, 3, 6, 7, 9, 11$). Para evaluar el desempeño, se calcularon el *accuracy* y el error cuadrático medio (MSE). Los resultados se presentan en la Figura 4.

La Figura 4 muestra que cuando $N' < N$, el MSE es más alto y el accuracy no alcanza el 100 % (logrado por el epsilon en el conteo de predicciones correctas). En particular, con $N' = 1$, la red no aprendió el algoritmo correctamente. El error disminuye y el accuracy mejora a medida que N' se acerca a N. Para $N' > N$, tanto el MSE como el accuracy son similares, indicando que la red puede resolver el problema de paridad, mientras que

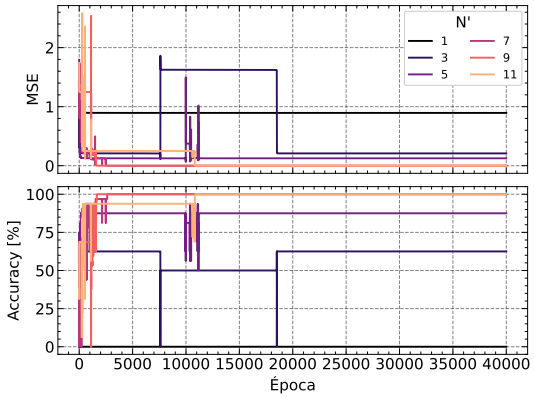


Figura 4: *métricas obtenidas durante el entrenamiento para diferentes valores de la capa oculta en la arquitectura del ejercicio 2.*

con $N' < N$, el desempeño es peor y empeora a medida que N' disminuye.

Ejercicio 3

Se implementó una red neuronal, según la arquitectura de la Figura 5, para aprender el mapeo logístico $x(t+1) = 4x(t)(1-x(t))$ mediante el algoritmo de retropropagación. La capa oculta utilizó la función de activación sigmoide $g(x) = 1/(1+\exp(-x))$, y la neurona de salida una función lineal. La red contaba con umbrales y 5 neuronas en la capa oculta. El entrenamiento se realizó en 500 iteraciones utilizando el paquete `TensorFlow`.

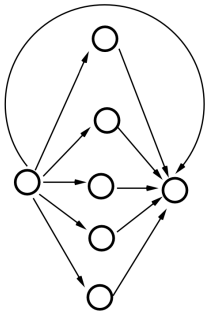


Figura 5: *arquitectura del ejercicio 3 para aprender el mapeo logístico.*

Se generaron aleatoriamente los valores de $x(t)$ y se aplicó la regla de selección. Se evaluó el desempeño de la red con 5, 10 y 100 ejemplos de entrenamiento, comparando el error de entrenamiento y el de generalización. Los resultados se presentan en la Figura 6.

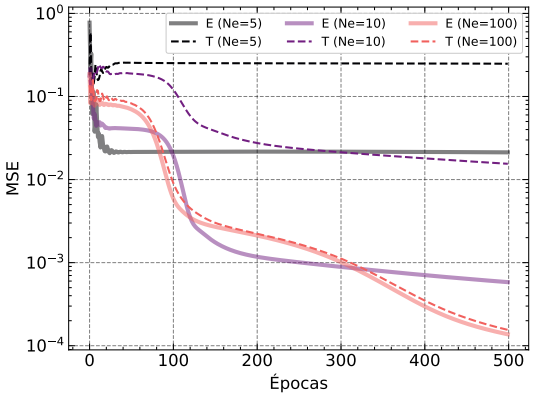


Figura 6: *métricas obtenidas durante el entrenamiento para diferentes cantidades de ejemplos de entrenamiento. E: entrenamiento, T: verificación o testing.*

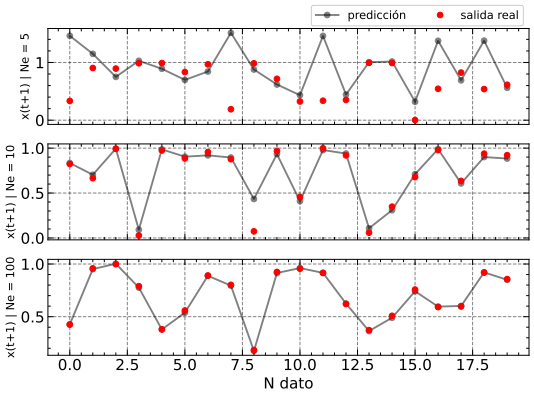


Figura 7: *predicciones de la red luego del entrenamiento para 20 datos de entrada diferentes a los del entrenamiento. Ne indica el número de datos utilizados para el entrenamiento.*

La Figura 6 muestra que aumentar el número de ejemplos en el entrenamiento reduce el error de generalización, lo que aumenta la probabilidad de obtener la salida correcta para datos no presentados durante el entrenamiento. La Figura 7 presenta 20 ejemplos de verificación y las salidas generadas por la red.

La Figura 7 confirma el comportamiento descrito anteriormente: con pocos ejemplos de entrenamiento, la red presenta un mayor error en las predicciones, pero este error disminuye a medida que aumenta la cantidad de ejemplos de entrenamiento.