Redes Neuronales Multicapa

Sistemas de Inteligencia Artifical - ITBA

Carlos Sessa Lucas Pizzagalli Nicolás Purita

Resumen

Se implementó una red neuronal supervisada multicapa para resolver los siguientes problemas:

1. **Paridad**
2. **Simetría**

Ambos problemas poseen una entrada de N bits, con $2 \leq N \leq 5$.

Desarrollo

Las entradas para estos problemas son todas las combinaciones posibles de un arreglo de N bits y la salida esperada es representada como un bit. Este bit se enciende cuando se cumple la condición de simetría o paridad según que función lógica se desee probar.
Se corren distintos experimentos, variando parámetros y arquitecturas con el fin de lograr disminur el tiempo de entrenamiento para la red neuronal implementada.
Dada que la salida es binaria, se utilizaron distintas funciones de activación en distintas capas. En la capa oculta se utiliza una función de activación *no lineal* y en la última capa, la de salida, se utiliza un función *lineal*. La arquitectura elegida, con el criterio de mejor performance a la hora de entrenarla, es de N entradas, N neuronas en la primer capa oculta y una única neurona en la capa de salida. Esta arquitectura se acomoda correctamente a los problemas 1 y 2.
Cabe destacar que al hacer las pruebas encontramos redes de menor tamaño que resolvían el problema. Por ejemplo, si usamos $N = 5$, el problema se puede resolver con $N-2$ neuronas en la capa oculta. Dado que poder elegir un número de neuronas menor a N es posible pero no se cumple para todos los N preferimos hacer las pruebas correspondientes con N neuronas en la capa oculta.

Resultados obtenidos

En la figura 1 se puede ver como utilizando la tangente hiperbólica como función de activación para el problema de la *Simetría* se logra un mejor error que utilizando la exponencial.
En la figura 2 se puede observar como el problema de *Paridad* utilizando la función de activación exponencial no se llega a entrenar la red con un error menor al deseado,*0.01*, en un lapso de 5000 épocas a diferencia de la función tangencial, donde la red logra entrenarse en 1487 épocas. El motivo por el cual la función de activación exponencial no termina en 5000 épocas es porque no se realiza ninguna corrección de los pesos ya que es 0 el valor de la función.
Como se observa en las figuras 3 y 4 al poner una funció lineal en la última capa la red es entrenada en menos tiempo, es decir en menos épocas. Esto se debe a que en la última capa la estructura de la red se comporta como un perceptron simple y el problema en ese nivel es linealmente separable. Por este motivo es mejor utilizar una función *lineal* frente a una *no-lineal*.

Conclusiones

La primera es que, dado que la salida de la red es binaria, podemos mejorar el tiempo de entrenamiento poniendo una función de activación *escalón* en la capa de salida. Para aplicar esto debemos realizar la transformación de la entrada de mapear los 0 al -1.
El segundo punto a destacar es el peso de la aleatoriedad de los pesos. Por ejemplo, notamos que poniendo distintas semillas se puede lograr que la red con $N = 2$ tarde menos en entrenarse que con $N = 5$.

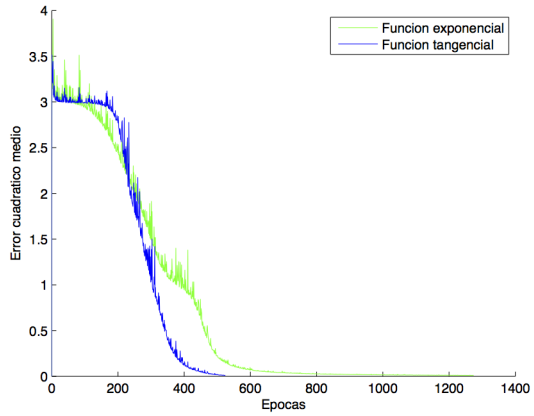

Figura 1: Comparación del error para distintas funciones de activación con $N = 5$ en el problema de Simetría

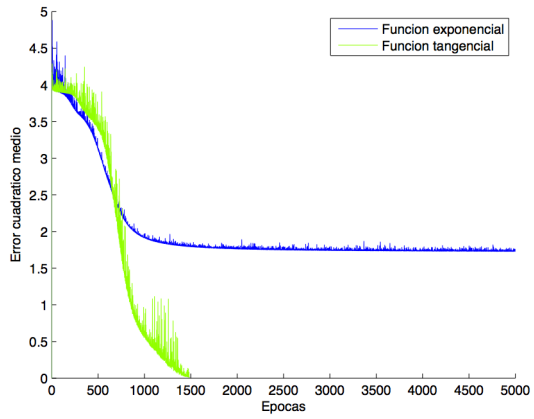

Figura 2: Comparación del error para distintas funciones de activación con $N = 5$ en el problema de Paridad

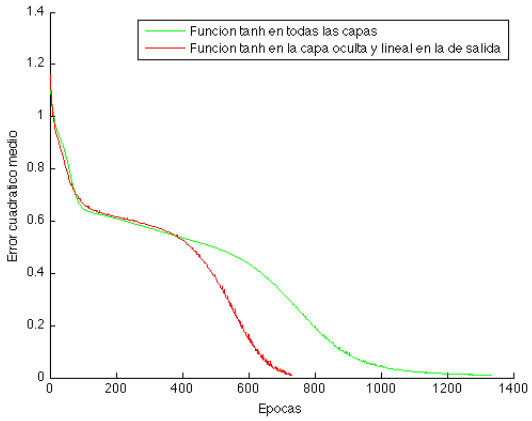

Figura 3: Comparación del error para distintas funciones de activación en la última capa, con $N = 3$ en el problema de Paridad

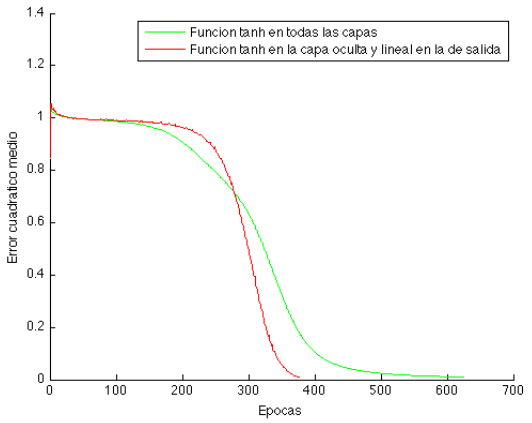

Figura 4: Comparación del error para distintas funciones de activación en la última capa, con $N = 3$ en el problema de Simetría