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, donde se encuentra encendido siempre y cuando se cumpla la condición de simetría o paridad.
Se corren distintos experimentos, variando parámetros y arquitecturas con el fin de lograr el mejor entrenamiento para la red neuronal implementada.
Un tema a destacar es la utilización de distintas funciones de activación que con el fin de comparar el aprendizaje de la red con cada una de ellas. Las mismas fueron *lineal*, *tangente hiperbólica* y *escalón*. para esta última se decidió utilizar una entrada modificada cambiando el número 0 por -1.
Durante las pruebas, se comprobó que el η elegido era determinando en cuanto a la velocidad de aprendizaje como también decisivo a la hora de poder enseñar o no a la red, por lo que se decidió hacer una busqueda exaustiva variando η entre 0.01 y 0.2 con un paso de 0.01 y con una cantidad máxima de 15000 epocas en cada prueba. Considerando que si se supera dicho tiempo, no se ha logrado entrenar a la red. Esta cota fue elegida al demorar una cantidad grande de tiempo en relación al tiempo que se tiene para realizar la investigación.
DE ACA A ABAJO A COMPROBAR ES TODO LO VIEJO! !!! 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

A simple vista se puede observar en todos los gráficos, que en caso de lograr entrenar a la red con el error requerido, es más fácil al tener una menor cantidad de bits de entrada. También se puede observar en las figuras 1 y 2 que al utilizar la función *tanh* se ha logrado llegar a una solución satisfactoria en todos los casos, mientras que al utilizar la función lineal (figuras 3 y 4, el error nunca disminuye lo suficiente como para considerarlo aceptable, pudiendo determinar que el entrenamiento no fue concretado.
En cuenta a la función escalonada, se puede observar en la figura 5 que solo logra entrenar redes con pocas entradas, mientras que falla al aumentar las mismas (ver 6). También se puede observar que el error es muy inestable durante todo el período de entrenamiento (teniendo picos de error muy altos). Esto se debe al ser una función que varía entre sus extremos de manera abrupta. Esta también es la razón por la cual llega a encontrar solución con un número bajo de entradas, ya que en una de sus picos hacia abajo llega a encontrar una configuración con error aceptable.

Conclusiones

- Función lineal no sirve.
- η es desisiva
- algun otro chamullo

DE ACA HACIA ABAJO VIEJO!!
Al realizar este trabajo notamos dos cosas importantes. 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. La segunda es notar lo tanto que pesa la aleatoriedad de los pesos. Por ejemplo, notamos que poniendo distintas semillas podíamos lograr que la red con $N = 2$ tarde menos en entrenarse que con $N = 5$.

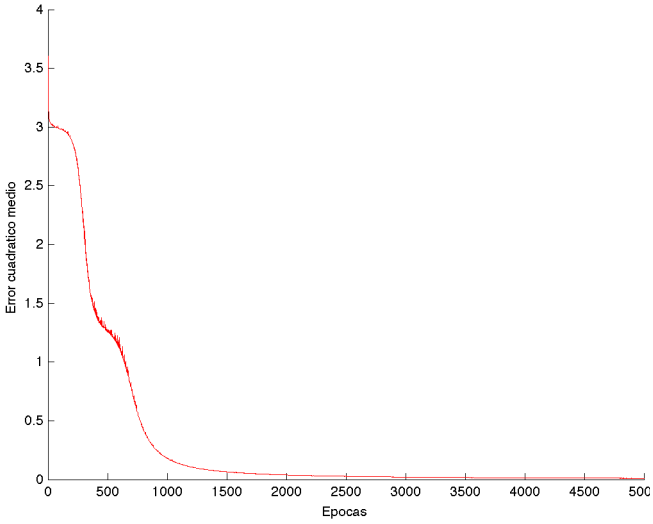

Figura 1: Comparación del error para distintas funciones de activación con N = 5 en el problema de Simetriía

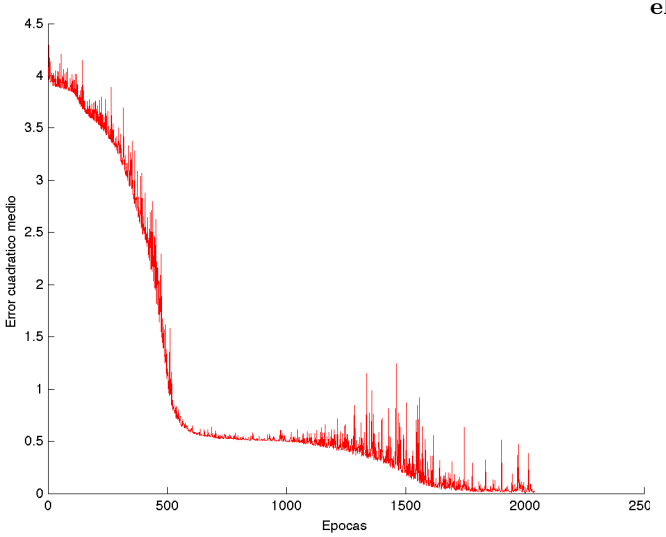

Figura 2: Comparación del error para distintas funciones de activación con N = 5 en el problema de Paridad

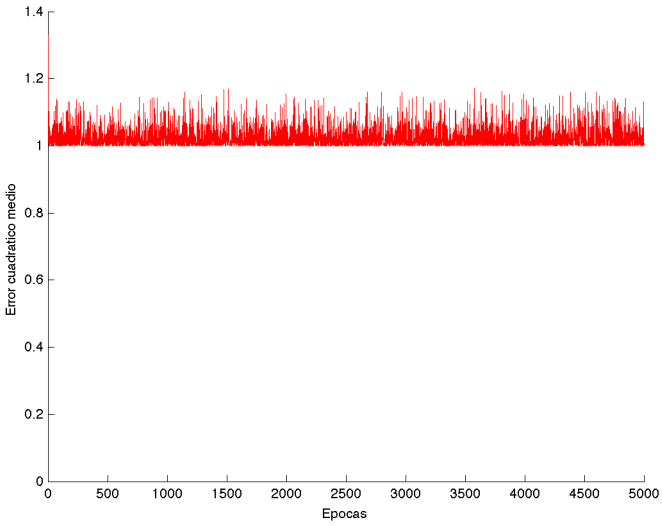

Figura 3: Comparación del error para distintas funciones de activación en la última capa, con N = 3 en el problema de Simetría

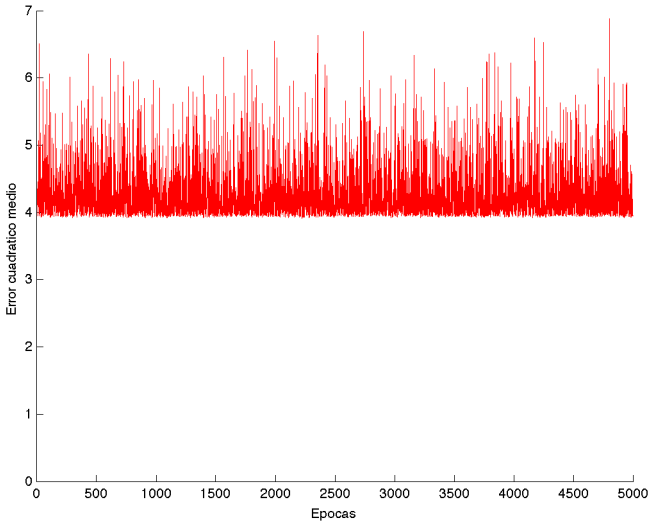

Figura 4: Comparación del error para distintas funciones de activación en la última capa, con N = 5 en el problema de Paridad

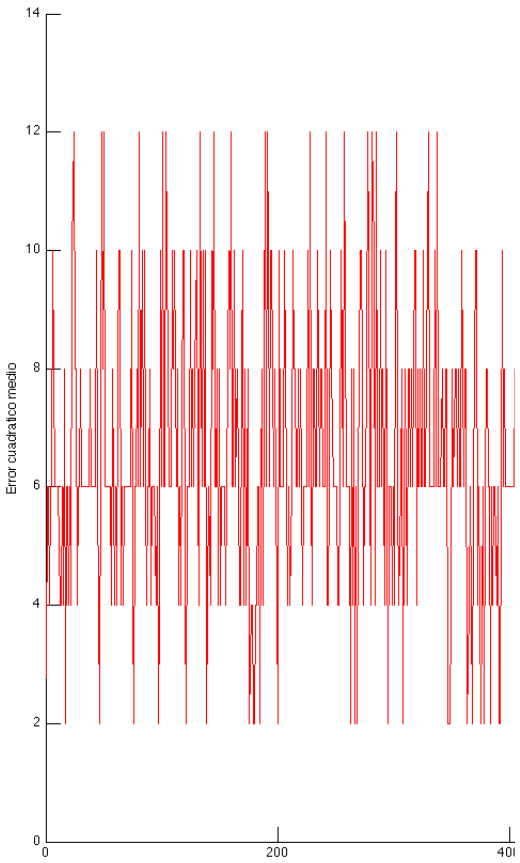

Figura 5: Comparación del error para distintas funciones de activación en la última capa, con N = 3 en el problema de Paridad

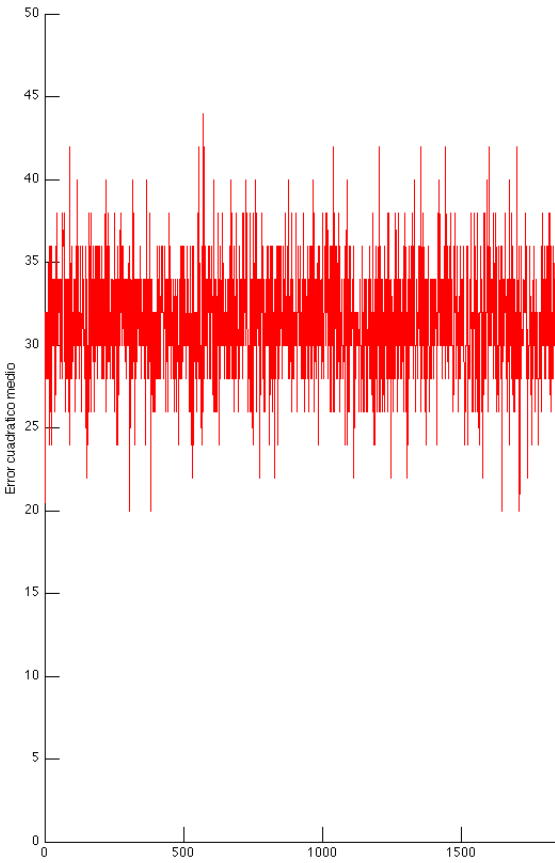

Figura 6: Comparación del error para distintas funciones de activación en la última capa, con N = 3 en el problema de Simetría