Práctica 4: Aprendizaje supervisado en redes multicapa

Evelyn G. Coronel
Redes Neuronales - Instituto Balseiro
(22 de abril de 2020)

En esta práctica se realiza el aprendizaje supervisado para distintas arquitecturas y problemas usando el algoritmo de retropropagación.

EJERCICIO 1: XOR

Casos base para el entrenamiento

Para este problema básico, usamos todos los casos posibles para el entrenamiento, es decir, no tenemos ejemplos para verificar la red neuronal. Estos casos se presentan en la Tabla I.

Entrada 1	Entrada 2	Salida
-1	-1	+1
-1	+1	-1
-1	-1	-1
+1	+1	+1

Tabla I: Todos los casos posibles para la función XOR.

Se utilizaron dos arquitecturas para el algoritmos de retropropagación, además de agregar una neuronas adicionales para simular los umbrales. La función de transferencia para la salida fue $\tanh(x)$ y la función error que se minimiza es la función error cuadrático medio. En todas las simulaciones se usó un *learning rate* $\eta = 0.05$, que cuantifica el cambio de los pesos en función de las épocas.

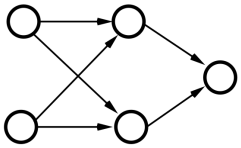

Fig. 1: Arquitectura 2-2-1

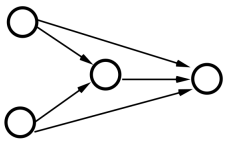

Fig. 2: Arquitectura 2-1-1

Arquitectura 2-2-1

Un esquema de la arquitectura utilizada en este caso se muestra en la Fig. 1. En la misma se observa dos entradas, una capa oculta con 2 neuronas conectadas a cada entrada, y una salida conectada a la capa oculta.

Pérdida en función de la época

La evolución de la pérdida, que en este caso es la función error cuadrático medio, para distintas condiciones iniciales para los pesos y el umbrales se muestra en la Fig.3. Se observa que en el trial 5 para esta ejecución del programa, convergen a un mínimo local de la función error.

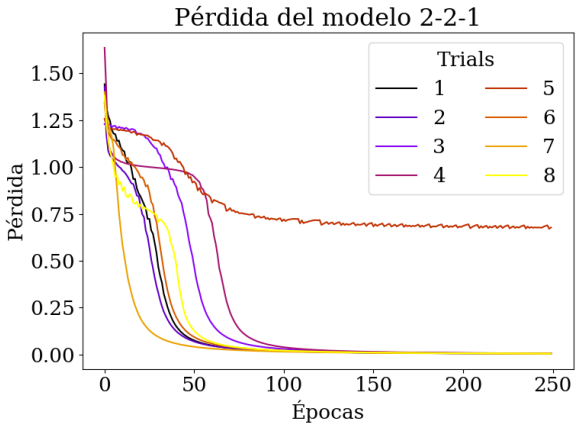

Fig. 3: Pérdida en función de las épocas para la arquitectura 2-2-1 para el problema del la función XOR para distintas condiciones iniciales.

Si consideramos la exactitud de la red neuronal, se observa que para el trial 5, su valor es de 0.5. Esto indica que la red neuronal predice la salida del 50 % de los casos de entrenamiento, es este caso 2 ejemplos de 4 en total.

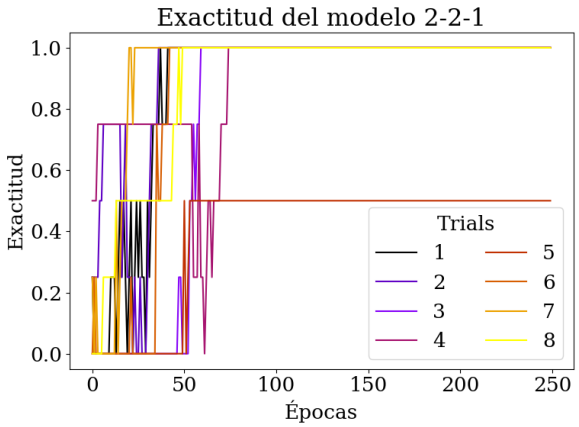

Fig. 4: Exactitud en función de las épocas para la arquitectura 2-2-1.

Arquitectura 2-1-1

Un esquema de la arquitectura utilizada en esta sección se observa en la Fig. 2. En la misma se observa dos entra-

das de la función XOR, una capa oculta con 1 neurona conectada a cada entrada, y una salida conectada a la capa oculta además de estar conectada a las entrada de la red.

Pérdida en función de la época

La Fig. 5 muestra la evolución de la pérdida con las épocas de la ejecución del programa para distintas condiciones iniciales. En la misma se ve como la red converge a una pérdida nula sólo para 3 de los 8 casos en las primeras 250 épocas. Esto se debe a que la cantidad de parámetros que tiene la red para ajustar la función XOR no es suficiente para regresionar la función. La cantidad de parámetros viene dada por la arquitectura del problema.

En la evolución de la exactitud con las épocas de la Fig. 6, muestra que los casos donde no se converge a una solución, la red es capaz de predecir 3 de los 4 ejemplos usando en el entrenamiento, ya que la exactitud vale 75 %.

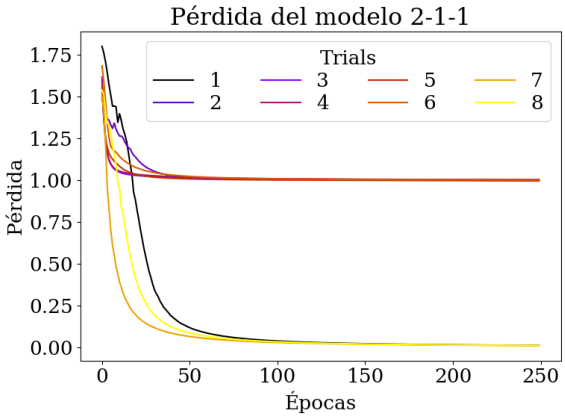

Fig. 5: Pérdida en función de las épocas para la arquitectura 2-1-1 para el problema del la función XOR.

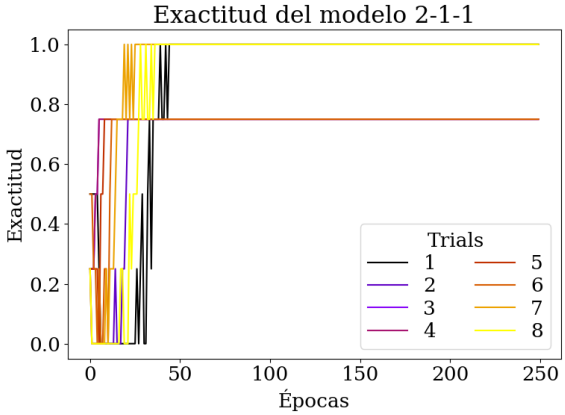

Fig. 6: Exactitud en función de las épocas para la arquitectura 2-1-1.

EJERCICIO 2: XOR GENERALIZADO

A diferencia del ejercicio anterior del XOR, ahora la función tiene N entradas, N′ neuronas en la capa oculta conectadas a las entradas, y una salida conectadas a todas las neuronas de la capa oculta, además de entradas adicionales para simular los umbrales . Esta arquitectura se muestra en la Fig. 7. La salida es +1 si el producto de las N entradas es +1 y −1 si el producto de las entradas es −1. Al igual que el ejercicio anterior, se utilizó la función $tanh(x)$ como función de activación.

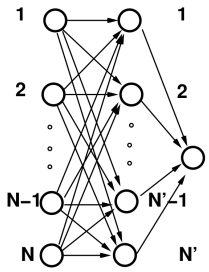

Fig. 7: Arquitectura N- N′ -1

Pérdida y exactitud en función de la época

Para esta sección utilizaremos la nomenclatura (N, N', N_{train}) para referirnos a la cantidad de entradas N, de neuronas en la capa oculta N' y la cantidad de ejemplos para el entrenamiento N_{train}. Una vez definido estos parámetros, se ejecuta el programa con valores iniciales aleatorios para los pesos y los umbrales. En la Fig.8 se muestra la pérdida en función de las épocas para distintas ejecuciones del programa con distintos valores para N, N' y N_{train}. Se observa que para la pérdida tiene a ser nula para la mayoría de los casos en las primeras 250 épocas, excepto para $(4, 1, 16)$, $(4, 3, 16)$ y $(7, 4, 100)$.

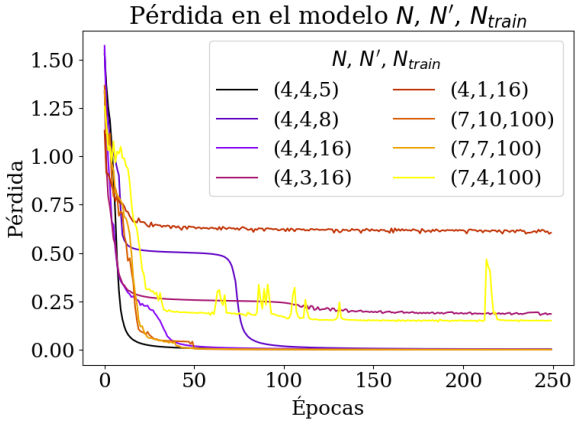

Fig. 8: Pérdida en función de la época para distintas arquitecturas

En los casos donde la pérdida no tiene a 0, si consideramos la exactitud de los mismos tal como se muestra en la Fig. 9, la misma no tiende a ser 1. Esto se debe a la presencia de mínimo locales en la convergencia.

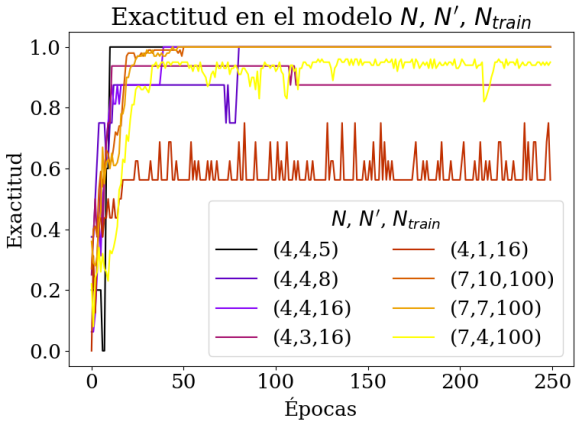

Fig. 9: Exactitud en función de la época para distintas arquitecturas

Para los casos $(4, 4, 16)$, $(4, 3, 16)$ y $(4, 1, 16)$, se entrena la red con todas las combinaciones posibles del XOR generalizado para 4 entradas. La diferencia entre las mismas es la cantidad de neuronas en la oculta. Se observa que cuando $N > N'$ la red tarda más en converger que cuando $N <= N'$, por ejemplo $(7, 4, 100)$. Esto se debe a la dimensión del problema no es suficiente para separar a los puntos de entrenamiento, como se espera con un perceptron.

EJERCICIO 3: MAPEO LOGÍSTICO

La función que la red neuronal busca solución es conocida, que la función logística, dada por la Ec. 1 del tipo

$$x(t+1) = 4x(t)(1 - x(t)). \tag{1}$$

Para generar los valores de entrenamiento, se define la cantidad de épocas T en las que los pesos van a actualizar. Con este valor, se calcula $x(T)$ usando como valor inicial $x[T-1]$ un valor aleatorio en el intervalo $(0, 1)$.

La arquitectura utilizada se muestra en la Fig. 10. En la ejecución del programa se agregaron neuronas adicionales para simular los umbrales y es utilizó un valor de $\eta = 0.05$.

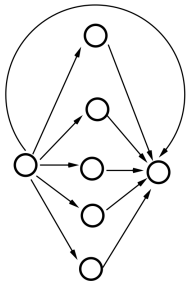

Fig. 10: Arquitectura para regresionar la función de mapeo logístico.

Error en función de la época

Durante la evolución de la red, se presentaron 15 ejemplo para verificar el error generalización en función de las épocas comparada con el error de entrenamiento, como se muestra en la Fig. 11. Se observa que para 5 ejemplos de entrenamiento, el error de entrenamiento, representada por una línea continua, converge a un valor no nulo en las primeras 250 épocas, mientras que el error de generalización, mostrada como una línea punteada, tiende a seguir disminuyendo. En cambio para 10 ejemplos de entrenamiento, ambos error disminuyen y la diferencia entre ambos disminuye en función de las épocas. Ya para 100 ejemplos, ambos errores convergen rápidamente a 0, esto se debe a la cantidad de ejemplos de entrenamiento que usa la red para actualizar sus pesos.

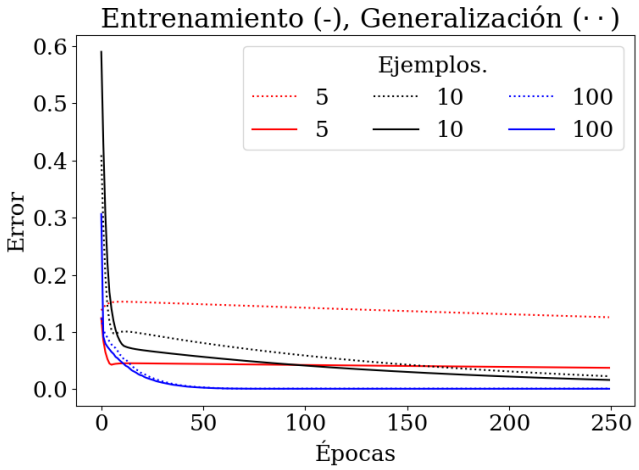

Fig. 11: Error en función de la época para distintas cantidades de ejemplos de entrenamiento.