Aprendizaje supervisado en redes multicapa

Avetta, Gastón
Redes neuronales

11 de octubre de 2023

Pérdida y precisión

Para todas las redes neuronales entrenadas y analizadas en este trabajo, la medida de desempeño está dada por la pérdida (*loss*) del modelo, dada por la expresión 0.1,

$$MSE = \sqrt{\frac{1}{2} \cdot \sum_{\mu=1}^{P} \sum_{i=1}^{N} (y_i^\mu - O_i^\mu)^2}, \qquad (0.1)$$

donde $\vec{y}$ es la salida deseada de la red y $\vec{O}$ la salida predicha por la red neuronal; la sumatoria es sobre todos los patrones de entrenamiento y todas las dimensiones de la salida correspondiente (aunque, en esta práctica, todas las redes utilizadas tienen salidas unidimensionales). *MSE* corresponde a las siglas de *mean squared error* (error cuadrático medio) y este parámetro también es denominado la *pérdida (loss)* de la red.

Por otro lado, para los ejercicios 1 y 2 se define también la precisión del modelo (*accuracy*). Esto surge por el hecho de que se desea modelar una red con entradas y salidas discretas (1 y -1) pero los algoritmos de entrenamiento utilizados dan como resultado salidas continuas. Por lo tanto, para definir la precisión, se redondea la salida al entero más cercano y se compara a este número con la salida esperada y se promedia para todos los patrones de entrenamiento utilizados, teniendo una precisión entre 0 (si no coincide ninguna salida) y 1 (si coinciden todas).

1. Regla XOR

La regla XOR tiene dos entradas (+1 o -1) y la salida es -1 si ambas son diferentes y +1 si ambas son iguales. Utilizar tanh(x) como función transferencia.

Para este problema simple, donde se tiene una entrada bidimensional, una salida unidimensional y una única capa oculta (para ambas redes propuestas), diseñamos a mano el mecanismo de **retropropagación de errores** (códigos adjuntos en el apéndice al final del informe).

1.1. Primer modelo - secuencial

En la figura 1.1 se puede observar el modelo de red neuronal utilizado para resolver este ejercicio, donde se tiene una capa oculta bidimensional entre la entrada y la salida y las conexiones entre las neuronas son **secuenciales**, es decir, no hay conexiones entre neuronas que no se hallan en capas contiguas.

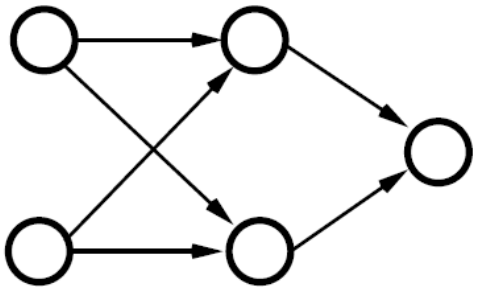


Figura 1.1: Red neuronal secuencial utilizada en este ejercicio.

Es necesario tener en cuenta que, para obtener un cierto margen de seguridad a la hora de entrenar a la red neuronal, consideramos una tercer dimensión *ficticia* en la entrada, una neurona que denominaremos *bias*, la cual tiene conexiones con todas las neuronas de las capas siguientes: las dos neuronas de la capa oculta y la neurona de la capa de salida. De es-

ta forma, este modelo cuenta con 7 parámetros para entrenar.

En la figura 1.2 se puede ver cómo evoluciona el desempeño de la red neuronal en las distintas épocas de entrenamiento para distintos valores del *learning rate*.

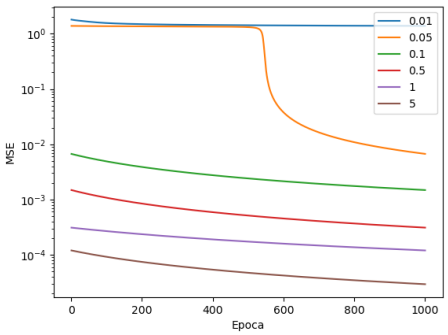


Figura 1.2: Pérdida de la red neuronal en función de las épocas de entrenamiento para distintos valores de learning rate, con eje vertical en escala logarítmica. Los valores iniciales de los parámetros fueron los mismos para cada una de estas realizaciones.

Si bien esta figura parece directa la relación entre un mayor *learning rate (lr)* y un mejor desempeño de la red, sabemos que para valores grandes de este parámetro la red puede tomar un comportamiento caótico y provocar que la red no converja al comportamiento esperado. De aquí en adelante, se muestran resultados obtenidos con un *lr* de 0,5.

Si analizamos el desempeño de la red para una única realización, puede ocurrir que esta no converja al comportamiento adecuado, sino que converja en un mínimo local con una pérdida considerable y no dé la salida deseada para los cuatro patrones posibles: esto depende de los valores con los que se inicialice los parámetros de la red. Para sortear esta cuestión en el análisis del desempeño de la red, analizamos la pérdida y la precisión (*accuracy*) de la misma en 100 realizaciones distintas y analizamos el promedio de las mismas, tal como se puede ver en la figura 1.3.

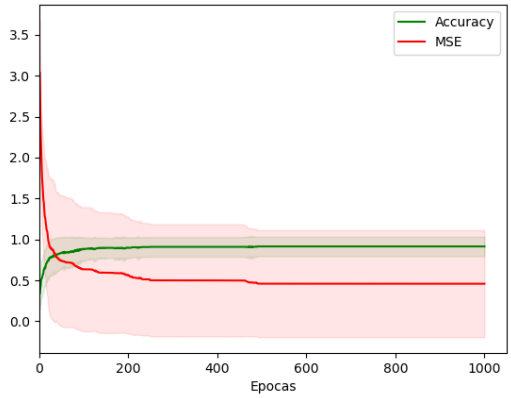


Figura 1.3: Pérdida y precisión promedio de la red neuronal a partir de 100 inicializaciones distintas de los parámetros de la misma. En cada realización hubo 1000 épocas de entrenamiento con lr = 0,5. Sombreado detrás de cada curva promedio, se tiene la desviación estándar de ambos parámetros.

1.2. Segundo modelo

En la figura 1.4 se puede observar el modelo de red neuronal utilizado en este caso, donde la capa oculta es unidimensional y la capa de entrada tiene conexiones no solo con la capa oculta sino también con la capa de salida.

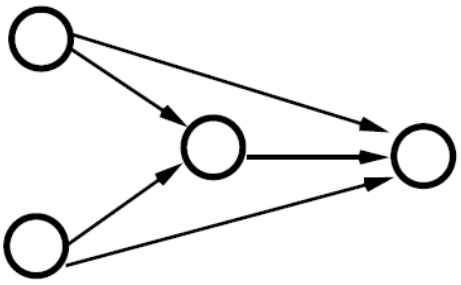


Figura 1.4: Red neuronal utilizada en este ejercicio.

Haciendo un análisis análogo al del otro modelo, obtenemos los resultados de la figura 1.5.

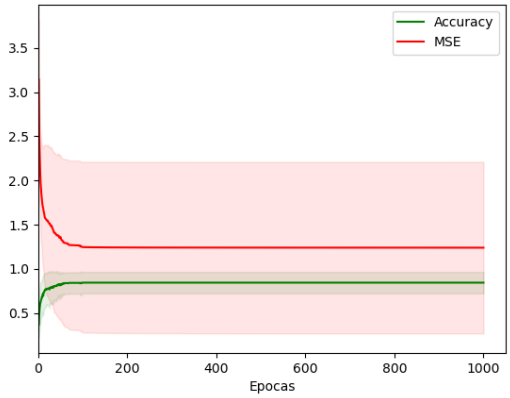
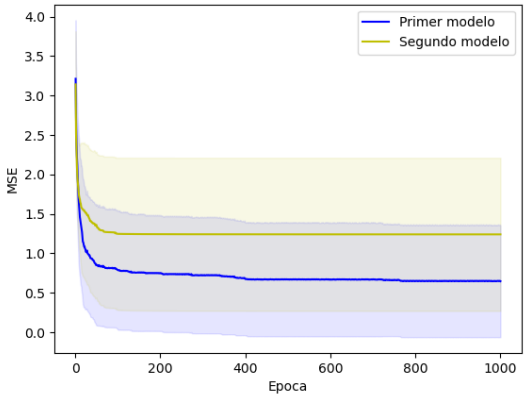
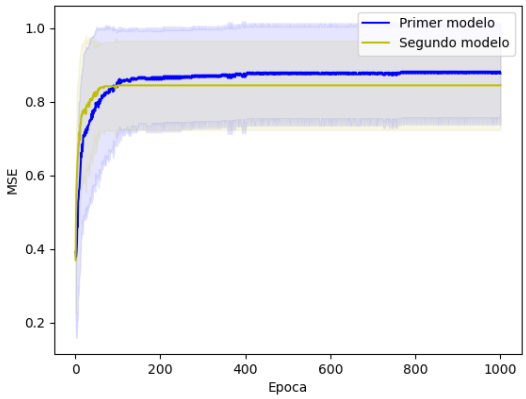


Figura 1.5: Pérdida y precisión promedio de la red neuronal a partir de 100 inicializaciones distintas de los parámetros de la misma. En cada realización hubo 1000 épocas de entrenamiento con lr = 0,5. Sombreado detrás de cada curva promedio, se tiene la desviación estándar de ambos parámetros.

Entonces, teniendo parámetros que caracterizan el desempeño promedio de ambas redes neuronales para resolver la **regla XOR bidimensional**, comparamos ambos modelos en la figura 1.6.



(a) Pérdida.



(b) Precisión.

Figura 1.6: Desempeño promedio tras 100 realizaciones con 1000 épocas de entrenamiento y un learning rate de 0,5.

Observando ambos parámetros podemos concluir que la primera red neuronal propuesta, la secuencial, tiene un mejor desempeño que la segunda, logrando un menor error cuadrático medio y una mayor precisión en las salidas predichas.

2. Generalización del XOR

El problema de paridad es una generalización del XOR para N entradas. La salida es +1 si el producto de las N entradas es +1 y -1 si el producto de las entradas es -1.

Tanto para este ejercicio como para el siguiente, no utilizamos un mecanismo de retropropagación de errores diseñado a mano (como en 1), sino que utilizamos la API **Keras** de la librería **Tensorflow** de Python.

En la figura 2.1 se puede ver la red neuronal propuesta para este problema, con una capa de entrada de dimensión N, una capa oculta de dimensión N' y una capa de salida unidimensional.

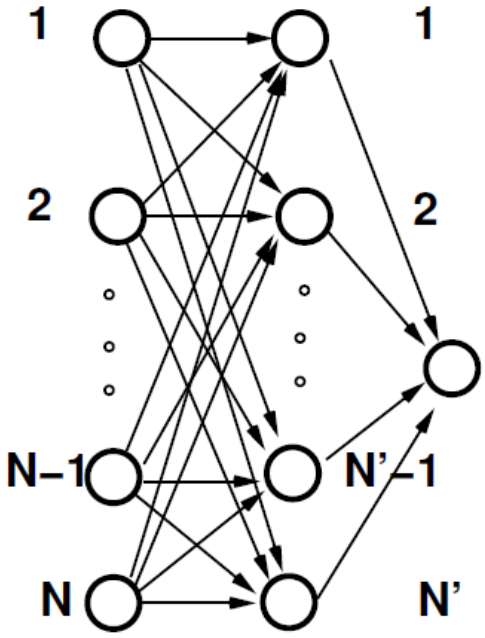
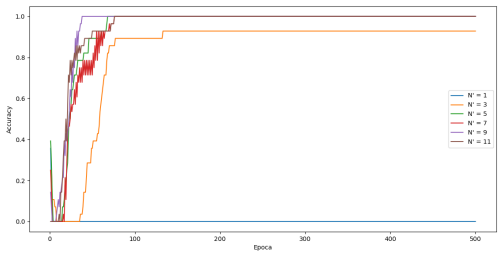
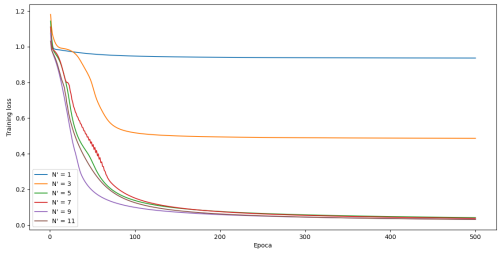


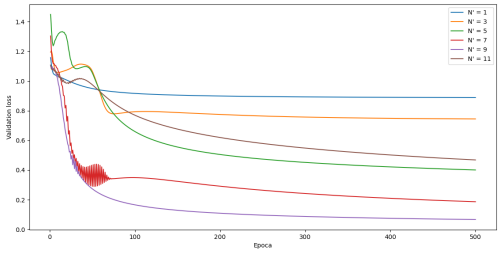
Figura 2.1: Red neuronal utilizada en este ejercicio.



(a) Pérdida.



(b) Pérdida de entrenamiento.



(c) Pérdida de validación.

Figura 2.2: Medidas de desempeño obtenidas para distintas dimensiones de la capa oculta

En esta práctica, se utilizo un único valor para la dimensión de las entradas a la red, N = 5, lo cual implica que se tienen $2^5 = 32$ patrones posibles para entrenar y validar a la red. Para la dimensión de la capa oculta se utilizaron valores entre 1 y 11, lo cual implica analizar casos donde la capa oculta reduce la dimensionalidad de la entrada y otros casos donde la aumenta. Para este ejercicio, se realizaron entrenamientos de 1000 épocas, con un *learning rate* de 0,5 y, de los 32 patrones posibles, se utilizaron 28 para entrenar al modelo y 4 para validarlo, elegidos aleatoriamente. Para todos los valores probados de N', se utilizó la misma inicialización de los parámetros de la red y los mismos patrones para el entrenamiento y la validación. En la figura 2.2 se pueden ver los resultados obtenidos.

Cabe aclarar que estos son los resultados obtenidos para una única realización, una única inicialización de los parámetros de la red, por lo cual hay que ser cuidadoso a la hora de generalizar conclusiones aquí observadas.

De todas formas, en la figura 2.2a podemos observar que en general, tras 100 épocas de entrenamiento, las redes con N' mayor o igual a la dimensión de la capa de entrada convergen a la precisión máxima, es decir, redondeando al entero más cercano, predicen todas las salidas correctamente. Por lo contrario, las redes donde la dimensión de la capa oculta es menor a la de la entrada convergen a una precisión menor, siendo particularmente nula cuando N' = 1, es decir, no acierta la salida de ninguno de los patrones presentados.

Por otro lado, en la figura 2.2b vemos que, en líneas

generales, a mayor dimensión, menor pérdida en el entrenamiento, lo cual se traduce en un mejor desempeño de la red. Al igual que para la precisión, vemos que las redes con capa oculta de dimensión igual o mayor a la entrada convergen a una pérdida mucho menor que aquellas con N' < N; también, la red con N'= 1 tiene un desempeño pésimo, donde la pérdida se mantiene prácticamente constante durante todo el entrenamiento.

Hasta aquí, podemos concluir que, si la capa oculta tiene una dimensión menor a la capa de entrada, no se puede esperar que la red tenga un buen desempeño. Por lo tanto, es determinante que dicha dimensión sea mayor o igual a la dimensión de los patrones de entrada. Una vez que se tiene N' $\geq$ N, parece que el desempeño mejora al aumentar dicha dimensión, pero esta mejora es mucho menos determinante. De hecho, para todas las dimensiones probadas mayores a 5, la precisión de la red convergió al valor máximo en aproximadamente la misma cantidad de etapas de entrenamiento.

Por último, en clase fue mencionado que, para el problema de paridad, "la validación da siempre mal", debido a que es un problema muy difícil de generalizar, ya que cambiando solo una dimensión de la entrada se obtiene la salida opuesta. En la figura 2.2c podemos ver que en líneas generales, la conclusión de que a mayor N' se tiene mejor desempeño sigue siendo válida, pero con algunas excepciones. Ya de por sí, el comportamiento de las distintas curvas resulta más irregular que para la pérdida de entrenamiento, creciendo o teniendo *oscilaciones* en algunas épocas del entrenamiento. Otro ejemplo de estas irregularidades: vemos que para N' = 11, donde el error de entrenamiento era mínimo, el error de validación resulta mayor que para las redes con dimensiones 5, 7 y 9 de la capa oculta.

3. Mapeo logístico

Aprender utilizando retropropagación el mapeo logístico $x(t+1) = 4x(t)\cdot(1\ x(t))$. La función de activación de las neuronas en la capa oculta es $g(x) = 1/(1+e^{-x})$. La función de activación de la neurona de salida es lineal.

Para este ejercicio, utilizamos la red neuronal que se puede ver en la figura 3.1, con las funciones de activación indicadas en el enunciado para las distintas capas.

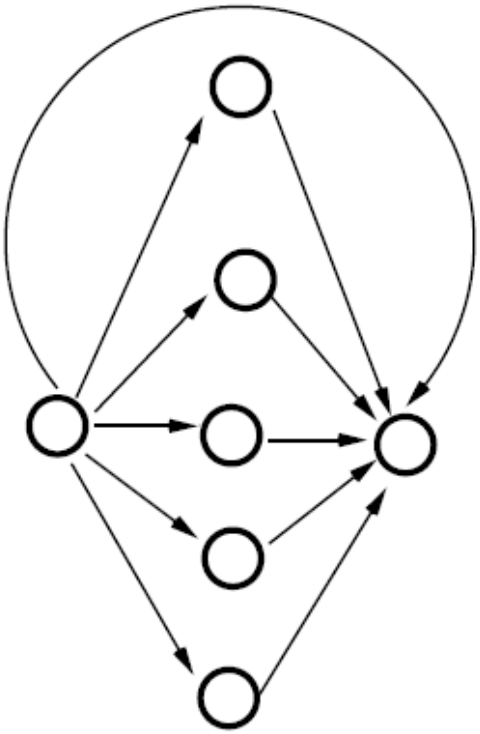


Figura 3.1: Red neuronal utilizada en este ejercicio.

El objetivo es entrenar esta neurona con una cierta cantidad de patrones para obtener salidas que se correspondan al mapeo logístico indicado en el enunciado. En la figura 3.2 se puede observar el mapeo en cuestión utilizando un valor inicial de 0,2.

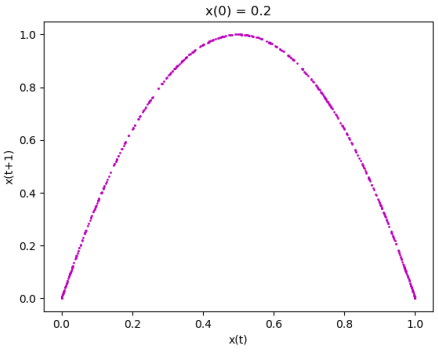
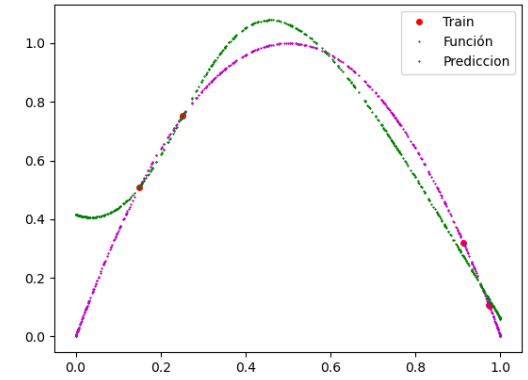


Figura 3.2: Mapeo logístico indicado, el cual se desea predecir con la red neuronal en cuestión.

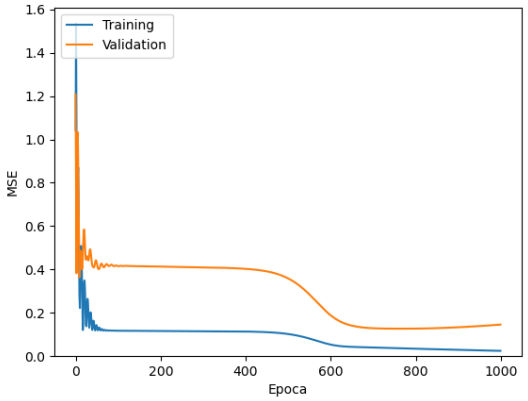
Utilizamos un total de 500 puntos del mapeo logístico como base de datos, los cuales serán utilizados para medir las pérdidas de validación de la red neuronal. Analizaremos el desempeño de la red dándole 5, 10 y

100 patrones para el entrenamiento, con un *learning rate* de 0,5 y 1000 épocas de entrenamiento en cada caso. Los patrones utilizados para el entrenamiento fueron elegidos aleatoriamente entre los 500 patrones de la base de datos. La inicialización de los parámetros de la red fue siempre con los mismos valores.

En la figura 3.3 se pueden observar los resultados obtenidos con solo cinco patrones de entrenamiento.



(a) Función deseada, patrones de entrenamiento y predicción de la red neuronal.

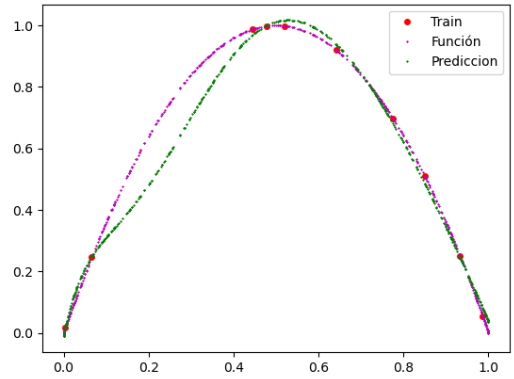


(b) Pérdidas de entrenamiento y de validación.

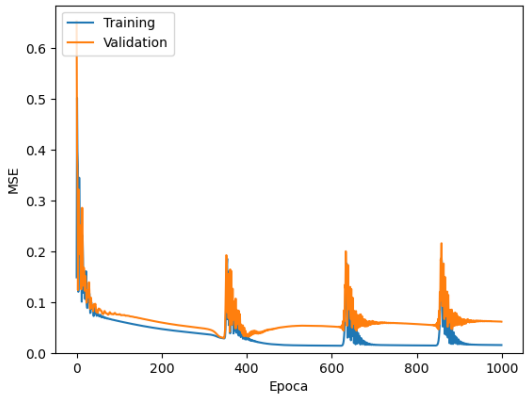
Figura 3.3: Resultados obtenidos con cinco patrones de entrenamiento.

De la figura 3.3a se puede concluir que 5 patrones de entrenamiento no son suficientes para entrenar a la red y obtener el comportamiento deseado, sin importar cómo se distribuyan estos entre toda la base de datos. Analizando la figura 3.3b se refuerza este teoría, donde vemos que el error de validación es considerablemente mayor al de entrenamiento, dado que la red predice salidas bastante acertadas para los patrones utilizados para el entrenamiento, pero difiere bastante con la predicción del resto de puntos de la base de datos.

En la figura 3.4 tenemos el mismo análisis pero para un conjunto de diez patrones de entrenamiento.



(a) Función deseada, patrones de entrenamiento y predicción de la red neuronal.



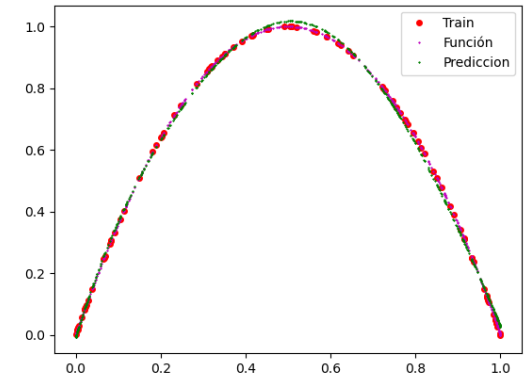
(b) Pérdidas de entrenamiento y de validación.

Figura 3.4: Resultados obtenidos con diez patrones de entrenamiento.

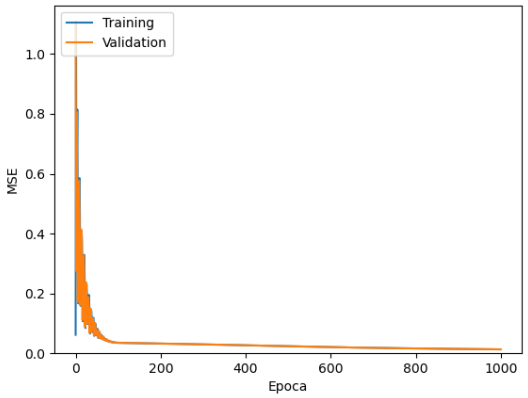
En este caso, la figura 3.4a muestra una predicción bastante más acertada que el caso anterior, aunque con regiones donde el error es notable. Esto dependerá de la distribución de los patrones de muestreo entre los valores de la base de datos. En la figura 3.4b vemos que, más allá del comportamiento errático de las curvas en algunas regiones, el error de validación sigue siendo considerablemente mayor al de entrenamiento, aunque ya menos que en el caso anterior, debido a que tenemos una predicción que representa fielmente a la función deseada para varios puntos de la base de

datos.

Por último, en la figura 3.5, el mismo desarrollo pero con cien patrones de entrenamiento.



(a) Función deseada, patrones de entrenamiento y predicción de la red neuronal.

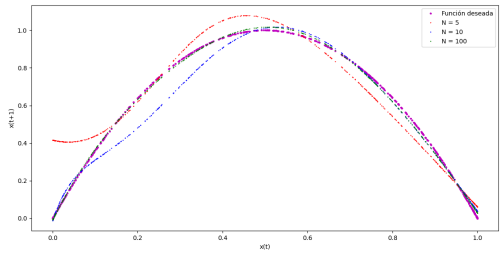


(b) Pérdidas de entrenamiento y de validación.

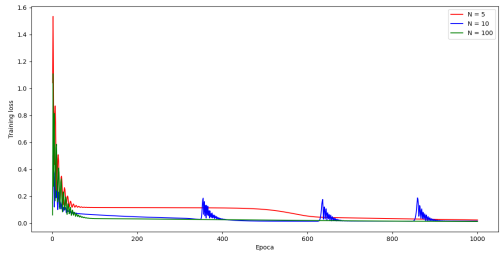
Figura 3.5: Resultados obtenidos con cien patrones de entrenamiento.

Ahora, en la figura 3.5a vemos que con cien patrones de entrenamiento ya obtenemos una predicción bastante acorde a la función deseada en todas las regiones de la base de datos. Esto también se puede concluir observando la figura 3.5b, donde vemos que, a diferencia de los casos anteriores, el error de entrenamiento y de validación convergen a un mismo valor.

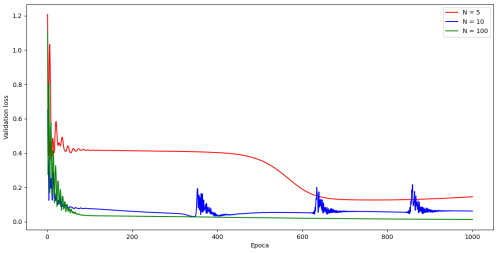
Por último, en la figura 3.6 se puede ver la comparación de los resultados obtenidos para los tres conjuntos de patrones de entrenamiento analizados.



(a) Función deseada y predicciones obtenidas.



(b) Pérdidas de entrenamiento obtenidas



(c) Pérdidas de validación obtenidas

Figura 3.6: Resultados obtenidos para los distintos conjuntos de patrones de entrenamiento utilizados.

De las figuras 3.6b y 3.6c vemos que la diferencia entre los desempeños obtenidos para los distintos conjuntos de patrones de entrenamiento es más notable en la pérdida de validación que en la de entrenamiento. Esto ocurre porque, en el caso de pocos patrones de entrenamiento, después de mil épocas, la red queda bien entrenada y predice correctamente las salidas indicadas, pero no tiene capacidad de generalización, por lo cual no predice correctamente las salidas esperadas para el resto de los puntos de la base de datos.

Códigos utilizados

A continuación, se adjunta una carpeta con los códigos utilizados para resolver los distintos ejercicios: https://drive.google.com/drive/folders/16Zd4TPu546HU35KnVilHBIPBGPjlkvVJ?usp=sharing.