TP2: REDES NEURONALES

Grupo 4: Martina Arco, Joaquín Ormachea,
Lucas Emery, Diego Orlando

Introducción

En este trabajo se desarrolló una red neuronal para poder aproximar los puntos en un espacio que fue provisto por la cátedra. El programa recibe por parámetros diferentes configuraciones que se le pueden aplicar a la red.

Descripción del programa

Las configuraciones, que en el *README.md* se describe cómo cambiarlas, son:

- La estructura de la red, es decir, cuántas capas se van a tener y cuántas neuronas va a haber en cada una. Tener en cuenta que siempre el primer valor debe ser 2 y el último 1.
- Qué función de activación se va a utilizar, que puede ser tangente hiperbólica o exponencial.
- Porcentaje de patrones a utilizar para el aprendizaje.
- Tamaño del batch a utilizar, si se pone 1 se corre en modo iterativo.
- Factor de aprendizaje.
- Cantidad máxima de épocas.
- Máximo error, luego que baja de este valor, no se sigue iterando. En caso de que se combine con la cantidad de épocas, cuando alguna de las dos condiciones se cumpla, se deja de iterar.
- Optimizadores y sus parámetros.

La salida del programa será la matriz de pesos que se terminó aproximando, el error final y la cantidad de épocas. Además, se grafica el error de aprendizaje y de testeo y el factor de aprendizaje a medida que se está corriendo. En caso de querer modificar variables mientras se corre el programa, se puede poner un breakpoint en la línea 170 y desde allí modificar las variables deseadas desde la consola.

Para inicializar la matriz de pesos se utilizó una distribución normal y se la dividió por la raíz cuadrada de la cantidad de entradas. Este es mejor que la distribución uniforme debido a la tangente hiperbólica y sigmoide. Más cerca del 0 tienen más pendiente y de esta forma la derivada no se anula.

Además, se normalizaron las entradas restando la media y dividiendo por la varianza como fue pedido.

Análisis de resultados

Para el análisis de resultados, como default se utilizó:

- Máximo de épocas: 1500.
- Máximo error: 0.0000001.
- Tamaño de bache: 4.
- Porcentaje de aprendizaje: 0.9.
- Factor de aprendizaje: 0.01.
- Estructura: una capa con 50 neuronas ([2, 50, 1]).
- Función de activación: tangente hiperbólica.
- Optimizador: RMSPprop.
- Gamma: 0.9.

Estos se mantuvieron constantes y luego se varió el parámetro correspondiente a cada caso.

Aproximación del plano

En el gráfico 1 se puede observar el terreno provisto por la cátedra. Por el otro lado, el gráfico 2 muestra una aproximación del terreno en donde se utilizó un porcentaje del 80% para el entrenamiento y 20% para el testeo. Se puede observar que aunque no se ven todos los detalles, debido a las pequeñas variaciones del plano, se pudieron obtener las alturas y la idea general del terreno original.

Variaciones en factor de aprendizaje

Se puede observar el gráfico 3, 4 y 8 con las diferencias. Los resultados fueron:

- 0.01 con un error de 0.027198.
- 0.03 con un error final de 0.068659.
- 0.001 con un error de 0.047504.

Llegamos a la conclusión que, en el caso de tener el optimizador RMSProp, el mejor factor es 0.01. Además, este va aumentando a medida que avanza el algoritmo.

El cambio del factor varía la velocidad a la que logra aprender el algoritmo y es por esto que cuanto más pequeño es el factor menos logra progresar. Si es muy grande, se utilizan pasos muy largos y no se llega a aproximar.

Variaciones en tamaño de batch

Se probaron diferentes tamaños de baches para probar cómo este parámetro afectaba a la red. Se pueden observar los resultados en el gráfico 5:

- Bache con tamaño 1: 1500 épocas y error de 0.077797.
- Bache con tamaño 4: 1500 épocas y error de 0.11349.
- Bache con tamaño 8: 1500 épocas y error de 0.024156.
- Bache con tamaño 16: 1500 épocas y error de 0.022567.

El error aumenta al disminuir el tamaño de bache. Esto puede ser debido a que el error sólo se calcula cada cierta cantidad de patrones por lo que el impacto de uno en particular no afecta significativamente el resultado.

Variaciones en función de activación

En el gráfico 6 se muestran las dos funciones de activación que se utilizaron:

- La tangente hiperbólica: error de 0.083924.
- La función sigmoide: error de 0.21912.

Se realizaron ambos testeos sin ninguna optimización y con el resto de valores en default. Se puede observar como la tangente hiperbólica aprende más rápido que la función sigmoide.

Variaciones en estructura de red

En el gráfico 7 se aprecia cómo la estructura de red afecta el resultado al aprender. Las diferentes estructuras fueron:

- [2, 50, 1]: consiste en una capa oculta de 50 neuronas. El resultado consiste en 1500 épocas y un error de 0.031753.

- [2, 25, 25, 1]: consiste en dos capas ocultas de 25 neuronas cada una. Se realizaron 1500 épocas y el error fue de 0.022830.
- [2, 20, 10, 1]: consiste de dos capas ocultas de 20 y 10 neuronas cada una. Se consiguió un error de 0.022664 en 1500 épocas.

Comparando estos 3 resultados podemos ver que la precisión de la red no es proporcional a la cantidad total de neuronas ya que con la 20-10 obtuvimos mejores resultados que con la 25-25. Pero con ambas redes de 2 capas obtuvimos mejores resultados que con una única capa. Sin embargo se pueden observar algunos incrementos en el error que se deben a la falta de optimización del learning rate para la estructura en uso.

Variaciones con optimizadores

Se implementaron 4 optimizadores diferentes:

- Momentum
- Adagrad (ETA Adaptativo)
- RMSProp
- Adam

Los primeros dos eran los requeridos por la cátedra y los otros dos son mejoras de los anteriores, RMSProp es una mejora de Adagrad que no sufre de decaimiento continuo del learning rate mediante la utilización de un EMA y Adam es una combinación de RMSProp con Momentum.

Los resultados fueron los siguientes:

- Sin optimizador: 1,3e-2
- Momentum: 1,9e-2
- Adagrad: 1,3e-1
- RMSProp: 8,9e-3
- Adam: 1e-2

En el gráfico 9 se puede observar que Adam y RMSProp obtuvieron menor error a grandes rasgos y Momentum estuvo cerca.

La diferencia entre los distintos optimizadores se puede apreciar mejor en el gráfico 10, que muestra el error medio contra las muestras de testeo. En él se puede observar que Momentum fue el que dio una mejor generalización, seguido por Adam y RMSProp.

En el gráfico 11 se puede apreciar cómo varía el factor de aprendizaje con los distintos optimizadores. En él se puede observar el problema de Adagrad, que es el decaimiento constante y hace que la red no pueda seguir aprendiendo después de muchas iteraciones.

Apéndice

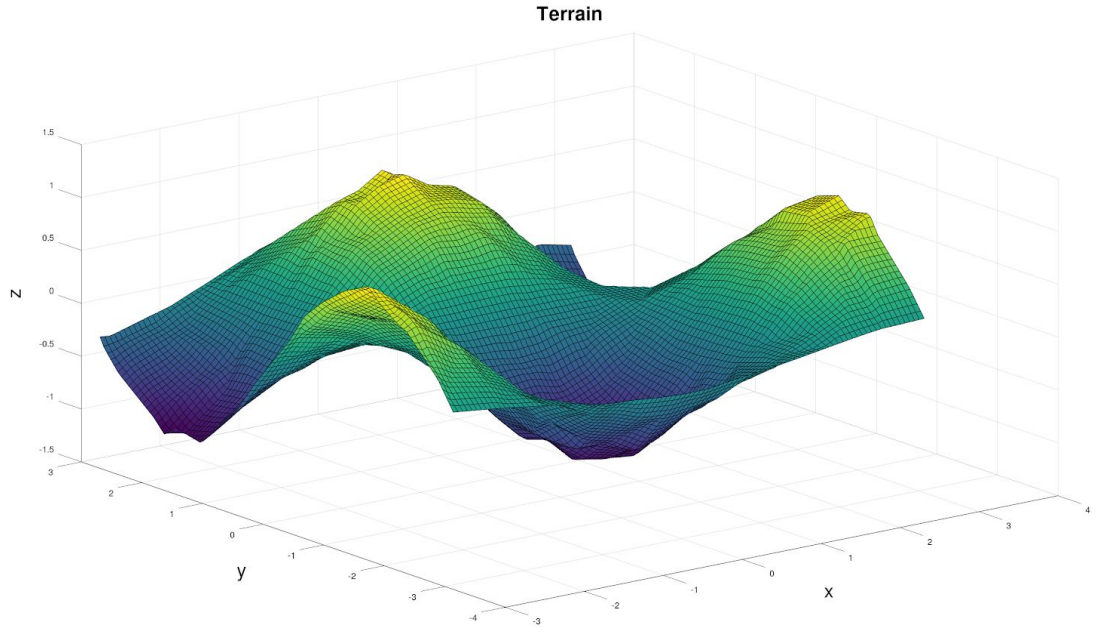

Gráfico 1: Terreno a aproximar.

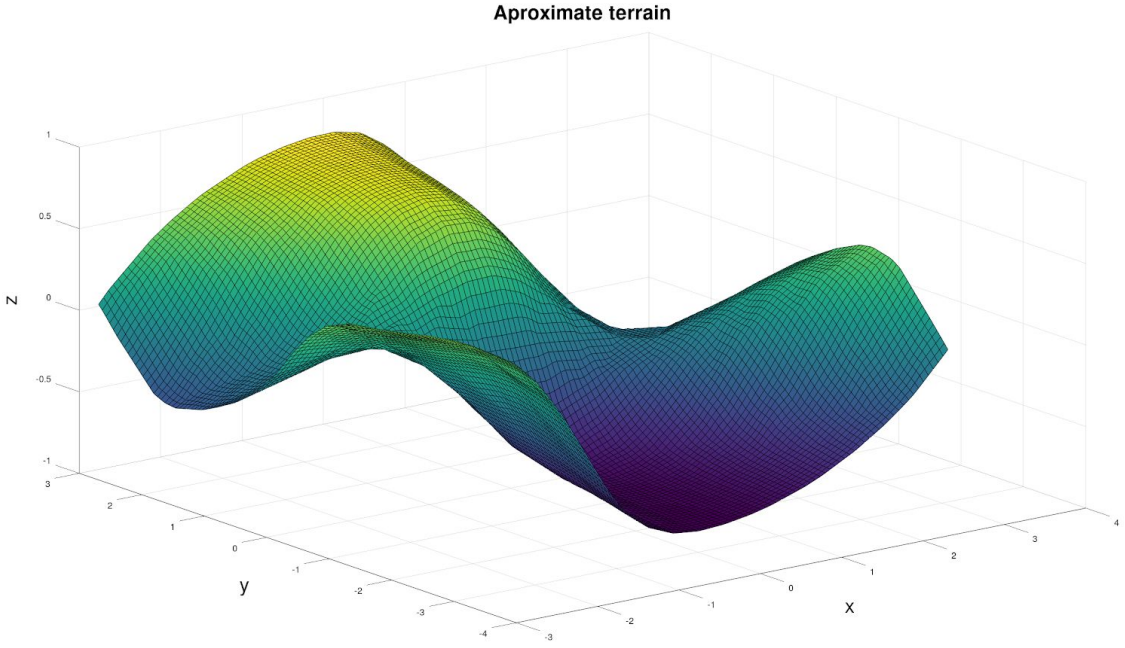

Gráfico 2: Terreno aproximado.

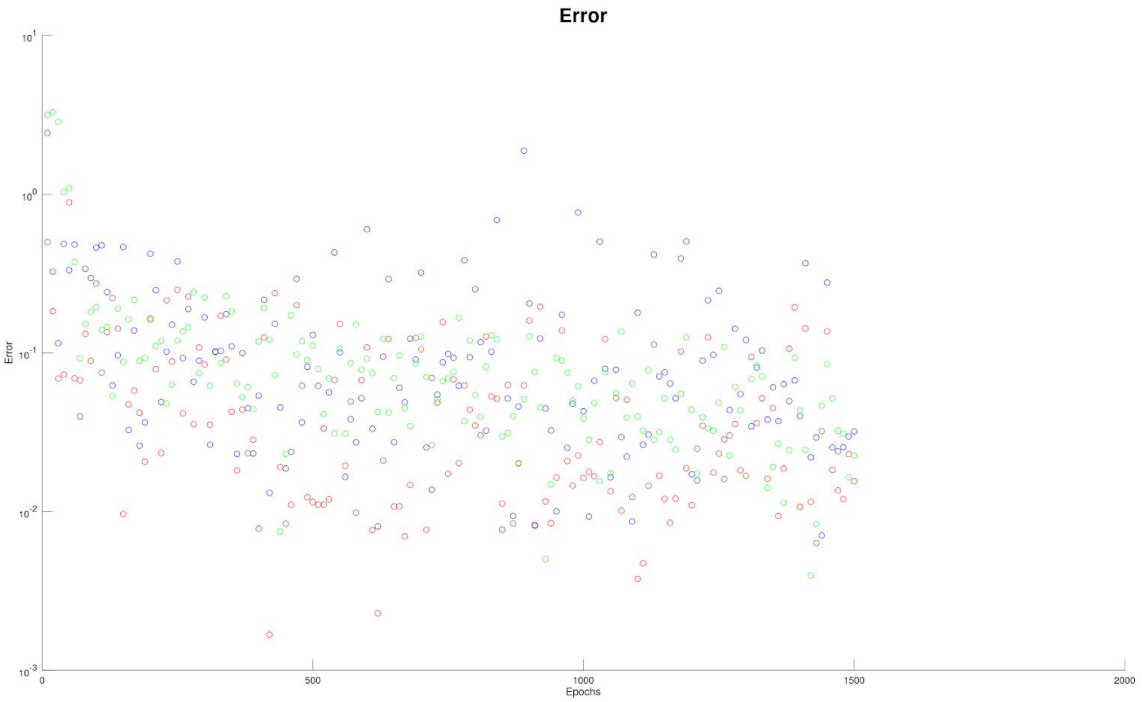

Gráfico 3: variaciones de error cambiando el factor de aprendizaje.

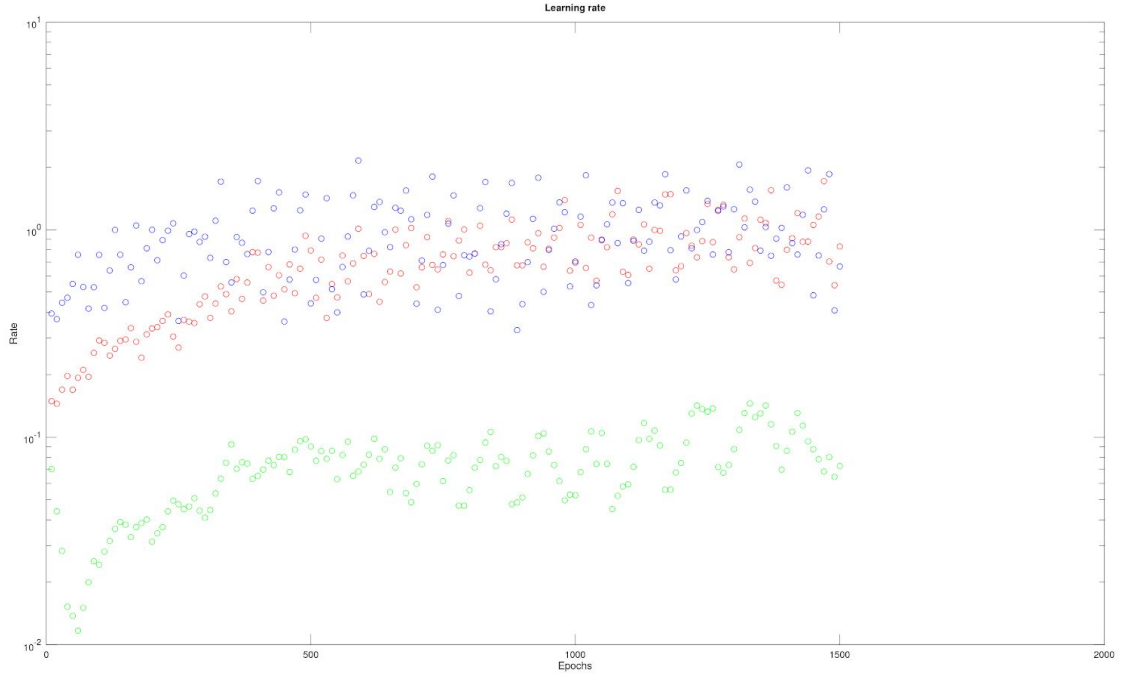

Gráfico 4: variaciones de factor de aprendizaje.

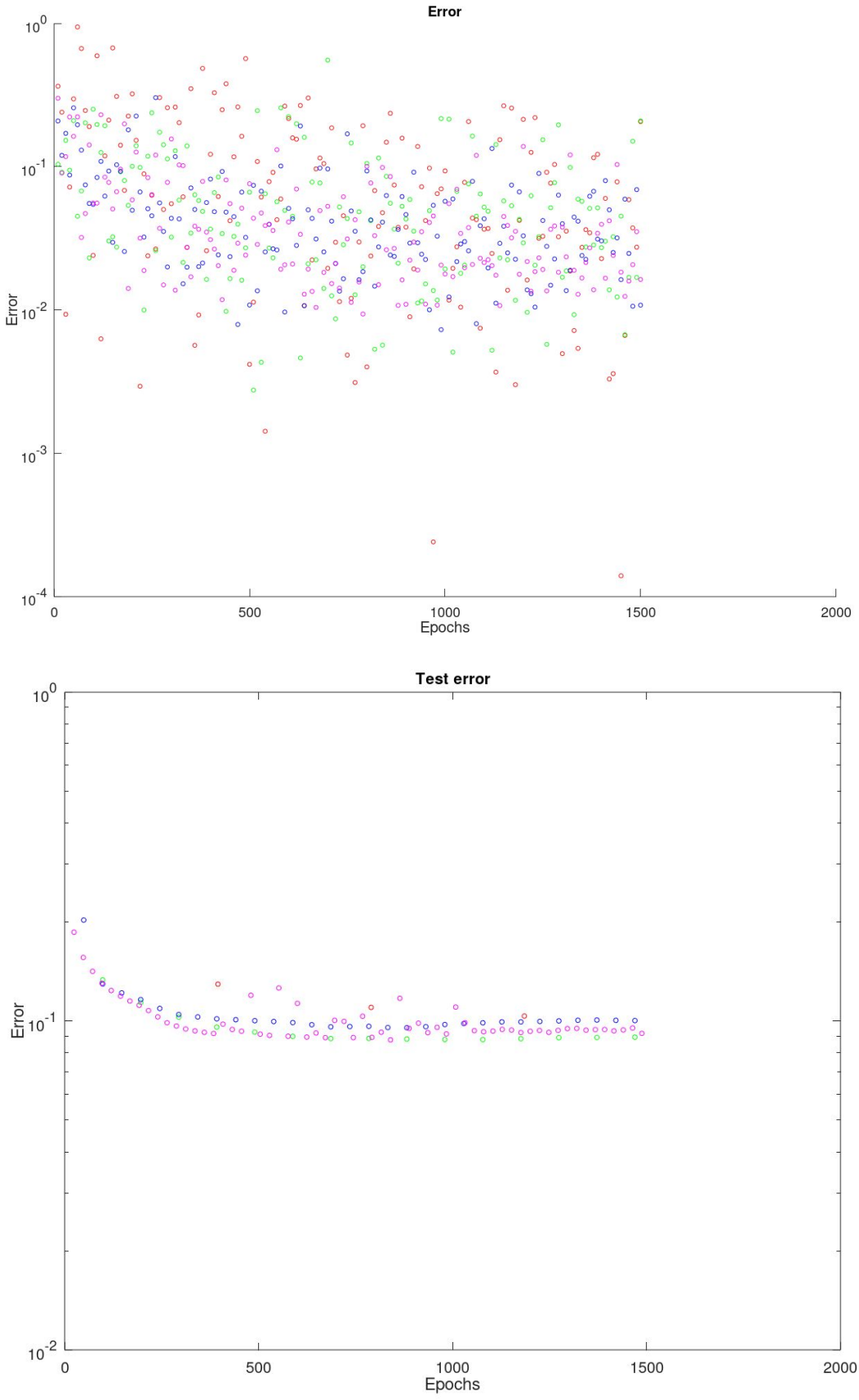

Gráfico 5: variaciones de tamaño de bache.

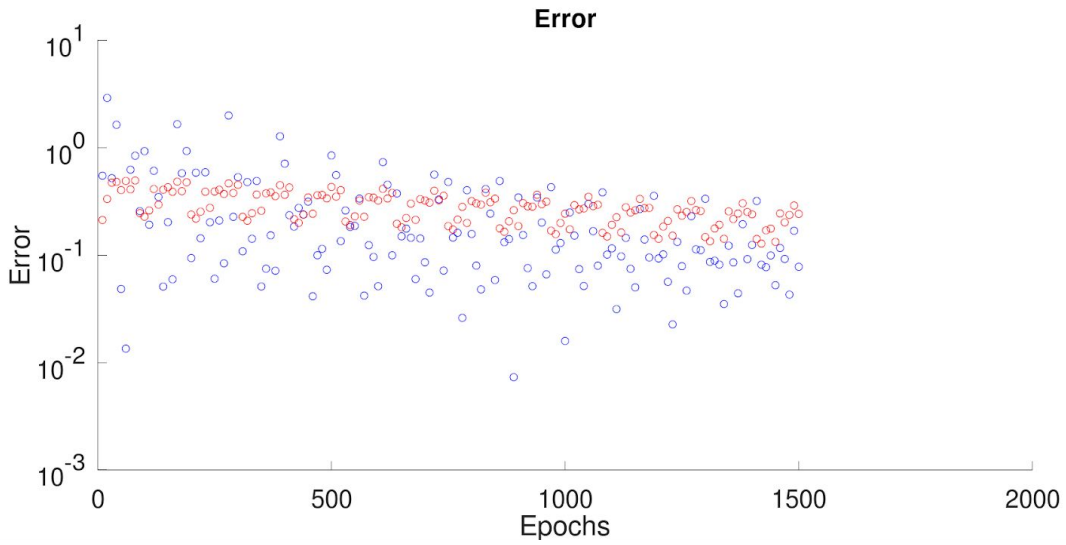

Gráfico 6: variaciones en la función de activación.

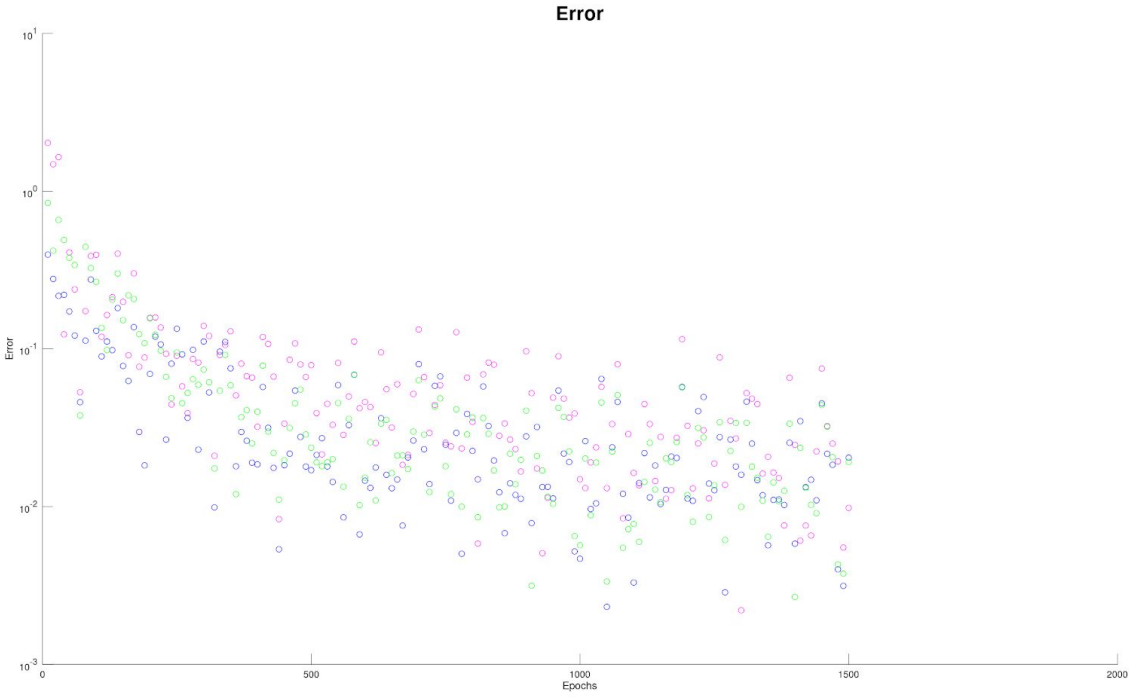

Gráfico 7: variaciones en la estructura de la red.

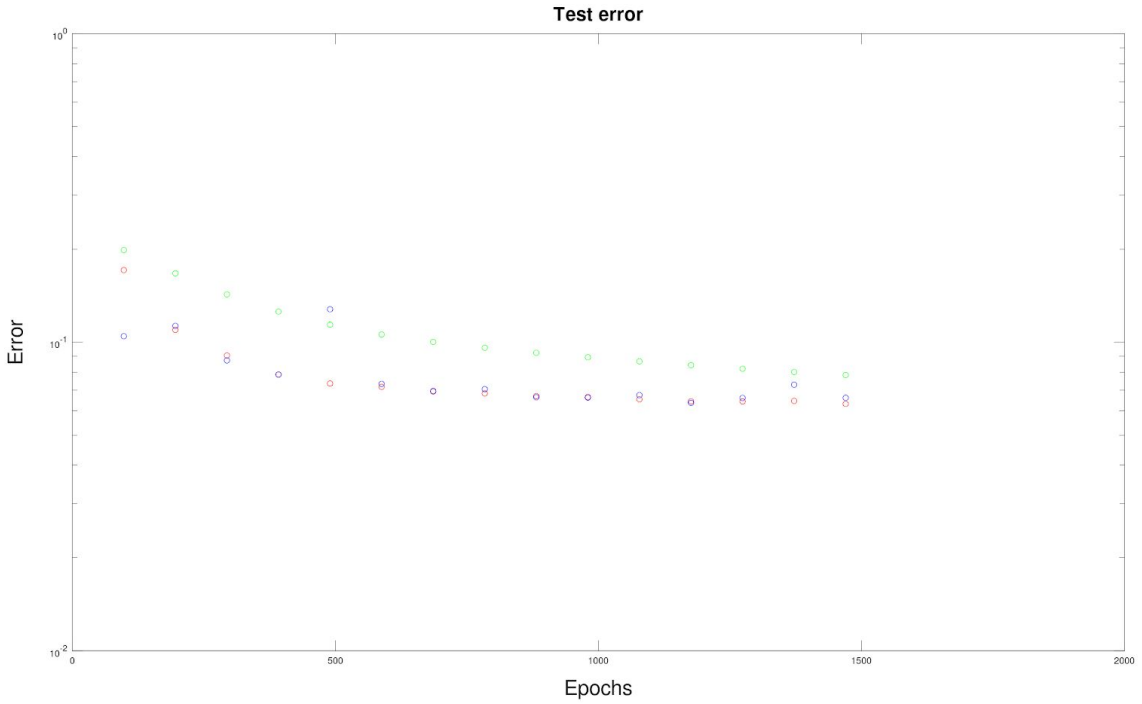

Gráfico 8: variación de error de testeo cambiando factor de aprendizaje.

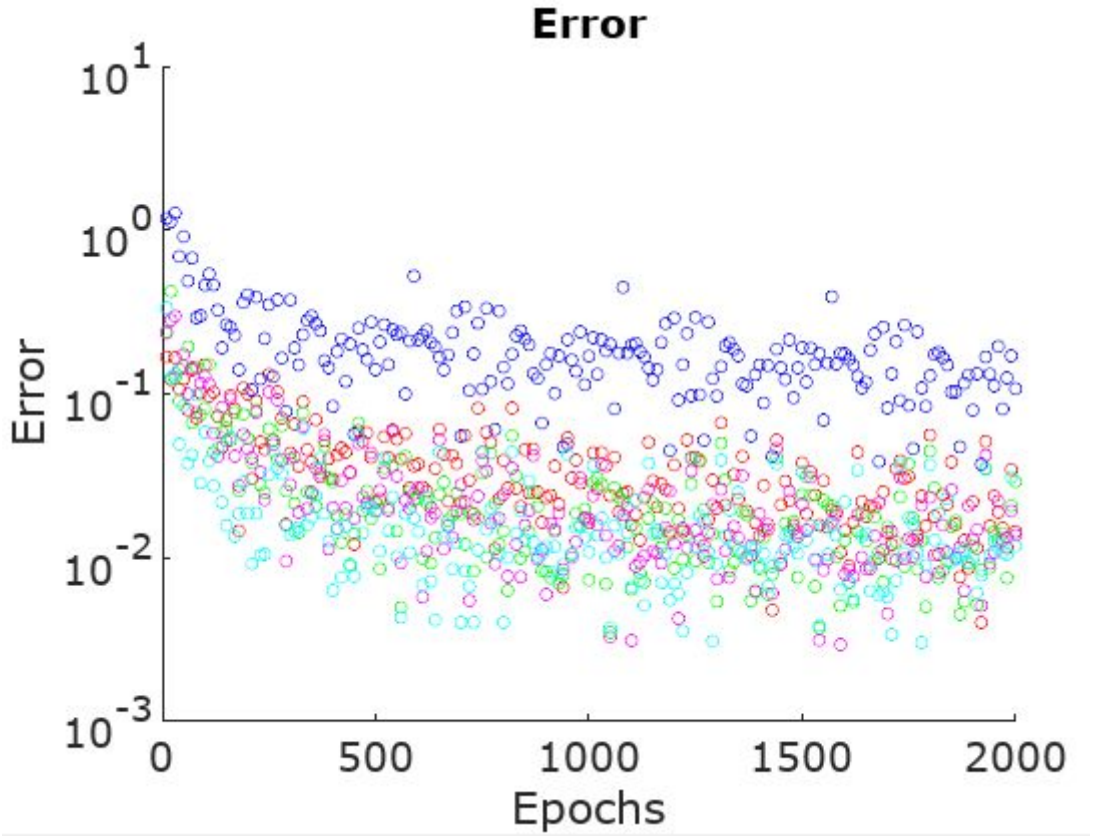

Gráfico 9: Variación del error en función del optimizador.

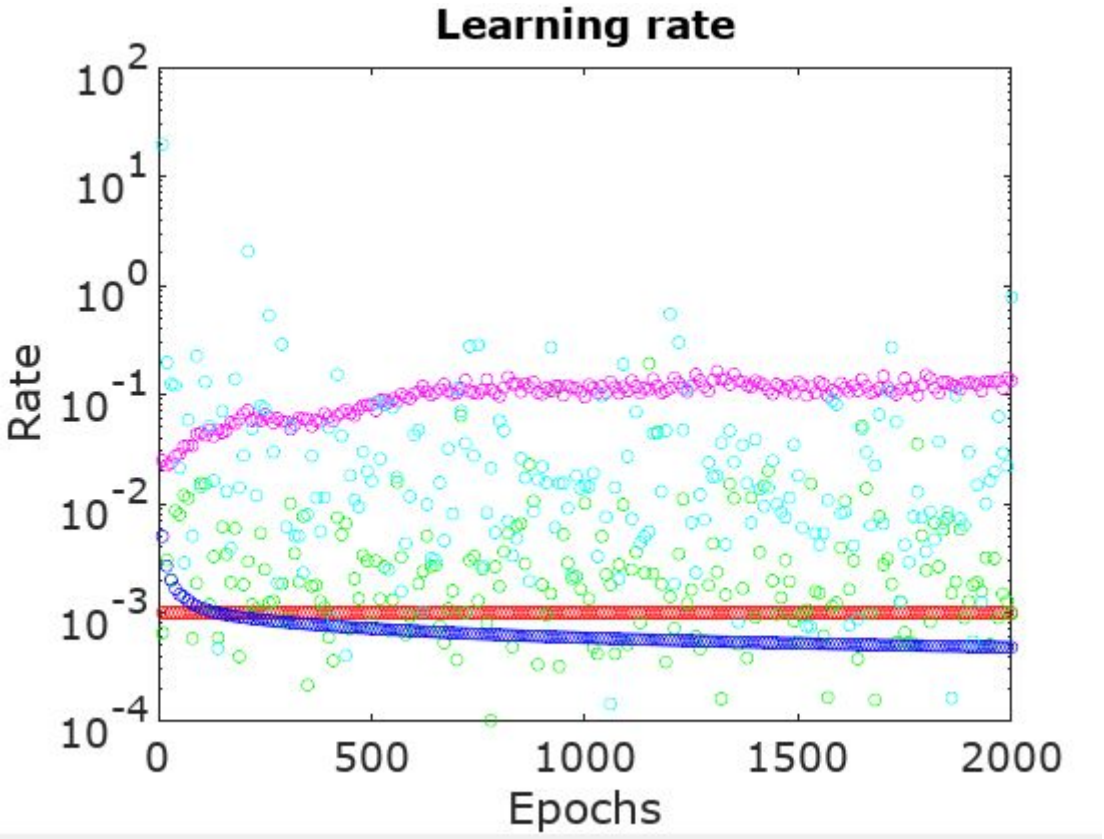

Gráfico 10: Variación del factor de aprendizaje en función del optimizador.

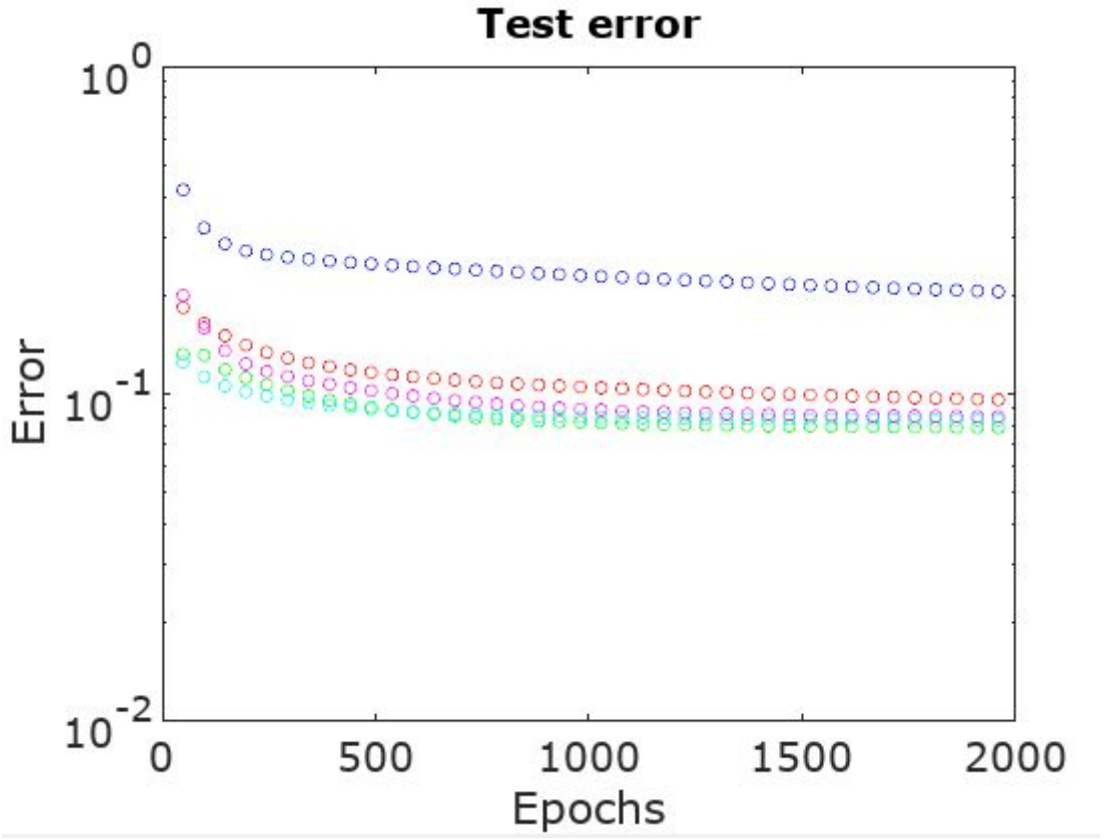

Gráfico 9: Variación del error de testeo en función del optimizador.