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.
- Tipo de corrida, es decir, batch o incremental.
- Factor de aprendizaje.
- Cantidad máxima de épocas.
- Máximo error, luego que el promedio 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, el porcentaje de aciertos que hubo 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: 0.02.
- Tipo: incremental.
- Porcentaje de aprendizaje: 0.9.
- Factor de aprendizaje: 0.03.
- Estructura: una capa con 50 neuronas ([2, 50, 1]).
- Función de activación: tangente hiperbólica.
- Optimizador: adagrad
- Gamma: 0.9.

- Epsilon: 0.01

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.

Batch o incremental

Se puede ver en los gráficos las diferencias. Los resultados fueron:

- Incremental con 27 épocas.
- Batch con 149 épocas.

Se puede observar que con incremental se logran mejores resultados porque se tiene en cuenta cada patrón individualmente, evitando así que patrones opuestos se cancelen.

Variaciones en factor de aprendizaje

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

- 0.01 con 341 épocas.
- 0.03 con 27 épocas.
- 0.001 con 500 épocas no logró terminar.

Llegamos a la conclusión que, en el caso de tener el optimizador adagrad, el mejor factor es 0.03. Además, este va disminuyendo 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 función de activación

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

- La tangente hiperbólica: 43 épocas.
- La función sigmoide: 200 épocas. Fue cortado porque no seguía aprendiendo.

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 porque está centrada en 0 yendo de -1 a 1, por lo que naturalmente se adapta más a nuestro problema.

Variaciones en estructura de red

Se puede observar 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 fue de 44 épocas.

- [2, 25, 25, 1]: consiste en dos capas ocultas de 25 neuronas cada una. Se realizaron 28 épocas.
- [2, 20, 10, 1]: consiste de dos capas ocultas de 20 y 10 neuronas cada una. Se terminar en 41 épocas.

Se puede ver que se obtuvieron mejores resultados con las estructuras de 2 capas. Y dentro de estas, observamos que la capa con más neuronas probablemente esté recordando más y así obteniendo resultados más precisos

Además, en el gráfico de aciertos se puede ver que el que el de 1 capa con 50 neuronas es el que más aciertos tiene porque está memorizando.

Variaciones con optimizadores

Los resultados fueron los siguientes:

- Sin optimizador: Se cortó la ejecución porque no estaba aprendiendo.
- Momentum: Se cortó la ejecución porque no estaba aprendiendo.
- Adagrad: 66 épocas
- Eta Adaptativo: 450 épocas.

Se puede observar que el único optimizador que dio buenos resultados para este caso fue adagrad. Esto se debe a que el learning rate utilizado es el mejor para este caso en particular.
Haciendo pruebas se pudo observar que para momentum, si se baja el rate a 0.001 logra terminar luego de 76 épocas. Sigue sin ser tan eficiente como eta adaptativo pero se acerca. Se puede observar en el último gráfico.
En cuanto al eta adaptativo también se recomienda utilizar un learning rate más bajo.

Apéndice

Planos encontrados

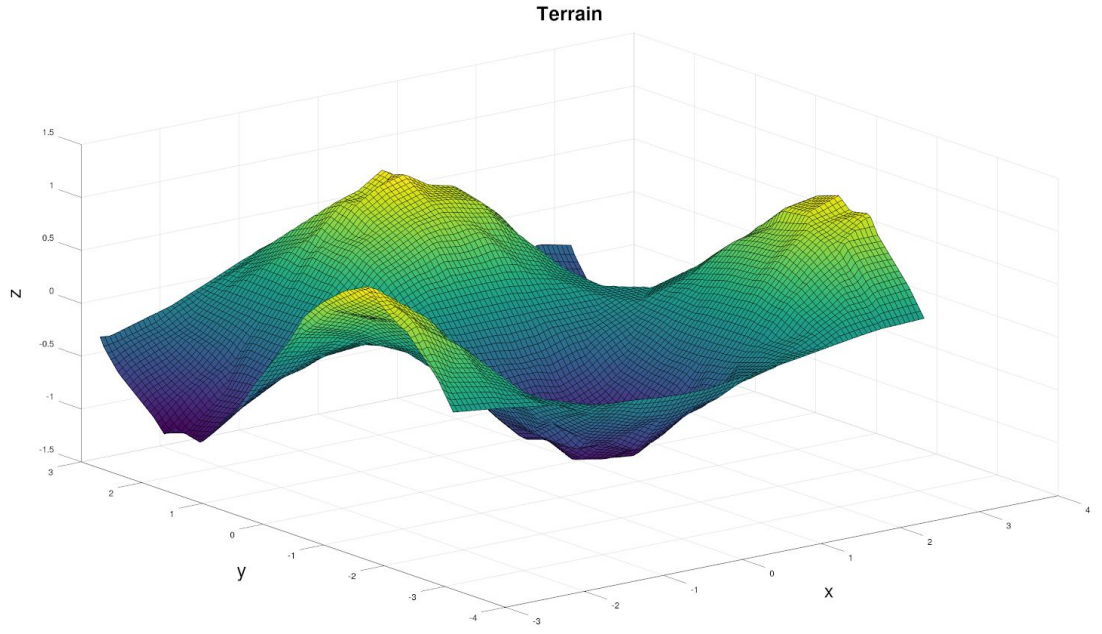


Gráfico 1: Terreno a aproximar.

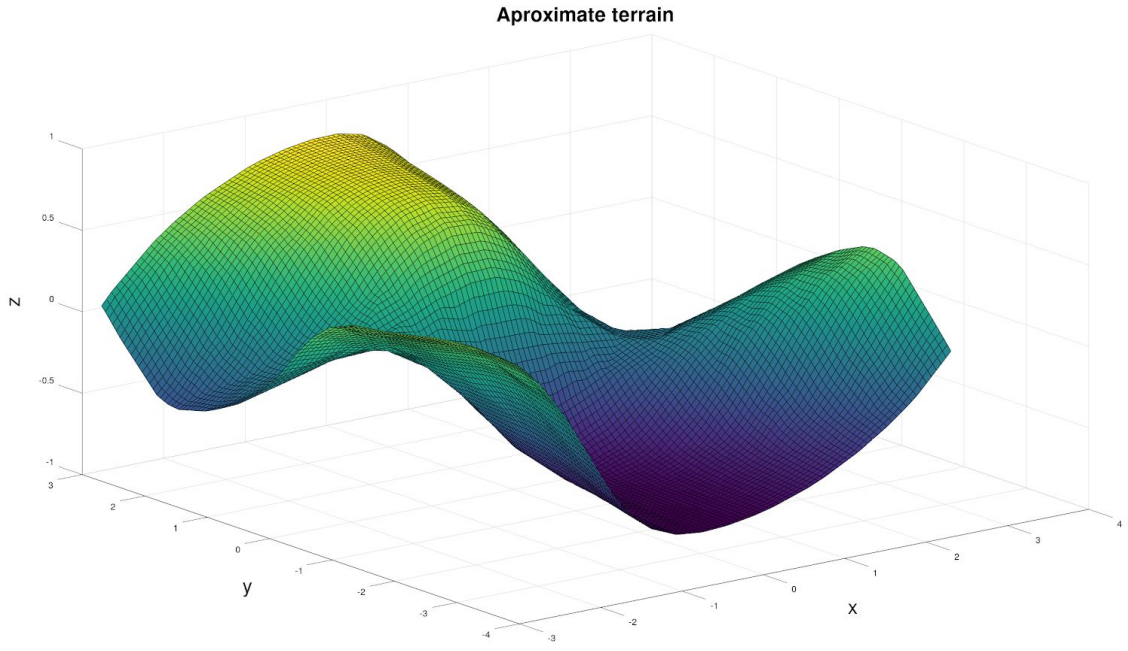
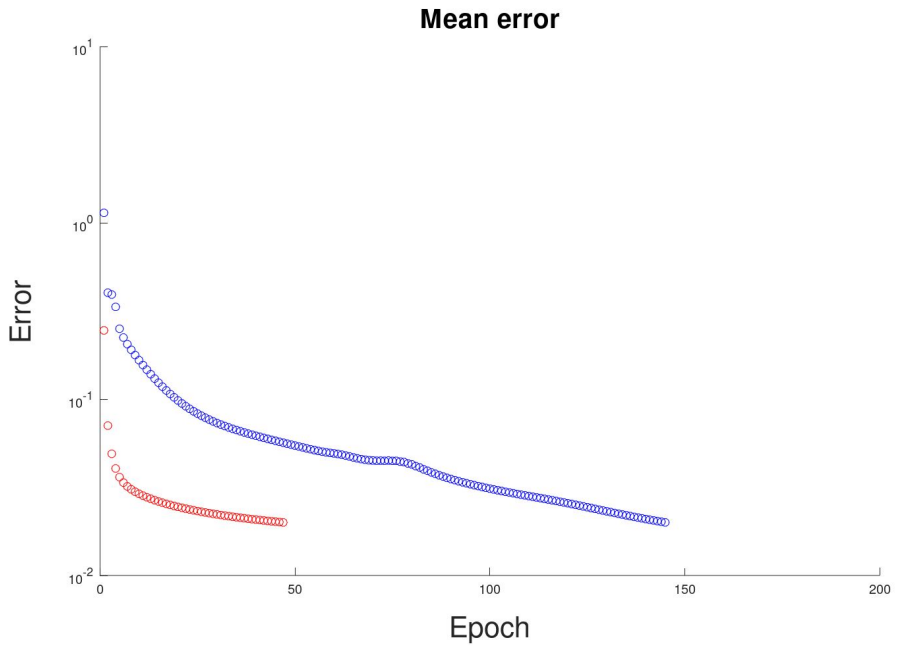
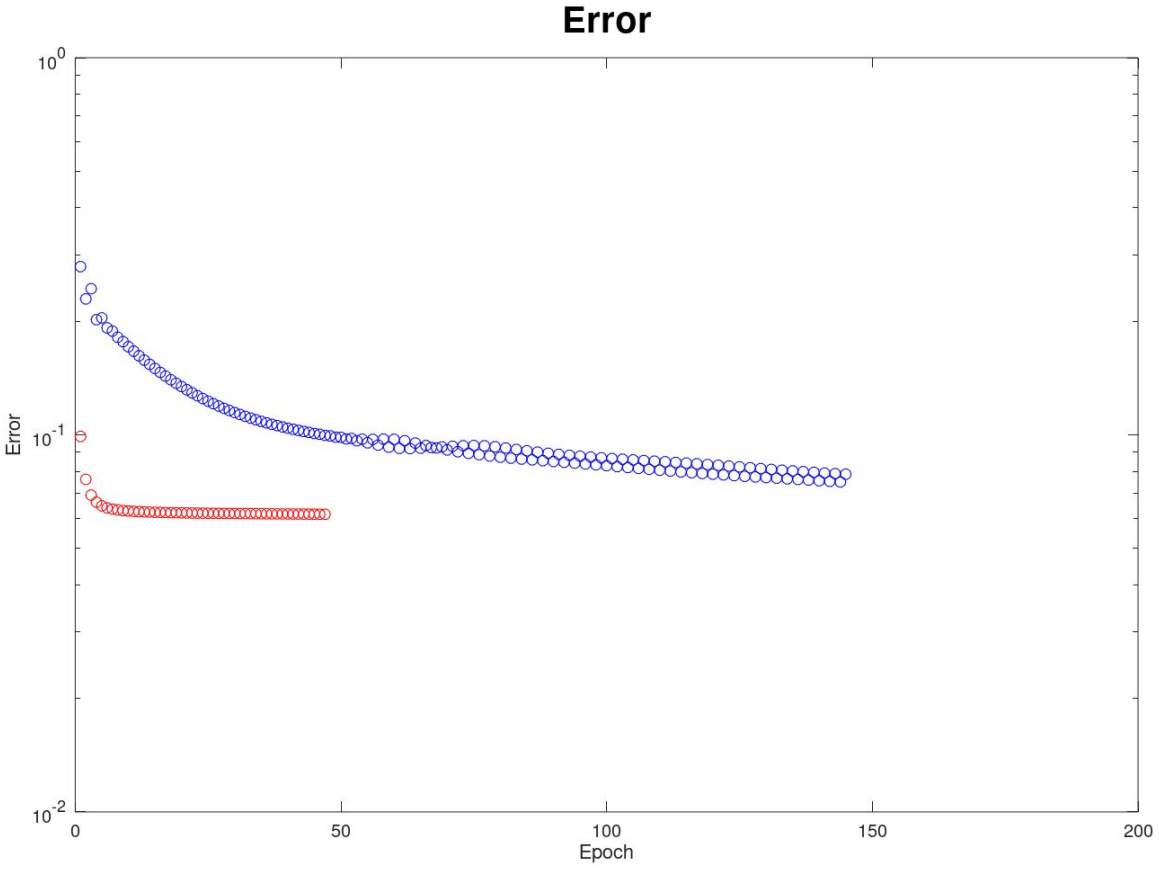
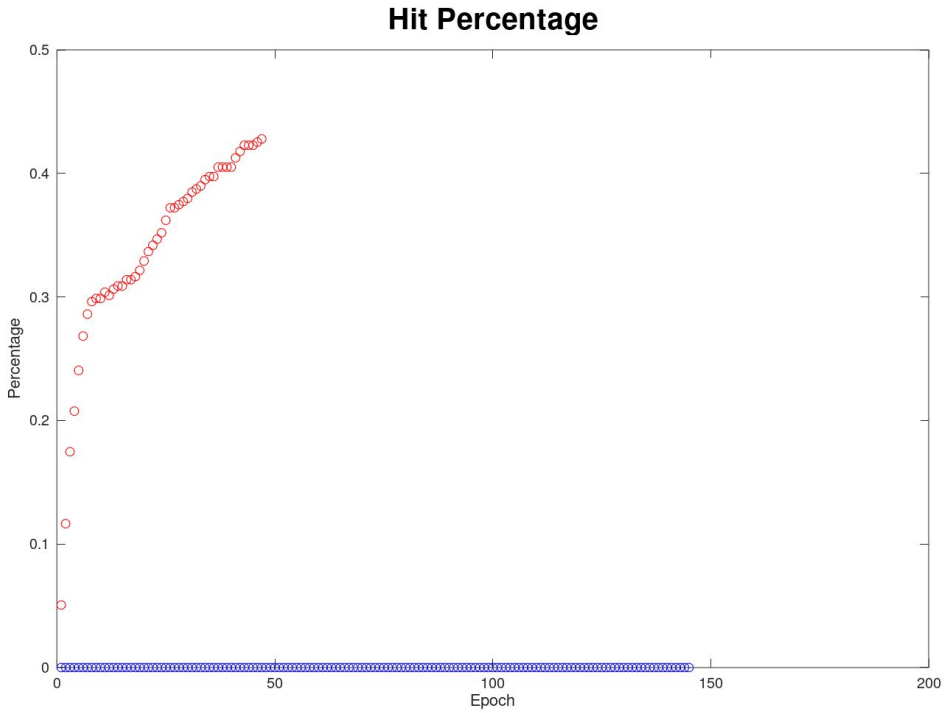


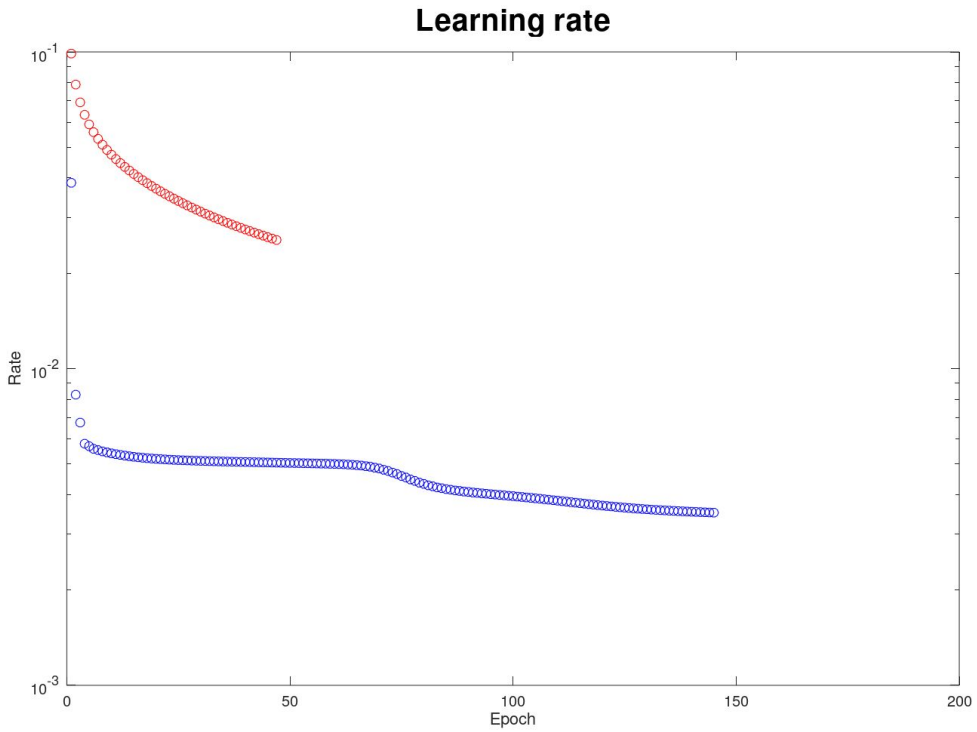
Gráfico 2: Terreno aproximado.

Variaciones para batch o incremental

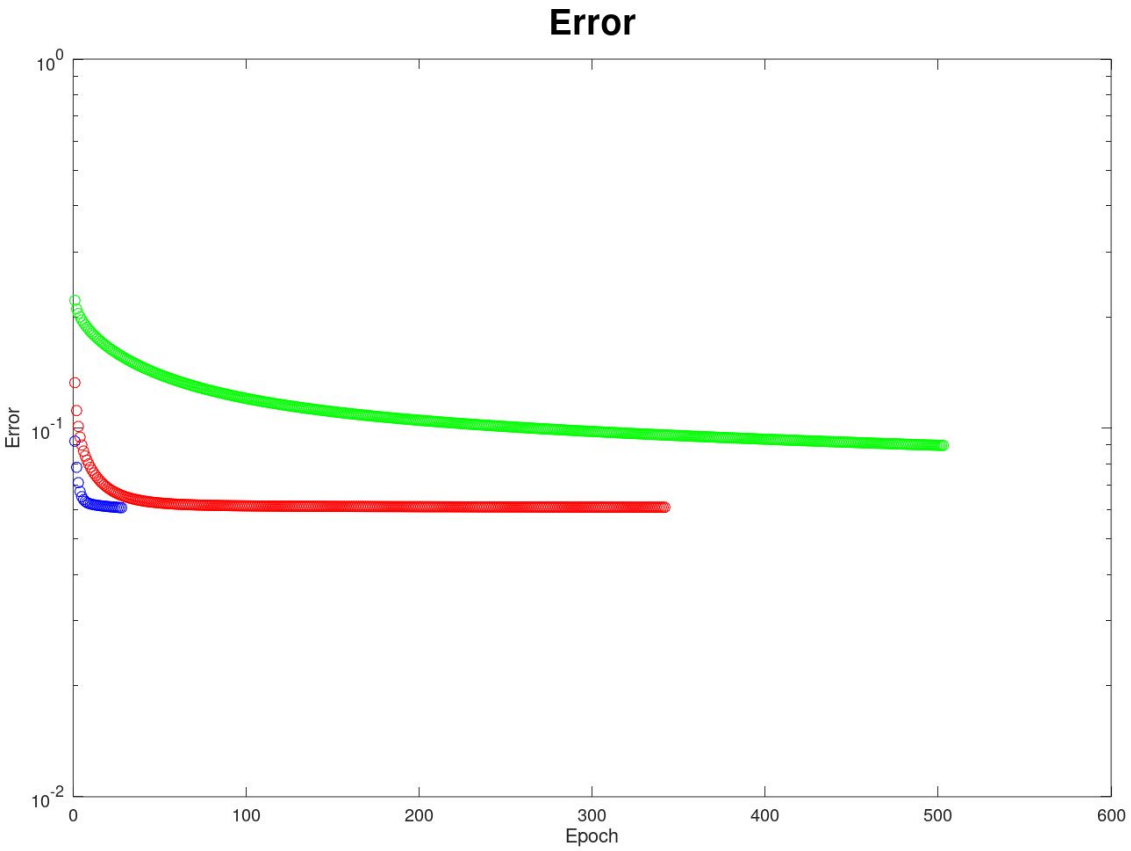


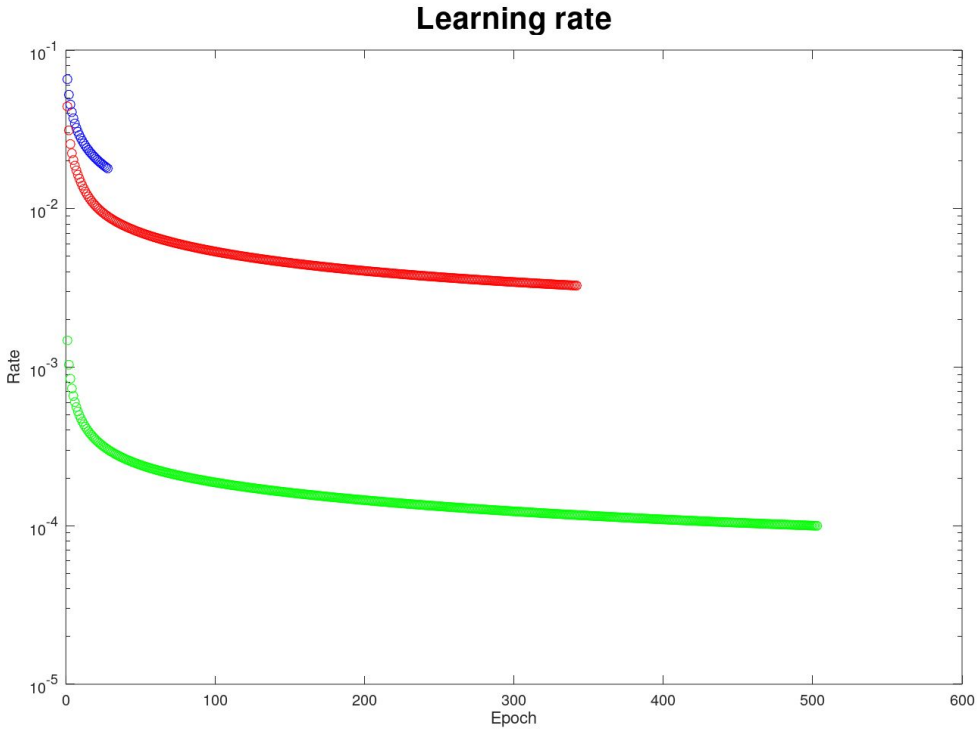


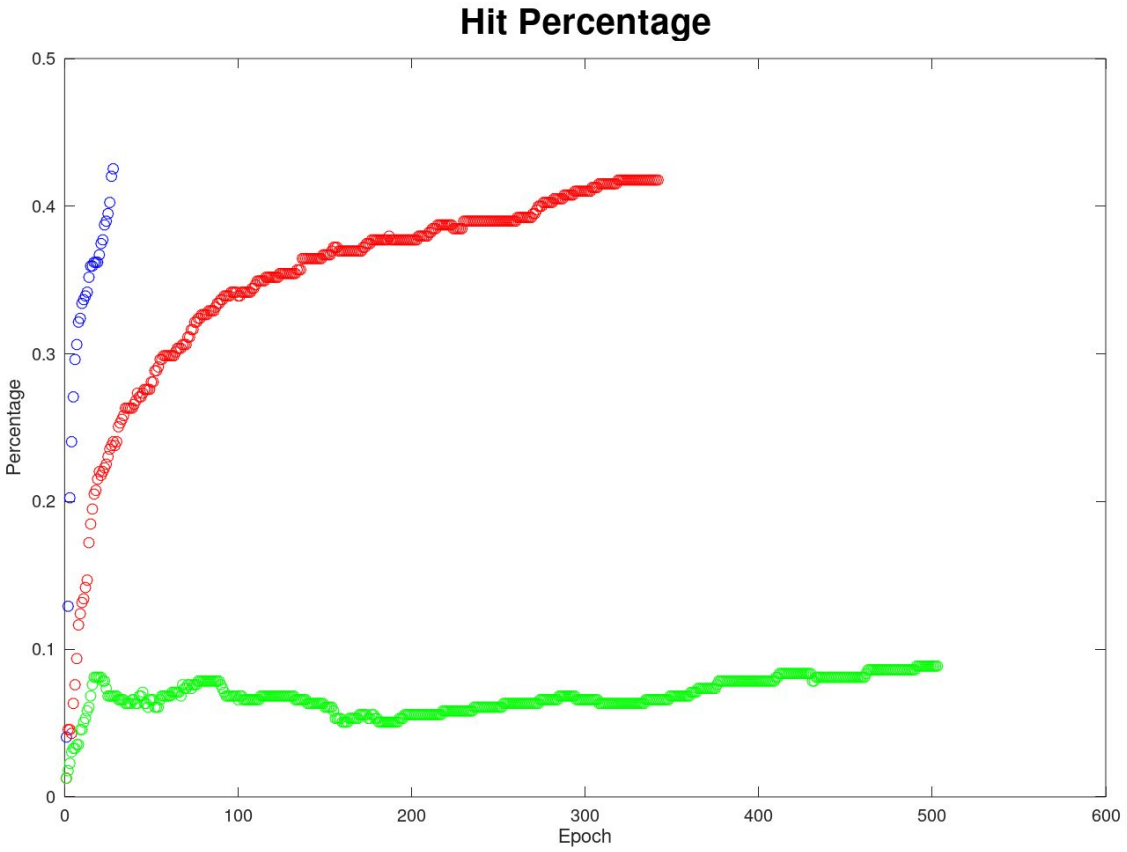

Hit Percentage
Percentage
Epoch
0
0.1
0.2
0.3
0.4
0.5
0
50
100
150
200


Learning rate
Rate
Epoch
10^-1
10^-2
10^-3
0
50
100
150
200

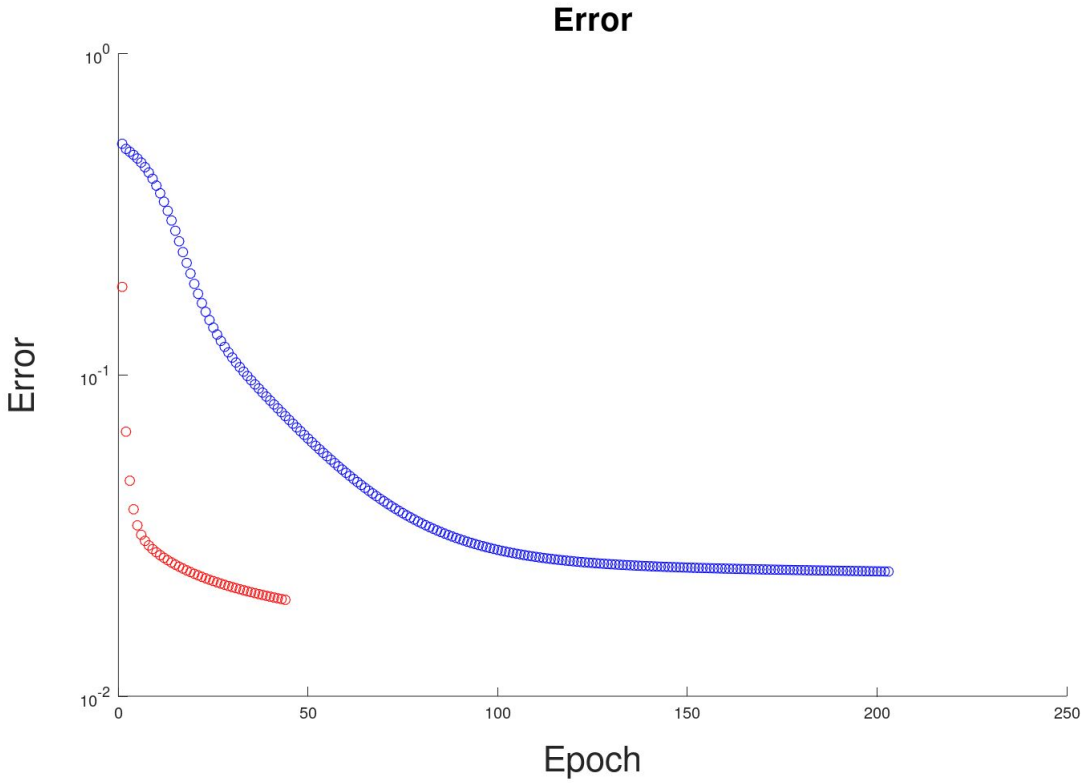
Variaciones en factor de aprendizaje

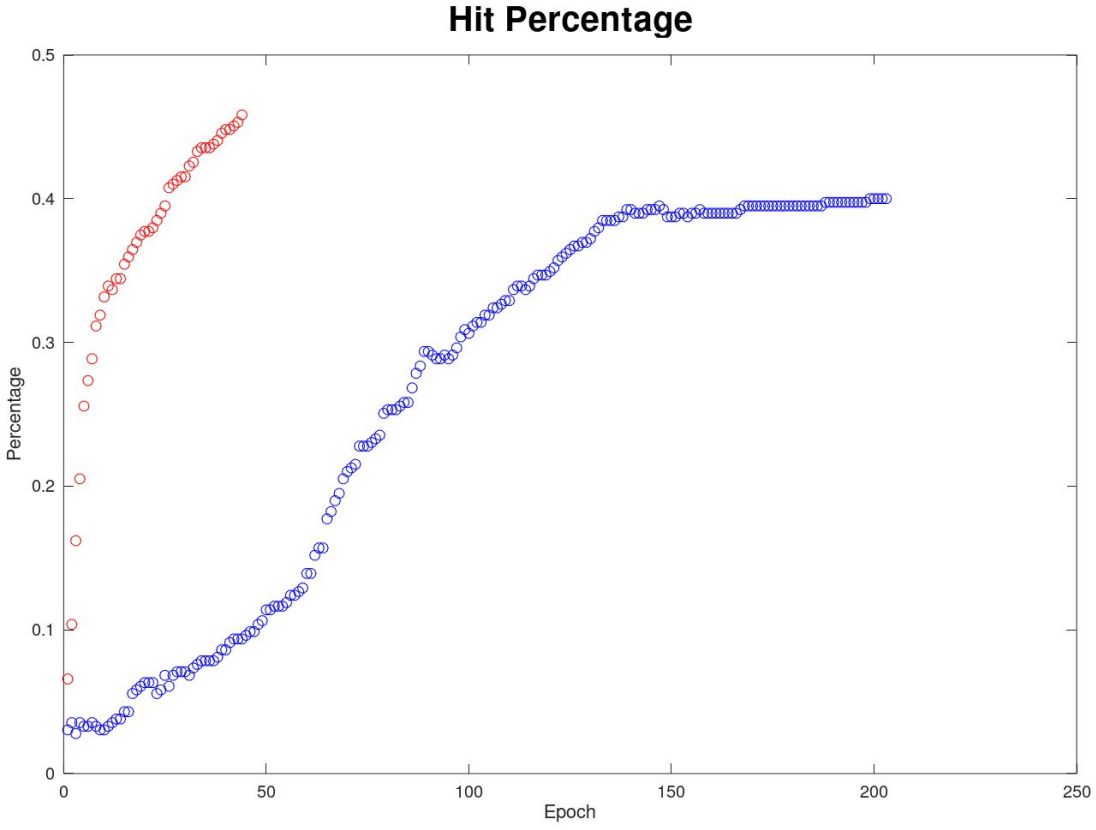


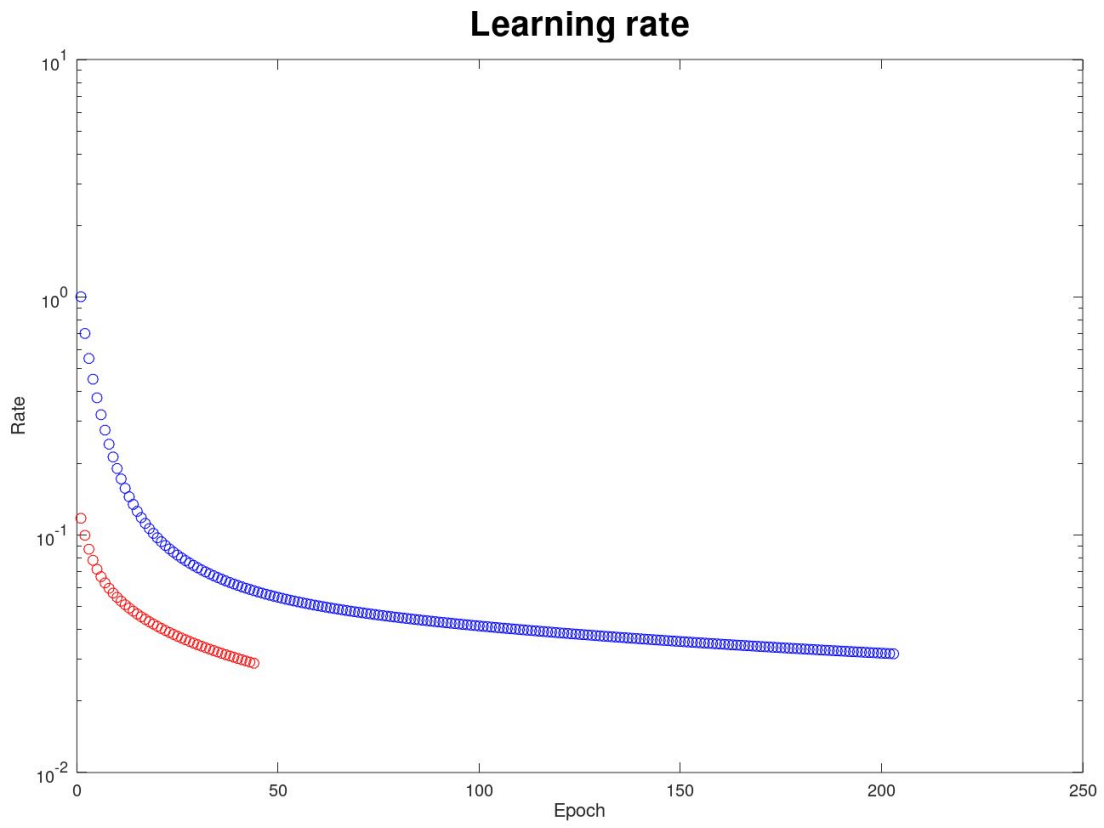


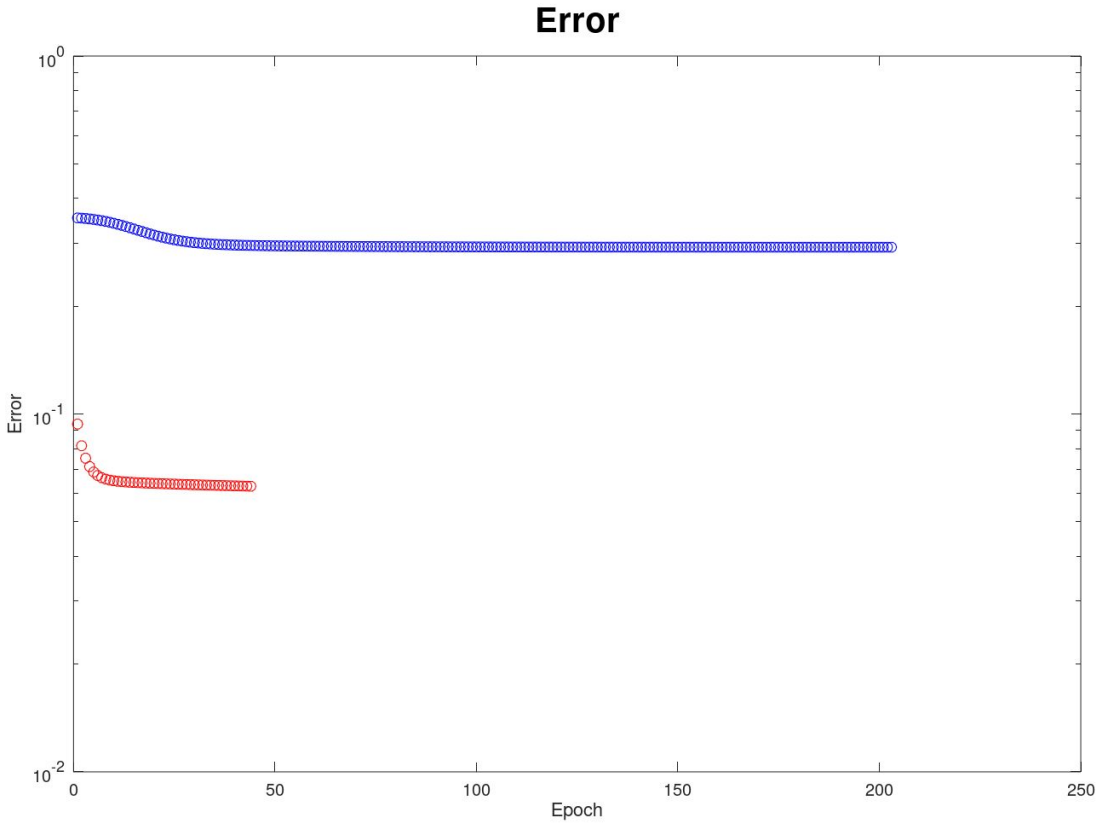

Hit Percentage
Percentage
Epoch
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
600

Variaciones en función de activación

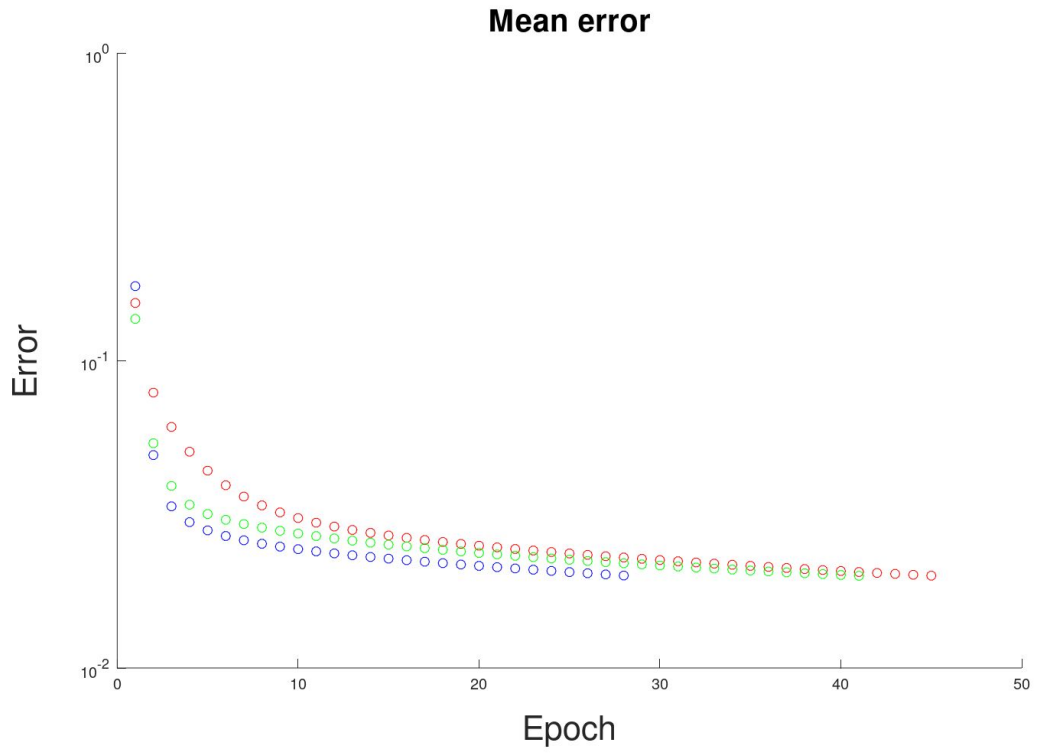


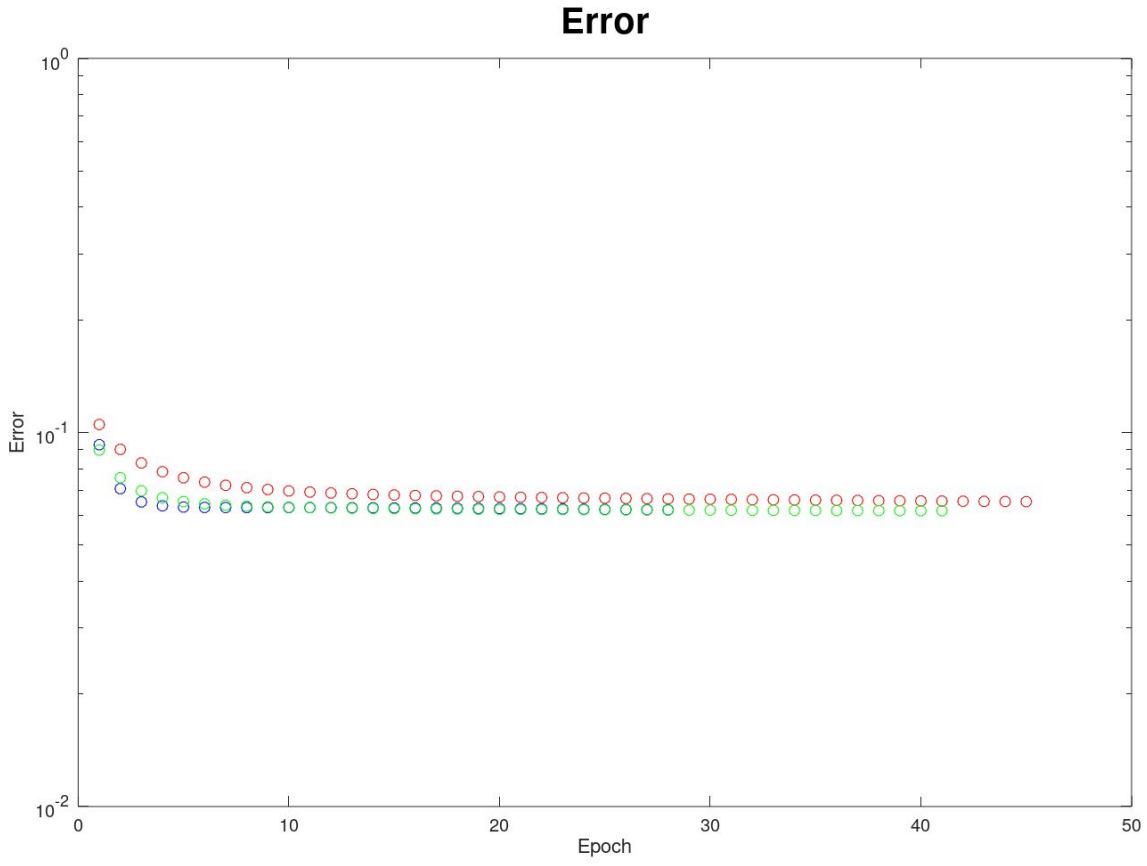


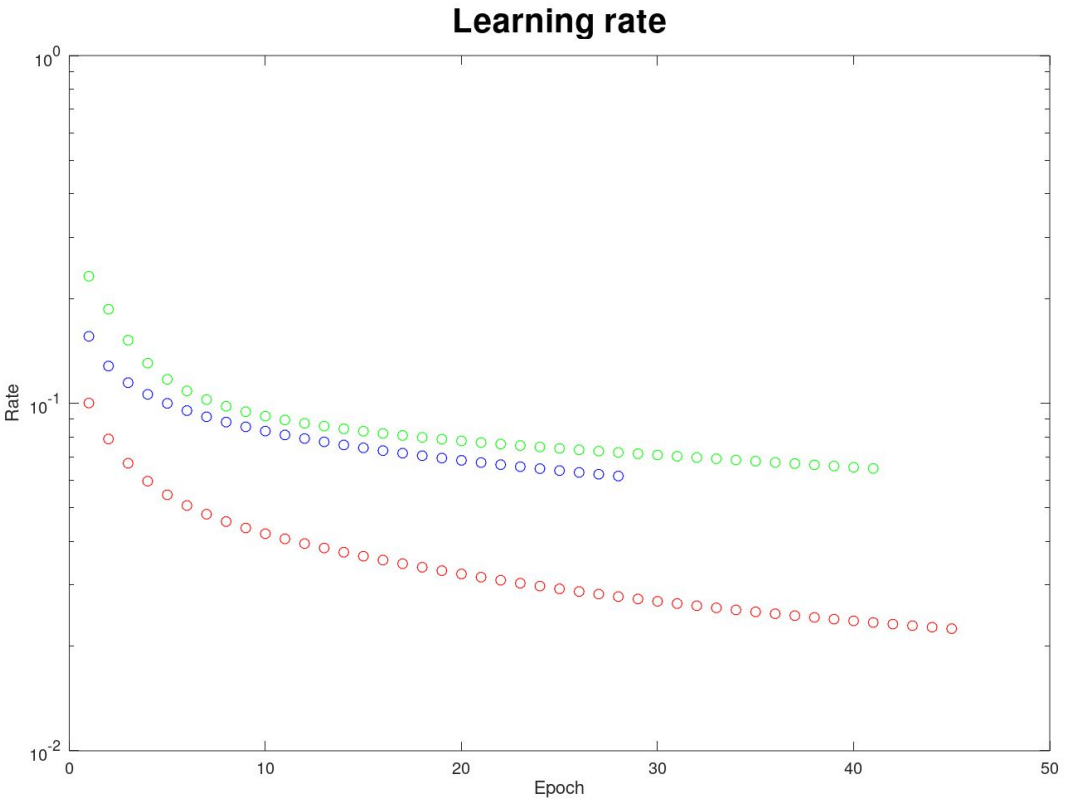

Learning rate
Rate
Epoch
10^{1}
10^{0}
10^{-1}
10^{-2}
0
50
100
150
200
250

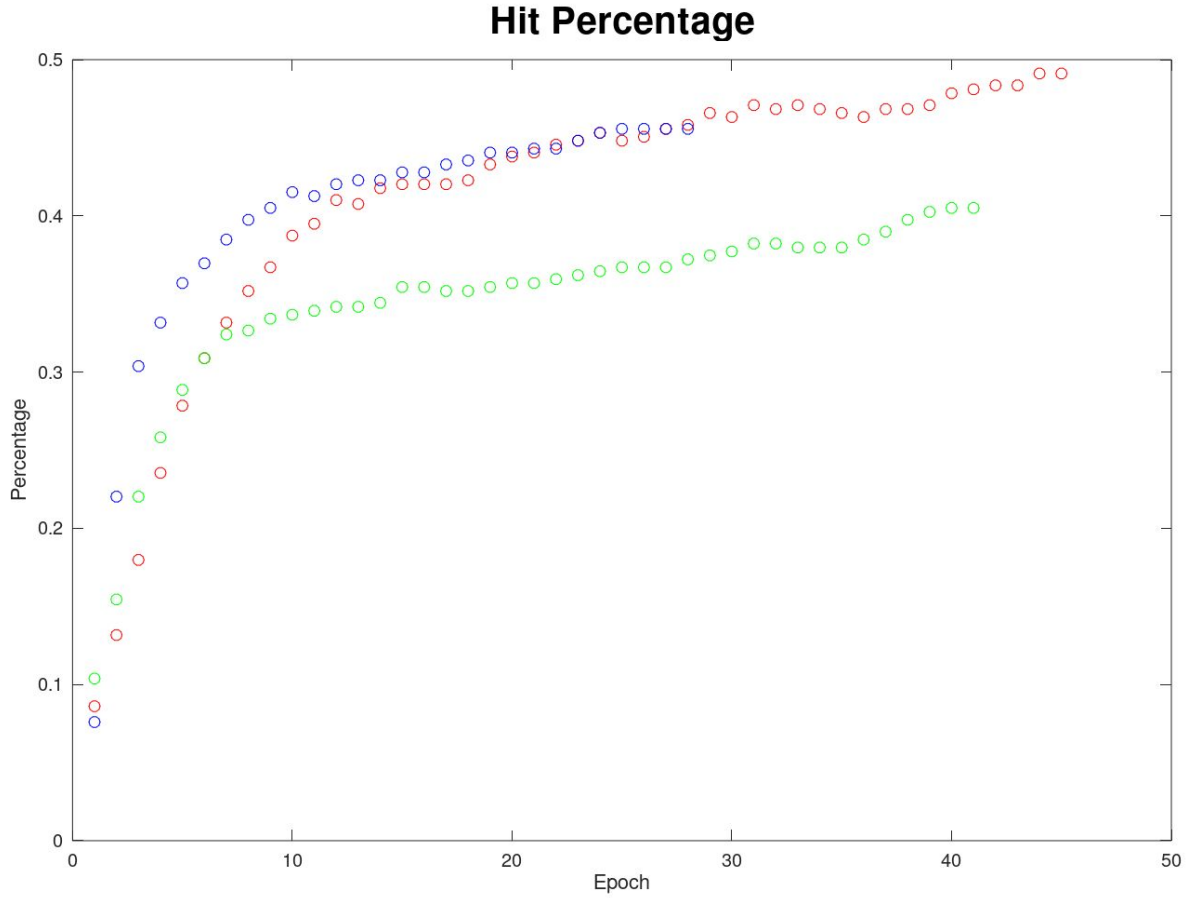

Error
Error
Epoch
10^{0}
10^{-1}
10^{-2}
0
50
100
150
200
250

Variaciones en estructura de red

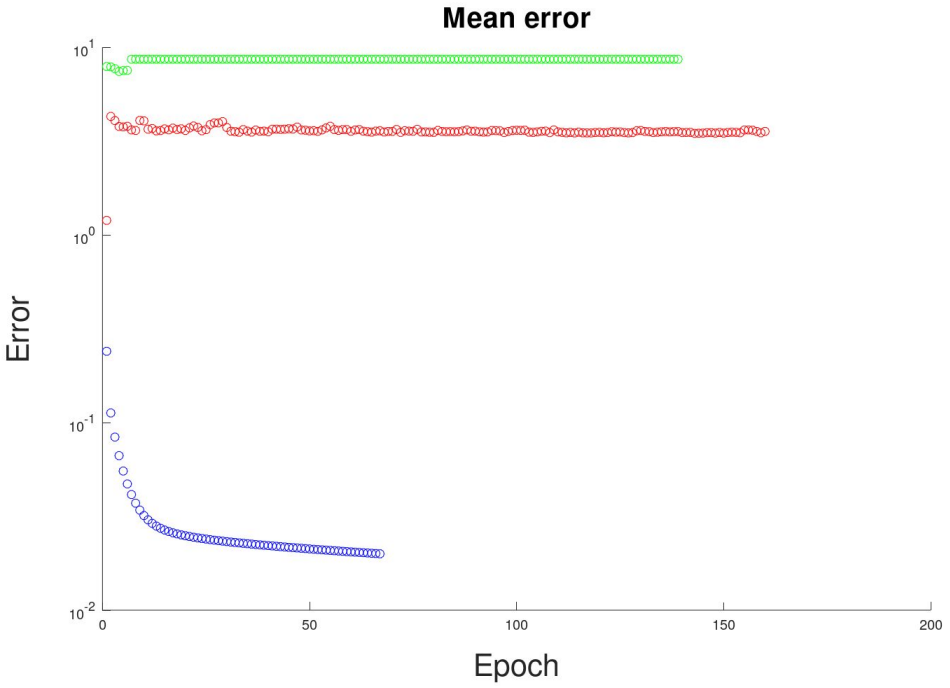




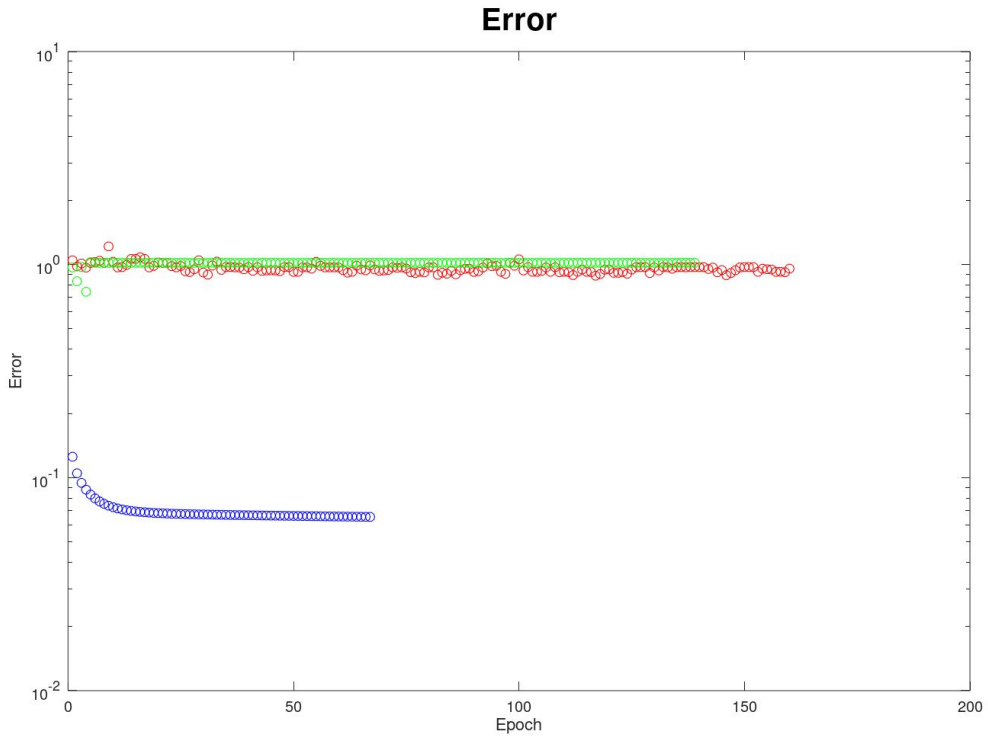

Learning rate
Rate
Epoch
10^0
10^{-1}
10^{-2}
0
10
20
30
40
50

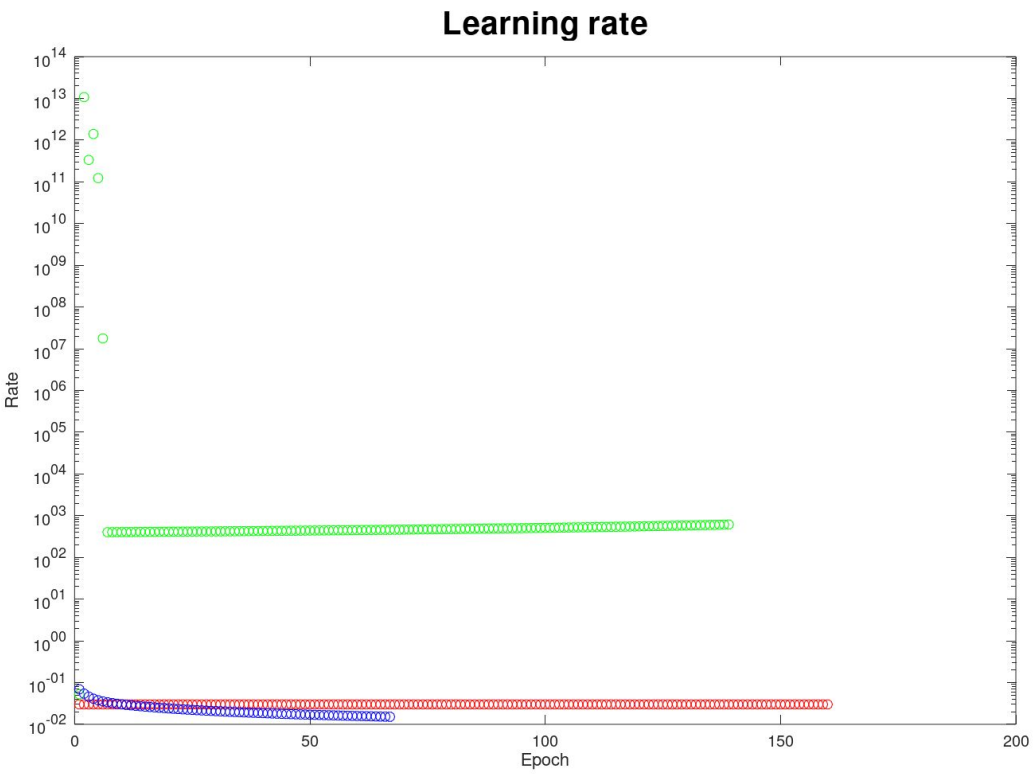

Hit Percentage
Percentage
Epoch
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50

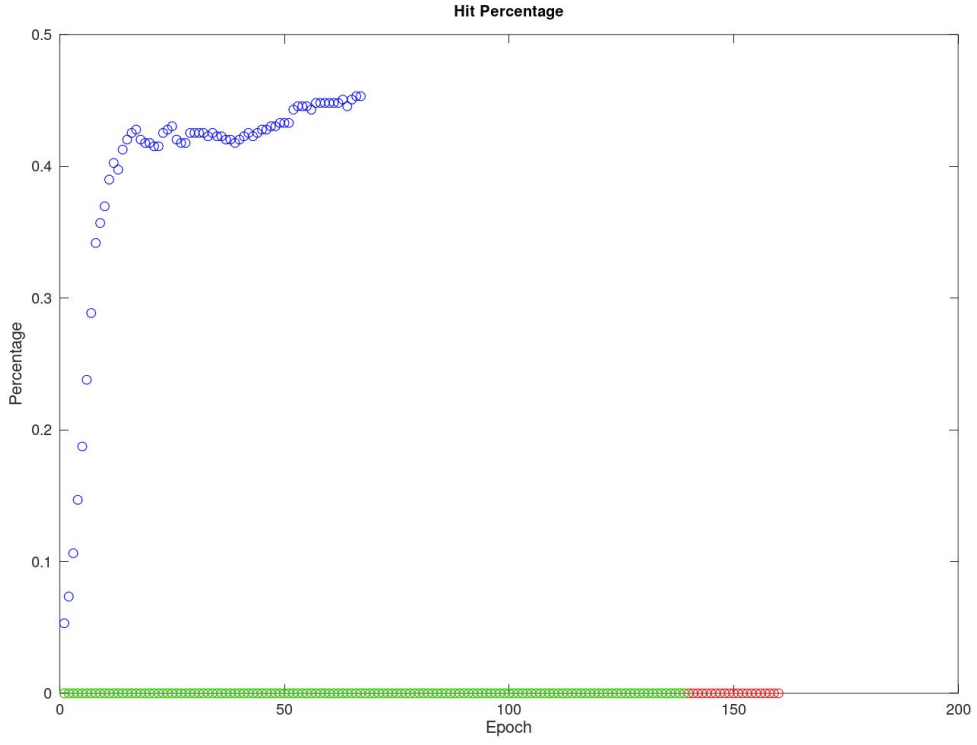
Variación en función del optimizador

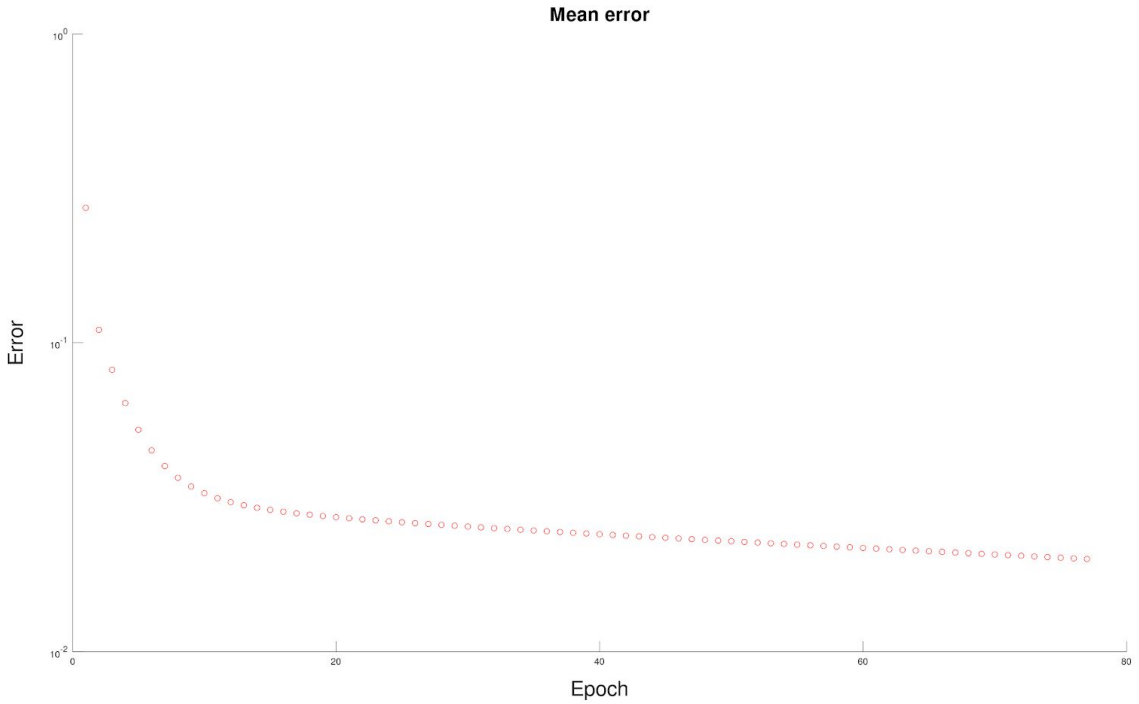


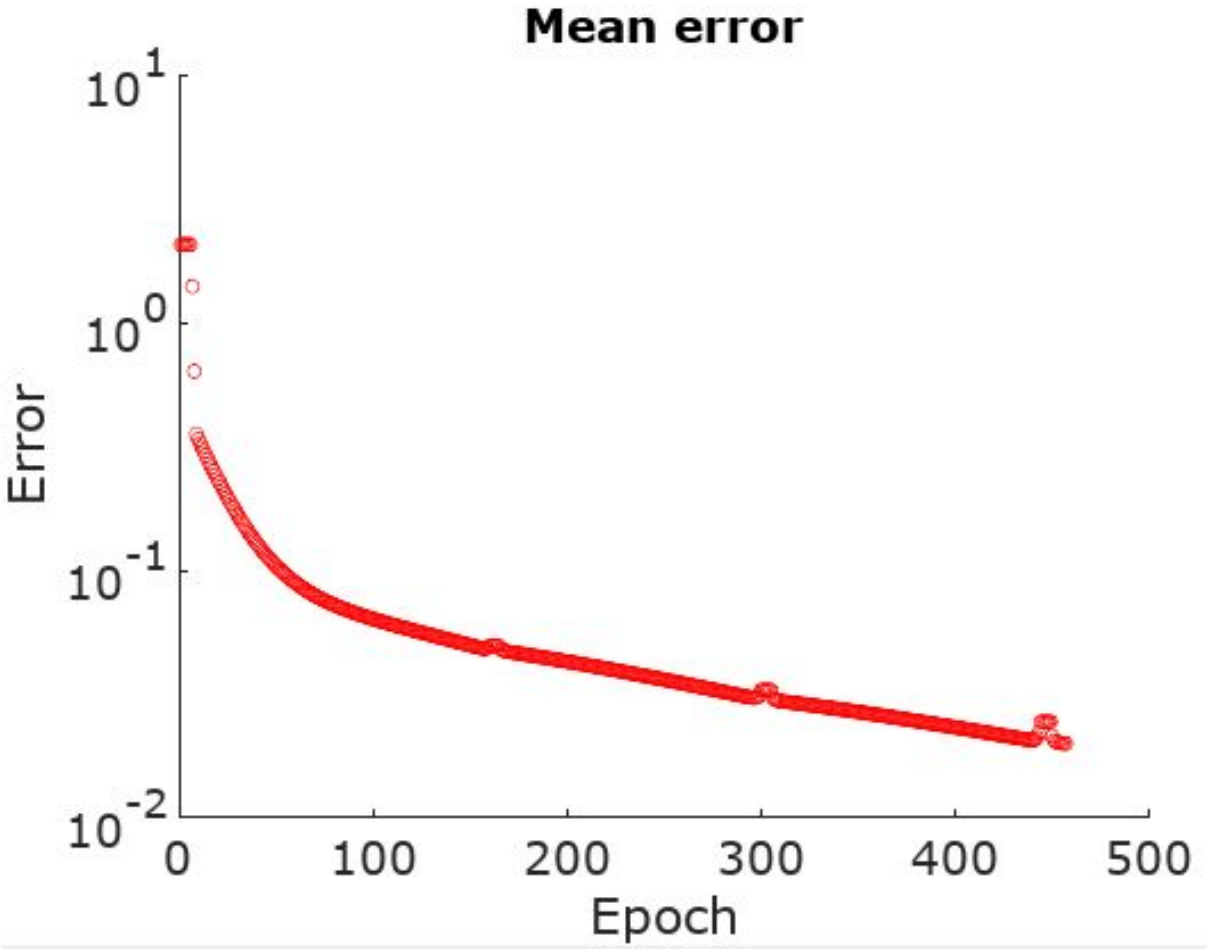
.

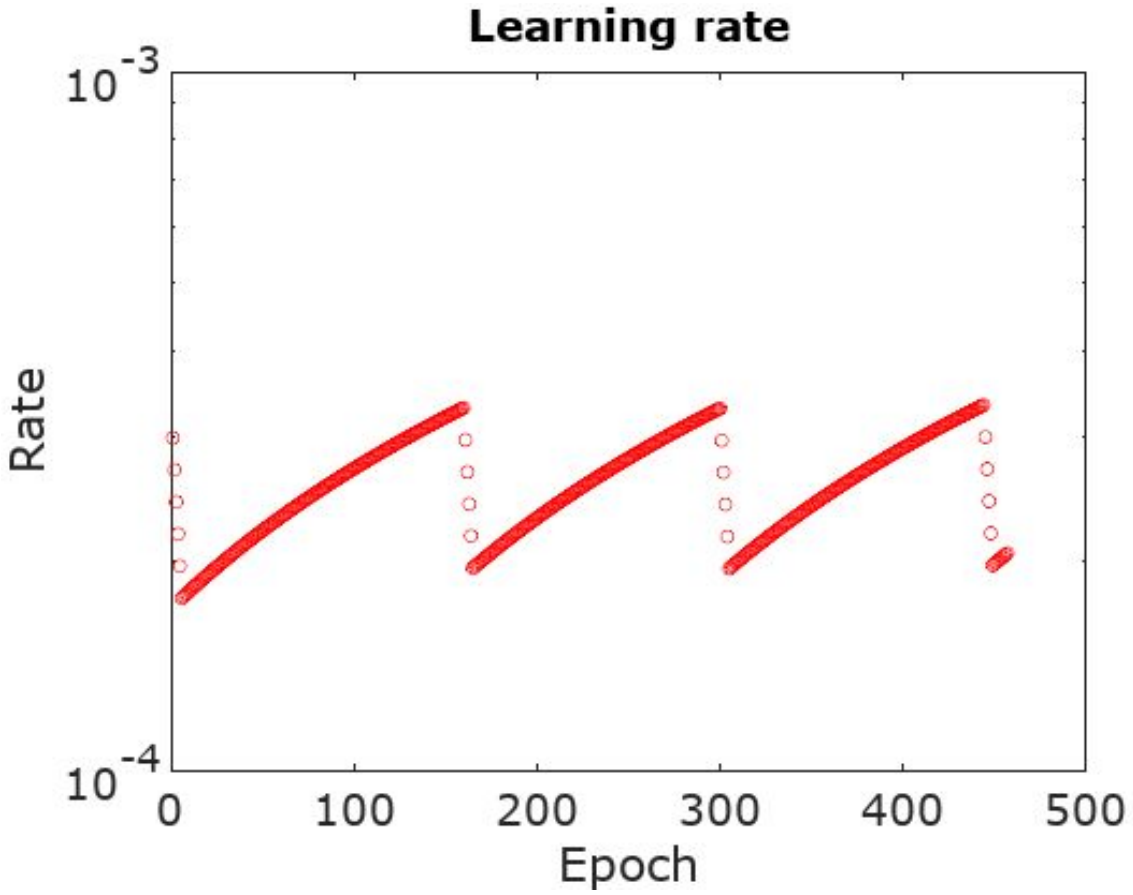



Learning rate
Rate
Epoch
0
50
100
150
200


Hit Percentage
Percentage
Epoch
0
0.1
0.2
0.3
0.4
0.5
0
50
100
150
200


Mean error
Error
Epoch
10^0
10^-1
10^-2
0
20
40
60
80


Mean error
Error
Epoch
10^1
10^0
10^-1
10^-2
0
100
200
300
400
500


Learning rate
10^-3
10^-4
Rate
0
100
200
300
400
500
Epoch