Sistemas de Inteligencia Artificial
TP1 : Redes Neuronales

Grupo 9

Marcos Abelenda, 55086
Santiago Manganaro Bello, 56239
Matias Perazzo, 55024
Agustín Ignacio Vázquez, 55354

Índice

Introducción

El objetivo de este proyecto es realizar la aproximación de un terreno de tres dimensiones a partir de la implementación de una red neuronal. La efectividad de la misma va a depender de la arquitectura propuesta y de su semejanza con la solución original.

La red requiere de dos etapas, una donde se le enseña a la misma otorgándole una capacidad de estimación y otra donde se pone a prueba esta última. Para esto, se cuenta con un set de datos de entrada que representan la función a aproximar y los mismos son divididos en dos grupos, uno utilizado para su aprendizaje y otro para el testeo respectivamente. Para estas etapas se implementaron las funciones de activación sugeridas y se analizó la influencia de cada una sobre la red.

Se tomaron distintas métricas de las arquitecturas propuestas y en base a estas se eligió la óptima. Entre los indicadores más relevantes se encuentran los errores cuadráticos medios de aprendizaje y de testeo, la cantidad de aciertos de la red y la rapidez de convergencia. Al mismo tiempo se analizaron distintas técnicas de mejora y se observó cómo las mismas impactaron sobre las métricas consideradas. Se trataron con problemas inherentes a redes neuronales como la saturación no deseada de capas intermedias, la búsqueda de mínimos locales óptimos, la convergencia y el sobre aprendizaje.

Las optimizaciones implementadas y las distintas arquitecturas consideradas serán detalladas a lo largo de este informe.

Arquitecturas propuestas

Se implementaron distintas arquitecturas de red y se analizó el error cuadrático medio de entrenamiento y de testeo, el porcentaje de aciertos, la saturación de capas intermedias, la cantidad de patrones de entrenamiento y otros indicadores de cada una para luego compararlas entre sí. También se contempló el efecto de las optimizaciones sobre las variables mencionadas generando una serie de observaciones las cuales forman parte de la conclusión final. Las distintas arquitecturas y todo lo mencionado previamente puede ser visualizado en la tabla del archivo Excel "*Cuadro comparativo de redes neuronales*".

Cabe destacar, que el algoritmo implementado es el *back propagation incremental*. En el aprendizaje incremental, el sistema aprende crecientemente con un algoritmo de actualización, y el conocimiento del sistema se actualiza cada vez. En el método basado en Bach, lleva menos tiempo procesar los datos que vienen en bloque, y se actualiza una vez que se procesa. La diferencia es que el aprendizaje ocurre solo cuando se completa el procesamiento por lotes. Uno se beneficia de una administración de datos eficiente, y el otro es más efectivo en "digerir" los datos a través del tiempo. Dependiendo de la aplicación, cada método podría ser más apropiado. Nuestro criterio de decisión, fue dejar en segundo plano al tiempo e

intentar que la red aprendiera lo mejor posible para obtener a largo plazo un menor error cuadrático medio.

Arquitectura elegida

A la hora de destacar una arquitectura por sobre las demás, hay que tener en cuenta diversos factores. En resumen, el tiempo, la precisión, y la generalidad.

En base al análisis, es posible considerar como “mejor red” a aquella red que presente el menor error cuadrático medio al finalizar cierta cantidad de épocas. En este caso, hay que entonces diferenciar dos situaciones. En primer lugar, la red que para la mayoría de las *seeds* probadas tuvo el menor error cuadrático promedio; en segundo lugar, la red que tuvo el menor error cuadrático medio para alguna *seed*.

Por otro lado, se puede definir que una red es mejor que otra, estableciendo un valor de error cuadrático medio máximo aceptado y analizando cuál de ellas llega más rápido a ese valor. Aquí también hay que diferenciar dos situaciones. En primer lugar, la red que para la mayoría de las *seeds* probadas llegó más rápido al valor máximo de error aceptado; en segundo lugar, la red que llegó más rápido al valor para alguna *seed*.

La postura que se eligió al momento de determinar cuál era la mejor red fue a partir del análisis del error cuadrático medio priorizando la que mejor generaliza sobre la que mejor aprende sin considerar su rapidez de convergencia (siempre y cuando sea una cantidad de tiempo razonable). En cuanto a las *seeds* se considera el error más bajo para alguna semilla, no es necesario que la red seleccionada tenga el menor error para la mayoría de las *seeds* probadas.

Manteniendo el criterio mencionado se seleccionó la red 2,15,10,1 (dos capas ocultas: la primera con 15 neuronas, la segunda con 10) y la semilla elegida fue la “D”. Se obtuvo 1,14E-04 de error cuadrático medio de entrenamiento, 1,96E-04 de error cuadrático medio de testeo, 88% de aciertos en el aprendizaje y 86,52% de aciertos para la generalización.

Optimizaciones

Saturación de la red

Dado que es necesario realizar sumas pesadas cuando se opera con una red neuronal esto puede generar problemas cuando las entradas son valores de gran magnitud, provocando la saturación indeseada de capas intermedias.

Normalización

Para tratar el problema mencionado previamente se consideró la normalización de los datos de entrada. De todas formas dado que nuestro terreno a aproximar contaba con datos de entrada pequeños (valores entre -1 y 1) se decidió no realizar la normalización de los mismos.

Mínimos locales

Dado que la red utiliza el método del gradiente descendente, la misma realiza una búsqueda de un mínimo local en la superficie obteniendo como resultado una aproximación a la solución original del problema. En un planteo ideal cualquier mínimo hallado sería óptimo, pero en la realidad esto no es cierto y se tiene que hacer lo posible para evitar falsos positivos (hallar mínimos locales que parecen óptimos pero no lo son). Para esto, se realizaron las técnicas mencionadas a continuación.

Shuffling del set de entrenamiento

Para evitar mínimos locales no muy profundos y poco óptimos se realiza una mezcla de los patrones de entrada de la red. Esto genera variaciones en el recorrido sobre la superficie evitando estos mínimos e influye en el aprendizaje de la misma, ya que no se la predispone a aprender los patrones de entrada siempre en el mismo orden.

Eta Adaptativo

Esta técnica se utiliza para evadir mínimos locales profundos y poco óptimos. Consta en analizar el error generado en los últimos *n* pasos y distinguir si el mismo incrementó o disminuyó. Si se trata del primer caso, se realiza un retroceso en la ejecución del algoritmo, volviendo a un punto anterior y decrementando el eta, para que la solución converja. El cómputo de datos es mayor, afectando la performance de la red. En el segundo caso se aumenta el eta para dar “saltos” más grandes en la superficie logrando salir de un mínimo local muy profundo que no sea óptimo. El incremento conlleva una disminución en la cantidad de cómputo.

Convergencia

Eta Adaptativo

Tal como se explicó anteriormente, si el error en los últimos *n* pasos viene decreciendo, se aumenta el valor del *learning rate.* De esta manera se logra, entre

otras cosas, acelerar la convergencia, pues se continúa avanzando en la dirección correspondiente, disminuyendo el error más rápidamente.

Momentum

La idea de este método es de acelerar la convergencia sumando una proporción de la variación de los pesos anteriores haciendo que nos movamos aún más en la superficie en la dirección actual. Como mencionamos anteriormente a partir del análisis de los últimos *n* pasos podemos conocer cuando el error aumenta o disminuye, pudiendo diferenciar cuándo aplicar momentum y cuando no. Es decir, si el error está creciendo no aplicamos esta técnica porque no queremos avanzar más en una dirección no favorable, por otro lado si el error está decreciendo queremos aplicarla para converger más rápido a esta solución.

Cota mínima de error

El objetivo de esta práctica es de finalizar el entrenamiento de la red cuando se llega a un error aceptable de la solución aproximada. Esto reduce el cómputo ya que se finaliza la iteración sobre la cantidad de épocas. Además resulta útil para evitar algunos casos de sobre aprendizaje de la red ya que probablemente cuando la misma tenga un error próximo al indicado, éste disminuya en unos pocos decimales en las siguientes iteraciones.

Capa de salida

Dado que la salida del terreno a simular se encontraban entre -1 y 1 se decidió que la función de activación de la capa de salida sea tangencial hiperbólica en vez de lineal, ajustándose a las características del problema y logrando la convergencia en casos donde no convergía la red.

Solución aproximada

Al momento de elegir una o más soluciones óptimas de las candidatas para una arquitectura de red se tuvieron en cuenta además de los indicadores mencionados a lo largo de este informe los siguientes factores.

Semillas óptimas

Dado que los pesos iniciales son pseudoaleatorios estos dependen de la semilla tomada al momento de su creación, por lo tanto la solución propuesta por una arquitectura puede variar considerablemente. Debido a esto se optó por almacenar aquellas semillas que generaron un alto rendimiento de nuestra arquitectura elegida.

Peso óptimo

Se observó que los errores cuadráticos medios (aprendizaje y *testing*) presentan oscilaciones que son cada vez más pequeñas a medida que la red converge, por lo que se optó por almacenar los pesos correspondientes al menor error hasta el momento. Como nuestra postura es priorizar aquellas redes que generalizan mejor sobre las que aprenden mejor se almacenan los pesos según el error cuadrático medio de testeo.

Observaciones

Se dedujeron las siguientes observaciones a partir del archivo mencionado que contiene más de 50 casos de prueba:

*Aclaración : dos redes con una misma arquitectura tienen un mismo estado inicial si tienen el mismo vector de entrenamiento, eta inicial, cantidad de épocas, flags (momentum, eta adaptativo, shuffle, etc) e iguales parámetros de optimización para cada mejora implementada.

- A partir del análisis de los casos de las líneas 23 y 24 se puede observar como la reducción del tamaño de entrenamiento para una misma arquitectura con un mismo estado (sin considerar en este caso el tamaño de entrenamiento como parte del estado) y una misma semilla afecta directamente al error medio de aprendizaje y de testeo y a los aciertos. Impacta considerablemente sobre el rendimiento de la red a las mil épocas (se puede observar un rendimiento 10 veces menor en el caso de los errores) y en menor medida sobre el final (casi la mitad en cuanto a aciertos).

- Comparando el caso de la línea 13 con el de la 15 y el de la 44 con el de la 50 se puede apreciar que las redes probadas tuvieron mejor convergencia y lo hicieron a una mayor velocidad cuando se aplicó la tanh como función de activación en vez de la exponencial.

- Comparando los casos de las líneas 44 y 45 se puede interpretar como incide el flag de momentum sobre la convergencia de una red con misma arquitectura, estado y semilla. Esto se puede apreciar aún más si se compara a las 1000 épocas teniendo errores más grandes en el caso en el que se desactivo el flag. De todas formas se puede ver que convergen al mismo valor a largo plazo.

- Algo similar sucede con los casos de las líneas 48 y 49. El desactivar el flag del momentum de una a otra, deriva en una pérdida de efectividad, es decir, aumenta el error cuadrático medio. En este caso, no solo influye en la convergencia a corto plazo, sino que también se puede observar una diferencia significativa una vez terminada la totalidad de las épocas.

- Interpretando los casos de las líneas 23 y 25 se puede observar que los indicadores de una misma red con un mismo estado pueden variar notablemente a partir de una semilla distinta. También se puede ver la influencia de las épocas en la convergencia al momento de comparar semillas. Todo parecía indicar que el caso 23 iba a ser elegido sobre el 25 al momento de compararlos en las 1000 épocas, pero al seguir iterando el último terminó teniendo mejor performance que el primero.

- Otro caso en el que se puede analizar la influencia de la semilla es teniendo en cuenta los casos de las líneas 3 y 34 con la semilla “D” y 31 y 32 con la semilla “C”. En estos cuatro casos, se prueban dos redes con esas dos semillas distintas: la red 2,5,5,1 (X) y la red 2,15,10,1 (Y). Al analizarlo con la semilla “C”, la red que converge al menor error en la misma cantidad de épocas es la X. Sin embargo, al analizar lo mismo con la semilla “D”, es la Y.

- Si se analizan los casos de las líneas 33 y 34 se puede apreciar que el primero tiene un error cuadrático medio de aprendizaje menor que el segundo (es decir, la red aprende mejor) pero que que su error cuadrático medio de testeo es mayor (la red generaliza peor). La red aprende mejor que la segunda pero generaliza peor que esta.

- Analizando los casos de las líneas 3 y 52, en los que se prueba la misma red con los mismos parámetros exceptuando el flag del *shuffle* (en el caso de la línea 3 desactivado, en el otro activado) se puede observar que no siempre el *shuffle* resulta útil. En este caso, se da que al activar el shuffle, el error medio cuadrático a las 1000 épocas aumenta.

- Algo similar sucede en los casos de las líneas 39 y 40. Sin embargo, en este caso, al tener la misma cantidad de épocas, podemos detectar que el *shuffle* resulta útil a largo plazo. Si bien el shuffle hace que converja más lento, llega un punto en el que, gracias a la implementación del mismo, se obtiene un error menor. Esta mejoría, se observa en mayor medida a la hora del testeo, no tanto en el aprendizaje.

- Si se observan los casos de las líneas 40 y 41, se puede ver la diferencia que conlleva la implementación del eta adaptativo. Se observa a corto plazo,

alrededor de un 35% de mejoría en el error cuadrático medio, porcentaje que a largo plazo se hace más pequeño, en lo que concierne al aprendizaje. Se observa a su vez, ahora en el testeo, alrededor de un 25% de mejoría en el error cuadrático medio, porcentaje que, a largo plazo, una vez más, disminuye.

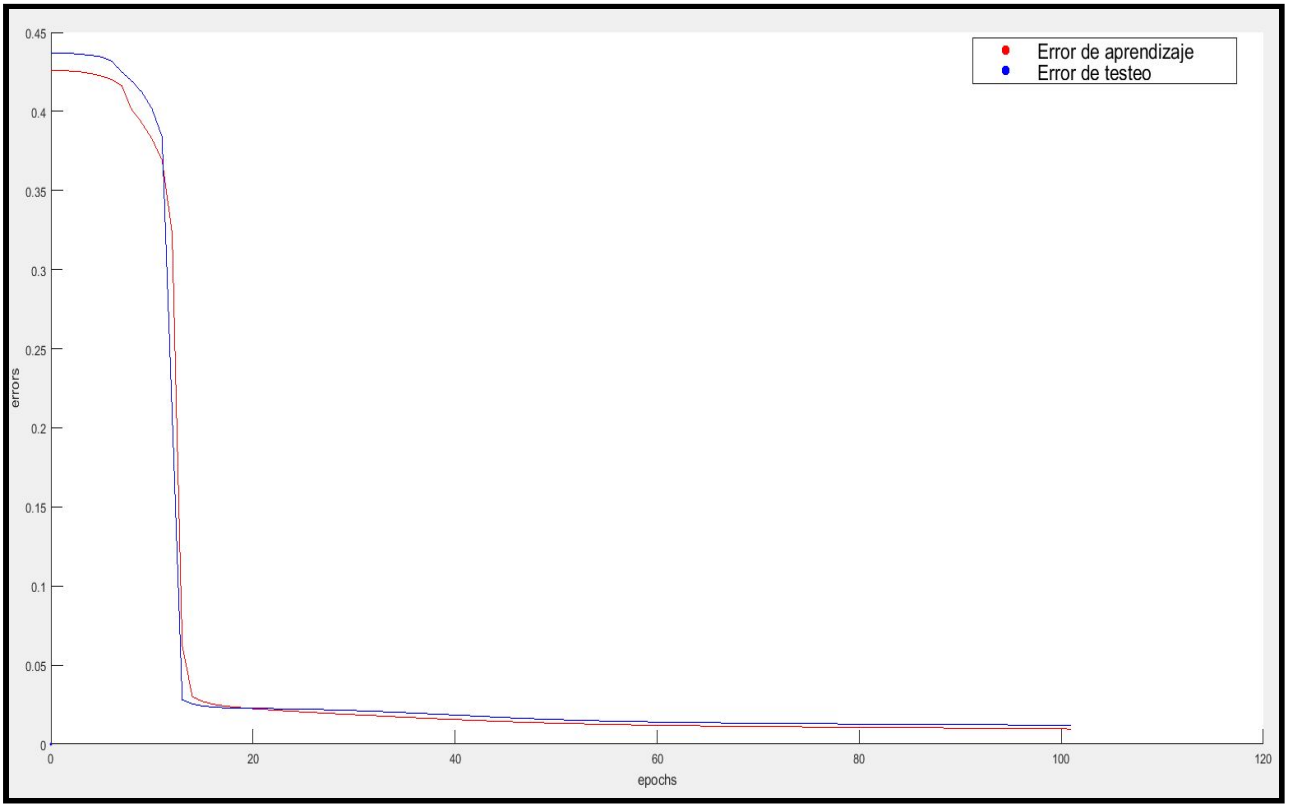
- Cuanto mayor es el eta inicial, más diverge la red. Sin embargo, gracias a la implementación de eta adaptativo, es posible lograr la convergencia de la misma.

Gráficos ilustrativos

Shuffle flag desactivado y activado

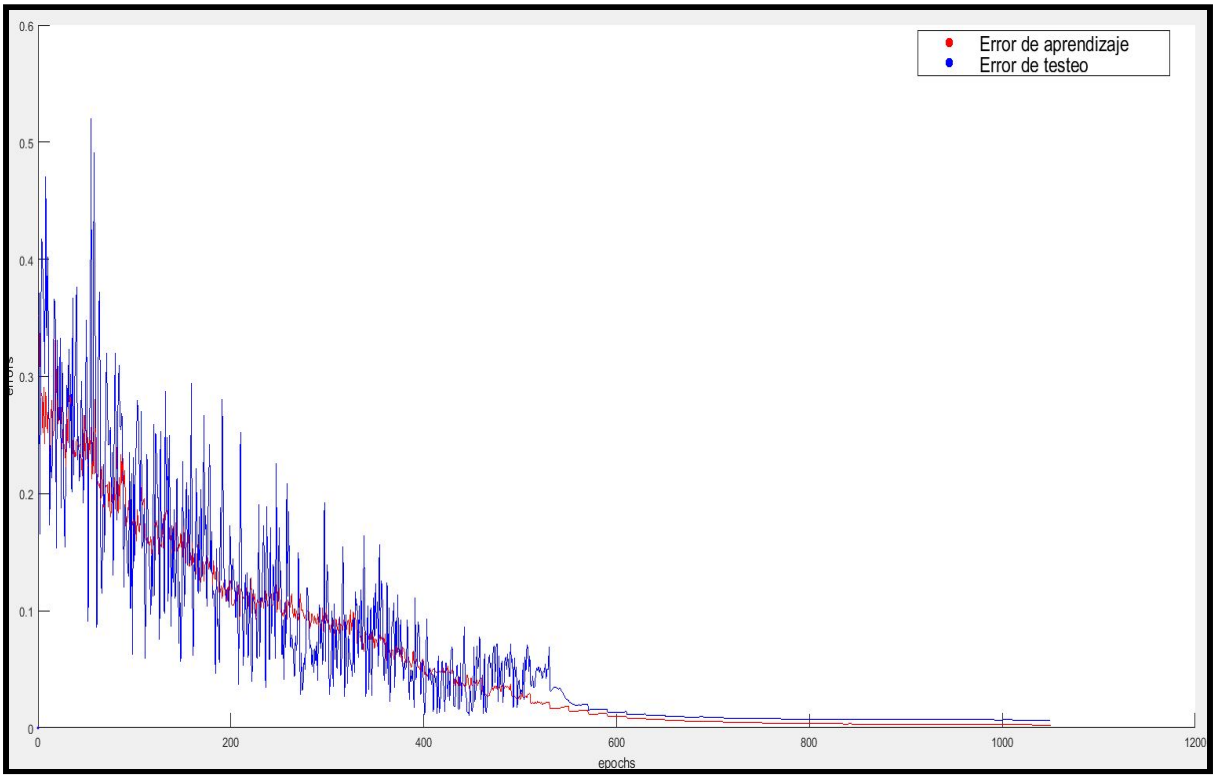
Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 0, alpha_momentum 0.5, seed “D”, tanh 1

Primeras 100 épocas



Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.5, shuffle_flag 1, alpha_momentum 0.5, seed “D”, tanh 1

Primeras 1000 épocas

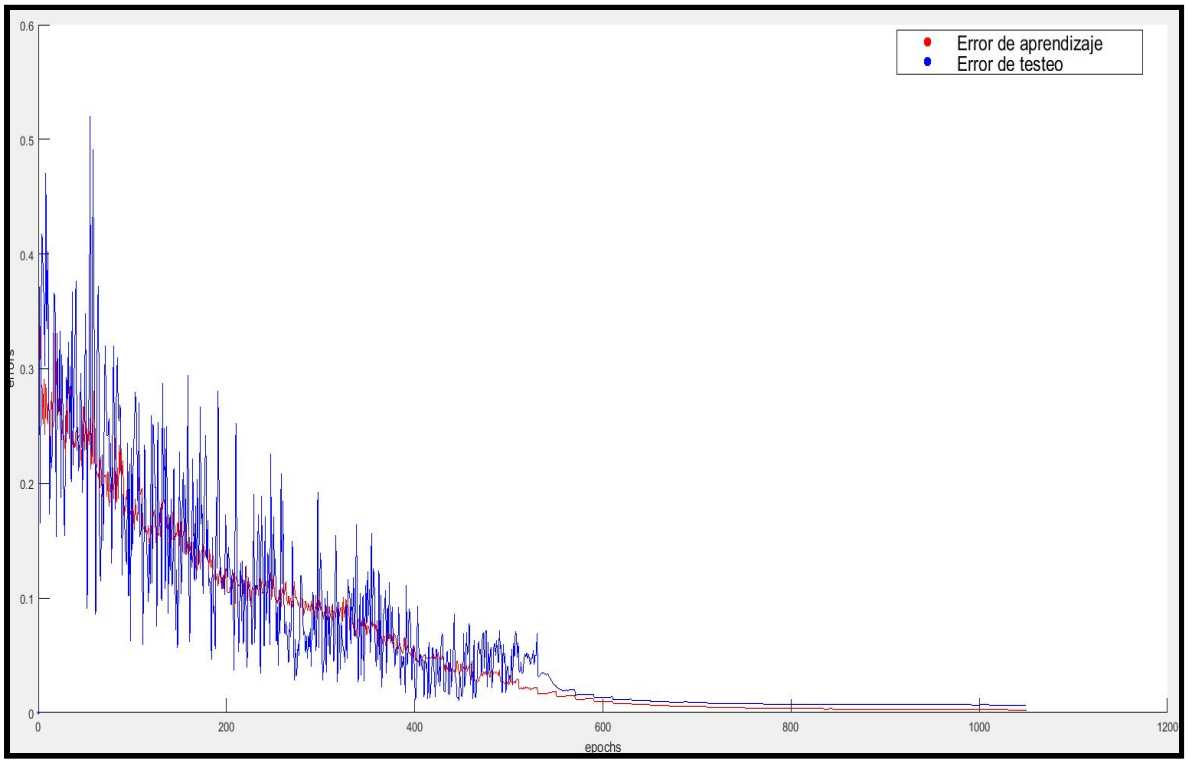


- Se puede observar como el shuffle incide en la convergencia y en la oscilación del error (la amplitud de sus valores)

Eta adaptativo activado y desactivado

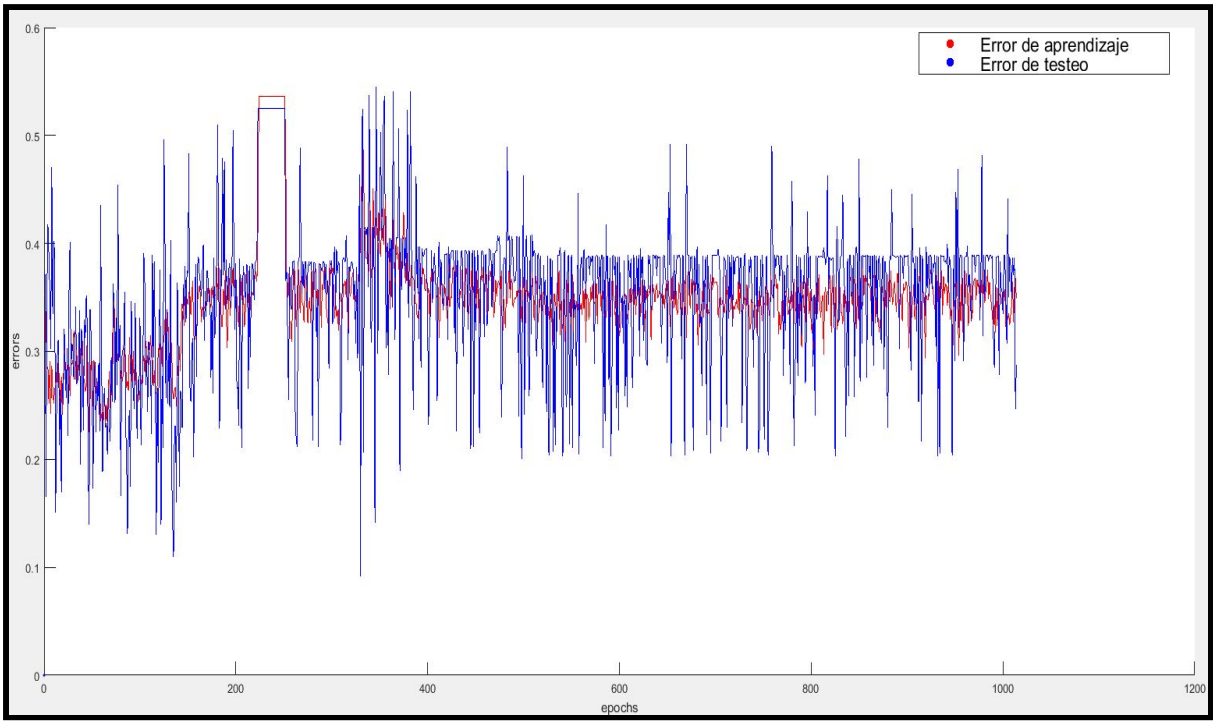
Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.5, shuffle_flag 1, alpha_momentum 0.5, seed “D”, tanh 1

Primeras 1000 épocas



Red 2,15,10,1 con: training_size 300, eta 0.5, shuffle_flag 1, alpha_momentum 0.5, seed “D”, tanh 1 (eta adaptativo desactivado)

Primeras 1000 épocas

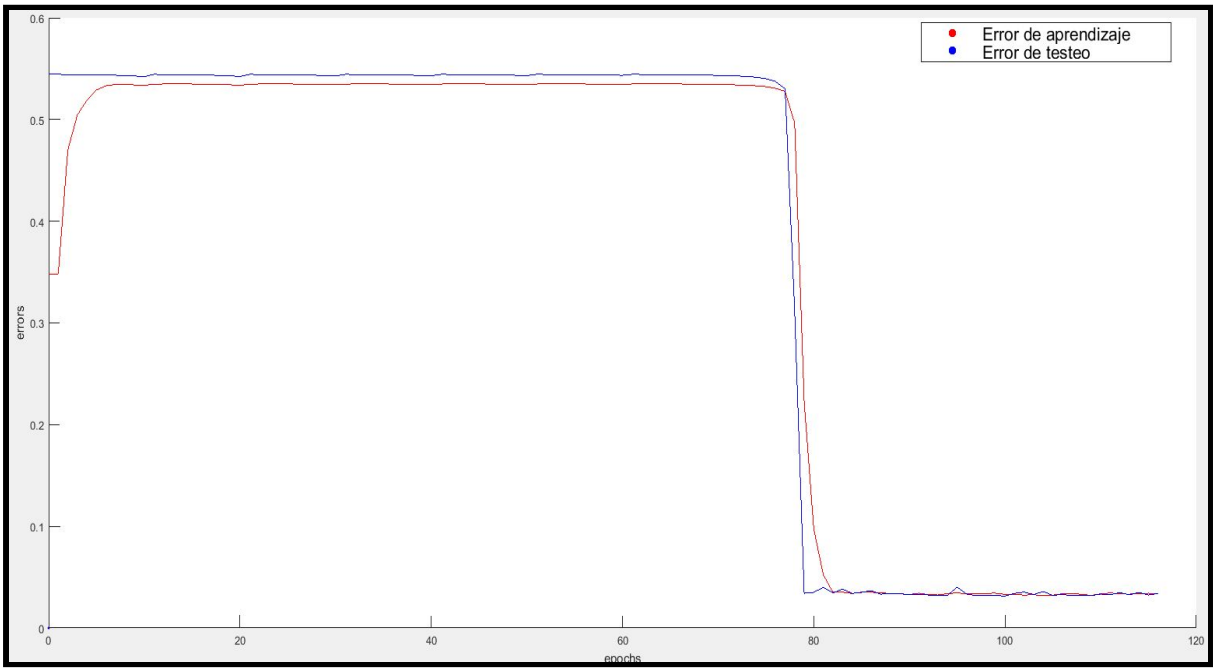


- En este caso al estar eta adaptativo desactivado la red encuentra un mínimo local que no es óptimo, está muy por debajo del error esperado.

Exponencial vs tangencial hiperbólica

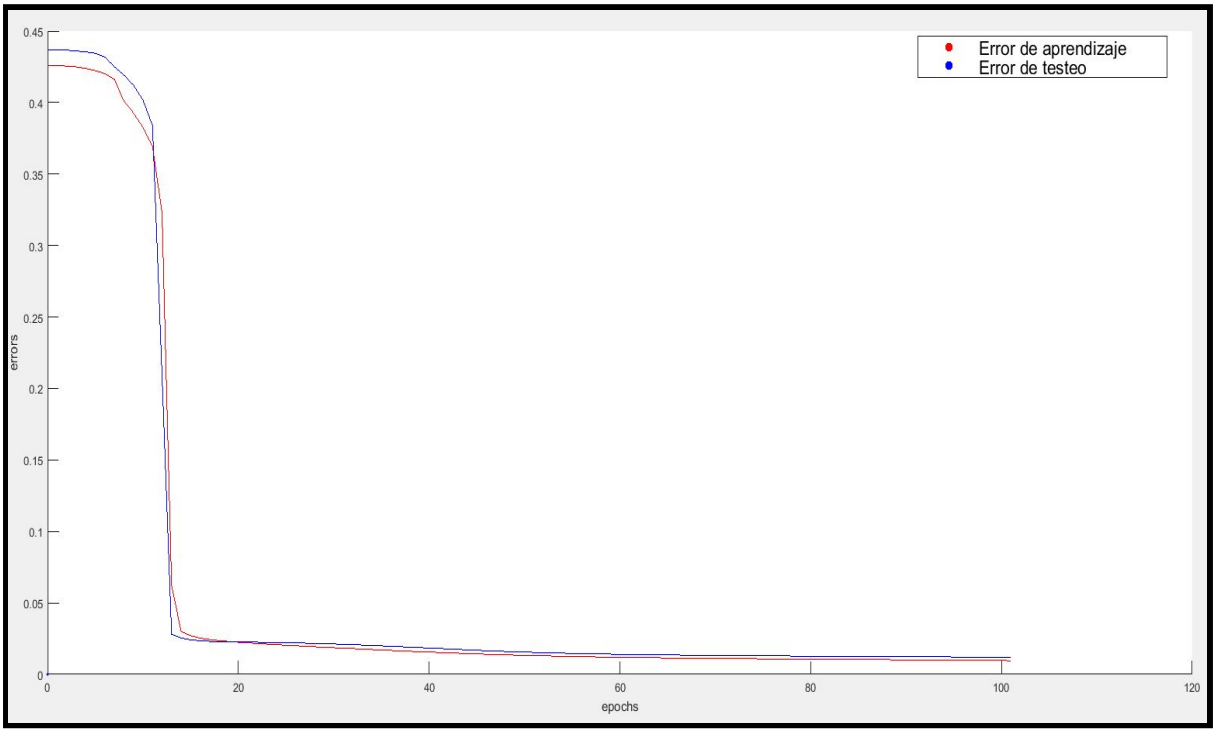
Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 1, alpha_momentum 0.5, seed “D”, tanh 0 (exponencial)

Primeras 100 épocas



Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 1, alpha_momentum 0.5, seed “D”, tanh 1

Primeras 100 épocas

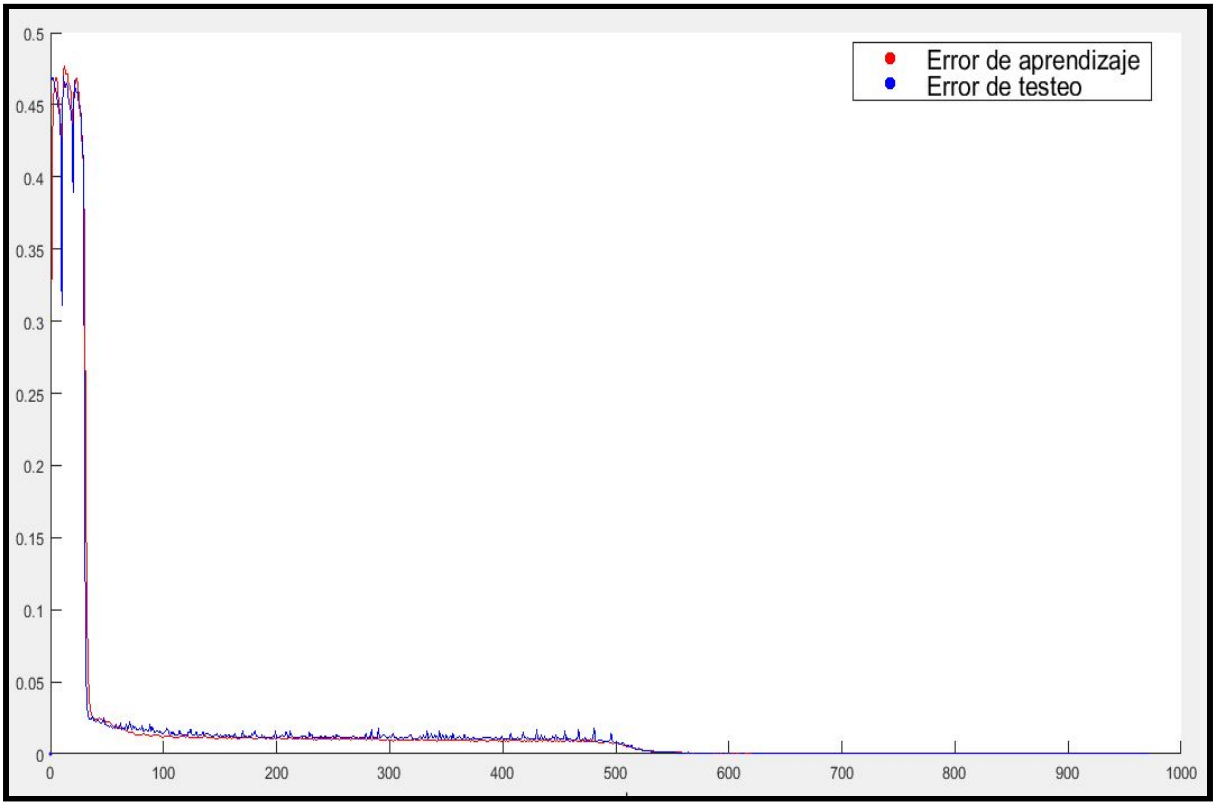


- La red converge a una mayor velocidad con la tangencial

Capa de salida tangencial y lineal

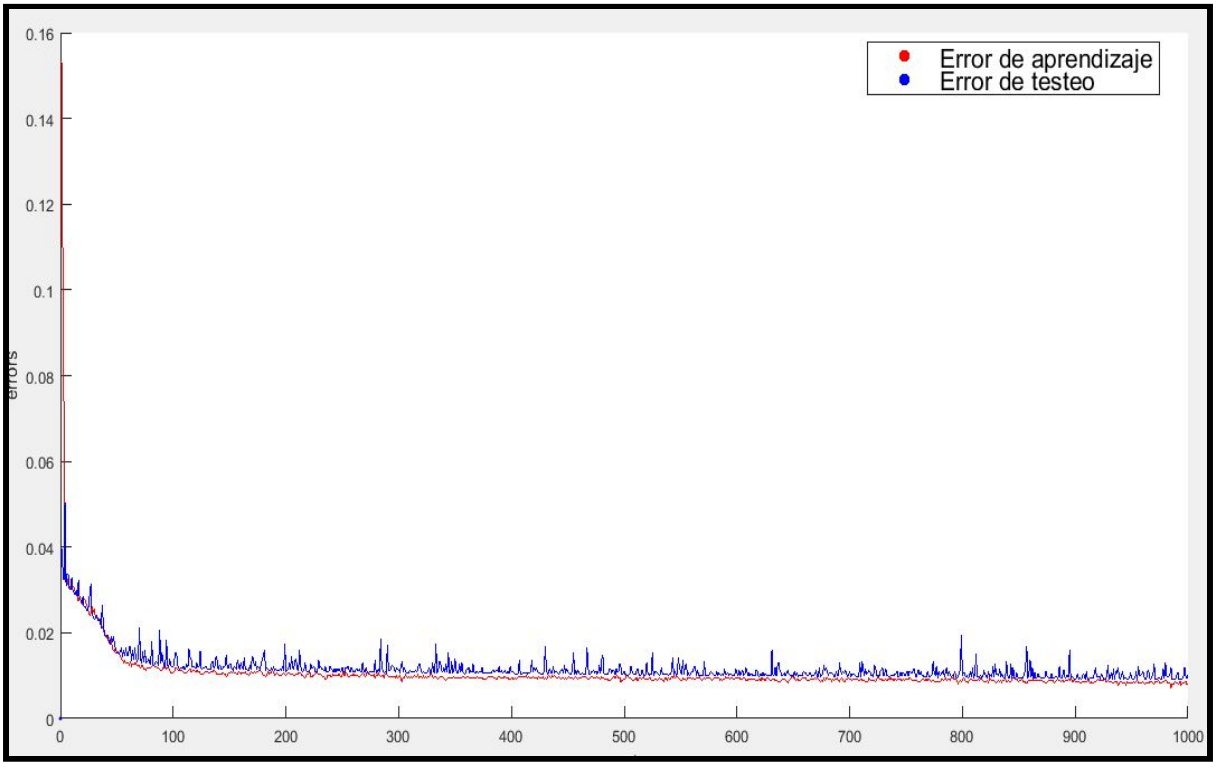
Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 1, alpha_momentum 0.5, seed “H”, tanh en capas intermedias con salida tangencial hiperbólica

Primeras 1000 épocas



Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 1, alpha_momentum 0.5, seed “H”, tanh en capas intermedias con salida lineal

Primeras 1000 épocas

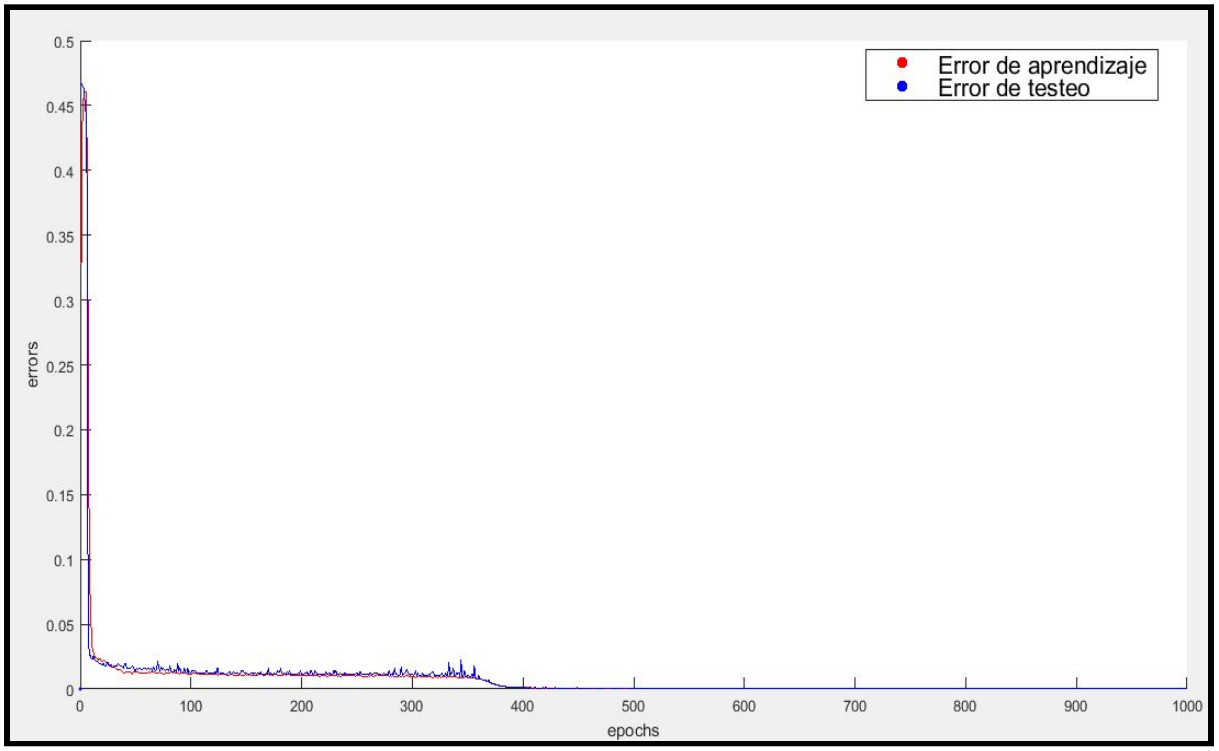


- Si bien la lineal converge a una mayor velocidad al principio, la misma se “estanca” al corto plazo en un error que está por debajo de lo esperado.

Momentum activado y desactivado

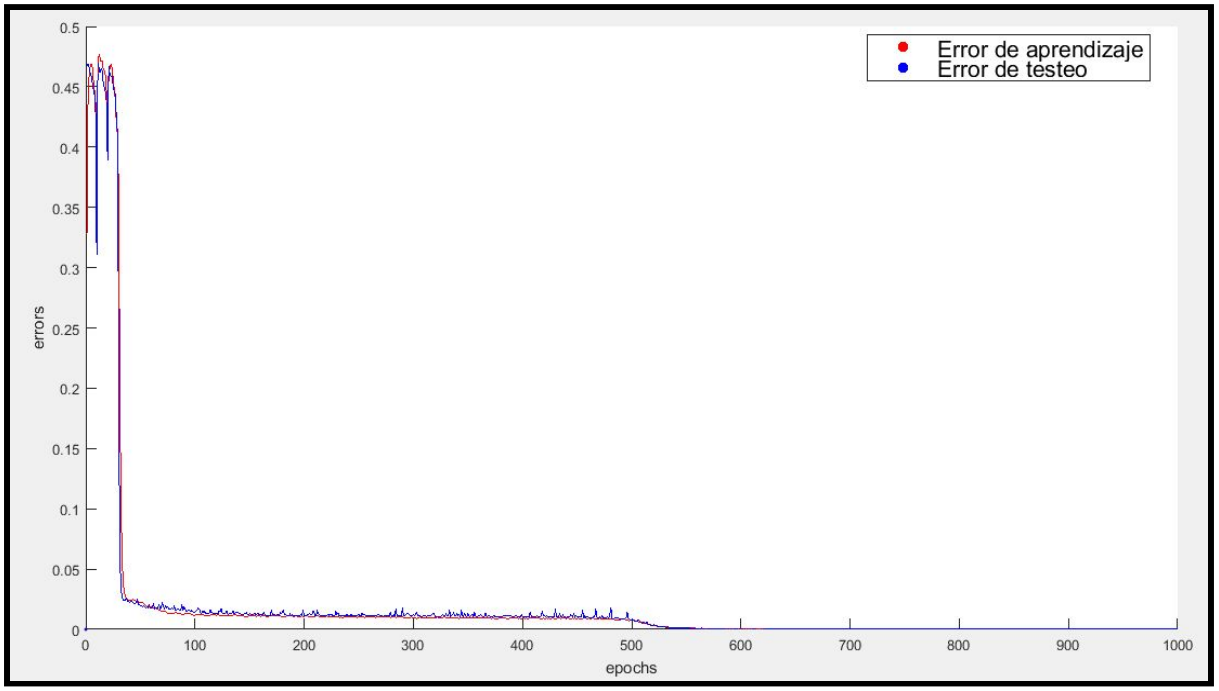
Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 1, alpha_momentum 0.5, seed “D”, tanh 1

Primeras 1000 épocas



Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 1, seed “D”, tanh 1, sin momentum

Primeras 1000 épocas

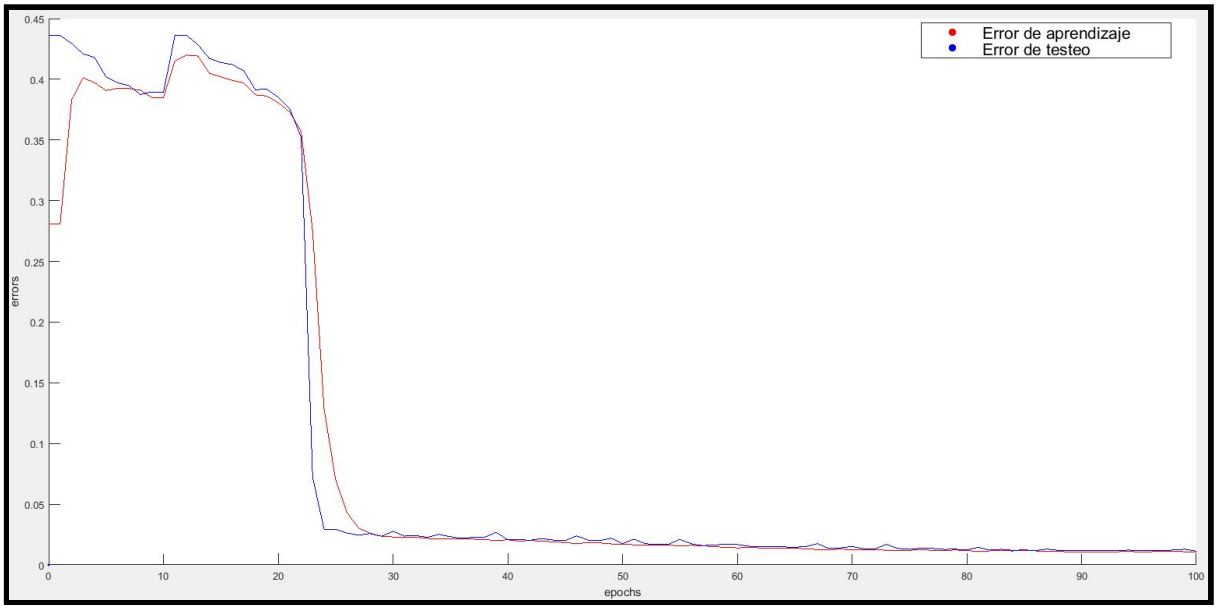


- Se puede observar como momentum influye en la rapidez de convergencia tanto inicialmente como en el corto plazo

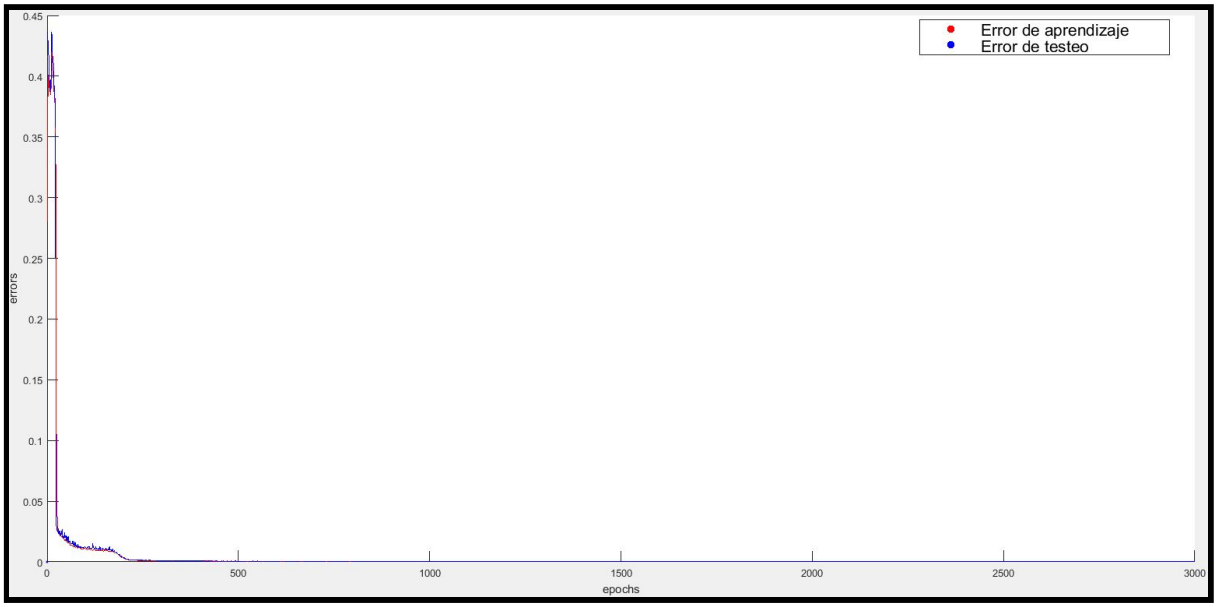
Red neuronal elegida

Red 2,15,10,1 con: eta_check_steps 10, eta_increase_value 0.001, eta_decrease_factor 0.95, training_size 300, eta 0.01, shuffle_flag 1, alpha_momentum 0.5, seed “D”, tanh 1

Primeras 100 épocas

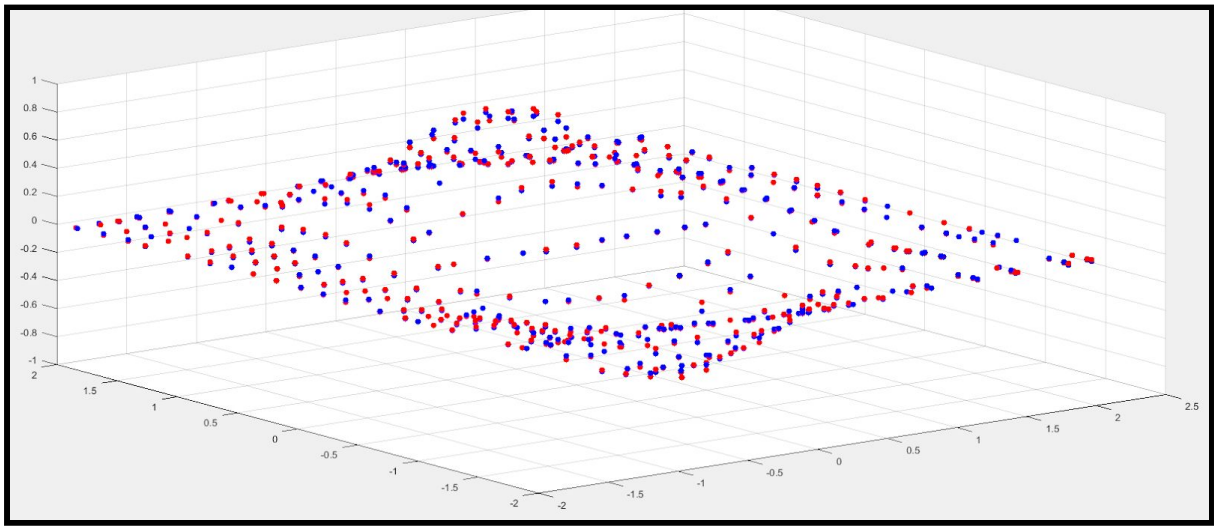


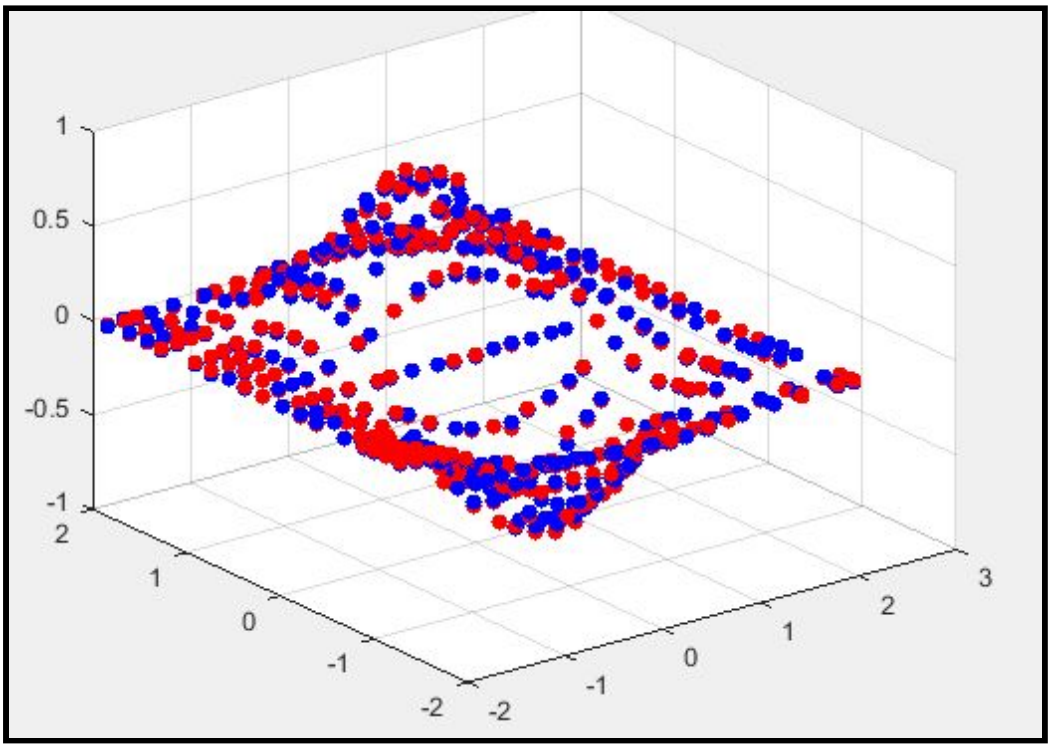
Al finalizar 3000 épocas



Terreno generado

Nota: los puntos rojos representan la solución exacta y los azules la aproximada (la salida de nuestra red).





Conclusiones

A partir de las observaciones realizadas previamente y de las optimizaciones propuestas se pueden deducir las siguientes conclusiones:

- No se puede garantizar que la red elegida genere la mejor solución aproximada al problema, ya que el espacio de prueba es mucho menor al espacio de redes posibles.

- La forma en la que un sistema busca la solución (como recorre la superficie) se ve afectada por la semilla en cuestión. Esto se debe a que los pesos iniciales son pseudoaleatorios, entonces cada vez que se pone a prueba la arquitectura se recorre la superficie comenzando por un punto de origen distinto.

- El shuffle demostró influir positivamente en la convergencia en largo plazo en una mayor medida que en la de corto plazo. Al mismo tiempo esta técnica es útil para salir de mínimos locales no óptimos que no sean muy profundos.

- Tanto momentum como eta adaptativo logran una mayor rapidez de convergencia que es más notable en el corto plazo que en el largo plazo y nos permite evadir mínimos locales profundos que no sean óptimos.

- La fijación de los parámetros correspondientes a las mejoras implementadas depende de cada arquitectura de red y es completamente empírica.

- Como la comparación entre distintas arquitecturas depende de la semilla elegida para cada una, no se puede asegurar que una arquitectura siempre va a ser mejor que otra (considerando que ambas convergen a un valor razonable).

- Si bien un sistema puede aprender mejor que otro (al tener un error cuadrático medio de aprendizaje menor), esto no implica que el mismo tenga una mayor capacidad de generalización.

- Para nuestro problema la tangente hiperbólica cuenta con una convergencia mucho más rápida y precisa que la exponencial como función de activación.

- Momentum influye considerablemente en la rapidez de convergencia de un sistema en corto plazo.

- El epsilon del margen de error de aceptación impacta notoriamente en el porcentaje de aciertos.

- El tamaño de los patrones de entrada para el entrenamiento incide directamente con la capacidad de generalización de la red.

- Cuanto mayor es el eta inicial, más diverge la red. Sin embargo, gracias a la implementación de eta adaptativo, es posible lograr la convergencia de la misma.

- Resultó de suma importancia la correcta representación de la capa de salida de la red. A partir del análisis de la salida del terreno se pudo apreciar que el mismo se encontraba comprendido entre -1 y 1, por lo que se aplicó una función de activación tangencial. Esto logró convergencia en casos donde no se cumplía con una capa de salida lineal.

- Dependiendo del problema particular que se esté intentando resolver, una arquitectura determinada con ciertos parámetros puede resultar mejor que otra. A la hora de elegir una red, el *trade-off* siempre va a existir; es trabajo del diseñador poder determinar cuales son los requisitos y en qué orden de

prioridad se encuentran para poder determinar el mejor resultado. Por ejemplo, la solución puede necesitar ser rápida y no tan precisa o viceversa.