

Trabajo Práctico N°2

Redes Neuronales

**Materia:** Sistemas de Inteligencia Artificial

**Grupo:** 3

**Integrantes:** Biagini, Martín

Clozza, Nicolás

Filipic, Joaquín

Mamone, Federico

**Fecha:** 08/05/2019

Índice

* Introducción 2
* Descripción del Trabajo 3
* Análisis de Resultados 6
* Conclusiones 10
* Anexo 11

Introducción

El objetivo de este proyecto es implementar una red neuronal multicapa (*Perceptrón*) que aproxime la altura del terreno dada la longitud y latitud del mismo. Para hacer esto es necesario entrenar el *Perceptrón* con patrones conocidos (brindados por la cátedra) de latitud, longitud y altura, para que luego el mismo pueda realizar la estimación solicitada.

El algoritmo utilizado para llevarlo a cabo es el de *backpropagation*, distinguiéndose el paso forward, el cálculo de los deltas y el paso backward para actualizar los pesos de las conexiones.

La efectividad de la red estará relacionada con la diferencia comprendida entre la altitud calculada y la presente en el patrón.

Descripción del Trabajo

Inicialización

Para la realización del trabajo se implementó una red neuronal multicapa llamada *Perceptron*, la cual posee una arquitectura variable, es decir que puede ser inicializada con distintos parámetros (capas totales, neuronas por capa, función de activación, etc.). Los mismos deben estar presentes en un archivo de configuración externo (dentro de la carpeta *configuration*, con prefijo *config\_* y con extensión *.txt*) e indicarle el nombre del mismo al momento de instanciar la clase *Perceptron* (ver Imagen 1 en anexo).

Toda arquitectura está afectada por una unidad neuronal con el potencial de salida en -1 en cada capa oculta y en la inicial. Los pesos de las conexiones entre las neuronas de capas consecutivas se inicializan con valores al azar comprendidos entre -0,5 y 0,5.

Para la obtención de los patrones, se procedió a normalizar los datos para trabajar correctamente en relación a los dominios de las funciones de activación.

Se evalúa el error de la salida de la red como el error cuadrático medio (ECM) con respecto a los valores de salida del archivo.

Entrenamiento

El entrenamiento de la red se hace cada una cantidad elegida de épocas, y finaliza al llegar a esa cantidad o hasta alcanzar el criterio de corte (si el ECM luego de una época da menor a este valor, se considera que la red está bien entrenada y corta). Para el mismo, se pueden distinguir dos maneras, siempre utilizando los datos provistos por la cátedra:

1. Tomando un porcentaje de los datos:

De esta forma, vamos a entrenar a la red con un parcial de los patrones brindados. Esto nos va a poder permitir utilizar los datos no usados para el testeo de la misma, pudiendo así obtener estadísticas más precisas.

1. Tomando todos los datos (indicando el 100% de los datos):

Con esta configuración la red va a utilizar la totalidad del archivo de entrenamiento para obtener su aprendizaje. La ventaja de realizarlo de esta manera es que la red posee una mayor cantidad de patrones con los cuales puede aprender, pero se pierde la posibilidad de saber la precisión de la misma (ya que no se poseen datos no vistos por la red).

Además se puede optar por entrenar la red mediante los métodos incremental y batch:

* Incremental:

Al utilizar este modo se selecciona un patrón de forma azarosa, el cual es aplicado en la red y propagado hacia la salida de la misma. Luego se compara la salida obtenida con la esperada, y se corrigen los pesos de las conexiones entre neuronas de forma inversa, es decir, desde la salida hacia la entrada. Se prosigue de esta manera para cada patrón que se quiera utilizar para entrenar la red.

Un beneficio de esta implementación es que se suelen evitar mínimos locales en la función de error, sin embargo se tarda más en ser entrenada la red.

* Batch:

La diferencia en este caso es que se van seleccionando de a cierta cantidad de patrones, los cuales son aplicados a la red, y la misma no corrige sus pesos hasta que no fueron ingresados todos los patrones pertenecientes a esa selección.

Evaluación

Así como el aprendizaje puede llevarse a cabo de dos maneras, se puede hacer la prueba de la red de forma análoga:

1. Tomando todos los datos

Se utiliza la totalidad del archivo con el que se entrenó la red.

1. Tomando un porcentaje restante de los datos

Si se optó por entrenar a la red con un porcentaje de los datos, entonces el restante puede ser utilizado para testear la misma.

Lo que se hace es simplemente realizar el paso forward con los patrones y obtener los potenciales de salida para cada uno.

Mejoras

La finalidad al implementar una red neuronal es encontrar el mínimo total en la función de error de la salida, para esto se hace uso del gradiente descendente de esta función en un punto particular. Encontrar o acercarse a este mínimo total puede resultar muy complicado, con lo que el algoritmo tomaría mucho tiempo en entrenarse.

Para mitigar este problema se optó por implementar dos posibles soluciones:

* Momentum

Este algoritmo se basa en tener cierto tipo de memoria sobre los cambios en el vector de pesos que se realizó en la iteración anterior. Es muy útil a la hora de querer evitar el problema de los mínimos locales de la función de error, evitando que la red se estanque en uno de ellos sin encontrar el mínimo total.

* ETA adaptativo

Otro gran problema que surge viene dado por la oscilación de la convergencia al mínimo total alrededor de este punto. Es decir que se producen saltos abruptos, tales que no permitan llegar al valor deseado. Para solucionar este problema se hace uso del ETA adaptativo, el cual analiza el error generado en los últimos n pasos y verificar si este disminuyó o no. Consiste en agregar un parámetro que se va modificando a medida que transcurren las corridas de la red en la etapa de aprendizaje, comenzando por un valor grande provocando que se aproxime de forma más rápida al mínimo total y luego disminuyéndolo, logrando así la convergencia deseada.

Además de evitar la oscilación en la convergencia también busca evadir los mínimos locales profundos, los cuales claramente no son deseados.

Métricas

La implementación fue hecha de forma tal que se pueda entrenar, evaluar y obtener métricas interactivamente, cuando sea requerido. Se proveen funciones para ver las métricas relacionadas al entrenamiento (cantidad de épocas corridas, valor del ECM de la última eṕoca, etc.) y a la evaluación (precisión, ECM).

Cabe hacer dos menciones: la primera es que se guarda registro del ECM a medida que avanzan las épocas, para tener un panorama de la forma en la que fue entrenándose la red, y registro del valor del Eta, relevante en casos donde se use la mejora de Eta adaptativo. La segunda es lo referente a la determinación de la precisión. La precisión del entrenamiento se realiza de manera porcentual tomando la última corrida de los patrones. Para cada uno, se verifica si el valor obtenido y el valor esperado difieren en menos de un valor *epsilon*. Análogamente, se calcula la precisión de la evaluación (tanto con la completitud de los datos como con el porcentaje que no fue utilizado para el entrenamiento).

Análisis de Resultados

Se realizaron diversas pruebas variando los parámetros a disposición, y se fueron colocando los resultados en el archivo adjunto *resultados.xlsx*. Esencialmente, la idea fue probar muchas variantes e ir comparando la respuesta de la red, hasta determinar alguna arquitectura mediante la cual se obtenga un terreno lo suficientemente aproximado al original.

Cabe destacar que se exponen los gráficos que sean lo más representativos posibles en las comparaciones que se estén efectuando, ya que hay casos en que, según la disposición de la arquitectura y los parámetros elegidos, no es visible la diferencia entre elegir un valor u otro para cierta variable.

A continuación, la lista de los parámetros con los que se realizaron las pruebas:

* Arquitectura de la red: cantidad de capas de la red y cantidad de neuronas de cada una (Ej: 2,15,10,1 es una arquitectura con 2 neuronas de entrada, 15 en la primera capa oculta, 10 en la segunda y 1 neurona de salida).
* Porcentaje del total de patrones a tomar en el entrenamiento.
* Modo de entrenamiento (Incremental o Batch)
* Criterio de corte. Para todas las pruebas se determinó el valor de 0,001.
* Epsilon: Valor por debajo del cual se considera que la diferencia entre el valor obtenido de salida de la red de un patrón y el valor esperado es aceptable. Para todas las pruebas se determinó el valor de 0,05, y de aquí se obtiene la precisión.
* Función de activación (tanh o exp).
* Eta inicial: factor de aprendizaje.
* Alpha (del momentum).
* K (del eta adaptativo): cada cuántas épocas se revisa el eta.
* a (del eta adaptativo): constante para el incremento.
* b (del eta adaptativo): constante para el decremento.

Y la lista de métricas obtenidas:

* Cantidad de épocas corridas: Ya sea el total indicado o las alcanzadas por el criterio de corte.
* Eta final (relevante en los casos de prueba con eta adaptativo)
* Precisión entrenamiento.
* EMC final del entrenamiento.
* Precisión de testeo con el 100% de la data.
* ECM de testeo con el 100% de la data.
* Precisión de testeo con el porcentaje de la data no usado en el entrenamiento.
* ECM de testeo con el porcentaje de la data no usado en el entrenamiento.

Comparación entre las funciones de activación *tanh* y *exp*

Como arquitectura representativa para esta comparación se tomó la siguiente (líneas 19 y 21 del archivo adjunto de resultados):

* Neuronas por capas: 2,20,15,1
* Porcentaje entrenamiento: 70%
* Eta inicial: 0,05
* Alpha: 0,7
* K: 10
* a: 0,1
* b: 0,5

Y los resultados de la evolución del error se ven en los gráficos 1 y 2 del anexo. Se evidencia claramente que el error se reduce más rápidamente utilizando la tangencial, sumado a que se llega al criterio de corte, cosa que no sucede con la función exponencial.

Comparación entre métodos de entrenamiento batch e incremental

Como arquitectura representativa para esta comparación se tomó la siguiente (líneas 14, 15 y 16 del archivo adjunto de resultados):

* Neuronas por capas: 2,4,3,2,1
* Porcentaje entrenamiento: 70%
* Eta inicial: 0,1
* Alpha: 0,5
* K: 10
* a: 0,01
* b: 0,1

Y los resultados de la evolución del error se ven en los gráficos 3, 4 y 5 del anexo. La gran deficiencia del modo batch es el estancamiento en un mínimo local de la solución; una vez que llega a esa situación, no puede mejorar. Otra cosa que vale la pena mencionar es el hecho de que cuesta más, para una arquitectura dada, encontrar valores de los parámetros para que el batch encuentre al menos una solución parcial. En otros casos de prueba (por ejemplo, para esta arquitectura y usando la función de activación tangencial), se estanca en valores del ECM tan elevados (y la precisión baja cerca del 0%) que no es relevante su análisis.

El modo incremental hace posible no estancarse y seguir buscando otro mínimo. Si bien en este caso con la función exponencial se ve una prolongada variación, esto se corrige con la tangencial, como ya se vio también en el inciso anterior.

Comparación entre el Momentum activado y desactivado

Como arquitectura representativa para esta comparación se tomó la siguiente (líneas 37 y 38 del archivo adjunto de resultados):

* Neuronas por capas: 2,30,1
* Porcentaje entrenamiento: 70%
* Eta inicial: 0,1
* Alpha: 0,5
* Función de activación: tanh

Y los resultados de la evolución del error se ven en el gráfico 6 del anexo. Si bien tarda un poco más en estabilizarse, se nota que el error disminuye a medida que pasan las épocas con el momentum activado.

Comparación entre el Eta Adaptativo activado y desactivado

Como arquitectura representativa para esta comparación se tomó la siguiente (líneas 39 y 40 del archivo adjunto de resultados):

* Neuronas por capas: 2,30,1
* Porcentaje entrenamiento: 70%
* Eta inicial: 0,001
* K: 10
* a: 0,005
* b: 0,01
* Función de activación: tanh

Los resultados de la evolución del error se ven en el gráfico 7 del anexo, y el de la evolución del Eta (para el caso en que está activado) en el gráfico 8 del anexo. Se puede observar, en primera instancia, que el uso del eta adaptativo logra evitar un posible mínimo local y sigue reduciendo el ECM a medida que pasan las épocas. Esto es gracias al cambio en el valor de eta a medida que sea requerido, como se muestra en el segundo de los gráficos mencionados.

Elección de la mejor red

Teniendo en cuenta los apartados anteriores, indudablemente la elección de una buena red tiene que empezar por el uso de la función de activación tangencial y el modo de aprendizaje incremental. Luego se empiezan a probar distintas arquitecturas y valores de los parámetros de las mejoras vistas. Un buen ejemplo (pues dió una precisión de entrenamiento del 94.1748%) obtenido fue el siguiente:

* Neuronas por capas: 2,4,3,2,1
* Modo incremental
* Porcentaje entrenamiento: 70%
* Eta inicial: 0,1
* K: 10
* a: 0,01
* b: 0,1
* Función de activación: tanh

En el gráfico 9 se observa la evolución del ECM a lo largo de las épocas, y en el gráfico 10, la evolución del eta. A continuación, se detallan las métricas obtenidas para una corrida de esta arquitectura (recordando que inicialmente los pesos de las conexiones se inicializan al azar):

Entrenamiento:

* Épocas corridas: 300
* Condición de corte: Sí
* ECM (última corrida): 0.000870694
* Precisión (última corrida): 94.1748%
* Eta final: 0.106946

Testeo con el 100% de los datos:

* ECM: 0.00146589
* Precisión: 83.6735%

Testeo con el 30% restante de los datos:

* ECM: 0.00158352
* Precisión: 81.203%

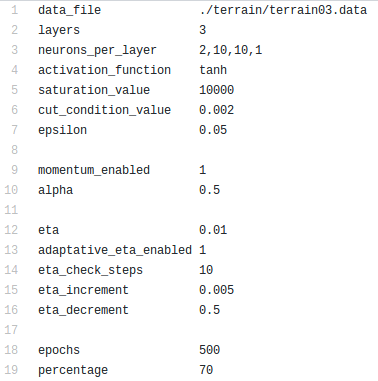
Finalmente, se pueden comparar el gráfico en 3D del terreno esperado y el generado a partir del testeo del 100% de los datos una vez entrenada la red (Gráficos 11 y 12, respectivamente).

Conclusiones

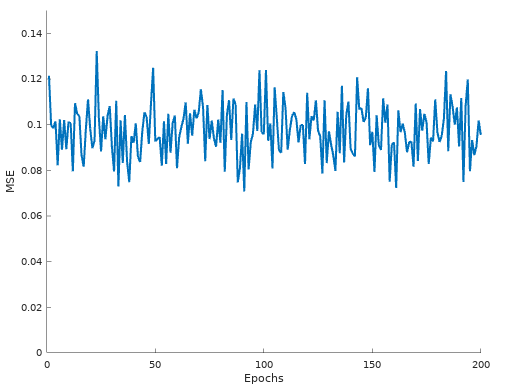
A partir de las observaciones de los resultados, se pueden enumerar las siguientes conclusiones:

* No hay certeza en la elección de la mejor red, depende de muchos parámetros, de la arquitectura y de los pesos iniciales de las conexiones.
* El eta está muy ligado a la arquitectura elegida.
* Las mejoras de momentum y eta adaptativo responden en general con reducciones del ECM a lo largo de las épocas, y evitan estancamientos en mínimos locales.
* La función de activación tangencial es significativamente mejor que la exponencial (converge más rápido y es más precisa).
* El modo incremental evita los mínimos locales que el batch no.
* La capacidad de generalización está muy relacionada con la precisión del entrenamiento y de la cantidad de patrones destinados a entrenamiento y a testeo (aunque se necesita un set de testeo mayor para comprobarlo).
* La determinación de parámetros y arquitectura se enfoca en un problema en particular, cada red neuronal puede responder diferente frente a distintos problemas.

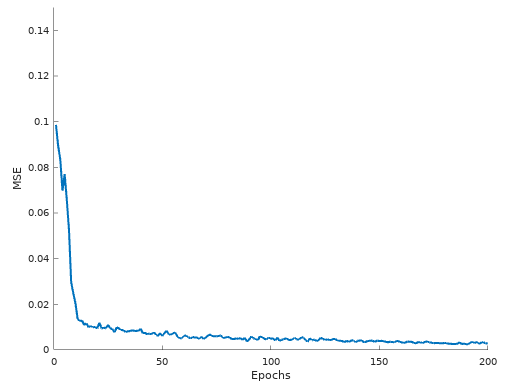
Anexo



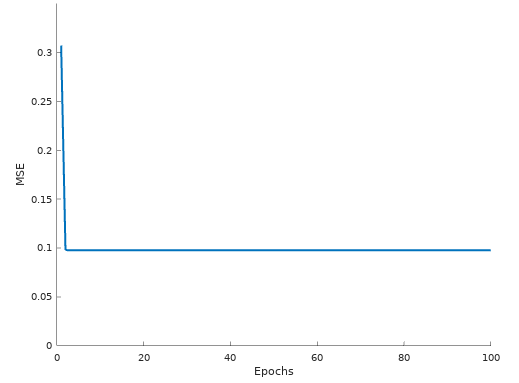
**Imagen 1:** Archivo de ejemplo de configuración.



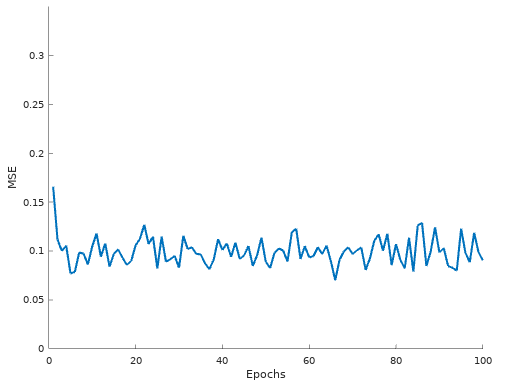
**Gráfico 1:** Variación del ECM con función exponencial y arquitectura correspondiente a línea 21.



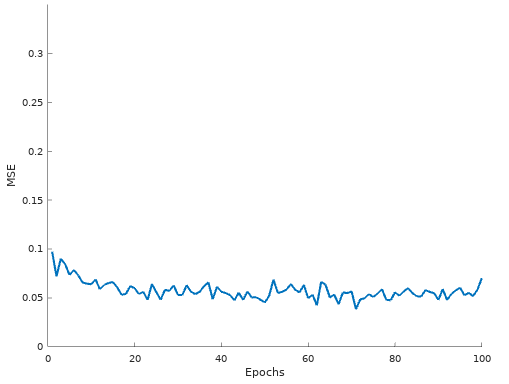
**Gráfico 2:** Variación del ECM con función tangencial y arquitectura correspondiente a línea 19.

**

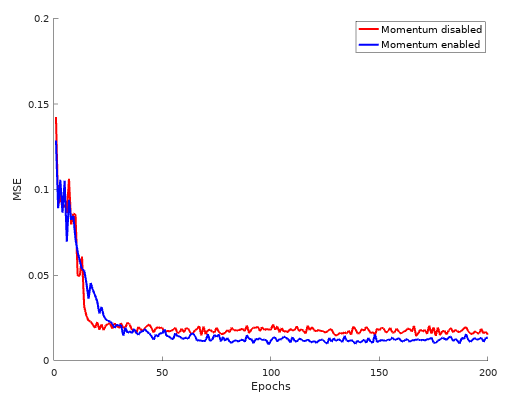
**Gráfico 3:** Variación del ECM en modo batch con función exponencial y arquitectura correspondiente a línea 14.



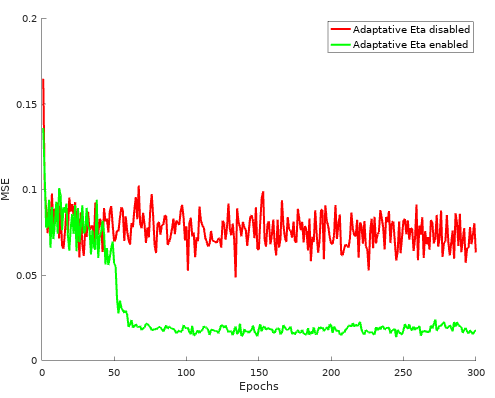
**Gráfico 4:** Variación del ECM en modo incremental con función exponencial y arquitectura correspondiente a línea 15.



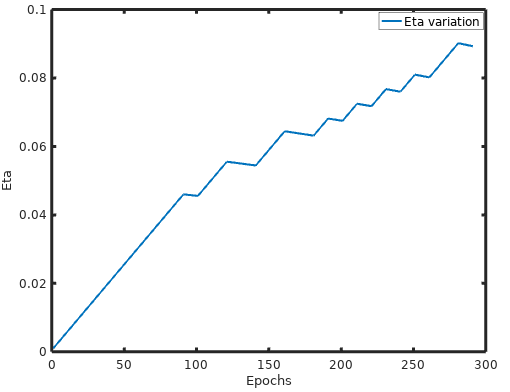
**Gráfico 5:** Variación del ECM en modo incremental con función tangencial y arquitectura correspondiente a línea 16.



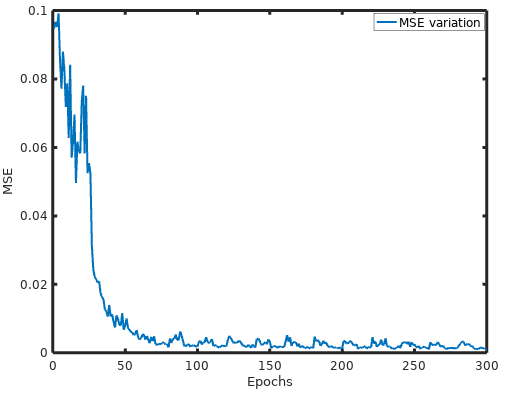
**Gráfico 6:** Comparación entre la variación de errores entre redes con y sin momentum.



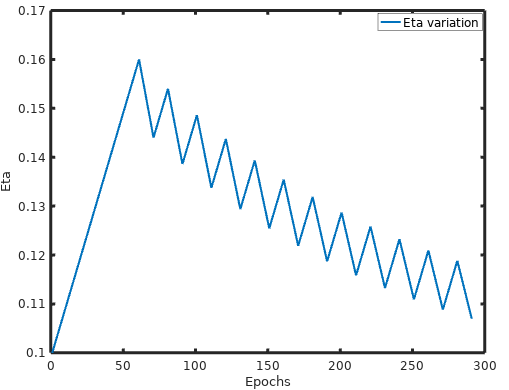
**Gráfico 7:** Comparación entre la variación de errores entre redes con y sin eta adaptativo.



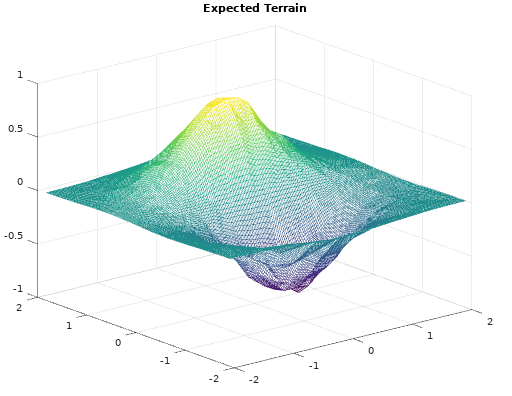
**Gráfico 8:** Evolución del valor del eta para el mismo caso de comparación del gráfico 7.



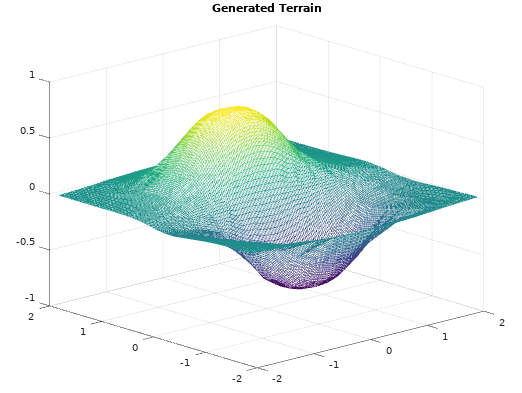
**Gráfico 9:** Evolución del ECM para la arquitectura elegida.



**Gráfico 10:** Evolución del Eta para la arquitectura elegida.



**Gráfico 11:** Terreno esperado.



**Gráfico 12:** Terreno generado a partir de la red neuronal con la arquitectura elegida.