Trabajo Práctico Especial
Sistemas de Inteligencia Artificial

Segundo Fariña - 56176
Martin Victory - 56086
Sebastian Favaron - 57044
Ximena Zuberbuhler - 57287

Introducción

El objetivo del trabajo práctico es desarrollar una red neuronal multicapa para aproximar lo mejor posible un terreno en tres dimensiones dado un set limitado de coordenadas iniciales. La red recibe una coordenada de dos dimensiones (pasando dos números) y responde con un tercer número que representa la altura de la coordenada anterior. La misma para aprender utiliza el algoritmo back propagation el cual fue desarrollado en clase.

Descripción del trabajo

Para cumplir el objetivo, se diseñó una red genérica que aceptara distintas arquitecturas y ciertos parámetros. Entre estos parámetros podemos destacar:

- Eta
- Finalizar en un cierto valor de error
- Función de activación
- Opción de utilizar eta adaptativo
- Opción de utilizar momentum

Se estudiaron distintas configuraciones de la red para intentar encontrar la más performante y que realice también la mejor aproximación. Para poder resolver este problema nos encontramos con varias situaciones las cuales debíamos tomar una decisión para llegar a un resultado.

En primer lugar había que definir que arquitecura usar. Es decir, sabiendo que la entrada eran dos parámetros (x,y) y la salida un único parámetro (z), debíamos decidir la cantidad de capas ocultas en la red y la cantidad de nodos en cada una de las capas ocultas. Como el espacio de búsqueda de configuraciones era tan grande, recurrimos a elegir configuraciones iniciales distintas entre si (segun nuestra opinion subjetiva) e intentar optimizarlas, manteniendo siempre fija la arquitectura en cada configuración.

En segundo lugar deberíamos decidir que parametros ibamos a variar y en que orden deberian variar los mismos según la configuración dada.

Como parámetro de performance analizamos en una cantidad fija de épocas (corridas de feed y backpropagation para todos los inputs de entrenamiento) el error final de la red.

Detalles del Algoritmo

Al comenzar el algoritmo de aprendizaje se genera un conjunto de matrices de pesos, con sus dimensiones correspondientes para poder ser luego utilizadas tanto para realizar el paso *feed* como para el de *backpropagation*. Estas matrices son inicializadas con valores aleatorios en el rango de *[-1 ; 1]*. Luego se comienza a iterar para poder entrenar a la red. En cada iteración se toman los pares de valores de entrada y se realiza el paso *feed*, de esta manera se logra obtener los valores de salida estimados. Basándose en el error existente entre la salida estimada y la salida real, se realiza el paso *backpropagation*, en el que las matrices de pesos son actualizadas de manera que minimicen el error. Dado que se eligió implementar la red con el esquema *incremental*, el paso de *backpropagation* es realizado uno a uno con cada una de las entradas. Este proceso se repite numerosas veces, hasta completar una de dos condiciones: se alcanza un valor mínimo de error, o se realizaron una cantidad fija de iteraciones (*épocas*).

Arquitecturas

En general, nos limitamos a arquitecturas con una o dos capas ocultas ya que la aproximación de terreno no nos parecía un problema excesivamente complejo para la red. Sin embargo, como no teníamos pruebas experimentales de esto incluimos una red con tres capas ocultas para (de manera incompleta) confirmar nuestra teoría.

Somos conscientes de que podría existir una arquitectura de tres capas ocultas (o más) con otra cantidad de neuronas que las que elegimos cuya performance sea mejor que todas las que elegimos. Sin embargo, debíamos limitar la exploración en cierto punto.

Elegimos como arquitecturas:

- [2 50 1]
- [2 30 1]
- [2 10 10 1]
- [2 15 30 1]
- [2 10 10 10 1]

Variación de parámetros

Una vez definidas las arquitecturas se pasó a probar las mismas entrenando a la red con distintos parámetros. Como condición de corte se estableció que el error fuera menor a 0.001, pero por un tema de poder realizar varias pruebas en

tiempos razonables tambien se decidio cortar el entrenamiento una vez alcanzadas las 500 épocas.

En primer lugar se partió de la primer arquitecura y se probaron, varios eta distintos para cada funcion de activacion y viendo para que rango de valores la aproximación tenía sentido y para cuales los resultados divergían o iban muy lento. A partir de esto, se probaron las arquitecturas con un poco mas de conocimiento sobre cómo deberían variar los etas según las funciones de activación.

Análisis de resultados

Se comenzó a analizar el problema con la arquitectura [2 50 1]. Como aún no se tenía ningún tipo de parámetro para guiarse, se probaron diversas combinaciones. Inicialmente se decidió fijar la función de activación como la *función sigmoide*, e ir variando los valores de eta hasta encontrar algunos que pudieran resolver relativamente bien el problema. Se buscaba alcanzar un valor bajo de error, o en su defecto el menor posible a lo largo del número de *épocas* fijadas. Se encontró que con valores muy pequeños (del orden 10^{-1}) la red no lograba aprender correctamente. También se observó que con valores grande (del order 10^{1}) tampoco se lograba aprender. Pero con valores intermedios como ser *1, 2 o 5*, si se lograba aprender.

Luego se mantuvo la arquitectura, pero se modifico la funcion de activacion a la *función hiperbólica* para poder contrastar los efectos de estas. Inicialmente se probó nuevamente con valores altos, y tampoco se obtuvieron resultados positivos. Pero esta vez, fue con los valores pequeños (del orden 10^{-2}) donde sí se observó señales de aprendizaje. Pero aun así, a diferencia del caso anterior la red no lograba alcanzar el objetivo de reducir el error a un valor menor de *0,001* en una cantidad de *épocas* menor a *500*.

Obtenida una primera arquitectura con valores aproximados, se decidió pasar a analizar la segunda arquitectura, [2 30 1]. En este caso, nuevamente con valores medios se eta para la *función sigmoide* se logró obtener resultados positivos de aprendizaje, aunque no fueron tan precisos como en el caso anterior. Únicamente se logró cumplir el objetivo con un valor de *eta* igual a *2 o* igual a *3*. Pero es cierto que con valores similares, a pesar de no lograrlos se estuvo muy cerca. Con la *función hiperbólica* nuevamente con valores pequeños fue donde más aprendizaje se observó, pero no se logró alcanzar el nivel de error en la cantidad de épocas, en ninguno de los casos.

Pasando a la tercer arquitectura, [2 10 10 1] se volvió a realizar el mismo procedimiento. Se comenzó haciendo pruebas con la *función sigmoide* y con valores de *eta* medianos. Pero esta vez, se observaron mejoras considerables con respecto a las arquitecturas anteriores, ya que los valores de error esperados se alcanzaron

sin problemas antes de las *500* épocas. También se encontraron mejoras en la *función hiperbólica*, siendo esta la primer arquitectura en la cual se logró alcanzar el objetivo.

En la cuarta arquitectura, [2 15 30 1] se encontraron los mejores resultados de todos. Tanto para la *función sigmoide* como para la *función hiperbólica* se logró superar ampliamente los valores obtenidos anteriormente. Para la primera, se consiguió alcanzar el error en el mejor de los casos, en tan solo *117 épocas*. Mientras que en la segunda, no solo se alcanzó el objetivo, sino que se lo completo en *191 épocas*.

Finalmente se analizó la única arquitectura de con tres capas ocultas, [2 10 10 10 1]. Si bien se obtuvieron buenos resultados, alcanzando el objetivo en diferentes ocasiones, nunca se alcanzaron resultados tan buenos como en la arquitectura anterior. Para la *función sigmoide* se obtuvo en *196 épocas*, pero para la *función hiperbólica* solamente se alcanzó en *496 épocas* un valor muy cercano a las 500 que podría estar totalmente influenciado por el estado inicial.

Por lo tanto, tras analizar cada una de las arquitecturas planteadas variando las distintas combinaciones de parámetros posibles, se decidió que la cuarta arquitectura, que se compone por dos capas ocultas es la mejor para solucionar el problema. Y a su vez, la mejor combinación dentro de esta, se alcanza utilizando la *función sigmoidal*.

Mejoras a Backpropagation

Se analizó la influencia del algoritmo de *backpropagation* en el aprendizaje de las distintas redes. Para esto se implementaron las mejoras de *eta adaptativo*, y *momentum*.

Eta adaptativo

A lo largo de todas las corridas del algoritmo, se observaba como inicialmente valores elevados de eta eran convenientes ya que lograban reducir el error rápidamente, pero en la mayoría de los casos, al mantener estos valores causaba que no se logre la precisión requerida. Este problema se basa en que al tener un valor elevado de eta, los pesos son actualizados fuertemente y se comienza a oscilar superando y reduciendo la solución precisa, sin poder acercarse a esta. Es por esto que un valor de eta el cual se pueda ir adaptando para poder evitar las oscilaciones es altamente efectivo.

Tal como era esperado, se encontraron mejoras al aplicar dicha mejora. En situaciones en las cuales anteriormente no se lograba alcanzar el objetivo. Pero en

las mejores situaciones de la cuarta arquitectura no se pudieron encontrar grandes mejoras. Inicialmente aparentaba que habria mejoras, pero luego no se terminaban de completar. Esto posiblemente se deba a que como el valor inicial de *eta* ya era optimizado, al intentar aplicarle el algoritmo este es alterado y no necesariamente siempre para mejor, por lo que esos pequeñas inconsistencias provocan saltos que por momentos reducían el progreso del aprendizaje de la red.

Momentum

La idea de esta mejora es poder evitar pequeñas oscilaciones en la búsqueda de la solución. En una función de gradiente existen mínimos locales, que pueden ser interpretados por el algoritmo como el mínimo deseado. Es por eso que al agregarle un término extra basándose en los términos anteriores permite tener un efecto de “inercia” que logra escapar de los mínimos locales. De esta manera se obtiene de manera más rápida y precisa el mínimo deseado.

Al aplicar esta mejora al algoritmo, también se lograron mejorar considerablemente el funcionamiento, principalmente usando la *función hiperbólica* que usualmente se quedaba estancada en mínimos locales. Al igual que en el caso anterior, no se observaron mejoras considerables en la cuarta arquitectura, no se la empeoraba, pero tampoco se la mejoraba.

Conclusiones

Se analizaron un conjunto de arquitecturas, y variables de *eta*, así como también mejoras al algoritmo. En primer lugar se observó que la función de activación *sigmoidal* requería un valor de *eta* dentro de un rango entre *1* y *5*. Ya que fuera de este se estanca, y no logra realizar el aprendizaje. Por otro lado, se encontró, que para la función *hiperbólica*, se requieren valores pequeños de *eta*, en el rango de *0,1* a *0,01.*

En cuanto a la función de activación más eficiente, en todos los casos, la función *sigmoidal* fue más efectiva para poder alcanzar nuestro objetivo donde se quería obtener un error menor a *0,001* en menos de *500 épocas*. Es por esto, que podemos concluir que esta función podría ser más conveniente. Esto no quiere decir que la función *hiperbólica* sea incorrecta, ya que pueden existir otras arquitecturas, en las que sea más efectiva.

Con respecto a la arquitectura de la red, lo primero a destacar, es que se noto un desempeño claramente inferior en aquellas que solo tenían una única capa oculta. En nuestro caso, se encontró una arquitectura superior a las otras, donde se pudieron alcanzar mejores resultados. Aun así, las redes con dos o tres capas ocultas, alcanzaron buenos resultados al momento de resolver el problema.

Por último, tanto la optimización de *eta adaptativo* como la de *momentum*, lograron realizar mejoras en diversas arquitecturas, pudiendo reducir el nivel de error. Pero no lograron optimizar la arquitectura elegida para resolver el problema.

Anexo

Épocas	ETA	Error	Función
279	1	0,001	sigmoid
182	2	0,001	sigmoid
178	5	0,001	sigmoid
187	3	0,001	sigmoid
500	0,01	0,0039	tanh
500	0,03	0,0043	tanh
500	0,05	0,0043	tanh
500	0,075	0,0054	tanh

Tabla 1: Red neuronal con una capa oculta de 50 neuronas

Épocas	ETA	Error	Función
500	1	0,0012	sigmoid
311	2	0,001	sigmoid
500	5	0,0013	sigmoid
493	3	0,001	sigmoid
500	0,01	0,0077	tanh
500	0,03	0,0059	tanh
500	0,05	0,0119	tanh
500	0,075	0,0061	tanh

Tabla 2: Red neuronal con una capa oculta de 30 neuronas

Épocas	ETA	Error	Función
297	1	0,001	sigmoid
249	2	0,001	sigmoid
227	5	0,001	sigmoid
445	3	0,001	sigmoid
500	0,01	0,0021	tanh
500	0,03	0,0011	tanh
306	0,05	0,001	tanh
252	0,075	0,001	tanh

Tabla 3: Red neuronal con dos capas ocultas de 10 neuronas cada una

Épocas	ETA	Error	Función
206	1	0,001	sigmoid
231	2	0,001	sigmoid
158	5	0,001	sigmoid
117	3	0,001	sigmoid
500	0,01	0,0016	tanh
309	0,03	0,001	tanh
191	0,05	0,001	tanh
223	0,075	0,001	tanh

Tabla 4: Red neuronal con dos capas ocultas de 15 y 30 neuronas

Épocas	ETA	Error	Función
500	1	0,0601	sigmoid
196	3	0,001	sigmoid
289	5	0,001	sigmoid
500	0,01	0,0011	tanh

Tabla 5: Red neuronal con tres capas ocultas de 10 neuronas cada una

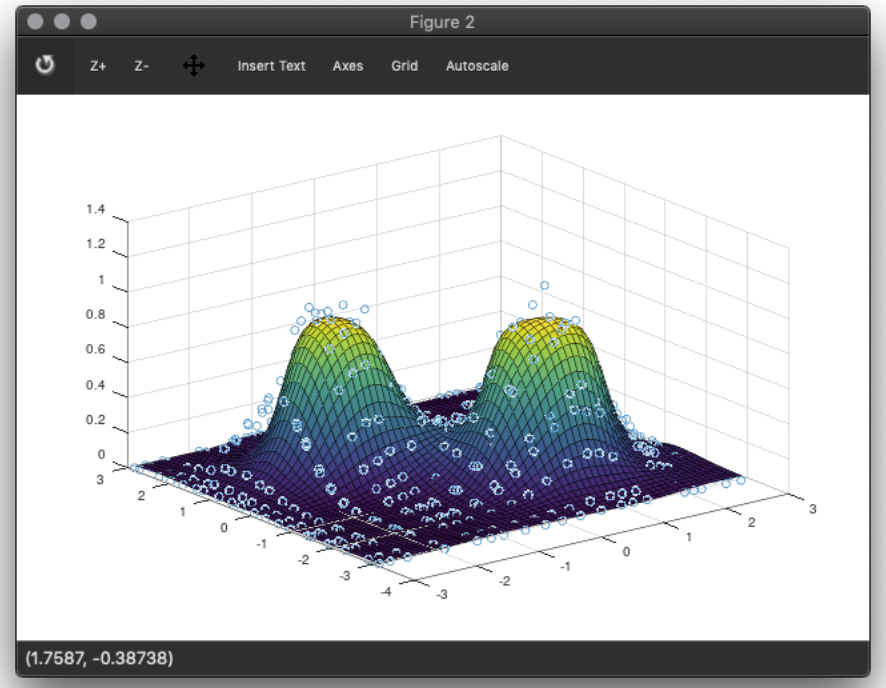

Figura 1: Terreno aproximado con red de 2 capas ocultas (15 y 30) con eta = 3

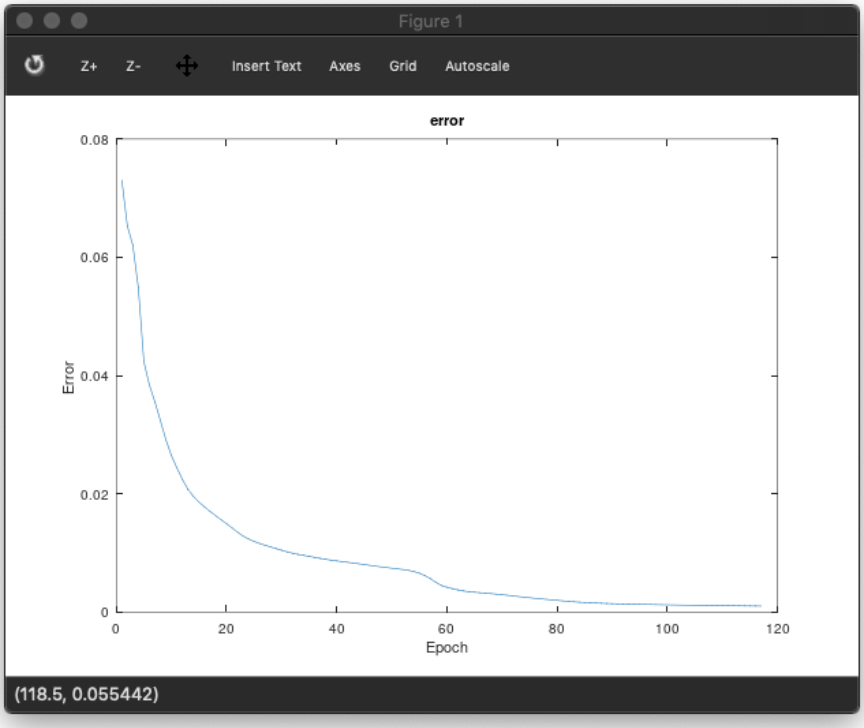

Figura 2: Error por época con red de 2 capas ocultas (15 y 30) con eta = 3