Generalización de Terrenos con Redes Neuronales Multicapa

Sistemas de Inteligencia Artificial
Instituto Tecnológico de Buenos Aires

Garrigó, Mariano 54393
Raies, Tomás A. 56099
Saqués, M. Alejo 56047

Resumen

Se ha analizado el potencial de una red neuronal multicapa supervisada a los efectos de aprender la forma de un terreno, y, posteriormente, generalizar sobre la misma.

Se ha evaluado una serie de variaciones sobre el algoritmo de *backpropagation*, particularmente *adaptive eta* y *momentum*, como así también combinaciones de ambas técnicas. Asimismo, se han comparado los resultados obtenidos actualizando los pesos de la red tanto de manera incremental como en *batch*.

Asimismo, se han entrenado diversas redes con capacidad de generalizar terrenos de diferentes ciudades del mundo, tomando como referencia puntos tomados de mapas de altura de dichas ciudades.

Por último, se han comparado estas técnicas con el estado del arte en algoritmos de generación aleatoria de terrenos, como el *Diamond-Square Algorithm* (DSA). El objetivo ha sido probar las capacidades de una red neuronal de aprender terrenos con una *apariencia* lo más aleatoria posible, tal como busca lograr DSA.

Palabras clave: Redes multicapa supervisadas, *feature scaling*, *backpropagation*, *adaptive eta*, *momentum*, actualización de pesos incremental / en *batch*, funciones de activación, *Diamond-Square Algorithm*.

1. Descripción del entrenamiento

En esta sección, se discutirán las decisiones tomadas por el Equipo para entrenar redes neuronales. Esto incluye tanto toda la instancia de pre-procesamiento, como así también el entrenamiento propiamente dicho.

1.1. Pre-procesamiento

1.1.1. *Feature scaling*

Dado que las imágenes de las funciones de activación son intervalos en $\mathbb{R}^2$ ($tanh : \mathbb{R} \to [-1, 1]$ y $logistic : \mathbb{R} \to [0, 1]$), si los datos del *set* de entrenamiento no se encuentran dentro de dichos intervalos, cierto ajuste es necesario.

Para ello, se utiliza *feature scaling* para estandarizar muestras en algún intervalo deseado. De todas las variaciones de dicho método, se utiliza el *rescaling* de datos:

$$x_i^{'} = \frac{x_i - min(X)}{max(X) - min(X)}(b - a) + a \qquad (1)$$

Donde X es el espacio de muestras a estandarizar, a y b son las cotas inferior y superior del intervalo de llegada, respectivamente.

1.1.2. Selección de muestras de entrenamiento

Dado que una de las características que se quiere probar es la capacidad de generalización de la red, es razonable que parte de las

muestras del terreno se utilicen para verificar que la red, efectivamente, aprendió el problema. Por ende, se ha decidido utilizar el 90 % de las muestras para el entrenamiento, y un 10 % para testeo. Los patrones de cada conjunto se toman de manera no determinística. Se debe notar que la proporción es un dato parametrizable.

1.2. Implementaciones de *backpropagation*

A continuación, se elaborará sobre los diferentes algoritmos utilizados para entrenar las redes neuronales. Las descripciones atacarán fundamentalmente aquellos rasgos distintivos de los mismos, que emanan de decisiones tomadas por el Equipo durante el proceso de desarrollo.

Nota: El error cuadrático medio sobre el conjunto de datos de entrenamiento se calcula, para cada versión del algoritmo, después de ejecutar cada *epoch*.

1.2.1. Algoritmo incremental con *adaptive eta* y *momentum*

Tal como su nombre lo indica, las actualizaciones de pesos en este algoritmo se realizan patrón a patrón. Esto lleva consigo la necesidad de iterar sobre todos los patrones en cada *epoch*, desaprovechándose así las mejoras que realiza *Octave/Matlab* sobre el producto matricial. Esto se verá reflejado en la lentitud de ejecución de cada *epoch* por parte de este algoritmo, relativa a la versión en *batch* que se mencionará a continuación.

En lo que respecta al orden de las muestras de entrenamiento en cada *epoch*, se ha tomado la decisión de presentarlas en un orden no determinístico, dado que se ha observado que, de esta forma, es menos probable que el algoritmo se atasque en mínimos locales.

Por último, se han considerado dos variaciones sobre el proceso de eta adaptativo, las cuales versan sobre si después de k iteraciones en las que el error cuadrático medio descendió consistentemente, dicho contador se restablece tras incrementar el eta o si se prosigue incrementando *epoch* tras *epoch* hasta que haya un incremento en el error. Se ha notado que ambas variaciones presentan comportamientos similares si para la primera versión se establecen valores de k *pequeños* y, para la segunda versión, *grandes*. Para el caso, *pequeños* y *grandes* valores se refieren a cifras en el orden de 10^1 y 10^2 respectivamente. Se ha decidido utilizar la versión que restablece k tras cada incremento.

En lo que respecta a las condiciones de corte, el algoritmo retorna cuando:

1. Se ha llegado al máximo de iteraciones,
2. Se ha llegado al valor del error cuadrático medio deseado,
3. El eta se ha hecho ~ 0.
4. La diferencia entre el error cuadrático medio anterior y el actual es 0.

Nota: El ítem 3. refleja la situación en la que el error ha crecido consistentemente, haciendo que tras sucesivas disminuciones del eta dicho valor se acerque a 0.

1.2.2. Algoritmo en *batch* con *momentum*

En este algoritmo, las actualizaciones de pesos se realizan tras cada *epoch*. A diferencia del caso anterior, esto permite tomar las muestras del set de entrenamiento de manera conjunta, sin tener que iterar por cada patrón. De esta forma, se aprovechan las mejoras que *Octave/Matlab* realizan sobre el producto matricial, haciendo que la ejecución de cada *epoch* sea notoriamente más rápida que en el algoritmo anterior.

En lo que respecta a las condiciones de corte, el algoritmo retorna cuando:

1. Se ha llegado al máximo de iteraciones,
2. Se ha llegado al valor del error cuadrático medio deseado,

3. La diferencia entre el error cuadrático medio anterior y el actual es ~ 0.

1.2.3. Algoritmo en *batches* de tamaño parametrizable

La idea de esta versión de *batch* es idéntica a la anterior, salvo que los *batches* se toman de a tamaños parametrizables.

2. Resultados del entrenamiento

En esta sección, se discutirán los mejores resultados logrados con ambos algoritmos, exhibiéndose la progresión del error y analizándose la precisión de la red obtenida al generalizar el terreno.

Glosario de terminología:

- f: Función de activación
- H: Capas ocultas (`array`)
- η: *Learning rate*
- *momentum*: factor del *momentum*
- α: constante de crecimiento del eta adaptativo
- β: factor de decrecimiento del eta adaptativo
- k: constante de decrecimiento consistente
- ϵ: cota inferior para η (condición de corte)
- ϵ_e: condición de corte del error cuadrático medio
- max_iters: número máximo de *epochs*.

2.1. Algoritmo incremental con *adaptive eta*

A continuación siguen los resultados obtenidos para el entrenamiento con el algoritmo incremental utilizando eta adaptativo.

2.1.1. Parámetros y resultados generales

- $f = \tanh$
- $H = [10, 10]$
- $\eta = 0{,}1$
- $momentum = 0$
- $\alpha = 0{,}05$
- $\beta = 0{,}1$

- $k = 5$
- $\epsilon = 0{,}00001$
- $\epsilon_e = {,}0001$
- $max_iters = 10000$

En este primer caso de análisis, se ha logrado obtener una red neuronal que aproxima el terreno con un error cuadrático medio sobre el *set* de entrenamiento de $6{,}5 * 10^{-4}$. Es decir, la ejecución del algoritmo finalizó por haberse alcanzado el máximo de iteraciones. El *set* de pesos que menor error cuadrático medio logró se correspondió con la iteración 9954.

Un aspecto a destacar es la elección de no utilizar *momentum*. Por lo general, al combinar *momentum* con *adaptive eta*, se ha encontrado que el comportamiento del error a lo largo del tiempo es mucho más errático que sin utilizar el primero. Por ende, se ha optado por no usar *momentum* en este caso, optándose por usarlo como mecanismo de aceleración de *backpropagation* en el algoritmo por *batch*.

2.1.2. Tasas de éxito

Un indicador de qué tan acertado fue el entrenamiento de la red se puede encontrar en las tasas de éxito, es decir, en qué proporción la red neuronal generalizó correctamente un nuevo dato. Para ello, se está tomando como punto de comparación el *set* de muestras de prueba extraído tal como se indicó más arriba. Este conjunto representa el 10 % del conjunto original. Los intervalos dentro de los cuales se considera que el valor fue correctamente generalizado están definidos por el Épsilon (ϵ): $[S_i - \epsilon, S_i + \epsilon]$.

Épsilon	Éxito (%)
.05	75.6
.06	80
.07	84.5
.08	91.2
.09	91.2
.1	97.8
.11	100

Cuadro 1: Tasas de éxito

Como se podrá ver, con valores relativamente pequeños del Épsilon se logra tasas altas de éxito, considerando que el espacio de llegada (la coordenada Z) toma valores en el intervalo $[-1, 1]$.

2.1.3. Errores cometidos en el set de comparación

En estrecha vinculación con el punto anterior, se puede analizar la diferencia entre el valor esperado y el valor emitido por la red. Esto está definido por:

$$D_i = abs(S_{c_i} - E_i) \tag{1}$$

Donde E_i es la evaluación del i-ésimo patrón, S_{c_i} es su salida esperada.

Este análisis arroja los siguientes resultados:

- $max(D) = 0{,}10006$
- $min(D) = 1{,}7035 * 10^{-4}$
- $mean(D) = 0{,}030684$

Cómo se puede ver, los valores están en línea con los resultados de la sección anterior. El valor más significativo en este punto es la media de las diferencias, que, como podrá observarse, es pequeña en relación a la longitud del espacio de llegada.

A continuación se presenta la gráfica de cada una de las diferencias entre los valores esperado y calculado.

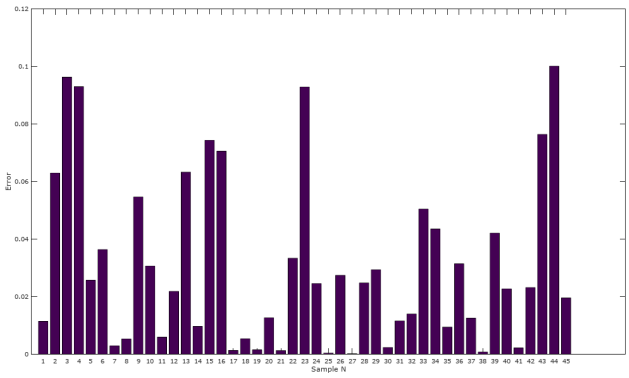

Figura 1: D_i

Como puede verse en la figura 1, existen picos en los cuales la diferencia es mucho mayor a la del resto. Analizar a qué puntos corresponden dichos valores permitirá comprender el comportamiento de la red.

Muestra	X	Y
44	-0.14806	0.31066
3	-1	-0.93007
23	1	-0.11425
4	0.061232	-0.240449

Cuadro 2: Puntos de mayor D_i

En la tabla 2 se pueden observar algunos datos interesantes. Las muestras 3 y 23 se localizan aproximadamente en dos diferentes puntas del terreno. Esto podría estar indicando que la red neuronal tiene mayores dificultades en aprender las puntas por ciertas características particulares que las mismas podrían tener.

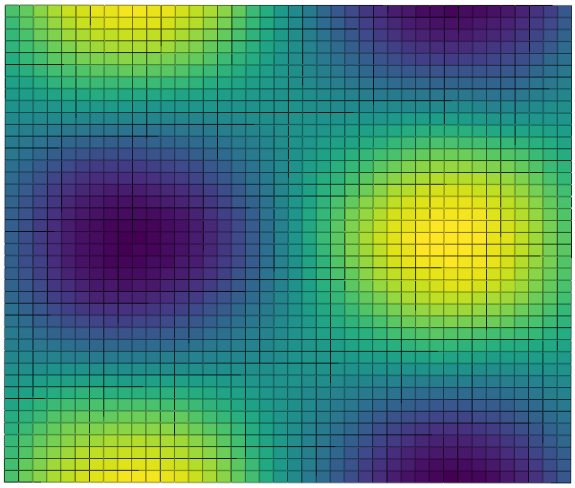

Figura 2: Mapa de altura

Como se podrá ver en el mapa de altura de la figura 2, realizado mediante una generalización realizada por la red, en las puntas hay o bien depresiones muy acentuadas o elevaciones pronunciadas. Nótese que colores mas azulados reflejan puntos más bajos.

Observando los valores de la muestra 4, puede afirmarse que este se encuentra en las inmediaciones de uno de los extremos del terreno, por lo que es de esperar que el comportamiento sea similar al de los casos anteriores.

2.1.4. Análisis del error cuadrático medio

Para analizar la evolución del error cuadrático medio en el tiempo, la gráfica a continuación puede ayudar a entender su comportamiento.

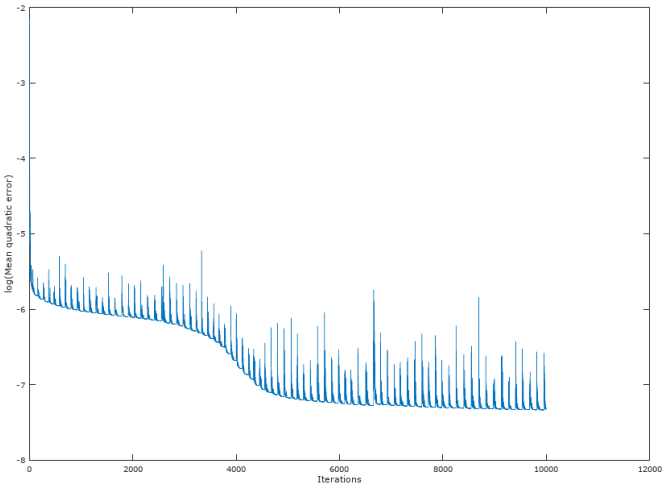

Figura 3: Error cuadrático medio

La figura 3 es una gráfica en escala logarítmica del error cuadrático medio en función del número de iteración. El aspecto quizás más interesante que se puede observar es la periodicidad con la que el error muestra picos de aumento, rápidamente corrigiéndose para mostrar el error una tendencia decreciente. Dicha periodicidad está relacionada con el valor k, es decir, el número de *epochs* que se considera que expresa un decrecimiento consistente en el error. Es probable que lo que se observe sea que, al aumentar el eta, el error inicialmente aumenta para luego corregirse.

Otro aspecto interesante es el abrupto descenso del error en torno a la iteración 4000, sobre todo teniendo en cuenta que la pendiente de la curva, hasta ese momento, era similar a lo que se encuentra en torno a la iteración 6000, es decir, después de dicho suceso. Queda como interrogante el analizar si un descenso como tal podría haber ocurrido de haberse proseguido la ejecución. De todas formas, se considera que el error alcanzado resuelve con creces el objetivo del problema con errores mínimos, tal como se ha discutido en secciones anteriores.