ISSN: 1692-7257 - Volumen 2 – Número 34 - 2019

Revista Colombiana de Tecnologías de Avanzada

Recibido: 16 de febrero de 2019
Aceptado16 de abril de 2019:

UN SOFTWARE INTERACTIVO PARA EL ENTRENAMIENTO DE REDES NEURONALES MULTICAPA USANDO EL MÉTODO SECANTE ESTRUCTURADO

AN INTERACTIVE SOFTWARE FOR THE TRAINING OF THE MULTILAYER NEURAL NETWORKS USING THE STRUCTURED SECANT METHOD

Mg. Favián Arenas A.[*], Mg. Hevert Vivas [*]

[*] Universidad del Cauca,
Departamento de Matemáticas, Grupo de Optimización.
Carrera 2 No. 2N-35, Popayán, Cauca, Colombia.
Tel. +5728209800 ext. 2366, 2379.
E-mail: farenas@unicauca.edu.co, hevivas@unicauca.edu.co.

Resumen: En este artículo, presentamos un nuevo software para el entrenamiento de redes neuronales artificiales, el cual, tiene como base el método de mínimos cuadrados no lineales y que utiliza los métodos secantes estructurados, implementados por primera vez en (Vivas, Martínez, & Pérez, 2018). La novedad de nuestra propuesta es que dicho software permite modificar la cantidad de capas ocultas, el número de neuronas en cada capa, y se incorporan los métodos secantes estructurados. Presentamos pruebas numéricas que muestran el buen desempeño del software en tres problemas de clasificación tomados de un base de datos libre (Newman, 2018).

Palabras clave: Redes Neuronales, secante estructurado, mínimos cuadrados no lineales.

Abstract: In this paper, we present a new software for the training of artificial neural networks, which is based on the method of nonlinear least squares and which uses the structured secant methods, implemented for the first time in (Vivas, Martínez, & Pérez, 2018). The novelty of our proposal is that this software lets to modify the number of hidden layers, the number of neurons in each layer, and the structured drying methods are incorporated. We present numerical tests that show the good performance of the software in three classification problems taken from a free database (Newman, 2018).

Keywords: Neural networks, structured secant , non-linear least squares.

1. INTRODUCCIÓN

Las *Redes Neuronales Artificiales* (RNA) son sistemas de procesamiento de información que funcionan de manera similar a las redes neuronales biológicas. Estas redes tienen en común con el cerebro humano la distribución de las operaciones a realizar en una serie de elementos básicos que, por analogía con los sistemas biológicos, se denominan *neuronas artificiales*, las cuales están relacionadas entre sí, mediante una serie de conexiones que se conocen como *pesos sinápticos*. Estos pesos y conexiones varían con el tiempo mediante un proceso, usualmente iterativo, conocido como *aprendizaje o entrenamiento* de la red.

Las redes neuronales que trataremos en este artículo son redes neuronales multicapa y unidireccionales,

su estructura está conformada básicamente, por una capa de entrada, otra de salida y el resto de las capas intermedias denominadas capas ocultas. El tipo de aprendizaje de esta red es supervisado; es decir, se presenta a la red un conjunto de *patrones*, junto con la salida esperada e iterativamente la red ajusta sus pesos hasta que la salida sea, lo más cercano posible, la deseada. Utilizando para ello, la información del *error* que se comete en cada paso (Martín del Brío, 2006).

Como mencionamos anteriormente, en el proceso de aprendizaje de una red neuronal artificial está inmersa una *función error*, la cual depende explícitamente de los pesos *sinápticos* y proporciona el error que comete la red, comparando en cada iteración del proceso de entrenamiento, la salida obtenida con la salida esperada.
Matemáticamente, la función error es el campo escalar:

$$E: \quad \mathbb{R}^n \longrightarrow \mathbb{R}$$
$$\boldsymbol{w} \longmapsto E(\boldsymbol{w}),$$

donde el vector $\boldsymbol{w}$ tiene como componentes los *pesos sinápticos* de la red. El proceso de aprendizaje en una red neuronal artificial consiste en encontrar un vector $\boldsymbol{w}_*$ que minimice la función E. Así, asociado al rendimiento de una red neuronal, tenemos el siguiente problema de minimización,

$$\begin{array}{l} Minimizar \quad E(\boldsymbol{w}) \\ \boldsymbol{w} \in \mathbb{R}^n \end{array} \qquad (1)$$

Si t^μ y $z^\mu(\boldsymbol{w})$ son, respectivamente, las salidas esperadas y las salidas obtenidas por la red, para un conjunto de p patrones de entrenamiento y unos pesos $\boldsymbol{w}$ y, $r_\mu(\boldsymbol{w}) = \|t^\mu - z^\mu(\boldsymbol{w})\|_2$, entonces la función error viene dada por

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{1}{2}\sum_{\mu=1}^{p}\sum_{i=1}^{m}(t_i^\mu - z_i^\mu(\boldsymbol{w}))^2 \\ &= \frac{1}{2}\sum_{\mu=1}^{p}\|t^\mu - z^\mu(\boldsymbol{w})\|_2^2 \\ &= \frac{1}{2}\sum_{\mu=1}^{p} r_\mu(\boldsymbol{w})^2 . \end{aligned}$$

2. PRELIMINARES

En esta sección, abordamos un problema especial de minimización sin restricciones que surge con frecuencia en problemas prácticos tales como ajuste de curvas, reconocimiento y clasificación de patrones y en redes neuronales artificiales, entre otros. Por su estructura particular, el problema es, por sí mismo, un tema de investigación en el área de optimización. Nos referimos al problema de *Mínimos Cuadrados No Lineales* (MCNL).

En general, los métodos usados para resolver este problema son iterativos; es decir, a partir de una aproximación inicial de la solución, se genera una sucesión de aproximaciones, la cual, bajo ciertas hipótesis, se espera converja a la solución del problema.

El método más popular e importante en la literatura de minimización es el método de *Newton*. Para el caso de los *Mínimos Cuadrados No Lineales* (MCNL), un paso de *Newton* está dado por

$$\begin{aligned} \nabla^2 E(\boldsymbol{w}_k)d_k &= -\nabla E(\boldsymbol{w}_k) \\ \boldsymbol{w}_{k+1} &= \boldsymbol{w}_k + d_k. \end{aligned} \qquad (2)$$

La estructura particular del problema (1), se observa claramente en las expresiones para *la matriz hessiana* de *f* en $\boldsymbol{w}_k$. En efecto, después de realizar algunos cálculos algebraicos, tenemos que la matriz hessiana $\nabla^2 E(\boldsymbol{w}_k)$ se puede expresar como

$$\nabla^2 E(\boldsymbol{w}_k) = J(\boldsymbol{w}_k)^T J(\boldsymbol{w}_k) + S(\boldsymbol{w}_k), \qquad (3)$$

donde $J(\boldsymbol{w}_k)$ es la matriz jacobiana de R en $\boldsymbol{w}_k$ con

$$R(\boldsymbol{w}_k) := \left(r_1(\boldsymbol{w}_k), \dots, r_p(\boldsymbol{w}_k)\right)^T$$

y $S(\boldsymbol{w}_k)$ es una matriz que contiene la información de las segundas derivadas parciales de $r_\mu(\boldsymbol{w})$, para $\mu = 1, \dots, p$, cuyo cálculo, desde el punto de vista computacional, tiene un costo muy alto.

Actualmente en el campo de la optimización los métodos *cuasi Newton*, han estado en auge, porque reducen los costos computacionales.

Estos métodos se caracterizan porque usan una aproximación de la matriz hessiana, en lugar de ella misma. Así, si B_k es una "adecuada" aproximación de la matriz hessiana, entonces la *dirección cuasi Newton* será la solución al sistema de ecuaciones lineales

$$B_k d_k = -\nabla E(\boldsymbol{w}_k).$$

Si además, actualizamos la aproximación B_k de tal forma que satisfaga la llamada *ecuación secante*, entonces surgen los métodos secantes cuya iteración básica es la siguiente

$$\begin{aligned} B_k d_k &= -\nabla E(\boldsymbol{w}_k) \\ \boldsymbol{w}_{k+1} &= \boldsymbol{w}_k + d_k \\ B_{k+1} &= B_k + \Delta_k, \end{aligned}$$

Donde Δ_k (corrección de actualización secante) se calcula dependiendo del método elegido (Nocedal, 2006).

En la Tabla 1, se presenta la manera como se actualiza en cada iteración, según el método, la matriz B_k.

Tabla 1: Métodos para resolver MCNL

Método	Actualización usada
Newton	$B_k = J(\boldsymbol{w}_k)^T J(\boldsymbol{w}_k) + S(x_k)$
Gauss Newton	$B_k = J(\boldsymbol{w}_k)^T J(\boldsymbol{w}_k)$
Levenberg - Marquardt	$B_k = J(\boldsymbol{w}_k)^T J(\boldsymbol{w}_k) + \alpha I_p$
Secante completo	$B_{\boldsymbol{k}}$, donde $B_{\boldsymbol{k}} \approx \nabla^2 E(\boldsymbol{w}_k)$
Secante estructurado	$B_k = J(\boldsymbol{w}_k)^T J(\boldsymbol{w}_k) + \boldsymbol{A_k}$, donde $\boldsymbol{A_k} \approx S(\boldsymbol{w}_k)$

Infortunadamente, los *métodos secantes* tal y como han sido descritos hasta aquí, no aprovechan la estructura de la matriz hessiana dada en (3) y los cálculos del jacobiano que ya han sido realizados. Una alternativa para ello, la representan los llamados *métodos secantes estructurados*. Estos métodos son apropiados para problemas en los cuales la matriz hessiana, tal como sucede en el problema (1), se puede expresar en la forma (3). En un método *secante estructurado*, para el problema de *mínimos cuadrados no lineales* (1), hacemos el proceso iterativo

$$\begin{aligned} B_k d_k &= -\nabla E(\boldsymbol{w}_k) \\ \boldsymbol{w}_{k+1} &= \boldsymbol{w}_k + d_k \\ A_{k+1} &= A_k + \Delta(\mathrm{s_k};\ \mathrm{y_k^{\#}};\ \mathrm{A_k};\ \mathrm{v_k}) \\ B_{k+1} &= A_{k+1} + J(\boldsymbol{w}_{k+1})^T J(\boldsymbol{w}_{k+1}), \end{aligned}$$

donde $\Delta(\mathrm{s_k};\ \mathrm{y_k^{\#}};\ \mathrm{A_k};\ \mathrm{v_k})$, llamada corrección de actualización secante está dada por

$$\Delta(\mathrm{s_k};\ \mathrm{y_k^{\#}};\ \mathrm{A_k};\ \mathrm{v_k}) = \frac{(\mathrm{y_k^{\#}} - \mathrm{A_k d_k})\mathrm{v_k^T} + \mathrm{v_k}(\mathrm{y_k^{\#}} - \mathrm{A_k d_k})^{\mathrm{T}}}{\mathrm{v_k^T d_k}} - \frac{(\mathrm{y_k^{\#}} - \mathrm{A_k d_k})^{\mathrm{T}} \mathrm{d_k v_k v_k^T}}{(\mathrm{v_k^T d_k})^2}.$$

$\mathrm{v_k}$, es llamado escala. La tabla 2 muestra para cada método secante estructurado, el correspondiente valor de $\mathrm{v_k}$. Uno de estos métodos es el BFGS estructurado (BFGSE) presentado en (Martínez, 1989) donde se demuestra que bajo ciertas condiciones, supera en eficiencia a los métodos antes existentes.

Tabla 2: valores de la escala ($\boldsymbol{v}$) según el método o actualización elegida

Método	Actualización usada
PSB estructurado	$\boldsymbol{v} = \boldsymbol{d}$
DFP estructurado	$\boldsymbol{v} = \boldsymbol{y}$
BFGS estructurado	$\boldsymbol{v} = y + \sqrt{\frac{y^T s}{s^T B s}} B\boldsymbol{d}$
SR1 estructurado	$\boldsymbol{v} = \boldsymbol{y} - B\boldsymbol{d}$

Los métodos secantes estructurados no están incluidos en el *toolbox* de MATLAB y, de hecho, fueron implementados por primera vez en (Vivas, Martínez, & Pérez, 2018) para resolver dos problemas y utilizando sendas redes con una estructura fija, es por ello que construimos un *software* que implemente los métodos secantes estructurados y que además permita modificar la estructura de la red, de manera que podamos verificar en la práctica las bondades de estos métodos aplicados al entrenamiento de redes neuronales artificiales.

3. DISEÑO DEL SOFTWARE

Para escribir los códigos de los algoritmos[1] de esta herramienta usamos el *software* MATLAB. El *script* de ejecución es *neuronal_net.m* de donde se eligen los parámetros de ejecución.

La función más relevante y que permitirá al usuario diseñar y entrenar la red, es la salida (*exit_net*), que a partir de un vector **W** (donde se encuentran los pesos sinápticos, junto con los *bias*), un registro **x** (contiene un dato del corpus de entrenamiento) y el vector **hiddenLayer** (indica cuántas capas ocultas y neuronas hay por capa), calcula el valor de salida, permitiendo al usuario, al modificar el vector **hiddenLayer** elegir la estructura de la red.

Por ejemplo, si **hiddenLayer = [3 4],** esto corresponde a una red con dos capas ocultas, tres neuronas en la primera capa oculta y cuatro en la segunda capa oculta. No hay límite para la cantidad de capas ocultas.

[1] Este software se puede descargar desde artemisa.unicauca.edu.co/~farenas/Soft_Redes.rar.

Utilizamos un tipo de estructura de datos llamado *cell* (MathWorks, s.f.), el cual permite almacenar vectores de diferentes tamaños y tipos en un solo vector.
En la primera parte de la función *exit_net* separamos el vector **W** en los respectivos pesos y bias de cada capa.

```
function t=exit_net(x,W,hiddenLayer)
%Hay que extaer w1,w2,w3,b1,b2,b3 a partir de w
xNumb=length(x);
H=[xNumb hiddenLayer 1];% indicador de todas las
capas, ocultas y visibles
HNumb=length(H);% total de capas
lim2=0;
%primero los pesos en vectores mas pequeños
wt=cell(HNumb-1,1);
   for p=2:1:HNumb
       lim1=lim2+1;
       lim2=lim1+H(p)*H(p-1)-1;
       wt{p-1}=W(lim1:lim2);
   end
% ahora los bias
   wbias=cell(HNumb-1,1);
   for q=2:1:HNumb
    lim1=lim2+1;
    lim2=lim1+H(q)-1;
    wbias{q-1}=W(lim1:lim2);
   end
% ahora llevamos los pesos a arreglos tri-indizados
ws(i,j,k)
% k indica la capa
% i indica la neurona de la capa
% j indica la neurona de la capa anterior
for p=1:1:HNumb-1
ws{p}=reshape(wt{p},H(p+1),H(p)); %creamos un
vector de matrices
end
% FIN DE LA SEPARACION
z=x;
 for capa=1:1:HNumb-1
    y=ws{capa}*z+wbias{capa};
    if capa~=HNumb-1
       z=tansig(y);
    else
       z=y;
    end
 end
 t=z;
 end
```

A continuación hablaremos de los resultados experimentales al utilizar este software.

4. EXPERIMENTACIÓN NUMÉRICA

Realizamos las pruebas numéricas en un computador con procesador Intel(R) Core(TM) i5-4200M CPU @2.50 GHZ.

Seleccionamos cuatro conjuntos de datos obtenidos de la base de datos (Newman, 2018). El entrenamiento de la red se realiza mediante la utilización de mínimos cuadrados no lineales, iniciando con un vector de pesos aleatorio y utilizando búsqueda lineal como estrategia de globalización. Finalmente, si el método converge, encontramos un vector $\mathbf{w}^*$ que minimiza el sistema y que corresponde a los pesos sinápticos de la red.

Los experimentos se realizaron para responder dos preguntas: ¿en qué caso es mejor usar un método secante estructurado para el entrenamiento de la red neuronal? En esos casos, ¿Cuál es la mejor configuración de la red neuronal artificial?

Para responder estas preguntas diseñamos 2 experimentos:

1. Comparar el método secante estructurado (con sus cuatro actualizaciones) con los que estánn implementados en el *toolbox* de MATLAB y otros métodos estructurados.
2. Elegir el método más eficiente y entrenar la red neuronal con diferentes configuraciones, desde una sola capa oculta con una sola neurona hasta 3 capas ocultas, cada una, con 10 neuronas y elegir la más eficaz, mediante la prueba de chi cuadrado (χ^2).

Usamos como criterio de convergencia

$$\|E(\boldsymbol{w}_k)\| < 10^{-6},$$

declarando divergencia si el número de iteraciones excede 500.

Los cuatro conjuntos de datos corresponden a los siguientes problemas:

Problema 1: Siluetas de vehículos

El propósito de este problema es clasificar una silueta de un vehículo entre cuatro tipos utilizando un conjunto de características extraídas de cada silueta. Se utilizaron cuatro modelos de vehículos para este experimento: un bus de dos pisos, una furgoneta *Chevrolet*, un *Saab* 9000 y un *Opel Manta* 400. Esta combinación particular de vehículos fue elegida esperando que fueran fácilmente distinguibles. Las características de las imágenes fueron extraídas de las imágenes de las siluetas por el Sistema de proceso jerárquico de

imagen (*Hierarchical Image Processing System-HIPS*) y los detalles de cómo se obtuvieron tanto las imágenes como sus características se pueden consultar en *Statlog* (Vehicle Silhouettes) *Data Set* de (Newman, 2018).

Los datos de entrada y salida son de tipo entero. Originalmente los datos de salida eran los tipos de vehículo, sin embargo los hemos cambiado a entero para facilitar el manejo de los datos. El número de ejemplos es 946 y el número de parámetros es 18 (17 de entrada y 1 de salida).

Problema 2: Cardiotocografía

El seguimiento de la *frecuencia cardíaca fetal* (FCF) sigue siendo ampliamente utilizado como un método para detectar cambios en la *oxigenación fetal* que pueden ocurrir durante el parto. Sin embargo, las muertes y discapacidad a largo plazo de la *hipoxia intraparto* siguen siendo una causa importante de sufrimiento para los padres y las familias, incluso en los países industrializados. Investigaciones confidenciales en el Reino Unido han puesto de manifiesto que el cincuenta por ciento de estas muertes podrían haberse evitado, ya que fueron causadas por falta de reconocimiento de patrones de frecuencia cardíaca fetal anormal, la falta de comunicación entre el personal, o el retraso en la adopción de las medidas oportunas (Ayres-de Campos, 2000).

En este problema se intenta predecir el estado de un feto a partir de un conjunto de datos que contiene mediciones de *frecuencia cardíaca fetal* y de contracción uterina UC. Cuenta con el *monitoreo fetal electrónico* (MFE) clasificado por obstetras expertos. Se procesaron 2126 *monitoreos fetales electrónicos* de forma automática registrando las respectivas medidas de los diagnósticos. Las cardiotocografías (CTG) también fueron clasificadas por tres obstetras y se asignó una etiqueta de clasificación a cada uno de ellos.

Problema 3: Localización inalámbrica en interiores

La detección de usuarios en un entorno interior basado en la intensidad de la señal de Wi-Fi tiene un amplio dominio de aplicaciones. Esto se puede usar para objetivos como ubicar usuarios en sistemas domésticos inteligentes, ubicar delincuentes en regiones limitadas, obtener el conteo de usuarios en un punto de acceso, etc.

Aunque otras Tecnologías como el GPS, Bluetooth y Wi-Fi podrían ser explotadas para proporcionar tales servicios ya se han usado métodos GPS para la localización del usuario, pero estos métodos se utilizaron para lograr precisión solo en ciertos rangos y no se pueden aplicar a ubicaciones interiores debido a señales débiles de los satélites.

Para predecir con precisión la ubicación del usuario, un modelo definido y consistente debe ser entrenado e implementado en un dispositivo de seguimiento o monitoreo. Para la obtención del corpus de entrenamiento, se consideró una configuración en una oficina en Pittsburgh, EE. UU que cuenta con siete enrutadores Wi-Fi y sus potencias de señal recibidas de éstos.

En la *tabla 3* se hace un resumen de los resultados obtenidos en el primer experimento.

Tabla 3:Comparación entre los métodos secantes.

		Problema 1	Problema 2	Problema 3
BFGSE	n	44	221	2
	r	7.117	7.931	4.0 E-6
PSBE	n	-	-	2
	r	-	-	4.0 E-6
DFPE	n	-	-	3
	r	-	-	4.0 E-6
SR1E	n	-	-	2
	r	-	-	4.0 E-6
LM	n	-	-	2
	r	-	-	4.0 E-6
BFGS	n	4	-	62
	r	7.137	-	4.0 E-6

El **Problema 1** solamente convergió con los métodos secantes BFGSE y BFGS, pero el estructurado resultó menos eficiente. En el **Problema 2**, el método BFGSE fue el único que convergió. En el **Problema 3** los métodos secantes estructurados se comportaron de manera eficiente.

Lo anterior nos permite concluir que el método más eficaz es el BFGS estructurado (BFGSE).

La *tabla 4* nos muestra la correspondiente configuración de la red para cada problema, con la cual las probabilidades de obtener la solución esperada es mayor, para ello utilizamos la prueba de bondad de ajuste chi cuadrado (χ^2).

Tabla 4:Estructura optima encontrada con BFGSE.

	hiddenLayer	n	r	probabilidad
Problema1	[2 10 2]	8	7.113	0.9999
Problema2	[7 4 10]	39	7.270	0.9986
Problema3	[5]	33	0.000	0.9167

5. CONCLUSIONES

En este artículo presentamos un software que incorpora el método BFGS estructurado al entrenamiento de redes neuronales artificiales. Además, presentamos algunas pruebas numéricas preliminares que muestran que el software propuesto es competitivo y al ser interactivo proporciona una herramienta útil para elegir el diseño más adecuado de la red para cada problema.

Finalmente, entre las cosas por hacer, y dado que hasta ahora, solo hemos entrenado y validado la red, hace falta un análisis de los *corpus* de entrenamiento, como por ejemplo, el pre-procesamiento de los datos, así podríamos intentar la resolución de estos problemas.

RECONOCIMIENTO

Los autores agradecen a la Universidad del Cauca por el tiempo concedido para esta investigación mediante el Proyecto de investigación VRI ID 4605.

Referencias

Ayres-de Campos, D. &.-d.-S.-L. (2000). SisPorto 2.0: a program for automated analysis of cardiotocograms. *J Matern Fetal Med*, 311–318.

Crisóstomo, J. &.-S.-R. (01 de 08 de 2016). Hyperresistinemia and metabolic dysregulation: a risky crosstalk in obese breast cancer. *Endocrine, 53*(2), 433-442.

Martín del Brío, B. y. (2006). *Redes Neuronales y Sistemas Borrosos.* Madrid: RA-MA editorial.

Martínez, H. J. (1989). Convergence theory for the structured BFGS secant method with an application to nonlinear least squares. *Journal Of Optimization Theory And Applications ISSN: 0022-3239, 61*, 161-178.

MathWorks. (s.f.). *Array de celdas - MATLAB.* Recuperado el 2018, de https://la.mathworks.com/help/matlab/ref/cell.html?s_tid=srchtitle

Newman, A. a. (01 de 09 de 2018). *UCI Machine Learning Repository*. Obtenido de http://archive.ics.uci.edu/ml/

Vivas, H., Martínez, H. J., & Pérez, R. (2018). Método secante estructurado para el entrenamiento del perceptrón multicapa. *Revista de Ciencias, 18*(2), 131-150.