Facultad de Ciencias Naturales y Exactas
Universidad del Valle

Structured Secant Method for the Multilayer Perceptron Training

Hevert Vivas Universidad del Cauca
Héctor Jairo Martínez Universidad del Valle
Rosana Pérez Universidad del Cauca

Received: May 6, 2014 Accepted: August 22, 2014

Pag. 131-150

Abstract

In the group of models of artificial neural networks, it is the multilayer perceptron, a unidirectional neural network consisting of three or more layers, its training is done by an algorithm called backpropagation. In this work, we introduced the structured secant method for the training of multilayer perceptron and we compare its numerical performance with other methods widely used with the same purpose. Some numerical experiments show a good performance of this algorithm.

Keywords: Multilayer perceptron, structured secant method, training neural network, nonlinear least squares.

Método secante estructurado para el entrenamiento del perceptrón multicapa

Resumen

Dentro del grupo de modelos de redes neuronales artificiales está el perceptrón multicapa: una red neuronal unidireccional constituida por tres o más capas, cuyo entrenamiento se hace mediante un algoritmo denominado retro-propagacion de errores. En este trabajo, proponemos e implementamos por primera vez, el método secante estructurado para el entrenamiento del perceptrón multicapa y analizamos su desempeño numérico comparandolo con métodos ampliamente usados con el mismo proposito. Pruebas numéricas preliminares muestran un buen desempeño numérico del método propuesto.

Palabras clave: método secante estructurado, entrenamiento de redes neuronales artificiales, perceptrón multicapa, mínimos cuadrados no lineales.

1 Introduccion

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. En las RNA supervisadas, estos pesos y conexiones varían mediante un proceso, usualmente iterativo, conocido como aprendizaje o entrenamiento de la red en el cual, está inmersa una función error que depende explícitamente de los pesos sinápticos y proporciona el error que comete la red. Matemáticamente, el proceso de aprendizaje consiste en encontrar un vector (una configuración de pesos) que minimice dicha función error. Esto conduce a un problema de minimización, más exactamente, un problema de mínimos cuadrados no lineales.

El modelo de redes neuronales artificiales más empleado en aplicaciones prácticas es el del perceptrón multicapa, una red neuronal unidireccional constituida por al menos, tres capas junto con su algoritmo de entrenamiento denominado retropropagación de errores (backpropagation) [8].

En este artículo, proponemos e implementamos por primera vez, el método secante estructurado para el entrenamiento del perceptrón multicapa, y analizamos su desempeño numérico comparándolo con métodos ampliamente usados con el mismo propósito, tales como: Gauss-Newton y Levenverg-Marquardt [8]. Pruebas numéricas preliminares muestran un buen desempeño numérico del método propuesto.

Organizamos la presentación de este documento de la siguiente forma. En la Sección 2, presentamos el problema de Mínimos Cuadrados No Lineales como un caso particular de minimización sin restricciones y describimos diferentes métodos de solución y sus propiedades de convergencia, centrándonos en el método secante estructurado. En la Sección 3, presentamos en forma descriptiva las redes neuronales artificiales, su estructura y su funcionamiento, profundizando en el perceptrón multicapa y en su algoritmo de entrenamiento denominado retropropagación de errores. En la Sección 4, proponemos e implementamos por primera vez, el método secante estructurado para el entrenamiento del perceptrón multicapa y analizamos numéricamente su desempeño para diferentes actualizaciones secantes. Finalmente, en la Sección 5, hacemos algunos comentarios finales y propuestas de trabajos futuros sobre el tema.

2 Mínimos cuadrados no lineales (MCNL)

En esta sección, abordamos un problema particular 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. Nos referimos al problema de Mínimos Cuadrados No Lineales.

2.1 Planteamiento del problema

Dada $R : \mathbb{R}^n \longrightarrow \mathbb{R}^m,\ m \geq n,\ R(\mathbf{x}) = (r_1(\mathbf{x}), \ldots, r_m(\mathbf{x}))^T$, una función no lineal y dos veces continuamente diferenciable, el problema de Mínimos Cuadrados No Lineales (MCNL) consiste en resolver el problema de minimización sin restricciones

$$\underset{\mathbf{x} \in \mathbb{R}^n}{Minimizar} \qquad f(\mathbf{x}) = \frac{1}{2} R(\mathbf{x})^T R(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{m} r_i(\mathbf{x})^2 \ . \tag{1}$$

La estructura particular del problema (1), se observa claramente en las expresiones para el vector gradiente y la matriz hessiana de f, en $\mathbf{x}$. En efecto,

$$\nabla f(\mathbf{x}) = J(\mathbf{x})^T R(\mathbf{x}) \qquad \text{y} \qquad \nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) + S(\mathbf{x}), \tag{2}$$

donde $J(\mathbf{x}) = \left(\nabla r_1(\mathbf{x})^T, \ldots, \nabla r_m(\mathbf{x})^T\right)^T$ y $S(\mathbf{x}) = \sum_{i=1}^{m} r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x})$.

Observemos que la matriz jacobiana $J(\mathbf{x})$ contiene solamente información de primer orden (primeras derivadas parciales) y $S(\mathbf{x})$ contiene información de segundo orden (es una combinación lineal de matrices hessianas). Esta estructura especial de la matriz hessiana de f, es la que se aprovecha en algunos de los métodos usados para resolver el problema (1) y es la razón por la cual no se usan métodos de propósito general para resolver el mismo. La información de primer orden es relativamente fácil de calcular, mientras que la de segundo orden es numéricamente difícil de calcular, ya que involucra el cálculo de m hessianos, lo que implica un alto costo computacional [2, 12, 5].

2.2 Métodos de solución

En esta sección, describimos algunos métodos de solución del problema MCNL. Incluimos, dada su popularidad e importancia, métodos de propósito general para resolver problemas de minimización tales como el método de Newton y los métodos secantes, los cuales debido a que no aprovechan la estructura particular del problema (1), no son muy apropiados para resolverlo. Además, describimos métodos especialmente diseñados para resolver el problema MCNL, como Gauss Newton, Levenberg-Marquardt y secante estructurado. Todos estos métodos coinciden en que una iteración básica incluye la solución de un sistema de ecuaciones lineales para posteriormente, generar la aproximación siguiente.

2.2.1 Método de Newton

La idea básica del método de Newton para resolver un problema de minimización sin restricciones consiste en, dada una aproximación a la solución del problema, resolver en cada iteración un sistema de ecuaciones lineales, cuya "solución" es usada para generar la aproximación siguiente. Es decir, su iteración básica es:

$$\begin{aligned} \nabla^2 f(\mathbf{x}_k)\mathbf{d}_k &= -\nabla f(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathbf{d}_k. \end{aligned} \tag{3}$$

La convergencia de este método es q-cuadrática, siempre y cuando $\nabla^2 f$ sea Lipschitz continua alrededor de $\mathbf{x}_k$ y $\nabla^2 f(\mathbf{x}_k)$ sea definida positiva [2]. A pesar de que las propiedades de convergencia del método de Newton son muy buenas comparadas con otros métodos, el término $S(\mathbf{x})$ es, computacionalmente, costoso de calcular.

2.2.2 Método Gauss-Newton

El método de Gauss-Newton es una variante del método de Newton y su iteración básica es:

$$\begin{aligned} J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\boldsymbol{s}_k &= -J(\boldsymbol{x}_k)^T R(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{s}_k \end{aligned}$$

La rapidez de convergencia de este método decrece a medida que el grado de no linealidad o el tamaño residual del problema crece; incluso, si estas medidas son grandes, el método podría no converger [2].

2.2.3 Método de Levenberg-Marquard

Este método puede verse también como una variante del método de Newton cuya iteración básica es

$$\begin{aligned} [J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + \mu_k I]\boldsymbol{s}_k &= -J(\boldsymbol{x}_k)^T R(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{s}_k. \end{aligned}$$

Este método surgió para evitar dificultades del método Gauss-Newton cuando, a lo largo del proceso iterativo, la matriz Jacobiana no tiene rango completo o está mal condicionada. Las propiedades de convergencia del método de Levenberg-Marquard son similares a las del método de Gauss-Newton; sin embargo, muchas implementaciones de su algoritmo superan las desventajas propias del método de Gauss-Newton [2].

2.2.4 Métodos Secantes

En la iteración de Newton (3), es necesario calcular la matriz hessiana de f en $\mathbf{x}_k$ y, para encontrar el paso de Newton, vía factorización de Cholesky, se requiere que la matriz $\nabla^2 f(\mathbf{x}_k)$ sea definida positiva; por lo cual, a no ser que ella tenga una estructura particular, dicha factorización es muy costosa computacionalmente. Si a lo anterior le agregamos el hecho de que, en general, el sólo cálculo del hessiano de f en $\mathbf{x}_k$ ya es muy costoso, se hace indispensable tener métodos que resuelvan el mismo problema de minimización, sin tener que realizar estos cálculos. De esta manera, surgen los denominados métodos cuasi Newton, los cuales usan una aproximación de la matriz hessiana en lugar de ella misma. Así, si B_k es una "buena" aproximación de la matriz hessiana $\nabla^2 f(\mathbf{x}_k)$, entonces la dirección cuasi newton será la solución al sistema de ecuaciones lineales

$$B_k \mathbf{d}_k = -\nabla f(\mathbf{x}_k).$$

Cuando decimos una "buena" aproximación, nos referimos a que requerimos que B_k conserve las buenas propiedades del hessiano (simetría) y que permita encontrar direcciones de descenso (definida positiva). Así, al resolver un problema de minimización, usando un método cuasi Newton, ganamos estabilidad numérica, eficiencia y convergencia [13]. Si además, actualizamos la aproximación B_k de tal forma que satisfaga la llamada ecuación secante[1]

$$B_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k),$$

entonces surgen los denominados métodos secantes, cuya iteración básica es la siguiente:

$$\begin{aligned} B_k\mathbf{s}_k &= -\nabla f(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathbf{s}_k \\ B_{k+1}\mathbf{s}_k &= \mathbf{y}_k, \end{aligned}$$

donde $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ y B_{k+1} es llamada actualización secante.

Existen varias actualizaciones secantes exitosas. Entre ellas, están la BFGS, propuesta independientemente por Broyden en (1969) y Fletcher, Goldfarb y Shano en (1970), dada por:

$$B_{k+1} = B_k + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k} - \frac{B_k\mathbf{s}_k\mathbf{s}_k^T B_k}{\mathbf{s}_k^T B_k\mathbf{s}_k}$$

y la actualización DFP propuesta por Davidon en (1959), Fletcher y Powell en (1963) definida por

$$B_{k+1} = B_k + \frac{(\mathbf{y}_k - B_k\mathbf{s}_k)\mathbf{y}_k^T + \mathbf{y}_k(\mathbf{y}_k - B_k\mathbf{s}_k)^T}{\mathbf{y}_k^T\mathbf{s}_k} - \frac{(\mathbf{y}_k - B_k\mathbf{s}_k)^T\mathbf{s}_k\mathbf{y}_k\mathbf{y}_k^T}{(\mathbf{y}_k^T\mathbf{s}_k)^2}.$$

En la práctica, se recomienda la actualización BFGS, debido a que su desempeño numérico es mejor; sin embargo, la actualización DFP fue la primera actualización secante propuesta, por lo cual tiene un gran interés tanto histórico como analítico [13, 2].

2.2.5 Métodos secantes estructurados

Desafortunadamente, los métodos secantes tal y como han sido descritos hasta aquí, no aprovechan la estructura de la matriz hessiana dada en (2). Es decir, no aprovechan los cálculos del jacobiano ya 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:

$$\nabla^2 f(\mathbf{x}) = C(\mathbf{x}) + S(\mathbf{x}),$$

[1]El nombre de ecuación secante se utiliza porque en el caso $n = 1$, B_{k+1} representa la pendiente de la recta secante a la gráfica de la función f' que une los puntos $(x_k, f'(x_k))$ y $(x_{k+1}, f'(x_{k+1}))$.

donde $C(\mathbf{x})$ contiene información "fácil" de obtener y $S(\mathbf{x})$ contiene información que es "difícil" o imposible de calcular. Así, en un método secante estructurado, basta hacer una aproximación secante de la parte "difícil", $S(\mathbf{x})$, conservando el resto de la estructura.

En particular, para el problema MCNL (1), tenemos que $\nabla^2 f(\mathbf{x}) = C(\mathbf{x}) + S(\mathbf{x})$, donde:

$$C(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) \qquad y \qquad S(\mathbf{x}) = \sum_{i=1}^{m} r_i(\mathbf{x})^T \nabla^2 r_i(\mathbf{x}).$$

Con ello, en un método secante estructurado, para el problema de mínimos cuadrados no lineales (1), hacemos el proceso iterativo

$$\begin{aligned} B_k \mathbf{s}_k &= -\nabla f(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + s_k \\ A_{k+1} &= A_k + \Delta(s_k, \mathbf{y}_k^{\#}, A_k, v_k) \\ B_{k+1} &= A_{k+1} + C(\mathbf{x}_{k+1}), \end{aligned}$$

donde A_k es una aproximación a $S(\mathbf{x}_k)$ y $\Delta(\mathbf{s}_k, \mathbf{y}_k^{\#}, A_k, \mathbf{v}_k)$, llamada corrección de actualización secante, está definida por

$$\Delta(\mathbf{s}_k, \mathbf{y}_k^{\#}, A_k, \mathbf{v}_k) = \frac{(\mathbf{y}_k^{\#} - A_k\mathbf{s}_k)\mathbf{v}_k^T + \mathbf{v}_k(\mathbf{y}_k^{\#} - A_k\mathbf{s}_k)^T}{\mathbf{v}_k^T\mathbf{s}_k} - \frac{(\mathbf{y}_k^{\#} - A_k\mathbf{s}_k)^T\mathbf{s}_k\mathbf{v}_k\mathbf{v}_k^T}{(\mathbf{v}_k^T\mathbf{s}_k)^2}.$$

El vector $\mathbf{v}_k \in \mathbb{R}^n$ es denominado la escala y con frecuencia es función de $\mathbf{s}$, $\mathbf{y}$ y B. Diferentes valores de la escala permiten obtener actualizaciones reconocidas y muy utilizadas [9] como por ejemplo:

$$\begin{aligned} PSB \quad \mathbf{v} &= \mathbf{s} \\ DFP \quad \mathbf{v} &= \mathbf{y} \\ BFGS \quad \mathbf{v} &= \mathbf{y} + \left[\frac{\mathbf{y}^T\mathbf{s}}{\mathbf{s}^T B\mathbf{s}}\right]^{1/2} B\mathbf{s} \\ SR1 \quad \mathbf{v} &= \mathbf{y} - B\mathbf{s} \end{aligned}$$

Los vectores $\mathbf{y}$ y $\mathbf{y}_k^{\#}$ son aproximaciones a $\nabla^2 f(\mathbf{x})\mathbf{s}$ y $\mathbf{S}(\mathbf{x}_k)\mathbf{s}$, respectivamente. Dennis, en (1976) y Bartholomew-Biggs, en (1977) sugirieron, independientemente, usar $\mathbf{y}_k^{\#} = (J(\mathbf{x}_{k+1})^T - J(\mathbf{x}_k)^T)R(\mathbf{x}_{k+1})$ y Al-Baali y Fletcher, en (1985) sugirieron $\mathbf{y}_k = \mathbf{y}_k^{\#} + J(\mathbf{x}_{k+1}^T)J(\mathbf{x}_{k+1})\mathbf{s}$ en lugar de la escogencia $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$, mejorando así el comportamiento del código $NL2SOL$[2].

Entre las ventajas del método secante estructurado, está el hecho de que no es necesario el cálculo analítico de la matriz hessiana; basta aproximar solo la

[2] $NL2SOL$ es una biblioteca en FORTRAN90 que implementa un algoritmo para resolver problemas de mínimos cuadrados no lineales y fue creada por John Dennis, David Gay y Roy Welsch. Este es un método de implementación para el algoritmo secante estructurado que usa una estrategia de región de confianza para globalizarlo [9, 2]

parte que contiene la información de segundo orden, la cual es costosa de obtener computacionalmente y difícil de obtener analíticamente. Además, su convergencia es q-superlineal y no depende del tamaño del residuo de la función objetivo [2].

La teoría de convergencia para los métodos secante estructurados PSB, DFP fue desarrollada en 1981 [3] mientras que la del método BFGS estructurado fue establecida en 1989 [10]. Una aplicación directa de esta teoría da la primera prueba de convergencia local y q-superlineal del método BFGS estructurado para el problema de mínimos cuadrados no lineales, el cual es usado por Dennis, Gay, y Welsh en la versión actual del código NL2SOL [4].

3 Redes Neuronales Artificiales (RNA)

La historia de las redes neuronales artificiales está llena de creatividad individual en diferentes campos y ha sido documentada por varios autores. Desde antes de la aparición del primer computador hasta hoy, han ocurrido varios hechos que marcaron la historia de las redes neuronales artificiales. En la actualidad, son numerosos los trabajos que se realizan y publican cada año las aplicaciones nuevas que surgen (sobretodo en el área de control) y las empresas que lanzan al mercado productos nuevos, tanto en hardware como en software (sobre todo para simulación) [8].

Informalmente, un sistema neuronal artificial tiene una estructura análoga al sistema neuronal biológico cuyos elementos básicos, llamadas neuronas artificiales, se conectan entre sí y se organizan en capas para formar la red neuronal.

3.1 Modelo general de neurona artificial

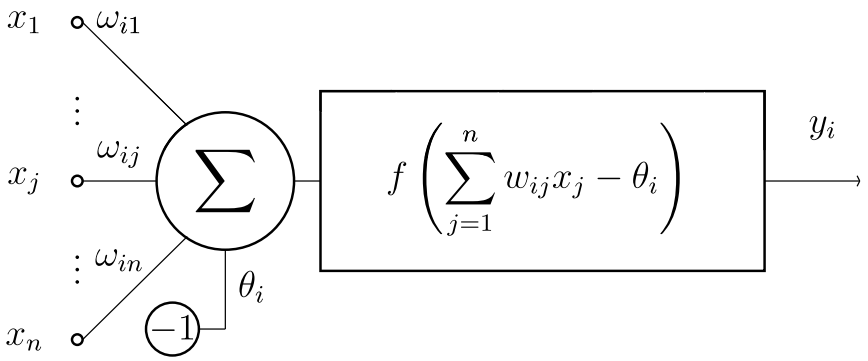

Figura 1: *Principales elementos de una neurona artificial.*

Al igual que una neurona biológica, una neurona artificial posee unas entradas que pueden provenir del exterior o de otras neuronas conectadas a ella y proporciona una única salida. En una red neuronal artificial, los elementos que constituyen la "neurona i" son las entradas $x_j(t)$, los pesos sinápticos, w_{ij}, la regla de propagación, $h_i(t) = \sigma(w_{ij}, x_j(t))$, donde $\sigma(\cdot,\cdot)$ proporciona el valor del potencial postsináptico; la función de activación, $f_i(a_i(t-1), h_i(t))$, donde $a_i(t)$ proporciona el estado de activación actual; finalmente, la función de salida, $F_i(a_i(t))$.

En general, el modelo que habitualmente se usa es aquel cuya regla de propagación es la suma ponderada de las entradas y de sus pesos respectivos, la función de activación proporciona su salida y tiene un parámetro adicional θ_i, conocido como umbral o bias, el cual puede tener diferentes usos dependiendo del modelo (Figura 1). Así, dicho modelo se puede expresar por la igualdad

$$y_i(t) = f_i\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right) = f_i\left(\sum_{j=0}^{n} w_{ij}x_j\right),$$

donde en la última parte hemos incluido el parámetro θ_i como si fuera el peso w_{i0}, con la convención de que $x_0 = -1$.

3.2 Estructura

En una red neuronal artificial, podemos distinguir tres tipos de capas. Una capa de entrada formada por las neuronas que reciben la información del medio exterior, una capa de salida formada por las neuronas que transfieren la información procesada al exterior y una capa intermedia u oculta (la cual puede o no existir), en la cual se procesa toda la información sin tener conexión con el entorno donde opera.

Dependiendo del enfoque, se pueden establecer diferentes arquitecturas de redes neuronales artificiales. De acuerdo al número de capas hablamos de redes neuronales monocapa compuestas por una única capa; o redes neuronales multicapa compuestas por varias capas. En relación a la manera como fluye la información, tenemos las redes neuronales unidireccionales (feedforward), en las cuales la información fluye en un solo sentido y las redes recurrentes o retroalimentadas (feedback), en las que la información puede fluir en cualquier sentido, incluido el de entrada-salida.

3.3 Aprendizaje

Una RNA no está completa si no podemos garantizar que funcione correctamente con un cierto grado de confianza; para ello, al igual que nuestro cerebro, funciona mejor en la medida que reciba un buen aprendizaje o entrenamiento, una vez definida su estructura, las redes neuronales artificiales necesitan pasar por un proceso de aprendizaje, en el cual se ajustan sus pesos sinápticos, con el fin de adaptar su desempeño al entorno donde operará. El tipo de aprendizaje es determinado de acuerdo a la manera en que dichos pesos son ajustados.

En efecto, con la estructura de la red fija, el proceso de aprendizaje consiste en modificar los pesos sinápticos siguiendo una cierta regla de aprendizaje. En este sentido, básicamente, existen dos tipos de aprendizaje: el supervisado que se caracteriza por tener un control externo a través de un supervisor o maestro, el cual conoce las salidas deseadas correspondientes a un conjunto de entradas; dichas salidas permiten definir una función error que se desea minimizar. El otro

tipo de aprendizaje es el no supervisado o autoorganizado que consiste en estimar los pesos de la red, en función de la caracterización de los datos de entrada de acuerdo a un objetivo específico que permita detectar sus patrones.

3.4 Perceptrón multicapa

Dentro del grupo de redes unidireccionales que usan aprendizaje supervisado, están el perceptrón simple, la adalina y el perceptrón multicapa. El perceptrón simple y la adalina son de gran interés histórico, pues su evolución representa la historia misma de las redes neuronales artificiales [8]. En general, la importancia de estos modelos se debe a su carácter de dispositivos entrenables.

El perceptrón multicapa es una red neuronal unidireccional constituida por tres o más capas: una capa de entrada, otra capa de salida y el resto de capas intermedias denominadas capas ocultas. La estructura de un perceptrón multicapa, con una capa oculta[3], se representa en la Figura 2.

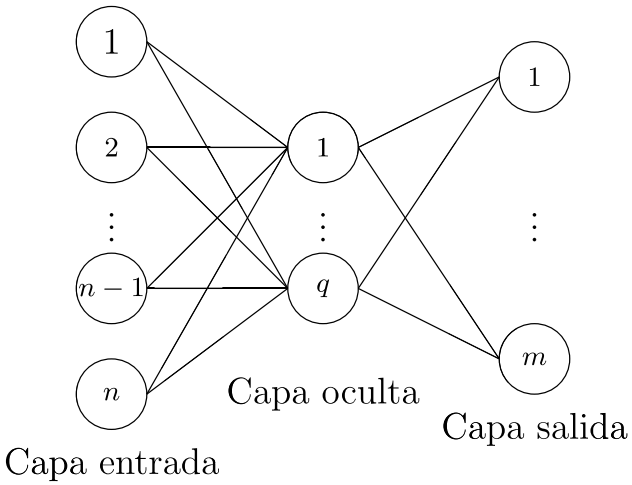

Figura 2: *Perceptrón multicapa* (MLP).

Sean x_i las entradas de la red; y_j, las salidas de la capa oculta; z_k, las salidas de la capa final; w_{ij}, los pesos de la capa oculta y θ_j, sus umbrales; w'_{kj}, los pesos de la capa de salida, y θ'_k, sus umbrales, para todo $i = 1, \ldots, n$, $j = 1, \ldots, q$ y para todo $k = 1, \ldots, m$. La operación de un perceptrón multicapa con estas características se expresa matemáticamente por la ecuación:

$$z_k = \sum_{j=1}^{q} w'_{kj} y_j - \theta'_k = \sum_{j=1}^{q} w'_{kj}\, f\left(\sum_{i=1}^{n} w_{ji} x_i - \theta_j\right) - \theta'_k, \tag{4}$$

donde f es la función de activación.

[3]Existen diversas demostraciones de que este modelo de perceptrón multicapa es un aproximador universal de funciones [8].

Como ya mencionamos, el aprendizaje de un perceptrón multicapa se hace a través de la minimización de una función error que mide la diferencia entre la salida $\mathbf{z}$ obtenida por la red y la salida deseada $\mathbf{t}$. Matemáticamente, la función error es es un campo escalar $E : \mathbb{R}^n \to \mathbb{R}$, en la variable $\mathbf{w}$, que para nuestro caso es un vector cuyas componentes son los pesos sinápticos. Así, asociado al aprendizaje de un perceptrón multicapa, tenemos el siguiente problema de optimización:

$$\begin{aligned} &Minimizar \quad E(\mathbf{w}). \\ &\mathbf{w} \in \mathbb{R}^n \end{aligned} \tag{5}$$

En el caso de una muestra finita formada por los patrones de entrada, $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^p$, vectores de $\mathbb{R}^n$, cada uno de los cuales tiene como componentes las entradas de la red, y de los vectores de $\mathbb{R}^m$, $\mathbf{t}^1, \mathbf{t}^2, \ldots, \mathbf{t}^p$, que contienen las salidas deseadas, la función error E es la siguiente

$$E(\mathbf{w}) = \frac{1}{p} \sum_{\mu=1}^{p} \sum_{i=1}^{m} (\mathbf{t}_i^\mu - \mathbf{z}_i^\mu(\mathbf{w}))^2 \equiv \frac{1}{p} \sum_{\mu=1}^{p} \|\mathbf{t}^\mu - \mathbf{z}^\mu(\mathbf{w})\|_2^2, \tag{6}$$

donde cada vector $\mathbf{z}^\mu(\mathbf{w}) \in \mathbb{R}^m$, con $\mu = 1, \ldots, p$, contiene las respuestas de la red correspondientes al patrón de entrada $\mathbf{x}^\mu$, cuando los pesos sinápticos están dados por $\mathbf{w}$. Así, la función E permite obtener el error cuadrático medio de las salidas de la red respecto de las deseadas [8].

En el proceso iterativo del algoritmo de entrenamiento de una red neuronal multicapa, se lleva a cabo una fase de ejecución de la red para los patrones de entrenamiento. Existen dos maneras de hacer esta ejecución; una denominada aprendizaje por lotes, que consiste en presentar a la red todos y cada uno de los patrones de entrenamiento, calcular para cada patrón, el error en la salida y por último, proceder a hacer la actualización de los pesos sinápticos; y la otra llamada aprendizaje en serie que consiste en calcular el error en la salida y actualizar los pesos sinápticos tras la presentación de cada patrón de aprendizaje, teniendo presente que en cada iteración, el orden en la presentación de los patrones sea aleatorio [8].

Para el perceptrón multicapa definido anteriormente, si $\mathbf{x}^\mu$ para $\mu = 1, \ldots, p$ es un patrón de entrada, la ejecución de la red (4) se expresa como:

$$z_k^\mu = g\left(\sum_{j=1}^{q} w'_{kj} y_j^\mu - \theta'_k\right) = g\left(\sum_{j=1}^{q} w'_{kj} f\left(\sum_{i=1}^{n} w_{ji} x_i^\mu - \theta_j\right) - \theta'_k\right),$$

donde g es la función de activación de las neuronas de salida y f de las ocultas. En este sentido, la función error cuadrático medio es

$$E\left(w_{ji}, \theta_j, w'_{kj}, \theta'_k\right) = \frac{1}{p} \sum_{\mu=1}^{p} \sum_{k=1}^{m} \left[t_k^\mu - g\left(\sum_{j=1}^{q} w'_{kj} y_j^\mu - \theta'_k\right)\right]^2. \tag{7}$$

El problema (5) con E dada por (7) es un problema MCNL [2].

4 Pruebas numéricas

En esta sección, implementamos por primera vez, el método secante estructurado para el entrenamiento del perceptrón multicapa. Con el propósito de comparar su desempeño numérico, también implementamos los métodos de Gauss-Newton y Levenverg-Marquardt, ampliamente utilizados con el mismo propósito, en paquetes (o programas) como el Toolbox de redes neuronales de MATLAB®. Usamos las cuatro fórmulas dadas en (4) para actualizar en cada iteración la matriz A_k dada por (4), con lo cual tenemos cuatro versiones del algoritmo secante estructurado, las cuales llamaremos: método PSBE, DFPE, BFGSE y SR1E, respectivamente.

Para las pruebas numéricas, consideramos el entrenamiento de dos redes del tipo perceptrón multicapa para resolver sendos problemas: evaluar la función seno y predecir el consumo de energía eléctrica en una determinada región, en un día dado y una hora determinada usando la red propuesta por [15]. Para escribir los códigos de los algoritmos y de las funciones objetivo de cada problema, usamos el software MATLAB® versión 2010. Realizamos las pruebas numéricas en un computador Intel (R) Core (TM) i5-CPU de 2.67 GHz. La presentación de los parámetros de entrenamiento la hicimos usando la técnica de entrenamiento por lotes descrita en la Sección 3.

4.1 Algoritmo general

Como lo mencionamos anteriormente, en el entrenamiento del perceptrón multicapa, resolvemos el problema de minimización (5) con E definida por (6). En general, para la obtención de los pesos iniciales, se recomienda iniciar con vectores aleatorios [8]; los algoritmos están implementados, usando una estrategia de globalización denominada búsqueda lineal, que permita iniciar desde cualquier punto [12, 2].

Los métodos globalizados, en cada iteración, determinan una dirección de descenso $\mathbf{s}_k$, y con una estrategia de búsqueda lineal [2] encuentran un tamaño de paso λ_k, con el cual se define la aproximación siguiente $\mathbf{w}_{k+1}$.

Usamos dos criterios de parada en nuestro algoritmo: uno relacionado con el tamaño del gradiente de la función objetivo ($\mathbf{g} = \nabla E(\mathbf{w})$) y el otro, relacionado con el número de iteraciones (n). Exactamente, declaramos convergencia si $\|\mathbf{g}\|_2 \leq tol$ y divergencia si $n > N$, donde tol es un número real muy pequeño que denota la tolerancia usada y N es el número máximo de iteraciones en el algoritmo.

A continuación, presentamos la estructura general del algoritmo para el entrenamiento de un perceptrón multicapa, en el cual se asume conocida la arquitectura de red neuronal artificial.

Algoritmo 1 *Dados los patrones de entrenamiento:* $\boldsymbol{x}^\mu$ *y* $\boldsymbol{t}^\mu$, $\mu = 1, \ldots, p$, *se procede como sigue:*

P.0. Inicialización
Generar los pesos iniciales $\boldsymbol{w}_0$.
Calcular la salida de la red para los p *patrones de entrenamiento, y el error en la salida.*

P.1. Criterios de parada
$\|\nabla E(\boldsymbol{w}_k)\|_2 > Tol \;\; y \;\; k \leq N$

P.2. Búsqueda direccional
Calcule B_k *y encuentre* $\boldsymbol{s}_k$ *tal que* $B_k \boldsymbol{s}_k = -\nabla E(\boldsymbol{w}_k)$.

P.3. Búsqueda lineal
Calcular λ_k *tal que* $E(\boldsymbol{w}_k + \lambda_k \boldsymbol{s}_k) \leq E(\boldsymbol{w}_k) + \alpha \lambda_k \nabla E(\boldsymbol{w}_k)^T \boldsymbol{s}_k$.

P.4. Actualización
Definir $\boldsymbol{w} : \; \boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \lambda_k \boldsymbol{s}_k$.

Para el paso P.3 del algoritmo, en las pruebas numéricas, usamos como valor inicial del tamaño de paso $\lambda_0 = 1$ y como valores de la constante α usamos dos valores 0.0001 y 0.01.

Realizamos dos tipos de pruebas numéricas:

1. Comparar el desempeño numérico de las cuatro versiones del método secante estructurado. Haremos referencia a estas versiones como métodos PSBE, DFPE, BFGSE y SR1E, respectivamente.

2. Comparar el desempeño numérico de los métodos BFGS, Levenverg-Marquard, Gauss-Newton y "el mejor" de los métodos estructurados mencionados en el numeral anterior.

Problema 1: Evaluación de la función seno

Este problema ilustra el uso del percetrón multicapa como aproximador universal de funciones. En efecto, para cualquier función de $\mathbb{R}^n$ en $\mathbb{R}^m$, siempre es posible diseñar y entrenar un perceptrón multicapa, de tal manera que realice un ajuste de los datos de dicha función con un grado de precisión predefinido [1]. En particular, es relativamente sencillo evaluar una función de variable y valor real tal como la función seno, mediante una de estas redes.

Resolvimos el problema mediante una percetrón multicapa de tres capas (Figura 3), donde el número de neuronas en la capa oculta (nco) es definido por el usuario. Los patrones de entrenamiento fueron $\mathbf{x} = (x_1, \ldots, x_p)^T$, $p = 41$ y $\mathbf{t} = (t_1, \ldots, t_p)^T$ con $t_i = sen\, x_i$, donde las componentes de $\mathbf{x}$ (entradas de la red) son números reales distribuidos uniformemente en el intervalo $[0, 2\pi]$. Usamos como función de activación la sigmoidal (logsig) y la identidad (purelin) en la capa oculta y de salida, respectivamente [1].

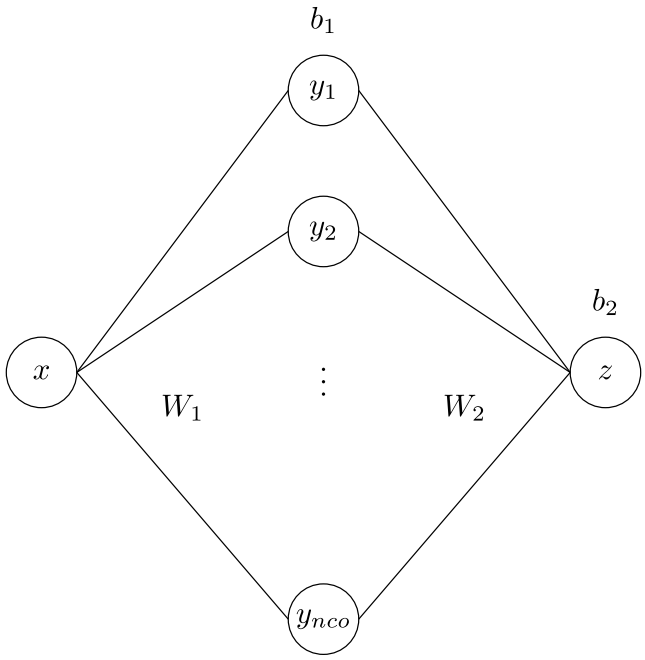

Figura 3: *Perceptrón multicapa que aproxima la función seno.*

El vector de pesos iniciales para cada valor de nco, $\mathbf{w}_0^{nco}$, lo generamos aleatoriamente con la función de MATLAB® $randn(\cdot,\cdot)$, exactamente $\mathbf{w}_0^{nco} = randn(3nco+1,1)$ [14].

Presentamos los resultados de estas pruebas en dos tablas, cuyas dos primeras columnas contienen la información sobre el número de neuronas en la capa oculta (nco) y la tolerancia usada (Tol). Las cuatro columnas siguientes contienen, para cada método, el tiempo de ejecución (t), medido en segundos, y el número de iteraciones (n). El símbolo "-" indica que hubo divergencia del método considerado (se excedió el número máximo de iteraciones permitido ($N = 500$)).

Tabla 1: Resultados de los métodos secantes estructurados para la evaluación de la función seno.

	$N = 500$	***PSBE***		***DFPE***		***BFGSE***		***SR1E***	
nco	Tol	t	n	t	n	t	n	t	n
4	10^{-3}	1	152	–	–	0.5	69	–	–
4	10^{-5}	–	–	–	–	1	134	–	–
4	10^{-6}	–	–	–	–	2	319	–	–
5	10^{-3}	–	–	–	–	1	116	–	–
5	10^{-5}	–	–	–	–	1	153	–	–
5	10^{-6}	–	–	–	–	–	–	–	–
6	10^{-3}	–	–	–	–	1	78	–	–
6	10^{-5}	–	–	–	–	–	–	–	–

Tabla 2: Resultados de los métodos de Gauss-Newton (GN), Levenberg-Marquardt (LM) y secante estructurado (BFGSE) para la evaluación de la función seno.

	$N = 500$	***GN***		***LM***		***BFGSE***	
nco	Tol	t	n	t	n	t	n
4	10^{-3}	-	-	1	160	0.5	69
4	10^{-5}	-	-	-	-	1	134
4	10^{-6}	-	-	-	-	2	319
5	10^{-3}	-	-	-	-	1	116
5	10^{-5}	-	-	-	-	1	153
5	10^{-6}	-	-	-	-	-	-
6	10^{-3}	-	-	-	-	1	78
6	10^{-5}	-	-	-	-	-	-

En la Tabla 1, podemos observar que en general, el método BFGSE convergió en el menor tiempo y número de iteraciones, mientras que los métodos DFPE y SR1E no convergieron en ningún de los casos. De la Tabla 2, observamos que para $Tol \leq 10^{-5}$ y $nco = 4$ y 5, el método BFGSE siempre converge y lo hace con un mejor desempeño numérico que los otros métodos comparados aquí. Además, el método Levenberg-Marquardt presentó mejor desempeño numérico que Gauss-Newton.

Cabe mencionar que, la divergencia en los métodos secantes estructurados PSBE, DFPE, Levenberg-Marquardt y Gauss-Newton, ilustrada en las Tablas 1 y 2, se debe a que se alcanzó el número máximo de iteraciones permitido. Sin embargo, las Tablas 3 y 4 muestran lo que sucede si el número máximo de iteraciones se aumenta suficientemente (o considerablemente).

Tabla 3: Otros resultados de los métodos secantes estructurados, $N = 25{,}000$, para la evaluación de la función seno.

	$N = 25{,}000$	***PSBE***		***DFPE***		***BFGSE***	
nco	Tol	t	n	t	n	t	n
4	10^{-3}	1	152	61	1000	1	69
4	10^{-7}	25	4103	–	–	3	329
5	10^{-3}	36	4225	45	5374	1	116
5	10^{-5}	70	5298	273	+25000	1	153
6	10^{-3}	43	5017	43	5005	1	68
6	10^{-7}	87	1090	–	–	40	4709

Por otra parte, la divergencia del método SR1E se presentó por el mal condicionamiento de las matrices que aproximaban $S(x)$, la parte del Hessiano de E que guarda la información de segundo orden.

Tabla 4: Otros resultados de los métodos de Gauss-Newton (GN), Levenberg-Marquardt (LM) y secante estructurado (BFGSE), $N = 25{,}000$, para la evaluación de la función seno.

	$N = 25{,}000$	***GN***		***LM***		***BFGSE***	
nco	Tol	t	n	t	n	t	n
4	10^{-3}	-	-	1	160	1	69
4	10^{-7}	-	-	23	1395	3	329
5	10^{-3}	-	-	146	665	1	153
5	10^{-5}	-	-	-	-	1	116
6	10^{-3}	172	4230	-	-	40	4709
6	10^{-7}	52	+25000	-	-	1	68

Problema 2: Predicción de consumo eléctrico [8, 15].

Una empresa abastecedora de energía eléctrica, en una población, debe garantizar que el servicio siempre llegue con buena calidad y, de ser posible, a un precio justo. Para que esto ocurra, el servicio debe entregar energía a todos los puntos que lo requieran, mantener los límites de la frecuencia y la tensión con valores dentro de un rango tolerable y operar con costos mínimos, tanto económicos como ambientales. Por tal motivo, es indispensable una planeación exhaustiva del sistema que nos permita, no solo conocer su estado actual en cualquier momento sino también estados futuros, con el fin de no producir en exceso, ya que habría desperdicio del servicio y daños en el medio ambiente; ni producir tan poco, que no sea suficiente para cubrir las necesidades del servicio.

Una de las partes indispensables en esta planeación es la predicción del consumo de carga eléctrica. El interés de esta predicción radica en la necesidad de que las empresas productoras o vendedoras de energía de la región, conozcan con antelación las necesidades de su mercado para poder planear la distribución futura de la energía eléctrica, con el fin de optimizar tanto la producción como su abastecimiento. Por tal motivo, este problema consiste en predecir la demanda de consumo eléctrico en una región para una hora y un día cualquiera, en años futuros.

Para resolver este problema, usamos el modelo propuesto en una investigación realizada en la Universidad Tecnológica de Pereira [15], este modelo consiste en una red neuronal de 4 capas (Figura 4): la capa de entrada, con dos neuronas que corresponden al día y la hora; la capa de salida, con una neurona que corresponde al consumo eléctrico en kilovatios (kw); la primera capa oculta, con doce neuronas, y la segunda capa oculta, con ocho neuronas. Como funciones de activación usaron la tangente sigmoidal (tansig), en ambas capas ocultas, y la identidad (purelin), en la capa de salida, y como algoritmo de entrenamiento, el método de Levenberg-Marquardt que aparece en el Toolbox de MATLAB®.

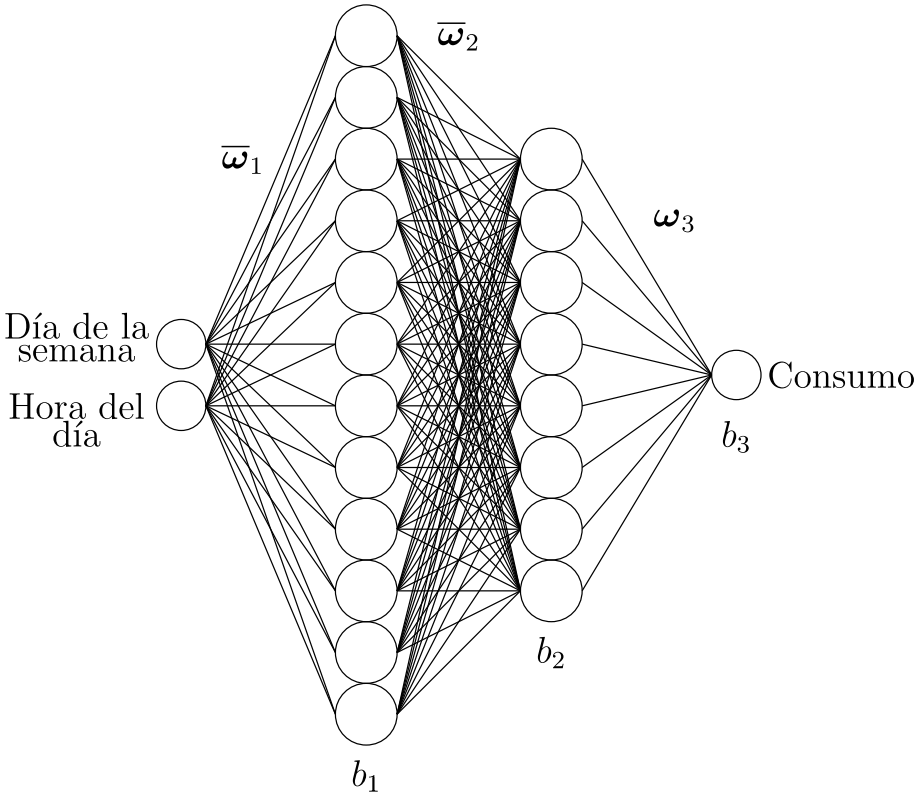

Figura 4: *Perceptrón multicapa para el consumo eléctrico.*

Para generar vectores iniciales, procedimos de la siguiente forma. Inicialmente, generamos aleatoriamente dos vectores $\boldsymbol{\omega}_0 = randn(149, 1)$ y $\boldsymbol{\omega}_1 = randn(149, 1)$. Luego, generamos otros vectores iniciales de la forma $\alpha\boldsymbol{\omega}_0$ y $\mu\boldsymbol{\omega}_1 + \boldsymbol{\omega}_0$, para $\bar{\alpha} = 1,\ 1.2,\ 2,\ 2.5,\ 10,\ -10,\ 10^2$ y $\mu = 1,\ 10,\ 10^2$, respectivamente [14].

En las pruebas numéricas para el Problema 2, usamos 112 datos de entrenamiento extraídos de los datos usados en [15] y los cuales, corresponden a los promedios históricos de consumo eléctrico, en una población de muestra a la que se le hizo un seguimiento, hora a hora, durante una semana.

Los resultados de las pruebas numéricas para el Problema 2, los presentamos en las Tablas 5 y 6. La primera tabla contiene información de los algoritmos secantes estructurados obtenida a partir de puntos iniciales de la forma $\bar{\alpha}\mathbf{w}_0$ y $\mu\mathbf{w}_1 + \mathbf{w}_0$. Exactamente, la primer columna contiene los valores de α y de μ utilizados. Las cuatro columnas siguientes contienen, para cada método secante estructurado, la tolerancia usada (Tol), el tiempo de ejecución (t), medido en segundos y el número de iteraciones (n). La segunda tabla, contiene los resultados de la comparación de los métodos de Gauss-Newton, Levenberg-Marquardt y secante estructurado (BFGSE), a partir de estos puntos iniciales. El símbolo "-" indica que hubo divergencia del método considerado (se excedió el número máximo de iteraciones permitido, ($N = 200$)).

En la Tabla 5, podemos observar que, en general, los 4 métodos secantes estructurados tienen un buen desempeño numérico (similar en todos los casos), en cuanto a tiempo de ejecución y número de iteraciones, excepto, cuando $\mu = 10^2$, y $Tol = 10^{-3}$, caso en el cual, el método PSBE empleó más tiempo y convergió en un número mayor de iteraciones que los otros métodos.

	$N = 200$	***PSBE***		***DFPE***		***BFGSE***		***SR1E***	
$\bar{\alpha}$	Tol	t	n	t	n	t	n	t	n
1.0	10^{-5}	3.5	6	6.9	6	10.3	6	13.7	6
1.2	10^{-2}	2.6	4	5.1	4	7.6	4	10.1	4
1.2	10^{-6}	4.4	8	8.8	8	13.1	8	17.4	8
10	10^{-2}	3.5	6	6.9	6	10.3	6	13.8	6
10	10^{-6}	5.3	10	10.5	10	15.8	10	20.2	8
-10	10^{-2}	3.5	6	6.9	6	10.4	6	13.8	6
-10	10^{-6}	6	10	6	10	6	10	6	8
100	10^{-2}	4.5	8	8.9	8	13.2	8	17.5	8
100	10^{-6}	5.4	10	10.7	10	16.0	10	20.4	8
μ	Tol	t	n	t	n	t	n	t	n
1.0	10^{-2}	2.6	6	6.2	4	9.6	6	12.2	4
1.0	10^{-6}	5.3	10	9.7	8	14.0	8	18.4	8
10	10^{-2}	3.5	6	6.1	4	8.7	4	11.1	4
10	10^{-6}	4.4	8	9.8	10	15.0	10	19.4	8
10	10^{-14}	5	3	8	7	7	5	6	6
10^2	10^{-3}	165	223	6	3	8	7	8	7

Tabla 5: Resultados de los métodos secantes estructurados para el problema del consumo eléctrico

		GN		***LM***		***BFGS***		***BFGSE***	
$\bar{\alpha}$	Tol	t	n	t	n	t	n	t	n
1.2	10^{-2}	-	-	64	112	15.1	4	7.6	4
1.2	10^{-6}	-	-	-	-	29.6	20	13.1	8
10	10^{-2}	-	-	-	-	27.0	24	10.3	6
10	10^{-6}	-	-	-	-	35.4	28	15.8	10
10^2	10^{-2}	-	-	-	-	31.7	26	13.2	8
10^2	10^{-6}	-	-	-	-	36.6	30	16.0	10
μ	Tol	t	n	t	n	t	n	t	n
1.0	10^{-2}	69	118	-	-	28.4	26	12.2	4
1.0	10^{-6}	92	164	-	-	30.1	24	18.4	8
10	10^{-2}	-	-	-	-	21.4	18	11.1	4
10	10^{-6}	-	-	-	-	35.7	30	19.4	8

Tabla 6: Resultados de los métodos Gauss-Newton (GN), Levenberg-Marquardt (LM), BFGS y secante estructurado BFGSE para el problema del consumo eléctrico

Con relación a la Tabla 6, podemos observar, como esperábamos, el buen desempeño numérico del método BFGSE en comparación con los métodos BFGS, Gauss-Newton y Levenberg-Marquardt, en todos los aspectos comparados. Ahora, si solo comparamos los métodos Gauss-Newton y Levenberg-Marquardt, vemos que en casi todos los casos hay divergencia. Esta divergencia está condicionada al número máximo de iteraciones permitido en cada algoritmo (N), tal como sucedió para el Problema 1.

Por otra parte, podemos observar que los resultados obtenidos están acordes con la teoría sobre los métodos estudiados, la cual garantiza que para problemas de gran tamaño, la convergencia de los métodos secantes estructurados, en especial el método BFGSE, en general, es mejor que la de los otros métodos con los cuales hicimos las comparaciones [2].

5 Comentarios finales

El estudio de redes neuronales artificiales constituye en la actualidad un amplio y activo campo en el que pueden interactuar investigadores de muchas y diferentes áreas, para resolver problemas prácticos y útiles tales como control de procesos industriales, reconocimiento de vehículos en los peajes de las autopistas, previsión de consumo eléctrico, entre otros. En este contexto, es quizá el perceptrón multicapa, con su algoritmo de entrenamiento de retropropagación de errores, el modelo neuronal más utilizado.

Métodos numéricos tradicionalmente usados en el entrenamiento supervisado del perceptrón multicapa como por ejemplo Newton, Gauss-Newton y Levenverg-Marquardt requieren del cálculo de la matriz hessiana de la función error; es decir, requieren información de segundo orden, lo que representa, a pesar de las buenas propiedades de los métodos, una desventaja de ellos, ya que los hace inadecuados para problemas con un elevado número de neuronas. En este caso, el cálculo analítico del hessiano es muy difícil o muy costoso, computacionalmente, dado el gran número de operaciones involucradas en el proceso. Una alternativa la representa el método secante estructurado, el cual no requiere explícitamente el cálculo directo de la matriz hessiana de la función a minimizar y además, aprovecha la estructura del problema, sin contar con las buenas propiedades de convergencia que posee.

Motivados por las buenas características del método secante estructurado, en este artículo lo proponemos e implementamos, por primera vez, para el entrenamiento del perceptrón multicapa, y analizamos numéricamente su desempeño comparándolo con los métodos Gauss-Newton y Levenverg-Marquardt. Resultados de pruebas numéricas presentadas indican un buen comportamiento numérico del método propuesto, pero creemos que es necesario realizar más experimentación numérica con diversos problemas de aplicación e introducir otros métodos de globalización, con lo cual se abre la puerta a nuevas investigaciones.

Agradecimientos

Los autores agradecen a la Universidad del Cauca por el tiempo concedido para este trabajo mediante el Proyecto de investigación VRI ID 3908.

Referencias bibliográficas

[1] Caicedo, E. F. y López, J. A. (2009). Una aproximación Práctica a las Redes Neuronales Artificiales. Programa Editorial Universidad del Valle. Primera Edición.

[2] Dennis, J. E. and Schnabel, R. B. (1983). Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, New Jersey.

[3] Dennis, J. E. and Walker, H. F. Covergence theorems for least change secant update methods. SIAM Journal Numerical Analisys 18, 949-987, 19, 443.

[4] Dennis, J. E.; Gay, D. M. and Welsch, R. E. Algorithm 573 NL2SOL-An adaptative nonlinear least squares. TOMS 7, 369-383.

[5] Fletcher, R. (2000). Practical Methods of Optimization. Wiley. Second Edition.

[6] Gavin, P. H. (2013). The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems. Duke University.

[7] Kandel, E. R.; Schwartz, T. H. and Jessel, T. M. (1999). Principles of Neural Science. McGraw-Hill. 4ª Edición.

[8] Martín del Brío, B. y Sanz, A. (2007). Redes Neuronales y Sistemas Borrosos. Alfaomega.

[9] Martínez R., H.J. and Engels J. (1991). Local and superlinear convergence for partially known quasi-Newton methods. Siam Journal on Optimization Vol. 1, No. 1. p.42 - 56.

[10] Martínez R., H.J.; Dennis J. and Tapia R. (1989). Convergence theory for the structured BFGS secant method with an application to nonlinear least squares. Journal of Optimization Theory And Applications. Vol.61 p.161 - 178.

[11] Moré, J. J., Garbow, B. S., and Hillstrom, K. E. (1980). User guide for MINPACK-1. Argonne National Labs Report ANL-80-74.

[12] Nocedal, J. and Wright, S. J. (2006). Numerical Optimization. Springer, Second Edition.

[13] Pérez, R. y Díaz, T. (2010). Minimización sin Restricciones. Editorial Universidad del Cauca.

[14] Vivas, H. (2014). Optimización en entrenamiento del perceptrón multicapa. Tesis de Maestría. Universidad del Cauca, Popayán-Cauca.

[15] http://medicinaycomplejidad.org/pdf/redes/Capitulo3.pdf

Direccion de los autores

Hevert Vivas
Departamento de Matemáticas, Universidad del Cauca, Popayán - Colombia
hevivas@unicauca.edu.co

Héctor Jairo Martínez
Departamento de Matemáticas, Universidad del Valle, Cali - Colombia
hector.martinez@correounivalle.edu.co

Rosana Pérez
Departamento de Matemáticas, Universidad del Cauca, Popayán - Colombia
rosana@unicauca.edu.co