OPTIMIZACIÓN MATEMÁTICA EN SUPERFICIES DE DESEMPEÑO DE REDES NEURONALES ARTIFICIALES

RESUMEN
Varios métodos de optimización son usados en este artículo con el propósito de mostrar su comportamiento sobre diferentes superficies de desempeño y concluir sobre la eficiencia de cada uno de ellos en el entrenamiento de las redes neuronales. Entre estos métodos están: gradiente descendente, Newton Raphson, gradiente conjugado y sus adaptaciones para ser usados en redes multicapa como son los métodos de propagación hacia atrás básico y Levenberg-Marquardt.

PALABRAS CLAVES: Optimización matemática, redes neuronales, superficies de desempeño, propagación hacia atrás.

ABSTRACT
Several optimization methods are used in this article in order to show the behavior on performance surfaces and to conclude about efficiency in neuronal networks training. These methods are: descendent gradient, Newton Raphson, conjugated gradient and its adaptations to multilayer networks: basic backpropagation and Levenberg-Marquardt.

KEYWORDS: Mathematical optimization, neural networks, performance surfaces, backpropagation.

MAURICIO GRANADA E.
Ingeniero Electricista, M.Sc.
Profesor Programa de Ingeniería Eléctrica
Universidad Tecnológica de Pereira
magra@utp.edu.co

ELIANA M. TORO
Ingeniera Industrial, M.Sc.
Profesora Facultad de Ingeniería Industrial
Universidad Tecnológica de Pereira
elianam@utp.edu.co

ANTONIO ESCOBAR Z.
Ingeniero Electricista, M.Sc.
Profesor Escuela de Tecnología Eléctrica
Universidad Tecnológica de Pereira
aescobar@utp.edu.co

1. INTRODUCCIÓN

Las redes neuronales artificiales (RNA) basadas en desempeño o supervisadas utilizan un conjunto de ejemplos que representan el comportamiento esperado de la red. De esta manera, calculando la diferencia existente entre la respuesta a un estimulo y la respuesta esperada (error), es posible tomar decisiones que permitan corregir los parámetros de la RNA a fin de minimizar esta diferencia. El proceso de minimización es llevado a cabo sobre la superficie construida por el valor que toma el error al variar los diferentes parámetros de la RNA. Este espacio de soluciones recibe el nombre de superficie de desempeño y puede obedecer a diferentes definiciones. Sin embargo, es una práctica común usar el error medio cuadrado para tal fin.

La forma de la superficie de desempeño es definida por la configuración de la red. Así, si se considera una red de una capa, la superficie que se obtiene, usando el error medio cuadrado, será una función cuadrática cuyo punto mínimo puede ser encontrado usando aproximaciones lineales o cuadráticas. En el caso de una aproximación lineal el proceso será iterativo y en caso de usar una aproximación cuadrática la convergencia al óptimo se dará en una iteración. A medida que el número de capas aumenta, la superficie de desempeño presentará diversos puntos óptimos. En este caso el objetivo es encontrar el óptimo global.

2 SUPERFICIE DE DESEMPEÑO EN REDES MONOCAPA

Se usa como ejemplo el esquema de cancelación de ruido a través de un filtro adaptivo como el mostrado en la figura 1.

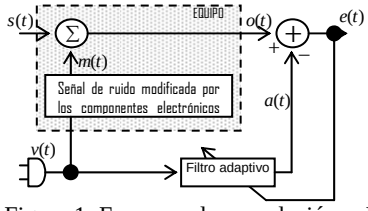


Figura 1. Esquema de cancelación de ruido

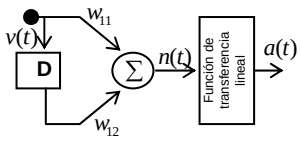


Figura 2. Filtro adaptivo

Este esquema representa un equipo electrónico que capta una señal *s(t)*. El equipo usa como fuente de alimentación una señal de voltaje $v(t) = 1.2 \cdot \sin(2\pi t/3)$ V. La señal captada es contaminada por *m(t)* que corresponde a la señal de la fuente modificada en su amplitud y ángulo debido a los componentes electrónicos del equipo, por lo cual *m(t)* está correlacionada con *v(t)*. El objetivo del filtro adaptivo es generar una señal *a(t)=m(t)*, de forma que al restarla a la señal contaminada *o(t)* se obtenga *s(t)*. Si se considera que no se tiene acceso al interior del equipo, entonces el filtro adaptivo deberá usar la correlación existente entre *v(t)* y *m(t)* para generar la señal *a(t)* requerida. Así, el error medio cuadrado estimado $E(e^2(w))$ podrá ser minimizado.

Fecha de Recepción: 31 Agosto de 2006
Fecha de Aceptación: 25 Octubre de 2006

El estudio del filtro adaptivo requiere de la definición de los siguientes parámetros:

- La estructura de la red es; una neurona, dos entradas y cero bias, como se muestra en la figura 2. Estas condiciones son suficientes para reconocer una señal senosoidal como *v(t)*. $a(t) = w_{11}v(t) + w_{12}v(t-1)$
- Si *t* toma valores enteros, entonces en un período de la señal *v(t)* se podrán realizar 3 muestras (*t = 1, 2, 3*). Así, el valor cuadrado estimado de *v(t)* puede ser calculado como:

$$E\left(v^2(t)\right) = \frac{1}{3}\cdot(1.2)\sum_{t=1}^{3}\sin^2\left(\frac{2\pi t}{3}\right) = 0.72 \quad (1)$$

- *s(t)* es una señal de valores aleatorios uniformemente distribuidos entre *[-0.2 0.2]*. El valor cuadrado estimado de *s(t)* es:

$$E\left(s^2(t)\right) = \frac{1}{0.4}\int_{-0.2}^{0.2} s^2 ds = \frac{1}{0.4}\cdot\frac{1}{3}s^3\Big|_{-0.2}^{0.2} = 0.0133 \quad (2)$$

- La señal *m(t)* se supone atenuada por 10 en su amplitud y desfasada 90º respecto a *v(t)*. Es decir:

$$m(t) = 0.12\cdot\sin\left(\frac{2\pi t}{3} + \frac{\pi}{2}\right) \quad (3)$$

Por lo tanto, el valor cuadrado estimado de m*(t)* es:

$$E\left(m^2(t)\right) = \frac{1}{3}\cdot(0.12)\sum_{t=1}^{3}\sin^2\left(\frac{2\pi t}{3} + \frac{\pi}{2}\right) = 0.0072 \quad (4)$$

- Los pesos iniciales asumidos son $w^0 = [0 \quad -2]$.

Para minimizar $E(e^2(w))$ es conveniente visualizar gráficamente la superficie de desempeño. Analizando la figura 1, se puede deducir que $e(t)=o(t)-a(t)$, por lo cual $E(e^2(w))=E((o(t)-a(t))^2)$. Resolviendo la expresión cuadrática se obtiene: $E(e^2(w))=E(o^2(t)-2.o(t).a(t)+a^2(t))$. Observando la figura 2 se tiene que $a(t)=w^T v(t)$. Por lo tanto:

$$E(e^2(w))=E(o^2(t))-2.w^T.E(o(t).v(t))+w^T E(v^T(t).v(t)).w(t) \quad (5)$$

De la figura 1, también se puede observar que $o(t)=s(t)+m(t)$. Reemplazando esta expresión en (6):

$$E(e^2(w)) = E\left[s^2(t) + \cancelto{0}{2s(t)m(t)} + m^2(t)\right] - \cancelto{0}{2w^T E\left[s(t)v(t)\right]} + 2w^T E\left[m(t)v(t)\right] + w^T E\left[v(t)v^T(t)\right] \quad (6)$$

Los términos llevados a cero en la ecuación anterior corresponden a aquellos que involucran señales no correlacionadas entre sí. Para los otros términos se debe calcular los valores esperados de manera similar a como se hizo en las expresiones (1), (2) y (4): $E\left[m(t)v(t)\right] = 0$, $E(v^2(t-1)) = E(v^2(t)) = 0.72$, $E(v(t)v(t-1)) = -0.36$. Ahora es posible encontrar una función matemática que representa la superficie de desempeño de la red.

$$E(e^2(w)) = 0.0205 + 0.1248w_{12} + 0.72w_{11}^2 + 0.72w_{12}^2 - 0.72w_{12}w_{11} \quad (7)$$

2.1 Optimización matemática.

Al graficar la expresión (7) se obtiene la superficie de desempeño correspondiente al error medio cuadrado estimado, como se muestra en la figura 3.

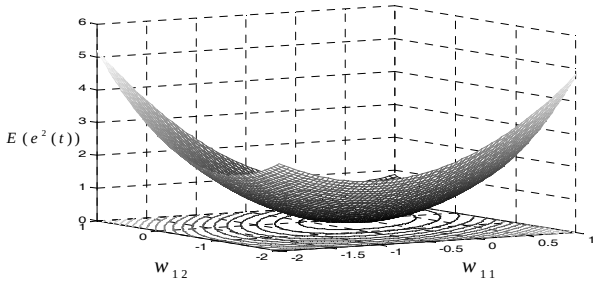


Figura 3. Superficie de desempeño

En este artículo, se presentan 3 algoritmos de optimización matemática que minimizan la función cuadrática de desempeño: método del gradiente descendente básico, método de Newton Raphson y el método de gradiente conjugado. Estos métodos optimizan la función obtenida usando valores estimados (fuera de línea). Sin embargo, al tratarse de un filtro adaptivo se incorpora, al final de esta sección, el método de Widrow-Hoff[3], el cual entrena la red a medida que capta la señal (aprendizaje en línea o adaptivo). Estos algoritmos aseguran encontrar un óptimo global en superficies convexas en donde en un punto óptimo se debe cumplir que el vector gradiente debe ser: $\nabla E(e^2(w)) = 0$.

2.1.1 Gradiente descendente básico.

Este algoritmo consiste en actualizar iterativamente las variables de estado (pesos) usando la dirección negativa del gradiente, dado que el problema es de minimización, hasta que se encuentre un punto óptimo. Usando la serie de Taylor para hacer una aproximación lineal de la función cuadrática que representa el desempeño de la red, se obtiene que las actualizaciones de los pesos son $w^{nuevo} = w^{anterior} - \alpha\nabla E(e^2(w^{anterior}))$, donde α representa el paso exploratorio o rata de aprendizaje de la red y determina la velocidad de convergencia o de entrenamiento de la misma.

Valores muy altos de α pueden, eventualmente, ocasionar inestabilidad en la convergencia. Por otro lado, valores muy pequeños pueden llevar a que el algoritmo consuma tiempos computacionales muy altos debido al

gran número de iteraciones implicadas en el proceso. Este valor, puede permanecer constante durante todo el proceso o puede ser recalculado en cada iteración. En el primer caso, para garantizar estabilidad al momento de escoger un valor constante para α se puede utilizar el criterio $\alpha = 2/\lambda_{\max}$. Donde $\lambda_{\max}$ es el mayor término de la diagonal de la matriz hessiana, es decir el mayor eigenvalor. En el segundo caso es posible usar reglas heurísticas que corrijan en cada iteración el valor de α. Por ejemplo, es posible incorporar un mecanismo que permita usar valores grandes de α al comienzo y que a medida que se avance en el proceso de optimización haga que este valor decrezca. Sin embargo, se puede optimizar el paso exploratorio si se minimiza el índice de desempeño respecto a α_k (*k=1,2..Número* de iteraciones). Haciendo $E(e^2(w)) = F(w)$, se puede obtener el índice de desempeño en función del paso exploratorio, así: $F(w^{k+1}) = F(w^k - \alpha_k \nabla F(w^k))$. El valor de α_k que minimiza la expresión anterior será el que se obtenga al resolver la ecuación:

$$\frac{d}{d\alpha_k}\left(F(w^k - \alpha_k \nabla F(w^k))\right) = 0 \qquad (8)$$

Expandiendo esta expresión en series de Taylor y haciendo una aproximación de segundo orden, se obtiene la siguiente expresión en forma matricial, donde el término $\nabla^2 F(w)^T$ se refiere a la matriz hessiana.

$$\alpha_k^{\acute{o}ptimo} = \frac{\nabla F(w)^T\Big|_{w=w^k} \nabla F(w)\Big|_{w=w^k}}{\nabla F(w)^T\Big|_{w=w^k} \nabla^2 F(w)^T\Big|_{w=w^k} \nabla F(w)\Big|_{w=w^k}} \qquad (9)$$

Ahora es posible retomar nuevamente el estudio de la superficie de desempeño del filtro adaptivo. En la figura 4 (a) se muestra el proceso de convergencia usando el algoritmo de gradiente descendente básico considerando un paso exploratorio constante (línea punteada) y un paso exploratorio óptimo (línea contínua). Para α constante se consideró un valor de 0.25.

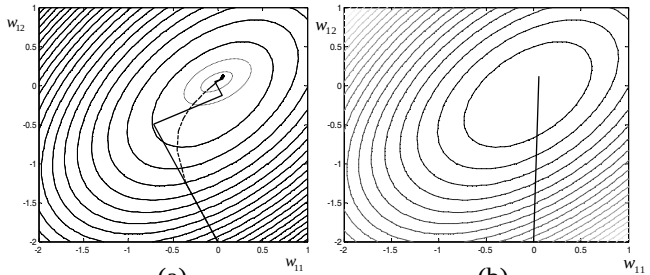


Figura 4. Proceso de convergencia usando el método: (a)gradiente descendente, (b)Widrow-Hoff.

El proceso se ejecutó en un computador Pentium 4 de 2.8 GHz. Las iteraciones requeridas usando α óptimo fueron 14 con un tiempo de $1\mu s$ y con α constante se necesitaron 71 iteraciones y *0.016* segundos.

2.1.2 Newton Raphson.

La función de desempeño corresponde a una función cuadrática. La expansión de la serie de Taylor hasta el término de segundo orden constituye entonces una representación exacta de la función cuadrática y por tal razón la convergencia al punto óptimo deberá realizarse en una iteración. Si la función evaluada es de mayor grado, el método de Newton se vuelve iterativo. Realizando la expansión se obtiene que las actualizaciones de los pesos obedecen a la siguiente expresión:

$$w^{nuevo} = w^{anterior} - \left[\nabla^2 F\left(w^{anterior}\right)\right]^{-1} \cdot \nabla F\left(w^{anterior}\right) \qquad (10)$$

Aunque, en este caso, encontrar los valores óptimos de los pesos requiere de una iteración, el proceso debe también calcular y realizar la inversa de la matriz hessiana, lo cual puede significar un alto esfuerzo y tiempo computacional. Esto implica que el número de iteraciones no es proporcional al tiempo computacional requerido en el proceso de convergencia. La figura 4 (b) muestra el proceso de convergencia en la superficie de desempeño del filtro adaptivo. El proceso se llevo a cabo en *0.0460* segundos en una iteración.

2.1.3 Gradiente conjugado.

Este método es un punto intermedio entre la exactitud del método de Newton y la velocidad de convergencia del gradiente descendente básico, con la ventaja de no tener que calcular e invertir la matriz hessiana [6]. El método consiste en corregir el gradiente usando un proceso similar a la ortogonalización de Gram Schmidt[1]. Conocido un punto inicial w^0 se siguen los siguientes pasos:

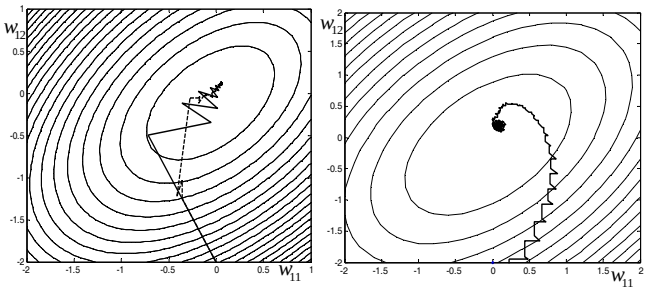
1. Se calcula el gradiente inicial g^0.
2. Se usa el gradiente descendente básico con α óptimo o constante, para encontrar los pesos de la *k-ésima* iteración w_k.
3. Se calcula el gradiente de la *k-ésima* iteración g^k.
4. Se corrige el vector gradiente usando un proceso similar al de Gram Schmidt:

$$g^k_{corregido} = g^k + \beta^k g^{k-1} \text{ con } \beta^k = \frac{\left[g^k\right]^T \cdot g^k}{\left[g^{k-1}\right]^T \cdot g^{k-1}} \qquad (11)$$

5. Se encuentran los pesos de la *k-ésima* iteración w_k usando el gradiente corregido.
6. Si el gradiente corregido es menor que una tolerancia especificada el proceso se detiene y se obtienen los pesos óptimos. Sino, se debe volver al paso 3.

En la figura 5 (a) se muestra el proceso de convergencia usando el algoritmo de gradiente conjugado considerando

un paso exploratorio constante (línea punteada) y un paso exploratorio óptimo (línea contínua). Para α constante se consideró un valor de *0.25*. Las iteraciones requeridas usando α óptimo fueron 48 con un tiempo de 0.022 segundos, mientras que usando un α constante se necesitaron 90 iteraciones y *0.031* segundos. En la figura 6 (a) se muestra (en línea continua) la señal original medida por el dispositivo electrónico y (en línea punteada) la señal descontaminada, obtenida después de que el filtro ha sido entrenado.



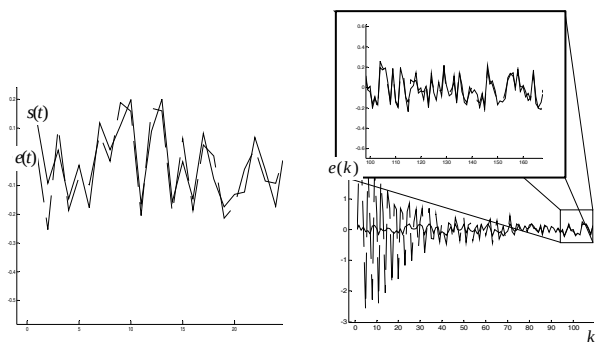
(a) (b)

Figura 5. Proceso de convergencia usando el método: (a)gradiente conjugado, (b) Newton Raphson

2.1.4 Aprendizaje de Widrow-Hoff.

Este algoritmo introduce un importante concepto el cual enuncia que el mínimo error medio cuadrado estimado puede ser encontrado en la *k-ésima* iteración. Es decir, se convierte un proceso basado en valores estimados en uno iterativo. Las actualizaciones de los pesos obedecen a la siguiente expresión:

$$w(k+1) = w(k) - 2\cdot\alpha\cdot e(k)\cdot\left[\nabla F(k)\right]^T \qquad (12)$$



(a) (b)

Figura 6. Comparación entre la señal medida y la señal procesada por el filtro: (a) optimización fuera de línea, (b) optimización adaptiva

El algoritmo consiste en un proceso iterativo donde se presenta a la red cada una de las entradas de forma aleatoria hasta que el error sea menor o igual a una tolerancia especificada. En la figura 5 (b) se muestra el comportamiento de la convergencia usando esta regla y en la figura 6 (b) se aprecia como el error entre la señal de entrada y la señal de salida disminuye a medida que transcurren las iteraciones.

3 SUPERFICIE DE DESEMPEÑO EN REDES MULTICAPA

Para estudiar la superficie de desempeño del error medio cuadrado en redes multicapa se usará un ejemplo de aproximación de una función. La red usada tendrá una configuración de una entrada, dos neuronas en la capa oculta y una neurona en la capa de salida, como se muestra en la figura 7. A fin de simplificar el análisis, la solución óptima del problema será conocida. Es decir, la función que se desea aproximar es la respuesta de la red a los siguiente valores nominales de pesos y bias: $w_{11}^1 = 10$, $w_{21}^1 = 10$, $b_1^1 = -5$, $b_2^1 = 5$, $w_{11}^2 = 1$, $w_{12}^2 = 1$ y $b^2 = -1$. La respuesta nominal de la red es mostrada en la figura 8, para un rango de p variando entre [-2,2]. El objetivo consiste en entrenar la red para aproximar la función mostrada en la figura 8, variando solamente los parámetros w_{11}^1 y w_{11}^2. Sólo se varían dos parámetros a fin de posibilitar la representación grafica del índice de desempeño (ver figura 9).

3.1 ALGORITMO BÁSICO DE PROPAGACIÓN HACIA ATRÁS.

El método de propagación hacia atrás o backpropagation es bastante conocido en el entrenamiento de redes multicapa. Es un método de aprendizaje basado en una generalización del método propuesto por Widrow-Hoff y utiliza un método de gradiente descendente aproximado.

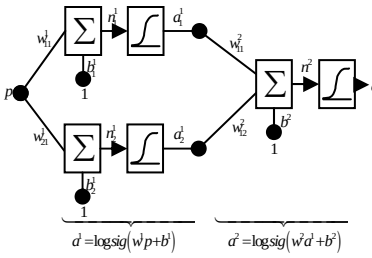


Figura 7. Red de aproximación de función

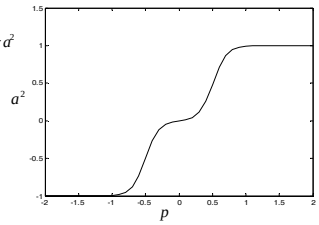


Figura 8. salida nominal de la red

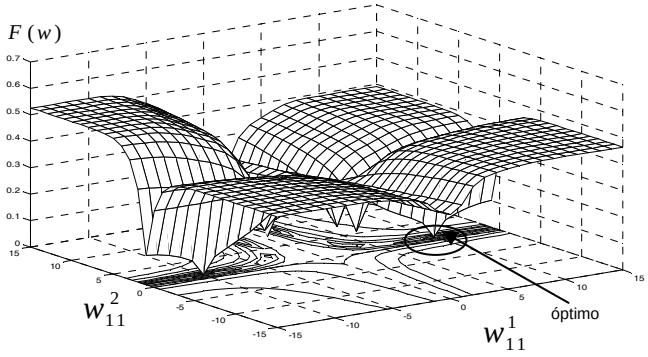


Figura 9. superficie de desempeño del error medio cuadrado

En redes monocapa, la relación entre la salida y la entrada es directa. En redes multicapa con funciones de transferencia no lineal el error total de la red, el cual se calcula en la última capa, sólo tiene relación directa con los pesos de la última capa. Para las otras capas será necesario utilizar una herramienta matemática conocida como regla de la cadena que permite encontrar la relación del error de la red con los pesos de cada capa. Estas relaciones proporcionan información sobre qué tan sensible es el error de la red a cambios en cada uno de los diferentes pesos de cada capa y reciben el nombre de sensibilidades[1]. Este tipo de red permite resolver problemas linealmente no separables, aproximar funciones no lineales de alto orden, resolver problemas complejos de predicción y realizar reconocimiento de patrones difusos con mayor precesión, entre otros. El algoritmo aproximado de gradiente descendente, usando la regla de Widrow-Hoff para redes multicapa, actualiza las variables de estado así:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^m} \qquad b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^m} \qquad (13)$$

Donde *m* indica el número de la capa cuyos parámetros están siendo actualizados. Calcular las derivadas parciales de la expresión (13) resulta complejo y es necesario aplicar la regla de la cadena [4].

3.1.1 Algoritmo

Paso 1: propagación hacia adelante.

$$a^0 = p$$

$$a^{m+1} = f^{m+1}\left(W^{m+1}a^m + b^{m+1}\right) \text{ para } m = 0,1,2,...,M-1$$

$$a = a^M$$

Paso 2: propagación hacia atrás.

$s^M = -2\dot{F}^M\left(n^M\right)(t-a)$ donde *t* es la salida esperada.

$$s^m = \dot{F}^m\left(n^m\right)\left(W^{m+1}\right)^T s^{m+1} \text{ para } m = M-1,...,2,1,0$$

Donde:

$$\dot{F}^m\left(n^m\right) = \begin{bmatrix} \dot{f}^m\left(n_1^m\right) & 0 & \cdots & 0 \\ 0 & \dot{f}^m\left(n_2^m\right) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m\left(n_{S^m}^m\right) \end{bmatrix}$$

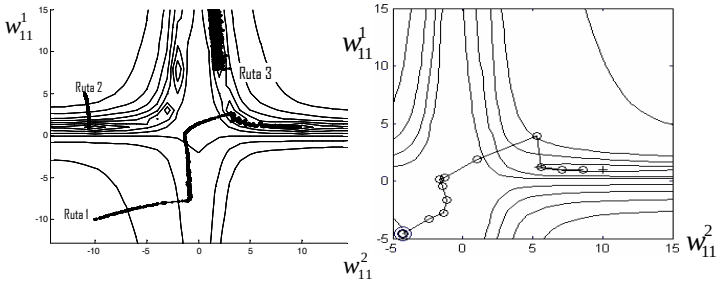
Paso 3: actualización de pesos (gradiente descendente aproximado).

$$W^m(k+1) = W^m(k) - \alpha s^m\left(a^{m-1}\right)^T$$

$$b^m(k+1) = b^m(k) - \alpha s^m$$

Paso 4: criterio de parada. Si $(t-a) < Tolerancia$ el proceso converge y se obtienen los pesos óptimos. Sino, volver al paso 1.

La aplicación del algoritmo anterior al problema de optimización de la figura 9, genera el comportamiento ilustrado en la figura 10 (a). Las rutas corresponden a las trayectorias seguidas por el algoritmo de optimización para diferentes puntos iniciales. La ruta 1 requirió de 14793 iteraciones, las cuales se llevaron a cabo en 7.453 s. La tolerancia usada fue 1e-8 y el paso exploratorio fue α =5. La ruta 1 ilustra el recorrido seguido por el proceso de optimización iniciando con $w_{11}^1 = -10$ y $w_{11}^2 = -10$. La ruta 2 utiliza el punto inicial $w_{11}^1 = -11$ y $w_{11}^2 = 5$ y la ruta 3 utiliza el punto inicial $w_{11}^1 = 3$ y $w_{11}^2 = 8$. Como se puede observar, la condición inicial juega un papel primordial en el proceso de optimización ya que se pueden encontrar diferentes subóptimos. La estrategia más usada para salvar este inconveniente consiste en hacer que el proceso comience en diferentes puntos iniciales y al final se selecciona el óptimo de mejor calidad, es decir, el que más reduzca el índice de desempeño.



(a) (b)

Figura 10. Proceso de convergencia usando: (a)backpropagation básico, (b) Levenberg-Marquardt

3.2 VARIACIONES AL ALGORITMO DE PROPAGACIÓN HACIA ATRÁS

El algoritmo básico de propagación hacia atrás visto en la sección anterior, resulta ser inapropiado para la mayoría de aplicaciones prácticas por su lenta convergencia. Algunas variaciones al algoritmo que mejoran el desempeño se pueden clasificar en heurísticas y en variaciones usando optimización numérica. En este artículo se describe solamente una técnica perteneciente al último grupo.

3.2.1 Algoritmo de Levenberg-Marquardt

Este algoritmo es presentado en la literatura [5] como uno de los más eficientes para entrenar redes neuronales con un número moderado de parámetros de red. Por otro lado, la desventaja del algoritmo radica en el cálculo, almacenamiento e inversión de la matriz Hessiana.

Paso1: Propagación hacia adelante. Se deben presentar todos los patrones de entrenamiento (entradas) a la red y calcular la correspondiente salida.

$a_q^0 = p_q$ para $q = 1, 2, ..., Q$

Q= número total de patrones de entrenamiento

$a_q^{m+1} = f^{m+1}\left(W^{m+1}a_q^m + b^{m+1}\right)$ para $m = 0, 1, ..., M-1$

M = número total de capas

Paso 2: Calcular los errores $e_q = t_q - a_q^M$. Nótese que para cada entrada q existirá un vector de errores de S^M elementos. Por tal razón es posible obtener un arreglo matricial de errores donde $e_{j,q}$ corresponderá al *j-ésimo* elemento del vector de errores de la *q-ésima* entrada.

Paso 3: Calcular el valor del índice de desempeño de la red

$$F(x) = \sum_{q=1}^{Q}\left(t_q - a_q\right)^T\left(t_q - a_q\right) \qquad (14)$$

Paso 4. Calcular cada uno de los elementos de la matriz jacobiana. Esta matriz es de dimensiones H_xL, donde $H=Q_xS^M$ y $L = S^1(R+1) + S^2\left(S^1+1\right) + ... + S^M\left(S^{M-1}+1\right)$. S^M = número de neuronas en la capa M. La estructura de la jacobiana es como sigue:

$$J(x) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{2,1}}{\partial w^1_{1,1}} & \frac{\partial e_{2,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{2,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{2,1}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{s^M,1}}{\partial W^1_{1,1}} & \frac{\partial e_{s^M,1}}{\partial W^1_{1,2}} & \cdots & \frac{\partial e_{S^M,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{S^M,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,2}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,2}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \end{bmatrix}$$

$\left[J\right]_{h,l} = \tilde{s}^m_{i,h} \times a^{m-1}_{j,q}$ para pesos y $\left[J\right]_{h,l} = \tilde{s}^m_{i,h}$ para bias

Las expresiones para calcular las sensibilidades son:

$$\tilde{S}^{\ddot{}M}_q = -\dot{F}^M\left(n^M_q\right), \tilde{S}^m_q = \dot{F}^m\left(n^m_q\right)\left(W^{m+1}\right)^T\tilde{S}^{m+1}_q$$

$$\tilde{S}^m = \left[\tilde{S}^m_1 \mid \tilde{S}^m_2 \mid \cdots \mid \tilde{S}^m_Q\right]$$

Paso 5. Actualizar las variables de estado usando:

$$X_{k+1} = X_k - \left[J^T(x_k)J(x_k) + \mu_k I\right]^{-1}J^T(x_k)e(x_k) \qquad (15)$$

Paso 6. Con los valores obtenidos en la iteración anterior, recalcular la suma de los errores cuadrados (índice de desempeño). El método converge cuando la norma del gradiente $\nabla F(x) = 2J^T(x)e(x)$ es menor que una tolerancia predeterminada. La aplicación del algoritmo anterior al problema de optimización de la figura 9, genera el comportamiento ilustrado en la figura 10 (b). Nótese que el número de iteraciones involucradas es menor (12 iteraciones) así como el tiempo computacional requerido (0.65 s).

4 CONCLUSIONES

El algoritmo de Levenberg-Marquardt tiene un excelente desempeño en el entrenamiento de redes multicapa a pesar que se debe calcular e invertir una matriz Jacobiana de alta dimensionalidad. El principal inconveniente de este algoritmo radica en lo difícil de su implementación.

La mayoría de los métodos aplicados en este trabajo a redes monocapa pueden ser extendidos a redes multicapa siempre y cuando se considere que la relación entre la entrada y la salida de la red multicapa es indirecta y por consiguiente es necesario usar la regla de la cadena.

Uno de los principales aportes introducidos por Widrow-Hoff consiste en convertir un proceso de optimización basado en valores estimados en un proceso iterativo. Se argumenta que el mínimo error estimado puede ser encontrado en la k-ésima iteración.

Las pruebas realizadas, muestran la importancia de utilizar diferentes condiciones iniciales en el proceso de entrenamiento, ya que una superficie de desempeño puede, eventualmente, contener múltiples subóptimos a los cuales el método puede converger.

El algoritmo básico de propagación hacia atrás resulta ser lento e inapropiado para la mayoría de las aplicaciones prácticas. Por tal razón es importante aplicar variaciones heurísticas o basadas en optimización matemática, tales como; método del momento, rata de aprendizaje variable, gradiente aproximado, entre otras.

Para una red con un alto número de capas, el proceso de optimización de los pesos o entrenamiento de la red puede requerir días o incluso semanas en computadores con las características actuales. En este tipo de problemas es posible incorporar técnicas de optimización combinatoria que aceleren la convergencia y garanticen óptimos de buena calidad.

5 BIBLIOGRAFÍA

[1] HAGAN M. T., DEMUTH H. and BEALE M. "Neural Network Design". PWS Publishing Company. Boston U.S.A. 1996.

[2] BATTITI R. "First and second order methods for learning: Between steepest descent and Newton's method", Neural Computation, vol. 4, no. 2, pp. 141-166, 1992.

[3] WIDROW B. y HOFF M. E. "Adaptive switching circuits", IRE WESCON Convention Record, New York: IRE Part 4, pp. 96-104, 1990.

[4] BROGAN W. L. "Modern control theory", 3ra Ed., Englewood Cliffs, NJ: Prentice Hall, 1991.

[5] HAGAN M.T. Y MENHAJ M., "Training feedforward networks with the Marquardt algithm,", IEEE Transactions on Neural Networks, vol. 5, no 6, 1994.

[6] CHARALAMBOUS C., "Conjugate gradient algorithm for efficient training of artificial neural networks", IEE Proceedings, vol. 139, no 3, pp. 301-310, 1992.