Conrprop: un algoritmo para la optimización de funciones no lineales con restricciones

Conrprop: an algorithm for nonlinear optimization with constraints

Fernán Villa, Juan Velásquez[*]*, Patricia Jaramillo*

Universidad Nacional de Colombia. Escuela de Sistemas. Grupo de Estadística Computacional y Análisis de Datos. Carrera 80 N.° 65-223, Bloque M8A, Oficina 201, Medellín, Colombia

(Recibido el 14 de diciembre de 2007. Aceptado el 24 de agosto de 2009)

Resumen

Resilent backpropagation (RPROP) es una poderosa técnica de optimización basada en gradientes que ha sido comúnmente usada para el entrenamiento de redes neuronales artificiales, la cual usa una velocidad por cada parámetro en el modelo. Aunque esta técnica es capaz de resolver problemas de optimización multivariada sin restricciones, no hay referencias sobre su uso en la literatura de investigación de operaciones. En este artículo, se propone una modificación de *resilent backpropagation* que permite resolver problemas no lineales de optimización sujetos a restricciones generales no lineales. El algoritmo propuesto fue probado usando seis problemas comunes de prueba; para todos los casos, el algoritmo de *resilent backpropagation* restringido encontró la solución óptima, y para algunos casos encontró un punto óptimo mejor que el reportado en la literatura.

----- *Palabras clave*: optimización no lineal, restricciones, propagación hacia atrás, rprop

Abstract

Resilent Backpropagation is a gradient-based powerful optimization technique commonly used for training artificial neural networks, which is based on the use of a velocity for each parameter in the model. However, although this technique is able to solve unrestricted multivariate nonlinear optimization problems there are not references in the operations research literature. In this paper, we propose a modification of Resilent Backpropagation that allows us to solve nonlinear optimization problems subject to general nonlinear restrictions. The proposed algorithm is tested using six common used

* Autor de correspondencia. teléfono: + 57 + 4 + 425 53 70, correo electrónico: jdvelasq@unal.edu.co (J. Velásquez)

benchmark problems; for all cases, the constrained resilent backpropagation algorithm found the optimal solution and for some cases it found a better optimal point that the reported in the literature.

-----*Keywords*: nonlinear optimization, restrictions, backpropagation, rprop

Introducción

En el caso más general, las técnicas de optimización buscan encontrar el argumento que minimiza una función $f(\cdot)$:

$$\arg\min f(\vec{x}), \ \vec{x} \in R^n \qquad (1)$$

Sujeta a un conjunto de restricciones descritas como igualdades o desigualdades que deben cumplirse obligatoriamente:

$$h_i(\vec{x}) = 0, \ i=1,\ldots,m \qquad (2)$$

$$g_i(\vec{x}) \leq 0, \ i=1,\ldots,m \qquad (3)$$

Donde $h_i(\cdot)$ y $g_i(\cdot)$ son funciones que permiten describir la región factible donde se debe buscar el mínimo de $f(\cdot)$. Debido a que el problema general definido por (1), (2) y (3) enmarca una gran cantidad de problemas prácticos de diversa índole, se han desarrollado, durante las últimas décadas, un número importante de técnicas de optimización que dependen de la especificación particular que se le de a $f(\cdot)$, $g_i(\cdot)$ y $h_i(\cdot)$ [1].

El problema general definido por (1), (2) y (3) puede ser resuelto mediante métodos determinísticos o estocásticos. Los métodos determinísticos utilizan el gradiente para guiar el proceso de búsqueda, por lo que $f(\cdot)$ debe ser continua y diferenciable. En esencia, la dificultad de la búsqueda del punto de mínima está relacionada con la complejidad de la superficie descrita por $f(\cdot)$, ya que existen puntos de mínima local, regiones de gradiente prácticamente nulo, cráteres, y muchas otras irregularidades que entorpecen el proceso de optimización. El proceso se ve aún más dificultado cuando el punto de óptima debe ser buscado dentro de una región factible, ya que el algoritmo utilizado debe garantizar la exploración dentro de dicha región. La optimización de funciones no lineales utilizando técnicas basadas en el gradiente ha sido un problema tradicional en la investigación de operaciones, y existen técnicas cuyo uso ha sido muy difundido, tales como los métodos de Newton-Raphson y de descenso acelerado.

La estimación de los parámetros de redes neuronales artificiales es un caso particular de (1) en donde se desea obtener los pesos de las conexiones que minimicen una función de error, tal como el error cuadrático medio. La complejidad de este problema esta asociada a que la superficie de error puede llegar a ser bastante compleja e irregular; esta situación ha motivado la utilización de las mismas técnicas generales usadas en investigación de operaciones para la optimización de funciones no lineales, así como también al desarrollo de nuevas metodologías. En [2] se presenta una amplia discusión sobre los problemas que adolecen este tipo de técnicas para el caso particular de la estimación de modelos de redes neuronales, aunque muchas de sus conclusiones pueden ser generalizadas en el contexto general del problema de optimización. Una de las técnicas que ha gozado de bastante aceptación en la comunidad de Inteligencia Computacional para la estimación de modelos de redes neuronales es el algoritmo RPROP (*Resilient Backpropagation*), que es una técnica basada en el gradiente de la función de error [3, 4]. Aunque RPROP puede ser utilizado para resolver problemas de optimización multivariable no restringida, no existen referencias sobre su uso en el contexto general del problema de optimización. Más aún, esta técnica no permite el manejo de restricciones, limitando su uso en casos prácticos.

Este artículo tiene dos objetivos: primero, presentar una modificación de RPROP para incorporar el manejo de restricciones a través del uso de funciones de penalización; y segundo, introducir el algoritmo propuesto dentro de las técni-

cas propias de la comunidad de investigación de operaciones.

Para cumplir con los objetivos propuestos, el resto de este artículo está organizado así: en la siguiente sección se discute la versión original de RPROP; seguidamente, se introduce un esquema para el manejo de restricciones a partir de la función de penalización; posteriormente, se utilizan varias funciones benchmark para comprobar la efectividad del algoritmo propuesto; y finalmente, se concluye.

RPROP (Resilient Backpropagation)

RPROP [3, 4] es una técnica ampliamente utilizada para el entrenamiento supervisado de redes neuronales artificiales tipo perceptrón multicapa, cuyo proceso de búsqueda es guiado por la primera derivada de $f(\cdot)$; en este caso, $f(\cdot)$ es una medida de la diferencia entre la salida arrojada por la red neuronal y el valor esperado. RPROP, y su variante iRprop+ por Igel y Hüsken [5], difiere de la técnica clásica de propagación hacia atrás del error (o algoritmo backpropagation) en que las derivadas parciales de la función error sólo son usadas para determinar el sentido en que deben ser corregidos los pesos de la red pero no las magnitudes de los ajustes. Los algoritmos basados en backpropagation modifican los valores de los parámetros proporcionalmente al gradiente de la función de error, de tal forma que en regiones donde el gradiente tiende a ser plano el algoritmo avanza lentamente; esta modificación se hace a través de un único parámetro que controla la velocidad de avance del algoritmo. RPROP utiliza parámetros independientes que controlan la velocidad con que se recorre la función objetivo para cada uno de los pesos de la red neuronal. RPROP tampoco se ve afectado por la saturación de las neuronas de la red neuronal, ya que solamente se usa la derivada para determinar la dirección en la actualización de pesos. Consecuentemente, converge más rápidamente que los algoritmos basados en backpropagation.

En ésta investigación se utilizó la variante iRprop+, en la cual los parámetros de la red neuronal en la iteración $k + 1$ son actualizados como:

$$w_{ij}^{(k+1)} \leftarrow w_{ij}^{(k)} + \Delta w_{ij}^{(k)} \tag{4}$$

donde Δw_{ij} es estimado con una función del cambio de signo de la derivada del error entre las iteraciones k y $k-1$, y del tamaño del paso Δ tal como se realiza tradicionalmente en las técnicas basadas en el gradiente. Así, si el signo de la derivada no cambia en las dos últimas iteraciones, entonces el tamaño de paso Δ es incrementado en un factor η^+ pero limitado a que su valor máximo no supere Δ_{max} que corresponde al tamaño máximo de la modificación de w_{ij}.

Cuando se presenta el cambio de signo de las derivadas el algoritmo sobrepaso el punto de mínima; consecuentemente, el tamaño de paso es reducido en un factor η^- pero limitando el tamaño mínimo de modificación de w_{ij} a un valor Δ_{max}. Si la derivada es cero no se modifica el tamaño de paso.

Función de penalización

Con el fin de aprovechar las ventajas mencionadas y el rendimiento que ofrece la técnica [6] en la optimización de perceptrones multicapa, se propone el algoritmo CONRPROP (constrained RPROP) como una adaptación del algoritmo iRprop+ que permite minimizar funciones no lineales sujetas a restricciones como las descritas en (2) y (3). Para ello es necesario que RPROPCON incorpore, además de la técnica de Rprop, una técnica de penalización para convertir el problema de optimización restringida en uno sin restricciones.

Hoffmeister y Sprave [7] proponen introducir una penalización dentro de la función objetivo $f()$ que permita medir el grado en que las restricciones descritas por (2) y (3) son violadas. Así, RPROP debe operar sobre una nueva función $F()$ definida como:

$$F(x) = \begin{cases} f(x) & x \in \Omega \\ M_c + d_c & x \notin \Omega \end{cases} \tag{5}$$

donde Ω representa la región factible; M_c es el máximo conocido en la región factible; y d_c es

una medida de penalización de las restricciones que no se cumplen. En esta implementación, d_c se define como:

$$d_c = \max\{H_1 \times |R_1|, \ldots, H_m \times |R_m|\} \qquad (6)$$

donde H_i es una variable que toma el valor de uno si la i-ésima restricción ha sido violada y cero en caso contrario; y R_i es el valor que toma $g_i(x)$ o $h_i(x)$ en el punto x. La versión propuesta es presentada en la figura 1.

Según (5), $F()$ es evaluada al valor máximo de f() en la frontera de la región factible causando una discontinuidad; así, cualquier punto al interior de la región factible será preferido a un punto en la frontera. Si el punto actual está por fuera de la región factible, la penalización d_c obliga a que el algoritmo se dirija hacia la región factible; una vez dentro de ella, iRprop+ opera de la forma tradicional. El efecto de la penalización es esquematizado en la figura 2,

for each x *in* $\vec{x}$ *do*

if $\left(\frac{dF(x)}{dx}\right)^{(k-1)} \cdot \left(\frac{dF(x)}{dx}\right)^{(k)} > 0$ *then*

$\Delta^{(k)} \leftarrow min\left(\Delta^{(k-1)} \cdot \eta^{+}, \Delta_{max}\right)$

$\Delta x^{(k)} \leftarrow \left(-sign\left(\left(\frac{dF(x)}{dx}\right)^{(k)}\right) \cdot \Delta^{(k)}\right)$

$x^{(k+1)} \leftarrow x^{(k)} + \Delta x^{(k)}$

elseif $\left(\frac{dF(x)}{dx}\right)^{(k-1)} \cdot \left(\frac{dF(x)}{dx}\right)^{(k)} < 0$ *then*

$\Delta^{(k)} \leftarrow max\left(\Delta^{(k-1)} \cdot \eta^{-}, \Delta_{min}\right)$

if $F(x)^{(k)} > F(x)^{(k-1)}$ *then* $x^{(k+1)} \leftarrow x^{(k)} - \Delta x^{(k+1)}$

$\left(\frac{dF(x)}{dx}\right)^{(k)} \leftarrow 0$

elseif $\left(\frac{dF(x)}{dx}\right)^{(k-1)} \cdot \left(\frac{dF(x)}{dx}\right)^{(k)} = 0$ *then*

$\Delta x^{(k)} \leftarrow \left(-sign\left(\left(\frac{dF(x)}{dx}\right)^{(k)}\right) \cdot \Delta^{(k)}\right)$

$x^{(k+1)} \leftarrow x^{(k)} + \Delta x^{(k)}$

end if

do until (*Converged*)

Figura 1 Algoritmo CONRPROP

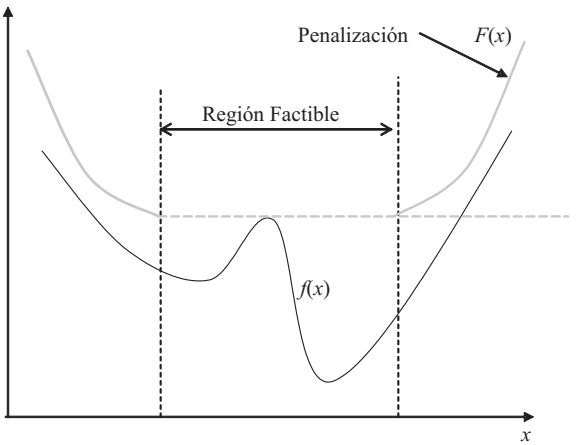

Figura 2 Efecto de la penalización en la evaluación de la función objetivo

Resultados experimentales y comparación

Para comprobar la efectividad del algoritmo CONRPROP se realizaron una serie de pruebas con los problemas de optimización presentados en la Tabla 1; cada uno de ellos presenta una relativa complejidad y han sido comúnmente usados en la literatura para la comparación de técnicas de optimización. El problema 1 posee dos restricciones: una inecuación no lineal y una ecuación lineal, fue abordado en [8]. El 2 no posee restricciones [9]; el 3 fue diseñado con un punto estacionario no óptimo en $f(\vec{x}) \approx 8$, esto puede causar una convergencia prematura [10]; en el 4 la función objetivo fue definida fuera de la región factible [11]; el 5 está sujeto a dos restricciones: una inecuación no lineal y otra lineal [8]; y el problema 6 posee 5 variables, está sujeto a 6 inecuaciones no lineales y 10 límites sobre las variables independientes, nótese que en la función objetivo los coeficientes de x_2 y x_4 son cero, es decir, x_2 y x_4 no están incluidos en la definición de $f(\vec{x})$.

En la tabla 1, se presentan los resultados experimentales obtenidos, estos muestran que el desempeño al resolver problemas de programación no lineal con CONRPROP es satisfactorio. En la mayoría de los problemas, el resultado de la optimización es mejor que el reportado por otros autores; este es el caso de los problemas 1, 5 y 6 donde el punto óptimo es menor que el reportado.

Para los problemas 2 y 3 los puntos obtenidos son iguales a los alcanzados por otras técnicas. Por otro lado, aunque para el problema 4 el óptimo logrado es mayor que el reportado, la aproximación obtenida es muy cercana. Cabe anotar que en el problema 3, CONRPROP superó el punto estacionario no óptimo y llegó al óptimo global, mientras que para el 4, donde la función fue definida fuera de la región factible, RPROPCON encontró un punto muy cercano al reportado.

Tabla 1 Soluciones obtenidas usando RPROPCON

	Problema	***Punto de inicio***		***Resultado reportado***	***RPROP***
1.	Minimizar:	No Factible	$f(\vec{x}^*)$	1,3930	1,3910
	$f(\vec{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$	$\vec{x}^{(0)} = [2{,}0 \quad 2{,}0]^T$	x_1^*	0,8230	0,8239
	Sujeto a:	$f(\vec{x}^{(0)}) = 1$			
	$h_1(\vec{x}) = x_1 - 2x_2 + 1 = 0$		x_2^*	0,9110	0,9118
	$g_1(\vec{x}) = \frac{x_1^2}{4} + x_2^2 - 1 \le 0$				
2.	Minimizar:	$\vec{x}^{(0)} = [-1{,}2 \quad 1{,}0]^T$	$f(\vec{x}^*)$	0,0000	0,0000
	$f(\vec{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$f(\vec{x}^{(0)}) = 24{,}20$	x_1^*	1,0000	1,0000
			x_2^*	1,0000	1,0000
3.	Minimizar:	Factible	$f(\vec{x}^*)$	0,0000	0,0000
	$f(\vec{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$\vec{x}^{(0)} = \begin{bmatrix} -3 \\ -1 \\ -3 \\ -1 \end{bmatrix}$	x_1^*	1,0000	1,0000
	$+90(x_4 - x_3^2)^2 + (1 - x_3)^2$		x_2^*	1,0000	1,0000
	$+10{,}1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right]$		x_3^*	1,0000	1,0000
	$+19{,}8(x_2 - 1)(x_4 - 1)$		x_4^*	1,0000	1,0000
	Sujeto a:	$f(\vec{x}^{(0)}) =$			
	$-10 \le x_i \le 10 \qquad i = 1, 2, 3, 4$	$19{,}192$			
4.	Minimizar:	$\vec{x}_i^{(0)} = 9$	$f(\vec{x}^*)$	-45,7780	-45,7785
	$f(\vec{x}) = \sum_{i=1}^{10}\left\{\left[\ln(x_i - 2)\right]^2 + \left[\ln(10 - x_i)\right]^2\right\}$	$i = 1, \ldots, 10$	x_1^*	9,3510	9,3503
		$f(\vec{x}^{(0)}) =$	x_2^*	9,3510	9,3503
	$-\left(\prod_{x=1}^{10} x_i\right)^{0,2}$	$-43{,}1343$	x_3^*	9,3510	9,3503
			$\vdots$	$\vdots$	$\vdots$
	Sujeto a:		x_{10}^*	9,3510	9,3503
	$2{,}001 \le x_i \le 9{,}999 \qquad i = 1, \ldots, 10$				

Tabla 1 continuación

	Problema	Punto de inicio		Resultado reportado	RPROP
5.	Minimizar $f(\vec{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$ Sujeto a: $g_1(\vec{x}) = x_1^2 - x_2^2 \leq 0$ $g_2(\vec{x}) = x_1 + x_2 - 2 \leq 0$	No Factible $\vec{x}^{(0)} = [2{,}0 \quad 2{,}0]^T$ $f(\vec{x}^{(0)}) = 1$	$f(\vec{x}^*)$	1,0000	0,9989
			x_1^*	1,0000	1,0005
			x_2^*	1,0000	0,9997
6.	Minimizar: $f(\vec{x}) = 5{,}3578547\, x_3^2 + 0{,}8356891 x_1 x_5 + 37{,}293239 x_1 - 40.792{,}141$ Restricciones: $0 \leq 85{,}334407 + 0{,}0056858\, x_2 x_5 + 0{,}0006262\, x_1 x_4 - 0{,}0022053\, x_3 x_5 \leq 92$ $90 \leq 80{,}1249 + 0{,}0071317\, x_2 x_5 + 0{,}0029955\, x_1 x_2 + 0{,}0021813\, x_3^2 \leq 110$ $20 \leq 9{,}300961 + 0{,}0047026\, x_3 x_5 + 0{,}0012547\, x_1 x_3 + 0{,}0019085\, x_3 x_4 \leq 110$ $78 \leq x_1 \leq 102$ $33 \leq x_2 \leq 45$ $28 \leq x_3 \leq 45$ $27 \leq x_4 \leq 45$ $27 \leq x_5 \leq 45$	Factible $\vec{x}^{(0)} = \begin{bmatrix} 78{,}62 \\ 33{,}44 \\ 31{,}07 \\ 44{,}18 \\ 35{,}22 \end{bmatrix}$ $f(\vec{x}^{(0)}) = -30.665{,}50$	$f(\vec{x}^*)$	-30.665,50	-30.531,00
			x_1^*	78,000	78,280
			x_2^*	33,000	33,440
			x_3^*	29,995	30,730
			x_4^*	45,000	45,001
			x_5^*	36,776	34,880

Conclusiones

En éste trabajo se presentan los resultados obtenidos al resolver problemas de optimización no lineal con restricciones usando una técnica propia de la optimización de redes neuronales, RPROP. Las contribuciones de este trabajo están relacionadas con los siguientes aspectos: Se propone CONRPROP como una adaptación y generalización de la técnica de RPROP, incorporando una función de penalización para resolver problemas de optimización no lineal. Dados los resultados presentados, se comprobó experimentalmente que la efectividad de RPROPCON al resolver problemas de programación no lineal es satisfactoria.

Agradecimientos

Los autores agradecen a los evaluadores anónimos cuyos comentarios permitieron mejorar la calidad de este artículo.

Referencias

1, P. M. Pardalos, M. G. C. Resende. *Handbook of Applied Optimization*. Ed. Oxford University Press. New York. 2002. pp. 263-299.

2, Y. LeCun, L. Bottou, B. Orr, K. R. Muller. "Efficient Backprop". *Neural Networks - Tricks of the Trade, Springer Lecture Notes in Computer Sciences*. Vol. 1524. 1998. pp. 5–50,

3. M. Riedmiller, H. Braun. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm". *Proceedings of the IEEE International Conference on Neural Networks*. San Francisco (CA). 1993. pp. 586-591.

4. M. Riedmiller. "Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms". *Computer Standards and Interfaces*. Vol. 16. 1994. pp. 265-278.

5. C. Igel, M. Hüsken. "Improving the Rprop learning algorithm". *Proceedings of the Second International Symposium on Neural Computation, NC2000,* ICSC Academic Press. Berlín. pp. 115-121.

6. D. Ortíz, F. Villa, J. Velásquez. "A Comparison between Evolutionary Strategies and RPROP for Estimating Neural Networks". *Avances en Sistemas e Informática*. Vol.4. 2007. pp. 135-144.

7. F. Hoffmeister, J. Sprave. "Problem-independent handling of constraints by use of metric penalty functions". L. J. Fogel, P. J. Angeline, T. Bäck (Editores). *Proceedings of the Fofth Annaul Conference on Evolutionary Programming. (EP'96)*. San Diego (CA). MIT press. 1996. pp. 289-294.

8. J. Bracken, G. McCormick. *Selected Applications of Nonlinear Programming*. Ed. Jhon Wiley & Sons. Inc. New York. 1968. pp. 16.

9. H. Rosenbrock. "An Automatic Method for Finding the Greatest and Least value of a Function". *Computer Journal*. Vol. 3. 1960. pp. 175-184.

10, A. R. Colville. *A comparative study on nonlinear programming codes.* IBM Scientific Center Report No. 320-2949. New York. 1968. pp. 12.

11, D. Paviani. *A new method for the solution of the general nonlinear programming problem*. Ph.D. dissertation. The University of Texas. Austin. Texas. 1969. pp. 18.