Artículo de Investigación. Revista Ciencia e Ingeniería. Vol. 39, No. 1, pp. 13-26, diciembre-marzo, 2018.
ISSN 1316-7081. ISSN Elect. 2244-8780. Universidad de Los Andes (ULA)

Algoritmos de Retropropagación con restricciones para la estimación de parámetros de curvas de titulación

Constrained Backpropagation algorithms for titration curves parameter estimation

Hernández, Jairo [1]***; Rodríguez, José** [2]
[1] Departamento de Evaluación, NER 489
Pregonero 5058, Venezuela.
[2] Decanato de Investigación, Universidad del Táchira
San Cristóbal 5001, Venezuela
*jairojhernandez@hotmail.com

Resumen

En este artículo se propone el diseño y programación en Matlab de los algoritmos de aprendizaje de Retropropagación: Gradiente Conjugado, Quasi-Newton DFP (Davidon, Fletcher y Powell) y BFGS (Broyden, Fletcher, Goldfard y Shanno) , con restricciones de no negatividad, con la finalidad de estimar pesos de una red neuronal de tres capas, tipo "caja gris", correspondiente a la ecuación de balance de un proceso de titulación de ácidos y bases. La entrada es el pH de la mezcla, la salida (r) es la razón entre el flujo del reagente (titulante) y el del influente (titulado), mientras que los pesos a estimar son las concentraciones de las sustancias presentes en el influente; debido a esto último es precisamente que los pesos no pueden ser negativos. El conjunto de datos ($\mathbf{pH}$,$\mathbf{r}$), son generados computacionalmente. Los algoritmos mencionados se comparan entre sí y también con la función predefinida de Matlab $lsqnonneg$. Los resultados obtenidos muestran que $lsqnonneg$ es el más rápido en la estimación de los referidos pesos o parámetros. En cuanto a la calidad en la precisión, se puede decir que los algoritmos propuestos toman mayor preponderancia respecto a $lsqnonneg$ a medida que algunas características de los datos suministrados tienden a ser más complejas, tales como escasa muestra de datos para el entrenamiento o alta presencia de ruido.

Palabras clave: Red neuronal artificial, caja gris, estimación de parámetros, Retropropagación con restricciones , curva de titulación.

Abstract

This article describes the development and Matlab programming of auto-learning backpropagation algorithms: Conjugate Gradient, Quasi-Newton DFP (Davidon, Fletcher and Powell) and BFGS (Broyden, Fletcher, Goldfarb and Shanno), with non-negativity constraints, in order to estimate weights in a "gray box" three-layer neural network describing the balance equation for a acid-base titration process. The input is pH, the output (r) is the ratio of the reagent flow (titrant or titrator) and influent flow (analyte or titrand), while weights represent concentration of the substances in the influent estimates; because of this last is precisely that the weights can not be negative. The dataset ($\mathbf{pH}$,$\mathbf{r}$) are generated computationally. The aforementioned algorithms are compared with each other and with $lsqnonneg$ a predefined function in Matlab. The results show that $lsqnonneg$ is faster in parameter estimation. As for accuracy, it can be said that the proposed algorithms take on a greater preponderance with respect to $lsqnonneg$ as some characteristics of the supplied data tend to be more complex, such as few data samples for training or high presence of noise.

Key words: Artificial neural network, gray box, parameter estimation, constrained Backpropagation, titration curve.

1 Introducción

Los algoritmos de aprendizaje de Retropropagación que comprenden, según (Hudson y col., 2013), a cualquier método de aprendizaje supervisado que se base en el cálculo del vector gradiente o de la matriz jacobiana, y por tanto en el cálculo de la matriz hessiana, son en esencia métodos de optimización. Estos algoritmos, que tienen como principio de funcionamiento el aprendizaje de la función subyacente a una serie de datos dados (ejemplos de entrenamiento), son diseñados e implementados en esta investigación para estimar los parámetros desconocidos de dichas funciones, bajo la restricción de que éstos sean necesariamente no negativos.

La Red Neuronal Artificial (RNA) propuesta bajo este tipo de algoritmos de aprendizaje, surge de la necesidad de determinar completamente el modelo matemático (con todos sus parámetros) que describe un proceso de mezcla de ácidos y bases, necesario para implementar en la práctica el control del pH de un fluido principal, denominado influente, agregándole a éste otro fluido de naturaleza distinta, generalmente de menor flujo y de mayor concentración, denominado reagente, tal como se muestra en la siguiente figura:

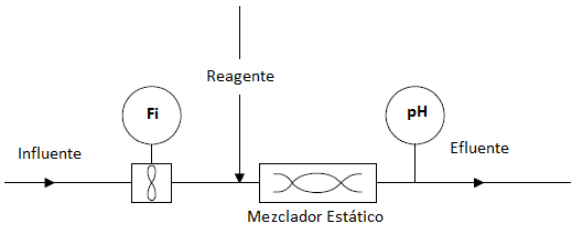

Fig. 1. Arreglo "En linea" (Extraida de (Rodríguez, 1999))

Para estimar a partir de la mencionada RNA los parámetros desconocidos de una función determinada, se procura que la estructura de dicha red represente esencialmente a la ecuación del proceso físico involucrado, bajo una concepción fenomenológica conocida como "caja gris", en alusión a su funcionamiento mixto, donde se cuenta con el modelo matemático del proceso ("caja blanca"), pero cuyos parámetros deben ser determinados a partir de patrones de entrenamiento basados en entradas y salidas a la RNA ("caja negra"). Bajo este enfoque, en (Cruz y col., 2011) se obtienen buenos resultados en comparación con modelos neuronales convencionales (tipo "caja negra"), debido al aprovechamiento del conocimiento a priori del modelo matemático; mientras que en (Xiong y col., 2001) se resalta la mayor rapidez de convergencia de este tipo de metodología.

En esta investigación se cuenta con la ecuación de balance de un proceso de titulación de ácidos y bases, extraída de (Rodríguez 1999), cuya representación gráfica se conoce como curva de titulación (aunque en adelante a la ecuación de balance también se le denominará curva de titulación), procurándose que la estructura de la red represente la curva de titulación, fundamentalmente la parte de dicha ecuación dependiente de parámetros desconocidos, aprovechando la similitud matemática de la función de activación de la red neuronal con el factor de disociación propio de la ecuación de balance químico.

Debido a lo recientemente mencionado, los componentes de la red tendrán un significado físico; así, por ejemplo, los pesos a la salida de la red **w** (parámetros a estimar) representan los valores de las concentraciones (desconocidas) de las diferentes sustancias presentes en el influente (**Ci**), la entrada es el nivel de *pH* de la mezcla y la salida es *r* (razón entre el flujo del reagente y el del influente). A continuación se muestra la estructura básica de la RNA propuesta, cuya nomenclatura será aclarada más adelante.

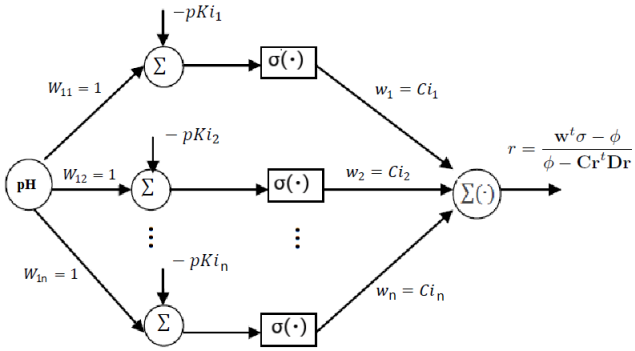

Fig. 2. RNA para la aproximación de curvas de titulación

De esta manera, la sumatoria a la salida de la red sería de la forma $\sum(\cdot)=\mathbf{w}^t\sigma$. Sin embargo, se puede considerar, con fines prácticos, a r como la salida real, puesto que $\phi = 10^{-pH} - 10^{pH-14}$, dependiendo únicamente del pH, que es el valor de entrada a la red, y por tanto resulta conocido; mientras que la expresión $\mathbf{Cr}^t\mathbf{Dr}$ depende de los datos para el reagente, el que se considera, en todo caso, completamente determinado.

La función de activación, expresión que a su vez coincide con lo que luego se conocerá como el factor de disociación para el influente (Di), vendría dada para cada componente como:

$$\sigma_j = \frac{1}{1+10^{pKi_j-pH}} \tag{1}$$

para j desde 1 hasta n (número de pesos a la salida), donde pKi es la constante de disociación de las sustancias en el influente.

Si se hace $x_j = pH - pKi_j$, se obtiene una función de activación equivalente a la logística (con exponencial de base 10):

$$\sigma(x_j) = \frac{1}{1+10^{-x_j}} \tag{2}$$

Resulta importante resaltar, tal como se puede observar en la figura 2, que existe una correspondencia entre las componentes de los vectores $\mathbf{pKi}$ y $\mathbf{Ci}$, ambos de tamaño n (número de potenciales sustancias presentes en el influente, físicamente hablando); es decir, la concentración Ci_j es la de una sustancia indeterminada del influente con constante de disociación pKi_j. El vector $\mathbf{pKi}$ se supondrá con el valor de sus componentes uniformemente distribuidas entre -3 y 15 (rango en el que se encuentran la mayoría de pKi en la práctica) y tan cercanos entre sí como lo permita n. Mientras que los valores de Ci, como ya se ha explicado antes, se han de estimar.

De esta manera, cada vez que se aumente el número de pesos de la RNA, se estará, en realidad, aumentando el número de posibles sustancias presentes en el influente. Adicionalmente, se puede inferir que si para un valor de pKi determinado, al estimar su correspondiente Ci, ésta vale cero, significa que no existe en el influente ninguna sustancia con tal valor de pKi. Desde otro punto de vista, las concentraciones con valores mayores a cero permiten identificar los componentes químicos del influente, lo que puede ser de mucha utilidad para fines de protección ambiental.

En concreto, la RNA planteada busca describir, en esencia, la curva de titulación, que expresada en términos matemáticos sería:

$$r = \frac{\mathbf{w}^t\sigma - \phi}{\phi - \mathbf{Cr}^t\mathbf{Dr}} = \frac{\mathbf{Ci}^t\mathbf{Di} - \phi}{\phi - \mathbf{Cr}^t\mathbf{Dr}} \tag{3}$$

Donde:
r: razón del volumen de reagente añadido por unidad de volumen del influente.
$\mathbf{Ci}^t$: transpuesta del vector de las concentraciones del influente.
$\mathbf{Cr}^t$: transpuesta del vector de las concentraciones del reagente.
$\mathbf{Di}$: vector de los factores de disociación del influente.
$\mathbf{Dr}$: vector de los factores de disociación del reagente.
$\phi = 10^{-pH} - 10^{pH-14}$

Ahora, según (Rodríguez 1999), los factores de disociación se calculan por cada componente, de la forma: $Di = 1/(1+10^{pKi-pH})$ y $Dr = 1/(1+10^{pKr-pH})$, siendo pKi y pKr las constantes de disociación del influente y reagente, respectivamente. Por razones prácticas, se ha expresado la ecuación (3) de tal forma que la entrada (implícitamente) sea el pH y la salida r, en contraste con la apariencia gráfica convencional de las curvas de titulación.

Por otra parte, debido a que los parámetros a estimar equivalen a las concentraciones de las sustancias presentes en el influente, éstos no pueden ser negativos (físicamente carecería de sentido). Bajo tal restricción, estos parámetros son estimados en (Rodríguez 1999) haciendo uso de la función de Matlab NNLS (siglas en inglés para mínimos cuadrados no negativos); siendo dicho trabajo el que presta su contexto a la presente investigación, la cual no tendrá por meta, en sí misma, la estimación de las mencionadas concentraciones, sino que más bien se usará tal contexto de optimización de parámetros químicos como criterio para verificar comparativamente la funcionalidad de los algoritmos aquí propuestos.

En otras palabras, la idea fundamental en este trabajo es deducir y programar (en Matlab) una serie de algoritmos de Retropropagación para una RNA de tipo caja gris, con restricciones de no negatividad en sus pesos, a los fines de comparar su desempeño computacional, en cuanto a eficiencia y precisión, con el de $lsqnonneg$ de Matlab (versión alternativa a NNLS, destinada igualmente a la estimación de parámetros no negativos) y entre sí. Para facilitar esta comparación se programó una interfaz de usuario denominada $rnaph$, que a su vez ejecuta un simulador de curvas de titulación ($tituinv$), validado experimentalmente e implementado en (Rodríguez 1999). Los algoritmos de aprendizaje restringido aquí desarrollados son: Gradiente Conjugado no negativo ($gcnn$), Quasi-Newton DFP no negativo ($qndfpnn$) y Quasi-Newton BFGS no negativo ($qnbfgsnn$).

En lo que respecta a la comparación del desempeño computacional entre las RNA y algunos métodos estadísticos o de investigación operativa ya establecidos (o algunos híbridos), se le puede considerar un tarea de gran interés en la actualidad, a los fines de plantear posibles mejoras de las RNA, siendo éstas unas herramientas bastante prometedoras en el campo de la inteligencia artificial y que se encuentran aún en pleno desarrollo. En este sentido, en (Jabbour y col., 2014, Jabbour y col., 2009, Conti y col., 2005) se han llevado a cabo algunas de tales comparaciones.

Lo que resta de este artículo ha sido organizado de la siguiente forma: *Sección 2*, cálculos y demostraciones previas; *Sección 3*, algoritmos de Retropropagación restringidos; *Sección 4*, simulaciones y resultados y *Sección 5*, conclusiones y recomendaciones.

2 Cálculos y demostraciones previas

A continuación se plantean una serie de resultados previos al desarrollo de los algoritmos de aprendizaje (en lote) para la red neuronal artificial propuesta.

2.1 Función Error o de Costo

Mediante el método de Retropropagación, en cualquiera de sus versiones aquí desarrolladas, se busca minimizar, para los N patrones de entrenamiento, la diferencia entre la salida deseada o proporcionada por el simulador (ds) y la salida a la red (ys), expresada mediante una función conocida como función error o de costo (E). Dada la condición de no negatividad de los pesos se requiere en realidad resolver un problema de minimización de dicha función con restricciones, de la forma:

$$\begin{aligned} &Minimizar\ E(\mathbf{w}) \\ &\quad \mathbf{w} \in S \end{aligned} \tag{4}$$

donde E es una función continua sobre $\mathbf{R}^n$ y S es un conjunto restringido en $\mathbf{R}^n$ (conjunto factible), en el que ningún elemento es negativo.

La solución a este problema, tal como lo propone (Shandiz 2011) , se puede plantear de forma que se requiera la minimización de una nueva función error sin restricciones (Ep), incorporando una "penalización decreciente", evitando a la vez que ocurra un probable mal condicionamiento del problema al aumentar el valor de c (parámetro de penalización). Matemáticamente dicha solución se formula así:

$$Minimizar \;\; Ep(\mathbf{w}) = \frac{E(\mathbf{w})}{c} + P(\mathbf{w}) \tag{5}$$

en donde P se conoce como función de penalización.

Para la función error sin penalizar E, expresada en forma de error cuadrático medio, se tiene:

$$E = \frac{1}{2N}\sum_{p=1}^{N}(ds_p - ys_p)^2 \tag{6}$$

Ahora, tomando en cuenta que $ys = r$:

$$E(\mathbf{w}) = \frac{1}{2N}\sum_{p=1}^{N}\Big(ds_p - \big(\frac{\mathbf{w}^t\mathbf{Di} - \phi}{\phi - \mathbf{Cr}^t\mathbf{Dr}}\big)_p\Big)^2 \tag{7}$$

Por otro lado, se tiene la forma general del conjunto factible, tal como se expresa en (Luenberger y col., 2008), $S = \{\mathbf{x} : g_j(\mathbf{x}) \leq 0,\ j = 1, 2, ...n\}$, donde $g_j(\mathbf{x})$ son las funciones de restricción. Adaptando la respectiva nomenclatura al caso en estudio y recordando que se ha de restringir las concentraciones del influente (pesos a estimar) para que sean no negativas; es decir, $w_j \geq 0$, por lo que el conjunto factible en particular queda como $S = \{\mathbf{w} : -w_j \leq 0,\ j = 1, 2, ...n\}$.

Mientras que como función de penalización se usará aquella equivalente a la primordial función propuesta en (Luenberger y col., 2008) para métodos de penalización:

$$P(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{n}(min[0, w_j])^2 \tag{8}$$

donde la función de restricción auxiliar $min[0, w_j]$ se simbolizará en adelante como w_j^-. La penalización a agregarse mediante P, sería nula si ningún w_j es negativo, pues en todo caso el mínimo (min) sería siempre cero.

Por lo que la función error penalizada de acuerdo con la ecuación (5), puede ser expresada como:

$$E_P(\mathbf{w}) = \frac{1}{2cN}\sum_{p=1}^{N}\Big(ds_p - \big(\frac{\mathbf{w}^t\mathbf{Di} - \phi}{\phi - \mathbf{Cr}^t\mathbf{Dr}}\big)_p\Big)^2 + \frac{1}{2}\sum_{j=1}^{n}(g_j^-(\mathbf{w}))^2 \tag{9}$$

Haciendo $b = d_s + \phi/(\phi - \mathbf{Cr}^t\mathbf{Dr})$ y $A = \mathbf{Di}/(\phi - \mathbf{Cr}^t\mathbf{Dr})$, se obtiene después de algunos artificios algebraicos:

$$E_P(\mathbf{w}) = \frac{1}{2cN}\sum_{p=1}^{N}\big(b_p - \mathbf{w}^t A_p\big)^2 + \frac{1}{2}\sum_{j=1}^{n}(g_j^-(\mathbf{w}))^2 \tag{10}$$

que por medio de la definición de norma euclidiana resulta

$$E_P(\mathbf{w}) = \frac{1}{2cN}\|A^t\mathbf{w} - \mathbf{b}\|^2 + \frac{1}{2}\|\mathbf{g}^-(\mathbf{w})\|^2 \tag{11}$$

donde: $A^t \in \mathbf{R}^{N*n}$, $\mathbf{w} = [w_1, w_2, ..., w_n]^t$ y $\mathbf{b} = [b_1, b_2, ..., b_N]^t$.

La ecuación (11) se considera compuesta por dos partes perfectamente distinguibles: $E(\mathbf{w})/c = 1/2cN\|A^t\mathbf{w} - \mathbf{b}\|^2$ y $P(\mathbf{w}) = 1/2\|\mathbf{g}^-(\mathbf{w})\|^2$.

2.2 Gradiente de la Función Error

El gradiente de la función error o de costo penalizada, $\nabla E_P(\mathbf{w})$, es esencial para los algoritmos de aprendizaje de Retropropagación que se desarrollarán posteriormente. A continuación, por razones de espacio, se muestra únicamente la deducción del gradiente correspondiente a la función de penalización $P(\mathbf{w})$, por ser realmente la parte de la función de costo con mayores particularidades inherentes a este trabajo.

2.2.1 Gradiente de $P(\mathbf{w})$

La función $P(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{n}(g_j^-(\mathbf{w}))^2$ se puede reescribir así:

$$P(\mathbf{w}) = \frac{1}{2}\Big(g_1^-(\mathbf{w})^2 + g_2^-(\mathbf{w})^2 + ... + g_n^-(\mathbf{w})^2\Big) \tag{12}$$

o lo que es lo mismo

$$P(\mathbf{w}) = \frac{1}{2}\Big((min[o, w_1])^2 + (min[o, w_2])^2 + ... + (min[o, w_n])^2\Big) \tag{13}$$

Siendo el vector gradiente de esta función

$$\nabla P(\mathbf{w}) = \frac{\partial P(\mathbf{w})}{\partial w_j} = \Big[\frac{\partial P(\mathbf{w})}{\partial w_1}, \frac{\partial P(\mathbf{w})}{\partial w_2}, ..., \frac{\partial P(\mathbf{w})}{\partial w_n}\Big] \tag{14}$$

Así, particularmente, se tiene para la primera componente de este vector, a partir de la ecuación (13)

$$\frac{\partial P(\mathbf{w})}{\partial w_1} = min[0, w_1]\frac{\partial(min[0, w_1])}{\partial w_1} \tag{15}$$

donde:

$$\frac{\partial(min[0, w_1])}{\partial w_1} = \begin{cases} \frac{\partial(0)}{\partial w_1} = 0 \; si \; w_1 \geq 0 & (a) \\ \frac{\partial w_1}{\partial w_1} = 1 \; si \; w_1 < 0 & (b) \end{cases} \tag{16}$$

Sin embargo, tomar en cuenta la parte (a) de la ecuación (16) no es necesario, puesto que si $w_1 \geq 0$ igualmente el valor de la ecuación (15) se haría cero, debido a que $min[0, w_1]$ también es cero en tal circunstancia. De manera que se puede simplificar la ecuación (15), así:

$$\frac{\partial P(\mathbf{w})}{\partial w_1} = min[0, w_1] * 1 = min[0, w_1] \tag{17}$$

Siguiendo un razonamiento similar para el resto de componentes del gradiente se obtiene, sustituyendo en la ecuación (14)

$$\nabla P(\mathbf{w}) = \Big[min[0, w_1], min[0, w_2], ..., min[0, w_n]\Big] \tag{18}$$

es decir

$$\nabla P(\mathbf{w}) = \Big[g_1^-(\mathbf{w}), g_2^-(\mathbf{w}), ..., g_n^-(\mathbf{w})\Big] = \mathbf{g}^-(\mathbf{w}) \tag{19}$$

2.2.2 Gradiente de $E(\mathbf{w})/c$

De forma análoga a la deducción del gradiente de $P(\mathbf{w})$, se obtiene el gradiente de $E(\mathbf{w})/c$, resultando:

$$\nabla E(\mathbf{w})/c = \frac{1}{cN}(AA^t\mathbf{w} - A\mathbf{b}) \tag{20}$$

2.2.3 Gradiente de la Función Error Penalizada

Finalmente, el gradiente para la función correspondiente a la ecuación (11) será:

$$\nabla E_P(\mathbf{w}) = \frac{1}{cN}(AA^t\mathbf{w} - A\mathbf{b}) + \mathbf{g}^-(\mathbf{w}) \tag{21}$$

2.3 Paso o Factor de Aprendizaje

Los diferentes algoritmos de aprendizaje de Retropropagación presentan la actualización de los pesos a estimar, para cada iteración k, bajo la forma:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k\mathbf{d}_k \tag{22}$$

Para determinar el valor óptimo del paso α_k, suponiendo la dirección de búsqueda del mínimo $\mathbf{d}_k$ como conocida, se plantea la función error de tal forma que quede en función del paso, es decir:

$$E_P(\mathbf{w}_{k+1}) = E_P(\mathbf{w}_k + \alpha_k\mathbf{d}_k) \tag{23}$$

Luego, se formula la condición necesaria de primer orden

$$\frac{dE_P(\mathbf{w}_{k+1})}{d\alpha} = \frac{dE_P(\mathbf{w}_k + \alpha_k\mathbf{d}_k)}{d\alpha} = 0 \tag{24}$$

cuya primera parte de acuerdo a la ecuación (11) queda como

$$\frac{d\Big(\frac{1}{2cN}\|A^t\mathbf{w}_{k+1} - \mathbf{b}\|^2 + \frac{1}{2}\|\mathbf{g}^-(\mathbf{w_{k+1}})\|^2\Big)}{d\alpha} = 0 \tag{25}$$

y cuya segunda parte, prescindiendo del subíndice k (todos los elementos de la ecuación pertenecen a esa iteración), resulta

$$\frac{d\Big(\frac{1}{2cN}\|A^t(\mathbf{w} + \alpha\mathbf{d}) - \mathbf{b}\|^2 + \frac{1}{2}\|\mathbf{g}^-(\mathbf{w} + \alpha\mathbf{d})\|^2\Big)}{d\alpha} = 0 \tag{26}$$

Antes de continuar es necesario redefinir $\mathbf{g}^-(\mathbf{w} + \alpha\mathbf{d})$ en función de nuevas variables auxiliares, asociadas a la restricción de no negatividad, que faciliten el despeje de α

$$\mathbf{g}^-(\mathbf{w} + \alpha\mathbf{d}) = \mathbf{w}^- + \alpha^-\mathbf{d}^- = \mathbf{w}^- + \alpha\mathbf{d}^- \tag{27}$$

donde $\mathbf{w}^-$, como ya se ha mencionado, se define en función de sus componentes de la forma $w_j^- = min[0, w_j]$, mientras que $\mathbf{d}^-$ se define en función de sus componentes así:

$$d_j^- = \begin{cases} d_j \; si \; w_j^- < 0 \\ 0 \; si \; w_j^- = 0 \end{cases} \tag{28}$$

En lo respecta a α^-, será simplemente igual a α. Esta simplificación tiene sentido puesto que al estar multiplicado por $\mathbf{d}^-$, se hará cero cuando las componentes de éste se hagan cero.

De esta forma, la ecuación (26) se puede escribir así:

$$\frac{d\Big(\frac{1}{2cN}\|A^t(\mathbf{w} + \alpha\mathbf{d}) - \mathbf{b}\|^2 + \frac{1}{2}\|\mathbf{w}^- + \alpha\mathbf{d}^-\|^2\Big)}{d\alpha} = 0 \tag{29}$$

Luego, resolviendo y despejando α resulta

$$\alpha = \frac{\mathbf{b}^tA^t\mathbf{d}/cN - \mathbf{w}^tAA^t\mathbf{d}/cN - \mathbf{w}^{t-}\mathbf{d}^-}{\mathbf{d}^tAA^t\mathbf{d}/cN + \mathbf{d}^{t-}\mathbf{d}^-} \tag{30}$$

que se puede expresar también como

$$\alpha = \frac{\mathbf{b}^tA^t\mathbf{d}/cN - \|A\|^2\mathbf{w}^t\mathbf{d}/cN - \mathbf{w}^{t-}\mathbf{d}^-}{\|A\mathbf{d}\|^2/cN + \|\mathbf{d}^-\|^2} \tag{31}$$

Obteniéndose, de esta forma, a partir de la condición necesaria $dE_P(\mathbf{w} + \alpha\mathbf{d})/d\alpha = 0$ el valor de α óptimo, el que sin duda es un punto estacionario, pero para estar seguros de que se trata de un mínimo local estricto, se debe cumplir la condición suficiente de segundo orden, es decir, $d^2E_P(\mathbf{w} + \alpha\mathbf{d})/d\alpha^2 > 0$.

Resulta entonces, luego de algunos cálculos

$$\frac{d^2E_P(\mathbf{w} + \alpha\mathbf{d})}{d\alpha^2} = \|A\mathbf{d}\|^2/cN + \|\mathbf{d}^-\|^2 \tag{32}$$

Siendo efectivamente esta última expresión mayor a cero, dado que c y N son siempre mayores a cero, mientras que A y $\mathbf{d}$ son siempre diferentes a cero. Esto último debido a que $A = \mathbf{Di}/(\phi - \mathbf{Cr}^t\mathbf{Dr})$, y teniendo $\mathbf{Di}$ la forma de una función logística, sólo se acerca a cero de forma asintótica; mientras que según (Rodríguez 1999), $\phi - \mathbf{Cr}^t\mathbf{Dr}$ nunca se hace cero, por razones físicas, lo que por su parte garantiza un resultado siempre válido para A. En cuanto a la dirección de búsqueda, $\mathbf{d} = -[\nabla^2 f(\mathbf{x_k})]^{-1}\nabla f(\mathbf{x_k})$, tampoco se hará cero, debido a que la inversa de la matriz hessiana, como se demuestra en (Luenberger y col., 2008) , es siempre definida positiva, y el gradiente, por su parte, no podría ser cero, en cuyo caso todo el proceso optimización carecería de sentido.

3 Algoritmos de Retropropagación restringidos

Los algoritmos o métodos propuestos han de resolver un problema de mínimos cuadrados no negativos, esto es:

Dada una matriz $A \in \mathbf{R}^{nxN}$ y un conjunto de observaciones $b \in \mathbf{R}^N$, encontrar un vector no negativo $\mathbf{w} \in \mathbf{R}^n$ para minimizar la función $E_P(\mathbf{w}) = \frac{1}{2cN}\|A^t\mathbf{w} - \mathbf{b}\|^2 + \frac{1}{2}\|\mathbf{w}^-\|^2$, donde $w_j^- = min[0, w_j]$, siendo $j = 1 \; hasta \; n$.

3.1 Método del Gradiente Conjugado no Negativo (gcnn)

Se tienen como *datos de entrada*: A, $\mathbf{b}$, $\mathbf{w}_0$ (valor inicial del vector $\mathbf{w}$, tomado por defecto como cero) , tol (tolerancia), $itermax$ (número máximo de iteraciones) y F (factor de incremento del parámetro de penalización c). Mientras que N (número de ejemplos de entrenamiento), n (número de pesos)

son el número de columnas y filas de A, respectivamente. Para obtener $\mathbf{w}$ optimizado se han de seguir los siguientes pasos:

1) Se inicia con una iteración $k = 0$, bajo el método del gradiente. Se inicializa el parámetro de penalización, $c_0 = 1$ (sin embargo, a la iteración " cero" no se le aplica restricción debido a que los pesos son necesariamente no negativos al inicio). En esta iteración se efectúan los siguientes cómputos:
   a) La *dirección inicial de búsqueda* del mínimo
   $$\mathbf{d}_0 = -\nabla E(\mathbf{w}_0)$$
   donde
   $$\nabla E(\mathbf{w_0}) = \frac{1}{N}(AA^t\mathbf{w}_0 - A\mathbf{b})$$
   b) Se calcula el *factor de aprendizaje inicial* α_0
   $$\alpha_0 = \frac{\mathbf{b}^t A^t \mathbf{d}_0 - \|A\|^2 \mathbf{w}_0^t \mathbf{d}_0}{\|A\mathbf{d}_0\|^2}$$
   c) El *error inicial* será:
   $$E(\mathbf{w}_0) = \frac{1}{2N}\|A^t\mathbf{w}_0 - \mathbf{b}\|^2$$
   para luego hacer $E_P(\mathbf{w_0}) = E(\mathbf{w}_0)$.
   d) Actualizando los pesos
   $$\mathbf{w}_1 = \mathbf{w}_0 + \alpha_0\mathbf{d}_0$$
2) Se inicia el proceso iterativo propiamente (Gradiente Conjugado), que se ejecuta mientras: $Ep(\mathbf{w_k}) > tol$ y $k \leq itermax$. Si no se cumplen estas dos condiciones se va directamente al paso 4.
   a) Se actualiza la iteración: $k = k + 1$
   b) Se calcula el *parámetro de penalización*: $c_k = F * c_{k-1}$.
   c) Para $j = 1\ hasta\ n$, se calcula la *función de restricción auxiliar*: $w_j^- = min[0, w_j]$, obteniéndose el vector $\mathbf{w}_k^- = (w_1^-, w_2^-, ..., w_n^-)$.
   d) La *dirección de búsqueda* del mínimo es:
   $$\mathbf{d}_k = -\nabla E_P(\mathbf{w_k}) + \beta_k \mathbf{d}_{k-1}$$
   donde
   $$\nabla E_P(\mathbf{w_k}) = \frac{1}{c_k N}(AA^t\mathbf{w}_k - A\mathbf{b}) + \mathbf{w}_k^-$$
   y usando la β_k de Fletcher y Reeves:
   $$\beta_k = \frac{\|\nabla E_p(\mathbf{w}_k)\|^2}{\|\nabla E_p(\mathbf{w}_{k-1})\|^2}$$
   e) Las direcciones d_j^- se calculan de la forma:
   $$d_j^- = \begin{cases} d_j\ si\ w_j^- < 0 \\ 0\ si\ w_j^- = 0 \end{cases}$$
   para obtener el vector $\mathbf{d}_k^- = (d_1^-, d_2^-, ..., d_n^-)$.
   f) Se calcula el *factor de aprendizaje* α_k
   $$\alpha_k = \frac{\mathbf{b}^t A^t \mathbf{d}_k / c_k N - \|A\|^2 \mathbf{w}_k^t \mathbf{d}_k / c_k N - \mathbf{w}_k^{t-} \mathbf{d}_k^-}{\|A\mathbf{d}_k\|^2 / c_k N + \|\mathbf{d}_k^-\|^2}$$
   g) El *error penalizado* es:
   $$E_P(\mathbf{w}_k) = \frac{1}{2c_k N}\|A^t\mathbf{w}_k - \mathbf{b}\|^2 + \frac{1}{2}\|\mathbf{w}_k^-\|^2$$
   h) Se actualizan los pesos
   $$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{d}_k$$
   i) Se asignan: $\mathbf{d}_{k-1} = \mathbf{d}_k$ y $\nabla E_P(\mathbf{w_{k-1}}) = \nabla E_P(\mathbf{w_k})$
3) Volver al paso 2.
4) Se aproxima a cero los pesos "pequeños". Para $j = 1\ hasta\ n$ se hace:
   si $w_j < 0{,}0001 \rightarrow w_j = 0$
5) Fin de la función

3.2 Método Quasi-Newton (DFP) no Negativo (qndfpnn)

Este método de segundo orden actualiza la inversa de la matriz hessiana de acuerdo con la siguiente fórmula:

$$H_{k+1}^{DFP} = H_k + \frac{\mathbf{s}_k \mathbf{s}_k^t}{\mathbf{s}_k^t \mathbf{y}_k} - \frac{H_k \mathbf{y}_k {\mathbf{y}_k}^t H_k}{{\mathbf{y}_k}^t H_k \mathbf{y}_k}$$

donde: $\mathbf{s}_k = \alpha_k \mathbf{d}_k$ y $\mathbf{y}_k = \nabla E_P(\mathbf{w_{k+1}}) - \nabla E_P(\mathbf{w_k})$. H_0 se puede suponer como la matriz identidad I, la cual es siempre definida positiva.

Los *datos de entrada* son los mismos del algoritmo anterior. Los pasos esenciales a seguir son:

1) Se inicializan las variables: iteración, $k = 0$; el parámetro de penalización, $c_0 = 1$; la inversa de la hessiana, $H_0 = H_1 = I$; $\mathbf{w}_0^- = \mathbf{0}$, debido a que todas las componentes de $\mathbf{w}_0$ son positivas.
2) Se calcula el *error inicial*:
   $$E(\mathbf{w_0}) = \frac{1}{2N}\|A^t\mathbf{w_0} - \mathbf{b}\|^2$$
   y se hace $E_P(\mathbf{w_0}) = E(\mathbf{w_0})$.
3) Se inicia el proceso iterativo , que se ejecuta mientras: $Ep(\mathbf{w_k}) > tol$ y $k \leq itermax$. Si no se cumplen estas dos condiciones se va directamente al paso 5.
   a) Se actualiza la iteración: $k = k + 1$
   b) Se calcula el *parámetro de penalización*: $c_k = F * c_{k-1}$
   c) El gradiente se calcula aquí (al inicio del proceso iterativo) una única vez, así que si $k = 1$ se tiene
   $$\nabla E_P(\mathbf{w_k}) = \frac{1}{c_k N}(AA^t\mathbf{w}_k - A\mathbf{b}) + \mathbf{w}_k^-$$
   con lo que ya se puede hallar la *dirección de búsqueda* (que sí se calcula aquí para todas las iteraciones):
   $$\mathbf{d}_k = -\nabla E_P(\mathbf{w_k}) H_k$$

d) Las direcciones d_j^- se determinan de la forma:

$$d_j^- = \begin{cases} d_j \; si \; w_j^- < 0 \\ 0 \; si \; w_j^- = 0 \end{cases}$$

para obtener el vector $\mathbf{d}_k^- = (d_1^-, d_2^-, ..., d_n^-)$.

e) Se calcula el *factor de aprendizaje* α_k

$$\alpha_k = \frac{\mathbf{b}^t A^t \mathbf{d}_k / c_k N - \|A\|^2 \mathbf{w}_k^t \mathbf{d}_k / c_k N - \mathbf{w}_k^{t-} \mathbf{d}_k^-}{\|A\mathbf{d}_k\|^2 / c_k N + \|\mathbf{d}_k^-\|^2}$$

f) El *error penalizado* es:

$$E_P(\mathbf{w}_k) = \frac{1}{2c_k N}\|A^t \mathbf{w}_k - \mathbf{b}\|^2 + \frac{1}{2}\|\mathbf{w}_k^-\|^2$$

g) Se actualizan los pesos

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{d}_k$$

h) Se calcula la *función de restricción auxiliar*: $w_j^- = min[0, w_j]$ para $\mathbf{w}_{k+1}$, obteniéndose el vector $\mathbf{w}_{k+1}^- = (w_1^-, w_2^-, ..., w_n^-)$.

i) El nuevo gradiente será

$$\nabla E_P(\mathbf{w_{k+1}}) = \frac{1}{c_k N}(AA^t \mathbf{w}_{k+1} - A\mathbf{b}) + \mathbf{w}_{k+1}^-$$

j) Para el cálculo de la inversa de la *matriz hessiana* aproximada se sigue:

i) $\mathbf{s}_k = \alpha_k \mathbf{d}_k$

ii) $\mathbf{y}_k = \nabla E_P(\mathbf{w_{k+1}}) - \nabla E_P(\mathbf{w_k})$

iii) $H_{k+1} = H_k + \frac{\mathbf{s}_k \mathbf{s}_k^t}{\mathbf{s}_k^t \mathbf{y}_k} - \frac{H_k \mathbf{y}_k \mathbf{y}_k{}^t H_k}{\mathbf{y}_k{}^t H_k \mathbf{y}_k}$

iv) Luego, se hace $\nabla E_P(\mathbf{w_k}) = \nabla E_P(\mathbf{w_{k+1}})$

4) Volver al item 3.

5) Se aproxima a cero los pesos "pequeños". Para $j = 1$ $hasta$ n se hace:
si $w_j < 0{,}0001 \rightarrow w_j = 0$

6) Fin de la función

3.3 Método Quasi-Newton (BFGS) no Negativo (qnbfgsnn)

Este método Quasi-Newton, igualmente de segundo orden, actualiza la inversa de la matriz hessiana de acuerdo con la siguiente fórmula:

$$H_{k+1}^{BFGS} = H_k + \left(1 + \frac{\mathbf{y}_k{}^t H_k \mathbf{y}_k}{\mathbf{s}_k{}^t \mathbf{y}_k}\right) \frac{\mathbf{s}_k \mathbf{s}_k{}^t}{\mathbf{s}_k{}^t \mathbf{y}_k} - \frac{H_k \mathbf{y}_k \mathbf{s}_k{}^t + \mathbf{s}_k \mathbf{y}_k{}^t H_k}{\mathbf{s}_k{}^t \mathbf{y}_k}$$

Nuevamente, H_0 es la matriz identidad I. Los *datos de entrada* siguen siendo los mismos. El algoritmo correspondiente a este método es exactamente igual que el de $qndfpnn$, con la única excepción del paso 3.j.3, donde en lugar de aplicarse la fórmula para hallar H_{k+1}^{DFP}, se usa H_{k+1}^{BFGS}.

4 Simulaciones y Resultados

A los fines de facilitar la comparación entre el desempeño computacional de los algoritmos de Retropropagación propuestos y el de $lsqnonneg$, se ha desarrollado haciendo uso del GUIDE (Entorno de desarrollo de interfaz gráfica de usuario) de Matlab, una interfaz de usuario denominada $rnaph$, cuya apariencia puede ser observada en la figura 3. Esta interfaz permite la introducción de datos relacionados

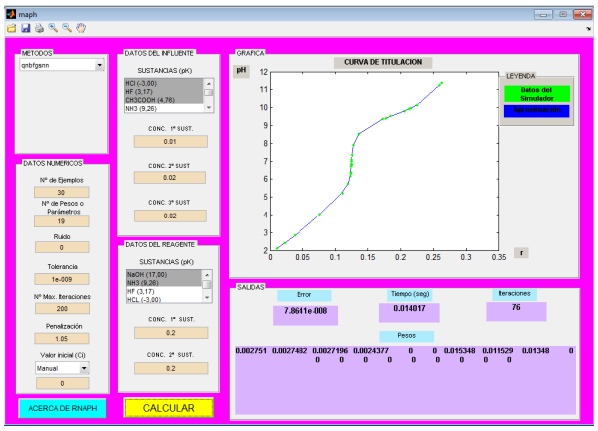

Fig. 3. Apariencia de la Interfaz RNAPH

con: los métodos a comparar (tipo de método de aprendizaje utilizado), los aspectos numérico-computacionales (número de ejemplos de entrenamiento, número de pesos, nivel de ruido, tolerancia, número máximo de iteraciones, factor de incremento del parámetro de penalización y valor de los pesos iniciales) y algunas variables químicas (concentraciones y constantes de disociación de las sustancias tanto del influente como del reagente). Adicionalmente, la interfaz muestra como datos de salida: la gráfica de la curva de titulación, el ECM (Error Cuadrático Medio) entre los datos suministrados por el simulador (representados gráficamente por puntos verdes) y la aproximación aportada por los algoritmos propuestos o el $lsqnonneg$ (representada como una línea continua azul), el tiempo (en segundos) empleado por cada método en el proceso de aprendizaje, y el número de iteraciones empleadas en dicho aprendizaje.

Los datos se generan mediante simulación computacional, a partir del modelo de la curva de titulación desarrollado en (Rodríguez 1999), haciendo igualmente uso de Matlab. Estos datos son suministrados en forma de pares $(\mathbf{pH}, \mathbf{r})$, siendo unos utilizados como ejemplos de entrenamiento y otros adicionales, para la validación o test.

En cuanto a las sustancias químicas usadas en el simulador, a menos que se indique expresamente lo contrario, se tiene para el $reagente$, cuyos datos se suponen completamente conocidos en el entrenamiento, la sustancia única Hidróxido de Sodio ($NaOH$), la cual es una base fuerte, con concentración de 0,2 $moles/litro$ = 0,2 M (Molar). Mientras

que para el *influente*, cuyos datos son completamente desconocidos en el entrenamiento, se tiene: ácido hidroclorhídrico (HCL), que es un ácido fuerte, con una concentración 0,01 M, y ácido acético (CH_3COOH), que es un ácido débil, con una concentración de 0,02 M. La naturaleza de las sustancias involucradas y/o sus concentraciones sufrirán variaciones en la parte de las simulaciones denominada "Modelo Físico y Desempeño Computacional".

4.1 Experimento previo

Ante todo sería interesante indagar sobre un aspecto esencial relacionado con la pertinencia de este trabajo, la conveniencia o no de implementar una red neuronal de tipo caja gris y su inherente estimación de parámetros internos. Para esto se evaluó el desempeño de una red neuronal convencional, es decir, de tipo caja negra, en donde no se hace tal estimación; siendo mucho más sencilla de implementar, puesto que toma en cuenta únicamente datos de entrada y salida, que serán procesados por un algoritmo de red neuronal estándar de Matlab ($newff$).

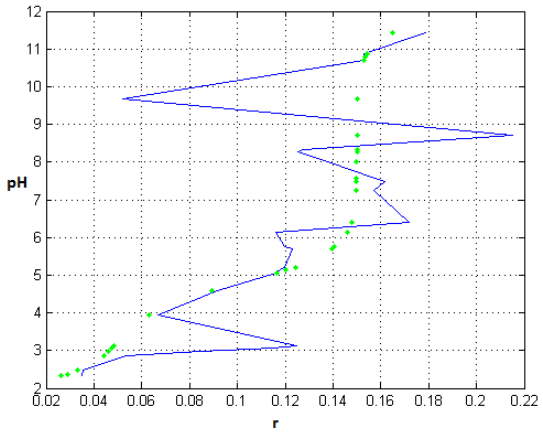

Fig. 4. Aproximación mediante newff de Matlab

Como se puede observar en la figura 4, la función aproximada no es biyectiva, y esto es lo que ocurre precisamente al no plantearse las restricciones de no negatividad para los parámetros de la curva de titulación; lo que trae como consecuencia que la curva no se pueda usar posteriormente en un hipotético proceso de control, donde, en muchos casos, para diferentes valores de pH se tendrá una misma cantidad de reagente a agregar. Adicionalmente, el tiempo y el error de aproximación, son en general considerablemente más grandes para $newff$ que para los algoritmos propuestos y el $lsqnonneg$, implementados bajo la estructura de una RNA de tipo caja gris.

4.2 Comparación variando los datos numéricos de entrada

Se plantea la comparación directa entre el método $lsqnonneg$ de Matlab y los algoritmos de aprendizaje de Retropropagación: $gcnn$, $qndfpnn$ y $qnbfgsnn$, en cuanto a la calidad del ajuste o precisión, variando algunos datos numéricos de entrada (número de ejemplos de entrenamiento, número de pesos y nivel de ruido introducido en los datos de entrada) y manteniéndose el resto de datos, incluyendo los referidos a variables químicas.

En cuanto a la eficiencia, debido a que en la totalidad de experimentos realizados los tiempos empleados por los métodos propuestos son mayores (aunque aceptables en la práctica) que los del método $lsqnonneg$, se prescindirá generalmente de esta variable con fines comparativos. Aunque es de aclarar que, por tratarse el $lsqnonneg$ de una función predefinida de Matlab, encuentra ventajas adicionales en su rapidez respecto a los algoritmos propuestos. Por lo que en adelante se mostrará únicamente el EMC como dato de salida relevante, cuyo valor es en realidad un promedio, obtenido tomando en cuenta 30 ensayos (procesos aprendizaje), en cada uno de los cuales a su vez se usaron, a menos que se especifique lo contrario, 30 ejemplos de entrenamiento.

4.2.1 Variando el número de ejemplos

En esta parte se evalúa el desempeño de los métodos variando el número de ejemplos de entrenamiento entre los valores: 10, 100 y 500. Los resultados se pueden observar en la figura 5.

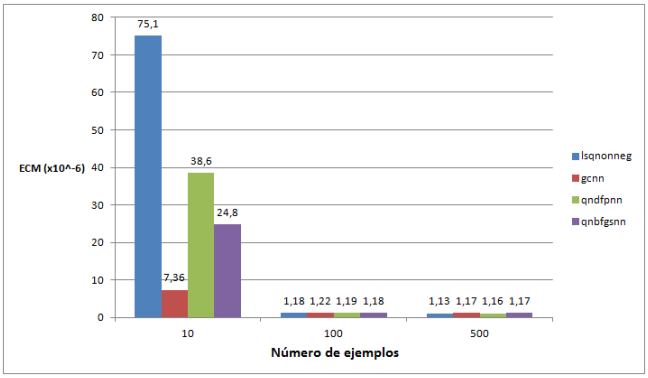

Fig. 5. Precisión variando el número de ejemplos

Se aplican las herramientas para el análisis de la varianza (ANOVA) de Matlab , $Anova1$ y $multcompare$, donde para cada método se obtiene la media de sus EMC (círculo) y su respectivo intervalo de confianza de 95 % (segmento horizontal). Tomando como método 1 el $lsqnonneg$, método 2 el $gcnn$, método 3 el $qndfpnn$ y método 4 el $qnbfgsnn$, se tiene que para 10 ejemplos no se producen diferencias significativas entre los errores medios arrojados por cada método, tal como se puede observar en la figura 6 (gráfico aportado directamente por la función $multcompare$). Para 100 ejemplos tampoco se producen diferencias significativas entre los distintos errores medios (ver figura 7). En cuanto al caso de 500 ejemplos, el método $lsqnonneg$ presenta

un error de aproximación significativamente menor que los demás (ver figura 8).

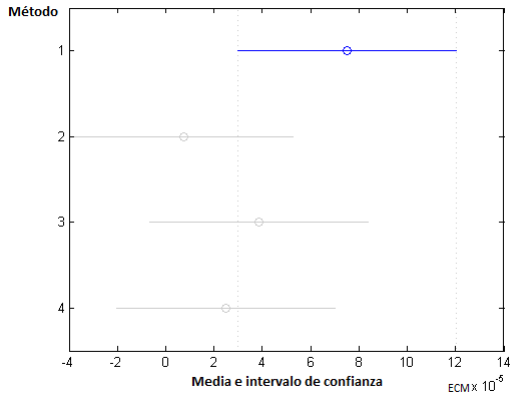

Fig. 6. Comparación múltiple para 10 ejemplos

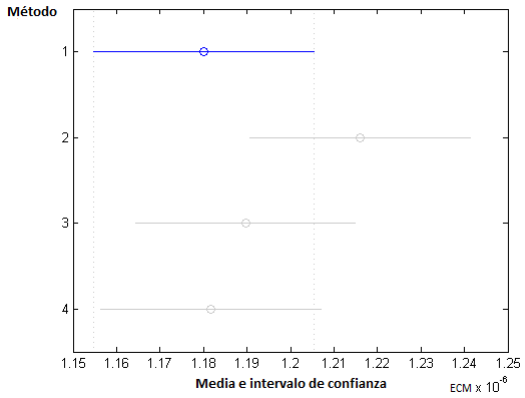

Fig. 7. Comparación múltiple para 100 ejemplos

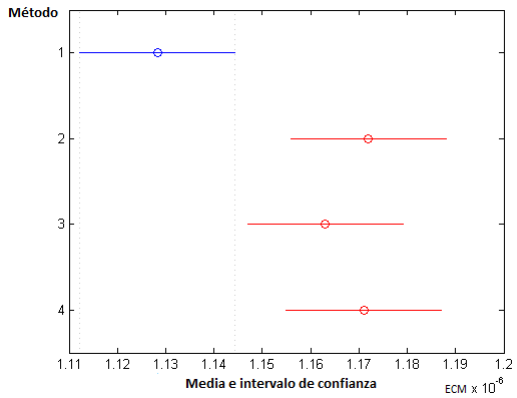

Fig. 8. Comparación múltiple para 500 ejemplos

Los resultados del ANOVA se presentarán en adelante únicamente como un resumen textual, sin hacer referencia a gráfico alguno.

4.2.2 Variando el número de pesos

Se evalúa el desempeño de los métodos variando el número de pesos entre: 10, 50 y 100 (ver figura 9).

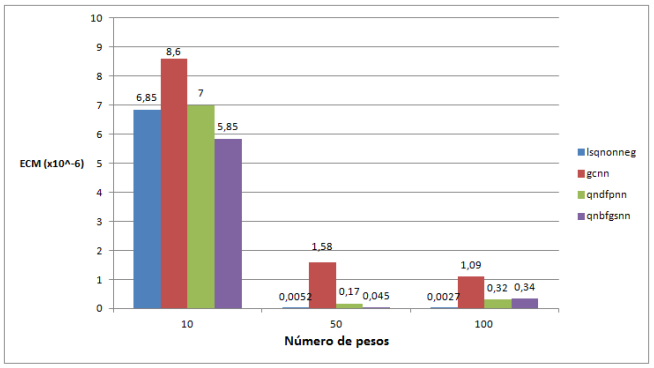

Fig. 9. Precisión variando el número de pesos

Según el respectivo análisis de la varianza, se tiene que para el caso de 10 pesos no existe una diferencia significativa entre los errores medios de cada método, mientras que para 50 y 100 pesos el error medio de $gcnn$ es significativamente mayor que el de los demás métodos. Es de resaltar aquí que, a pesar del aumento del número de pesos (de 50 a 100), los métodos Quasi-newton aumentaron su error de aproximación.

4.2.3 Variando el nivel de ruido en la entrada

Ahora se variarán los niveles de ruido (nr) que afectan a los valores de las componentes del vector $\mathbf{pH}$ de entrada. Se realizarán experimentos para valores de nr de 0,1, 0,2 y 0,3. El ruido es inevitablemente introducido, en la práctica, a través de los sensores de pH. En este trabajo es simulado mediante la fórmula para la entrada: $\mathbf{pH} = \mathbf{pH} + randn(size(\mathbf{pH})) * nr$, donde $randn$ y $size$ son funciones de Matlab. La función $randn$ genera un vector de números aleatorios cuyos elementos están normalmente distribuidos, con media 0 y desviación estandar de 1.

Recordando que, de acuerdo con la ecuación (3), a pesar de la apariencia de la gráfica de la curva de titulación (figura 10), matemáticamente $\mathbf{pH}$ es la entrada y $\mathbf{r}$ la salida. En la figura 11 se pueden observar los resultados de esta comparación.

En cuanto al análisis de la varianza, se tiene que no existe diferencia significativa entre la media de los errores de ninguno de los métodos para ninguno de los niveles de ruido.

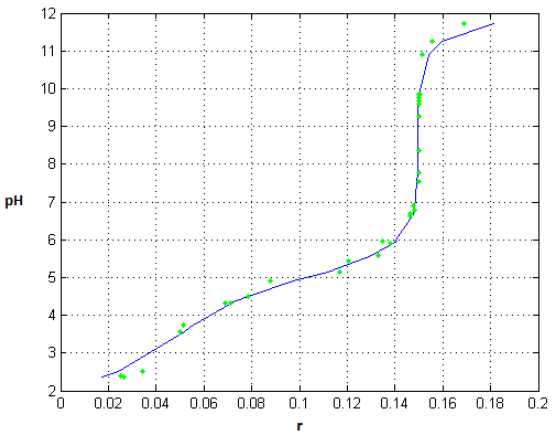

Fig. 10. Aproximación con intervención de ruido

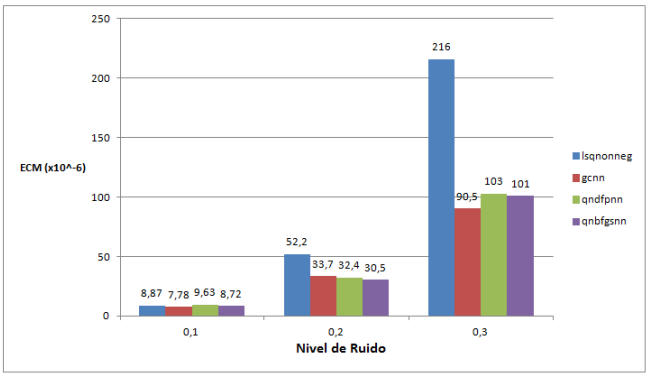

Fig. 11. Precisión variando el nivel de ruido

4.3 *Modelo Físico y Desempeño Computacional*

En esta parte se hace especial énfasis en la variación de datos relacionados con los aspectos químicos, que conllevan a cambios en la forma de la curva de titulación, y en la equivalencia entre variables numérico-computacionales y las propiamente químicas.

Manteniendo los datos numéricos de entrada (a menos que se indique explícitamente lo contrario), se verificará la calidad del ajuste de los métodos comparados, variando la concentración o la naturaleza de las sustancias químicas involucradas.

4.3.1 *Variando la concentración de las sustancias del influente*

Para una concentración fija del reagente ($NaOH$) en 0,2 M, se procede ahora a variar en el simulador el valor de la concentración de las sustancias del influente (parámetros a estimar o conjunto solución), sin presencia de ruido. Los resultados se pueden ver en la figura 12.

Según el análisis de la varianza no existe para los dos primeros casos una diferencia significativa entre las medias

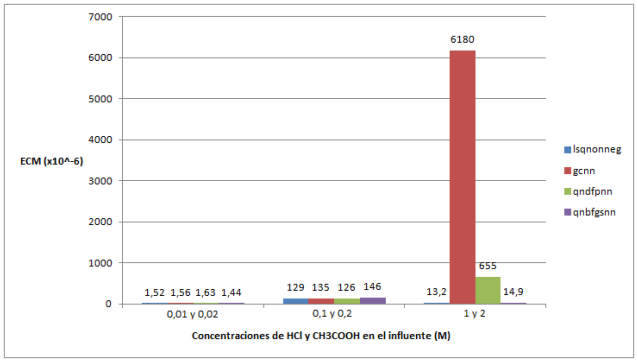

Fig. 12. Precisión variando las concentraciones en el influente

de los errores cometidos por cada método; mientras que para las concentraciones de 1 M y 2 M de los ácidos presentes en el influente, clorhídrico y acético, respectivamente, ese mismo análisis arroja como resultado un error de aproximación significativamente diferente para $gcnn$ con respecto a los demás métodos.

4.3.2 *Titulación con múltiples sustancias bajo condiciones no ideales*

En esta parte se propone la simulación de condiciones que se presentan regularmente en los procesos de titulación reales y que son propicias para evaluar el desempeño de los métodos de optimización ante situaciones más demandantes, tales como: la presencia de múltiples sustancias en el influente y en el reagente, titulación de ácidos con bases o viceversa, la presencia de ruido y/o la disposición de pocos ejemplos para el entrenamiento.

Es de tomar en cuenta que una mayor cantidad de sustancias requiere de más constantes de disociación, en consecuencia, se debe incrementar el número de pesos. A su vez, la presencia de varias sustancias implicará una modificación sustancial en la pendiente de la curva de titulación. Desde el punto de vista del control del sistema, aunque elevan la complejidad de la red por requerir más elementos, la presencia de múltiples sustancias simplifica el comportamiento, puesto que la alta no linealidad característica de los sistemas simples, con ácidos y bases fuertes, se ve atenuada con dichos múltiples componentes.

Por otra parte, la presencia de ruido y pocos ejemplos disponibles para el entrenamiento , implica un desafío a la capacidad de generalización de la red.

Para todas las sustancias del influente se tomará una concentración de 0,02 M y para todas las del reagente una de 0,2 M. A continuación se detallan los 4 experimentos computacionales realizados.

Experimento 1: en esta parte se simula la titulación de varios ácidos en el influente: HCl (ácido clorhídrico), HF (ácido fluorhídrico) y CH_3COOH (ácido acético), con varias bases

en el reagente: $NaOH$ (hidróxido de sodio) y NH_3 (amoníaco), ante la presencia de ruido (nivel de 0,1) e incrementando la cantidad de pesos en busca de un posible mejor ajuste (ver figura 13).

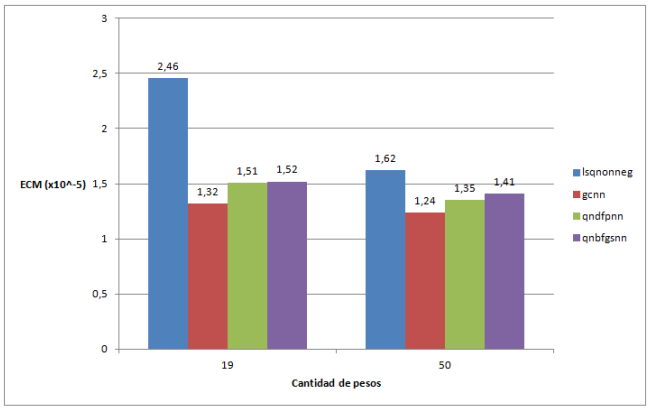

Fig. 13. Precisión para titulación de ácidos con bases

El análisis de la varianza da como resultado, para el caso de 19 pesos, que existe una diferencia significativa entre la media de los errores de aproximación de $lsqnonneg$ y de $gcnn$. En cuanto al caso de 50 pesos, no existe diferencia significativa entre las medias de los errores de ninguno de los métodos.

Experimento 2: en este caso, se simula la titulación de varias bases en el influente: NH_3 (amoníaco), CH_3NH_2 (metilamina) y $(C_2H_5)_2NH$ (dietilamina), con varios ácidos en el reagente: HF (ácido fluorhídrico) y HCl (ácido clorhídrico), igualmente en presencia de ruido (nivel de 0,01) y variando la cantidad de pesos (ver figura 14).

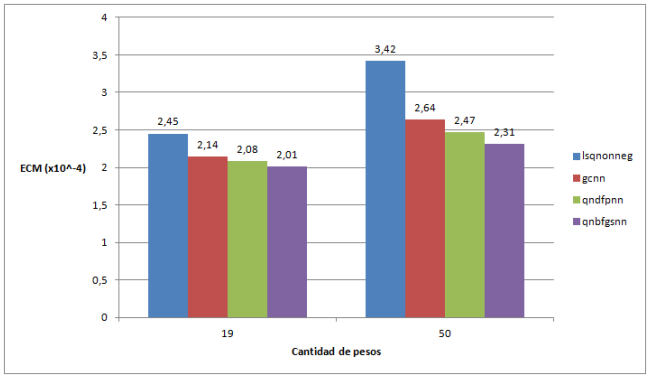

Fig. 14. Precisión para titulación de bases con ácidos

El análisis de la varianza no señala alguna diferencia significativa entre la media de los errores de aproximación de los métodos, tanto para 19 como para 50 pesos.

Experimento 3: las sustancias consideradas aquí son las mismas del experimento 1, pero en este caso los ejemplos de entrenamiento serán 10. El nivel de ruido es igualmente 0,1 (ver figura 15).

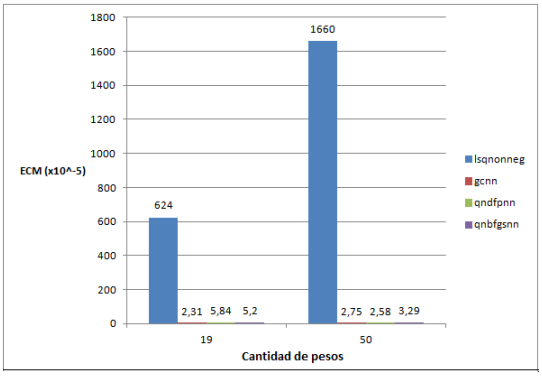

Fig. 15. Precisión para titulación de ácidos con bases (10 ejemplos)

De acuerdo con el análisis de la varianza en ningún caso se observa diferencia significativa entre las medias de los errores de los diferentes métodos. En la figura 16 se puede observar la curva aproximada para este caso.

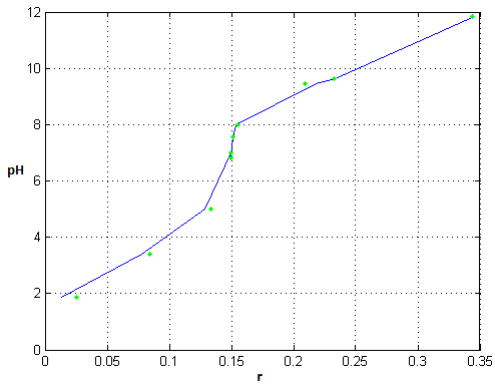

Fig. 16. Aproximación para titulación de ácidos con bases (10 ejemplos)

Experimento 4: las sustancias son aquí las mismas del experimento 2. Aunque se emplean ahora 10 ejemplos de entrenamiento, el nivel de ruido es igualmente de 0,01. En la figura 17 se pueden observar los resultados de esta comparación . El análisis de la varianza, nuevamente, no señala alguna diferencia significativa entre las medias de los errores de aproximación de los métodos. En la figura 18 se puede observar la curva aproximada para este caso.

4.4 Análisis de Resultados

Debido a que a lo largo de las simulaciones o experimentos realizados se llevó a cabo el respectivo análisis de la varianza, para corroborar alguna diferencia significativa entre las medias de los ECM de los diversos métodos, se hace necesario aclarar que aunque dicho análisis indique que no

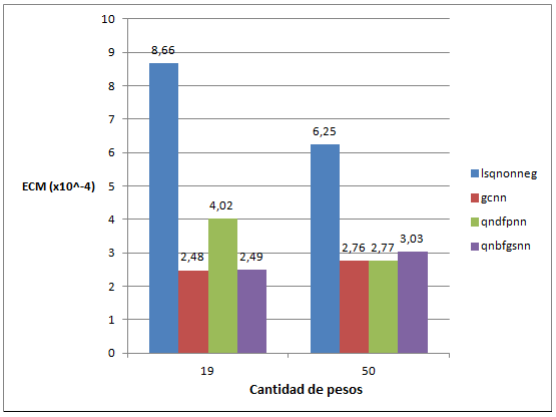

Fig. 17. Precisión para titulación de bases con ácidos (10 ejemplos)

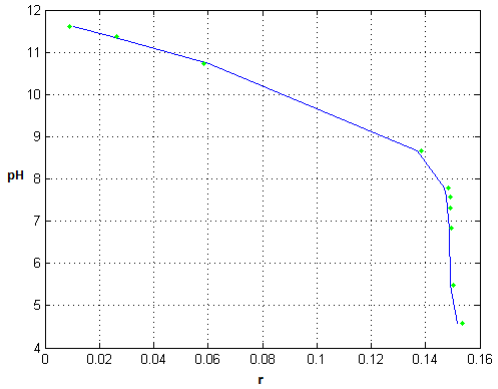

Fig. 18. Aproximación para titulación de bases con ácidos (10 ejemplos)

existe tal diferencia significativa, los métodos que arrojen un ECM promedio considerablemente superior a otros (haciendo abstracción de la varianza), habrán presentado las peores aproximaciones (dentro de las 30 tomadas en cuenta para cada método), las que si son de magnitud relevante, pudiesen ocasionar algún fallo en el proceso de control. Tal situación se evidencia, por ejemplo, en la figura 15 con el $lsqnonneg$ de Matlab.

Ahora, tomando en cuenta la calidad en el ajuste de los métodos comparados, al aumentar el número de ejemplos de entrenamiento, se tiene una mejor respuesta para $lsqnonneg$; aunque, en general, a partir de cierto número de ejemplos, todos los métodos mejoran en muy poco su precisión. No obstante, cuando lo que se incrementa es el número de pesos, es notable la mejora en el ajuste de $lsqnonneg$ (método que aproxima la función objetivo mediante funciones lineales), mientras que los métodos propuestos (métodos que aproximan la función objetivo mediante funciones cuadráticas), ante un número elevado de pesos, pueden aumentar su error de aproximación, debido a la mayor tendencia a sufrir de sobreajuste por parte de estos métodos no lineales. Sin embargo, resulta pertinente aclarar que, en la práctica, generalmente no es recomendable, ni necesario, un elevado número de pesos, sobre todo ante la presencia de ruido.

Cuando lo que se varía son las concentraciones de las sustancias del influente (conjunto solución del problema de optimización), entonces se estará variando la distancia del punto de partida de optimización al conjunto solución. Tomando en cuenta que los métodos comparados parten por defecto de cero, se puede decir, haciendo uso del respectivo análisis de la varianza, que estos métodos no presentan diferencia significativa en la calidad del ajuste cuando las concentraciones son cercanas a cero; sin embargo, $gcnn$ presenta una menor robustez ante puntos de partida alejados del conjunto solución.

Por otra parte, en la simulación de situaciones "reales", donde se cuenta con la presencia de múltiples sustancias tanto en el reagente como en el influente, y de ruido en la entrada, la ventaja de $lsqnonneg$ de responder de forma bastante positiva al aumento del número de pesos, pierde importancia, puesto que en algunos casos al aumentar la cantidad de éstos, su error de aproximación tiende a aumentar. Adicionalmente, cuando se considera la situación de pocos ejemplos de entrenamiento, es aún más notable un mejor ajuste por parte de los métodos propuestos, principalmente para el caso de titulación de ácidos con bases; por lo que se puede decir que, ante la presencia de situaciones altamente demandantes, los algoritmos propuestos cobran preponderancia.

5 Conclusiones y Recomendaciones

1) Ante la pregunta decisiva, sobre cuál es el método con el mejor desempeño computacional, la respuesta no es única. Por lo que se plantean las siguientes situaciones:
   a) Dado el caso de alguna aplicación práctica, inclusive más allá del control de pH, que posea un modelo matemático relativamente lineal, sin presencia importante de ruido, con la suficiente disposición de ejemplos de entrenamiento, y que por su naturaleza requiera de un método altamente eficiente y/o preciso, la elección sería en favor del $lsqnonneg$ de Matlab.
   b) Mientras que ante una situación donde se cuente con una presencia importante de ruido, posiblemente escasos ejemplos para el entrenamiento y/o con múltiples sustancias involucradas (en el caso de control de pH), tendríamos las siguientes alternativas:
      i) Si la aplicación práctica no requiere de una alta precisión y en todo momento se tiene una idea del conjunto solución (no se necesita de un método muy robusto que pueda empezar la optimización en un punto alejado de la solución), la elección favorece a $gcnn$.
      ii) En cambio, si no se requiere de la estimación de un número muy elevado de pesos o parámetros, los métodos a utilizar serían los Quasi-Newton

restringidos ($qndfpnn$ o $qnbfgsnn$), debido fundamentalmente a su mayor robustez en general.

2) Puesto que Matlab está concebido para usos bastante específicos, se recomienda implementar los diferentes algoritmos propuestos bajo herramientas computacionales de uso más generalizado, con la finalidad de brindarles un sentido más práctico. Aunque, adicionalmente, también sería bastante útil compilar en el mismo Matlab dichos algoritmos, de manera que, aparte de una mayor facilidad para su implementación, se les procure una mayor rapidez en su proceso de aprendizaje.
3) Sería interesante, dada su mayor robustez en general, hacer un estudio comparativo más intensivo sobre el desempeño de los Quasi-Newton no negativos, con la finalidad de averiguar más rigurosamente bajo cuáles condiciones específicas cada uno de éstos funciona mejor.
4) La interfaz de usuario $rnaph$, desarrollada en esta investigación, representa una herramienta que se pudiese aprovechar tanto en la didáctica de la programación matemática y áreas afines, como en la de la química, dada su concepción como simulador de procesos matemáticos y químicos (con la ayuda de $tituinv$) simultáneamente.

Referencias

Conti D, Simó C, 2005, Teoría de carteras de inversión para la diversificación del riesgo: enfoque clásico y uso de redes neuronales artificiales (RNA), Ciencia e Ingeniería, vol. 26, No.1, pp. 35-42.

Cruz F, Acuña G, Badillo G, 2011, Propuesta metodológica para la creación de modelos neuronales de caja gris utilizando Matlab, 9th Latin American and Caribbean conference for engineering and technology, Medellín.

Hudson M, Hagan M, Demuth H, 2013, Neural netwok toolbox, The Mathworks Inc., Natick.

Jabbour G, Maldonado J, 2009, Predicción de índices bursátiles mediante un sistema híbrido basado en modelos ocultos de Márkov y redes neuronales artificiales, Ciencia e Ingeniería, vol. 30, No. 2, pp.127-136.

Jabbour G, Maldonado J, 2014, Reconocimiento automático de fonemas en habla continua venezolana por medio de sistemas híbridos basados en modelos ocultos de Márkov y redes neuronales artificiales, Ciencia e Ingeniería, vol. 35, No. 1, pp.29-38.

Luenberger D, Ye Y, 2008, Linear and nonlinear programming (3º Ed.), Editorial Springer, New York.

Rodríguez J, 1999, Modeling, identification and predictive control of pH processes, Tesis doctoral no publicada, Case Western Reserve University, Cleveland .

Shandiz R, Tohidi E, 2011, Decrease of the penalty parameter in differentiable penalty function methods, Theoretical Economics Letters, vol. 1 , No. 1 , pp. 8-14.

Xiong Q, Jutan A, 2001, Grey-Box modelling and control of chemical processes, Chemical Engineering Science, vol. 57, pp. 1027-1039.

***Recibido**: 18 de febrero de 2016*

***Aceptado**: 15 de julio de 2017*

***Hernández Rodríguez, Jairo Javier**: Ingeniero Mecánico, Licenciado en Educación Mención Matemática y Magister en Matemática. Coordinador del departamento de evaluación del NER 489, adscrito al Ministerio del Poder Popular para la Educación, en Pregonero, estado Táchira.*

***Rodríguez Pérez, José Luis**: Ingeniero Mecánico, M.Sc. y Ph.D. en Ingeniería de Sistemas y Control. Profesor Titular a Dedicación Exclusiva (jubilado) de la Universidad Nacional Experimental del Táchira. Decano de Investigación UNET (2007-2014). Correo electrónico: jlrodriguezp@unet.edu.ve.*