Información Tecnológica
Vol. 30(4), 227-236 (2019)
http://dx.doi.org/10.4067/S0718-07642019000400227

Entrenamiento Comprimido Basado en Máquinas de Aprendizaje Extremo

Fausto M. Castro y Pablo E. Jojoa
Grupo de Nuevas Tecnologías de Telecomunicaciones, Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad del Cauca, carrera 9 # 8-51 barrio San Camilo, Sector Tulcán, Popayán, Colombia.
(e-mail: faustocastro@unicauca.edu.co; pjojoa@unicauca.edu.co)

Recibido Nov. 20, 2018; Aceptado Feb. 7, 2019; Versión final Feb. 9, 2019, Publicado Ago. 2019

Resumen

En este documento se presenta el diseño y prueba de un nuevo modelo de entrenamiento para redes neuronales no realimentadas de una sola capa oculta basado en las propiedades de la Máquina de Aprendizaje Extremo o ELM. El modelo actúa comprimiendo la información proveniente de la capa oculta por medio de un subconjunto de nodos de la misma capa, esto permite disminuir considerablemente la complejidad computacional en comparación con ELM. Resultados experimentales basados en simulación para diferentes problemas de clasificación indican que el modelo propuesto permite reducir considerablemente los tiempos de entrenamiento en comparación con ELM, alcanzando a la vez rendimientos similares en términos de generalización.

Palabras clave: ELM; redes neuronales; entrenamiento automático; técnicas de clasificación; aprendizaje supervisado

Compressed Training Based on Extreme Learning Machine

Abstract

This paper presents the design and testing of a new training model for single hidden layer feedforward network based on the same properties of Extreme Learning Machine (ELM). The model acts by compressing the information coming from the hidden layer by means of a subset of nodes from the same layer. This allows to considerably reduce the computational complexity compared to ELM. Experimental results based on simulation for different classification problems indicate that the proposed model achieves the same ELM performances in terms of generalization, exceeding it in speed

Keywords: ELM; neural networks; machine learning; classification technics; supervised learning

INTRODUCCIÓN

Con el desarrollo de las nuevas tecnologías de la información y las comunicaciones se ha disparado la cantidad de datos que se generan como resultado de las diversas actividades modernas. Este fenómeno tiene gran impacto en casi todos los sectores de la sociedad y se incluye dentro de lo que hoy en día se conoce como "*Big Data*" (Torrecilla y Romo, 2018). Un gran número de actividades modernas generan datos que contienen infor-mación potencialmente valiosa para la toma de decisiones, dicha información representa un valor de uso que debe ser capturado y aprovechado para mejorar un determinado proceso (Wang et al., 2016). Sin embargo, el volumen, la velocidad y la variabilidad con la que se generan los datos actualmente, hacen que esto sea un importante desafío tecnológico que debe ser abordado mediante la investigación y diseño de potentes herra-mientas para el tratamiento de datos (Valenzuela et al., 2016). En este contexto han venido cobrando importancia aquellos modelos que aprenden a partir de muestras, o técnicas de aprendizaje automático entre las que se destacan las máquinas de aprendizaje extremo ELM ("*Extreme Learning Machine*") y sus variantes (Akusok et al., 2015; Xin et al., 2012). La ELM es un modelo de red no realimentada de una sola capa oculta o SLFNs ("*Single-hidden Layer Feedforward Networks*") utilizado principalmente en tareas de aproximación y clasificación. En ELM los nodos ocultos se seleccionan aleatoriamente y los nodos de salida se determinan mediante un proceso de regresión lineal que minimiza el error de entrenamiento y la norma de los pesos de salida (Huang et al., 2006; Huang et al., 2015). Su importancia práctica se fundamenta principalmente en: 1) estudios teóricos que garantizan el que una red SLFN mantiene sus capacidades de aproximación universal y clasificación universal, incluso si sus pesos ocultos son seleccionados aleatoriamente (Huang et al., 2006; Huang y Chen, 2007) y 2) la teoría de Barlett (1998), según la cual entre menor es el error de entrenamiento y la norma de los pesos mejor tiende a ser la generalización.

En los últimos años ELM ha despertado mucho interés en la comunidad científica debido a que, para un amplio rango de aplicaciones sus prestaciones en términos de generalización y tiempo de entrenamiento son mucho mejores en comparación con las de los tradicionales algoritmos basados en gradiente (Huang et al., 2006) y, en general mejores que las de las máquinas de soporte vectorial (Cortes y Vapnik, 1995) y las máquinas de soporte vectorial por mínimos cuadrados (Suykens y Vandewalle, 1999; Huang et al., 2010; Huang et al., 2012; Huang et al., 2015). Por lo que ELM es una buena alternativa para abordar aplicaciones en las que los datos se presenten de forma masiva. Sin embargo, es preciso mencionar que en este tipo de escenarios se pueden presentar problemas de procesamiento relacionados con la necesidad de multiplicar e invertir grandes matrices (Chen et al., 2017). Entre las iniciativas que buscan resolver dichos problemas se incluye el desarrollo de herramientas de alto rendimiento como la propuesta por Akusok et al., (2015) y algunas variantes de tipo paralelo (Heeswijk et al., 2011; Krawczyk, 2016) y distribuido (Xin et al., 2014; Sun et al., 2011) implementadas sobre GPU ("*Graphics Processing Unit*") o usando MapReduce (Vidal et al., 2018). No obstante, en estas propuestas persisten los problemas de procesamiento tipo cuello de botella cuando el número de nodos ocultos es elevado.

Por otra parte, Zhang y Joo (2016) sacando provecho de la similitud que existe entre los modelos metacognitivos (Babu y Suresh, 2012) y el aprendizaje activo (Settles, 2012), proponen un método de entrenamiento para problemas de clasificación con grandes volúmenes de datos. Esta iniciativa se basa en una variante secuencial de ELM conocida como OS-ELM ("*Online Sequential Extreme Learning Machine*") que funciona por lotes o por muestras (Huang et al., 2005). La idea consiste básicamente en entrenar un modelo utilizando un conjunto inicial de muestras, dicho modelo se usa posteriormente para clasificar nuevas muestras dependiendo de la información que aportan. Las muestras que aportan más información se usan para entrenar, las que aportan poca información se descartan y las restantes se ponen en cola de espera. Una vez seleccionado un número determinado de muestras se actualiza el modelo y el proceso se repite nuevamente. Otro trabajo similar es el propuesto por Qian et al. (2017). En general, estas iniciativas disminuyen el costo computacional asociado al número de muestras y mejoran la generalización, pero el costo computacional total puede elevarse dependiendo del número de lotes y nodos ocultos. Comúnmente el número de nodos ocultos en ELM es elevado debido a que éstos se seleccionan aleatoriamente, lo que ocasiona problemas de procesamiento en aplicaciones que involucran grandes cantidades de datos (Cai, 2018). Buscando superar este inconveniente, en este documento se propone un nuevo modelo de entrenamiento para redes SLFN llamado máquina de entrenamiento comprimido basada en ELM o MEC-ELM, el cual actúa comprimiendo la información proveniente de la capa oculta por medio de un subconjunto de nodos ocultos denominados nodos de compresión, esto permite reducir considerablemente los requerimientos computacionales en relación a la ELM.

METODOLOGÍA

Este estudio inicia con la descripción de los fundamentos teóricos de ELM, luego detalla el planteamiento matemático de MEC-ELM, posteriormente se presenta un análisis comparativo entre ELM y MEC-ELM basado en simulación para diferentes tareas de clasificación.

Máquina de Aprendizaje Extremo

La máquina de aprendizaje extremo o ELM (Huang et al., 2006; Huang et al., 2012) es un modelo de red neuronal SLFN en el que los nodos de la capa oculta son seleccionados aleatoriamente y los nodos de la capa de salida se determinan analíticamente. Para una entrada arbitraria $x_j \in R^n$, la salida de un modelo ELM con *L* nodos ocultos está dada como sigue a continuación;

$$f_L(x_j)=\sum_{i=1}^{L}\beta_i\, g(<w_i.x_j>+b_i) \tag{1}$$

donde $w_i=[w_{i,1}\ \ w_{i,2}\ \dots\ w_{i,n}]^T$ es el vector de pesos entre el *i* ésimo nodo oculto y la capa de entrada; $\beta_i=[\beta_{i,1}\ \ \beta_{i,2}\ \dots\ \beta_{i,m}]$ es el vector de pesos entre el *i* ésimo nodo oculto y la capa de salida; b_i el bias del *i* ésimo nodo oculto; g(.) es la función de activación de los nodos ocultos y $<w_i.x_i>$ denota el producto punto entre w_i y x_i (Huang et al., 2006).

Ahora bien, sea $\eta=\{(x_j, t_j) \mid x_j \in R^n, t_j \in R^m$, para $j=1,2,\dots,N\}$ un conjunto de *N* entradas con sus respectivas salidas deseadas. Entonces, con base en la ecuación (1), la salida para todas las muestras del conjunto η puede ser expresada como una matriz $F \in R^{N\times m}$ tal que

$$F=HB \tag{2}$$

donde, $B \in R^{L\times m}$ es la matriz de pesos de salida tal que $B=[\beta_1\ \ \beta_2\ \dots\ \beta_L]^T$ y $H \in R^{N\times L}$ la matriz de salida de la capa oculta dada por

$$H=\begin{bmatrix} g(w_1.x_1+b_1) & \cdots & g(w_L.x_1+b_L) \\ \vdots & \ddots & \vdots \\ g(w_1.x_N+b_1) & \cdots & g(w_L.x_N+b_L) \end{bmatrix} \tag{3}$$

Para calcular la matriz de pesos de salida B, se propuso inicialmente minimizar el error de entrenamiento, esto es

$$B=\min_{\dot{B}}\left\|H\dot{B}-T\right\|^2 \tag{4}$$

donde $T \in R^{N\times m}$ definida como

$$T=[t_1\quad t_2\quad \dots\quad t_N]^T \tag{5}$$

es la matriz de salidas deseadas y $\dot{B} \in R^{L\times m}$ cualquier configuración posible de pesos de salida.

La solución original de ELM (Huang et al., 2006) se obtiene al resolver el problema planteado en la ecuación (4) y está dada por

$$B=H^{\dagger}T \tag{6}$$

donde $H^{\dagger}$ es la pseudoinversa de *Moore-Penrose* de la matriz H**.**

Sin embargo, desde la perspectiva de la teoría de optimización, el objetivo de ELM es minimizar a la vez el error de entrenamiento y la norma de los pesos de salida, esto es:

$$B=\min_{\dot{B}}\left(\frac{1}{2}\left\|\dot{B}\right\|^2+\frac{C}{2}\left\|H\dot{B}-T\right\|^2\right) \tag{7}$$

donde, *C* es un parámetro de regularización. Esto se justifica en la teoría de Barlett (1998) según la cual, entre más pequeños sean el error de entrenamiento y la norma de los pesos de salida, mejor tiende a ser la generalización.

Con base en lo anterior, la solución de ELM conocida también como solución regularizada de ELM se obtiene al resolver la ecuación (7) (Huang et al., 2012) y se define como

$$B=\begin{cases} H^T\left(\frac{I_{LxL}}{C}+HH^T\right)^{-1}T & \text{si} \quad N<L \\ \left(\frac{I_{LxL}}{C}+H^TH\right)^{-1}H^TT & \text{si} \quad N\geq L. \end{cases} \tag{8}$$

donde $I_{L\,x\,L}$ es una matriz identidad tal que $I_{L\,x\,L} \in R^{L\,x\,L}$.

El proceso de entrenamiento en ELM puede ser resumido en tres pasos: primero, se seleccionan aleatoriamente los pesos w_i y bias b_i de la capa oculta para i=1,2,…,L, luego se calcula la matriz de salida de la capa oculta H y finalmente se determina la matriz de pesos de salida B usando la ecuación (8). En relación a la capacidad de aproximación de un modelo ELM, Huang et al. (2006) demuestran el siguiente resultado teórico:

Teorema 1. *Dado un pequeño valor positivo* $\varepsilon>0$*, para cualquier función de activación* $g:\rightarrow R$ *infinitamente diferenciable en algún intervalo y N distintas muestras* (x_j, t_j) *seleccionadas arbitrariamente, donde* $x_j \in R^n$ y $t_j \in R^m$ *existe* $L\leq N$ *tal que, si* (w_i, b_i) *para* i=1,2,…,L *son generados aleatoriamente a partir de cualquier intervalo* R^nxR *de acuerdo con cualquier función de probabilidad continua, entonces con probabilidad uno,* $||HB-T||\leq\varepsilon$*. Mas aún, si* $L=N$ *con probabilidad uno* H *es invertible y* $||HB-T||=0$. El teorema 1 garantiza que para cualquier conjunto de entrenamiento siempre existirá un modelo ELM que asocie los patrones con un error de salida lo suficientemente pequeño (en el sentido de los mínimos cuadrados). Además, que el número de nodos ocultos de dicho modelo no es mayor al número de muestras de entrenamiento distintas y en el caso de que el número de nodos ocultos sea igual al número de muestras distintas, con probabilidad uno el error de entrenamiento será cero. Por consiguiente, ELM puede ajustar cualquier conjunto de entrenamiento siempre y cuando el número de nodos ocultos de la red sea lo suficientemente grande.

Máquina de Entrenamiento Comprimido Basada en ELM

Considérese inicialmente la solución óptima de ELM para el caso en que $N\geq L$, esto es

$$B=\left(\frac{I_{L\times L}}{C}+H^TH\right)^{-1}H^TT \tag{9}$$

y un conjunto de entrenamiento $\eta_q=\{(x_j, h_j) \mid x_j \in R^n, h_j \in R^L$, para j=1,2,…,$N\}$ tal que

$$H=[h_1 \quad h_2 \quad \dots \quad h_N]^T \tag{10}$$

Entonces, con base en el teorema 1, las muestras del conjunto de entrenamiento η_q pueden ser aproximadas tanto como se desee usando un modelo ELM auxiliar de $q \leq N$ nodos ocultos. Por simplicidad, en adelante los nodos ocultos de este segundo modelo serán referidos como nodos de compresión. Así entonces, siempre existirá un modelo ELM auxiliar tal que

$$||QD-H||\leq\varepsilon_0 \tag{11}$$

donde $Q \in R^{N\,x\,q}$ es la matriz de salida de los nodos de compresión, $D \in R^{q\,x\,L}$ la matriz de pesos de salida de los nodos de compresión y ε_0 un valor real positivo tan pequeño como se quiera.

La ecuación (11) sugiere que se pueden seleccionar un conjunto de nodos de compresión que permitan comprimir la información proveniente de la capa oculta de un modelo ELM en función de la matriz de pesos D para calcular una aproximación B_ϵ de la ecuación (9), esto es

$$B_\epsilon=\left(\frac{I_{L\times L}}{C}+U_a\right)^{-1}H^TT \tag{12}$$

donde

$$U_a=(QD)^TH\approx H^TH \tag{13}$$

Ahora, remplazando la ecuación (13) en la ecuación (12)

$$B_\epsilon=\left(\frac{I_{L\times L}}{C}+D^TQ^TH\right)^{-1}H^TT \tag{14}$$

o equivalentemente

$$B_\epsilon=V^{-1}H^TT \tag{15}$$

donde

$$V=\frac{I_{LxL}}{C}+D^TQ^TH \tag{16}$$

Posteriormente, aplicando la fórmula de inversión matricial de Sherman-Morrison-Woodbundry (Golub y Van Loan, 2013) en la ecuación (16)

$$\begin{aligned} V^{-1}&=CI_{LxL}-C^2D^T\left(I_{q\times q}+CQ^THD^T\right)^{-1}Q^TH \\ &=CI_{LxL}-CD^T\left(\frac{I_{q\times q}}{C}+Q^THD^T\right)^{-1}Q^TH. \end{aligned} \tag{17}$$

Al remplazar la ecuación (17) en la ecuación (15) se obtiene el modelo general de una Maquina de Entrenamiento Comprimido Basada en ELM o MEC-ELM.

$$B_\epsilon=CHT-CD^T\left(\frac{I_{q\times q}}{C}+Q^THD^T\right)^{-1}Q^THHT \tag{18}$$

Ahora bien, para calcular la matriz de pesos de compresión D se plantea usar la solución original de ELM (ecuación (4)) sobre el conjunto de entrenamiento η_q. Esto es, minimizando ||QD-H|| como sigue

$$D=\min_{\dot{D}}\left\|Q\dot{D}-H\right\|^2 \tag{19}$$

de modo que, si Q^ι es la *pseudoinversa de More Penrose* de la matriz Q**,** entonces

$$D=Q^\iota H \tag{20}$$

Por otro lado, si el rango de Q es igual a *q* entonces Q^TQ es no singular y D puede calcularse como

$$D=(Q^TQ)^{-1}Q^TH. \tag{21}$$

La solución B_ϵ con *q* nodos de compresión es una aproximación de la solución óptima B de un modelo ELM con
L nodos ocultos. En relación al número de nodos de compresión, el teorema 1 garantiza que la matriz H puede ser aproximada sin error usando la solución original de ELM con *L=N* nodos ocultos. Por lo tanto, el teorema 1 también asegura la existencia de una solución B_ϵ con *q=N* nodos de compresión seleccionados aleatoriamente tal que

$$B_\epsilon=B \tag{22}$$

No obstante, para que se satisfaga la ecuación (22) solo son necesarios *L* nodos de compresión. Esto se puede comprobar fácilmente seleccionando como nodos de compresión los mismos nodos de la capa oculta. De esta forma *q=L*, Q=H, $D=I_{L\,x\,L}$ y la ecuación (18) se reduce a la ecuación (9).

Lo anterior sugiere que los nodos de compresión deben seleccionarse entre los nodos ocultos. Siguiendo esta idea la matriz de salidas de la capa oculta H puede ser expresada como una matriz partida, esto es

$$H=[H_{oc} \mid H_o] \tag{23}$$

donde, $H_{oc} \epsilon R^{N\,x\,q}$ es la matriz de salida de los nodos de la capa oculta que a su vez son nodos de compresión y $H_o \epsilon R^{N\,x\,(L-q)}$ es la matriz de salida de los nodos ocultos que no son nodos de compresión. Así entonces

$$Q=H_{oc} \tag{24}$$

y la ecuación (21) puede reescribirse como

$$D=(H_{oc}^T H_{oc})^{-1} H_{oc}^T [H_{oc} \mid H_o] \quad (25)$$

o equivalentemente

$$D=[I_{q\times q} \mid V_{oc}^{-1} W_{oc}] \quad (26)$$

donde

$$V_{oc}=H_{oc}^T H_{oc} \quad (27)$$

y

$$W_{oc}=H_{oc}^T H_o \quad (28)$$

Finalmente, la solución B_ϵ planteada en la ecuación (18) se puede reescribir como

$$B_\epsilon=CW-CD^T\left(\frac{I_{q\times q}}{C}+ZD^T\right)^{-1} ZW \quad (29)$$

donde

$$W=H^T T \quad (30)$$

y

$$Z=[V_{oc} \mid W_{oc}]. \quad (31)$$

El proceso de aprendizaje de una Maquina de Entrenamiento Comprimido Basada en ELM se presenta en la tabla 1.

Tabla 1: Entrenamiento en MEC-ELM

	Entradas	$\eta=\{(x_j, t_j) \mid x_j \in R^n, t_j \in R^m$, para $j=1,2,\ldots,N\}$ // Datos de Entrenamiento.
		g(.) // Funciones de activación de los nodos ocultos.
		L // Número de nodos ocultos.
		q // Número de nodos de compresión.
		C // Constante de regularización.
	Salidas	$\{(w_i, b_i) \mid w_i \in R^n, b_i \in R$, para $i=1,2,\ldots,L\}$ // Datos de Entrenamiento.
		B_ϵ // Solución MEC-ELM para pesos de salida.
1		Seleccionar aleatoriamente los pesos w_i y b_i para $i=1,2,\ldots,L$
2		Calcular la matriz H de acuerdo con la ecuación (3).
3		Identificar H_{oc} y H_o según la ecuación (23) y calcular V_{oc} y W_{oc} como: $V_{oc}=H_{oc}^T H_{oc}$ $W_{oc}=H_{oc}^T H_o$
4		Calcular $V_{oc}^{-1} W_{oc}$
5		Conformar las matrices D y Z según las ecuaciones (26) y (31) respectivamente.
6		Calcular W como $W=H^T T$, donde T es la matriz de salidas deseadas indicada en la ecuación (5).
7		Calcular B_ϵ utilizando la ecuación (29).

Consideraciones en relación a los requerimientos computacionales

El modelo de entrenamiento comprimido es un esquema de aprendizaje propuesto para aplicaciones que involucran grades conjuntos de datos. En este tipo de escenarios las máquinas de entrenamiento extremo han ganado gran interés puesto que su solución se puede calcular en forma relativamente rápida (Akusok et al., 2015). Por lo anterior resulta interesante analizar los requerimientos computacionales y de almacenamiento de MEC-ELM en contraste con los que demanda ELM. Para esto considérese la tabla 2, en

la que se presentan todas las operaciones que intervienen en el cálculo de la solución para ambos modelos junto a sus respectivos órdenes de complejidad computacional y de almacenamiento.

En la tabla 2 se asume que el número de muestras de entrenamiento es mayor al número de nodos ocultos, es decir $N \geq L$, teniendo en cuenta que bajo esta condición se propuso el modelo MEC-ELM. En lo que respecta a MEC-ELM, las operaciones con mayor orden computacional son según el caso: calcular la matriz H si $n>q$ y $n>m$; calcular la matriz W_{oc} si $q>n$ y $q>m$ o bien calcular W si $m>q$ y $m>n$. Con lo que la complejidad computacional de MEC-ELM está dada por $O(NL\max(q,n,m))$. En la misma tabla 2, se puede observar que la complejidad computacional de ELM cuando $N \geq L$ está dada por $O(NL\max(L,m,n))$. Esto implica que los requerimientos computacionales de MEC-ELM se pueden reducir en relación a los de ELM dependiendo del número de nodos de compresión siempre y cuando $m<q$ y $n<q$.

Tabla 2: Requerimientos computacionales y de almacenamiento para MEC-ELM y ELM cuando N≥L.

Operación	Complejidad Computacional		Requerimientos memoria por Lotes $\tilde{N}<N$	
	MEC-ELM	ELM	MEC-ELM	ELM
Dat. $\{x_j \mid x_j \in R^n, \text{ para } j=1, 2, N\}$	$O(Nn)$		$O(\tilde{N}n)$	
H ecuación (3)	$O(NLn)$		$O(\tilde{N}L)$	
$V_{oc}=H_{oc}^T H_{oc}$	$O(Nq^2)$	--	$O(q^2)$	--
$W_{oc}=H_{oc}^T H_o$	$O(NLq)$	--	$O(Lq)$	--
$D=[I_{qxq} \mid V_{oc}^{-1} W_{oc}]$	$O(Lq^2)$	--	$O(Lq)$	--
$Z=[V_{oc} \mid W_{oc}]$	--	--	--	--
$W=H^T T$	$O(NLm)$		$O(Lm)$	
$A=I_{q \times q}/C+ZD^T$	$O(Lq^2)$	--	$O(q^2)$	--
$B_\epsilon=CW-CD^T(A)^{-1}ZW$	$O(L^2q+L^2m)$	--	$O(L^2)$	--
$S=I_{L \times L}/C+H^TH$	--	$O(NL^2)$	--	$O(L^2)$
$B=S^{-1}W$	--	$O(L^3+L^2m)$	--	$O(L^2+Lm)$
Total	$O(NL\max(q,n,m))$	$O(NL\max(L,m,n))$	$O(L^2+Lm)$	$O(L^2+Lm)$

Lo anterior resulta conveniente teniendo en cuenta que en el contexto de las redes multicapa determinar el número de nodos ocultos L es un problema que aún no está lo suficientemente esclarecido. Como consecuencia en la literatura solo suele especificarse que se debe usar un número suficiente de nodos en la capa oculta, por lo que en la práctica generalmente se selecciona de inicio un número grande de nodos ocultos. Cabe anotar que, si bien seleccionar un número elevado de nodos ocultos en algunas aplicaciones puede disminuir el rendimiento en términos de generalización, esto no representa un gran problema cuando se cuenta con grandes conjuntos de entrenamiento puesto que la gran cantidad de datos evita el sobreajuste.

En lo que respecta a los requerimientos de memoria, nótese en la tabla 2 que estos son constantes para MEC-ELM y ELM independientemente del número de muestras de entrenamiento N, ya que las matrices H^TH, $H_{oc}^T H_{oc}$, $H_{oc}^T H_o$ y H^TT pueden ser calculadas por lotes de $\tilde{N}<N$ muestras, de modo que al final el resultado se puede obtener sumando los resultados parciales de cada lote. Esto no incrementa el costo computacional dado que a nivel de hardware y software la multiplicación y suma de matrices se implementan en una única operación (Golub y Van Loan, 2013).

RESULTADOS Y DISCUSIÓN

En esta sección se comparan los resultados de MEC-ELM y los obtenidos con ELM para diferentes problemas de clasificación. Todas las simulaciones se realizaron en un procesador AMD de 2.70 GHz y haciendo uso del lenguaje de programación *R*. El lenguaje de programación *R* cuenta con paquetes especializados para el tratamiento de datos y "*Machine Learning*" entre los que se incluye el paquete CARET ("*Classification And REgression Training*") que dispone de funciones para el diseño y evaluación de modelos predictivos, así como para pre-procesamiento, selección de características entre muchas más.

Las implementaciones de MEC-ELM y ELM utilizadas se diseñaron de modo que fueran compatibles con el paquete CARET. El rendimiento de MEC-ELM se puso a prueba utilizando una colección de conjuntos de entrenamiento relacionados con aplicaciones de clasificación binaria y multiclase tomados de *UCI Machine Learning Repository* (Dua y Taniskidou, 2017) y el portal LIBSVM (Chang y Lin, 2011). La información referente al número de datos, atributos y clases se presentan en la Tabla 3.

Tabla 3: Información relacionada con los de datos utilizados en las simulaciones.

Datos de datos	# de muestras de entrenamiento	# de muestras de prueba	Atributos	Clases
Mushroom	4062	4062	22	2
SVDguide1	3089	4000	4	2
Magic	9510	9510	10	2
COD RNA	29768	29767	8	2
Spambase	2301	2300	57	2
Adult	6414	26147	14	2
SatImage	4435	2000	36	7
Segment	1155	1155	18	7

Para empezar, cada conjunto de datos se normalizó de tal modo que las entradas tengan media cero y varianza uno. En lo que respecta a las etiquetas de clase, la *k* ésima clase se representó por un vector *m* dimensional en el que todos los elementos eran -1 excepto el *k* ésimo elemento del vector, siendo *m* el número total de clases. Todos los modelos utilizados en las simulaciones eran de 1000 nodos ocultos con funciones activación tangente hiperbólica $g(a)=(e^{a}-e^{-a})/(e^{a}+e^{-a})$. Para los modelos MEC-ELM simulados el número de nodos de compresión se fijó en la mitad del número de nodos ocultos, es decir 500 en todos los casos.

El parámetro de regularización *C* de MEC-ELM y ELM se seleccionó utilizando validación cruzada de 10 iteraciones. En este proceso se probaron los siguientes valores de *C* $\{2^{3}, 2^{1}, 2^{-1}, 2^{-3}, 2^{-5}, 2^{-7}, 2^{-9}, 2^{-11}, 2^{-13}\}$. Posteriormente, se realizaron cincuenta simulaciones con MEC-ELM y cincuenta simulaciones con ELM. Para medir el rendimiento se obtuvo el porcentaje de correcta clasificación en la fase de entrenamiento y en la fase de prueba. El promedio y la respectiva desviación estándar de los resultados para cada conjunto de datos se presentan en la tabla 4. Los resultados de rendimiento en entrenamiento y prueba también se pueden contrastar con los obtenidos por Inaba et al (2017).

Tabla 4: Resultados de simulación de MEC-ELM y ELM para cada conjunto de datos

Datos de Entrenamiento	*Modelo*	*Rendimiento (%)*				*Tiempo (s)*		*C*
		Entrenamiento		*Prueba*				
		Tasa	*Dev. Est*	*Tasa*	*Dev. Est*	*Entrenam.*	*Prueba*	
Mushroom	MEC-ELM	100	0.00	99.99	9.59×10^{-5}	8.91	1.60	2^{-7}
	ELM	100	0	99.99	5.69×10^{-5}	15.55	1.67	2^{-1}
SVDguide1	MEC-ELM	97.36	0.04	96.61	0.05	6.90	1.053	2^{-1}
	ELM	97.37	0.04	96.61	0.06	11.99	1.13	2^{-1}
Magic	MEC-ELM	87.20	0.09	86.47	0.11	15.81	2.32	2^{-5}
	ELM	88.22	0.08	86.54	0.14	31.48	2.31	2^{-1}
COD RNA	MEC-ELM	95.18	0.04	95.09	0.06	42.63	6.49	2^{-3}
	ELM	95.22	0.04	95.15	0.05	94.85	6.54	2^{-3}
Spambase	MEC-ELM	95.65	0.22	93.01	0.27	5.67	0.58	2^{-7}
	ELM	95.35	0.18	93.09	0.26	8.95	0.57	2^{-7}
Adult	MEC-ELM	84.93	0.15	83.63	0.09	12.01	9.18	2^{-13}
	ELM	84.92	0.14	83.61	0.09	22.08	9.11	2^{-13}
Segment	MEC-ELM	98.85	0.05	95.98	0.34	4.09	0.28	2
	ELM	98.57	0.05	95.95	0.32	5.29	0.28	2
Satimage	MEC-ELM	93.68	0.16	90.16	0.31	8.79	0.47	2^{-5}
	ELM	95.86	0.15	90.16	0.28	15.40	0.6184	2^{3}

El rendimiento en la fase de prueba permite estimar qué tan buena es la generalización de un modelo, es decir, qué tan bien responde ante patrones que no han sido utilizados en el entrenamiento. Los resultados indican que MEC-ELM y ELM alcanzaron rendimientos similares en la fase de prueba y como tal que ambos modelos alcanzaron rendimientos similares en términos de generalización.

En lo que respecta al rendimiento en la fase de entrenamiento, este indica que tan acertado es el modelo al clasificar las muestras utilizadas en el entrenamiento. Los resultados reflejan que el rendimiento de ELM en entrenamiento es levemente mayor en comparación con MEC-ELM. Sin embargo, cabe mencionar que en la práctica lo que se busca es que un modelo aprenda a dar respuestas correctas ante patrones jamás vistos en el entrenamiento, es decir, que generalice. En la tabla 4 también se muestra los tiempos que en promedio tomaron la fase de prueba y la fase de entrenamiento para todos los casos de clasificación tratados. Nótese que los tiempos de entrenamiento de MEC-ELM son significativamente menores a los de ELM y como era de esperarse, los tiempos en la fase de pruebas en ambos modelos es relativamente igual.

CONCLUSIONES

En relación al modelo MEC-ELM propuesto en este documento y a los resultados de simulación obtenidos, se pueden establecer las siguientes conclusiones; 1) MEC-ELM actúa comprimiendo la información proveniente de la capa oculta por medio de un subconjunto de nodos de la misma capa llamados nodos de compresión, esto permite bajo ciertas condiciones ($n<q<N$ y $m<q$) que la complejidad computacional en relación a ELM pueda ser reducida dependiendo del número de nodos de compresión utilizados; y 2) Experimentos basados en simulación para diferentes problemas de clasificación binaria y multiclase indican que MEC-ELM reduce considerablemente los tiempos de entrenamiento en comparación con ELM manteniendo rendimientos similares en términos de generalización.

NOTACIÓN

Símbolos

R	Conjunto de los números reales.
$.^T$	Denota transpuesta de un vector o de una matriz
E	Mayúscula sin cursiva denota Matriz
e	Minúscula sin cursiva denota Vector
E, e	Minúscula o mayúscula con cursiva denota escalar
<e.v>	Producto escalar entre el vector e y el vector v
$I_{r \times r}$	Matriz identidad de dimensión *r* x *r*
$E^{\dagger}$	Pseudoinversa de *More Penrose* de la matriz E
x_j	*j* ésimo patrón de entrada
t_j	*j* ésimo patrón de salida
w_i	Vector de pesos de entrada del *i* ésimo nodo oculto
b_i	Bias del *i* ésimo nodo oculto
β_i	Vector de pesos entre el *i* ésimo nodo oculto y la capa de salida
H	Matriz de salidas de la capa oculta
Q	Matriz de salida de los nodos de compresión
T	Matriz de salidas deseadas
B	Matriz de pesos de salida en ELM
D	Matriz de pesos de salida de los nodos de compresión
B_ϵ	Matriz de pesos de salida en MEC-ELM
N	Número de datos de entrenamiento
n	Dimensión del espacio de entrada
L	Número de nodos ocultos
q	Número de nodos de compresión
m	Dimensión del espacio de salida o número de clases
C	Constante de regularización en ELM y MEC-ELM

Abreviaciones

ELM	Máquina de aprendizaje extremo ("*Extreme Learning Machine*")
OS-ELM	Máquina de entrenamiento secuencial en línea ("*Online Sequential ELM"*)
MEC-ELM	Máquina de aprendizaje comprimido basada en ELM
SLFN	Redes neuronales no realimentadas de una sola capa oculta ("*Single-hidden Layer Feedforward Networks"*)

REFERENCIAS

Akusok, A., K. M. Björk y otros dos autores, High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications, doi: 10.1109/ACCESS.2015.2450498, IEEE Access 3, 1011-1025 (2015)

Babu, G. S. y S. Suresh, Meta-Cognitive Neural Network for Classification Problems in a Sequential Learning Framework, doi: 10.1016/j.neucom.2011.12.001, Neurocomputing 81, 86-96 (2012)

Bartlett, P. L., The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is more Important than the Size of the Network, doi: 10.1109/18.661502, IEEE Trans. Inf. Theory 44 (2), 525-536 (1998)

Cai, Y., X. Liu y otros dos autores, Hierarchical Ensemble of Extreme Learning Machine, doi: 10.1016/j.patrec.2018.06.015, Pattern Recognition Letters (2018)

Chang, C. C. y C. J. Lin, LIBSVM: A Library for Support Vector Machines, ACM Transactions on Intelligent Systems and Technology, 2:27:1-27:27 (2011)

Chen, C., X. Li y otros dos autores, Extreme Learning Machine and Its Applications in Big Data Processing, doi: 10.1016/B978-0-12-809393-1.00006-4, Big Data Analytics for Sensor-Network Collected Intelligence, 117-150 (2017)

Cortes, C. y V. Vapnik, Support-Vector Networks, Machine learning, 20.3, 273-297 (1995)

Dua, D. y K. Taniskidou, UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science (2017)

Golub, G. y C. Van Loan, Matrix computations. 4ª Ed., The Johns Hopkins University Press (2013)

Heeswijk, M., Y. Miche y otros dos autores, GPU-Accelerated and Parallelized ELM Ensembles for Large-Scale Regression, doi: 10.1016/j.neucom.2010.11.034, Neurocomputing, 74.16, 2430-2437 (2011)

Huang, G.B. y L. Chen, Convex Incremental Extreme Learning Machine, doi: 10.1016/j.neucom.2007.02.009, Neurocomputing, 70.16-18, 3056-3062 (2007)

Huang, G. B., H. Zhou y otros dos autores, Extreme Learning Machine for Regression and Multiclass Classification, doi: 10.1109/TSMCB.2011.2168604, IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 42.2, 513-529 (2012)

Huang, G. B., N. Y. Liang y otros tres autores, On-Line Sequential Extreme Learning Machine, Computational Intelligence 2005, 232-237 (2005)

Huang, G.B., Q.Y. Zhu y C.K., Siew, Extreme Learning Machine: Theory and Applications, doi: 10.1016/j.neucom.2005.12.126, Neurocomputing Intelligence, 70.1-3, 489-501 (2006)

Huang, G.B., X. Ding y H., Zhou, Optimization Method Based Extreme Learning Machine for Classification, doi: 10.1016/j.neucom.2010.02.019, Neurocomputing, 74.1-3, 155-163 (2010)

Huang, G., G.B. Huang y otros dos autores, Trends in Extreme Learning Machines: A Review, doi: 10.1016/j.neunet.2014.10.001, Neural Networks, 61, 32-48 (2015)

Inaba, F. K., E. O. T. Salles y otros dos autores, DGR-ELM–Distributed Generalized Regularized ELM for classification, doi: 10.1016/j.neucom.2017.09.090, Neurocomputing, 275, 1522-1530 (2017)

Krawczyk, B., GPU-Accelerated Extreme Learning Machines for Imbalanced Data Streams with Concept Drift, doi: 10.1016/j.procs.2016.05.509, Procedia Computer Science, 80, 1692-1701 (2016)

Qian, K., Active Learning for Bird Sound Classification Via a Kernel-Based Extreme Learning Machine, doi: 10.1121/1.5004570, The Journal of the Acoustical Society of America, 142.4, 1796-1804 (2017)

Settles, B., Active Learning, doi: 10.2200/S00429ED1V01Y201207AIM018, Synthesis Lectures on Artificial and Machine Learning, 6.1, 1-114, (2012)

Sun, Y., Y. Yuan y G. Wang, An OS-ELM Based Distributed Ensemble Classification Framework in P2P Networks, doi: 10.1016/j.neucom.2010.12.040, Neurocomputing, 74.16, 2438-2443 (2011)

Suykens, J.A.K. y J. Vandewalle, Least Squares Support Vector Machine Classifiers, Neural Processing Letters, 9.3, 293-300 (1999)

Torrecilla, J. L. y J., Romo, Data Learning from Big Bata, doi: 10.1016/j.spl.2018.02.038, Statistics & Probability Letters, 136, 15-19 (2018)

Valenzuela, S.A., C.L. y otros dos autores, Ejemplos de Aplicabilidad de Giraph y Hadoop para el Procesamiento de Grandes Grafos, doi: 10.4067/S0718-07642016000500019, Información Tecnológica, 27(5), 171-180 (2016)

Vidal, C. L., M. A. Bustamante y otros dos autores, En la Búsqueda de Soluciones MapReduce Modulares para el Trabajo con BigData: Hadoop Orientado a Aspectos, doi: 10.4067/S0718-07642018000200133, Información Tecnológica, 29(2), 133-140 (2018)

Wang, H., X. Zeshui y otros dos autores, Towards Felicitous Decision Making: An Overview on Challenges and Trends of Big Data, doi: 10.1016/j.ins.2016.07.007, Information Sciences, 367, 747-765 (2016)

Xin, J., Z. Wang y otros cuatro autores, ELM∗: Distributed Extreme Learning Machine with MapReduce, doi: 10.1007/s11280-013-0236-2, World Wide Web, 17.5, 1189-1204 (2014)

Xin, J., Z. Wang y otros dos autores, Elastic Extreme Learning Machine for Big Data Classification, doi: 10.1016/j.neucom.2013.09.075, Neurocomputing, 149, 464-471 (2015)

Zhang, Y. y M. Joo, Sequential Active Learning Using Meta-Cognitive Extreme Learning Machine, doi: 10.1016/j.neucom.2015.08.037, Neurocomputing, 173, 835-844 (2016)