Entrenamiento de máquinas de aprendizaje extremo con el algoritmo acelerador

Phd. Pablo Emilio Jojoa Gómez
Universidad del Cauca - Colombia
pjojoa@unicauca.edu.co

Phd (c). Fausto Miguel Castro Caicedo
Universidad del Cauca - Colombia
faustocastro@unicauca.edu.co

Fecha Recepción: 08/11/15 - Fecha Aprobación: 29/11/15

Resumen: En este artículo se presenta un nuevo algoritmo para el entrenamiento de redes neuronales no realimentadas de una sola capa oculta el cual combina las propiedades de la máquina de aprendizaje extremo con las del algoritmo ARγ. El algoritmo propuesto presenta una baja complejidad computacional y las pruebas realizadas muestran un alto desempeño en términos de generalización.

Palabras clave: Máquina de aprendizaje extremo, Algoritmo ARγ, redes neuronales sin realimentación, filtros adaptativos.

Abstract: This paper presents a new algorithm for training neural networks fed back not one hidden layer which combines the properties of the end machine learning algorithm with ARγ occurs. The proposed algorithm has low computational complexity and tests show a high performance in terms of generalization.

Keywords: Extreme learning machine, ARγ algorithm, neural networks without feedback, adaptive filters.

1. Introducción

Las redes neuronales artificiales han sido aplicadas extensivamente en muchos campos debido a su habilidad para aproximar complejos mapeos no lineales, a partir de muestras de entrada y por su capacidad para modelar una gran variedad de fenómenos naturales y artificiales que difícilmente podrían manejarse usando técnicas paramétricas [1]. La máquina de aprendizaje extremo ELM (*Extreme Learning Machine*) es un algoritmo para redes neuronales no realimentadas de una única capa oculta SLFN (*Single Layer Feedforwared Network*) que supera los principales problemas de los algoritmos basados en gradiente, como la presencia de mínimos locales, la imprevisibilidad de la tasa de aprendizaje y la lenta velocidad de convergencia [2]. El algoritmo ELM implementa una red SLFN similar al modelo tradicional, con la única diferencia de que los pesos de la capa oculta se escogen aleatoriamente y solo se determinan analíticamente los pesos de la capa de salida [3], esto permite transformar el entrenamiento (que tradicionalmente ha sido un problema no lineal) en un problema lineal, con lo que se reduce significativamente la complejidad computacional [4].

Por su parte, el algoritmo Acelerador Regresivo Versión Gamma ARγ es utilizado en filtrado adaptativo para ajustar los parámetros de estructuras de respuesta finita al impulso FIR (*Finite Impulse Response*) [5]. Este algoritmo ajusta la segunda derivada de parámetros alcanzando altos niveles de velocidad de convergencia y paralelamente una reducción en la medida del error final, superando considerablemente los resultados de otros algoritmos propuestos para la misma tarea como el algoritmo de Mínimos Cuadrados Medio LMS (*Least Mean Squared*), y el Algoritmo de Mínimos Cuadrados Medio Normalizado NLMS (*Normalized Least Mean Squared*) [6].

De acuerdo con lo anterior, ambos algoritmos (ELM y ARγ) presentan características sobresalientes en sus respectivos campos de aplicación. Por un lado

la baja complejidad computacional de ELM aplicado a la solución de problemas no lineales, y por otro la alta velocidad de convergencia del ARγ ajustando estructuras lineales tipo FIR. Así entonces, resulta atractivo combinar las propiedades de los algoritmos ELM y ARγ buscando mantener sus respectivas ventajas, surgiendo así el algoritmo propuesto en este documento. Sin embargo, no se debe olvidar que el fin último de una red neuronal es generalizar, es decir, dar respuestas adecuadas ante entradas no utilizadas en el entrenamiento y no simplemente memorizar el conjunto de muestras. Cabe destacar que en las pruebas realizadas la implementación propuesta alcanza muy buenos resultados de generalización.

Este documento se organiza en cinco secciones: La sección 2 introduce los conceptos teóricos fundamentales de la máquina de aprendizaje extrema y el algoritmo ARγ; la sección 3 describe la implementación propuesta; la sección 4 presenta un caso de aplicación relacionado con ajuste funcional que permite validar experimentalmente el algoritmo propuesto; en la sección 5 se presentan los resultados del estudio experimental y, finalmente, en la sección 6 se enuncian las conclusiones.

2. Conceptos Teóricos

2.1 Máquina de Aprendizaje Extremo

La máquina de aprendizaje extremo o ELM actualmente es un importante paradigma conceptual y computacional dentro de las redes neuronales puesto que su velocidad de convergencia y rendimiento en términos de generalización son muchísimo mejores en comparación con los tradicionales algoritmos basados en el método de gradiente [7], se trata de un eficiente método para el entrenamiento de SLFN, las cuales requieren por lo general un alto número de neuronas ocultas. La arquitectura de una red SLFN se presenta en la Figura 1.

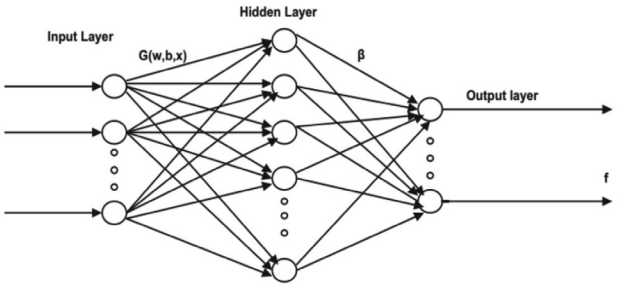

Figura 1. Modelo de Red Neuronal sin realimentación de una capa oculta [7].

Matemáticamente, la función de salida de una SLFN con $\tilde{N}$ nodos en la capa oculta se puede representar como:

$$f_j = \sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i g(\boldsymbol{w}_i, \boldsymbol{x}_j, b_i) = \sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i g(\boldsymbol{w}_i . \boldsymbol{x}_j + b_i) \qquad \boldsymbol{x}_j \in R^n, \ \boldsymbol{\beta}_i \in R^m \qquad (1)$$

donde $\boldsymbol{x_j}$ es el vector de entrada, $\boldsymbol{w_i}$ es el vector de pesos entre el nodo i oculto y los nodos de entrada, $\boldsymbol{w_i}.\boldsymbol{x_j}$ denota el producto punto entre $\boldsymbol{x_j}$ y $\boldsymbol{w_i}$, $b_i \in$ R es el umbral o bias del nodo i, $g(.)$ es la función de activación sigmoidal, $\boldsymbol{\beta}_i$ es el vector de pesos entre el nodo i oculto y los nodos de salida y f_j es la salida de la SLFN, $\tilde{N}$ es el numero de nodos ocultos y N el número de datos de entrenamiento. Los nodos de salida son lineales [8].

En el entrenamiento con ELM tanto $\boldsymbol{w_i}$ como b_i son seleccionados aleatoriamente.

La ecuación (1) puede ser expresada en forma matricial como:

$$\boldsymbol{AB=F} \qquad (2)$$

Donde:

$$\boldsymbol{A}(\boldsymbol{w}_1 \dots \boldsymbol{w}_L, b_1 \dots b_L, \boldsymbol{x}_1 \dots \boldsymbol{x}_N) = \begin{bmatrix} g(\boldsymbol{w}_1 \boldsymbol{x}_1 + b_1) & \dots & g(\boldsymbol{w}_{\tilde{N}} \boldsymbol{x}_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{w}_1 \boldsymbol{x}_N + b_1) & \dots & g(\boldsymbol{w}_{\tilde{N}} \boldsymbol{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N x \tilde{N}} \qquad (3)$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} x m} \qquad (4)$$

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}_1^T \\ \vdots \\ \boldsymbol{f}_N^T \end{bmatrix}_{N x m} \qquad (5)$$

La matriz $\boldsymbol{A}$ es llamada matriz de salida de la capa oculta del ELM, en la cual la i-ésima columna de la matriz $\boldsymbol{A}$ es el vector de salida del i-ésimo nodo oculto respecto a las entradas $x_1, x_2, x_3 \dots x_N$ y la j-ésima fila de la matriz $\boldsymbol{A}$ denotada como $\boldsymbol{a}_j$ es el vector de salida de la capa oculta con respecto a la entrada $\boldsymbol{x}_j$ [8].

Si la red SLFN con $\tilde{N}$ nodos ocultos puede aprender los N ejemplos de entrenamiento entonces existe un $\boldsymbol{B}$ tal que

$$\sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i g(\boldsymbol{w}_i . \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j \qquad j = 1,2,3 \dots \dots N \qquad (6)$$

donde $\boldsymbol{t}_j$ son las salidas deseadas, por tanto :

$$\boldsymbol{AB=T} \qquad (7)$$

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{Nxm} \qquad (8)$$

Definición 1: supóngase una SLFN con $\tilde{N}$ nodos ocultos cada uno con función de activación g:R→R infinitamente diferenciable en algún intervalo y pesos $\boldsymbol{w_i}$ y bias b_i seleccionados aleatoriamente en algún intervalo R^n y R respectivamente de acuerdo con alguna función de distribución continua. Para una SLFN de estas características y dado un conjunto de N muestras distintas $(\boldsymbol{x_j}, \boldsymbol{t_j})$ donde $\boldsymbol{x_j} \in R^n$ y $\boldsymbol{t_j} \in R^m$, en [1] se prueba rigurosamente que:

Si $\tilde{N}=N$ con probabilidad uno, la matriz $\boldsymbol{A}$ de la SLFN es invertible y $\|\boldsymbol{AB-T}\|=0$

Dado un valor positivo $\varepsilon>0$ pequeño existe un número de nodos ocultos $\tilde{N} \leq N$ tal que con probabilidad uno, $\|\boldsymbol{A}_{Nx\tilde{N}}\boldsymbol{B}_{\tilde{N}xm}-\boldsymbol{T}_{Nxm}\|<\varepsilon$.

Los anteriores resultados son la base teórica sobre la que se plantea el algoritmo ELM para redes SLFN. Este eficiente método de aprendizaje se resume a continuación.

Supóngase una SLFN con las características de la definición 1. Entonces:

1. Asignar aleatoriamente los pesos de entrada $\boldsymbol{w_i}$ y bias b_i de entrada para i=1,2,3,...,$\tilde{N}$

2. Calcular la matriz de salida de la capa oculta $\boldsymbol{A}$.

3. Calcular los pesos de salida $\boldsymbol{\beta}$ como

$$\boldsymbol{\beta}=\boldsymbol{A}^{\dagger}\boldsymbol{T} \qquad (9)$$

donde $\boldsymbol{A}^{\dagger}$ es la inversa generalizada de More-Penrose de la matriz $\boldsymbol{A}$ [1].

2.2 Filtros adaptativos y Algoritmo ARγ

Un filtro adaptativo puede entenderse como un sistema capaz de ajustar algunos de sus parámetros con el propósito de modelar su relación entrada salida, de tal manera que cumpla con un criterio definido por el propio sistema o por su entorno. La Figura 2 muestra el diagrama general de un filtro adaptativo, el cual está conformado por tres módulos fundamentales.

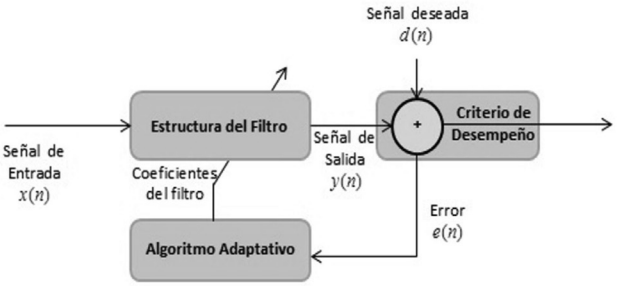

Figura 2. Diagrama general de un filtro adaptativo

Estructura del filtro: Define la forma en como la señal de salida será computada a partir de la señal de entrada, su funcionamiento está definido por un conjunto de coeficientes, los cuales pueden ser modificados para cumplir con una determinada relación de entrada salida. La estructura del filtro puede ser de tipo FIR o de tipo IIR.

Criterio de desempeño: Evalúa el rendimiento del sistema respecto a los requerimientos de una aplicación en particular, para lo cual usa alguna función de error que le permite comparar la salida del filtro con la señal deseada.

Algoritmo adaptativo: Es un procedimiento iterativo que permite ajustar los coeficientes del filtro en función del criterio de desempeño, de tal manera que se minimice la función de error.

En la literatura se pueden encontrar diferentes tipos de algoritmos adaptativos, basados en uno de los dos criterios más comunes de optimización, el error cuadrático medio (LMS, NLMS y **ARγ**) o los mínimos cuadrados (RLS y Kalman).

El algoritmo ARγ es una variante del algoritmo acelerador en tiempo continuo propuesta en 1998 que ajusta la segunda derivada de parámetros, el cual al ser discretizado genera tres versiones conocidas como algoritmo acelerador completo (AAC), algoritmo acelerador progresivo (APCM) y algoritmo acelerador regresivo (ARCM). Es destacable la elevada complejidad computacional del algoritmo APCM en relación con el buen desempeño del algoritmo ARCM y los excelentes resultados que arrojó el análisis de estabilidad para los algoritmos ARCM y AAC. Partiendo de esto, los esfuerzos se enfocaron en disminuir la complejidad de ARCM, con lo que surgió el algoritmo ARγ [5], que tiene como características el tener tres parámetros de

ajuste, lograr una buena velocidad de convergencia y paralelamente una considerable reducción del error de medida final [6]. Las ecuaciones que describen el algoritmo ARγ son:

$$e[n] = \boldsymbol{x}^T[n]\boldsymbol{w}[n-1] - d[n] \qquad (10)$$

$$\breve{g}[n] = \frac{e[n] + \gamma \boldsymbol{x}^T[n]\boldsymbol{w}[n-1]}{1 + \alpha\gamma m_1 \boldsymbol{x}^T[n]\boldsymbol{x}[n]} \qquad (11)$$

$$\boldsymbol{q}[n] = \frac{\gamma}{\alpha+\gamma}\left[\boldsymbol{q}[n-1] - \alpha m_1 \breve{g}[n]\boldsymbol{x}[n]\right] \qquad (12)$$

$$\boldsymbol{w}[n] = w[n-1] + \alpha \boldsymbol{q}[n] \qquad (13)$$

donde $\boldsymbol{x}[n]$es el vector de la señal de entrada, $d[n]$ un escalar correspondiente a la señal deseada, e[n] un escalar correspondiente a la señal de error, $\breve{g}$ [n] un escalar auxiliar en el instante n, $\boldsymbol{q}[n]$ un vector auxiliar del filtro adaptativo, $\boldsymbol{w}[n]$ es el vector de coeficientes del filtro adaptativo y las constantes α, γ y m_1 son parámetros de ajuste fijo. Los parámetros γ, α y m_1 de ajuste son cantidades escalares donde $\gamma>0$, $\alpha>0$ y $m_1>0$ para garantizar la convergencia del algoritmo [6].

3. Implementación propuesta: ELM entrenado con el algoritmo ARγ.

El procedimiento seguido para realizar la implementación se resume a continuación.

Dada una SLFN con las características de la definición 1. Entonces:

1. Asignar aleatoriamente los pesos de entrada $\boldsymbol{w_i}$ y bias b_i de entrada para i=1,2,3,...,$\tilde{N}$.
2. Para cada muestra j=1,2,3...N del conjunto de datos.
   2.1. Calcular el vector de salida de la capa oculta $\boldsymbol{a_j}$ y la salida general de la SLFN $\boldsymbol{f_j}$ para la muestra.
   2.2. Calcular el error de salida $\boldsymbol{e_j}$= $\boldsymbol{t_j}$ - $\boldsymbol{f_j}$
   2.3. Calcular el incremento parcial de los pesos con:

$$\boldsymbol{Q_j} = \frac{\gamma}{\alpha+\gamma}\left[\boldsymbol{Q_{j-1}} - \alpha m_1 \breve{\boldsymbol{g}}_j \boldsymbol{a_j}\right] \qquad (14)$$

$$\breve{\boldsymbol{g}}_j = \frac{\boldsymbol{e_j} + \gamma \boldsymbol{a_j^T B}}{1 + \alpha\gamma m_1 \boldsymbol{a_j^T a_j}} \qquad (15)$$

donde $\breve{\boldsymbol{g}}$ es un vector auxiliar y $\boldsymbol{Q_j}$ es una matriz que contiene los incrementos parciales de pesos y biases; γ, α y m_1 son parámetros de ajuste fijo.

   2.4. Actualizar los pesos sinápticos con

$$\boldsymbol{B_j} = \boldsymbol{B_j} + \alpha \boldsymbol{Q_j} \qquad (16)$$

3. Regresar al paso 2 si el resultado no es satisfactorio de acuerdo con algún criterio de error.

Por tratarse de un algoritmo que hereda las propiedades del algoritmo ARγ converge para valores positivos de γ, α y m_1 ($\gamma>0$, $\alpha>0$ y $m_1>0$). El procedimiento se implementó en la herramienta computacional Matlab® y se validó mediante un problema de regresión. El planteamiento y diseño del experimento se presentan a continuación.

4. Planteamiento y Diseño del Experimento

El objetivo es determinar el grado de generalización alcanzado con múltiples redes SLFN con diferente número de nodos ocultos cuando se entrenan para determinar una función, de tal forma que sea lo más cercana posible a la función propuesta por Saito y Nakano [9], que se define como:

$$y = 2 + x_1^{-1}x_2^3 + 4x_3x_4^{1/2}x_5^{-1/3}. \qquad (17)$$

Para estas simulaciones se generaron 500 patrones de entrenamiento, de los cuales 300 se usan para entrenamiento, 100 para generalización y 100 para prueba. Los valores de las variables independientes se han elegido aleatoriamente en el intervalo [0, +1]. Las salidas deseadas se obtienen aplicando directamente la ecuación 17 y los parámetros de entrenamiento son α=0,1 , γ=0,1 y m_1=0,1 para todas las experiencias. Además, se adiciona una señal de ruido blanco gaussiano del 10% (relación entre la desviación estándar del ruido y la desviación estándar de la señal) a las muestras de entrenamiento y validación. Las muestras de prueba se mantienen libres de ruido.

Para calcular el número de nodos ocultos se entrenan diferentes arquitecturas con las muestras de entrenamiento y se calcula la raíz cuadrada del error cuadrático medio (RMSE) con las muestras de validación para cada una de las redes. El número de nodos ocultos de las arquitecturas utilizadas están espaciados de acuerdo con [5, 10, 15,...,300]. Finalmente se escoge la arquitectura de red que obtuvo los mejores resultados de validación y se obtienen las gráficas de regresión haciendo uso de los datos de prueba.

La raíz cuadrada del error cuadrático medio se define como [8]:

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N} e_j^T e_j} \qquad (18)$$

5. Resultados y discusión

Los resultados obtenidos en función del número de nodos ocultos se presentan en la Figura 3. Un aspecto importante es que no se presenta ningún hecho experimental relacionado con sobre-aprendizaje. Nótese que el error de validación no cambia significativamente al aumentar el número de nodos ocultos. Con los métodos convencionales basados en gradiente generalmente ocurre que la arquitectura se adapta progresivamente al conjunto de entrenamiento, acomodándose al problema y mejorando la generalización, pero en determinado momento el modelo se empieza a ajustar demasiado a las particularidades de los patrones empleados en el entrenamiento, por lo que crece el error que comete ante patrones nuevos. Las arquitecturas con más nodos ocultos tiene más memoria asociativa, con lo que, pueden incluso, simplemente memorizar dichas particularidades, y alcanzar errores de entrenamiento más bajos sin captar la verdadera tendencia del problema, esto implica un error alto de validación. De acuerdo con los resultados obtenidos, son notorias las ventajas que el algoritmo propuesto tiene para sobrellevar este problema.

Los datos que miden más objetivamente la eficacia de la red son los que pertenecen al conjunto de prueba. En la Figura 4 se presenta el diagrama de muestras ante los datos de prueba en contraste con los resultados proporcionados por el modelo teórico para los mismos datos, en la Figura 5 se presenta el error del modelo estimado respecto a los datos de prueba obtenidos teóricamente y en la Figura 6 se presenta el correspondiente diagrama de regresión que indica mediante el coeficiente de ajuste (o coeficiente de correlación) una medida de la confiabilidad de la estimación proporcionada por el modelo encontrado, de tal manera que entre más cercano a 1 sea el coeficiente de ajuste, más confiable será la estimación realizada por la red; para un ajuste ideal los datos deberían quedar sobre la línea punteada, en tal caso coincidirían también con la recta de ajuste (línea continua).

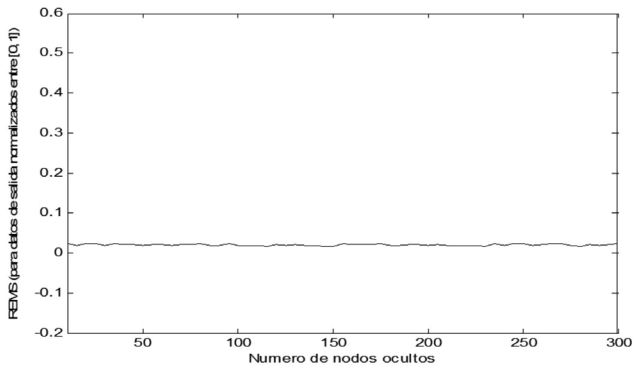

Figura 3. Relación entre el número de nodos ocultos y el RMSE con datos de validación (Función de Saito-Nakano).

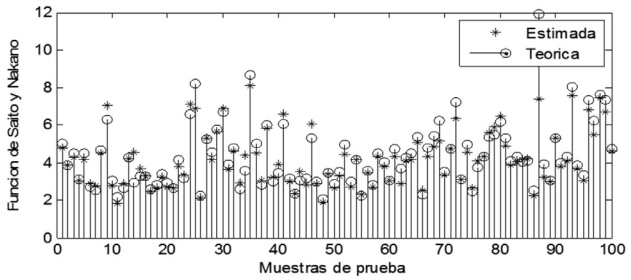

Figura 4. Relación entre el número de nodos ocultos y el RMSE (Función de Saito-Nakano).

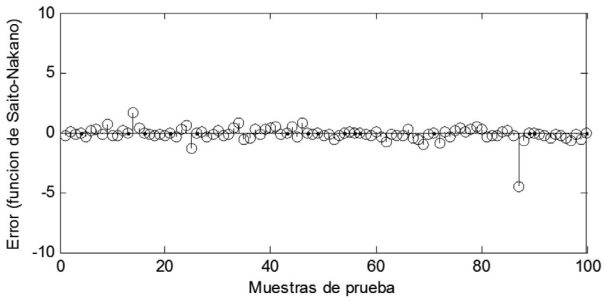

Figura 5. Error del modelo estimado usando datos de prueba.

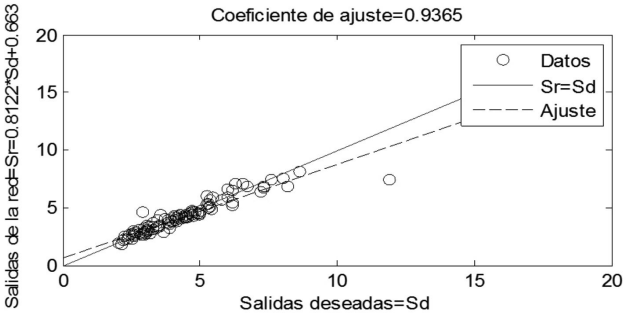

Figura 6. Diagrama de regresión usando datos de prueba.

Los resultados permiten afirmar que la implementación realizada es consistente experimentalmente, tiene capacidades bastante atractivas en lo que respecta a la aproximación de funciones y además evidencian bajas posibilidades de incurrir en problemas de sobre-aprendizaje, a diferencia de otras técnicas cuando no se controlan algunos aspectos.

6. Conclusiones

El algoritmo propuesto permite el entrenamiento de redes neuronales SLFN y resulta de combinar las propiedades de los algoritmos ELM y ARγ cuyas características principales son baja complejidad computacional y alta velocidad de convergencia respectivamente.

El algoritmo propuesto permite resolver problemas de estimación funcional (como el de Saito y Nakano) con un alto coeficiente de ajuste.

El algoritmo propuesto obtuvo un alto rendimiento en términos de generalización evitando el sobre-aprendizaje en todas las pruebas realizadas.

Bibliografía

[1] G. Huang, Q. Zhu y C. Siew. "Extreme learning machine: Theory and applications". Contenido disponible en Science Direct, Neurocomputing 70, Pag. 489–501, 2006.

[2] S. Balasundaram y D. Gupta. "1-Norm extreme learning machine for regression and multiclass classification using Newton method". Contenido disponible en Science Direct, Neurocomputing 128, Pag. 4-14, 2014.

[3] A. Lima, A. Cannon y W. Hsieh, "Nonlinear regression in environmental sciences using extreme learning machines: A comparative evaluation", Contenido disponible en Science Direct, Environmental Modelling & Software, 73, Pag. 175-188, 2015.

[4] X. Liu y A. Wan. "Universal consistency of extreme learning machine for RBFNs case". Contenido disponible en Science Direct, Neurocomputing 168, Pag. 1132–1137, 2015.

[5] V. Solart y P. Jojoa. "El Algoritmo Acelerador Regresivo versión γ (ARγ) y los efectos de cuantificación". Revista Universitaria en Telecomunicaciones Informática y Control. Vol 1. N° 2, 2012.

[6] P. Jojoa. "Um algoritmo acelerador de parámetros" Tesis de Doctorado. Escola Politecnica da Universidad de Sao Paulo. Brasil, 2003.

[7] P. Mohapatra, S. Chakravarty y P. Dash. "An improved cuckoo search based extreme learning machine for medical data classification". Contenido disponible en Science Direct, Swarm and Evolutionary Computation 24, Pag. 25–49, 2015.

[8] X. Wang y M. Han. "Improved extreme learning machine for multivariate time series online sequential prediction". Contenido disponible en Science Direct, Engineering Applications of Artificial Intelligence 40, Pag. 28–36, 2015.

[9] A. Estudillo. "Modelos de regresión basados en redes neuronales de unidades producto diseñadas y entrenadas mediante algoritmos de optimización híbrida: aplicaciones". Tesis de Doctorado. Universidad de Granada, 2005.