# ALGORITMO ACELERADOR REGRESIVO VERSIÓN GAMMA CON GRADIENTE LOCAL DE ERROR PARA ENTRENAMIENTO DE REDES NEURONALES PERCEPTRÓN MULTICAPA

## GAMMA VERSION REGRESSIVE ACCELERATOR ALGORITHM WITH LOCAL ERROR GRADIENT FOR MULTILAYER PRECEPTRON NEURAL NETWORK TRAINING

**AUTOR**
FAUSTO MIGUEL CASTRO CAICEDO
Magister en Electrónica y Telecomunicaciones
Universidad del Cauca
Estudiante de Doctorado en Ciencias de la Electrónica
Facultad de Ingeniería
Electrónica y Telecomunicaciones.
faustocastro@unicauca.edu.co
COLOMBIA

**AUTOR**
PABLO EMILIO JOJOA GOMEZ
Doctor en Ingeniería Eléctrica
Universidad del Cauca
Profesor
Facultad de Ingeniería
Electrónica y Telecomunicaciones.
pjojoa@unicauca.edu.co
COLOMBIA

***INSTITUCIÓN**
Universidad del Cauca
UNICAUCA
Universidad publica
Calle 5 No. 4-70
rectoria@unicauca.edu.co
COLOMBIA

**INFORMACIÓN DE LA INVESTIGACIÓN O DEL PROYECTO:** Este trabajo es uno de los productos desarrollados en el proyecto de maestría "Diseño de un Algoritmo de Aprendizaje para Redes Neuronales MLP Basado en las Propiedades del Algoritmo Acelerador Regresivo Versión γ". El proyecto se desarrolló en el Grupo de investigación Nuevas Tecnologías en Telecomunicaciones GNTT de la Universidad del Cauca.

**RECEPCIÓN:** Enero 17 de 2015 **ACEPTACIÓN:** Julio 22 de 2015

**TEMÁTICA:** Ingeniería Eléctrica, Electrónica, Telecomunicaciones y Telemática

**TIPO DE ARTÍCULO:** Artículo de investigación científica y tecnológica.

**Forma de citar:** Castro Caicedo, F. M. (2015). Algoritmo acelerador regresivo versión gamma con gradiente local de error para entrenamiento de redes neuronales perceptrón multicapa. En R, Llamosa Villalba (Ed.). Revista Gerencia Tecnológica Informática, 14(39), 65-74. ISSN 1657-8236.

**RESUMEN ANALÍTICO**

Se presenta un nuevo algoritmo para el entrenamiento de redes neuronales perceptrón multicapa llamado Acelerador Regresivo versión Gamma con Gradiente Local de Error. Este algoritmo se basa en los mismos principios que rigen la actualización de parámetros en el algoritmo Acelerador Regresivo versión Gamma. El algoritmo Acelerador Regresivo versión Gamma con Gradiente Local de Error se valida mediante diferentes problemas relacionados con aproximación de funciones y reconocimiento de patrones. Los resultados muestran buen comportamiento en cuanto a convergencia y generalización, mejorando la tasa de aprendizaje del algoritmo "backpropagation".

**PALABRAS CLAVES:** Redes Neuronales, Perceptrón Multicapa, Algoritmo ARγ, Gradiente Local de Error, Reconocimiento de Patrones, Aproximación de Funciones.

**ANALYTICAL SUMMARY**

A new algorithm is presented for Multi-Layer Perceptron Neural Networks training, which is called Gamma version regressive accelerator algorithm with local error gradient. This algorithm is based on the same principles as parameter actualization in Gamma version regressive accelerator algorithm. Gamma version regressive accelerator algorithm with local error gradient is validated through different problems related to pattern recognition and fitting function. Results show good convergence and generalization, and improving the learning rate of back propagation algorithm.

**KEYWORDS:** Artificial Neural Networks, Multi-Layer Perceptron, ARγ Algorithm, Local Gradient, Pattern Recognition, Fitting Function.

## INTRODUCCIÓN

Un sistema neuronal artificial o ANS (*Artificial Neural System*) es un esquema de procesamiento inspirado en la estructura de los sistemas nerviosos biológicos que intenta reproducir sus capacidades [1]. Estos sistemas han sido ampliamente utilizados en diversos campos científicos e ingenieriles tales como la minería de datos, diagnosis y clasificación, procesamiento de señales, diseño de sistemas de control, reconocimiento del habla, visión de objetos inmersos en el espacio natural y resolución de ecuaciones [2].

Una de las características principales de los ANS es su capacidad de aprender y mejorar su rendimiento a través del entrenamiento para lo cual debe modificar los parámetros de su estructura en función de las señales provenientes del entorno [3]. Un ANS tiene como unidad funcional a la neurona artificial, la cual al agruparse con otras formas capas, y a su vez varias capas constituyen una red [4]. Sin embargo, una red neuronal no es capaz de aprender por sí sola a realizar una tarea por lo cual debe adicionarse un algoritmo de entrenamiento [5].

Por su parte, el perceptrón multicapa o red MLP (*Multi-Layer Perceptron*), es el modelo más ampliamente utilizado en el contexto de las redes neuronales [6]. Se compone de varias capas de neuronas unidas sin ningún tipo de realimentación y permite abordar problemas complejos de clasificación y aproximación funcional de una manera eficaz y relativamente simple [7][8]. El algoritmo más ampliamente utilizado para entrenar esta arquitectura es el denominado "backpropagation" o BP (*Back Propagation*), basado en el método de gradiente descendiente [9]. Sin embargo, no se garantiza que usando esta técnica se tome el mejor camino hacia el mínimo y aunque varias propuestas intentan solucionar este problema, no existe un algoritmo superior en general, por lo que mejorar el aprendizaje de las redes MLP sigue siendo un desafío [10]. De aquí la importancia de formular nuevos algoritmos de aprendizaje que busquen mejorar los resultados de los algoritmos actuales.

Teniendo en cuenta lo anterior, se presenta en este artículo un nuevo algoritmo para el entrenamiento de redes neuronales perceptrón multicapa denominado Acelerador Regresivo versión Gamma con Gradiente Local de Error o ARγ-GLE.

Este documento se organiza de la siguiente manera: en la sección 1 se presenta los aspectos fundamentales de las redes neuronales MLP, el algoritmo BP y una de sus variantes más representativas (BP con momento); en la sección 2 se expone el funcionamiento básico del algoritmo ARγ-GLE; en la sección 3 y sección 4 se presenta un estudio experimental del algoritmo ARγ-GLE basado en simulación el cual gira en torno a dos problemas relacionados respectivamente con reconocimiento de patrones y aproximación funcional; en la sección 5 se realiza un análisis comparativo de

los resultados obtenidos con el algoritmo ARγ-GLE y el algoritmo BP ante dos reconocidos problemas de "benchmarking". Finalmente en la sección 6 se enuncian las conclusiones.

## 1. PERCEPTRÓN MULTICAPA (MLP)

El perceptrón multicapa es un modelo neuronal no lineal compuesto de varias capas de neuronas entre la entrada y la salida, dichas capas se unen sin ningún tipo de realimentación, de tal manera que, las entradas se conectan con la primera capa y las salidas de ésta con la siguiente capa y así sucesivamente hasta la última capa [6]. Esta arquitectura puede establecer regiones de decisión complejas, se utiliza para resolver problemas de clasificación y también como un aproximador universal de funciones [7].

La figura 1 muestra la arquitectura abreviada de un MLP con una capa oculta. Si se denominan $x_i[n]$ a las entradas de la red, $y_j[n]$ a las salidas de la capa oculta y $z_k[n]$ a las salidas de la capa final o de salida, y por otro lado, $w_{i,j}[n]$ a los pesos de la capa oculta y $b_j[n]$ a sus bias, $w'_{k,j}[n]$ a los pesos de la capa de salida y $b'k[n]$ a sus bias, entonces la operación de un MLP con una capa oculta se expresa matemáticamente como:

$$Z_k[n] = f_2(\sum_j w'_{k,j}[n]\, y_j[n] - b'_k[n], \quad (1)$$

donde,

$$y_j[n] = f_1(\sum_i w_{j,i}[n] - b_j[n]),$$

siendo $f_1$ y $f_2$ las funciones de activación para las neuronas de la capa oculta y la capa de salida respectivamente [5]. Por lo general se utilizan funciones de activación sigmoideas para las capas ocultas y funciones de activación lineales para la capa de salida, esta es la arquitectura más común de MLP. Sin embargo, existen numerosas variantes (por lo general en problemas de clasificación) que utilizan funciones de activación sigmoideas en la capa de salida [5].

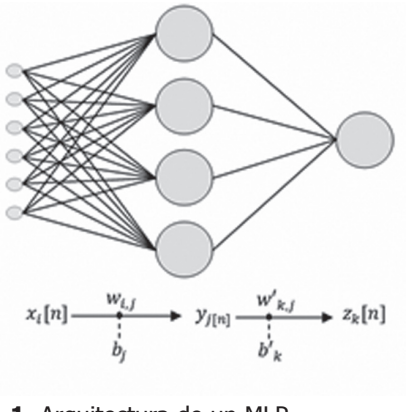



**FIGURA 1.** Arquitectura de un MLP.

A menudo, para hacer referencia a una arquitectura MLP se utiliza una secuencia de números cuyos valores indican respectivamente el número de entradas, número de neuronas en cada capa oculta y número de neuronas de salida. Así entonces, la arquitectura mostrada en la figura 1 es 6-4-1.

### 1.1 ALGORITMO BP

Es un algoritmo de aprendizaje utilizado en el entrenamiento de redes MLP que se deduce de aplicar el método de gradiente descendiente, por lo que calcula el incremento de parámetros siguiendo el sentido contrario al gradiente de la energía del error $E[n]$ [7][2]. Así entonces, la actualización de los pesos queda definida como sigue a continuación:

$$w'_{k,j}[n+1] = w'_{k,j}[n] + \hat{\alpha}\, \frac{\partial E[n]}{\partial w'_{k,j}[n]} \quad (2)$$

$$w_{j,i}[n+1] = w_{j,i}[n] + \hat{\alpha}\, \frac{\partial E[n]}{\partial w_{j,i}[n]}, \quad (3)$$

Donde $\hat{\alpha}$ representa la tasa de aprendizaje [11]. Por lo general durante el entrenamiento $\hat{\alpha}$ se mantiene constante, sin embargo, si $\hat{\alpha}$ es muy alta el algoritmo puede oscilar y entrar en inestabilidad, si por el contrario $\hat{\alpha}$ es muy pequeña, el entrenamiento puede tornarse muy lento [12].

### 1.2 ALGORITMO BP CON MOMENTO

Comúnmente al algoritmo BP se le adiciona un término proporcional al incremento de la iteración anterior (*momentum*) con el propósito de introducir una cierta inercia en el entrenamiento [11]. Esta variante se conoce como algoritmo BP con momento y las expresiones para la actualización se definen a continuación:

$$w'_{k,j}[n+1] = w'_{k,j}[n] + \hat{\alpha}\, \frac{\partial E[n]}{\partial w'_{k,j}[n]} + \hat{\mu}\Delta w'_{k,j}[n-1] \quad (4)$$

$$w_{j,i}[n+1] = w_{j,i}[n] + \hat{\alpha}\, \frac{\partial \mathrm{E}[n]}{\partial w_{j,i}[n]} + \hat{\mu}\Delta w'_{k,j}[n-1], \quad (5)$$

Siendo $\hat{\mu}$ una constante que ajusta el término momento, la cual puede tomar valores entre 0 y 1 [5].

## 2. ALGORITMO ARΓ-GLE

A diferencia del algoritmo BP el cual se basa en el método de gradiente descendiente, el algoritmo ARγ-GLE se basa en los principios que rigen la actualización de parámetros en el algoritmo ARγ (Acelerador Regresivo versiónGamma), que fue desarrollado en el contexto del filtrado adaptativo a partir de la discretización de un algoritmo en tiempo continuo que ajusta la segunda derivada de parámetros [13]. El procedimiento general se resume a continuación:

1. Iniciar aleatoriamente los pesos sinápticos y bias de la red.

2. Para cada patrón de aprendizaje:

**2.1 OBTENER LA RESPUESTA GENERAL DE LA RED $z'_k[n]$ COMO SIGUE**

$$z'_k[n]=f_2(a'_k[n]) \tag{6}$$

con $a'_k[n]=\sum_{j=0}^{U} y_i[n]w'_{k,j}\ [n-1]$

siendo

$$y_j[n]=f_1(a_j[n]) \tag{7}$$

con $a_j[n]=\sum_{i=0}^{R} x_i[n]w_{j,i}[n-1],$

Donde $R$ es el número de entradas, $U$ el número de neuronas ocultas y $a'_k[n]$ y $a_j[n]$ los potenciales postsinápticos [3]. Se consideran los bias como un peso sináptico adicional al cual se le asocia permanentemente una entrada negativa de valor unitario, por tanto $b'_k[n]=w'_{k,0}[n]$, $b_j[n]=w_{j,0}[n]$, y $x_0=y_0=-1$.

**2.2. CALCULAR LOS GRADIENTES LOCALES DE ERROR $\delta'_k[n]$ Y $\delta_j[n]$ PARA CADA NEURONA DE LA RED CON**

$$\delta'_k[n]=(z_k[n]-t_k[n])\ \frac{\partial f_2(a'_k[n])}{\partial a'_k[n]}, \tag{8}$$

$$\delta_j[n]=\frac{\partial f_1(a_j[n])}{\partial a_j[n]}\sum_k \delta'_k[n]w'_{k,j}[n-1], \tag{9}$$

Siendo $t_k[n]$ las salidas deseadas.

3. Calcular el incremento parcial de los pesos y bias con

$$g'[n]=\frac{\delta'_k[n]+\gamma\Sigma y_j[n]w'_{k,j}[n-1]}{1+\alpha\gamma m_1\Sigma y_j^2[n]}, \tag{10}$$

$$q'_{k,j}[n]=\frac{\gamma}{\alpha+\gamma}\left(q'_{k,j}[n-1]-\alpha g'[n]m_1 y_j[n]\right), \tag{11}$$

$$g[n]=\frac{\delta_j[n]+\gamma\Sigma x_i[n]w_{j,i}[n-1]}{1+\alpha\gamma m_1\Sigma x_i^2[n]}, \tag{12}$$

$$q_{j,i}[n]=\frac{\gamma}{\alpha+\gamma}\ (q_{j,i}[n-1]-\alpha g[n]m_1 x_i[n]), \tag{13}$$

Donde: $g[n]$ y $g'[n]$ son escalares auxiliares; $q_{j,i}[n]$ y $q'_{k,j}[n]$ incrementos parciales de los pesos y bias; y finalmente $\alpha$, $\gamma$ y $m_1$ son parámetros de entrenamiento fijos. Por tratarse de un algoritmo que hereda las propiedades del algoritmo ARγ, los parámetros $\alpha$, $\gamma$ y $m_1$ deben ser positivos mayores que cero ($\alpha>0, \gamma>0$ y $m_1>0$).

4. Actualizar los pesos y bias con

$$w'_{k,j}[n]=w'_{k,j}[n-1]+\alpha q'_{k,j}[n], \tag{14}$$

$$w'_{j,i}[n]=w'_{j,i}[n-1]+\alpha q'_{j,i}[n], \tag{15}$$

En [14] se presenta un estudio relacionado con el efecto que los parámetros $\alpha$, $\gamma$ y $m_1$ tienen en el entrenamiento, a partir del cual se establece experimentalmente un orden lógico para la escogencia de dichos parámetros, siendo $\alpha$ (seguido respectivamente por $m_1$ y $\gamma$), el que más efectos tiene en términos de convergencia y desajuste para un mínimo local dado.

A continuación se presenta una serie de resultados con el algoritmo ARγ-GLE basados en simulación. Este estudio gira en torno a diferentes problemas relacionados con reconocimiento de patrones y aproximación funcional.

## 3. DISCRIMINACIÓN DE CLASES GAUSSIANAS.

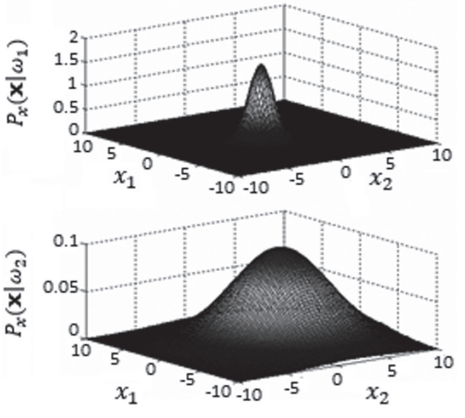
El experimento consiste en discriminar patrones x pertenecientes a dos clases bidimensionales con distribución gaussiana. Las funciones de probabilidad $P_x(x|\omega_1)$ y $P_x(x|\omega_2)$ para las clases 1 y 2 respectivamente, están dadas por:

$P_x(X|\omega_1)=\frac{1}{2\pi\sigma_1}\ e^{\left(\frac{1}{2\sigma_1}\|x-\mu_1\|\right)}$, con media $\mu_1=[0,0]^T$ y varianza $\sigma_1=1$.
$P_x(x|\omega_2)=\frac{1}{2\pi\sigma_2}e^{\left(\frac{1}{2\sigma_2}\|x-\mu_2\|\right)}$, con media $\mu_2=[2,0]^T$ y varianza $\sigma_2=4$.

La figura 2 muestra las funciones de probabilidad $P_x(x|\omega_1)$ y $P_x(x|\omega_2)$ en tres dimensiones. En cada distribución puede identificarse la zona donde la probabilidad de ocurrencia toma valores considerablemente mayores, cabe destacar que dichas zonas están traslapadas.

Las clases 1 y 2 se asumen equiprobables [15].

**FIGURA 2.** Funciones de probabilidad $P_x(x|\omega_1)$ y $P_x(x|\omega_2)$.



### 3.1 PROBABILIDAD TEÓRICA DE CORRECTA CLASIFICACIÓN

Luego de aplicar y desarrollar matemáticamente el criterio de Bayes para mínimo error de clasificación se obtiene una frontera de decisión cuya ecuación característica es la de una circunferencia con centro $x_c$ y radio $r$ [15]

$$||x - x_C||^2 = r^2. \quad (16)$$

Concretamente para las condiciones experimentales planteadas al inicio de la sección 3, se puede demostrar que $xc = [2/3\ 0]^T$ y $r \approx 2{,}34$.

Por otra parte, la probabilidad de error $P(e_r)$ definida en el sentido de Bayes está dada por

$$P(e_r) = P(\omega_1)P(e_r|\omega_1) + P(\omega_2)P(e_r|\omega_2), \quad (17)$$

Donde

$$P(e_r|\omega_1) = \int_{C(\Omega_1)} P_x(x|\omega_1)dx, \quad (18)$$

$$P(e_r|\omega_1) = \int_{C(\Omega_1)} P_x(x|\omega_1)dx, \quad (18)$$

Siendo $\Omega_1$ la región interna del círculo de decisión y $C(\Omega_1)$ su complemento [15].

Evaluando numéricamente las integrales (18) y (19), y remplazando valores en la ecuación (17) (teniendo en cuenta que $P(\omega_1) = P(\omega_1) = 1/2$ por tratarse de clases equiprobables), se obtiene una probabilidad de error de clasificación $P(e) = 0.1849$ y equivalentemente una probabilidad de correcta clasificación $P_c$,

$$P_c = 1 - P(e_r) \approx 0{,}8151.$$

Por lo que en teoría un clasificador bayesiano puede distinguir entre patrones de la clase 1 y 2 con un rendimiento de del 81,51 %.

### 3.2 RESULTADOS DE CLASIFICACIÓN

A partir de una serie de pruebas previas se estableció que una arquitectura de red MLP con dos neuronas ocultas da buenos resultados. Así mismo, se estableció como parámetros de entrenamiento adecuados, los valores de $\alpha = 0.1$, $\gamma = 1.6133$ y $m_1 = 1$. El conjunto de entrenamiento utilizado consta de 1000 patrones aleatoriamente ordenados con igual probabilidad de ocurrencia para las clases 1 y 2.

Posteriormente, teniendo en cuenta que la función de aprendizaje a obtenerse es puramente estocástica dado que se utilizan conjuntos de entrenamiento finitos, las mediciones de rendimiento (en términos de probabilidad de correcta clasificación) se obtuvieron promediando los resultados de 20 redes independientemente entrenadas. Se utilizaron 320 épocas de entrenamiento en cada sesión.

La probabilidad de correcta clasificación se determina en modo de ejecución, aplicando como entradas una cantidad de muestras que no hayan sido usadas en el entrenamiento, posteriormente se determina el porcentaje de aciertos, el cual puede considerarse como un estimativo de la probabilidad de correcta clasificación. En [15] se determina experimentalmente que cuando , el estimativo hallado tiene una certidumbre superior al . Como tal se emplearon 32000 muestras para estimar la probabilidad de correcta clasificación.

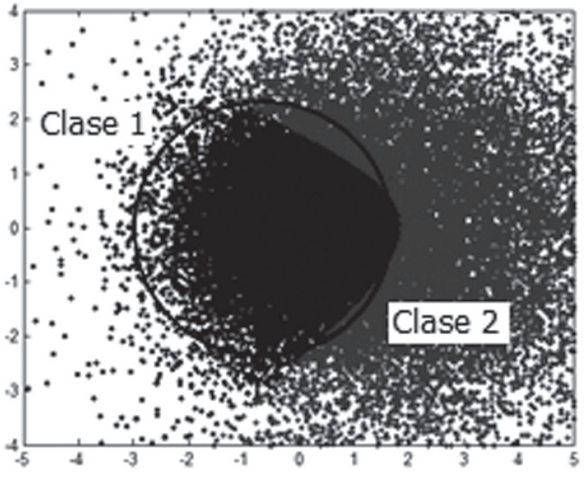
La Tabla 1 muestra estadísticas relacionadas con las 20 sesiones de entrenamiento. La medida de rendimiento es la probabilidad de correcta clasificación, para la cual se indica el promedio y la desviación estándar obtenidas.

**TABLA 1.** Estadísticas de Rendimiento para 20 Sesiones de Entrenamiento

| Rendimiento | Media | Desviación estándar |
|---|---|---|
| Probabilidad de correcta clasificación | 79,8530% | 1,1374% |

La red que obtuvo el mejor resultado alcanzó un rendimiento en clasificación del 80.82 % y establece una frontera de decisión con forma convexa. En la figura 3 se presenta el diagrama de clasificación para dicha red, en contraste con la frontera circular de decisión bayesiana. Cabe recordar además que el límite teórico para la probabilidad de correcta clasificación definido con base en el criterio de Bayes es 81.51 %.

**FIGURA 3**. Diagrama de clasificación para la mejor frontera de decisión obtenida.



## 4. APROXIMACIÓN DE LA FUNCIÓN DE SAITO NAKANO

El objetivo es determinar mediante validación cruzada el grado de generalización alcanzado con múltiples arquitecturas de red cuando se entrenan para determinar una función, de tal forma que sea lo más cercana posible a la función propuesta por Saito y Nakano, la cual se define como [16]:

$$y = 2 + x_1^{-1} x_2^3 + 4 x_3 x_4^{1/2} x_5^{-1/3} \quad (20)$$

### 4.1 CONDICIONES EXPERIMENTALES

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 (20)

Los parámetros de entrenamiento son $\alpha = 0{,}1$, $\gamma = 0{,}1$ y $m1 = 0{,}1$ para todas las experiencias.

Se utilizan funciones de activación sigmoidal en las capas ocultas y funciones de activación lineal (identidad) en las capas de salida.

### 4.2 DETERMINACIÓN DE ARQUITECTURAS DE RED

En este experimento se prueban varias arquitecturas MLP de una y dos capas ocultas aplicando la técnica de validación cruzada. El error de validación se determina cada que transcurre una época de entrenamiento, guardando cada vez la configuración de pesos más adecuada, es decir aquella que vaya obteniendo el menor error de validación [17]. El número de épocas utilizadas para el entrenamiento en todas las experiencias es 1200. Los resultados obtenidos se muestran en la Tabla 2.

**TABLA 2.** Resultados de Validación para Arquitecturas de Una Capa Oculta

| Arquitectura | Error de validación (MSE) |
|---|---|
| 5-2-1 | 15,2319 |
| 5-3-1 | 16,626 |
| 5-5-1 | 18,3466 |
| 5-8-1 | 26,2607 |
| 5-12-1 | 17,972 |
| 5-18-1 | 24,7685 |
| 5-3-2-1 | 41,7988 |
| 5-12-6-1 | 57,2675 |
| 5-18-14-1 | 93,9097 |
| 5-32-18-1 | 114,2566 |
| 5-65-35-1 | 140,9689 |
| 5-150-75-1 | 163,3494 |

### 4.3 RESULTADOS

Se realizaron mediciones promediando los resultados de 20 sesiones de entrenamiento para la mejor y peor arquitectura de red, nuevamente las condiciones y los parámetros de entrenamiento fueron los mismos estipulados en la sección 4.1. Los resultados se presentan en la Tabla 3.

**TABLA 3.** Estadísticas de Rendimiento para 20 Sesiones de Entrenamiento con las Arquitecturas Seleccionadas

| Arquit. | MSE Entrenamiento | | | |
|---|---|---|---|---|
| | Media | Desviación | Mejor | Peor |
| 5-2-1 | 6,1718 | 0,2785 | 5,9497 | 7,2174 |
| 5-150-75-1 | 5,43 E-4 | 3,36 E-4 | 4,40 E-5 | 0,0012 |
| | MSE Validación | | | |
| | Media | Desviación | Mejor | Peor |
| 5-2-1 | 15,2556 | 0,0751 | 15,1708 | 15,5529 |
| 5-150-75-1 | 162,6697 | 12,18 | 141,569 | 191,1421 |

En la Tabla 3 se presenta un hecho experimental relacionado con sobreaprendizaje, ambas arquitecturas se entrenaron con el mismo número de épocas, pero la arquitectura 5-150-75-1 logró alcanzar en el entrenamiento una cota de error mucho menor en relación a la arquitectura 5-2-1. Sin embargo, la eficacia general del sistema o generalización para la arquitectura 5-150-75-1 es mucho menor dado que su error de validación es más alto.

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, y en la figura 5 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).

Los resultados indican que el algoritmo ARγ-GLE tiene capacidades bastante atractivas en lo que respecta a aproximación de funciones. Sin embargo, también evidencian sus posibilidades de incurrir en problemas de sobreaprendizaje, al igual que otras técnicas, si no se controlan algunos aspectos.

**FIGURA 4.** Diagrama de Muestras para el Mejor Modelo Estimado (Arquitectura 5-2-1) Usando Datos de Prueba.

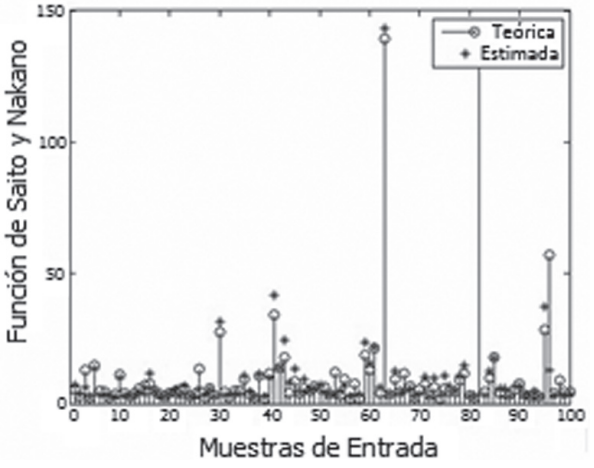
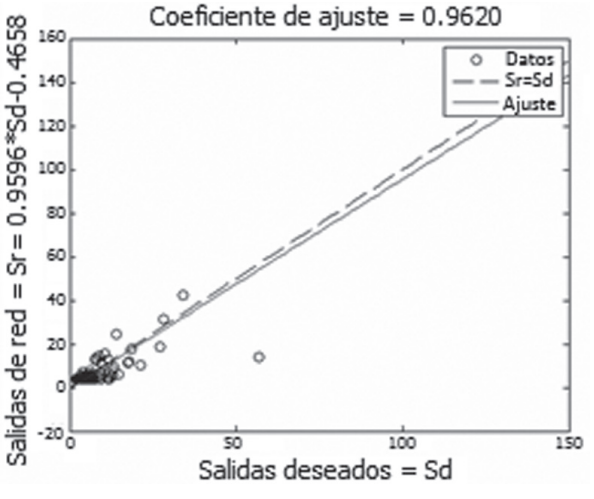



**FIGURA 5.** Diagrama de Regresión para el Mejor Modelo Estimado (Arquitectura 5-2-1) Usando Datos de Prueba.




## 5. ANÁLISIS COMPARATIVO DEL ALGORITMO ARΓ-GLE CON EL ALGORITMO BP.

### 5.1 DESCRIPCIÓN DE LOS DATOS UTILIZADOS

El análisis comparativo que se presenta a continuación se basa en dos tipos de datos usados comúnmente en problemas de "benchmarking". El primer conjunto de datos (función coseno) se puede obtener directamente, el segundo conjunto (precio de vivienda) está disponible en *UCI Machine Learning Repository* [18]. A continuación se describen los datos utilizados.

#### 5.1.1 FUNCIÓN COSENO

Inicialmente se compara el rendimiento de ambos algoritmos mediante un problema de regresión que consiste en aproximar la función coseno $y = \cos x$. Con el propósito de mantener la señal de salida en el rango [0, +1], la función es cambiada por $y = (\cos 2x + 1)/2$. Los patrones de entrenamiento se obtienen evaluando la función $y = (\cos 2x + 1)/2$ en 315 puntos escogidos aleatoriamente en el intervalo $[0, +\pi]$.

#### 5.1.2 PRECIO DE VIVIENDA (*HOUSE PRICING DATASET*)

Se cuenta con 506 patrones de entrenamiento, cuyas entradas son 13 características asociadas a un vecindario, en función a las cuales se puede estimar el valor promedio de una vivienda. Los atributos de vecindario se relacionan respectivamente con:

Tasa de crimen per cápita
Proporción de área residencial.
Proporción de negocios no minoristas.
Presencia de ríos en la zona.
Concentración de óxidos nitrosos.
Promedio de habitaciones por vivienda.
Proporción de casas ocupadas por sus propietarios.
Distancia a centros de empleo.
Índice de accesibilidad.
Valor de los impuestos.
Tasa maestros/alumnos en la zona.
Proporción de grupos raciales.
Porcentaje de la población de estrato bajo.

### 5.2 PREPROCESAMIENTO Y CARACTERÍSTICAS DE ENTRENAMIENTO

La actualización de parámetros en el algoritmo BP se lleva a cabo en modo secuencial. El procesamiento se realiza bajo las siguientes condiciones:

*Escalado de datos*: primeramente se realiza un estandarizado que consiste en transformar las variables de entrada y salida de tal modo que presenten media cero y varianza unitaria. Posteriormente se normalizan los datos de tal manera que tomen valores en el rango [-1, +1].

*Parámetros de red*: para este análisis se utilizan redes de una y dos capas ocultas, y las pruebas involucran diferente número de neuronas en estas capas. Se hace uso de funciones de activación sigmoidales $Tanh(.)$ en las capas ocultas y funciones de activación lineales $y(.)$ en las capas de salida. Las expresiones matemáticas de las funciones de activación se presentan a continuación:

$$Tanh\,(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \tag{21}$$

$$y\,(a) = a, \tag{22}$$

Donde $Tanh(a)$ es la tangente hiperbólica y $y(a)$ es la función identidad, siendo $a$ el potencial postsináptico de una neurona cualquiera.

*Parámetros de entrenamiento*: se utilizan los mismos parámetros de entrenamiento en todas las neuronas de la red.

*Rendimiento de la red*: los resultados de entrenamiento, validación y prueba se miden en términos del error cuadrático medio o MSE (*Mean Squared Error*). La expresión matemática para calcular el MSE se define como [15]

$$\mathrm{MSE} = \frac{1}{2P}\sum_{u=1}^{P} ||t_u - F(x_u)||, \qquad (23)$$

Siendo $t_u$ el vector de salidas deseadas para el vector de entrada $x_u$, $P$ el número de muestras en el conjunto de datos utilizado y $F(.)$ la función de red propia del modelo utilizado (ecuación (1) para el caso de una capa oculta), de tal forma que $F(xu)$ es la salida de la red. Teniendo en cuenta que el entrenamiento se realiza con salidas deseadas a las cuales se les aplica un preprocesamiento previo, antes de calcular el MSE se debe aplicar a todas las salidas un procesamiento que revierta el preprocesamiento inicial.

*Inicialización de pesos*: una heurística (de la cual se hace uso en las siguientes simulaciones) que suele dar buenos resultados consiste en elegir los pesos sinápticos iniciales aleatoriamente en el intervalo [-2.4/R, +2.4/R], siendo R el número de entradas de la red [19].

### 5.3 RESULTADOS Y DISCUSIÓN

Los resultados que se presentan a continuación obedecen a promedios calculados a partir de 20 sesiones de entrenamiento para cada caso. Para la elección de los parámetros de entrenamiento se tuvo en cuenta el hecho de que $\alpha$ en el algoritmo ARγ-GLE y $\hat{\alpha}$ en el algoritmo BP son parámetros análogos similares relacionados con la tasa de aprendizaje [13]. Para cada $\alpha$ y $\hat{\alpha}$ dados se probaron múltiples configuraciones de $\gamma$ y $m_1$ para el algoritmo ARγ-GLE y múltiples momentos para el algoritmo BP.

Realizadas las pruebas planteadas se encuentra que el comportamiento del algoritmo ARγ-GLE en relación con el algoritmo BP para diferentes ritmos de aprendizaje es más adecuado, puesto que cuando $\hat{\alpha}$ toma valores altos el error de entrenamiento con el algoritmo BP presentó fluctuaciones indeseables e incluso inestabilidad, este hecho no se presentó en ninguna de las pruebas realizadas con el algoritmo ARγ-GLE [14].

Se probaron distintas configuraciones de parámetros de entrenamiento tomando para cada algoritmo la que alcanzó el mejor rendimiento en términos de error de validación y se obtuvieron los resultados que se presentan en la tabla 4 y tabla 6. Durante estas simulaciones se midió para diferentes épocas, el error de entrenamiento, validación y prueba [20]. Posteriormente, haciendo uso de las configuraciones de parámetros obtenidas, se entrenaron diferentes arquitecturas guardando en cada prueba el mínimo error de entrenamiento y validación. Los resultados se muestran en la tabla 5 y tabla 7.

**TABLA 4.** Resultados Función Coseno: Validación Cruzada en Arquitectura 1-2-1 con las Mejores Configuraciones de Parámetros.

| Épocas | Datos | ARγ - GLE | BP |
|---|---|---|---|
| | | $\alpha = 0{,}1$ $\gamma = 2.04$ $m_1 = 1$ | $\hat{\alpha} = 0{,}1$ $\hat{\mu} = 0{,}5$ |
| 300 | Entrenamiento | 2.21 E-05 | 2,77E-05 |
| | Validación | 1,65E-05 | 2,49E-05 |
| | Prueba | 1,85E-05 | 2,62E-05 |
| 600 | Entrenamiento | 1,78E-05 | 2,77E-05 |
| | Validación | 1,31E-05 | 2,47E-05 |
| | Prueba | 1,51E-05 | 2,62E-05 |
| 900 | Entrenamiento | 1,64E-05 | 2,77E-05 |
| | Validación | 1,20E-05 | 2,45E-05 |
| | Prueba | 1,39E-05 | 2,59E-05 |
| 1200 | Entrenamiento | 1,63E-05 | 2,76E-05 |
| | Validación | 1,20E-05 | 2,49E-05 |
| | Prueba | 1,39E-05 | 2,64E-05 |

**TABLA 5.** Resultados Función Coseno: Entrenamiento y Validación Usando Diferentes Arquitecturas.

| Arquitectura | ARγ-GLE | | BP | |
|---|---|---|---|---|
| | Entrena. | Val. | Entrena. | Val. |
| 1 - 2- 1 | 2,22E-05 | 1,84E-05 | 2,99E-05 | 2,51E-05 |
| 1- 5 -1 | 1,41E-05 | 1,12E-05 | 2,52E-05 | 1,35E-05 |
| 1- 12-1 | 9,20E-06 | 6,61E-06 | 3,37E-05 | 9,36E-06 |
| 1- 18 -1 | 7,56E-06 | 5,99E-06 | 6,39E-01 | 1,09E-01 |
| 1- 28- 1 | 5,42E-06 | 3,56E-06 | NaN | NaN |
| 1- 42 -1 | 4,25E-06 | 4,05E-06 | NaN | NaN |
| 1-3-2-1 | 4,90E-03 | 3,53E-03 | 3,66E-03 | 2,57E-03 |
| 1-8-4-1 | 5,13E-06 | 4,00E-06 | 2,55E-04 | 1,06E-04 |
| 1-12-6-1 | 5,19E-06 | 4,68E-06 | 2,61E-03 | 6,86E-04 |
| 1-18-10-1 | 1,34E-05 | 4,41E-06 | 7,92E-02 | 1,57E-02 |

**TABLA 6.** Resultados Precio de Vivienda: Validación Cruzada en Arquitectura 13-15-1 con las Mejores Configuraciones de Parámetros.

| Épocas | Datos | ARY-GLE | BP |
|---|---|---|---|
| | | $\alpha = 0{,}01$ $\gamma = 2.04$ $m_1 = 1$ | $\hat{\alpha} = 0{,}01$ $\hat{\mu} = 0{,}5$ |
| 50 | Entrenamiento | 3,77 | 2,43 |
| | Validación | 23,01 | 23,35 |
| | Prueba | 39,8 | 37,48 |
| 100 | Entrenamiento | 3,24 | 2,15 |
| | Validación | 17,05 | 25,58 |
| | Prueba | 36,4 | 40,09 |
| 200 | Entrenamiento | 2,89 | 1,77 |
| | Validación | 15,1 | 30,05 |
| | Prueba | 37,05 | 41,93 |
| 400 | Entrenamiento | 2,23 | 1,38 |
| | Validación | 22,87 | 36,71 |
| | Prueba | 43,01 | 44,16 |

**TABLA 7.** Resultados Precio de Vivienda: Entrenamiento y Validación Usando Diferentes Arquitecturas.

| Arquitectura | ARγ- GLE | | BP | |
|---|---|---|---|---|
| | Entrena. | Val. | Entrena. | Val. |
| 9-2-2 | 13,583 | 12,503 | 3,082 | 22,185 |
| 9-5-2 | 13,039 | 11,01 | 1,924 | 20,052 |
| 9-12-2 | 12,816 | 10,057 | 1,288 | 19,316 |
| 9-18-2 | 12,682 | 9,601 | 1,214 | 15,647 |
| 9-28-2 | 12,655 | 10,005 | 1,145 | 16,644 |
| 9-42-2 | 12,645 | 9,766 | 1,136 | 16,288 |
| 9-3-2-2 | 13,142 | 12,405 | 2,313 | 13,405 |
| 9-8-4-2 | 12,641 | 12,405 | 1,107 | 14,101 |
| 9-12-6-2 | 12,472 | 11,285 | 0,874 | 11,781 |
| 9-18-10-2 | 12,431 | 10,857 | 0,782 | 12,667 |

El algoritmo ARγ-GLE obtuvo en promedio los mejores resultados en términos de error de validación para las diferentes arquitecturas utilizadas (ver tablas 5 y 7). En el caso de la función coseno, las ventajas del algoritmo ARγ-GLE son mucho más marcadas para redes con más neuronas ocultas.

En el caso de los datos precio de vivienda, el algoritmo ARγ-GLE obtuvo mejores resultados que el algoritmo BP, siendo más significativas las ventajas en aquellas arquitecturas con menos neuronas ocultas.

Un aspecto positivo del algoritmo ARγ-GLE es que los resultados de validación obtenidos con las diferentes arquitecturas son más constantes que los del algoritmo BP [3]. Comúnmente, en el algoritmo BP la escogencia de la tasa de aprendizaje resulta imprevisible, por lo que tasas de aprendizaje que dan buenos resultados en una arquitectura pueden ocasionar comportamientos indeseables en otras (ver tabla 5), por lo general (aunque no necesariamente) para arquitecturas muy grandes suelen ser necesarias tasas de aprendizaje muy bajas, lo cual se traduce en una disminución de la velocidad de convergencia [6]. En las pruebas realizadas son notorias las ventajas que tiene el algoritmo ARγ-GLE para sobrellevar este problema.

## 6. CONCLUSIONES

El algoritmo ARγ-GLE permite el entrenamiento de redes neuronales MLP haciendo uso de los principios que rigen la actualización de parámetros en el algoritmo ARγ.

La combinación Arquitectura MLP + Algoritmo ARγ-GLE resuelve problemas de discriminación de patrones alcanzando probabilidades de clasificación muy cercanas a las teóricas en el sentido de Bayes y estima funciones (como la de Saito y Nakano) con un alto coeficiente de ajuste.

El ARγ-GLE presenta un mejor comportamiento que el algoritmo BP en lo relacionado con la tasa de aprendizaje evitando la inestabilidad en todas las pruebas realizadas.

## 7. REFERENCIAS BIBLIOGRÁFICAS

[1] Szymczyk, P. (2015) Z-transform artificial neural networks. *Neurocomputing*, 168, 1207-1210.

[2] Guo, D., Zhang, Y., Xiao, Z., Mao, M., & Liu, J. (2015) Common nature of learning between BP-type and Hopfield-type neural networks. *Neurocomputing*, 167, 578-586.

[3] Chetehounaa, K., El Tabach, E., & Bouazaoui, L. (2015) Predicting the flame characteristics and rate of spread in fires propagating in a bed of Pinus pinaster using Artificial Neural Networks. *Process Safety and Environmental Protection*, 98, 50-56.

[4] Ata, R. (2015) Artificial neural networks applications in wind energy systems: a review. *Renewable and Sustainable Energy Reviews*, 49, 534-562.

[5] Brio, B., & Sanz A. (2007). *Redes neuronales artificiales y sistemas borrosos, 3° Edición*. México: Alfaomega Ra-Ma.

[6] Mirjalili, S., Mirjalili, S., & Lewis, A. (2014) Let a biogeography-based optimizer train your Multi-Layer Perceptron. *Information Sciences*, 269, 188-209.

[7] Malalur, S., Manry, M., & Jesudhas, P. (2015) Multiple optimal learning factors for the multi-layer

perceptron. *Neurocomputing*, 149, 1490-1501.

[8] Albuquerque, V., Alexandria, A., Cortez, P., & Tavares, J. (2009). Evaluation of multilayer perceptron and self-organizing map neural network topologies applied on microstructure segmentation from metallographic images. *NDT & E International*, *42, 7*, 644–651.

[9] Jayalakshmi, T., & Santhakumaran, A. (2011). Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering, 3, 1*, 1793–8201.

[10] Zhang, J., Ji, N., Liu, J., Pan, J., & Meng, D. (2015) Enhancing performance of the backpropagation algorithm via sparse response regularization. *Neurocomputing*, 153, 20-40.

[11] Yasin, U., & Kemal, B. (2012). A second look at the performance of neural networks for keystroke dynamics using a publicly available dataset. *Computers & Security*, 3, 717–726.

[12] Mukherjee, I., & Routroy, S. (2012). Comparing the performance of neural networks developed by using levenberg-marquardt and quasi-newton with the gradient descent algorithm for modeling a multiple response grinding process. *Expert Systems with Applications, 39, 3*, 2397–2407.

[13] Jojoa, P. (2003). *Um algoritmo acelerador de parâmetros*. Tesis de Doctorado no publicada. Escola Polit'ecnica da Universidad de S'ao Paulo, S'ao Paulo, Brasil.

[14] Castro, F. (2014). *Diseño de un algoritmo de aprendizaje para redes neuronales MLP basado en las propiedades del algoritmo acelerador regresivo versión* $\gamma$. Tesis de Maestría. Universidad del Cauca, Popayán, Colombia.

[15] Haykin, S. (1998) *Neural networks: a comprehensive foundation, 2° Edición*. Ontario: McMaster University.

[16] Estudillo, A. C. (2005) 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.

[17] Demuth, H., Beale, M., & Hagan, M. (2010). *Neural network toolbox 7, user's guide*. Natick, MA: The MathWorks.

[18] Asuncion, A., & Newman, D. (2007). *UCI Machine Learning Repository*. Recuperado (2014, Enero14) de http://mlearn.ics.uci.edu/MLRepository.html

[19] Principe, J., Euliano, N., & Lefebvre, W. (2000) *Neural adaptive systems*: Fundamentals through simulation. New York: John Wiley & Sons.

[20] Zhen-Guo, C., Tzu-An, C., & Zhen-Hua, C. (2011) Feed-Forward neural networks training: A comparison between genetic algorithm and back-propagation learning algorithm. *Information and Control ICIC International, 7, 10*, 5839-5850.