ABYA YALA UNIVERSIDAD POLITÉCNICA SALESIANA

Artículo científico / Scientific paper

METEOROLOGÍA COMPUTACIONAL

LA GRANJA: REVISTA DE CIENCIAS DE LA VIDA

DOI: 10.17163/lgr.n21.2015.04

ANÁLISIS COMPARATIVO DE AJUSTE EN ENTRENAMIENTO DE REDES NEURONALES ARTIFICIALES A PARTIR DE LAS LIBRERÍAS OPEN NN Y ALGLIB

COMPARATIVE ANALYSIS OF ADJUSTMENT IN ARTIFICIAL NEURAL NETWORKS TRAINING USING OPEN NN AND ALGLIB LIBRARIES

Erith Muñoz[1,2] y Cesar Seijas[1]

[1]*Facultad de Ingeniería, Universidad de Carabobo, Valencia-Venezuela*
[2]*Universidad San Francisco de Quito, Colegio de Ciencias e Ingeniería - El Politécnico. Calle Diego de Robles y Vía Interoceánica, Campus Cumbayá, Edif. Newton. Casilla Postal 17-1200-841, Quito, Ecuador.*

Autor para correspondencia: erith7@gmail.com

Manuscrito recibido el 19 de agosto de 2014. Aceptado, tras revisión, el 1 de junio de 2015.

Resumen

En las últimas décadas son muchos los avances que han tenido lugar en el desarrollo de aplicaciones y alcances de las redes neuronales artificiales, y de igual modo el desarrollo tecnológico en el área de la computación. Este tipo de avances han incidido directamente en el número de publicaciones de aplicaciones, en diversas áreas del conocimiento, basadas en este método de inteligencia artificial. Ahora bien, hasta el presente sigue siendo tema de discusión la idoneidad y aplicabilidad de herramientas de software libre para facilitar la implementación y la calidad de resultados. En este contexto, este trabajo representa un análisis comparativo de aplicaciones usando las librerías ALGLIB y Open NN (Open Source Neural Networks C++ Library), orientadas al entrenamiento y reproducción de redes neuronales artificiales. De igual modo, se establece una evaluación de los resultados obtenidos a partir de los niveles de correlación entre la salida de valores para redes entrenadas y un conjunto de datos de entrenamiento simulados.

Palabras claves: Redes Neuronales Artificiales, ALGLIB, Open NN, Algoritmo Quasi-Newton, C++.

Abstract

In the last decades, there have been a considerable amount of innovations in the development of applications and the scope of artificial neural networks, and likewise the technological development in computer science. These improvements have had a direct effect in the number of publications on applications, in diverse areas of knowledge, based on this artificial intelligence method. Until now, the adequacy and applicability of free software tools to facilitate the implementation and the quality of results is still under discussion. In this context, this work presents a comparative analysis of such applications using libraries ALGLIB and Open NN, oriented to training and reproduction of artificial neural networks. Also, we propose an evaluation of the results obtained from the levels of correlation between the output values for trained networks and a set of data for simulated training.

Keywords: Artificial neural networks, ALGLIB, Open NN, Quasi-Newton algorithm, C++

Forma sugerida de citar: Muñoz, E. y C. Seijas 2015. **Análisis Comparativo de Ajuste en Entrenamiento de Redes Neuronales artificiales a partir de las Librerías Open NN y ALGLIB.** La Granja: Revista de Ciencias de la Vida. Vol. 21(1): 49-60. ISSN: 1390-3799.

1. Introducción

El objetivo de este trabajo es presentar un análisis comparativo entre las librerías Open NN y ALGLIB para entrenamiento y reproducción de redes neuronales artificiales, ambas de características de software libre, a partir de la evaluación del ajuste en un proceso de entrenamiento. Los datos usados para el entrenamiento de las redes neuronales fueron simulados siguiendo la metodología presentada en la guía de usuarios del RTTOV v10 (Radiative Transfer for TOVS satellite), para la simulación de radianzas debido a la dispersión de microondas por nubes y precipitación (Hocking *et al.*, 2012). Estos datos están compuestos por 22 variables de entrada correspondientes a los 22 canales del sensor ATMS a bordo de la plataforma Suomi-NPP, y una variable de salida representada por la tasa de precipitación. Además, los datos comprenden observaciones sobre océano y continente en período de verano.

En este contexto, se entrenan redes neuronales para que las salidas generen tasas de precipitación a partir de las temperaturas de brillo proporcionadas por datos de los canales del sensor ATMS a bordo de la plataforma Suomi-NPP. Con la finalidad de evaluar las diferencias inherentes a la estacionalidad de procesos de precipitación se entrenan redes neuronales artificiales sobre superficie de océano y continente en condiciones de verano en referencia al hemisferio norte.

2. Generalidades

Las redes neuronales artificiales representan una de las técnicas más conocidas de las que forman parte del área de Inteligencia Artificial, la cual algunos autores consideran, que ha sido inspirada por la naturaleza de la inteligencia humana en su afán por comprender y desarrollar entidades inteligentes simples que permitan crear sistemas de inteligencia complejos. Haykin (1999) proporcionó el siguiente concepto para ANNs: Una Red Neuronal Artificial es un procesador distribuido dotado de alto nivel de paralelismo construido a partir de unidades de procesamientos simples, las cuales tienen una capacidad natural de almacenar conocimiento conforme a la experiencia y disponer de dicho conocimiento para su uso (Haykin, 1999).

De acuerdo a Mas y Flores (2008), la definición presentada por Haykin (1999) asemeja a las ANNs con el cerebro en dos aspectos, en primer lugar resalta el hecho de que el conocimiento es adquirido por la red desde su medio, a través de un proceso de aprendizaje y por otra parte las fuerzas de las conexiones inter-neuronas, conocidas como pesos sinápticos, son usadas para almacenar el conocimiento adquirido. El primer aspecto está relacionado a las características de los datos de entrada y salida, así como con los mecanismos de entrenamiento de la red, mientras que el segundo aspecto está relacionado con la estructura de la ANN. A continuación se proporciona un tratamiento formal del conjunto de tópicos asociados con ANNs usados en esta investigación.

De acuerdo con Haykin (1999), toda estructura compleja de ANNs está conformada por una unidad de procesamiento simple denominada neurona (Haykin, 1999). Una neurona puede considerarse como una entidad que emite una respuesta (salida) debido a la estimulación generada (activación) por el recibimiento de señales conocidas (entradas). En otras palabras, la neurona recibe información de entrada para la cual ella emite respuestas en función de un conocimiento previo, lo cual implica que si la neurona recibe estímulos desconocidos, la respuesta pudiera ser no acorde a lo esperado. La corrección de respuestas no esperadas por parte de la neurona que recibe nuevos estímulos, se resuelve mediante la aplicación de procesos de entrenamiento.

En la Figura 1 se aprecia el modelo matemático de la estructura de una neurona, donde considerando que j es el índice de identificación de la neurona estimulada, se tiene entonces que $X = \{x_1, x_2, x_3, \ldots, x_n\}$ es el vector de entradas que contiene información de n neuronas inter-conectadas con j. Por otro lado w_{ij} es el peso sináptico entre cada una de las n neuronas, identificadas con $i = 1, 2, 3, \ldots, n$, y la neurona j; θ_j es el umbral de activación de la neurona j también conocido como umbrales. Por otra parte, f es la función de activación asociada a la neurona j, generalmente se usa una función sigmoidal, tal como la tangente hiperbólica. La relación matemática entre las entradas y la salida y_j de la neurona esta dado como:

$$y_j = f\left(\sum_{i=1}^{n} w_{ij}x_i - \theta_j\right) \tag{1}$$

La ecuación 1 representa una generalización simple de la ecuación de McCulloch-Pitts (Hertz *et al.*,

1991), lo cual hace referencia a la similitud entre el modelo biológico neuronal con el modelo matemático presentado, con la salvedad de que en este caso, la relación entrada-salida no considera dependencia temporal, es decir es una relación atemporal y por otro lado no se consideran en la salida efectos de acumulación debido a los datos de entrada, lo cual se traduce en que no se almacena información sobre estados anteriores de las entradas, sólo el momento actual (Xingui y Shaohua, 2010). La forma en que las neuronas son dispuestas en una red determinan la arquitectura o topología de la red, la cual está estrechamente relacionada con la selección del algoritmo de entrenamiento (Mas y Flores, 2008).

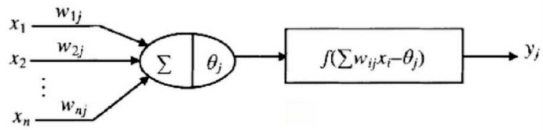

Figura 1. Modelo matemático de una neurona (Xingui y Shaohua, 2010).

2.1 El modelo del perceptrón

El modelo del perceptrón fue introducido por Rosenblatt (1958) y también de forma independiente por Widrow y Hoff (1960), como un mecanismo capaz de ser entrenado en forma supervisada y a su vez como la arquitectura más simple de ANNs. En la Figura 2 se muestra la arquitectura del perceptrón, el mismo consta de una capa de entrada conectada a una capa de salida, en la capa de entrada el número de neuronas n es igual a la cantidad de datos de entrada x_i, mientras que en la capa de salida se tienen m neuronas de salida y_j.

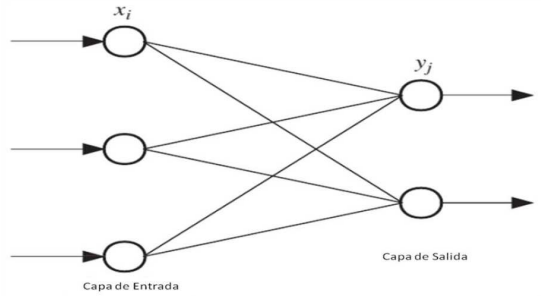

Figura 2. Algoritmo para la estimación de Temperatura (Hsieh, 2009).

La salida y_j de una neurona en términos de la combinación lineal de entradas x_j, está definida por una función de activación. Existen diferentes tipos de funciones de activación entre las cuales se destacan, por ser usados en la presente investigación, la función lineal y las funciones sigmoidales (Hiperbólica y Logística).

La forma de las funciones sigmoidales permiten simular la actividad de una neurona biológica, debido a que a partir de las combinaciones de los diferentes estímulos recibidos por una neurona, es posible describir estados de transición intermedios entre los estados absolutos de activación y no- activación en la respuesta neuronal. Tal y como se aprecia en la figura 3, la función de activación para respuestas binarias está representada en la subfigura 3.a por la función escalón, en la subfigura 3.b se muestra la función lineal, y en la 3.c y 3.d las funciones sigmoidales.

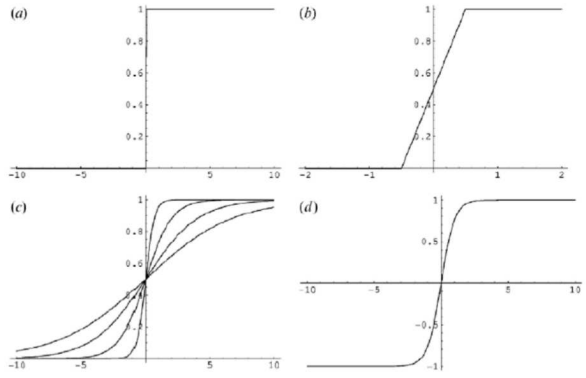

Figura 3. Funciones de activación: a) Escalón , b) Lineal, c) Logística, d) Hiperbólica (Mas y Flores, 2008).

En referencia al modelo del perceptrón debe añadirse que las limitaciones fueron puestas en evidencia en el trabajo de Minsky y Papert (1969), en el cual tras un estudio basado en aplicaciones de lógica operacional, el perceptrón no permitió conseguir soluciones para el problema del *XOR*, debido a que es un problema linealmente no separable, destacando por ende, la aplicabilidad de la arquitectura del perceptrón exclusivamente a problemas linealmente separables.

2.2 El modelo del perceptrón multicapa

Posterior al conocimiento de las limitaciones del modelo del perceptrón relacionadas al requerimiento de la linealidad de los datos de entrada y salida, se pensaba que esta dificultad podía ser superada mediante la inclusión de capas ocultas entre la capa de entrada y salida, sin embargo para la época no había ningún algoritmo para resolver ANNs multicapas. El trabajo de Rumelhart *et al.* (1986), fijó las bases que permitieron posteriormente el desarrollo del Perceptrón Multicapa (MLP, por sus siglas en inglés). En el mismo se presenta el re-descubrimiento del Algoritmo de Propagación Dirigida hacia Atrás, el cual había sido presentado previamente por Werbos (1974).

En la Figura 4 se muestra la estructura de un MLP de arquitectura $x_i - h_j - y_k$ (Una capa de entrada x conformada por i neuronas, una oculta h de j neuronas y una capa de salida y de k neuronas). Esta arquitectura de ANN es conocida como Perceptrón Multicapa Dirigido hacia Adelante (MLPFF, por sus siglas en inglés) la cual se caracteriza por el hecho de que, cada neurona de cada capa, está conectada con cada neurona de la próxima capa.

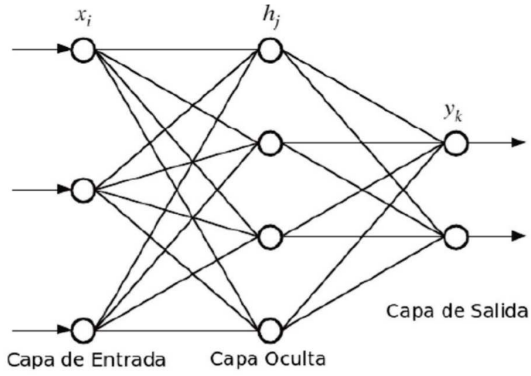

Figura 4. Arquitectura de un MLP, en la cual se muestra la capa de entrada de datos, una capa oculta, y una de capa de salida (Hsieh, 2009).

En este caso se deben implementar dos funciones de activación una para activar a las neuronas de la capa oculta al ser estimuladas por las neuronas de la capa de entrada , y otra para activar las neuronas de la capa de salida tras ser estimuladas por las neuronas de la capa oculta. En este contexto considerando en primer lugar la salida de las neuronas h_j de la capa oculta, se tiene la siguiente ecuación:

$$h_j = f\left(\sum_{i=1}^{n} w_{ij}x_i - b_j\right) \tag{2}$$

Donde, de forma similar a la ecuación 1, w_{ij} es el peso sináptico entre cada neurona de entrada i con la j-esima neurona de la capa oculta, f es la función de activación y b_j es el umbral de activación de la neurona de la capa oculta j. Por otra parte, la salida para la neurona y_k de la capa de salida está dada por la ecuación siguiente:

$$y_k = g\left(\sum_{j=1}^{n} \breve{w}_{jk}h_j - \breve{b}_k\right) \tag{3}$$

Donde $\breve{w}_{jk}$ es el peso sináptico entre cada neurona j de la capa oculta con cada neurona y_k de la capa de salida y g es la función de activación. La extensión de la figura 3, para el caso de N capas ocultas resulta natural, siendo necesario obtener la salida en dependencia de la función de activación entre cada dos capas sucesivas, de tal forma como se mostró para h_j y y_k en este ejemplo.

2.3 Entrenamiento de ANNs.

De acuerdo con Blackwell y Chen (2009), generalmente los problemas atmosféricos basados en ANNs están enfocados como un problema de clasificación mediante reconocimiento de patrones (*RP*, por sus siglas en inglés) o como un problema de regresión de funciones. En estimaciones de precipitación, la técnica de *RP* es empleada generalmente para discriminar entre pixeles de precipitación y de no precipitación en una escena, mientras que la regresión de funciones es empleada para estimar valores de tasas de precipitación. En esta investigación las ANNs se entrenan para generar en la salida valores de tasa de precipitación, por lo tanto, el enfoque teórico que sustenta esta investigación está vinculado al entrenamiento de ANNs orientadas a la resolución de problemas de estimación de precipitación mediante regresión de funciones.

En este sentido es propicio el contexto para denotar que, se define entrenamiento de una ANN al proceso de determinar valores óptimos de pesos sinápticos y bias que maximicen la aproximación del conjunto de salidas de datos predichos mediante la

ANN, con el conjunto de datos de salida que representan los datos objetivos de ajuste o de predicción durante la fase de entrenamiento. Las características intrínsecas de la ANN determinan en gran medida su capacidad y propiedad para ser entrenadas, entre estas características se pueden mencionar la estructura de la ANN, forma de inicialización de la red, así como también las fases de regularización que permiten la autoevaluación en tiempo real del proceso de entrenamiento.

El proceso de post-entrenamiento requiere además un proceso de evaluación de la calidad del proceso, el cual puede ser realizado mediante una regresión lineal entre la salida de la red y los datos objetivos, así como también mediante la estimación del coeficiente de correlación. Hsieh (2009) afirma que el proceso de entrenamiento generalmente es llevado a cabo al minimizar una función de costo J, que conforme a su metodología, ha sido definida como un medio del error medio cuadrático (*MSE*, por sus siglas en inglés) entre la salida obtenida por la ANN y las salidas del conjunto de datos usados durante el proceso de entrenamiento (Hsieh, 2009). Matemáticamente la función J tiene la siguiente forma:

$$J = \frac{1}{N}\sum_{n=1}^{N}\left\{\frac{1}{2}\sum_{k}[y_k^n - y_{dk}^n]^2\right\} \quad (4)$$

En la ecuación 4 y_{dk} representa el dato objetivo (meta proporcionada en el entrenamiento), y_k es la salida generada mediante la ANN, $n = 1, 2, 3, \ldots, N$, son el número de casos, observaciones o mediciones. De esta manera, se puede plantear que el proceso de entrenamiento de ANNs consiste en el empleo de algoritmos de optimización (comúnmente denominados algoritmos de entrenamiento) que permitan determinar valores w y b para los cuales la función de costo J resulte minimizada.

2.4 Entrenamiento mediante propagación de errores hacia atrás

El término Propagación hacia Atrás (BP, por sus siglas en inglés) tiene diferentes connotaciones en el campo de las ANN, por una parte se suele asociar este término a la arquitectura de la ANN, y por otra, también se denomina BP al mecanismo de minimización de la función de costo presentada por Rumelhart *et al.* (1986), que es en el contexto en el cual este término es empleado en la presente investigación.

Según Bishop (2005), el algoritmo BP consiste en el entrenamiento de MLPs mediante la aplicación del método del Gradiente Descendiente (DG, por sus siglas en inglés) sobre la función de costo, sin embargo cabe destacar que Xingui (2010) menciona que existen varios métodos mejorados basados en BP, como por ejemplo el Gradiente Conjugado, Newton, Quasi-Newton, Levenberg-Maquardt, entre otros (Xingui y Shaohua, 2010). La optimización de pesos y bias es llevada a cabo mediante BP usando Gradiente Descendente a partir de la actualización iterativa en la dirección en la que la función de costo disminuye mas rápido, es decir en la dirección negativa del gradiente.

En esta investigación se usa el algoritmo Quasi-Newton para el proceso de entrenamiento de las ANNs, debido principalmente a que es un algoritmo de segundo orden mejorado, lo cual implica una buena relación entre rendimiento y tiempo requerido para el entrenamiento, por otro lado este algoritmo está implementado en las dos librerías de licencia abierta para C++, que son OPEN NN y ALGLIB, lo cual permite diseñar condiciones estándares para llevar a cabo el análisis comparativo.

3. Metodología de la investigación

De acuerdo Bellerby *et al.* (2000), una red neuronal artificial de dos capas ocultas proporciona niveles de complejidad adecuados para modelar la no-linealidad entre los datos de entrada y salida asociados a problemas de segundo orden, garantizando además, costos computacionales admisibles para realizar el proceso de regresión no-lineal convencional que involucran los métodos generales de estimación de tasa de precipitación usando ANNs en base a datos satelitales.

En este sentido, con la finalidad de emular los fenómenos físicos involucrados en el proceso de precipitación, en este trabajo se entrenan diferentes modelos de redes neuronales artificiales, todos con una capa de entrada, dos capas ocultas, una capa de salida y se utilizan como datos de entrada diferentes combinaciones de canales del sensor ATMS. Esta topología de red es constante en los diversos experimentos numéricos realizados en esta investigación, sin embargo, se presentan algunas variantes en cuanto al número de neuronas que componen la ca-

pa de entrada y las capas ocultas.

En este orden de ideas, con la primicia de proporcionar mayor información sobre la arquitectura de red neuronal usada en este trabajo, en la Fig. 4 se muestran las diferentes partes que conforman cada red neuronal artificial.

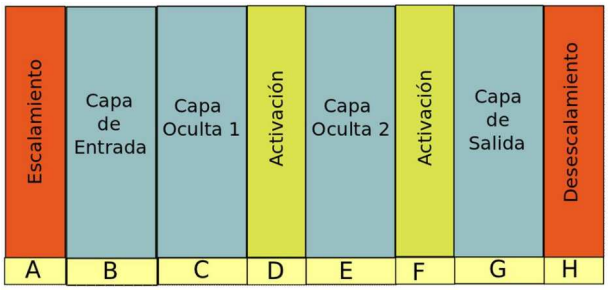

Figura 5. Diseño secuencial de la topología de red neuronal de dos capas ocultas.

3.1 Capa de entrada

La importancia de la capa de entrada se encuentra implícita en el hecho de que la red neuronal debe emular las relaciones físicas y matemáticas que tienen lugar entre las variables que conforman los datos de entrada y salida que representan las magnitudes predichas. En este sentido, deben realizarse varias consideraciones con la finalidad de llevar a cabo una óptima selección de los datos de entrada, entre las que cabe resaltar el hecho de que dichas variables tengan poca correlación entre sí, pero alta correlación con la salida de la ANN, así como también que contengan poco contenido de ruido en el caso de que los datos provengan de instrumentos de medición.

Con respecto a la estructura de la capa de entrada, es importante precisar que el conjunto de variables de entrada define el número de neuronas en esta capa. Así pues, el número de variables de entrada tiene una relación directa con el diseño de la estructura de neuronas de las capas ocultas de la red y por ende con el desempeño de la red entrenada. Es importante además resaltar que en el proceso de entrenamiento de la red neuronal, los datos de inicialización del proceso de entrenamiento entrada están conformados por variables de entrada y variables de salida.

Las variables de entrada están representadas por el conjunto de canales del ATMS que tienen una alta relación con la posibilidad de predecir la tasa de precipitación, y por otro lado los datos de salida son valores de tasa de precipitación simulados para las diferentes combinaciones de los datos de entradas. Posterior a esta separación descriptiva sobre los datos entrada, los mismos son divididos en tres subconjuntos de datos; el primero corresponde a los datos de entrenamiento para la red, el segundo a datos que sirven para la verificación de ajuste en el tiempo del proceso de entrenamiento y el tercero para realizar pruebas sobre el desempeño predictivo de la red neuronal.

Las variables de entrada están representadas por el conjunto de canales del ATMS que tienen una alta relación con la posibilidad de predecir la tasa de precipitación, y por otro lado los datos de salida son valores de tasa de precipitación simulados para las diferentes combinaciones de los datos de entradas. Posterior a esta separación descriptiva sobre los datos entrada, los mismos son divididos en tres subconjuntos de datos; el primero corresponde a los datos de entrenamiento para la red, el segundo a datos que sirven para la verificación de ajuste en el tiempo del proceso de entrenamiento y el tercero para realizar pruebas sobre el desempeño predictivo de la red neuronal.

3.2 Escalamiento y desescalamiento de los datos

En la Fig. 4 se puede apreciar en la etiqueta A que, antes de que los datos sean integrados a la capa de entrada, hay un proceso de escalamiento. El proceso de escalamiento puede ser llevado a cabo cuando los datos representan linealidad en cuanto a la escala, sin embargo es conveniente hacerlo para expresar los datos de entrada en orden de magnitud cero, lo cual ayuda a alcanzar una mejor aproximación por parte del algoritmo de entrenamiento. De igual forma se aprecia que en la etiqueta H, nuevamente se hace el desescalamiento de los datos, lo cual es imprescindible para expresar los resultados en las magnitudes originales.

3.3 Capas ocultas

En la arquitectura de las redes que son entrenadas para los efectos de este trabajo, se han incluido dos capas ocultas, siguiendo las recomendaciones de Blackwell y Chen (2005). De este modo, la ar-

quitectura base para la evaluación del entrenamiento de redes neuronales es 10 neuronas en la primera capa oculta y 5 en la segunda, tomando como referencia el trabajo de Chen *et al.* (2006). A partir de esta arquitectura base, se evalúa el desempeño de cada red entrenada y en función de la calidad de resultados obtenidos se plantean experimentos alternativos con diferentes números de neuronas en ambas capas ocultas.

3.4 Función de activación

En la Fig. 4 se aprecian 2 capas de activación, una con la etiqueta D que corresponde a las neuronas de la capa oculta 1 y otra con la etiqueta F que está asociada a la capa oculta 2, en ambos casos se ha seleccionado como función de activación a la función hiperbólica. Es importante mencionar, que en la Fig. 4 ha sido ignorada una tercera capa de activación que corresponde a las neuronas de la capa de salida, esta omisión se debe al hecho de que la función de activación aplicada en esta etapa es la lineal, la cual para los efectos este trabajo ha sido implementada en una relación uno a uno.

3.5 Características de ALGLIB

ALGLIB es un conjunto de herramientas computacionales y numéricas con versiones comerciales y gratuitas, orientadas a implementaciones matemáticas aplicadas. ALGLIB contiene interfaces de desarrollo para la reproducción y entrenamiento de ANNs, que están compuestas por librerías disponibles para ser compiladas y usadas en varios lenguajes de programación, sin embargo en este trabajo las implementaciones han sido llevadas a cabo en C++ para establecer una comparación con Openn NN, que está desarrollada exclusivamente en este lenguaje.

Dentro de las ventajas que aporta ALGLIB para implementaciones de ANNs resalta el hecho de que se pueden entrenar ANNs tanto para clasificación como para regresión lineal, por otra parte cuenta con tres algoritmos de entrenamiento que son, el L-BFGS (Limited Memory Broyden-Fletcher-Goldfarb-Shanno) que es un método Quasi- Newton con costo de iteraciones fijas cuyo uso es recomendable en problemas con gran cantidad de datos, el

Levenberg-Marquardt que está fundamentado en el uso de la función de error exacta Hessiana, y el método de parada temprana, el cual evita el sobreentrenamiento a partir de la finalización automática del proceso de entrenamiento mediante parámetros determinados en el algoritmo (http://www.alglib.net/dataanalysis/neuralnetworks.php).

Una ventaja destacable de la implementación del algoritmo Quasi-Newton en ALGLIB, es el hecho de que presenta criterios de parada predefinidos, lo cual garantiza una convergencia óptima durante el proceso de entrenamiento, y al mismo tiempo proporciona 3 parámetros que permiten la modificación del criterio de parada sin perder generalidad en las características de optimización. De este modo, es posible establecer procedimientos de entrenamiento de ANNs, con pocos pasos de configuración del algoritmo y de forma óptimizada.

3.6 Características de Open NN

Open Neural Network (Open NN) es una librería gratuita basada en C++ para el entrenamiento y reproducción de ANNs. Posee herramientas para el manejo y disposición de datos, además de contar con una amplia variedad de algoritmos de entrenamiento entre los que resaltan el método del Gradiente Descendiente, Gradiente Conjugado, Newton, Quasi-Newton, Levenberg-Marquardt, entre otros. Al igual que ALGLIB proporciona herramientas para entrenar redes para clasificación y regresión lineal, pero a diferencia de ALGLIB, Open NN tiene una documentación que incluye más de 10 ejemplos de soluciones de problemas a partir del entrenamiento de ANNs por diferentes algoritmos (http://www.cimne.com/flood/Links.asp).

La arquitectura de Open NN, y la implementación del algoritmo Quasi-Newton, permiten al usuario crear estrategias de entrenamiento a partir de la configuración de un criterio de parada por un incremento mínimo de desempeño, cumplimiento de meta, meta de gradiente normal, y también por la evaluación de decremento de generalización máxima. También es posible establecer configuraciones para llevar a cabo el proceso de entrenamiento en 3 fases, de este modo se usa un primer algoritmo para llevar a estabilidad y convergencia los bías y pesos sinápticos (Generalmente un algoritmo de primer orden como el Gradiente Descendente), uego el segundo algoritmo que realiza el proceso de entrenamiento, y un tercer algoritmo para optimizar el

proceso.

Para los efectos de esta investigación se ha usado un solo algoritmo, para evaluar las características de ajuste de cada librería en condiciones similares. De igual forma es importante mencionar que el criterio de parada utilizado fue el de la evolución de generalización, ya que evita el sobreentrenamiento de la ANN.

4. Resultados

A continuación se presenta el diseño de las ANNs para cada experimento, así como también los resultados derivados para cada una de las librerías:

4.1 Entrenamiento sobre superficie de continente

En la tabla 1 se aprecia el conjunto de experimentos diseñados para el proceso de entrenamiento en verano, cada uno tiene asociado un número de referencia, un código para facilitar su identificación durante el entrenamiento de la red neuronal, y finalmente los canales seleccionados como datos de entrada.

Número	Código	Canales
1	LS_1	3,4,5,16,17,18,19,20,21,22
2	LS_2	18,19,20,21,22
3	LS_3	5,17, 18,19,20,21,22

Tabla 1. Diseño de experimentos de entrenamiento de la ANN para la estimación de tasa de precipitación sobre continente en período de Verano (LS son siglas en inglés que hacen referencia a Land-Summer, lo cual se interpreta como dato para pixel sobre continente en época de Verano).

El conjunto de datos simulados usados para este entrenamiento, involucra 25384 casos de combinaciones de los canales ATMS asociados con diferentes niveles de tasa de precipitación que comprenden valores entre 6.30×10^{-4} y 61.43 mm/h. Los resultados del entrenamiento de redes, obtenidos para esta experiencia, se muestran en la Figura 5 y Figura 6, donde se han representado los resultados derivados a partir de la librería ALGLIB y los obtenidos mediante la librería Open NN respectivamente.

La primera observación a destacar es que ALGLIB proporciona un coeficiente de correlación 21,41 % más alto que Open NN, este resultado puede ser observado de forma visual a partir de la regresión lineal y la línea de ajuste mostrada en cada caso, siendo relevante la calidad del ajuste obtenido a partir de ALGLIB.

En este experimento, ALGLIB reportó un valor de 2,59 para el error cuadrático medio, mientras que Open NN la figura 7 muestra la diferencia de errores, durante el proceso de entrenamiento, entre los datos de salida de la red y los datos de prueba.

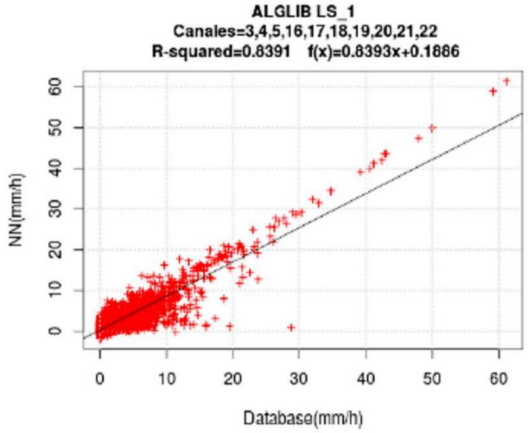

Figura 6. Gráfico de dispersión entre los valores de tasa de precipitación estimados por la ANN y el valor objetivo de los datos de entrenamiento (Database) para entrenamiento mediante ALGLIB del experimento LS_1.

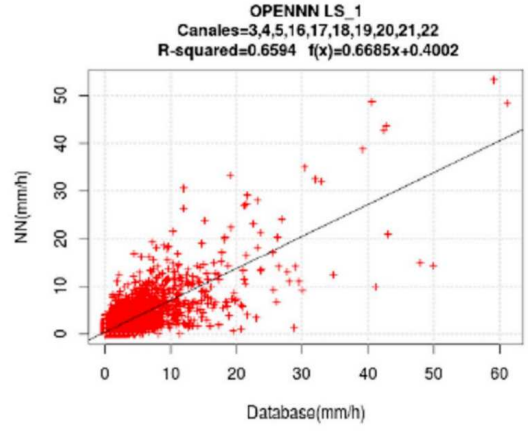

Figura 7. Gráfico de dispersión entre los valores de tasa de precipitación estimados por la ANN y el valor objetivo de los datos de entrenamiento para entrenamiento mediante OpenNN del experimento LS_1.

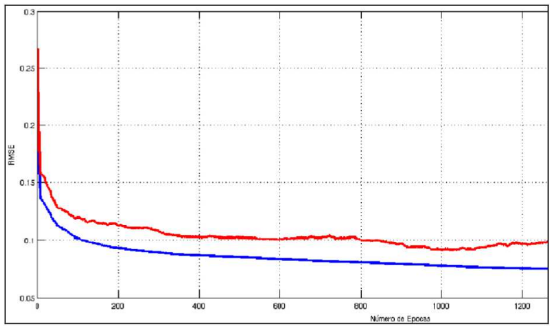

Figura 8. Error Cuadrático Medio del desempeño del proceso de entrenamiento(azul) y de los datos de prueba(rojo) durante el proceso de entrenamiento de la red en Open NN del caso LS_1.

Idealmente, en la Figura 7 es de esperarse dos curvas superpuestas, lo cual indica que el error entre la salida de la red(curva inferior de la figura) en entrenamiento reproduce perfectamente los datos de prueba(curva superior de la figura), sin embargo, en lugar de esto se tiene entonces que cuando la curva roja asociada a los datos de prueba(Generalization) se aproxima a los datos de salida de la red en función de las épocas se tiene un proceso de entrenamiento normal, ahora bien cuando esta curva de prueba se aleja por arriba de la curva azul(Performance), como sucede en la Fig. 7 a partir de la época 1000, se dice que la red está sobreentrenando lo cual se traduce en pérdida de generalidad por parte del proceso de entrenamiento.

Por otro lado, cuando la curva roja se aleja de la azul por debajo, el resultado es un subentrenamiento de la red. En todo caso es necesario evitar tanto el sub-entrenamiento, como el sobreentrenamiento de la red, en orden de llevar a cabo un buen proceso de entrenamiento. Particularmente, en este caso se tomó como pesos y bias de la red, los valores correspondientes a 1000 épocas.

Es importante mencionar, que esta gráfica no fue posible generarla para el entrenamiento mediante ALGLIB, porque esta librería no proporciona la posibilidad de almacenar valores de RMSE para cada ciclo, mientras que con Openn NN fue posible obtenerla con una sencilla modificación de un código, también es importante aclarar que estas graficas son naturales en Open NN ya que el criterio de parada no es automático y un criterio para establecer paradas de entrenamiento en Open NN es definir un valor máximo de diferencia en RMSE entre los datos de performance y generalization.

Las Figs. 8 y 9 corresponden al segundo experimento, etiquetado como LS_2. En este caso el coeficiente de correlación obtenido por ALGLIB es 24,67 % superior al de Open NN.

Respecto a la calidad de este resultado, se puede destacar que una correlación de 0.451 es baja y más aun considerando que esta correlación describe la capacidad de predicción de datos simulados y no de datos reales, en cuyo caso se espera disminuya aun más dicha correlación. Este resultado se atribuye al hecho de que estos 5 canales, por si solos no contienen suficiente información como para estimar precipitación. Por tal motivo, la inclusión de canales que permitan aumentar la correlación entre los datos de entrada y de salida, son una excelente opción para aumentar los niveles de predicción.

Experimento	R^2 con ALGLIB	R^2 con OPENNN	Diferencia porcentual (%)
LS_1	0.8391	0.6594	21.41
LS_2	0.4510	0.3397	24.67
LS_3	0.5854	0.3846	34.30

Tabla 2. Resumen de resultados de coeficientes de correlación para las redes neuronales de estimación de tasa de precipitación sobre continente en época de verano.

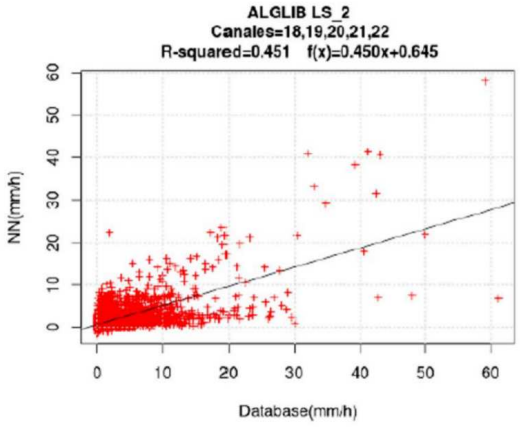

Figura 9. Gráfico de dispersión entre los valores de tasa de precipitación estimados por la ANN y el valor objetivo de los datos de entrenamiento para entrenamiento mediante ALGLIB del experimento LS_2.

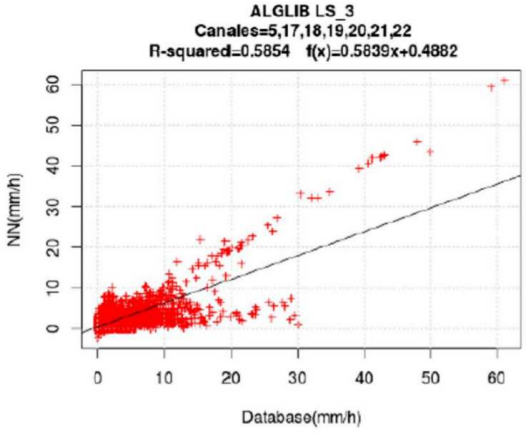

Figura 11. Gráfico de dispersión entre los valores de tasa de precipitación estimados por la ANN y el valor objetivo de los datos de entrenamiento para ntrenamiento mediante ALGLIB del experimento LS_3.

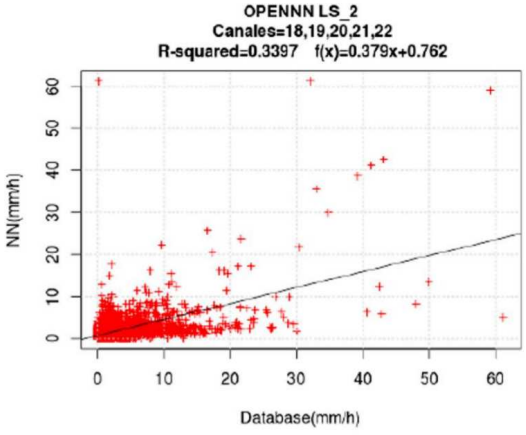

Figura 10. Gráfico de dispersión entre los valores de tasa de precipitación estimados por la ANN y el Open NN del experimento LS_2.

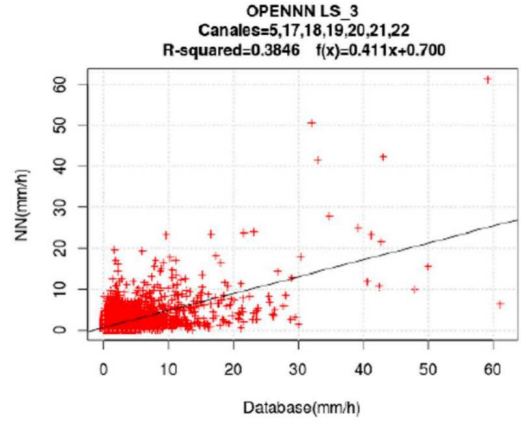

Figura 12. Gráfico de dispersión entre los valores de tasa de precipitación estimados por la ANN y el valor objetivo de los datos de entrenamiento para entrenamiento mediante Open NN del experimento LS_3.

Ahora bien, en el caso del tercer experimento, etiquetado como LS_3, ALGLIB arrojó un coeficiente de correlación 34,30 % mayor que Open NN, lo cual hace referencia a un mejor ajuste entre las tasa de precipitación predicha por la red neuronal y los datos simulados de entrenamiento, por parte de ALGLIB, tal y como se puede apreciar a partir de las figuras 10 y 11.

En la tabla 2, se puede observar un resumen sobre los resultados en coeficiente de correlación obtenidos mediante el entrenamiento con ambas librerías, así como la diferencia porcentual en el ajuste.

4.2 Entrenamiento sobre superficie de océano

Las combinaciones de canales para el entrenamiento de redes neuronales para la estimación de tasa de precipitación sobre océano en época de verano se pueden apreciar en la tabla 3, donde se muestran los tres experimentos diseñados. Los resultados obtenidos a partir de cada una de las tres redes neuronales entrenadas, para un conjunto de datos de entrenamiento compuesto por 9006 casos, se apre-

cian resumidos en la tabla 4 donde nuevamente se pueden apreciar resultados tanto para los entrenamientos llevados a cabo mediante ALGLIB, así como también por Open NN.

Número	Código	Canales
1	OS_1	3,4,5,16,17,18,19,20,21,22
2	OS_2	18,19,20,21,22
3	OS_3	5,17, 18,19,20,21,22

Tabla 3. Diseño de experimentos de entrenamiento de la ANN para la estimación de tasa de precipitación sobre océano en periodo de verano (OS son siglas en inglés que hacen referencia a Ocean-Summer, lo cual se interpreta como dato para pixel sobre océano en época de verano).

Experimento	R^2 con ALGLIB	R^2 con OPENNN	Diferencia porcentual (%)
OS_1	0.9066	0.7327	19.18
OS_2	0.5530	0.2382	56.92
OS_3	0.7493	0.4655	37.87

Tabla 4. Resumen de resultados de coeficientes de correlación para las redes neuronales de estimación de tasa de precipitación sobre continente en época de verano.

5. Conclusiones

En la tabla 2 se puede apreciar una diferencia porcentual de ajuste cercana a 50 % entre ALGLIB y Open NN cuando del experimento 1 se extraen las entradas de los canales 3, 4, 5, 16 y 17, de tal modo que ha sido posible constatar que el numero de variables de entrada, el nivel de correlación entre sí, y con la variable de salida juega un papel predominante en las características de predicción de la ANN entrenada. En este sentido, una recomendación de paso previo al ingreso de datos de entrada en el proceso de entrenamiento, consiste en ingresar datos con poca correlación lineal entre sí, y con alta correlación con la variable a predecir.

El nivel de relación entre las variables de entrada y salida, puede ser inferido a partir del conocimiento previo sobre las características propias de la relación entre las variables, si bien es cierto que en este trabajo se ha evitado discutir sobre las características físicas de cada uno de los canales del sensor ATMS y las relaciones que tienen con la variable precipitación, la selección de los canales considerados en cada experimento tiene su justificación en conocimientos previos por parte del autor sobre principios de teledetección de la atmósfera y fenómenos de transferencia radiativa, cuya descripción se ha omitido en el presente trabajo para evitar desenfoques en el objetivo de la investigación, que es proporcionar al lector un estudio comparativo sobre el nivel de ajuste proporcionado por ALGLIB y Open NN para los experimentos objetos de este estudio.

Con base en los resultados mostrados en las tablas 2 y 4 se concluye que bajo la configuración diseñada para Open NN, en cuanto a criterio de parada, estrategia de entrenamiento y parametrización, se tiene en todos los casos un mayor ajuste y capacidad de predicción mediante ALGLIB. Por otra parte es importante mencionar, que el tiempo requerido para el proceso de entrenamiento que Open NN invirtió para cada proceso es de aproximadamente 6 horas en promedio, mientras que ALGLIB solo dos. Estos resultados, pudieran mostrar otra tendencia si la configuración de corrida para el proceso de entrenamiento es optimizada para Open NN, lo cual es factible para usuarios con conocimientos avanzados en este tipo de algoritmos, esto permite inferir que para usuarios con conocimientos introductorios o intermedios en ANNs y algoritmos de entrenamientos, factiblemente obtendrán mejores resultados a partir de las configuraciones preestablecidas en ALGLIB.

Finalmente, cabe destacar que la arquitectura de ANN utilizada en cada caso fue una cada de entra-

da, dos capas ocultas, la primera con el doble número de neuronas que la capa de entrada y la segunda con igual número, y una capa de salida conformada por una neurona.

Referencias

Bellerby, T., M. Todd, D. Kniveton y C. Kidd. 2000. **Rainfall estimation from a combination of trmm precipitation radar and goes multispectral satellite imagery through the use of an artificial neural network**. American Meteorological Society, págs. 2115–2128.

Bishop, C. 2005. **Neural network for pattern recognition**.

Blackwell, W. y F. Chen. 2009. **Neural Networks in Atmospheric Remote Sensing**. Massachusetts Institute of Technology.

Blackwell, W. J. y F. Chen. 2005. **Neural network applications in high-resolution atmospheric remote sensing**. Lincoln Laboratory Journal, págs. 299–322.

Chen, F., L. Bickmeier, W. Blackwell, R. Leslie, D. Staelin y C. Surussavadee. 2006. **Satellite-based estimation of precipitation using passive opaque microwave radiometry**. Workshop of the International Precipitation Work-ing Group, Melbourne, Australia.

Haykin, S. 1999. **Neural networks. a comprehensive foundation**. Upper Saddle River.

Hertz, J., A. Krogh y R. Palmer. 1991. **Introduction to the theory of neural computation**.

Hocking, J., P. Rayer, R. Saunders, M. M. A. Geer y P. Brunel. 2012. **Rttov v10 users guide**. ECMWF, KNMI and MeteoFrance.

Hsieh, W. 2009. **Machine learning methods in the environmental sciences. neural networks and kernels**.

http://www.alglib.net/dataanalysis/neuralnetworks.php.

http://www.cimne.com/flood/Links.asp.

Mas, J. y J. Flores. 2008. **The application of artificial neural networks to the analysis of remotely sensed data**. International Journal of Remote Sensing, págs. 617–663.

Minsky, M. y S. Papert. 1969. **Perceptrons**.

Rosenblatt, F. 1958. **The perceptron: A probabilistic model for information storage and organization in the brain**. Psychological Review, págs. 386–408.

Rumelhart, D. E., G. E. Hinton y R. J. Williams. 1986. **Learning internal representations by error propagation. Parallel Distributed Processing**. págs. 318–62.

Werbos, P. J. 1974. **Beyond regression: new tools for prediction and analysis in the behavioural sciences**.

Widrow, B. y M. E. Hoff. 1960. **Adaptive switching circuits**. IRE WESCON Convention Record, págs. 96–104.

Xingui, H. y X. Shaohua. 2010. **Process neural network: Theory and applications**.