ACOTACIÓN DEL ERROR DE MODELOS DE REDES NEURONALES APLICADOS AL PRONÓSTICO DE SERIES DE TIEMPO

JUAN DAVID VELASQUEZ H.
M.Sc. en Ingeniería de Sistemas, Dr. en Ing. – Sistemas Energéticos
Profesor Asociado
Universidad Nacional de Colombia
jdvelasq@unal.edu.co

Fecha de recibido: 31/05/2010
Fecha de aprobación: 15/06/2011

RESUMEN

Las redes neuronales artificiales son una importante técnica en el pronóstico de series de tiempo no lineales. Sin embargo, el entrenamiento de las redes neuronales es una tarea difícil, a causa de la presencia de muchos puntos óptimos locales y a la irregularidad de la superficie de error. En este contexto, es muy fácil obtener modelos sub-entrenados o sobre-entrenados sin poder de pronóstico. Así, los investigadores y los profesionales necesitan contar con criterios para detectar esta clase de problemas. En este artículo, se demuestra que el uso de metodologías bien conocidas en el pronóstico de series de tiempo lineales, tales como la metodología de Box-Jenkins o los modelos de suavizado exponencial, son valiosas herramientas para detectar modelos de redes neuronales mal especificados.

PALABRAS CLAVE: pronóstico, suavizado exponencial, modelos ARIMA, perceptrones multicapa, modelos no lineales.

ABSTRACT

Artificial neural networks are an important technique in nonlinear time series forecasting. However, training of neural networks is a difficult task, because of the presence of many local optimal points and the irregularity of the error surface. In this context, it is very easy to obtain under-fitted or over-fitted forecasting models without forecasting power. Thus, researchers and practitioner need to have criteria for detecting this class of problems. In this paper, we demonstrate that the use of well known methodologies in linear time series forecasting, such as the Box-Jenkins methodology or exponential smoothing models, are valuable tools for detecting bad specified neural network models.

KEY WORDS: forecasting, exponential smoothing, ARIMA models, multilayer perceptrons, nonlinear models.

1. INTRODUCCIÓN

En la últimas dos décadas se ha presentado un interés creciente por la predicción de series de tiempo que presentan características no lineales; véase, por ejemplo, [1] y [2]. Dentro de la amplia gama de modelos existentes, las redes neuronales artificiales han gozado de amplia aceptación, especialmente, los perceptrones multicapa [3].

No obstante, su proceso de especificación sigue siendo basado en criterios heurísticos y en el juicio del modelador [4]; aunque recientemente se han presentado avances importantes en el desarrollo de metodologías para determinar su configuración óptima, no hay un procedimiento aceptado de forma general.

Posiblemente, una de las razones que explican la persistencia de esta problemática a lo largo de los años, es la dificultad de estimar los parámetros del modelo [5]; ello se debe a que decisiones como la cantidad de neuronas ocultas o la selección de los regresores o variables explicativas parten del supuesto de que el modelo está correctamente entrenado y se han encontrado los mejores pesos (parámetros) posibles para una configuración dada [6].

Es bien sabido, que los modelos no lineales sufren de dos problemas fundamentales en lo referente a la estimación de sus parámetros: por un lado, el modelo puede estar subestimado, esto es, el proceso de optimización se detuvo de forma temprana y es posible, aún, obtener un menor error de ajuste a la muestra de calibración. En este caso, el modelo se ajusta pobremente a los datos, y sus pronósticos tienen un error muy alto.

Por el otro lado, el modelo puede memorizar los datos de calibración (entrenamiento), fenómeno conocido usualmente como sobre-entrenamiento, de tal manera, que se ajusta con un error muy bajo a la muestra de calibración, pero los pronósticos son bastante lejanos de la realidad; consecuentemente, el modelo obtenido carece de valor práctico.

El problema fundamental radica en que no se poseen herramientas conceptuales o teóricas que permitan concluir cuando se ha presentado uno de los dos problemas anteriores (sub-entrenamiento o sobre-entrenamiento). Los objetivos de este artículo son: clarificar este problema y demostrar como otras técnicas más simples de pronóstico pueden ser usadas para juzgar si se ha obtenido un modelo adecuado.

Este artículo está organizado como sigue: en la Sección 2, se define que es una serie de tiempo y el concepto de no linealidad; seguidamente en la Sección 3, se discuten las principales razones que explican la dificultad del proceso de entrenamiento de una red neuronal. En la Sección 4, se identifica el problema práctico y cómo se pueden utilizar modelos más tradicionales para su detección. A continuación, en la Sección 5, se discuten algunos aspectos prácticos. En la Sección 6, se concluye.

2. SERIES DE TIEMPO

Una serie de tiempo es un conjunto ordenado de observaciones:

$$y[1], y[2],\ldots,y[T] \quad \textbf{(1)}$$

El objetivo del pronóstico es obtener una función $f()$, que permita calcular un estimado del valor en el instante actual, dados sus valores pasados:

$$y[t]=f(y[t\text{-}1],y[t\text{-}2],\ldots)+e[t] \quad \textbf{(2)}$$

Donde e es una variable aleatoria que representa el error inherente al hacer el pronóstico. Se considera que la función $f()$ es lineal cuando ella es definida como una sumatoria y el error sigue una distribución normal con media cero y desviación estándar desconocida σ. Por ejemplo, si el valor actual $y[t]$ es función de los dos valores anteriores, entonces (2) puede escribirse como:

$$y[t]=\phi_1 y[t\text{-}1]+\phi_2 y[t\text{-}2]+e[t] \quad \textbf{(3)}$$

Con $e[t]\sim\sigma N(0,1)$. ϕ_1 y ϕ_2 son los parámetros del modelo. El modelo definido en (3) se conoce como autorregresivo. Un modelo es no lineal cuando su representación matemática no puede reducirse a (3).

En el uso de (3) se parte del supuesto que la serie de tiempo es ergódica, que en términos simples, puede entenderse como que su media y varianza no cambian en el tiempo. Ello implica que la serie no exhibe una tendencia determínistica ni patrones cíclicos repetitivos. Cuando la serie presenta estas características, se requiere realizar un proceso mucho más elaborado para obtener un modelo aditivo como el especificado en (3). Uno de estos procesos es la metodología de Box y Jenkins cuyos modelos se conocen de forma genérica como ARIMA. Una discusión profunda sobre las propiedades de esta familia de modelos y su proceso de especificación está por fuera de los alcances de este artículo, y se invita al lector a consultar otras fuentes como [7].

3. REDES NEURONALES

Un perceptrón multicapa (MLP, por su sigla en inglés) es un modelo estadístico [8] inspirado en la arquitectura de los circuitos neuronales del cerebro [véase la Figura 1]; puede aproximar cualquier función continua definida en un dominio compacto con un nivel de precisión arbitrario definido de antemano [9]; se caracteriza por ser muy tolerante a información incompleta, inexacta o contaminada con ruido [10] [11]. Uno de sus principales usos es el pronóstico de series de tiempo [3]. En este caso, la forma funcional del MLP de la Figura 1 se puede escribir en términos de la propagación de la señal a través de la red neuronal. En este caso, la propagación desde la capa de entrada hasta la salida de las neuronas de la capa oculta será:

$$\Psi_h = \tanh\left(\alpha_{0,h}+\sum_{p=1}^{P}\alpha_{p,h}\,y[t-p]\right) \quad \textbf{(4)}$$

Mientras que la salida de la red neuronal puede calcularse como

$$f\ (y[t\text{-}1], y\ [t\text{-}2],\ldots)=\tanh[\beta_0+\sum_{h=1}^{H}\beta_h\Psi_h] \quad \textbf{(5)}$$

El problema de identificación o estimación de parámetros consiste en determinar los valores del

vector $\Omega=[\alpha_{0,1},\ldots,\alpha_{P,H},\beta_0,\ldots,\beta_H]$ cuando se ha definido previamente las entradas del modelo (P) y la cantidad de neuronas en la capa oculta (H).

La estimación de los parámetros del MLP definido en (4) y (5) se basa, usualmente, en la minimización del error cuadrático medio (MSE, por su sigla en inglés):

$$MSE=\sum_{t=P+1}^{T}[y[t]-f\ (y[t-1],\ldots)]^2 \qquad \textbf{(6)}$$

Se sabe que la estimación de los valores del vector Ω es un problema numérico difícil, debido a la multiplicidad de puntos de mínima local y a la complejidad de función de error (5) debido a que:

1. El modelo puede estar mal condicionado, lo cual es síntoma de que las diferencias entre las magnitudes de los parámetros Ω son muy grandes; ello puede causar que las neuronas en la capa oculta estén en su nivel de saturación (diferentes entradas netas producen la misma salida), y consecuentemente que el gradiente de (5) sea cercano a cero, por lo que la optimización numérica se hace ineficiente.

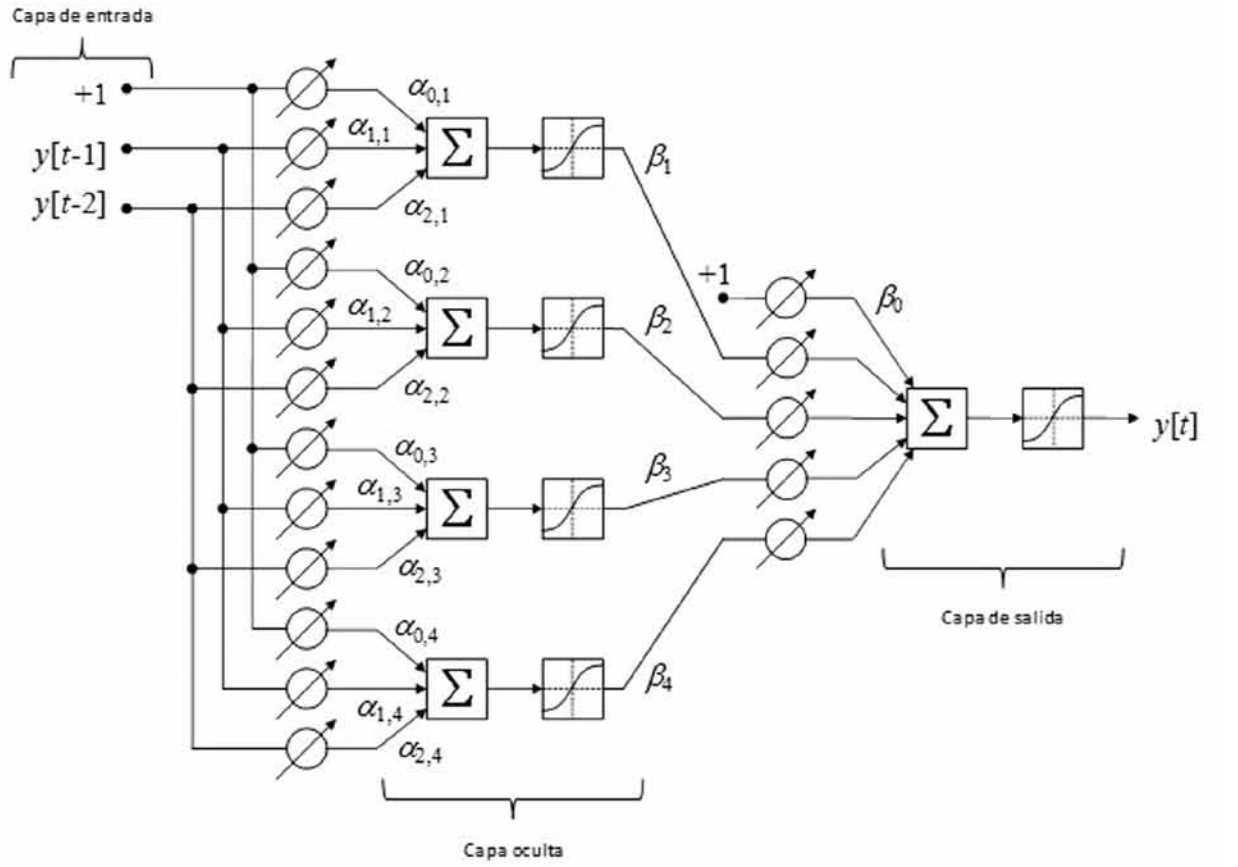

Figura 1. *Esquema representativo de un perceptrón multicapa de dos entradas, una capa oculta con cuatro neuronas y una salida.*

2. Se pueden obtener modelos diferentes con el mismo error al permutar las neuronas de la capa oculta sin variar los valores de los parámetros, lo que equivale a intercambiar algunos de los elementos de Ω.

3. Se pueden obtener modelos diferentes con el mismo error, cambiando de signo los parámetros asociados a las conexiones que entran y salen de la misma neurona oculta, ya que para funciones de activación tal como la tangente hiperbólica, se cumple que $\tanh u = -\tanh(-u)$

El modelo no es globalmente identificable, si la salida de una neurona oculta o el parámetro asociado a la conexión entre ella y la neurona de salida son cero, ya que algunos de los parámetros restantes podrían tomar cualquier valor sin cambiar el MSE resultante.

Las razones expuestas dificultan enormemente el proceso de estimación de los parámetros, a tal punto, que este es un punto crítico en el desempeño del modelo final obtenido [4].

La dificultad del entrenamiento de modelos de redes neuronales tipo perceptrón multicapa es estudiada por Sánchez y Velásquez [12]. En dicho trabajo [12] se plantea que el comportamiento teórico de los algoritmos de entrenamiento es que el error de ajuste disminuya progresivamente a medida que se aumenta la complejidad de la red neuronal, a tal punto que sea arbitrariamente cercano a cero si la complejidad del modelo es la suficiente. Igualmente, en [12] se demuestra empíricamente que algoritmos clásicos para el entrenamiento de redes neuronales, tal como la regla delta generalizada y Rprop, no cumplen con dicho comportamiento teórico debido a que en la práctica no hay forma de garantizar la disminución del error de ajuste.

Desde lo práctico, el modelador se enfrenta a los problemas de sub-entrenamiento y sobre-entrenamiento. En el sub-entrenamiento, para una configuración determinada de un MLP, el algoritmo de aprendizaje queda atrapado en un punto de mínima local y el proceso de aprendizaje se detiene de forma temprana, impidiendo que se extraiga toda la información contenida en los datos; en este caso, el modelo tiene un ajuste pobre a la muestra de entrenamiento y sus pronósticos son de baja calidad. La detección del sub-entrenamiento resulta fácil en casos extremos, pero, resulta muy difícil en puntos relativamente cercanos al punto donde se inicia el sobre-entrenamiento. El uso de algunos criterios para abordar los problemas de sobre y sub-entrenamiento son abordados en [10] [13]. Soluciones típicas a este problema son el uso de reinicios múltiples en el caso de algoritmos de búsqueda local (como los de gradientes) [5] [10], o el uso de metaheuristicas (véase, por ejemplo, la referencia [14])

4. DEFINICIÓN DEL PROBLEMA Y SOLUCIÓN PROPUESTA

Es bien conocido que el problema de la estimación de los parámetros de un MLP es dependiente, entre otros, de los datos usados, la cantidad de pesos a ser estimados, el algoritmo de entrenamiento utilizado y los valores de los parámetros del algoritmo de entrenamiento. De ahí, que resulta prácticamente imposible realizar recomendaciones generales que permitan obtener siempre un modelo adecuadamente entrenado. Consecuentemente, una recomendación típica en la literatura más relevante es realizar el proceso de entrenamiento muchas veces (independientemente del algoritmo de optimización utilizado) variando diferentes aspectos, tal como el punto de arranque, y seleccionar el mejor modelo encontrado. Sin embargo, no existen garantías, teóricas o conceptuales, de que realmente se pueda obtener un modelo entrenado adecuadamente.

A continuación se describe un caso práctico de sub-entrenamiento. Ghiassi., Saidane. y Zimbra [13] utilizan un MLP como referente para demostrar las bondades de un nuevo tipo de red neuronal llamada DAN2. En uno de los casos de aplicación (Sección 3.8 de [13]), se realiza el pronóstico de las ventas industriales del papel para impresión y escritura (en miles de francos) entre enero de 1963 y diciembre de 1972, utilizando un modelo ARIMA, un MLP y DAN2: la serie es presentada en la Figura 2. Para ello, se utilizaron las primeras 100 observaciones en la estimación de los parámetros, mientras que las 20 finales se usaron para evaluar la capacidad predictiva de los modelos. En la Tabla 1, se reproduce el MSE reportado en [13] para el modelo ARIMA y el MLP (nombrado como ANN en la referencia citada). Se indica en [13], que los resultados reportados para el modelo ARIMA fueron calculados usando el procedimiento ARIMA implementado en el software SAS [15]. Para el MLP, se usó el software de minería de datos Clementine de SPSS [16]. La selección de SPSS [16] y SAS [15] está basada en que el modelo DAN2 [13] debe ser comparado contra modelos alternativos adecuadamente ajustados (entrenados) y especificados teniendo en cuenta las metodologías del estado del arte, ya que en caso contrario, los resultados serían cuestionables; esto es, se esperaría que los resultados presentados correspondían a los mejores modelos ARIMA y MLP que podían obtenerse para pronosticar dicha serie utilizando la información dada. Ya que ambas casas de software gozan de amplia reputación y popularidad entre la comunidad científica, es de esperarse que las herramientas computacionales utilizadas para especificar los modelos alternativos en [13] sean de alta calidad.

Los modelos finales reportados en [13] fueron los mejores modelos predictivos entre un grupo de modelos alternativos que diferían en su configuración y en los retardos utilizados. En [13] no se profundiza en la comparación entre el MLP y el ARIMA, ya que el objetivo de dicho trabajo es demostrar las bondades de DAN2 para la predicción de series de tiempo.

Durante el proceso de entrenamiento de un MLP, un modelo ARIMA podría ser usado para fijar un límite máximo del error en el entrenamiento. Esto se basa en dos hipótesis fundamentales:

1. Si la serie analizada es lineal, el MLP no puede capturar un comportamiento no lineal inexistente, de tal forma, que el MLP se ajustaría de forma similar a como lo hace un modelo ARIMA.

2. Si existe una componente determinística no lineal en los datos, un modelo ARIMA es incapaz de capturarla, por lo que un MLP debería tener un mejor comportamiento al ser un aproximador universal de funciones.

En los dos casos anteriores, el MLP debe tener, en el peor caso, un MSE de entrenamiento, a lo sumo, ligeramente superior al del modelo ARIMA (esto por cuestiones numéricas); y en la gran mayoría de las veces, debería ser inferior. En el caso de sobreajuste del MLP, su MSE debería ser mucho más bajo que el del modelo ARIMA, pero significativamente más alto durante el pronóstico.

Retornado a la Tabla 1, se observa que para el modelo ANN, los MSE de entrenamiento y validación son superiores a los calculados para el modelo ARIMA, lo que permite concluir que el modelo ANN no está correctamente entrenado (subentrenamiento), y que podrían obtenerse mejores parámetros. Para validar numéricamente la conclusión obtenida, se procedió a estimar un MLP como el presentado en la Figura 1, con las siguientes características:

- Se usaron los retardos 1, 7 y 12, al igual que en [13].

- Se utilizó una sola capa oculta. Se evaluaron diferentes cantidades de neuronas en la capa oculta encontrándose que su número óptimo es 5.

- Se usó la función tanh() para la activación de las neuronas de la capa oculta y la neurona de la capa de salida.

- Para entrenar cada configuración, que difiere en el número de capas ocultas, se utilizó el algoritmo GRG, con 100 reinicios aleatorios y un máximo de 100 iteraciones por cada optimización; las derivadas fueron estimadas numéricamente al interior del algoritmo.

Los resultados obtenidos corresponden al modelo MLP de la Tabla 1 (última línea), el cual presenta errores de entrenamiento y predicción inferiores al ANN, lo que permite validar numéricamente el razonamiento realizado. El pronóstico obtenido con el modelo MLP es presentado en la Figura 2.

Como un segundo ejemplo, se ilustra la detección de un caso de sobre-entrenamiento. A continuación se analizan a continuación los resultados reportados en [13] para la serie de conformada por la cantidad de usuarios registrados por minuto en un servidor de internet durante un lapso de 100 minutos. Para esta segunda serie, se tomaron las primeras 80 observaciones para la estimación de los modelos, y las 20 restantes para su validación. En la Tabla 2 se presentan los resultados reportados en [13]; en este caso, el MSE de entrenamiento para la red neuronal artificial es significativamente inferior (72%) en relación al obtenido para el modelo ARIMA; no obstante, el MSE de validación para la red neuronal es el 114% del valor reportado para el modelo ARIMA. Si la red neuronal artificial realmente captura mejor la dinámica de la serie, sus resultados deberían ser inferiores tanto en entrenamiento como en validación; ello permite concluir que es muy posible que se presente un fenómeno de sobre-entrenamiento. Ello implicaría que es necesario ajustar nuevamente el modelo.

Tabla 1. *Estadísticos de ajuste para el primer ejemplo.*

Modelo	Retardos	MSE Entrenamiento	MSE Validación
ARIMA	1, 12, 13	1715.5	4791.7
ANN	1, 7, 12	1951.5	5874.8
MLP	1,7, 12	1651.0	4650.0

Tabla 2. *Estadísticos de ajuste para el segundo ejemplo.*

Modelo	Retardos	MSE Entrenamiento	MSE Validación
ARIMA	1,2,3,4	9.76	8.11
ANN	1,2,3,4	7.00	9.25

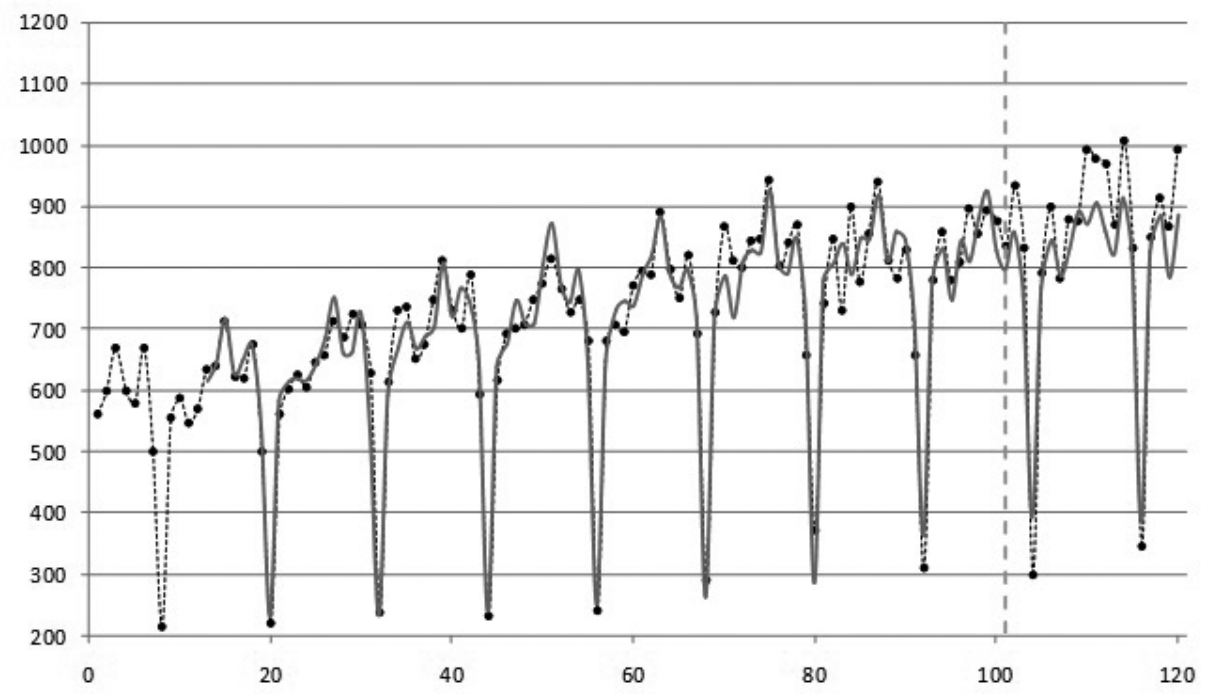

Figura 2. *Serie real (línea punteada negra) y pronóstico (línea gris continua) obtenido usando un perceptrón multicapa.*

5. DISCUSIÓN

Los dos ejemplos anteriores demuestran claramente la problemática del entrenamiento de un MLP; este no es un problema trivial, y la solución clásica de "realice muchas optimizaciones y conserve la mejor" no da garantía sobre la calidad del modelo obtenido; este no es un problema propio de la falta de experiencia por parte del pronosticador, sino de la dificultad misma del proceso de optimización. Este argumento se refuerza al considerar que los resultados reportados en [13] fueron obtenidos con herramientas comerciales, en las cuales se implementan algoritmos del estado del arte para la optimización de los modelos considerados.

De esta forma, surge la necesidad de incorporar límites en los errores de entrenamiento y validación durante la especificación de modelos de pronóstico basados en redes neuronales artificiales, con el fin de detectar modelos mal especificados debido a un entrenamiento inadecuado.

Una dificultad inherente a la propuesta realizada es la construcción de un benchmark contra el cual se debe realizar la comparación. Este no está limitado necesariamente a los modelos ARIMA, sino que otras alternativas podrían incorporarse, como por ejemplo, los modelos de suavizado exponencial; véase por ejemplo [7].

Una segunda dificultad es la estimación de los modelos usados como benchmark, debido a que el dominio de metodologías estadísticas de predicción es poco común en la comunidad de usuarios de redes neuronales. Para solucionar este problema, es posible utilizar el paquete forecast [17] implementado en el lenguaje de programación R para la computación estadística (www.r-project.org), ya que incorpora métodos automáticos para la selección de modelos ARIMA y de suavizado exponencial.

6. CONCLUSIONES

En este artículo se ha demostrado la importancia de la comparación de los errores de entrenamiento y predicción en modelos de redes neuronales artificiales. El principal aporte de este trabajo, es ejemplificar como se pueden utilizar los errores calculados para las muestras de entrenamiento y pronóstico de modelos lineales bien establecidos como criterios para detectar problemas de sobre-ajuste y sub-entrenamiento en modelos de redes neuronales artificiales.

La principal recomendación dada al lector, es que antes de realizar la especificación de un modelo de redes neuronales, se especifiquen modelos más tradicionales con el fin de establecer, a priori, límites en los errores, con el fin de tener un referente respecto a los resultados obtenidos con una red neuronal artificial.

7. REFERENCIAS

[1] D. van Djck Smooth Transition Models: Extensions and Outlier Robust Inference [PhD thesis]. Erasmus University - Rotterdam. 1999.

[2] M.P. Clements, P.H. Frances and N.R. Swanson. "Forecasting economic and financial time-series with non-linear models". International Journal of Forecasting, vol. 20, 2004, pp.168-183

[3] G. Zhang, B. Patuwo and M. Hu. "Forecasting with artificial neural networks: the state of the art", International Journal of Forecasting, vol. 14, 1998, pp. 35–62.

[4] I. Kaastra, and M. Boyd. "Designing a neural network for forecasting financial and economic series", Neurocomputing, vol. 10, 1996, pp. 215–236.

[5] Y. LeCun, L. Bottou, G.B. Orr and K.-R. Muller. "Efficient Backprop. En Neural Networks - Tricks of the Trade". Springer Lecture Notes in Computer Sciences 1524, 1988, pp. 5-50.

[6] U. Anders and O. Korn. "Model selection in neural networks", Neural Networks, vol. 12, 1999, pp. 309-323.

[7] S.G. Makridakis, S.C. Wheelwright and R.J. Hyndman. Forecasting: Methods and applications. 3rd edition. New York. John Wiley & Sons. 1998.

[8] W. Sarle. "Neural networks & and statistical models", Proceedings of the Nineteenth Annual SAS Users Group International Conference. April, 1994 The 19th Annual SAS Users Group Int. Conference. Cary, NC: The SAS Institute, 1994, pp. 1538-1550.

[9] G. Cybenko. "Approximation by superpositions of a sigmoidal function", Mathematics of Control: Signals and Systems, vol. 2, 1989, pp. 202–314.

[10] T. Masters, T. Practical Neural Network Recipes in C++ (1ra. Ed). San Diego, CA, USA Academic Press Professional, Inc. 1993.

[11] T. Masters. Neural, Novel and Hybrid Algorithms for Time Series Prediction (1ra ed.). New York, NY, USA. John Wiley and Sons. 1995.

[12] P. Sánchez y J.D. Velásquez. "El rol del algoritmo de entrenamiento en la selección de modelos de redes neuronales". Revista U.D.C.A. Actualidad & Divulgación Científica, vol. 14, no. 1, 2011, pp. 149-156.

[13] M. Ghiassi, H. Saidane and D.K. Zimbra. A dynamic artificial neural network model for forecasting time series events. International Journal of Forecasting, vol. 21, 2005. pp. 341-362.

[14] M. Pant, R. Thangaraj, R. y A. Abraham, "DE-PSO: a new hybrid meta-heuristic for solving global optimization problems". New Mathematics and Natural Computation, vol. 7, no. 3, 2011, pp. 363-381.

[15] SAS. The SAS System 8e for MS-Windows, Release 8.02, SAS Institute Inc. 2001

[16] SPSS. Clementine Data Mining System, Version 5, SPSS, Inc. 1998.

[17] R.J. Hyndman and Y. Khandakar. "Automatic Time Series Forecasting: The forecast Package for R". Journal of statistical software, vol. 27, No. 3, July 2008.