Hurst Parameter Estimation Using Artificial Neural Networks

S. Ledesma-Orozco*[1], J. Ruiz-Pinales[2], G. García-Hernández[3], G. Cerda-Villafaña[4], D. Hernández-Fusilier[5]

[1,2,3,4,5] Universidad de Guanajuato
Comunidad de Palo Blanco
C.P.36885 Salamanca, Guanajuato, Mexico
*selo@ugto.mx

ABSTRACT
The Hurst parameter captures the amount of long-range dependence (LRD) in a time series. There are several methods to estimate the Hurst parameter, being the most popular: the variance-time plot, the R/S plot, the periodogram, and Whittle's estimator. The first three are graphical methods, and the estimation accuracy depends on how the plot is interpreted and calculated. In contrast, Whittle's estimator is based on a maximum likelihood technique and does not depend on a graph reading; however, it is computationally expensive. A new method to estimate the Hurst parameter is proposed. This new method is based on an artificial neural network. Experimental results show that this method outperforms traditional approaches, and can be used on applications where a fast and accurate estimate of the Hurst parameter is required, i.e., computer network traffic control. Additionally, the Hurst parameter was computed on series of different length using several methods. The simulation results show that the proposed method is at least ten times faster than traditional methods.

Keywords: Parameter estimation, time series, network traffic analysis, neural network.

RESUMEN
El parámetro de Hurst captura la cantidad de dependencia de rango amplio (LRD) en las series de tiempo. Hay varios métodos para estimar el parámetro de Hurst, siendo los más populares: la gráfica de varianza contra tiempo, la gráfica R/S, el periodograma, y el estimador de Whittle. Los tres primeros son métodos gráficos, y la precisión de la estimación depende de cómo se interprete y calcule la gráfica. Por otro lado, el estimador de Whittle se basa en una técnica de máxima probabilidad y no depende de una lectura gráfica; sin embargo, éste requiere una gran demanda computacional para su cálculo. Se propone un nuevo método para estimar el parámetro de Hurst. Este nuevo método está basado en una red neuronal artificial. Los resultados experimentales muestran que este método supera a los métodos tradicionales, y que puede ser usado en aplicaciones que requieran una estimación precisa y rápida del parámetro de Hurst, por ejemplo en control de tráfico en redes de computadoras. Adicionalmente, el parámetro de Hurst se calculó en series de diferentes tamaños utilizando varios métodos. Los resultados de la simulación muestran que el método propuesto es por lo menos diez veces más rápido que los métodos tradicionales.

1. Introduction

Long-range dependence (LRD) captures the persistence phenomenon observed in many empirical time series that manifests itself in clusters of consecutive large (or small) values, see [1, 2]. A typical hypothesis of time series analysis is that observations separated by a great time span are almost independent. Nevertheless, for LRD, these observations are not independent [3, 4, 5, 6]. The Hurst parameter, symbolized by H, is the measure of LRD. The classic LRD analysis involves the estimation of the Hurst parameter from data series. Several traditional methods can be used to estimate *H* [7]; however, some of them may be computationally expensive, require a graph reading or are affected by irregularities in the data [8]. Since artificial neural networks (ANNs) can learn and adapt to a great diversity of data without being affected by noise and errors [9], it will be shown that they are an excellent option to estimate *H*.

This paper is organized as follows. In Section 2, we present background information about short-range dependence and long-range dependence. In Section 3, we present a brief explanation on

artificial neural networks focusing mainly on mapping and classification problems. In Section 4, we develop a method to reduce the computation complexity to estimate the Hurst parameter in data series with LRD. It is shown that this method can also be used to classify data series according to their amount of LRD. In Section 5, we perform a statistical analysis using existing methods to estimate the Hurst parameter, and the method developed in this paper. Finally, Section 6 presents some conclusions and key points.

2. Long-range dependence

Consider a second-order stationary process $X = \{X_n\}$, where all $\{X_n\}$ have a common mean μ and a common finite variance σ. Let $r(k)$ denote the autocorrelation function defined by

$$r(k) = E[X_n X_{n+k}] \quad (1)$$

Thus, short-range dependence (abbreviated SRD) can be observed in processes characterized by an autocorrelation function that decreases exponentially fast and have *H*=0.5. In contrast, LRD processes have an autocorrelation function that slowly decreases, that is

$$r(k) \sim k^{-(2-2H)} L_1(k) \text{ as } k \to \infty \text{ with } 1/2 < H < 1 \quad (2)$$

where the symbol ~ denotes that the expressions on both sides are asymptotically proportional to each other, see [1, 2], and $L_1(\cdot)$ is slowly varying at infinity, specifically

$$\lim_{y\to\infty} \frac{L_1(ay)}{L_1(y)} = 1 \text{ for all } a > 0. \quad (3)$$

3. Artificial neural networks

An ANN is a computational method motivated by biological models. ANNs attempt to mimic the fundamental operation of the human brain and can be used to solve a broad variety of problems [10]. One of the most important features of ANNs is that they can discover hidden patterns from data sets [11], and solve complex problems when there is not a mathematical model (or when the model is not suitable for the case at hand). Furthermore, ANNs are commonly immune to noise and irregularities present in the data [12, 13]. ANN learning is typically based on two data sets: the training set and the validation set. The training set is used on a new artificial neural network, as its name indicates, for training. The validation set is used after the neural network has been trained to assess its performance. The validation set is, in most cases, similar to the training set but not equal [14, 15].

3.1 Data mapping

In artificial intelligence, a desired output is commonly known as the target. For the specific case of ANNs, the target is used for network training [9]. Generally, ANNs can map a given input to a desired output; when an ANN is used for this purpose, the ANN is typically called a mapping ANN. For this kind of setup, the network is trained by applying the desired input to the ANN, and monitoring the actual ANN output. The difference between the actual ANN output and the desired output is normally used to manage the learning process. Thus, during training, the learning algorithm attempts to reduce the error measured between the actual network output and the target for each case in the training set [9, 11]. The training process may be time consuming, but after this process has been successfully completed, an ANN can quickly calculate its output once the input data has been applied to the network input.

3.2 Data classification

Data classification or just classification is the process of identifying an object from a set of possible outcomes [9, 12]. An ANN can be trained to identify and classify any kind of objects. These objects can be numbers, images, sounds, signals, etc. An ANN used for this purpose is also known as a classifier.

4. Proposed method

Current research on LRD includes the accurate estimation of the Hurst parameter. For instance, in [16], a new fractional Fourier transform (FrFT)-based estimation method to estimate the Hurst parameter is presented. This paper

proposes the use of ANNs to analyze data series that exhibit LRD.

Consider a second-order stationary process $X = \{X_n\}$ as in Section 2, but now suppose that all $\{X_n\}$ have a common mean $\mu = 0$ and a common finite variance $\sigma^2 = 1$; observe that for those processes that do not have $\mu = 0$ and $\sigma^2 = 1$, there is always a linear transformation that can be applied to the data so that these requirements can be easily met. For discrete-time processes, the discrete-time Fourier transform can be used [17, 18, 19], and the power spectral density of X is defined, for $\sigma^2 = 1$, as

$$f(\omega) = \sum_{k=-\infty}^{\infty} r(k)\, e^{-j\omega k} \qquad (4)$$

Hence, the slow decrease of $r(k)$ of Equation 2 can be similarly expressed in terms of the behavior of the power spectral density $f(\omega)$ as

$$f(\omega) \sim \omega^{-(2H-1)} L_2(\omega) \text{ as } \omega \to 0 \text{ with } 1/2 < H < 1 \qquad (5)$$

where the symbol ~ has the same meaning as in Equation 2, and is slowly varying at the origin, that is

$$\lim_{y \to 0} \frac{L_2(ay)}{L_2(y)} = 1 \text{ for all } a > 0. \qquad (6)$$

Equation 5 indicates that the power spectral density, $f(\omega)$, depends on the Hurst parameter, meaning that different values of H will produce different power spectral density functions, see [18, 19]. In consequence, Equation 5 may be used to estimate H through the power spectrum components of the data series, i.e., by the amount and distribution of these components. This is the foundation of the method proposed, and this thought will be now expanded and reviewed in detail. As far as we know, this is the first time that ANNs are used to analyze LRD series.

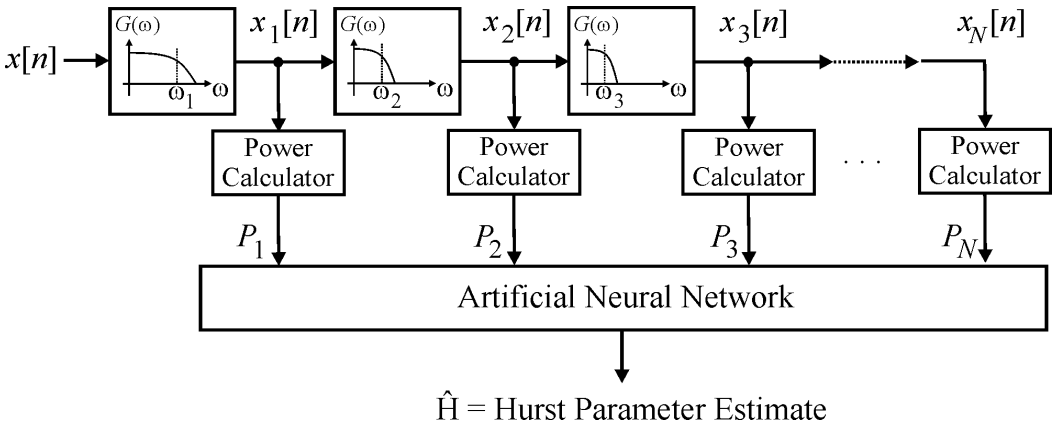


Figure 1. Neural network method to estimate the Hurst parameter.

Consider now Figure 1 and suppose that an LRD data series $x[n]$ is applied to the input of the first low-pass filter as shown. This low-pass filter has a cut-off frequency ω_1, and the output sequence, $x_1[n]$, is applied to the second low-pass filter as shown in the same figure. Note that the cut-off frequency of the second filter, ω_2, is chosen so that $\omega_1 > \omega_2$. Then, $x_2[n]$ is applied to the input of the next low-pass filter and the process continues N times as shown. This special configuration will produce N new sequences, $x_1[n]$, $x_2[n]$, ..., $x_N[n]$. Observe that each of these sequences has a different power spectral density, and that these differences in the spectral density provide information about the value of H. Observe also, that we are not concerned much about $x_1[n]$, $x_2[n]$, ..., $x_N[n]$, neither of their power spectral densities, but we care more about the power of each of them, namely P_1, P_2, ..., P_N. In practice, the value of N may be something around 10 as the power of sequence-*i,* namely P_i, goes to zero quickly when i increases.

As it is well known, an ANN can easily learn and find patterns from a data set [9, 11, 13]. This feature of ANNs can be applied to estimate the Hurst parameter, in particular, the values of P_1, P_2, ..., P_N (produced by the low-pass filters and power calculators of Figure 1) can be used to create a suitable training set. For this particular case, the required ANN must have N inputs, and one single output (the estimate of H). Specifically, P_1, P_2, ..., P_N can be applied to the ANN inputs, and the learning algorithm (used for training) must adjust the ANN's weights to produce the specific target, i.e., the actual Hurst parameter of the series. If the training set is built with enough training cases, and these cases represent appropriately the majority of all LRD series, it will be possible to estimate the Hurst parameter using an ANN.

Before concentrating on the implementation of the algorithm described in Figure 1, note that in order to get appropriate frequency samples, the cut-off frequency of each filter in this figure must be chosen so that

$$\omega_1 > \omega_2 > \omega_3 > ... > \omega_N \qquad (7)$$

Even though the algorithm of Figure 1 can be used to estimate H, it is important to note that several simplifications can be made on the implementation to reduce its computation complexity. First consider the low-pass filters of Figure 1, there are several factors such as the order and approximation of the filter that may affect the overall performance. For this specific application and because of the use of ANNs, it will be seen that the specific shape of the filter is not important, that is, the filter may have a rough low-pass filter approximation without affecting the accuracy of the estimate of *H*. One way to build such a filter is using a simple moving average system with impulse response:

$$h[n] = \begin{cases} \frac{1}{2}, & 0 \le n \le 1 \\ 0, & otherwise \end{cases} = \frac{1}{2}\left[\delta[n] + \delta[n-1]\right]$$

(8)

It can be easily shown that the frequency response of the moving average system, described in Equation 8, is

$$H(e^{j\omega}) = e^{-j\omega/2}\cos\left(\frac{\omega}{2}\right) \qquad (9)$$

The magnitude of $H(e^{j\omega})$ is plotted in Figure 2, note that it is periodic as required by the frequency response of a discrete-time system. Also note that the magnitude of $H(e^{j\omega})$ falls off at high frequencies. This attenuation of the high frequencies suggests that the system can be used as a rough approximation to a low-pass filter, see [20].

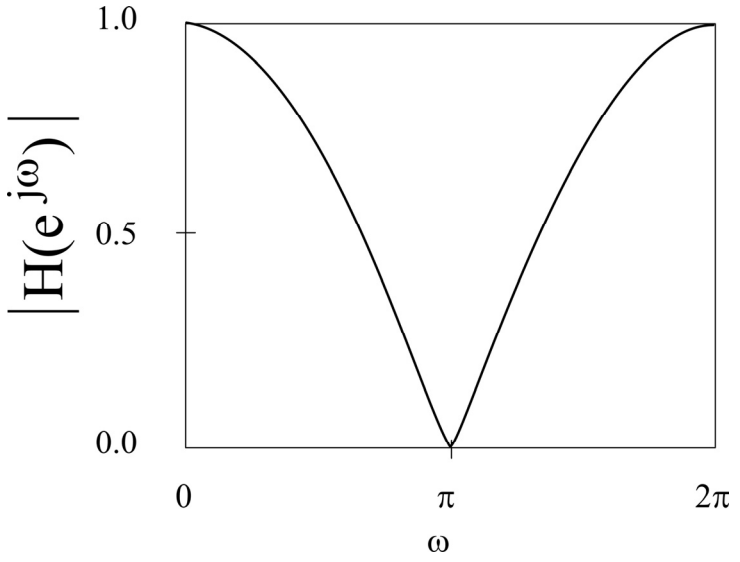


Figure 2. Magnitude of $H(e^{j\omega})$.

Another simplification that can be done in the implementation of the proposed method is to reduce the sequence length each time they pass through each of the low-pass filters by using a downsampler. In general, this can improve the overall performance as the number of operations is performed with shorter sequences than the original. After the sequences $x_2[n]$ …, $x_N[n]$ have passed through the filters, and part of their frequency content has been removed, the sequence length can be reduced with little aliasing. As it is well known, the operation of reducing the sampling rate (including any prefiltering) is typically called downsampling [20]. Figure 3 shows one simple way to implement the moving average system and the downsampler just described for a sampling rate reduction of 2.

A careful inspection of Figure 3 will reveal that it is possible to make one final optimization on the implementation of these low-pass filters. Specifically, it is not necessary to compute all the values of $y[n]$ as some of these values will be discarded by the downsampler. In practice, all these optimizations account for great part of the computation, and they will considerably reduce the computation complexity as it will be seen in the next section.

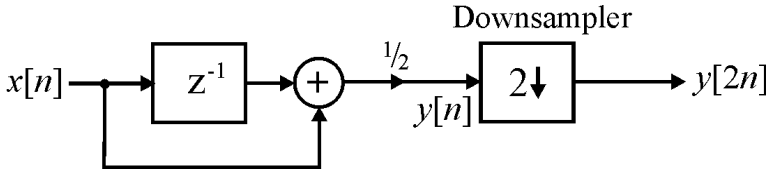


Figure 3. Rough approximation to a low-pass filter.

5. Experimental results

We proceed to evaluate the computation complexity, accuracy and immunity to noise of the proposed and classic methods to estimate *H*.

5.1 Computation complexity

One of the simplest methods that exhibit LRD is the Fractional Gaussian Noise (FGN). The approach presented in [17] for generating FGN is based on the Fast Fourier Transform (FFT). The main difficulty with this approach lies in accurately computing the power spectrum of FGN in a timely manner. The second-order approximation (SOA) presented in [18] estimates the power spectrum of FGN efficiently and accurately.

First, 10,000 series of length 8,192 using the SOA method, see [18], were created using uniformly distributed values of H ($0.5 < H < 1.0$). The values of P_1, P_2, ..., P_{10} were computed from these series, and a suitable training set was built. Following the same procedure, the validation set was built as well. After that, a multilayer feed-forward network (using the logistic activation function) with ten inputs, four hidden neurons and one output was built and trained (note that appropriate input and output scaling was applied to the data, and that the number of hidden neurons was adjusted to get a comparable mean-squared error during training and validation). Then, 100 series for several values of the Hurst parameter ($H = 0.55, 0.65, ..., 0.95$) were created as before; the Hurst parameter of these series was computed for several methods and the computation time was measured. Table 1 shows the average running time in microseconds to estimate the Hurst parameter using a Intel (R) Core (TM) i7 CPU 940 @ 2.93 GHz for several of the methods. As it can be seen from this table, Whittle's estimator is the slowest method, and the method proposed (Neural Network) is the fastest. Same results can be observed from Table 2 that shows the relative running time to estimate *H*. Specifically, it can be observed that Whittle's estimator is around 700 times slower than the proposed method, while the Variance-Time plot and the R/S-statistics are around 10 times slower than the proposed method.

	H = 0.55	H = 0.65	H = 0.75	H = 0.85	H = 0.95
Variance-Time Plot	1.22	1.23	1.22	1.22	1.22
R/S - statistics	0.85	0.86	0.86	0.86	0.86
Whittle's Estimator	68.11	61.89	67.80	62.90	67.78
Neural Network	0.09	0.09	0.09	0.09	0.09

Table 1. Running time in milliseconds to estimate the Hurst parameter.

	H = 0.55	H = 0.65	H = 0.75	H = 0.85	H = 0.95
Variance-Time Plot	13.56	13.67	13.56	13.56	13.56
R/S - statistics	9.44	9.56	9.56	9.56	9.56
Whittle's Estimator	756.78	687.67	753.33	698.89	753.14
Neural Network	1.00	1.00	1.00	1.00	1.00

Table 2. Relative running time to estimate the Hurst parameter.

For a given length, one hundred LRD series were created using a Hurst parameter of 0.5. The average running time as a function on the series length is presented in Figures 4 and 5. Figure 4 shows the results for the Variance-Time plot, the R/S statistics and the ANN. Figure 5 shows the results for Whittle's estimator and the Variance-Time plot. Because Whittle's estimator is based on a minimization algorithm using the power spectrum of FGN, it is extremely slow. On the other hand, the Variance-Time plot and R/S statistics are quick methods to estimate H. From Figure 4, it can be seen that the neural network can quickly estimate the Hurst parameter for short and long data series.

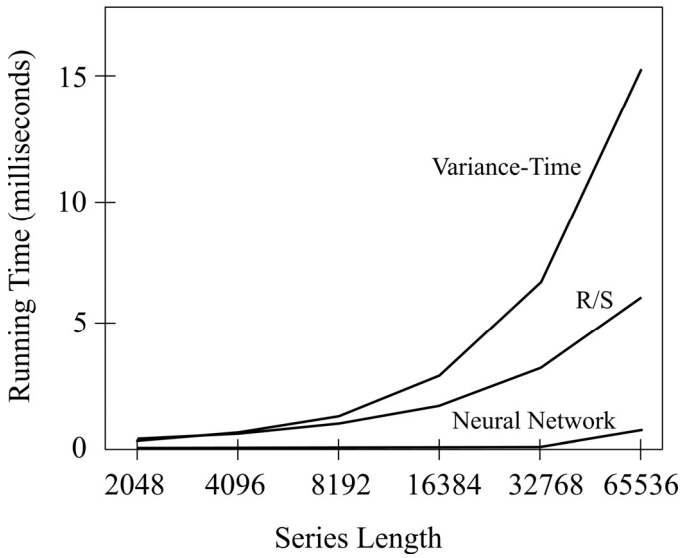


Figure 4. Running time to estimate the Hurst Parameter.

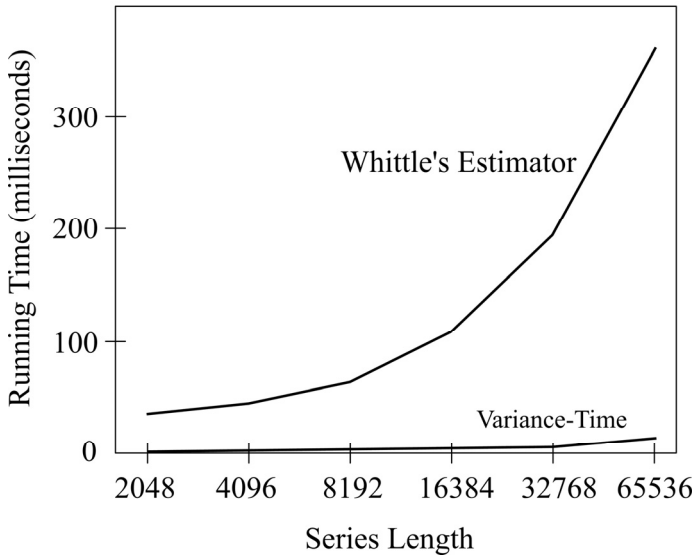


Figure 5. Running time to estimate the Hurst Parameter.

5.2 Estimation accuracy

The areas of application of LRD series includes: Agronomy, Astronomy, Chemistry, Economics, Engineering, Environmental sciences, Hydrology, Math, Physics and Statistics. Thus, current research focuses on the accurate estimation of the Hurst parameter, see [22, 23, 24].

To compare the estimation accuracy of the proposed method with other traditional methods, one hundred LRD series of length 8,192 were created (using uniformly distributed values for H in the range from 0.5 to 0.99). We proceeded to estimate the Hurst parameter using some of the classic methods as well as the method proposed. Finally, the mean-squared error (MSE) was computed and plotted as shown in Figures 6, 7 and 8. As it can be seen from Figure 6, the proposed method outperformed the variance-time plot (particularly for $H > 0.7$). For the case of the R/S statistics, it can be observed from Figure 7, that this method has little error for *H* close to 0.8, however, the neural network has generally a smaller MSE than the R/S statistics for all values of *H*. Figure 8 shows the respective plot to compare Whittle's estimator with the proposed method; a quick look to the plot allows observing that Whittle's estimator has a bigger error than the proposed method.

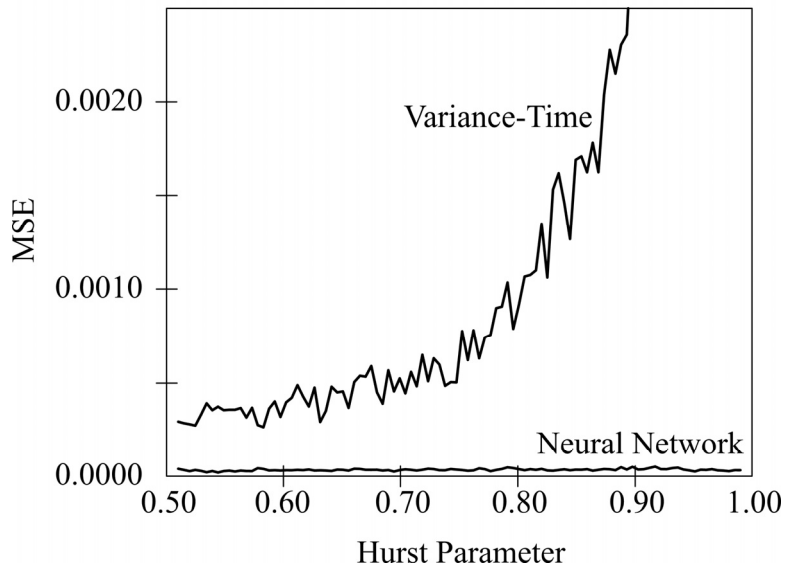


Figure 6. Mean-squared error for the R/S-statistics and the proposed method.

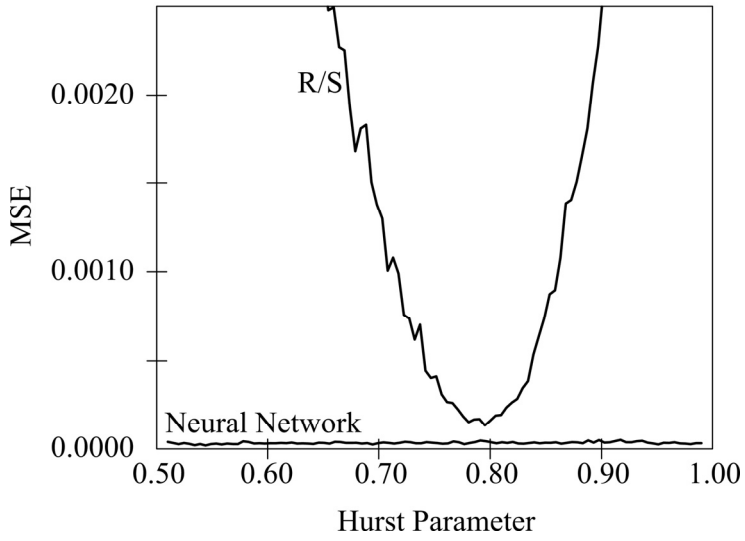


Figure 7. Mean-squared error for the R/S-statistics and the proposed method.

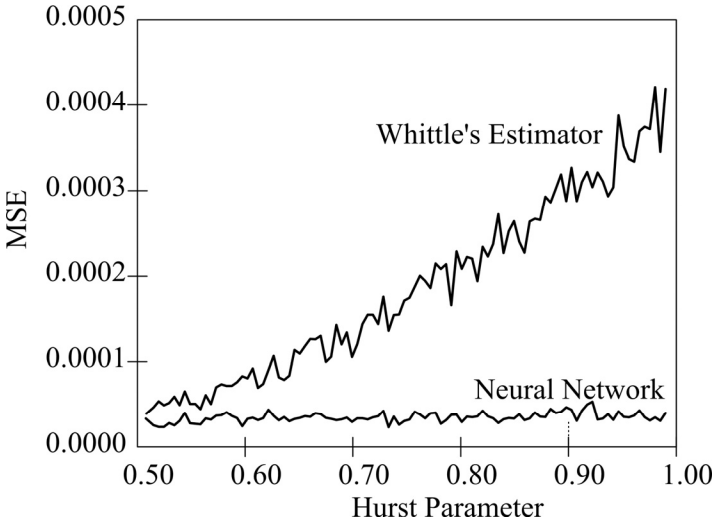


Figure 8. Mean-squared error for Whittle's estimator and the proposed method.

5.3 Data series classification

An ANN used for classification typically has the same number of outputs as classification categories. In the field of artificial intelligence, a confusion matrix is a visualization tool typically used in supervised learning [25]. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly misidentifying one as another) [26].

For the purpose of classification, another training set with five classes and 1000 training cases was built. In this case, each class was made with series with values of H in a precise range, specifically, class 1 has values of H around 0.55, class 2 has values of H around 0.65, and so on. The training set had 200 training cases for each class. After the training set was built using the of P_1, P_2, ..., P_{10}, the ANN was trained. Finally, we proceeded to classify 1000 LRD series; but to make the test more difficult, 30% of the data in the series were replaced with zeros before computing P_1, P_2, ..., P_{10}. Table 3 shows the classification results for the variance-time plot. As it can be seen, this method made 273 errors, and these errors were more common for bigger values of H (classes 4 and 5). Table 4 shows the confusion matrix for the R/S-statistics, as it can be seen from this table, this method did not make any errors for class 3; however, it made a total of 474 errors. Table 5 shows the classification results for Whittle's estimator, with only 210 errors made by this method. As it can been concluded, Whittle's estimator is the most accurate method when compared with the variance-time plot or the R/S statistics. Finally, Table 6 shows the results for the proposed method; as it can be seen, the ANN did not make any error for classes 1, 2 and 3; the total number of errors was only 4.

	Class 1	Class 2	Class 3	Class 4	Class 5	Reject
Class 1	198	0	0	0	0	2*
Class 2	12*	188	0	0	0	0
Class 3	0	21*	178	1*	0	0
Class 4	0	0	68*	132	0	0
Class 5	0	0	0	169*	31	0

Table 3. Confusion matrix for the variance-time plot.

	Class 1	Class 2	Class 3	Class 4	Class 5	Reject
Class 1	1	199*	0	0	0	0
Class 2	0	163	37*	0	0	0
Class 3	0	0	200	0	0	0
Class 4	0	0	38*	162	0	0
Class 5	0	0	0	200*	0	0

Table 4. Confusion matrix for the R/S-statistics.

	Class 1	Class 2	Class 3	Class 4	Class 5	Reject
Class 1	200	0	0	0	0	0
Class 2	0	200	0	0	0	0
Class 3	0	0	200	0	0	0
Class 4	0	0	23*	177	0	0
Class 5	0	0	0	187*	13	0

Table 5. Confusion matrix for Whittle's estimator.

	Class 1	Class 2	Class 3	Class 4	Class 5	Reject
Class 1	200	0	0	0	0	0
Class 2	0	200	0	0	0	0
Class 3	0	0	200	0	0	0
Class 4	0	0	1*	199	0	0
Class 5	0	0	0	3*	197	0

Table 6. Confusion matrix for the proposed method (Neural Network).

3. Conclusions

A new method to estimate the Hurst parameter was proposed. This new method is based on an ANN. A set of simple and fast low-pass filters was used to extract the information from the LRD data series. This set of low-pass filters produced the sequences: $x_1[n], x_2[n], \ldots, x_N[n]$. The power of these sequences was used to build a training set. After a suitable ANN was trained, several experiments were performed to validate our approach. The proposed method turned out to be the fastest and most accurate when compared to traditional methods to estimate the Hurst parameter. Finally, some classification experiments were performed, the results show that the proposed method made only 4 misclassifications, while the traditional methods made from 210 to 473 errors.

LRD data series were found in water reservoir analysis several decades ago [6]; nowadays, LRD data series are present in computer network traffic, see [27].

References

[1] J. Beran, Statistical methods for data with long-range dependence. Statistical Science, Volume 7, Number 4, pp. 404-427, (1992).

[2] A. Erramilli, O, Narayan and W. Willinger, Experimental queueing analysis with long-range dependent packet traffic. IEEE/ACM Transactions on Networking, Volume 4, Number 29, pp. 209-223, (1996).

[3] R. Fox and M. S. Taqqu, Large-sample properties of parameter estimates for strongly dependent stationary Gaussian time series. The Annals of Statistics, Volume 4, Number 2, pp. 517-532, (1985).

[4] T. Karagiannis, M. Molle, and M. Faloutsos, Long-range dependence: ten years of Internet traffic modeling. IEEE Internet Computing, Volume 8, Number 5, pp. 57-64, (2004).

[5] M. Krunz and I. Matta, Analytical investigation of the bias effect in variance-type estimators for inference of long-range dependence. Computer Networks, Volume 40, Number 3, pp. 445-458, (2002).

[6] B. B. Mandelbrot and J. R. Wallis, Some long-run properties of geophysical records. Fractals in the earth sciences, New York: Plenum Press, (1969).

[7] O. Rose, Estimation of the Hurst Parameter of Long-Range Dependent Time Series. Report 137, Institute of Computer Science, University of Wurzburg, (1996).

[8] S. Stoev, M. S. Taqqu, C. Park, G. Michailidis and J. S. Marron, LASS: a tool for the local analysis of self-similarity. Computational Statistics & Data Analysis, Volume 50, Number 9, pp. 2447-2471, (2006).

[9] T. Masters, Practical Neural Network Recipes in C++. Preparing Input Data (C-16), Academic Press, Inc., pp. 253-270, (1993).

[10] S. J. Russel and P. Norvig, Artificial Intelligence: A Modern Approach. Prentice-Hall of India, Second Edition. Statistical Learning Methods (C-20), pp. 712-762, (2006).

[11] T. Masters, Neural, Novel & Hybrid Algorithms for Time Series Prediction. Neural Network Tools (C-10), John Wiley & Sons Inc., pp. 367-374, (1995).

[12] T. Masters, Signal and Image Processing With Neural Networks. Data Preparation for Neural Networks (C-3), John Wiley & Sons Inc., pp. 61-80, (1994).

[13] T. Masters, Advanced Algorithms for Neural Networks. Assessing Generalization Ability (C-9), John Wiley & Sons Inc., pp. 335-380, (1995).

[14] R. D. Reed and R. J. Marks II, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks. Factors Influencing Generalization (C-14), The MIT Press, pp. 239-256, (1999).

[15] http://en.wikipedia.org/wiki/Neural_Lab

[16] Y. Chen, R. Sun and A. Zhou, An improved Hurst parameter estimator based on fractional Fourier transform, Volume 43, Numbers 3-4, Telecommunication Systems, pp. 197-206, (2010).

[17] V. Paxson, Fast, approximate synthesis of fractional Gaussian noise for generating self-similar network traffic, Computer Communications Review, Volume 27, pp. 5–18, (1997).

[18] S. Ledesma, D. Liu, and D. Hernandez, Two approximation methods to synthesize the power spectrum of fractional Gaussian noise. Computational Statistics and Data Analysis, Volume 52, Number 2, pp. 1047-1062, (2007).

[19] S. Ledesma and D. Liu, Synthesis of fractional Gaussian noise using linear approximation for generating self-similar network traffic. ACM Computer Communication Review, Volume 30, Number 2, pp. 4-17, (2000).

[20] A. V. Oppenheim and R. W. Schafer, Discrete-time signal processing. Sampling of Continuos-Time Signals (C-3), Upper Saddle River, N.J., Prentice Hall, pp. 80-148, (1989).

[21] J. Purczynski and P. Wlodarski, On fast generation of fractional Gaussian noise. Computational Statistics and Data Analysis, Volume 50, Number 10, pp. 2537-2551, (2005).

[22] M. Kulikovs, S. Sharkovsky and E. Petersons, Comparative studies of methods for accurate Hurst parameter estimation, Volume 7, Number 103, Electronics and Electrical Engineering, pp. 113-116, (2010).

[23] J. Park and C. Park, Robust estimation of the Hurst parameter and selection of an onset scaling, Volume 19, Statistica Sinica, pp. 1531-1555, (2009).

[24] N. Wang, Y. Li and H. Zhang, Hurst exponent estimation based on moving average method, Volume 72, Advances in wireless networks and information systems, Lecture Notes in Electrical Engineering, pp. 137-142, (2010).

[25] M. T. Jones, AI Application Programming. Introduction to Neural Networks and the Backpropagation Algorithm (C-8), Charles River Media, 2nd edition, pp. 165-204, (2005).

[26] http://en.wikipedia.org/wiki/Confusion_matrix

[27] J. C. Ramirez and R. D. Torres, Local and Cumulative Analysis of Self-similar Traffic Communications and Computers, Volume 37, Number 1, pp. 27-33, (2006).

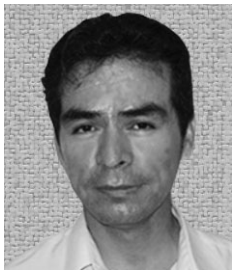
Acknowledgments

This work was sponsored by CONACYT and PROMEP.

Authors´ Biographies

Sergio LEDESMA-OROZCO

Dr. Ledesma-Orozco got his M.S. from Universidad de Guanajuato while working on the setup of Internet in Mexico. In 2001, he got his Ph.D. from Stevens Institute of Technology in Hoboken, New Jersey. After graduating, he worked for Barclays Bank as part of the IT-HR group. He has worked as a software engineer for several years, and he is the creator of the software Neural Lab and Wintempla. Neural Lab offers a graphical interface to create and simulate artificial neural networks. Neural Lab is free, and the latest version can be downloaded from Wikipedia. He is the author of the software Wintempla which provides a thin layer of encapsulation to ease program design and implementation for business and research. Currently, he is a research professor at Universidad de Guanajuato (University of Guanajuato) in Mexico. His areas of interests are artificial intelligence and software engineering.



José RUIZ-PINALES

Dr. Ruiz-Pinales received the BA degree in electronics and communications engineering and the MSE degree in electrical engineering from Universidad de Guanajuato in 1993 and 1996. He received the PhD degree in computer science from Ecole Nationale Supérieure des Télécommunications de Paris (Higher Education National School of Telecommunications of Paris) in 2001. He joined the department of Electronics Engineering of Universidad de Guanajuato as a professor in 2001. His research interests are in computer vision and artificial intelligence including face recognition, handwriting recognition, neural network models, support vector machines, models of the human visual system, instrumentation, communications and electronics. He has coauthored more than 32 research papers.

Ma. de Guadalupe GARCÍA-HERNÁNDEZ

Dr. García-Hernández is a researcher and full-time professor in the Electronics Department at Universidad de Guanajuato. She graduated in chemical engineering from Universidad de Guanajuato in 1985 and obtained her master's degree in mechanical engineer from the same university in 1992. Dr. García-Hernández got her doctorate degree in form recognition and artificial intelligence from Universitat Politècnica de València (Polytechnic University of Valencia), Spain. She has published papers in international and national scientific events, and in indexed and national journals. She has been a member of the Spanish Association for Artificial Intelligence since January, 2006.

Gustavo CERDA-VILLAFAÑA

Gustavo Cerda-Villafaña received his B.S. degree in mechanical engineering from the Mechanical Engineering Department of Universidad de Guanajuato in 1994, and his M.S. from the same university in 2000. He received his Ph.D. from the University of Bristol, United Kingdom in 2005. He is currently working on the development of systems in artificial intelligence and statistical signal processing.

Donato HERNANDEZ-FUSILIER

Donato Hernández Fusilier received his M. S. (1990) in electrical engineering from the school of engineering (FIMEE) of Universidad de Guanajuato and his B. S. (1986) in communications and electronics engineering from the same university. Currently, he holds an associated professor position at the DICIS of Universidad de Guanajuato. His major research interests are in neural networks and signals optics.