Adaptive Normalization: A Novel Data Normalization Approach for Non-Stationary Time Series

Eduardo Ogasawara, Leonardo C. Martinez, Daniel de Oliveira,
Geraldo Zimbrão, Gisele L. Pappa and Marta Mattoso

Abstract - **Data normalization is a fundamental preprocessing step for mining and learning from data. However, finding an appropriated method to deal with time series normalization is not a simple task. This is because most of the traditional normalization methods make assumptions that do not hold for most time series. The first assumption is that all time series are stationary, *i.e.*, their statistical properties, such as mean and standard deviation, do not change over time. The second assumption is that the volatility of the time series is considered uniform. None of the methods currently available in the literature address these issues. This paper proposes a new method for normalizing non-stationary heteroscedastic (with non-uniform volatility) time series. The method, named Adaptive Normalization (AN), was tested together with an Artificial Neural Network (ANN) in three forecast problems. The results were compared to other four traditional normalization methods, and showed AN improves ANN accuracy in both short- and long-term predictions.**

I. Introduction

Any application that deals with data requires a lot of time and effort for data preparation [1-3]. The main goal of data preparation is to guarantee the quality of the data before it is fed to any learning algorithm, and includes data cleaning, integration and transformation, and reduction. This paper focuses on data transformation methods, especially normalization, when dealing with time series data.

The most common normalization methods used during data transformation include the min-max (where the data inputs are mapped into a predefined range, varying from 0 or −1 to 1), the z-score (where the values of an attribute *A* are normalized according to its mean and standard deviation), and the decimal scaling (where the decimal point of the values of an attribute *A* are moved according to its maximum absolute value). However, these methods are not always applicable to time series data. Consider the min-max and the decimal scaling methods, for instance. Their applicability depends on knowing the minimum and/or maximum values of a time series, which is not always possible. We can assume these values are present in a time series sample, but future data might be out of bounds.

The z-score method, in contrast, is useful when the minimum and maximum values of an attribute are unknown, and can be applied to stationary time series [4,5], i.e., time series whose statistical properties, such as mean, variance, and autocorrelation, are constant over time. However, in the real world, most of the financial and economical time series are non-stationary [6]. In contrast with stationary time series, in non-stationary series data statistical properties do vary over time.

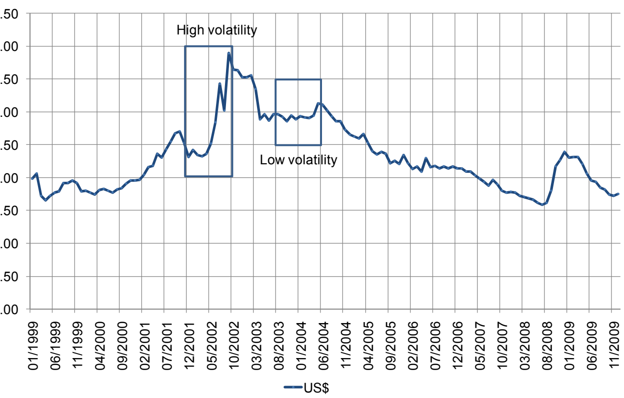

Fig. 1 Monthly average exchange rate of U.S. Dollar to Brazilian Real time series

In order to illustrate the concepts described above, Fig. 1 presents the monthly time series of average exchange rates of U.S. Dollar to Brazilian Real from January 1999 to December 2009. Complementary, Table I shows the mean, standard deviation, and minimum and maximum values of the series per year. Observe that these values change over time. For instance, the exchange rate mean in 1999 was 1.81, and this value rose to 3.08 in 2003. These variations shows that time series in Fig. 1 is non-stationary.

TABLE I
STATISTICS OF THE MONTHLY AVERAGE EXCHANGE RATE OF U.S. DOLLAR TO BRAZILIAN REAL TIME SERIES

Year	Mean	Std. Deviation	Min	Max
1999	1.81	0.13	1.50	1.97
2000	1.83	0.07	1.74	1.96
2001	2.35	0.25	1.95	2.74
2002	2.92	0.56	2.32	3.81
2003	3.08	0.26	2.86	3.59
2004	2.93	0.12	2.72	3.13
2005	2.44	0.17	2.21	2.70
2006	2.18	0.04	2.13	2.27
2007	1.95	0.13	1.77	2.14
2008	1.83	0.28	1.59	2.39
2009	2.00	0.23	1.73	2.31

Eduardo Ogasawara, Daniel de Oliveira, Geraldo Zimbrão and Marta Mattoso are with the Department of Computer Science, Federal University of Rio de Janeiro – Brazil (email: {ogasawara, danielc, zimbrao, marta}@cos.ufrj.br).
Leonardo Martinez and Gisele Pappa are with the Department of Computer Science, Federal University of Minas Gerais - Brazil (email: {leocm, glpappa}@dcc.ufmg.br).

Although there are ways of transforming non-stationary in stationary time series through mathematical manipulations, such as differencing transformations [7], this process is not always appropriated to handle non-stationary series [8,9], as it removes long-run information from data.

One traditional approach that attempts to overcome the problems of the aforementioned normalization methods to handle non-stationary time series uses the sliding window technique [10,11]. This approach divides data series into sliding windows, extracts statistical properties from data considering only the last ω items of the series, where ω is the length of the window, and normalizes each window considering these statistical properties. The sliding window technique works well for time series with uniform volatility [6,12,13], but most time series present non-uniform volatility [6], that is, high volatility for certain time periods and low for others.

For instance, consider the time series presented in Fig. 1. Its volatility is non-uniform, as it is high from December 2001 to October 2002 and low from April 2003 to February 2004. Time series presenting this behavior are said to be heteroscedastic [4] and the sliding window technique do not deal well with them, since all the normalized sliding windows present the same volatility.

This paper addresses the problem of normalizing non-stationary heteroscedastic time series. We propose a new method, named Adaptive Normalization (AN), which is a variation of the sliding window technique. The main difference between the two methods is that AN transforms the time series into a data sequence from which global statistical properties, obtained from a sample set, can be calculated and considered in the normalization process. Thus, the sliding windows of AN are able to represent different volatilities.

We also studied how Adaptive Normalization affects time series forecasting with artificial neural networks (ANN). We chose to start its analysis by using an ANN due to the great impact data normalization has in neural networks, as it prevents attributes with initially large ranges from outweighing attributes with initially smaller ranges [1,14], while improving error estimations and reducing training time [10,15,16].

Experiments with Adaptive Normalization were performed using three time series: U.S. Dollar to Brazilian Real Exchange Rate, Brazilian Agriculture Gross Product, and São Paulo Unemployment Rate. We compared the method with four other normalization techniques and the results showed that AN achieved better results for short- and long-term forecasts.

The remainder of this paper is organized as follows. Section II discusses traditional data normalization techniques. Section III introduces the new method, Adaptive Normalization. Section IV presents experimental results using AN with neural networks for time series forecasting. Finally, Section V presents some conclusions and directions for future work.

II. Traditional Data Normalization Methods

In this section, we briefly describe the three data normalization methods most commonly used in the literature: min-max, decimal scaling and z-score [1]. In addition, we also discuss the sliding window technique, usually applied to normalize time series data. Table II summarizes the notation that is used throughout this paper. Note that the term *sequence* is used to represent any ordered list of values (*e.g.*, a time series).

TABLE II
SUMMARY OF NOTATION

Symbol	Definition
min_A, max_A	The minimum and maximum values of attribute *A*
$\mu(A), \sigma(A)$	Mean and standard deviation of attribute *A*
$\lvert S \rvert$	Length of sequence *S*
$S[i]$	The *i*th entry of sequence *S* $(1 \le i \le \lvert S \rvert)$
$S[i:j]$	Subsequence os *S*, from index *i* to index *j*
$S^{(k)}$	The *k*-moving average sequence of *S*
$S_s^{(k)}$	The *k*-simple moving average sequence of *S*
$S_e^{(k)}$	The *k*-exponential moving average sequence of *S*
ω	Length of the disjoint/sliding windows
ϕ	Number of disjoint/sliding windows of training set
s_i	The ith disjoint/sliding window of sequence *S* $(= S[(i-1) \times \omega + 1 : i \times \omega], i \ge 1)$
$\delta(S^{(k)}, r_i)$	The level of adjustment of the *k*-moving average sequence of *S* to the disjoint sliding window r_i
$\delta(S^{(k)}, R)$	The level of adjustment of the *k*-moving average sequence of *S* to all the disjoint sliding windows of *R*

The min-max method normalizes the values of an attribute *A* according to its minimum and maximum values. It converts a value *a* of *A* to *a′* in the range [low, high] by computing:

$$a' = (\mathrm{high} - \mathrm{low}) \times \frac{a - \min_A}{\max_A - \min_A} + \mathrm{low}$$

The main problem of using the min-max normalization method in time series forecast is that the minimum and maximum values of out-of-sample data set are unknown. A simple way to overcome this problem is to consider the minimum (min_A) and maximum (max_A) values presented in the in-sample data set, and then map all out-of-sample values below min_A and above max_A to low and high, respectively. However, this approach leads to significant information loss and to a concentration of values on certain parts of the normalized range [1], which implies more computational effort and loss of quality in learning techniques [14,16].

Fig. 1 illustrates this problem in the monthly average exchange rate of U.S. Dollar to Brazilian Real time series. If we use the min-max normalization with an in-sample set from January 1999 to December 2001, we observe that from the middle of 2002, the min-max would lead to an out of bounds normalization.

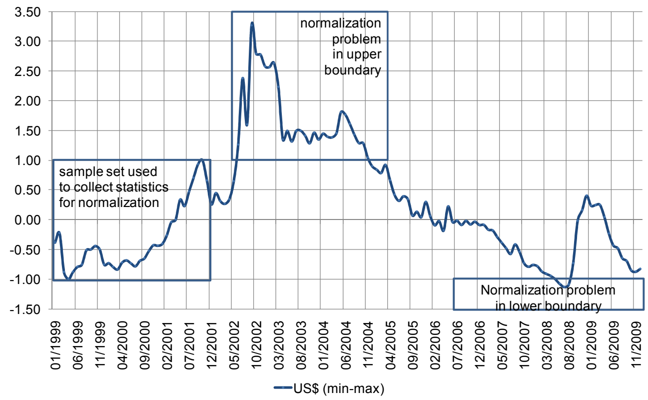

Fig. 2 U.S. Dollar to Brazilian Real Exchange Rate using min-max

Another common normalization method, the decimal scaling normalization, moves the decimal point of the values of an attribute A according to its maximum absolute value. Hence, a value a of A is normalized to a' by computing:

$$a' = \frac{a}{(10^d)}$$

where d is the smallest integer such that $Max(|a'|) < 1$. This method also depends on knowing the maximum values of a time series and has the same problems of min-max when applied to time series data.

Finally, in the z-score normalization, the values of an attribute A are normalized according to their mean and standard deviation. A value a of A is normalized to a' by computing:

$$a' = \frac{a - \mu(A)}{\sigma(A)}$$

This method is useful in stationary environments when the actual minimum and maximum values of attribute A are unknown, but it cannot deal well with non-stationary time series since the mean and standard deviation of the time series vary over time.

Another traditional approach commonly used for data normalization is the sliding window technique [10,11]. The basic idea of this approach is that, instead of considering the complete time series for normalization, it divides the data into sliding windows of length ω, extracts statistical properties from it considering only a fraction of ω consecutive time series values [1,17], and normalizes each window considering only these statistical properties. The rationale behind this approach is that decisions are usually based on recent data. The sliding window technique has the advantage of always normalizing data in the desired range. However, it has a drawback of assuming that the time series volatility is uniform, which is not true in many phenomena [4,6,18].

In order to illustrate this problem, Fig. 3 and Fig. 4 present two sample data of the monthly average exchange rate of U.S. Dollar to Brazilian Real time series using ω = 5. The first one is from August 2000 to December 2000 and the second is from April 2001 to August 2001. Fig. 3 shows the original time series values, while Fig. 4 shows the values normalized by the min-max method. Although the dataset from window number 1 had less volatility than the dataset of window number 2, the normalized windows do not preserve this behavior.

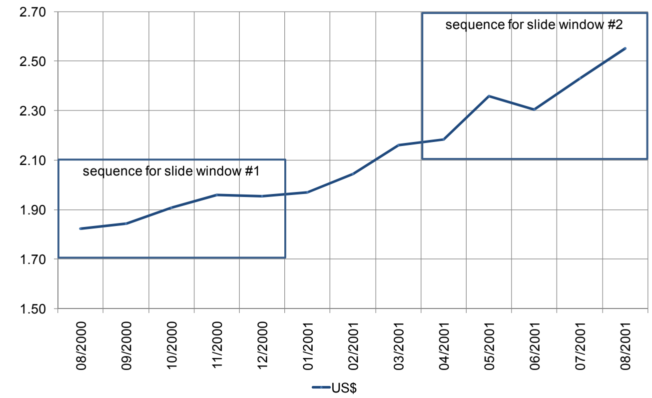

Fig. 3 U.S. Dollar to Brazilian Real Exchange Rate from aug/2000 to dec/2000 and from apr/2001 to aug/2001

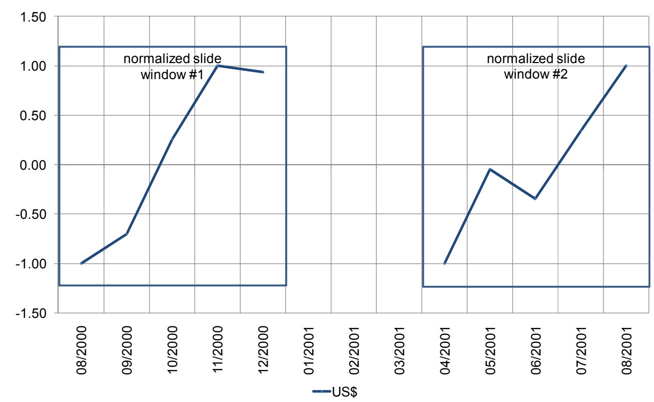

Fig. 4 U.S. Dollar to Brazilian Real Exchange Rate from aug/2000 to dec/2000 and from apr/2001 to aug/2001 normalized by sliding window

Traditional normalization methods, as discussed above, are successful in stationary time series. The sliding window technique tries to overcome the limitations of those methods, but with an implicit assumption of uniform volatility. As these techniques are extrapolated to non-stationary time series with heteroscedasticity, they are not capable to represent the time series correctly in a normalized range.

III. Adaptive Normalization

Adaptive Normalization is a novel data normalization approach specially developed to be applied to non-stationary heteroscedastic time series. Its complete process of data normalization can be divided into three stages: (i) transforming the non-stationary time series into a stationary sequence, which creates a sequence of disjoint sliding windows (that do not overlap); (ii) outlier removal; (iii) data normalization itself. The data resulting from this process are given as input to a learning method, such as an ANN. In this case, after forecasts are made, a denormalization and detransformation process maps the output values to the original values in the time series.

In order to illustrate AN, this section introduces the concepts and shows the method working in a toy example, based on the daily time series presented in Table III, which contains the exchange rate of U.S. Dollar to Brazilian Real from the 1st to the 17th of December 2009. This time series contain 13 points, and in this example we show how AN is used to forecast the 13th element of sequence *S*.

TABLE III
SAMPLE DAILY EXCHANGE RATE OF U.S. DOLLAR TO BRAZILIAN REAL TIME SERIES

i	Date	US$/R$: *S*	EMA: $S_e^{(5)}$
1	2009-12-01	1.734	1.721
2	2009-12-02	1.720	1.729
3	2009-12-03	1.707	1.734
4	2009-12-04	1.708	1.742
5	2009-12-07	1.735	1.745
6	2009-12-08	1.746	1.747
7	2009-12-09	1.744	1.752
8	2009-12-10	1.759	1.752
9	2009-12-11	1.751	1.760
10	2009-12-14	1.749	-
11	2009-12-15	1.763	-
12	2009-12-16	1.753	-
13	2009-12-17	1.774	-

A. Data Transformation

In the first stage of Adaptive Normalization, the original non-stationary time series is transformed into a stationary sequence. This transformation is based on the concepts of moving averages and proceeds in two steps. First, the moving average of the original time series is calculated. Then, its values are used to create a new stationary sequence which is divided into disjoint sliding windows (DSWs).

Moving averages (MAs) [7] have been widely used in many areas, such as finance [6] and econometrics [19]. They are useful for finding trends and patterns in time series data, and detecting changes in their behavior by reducing the effect of noise [20]. Furthermore, MAs can implicitly deal with inertia [4], an important concept in time series data. Inertia is a physical and mathematical concept that is expressed as the resistance an object offers to a change in its state of motion [21]. In time series data, inertia is an important property that allows handling the stability-plasticity dilemma [22]. In AN, in a moving average of order *k, k* corresponds to the number of periods used to introduce inertia to the new stationary sequence, as explained below.

MAs convert a given sequence *S* into a new sequence $S^{(k)}$, where each value in $S^{(k)}$ represents the average of *k* consecutive values of sequence *S*. In other words, given a sequence $S = \{S[1], S[2], \ldots, S[n]\}$ of length *n* and a moving average order k $(1 \le k \le n)$, the *i*-th value $S^{(k)}[i]$ $(1 \le i \le n-k+1)$ of the *k*-moving average sequence $S^{(k)}$ is defined as an average of the values of the subsequence $S[i : i + k - 1]$.

Two types of MA can be used in the adaptive normalization process: a Simple Moving Average (SMA) or an Exponential Moving Average (EMA) [23]. While a SMA is a sequence of non-weighted averages, EMAs are sequences of weighted averages with weighting factors decreasing exponentially. EMAs give more weight to recent observations, where the weights decrease by a constant smoothing factor α $(0 \le \alpha \le 1)$, w is usually expressed in terms of the EMA order k: $\alpha = 2/(k + 1)$.

When using SMA with Adaptive Normalization, $S_s^{(k)}[i]$ can be defined as:

$$S_s^{(k)}[i] = \frac{1}{k}\sum_{j=i}^{i+k-1} S[j], \forall i /\ 1 \le i \le n-k+1.$$

When using EMA, in contrast, $S_e^{(k)}[i]$ can be recursively defined as:

$$S_e^{(k)}[1] = S_s^{(k)}[1]$$

$$S_e^{(k)}[i] = (1-\alpha)S_e^{(k)}[i-1] + \alpha S[i+k-1],$$
$$\forall i /\ 2 \le i \le n-k+1$$

The third column of Table III shows the EMA with $k = 5$ and $\alpha = 0.333$ for the time series listed in the second column. For instance, $S_e^{(5)}[2] = 0.667 \times S_e^{(5)}[1] + 0.333 \times S[6] = 1.729$. After creating the moving average sequence, Adaptive Normalization transforms the original non-stationary time series into a stationary sequence divided into disjoint sliding windows, as explained below.

Given a sequence *S* of length *n*, its *k*-moving average S(k) of length $n - k + 1$, and a sliding window length ω, a new sequence *R* can be defined as:

$$R[i] = \frac{S[\lceil \frac{i}{\omega} \rceil (i-1) \bmod \omega]}{S^{(k)}[\lceil \frac{i}{\omega} \rceil]}, \tag{1}$$

for all $1 \le i \le (n - \omega + 1) \times \omega$. This sequence *R* is divided into $n - \omega + 1$ disjoint sliding windows. Considering our example, Table IV shows the sequence R divided into eight DSWs with ω = 6. For instance, the 9th element of sequence *R* appears in the third element of the second DSW. Its value can be calculated by Equation 1:
$R[9] = \frac{S[4]}{S_e^{(5)}[2]} = \frac{1.708}{1.729} = 0.988$. The DSWs $\{r1, \ldots, r7\}$ are used in the training data set while r_8 is used in the testing data set.

As can be seen, for each DSW r_i, all the fraction's denominator are the same ($S^{(k)}[i]$). This factor is important to preserve the original trend of the time series and to bring the same inertia to all the values in a DSW. Each DSW contains ω −1 input values and 1 output value. Note that, if $k > \omega - 1$, *i.e.*, if the moving average order is larger than the number of inputs, we should first calculate $S^{(k)}$ and then discard the $k-(\omega-1)$ first values of S in order to create the sequence *R*. For instance, if $k = \omega = 3$, we should remove the first term of sequence S. Then, the first DSW of R would be $\frac{S[2]}{S_e^{(3)}[1]}$, $\frac{S[3]}{S_e^{(3)}[1]}$ and $\frac{S[4]}{S_e^{(3)}[1]}$.

TABLE IV
SAMPLE OF DISJOINT SLIDING WINDOWS OF SEQUENCE R

r_i	$\frac{S[i]}{S_e^{(5)}[i]}$	$\frac{S[i+1]}{S_e^{(5)}[i]}$	$\frac{S[i+2]}{S_e^{(5)}[i]}$	$\frac{S[i+3]}{S_e^{(5)}[i]}$	$\frac{S[i+4]}{S_e^{(5)}[i]}$	$\frac{S[i+5]}{S_e^{(5)}[i]}$
1	1.008	1.000	0.992	0.993	1.008	1.015
2	0.995	0.987	0.988	1.003	1.010	1.009
3	0.984	0.985	1.000	1.007	1.006	1.014
4	0.980	0.996	1.002	1.001	1.010	1.005
5	0.994	1.000	0.999	1.008	1.003	1.002
6	1.000	0.999	1.007	1.003	1.001	1.009
7	0.995	1.004	0.999	0.998	1.006	1.001
8	1.004	0.999	0.998	1.006	1.000	1.012

The type of the MA to be used in Adaptive Normalization and its order vary according to the time series characteristics. We test the adjustment level of all the combinations of "MA type" and "order k", and the one with the best adjustment to all the DSWs is selected. First, we calculate the level of adjustment of each $S^{(k)}$ to each one of the training set DSWs:

$$\delta\left(S^{(k)}, r_i\right) = \frac{1}{\omega}\sum_{j=i}^{i+\omega-1}\left(S[j] - S^{(k)}[i]\right)^2, \forall i / 1 \le i \le \phi \quad (2)$$

where ϕ is the number of DSWs used in the training data set. Then, we calculate the level of adjustment of $S^{(k)}$ according to all the DSWs of R:

$$\delta\left(S^{(k)}, R\right) = \frac{1}{\phi}\sum_{i=1}^{\phi}\delta\left(S^{(k)}, r_i\right) \quad (3)$$

The $\delta\left(S^{(k)}, R\right)$ that achieves the lowest adjustment level is selected to be used in Adaptive Normalization. Although there are other ways to calculate the adjustments levels, we decided to use Equations 2 and 3 to calculate them for all the experiments. The concept of adjustment here (and consequently the definition of Equations 2 and 3) is the same used in linear regression: minimize the sum of the squares of some measure. We used the difference between the numerators and denominators of each fraction as our measure, with the main goal to keep the values of sequence R closest to 1.

B. Outlier Removal

The second stage of Adaptive Normalization is dedicated to outlier removal [1,3,24]. Outlier removal of sample data is a key step in the data preprocessing phase, and is also important for time series analysis. The main problem to the data normalization process arises when outliers occur in extreme boundaries of the time series, leading to incoherent minimum and/or maximum values. This affects the global statistics of the time series and also the data normalization quality, since values may be concentrated on a specific range of the normalized range.

To avoid this inconvenience, a method based on Box plots [25,26] for detecting outliers in a data sample set can be applied. The method prune any value smaller than the first quartile minus 1.5 times the interquartile range, and also any value larger than the third quartile plus 1.5 times the interquartile range [27], that is, all the values that are not in the range [$Q1-1.5\times IQR$, $Q3+1.5\times IQR$] are considered outliers. In Adaptive Normalization, any DSW that contains at least one outlier is not considered during the algorithm training phase.

In our example, the subsequence $R[1 : 42]$ (the training data set) illustrated by Table IV has $Q1 = 0.996$ and $Q3 = 1.006$. Then, $IQR = Q3-Q1 = 0.010$, $Q1-1.5\times IQR = 0.981$ and $Q3 + 1.5 \times IQR = 1.021$. Thus, only r_4 is discarded from the training data set, since it contains a value smaller than 0.981.

The multiplier 1.5 for the interquartile range may be adjusted depending on the data set. The value 3.0 is also commonly used to remove only the extreme outliers [27]. In the experiments reported in Section IV, we used the value 3.0.

C. Data Normalization

Adaptive normalization uses the min-max method to normalize the values of sequence R in the range [−1, 1], but in a different way of the traditional sliding window approach. The idea is to explore all the disjoint sliding windows in order to obtain global data statistics (including global minimum and global maximum), and to use these values as inputs for the min-max normalization method. However, the value $Q1 - 1.5 \times IQR$ ($Q3 + 1.5 \times IQR$) is considered the global minimum (maximum) if R contains any value smaller (larger) than it.

Continuing with our example, the minimum and maximum values used in the min-max normalization method were, respectively, 0.981 ($Q1-1.5\times IQR$) and 1.015 (maximum value of R). Table V shows the sequence R after normalization.

It is now possible to return to the example of Fig. 3, and compare the traditional sliding window normalization showed by Fig. 4 with Adaptive Normalization. Fig. 5 presents the normalized values of the monthly average exchange rate of U.S. Dollar to Brazilian Real time series from August 2000 to December 2000 and from April 2001 to July 2001 using AN, with $k = 4$ and $\omega = 5$. The problems observed using traditional sliding window methods do not occur when using AN. In the sequence of sliding window number 1 in Fig. 3 there is an upward trend, with lower volatility than the one of sliding window number 2. This still occurs when observing the normalized windows in Fig. 5. It is worth noticing that values are not stretched to reach -1 to 1. Indeed they respect the global volatility obtained from the whole sample set.

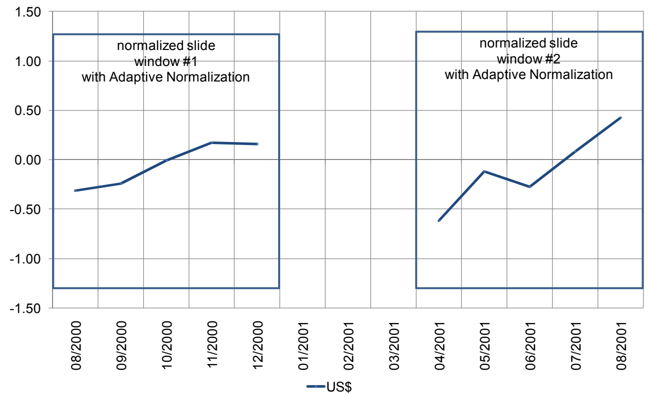

Fig. 5 U.S. Dollar to Brazilian Real Exchange Rate from aug/2000 to dec/2000 and from apr/2001 to aug/2001 normalized by Adaptive Normalization

TABLE V
SAMPLE OF DISJOINT SLIDING WINDOWS OF SEQUENCE R NORMALIZED WITH ADAPTIVE NORMALIZATION IN THE RANGE $[-1, 1]$

r_i	Sequence R					
1	0,585	0,102	-0,347	-0,313	0,620	1,000
2	-0,187	-0,634	-0,599	0,329	0,707	0,638
3	-0,801	-0,766	0,159	0,536	0,468	0,982
4	-	-	-	-	-	-
5	-0,221	0,154	0,086	0,597	0,324	0,256
6	0,112	0,044	0,554	0,282	0,214	0,690
7	-0,142	0,366	0,095	0,027	0,502	0,163
8	0,355	0,084	0,016	0,491	0,152	0,864

D. Data Denormalization and Detransformation

The transformed and normalized data resulting from the complete process described in the last three subsections are given as input to a learning method, such as an ANN. After forecasts are made, a denormalization and detransformation process maps the output values to the original values in the time series.

Given an attribute A, the denormalization process for min-max converts a value a' in the range [low, high] to a value a of A by computing:

$$a = \frac{a' - low}{high - low} \times (max_A - min_A) + min_A$$

Consider our toy example, where the output value of the ANN (the normalized forecast for *S[13]*) was 0.888. Then, its denormalized forecasted value was $\frac{0.888-(-1)}{1-(-1)} \times (1.015 - 0.981) + 0.981 = 1.013$.

In the detransformation phase it is necessary to convert the denormalized value to the original time series value. In order to do this, we only need to multiply the denormalized value by the correct $S^{(k)}[i]$. In our example, we proposed to forecast S[13], represented in the sequence R by the element $R[42] \frac{S[13]}{S_e^{(5)}[8]}$. Since the denormalized forecasted value for *S[13]* was 1.013, the (detransformed) forecasted value for *S[13]* was $1.013 \times S_e^{(5)}[8] = 1.013 \times 1.752 = 1.775$.

IV. EXPERIMENTS AND RESULTS

This section presents experimental results performed to evaluate the proposed adaptive normalization method. As explained before, the method was tested with an ANN, and from now on it is referred as NN-AN. Here, NN-AN is compared to four neural networks using different data normalization approaches: traditional min-max normalization (NN-MM), decimal-scaling normalization (NN-DS), z-score normalization (NN-ZS) and sliding windows/min-max normalization (NN-SW). In order to analyze the quality of the forecasts obtained by these methods, we also present the results achieved by the auto regression (AR) model [28].

The method was evaluated considering three different time series: (i) U.S. Dollar to Brazilian Real Exchange Rate, (ii) Brazilian Agriculture Gross Product, and (iii) São Paulo Unemployment Rate (%). The three time series are available for download from the Brazilian Institute of Applied Economic Research [29], but, for convenience, they can also be downloaded from first author's homepage [30].

Our main goal is to evaluate the forecast performance for short- and long-term horizons, forecasting the next twelve observations after the sample/training set, using 1-step-ahead and 12-step-ahead forecasts. For the 12-step-ahead forecast, we have used the recursive strategy [31], which considers the initial forecasted values to forecast the next ones. The performance of out-of-sample forecast is evaluated by two commonly used error measures, which represents different angles to evaluate forecasting models: the root mean square error (RMSE) and the mean absolute percentage error (MAPE). While RMSE is a measure of absolute performance, MAPE evaluates the ANN relative performance.

In all experiments, we used a classical feed-forward neural network [10], trained with the back-propagation algorithm, three neurons in the hidden layer and one neuron in the output layer, implementing a hyperbolic tangent function. The number of neurons in the input layer is the result of an autocorrelation analysis [28], commonly used for checking randomness in a data set by computing autocorrelations for data values at different time-lag separations. If the data set is random, the autocorrelations should be small (near zero) for all time-lag separations. If the data set is non-random, one or more of the autocorrelations are significantly non-zero.

Fig. 6 shows the sample autocorrelation function for the monthly average exchange rate of U.S. Dollar to Brazilian Real time series calculated using MatLab [32]. The horizontal highlighted lines are placed at zero plus and minus two approximate standard errors of the sample autocorrelations, namely $\pm \frac{2}{\sqrt{n}}$, where n is the length of the time series [7]. As observed, there are two relevant lags: lag 1, that exceeds two standard errors above zero and lag 7 that exceeds two standard errors below zero. Since the last relevant lag was lag 7, the number of selected inputs for this time series was equal to 7 (lags 1, 2, … , 7). For all the experiments, the number of inputs was calculated based on similar autocorrelation analysis.

The length of the sliding windows of NN-SW and the disjoint windows of NN-AN is the same, and equals to the number of inputs plus one (the output), giving to both methods the same dataset size for training. The value of the inertia parameter (order of the used moving average) of NN-AN is at most 10% of the sample size and is chosen by the best level of adjustment of all the possible moving averages, as described in Subsection III-A.

When training the network, the learning rate was set to 0.64, and the back-propagation algorithm also used a momentum of 0.8. The network was trained for 200,000 epochs. This choice was made to give equal conditions to all different normalization techniques and is in agreement with previous work [15] in which this structure usually presented results that were near to the optimal ones.

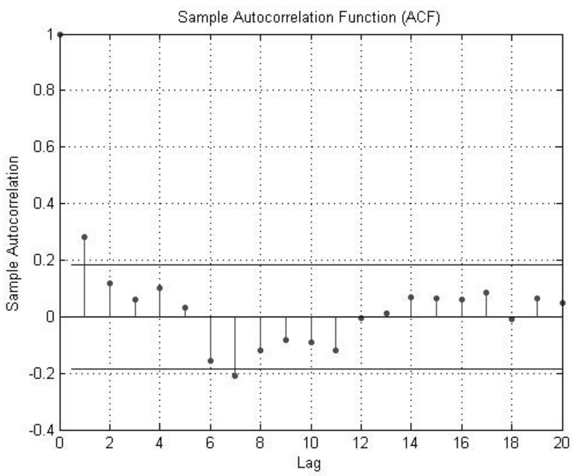

Fig. 6 Sample Autocorrelation of the monthly average exchange rate of U.S. Dollar to Brazilian Real time series

A. *Forecasting U.S. Dollar to Brazilian Real Exchange Rate*

The first time series used in our experiments was the monthly average exchange rate of U.S. Dollar to Brazilian Real. This time series represents a period of 132 months, varying from January 1999 to December 2009. The training set covered the first 120 months and the test set covered the last 12 months. The number of inputs used in this example was 7. The EMA was used with inertia 8.

Table VI shows the RMSE and MAPE obtained by the six algorithms for the 1-step-ahead and 12-step-ahead forecasts during the test period. The NN-AN achieved the best results for both the 1-step-ahead horizon, with RMSE and MAPE 24% and 14% smaller than the AR (2nd best), and the 12-step-ahead horizon. For the 12-step-ahead horizon, the RMSE and MAPE were 24% and 22% smaller than the NN-SW (2nd best).

B. *Forecasting Brazilian Agricultural Gross Product*

The second time series used in our experiments was the quarterly Brazilian Agriculture Gross Product - Index Linked (average in 1995 = 100). This time series is seasonally adjusted, i.e., it does not contain the seasonal component. By removing the seasonal component, it is better to reveal certain non-seasonal features and easier to focus on the trend and cyclical components of the time series. The series represents a period of 119 quarters, varying from 1980 Q1 to 2009 Q3. The training set covered the first 107 quarters and the test set covered the last 12 quarters. Both the number of inputs and the SMA order were equal to 4.

TABLE VI
PERFORMANCE OF ALGORITHMS TO FORECAST THE MONTHLY AVERAGE EXCHANGE RATE OF U.S. DOLLAR TO BRAZILIAN REAL TIME SERIES

	RMSE		**MAPE (%)**	
Algorithm	1-step	12-step	1-step	12-step
AR	0.082	0.545	3.174	27.099
NN-MM	0.177	1.173	8.446	57.611
NN-DS	0.094	1.444	3.545	69.517
NN-ZS	0.126	0.814	4.526	40.931
NN-SW	0.088	0.451	3.661	20.917
NN-AN	0.062	0.345	2.730	16.398

Table VII presents the RMSE and MAPE of the forecasts. As in the previous experiment, the NN-AN achieved the best results considering both RMSE and MAPE and the two forecasting horizons. Considering the 1-step-ahead horizon, the NN-AN obtained both the RMSE (4%) and MAPE (17%) smaller than the AR. For the 12-step-ahead horizon, both the RMSE (3%) and the MAPE (8%) were smaller than the NN-DS.

TABLE VII
PERFORMANCE OF ALGORITHMS TO FORECAST THE QUARTERLY BRAZILIAN AGRICULTURE GROSS PRODUCT TIME SERIES

	RMSE		**MAPE (%)**	
Algorithm	1-step	12-step	1-step	12-step
AR	0.082	0.545	3.174	27.099
NN-MM	0.177	1.173	8.446	57.611
NN-DS	0.094	1.444	3.545	69.517
NN-ZS	0.126	0.814	4.526	40.931
NN-SW	0.088	0.451	3.661	20.917
NN-AN	0.062	0.345	2.730	16.398

C. *Forecasting São Paulo Unemployed Rate*

The third experiment used the time series of the monthly Unemployment Rate of São Paulo. This series considers 130 months, varying from January 1999 to October 2009. The first 118 months were chosen for training and the last 12 months were chosen for test. For this experiment, we used 16 inputs and a SMA of order 15.

Table VIII presents the RMSE and MAPE of the forecasts. Again, the NN-AN gave the best results for all the tests, with the RMSE 15% and the MAPE 14% smaller than the one obtained by the AR for the 1-step-ahead forecasts. For the 12-step-ahead horizon, the RMSE (9%) and the MAPE (17%) were smaller than the NN-SW.

TABLE VIII
PERFORMANCE OF THE ALGORITHMS TO FORECAST THE MONTHLY SÃO PAULO UNEMPLOYMENT RATE TIME SERIES

Algorithm	RMSE		MAPE (%)	
	1-step	12-step	1-step	12-step
AR	0.687	2.354	4.356	15.384
NN-MM	1.288	2.014	8.878	14.091
NN-DS	0.794	2.912	4.668	18.793
NN-ZS	2.730	2.885	20.455	21.472
NN-SW	0.778	1.067	5.645	7.923
NN-AN	0.587	0.975	3.742	6.553

V. CONCLUSIONS AND FUTURE WORK

This paper presented Adaptive Normalization (AN), a new method for normalizing non-stationary heteroscedastic time series. AN is a variation of the sliding window technique and has the advantage of transforming the time series into a data sequence, from which global statistical properties of a sample set can be calculated and considered during the normalization process. This allows us to build sliding windows that are capable to represent different volatilities, *i.e.*, preserve the original time-series properties inside each relative slide window. Moreover, this method does not require renormalizing the entire dataset as more time series data become available.

We studied how Adaptive Normalization affects time series forecasting with artificial neural networks (ANN), since ANN are very sensitive to data normalization. Experiments were performed in three datasets, and the results compared to four other normalization methods. The neural network using adaptive normalization outperformed both the ANN using other normalization methods as well as an auto regression method.

As future work, we plan to analyze the behavior of adaptive normalization with other learning methods, such as Support Vector Machines, as well as its combination with ANN for clustering.

ACKNOWLEDGMENTS

The authors would like to thank CNPq and CAPES for financial support. The authors are grateful to the High Performance Computing Center (NACAD-COPPE/UFRJ), where the experiments were performed.

REFERENCES

[1] P. Tan, M. Steinbach, and V. Kumar, 2005, *Introduction to Data Mining*. Addison Wesley.

[2] J. Han and M. Kamber, 2006, *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

[3] D. Pyle, 1999, *Data Preparation for Data Mining*. 1 ed. Morgan Kaufmann.

[4] D.N. Gujarati and D.C. Porter, 2008, *Basic econometrics*. McGraw-Hill New York.

[5] M.G. Kendall, 1976, *Time Series*. 2 ed. Oxford Univ Pr (Txt).

[6] R.S. Tsay, 2001, *Analysis of Financial Time Series*. 1 ed. Wiley-Interscience.

[7] J.D. Cryer and K. Chan, 2008, *Time Series Analysis: With Applications in R*. 2 ed. Springer.

[8] C.R. Nelson and C.R. Plosser, 1982, Trends and random walks in macroeconmic time series : Some evidence and implications, *Journal of Monetary Economics*, v. 10, n. 2, p. 139-162.

[9] D.A. Pierce, 1977, Relationships--and the Lack Thereof--Between Economic Time Series, with Special Reference to Money and Interest Rates, *Journal of the American Statistical Association*, v. 72, n. 357 (Mar.), p. 11-26.

[10] S. Haykin, 2008, *Neural Networks and Learning Machines*. 3 ed. Prentice Hall.

[11] J. Lin and E. Keogh, 2004, Finding or not finding rules in time series, p. Emerald Group Publishing Limited.

[12] V. Fang, V.C. Lee, and Y.C. Lim, 2005, "Volatility Transmission Between Stock and Bond Markets: Evidence from US and Australia", *Intelligent Data Engineering and Automated Learning - IDEAL 2005*, , p. 580-587.

[13] E.H. Wu and P.L. Yu, 2005, "Volatility Modelling of Multivariate Financial Time Series by Using ICA-GARCH Models", *Intelligent Data Engineering and Automated Learning - IDEAL 2005*, , p. 571-579.

[14] L.A. Shalabi and Z. Shaaban, 2006, Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix, In: *Proceedings of the International Conference on Dependability of Computer Systems*, p. 207-214

[15] E. Ogasawara, L. Murta, G. Zimbrão, and M. Mattoso, 2009, Neural networks cartridges for data mining on time series, In: *IJCNN*, p. 2302-2309, Atlanta, USA.

[16] J. Sola and J. Sevilla, 1997, Importance of input data normalization for the application of neural networks to complex industrial problems, *IEEE Transactions on Nuclear Science*, v. 44, n. 3, p. 1464-1468.

[17] H. Li and S. Lee, 2009, Mining frequent itemsets over data streams using efficient window sliding techniques, *Expert Syst. Appl.*, v. 36, n. 2, p. 1466-1477.

[18] J.C. Hull, 2005, *Options, Futures and Other Derivatives*. 6 ed. Prentice Hall.

[19] C. Chatfield, 2003, *The Analysis of Time Series: An Introduction, Sixth Edition*. 6 ed. Chapman and Hall/CRC.

[20] Y. Moon and J. Kim, 2007, Efficient moving average transform-based subsequence matching algorithms in time-series databases, *Inf. Sci.*, v. 177, n. 23, p. 5415-5431.

[21] I. Newton, 1999, *The Principia : Mathematical Principles of Natural Philosophy*. 1 ed. University of California Press.

[22] S. Grossberg, 1988, *Neural Networks and Natural Intelligence*. Bradford Book.

[23] Rui Jiang and K. Szeto, 2003, Extraction of investment strategies based on moving averages: A genetic algorithm approach, In: *International Conference on Computational Intelligence for Financial Engineering*, p. 403-410

[24] K. Choy, 2001, Outlier detection for stationary time series, *Journal of Statistical Planning and Inference*, v. 99, n. 2 (Dezembro.), p. 111-127.

[25] D.C. Hoaglin, F. Mosteller, and J.W. Tukey, 2000, *Understanding Robust and Exploratory Data Analysis*. 1 ed. Wiley-Interscience.

[26] R. Matignon, 2005, *Neural Network Modeling Using SAS Enterprise Miner*. AuthorHouse.

[27] Kvanli/Pavur/Keeling, 2006, *Concise Managerial Statistics*. 1 ed.

[28] G.E. Box, G.M. Jenkins, and G.C. Reinsel, 2008, *Time series analysis: forecasting and control*. 4 ed. Wiley.

[29] Ipeadata, 2010, *Ipeadata database*, http://www.ipeadata.gov.br.

[30] E. Ogasawara, 2010. IJCNN 2010 Datasets. Dispon?vel em: http://www.cos.ufrj.br/~ogasawara/ijcnn2010. Acesso em: 24 Mar 2010.

[31] J. Tikka and J. Hollmén, 2008, Sequential input selection algorithm for long-term prediction of time series, *Neurocomput.*, v. 71, n. 13-15, p. 2604-2615.

[32] Matlab, 2009, *The Mathworks MatLab & Simulink*, http://www.mathworks.com/.