Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2006) 79–92
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Detecting novelties in time series through neural networks forecasting with robust confidence intervals

Adriano L.I. Oliveira[a,*], Silvio R.L. Meira[b]

[a]*Department of Computing Systems, Polytechnic School of Engineering, Pernambuco State University, Rua Benfica, 455, Madalena, Recife – PE 50.750-410, Brazil*
[b]*Center for Informatics, Federal University of Pernambuco, P.O. Box 7851, Cidade Universitaria, Recife – PE 50.732-970, Brazil*

Received 16 August 2005; received in revised form 13 March 2006; accepted 18 May 2006
Communicated by G.P. Zhang
Available online 8 July 2006

Abstract

Novelty detection in time series is an important problem with application in a number of different domains such as machine failure detection and fraud detection in financial systems. One of the methods for detecting novelties in time series consists of building a forecasting model that is later used to predict future values. Novelties are assumed to take place if the difference between predicted and observed values is above a certain threshold. The problem with this method concerns the definition of a suitable value for the threshold. This paper proposes a method based on forecasting with robust confidence intervals for defining the thresholds for detecting novelties. Experiments with six real-world time series are reported and the results show that the method is able to correctly define the thresholds for novelty detection.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Time series; Novelty detection; Fraud detection; Anomaly detection; Forecasting; Confidence intervals; Neural network

1. Introduction

Time series data appear in virtually every area of human knowledge. Examples of time series include electrocardiograms, daily values of stocks and monthly sales of companies [4,38,39]. Novelties in time series can be informally defined as unexpected values or sequences of values. Applications of novelty detection in time series include the detection of machine failures and the detection of frauds in a financial system [5,23]. The method proposed in this work was designed especially for the latter application.

In classification problems, novelty detection is the ability of the classifier to reject patterns that do not belong to any of the classes of the training set [3,16–19,32]. This is a very important issue in many practical problems such as scene analysis [32] and intrusion detection in computer systems [28]. In scene analysis with neural networks, for example, it is important to design the network to reject images that do not correspond to any of the classes of the patterns available during training [32]. Popular neural networks architectures such as multilayer perceptrons (MLPs) do not have this inherent ability and therefore additional mechanisms must be designed to make these networks detect novelties properly [16,17,36,37]. A number of approaches for novelty detection in classification problems have recently appeared in the literature. Among these methods, some are based on statistical methods [16], others on artificial neural networks [17,18], and others on support vector machines (SVMs) [29,31].

In the problem of novelty detection, the training phase involves building a description of a target set of objects [33]. In the test phase, the novelty detection method should detect which (new) objects of the test set resemble the training set and which do not. The objects which do not resemble the training set are referred to as *novelties* or *outliers*.

The problem of novelty detection is also referred to in the literature as *one-class classification*, *data description*, *outlier detection* and *concept learning* [33]. The different

*Corresponding author. Tel.: +55 81 99764841; fax: +55 81 34137749.
E-mail address: adriano@dsc.upe.br (A.L.I. Oliveira).
URL: http://adriano.dsc.upe.br.

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.05.008

terms originate from the different applications of novelty detection [33]. For example, in outlier detection we are interested in excluding outliers from the training data before training a classifier. Such outliers can be caused by errors or noise in the measurement of feature values. They should be excluded from the training data since training without them usually helps producing better classifiers [33]. An example of application which is termed novelty detection is face recognition with the reject option. In this application, we are interested in training a classifier to recognize faces, yet we would like it to be able to detect—in the test phase or when used in practice—new pictures of faces which do not belong to any person of the training set [36]. Such pictures would not be recognized. Instead, the system would reject them, that is, it would label them as *novelties*.

One of the most important problems in time series analysis is time series forecasting. This problem involves the prediction of future values of the time series based on its past behavior [39]. Nevertheless, the detection of novelties in time series has gained recent interest with a number of methods proposed in the literature [15,21–25,6,7,30,10]. The problem of novelty detection in time series is referred under various names in the literature including anomaly detection [6,7] and surprise detection [10].

The applications of techniques for novelty detection in time series are very important. One example is the detection of faults in machines. In this case, the dynamic behavior of a machine is modeled by a signal that is, in fact, a time series. This is the case for example for mill machines [5]. A novelty detection system would learn the normal behavior of the machine through the time series. Later, the system would be able to detect novelties which, in this case, would mean machine breakage [5].

Fraud detection in financial systems is another important application of novelty detection in time series. Examples of this application include the detection of frauds in payroll systems [21] and in accounting systems [13,14]. The monthly earnings in a payroll and the accounts in an accounting system are time series. In a fraud detection application, the normal behavior of these series would be learned by a novelty detection system. Later, the system would be used to detect abnormal behavior which, in this case, could mean fraud.

The problem of detecting novelties in time series has recently received increased attention, with a number of different methods proposed in the literature [13,21,15, 23,24]. A straightforward method for novelty detection in time series is based on forecasting [13,21]. A forecasting model is built from the historical values and is used later to predict future values. If a predicted value differs from the observed value beyond a certain threshold, a novelty would be indicated. The definition of the value of the threshold to be used for detecting novelties is the main problem of novelty detection based on forecasting [13].

The difficulties of forecasting-based novelty detection have motivated the proposal of other methods, based on classification of time series windows [15,22–25,9,1]. These methods are meant mainly to analyze a sequence of points of a given length w (also referred to as a window) and to indicate whether they correspond to normal or anomalous behavior. For instance, the method by Keogh et al. is devoted to detect *discords* in a time series, where a discord is defined as a "sequence that is *least* similar to all other sequences of the time series" [9]. Nonetheless, some of these classification-based methods can also be used for detecting novelties in a single point of a time series [1].

This paper proposes the use of *robust confidence intervals* as a means for defining the thresholds for detecting novelties in time series and therefore improve performance of forecasting-based time series novelty detection. Robust confidence intervals for forecasting are directly computed from the prediction errors of the forecasting algorithm for the series under analysis [20].

In many papers on time series forecasting only the predictions are given [38,2,11,12]. Ideally, however, confidence intervals should also be given for each prediction in order to express the confidence in that prediction [20]. In other works, confidence intervals are computed by assuming that errors follow some probability distribution defined by a number of parameters. The errors are quite often assumed to follow a Gaussian distribution. The errors are then used to estimate the parameters and compute the confidence. The problem is that in practice the distribution assumed for the errors may not be correct. Robust confidence intervals are more general because they do not make any assumptions about the errors [20].

We report on a number of experiments using six real-world time series performed to evaluate the proposed method. The experiments reported here were performed for short-term forecasting (one step ahead). The results show that the real values on the test set of each time series lie within the confidence intervals. Thus, these intervals can be used as reliable thresholds for the detection of novelties. Two of the time series used in the experiments have known novelties. Our experiments show that the proposed method was able to detect these novelties.

This paper is organized as follows. In Section 2, we discuss the difficulties of detecting novelties in time series via forecasting. In particular, we discuss the difficulty to define the value of the threshold in current forecasting-based novelty detection techniques. Section 3 presents the proposed method for novelty detection in time series based on robust confidence intervals. A number of experiments with the proposed method using six real-world time series are reported in Section 4. Finally, Section 5 presents conclusions and suggestions for further research.

2. Difficulties of novelty detection in time series based on forecasting

Forecasting-based novelty detection relies on the forecasting capabilities of the algorithm used for the prediction. The idea is to use the algorithm to predict the future values of the series for a given period (window). If the predicted value deviates significantly from the real value, the algorithm alerts the user. This approach was used in the analysis of monthly balances of a manufacturing company [13], which compared two MLP models.

Koskivaara applied MLPs as the forecasting method and has performed multivariate analysis, that is, used only one MLP to simultaneously predict the values of nine financial accounts [13]. Each account had only 66 months available—from May 1990 to December 1995. The first 54 months were used for training while the 12 months of year 1995 were used for testing. Two MLP models were used: model 1 used two months to predict a third one and model 2 used four months to predict a fifth one.

The results obtained in the test set by Koskivaara are summarized here in Table 1. The test set had only normal data points [13]. Table 1 shows the forecasting performance of the method and the relative differences between predicted and real (target) values obtained by the neural networks. For example, the third line of table 1 shows that 24% of the test months had a difference between predicted and target outputs smaller than 5% (for model 1). For model 2, 26% of the months of the test set had a difference between predicted and target outputs smaller than 5%. Notice that there are $12 \times 9 = 108$ points in the test sets because they had nine time series and 12 months in the test set. Unfortunately, in these experiments the time series did not have novelty, that is, it had only normal behavior. Hence, it was not possible for the author to demonstrate whether his approach was able to detect real novelties [13].

Model 2 obtained slightly better results than model 1. Note that there are 3% and 4% (for models 1 and 2, respectively) of the 108 data points for which the relative difference between predicted and real values is greater than 30%. Koskivaara does not discuss what threshold should be used to distinguish between normal and fraudulent behavior [13]. If we set this threshold to 10%, that is, if we admit frauds (that is, deviations) above this value, then, as shown in the table, the system would produce false alarms (that is, false negatives, also referred to as *type I error* in the literature [33]) for around half of the months (to be exact, for model 1, the false alarm rate would be 58%, whereas for model 2 it would be 48%). In practice, we would like this threshold to be small so that we could detect small frauds, that is, small deviations from the normal behavior of the time series. Yet this has lead to great false alarm rates in this case. An alternative would be to use 30% relative error as threshold. In this case, one would get much smaller false alarm rates: 3% for model 1 and 4% for model 2. The problem is that, in this case, the system would permit frauds up to 30% (see Table 1). As we have mentioned, the time series employed by Koskivaara contained only normal behavior, therefore he did not report anything about false positives (also referred to as *type II error* in the literature [33]), that is, situations where a known novelty was not detected by the method [13].

Table 1
Results from Table 3 of [13], comparing two MLP models for monthly balances forecasting

	Model 1	Model 2
Test set size (months)	12	12
RMSE	0.15225	0.13942
Difference between predicted and real output: <5%	24%	26%
Difference between predicted and real output: <10%	42%	52%
Difference between predicted and real output: <30%	97%	96%
Average difference	12%	10%

3. A method based on robust confidence intervals

In many papers on time series forecasting, especially those using neural networks for forecasting, only the predicted values are provided. In practice, however, it is very important to provide an indication of the reliability of these predictions as well [20]. This can be achieved by using confidence intervals. In many cases these intervals are computed by supposing that the errors follow a normal (Gaussian) distribution or another more general distribution with a number of parameters. Nevertheless, it is argued that in practice this assumption rarely holds and therefore it can lead to wrong confidence intervals. Robust confidence intervals were proposed to overcome this limitation, since they do not make assumptions about the distributions of the errors [20].

Robust confidence intervals are generic and can be computed for predictions made by any model, such as exponential smoothing, ARIMA and neural networks models [20]. They can be computed for both univariate and multivariate time series forecasting. They also work for both short-term and long term forecasting. In this work we consider only short-term (one step ahead) forecasting.

Robust confidence intervals for predictions are computed from errors collected from a set of predictions within a known data set. Ideally, this data set should be different from the training set. Nonetheless, experience indicates that reuse of the training set may be acceptable if care is taken to avoid overfitting [20]. In this work, overfitting is avoided by using the well known *early stopping* method [8,26]. This method avoids overfitting by using a validation set different from the training set and by stopping training when the validation error starts to increase.

3.1. Collecting prediction errors

In this work the errors are collected on both the training and validation sets, after training the neural networks. Suppose that p previous data points from the time series are used to predict the following $p+1$. For a given data set $x_1, \ldots, x_n$, the first short-term prediction error is computed by feeding the trained network with the first p values from the set (training or validation) and taking the difference between the predicted and known correct values, $\hat{x}_{p+1} - x_{p+1}$. The sign of the computed errors must be preserved [20]. To compute the next error, p values from the data set starting from the second one are fed to the to network to obtain the predicted value $\hat{x}_{p+2}$. The second error is $\hat{x}_{p+2} - x_{p+2}$. This process is carried out up to the end of the data set.

3.2. Computing robust confidence intervals

A robust confidence interval for predictions is computed from errors collected from the training and validation sets. The intuition behind robust confidence intervals is that errors of a given size will tend to occur with about the same frequency that they have occurred previously. In other words, the frequency of occurrence of errors of a given size in the training and validation sets is assumed to be the same when the network is used for forecasting. For example, suppose that about one-quarter of the errors collected from the training and validation sets exceeds 2.3. Then, it is reasonable to expect that when the model is used for forecasting, the errors will exceed 2.3 about one-quarter of the times. As commented previously, this is true if the model does not overfit training data.

In order to build robust confidence intervals, the collection of n error measurements is initially sorted in ascending order, yielding $\{e_1, \ldots, e_n\}$. Next, the *sample distribution function* of the errors, $S_n(e)$, is computed according to Eq. (1).

$$S_n(e) = \begin{cases} 0 & \text{if } e < e_1, \\ r/n & \text{if } e_r \leqslant e < e_{r+1}, \\ 1 & \text{if } e_n \leqslant e. \end{cases} \tag{1}$$

If the collection of errors used to compute $S_n(e)$ is representative of the errors that will be encountered during forecasting, and if n is large enough, then $S_n(e)$ can be assumed to be close to $F(e)$, the true error distribution. In this case the confidence intervals for upcoming predictions are computed simply by keeping as much of the interior of $S_n(e)$ as is desired and by using the limits of this truncated collection to define the confidence interval. In practice it is usually desirable to have intervals symmetric in probability, even though this implies that they will not generally be symmetrical in the error.

For very large error collections and moderate values of p, $n \times p$ values should be discarded from each extreme in order to build the confidence intervals. For smaller samples, however, the recommended amount to be discarded from each extreme is $n \times p - 1$ [20]. If the result is a real number it should be truncated. p is the fraction of probability in each tail of the distribution function. For example, if a 90% confidence interval is desired, $p = (1.0 - 0.9)/2 = 0.05$.

The following example illustrates the procedure used to compute robust confidence intervals. Suppose there are $n = 10$ collected errors. This collection is firstly sorted to give $\overrightarrow{e} = \{-0.4, -0.3, -0.1, 0.0, 0.0, 0.0, 0.1, 0.1, 0.2, 0.4\}$. Suppose one wants to build a 60% confidence interval based on this error collection. Notice that this is an unrealistically small collection used only to illustrate the procedure. To build a 60% confidence interval for predictions, $p = (1.0 - 0.6)/2 = 0.2$, which gives $n \times p - 1 = 1$. Thus, one sample must be discarded from each extreme of the error collection. Hence, the 60% confidence interval for a future predicted value $\hat{x}_p$ would be $\{\hat{x}_p - 0.3, \hat{x}_p + 0.2\}$.

3.3. Increasing the error collection

In order to build robust confidence intervals it is important to collect a reasonable number of errors on training and validation sets [20]. The problem is that in many practical financial systems, the time series of interest are quite short. For example, time series with only 66 months were analyzed by Koskivaara in an accountancy problem [13]. In our work regarding payroll auditing we have analyzed time series with 84 months [21].

In the payroll auditing problem, suppose we reserve 12 months for test and use four months in order to predict a fifth. Then, we are left with only 68 months for training and validation. Thus, our error collection will have only 68 points. It would be desirable to have more errors in order to build more reliable confidence intervals.

We propose the system shown in Fig. 1 to tackle this problem. The proposed method is based on using a machine committee composed of several neural networks forecasters [8]. Firstly, the time series is pre-processed. This step may include simple differentiation or seasonal differentiation, depending on the characteristics of the time series under analysis [4]. Subsequently, several neural networks forecasters are trained independently with the same time series for one-step ahead forecasting. Finally, the prediction furnished by this system will be the mean of the predictions of the neural networks forecasters for the same month. The error collection is formed by considering the forecasting error of each neural network forecaster. Thus, if we use 10 forecasters, for example, we are able to increase our error collection by a factor of 10. In practice one could use any forecasting model in the committee of Fig. 1. In this work we consider only the use of MLPs or Elman networks as forecasters.

The machine committee proposed here is based on the ensemble mean, as shown in Fig. 1 [8]. In our case, the main motivation for using such a committee is to increase the size of our error correction. Nevertheless, there is another motivation for using such an architecture [8]. This motivation has two aspects, namely, (a) if the combination of neural networks of Fig. 1 is replaced by a single neural network we would have a network with a large number of parameters which would take more time for training than the training time of the machine committee (whose networks can be trained in parallel) and (b) using a big network increases the risk of overfitting training data. Thus, it is better to use a combination of smaller networks

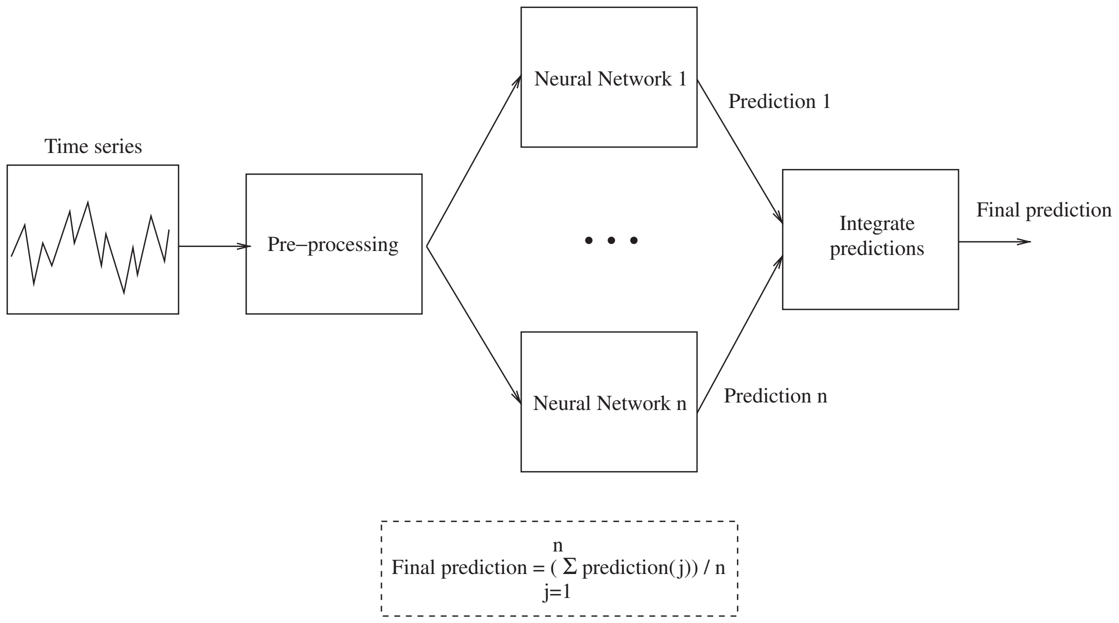

Fig. 1. System based on machine committee used for increasing the error collection for computing robust confidence intervals.

[8]. Finally, it is demonstrated that the variance of the ensemble mean produced by the machine committee is smaller than the variance of a single neural network [8].

3.4. Detecting novelties with robust confidence intervals

The robust confidence intervals discussed in this section provide an adequate threshold for novelty detection in time series. The forecasting-based novelty detection method proposed in this paper states that a future value from a time series is to be considered as a novelty if it lies beyond the confidence interval of the respective prediction. This method is robust because it takes into account the previous network forecasting performance in order to establish the threshold for detection of novelties in the future. If the observed value of the series lies within the confidence interval for the respective prediction, one assumes that the difference between observed and predicted values does not correspond to a novelty, but is due to the inherent uncertainty of the forecasting model.

The novelty detection method with robust confidence intervals can be used in practical auditing applications such as payroll and accounting auditing [21,13]. Consider the application of the proposed method in payroll auditing. Each time series is supposed to be free of frauds, because they were previously audited. A neural network forecasting model would be built for each time series based on historical values. Auditing would take place monthly, after the payroll is computed. Firstly, the one step ahead forecasting would be computed along with the respective robust confidence intervals. Next, the value furnished by the payroll system for this month would be compared to the forecasted value with confidence intervals. If the furnished value is outside the confidence intervals for the forecasting the system would detect a novelty. The auditors would concentrate their work on those earnings/deductions of the payroll for which a novelty was indicated by the model.

An architecture for an intelligent system built using the methods proposed in this paper is depicted in Figs. 2 and 3. Fig. 2 illustrates the training phase, whereby each time series of the financial system under analysis is fed to a different novelty detector. Each time series is assumed to be free of frauds and therefore each novelty detector learns to recognize the normal behavior of the respective time series in the training phase. In other words, we assume that each time series has been analyzed before and that they represent the normal behavior.

Fig. 3 illustrates the intelligent system for fraud detection in financial systems in operation. After the training phase the system can be put in operation. Suppose that the system is used to detect frauds in payroll systems. In this case, each earning and deduction is represent by a monthly time series and in many cases there are hundreds of them in a company or organization. Furthermore, the company or organization can have hundreds or even thousands of employees.

The system described here could be used to analyze earnings or deductions per employee or the total value of the earnings and deductions considering all employees of the company. In both cases, after the training phase the system would be fed with new values of each time series to be analyzed. In many practical cases, auditing takes place annually. In this case, the final decision provided by the system would consist of a list of months and earnings (or deductions) whose behavior is more likely to represent fraud. Using this information, an auditor would focus the investigations in these cases.

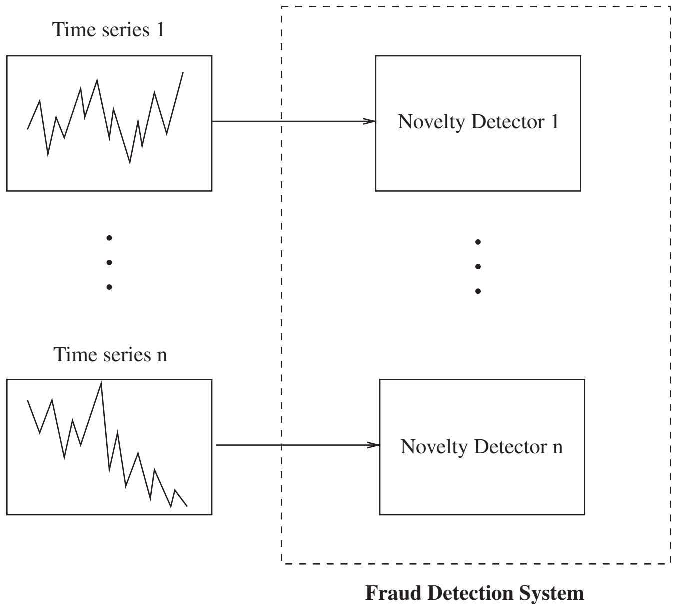

Fig. 2. Intelligent system for fraud detection in financial systems: training phase.

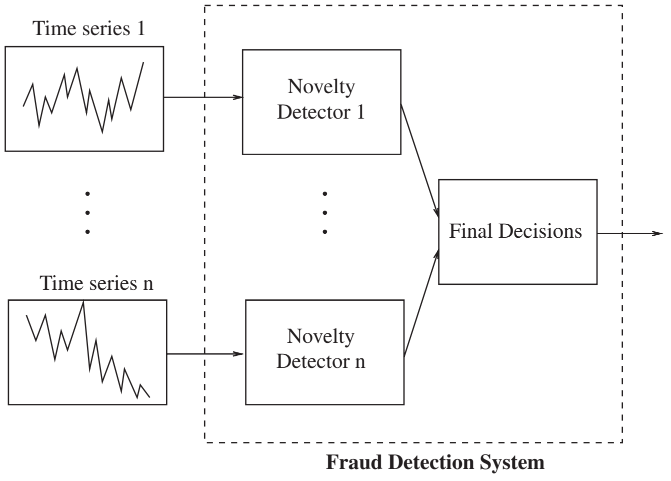

Fig. 3. Intelligent system for fraud detection in financial systems: system in operation.

4. Experiments

In this section we report on a number of experiments carried out to show the effectiveness of the proposed method. Firstly, we report on experiments performed using financial time series that do not have novelties in order to show that the proposed method is able to correctly establish the confidence interval for normal behavior. Subsequently, we study the effect of using the machine committee of Fig. 1 in conjunction with our method based on robust confidence intervals. We compare the performance of the method proposed in this paper using a machine committee with the same method using a single Elman neural network. Finally, we describe experiments with two real-world time series which have known novelties. These experiments show that the proposed method successfully detects such novelties.

4.1. Experiments with financial time series without novelties

4.1.1. Experimental setup

In this subsection we report on a number of experiments using four real-world time series which The time series used in these experiments are depicted in Figs. 4–7. These series have been used in previous novelty detection works [23–25,21]. The first three series have 84 values corresponding to the months from January 1996 to December 2002. The first and second series are available at the US Census Bureau web site [35]. The third series was extracted from a real Brazilian payroll. The fourth series corresponds to the monthly number of employed persons in Australia, February 1978 – April 1991, and is available in a time series library [34]. The correlograms of the two first series, which are not depicted here, indicate that they are non-stationary

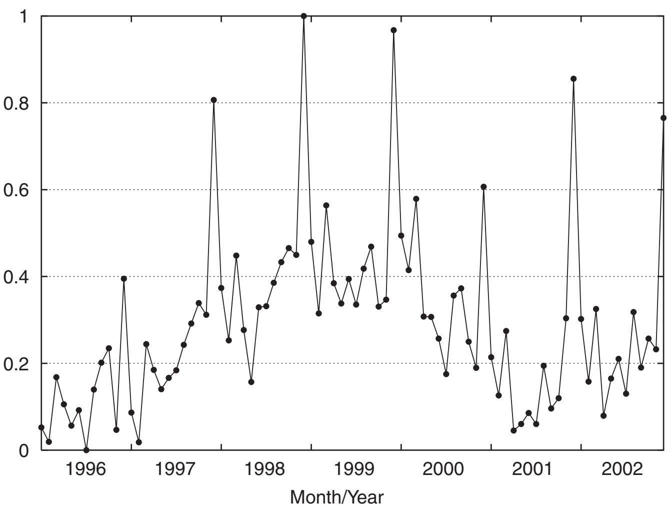

Fig. 4. USA computer and software stores sales time series.

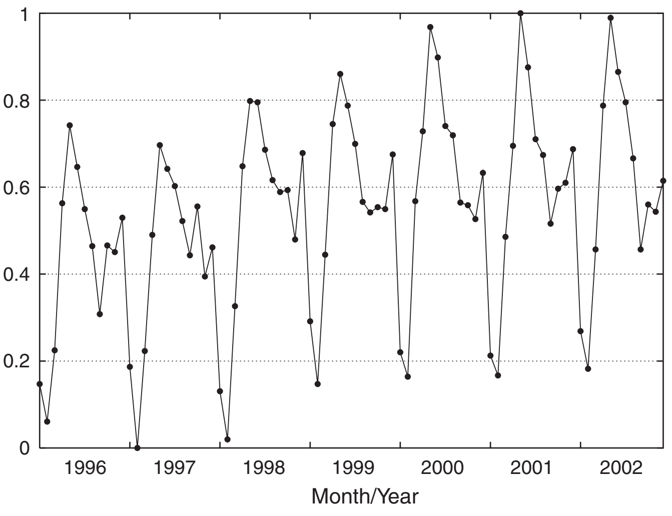

Fig. 5. USA hardware stores sales time series.

and seasonal with period 12. On the other hand, the two last series are non-seasonal and non-stationary.

In the experiments both the original and the differenced versions of each time series are considered. Differencing is carried out in order to obtain a stationary version of a non-stationary time series [4]. This technique is important to improve the prediction capability of time lagged feedforward networks (TLFNs), such as the MLP [8]. Given a time series $\{x_1, ..., x_N\}$, a differenced time series $\{y_2, ..., y_N\}$ is obtained by $y_t = x_t - x_{t-1}$. Each time series, original or differenced, is normalized between 0 and 1 before being used for training and testing the neural networks. It is important to stress that we firstly normalize the training portion of each time series using the following equation:

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{2}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the *training* portion of the time series. Subsequently, the validation and test sets are normalized using Eq. (2) as well. That is, the validation and test sets are normalized considering the minimum and maximum values of the training data.

Experiments were carried out by using both MLP and Elman networks. For each architecture and each series, topologies with both one and two hidden layers were tried. One hidden layer networks used 3, 7, 14 and 28 hidden units whereas networks with two hidden layers had 2, 3, 4 and 5 hidden units. The use of networks with a single hidden layer is more common in time series forecasting problems [39]. Nevertheless, we also considered here networks with two hidden layers because it was shown that in some cases they can yield improved forecasting performance [39]. In each case, time delay orders of 2, 4, 6 and 12 were used [8]. These architectures and topologies were also used in a previous work [21]. Zhang et al. have reported that the time delay order, which correspond to the number of inputs of the neural network, is one of the parameters which mostly influence forecasting performance [39]. Nevertheless, there is no systematic way to determine the value of this parameter, therefore, they should be obtained experimentally for a given time series. It is recommend to use small values, such as the one adopted in our simulations [39]. Furthermore, some

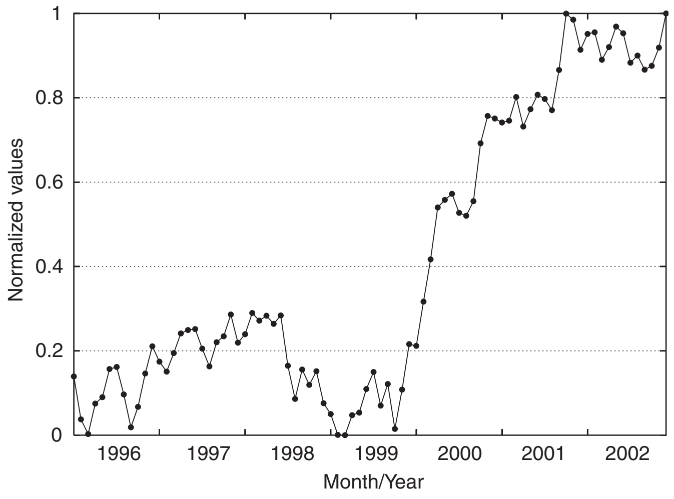

Fig. 6. Earning 2 time series.

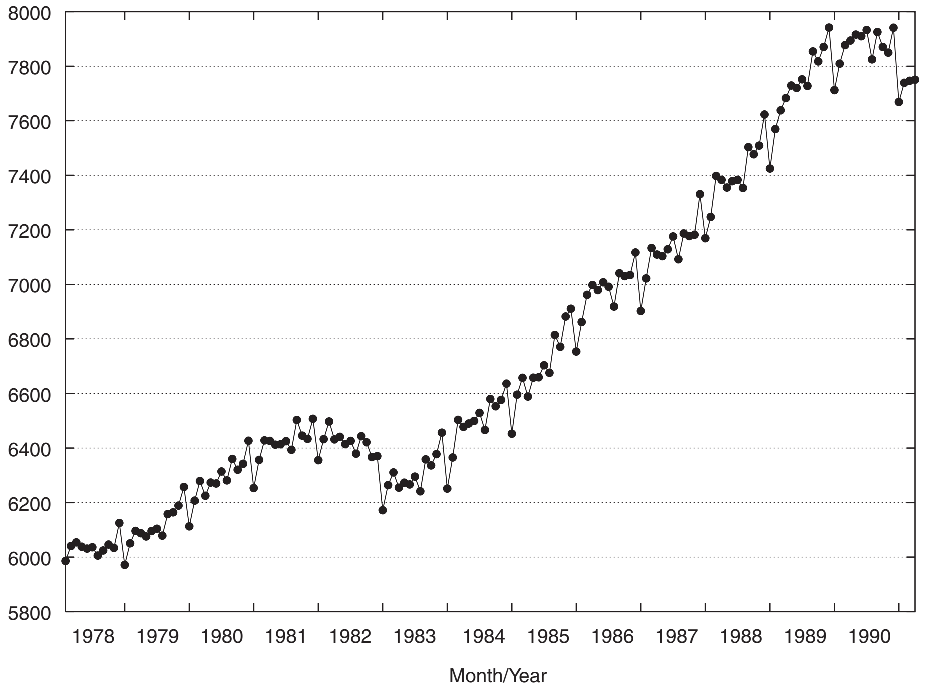

Fig. 7. Empper time series.

authors suggest the use of time delay order of 12 for monthly time series, such as the ones we consider here [39].

Both MLP and Elman networks were trained using the Rprop algorithm with default parameters [27]. Early stopping with the GL_5 criterion from Proben 1 was used to avoid overfitting [26]. For each series, the 12 last months were used as test set. The validation set was formed by the previous 12 months and the remaining data points formed the training sets.

Each architecture and topology was trained 10 times. The best architecture for each time series is the one that obtains the smallest mean square error (MSE) on the validation set. The best architecture is used to build forecasting with 95% confidence intervals for one step ahead predictions. In this case, $p = (1 - 0.95)/2 = 0.025$. Each network was trained 10 times, in order to increase the size of the error collection, as described in Section 3.3. The prediction for each month in the test set was given by the mean prediction across the 10 runs. The robust confidence intervals were computed considering the errors collected on both the training and validation sets.

4.1.2. Results

Elman networks have obtained the best forecasting results in all four time series considered in this subsection. Table 2 shows the best Elman topology and mean and standard deviations for the MSE on the test set for each time series. For the first, second, and fourth time series, the best results were obtained using differenced versions of the series whereas for the third one the use of the original time series resulted in the best forecasting.

Table 3 shows the number of errors collected to build the robust confidence intervals (n) and the number of sample errors discarded from each extreme of the ordered error collection ($n \times p - 1$). In each case, the errors were collected from both training and validation sets in 10 runs of the experiments with different weights and bias initializations. For example, for the first differenced series, which has 83 data points, using delay line of order 12 and 12 points for the test set, results in $83 - 12 - 12 = 59$ points for the training and validation sets. The number of sample errors in 10 runs is, therefore, 590 in this case. The number of samples discarded from each extreme for 95% confidence intervals is $590 * 0.025 - 1 = 13$.

Figs. 8–11 depict real and predicted values in the test sets for the *software*, *hardware*, *earning 2*, and *empper* time series, respectively. In each of these graphics the robust confidence intervals for the predictions are also depicted. Recall that the test sets for all series do not contain novelties. These graphics show that the real values from the test sets all lie within the confidence intervals boundaries for the predictions, as is desirable. Some of the real values are very near the respective predicted value, but others are more distant, all being in the limits created by the confidence intervals for predictions.

A novelty detection method based on robust confidence intervals would indicate a novelty if the real value lies outside the confidence interval, as described in Section 3.4. The experiments reported here showed that these intervals, built from the errors on training and validation sets, can correctly bound the region of normality. Furthermore, these intervals could be used to indicate the *level of suspicion* associated with a given point in the future. The level of suspicion is directly related to the distance from the value in that point to the limit established by the confidence interval (if the value is outside the normality region). In fraud detection applications, the auditor can use these suspicions levels in order to prioritize investigations. Therefore, one could prioritize investigations on points of the time series (for instance, months) with greater suspicions levels.

4.2. Comparison of single ANN and ANN committee for novelty detection in time series

The method proposed in this paper includes a machine committee, introduced in Section 3.3, which aims to increase the size of the error collection as well as to reduce the variance of the predictions. In this subsection we provide an experimental comparison of our method using the proposed machine committee and the same method

Table 3
Characteristics of error collections

Time series	Number of errors (n)	$n \times p - 1$
Software	590	13
Hardware	590	13
Earning 2	660	15
Empper	1340	32

Table 2
The best Elman topology and performance for each series

Time series	Elman hidden layers/neurons	Order of delay line	Test set MSE	
			Mean	Standard deviations
Software	2/3	12	0.00623	0.00347
Hardware	2/5	12	0.00662	0.00200
Earning 2	1/28	6	0.00508	0.00367
Empper	1/7	12	0.01160	0.00276

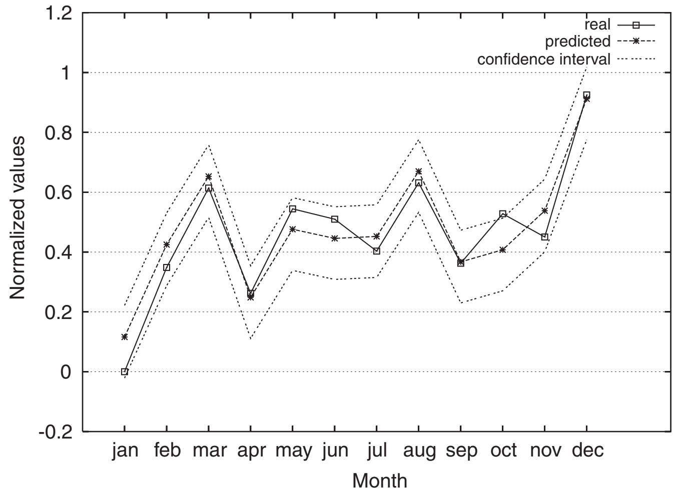

Fig. 8. Robust confidence intervals for predictions in the test set of the *software* series.

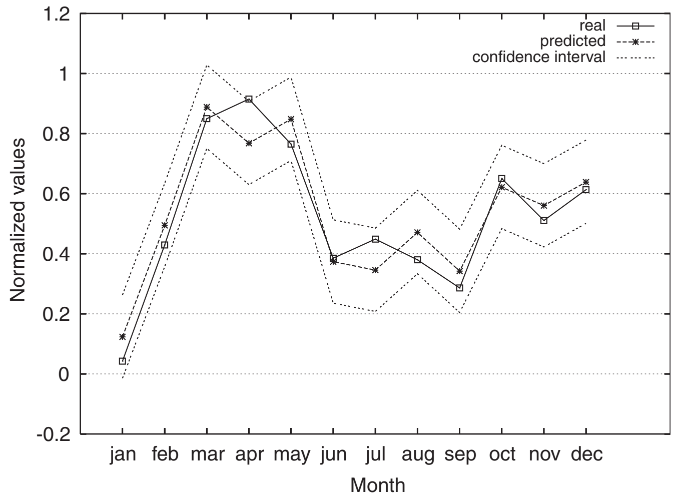

Fig. 9. Robust confidence intervals for predictions in the test set of the *hardware* series.

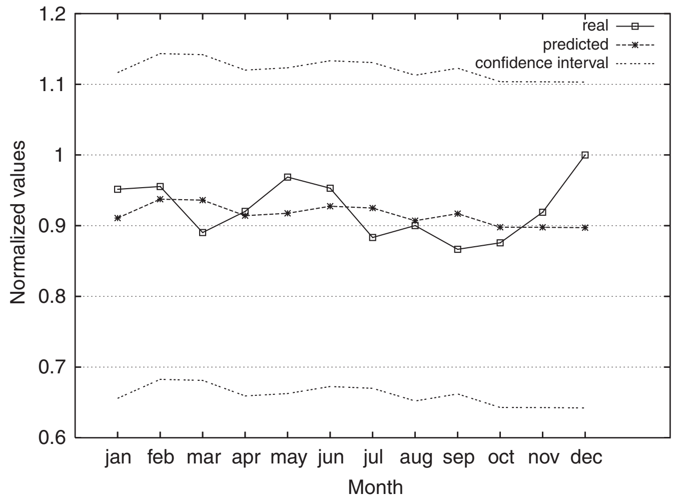

Fig. 10. Robust confidence intervals for predictions in the test set of *earning* 2 series.

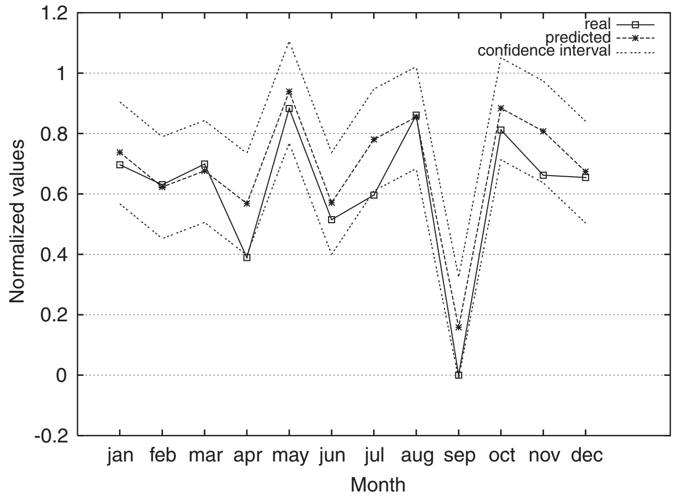

Fig. 11. Robust confidence intervals for predictions in the test set of *empper* series.

based on the robust confidence intervals, yet using a single neural network instead of a machine committee.

The experiments were conducted using the same time series of Section 4.1. We report the results of the experiments using the time series *software* (Fig. 4) and *empper* (Fig. 7). The experimental setup used here was exactly the same employed in Section 4.1. The only difference is that instead of using the machine committee of Fig. 1 we employ here a single Elman neural network as the forecaster.

We have applied a single Elman network with two hidden layers and three neurons in each layer (see Table 2) to predict the values of the test set of the *software* time series with robust confidence intervals. It is well known that the performance of the network depends on the initialization of its parameters (weights and bias). Therefore, we investigate the performance of the method as a function of the seed of the random number generator.

Fig. 12 we show the results for the case where the seed of the random number generator was $s = 2$. Fig. 13 depicts the results using $s = 6$. In Fig. 12 one can observe that the method would indicate a novelty in August since the real value is outside the confidence interval. Such false alarm did not take place when we used the machine committee (see the results of Fig. 8).

When we used $s = 6$ there was a false alarm in December, as the results of Fig.13 show. In addition, the prediction performance in this case (MSE = 0.027566) was much worse than with $s = 2$ (MSE = 0.010501) as well as compared to the machine committee (mean MSE = 0.00623—see Table 2). Notice also that the confidence interval generated in Fig. 13 is much wider than that of Fig. 8 (generated with the machine committee). This means that in this case the system with a single neural network would have a considerable likelihood of producing false positives. For instance, observe that the difference between real and predicted values for January in Fig. 13 is about 0.35. This value is much greater than the width of the

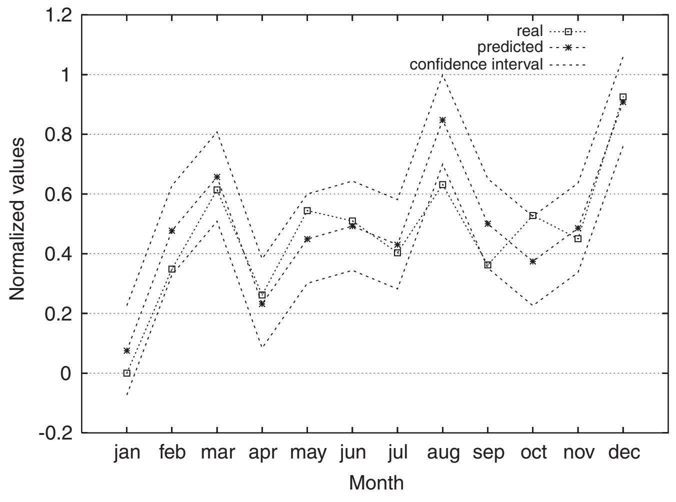

Fig. 12. Robust confidence intervals for predictions in the test set of the *software* series using a single Elman neural network. Seed of random number generator $s = 2$. MSE in test set 0.010501.

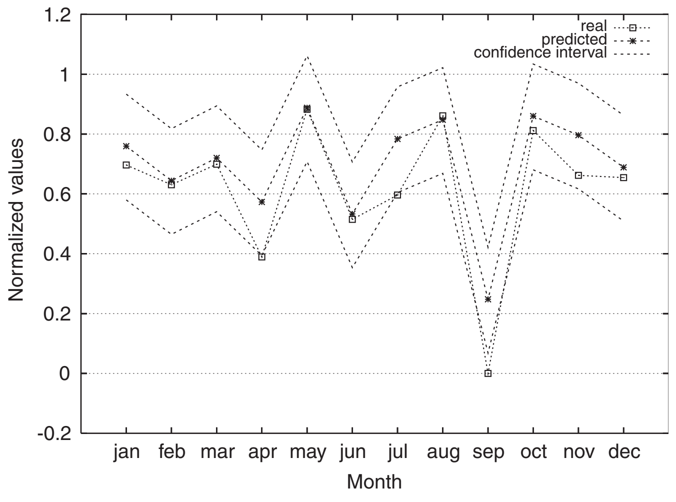

Fig. 14. Robust confidence intervals for predictions in the test set of the *empper* series using a single Elman neural network. Seed of random number generator $s = 1$. MSE in test set 0.013075.

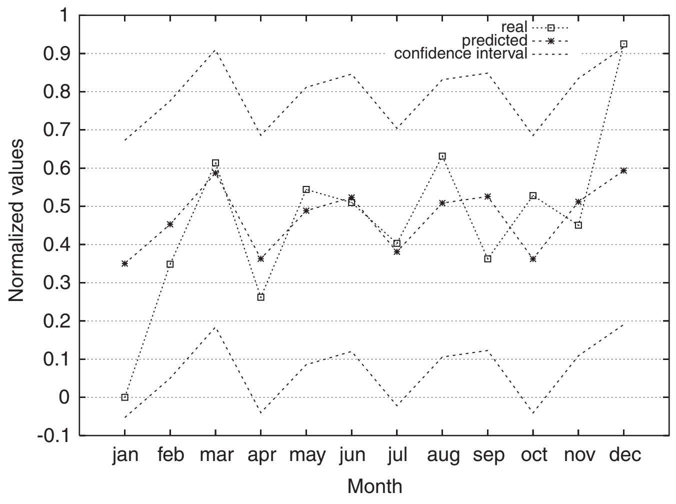

Fig. 13. Robust confidence intervals for predictions in the test set of the *software* series using a single Elman neural network. Seed of random number generator $s = 6$. MSE in test set 0.027566.

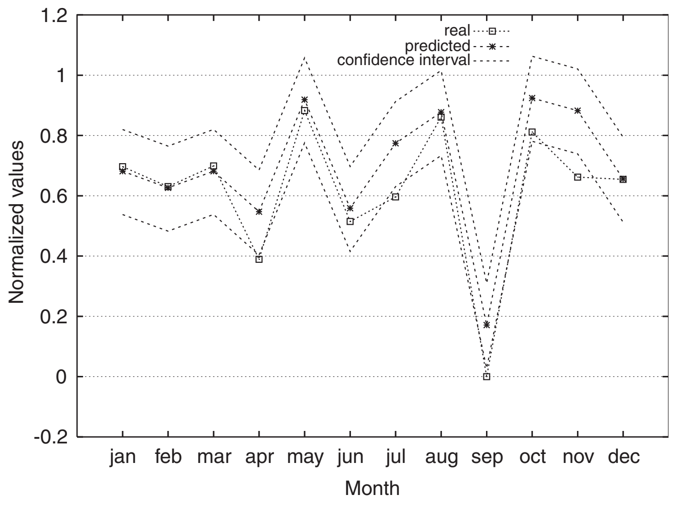

Fig. 15. Robust confidence intervals for predictions in the test set of the *empper* series using a single Elman neural network. Seed of random number generator $s = 5$. MSE in test set 0.012597.

confidence interval produced by the machine committee (about 0.22, see Fig. 8). Hence, the prediction furnished for January in Fig. 13 would be interpreted as a novelty if we had used the machine committee. Yet, the single Elman network accepted it as normal value.

Similar results were obtained for the other time series considered in Section 4.1. For example, Figs. 14 and 15 depict the results obtained using a single Elman network and the time series *empper* with two different values of the seed of the random number generator. In both cases there are false alarms (in Fig. 14 it happens in September whereas in Fig. 15 it happens in July and in November). Conversely, the machine committee has not produced false alarms in the test set as explained in Section 3.3 and depicted in Fig. 11. Furthermore, the machine committee, which integrates the predictions of 10 networks, was able to produce a smaller MSE in the test set in this case as well (MSE = 0.01160—see Table 2 versus MSE = 0.013075—Fig. 14 and MSE = 0.012597—Fig. 15).

We have also conducted several experiments with the seed for the random number generator with values varying from $s = 1$ to 20 for each time series. For most values of s the single network produced one or more false alarms or a confidence interval wider than that built by a machine committee (experiments of Section 3.3). For some values of s the single network produced a result similar to the machine committee. We have also carried out experiments with different values of the seeds of the number generators for the machine committee. The results of these experiments were similar to those reported on Section 3.3. This occurs because the committee integrates the predictions of the networks which makes the method less sensitive to the initial values of the parameters of the networks, thereby producing less variable results compared to those of a single network.

4.3. Experiments with time series with novelties

This subsection describes a set of experiments using time series which have sequences with known novelties. Our intention was to test our proposed method on financial time series. However, we were not able to find such time series with known novelties. There are a few time series benchmark datasets available on the Web such as [34]. Yet we have found only one collection of time series with novelties, which is made available by Eamonn Keogh (available at http://www.cs.ucr.edu/~eamonn/discords). This collection was used on the work by Keogh et al. on the discovery of *discords* in time series [9]. A discord is defined as a sequence within the time series that is least similar to all other sequences [9]. This concept can be used to detect anomalies or changes in the behavior of time series [9]. This time series collection contains time series from domains such as medicine and telemetry monitoring, yet it does not contain financial time series. Therefore, we decided to use two time series from the medical domain in these experiments.

Initially, we consider a time series of a patient's respiration, depicted in Fig. 16. The novelty takes place from time 3050. Before this instant the patient is sleeping and at this instant he takes a deep breath and opens his eyes [9]. It is an obvious novelty which is easily detected visually. Thus, we use it as a first test of our method. We have used only a portion of the time series available in Eamonn's collection (http://www.cs.ucr.edu/~eamonn/discords). In our experiments we have used a subsequence of the original time series from 17500 to 20900 (depicted in Fig. 16).

In these experiments we have normalized the time series as explained in Section 4.1, yet we have not differenced it. The experimental setup was the same as Section 4.1, that is, we considered a machine committee which combined the predictions of 10 Elman neural networks with the same architecture yet with different initial weights and biases. Each Elman network had twelve inputs and two hidden layers, each with three neurons (the same we used for the *software* time series, as shown in Table 2).

In our first simulation we have not considered the whole time series shown in Fig. 16. Instead, we considered a subset of this series which did not include the last portion. This produced a reduced version of the time series without the last 434 points. This was done so that we have a time series without novelties. We aimed to show that our method would identify the region of normality as in the series of Section 4.1. Our simulation with this reduced time series (without novelties) used the last 434 points as the test set, the previous 434 as validation set and the remaining points as the training set. The results of this simulation is shown in Fig. 17. This figure is similar to those shown in Section 4.1. Note that all real values of the time series in the test set lie within the confidence interval built by considering the errors on the training and validation sets. Thus, this shows that the method has worked correctly, since, in this simulation, the test set did not contain novelties.

Now we consider the whole patient's respiration time series shown in Fig. 16. Our simulation used the last 434 for test, the previous 434 for validation and the remaining points for training. Now we have novelties in the test set of this series. The results of this simulation in the test set are depicted in Fig. 18. Note that our method was able to successfully detect the transition of the patient from the sleeping to awake state, since the real values of the time series in Fig. 18 from point 50 ahead are very far from the confidence intervals built by our method.

We have also carried out simulations considering the electrocardiogram (ECG) time series with novelties depicted in Fig. 19. This series corresponds to the first 4600 points of the series depicted in Fig. 10 of [9]. Note that

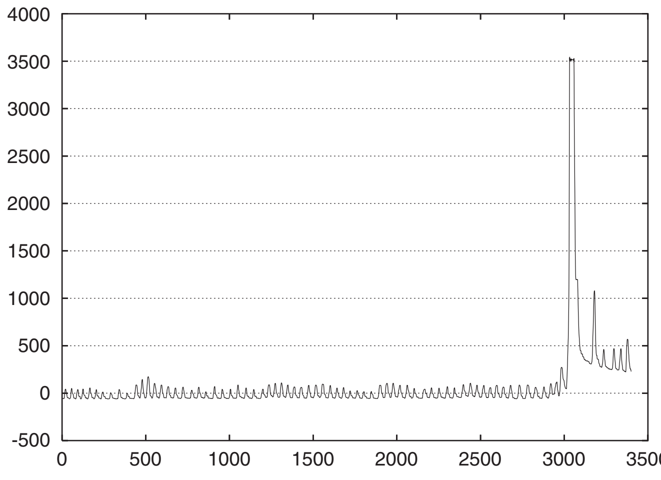
Fig. 16. Time series of a patient's respiration with novelty. From time 1 to 3050 the patient is sleeping and afterwards he takes a deep breath and wakes up.

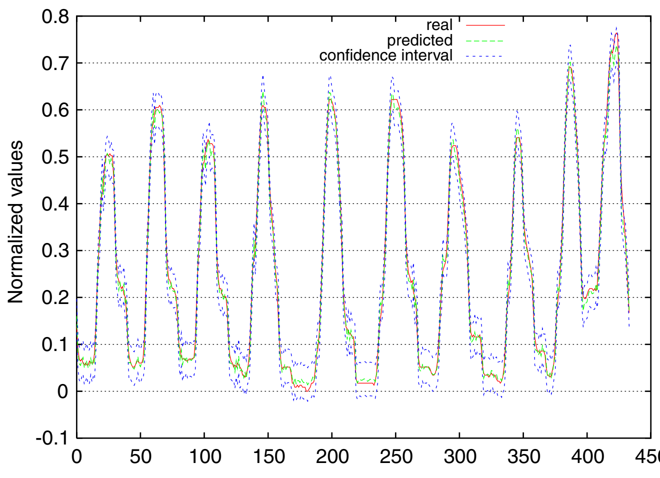

Fig. 17. Robust confidence intervals for predictions in the *test* set of the reduced patient's respiration time series.

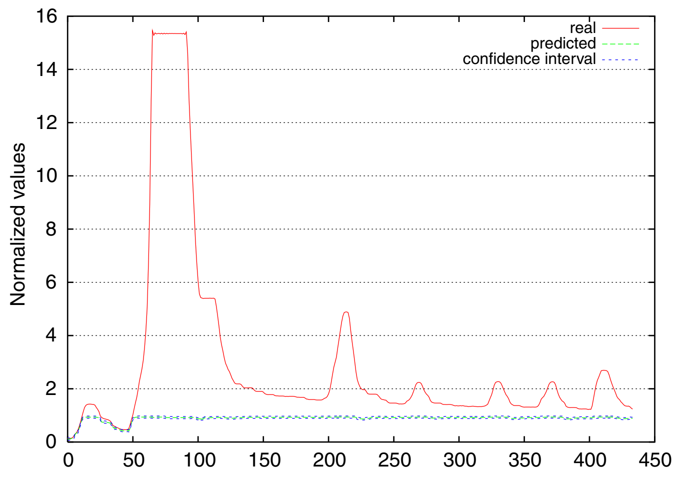

Fig. 18. Robust confidence intervals for predictions in the *test* set (*with novelties*) of the complete patient's respiration time series.

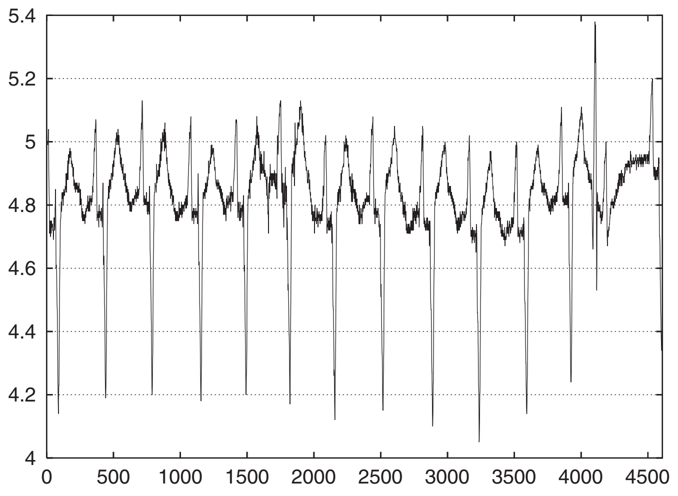

Fig. 19. Electrocardiogram (ECG) time series with novelty.

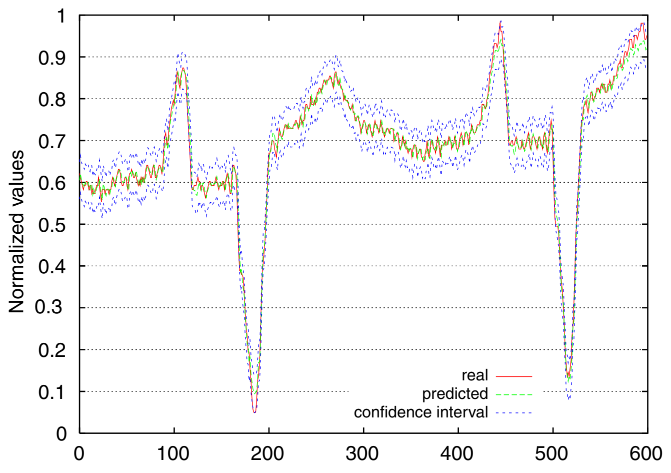

Fig. 20. Robust confidence intervals for predictions in the *test* set of reduced ECG time series.

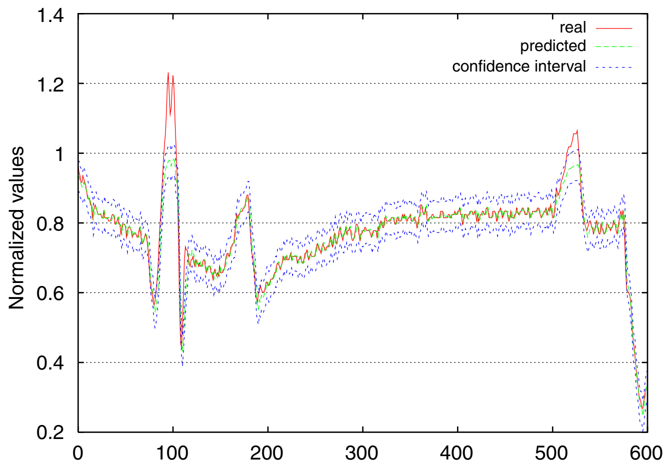

Fig. 21. Robust confidence intervals for predictions in the *test* set (*with novelties*) of the complete ECG time series.

there is an anomaly in this ECG starting in approximately in point 4080. This anomaly has been reported by a cardiologist [9] and is much more subtle than that of the patient's respiration time series (Fig. 16) and therefore it is harder to detect.

The experiments in this case also employed a machine committed comprising 10 Elman neural networks each having 12 inputs and two hidden layers, with three neurons in each hidden layer. As in the case of the patient's respiration time series, we have conducted two simulations. The first simulation was carried out using a reduced version of the ECG time series, which was formed by considering only the first 4000 points of the complete ECG time series shown in Fig. 19. Therefore, the reduced time series contains only normal behavior. Our method was applied to the reduced time series using the last 600 points for the test set, the previous 300 for the validation set and the remaining for training. The results of this simulation are depicted in Fig. 20. It can be observed that the method was able to correctly define the confidence intervals with no false alarm in this case as well.

Finally, Fig. 21 depicts the results of the experiments considering the whole ECG time series shown in Fig. 19. In this case we have used the last 600 points for the test set, the previous 300 for the validation set and the remaining for training, thus, the test set contains an anomaly (novelty) in the ECG. Observe that there are two sequences in Fig. 21 where the real values are clearly outside the confidence intervals for the prediction. The first one is around $t = 100$ whereas the second is around $t = 500$. The test set employed in this simulation had both normal and novelty behavior. The novelty takes place from $t = 80$ and, as shown in Fig. 21 this is immediately detected by our method. There are points between $t = 100$ and 500 where our confidence intervals did not detect novelty. Nevertheless, all this window (from $t = 80$ ahead) should be considered a novelty, since in the previous windows the system has not detected novelty in any point (see Fig. 20). Keogh et al. have analyzed this time series using sequences (windows) of size $w = 600$ [9]. The sequence shown in Fig. 21 was one of the discords found by his method, that is, one of the sequences least similar to the other sequences of the series. In our method we can say that this is also a discord because in this sequence we discovered two subsequences (one around $t = 100$ and the other around $t = 500$) with real values outside the confidence interval whereas in a normal sequence of the time series such as the one of Fig. 20 all real values lay within the confidence interval.

5. Conclusions

We have proposed the use of robust confidence intervals in conjunction with neural network forecasting models in order to detect novelties in time series. A forecasting model is built from the historical values of the time series and the errors on training and validation sets are used to build robust confidence intervals for future predictions. When the model is put to use, a novelty is detected whenever an observed value lies outside the robust confidence interval for the respective prediction. In our method we employ a machine committee in order to increase the number of errors collected for computing the robust confidence intervals, since in practice the time series are very short. Experiments using six real-world time series were reported. Four of them were financial time series without novelties. We also employed two medical time series (electrocardiogram and patient's respiration) with known novelties in our simulations. In the experiments Elman neural networks were used for one step ahead forecasting and 95% robust confidence intervals were built.

The results have shown that the proposed method was able to correctly define the region of normality in the test sets for all time series considered and thus adequately define thresholds for novelty detection. Furthermore, we have demonstrated that the use of a machine committee produces better results than those of a single neural network. In fact, in our experiments with a single neural network the method produced false alarms in many cases as well as wide confidence intervals that would lead to false positives. Finally, experiments with two medical time series with known novelties have shown that the proposed method is able to correctly detect such novelties.

Acknowledgment

The author would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] A. Al-Habaibeh, F. Zorriassatine, R.M. Parkin, M.R. Jackson, Application of infrared technology for quality control of diesel engine glow plugs, J. Eng. Manufacture 219 (6) (2005) 483–490, Proceedings of the Institute of Mechanical Engineers, Part B.

[2] L. Cao, Support vector machines experts for time series forecasting, Neurocomputing 51 (2003) 321–339.

[3] L.J. Cao, H.P. Lee, W.K. Chong, Modified support vector novelty detector using training data with outliers, Pattern Recognition Lett. 24 (14) (2003) 2479–2487.

[4] C. Chatfield, The Analysis of Time Series—An Introduction, fourth ed., Chapman & Hall, London, 1989.

[5] D. Dasgupta, S. Forrest, Novelty detection in time series data using ideas from immunology, in: Proceedings of the Fifth International Conference on Intelligent Systems, 1996.

[6] F. Gonzalez, D. Dasgupta, Neuro-immune and self-organizing map approaches to anomaly detection: a comparison, in: Proceedings of the First International Conference on Artificial Immune Systems, 2002, pp. 203–211.

[7] F. Gonzalez, D. Dasgupta, R. Kozma, Combining negative selection and classification techniques for anomaly detection, in: Proceedings of IEEE Congress on Evolutionary Computation, 2002, pp. 705–710.

[8] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1998.

[9] E. Keogh, J. Lin, A. Fu, HOT SAX: efficiently finding the most unusual time series subsequence, in: Proceedings of Fifth IEEE International Conference on Data Mining (ICDM'05), 2005, pp. 226–233, longer version of the paper and datasets available at ⟨http://www.cs.ucr.edu/~eamonn/discords⟩.

[10] E. Keogh, S. Lonardi, W. Chiu, Finding surprising patterns in a time series database in linear time and space, in: Proceedings of the ACM Knowledge Discovery and Data Mining—SIGKDD'02, 2002, pp. 550–556.

[11] K.-J. Kim, Financial time series forecasting using support vector machines, Neurocomputing 55 (2003) 307–319.

[12] T. Koskela, M. Lehtokangas, J. Saarinen, K. Kaski, Time series prediction with multi-layer perceptron, FIR and Elman neural networks, in: Proceedings of the World Congress on Neural Networks, San Diego, USA, 1996, pp. 491–496.

[13] E. Koskivaara, Artificial neural network models for predicting patterns in auditing monthly balances, J. Oper. Res. Soc. 51 (9) (2000) 1060–1069.

[14] E. Koskivaara, Artificial neural networks in auditing: state of the art, Technical Report, TUCS-TR-509, ISBN 952-12-1120-2, (February 2003).

[15] J. Ma, S. Perkins, Time series novelty detection using one-class support vector machines, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN'2003), vol. 3, 2003, pp. 1741–1745.

[16] M. Markou, S. Singh, Novelty detection: a review, part i: statistical approaches, Signal Process. 83 (2003) 2481–2497.

[17] M. Markou, S. Singh, Novelty detection: a review, part ii: neural network based approaches, Signal Process. 83 (2003) 2499–2521.

[18] S. Marsland, Novelty detection in learning systems, Neural Comput. Surv. 3 (2003) 157–195.

[19] S. Marsland, U. Nehmzow, J. Shapiro, On-line novelty detection for autonomous mobile robots, Robotics Autonomous Syst. 51 (2005) 191–206.

[20] T. Masters, Neural, Novel & Hybrid Algorithms for Time Series Prediction, Wiley, New York, 1995.

[21] A.L.I. Oliveira, G. Azevedo, A. Barros, A.L.M. Santos, A neural network based system for payroll audit support (in Portuguese), in: Proceeding of the IV Brazilian National Artificial Intelligence Meeting, 2003, pp. 487–496.

[22] A.L.I. Oliveira, F.B.L. Neto, S.R.L. Meira, Novelty detection for short time series with neural networks, in: A. Abraham, M. Köppen, K. Franke (Eds.), Design and Application of Hybrid Intelligent Systems, Frontiers in Artificial Intelligence and Applications, vol. 104, IOS Press, 2003, pp. 66–76.

[23] A.L.I. Oliveira, F.B.L. Neto, S.R.L. Meira, Improving novelty detection in short time series through RBF-DDA parameter adjustment, in: Proceedings of International Joint Conference on Neural Networks (IJCNN'2004), vol. 3, IEEE Press, Budapest, Hungary, 2004, pp. 2123–2128.

[24] A.L.I. Oliveira, F.B.L. Neto, S.R.L. Meira, A method based on RBF-DDA neural networks for improving novelty detection in time series, in: Proceedings of the International FLAIRS Conference, AAAI Press, New York, 2004, pp. 670–675.

[25] A.L.I. Oliveira, F.B.L. Neto, S.R.L. Meira, Combining MLP and RBF neural networks for novelty detection in short time series, in: Proceedings of Mexican International Conference on Artificial Intelligence, Lecture Notes in Computer Science, vol. 2972, Springer, Berlin Heidelberg, 2004, pp. 844–853.

[26] L. Prechelt, Proben1—a set of neural networks benchmark problems and benchmarking rules, Technical Report 21/94, Universität Karlsruhe, Germany, 1994.

[27] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, in: Proceedings of the IEEE International Conference on Neural Networks (ICNN 93), vol. 1, 1993, pp. 586–591.

[28] J. Ryan, M.J. Lin, R. Miikkulainen, Intrusion detection with neural networks, in: M.J. et al. (Ed.), Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 1998, pp. 943–949.

[29] B. Scholkopf, R.C. Williamson, A.J. Smola, J. Shawe-Tayler, J. Platt, Support vector method for novelty detection, Neural Inform. Process. Syst. (2000) 582–588.

[30] C. Shahabi, X. Tian, W. Zhao, TSA-tree: a wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data, in: Proceedings of 12th International Conference on Scientific and Statistical Database Management, 2000, pp. 55–68.

[31] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, 2004.

[32] S. Singh, M. Markou, An approach to novelty detection applied to the classification of image regions, IEEE Trans. Knowl. Data Eng. 16 (4) (2004) 396–406.

[33] D.M.J. Tax, One-class classification—concept-learning in the absence of counter-examples, Ph.D. Thesis, Technische Universiteit Delft, 2001.

[34] Time series library ⟨http://www-personal.buseco.monash.edu.au/~hyndman/TSDL⟩.

[35] U.S. census bureau ⟨http://www.census.gov/mrts/www/mrts.html⟩.

[36] G.C. Vasconcelos, An investigation of feedforward neural networks with respect to the detection of spurious patterns, Ph.D. Thesis, University of Kent, Canterbury, 1995.

[37] G.C. Vasconcelos, M.C. Fairhurst, D.L. Bisset, Investigating feedforward neural networks with respect to the rejection of spurious patterns, Pattern Recognition Lett. 16 (2) (1995) 207–212.

[38] J. Yao, C.L. Tan, A case study on using neural networks to perform technical forecasting of forex, Neurocomputing 34 (2000) 79–98.

[39] G. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with artificial neural networks: the state of the art, Int. J. Forecasting 14 (1998) 35–62.

Adriano L. I. Oliveira obtained his B.Sc. degree in Electrical Engineering and M.Sc. and Ph.D. degrees in Computer Science from the Federal University of Pernambuco, Brazil, in 1993, 1997 and 2004, respectively. In 2002 he joined the Department of Computing Systems of Pernambuco State University as an Assistant Professor. He is also a Systems Analyst of Pernambuco Court of Accounts since 1995. He is a Senior Member of the IEEE and a Member of the IEE. His current research interests include machine learning, pattern recognition, data mining, and applications of these techniques to information systems, software engineering, and biomedicine.

Silvio Meira is Professor of Software Engineering at the Federal University of Pernambuco at Recife, Brazil and Chief Scientist of C.E.S.A.R (Recife Center for Advanced Studies and Systems). Professor Meira has authored dozens of articles in learned media and a many others on the impact of information technology in society, in newspapers and magazines. He is member of the Order of Merit in Science and the Rio Branco Order, both awarded by the President of Brazil. Professor Meira is a former president of the Brazilian Computer Society and founding member of the Brazilian Internet Steering Committee. Professor Meira holds a BSEE from ITA (Aeronautics Technological Institute, 1977), a M.Sc. in Computer Science (Federal University of Pernambuco, '81) and a Ph.D. in Computing (University of Kent at Canterbury, 1985). Meira blogs at blog.meira.com.