Image-Based Learning Approach Applied to Time Series Forecasting

K.Ramírez-Amáro[1,2], J. C. Chimal-Eguía*[2]

[1]Technische Universität München,Department of Computer Science Informatik 9,
Boltzmannstrasse 385748 Garching, Germany*mescobar@ing.uc3m.es
[2]Centro de Investigación en Computación, CIC-IPN,
Av. Juan de Dios Bátiz s/n esq. Miguel Othón de Mendizabal,
col. San Pedro Zacatenco, C. P. 07738, México D. F., México
*jchimale@gmail.com

ABSTRACT
In this paper, a new learning approach based on time-series image information is presented. In order to implement this new learning technique, a novel time-series input data representation is also defined. This input data representation is based on information obtained by image axis division into boxes. The difference between this new input data representation and the classical is that this technique is not time-dependent. This new information is implemented in the new Image-Based Learning Approach (IBLA) and by means of a probabilistic mechanism this learning technique is applied to the interesting problem of time series forecasting. The experimental results indicate that by using the methodology proposed in this article, it is possible to obtain better results than with the classical techniques such as artificial neuronal networks and support vector machines.

Keywords: Time series, Imaged based learning approach (IBLA), data representation, Forecasting.

RESUMEN
En este trabajo se presenta un nuevo enfoque para obtener información de una serie de tiempo. Para implementar esta nueva técnica, se ha definido una nueva representación de los datos de entrada de una serie de tiempo. Esta nueva representación está basada en la información obtenida mediante la división del eje de la imagen de la serie de tiempo en cajas. La diferencia entre esta nueva técnica de representación de datos y la forma clásica, se basa en que no es dependiente del tiempo. La nueva representación se ha implementado en una nueva técnica denominada Técnica de Aprendizaje Basada en la Imagen (IBLA por su siglas en inglés) y por medio de un mecanismo probabilístico, esta técnica se aplica al muy interesante problema de predicción en una serie de tiempo. Los resultados experimentales indican que usando esta metodología es posible obtener mejores resultados que los obtenidos por medio de Redes Neuronales Artificiales y Máquinas de Soporte Vectorial.

1. Introduction

Time is both very complex and important in many real-world problems [22]. Its importance is in the fact that almost every kind of data contains time-dependent information, either explicitly in the form of time steps, or implicitly in the way the data is collected from a process that varies with time (e. g., a machine that is worn out, sales influenced by changing tastes or changing contents of web sites). One reason for its complexity is that time can be presented in a multitude of different representations.

Time series of all the possible representations of time are the most commonly used. A time series can be defined as a $x(t)$ which records a sequence of experimental values [2, 3, 8], time values system:

$$x(t_1), x(t_2), x(t_3), ..., x(t_n) \qquad (1)$$

for some interval $t = n$ with $t_0 < t_1 < t_2 ... < t_n$. The time series (TS) are mostly used when the phenomena are not calculated or measured by mathematical models but rely on observation or experiment such as temperature, electronic signals, economic and social data, and so on.

There are two main goals in time series analysis: the first one refers to understanding the behavior of such time series and understanding the underlying theory of data points (where did they come from?, what generated them?). The other one refers to making TS forecasts (predictions). This article is concerned with the complicated but interesting problem of forecasting by means of a new learning technique that uses the structure of artificial intelligence systems.

An artificial intelligence (AI) system must be capable of doing three things: (1) storing knowledge, (2) applying the knowledge to solve problems, and (3) acquiring new knowledge through experience. An AI system has at least two components: knowledge representation and learning [7].

"Knowledge", as used by AI researchers, is just another term for data. Knowledge is represented as a static collection of facts with a small set of general procedures used to manipulate facts. In real-world applications of "intelligent" machines, a good solution depends on a good representation of knowledge [26].

"Learning", in the simple model of machine learning, has four elements: the *environment* supplies some information to a *learning element*. The learning element then uses this information to make improvements in a *knowledge base*, and finally the *performance element* uses the knowledge base to perform its task.

In fact, for the problem of time-series forecasting, there are AI techniques [1] such as artificial neural networks (ANNs) and support vector machines (SVM) that deal with this problem. A complete review of these non-linear techniques applied to time series forecasting can be found in [6, 7]. These AI techniques have the following two components:

Input data representation (knowledge): As Wood stated in 1986, "a good solution depends on a good representation of knowledge" [26]; in this sense, the input data representation used by most of the techniques for time series prediction is often based on information obtained by the time axis in different manners.

For instance, if an ANN technique is implemented then, the input data representation could be the segmentation of the TS: $x(t_1), x(t_2), x(t_3), ..., x(t_n)$, into the following vector:

$$(x(t_i), x(t_{i+d}), ..., x(t_{i+kd})) \quad (2)$$

of size *k* where *d* is the delay between data. Figure 1(a) illustrates the input data representation used by the ANN technique. It is possible to observe that the input data to the ANN is time-dependent; i.e., suppose that the input window size is equal to 5 and the delay is equal to 3, as shown in Figure 1(b) then, the points that belong to the first input window are $x(1)$, $x(4)$, $x(7)$, $x(10)$ and $x(13)$, then the second input data are points $x(2)$, $x(5)$ $x(8)$, $x(11)$ and $x(14)$, and so on. For further information see [4, 7, 9].

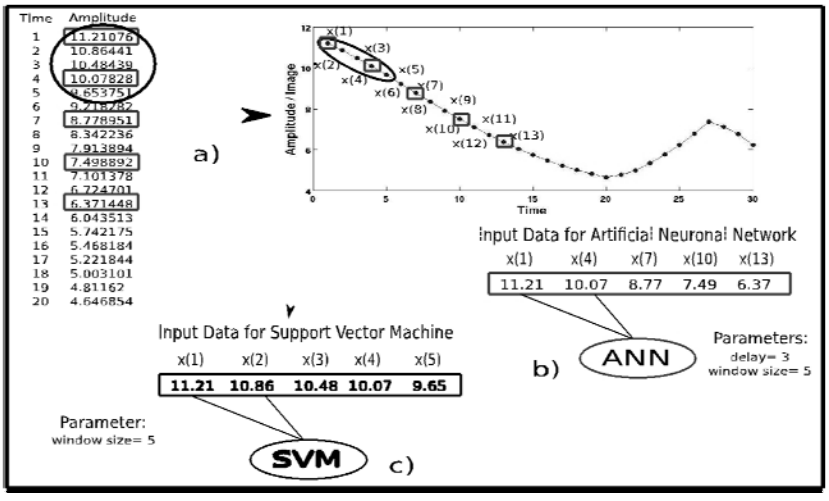

Figure 1. Examples of some input data representations. Firstly, a) shows the values of a time series; then, b) shows the input data for ANN (squares) and its corresponding parameters; finally, c) shows the input window for SVM (circle) and its parameter.

On the other hand, if a SVM approach is implemented, then the TS are split into the following windows:

$$(x(t_i),...,x(t_{i+k-1})) \quad (3)$$

where k is the input window size [22]. For instance, suppose the window size is equal to 5, as shown in Figure 1(c), then the first input window is equal to points $x(1)$, $x(2)$, $x(3)$, and $x(5)$ whilst the second window will be equal to points $x(2)$, $x(3)$, $x(4)$, $x(5)$ and $x(6)$, and so on.

It can be observed that the classical input data representation usually has the following characteristics:

1. The input data is time-dependent.

2. The above techniques do not analyze the whole TS, but they firstly divide the data into small sections following different methodologies.

3. The data window size is defined by one or several parameters and these make the input data windows constant during time.

It is important to notice that if the parameters which define the input data are not optimal, then the learning approach will not obtain the best results.

Learning process: This process is usually called "training" and the forecasting technique learns the behavior of the time series from the training set (input data representation). For instance, the learning process implemented in ANN is based on the idea of mimicking natural brains [13]. The ANN consists of a number of components, the neurons. In the most popular ANN model, the Multi-Layer Perceptron (MLP), neurons are arranged in layers. Connections are present between any neuron in a given layer with any other neuron belonging to the previous layer, and with any other belonging to the next layer. Each connection is weighed by means of a coefficient expressing "how strong the connection is". Neurons in the input layer receive stimuli from the environment and send them to the neurons in the next layer. Each receives inputs from those in the previous layer. Then, the neuron computes its output by applying an activation function of its input value [7, 16].

On the other hand, the SVM are based on the work by Vladimir Vapnik [24] about statistical learning theory. This theory deals with the question of how a function f from a class of functions $(f_\alpha)_{\alpha\in\Lambda}$ can be found; that minimizes the expected risk $R[f]=\int\int L(y, f(x))dp(y,x)dP(x)$ with respect to a loss function L when the distributions of examples P(x) and their classifications P(y,x) are unknown and have to be estimated from examples $(x_i, y_i)_{i\in I}$ [5,12,22].

The learning techniques explained above have some inconveniences; for example, in ANN, it is necessary to decide the optimal architecture and function to train the network. Also, it is necessary to define the appropriate delay between the data. For the SVM, it is necessary to decide the type of kernel function either linear or non-linear and the optimal data window size. An incorrect choice of either the input data representation or the learning process would affect the final result.

Considering how important the input data representation is to the learning techniques, this work presents a new methodology to obtain new input data using image axis information, and by implementing this new methodology, a new learning technique based on statistical procedures is also studied.

Finally, the information obtained from the new input data representation and learning technique is implemented in a new TS forecasting approach. In order to validate the robustness of this approach, the results are compared with some classical techniques, e.g., ANN and SVM.

This paper is organized as follows: in Section 2, the methodology to obtain the new input data representation is presented; in Section 3, the new learning approach is explained; in Section 4, the procedure to generate the new forecasting approach is introduced, and finally, in Section 5, this new technique is validated with experimental results.

2. New Input Data Representation

In order to define this new methodology for input data representation, the principle termed "sub-goals" was studied. This was first published by Newell, Shaw and Simon in 1960 [14]; then, in 1964, it was used by Donald Michie in his MENACE machine [10, 11]. This principle states the next rule: "It is easier to solve a complex problem by dividing it into many easy sub-problems, which are sequentially linked, rather than solve the complete problem". Then, using this idea, the image axis of the TS is divided into these small sections and new divisions are called "boxes".

These boxes are defined by the upper and lower image limits y_1 and y_2, and the upper and lower time limits x_1 and x_2, respectively. In other words, the box is defined as follows:

$$[Box] = [y_{1i}, y_{2i}] \times [x_1, x_2] \forall i = 1,2,3,...,n \qquad (4)$$

where *n* represents the number of divisions made in the image axis. Then, instead of the classical input data representation from which the information is obtained by the time axis of the TS, now the image (amplitude) information is used as a source of the input data.

Using the image axis information, it is possible to observe that the input window size is not constant (see Figure 2). Nevertheless, the following parameters are obtained from the input windows:

Box: This parameter indicates the current box. For instance, points $x(1), x(2), \ldots x(11)$ from the TS of Figure 2(b) belong to $Box = 1$ and $x(12), x(12), x(13)$ and $x(14)$ belongs to $Box = 2$ and so on.

Tendency: This parameter indicates the movement or prevailing movement in a given direction of the TS points. This parameter is a binary value; either the points are decreasing or the points are rising, 0 and 1, respectively.

Visit: This parameter refers to the number of times that a set of points enter a specific box with the same tendency. For instance, observe that the points $x(12)$, $x(13)$ and $x(14)$ in Figure 2(b), visit for the first time $Box = 2$ with tendency=0, then the second visit with the same parameters is made by points $x(29)$ and $x(30)$, and so on. Also, this parameter indicates the number of times that a box is trained.

Then using this new input data representation, the TS is split into the following windows:

$$(x(t_{m(i,j,k),1}), ..., x(t_{m(i,j,k),l})) \qquad (5)$$

Where $x(t_{m(\bullet),1}) > y_{1i}$ and $x(t_{m(\bullet),l}) \leq y_{2i}$, with *i* as the number of the current box, *j* is the tendency of those points, *k* corresponds to the number of visits and *l* indicates the length of the points (see Definition 1) within a box. For example, using the TS observed in Figure 2, the first input window is equal to points $x(1), x(2), x(3), ..., x(11)$ and these points have the following parameters: $Box = 3$, $tendency = 0$ and $visit = 1$. The second window is equal to points $x(12)$, $x(13)$ and $x(14)$ and its parameters are $Box = 2$, $tendency = 0$ and $visit = 1$. Notice, that the number of points within the input window are variable depending on parameters "current box", "tendency" of the points and the "visits".

This new input data representation depends on how the TS image is divided into boxes, i.e., the interval of each box called $interval_{(Box)}$ could be variable as shown in Figure 2. Then, the next subsection is devoted to explain the two different methodologies proposed in this article to divide the image axis.

2.1 Methodology to divide the image axis

The TS image axis is divided into boxes; these boxes have the form of Equation (4) and only the lower (y_1) and the upper (y_2) image bounds of the boxes are defined since the lower and upper time bounds are constant and equal to $x_1 = x(t_1)$ and $x_2 = x(t_n)$. Then, the image interval of each box is defined as

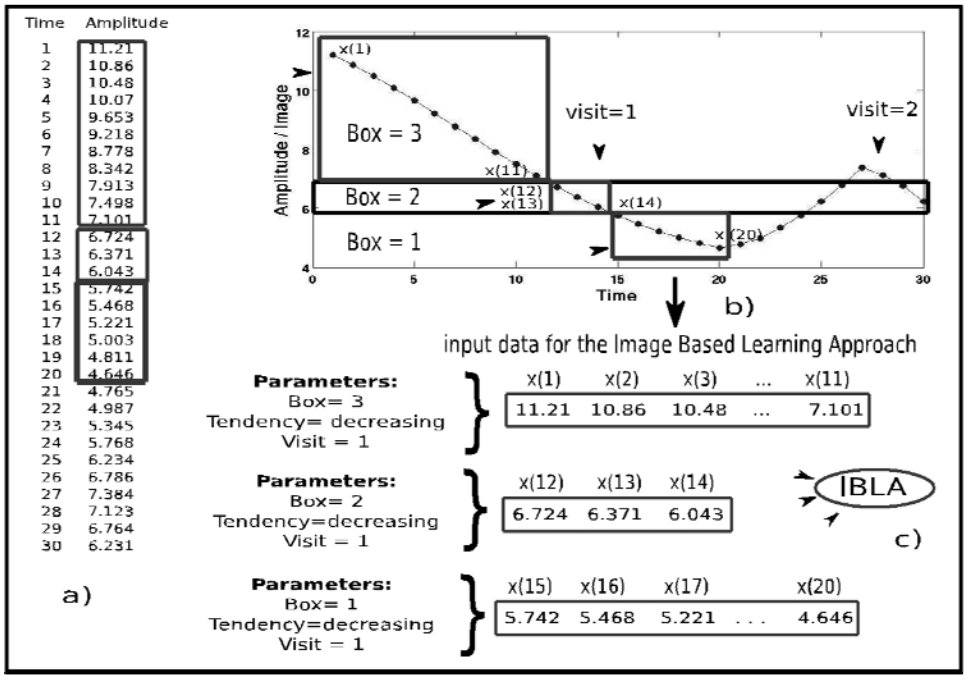

Figure 2. Example of the new input data representation in the form of boxes. In a), the points of the TS are shown; in b), the definitions of Box = 3, Box = 2 and Box = 1, and c) shows its corresponding input windows with tendency=0 and visit=1.

$$image(\text{int}erval_{(box)}) = [y_{1i}, y_{2i}] \forall i = 1,2,3,...,n \quad (6)$$

where y_{1i} is the lower limit and y_{2i} is the upper limit of the Box=*i* and *n* is the total number of divisions made to the TS image axis. For instance, the $image(\text{int}erval_{(1)})$ shown in Figure 2 is equal to $(4,5.8)$ and $image(\text{int}erval_{(2)}) = (5.9,7)$ and $image(\text{int}erval_{(3)}) = (7.1,12)$ .The *width*[1] between the intervals is variable, therefore the input windows are also variables. The new input data representation proposed in this article depends on the divisions of the time series image axis. Then, two different types of divisions are implemented and they are explained as follows;

Width of the division: In order to use this division, it is necessary to define the total number of boxes firstly and then, compute the *width* between the lower and upper bound of each box as follows:

$$Width[y_1, y_2] = \frac{y_2 - y_1}{N}$$

where N is the number of desired boxes. The distribution of points within the boxes is Gaussian. i.e., the first and last box have a lesser number of points than the others. Using this division, the interval of each box has the same width. For instance, if the TS shown in Figure 2 is divided into three boxes (N = 3), then width=2.6. Therefore, the interval of Box(1) will be $image(\text{int}erval_{(1)}) = (4,6.6)$ and the interval of

[1] The *width* of any non empty interval is equal to the difference between the upper and lower limits of the interval, i.e., *width* = $y_2 - y_1$.

Box(2) should be equal to $image(\text{int}erval_{(2)}) = (6.7,9.4)$, and finally, the interval of Box(3) will be $image(\text{int}erval_{(3)}) = (9.5,12)$

Division of the percentage of total points: This division is based on the number of points contained in one box. It splits the TS image using some percentage (x) of the whole data as follows:

$$area_f = size(TS) * (x/100) \quad (7)$$

where *size* (*T S*) stands for the total number of TS points. For instance, the TS of Figure 2 has *size* (*T S*) = 30, and suppose *x* is equal to 10, then $area_f$ = 3 points, i.e., each box should have three points. Notice the number of boxes is computed automatically and, using this division, the number of points within each box is uniform. A complete review of these types of division can be seen in [17].

Hence, the number of boxes and the number of points within each box depends on the type of division. Now, in order to define the box where a point belongs, it is only necessary to verify the lower and upper bounds of the TS image axis.

This new input data representation has the following characteristics:

1. The input data is not time-dependent because now the data is obtained by its image.
2. This technique does not analyze the whole TS, but it firstly divides the data into small sections called Boxes.
3. The window size of the input data is variable.

Since the input data presents the above characteristics, a new learning technique implementation is required as follows.

3. Image-Based Learning Approach (IBLA)

In order to learn[2] the TS behavior by applying the IBLA technique, it is necessary to acquire and accumulate information from the time series by generating a new time series that tracks the behavior of the original.

In order to implement this new learning approach, the next pseudo-code is executed as follows:

Pseudocode to implement the IBLA

```
1  tendency=1, length=0, visit=0;
2  for i=1; i++; to lastPointTimeSeries-1
3     if x(i)>lowerBound(box(j))and
      x(i)<=upperBound(box(j))
4        length++;
5        if x(i)<x(i-1) and i>1
6          tendency=0;
7        else
8          tendency=1;
9        end if
10       {Ask if this box was previously
         trained}
11       if visit>1 {recompute the tracking
         point}
12          lastError=Ue(length,box(j),tendency
            );
13          if lastError>errorMin
14             max=Ua(length,box(j),tendency);
15             min=Ua(length-
               1,box(j),tendency);
16          else if lastError<-errorMin
17             max=Ua(length+1,box(j),tendenc
               y);

18             min=Ua(length,box(j),tendency);
19          else {if the computed point is
            closed to the original}
20             max=Ua(length,box(j),tendency);
21             min=Ua(length,box(j),tendency);
22          end if
23       else {compute the tracking point for
         the first time}
24          max=upperBound(box(j));
25          min=lowerBound(box(j));
26       end if
27       randPoint(i)=rand(min,max);
28       error(i)=                    x(i)-
         randPoint(i);
29       Ua(lenght,box(j),tendency)=randPoint(i
         );
30       Ue(lenght,box(j),tendency)=error(i);
31    end if
32 end for.
```

[2] Learning is the process of acquiring and accumulating knowledge about some task through experience, which allows a better execution of the same task in the future than the last time [23].

For example, observe point x (6) from the TS shown in Figure 3. This point belongs to Box (4) because 0.68 > 0.6 and 0.68 ≤ 0.8; also, point x (7) belongs to *box* **(**4**)** and the tendency of point *x* **(**6**)** rises. This is the first visit to *box***(**4**);** therefore, lines 23-25 from the above pseudo-code are executed as follows: *randPoint* **=** rand **(**0.6**,** 0.8**) =** 0.681. Then, the error between the observed and computed points is obtained. Finally, the computed value and its error are stored into matrices *Ua* and *Ue* (see Figure 3(c)). Observe that point *x* **(**21**)** from the TS shown in Figure 3(a) visits the box for the second time **(**4**)**; after that, line 13 of the pseudo-code is executed and *lastError* **=** −0.001 suppose that *minError* is set to 0.009, then lines 19-21 are executed and the tracking point remains the same, but the *error***(**21**)** changes. Now, in order to compute point *x* **(**22**),** line 13 remains at *lastError* **=** −0.03 and if *minError* **=** 0.009 then, lines 16-18 are performed. Therefore, *min* **=** 0.75 and *max* **=** 0.8; after that, lines 27-30 are executed, thus, *randPoint***(**22**) =** *rand***(**0.75**,** 0.8**) =** 0.78**.**

This learning approach is very important in the problem of forecasting because it is necessary to verify if the learning technique is able to track time series behavior before the prediction is executed. The results obtained using this learning technique are shown in [17]. The learning approach is also important to estimate the number of data needed to learn the point dynamics in order to obtain the best result in the prediction of unknown data. Using this new image-based learning approach (IBLA), it is possible to use additional information obtained from this process in order to solve the forecasting problem. This is explained in the next subsection.

3.1 Additional information

On the basis of the above learning process, matrices Ua and Ue are generated but these matrices are not the only information extracted from the learning process. It is possible to define and extract the following additional important information:

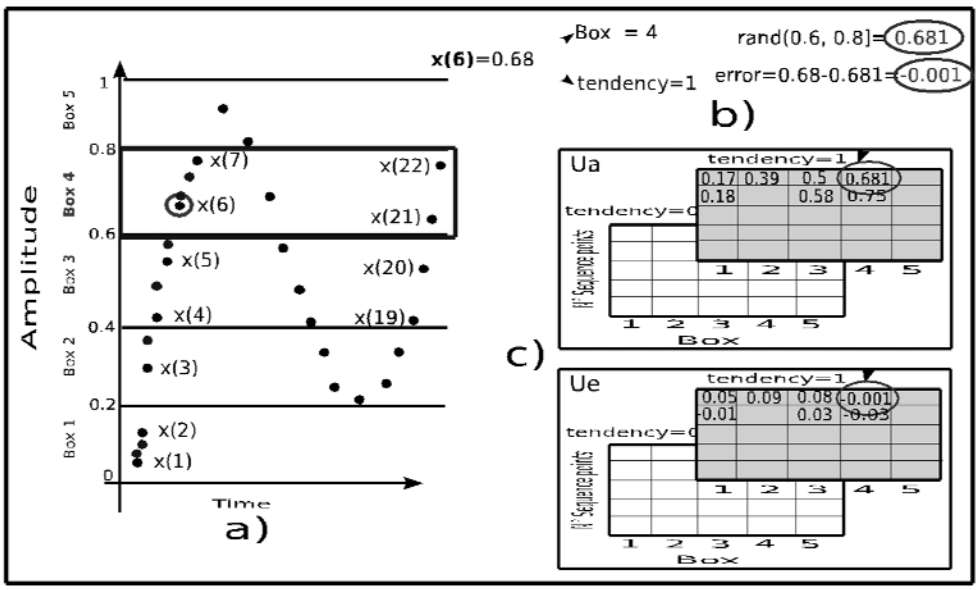

Figure 3. Example of the learning process. In a), the identification of an observed point (dark dots) in a specific box and its tendency is observed; b) shows how the tracking points (light dots in a)) and its errors are computed. Finally, in c) the stored information achieved with this learning process.

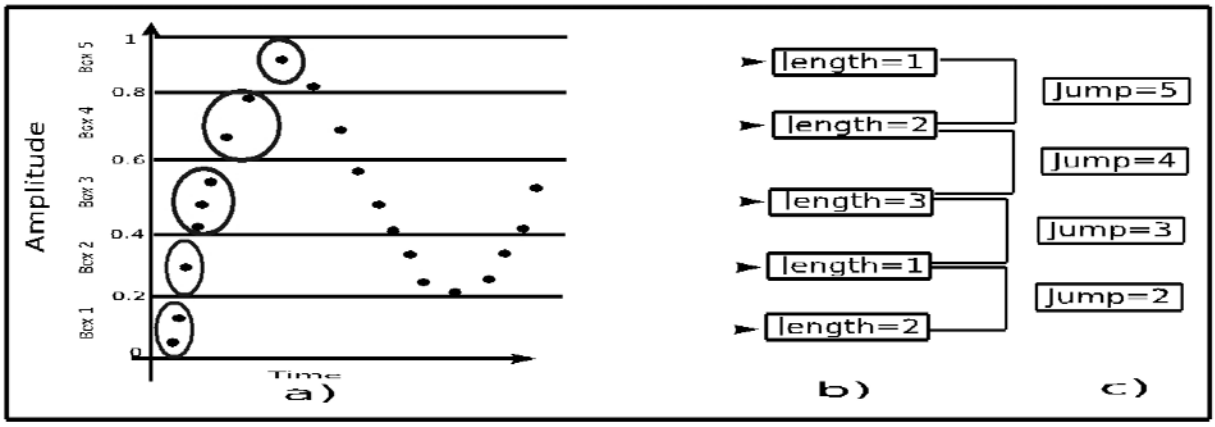

Figure 4. Definition of length and jumps' illustrations. In a), observe that the first points have a rising tendency; in b), the length belonging to those points is shown; and in c), the jumps of the points are observed.

Definition 1. **Length** "l" of points indicates the sequence of points that belong to a specific box during a visit and it presents one of the following conditions (see Figure 4(b):

- if the points rise then,

$$randPo\,\mathrm{int}(t_1) < randPo\,\mathrm{int}(t_2) < ... < randPo\,\mathrm{int}(t_l) \quad (8)$$

- if the points fall then,

$$randPo\,\mathrm{int}(t_1) > randPo\,\mathrm{int}(t_2) > ... > randPo\,\mathrm{int}(t_l) \quad (9)$$

Definition 2. The **jumps** store the next box where a specific set of points jumps.

For example, suppose that *randPoint* **(**t_i**)** belongs to box(p), and the next *randPoint* **(**t_{i+1}**) jumps** to *box*(*n*) without changing its tendency; then **jumps** are equal to *n* (see Figure 4(c)).

The length and jumps are stored in two new matrices called *longs* and *jumps*, respectively.

From these matrices, it is possible to extract the following information:

- the length of a set of points within a box during a visit,
- how many times the boxes are being visited (trained), and
- the next box where the points jump after visiting a specific box.

If matrices *longs* and *jumps* are merged into a single one then it is possible to obtain a new representation of this information. This new matrix is called *longJump* and it is defined as

$$longJump := longJump_{l,\,j,k,t}\ , \quad (10)$$

$\forall 1 \le l \le m$, $1 \le j \le n$, $1 \le k \le o$, $1 \le t \le p$, where *l* is the length of points during a visit, *j* is the next box, *k* is the current box and *t* is the tendency of the points. Matrix *longJump* **(***l*, *j*, *k*,*t***)** stores the number of times that some points show length *l* and jump from box *k* to box *j* with the same tendency *t* (see Figure 5).

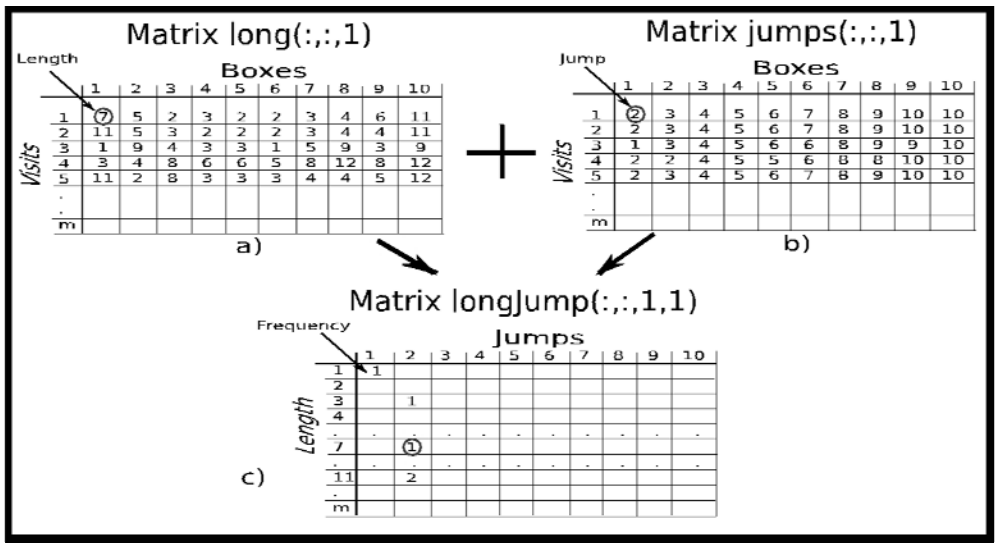

Figure 5. Example of matrix *longJump*. In a), matrix *long* is observed; in b), matrix *jumps* is observed, and finally, c) shows the merging of both matrices. Observe that from matrix *long* **(**1**,** 1**,** 1**) =** 7 (circle) and from matrix *jumps* **(**1**,** 1**,** 1**) =** 2 (circle), this means that when the points have a length equal to 7, they jump to the box equal to 2. It is important to notice that the length is the row and the jumps are the columns of the new matrix *longJump* and this matrix stores the frequency of the points that follows that specific pattern, which is equal to 1 (circle).

For instance, *longJump* **(**11**,** 2**,** 1**,** 1**) =** 2 means that two sets of points repeat the following pattern: the points have a length equal to 11, they jump to box=2 while their initial position was box=1 and they have a rising tendency; observe Figure 5(c).

Also, the patterns obtained from the *longJump* matrix could be expressed by the next decision rule [25]:

> *i f length* **=** 11 and *tendency* **=** 1 and *actualBox* **=** 1
> *then* the next points will jump to *nextBox* **=** 2

Or this rule,

> *i f tendency* **=** 1 and *actualBox* **=** 1 and *nextBox* **=** 2
> *then* the following points will have *length* **=** 11

The above decision rules are used in order to forecast a TS and the next section is devoted to explain the forecasting procedure. The information obtained from the IBLA technique is as follows:

- The *Ua* matrix stores the learned points amplitudes of the points and this information will be implemented in the forecasting approach.
- The *Ue* matrix saves the learned errors of the points, and this information is used to improve the learning technique.
- The *longJump* matrix keeps the frequencies of the decision rules which will be implemented in the TS prediction.

4. Forecasting method

Suppose that an observed TS is $x(t_1)$, $x(t_2)$, $x(t_3)$, ..., $x(t_n)$ and wishes to forecast future values such as $\hat{x}(t_{n+h})$. Integer *h* is called the *lead time* or the *forecasting horizon* (*h,* for horizon) and the forecast of $x(t_{n+h})$ made at time *n* for *h* steps ahead will be denoted by $\hat{x}(h)$. Note that it is essential to specify both the time the forecast is made and the lead time. If $\hat{x}(h)$ only uses information up to time *n*, the resulting forecast is called *out-of-sample* forecasts [4].

A forecasting method is a procedure that takes into account present and past values. The forecasting method implemented in this work is univariate. A univariate method only depends on present and past values of the single series being forecast. Additionally, in this article an iterative prediction which is one of two standard methods to solve a multi-step prediction problem (predicting *h*-steps in advance, typically in a time series) is implemented [4].

The following forecasting method uses the *Ua* and *longJump* matrices obtained from IBLA and this follows the next methodology.

4.1 Methodology

In order to forecast the next TS point(s), it is necessary to follow these steps[3]:

a) Firstly, it is necessary to obtain the information about the last point from the learning process, i.e., *randPoint* **(**t_n**)** (see Figure 6(a)).

b) Then, the box where the future point(s) is (are) going to jump is estimated (see Figure 6(b)).

c) Afterwards, the tendency of the point(s) will be defined (Figure 6(c)).

d) Then, it is necessary to estimate the length of the new point(s) (see Fig. 6d).

e) Finally, the predicted point(s) $\hat{x}(t_{n+l})$ is (are) plotted. Notice that *h* is the lead time and *l* is the number of points predicted for a specific box. For example, in Fig. 6e, the lead time (*h*) is equal to 10 and *l* is equal to two; it is necessary to execute the forecasting process one more time until the 10 predicted points are reached.

In Fig. 6 there is a loop which begins from step e) using the information of the last predicted point $\hat{x}(t_{n+2})$ in order to execute steps b), c), d) and e), until the total number of desired points is reached which in the case of the example in Fig. 6 is equal to *h* **=**10. Now, the above steps are going to be explained in detail.

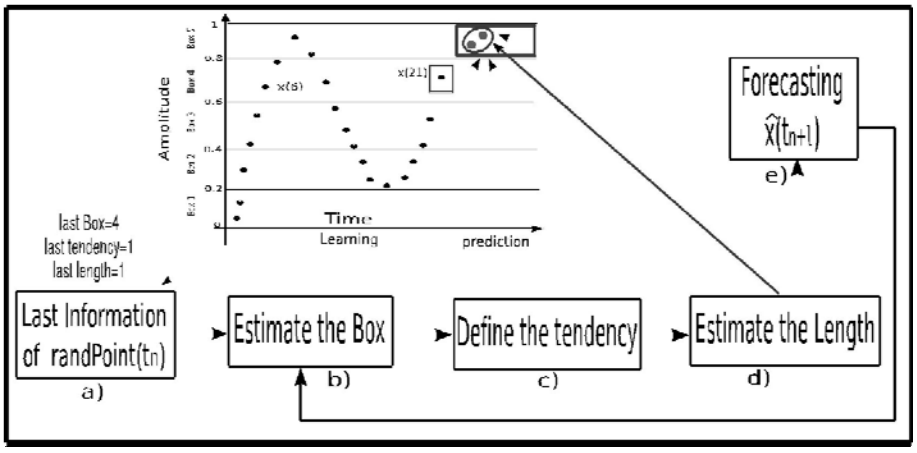

Figure 6. Methodology to forecast the time series points. Firstly, a) shows the last information from the learning process; then, b) shows the estimation of the next box; in c), the definition of the tendency is shown; d) shows the estimation of the length and finally, in e), the forecast points are plotted. Then, the loop begins again with the last predicted point information. Note that when the points prediction is made, there is no learning from experience (reinforcement) like IBLA.

[3] Note, that this forecasting methodology is made only to the out-of-sample data, i.e., there is no modification of any of the matrices used in this process.

4.1.1 Latest information

Firstly, it is necessary to obtain the following information from the latest learned point *randPoint* **(**t_n**)** (see Fig. 6a):

- the last box this point visits,
- the tendency of this point
- its length.

For instance, suppose that the IBLA technique is applied to the TS of Fig. 6 and the last learned point is *x* **(**21**)** then, the information obtained from this point is last box=4, last tendency=1 and length=1. Then, the following step is to estimate the next box as follows.

4.1.2 Estimation of the next box

From the above information, it is possible to generate the next decision rule: *if lastBox* **=** 4 *and tendency* **=** 1 *and length* **=** 1 *then nextBox* **=**?. The first part of the implication is called *antecedent* and the second part is called *successor* [20, 25]. Then, noticing that the successor is missing, it is necessary to implement a mechanism in order to determine the successor which in this case is equal to the next box where the predicted points will jump. In order to do that, the above information is used as coordinates to extract a new vector from the *longJump* matrix, as follows:

$$vector = longJump(length, :, lastBox, tendency) \quad (11)$$

This new vector stores the frequencies that an antecedent reached during the learning process and this new vector also indicates the successor which follows that antecedent. For example, observe the vector extracted from Fig. 7b using the information: box=4, length=1 and tendency=1, i.e. *vector* **=** *longJump***(1, :, 4, 1)**, this vector indicates that when the points have the above information as an antecedent, they follow the successor either to *nextBox*=4, one time or to *nextBox*=5, five times.

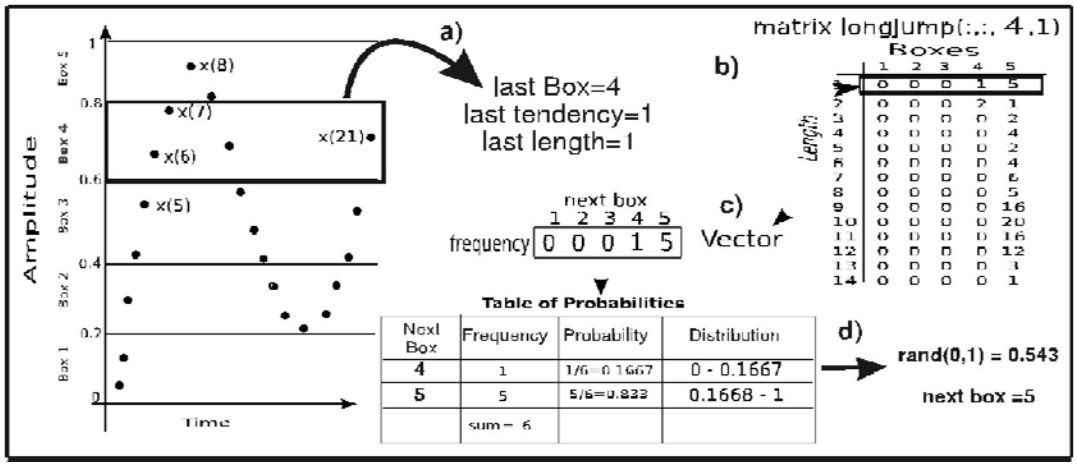

Figure 7. Procedure to estimate the next box. First, in a) the information from the last tracked point is extracted, then in b) the vector that contains the information of the visited boxes is extracted. In c) a probabilistic table is obtained for the visited boxes and finally, in d) the random process implemented to choose the next box.

Then the following information is obtained:

$$if\ lastBox = 4\ and\ tendency = 1\ and\ length = 1 \begin{cases} netBox = 4\ (one\ time) \\ netBox = 5\ (five\ times) \end{cases}$$

In order to estimate the successor, a probabilistic table is computed for every box into the new vector that has a frequency greater than zero. Box=5 is more likely to occur in the future than box=4 due to its frequencies. Then, the successor is chosen by random weight guessing, e.g., suppose that the random value is equal to 0.543 and observe in Fig. 7d that this value correspond to box=5, then the successor is equal to *nextBox* **=** 5.

4.1.3 Define the tendency

It is necessary to determine if the tendency of the predicted points changes or remains the same, once the *nextBox* is estimated. This is achieved as follows:

where *Tend* (*i*) is the tendency of the set of points of the *lastBox*. For example, the next estimated box in Fig. 7 is equal to 5, then the defined tendency for this new box will be "rising".

4.1.4 Length estimation

Once the future box and tendency are estimated, the next step is to define the length of the future set of points, in order to determine how many points will be predicted for that box, i.e., $\hat{x}(t_{n+l})$ and this is done as:

1. From matrix *longJump* a sub-matrix is extracted. This matrix is called *newMatrix*, (see Fig. 8a):

$$newMatrix = longJump\ (:, :, box, tendency) \tag{12}$$

2. The columns from the newMatrix are added and a new vector called *sum* is obtained. The vector rows *sum* represent the future length of points. In other words, this new vector is obtained as (see Fig. 8b):

$$sum(i) = \sum_{i=1}^{m} newMatrix(i,:)$$

$$if\ \ tend(i) = Rai\sin g\ \ and \begin{cases} lastBox = nextBox \Rightarrow Tend(nextBox) = Decrea\sin g \\ lastBox < nextBox \Rightarrow Tend(nextBox) = Decrea\sin g \\ lastBox > nextBox \Rightarrow Tend(nextBox) = Rai\sin g \end{cases}$$

$$if\ \ tend(i) = Decrea\sin g\ \ and \begin{cases} lastBox = nextBox \Rightarrow Tend(nextBox) = Rai\sin g \\ lastBox < nextBox \Rightarrow Tend(nextBox) = Decrea\sin g \\ lastBox > nextBox \Rightarrow Tend(nextBox) = Rai\sin g \end{cases}$$

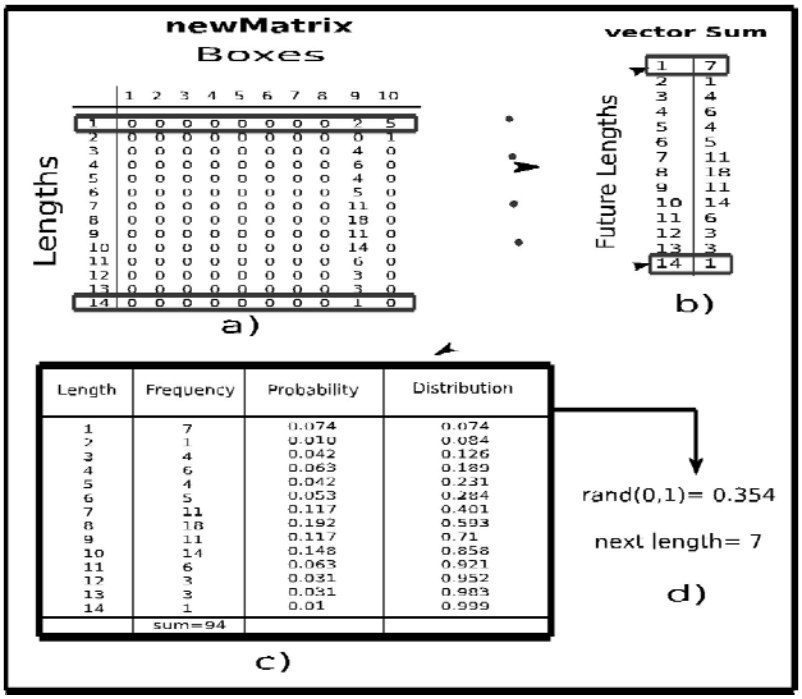

Figure 8. Example of the procedure to estimate the future length of points. In a) the extraction of the matrix newMatrix is observed, in b) the new vector sum formed by the sum of the columns of the newMatrix shown in b), and in c) the table of probabilities of the lengths, and finally, in d) the randommechanism to obtain the future length.

3. As in association with the estimation of the next box, the procedure used to estimate the next length of points is very similar. This means that a probability table of length frequency is generated and then the *nextLength* of the future points set is randomly chosen (see Fig. 8c y d).

4.1.5 Plot the predicted point(s)

Finally, it is necessary to plot the predicted points to the estimated box and this is done by extracting a vector from the *Ua* matrix (obtained during the learning process) using all the estimations as coordinates, i.e., the *nextBox*, the *tendency* and the *nextLength*. Then, that vector is as follows:

$$Ua(nextLength, nextBox, tendency) = \left[\hat{x}(t_{n+i}), \hat{x}(t_{n+(i+1)}), \ldots, \hat{x}(t_{n+(i+l)})\right] \quad (14)$$

where *l* indicates the length of the predicted points and *i* is equal to 1 which is the first time that the forecasting methodology is implemented and is the second time that the forecasting loop is executed then $i = l$ and so on. Finally, that (those) point(s) is (are) plotted (see Fig. 9).

4.1.6 Forecasting evaluation

Finally, in order to measure the performance of this new approach, the root mean square error (RMSE) is implemented:

$$RMSE = \sqrt{\frac{\sum_{t=n+1}^{n+h} (x_t^0 - x_t^F)^2}{h}} \quad (15)$$

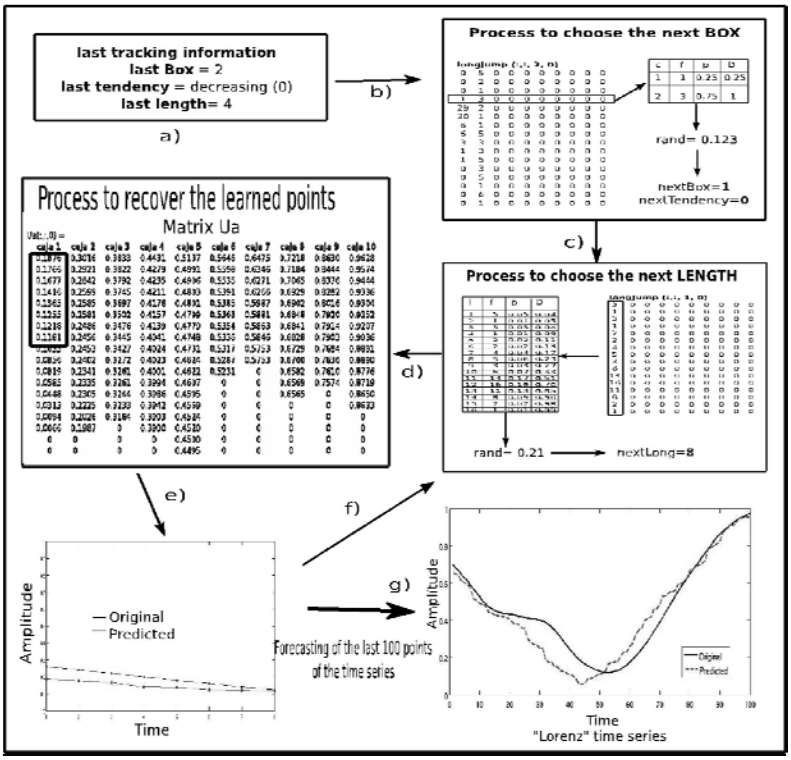

Figure 9. The last learning (tracking) information extracted from the TS is shown in a), in b) the procedure to estimate the future box, in c) the process to estimate the future length, in d) is shown the procedure to recovery the estimated points and finally in e) these new points are plotted and f) shows that this process is repeated until the number of estimated points are equal to the number of points desired and g) shows the last 100 points of the TS "Lorenz".

where x_t^o denotes de observation at time t, x_t^F denotes the forecast of the x_t^o and h indicates the lead time [8]. In order to test the accuracy of this new approach the TS data ($x(t_1), x(t_2), \ldots, x(t_N)$) was split into two parts. In the first part of the data ($x(t_1), \ldots, x(t_n)$), the IBLA is implemented in order to obtain the information to perform the forecasting. In the second part of the data ($x(t_{n+1}), \ldots, x(t_{n+h}) = x(t_N)$), the forecasting methodology is executed and this procedure is done *out-of-sample*.

5. Experimental Results

In order to verify the performance of this new forecasting procedure the obtained results were compared with two of the most used techniques in literature, i.e., the artificial neuronal network and support vector machines.

5.1 Artificial Neuronal Network vs. Image Based Learning Approach

Landassuri-Moreno [9] implemented an input data representation as explained in Section (1) and in order to obtain the best ANN architecture, he implemented a genetic algorithm, also added time series information using Empirical Mode Decomposition (EMD) before analyzing the time series points into the ANN. The author also implemented an iterative prediction. A complete review on this work can be found in [9].

In order to compare the ANN technique with the Image-Based Learning Approach (IBLA), it is necessary to use the same time series, the same forecasting horizon (*h*) which in this case is equal to 30 points, and implement the iterative prediction. Since the author normalized the time series to the interval **[−**1**,** 1**],** it is also necessary to add it in our analysis, but, actually this step can be omitted using the proposed approach. Then Table 1 is obtained with the RMSE produced from both techniques under the same conditions.

The RMSE values obtained from the IBLA are about 30% better than the results obtained with ANN for all the TS presented in Table 1.

5.2 Support Vector Machine vs. Image Based Learning Approach

Bautista-Thompson et al. [3] implemented a multistep prediction with a moving window over the "x" axis. In that research, the authors also used two different kernel functions such as the linear kernel and the radial basis kernel. The best forecasting results obtained in that work were those obtained using the radial basis kernel. The authors performed a modification of the tool MySVM [21] in order to obtain the results presented in [3].

In order to compare the SVM technique with IBLA, it is necessary to use the same time series, the same lead time (*h*) that in this case is equal to 50 points. Since the authors [3] normalized the time series into the interval **[−**1**,** 1**],** it is also necessary to add in this proposed approach.

Fig. 10 shows the results of the last 50 forecast points of the classical time series called "Seno" which is characteristic of the periodic dynamical behavior [3]. Fig. 10a shows the best results obtained in [3] with an RMSE equal to 0.0682. It can be observed that the first 25 predicted points have an error near to zero, but from point 26 there

No.	Time series name	Artificial neuronal networks	IBLA
1	Qperiodic2	0.1048	**0.0895**
2	Qperiodic3	0.2470	**0.1121**
3	Lorenz	0.2571	**0.0246**
4	Ikeda	0.4630	**0.3690**
5	Henon	0.6229	**0.4134**
6	D1	0.2793	**0.1618**
7	Laser	22.407	**0.3902**
8	Lovaina	0.1186	**0.0773**
9	S & P 500	1.7547	**0.2866**
10	Star	0.3577	**0.1841**
11	Brownian Motion	0.8108	**0.0930**
12	Gold Price	6.9932	**0.0378**
13	Nile River	1.1589	**0.6186**
14	Sunspot	25.982	**0.0746**
15	Souther Oscilation	8.068	**0.2355**

Table 1. The lower RMSE obtained by implementing the ANN and IBLA are presented in this table which represents the errors of the last 30 time series forecasting points.

is a delay between the original point and the predicted, and that error is propagated to the next predicted points. That is one of the disadvantages of the iterative predictor, i.e., once an incorrect predicted point is computed, this error will be propagated during the next predictions. Fig. 10b shows the results obtained using IBLA to the same TS observing that the error between the original points and the predicted one is near to zero and the RMSE produced is equal to 0.0044 with no delay between the forecast points and the original.

Fig. 11 shows the results of the last 50 forecast points of the time series called “Tent” [15] which is characteristic of chaotic dynamical behavior and is generated by a linear map [3]. Fig. 11a shows the best result obtained from the SVM implementation and the RMSE is equal to 0.20708. It is possible to observe that the forecasting is not accurate. Fig. 11b the result obtained with IBLA shows that the first 35 predicted points have an error near to zero, but the next four predicted points despite having an incorrect predicted amplitude, its tendency is preserved. The prediction of point 40 onwards had the same amplitude and tendency value, i.e., the error from point 40 to 50 is near zero once again.

This means that the prediction error is not propagated during future predicted points. Finally, in Table 2 the best results obtained using the SVM and IBLA under the same conditions in order to forecast the last 50 points of a different TS is observed. All the results obtained using the IBLA have a better performance than using the SVM technique. This is possible since the SVM has the error propagation problem.

6. Concluding Remarks

In this paper a new forecasting approach of a learning technique based on a new methodology to obtain the input data representation of the time series using the information of its image axis is presented. This new approach can predict points from time series with a different behavior that ranges from the easiest time series (periodical) to a more complex behavior (chaotic).

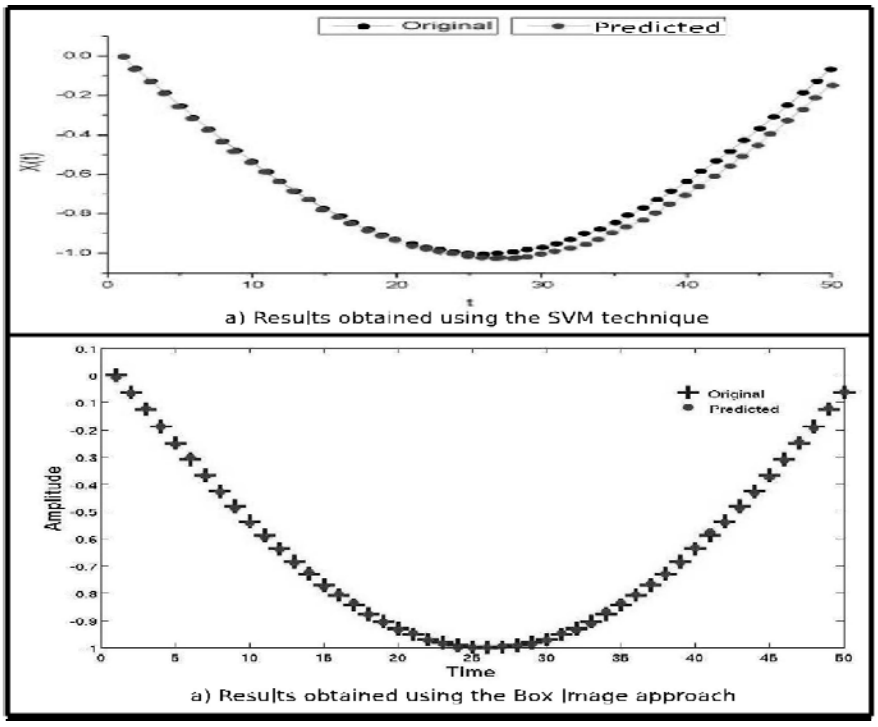

Figure 10. a) shows the results obtained from the implementation of SVM, and b) shows the results obtained using the Image-Based Learning Approach proposed in this article.

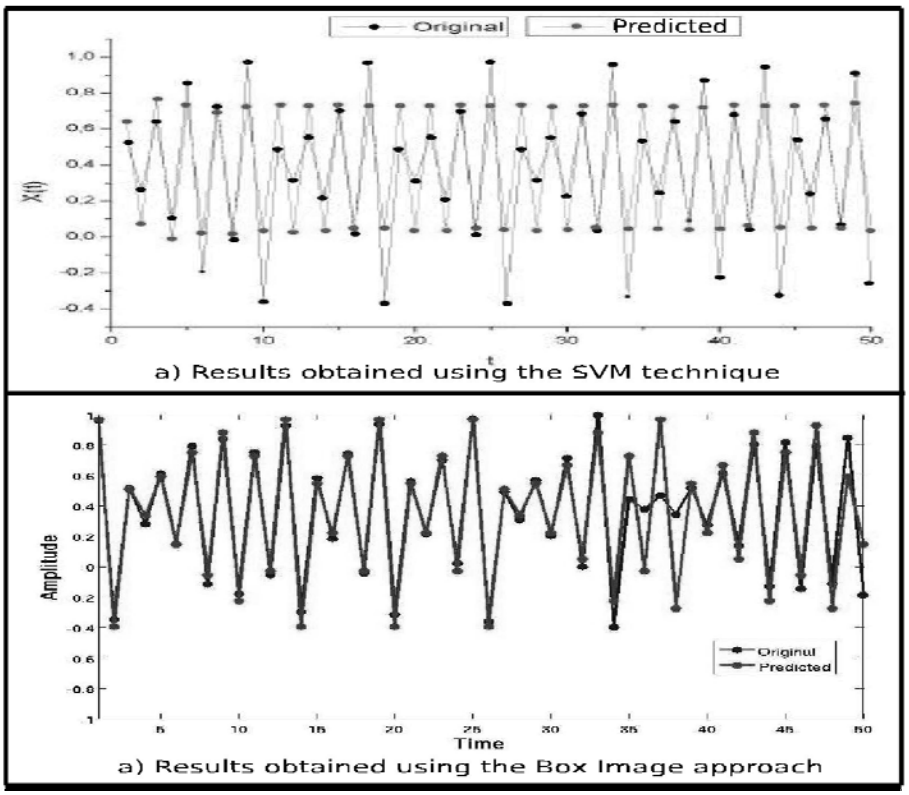

Figure 11. a) shows the results obtained using the SVM technique, and b) shows that the IBLA technique exhibiting the best results. As we can observe the last has the best results.

The novelty of this approach lies in the fact that a new input data representation is proposed which is not time-dependent like most of the classic techniques used in literature at the present time.

The forecasting approach proposed in this paper has two main characteristics: the first refers that despite not predicting an accuracy amplitude, the predicted point preserves the tendency of the original. The second characteristic is that in spite of implementing an iterative prediction technique, an incorrect prediction of some points does not propagate its error through future predicted points. These characteristics are due to the fact that the points estimation into a box are independent of the other boxes.

By using the methodology proposed in this paper the results obtained using IBLA have better forecasting results than the approaches of artificial neuronal networks and support vector machines.

The information obtained from the training process can also be used to expand these results to complex networks. Any natural phenomena can be represented as a time series, and it is possible to apply this approach to several phenomena.

No.	Time series name	Support Vector Machine	IBLA
1	Seno	0.0682	**0.00002**
2	Vanderpol	0.70549	**0.00464**
3	Qperiodic2	0.17429	**0.00349**
4	Qperiodic3	0.46989	**0.04423**
5	Mackey & Glass	0.5007	**0.24519**
6	Logistic	1.08828	**0.53587**
7	Lorenz	0.17389	**0.00067**
8	Rossler	0.4131	**0.01557**
9	Tent	0.20708	**0.03503**
10	A1	7.37899	**0.04300**
11	Laser	0.32574	**0.14717**
12	Lovaina	0.16296	**0.01268**

Table 2. The RMSE is observed in this table implementing the SVM technique and the Image Approach in order to predict the next 50 points of the following time series.

Acknowledgments

This research was partially supported by Consejo Nacional de Ciencia y Tecnologia (CONACyT, Mexico), COFAA-IPN and EDI-IPN*

References

[1] Armstrong J. & Fildes R. (2006). Making Progress in Forecasting. *International Journal of Forecasting*, 22, 433-441.

[2] Box, G.E.P. & Jenkins, G.M. (1970). *Time Series Analysis, Forecasting and Control*. San Francisco: Holden-Day (revised edn., 1976).

[3] Bautista-Thompson, E., Guzmán-Ramírez & Figueroa-Nazuno J. (2004). Predicción de Múltiples Puntos de Series de Tiempo Utilizando Support Vector Machines. *Computación y Sistemas*, 7(3), 148-155, ISSN 1405-5546.

[4] Chatfield C. (2000). *Time Series Forecasting*. United Kingdom: Chapman & Hall/crc, ISBN: 1-58488-063-5.

[5] Cristianini N. and Shawe-Taylor J. (2001). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge: Cambridge University Press. ISBN-10: 0521 780195.

[6] De Gooijer J. G, Hyndman R.J. (2006). 25 Years of Time Series Forecasting. International Journal of Forecasting, 22, 443-473.

7] Haykin S. (1999). *Neural Networks: a Comprehensive Foundation*. Prentice Hall Press. ISBN: 0-13-273350-1.

[8] Kantz, H. & Schreiber, T. (1997). *Nonlinear Time Series Analysis*. Cambridge: Cambridge University Press.

[9] Landassuri-Moreno V. M.: *Predicción de Series de Tiempo con Descomposición Empírica en Modos, Algoritmo Genético y Redes Neuronales Artificiales*. México, D.F. Centro de Investigación en Computación (CIC-IPN). Maestría en Ciencias de la Computación.

[10] Michie, D. (1963). Experiments on the mechanization of game-learning Part I. Characterization of the model and its parameters. *Computer Journal*, 6, pp. 232-236.

[11] Michie, D. & Chambers, R. A. (1968). Boxes: An Experiment in Adaptive Control. eds. E. Dale and D. Michie. *Machine Intelligence* 2, pp. 125-133.

[12] Mitchell T. (1997). *Machine Learning*. Mc Graw Hill Education. ISBN-10: 007154671.

[13] McCulloch, W. & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 7, 115 - 133.

[14] Newell A., Shaw J.C. & Simon H.A. (1960). A Variety of Intelligent Learning. eds. C Mar-shall, Yovits and Scott Cameron. London: Pergamon: *in a general problem solver in Self-organizing Systems*, pp. 153-189.

[15] Ott E. (2000). *Chaos in Dynamical Systems*. Cambridge University Press, Cambridge.

[16] Paliouras G., Karkaletsis V. and Spyropoulos C. D. (Eds.) (2001). Machine Learning and Its Applications, Advanced Lectures. Lecture Notes in Computer Science 2049 Springer 2001, ISBN 3-540-42490-3

[17] Ramírez-Amaro K. (2007). Técnica de Aprendizaje de Series de Tiempo Estacionarias a partir de la Información de su Imagen . México, D.F. Centro de Investigación en Computación (C IC-IPN). Maestría en Ciencias de la Computación.

[18] Ramírez-Amaro K. & Chimal-Eguía J.C. (2007). New Approach to Time Series Tracking. 27th International Symposium on Forecasting, New York City, USA.

[19] Ramírez-Amaro, K. & Figueroa-Nazuno, J.G. (2006). Empleo de la Técnica Mapa Recurrente en el Análisis de los Indices de Teleconexión Atmosféricos . México, D.F. Centro de Investigación en Computación, I.P.N. Reporte Técnico. ISBN: 97 0-36-0330-0.

[20] Ramírez-Amaro K., Ortega-González V. and Figueroa-Nazuno J. (2006). “Automatic Generation of Hypotheses Using the GUHA Method”. Data Mining an d Information Systems. Research in Computing Science 22, pp. 31-41. ISSN: 1870-4069.

[21] Ruping S. (2000). mySVM-Manual. University of Dortmund, Lehrstuhl Informatik 8, http://www-ai.cs.unidortmund.de/SOFTWARE/MYSVM/.

[22] Ruping S.(2001). SVM Kernels for Time Series Analysis. En Klinkenberg R. and S. Ruping, A. Fick, N. Henze, C. Herzog, R. Molitor y O. Schroder, editores, LLWA 01 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivit¨at in series Forschungsberichte des Fachbereichs Informatik der Universit¨at Dortmund, pp. 43-50.

[23] Simon H. A. (1983). Why Should Machines Learn? In R. Michalski, J. Carbonell, and T. Mitchell, editors, Machine Learning: An Artificial Intelligence Approach, pp. 25-38. Tioga, Palo Alto, CA.

[24] Vapnik V. (1998). Statistical Learning Theory, John Wiley & Sons, New York.

[25] Witten I. H. and Frank Eibe (2005). Data Mining: Practical machine learning tools and techniques. Second edition, editor: Morgan Kaufmann. ISBN-10: 0120884070.

[26] Woods, W.A. (1986). Important issues in knowledge representation, Proceedings of the Institute of Electrical and Electronics Engineers, vol. 74, pp. 1322-1334.