Neural Comput & Applic (2009) 18:1021–1029
DOI 10.1007/s00521-009-0266-y

ORIGINAL ARTICLE

A comparative evaluation of nonlinear dynamics methods for time series prediction

Francesco Camastra · Maurizio Filippone

Received: 28 March 2008 / Accepted: 6 April 2009 / Published online: 29 April 2009

Abstract A key problem in time series prediction using autoregressive models is to fix the model order, namely the number of past samples required to model the time series adequately. The estimation of the model order using cross-validation may be a long process. In this paper, we investigate alternative methods to cross-validation, based on nonlinear dynamics methods, namely Grassberger–Procaccia, Kégl, Levina–Bickel and False Nearest Neighbors algorithms. The experiments have been performed in two different ways. In the first case, the model order has been used to carry out the prediction, performed by a SVM for regression on three real data time series showing that nonlinear dynamics methods have performances very close to the cross-validation ones. In the second case, we have tested the accuracy of nonlinear dynamics methods in predicting the known model order of synthetic time series. In this case, most of the methods have yielded a correct estimate and when the estimate was not correct, the value was very close to the real one.

1 Introduction

Time series prediction is the problem of determining the future values of a given time series. This problem has great importance in several fields, ranging from finance (e.g. predicting the future behavior of stock markets) to engineering (e.g. estimating future electrical consumption). A key problem in time series prediction is to fix the *model order*, namely the number of past samples required to model the time series adequately. In principle, *cross-validation* [6, 26] is the simplest solution, just picking the model order which gives the lowest prediction error. The computational cost of cross-validation, however, may be high, an estimate of the model order can be helpful, either to be used directly, or to narrow down the range for cross-validation. The problem of estimating the order of a model is crucial not only in time series analysis, but also in data modeling in general. Popular approaches to model order estimation are based on concepts of Information Theory and Statistics, such as the Akaike Information Criterion (AIC) [2], methods based on the Minimum Description Length (MDL) principle [21], Bayesian Information Criterion (BIC) [23], and hypothesis testing [3, 16]. Although these methods have a solid theoretical foundation, the model assumption plays a key role, and a deviation from the real model can lead to a degradation of their performances [9]. A robust approach to the model order estimation in the framework of uncertain statistics can be found in [9]. In this paper, we focus on the time series prediction problem. In this context, the False Nearest Neighbors algorithms has already been used in several time series prediction applications [4] to estimate the model order. Here, we compare a set of methods for model order estimation based on nonlinear dynamics theory, namely Grassberger–Procaccia, Kégl, Levina–Bickel, and False Nearest Neighbors algorithms. The model order is used to carry out the prediction by means of a *support vector machine for regression* (*SVM*) [20, 22, 30]. We assess the performances of the estimators on the basis of the prediction accuracy achieved by the SVMs trained using the selected model

F. Camastra (✉)
Department of Applied Science, University of Naples Parthenope, Centro Direzionale Isola C4, 80143 Naples, Italy
e-mail: francesco.camastra@uniparthenope.it

M. Filippone
Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK
e-mail: m.filippone@dcs.shef.ac.uk

order. We also investigate the effectiveness of nonlinear dynamics methods comparing their performances with those obtained by cross-validation. Finally, we test the accuracy of nonlinear dynamics methods in predicting the known model order of synthetic time series. The paper's structure is as follows: in Sect. 2, a description of the nonlinear dynamics methods investigated is provided; in Sect. 3 some experimental results are reported; in Sect. 4 conclusions are drawn.

2 Nonlinear dynamics methods

We consider a time series $x(t)$, with ($t = 1,2,\ldots, \ell$). An autoregressive model describes a time series as: $x(t) = f(x(t-1),\ldots, x(t-d+1)) + \varepsilon_t$. The function $f(\cdot)$ is called the *skeleton* of the time series [12, 29], the term ε_t represents the noise. The key problem in the autoregressive models is to fix the model order $(d-1)$. Nonlinear Dynamics methods can be used for the *model reconstruction* of the time series. This is performed by the method of delays [7, 19]. The time series can be represented as a series of a set of points $\{\vec{X}(t) : \vec{X}(t) = [x(t), x(t-1), \ldots, x(t-d+1)]\}$ in a d-dimensional space. If d is adequately large, between the manifold[1] M obtained by the points $\vec{X}(t)$ and the attractor U of the dynamic system that generated the time series, there is a diffeomorphism.[2] The *Takens–Mañé embedding theorem*[3] [15, 27] states that to obtain a faithful reconstruction of the system dynamics, it must be

$$2S + 1 \leq d \tag{1}$$

where S is the dimension of the system attractor U and d is called the *embedding dimension* of the system. Hence, it is adequate to measure S to infer the embedding dimension d and the model order $d-1$. A unique definition of the dimension has not been given yet. Popular definitions of set dimensions are the *Box-counting* dimension [18] and the *Correlation dimension* [8]. In the next sections, we shall discuss three methods to estimate attractor dimension (Grassberger–Procaccia, Levina–Bickel and Kégl methods) and a method to estimate the embedding dimension, without using Takens–Mañé embedding theorem (False Nearest Neighbors method).

2.1 Kégl's algorithm

Let $\Omega = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_\ell\}$ be a set of points in $\mathbb{R}^n$ of cardinality ℓ. We denote with $v(r)$ the number of the boxes (i.e. hypercubes) of size r needed to cover Ω. It can be proven [18] that $v(r)$ is proportional to $\left(\frac{1}{r}\right)^d$, where d is the *dimension* of the set Ω. This motivates the following definition. The Box-counting dimension (or *Kolmogorov capacity*) D_B of the set Ω [18] is defined by

$$D_B = \lim_{r\to 0} \frac{\ln(v(r))}{\ln(\frac{1}{r})} \tag{2}$$

where the limit is assumed to exist.

Recently, Kégl [14] has proposed a fast algorithm (*Kégl's algorithm*) to estimate the Box-counting dimension. The algorithm was originally proposed for intrinsic data dimensionality estimation. In this paper, we propose a novel application of Kégl's algorithm, consisting the dimension estimation of an attractor. Kégl's algorithm is based on the observation that $v(r)$ is equivalent to the cardinality of the maximum independent vertex set $MI(G_r)$ of the graph $G_r(V,E)$ with vertex set $V = \Omega$ and edge set $E = \{(\vec{x}_i, \vec{x}_j) | \mathrm{d}(\vec{x}_i, \vec{x}_j) < r\}$. Kégl has proposed to estimate $MI(G)$ using the following greedy approximation. Given a data set Ω, we start with an empty set $\mathcal{C}$. In an iteration over Ω, we add to $\mathcal{C}$ data points that are at distance of at least r from all elements of $\mathcal{C}$. The cardinality of $\mathcal{C}$, after every point in Ω has been visited, is the estimate of $v(r)$. The Box-counting dimension estimate is given by:

$$D_B = -\frac{\ln v(\mathrm{r}_2) - \ln v(\mathrm{r}_1)}{\ln r_2 - \ln r_1} \tag{3}$$

where r_2 and r_1 are values that can be set up heuristically.

It can be proven [14] that the complexity of Kégl's algorithm is given by $O(D_B \ell^2)$, where ℓ and D_B are the cardinality and the dimensionality of the data set, respectively.

2.2 Grassberger–Procaccia algorithm

The *Correlation dimension* [8] of a set Ω is defined as follows. If the *correlation integral* $C_m(r)$ is defined as:

$$C_m(r) = \lim_{\ell\to\infty} \frac{2}{\ell(\ell-1)} \sum_{i=1}^{\ell} \sum_{j=i+1}^{\ell} I(\|\vec{x}_j - \vec{x}_i\| \leq r) \tag{4}$$

where I is an *indicator function*,[4] then the Correlation dimension D of Ω is:

[1] A manifold (http://en.wikipedia.org/wiki/Manifold) is a mathematical space in which every point has a neighborhood which resembles Euclidean space, but in which the global structure may be more complicated. In a one-dimensional manifold (e.g. a line, a circle) every point has a neighborhood that looks like a segment of a line. In a two-dimensional manifold (e.g. a plane, the surface of a sphere) the neighborhood looks like a disk. $\mathbb{R}^n$ is a n-dimensional manifold.

[2] M is *diffeomorphic* to U iff there is a differentiable map $m : M \mapsto U$ whose inverse m^{-1} exists and is also differentiable.

[3] Takens–Mañé embedding theorem is a consequence of a *Whitney Embedding Theorem* [32] stating that a generic map from an S-dimensional manifold to a $(2S+1)$-dimensional Euclidean space is an *embedding*, i.e. the image of the S-dimensional manifold is completely unfolded in the larger space. Therefore, two points in the S-dimensional manifold do not map to the same point in the $(2S+1)$-dimensional space.

[4] $I(\lambda)$ is 1 iff condition λ holds, 0 otherwise.

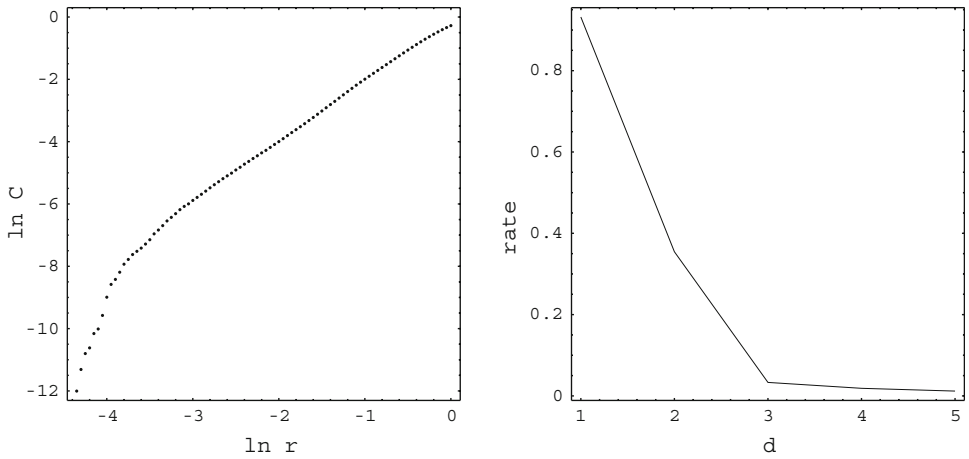

Fig. 1 The Grassberger–Procaccia (**a**) and False Nearest Neighbors (**b**) algorithms on Data Set A

$$D = \lim_{r \to 0} \frac{\ln(\mathrm{C_m(r)})}{\ln(r)}. \tag{5}$$

It can be proven that the Correlation dimension is a lower bound of the Box-counting dimension. The most popular method to estimate Correlation dimension is the *Grassberger–Procaccia algorithm* [8]. This method consists in plotting $\ln(C_m\ (r))$ versus $\ln(r)$. The Correlation dimension is the slope of the linear part of the curve (see Fig. 1a). For increasing values of d, one can notice a saturation effect. The limit value is the correlation dimension.

The computational complexity of the Grassberger–Procaccia algorithm is $O(\ell^2 s)$ where ℓ is the cardinality of the data set and s is the number of different times that the integral correlation is evaluated, respectively. This implementation was adopted for the experiments reported in Sect. 3. However, there are efficient implementations of the Grassberger–Procaccia algorithm whose complexity does not depend on s. For these implementations, the computational complexity is $O(\ell^2)$.

2.3 Levina–Bickel algorithm

The Levina–Bickel algorithm provides a maximum likelihood estimate of the correlation dimension. Like the Kégl's algorithm, the Levina–Bickel algorithm was proposed for the estimation of the intrinsic data dimensionality. Therefore, the application of the Levina–Bickel algorithm for estimating the dimension of an attractor is a novelty. The Levina–Bickel algorithm derives the maximum likelihood estimator (MLE) of the intrinsic dimensionality D of a manifold $\Omega = (\vec{x}_1, \ldots, \vec{x}_\ell)$. The dataset Ω represents an embedding of a lower-dimensional sample, i.e. $\vec{x}_i = g(Y_i)$ where Y_i are sampled from an unknown smooth density f on $\mathbb{R}^D$ with $D < n$, g is a smooth mapping. This last assumption guarantees that close datapoints in $\mathbb{R}^D$ are mapped to close neighbors in the embedding.

That being said, we fix a data point $\vec{x} \in \mathbb{R}^n$ assuming that $f(\vec{x})$ is constant in a sphere $S_{\vec{x}}(r)$ centered in $\vec{x}$ of radius r and we view Ω as a homogeneous Poisson process in $S_{\vec{x}}(r)$. Given the inhomogeneous process, $\{P(t,\vec{x}),\ 0 \le t \le r\}$

$$P(t,\vec{x}) = \sum_{i=1}^{\ell} I(\vec{x}_i \in S_{\vec{x}}(t)), \tag{6}$$

which counts the datapoints whose distance from $\vec{x}$ is less than t. If we approximate it by means a Poisson process and we neglect the dependence on $\vec{x}$, the rate $\lambda(t)$ of the process $P(t)$ is given by:

$$\lambda(t) = f(\vec{x})V(D)Dt^{D-1}, \tag{7}$$

where $V(D)$ is the volume of a D-dimensional unit hypersphere.

The Eq. 7 is justified by the Poisson process properties since the surface area of the sphere $S_{\vec{x}}(t)$ is $\frac{\mathrm{d}}{\mathrm{d}t}[V(D)t^D] = V(D)Dt^{D-1}$. If we define $\theta = logf(\vec{x})$, the log-likelihood of the process $P(t)$ [25] is:

$$L(D,\theta) = \int_0^r \log\lambda(t)\mathrm{d}P(t) - \int_0^r \lambda(t)\mathrm{d}t. \tag{8}$$

The equation describes an exponential family for which a maximum likelihood estimator exists with probability that tends to 1 as the number of samples ℓ tends to infinity. The maximum likelihood estimator is unique and must satisfy the following equations:

$$\frac{\partial L}{\partial \theta} = \int_0^r \mathrm{d}P(t) - \int_0^r \lambda(t)\mathrm{d}t = P(r) - e^{\theta}V(D)r^D = 0. \tag{9}$$

$$\frac{\partial L}{\partial D} = \left(\frac{1}{D} + \frac{V'(D)}{V(D)}\right)P(r) + \int_0^r \log t \, dP(t) + - e^{\theta} V(D) r^D \left(\log r + \frac{V'(D)}{V(D)}\right) = 0. \quad (10)$$

If we plug the Eq. 9 into the Eq. 10, we obtain the maximum likelihood estimate for the dimensionality D:

$$\hat{D}_r(\vec{x}) = \left[\frac{1}{P(r,\vec{x})}\sum_{j=1}^{P(r,\vec{x})} \log\frac{r}{T_j(\vec{x})}\right]^{-1}, \quad (11)$$

where $T_j(\vec{x})$ denotes the Euclidean distance between $\vec{x}$ and its jth nearest neighbor.

Levina and Bickel suggest to fix the number of the neighbors k rather than the radius of the sphere r. Therefore the estimate becomes:

$$\hat{D}_k(\vec{x}) = \left[\frac{1}{k-1}\sum_{j=1}^{k-1} \log\frac{T_k(\vec{x})}{T_j(\vec{x})}\right]^{-1}. \quad (12)$$

The estimate of the dimensionality is obtained averaging on all points of the data set Ω, that is:

$$\hat{D}_k = \frac{1}{\ell}\sum_{i=1}^{\ell} \hat{D}_k(\vec{x}_i) \quad (13)$$

The estimate of the dimensionality depends on the value of k. Levina and Bickel suggest to average over a range of values of $k = k_1,\ldots, k_2$ obtaining the final estimate of the dimensionality, i.e.

$$\hat{D} = \frac{1}{k_2 - k_1 + 1}\sum_{k=k_1}^{k=k_2} \hat{D}_k. \quad (14)$$

Regarding the computational complexity, the Levina–Bickel algorithm requires a sorting algorithm,[5] whose complexity is $O(\ell \log \ell)$, where ℓ denotes the cardinality of the data set. Hence the computational complexity for estimating $\hat{D}_k$ is $O(k\ell^2 \log \ell)$, where k denotes the numbers of the neighbors that have to be considered. Besides, Levina and Bickel suggest to consider an average estimate repeating the estimate D_k s times, where s is the difference between the maximum and the minimum value that k can assume, i.e. k_2 and k_1, respectively. Therefore the overall computational complexity of the Levina–Bickel algorithm is $O(k_2 s\ell^2 \log \ell)$.

The Levina–Bickel algorithm provides an optimal estimate of the correlation dimension. This topic has been widely investigated in nonlinear dynamics, e.g. [24, 28]. In particular, Takens proposed an algorithm that provides a maximum likelihood estimate, like the Levina–Bickel algorithm. The difference between the Takens and Levina–Bickel algorithms is the following. The former provides an estimate that depends on a radius r that has to be fixed properly; the latter depends on the number, chosen appropriately, of the neighbors that have to be taken into account for each datapoint of the manifold.

2.4 Method of False Nearest Neighbors

The Kégl, Grassberger–Procaccia and Levina–Bickel algorithms estimate the attractor dimension and compute the model order of the time series by the Takens–Mañé embedding theorem. An alternative approach is proposed by the *False Nearest Neighbors method* (*FNN*) [4, 13]. This method estimates directly the embedding dimension without using the Takens–Mañé theorem. The False Nearest Neighbors method is based on a simple geometric concept. If the dimension d used to reconstruct the attractor is too small, many points that appear *near* will become widely separated when $d + 1$ dimensions are used in the attractor reconstruction. Nearest neighbor points that show this wide separation when comparing their distance in dimension d and $d + 1$ are called *False Nearest Neighbors* in dimension d. Conversely, true nearest neighbors will remain near each other in attractor reconstructions of both d and $d + 1$ dimensions. More formally a pair of points are considered False Nearest Neighbors in dimension d if $\frac{R_{d+1}^2(j)}{R_d^2(j)} > \alpha$ where $R_d(j)$ and $R_{d+1}(j)$ are, respectively, the Euclidean distance between the jth point and its nearest neighbors in d and $d + 1$ dimensions and α is an heuristic threshold. Typical values for α are suggested in [4]. The adequacy of dimension d for reconstructing an attractor can be evaluated by finding for each data point of the attractor the nearest neighbors in dimension d and then evaluating the percentage of False Nearest Neighbors. Then the percentage of False Nearest Neighbors is plotted versus the dimension d. The lowest dimension corresponding to this minimum value of the percentage of False Nearest Neighbors is the embedding dimension. The main operation involved in the method is the nearest neighbor search. The time complexity of finding the nearest neighbor of a point in a d dimensional set of cardinality ℓ, is $O(d\ell)$. Since we are interested in computing the nearest neighbors for all the ℓ points of the set, we can simply use the nearest neighbor algorithm for all the ℓ points, that is $O(d\ell^2)$ in total. It is worth noting, however, that the complexity of this approach can be reduced[6] by storing the data in suitable data structures [31]. Once we obtain the nearest neighbors for the ℓ data points, we need to compute the

[5] The complexity of effective sorting algorithms (e.g. mergesort and heapsort) is $\ell \log \ell$, where ℓ is the number of elements that have to be sorted.

[6] In our implementation, we use the linear search having complexity $O(d\ell^2)$.

Table 1 The False Nearest Neighbors, Kégl, Grassberger–Procaccia and cross-validation method on Data Set A

Algorithm	Attractor dimension	Embedding dimension	Model order	Mean square error
False Nearest Neighbors			3	214.2 (12.83)
Kégl	2.02	∼5	4	138.3 (10.17)
Levina–Bickel	2.35	∼6	5	167.5 (10.44)
Grassberger–Procaccia	2.00	∼5	4	138.3 (10.17)
Cross-validation			4	138.3 (10.17)

Average error is reported in brackets

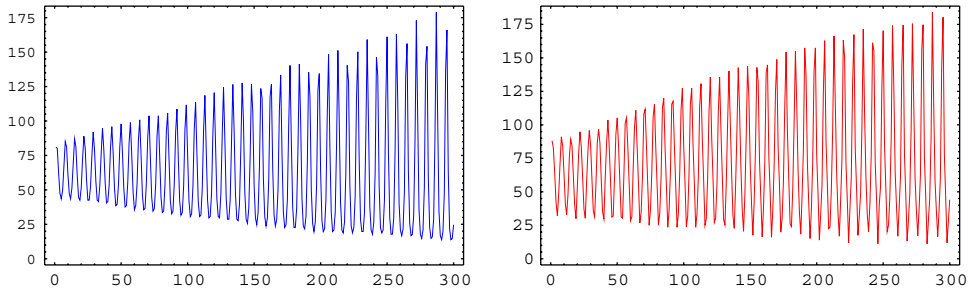

Fig. 2 Data Set A of Santa Fe Competition. The original target data and the results yielded by SVM (model order = 4) are shown on the *left* and the *right*, respectively

distances in the augmented space and compute the percentage of false nearest neighbors; these operations have a complexity that is lower than $O(d\ell^2)$. Finally, the procedure has to be repeated for all the values of d for which we are interested in computing the percentage of False Nearest Neighbors.

It is necessary to remark that in literature [4] when the FNN method is applied, the obtained result is used directly as the model order estimate. In the rest of the manuscript this procedure is used.

3 Experimental results

The False Nearest Neighbors, Grassberger–Procaccia, Kégl and Levina–Bickel algorithms have been tested on three synthetic time series and on three benchmarks of real data.

3.1 Real data time series

The False Nearest Neighbors, Grassberger–Procaccia, Kégl and Levina–Bickel algorithms have been tested on three real data time series, e.g. the *Data Set A* [10] of the Santa Fe time series competition, the *Paris-14E Parc Montsouris* [33] and the *DSVC1* [1] time series.[7] On the real data sets, we have compared the prediction accuracy obtained by cross-validation with the SVM for regression trained using the model order estimated by the presented methods. In these experiments, we have used tenfold cross-validation. We recall that in K-fold cross-validation, the training set is partitioned into K subsets. Of the K subsets, a single subset is retained as the validation data for testing the model, and the remaining $K-1$ subsets are used as training data. The cross-validation process is then repeated K times, i.e. the folds, with each of the K subsets used exactly once as the validation data. The K results from the folds then can be averaged to produce a single estimation.

3.1.1 Data Set A

Data Set A is a real data time series, formed by 1,000 samples, generated by a Lorenz-like chaotic system, implemented by NH_3-FIR lasers. First, the model order of the time series has been estimated by means of the False Nearest Neighbors, Grassberger–Procaccia, Kégl and Levina–Bickel algorithms. The estimates of the attractor dimension using Grassberger–Procaccia and Kégl algorithms are 2.00 and 2.02, respectively. Since the attractor dimension of data set A is 2.06, the estimates of both algorithms can be considered quite satisfactory. Applying the Eq. 1 of the Takens–Mañé theorem, we see that the embedding dimension estimate, provided by Grassberger–Procaccia and Kégl algorithms, is ∼5. Hence the model order is 4. The estimate of Levina–Bickel for the attractor dimension of data set A is 2.35, therefore the model order for Levina–Bickel is 5. Then we have estimated the model order using the False Nearest Neighbors method. As shown in Fig. 1b, the percentage of False Nearest Neighbors is negligible for an embedding dimension value of 3. Hence the model order estimated by False Nearest Neighbors is 2. Then the model order,

[7] Paris-14E Parc Montsouris and DSVC1 time series can be downloaded from http://www.knmi.nl/samenw/eca and http://www.cpdee.ufmg.br/∼MACSIN/services/data/data.htm, respectively.

Table 2 The False Nearest Neighbors, Kégl, Grassberger–Procaccia and cross-validation method on the Data Set Paris-14E Parc Montsouris

Algorithm	Attractor dimension	Embedding dimension	Model order	Mean square error
False Nearest Neighbors			5	436.5 (16.66)
Kégl	4.03	∼9	8	434.3 (16.65)
Levina–Bickel	5.96	∼13	12	434.6 (16.65)
Grassberger–Procaccia	4.91	∼11	10	429.6 (16.53)
Cross-validation			10	429.6 (16.53)

Average error is reported in brackets

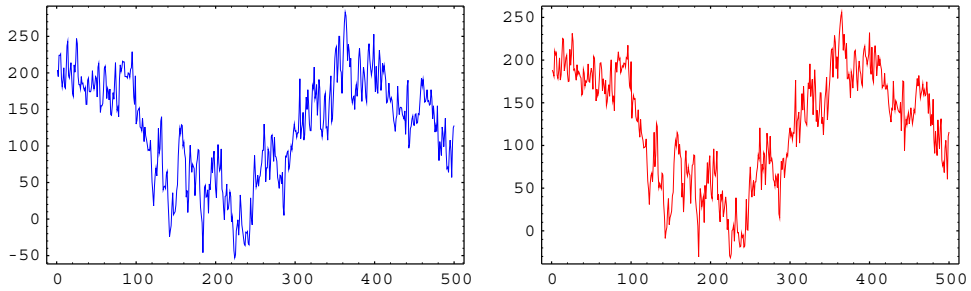

Fig. 3 Data Set Paris-14E Parc Montsouris. The original target data and the results yielded by SVM (model order = 10) are shown on the *left* and the *right*, respectively

Table 3 The False Nearest Neighbors, Kégl, Grassberger–Procaccia and cross-validation method on DSVC1 Time Series

Algorithm	Attractor dimension	Embedding dimension	Model order	Mean square error
False Nearest Neighbors			6	0.10 (0.23)
Kégl	2.14	[5…6]	[4…5]	[0.10…0.075] ([0.26…0.20])
Levina–Bickel	2.26	∼6	5	0.075 (0.20)
Grassberger–Procaccia	2.20	[5…6]	[4…5]	[0.10…0.075] ([0.26…0.20])
Cross-validation			5	0.075 (0.20)

Average error is reported in brackets. Since the model order estimated by Grassberger–Procaccia and Kegl is between 4 and 5, mean square error is between 0.10 and 0.075, and average error is between 0.26 and 0.20

estimated by three different algorithms, has been used to carry out *one-step ahead prediction*, i.e. the prediction of the next value of the time series. The former 70% of time series, i.e. 700 samples, has been used for the training set, while the latter one has been used for the test set, that is formed by 300 samples. The prediction stage has been performed using *SVM-Light* [11], an implementation of SVM for Regression [17, 20]. In our experiments, we have used the gaussian kernel and the kernel variance has been set up using cross-validation. Finally, as a comparison we have set up the model order by means of the cross-validation. The results are reported in the Table 1 and shown in Fig. 2.

3.1.2 Paris-14E Parc Montsouris

Paris-14E Parc Montsouris is a real data time series formed by the daily average temperatures, expressed in tenths of Celsius degrees, in Paris. The time series covers the whole period from January 1, 1958 to December 31, 2001 and has 15,706 samples. The former 64% of time series (10,043 samples) has been used for the training set, while the latter one has been used for the test set, formed by 5,663 samples. We have estimated the model order using the False Nearest Neighbors, Grassberger–Procaccia, Kégl and Levina–Bickel algorithms and we have performed the prediction stage using SVM-Light. Even in this case, we have used the gaussian kernel, setting the variance using cross-validation. As a comparison we have also estimated the model order by means of the cross-validation. The results on the test set, expressed in terms of quadratic loss, are reported in the Table 2 and shown in Fig. 3.

3.1.3 DSVC1

DSVC1 is a real data time series, formed by 5,000 samples, measured from a hardware realization of Chua's circuit [5]. The former 65% of time series (3250 samples) has been used for the training set, while the latter one has been used

Table 4 Average CPU Time, measured in seconds, required by False Nearest Neighbors, Kégl, Grassberger–Procaccia and cross-validation methods for estimating the model order on Data Seta A, Paris-14E Parc Montsouris and DSVC1 benchmarks

Algorithm	Data Set A	Paris-14E Parc Montsouris	DSVC1
False Nearest Neighbors	1	65	5
Kégl	32	1,230	202
Levina–Bickel	30	1,170	189
Grassberger–Procaccia	16	536	106
Cross-validation	125	2,810	756

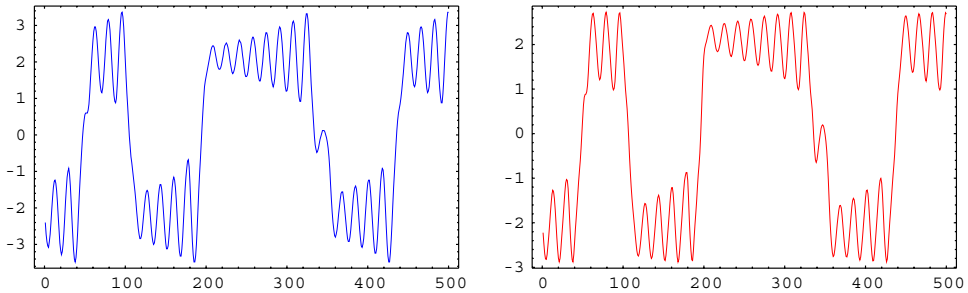

Fig. 4 Chua Time Series. The original target data and the results yielded by SVM (model order = 5) are shown on the *left* and the *right*, respectively

Table 5 Parameters of the three synthetic data sets

	$a(0)$	σ	$\vec{a}$
$d = 4$	−0.2	0.005	(−0.38, 0.493, 0.485, −0.535)
$d = 5$	−0.05	0.005	(−0.5, 0.3, 0.4, 0.25, −0.35)
$d = 6$	−0.3	0.005	(0.22, 0.38, −0.26, −0.23, −0.126, 0.4)

for the test set, that has 1,750 samples. The model order was estimated using the False Nearest Neighbors, Grassberger–Procaccia, Kégl and Levina–Bickel algorithms. The prediction stage was performed using SVM-Light. Even in this case, we have used the gaussian kernel, setting the variance using cross-validation. The estimates of the attractor dimension using Grassberger–Procaccia, Kégl and Levina–Bickel algorithms are 2.20, 2.14 and 2.26, respectively. Since the attractor dimension of data set A is $\sim$2.26, the estimates of the algorithms can be considered satisfactory. As a comparison, the model order was also estimated by means of cross-validation. The results expressed on the test set are reported in the Table 3 and are shown in Fig. 4. Finally, the average CPU times required by all nonlinear methods for estimating the model order are reported in Table 4. Note that in the FNN method, percentage of false neighbors has been computed up to a maximum dimension of 12. The experiments were performed on a Windows Vista.[8] PC with a Dual Core 1,83 GhZ Intel Processor and 3 GByte RAM. False Nearest Neighbors, Grassberger–Procaccia and cross-validation were implemented in C/C++, whereas Kégl and Levina–Bickel were implemented using Mathematica.[9]

3.2 Synthetic time series

The synthetic time series[10] have been generated with a fixed and known model order, and we were interested in evaluating the ability of the presented methods to estimate it. We generated three time series in the following way:

$$x(t+1) = \sum_{i=1}^{d} a(i)x(t-i+1) + a(0) + \varepsilon$$

The vector $\vec{a}$ contains the coefficients of the linear combination of the past d samples, and $\varepsilon \sim \mathcal{N}(0, \sigma)$ is a white Gaussian noise term. The time series have been generated using the parameters shown in Table 5. The series have been generated starting from few random numbers between -1 and 1 (that have been discarded), and have length 10,000.

Tables 6, 7 and 8 show the model order of the three synthetic time series, estimated using the False Nearest Neighbors, Grassberger–Procaccia, Kégl and Levina–Bickel algorithms. In all the tables, the non-integer value of the embedding dimension is approximated to the closest integer. As shown in the tables, Kégl's algorithm underestimates the model order in all time series. False Nearest Neighbor provides the correct value of the model order for two time series and underestimates the model order of the remaining one. Grassberger–Procaccia and Levina–Bickel

[8] Windows Vista is a registered trademark of Microsoft Inc.

[9] Mathematica is a registered trademark of Wolfram Inc.

[10] Available on request for further investigations and comparisons.

Table 6 The False Nearest Neighbors, Kégl, Grassberger–Procaccia methods on a Synthetic Data series, whose model order is 4

Algorithm	Attractor dimension	Embedding dimension	Model order estimated
False Nearest Neighbors			4
Kégl	1.66	~4	3
Levina–Bickel	1.95	~5	4
Grassberger–Procaccia	1.98	~5	4

Table 7 The False Nearest Neighbors, Kégl, Grassberger–Procaccia methods on a synthetic time series, with model order 5

Algorithm	Attractor dimension	Embedding dimension	Model order estimated
False Nearest Neighbors			5
Kégl	1.90	~5	4
Levina–Bickel	2.69	~6	5
Grassberger–Procaccia	2.51	~6	5

Table 8 The False Nearest Neighbors, Kégl, Grassberger–Procaccia methods on a synthetic time series, with model order 6

Algorithm	Attractor dimension	Embedding dimension	Model order estimated
False Nearest Neighbors			5
Kégl	1.92	~5	4
Levina–Bickel	2.89	~7	6
Grassberger–Procaccia	3.02	~7	6

algorithms estimate correctly the model order in all time series.

4 Conclusion

In this paper, we have investigated four nonlinear dynamics methods, i.e. False Nearest Neighbors, Grassberger–Procaccia, Kégl and Levina–Bickel algorithms, to estimate the model order of a time series, namely the number of past samples required to model the time series adequately. The experiments have been performed in two different ways. In the first case, the model order has been used to carry out the prediction, performed by a SVM for regression on three real data time series. The experiments have shown that the model order estimated by nonlinear dynamics methods is quite close to the one estimated using cross-validation. In the second case, the experiments have been performed on synthetic time series, generated with a fixed and known model order, and we were interested in evaluating the ability of the presented methods to estimate it. In this case, most of the methods have yielded a correct estimate and when the estimate was not correct, the value was very close to the real one.

As a general comment, even if cross-validation remains the simplest way to set up the model order of a time series, nonlinear dynamics methods can be very useful. They can be effectively used to narrow down the range for cross-validation, speeding up the crossvalidating process.

Acknowledgments The authors wish to thank the anonymous reviewers for their valuable comments.

References

1. Aguirre LA, Rodrigues GG, Mendes EM (1997) Nonlinear identification and cluster analysis of chaotic attractors from a real implementation of Chua's circuit. Int J Bifurcat Chaos 6(7):1411–1423
2. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723
3. Anderson TW (1963) Determination of the order of dependence in normally distributed time series. In: Rosenblatt M (ed) Time series Analysis. Wiley, London, pp 425–446
4. Arbabanel HDI (1996) Analysis of observed chaotic data. Springer, Berlin
5. Chua LO, Komuro M, Matsumoto T (1985) The double scroll. IEEE Trans Circuits Syst 32(8):797–818
6. Duda RO, Hart PE, Stork DG (2000) Pattern Classification. Wiley, New York
7. Eckmann JP, Ruelle D (1985) Ergodic theory of Chaos and strange attractors. Rev Mod Phys 57:617–659
8. Grassberger P, Procaccia I (1983) Measuring the strangeness of strange attractors. Physica D9:189–208
9. Hirshberg D, Merhav N (1996) Robust methods for model order estimation. IEEE Trans Signal Proces 44:620–628
10. Hübner U, Weiss CO, Abraham NB, Tang D (1994) Lorenz-Like Chaos in NH_3-FIR Lasers. Time Series Prediction. Forecasting the Future and Understanding the Past. Addison Wesley, Reading, pp 73–104
11. Joachim T (1999) Making large-scale SVM learning practical. Advances in Kernel methods—support vector learning, MIT Press, Cambridge
12. Kantz H, Schreiber T (1997) Nonlinear time series analysis. Cambridge University Press, London
13. Kennel MB, Brown R, Arbabanel HDI (1992) Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys Rev A 45(6):3403–3411
14. Kégl B (2003) Intrinsic dimension estimation using packing numbers, vol 15. Advances in Neural Information Processing. MIT Press, Cambridge
15. Mañé R (1981) On the dimension of compact invariant sets of certain nonlinear maps. Dynamical Systems and Turbolence, Warwick 1980. In: Lecture Notes in Mathematics no. 898, Springer, Berlin, pp 230–242
16. Merhav N (1989) The estimation of the model order in exponential families. IEEE Trans Inf Theory 35(5):1109–1114
17. Müller K-R, Rätsch G, Kohlmorgen J, Smola A, Schölkopf B, Vapnik V (2000) Time series prediction using support vector regression and neural network. In: Higuchi T, Takizawa Y (eds) Proceedings of second international symposium on frontiers of time series modelling: nonparametric approach to knowledge discovery. Institute of mathematical statistic publication

18. Ott E (1993) Chaos in dynamical systems. Cambridge University Press, London
19. Packard N, Crutchfield J, Farmer J, Shaw R (1980) Geometry from a time series. Phys Rev Letters 45(1):712–716
20. Schölkopf B, Smola A (2002) Learning with Kernels. MIT Press, Cambridge
21. Rissanen J (1978) Modeling by shortest data description. Automatica 14(5):465–471
22. Shawe-Taylor J, Cristianini N (2004) Kernel Methods for Pattern Analysis. Cambridge University Press, London
23. Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6(2):461–464
24. Smith RL (1992) Optimal estimation of fractal dimension. Nonlinear modeling and forecasting. In: Casdagli M, Eubank S (eds) SFI studies in the sciences of complexity, vol. XII, Addison-Wesley, New York, pp 115–135
25. Snyder DL (1975) Random point processes. Wiley, New York
26. Stone M (1974) Cross-validatory choice and assessment of statistical prediction. J R Stat Soc 36(1):111–147
27. Takens F (1981) Detecting strange attractor in turbolence. Dynamical systems and turbolence, Warwick 1980. In: Lecture Notes in Mathematics no. 898. Springer, Berlin, pp 366–381
28. Takens F (1985) On the numerical determination of the dimension of an attractor. Dynamical systems and bifurcations. In: Braaksma B, Broer H, Takens F (eds) Proceedings Groningen 1984, Lecture Notes in Mathematics No. 1125, Springer, Berlin, pp 99–106
29. Tong H (1990) Nonlinear time series. Oxford University Press, NY
30. Vapnik VN (1998) Statistical Learning Theory. Wiley, New York
31. Vaidya PM (1989) An O(n log n) algorithm for the All-Nearest-Neighbors Problem. Discret Comput Geom 4(1):101–115
32. Whitney H (1936) Differentiable manifolds. Ann Math 37:645–680
33. Wijngaard JB, Klein Tank AMG, Konnen GP (2003) Homogeneity of 20th century European daily temperature and precipitation series. Int J Clim 23:679–692