Neurocomputing 73 (2010) 2017–2029

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Approximate *k*-NN delta test minimization method using genetic algorithms: Application to time series

Fernando Mateo [a,*], Dušan Sovilj [b], Rafael Gadea [a]

[a] Institute of Applications of Information Technologies and Advanced Communications, Universidad Politécnica de Valencia, Valencia, Spain
[b] Laboratory of Information and Computer Science, Helsinki University of Technology, Espoo, Finland

ARTICLE INFO

Available online 7 March 2010

Keywords:
Genetic algorithm
Delta test
Variable selection
Approximate *k*-nearest neighbors
Variable scaling
Variable projection
Time series

ABSTRACT

In many real world problems, the existence of irrelevant input variables (features) hinders the predictive quality of the models used to estimate the output variables. In particular, time series prediction often involves building large regressors of artificial variables that can contain irrelevant or misleading information. Many techniques have arisen to confront the problem of accurate variable selection, including both local and global search strategies. This paper presents a method based on genetic algorithms that intends to find a global optimum set of input variables that minimize the Delta Test criterion. The execution speed has been enhanced by substituting the exact nearest neighbor computation by its approximate version. The problems of scaling and projection of variables have been addressed. The developed method works in conjunction with MATLAB's Genetic Algorithm and Direct Search Toolbox. The goodness of the proposed methodology has been evaluated on several popular time series examples, and also generalized to other non-time-series datasets.

1. Introduction

In many fields like science, industry and finance it is necessary to accurately predict future values of a time series. Some examples of problems that would benefit from an accurate prediction are: industrial processes, that can be modeled, predicted and controlled based on sensory data; natural phenomena, such as daily rainfall or seismic events; medical applications like the modeling of biological signals such as EEG or ECG; and financial problems like the prediction of stock market prices.

The correct estimation of future values of time series is usually affected by complex processes like random fluctuations, sudden trend changes, volatility and noise. The horizon of prediction and the number of available samples to obtain one or more future estimations are important issues too. When building a regressor, which can be understood as the number of past events used to predict their next one, the number of inputs to the model (which is translated as the size of the regressor) can become very large, depending on the periodicity of the particular time series. With large regressors, the learning procedure of the involved predictive models becomes slow and tedious.

Historically, the different models employed to estimate time series have been differentiated in two groups: linear and nonlinear methods. The most popular linear methods are based on the Box–Jenkins methodology [1]. They include autoregressive (AR) models, integrated (I) models, and moving average (MA) models. Their combination has given rise to autoregressive moving average (ARMA) models, autoregressive integrated moving average (ARIMA) models and their seasonal generalization (SARIMA) [2]. However, these models are too limited and simplistic for the average complexity of a time series. In contrast, nonlinear methods are more suitable for complex series that contain irregularities and noise, such as chaotic time series. There is abundant literature on nonlinear models for time series forecasting [3–8]. Among the existing methods are neural networks [9–15], radial basis function networks [11,16–18], support vector machines [19–22], self organizing maps [23,24] and other variants of these models [11,25–28]. However, building these models takes considerable computational time compared to linear models. Recently, several hybrid methods (ARIMA+fuzzy or neural networks) have been employed in the literature [29–31].

Both linear, nonlinear, and hybrid methods have the same purpose: to gather enough information from past samples to give a reliable prediction of the immediate future samples (short-term prediction) or give estimations about far-future samples (long-term prediction). Long term prediction (i.e. predicting multiple steps ahead towards the future) is usually more challenging because the accumulation of errors and inherent uncertainties of a multiple-step-ahead in time yields deteriorated estimates of future samples.

Time series prediction can be considered a modeling problem [32]. The inputs to the model are composed of a set of consecutive

* Corresponding author. Present address: European Organization for Nuclear Research (CERN), Geneva, Switzerland.
E-mail addresses: fermaji@upvnet.upv.es, fernando.mateo.jimenez@cern.ch (F. Mateo), dusans@cis.hut.fi (D. Sovilj), rgadea@eln.upv.es (R. Gadea).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.11.032

regressor instances, while the output is the next value or values of the series that have to be predicted after each regressor instance.

Normally, the size of the regressor is chosen according to the periodicity components of the time series. Consequently, time series with long periodic components may yield very large regressors that can be troublesome to handle by predictive models. Most modeling techniques do not deal well with datasets having a high number of input variables, due to the so called *curse of dimensionality* [33]. As the number of dimensions grows, the number of input values required to sample the solution space increases exponentially. Many real life problems present this drawback since they have a considerable amount of variables to be selected in comparison to the small number of observations. Therefore, efficient variable selection procedures are required to reduce the complexity while also improving the interpretability [34] of multidimensional problems.

Recently, it has been shown that delta test (DT) can be a powerful tool to determine the quality of a subset of variables by estimating the variance of the noise at the output [35]. Several studies related to feature selection using the DT have been developed using both local search strategies such as forward search (FS) [36] and forward-backward selection (FBS) [37,38] and also global search techniques like tabu search [37,39] and genetic algorithm (GA) [37]. Global search methods have the advantage of being able to escape from local minima, to which local methods are prone to converge [40].

This paper presents a global search technique based on GAs that manages not only to select, but also scale and project the input variables in order to minimize the DT criterion. The computation of the DT has been accelerated by using the approximate k-nearest neighbor approach [41]. The designed method makes use of MATLAB's Genetic Algorithm and Direct Search toolbox for the GA-based search. This methodology can be generalized to all regression problems. In this study we are going to focus mainly on time series processing, but we have also included non-time series datasets to show that the methodology can be generalized to all regression problems regardless of their nature. The predictive study of the time series is not going to be addressed, as the methodology only provides reduced datasets for their later modeling.

This paper is organized as follows: Section 2 explains the DT criterion and its role in the present study. Section 3 presents the approximate k-nearest neighbors algorithm that has been used to speed up the DT calculation. Section 4 introduces the motivation for the GA and the fitness function optimization methods, such as scaling, projection and their variations with a fixed number of variables. Section 5 describes the datasets tested, preprocessing and hardware/software specifications for the performed experiments, and finally, Section 6 discusses the most relevant results obtained.

2. Delta test

Delta Test was introduced by Pi and Peterson for time series [42] and recently further analyzed by Liitiäinen et al. [43]. However, its applicability to variable selection was proposed in [35]. The DT is a nonparametric noise estimator, i.e. it aims to estimate the variance of the noise at the output, or the mean squared error (MSE) that can be achieved without overfitting. Given N input–output pairs $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, the relationship between $\vec{x}_i$ and y_i can be expressed as

$$y_i = f(\vec{x}_i) + \eta_i, \quad i = 1, \ldots, N, \tag{1}$$

where f is the unknown function and η is the noise. The DT estimates the variance of the noise η.

The DT is useful for evaluating the nonlinear correlation between input and output variables. According to the DT, the selected set of input variables is the one that represents the relationship between the input variables and the output variable in the most deterministic way.

The DT is based on the hypothesis of continuity of the regression function. If two points $\vec{x}_1$ and $\vec{x}_2$ are close in the input variable space, the continuity of regression function implies that the outputs $f(\vec{x}_1)$ and $f(\vec{x}_2)$ will be close enough in the output space. If this is not accomplished, it is due to the influence of the noise.

The DT can be interpreted as a particularization of the Gamma Test [44] considering only the first nearest neighbor. This yields a fully nonparametric method as it removes the only hyperparameter (number of neighbors) that had to be chosen for the Gamma Test. Let us denote the nearest neighbor of a point $\vec{x}_i \in \mathbb{R}^d$ as $\vec{x}_{NN(i)}$. The nearest neighbor formulation of the DT estimates $\mathrm{Var}[\eta]$ by

$$\mathrm{Var}[\eta] \approx \delta = \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{NN(i)})^2, \tag{2}$$

where $y_{NN(i)}$ is determined from the input–output pair $(\vec{x}_{NN(i)}, y_{NN(i)})$. For a proof of convergence the reader should refer to [44].

3. Approximate k-nearest neighbors

Nearest neighbor search is an optimization technique for finding closest points in metric spaces. Specifically, given a set of n reference points R and query point q, both in the same metric space V, we are interested in finding the closest or nearest point $c \in R$ to q. Usually, V is a d-dimensional space $\mathbb{R}^d$, where distances are measured using Minkowski metrics (e.g. Euclidean distance, Manhattan distance, max distance).

The simplest solution to this neighbor search problem is to compute the distance from the query point to every other point in the reference set, while registering and updating the position of the nearest or k-nearest neighbors of every point. This algorithm, sometimes referred to as the naive approach or brute-force approach, works for small datasets, but quickly becomes intractable as either the size or the dimensionality of the problem becomes large, because the running time is $O(dn)$. In practice, computing exact nearest neighbors in dimensions much higher than 8 seems to be a very difficult task [41].

Few methods allow to find the nearest neighbor in less time than the brute-force computation of all distances does. In 1977, Friedman et al. [45] showed that $O(n)$ space and $O(\log n)$ query time are achievable through the use of kd-trees. However, even these methods suffer as dimension increases. The constant factors hidden in the asymptotic running time grow at least as fast as 2^d (depending on the metric).

In some applications it may be acceptable to retrieve a "good guess" of the nearest neighbor. In those cases one may use an algorithm which does not guarantee to return the actual nearest neighbor in every case, in return for improved speed or memory saving. Such an algorithm will find the nearest neighbor in the majority of cases, but this depends strongly on the dataset being queried. It has been shown [41] that by computing nearest neighbors approximately, it is possible to achieve significantly faster running times (on the order of tens to hundreds) often with relatively small actual errors.

The authors [41] state that given any positive real ε, a data point p is a $(1+\varepsilon)$−approximate nearest neighbor of q if its distance from q is within a factor of $(1+\varepsilon)$ of the distance to the true nearest neighbor. It is possible to preprocess a set of n points