Knowledge and Information Systems (2021) 63:1093–1116
https://doi.org/10.1007/s10115-021-01544-w

REGULAR PAPER

A scalable framework for large time series prediction

Youssef Hmamouche[1] · Lotfi Lakhal[1] · Alain Casali[1]

Received: 2 November 2018 / Revised: 21 May 2020 / Accepted: 21 December 2020 /
Published online: 5 February 2021

Abstract
Knowledge discovery systems are nowadays supposed to store and process very large data. When working with big time series, multivariate prediction becomes more and more complicated because the use of all the variables does not allow to have the most accurate predictions and poses certain problems for classical prediction models. In this article, we present a scalable prediction process for large time series prediction, including a new algorithm for identifying time series predictors, which analyses the dependencies between time series using the mutual reinforcement principle between Hubs and Authorities of the Hits (Hyperlink-Induced Topic Search) algorithm. The proposed framework is evaluated on 3 real datasets. The results show that the best predictions are obtained using a very small number of predictors compared to the initial number of variables. The proposed feature selection algorithm shows promising results compared to widely known algorithms, such as the classic and the kernel principle component analysis, factor analysis, and the fast correlation-based filter method, and improves the prediction accuracy of many time series of the used datasets.

Keywords Time series · Machine learning · Feature selection · Prediction · Scalability

1 Introduction

Time series are sequential data that are generally used to model dynamic systems and processes. In various fields, the ability to make accurate predictions is very important because they provide possible future information about the system being studied. One of the main challenges in time series prediction is improving the forecast accuracy. The first prediction models were univariate, which predict a single time series based on its own history. Then, multivariate models were introduced, which consider multidimensional time series and predict each variable using its previous values and the previous values of the other predictive variables. A way of improving the forecast accuracy consists in developing new prediction

✉ Youssef Hmamouche
youssef.hmamouche@lis-lab.fr

Lotfi Lakhal
lotfi.lakhal@lis-lab.fr

Alain Casali
alain.casali@lis-lab.fr

[1] Aix Marseille University, LIS, CNRS UMR 7020, 413 Avenue Gaston Berger, Aix en Provence, France

models by changing the structure of existing models and how they analyze the history of data in order to make predictions. Another way seeks to focus on the other factors that influence the predictions, by considering this problem as a process where the application of the prediction models is just a step. As such, the forecast accuracy can be improved by many ways, for instance, (i) determining the most optimized structures of the prediction models with respect to the underlying set of predictors, (ii) improving the quality of the input data, (iii) adopting model matching techniques, *etc*.

In the context of time series prediction with many variables, a common goal is detecting the most independent and relevant predictors with regard to a given target time series. In the literature, several approaches were proposed to handle this problem. One of the main motivations behind these approaches is that using all the available predictor variables does not necessarily yield to the best forecast accuracy and sometimes renders some statistical prediction models non-solvable due to the high number of variables compared to the number of observations [30]. Practically speaking, these approaches can be grouped into two main classes: (i) prediction models based on regularization or wrapper feature selection models, where the selection step is performed using the prediction model itself, and (ii) models that execute the reduction step independently from the prediction step. Such models are usually based on applying filter feature selection or dimension reduction in a first step and then using a prediction model on the selected variables.

Despite the advantages of existing methods in the literature, there are still some problems when dealing with large time series, where the distribution of the data storage and processing is also a challenge. Generally, on distributed platforms, partitioning data on multiple nodes makes the adaptation of machine learning algorithms delicate. In addition, it is not always possible to distribute complex algorithms without loss of information.

The problem we are addressing in this article is the prediction of multivariate time series that contain many predictors, i.e., predictive variables. The main question we are dealing with is how to select the subset from the predictors set that allows to obtain the best forecasts for a given target variable. Theoretically, if we have n variables to predict a target variable, there are 2^n possible sub-set of features, and this problem is NP-hard. In general, algorithms that deal with this problem are based on heuristics [24], or on statistical models.

This article is an extended version of the paper presented in [9], where we proposed an novel algorithm for feature selection specific to high-dimensional time series. The extensions consist in presenting a scalable version of the proposed algorithm and proposing a complete and scalable framework for large time series prediction, which includes the feature selection algorithm. The framework takes as input a multivariate time series and permits to (i) make future predictions and (ii) find the subset of variables yielding the best predictions for each target time series. Additional functionalities are also provided like the causality graphs of the input time series and comparisons between the models used by the framework in terms of their applicability.

The organization of this article is as follows: in Sect. 2, we describe existing concepts that are connected to our problematic, and then, we discuss related works. In Sect. 3, we detail the proposed feature selection method. Section 4 is for the experiments setup. In Sect. 5, we show and discuss the obtained results. In Sect. 6, we discuss the scalability of proposed prediction process and present the distributed version of the proposed algorithm. Finally, in the last section, we summarize our contributions.

2 Related work

The aim of this section is to discuss some approaches from related works that address the problem of large-scale time series reduction for prediction purposes. The cause behind reducing the number of predictors in a multivariate prediction model is generally based on two reasons; (i) for some linear models, if the number of variables is large, the estimation of the parameters may not be achievable [30] and (ii) applying a prediction model with all available variables is not necessarily the best choice in terms of prediction accuracy, because of the existence of redundant information within the set of variables.

Researches that handle the problematic studied in this article belong to the intersection of three topics; feature relevance, dimension reduction and feature selection, and multivariate time series prediction. Therefore, this section involves a discussion of these topics. Afterward, we present some related works that combine them to build models for large time series.

2.1 Feature relevance

Determining the relevance between variables is one of the central building blocks in feature selection. Therefore, it can be exploited to evaluate the importance of predictors in a multivariate prediction model.

In [15], a relevance definition was proposed by considering that a variable X_i is relevant to Y if there exist x_i and y, where $P_r(X_i = x_i) > 0$ such that:

$$P_r(Y = y|X_i = x_i) \neq P_r(Y = y), \tag{1}$$

where P_r represents the probability. The idea behind this definition is that X_i is relevant to Y, if the fact of knowing and taking into consideration the information of X_i results in changing the estimate of Y compared to the situation where X_i is not used. In the same work [15], other similar definitions were presented, where more than two variables are considered, but they are based on the same principle.

The predictive causality is also a very interesting concept in time series analysis. Generally, there are two common predictive causality measures for modeling complex time series; the Granger causality [8] and its nonlinear extensions such as the one based on kernel generalization [36], and the transfer entropy [28]. The original Granger causality definition presented in [8] rests on the prediction aspect to define the causality. Let us consider two univariate time series x, y with t observations. x_{t+1} is unknown at time t and therefore can be expressed using probability in a set B. Consider I the set of information available at time t within y_t. y_t causes x_{t+1} if:

$$P_r(x_{t+1} \in B|I) \neq P_r(x_{t+1} \in B|I \backslash y_t). \tag{2}$$

Practically, the Granger causality test [8] uses two VAR (vector auto-regressive) models to evaluate the causality from one variable to another one. It evaluates two models: (i) the first one uses just the previous values of y_t and (ii) the second uses both x_t and y_t in order to predict y_t:

$$y_t = \alpha_0 + \alpha_t + \sum_{i=1}^{p} \alpha_i y_{t-i} + \epsilon_t, \tag{3}$$

$$y_t = \alpha_0 + \alpha_t + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{p} \beta_i x_{t-i} + \epsilon_t, \tag{4}$$

where $[\alpha_t, \alpha_0, \ldots, \alpha_p]$ and $[\beta_t, \beta_0, \ldots, \beta_p]$ are the parameters of the models and ϵ_t is a white noise error term. The parameter α_t is used in case of the time series y_t contains some trends, and it is not required for stationary time series. The Granger causality test evaluates these two models using the residual sum of squares, to check if adding x_t to the model improves the predictions of y_t. A statistical significance can be obtained using the Fisher test, by providing a p-value indicating the probability of non-causality, if this probability is low based on a threshold value (e.g., 5%), then we can say that x_t causes y_t.

Similarly, the transfer entropy proposed in [28] is based on the same general principle of the previous definition, but uses information theory instead of prediction models. In addition, it is considered as nonlinear extension of the Granger causality. The transfer of information from a variable x to another variable y is based on the difference between two conditional entropies, where in the first one, only previous values of y are used, and in the second, both previous values of x and y are considered. Formally, the transfer entropy between two time series, or two processes x and y, measures the information flow from x to y, and can be expressed as follows:

$$\mathrm{TE}_{x \to y} = H(y_t | y_{t-1}, \ldots, y_{t-p}) \tag{5}$$

$$- H(y_t | (y_{t-1}, \ldots, y_{t-p}), (x_{t-1}, \ldots, x_{t-p})), \tag{6}$$

where H is a Shannon conditional entropy and p is a time delay parameter. Let us underline that feature relevance concept is quite close to the predictive causality in the sense that both are based on analyzing two situations; one using the predictive and the target variable and the other one using only the target variable, and then evaluating the difference between these two situations. The main difference is that the predictive causality takes time into account, and this is very important in forecasting with lagged variables and therefore seems more adequate to model interactions between time series in our problematic.

2.2 Dimension reduction

Dimension reduction methods generate new variables computed as a combination of the original variables. These variables are generally constructed with regard to two criteria: (i) they are uncorrelated between them and (ii) must keep as much as possible the characteristics of the original variables. PCA is one of the most common dimension reduction methods. The fundamental principle of this method is to find the principal components on which the data can be better explained. Consider a vector of n variables $Y = [y_1, \ldots, y_n]$ containing l observations, and $\Sigma = \frac{1}{n} Y^\intercal Y$ is its covariance matrix.

The idea of the PCA is to generate $k << n$ linearly uncorrelated factors $[p_1, \ldots, p_k]$, while preserving the information contained in original variables [16]. These factors can be expressed as follows:

$$p_j = a_j^\intercal Y = \sum_{i=1}^{n} a_{ji} y_i \quad \forall j \in [1, k], \tag{7}$$

One approach to find the principal components is to use the decomposition of the covariance matrix Σ. Σ is a positive square and symmetric matrix, so it is diagonalizable, so it can be decomposed as follows:

$$\Sigma = U A U^\intercal, \tag{8}$$

where A is a diagonal matrix and U is a matrix where each column is an eigenvector and $UU^\intercal = 1$. Based on this property, the principal components are determined such that

$(a_1^\intercal, \ldots, a_k^\intercal)$ are the eigenvectors of Σ in the ascending order, i.e., a_k is associated with the k^{th} eigenvalue.

Similarly, factor analysis reduces the size of a vector of n variables by explaining them by a vector of uncorrelated factors of dimension $p \leq n$. But contrary to the PCA, where the principal components are determined via a decomposition of the covariance matrix, the determination of the factors with FA is based on a multivariate statistical model [16,27]. Factorial analysis constructs a model where the original variables are expressed linearly according to the factors:

$$Y = BF + E, \tag{9}$$

where B, F, and E represent *resp*. the parameters of the model, the factors, and the error terms.

2.3 Feature selection

Feature selection is the process of eliminating redundant information by extracting a subset of the most relevant variables. There are two types of feature selection methods: filter and wrapper-based methods. Wrapper-based feature selection methods use the prediction algorithm itself to select variables, by refining the selected variables with respect to the model's performance at each iteration, while filter methods select variables independently from the prediction model based on the relationships between the variables. Filter feature selection includes two classes of algorithms: univariate algorithms and subset search algorithms [35]. On the one hand, univariate algorithms consist in ranking variables according to a relevance measure that we discussed like correlation, mutual information, Euclidean distance, or relevance measures specific to time series like the ones discussed in the first part of this section, such as Granger causality and transfer entropy. The final variables are then selected by setting a limit threshold of the measurement used, or according to a desired number of variables. On the other hand, subset search algorithms evaluate many subsets of features. In general, their complexity is exponential, because exploring all the search space, i.e., all possible subsets of a set of variables of size n, requires 2^n evaluations. In the literature, many feature selection methods that are based on correlation have been investigated. The correlation-based feature selection (CFS, [21]) seeks to find the variables that are correlated with the target and uncorrelated between them. Considering S a set of k features, and a target variable y, the quality of the set S is computed via the following measure:

$$Q_s = \frac{k\bar{r}_{ci}}{\sqrt{k + (k-1)\bar{r}_{ii'}}}, \tag{10}$$

where kr_{ci} is the mean of feature-target correlations and $kr_{ii'}$ is the mean of feature-feature correlations of the set S.

In [35], as an extension of the CFS method, and in order to avoid the shortcoming of linear correlation measures, the FCBF (fast correlation-based filter) algorithm was proposed for feature selection of high-dimensional datasets. The algorithm uses the symmetrical uncertainty as a relevance measure [26]. It is mainly based on the mutual information, which can be considered as a nonlinear correlation. In a first step, the algorithm selects the most relevant features, according to the class (the target) based on a threshold on the values of symmetrical uncertainty. Second, the algorithm eliminates redundant variables by calculating pairwise correlation between them.

2.4 Discussion

Feature selection and dimension reduction methods can play the same role in reducing the dimensionality of the set of predictor variables in a multivariate prediction model. Nevertheless, their principles are totally different, since dimension reduction methods generate new variables, which are generally a combination of the original variables, while feature selection methods extract variables from the originals. In practice, it is important to know exactly the relevant and real variables, especially when we are dealing with financial and macroeconomic data. Furthermore, knowing the most relevant variables is itself an important information as it can be used for other purposes besides predicting. On the other hand, sometimes we are just interested in determining input features of the prediction models, even if they are not part of the original variables. Hence, choosing the appropriate method depends not only on the accuracy of the predictions obtained, but also on the current problem.

In the literature, many approaches were proposed to handle large time series prediction, with applications in different fields, like industry, finance, weather forecasting, and so on. In each application, the time series structure and the domain knowledge hypothesis are specific to the studied system. The common goal between these approaches is to find the main time series that cause the future change of the system. In terms of methodology, they generally differ in the way of constructing a reduced set of features from the set of initial variables before applying the multivariate prediction model. Dimension reduction and feature selection methods, in particular principal component analysis (PCA), factor analysis (FA), and filter-based feature selection methods, are widely used to deal with this problem [6,37]. In [18], a clustering-based sample entropy approach is proposed to identify the temporal information between time series, and then it is used in a feature selection method for black-box weather forecasting. In [22], a forecasting approach was proposed to forecast electricity load time series. First, correlation, mutual information and instance-based feature selection methods were applied in order to extract the relevant informative lag variables. And second, a combination of multivariate artificial neural network and statistical models is applied to make forecasts. In [4], a similar forecasting process was proposed. It includes a filter and wrapper feature selection approaches in order to select the most important variables, and then a multilayer perceptron model is used for forecasting. In [1], a forecast approach was presented by proposing a new feature selection method based on information theory, for the purpose of modeling nonlinear dependencies between variables. Similarly, in [37], a data mining process was proposed for forecasting daily stock market using PCA and a feed-forward neural network prediction model.

3 Our approach

This section covers a description of our approach. First, we present the main steps of our prediction framework. Then, we detail the proposed feature selection algorithm and illustrate an example.

3.1 The prediction process

From an input multivariate time series, the process allows to predict target variables based on their history and the history of other related variables. The particularity of this system is that is based on a description step that consists in calculating causality graphs and uses a novel

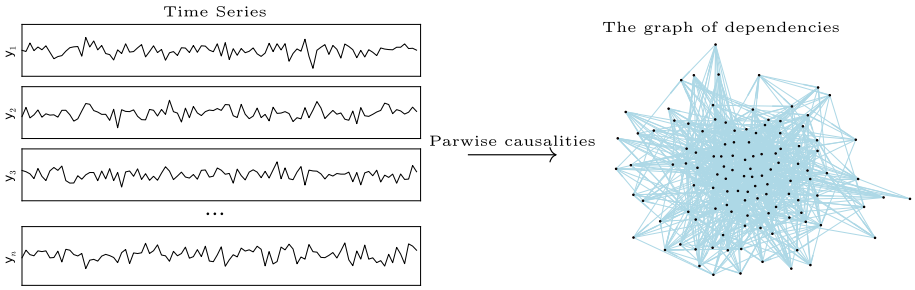


Fig. 1 Illustration of the transformation from time series to the graph of dependencies through pairwise causalities, where nodes represent the variables and the edges indicate the causalities

feature selection algorithm that exploits these causalities to find the most relevant predictors for each target variable. The prediction process consists of main 3 steps:

1. The computation of dependencies between variables via causality graphs: here, we compute the matrix of dependencies (or a graph) between variables using a causality measure. For example, the graph shown in Fig. 1 represents the graph of dependencies of the second dataset used in the experiments (*cf.* Table 1) using the Granger causality test, where the nodes represent the time series, and the edges represent the pairwise causalities. Using a matrix representation, if M is the matrix of pairwise causalities, then $M[i, j]$ is the value of the causality from the variable number i to the variable number j.
2. Feature selection and dimension reduction with different methods including the proposed algorithm.
3. Prediction using several prediction models.

3.2 The proposed feature selection method

Let us consider a set of predictors time series $P = \{y_1, \ldots, y_n\}$, and a target time series x. Consider also V as the vector of causalities from variables of P to the target, where $V[i]$ is the causality from y_i to x. The goal is to select a set from P of size $k << n$, which contains the most important variables in terms of prediction accuracy with regard to the variable x. The idea is to consider the predictor–predictor dependencies and the predictor–target dependencies. More precisely, we select predictor variables based on two criteria:

1. Having significant causality to the target: the first criteria is that the predictors should cause the target. However, this is not sufficient, because if we rank variables based on their causalities to the target, we get the problem of dependencies. Figure 2 illustrates this problem: what should be the best two predictors for y_4? Selecting y_1 and y_2 may not be the best choice, even if they strongly cause y_4, because if y_1 causes y_2, then they may provide the same information to y_4. In this case, it is worth to diversify the sources of information to predict y_4.
2. Summarizing as much as possible the information of predictor variables. Similar to the idea of the PCA technique that, based on the co-variance matrix, constructs the principal components that drive the others variables, here the second criteria are to choose variables that represent the source of information in the graph of predictor–predictor causalities.

We can connect our problem to the one of ranking web pages by making an equivalence between the matrix of dependencies of web pages and the matrix of causalities of time series.

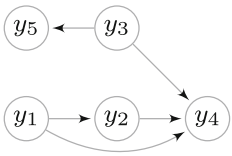


Fig. 2 Example of dependencies between time series

Nevertheless, the difference between our problem and web page ranking is that the latter is an unsupervised model. In our case, we exploit information about the target variable and add it in the graph of dependencies. Therefore, our approach is based on an adaptation of the Hits algorithm. In the rest of this part, we first discuss the original Hits algorithm, and then we describe our approach.

3.2.1 Generalities on the hits algorithm

The Hits (Hyperlink-Induced Topic Search) algorithm, proposed in [20], was originally used for web page analysis. It was developed to detect the most relevant pages from the Web graph by analyzing the dependencies between them. It considers two notions of pages relevance: the *Hubs* and the *Authorities*. And this is important in our case (forecasting using many variables), because we are not only interested in extracting the most relevant variables, but we distinguish between variables that transfer information, and variables that receive the information. Each page has an *Authority* and a *Hub* score. Good Hubs are pages that point to many pages (transferring information), while good *Authorities* are pages that are pointed by good *Hub*. The final purpose of the original application of the Hits algorithms is to rank pages based on their *Authorities* weights, because the goal was to find pages that are the most pointed.

The equations for updating the *Authorities* and a *Hubs* weights of each page are as follows:

$$a(p) = \sum_{q:q\to p} h(q), \tag{11}$$

$$h(p) = \sum_{q:p\to q} a(q). \tag{12}$$

3.2.2 The principle of the proposed method

Our approach is a particular adaptation and a new application of the Hits algorithm. It aims to to select the most relevant variables that cause the target while minimizing dependencies between them. A similar idea was investigated in [34], which uses the Hits algorithm for ranking items that maximize a profit in an association rule model. But from our knowledge, this concept was not used before to select predictors for multiple time series prediction based on the causality graphs.

To do that, we weight the coefficient of the *Hubs* and *Authorities* with *predictor–predictor* and *predictor–target* causalities as expressed through the following equations:

$$h(p) = \sum_{q:p\to q} a(q) \times M[p,q] \times V[p], \tag{13}$$

$$a(p) = \sum_{q:q\to p} h(q) \times M[q,p] \times V[q]. \tag{14}$$

These equations can be compactly expressed as follows:

$$h = Ga, \tag{15}$$

$$a = G^{\intercal}h, \tag{16}$$

where M is the matrix of pairwise causalities, $G[i, j] = M[i, j] \times V[i]$. With the use of this weighting method, the algorithm computes for each variable the *Hub* and the *Authority* scores by including the causality to the target. In the end, contrary to the first application of Hits algorithm were pages are ranked based on the *Authorities* scores, we are interested in the top variables based on their *Hub* scores, because this is our objective, i.e., the computation of these scores is basically designed to detect variables that represent the source of information in the causality graph by considering the causality to the target (*cf.* Eqs. 13 and 14).

3.2.3 The algorithm of the proposed method

In this part, we describe the proposed *PEHAR* (predictors extraction using *Hubs* and *Authorities* ranking) algorithm (*cf.* Fig. 1). The exact method for finding the *Hubs* and the *Authorities* vectors is based on matrix resolution using linear algebra [3]; more precisely, it is about eigendecomposition of diagonalizable matrices. Formally, by replacing the *Hubs* and the *Authorities* vectors in Eqs. 15 and 16, we obtain the following equations:

$$a = G^{\intercal}Ga, \tag{17}$$

$$h = GG^{\intercal}h. \tag{18}$$

Consequently, the *Authorities* vector converges to the eigenvector of the matrix $G^{\intercal}G$, and the *Hubs* vector converges to the eigenvector of the matrix $GG^{\intercal}$. In our case, what is important is the first eigenvector of $GG^{\intercal}$, which corresponds to the highest eigenvalue. Although this method is exact, it has some drawbacks. First, the computation time to find the eigenvectors can be important, especially if the matrix G is large and dense. Second, it requires a computation of all eigenvectors of $GG^{\intercal}$, but in our case, just one is needed.

In this paper, we adopt an iterative method based on the power iteration technique, as it generally converges in a reasonable number of iterations. The proposed Algorithm 1 shows an implementation of this method. It consists in executing Eqs. 13 and 14 iteratively, with a normalization step in each iteration, until stability [20]. Thus, the user can control the computational time through the number of iterations or an expected error. We consider that the values of dependencies between variables are computed separately, in such a way that the algorithm takes as input the matrix of *predictor–predictor* causalities M and the vector of *predictor–target* causalities V.

3.2.4 Example

In this example, we apply the proposed algorithm on a small set of variables. Consider a set of 5 predictors $P = \{y_1, \ldots, y_5\}$, and a target variable x. Consider also M as the matrix of causalities of P, and V the vector of causalities from P to x (following the same notations in Sect. 3.2.2), which are computed using the transfer entropy as follows:

$$M = \begin{pmatrix} 0 & 0.38 & 0.52 & 0.51 & 0.70 \\ 0.88 & 0 & 0.91 & 0.401 & 0.89 \\ 0.89 & 0.34 & 0 & 0.96 & 0.71 \\ 0.95 & 0.62 & 0.56 & 0 & 0.67 \\ 0.92 & 0.96 & 0.99 & 0.77 & 0 \end{pmatrix}, \quad V = \begin{pmatrix} 0.07 \\ 0.90 \\ 0.65 \\ 0.16 \\ 0.35 \end{pmatrix}.$$

Algorithm 1: The PEHAR algorithm.

Input : n: the number of predictors;
M: the causality matrix between predictors of size $(n \times n)$;
V: the vector of *predictor–target* causalities of size n;
$iters$: the number of iterations;
min_error: the minimum error;
Output: h: the vector of *Hubs* scores of size n;

$h(p) \leftarrow \frac{1}{n}, \quad p = 1, \ldots, n;$
while $i < iters$ *or* $min\_error < error$ **do**
  $hlast \leftarrow h;$
  $h(p) \leftarrow 0, \quad p = 1, \ldots, n;$
  $a(p) \leftarrow 0, \quad p = 1, \ldots, n;$
  for $p \in [1, n]$ **do**
    for $q \in \{[1, n], p \neq q\}$ **do**
      $a(q) \leftarrow a(q) + hlast(p) \times M[p, q] \times V[p];$
    end for
  end for
  for $p \in [1, n]$ **do**
    for $q \in \{[1, n], p \neq q\}$ **do**
      $h(p) \leftarrow h(p) + a(q) \times M[p, q] \times V[p];$
    end for
  end for
  $s_a \leftarrow max(a);$
  $s_h \leftarrow max(h);$
  $h(p) \leftarrow h(p)/s_h, \quad p = 1, \ldots, n;$
  $a(p) \leftarrow a(p)/s_a, \quad p = 1, \ldots, n;$
  $error \leftarrow 0;$
  for $p \in [1, n]$ **do**
    $error \leftarrow error + abs(h(p) - hlast(p));$
  end for
  if $error > min\_error$ **then**
    return $h;$
  end if
end while
return $h;$

We compute the matrix G as indicated in Sect. 3.2.2: $G[i, j] = M[i, j] \times V[i]$,

$$G = \begin{pmatrix} 0 & 0.026 & 0.036 & 0.035 & 0.049 \\ 0.792 & 0 & 0.819 & 0.360 & 0.801 \\ 0.578 & 0.221 & 0 & 0.624 & 0.461 \\ 0.152 & 0.099 & 0.0896 & 0 & 0.107 \\ 0.322 & 0.336 & 0.3465 & 0.2695 & 0 \end{pmatrix}.$$

As discussed in Sect. 3.2.3, the algebraic resolution method for finding the *Hubs* vector is based on the eigenvectors of $GG^\intercal$. By computing all normalized (l_1 norm) eigenvector of $G^\intercal G$, and ranking them according to eigenvalues, the first eigenvector is as follows:

$$V_{alg}^\intercal = \begin{pmatrix} 0.0196 & 0.4639 & 0.2853 & 0.0661 & 0.1651 \end{pmatrix}. \tag{19}$$

If we apply the power iteration method as detailed in Fig. 1, we obtain the following *Hubs* vector in just 3 iterations:

$$V_{iter}^\intercal = \begin{pmatrix} 0.0196 & 0.4638 & 0.2854 & 0.0661 & 0.1651 \end{pmatrix}. \tag{20}$$

Table 1 Datasets

Datasets	Description
Ausmacrodata	An Australian quarterly macroeconomic time series in the period 1984–2015, containing 117 variables and 119 observations [12]
Stock-and-Watson	an USA quarterly macroeconomic time series in the period 1960–2008, containing 143 variables and 200 observations [31]
Sales transactions	Weekly purchased quantities of products over 52 weeks, containing 811 variables and 52 observations [5,32]

The final step consists simply in ranking variables based on the *Hubs* vector. In the example, despite the small difference between the two vectors obtained by the iterative and the algebraic methods (Eqs. 19, 20), they lead to the same ranking.

4 Experiments setup

In this section, we present the setup used to perform the experiments. We describe the used datasets, the prediction procedure, and the evaluated prediction models and the reduction methods. The aim of these experiments is to select the best reduction algorithm, the appropriate number of predictors and the best prediction model, not only for each dataset, but for each variable. These evaluations will help us to identify and rank algorithms based on their applicability on each variable in terms of prediction accuracy.

4.1 Datasets

The experiments are conducted on 3 real time series datasets. A description of these datasets is provided in Table 1.

4.2 The used reduction methods

We compare our algorithm to dimension reduction and feature selection methods that are widely utilized in the literature, including PCA, kernel PCA, factor analysis, and FCBF (fast correlation-based filter) (see Sect. 2 for more details). Two versions of our algorithm are implemented, the first one using the Granger causality test for computing causalities (PEHAR-gc), and the second using the transfer entropy (PEHAR-te). We use a p-value threshold equal to 0.05 for the statistical significance of the causality test.

4.3 The used prediction models

Different prediction models are used of type univariate and multivariate and belong to two categories. The first one includes multivariate models that are used after the reduction step. Essentially, the chosen models are characterized by the ability to consider long-term relationships between variables, but using two different mechanisms:

– A statistical model: the vector error correction model (VECM) [14]. It uses the cointegration of non-stationary time series to exploit long-term dependencies between variables.

The cointegration is simply when two time series are non-stationary, and there is a stationary linear combination of them. Let us consider a non-stationary multivariate time series Y_t that is integrated of order 1. The VECM model can be written as follows:

$$\Delta Y_t = \Pi Y_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta Y_{t-i} + u_t, \tag{21}$$

where ΔY_t is the stationary time series resulting from the differentiation of Y_t (computing the difference between two consecutive observations), Π is the matrix representing the co-integration equations, Γ_i are the coefficients of the model, and u_t are white noise error terms. If $rk(\Pi) = 0$, then there are no cointegration relationships. In this case, the VECM model is reduced to the VAR model:

$$\Delta Y_t = \sum_{i=1}^{p-1} \Gamma_i \Delta Y_{t-i} + u_t. \tag{22}$$

– An artificial neural network model: the LSTM model [10]. It is a particular recurrent neural network, which supports time lags of the input variables, and it becomes very useful for time series prediction. Its structure contains a specific unit that includes artificial neurons responsible for forgetting and remembering old information passed through the network. For this model, we used an architecture composed of one LSTM hidden layer and a fully connected output layer containing one neuron to provide one prediction each time using the sigmoid activation function. The network is trained using the ADAM optimization algorithm [19], and the mean squared error is considered for the loss function.

The second category includes prediction models that do not require a selection or a dimension reduction step. Here, we use the following models:

– The ARIMA (auto-regressive integrated moving average) model. It used as a baseline model and in order to check if there exist some variables that could be predicted better without the use of predictors. Let us consider a univariate time series y_t that is integrated of order d (i.e., we have to differentiate it d times to become stationary).
The ARMA(p, q) model expresses y_t according to the last q errors terms and the last p past values of y_t:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t.$$

where α_i and β_i are the coefficients of the model and ϵ_t is an error term. The ARIMA(p, d, q) model is an extension of ARMA for non-stationary time series. It transforms the time series to be stationary, and then it applies the ARMA model. In our case, the lag parameter p is fixed to 4 (*cf.* Sect. 4.4), the parameter d depends on the stationary order of the time series, and q is determined based on the Akaike information criterion (AIC) [2].
– A vector auto-regressive combined with shrinkage methods. Shrinkage methods represent another approach to estimate the coefficients of linear models that contain many variables. They consist in minimizing the impact of irrelevant variables of a regression model iteratively by the prediction model itself. Thus, a regression model after estimating their coefficients with the regularization methods can contain many variables with a regularization on the non-important coefficients, for example, by pushing them close

to 0 [11]. Classical shrinkage methods are in fact specific to linear multiple regression models. But in our case it is different because we build models based on lagged variables. To overcome this limitation, we implemented an adaptation of the vector auto-regressive model with shrinkage representation. In this category, we employ Bayesian ridge, ridge and Lasso methods.

4.4 The prediction procedure

The prediction procedure used is similar to the one used in [25,31], where the two first datasets described in Table 1 have been used. We forecast all variables for each dataset and evaluate the last 75 and 100 values of Australian and US datasets, resp., and for Sales-Transactions dataset, we evaluate the models with the last 20% observations. These predictions are performed based on a rolling window procedure with one step ahead forecast. The lag parameter for prediction models is set to 4, which is equivalent to 4 quarters for US (Stock-and-Watson) and Australian (Ausmacrodata) datasets, and 4 weeks for the third dataset. Two prediction accuracy measures are used, the root mean square error (RMSE), and the mean absolute scaled error (MASE). The MASE is a new forecast accuracy proposed in [13], independent of the data scale, and based on the errors of the forecasts and the mean absolute error of the naive method:

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{h} {e_t}^2}{h}},$$

$$\text{MASE} = \frac{\frac{\sum_{t=1}^{h} |e_t|}{h}}{\frac{1}{h-1}\sum_{t=2}^{h} |y_t - y_{t-1}|},$$

where e_t is the prediction error and h is the number of predictions.

To summarize, the main steps of the prediction procedure, including the used methods and models, are as follows:

- Description step: computing the matrices of causalities using two measures, the Granger causality test and the transfer entropy.
- Reduction: this step includes the application of feature selection and dimension reduction methods. All available methods are applied using different reduction sizes k. The methods used are: PCA, kernel PCA, FCBF, factor analysis, and our method PEHAR with two variants, PEHAR-gc that uses the Granger causality test, and PEHAR-te that uses the transfer entropy. We evaluate values of k from 1 to 20, since using higher values does not improve the predictions.
- Prediction: applying all prediction models: ARIMA, VECM, LSTM, and the VAR model with shrinkage methods.
- Evaluation: evaluating the prediction accuracy using the RMSE and the MASE measures.

5 Results

In this section, we show the obtained results associated with the selection and the prediction steps, and we discuss the performances of the algorithms evaluated. Since we predict all variables of each dataset, the total number of variables evaluated is 1071. For the sake of readability, we present only a global comparison between methods and models by computing

Table 2 Prediction models applicability

Datasets	Models	Number of usages	
		RMSE	MASE
Stock-and-Watson	Arima	8	17
	Var+BayesianRidge	2	3
	Var+Lasso	3	2
	Var+Ridge	6	6
	Vecm	34	26
	Lstm	75	74
Ausmacrodata	Arima	49	56
	Var+BayesianRidge	0	1
	Var+Lasso	3	3
	Vecm	13	12
	Lstm	52	45
Sales-Transactions	Arima	15	20
	Var+BayesianRidge	7	24
	Var+Lasso	7	4
	Var+Ridge	12	12
	Vecm	294	264
	Lstm	477	487

the number of time where each algorithm outperform the others, and the best method's average RMSE and MASE. However, the detailed results of each variable separately, in addition to the implementations, are available online[1].

5.1 Evaluation of prediction models

In this part, we compare the applicability of the prediction models. To do this, we execute the prediction process on each dataset separately using the procedure described in Sect. 4.4, and based on the predictions obtained of the test data, we count the number of variables for which each prediction model leads to the best predictions. Consequently, the best models are those how lead to better predictions for more variables.

Table 2 shows the usages of the prediction models. We remark that each one outperforms the others on a subset of variables. But overall, the LSTM model is the most selected for Ausmacrodata and Stock-and-Watson datasets, and the VECM is the most selected for Sales-Transactions dataset. Shrinkage models are the less selected, especially for Ausmacrodata dataset, as they are not selected for any variable.

As for the ARIMA model, we notice a quite surprising result. It also provides good results and allows to have the best forecasts for a serious number of variables, especially for Australian datasets. Let H be the set of variables that are predicted better by the ARIMA model. The question that might arise is why such a simple univariate model, which needs just the history of a univariate time series to generate forecasts, predicts better the variables of H than the used multivariate models? We can tackle this question based on three explanations:

[1] https://github.com/Hmamouche/Large-Time-Series-Prediction.

1. Variables of the set H are independent, i.e., they do not require external information from other variables.
2. The used multivariate models do not exploit well the dependencies between variables of H and the associated predictors.
3. The selected variables by the used reduction methods are not the best predictors for variables of H.

In our opinion, the first point is the most logical, because it is very normal to find independent variables in a given dataset, those which are at the top of the causality graph. We can consider them as pure *Hubs*, since they just transfer information to other variables.

5.2 Evaluation of reduction methods

In this part, we compare the reduction methods with respect to the obtained prediction accuracy. Like the previous part, we evaluate the performance of the reduction methods in terms of RMSE and MASE measures.

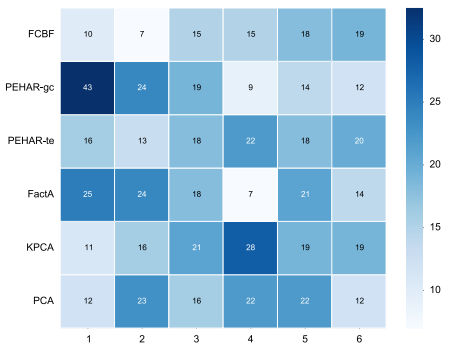
5.2.1 Methods applicability

The goal of this evaluation is to rank methods by their usages on variables of each datasets. Here again, we count the number of variables on which each method provides predictors leading to the best predictions. These results are shown in Fig. 3. The first column of each plot shows the number of variables where each method is the first best method. The second column shows the number of variables where each method is the second best method, and so on.

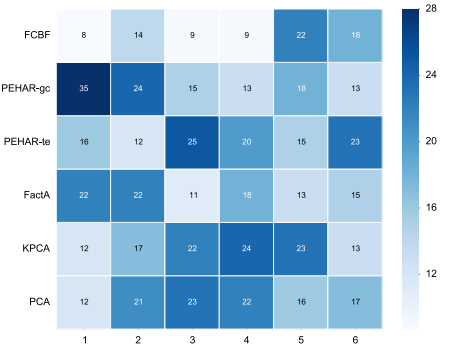
Figure 3 shows that there is no method that outperforms the others on all variables. However, the distribution methods applicability is not uniform. These results show that the proposed feature selection algorithm is the most appropriate reduction method for many variables compared to other methods. More precisely, it the first best algorithm for Ausmacrodata and Stock-and-Watson datasets in terms of RMSE and MASE, and it is the second best algorithm for Sales Transactions dataset. For example, in terms of RMSE, it is the best for 29 variables for Ausmacrodata, and the second algorithm is PCA with 11 variables. For Stock-and-Watson dataset, our algorithm is also the first with 43 variables, and the second algorithm is factor analysis with 25 variables. For Sales-Transactions data, factor analysis is the first method with 181 variables, followed by PCA with 157 variables, and then PEHAR-te with 150 variables.

5.2.2 Methods average RMSE and MASE

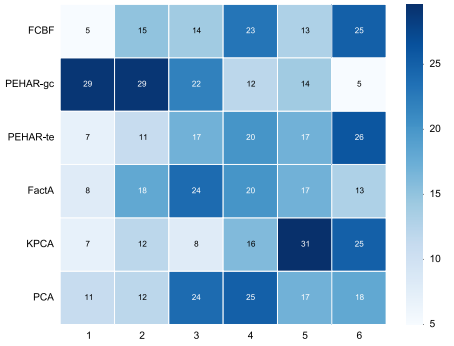
In this part, we compare method's best normalized RMSE and MASE obtained when applying reduction methods on variables of each datasets. The results show that the proposed algorithm provides the best average RMSE and MASE for Stock–Watson and Ausmacrodata datasets. For Sales-Transactions dataset, the best algorithms are factor analysis and PCA. Note that the MASE measure indicates a implicitly a comparison with a naive benchmark model (predicting next values based on the average of p last values, where p is the lag parameter), where the MASE is equal to 1. Therefore, acceptable MASE values must be less than 1. This is achieved for all datasets, except for the one, where only FCBF and PEHAR-gc methods satisfy this condition.



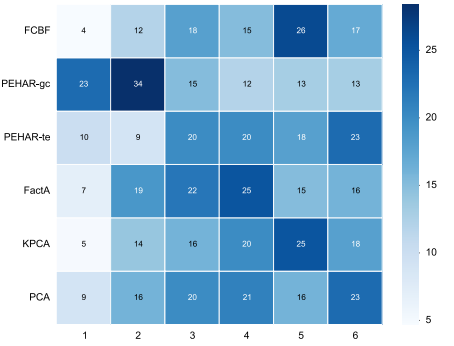
(a) Results of Stock&Watson data (Rmse).



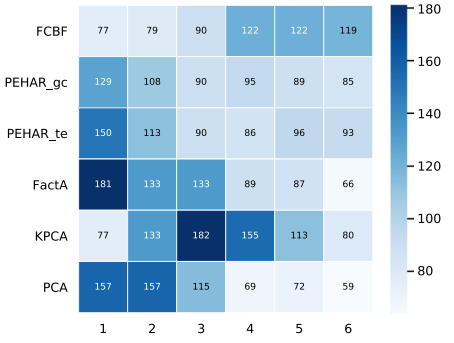
(b) Results of Stock&Watson data (Mase).



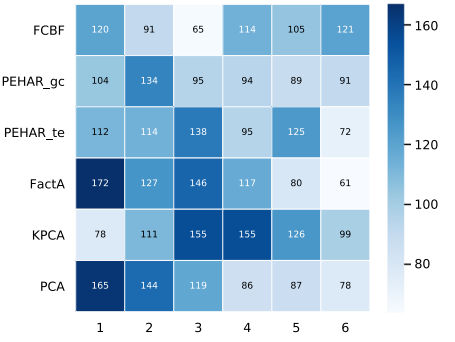
(c) Results of Ausmacrodata (Rmse).



(d) Results of Ausmacrodata (Mase).



(e) Results of Sales-Transactions data (Rmse).



(f) Results of Sales-Transactions data (Mase).

Fig. 3 Comparison between the applicability of the reduction methods in terms of RMSE and MASE measures. The number associated with a method on the column number i indicates the number of variables where the method is the ith best method

5.2.3 The number of predictors

Table 4 shows the average number of predictors found by each method which allow obtaining the best predictions. There is no major difference between methods. But for all of them, the obtained numbers are very small compared to the initial number of variables. We have tested

Table 3 Methods average RMSE and MASE for each dataset

Methods	Stock–Watson data		Ausmacrodata		Sales-transactions	
	RMSE	MASE	RMSE	MASE	RMSE	MASE
FCBF	0.14	0.98	0.12	0.59	0.20	0.81
PEHAR-gc	0.10	0.93	0.12	0.64	0.22	0.86
PEHAR-te	0.13	1.22	0.10	0.53	0.23	0.96
FactA	0.13	1.54	0.12	0.55	0.19	0.76
Kernel PCA	0.14	1.37	0.13	0.55	0.19	0.80
PCA	0.13	1.61	0.13	0.62	0.19	0.76

Table 4 The average best number of predictors

Methods	Stock–Watson data		Ausmacrodata		Sales-transactions	
	RMSE	MASE	RMSE	MASE	RMSE	MASE
FCBF	2	4	5	2	2	2
PEHAR-gc	5	5	5	6	3	2
PEHAR-te	4	2	4	4	2	2
FactA	3	2	3	1	3	2
Kernel PCA	2	2	4	4	3	3
PCA	4	3	3	2	3	2

different numbers, and we found out that using more than 20 variables decrease the prediction accuracy. This finding confirms results of work presented in [25], which is conducted on the Ausmacrodata dataset, where the authors show that the best prediction accuracy is obtained with a number of predictors less that 20-40. In our case, we found less number of predictors for all datasets, which does not exceed 10 (*cf.* Table 4). This reduction is very important for our research in terms of computational time, especially when using neural network prediction models. For example, in our case, the computation time of the LSTM model is the most important among the used models, and reducing the search space from all variables to less than 10 is a significant gain.

5.3 Discussion

Through these experiments, we demonstrated an application of the presented prediction process on three datasets, which allows to find the best reduction method and prediction model for each variable of the multivariate time series. The proposed algorithm (PEHAR) is included in the process and can also be used separately for feature selection, but it is specific to time series because it considers temporal dependencies between variables.

Starting with discussing the evaluations, the results show that the proposed method provides competitive predictions in terms of usage, and also in terms of average of RMSE and MASE compared to existing methods of type dimension reduction (PCA, kernel PCA, factor analysis) and feature selection (FCBF). Let us underline a difference between the used reduction methods, the used dimension reduction methods belong to the unsupervised learning category, because they reduce the dimension of variables just based on their statistical

characteristics, without considering the target variable. Usually, in forecasting, the data history contains passed values of target variables. In this case, it is worth to exploit it. This may be one cause why dimension reduction methods do not outperform in our experiments.

This is one of the motivations behind the proposed feature selection algorithm (PEHAR), which is a supervised learning model that considers relationships between predictors and target variables. However, it is possible to make an unsupervised version of it, by simply not considering causalities to the target in Eqs. 13 and 14. In this case, it will be close to the classic Hits algorithm.

The Ridge and Lasso shrinkage methods have been experimented on the first dataset in [31] and show good performance compared to the dynamic factor model. Unlike these findings, we found that the evaluated shrinkage methods are not competitive compared to models based on feature selection methods (*cf.* Table 2). In our opinion, the explanation of these results comes back to the basic principle of shrinkage models. They eliminate the impact of non-important variables, but these variables sometimes remain in the model. Therefore, the presence of these variables, even with low coefficients, may affect the prediction accuracy.

One common problem is how to allow feature selection methods to provide an automatic number of variables. This does not necessarily guarantee finding the optimal set of predictors, but it is useful in practice. For the proposed PEHAR algorithm, to allow it to generate his own number of variables, one idea is to fix a threshold on the *Hubs* scores using a statistical test. This method is practical if we use the Granger causality, since it is naturally computed using a statistical test. However, with transfer entropy as a causality measure, it is not clear how doing a significance test for the obtained values.

6 Scalability of the prediction process

When forecasting big time series, there are two problems related to the size of datasets; the number of observations, and the number of variables. In our opinion, the latter is the most difficult problem because some prediction models, like the ones based on artificial neural networks, are hard to distribute when they are applied to this type of data, or they require a change in their structure in order to adapt to distributed platforms. In this case, we may lose the original form of the model. In addition, we have seen in the previous section that using all variables does not yield to the best results; instead, the best predictions are obtained with a small number of variables (*cf.* Table 4), therefore, the more variables there are, the more difficult it is to find the best predictors.

In the other hand, time series with small number of variables and many observations do not present a real problem, because it is possible to use some techniques to reduce the number of observations like data discretization and quantization, or using sliding window techniques and batch based parallel training [33]. In addition, for financial forecasting, for example, all history of data is not necessary useful; that is, at a certain time, old observations are not very relevant.

From this point of view, we propose a distributed version of the presented prediction process that can process large multidimensional time series. The particularity of this system is that it rests on a distributed feature selection step as a means of reducing multivariate prediction models into many small models that can be executed in parallel. The implementation of this process is performed using Spark and Hadoop[1].

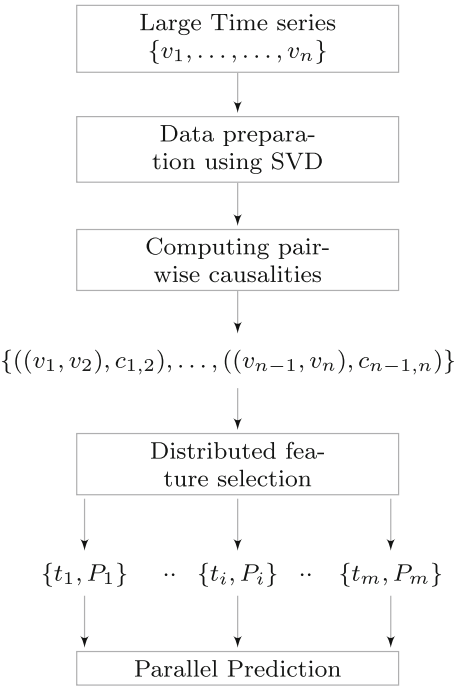


Fig. 4 Illustration of the scalable prediction process

6.1 The structure of the proposed system

To distribute the prediction process, we propose a system that boils down into the following steps:

- Storing time series in a set of partitions of variables.
- Preparing data.
- Computing dependencies between time series.
- Applying a distributed version of the PEHAR algorithm to determine predictors of each target variable, and generating models input ({target, predictors}).
- Running prediction models in parallel.

The schema of this system is shown in Fig. 4, where $\{v_1, \ldots, v_n\}$ are the original variables, $c_{i,j}$ represents the causality from v_i to v_j, and $\{t_i, P_i\}$ constitutes a prediction model input (target variable and the associated set of predictors), $\{t_1, \ldots, t_m\}$ are the target variables, and $\{P_1, \ldots, P_i\}$ is the set of the selected predictors. The size of each set P_i is $k << n$. Let us underline that different values of k can be tested, like in the experiments shown in the previous section. In this case, the results must be grouped by target variables to find the appropriate number of variables.

6.2 Data preparation

In big time series, the data may contain outliers and noises that may affect the quality of the information extracted from these data. On the other hand, eliminating outliers is a risky task because it is difficult to detect them. In addition, extreme values can provide important information and they may be detected as outliers. In our process, we set up a

preparation step by filtering data using low-rank matrix approximation via the singular value decomposition (SVD). SVD is a matrix decomposition technique and a generalization of the PCA technique for rectangular matrices. Consider a real matrix M of dimension $(m \times n)$, the SVD factorization of M is as follows:

$$M = USV^{\intercal}, \tag{23}$$

where U and V are two orthogonal and square matrix of size $(m \times m)$ and $(n \times n)$ resp. And S is the matrix of singular values (diagonal) of size $(m \times n)$.

SVD has many applications; one of them is dimension reduction. Another specific application of SVD is low-rank matrix approximation, called also the truncated SVD. The idea behind this application is to reconstruct the original matrix with only the most important singular values, i.e., those who represent the most variation in the data. The approximation of the matrix M using the top k singular components is expressed as follows:

$$M \sim U_k S_k V_k^{\intercal}, \tag{24}$$

where $U_k(m \times k)$, $S_k(k \times k)$, and $V_k(n \times k)$ are the truncated matrices of U, S, and V resp., according to the top k singular values (diagonal values of S). Several works have shown that this decomposition allows to filter and attenuate noise when processing signal data and image restoration [7,17,29]. We adopt this technique in our process using an implementation of SVD from the Spark machine learning library (MLlib) [23]. Nevertheless, we let this step optional, which means the user can avoid it in case of the input data do not contain noises, or if he wants to execute the process on the original data directly.

6.3 Computing causalities

In the actual version of this process, we store time series in partitions of variables on a Hadoop distributed file system and process them using Spark.

In this step, we describe the dependencies between variables by computing the matrix of causalities of each datasets using two measures; the Granger causality test and the transfer entropy. For both of these measures, this step requires the calculation of a set of tuples $G = (P \times P)$, where P is a set of time series. Consequently, all computations can be performed independently, and the global matrix of causalities can be stored by blocks of sub-matrix in the available nodes.

Note that here we suppose that we can compute each pairwise causality in at most one node, which means that we suppose that each pair of time series is short enough, in terms of the number of observations, to be processed in one node. For time series with many observations, a distribution of each single causality computation is required, and this is possible for the Granger causality for example, because it is mainly based on two VAR models, and the VAR model uses linear multiple regression to fit its parameters, which is scalable using for example the Spark MLlib library. However, this out of scope of this research because we concentrate on the problem of data with many variables.

6.4 Feature selection

In this step, we apply feature selection to reduce the set of input variables for multivariate prediction models. We use a distributed version of the proposed algorithm (*cf.* Algorithm 2). One property of the PEHAR algorithm is that it is flexible and easily scalable, because the steps of normalizing and updating the *Hubs* and *Authorities* vectors can be performed

using straightforward *MapReduce* operations. Algorithm 2 allows to have the Hubs scores of the predictor variables regarding a target variable. It requires as input a causality matrix of predictors M organized by pairs, as described in Sect. 4,

$$M = \{(v_1, v_2, c_{1,2}), \ldots, (v_{n-1}, v_n, c_{n-1,n})\}, \tag{25}$$

which can be expressed as a distributed list, where each element $(v_i, v_j, c_{i,j})$ contains the names of two predictor variables (v_i, v_j) and the causality $(c_{i,j})$ from the first variable to the second one. The algorithm also requires the vector of *predictor–target* causalities V and the list of predictors names. First, the algorithm computes the distributed matrix $G = M \times V$ as described in Sect. 3.2.2 and then applies the same steps of Algorithm 1 with a distributed way.

Algorithm 2: A Distributed PEHAR Algorithm.

Input : n: the number of predictor variables;
M: a distributed matrix of causalities between predictors of size $(n^2 \times 3)$;
V: the vector of *predictor-target* causalities of size $(n \times 2)$;
$colnames$: list of predictors names;
$iters$: the number of iterations;
min_error: the minimum error;
Output: *hubs*: the vector of *Hubs* scores of size n;

```
/* Compute the matrix G                                                    */
G ← M. join (V, on="M.v1"). rdd. map (lambda x: (x[0], (x[1], (x[2]) * (x[3])))));
/* Initialize the Hubs vector                                              */
hubs = colnames. rdd. map (lambda x: (x[0], 1/n))
while (i <iters or min_error <error) do
    hlast ← hubs;
    hubs ← hubs. map (lambda x: (x[0], 0));
    auths ← hubs;
    /* Update authorities                                                  */
    G' ← G. map (lambda x: (x[1][0], (x[0], x[1][1])));
    auths ← hubs.join (G'). map (lambda x: (x[1][1][0], x[1][0] * x[1][1][1])).
                reduceByKey (lambda x, y: x + y);
    /* Update hubs                                                         */
    hubs ← auths.join (G. map (lambda x: (x[1][0], (x[0], x[1][1]))).
                map (lambda x: (x[1][1][0], x[1][0] * x[1][1][1])).
                reduceByKey (lambda x, y: x + y);
    /* Normalization                                                       */
    normalize_rdd (hubs);
    normalize_rdd (auths);
    /* Compute the error                                                   */
    error ← hubs. join (hlast). map (lambda x: (abs (x[1][0] - x[1][1]))).
                reduce (lambda x, y: x + y);
    iter ← iter + 1;
end while
return hubs;
```

6.5 Prediction

After the selection step, we obtain a set of models inputs where each one contains a target variable and the related predictors. In the prediction step, we compute predictions by executing prediction models in parallel. In this case, parallel computing is sufficient and we do not need

to distribute prediction models, because each model will predict a target variable separately with a small number of predictors.

7 Conclusions

In this article, we have proposed a complete prediction process for large multidimensional time series prediction. The methodology adopted is based on (i) a graph representation of time series, in which we introduced a way for modeling dependencies between time series and (ii) on a selection step by representing a new feature selection algorithm that considers direct links between variables using *Hubs* and *Authorities* concept. The experiments are performed on real macroeconomic and financial datasets, and several algorithms are evaluated. We have found very important results, in terms of prediction accuracy, and also in terms of the number of predictor variables that lead to the best predictions. The proposed feature selection algorithm can be used separately from the process, and based on the evaluations performed, it is an interesting alternative to existing methods. Finally, a distributed version of the prediction process is presented, including the proposed feature selection algorithm, and implementation is provided using Spark/Hadoop.

References

1. Abedinia O, Amjady N, Zareipour H (2017) A new feature selection technique for load and price forecast of electrical power systems. IEEE Trans Power Syst 32(1):62–74. https://doi.org/10.1109/TPWRS.2016.2556620
2. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723. https://doi.org/10.1109/TAC.1974.1100705
3. Benzi M, Estrada E, Klymko C (2013) Ranking hubs and authorities using matrix functions. Linear Algebra Appl 438(5):2447–2474. https://doi.org/10.1016/j.laa.2012.10.022
4. Crone SF, Kourentzes N (2010) Feature selection for time series prediction—A combined filter and wrapper approach for neural networks. Neurocomputing 73(10):1923–1936. https://doi.org/10.1016/j.neucom.2010.01.017
5. Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
6. Forni M, Hallin M, Lippi M, Reichlin L (2000) The generalized dynamic-factor model: identification and estimation. Rev Econ Stat 82(4):540–554. https://doi.org/10.1162/003465300559037
7. Gan S, Chen Y, Zu S, Qu S, Zhong W (2015) Structure-oriented singular value decomposition for random noise attenuation of seismic data. J Geophys Eng 12(2):262. https://doi.org/10.1088/1742-2132/12/2/262
8. Granger CWJ (1980) Testing for causality. J Econ Dyn Control 2:329–352. https://doi.org/10.1016/0165-1889(80)90069-X
9. Hmamouche Y, Lakhal L, Casali A (2018) Predictors extraction in time series using authorities-hubs ranking. In: 2018 IEEE international conference on data mining workshops (ICDMW), pp 1070–1079. https://doi.org/10.1109/ICDMW.2018.00155
10. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

11. Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12(1):55–67. https://doi.org/10.2307/1267351
12. Hyndman RJ (2017) Australia datasets website. https://robjhyndman.com/hyndsight/ausmacrodata, (accessed 2020-04-12)
13. Hyndman RJ et al (2006) Another look at forecast-accuracy metrics for intermittent demand. Foresight: Int J Appl Forecast 4(4):43–46
14. Johansen S (1988) Statistical analysis of cointegration vectors. J Econ Dyn Control 12(2):231–254. https://doi.org/10.1016/0165-1889(88)90041-3
15. John GH, Kohavi R, Pfleger K (1994) Irrelevant features and the subset selection problem. In: Machine learning: proceedings of the eleventh international, Morgan Kaufmann, pp 121–129
16. Jolliffe IT (1986) Principal component analysis and factor analysis. Springer series in statistics. Springer, New York, pp 115–128
17. Kamm J, Nagy JG (1998) Kronecker product and SVD approximations in image restoration. Linear Algebra Appl 284(1):177–192. https://doi.org/10.1016/S0024-3795(98)10024-1
18. Karevan Z, Suykens JA (2018) Transductive feature selection using clustering-based sample entropy for temperature prediction in weather forecasting. Entropy 20(4):264
19. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. CoRR abs/1412.6980
20. Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. J ACM 46(5):604–632. https://doi.org/10.1145/324133.324140
21. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1):273–324. https://doi.org/10.1016/S0004-3702(97)00043-X
22. Koprinska I, Rana M, Agelidis VG (2015) Correlation and instance based feature selection for electricity load forecasting. Knowl-Based Syst 82:29–40. https://doi.org/10.1016/j.knosys.2015.02.017
23. Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S et al (2016) Mllib: machine learning in apache spark. J Mach Learn Res 17(1):1235–1241
24. Narendra PM, Fukunaga K (1977) A branch and bound algorithm for feature subset selection. IEEE Trans Comput 9(C—-26):917–922. https://doi.org/10.1109/TC.1977.1674939
25. Panagiotelis A, Athanasopoulos G, Hyndman RJ, Jiang B, Vahid F (2019) Macroeconomic forecasting for Australia using a large number of predictors. Int J Forecast 35(2):616–633
26. Press WH, Flannery BP, Teukolsky SA, Vetterling WT (1988) Numerical recipes in C: the art of scientific computing. Cambridge University Press, Cambridge
27. Schneeweiss H, Mathes H (1995) Factor analysis and principal components. J Multivar Anal 55(1):105–124. https://doi.org/10.1006/jmva.1995.1069
28. Schreiber T (2000) Measuring information transfer. Phys Rev Lett 85(2):461–464. https://doi.org/10.1103/PhysRevLett.85.461
29. Shim Y, Cho Z (1981) SVD pseudoinversion image reconstruction. IEEE Trans Acoust Speech Signal Process 29(4):904–909. https://doi.org/10.1109/TASSP.1981.1163632
30. Stock JH, Watson MW (2006) Chapter 10 forecasting with many predictors. In: Elliott CWJG, Timmermann A (eds) Handbook of economic forecasting, vol 1. Elsevier, Amsterdam, pp 515–554
31. Stock JH, Watson MW (2012) Generalized shrinkage methods for forecasting using many predictors. J Bus Econ Stat 30(4):481–493. https://doi.org/10.1080/07350015.2012.715956
32. Tan SC, San Lau JP (2014) Time series clustering: a superior alternative for market basket analysis. In: Proceedings of the first international conference on advanced data and information engineering (DaEng-2013), Springer, Singapore, pp 241–248
33. Turchenko V, Grandinetti L, Sachenko A (2012) Parallel batch pattern training of neural networks on computational clusters. In: 2012 International conference on high performance computing simulation (HPCS), pp 202–208. https://doi.org/10.1109/HPCSim.2012.6266912
34. Wang K, Su MYT (2002) Item selection by "Hub-authority" profit ranking. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, USA, KDD '02, pp 652–657. https://doi.org/10.1145/775047.775144
35. Yu L, Liu H (2003) Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Fawcett T, Mishra N (eds) Proceedings, twentieth international conference on machine learning, vol 2, pp 856–863
36. Zaremba A, Aste T (2014) Measures of causality in complex datasets with application to financial data. Entropy 16(4):2309–2349. https://doi.org/10.3390/e16042309
37. Zhong X, Enke D (2017) Forecasting daily stock market return using dimensionality reduction. Expert Syst Appl 67:126–139. https://doi.org/10.1016/j.eswa.2016.09.027

Youssef Hmamouche is a Post-doctoral researcher at LIS (UMR 7020) and LPL (UMR 7309) laboratories, Aix-Marseille Université, France. He received his Ph.D. in 2018 from Aix-Marseille Université. He does research in time series prediction, causality detection, and machine learning.

Lotfi Lakhal is Full Professor at Aix-Marseille Université, France. He received his Ph.D. (1986) and the Habilitation for Directing Research (1992) from the University of Nice-Sophia Antipolis, France. He joined the Department of Mathematics and Computer Science of Blaise Pascal University, Clermont-Ferrand, France, and became a full professor in 1993. In 2000, he joined the Mediterranean University, IUT d'Aix-en-Provence, France. His research, at LIS laboratory UMR 7020, interests include statistical databases, database design, formal concept analysis, data mining, and data warehouses.

Alain Casali obtained the Ph.D. degree in computer science from the Aix Marseilles University (France) in 2005. He is an assistant professor at the Aix Marseilles University - IUT of Aix en Provence and is a member of the LIS laboratory. He studies lattice algorithmic and temporal database mining.