


Time Series Forecasting with Many Predictors

Shuo-Chieh Huang and Ruey S. Tsay * 

Booth School of Business, University of Chicago, 5807 S. Woodlawn Avenue, Chicago, IL 60637, USA; shuochieh@chicagobooth.edu

* Correspondence: ruey.tsay@chicagobooth.edu

Abstract: We propose a novel approach for time series forecasting with many predictors, referred to as the GO-sdPCA, in this paper. The approach employs a variable selection method known as the group orthogonal greedy algorithm and the high-dimensional Akaike information criterion to mitigate the impact of irrelevant predictors. Moreover, a novel technique, called peeling, is used to boost the variable selection procedure so that many factor-relevant predictors can be included in prediction. Finally, the supervised dynamic principal component analysis (sdPCA) method is adopted to account for the dynamic information in factor recovery. In simulation studies, we found that the proposed method adapts well to unknown degrees of sparsity and factor strength, which results in good performance, even when the number of relevant predictors is large compared to the sample size. Applying to economic and environmental studies, the proposed method consistently performs well compared to some commonly used benchmarks in one-step-ahead out-sample forecasts.

Keywords: supervised principal component analysis; diffusion index; Lasso; dynamic dependence; group orthogonal greedy algorithm

MSC: 62M10; 62M20; 62J05



Citation: Huang, S.-C.; Tsay, R.S. Time Series Forecasting with Many Predictors. *Mathematics* **2024**, *12*, 2336. <https://doi.org/10.3390/math12152336>

Academic Editors: Luiz Koodi Hotta and Pedro Luiz Valls Pereira

Received: 12 June 2024

Revised: 17 July 2024

Accepted: 21 July 2024

Published: 26 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the current data-rich environment, many predictors are often available in time series forecasting. However, conventional methods have encountered serious difficulties in exploiting the information contained in such high-dimensional data. In particular, the curse of dimensionality often leads to unreliable forecasts when the conventional methods are applied naively. To exploit the information in the high-dimensional predictors, the use of latent factors has emerged among the first successful approaches. See, for instance, Peña and Box [1], Stock and Watson [2,3], Lam et al. [4], and Chapter 6 of Tsay [5]. These factor-based methods are also widely used in the econometric analysis of high-dimensional time series (e.g., [6–9]).

Nevertheless, existing factor-based methods may still be inadequate, and several issues have been constantly observed in practice. For example, weak factors are prevalent in real-world data, and extracting factors using all predictors may not be optimal [10]. In addition, some factor models may still contain many parameters when the factor dimension is not small. On the other hand, Kim and Swanson [11] reported, from an extensive experiment, that combining shrinkage methods with factor-based approaches can yield more accurate forecasts among many targeted macroeconomic time series. This implies that there might exist some irrelevant predictors in an empirical application. The recently proposed scaled principal component analysis of Huang et al. [12] and the supervised dynamic PCA (sdPCA) of Gao and Tsay [13] partially remedy this issue by constructing factors in a supervised fashion, so that the effects of irrelevant predictors are reduced. Limited experience indicates that when the number of predictors far exceed the sample size, or when the predictors are highly correlated, the performance of such supervised approaches can still be compromised.

To efficiently extract predictive factors from high-dimensional data, in this paper, we propose a new method that blends variable selection in factor estimation. The proposed method uses the group orthogonal greedy algorithm (GOGA, Chan et al. [14]) to screen variables that are relevant to the prediction problem at hand, and applies the sdPCA of Gao and Tsay [13] to extract factors from the selected variables. The orthogonal greedy algorithm, as well as its variant GOGA for grouped predictors, have been employed for variable selection for high-dimensional linear models, especially with dependent data and highly correlated predictors [14–16]. Since both GOGA and sdPCA serve to minimize the effects of noisy irrelevant variables, the combined procedure, which we call GO-sdPCA, can effectively construct highly predictive factors.

It is worth pointing out that the novelty of our method lies in a key step that successfully combines variable selection with factor-based methods. Indeed, variable selection techniques commonly employed, such as the Lasso of Tibshirani [17] or the OGA of Ing and Lai [15], tend to select a sparse representation of the data. These methods will select only a few variables when many variables are driven by common factors and thereby highly correlated. For factor estimation, this can be undesirable, since factor recovery typically benefits from the many variables that are loaded on the shared factors (see, e.g., [4,13]). This issue can be even more pronounced when the factors are strong, in which case the factors can be accurately recovered by employing many predictors. To circumvent this predicament, we propose using a “peeling” technique, which repeatedly applies GOGA to the data after previously selected variables are dropped from the set of candidate predictors. In this way, our method can select more variables in the model and discriminate between relevant and irrelevant predictors. In Section 4, we introduce the proposed method in more detail. To better motivate the proposed method, we briefly review the orthogonal greedy algorithm (OGA) and its variants in Section 2. We also review the high-dimensional information criteria that are often used to balance the model complexity and the model fit along OGA iterations. The sdPCA is reviewed in Section 3, where we particularly emphasize the method’s design principles.

We use simulation studies and some empirical analyses to examine the performance of the proposed GO-sdPCA approach. In addition to comparing with some factor-based methods, such as the diffusion index approach of Stock and Watson [2,3], the sdPCA, and the time series factor model of Lam et al. [4], we also compare GO-sdPCA against the Lasso and the random forests of Breiman [18], both of which are widely-used and versatile tools in machine learning. We found that GO-sdPCA improves upon most of factor-based approaches and fares well with selected competing methods, even when the number of relevant variables is comparable with the sample size. The simulation results are discussed in Section 5.

In Section 6, we apply the proposed method to two real datasets. The first dataset is the well-known FRED-MD macroeconomic data [9]. Time series forecasting has been a vital topic both in econometric research and in policy-making pertaining to these data set. The other dataset contains the hourly particulate matter (PM_{2.5}) measurements in Taiwan. The PM_{2.5} data play an important role in the environmental studies, as well as informing public health policies. However, since its measurements are taken over time and space, the dataset is naturally of high dimension and presents serious challenges to forecasting. Our findings demonstrate that the proposed method consistently offers more accurate forecasts than the competing methods, validating its practical utility.

2. Orthogonal Greedy Algorithm

In this section, we briefly review the (group) orthogonal greedy algorithm. The OGA is an iterative algorithm that sequentially chooses predictors to form a regression model. Theoretically grounded in approximation theory [19], the OGA is also easy to implement computationally. In the statistical learning and high-dimensional linear regression literature, Barron et al. [20] and Ing and Lai [15] analyzed its convergence rate as

well as variable screening capability. Its generalization to grouped predictors was studied by Chan et al. [14] with application to threshold autoregressive time series models.

Throughout this paper, we denote by $\mathbf{y} = (y_1, \dots, y_n)^\top$ the data of the response variable we are interested in, where n is the sample size. We also have data for the p predictors, $\{\mathbf{x}_{(j)} : j = 1, 2, \dots, p\}$, where $\mathbf{x}_{(j)} = (x_{1,j}, \dots, x_{n,j})^\top$. The OGA is defined as follows. Starting with $\hat{J}_0 = \emptyset$ and $\mathbf{u}^{(0)} = \mathbf{y} = (y_1, y_2, \dots, y_n)^\top$, the OGA computes, at iteration $k = 1, 2, \dots$,

$$\hat{J}_k = \arg \min_{1 \leq j \leq p} \|\mathbf{u}^{(k-1)} - \mathbf{x}_{(j)}(\mathbf{x}_{(j)}^\top \mathbf{x}_{(j)})^{-1} \mathbf{x}_{(j)}^\top \mathbf{u}^{(k-1)}\|^2, \tag{1}$$

and updates

$$\begin{aligned} \hat{J}_k &= \hat{J}_{k-1} \cup \{\hat{J}_k\}, \\ \mathbf{u}^{(k)} &= (\mathbf{I} - \mathbf{H}_{(k)})\mathbf{y}. \end{aligned}$$

where $\mathbf{H}_{(k)}$ is the orthogonal projection matrix associated with the linear space spanned by $\{\mathbf{x}_{(j)} : j \in \hat{J}_k\}$. Clearly, the OGA sequentially selects the variable to include in the model; the set \hat{J}_k denotes the index set corresponding to the predictors already selected at iteration k . Intuitively, in each iteration, OGA selects the variable that best explains the current residuals. There are numerical schemes to speed up the computation of the residuals such as using sequential orthogonalization. We refer to Ing and Lai [15] and Chan et al. [14] for details.

For the purpose of this paper, we will make use of a slight generalization of the OGA that deals with grouped predictors. Chan et al. [14] studied the group OGA (GOGA) and applied the method to estimate threshold time series models. Consider the j -th predictor, $\{x_{t,j}\}_{t=1}^n$. Instead of being a scalar predictor, it contains d_j component predictors, $\{(x_{t,j,1}, \dots, x_{t,j,d_j}) : t = 1, 2, \dots, n\}$, thereby forming a ‘‘group’’ predictor. Then, by substituting

$$\mathbf{x}_{(j)} = \begin{pmatrix} x_{1,j,1} & x_{1,j,2} & \dots & x_{1,j,d_j} \\ x_{2,j,1} & x_{2,j,2} & \dots & x_{2,j,d_j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,j,1} & x_{n,j,2} & \dots & x_{n,j,d_j} \end{pmatrix}$$

in (1), the procedure becomes the GOGA.

After K_n iterations, the GOGA selects the (group) predictors corresponding to $\hat{J}_{K_n} = \{\hat{J}_1, \hat{J}_2, \dots, \hat{J}_{K_n}\}$. Bias-variance trade-off manifests when selecting the number of predictors to be included, as a large K_n may lead to over-fitting. To select a desirable model complexity, Ing [16] suggests using the high-dimensional Akaike information criterion (HDAIC). The HDAIC of the model at iteration k is defined as

$$\text{HDAIC}(\hat{J}_k) = \left(1 + C \frac{k \log p}{n}\right) \hat{\sigma}_{(k)}^2, \tag{2}$$

where $\hat{\sigma}_{(k)}^2 = n^{-1} \|\mathbf{u}^{(k)}\|^2$, and C is a constant to be tuned. Then, the model selected by HDAIC is

$$\hat{J}_{\hat{k}} = \{\hat{J}_1, \dots, \hat{J}_{\hat{k}}\}, \quad \text{where } \hat{k} = \arg \min_{1 \leq k \leq K_n} \text{HDAIC}(\hat{J}_k). \tag{3}$$

Theoretically, Ing [16] proved that the resulting model selected by OGA+HDAIC adapts to the underlying sparsity structure. In particular, consider the high-dimensional regression model,

$$y_t = \sum_{j=1}^p \beta_j x_{t,j} + \epsilon_t.$$

Then, the conditional mean squared prediction error of OGA+HDAIC is of rate

$$\begin{cases} \left(\frac{\log p}{n}\right)^{1-1/2\gamma}, & \text{if } \sum_{j \in J} |\beta_j| \leq D_1 \left(\sum_{j \in J} \beta_j^2\right)^{(\gamma-1)/(2\gamma-1)} \text{ for all } J \\ \frac{\log n \log p}{n}, & \text{if } \sum_{j \in J} |\beta_j| \leq D_2 \max_{j \in J} |\beta_j| \text{ for all } J \\ \frac{k_0 \log p}{n}, & \text{if } \min_{j: \beta_j \neq 0} |\beta_j| \geq D_3, \end{cases}$$

for some $\gamma \geq 1$ and positive constants D_1, D_2, D_3 . See Theorem 3.1 of Ing [16] for details. Note that these rates are minimax optimal, and are automatically achieved by OGA+HDAIC without any knowledge about the sparsity pattern. In this paper, we leverage this property to select good predictors in constructing forecasts. In this way, the variables used in our method is carefully selected in a supervised fashion, which is more effective than employing all observed predictors in the high-dimensional data.

3. Supervised Dynamic PCA

Next, we review the supervised dynamic PCA (sdPCA) proposed by Gao and Tsay [13] for forecasting. The sdPCA is a factor-based forecasting approach, which took a major role in the literature of time series forecasting. Tailored to incorporate dynamic information, the sdPCA has been shown to outperform some existing factor-based approaches such as the diffusion index model of Stock and Watson [2,3] and the scaled PCA of Huang et al. [12].

We first outline the sdPCA procedure. Given a forecast horizon h , the sdPCA first constructs intermediate predictions using each individual predictor and its lagged values. For instance, one can regress y_{t+h} on $x_{t,j}, x_{t-1,j}, \dots, x_{t-q_2+1,j}$, where $q_2 \in \mathbb{N}$ is a user-specified lag value, and obtain the fitted values

$$\hat{\mu}_j + \sum_{k=0}^{q_2-1} \hat{\gamma}_{j,k} x_{t-k,j} =: \hat{\mu}_j + \hat{x}_{t,j}, \tag{4}$$

where $\hat{\mu}_j$ is the intercept estimate and $\hat{\gamma}_{0,1}, \dots, \hat{\gamma}_{q_2-1,j}$ are the slope estimates. For different values of j , the number of lags q_2 used in the regression can differ. For instance, q_2 can be selected by the BIC. Then, with the constructed intermediate predictions, $\hat{x}_{t,j}$, the sdPCA apply PCA to estimate a lower dimensional factor $\hat{\mathbf{g}}_t \in \mathbb{R}^r$, where $r < p$. Therefore, the data to which the PCA applies, $\hat{x}_{t,1}, \dots, \hat{x}_{t,p}$, are in the same unit (as y_t), which scales the variables according to their predictive power.

Finally, one may employ a linear model for the predictive equation,

$$y_{t+h} \sim \hat{\alpha} + \hat{\beta}^\top \hat{\mathbf{g}}_t, \tag{5}$$

where $\hat{\alpha}$ and $\hat{\beta}$ are intercept and slope estimates, respectively, and \sim signifies that we run the linear regression while the underlying relationship of y_{t+h} and $\hat{\mathbf{g}}_t$ may not be exactly linear. Gao and Tsay [13] also suggests using Lasso to estimate (5) instead of the usual linear regression if the number of common factors is large. Additionally, one can also include some lags of $\hat{\mathbf{g}}_t$ from (5) and let Lasso perform variable selection.

The sdPCA has several advantages over the conventional PCA methods. First, the PCA is not scale-invariant. On the contrary, the sdPCA constructs predictors that are in the same unit, which naturally scales the predictors according to their predictive capabilities. Second, instead of performing PCA directly on contemporaneous data x_t , the sdPCA sources from the lagged information in $x_{t-l}, l = 0, 1, \dots, q_2 - 1$, where $x_t = (x_{t,1}, \dots, x_{t,p})^\top$. For the

conventional PCA to use the lagged information, one often needs to augment the data by appending the lagged variables so that PCA is performed on $(\mathbf{x}_t, \dots, \mathbf{x}_{t-q_2+1})$, which leads to even higher dimensionality. Lastly, the usual PCA is performed in an unsupervised fashion, whereas the sdPCA constructs the factors in a supervised fashion. While the usual principal component directions are not necessarily predictive of the response, factors extracted by sdPCA can potentially yield better forecasts. In fact, Gao and Tsay [13] have shown that sdPCA has a lower mean square forecasting error than the approaches of Stock and Watson [2,3] and Huang et al. [12].

In this paper, we employ the sdPCA to capitalize on the aforementioned properties. However, for noisy high-dimensional data, the performance of sdPCA may be severely compromised. Hence, it is desirable to perform a careful dimension reduction before applying sdPCA. In the next section, we describe the proposed procedure, which combines GOGA and HDAIC with sdPCA to improve the accuracy of prediction.

4. The Proposed GO-sdPCA

In this section, we introduce the proposed method, GO-sdPCA, which screens variables by the GOGA and HDAIC and then estimates factors by the sdPCA approach. In order to facilitate factor recovery, we apply a “peeling” technique to select more factor-relevant variables in the new procedure.

To tackle the difficulties encountered by the factor-based approaches when applied to high-dimensional data, our method begins with dimension reduction by employing the GOGA introduced in Section 2. Because of the serial dependence in the data, it is beneficial to select variables based not only on cross-sectional correlation, but also on the lagged information. To this end, for each predictor $\mathbf{x}_j = (x_{1,j}, \dots, x_{n,j})^\top$, we consider the group predictor,

$$\mathbf{x}_{(j)} = \begin{pmatrix} x_{q_1,j} & x_{q_1-1,j} & \dots & x_{1,j} \\ x_{q_1+1,j} & x_{q_1,j} & \dots & x_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-h,j} & x_{n-h-1,j} & \dots & x_{n-h-q_1+1,j} \end{pmatrix}, \quad j = 1, 2, \dots, p, \tag{6}$$

where q_1 is a pre-specified integer for the number of lagged values to consider, and $h \in \mathbb{N}$ is the forecast horizon. Let

$$\mathbf{y}_{q_1} = (y_{q_1+h}, y_{q_1+h+1}, \dots, y_n)^\top \tag{7}$$

be the response vector. Then, we employ the GOGA algorithm in Section 2, with $\mathbf{x}_{(j)}$ and \mathbf{y} substituted by (6) and (7), respectively. For notational simplicity, we drop the dependence on q_1 and write \mathbf{y}_{q_1} as \mathbf{y} in the following. Since each group predictor already incorporates past information, the GOGA algorithm can select variables while accounting for the dynamic information.

We also employ the HDAIC (defined in (2)) after GOGA to prevent the inclusion of irrelevant variables. Since the above procedure depends on the number of GOGA iterations K_n , the HDAIC constant C in (2), the number of lags q_1 used in forming the group predictors, and the pool of candidate predictors $\mathcal{I} = \{1, 2, \dots, p\}$, we conveniently denote it as a set-valued function,

$$\hat{J} = \mathcal{A}(K_n, C, q_1, \mathcal{I}), \tag{8}$$

where \hat{J} is the index set outputted by GOGA+HDAIC, as in (3). For completeness, we summarize the GOGA+HDAIC method introduced in Section 2 in Algorithm 1.

Algorithm 1: GOGA+HDAIC (\mathcal{A})

Input: Number of maximum iterations K_n , HDAIC parameter C , number of lags q_1 , candidate set \mathcal{I}

Initialization: $\mathbf{u}^{(0)} = \mathbf{y}$; selected index $\hat{J} = \emptyset$

1 **for** $k = 1, 2, \dots, K_n$ **do**

2 Select

$$\hat{j}_k = \arg \min_{j \in \mathcal{I}} \|\mathbf{u}^{(k-1)} - \mathbf{x}_{(j)}(\mathbf{x}_{(j)}^\top \mathbf{x}_{(j)})^{-1} \mathbf{x}_{(j)}^\top \mathbf{u}^{(k-1)}\|^2,$$

where $\mathbf{x}_{(j)}$ is defined in (6).

3 Update $\hat{J} \leftarrow \hat{J} \cup \{\hat{j}_k\}$

4 Update $\mathbf{u}^{(k)} = (\mathbf{I} - \mathbf{H}_{(k)})\mathbf{y}$, where $\mathbf{H}_{(k)}$ is the projection matrix associated with $\{\mathbf{x}_{(j)} : j \in \hat{J}\}$

5 **end**

6 Choose, as in (3),

$$\hat{m} = \arg \min_{1 \leq m \leq K} \text{HDAIC}(\{\hat{j}_1, \dots, \hat{j}_m\}),$$

where HDAIC is defined in (2) with C set to the inputted value.

Output: selected indices $\{\hat{j}_1, \dots, \hat{j}_{\hat{m}}\}$

Because the orthogonal residuals are used in the greedy search, GOGA tends to differentiate highly correlated predictors and only selects those predictors that explain distinct (close to orthogonal) directions of the response. However, in factor estimation, the relevant variables loaded on the common factors tend to be highly correlated, and failing to employ these correlated predictors may lose some statistical efficiency for the inference of the underlying factors (the blessing of dimensionality, [4,13]). Therefore, to encourage GOGA to screen the factor-relevant, correlated predictors, we propose a technique, referred to as “peeling,” which allows careful inclusion of more variables. The idea is to repeatedly apply GOGA+HDAIC, with variables selected from previous implementations discarded from the candidate set. Hence, in each iteration, GOGA+HDAIC is forced to select a new set of variables that best predict \mathbf{y} . Formally, the peeling algorithm is described in Algorithm 2. Let M be the number of peeling iterations. In the following, we denote the peeling procedure, which depends on M , K_n , and C , by the set-valued function \mathcal{P} . Namely,

$$\hat{Q} = \mathcal{P}(M, K_n, C, q_1).$$

Algorithm 2: Peeling (\mathcal{P})

Input: Number of peeling iterations M , number of GOGA iterations K_n , HDAIC parameter C , number of lags q_1 in group predictors

Initialization: $\hat{Q} = \emptyset, \mathcal{I} = \{1, 2, \dots, p\}$

1 **for** $m = 1, 2, \dots, M$ **do**

2 Run GOGA+HDAIC

$$\hat{j}^{(m)} = \mathcal{A}(K_n, C, q_1, \mathcal{I})$$

3 Update $\hat{Q} \leftarrow \hat{Q} \cup \hat{j}^{(m)}$

4 Discard selected variables from the candidate set $\mathcal{I} \leftarrow \mathcal{I} - \hat{Q}$

5 **end**

Output: selected indices \hat{Q}

It is worth noting that peeling is different from simply running GOGA for many iterations. Although GOGA also selects distinct variables along its iteration because of orthogonalization (that is, GOGA does not revisit any previously selected variable), peeling, which discards previously selected variables from the candidate set, would produce very different results. For high-dimensional data, running many iterations of GOGA will lead to extremely small residuals, which may no longer carry sufficient variations to detect the remaining relevant variables (with a finite sample). On the other hand, peeling re-starts GOGA in every iteration, using \mathbf{y} instead of the previous residuals in the GOGA algorithm. Since GOGA is re-started with a smaller pool of candidate variables, the residuals in peeling will not shrink to zero after many (potentially more than n) variables are already contained in $\hat{\mathbf{Q}}$. We also remark that the idea of peeling is akin to the random forest, which uses randomly selected variables in building each tree. Hence, peeling can be used to detect the weak predictors in the case where a few of the predictors are highly predictive and many others are only weakly predictive (see, e.g., Chapter 8 of [21]).

After variable selection by peeling, sdPCA is employed to estimate the factors $\hat{\mathbf{f}}_t$ from the selected variables. Let q_2 be the number of lags used in constructing the intermediate predictions (4). Finally, the predictive model

$$y_{t+h} = \sum_{k=1}^{q_3} \alpha_k y_{t-q+1} + \boldsymbol{\beta}^\top \hat{\mathbf{f}}_t + \epsilon_{t+h}, \quad (9)$$

is estimated by OLS or Lasso, where q_3 is the number of autoregressive variables. The above procedure, which combines GOGA, peeling, and sdPCA, is called GO-sdPCA.

In closing this section, we briefly discuss the selection of the number of lags. In practice, the number of lags q_1 used in GOGA, q_2 used in the sdPCA step, and q_3 in the predictive equation can all differ. After factor extraction, selecting q_3 in (9) amounts to selecting the AR lags in an autoregressive model with exogenous inputs (ARX model), for which one can routinely apply the AIC or the BIC. For selecting q_1 , a larger q_1 implies that GOGA can detect the predictors whose distant lags have some predictive power. However, because a group of predictors is included in the linear regression, much variations are consumed in each GOGA iteration, resulting in a noisy selection path. Moreover, this also implies less number of GOGA iterations K_n can be allowed in each peeling iteration. To boost the variable selection capability of GOGA and peeling, we therefore suggest using a smaller q_1 , and if the distant past of the selected predictors is important, employ a larger q_2 in the sdPCA step. Finally, for selecting q_2 , we may use the data-driven technique of Gao and Tsay [13]: first, choose a maximum number of lags \bar{Q} , and in constructing the intermediate predictions in the sdPCA step, use the BIC to select the appropriate lag $q_2 \leq \bar{Q}$. For tuning these parameters, one can also reserve a validation data and pick the combination that yields the lowest forecast errors on the validation set. In Appendix A, we conduct a simulation study to examine the sensitivity of GO-sdPCA to these tuning parameters. The results show while choosing a smaller q_1 is indeed best for variable selection, the forecasting performance is quite robust for most reasonable parameter choices.

5. Simulation Studies

In this section, we assess the finite-sample performance of the proposed GO-sdPCA method via simulation studies. Some existing factor-based methods are employed as benchmarks, such as the diffusion index approach of Stock and Watson [2,3], the time series factor model of Lam et al. [4], and the supervised dynamic PCA of Gao and Tsay [13]. These benchmarks are referred to as SW, LYB, and sdPCA, respectively.

After factor estimation, the predictive model

$$y_{t+1} = \sum_{k=1}^q \alpha_k y_{t-q+1} + \boldsymbol{\beta}^\top \hat{\mathbf{f}}_t + \epsilon_{t+1}, \quad (10)$$

is estimated by OLS, where q is an integer specified later, and $\hat{\mathbf{f}}_t$ is constructed using different approaches. For the proposed GO-sdPCA (shorthanded as GsP* hereafter), we set $C = 2$ in the HDAIC definition (2) and use 10 peeling iterations. In each peeling iteration, GOGA is applied with q_1 set to 2 for the group predictors. Then, in the sdPCA step, r factors are extracted with $q_2 = q$ lags used in constructing the intermediate predictions in (4). To demonstrate the usefulness of the peeling technique, we also consider implementing GO-sdPCA by naively combining GOGA+HDAIC with sdPCA. That is, we only run one peeling iteration and the variables selected are exactly the ones selected by GOGA+HDAIC. This method is denoted as GsP in the following. Similarly, the forecasts of sdPCA are constructed by estimating (10) with the factors estimated as in Section 3. The time series factors of Lam et al. [4] are estimated using the eigenanalysis of a non-negative definite matrix computed from the autocovariance matrices at nonzero lags. In our implementation, q lags of past predictors are used by LYB in the eigenanalysis. Finally, SW follows that of Stock and Watson [2,3] and uses PCA to extract the factors.

In addition, we employ some commonly used alternatives, including the Lasso of Tibshirani [17] and the random forests of Breiman [18] as competing methods. The Lasso is a versatile tool for building sparse linear regression models, while the random forest (RF) excels in capturing non-linear relationships. Recently, Chi et al. [22] investigated the asymptotic consistency of RF for high-dimensional data. See also Saha et al. [23] for application of RF to dependent data. The tuning parameters for the Lasso are selected by the BIC, as suggested by Medeiros and Mendes [24], whereas we adopt the hyper-parameters for RF as recommended by the randomForest [25] package in R (ver. 4.2). Therefore, about one-third of the candidate variables is randomly selected at each split. For both methods, q lags of the dependent variable and the predictors are used for fair comparison.

Simulation Designs and Results

In the simulations, we consider three data generating processes (DGP) to generate the synthetic data. Throughout the experiments, we use one-step-ahead forecasts ($h = 1$) where each method makes a forecast for y_{n+1} , which is not in the training sample. The root mean squared forecast errors, averaged over 500 Monte Carlo simulations, are used for comparing different approaches.

DGP 1. Let $\mathbf{f}_t \sim_{\text{i.i.d.}} N(0, \mathbf{I}_{r_{\text{DGP}}})$, where $r_{\text{DGP}} \in \mathbb{N}$ is the number of underlying factors. The predictors $\mathbf{x}_t \in \mathbb{R}^p$ are generated by

$$\mathbf{x}_t = \mathbf{B}\mathbf{f}_t + 2\delta_t,$$

where $\{\delta_t\}$ are independent p -dimensional t -distributed random vectors with independent components and five degrees of freedom, and $\mathbf{B} \in \mathbb{R}^{p \times r_{\text{DGP}}}$ has independent $\text{Unif}(-2, 2)$ entries, with $p - s$ rows randomly set to zero. That is, \mathbf{B} only has s nonzero rows. Randomly generate $\beta_1 = (\beta_{1,1}, \dots, \beta_{r_{\text{DGP}},1})^\top$ and $\beta_2 = (\beta_{1,2}, \dots, \beta_{r_{\text{DGP}},2})^\top$ via $\beta_{j,1} \sim \text{Unif}(1.0, 2.5)$ and $\beta_{j,2} \sim \text{Unif}(-2.0, -0.8)$. Finally,

$$y_t = 0.6y_{t-1} + 0.2y_{t-2} + \beta_1^\top \mathbf{f}_{t-1} + \beta_2^\top \mathbf{f}_{t-2} + \epsilon_t,$$

where $\{\epsilon_t\}$ are independent standard Gaussians.

In this DGP, the parameter s dictates both the number of relevant variables and the strength in recovering the factors. The larger s is, the stronger the factors are. Factor strength plays a critical role in factor recovery [2–4]. In practice, s is seldom known. Therefore, we will consider the cases $s \in \{0.25n, 0.5n, 0.75n\}$, where n is the sample size, to see whether the methods adapt well to various levels of factor strength.

Table 1 reports the root mean squared forecast error (RMSFE) averaged over 500 Monte Carlo simulations. Across all sparsity levels $s \in \{0.25n, 0.5n, 0.75n\}$, the proposed GsP* delivered the most accurate forecasts and the sdPCA ranks the second. This suggests

that the proposed peeling procedure and GOGA improves accuracy in forecasting. The GsP, which naively combines GOGA+HDAIC with sdPCA, shows limited forecasting capabilities with RMSFE being higher than those of the Lasso. This indicates, again, the peeling strategy markedly improved the forecasting performance by selecting more factor-relevant variables. The other forecasting methods, including SW, LYB and RF, seem to suffer from the effect of employing many irrelevant variables in the high-dimensional data. We remark that DGP 1 is essentially a sparse model because r_{DGP} used is relatively small. Therefore, it is not surprising to see that Lasso fares reasonably well.

Table 1. Root mean squared forecast error of various competing methods. The data are generated from DGP 1, and the results are averaged over 500 Monte Carlo simulations. n and p stand for the sample size and number of observed predictors. r_{DGP} is defined in DGP 1, and r is the number of factors extracted using various methods.

(r_{DGP}, s)	GsP*	GsP	sdPCA	SW	LYB	Lasso	RF
$(n, p, r) = (200, 1000, 10)$							
(5, 50)	1.894	2.443	1.993	2.657	2.136	2.207	2.860
(10, 100)	2.406	3.299	2.488	3.288	2.909	2.831	4.353
(15, 150)	3.315	3.908	3.584	5.368	5.201	3.537	5.776
$(n, p, r) = (400, 2000, 30)$							
(10, 100)	2.391	2.947	2.590	3.482	2.077	2.544	4.428
(20, 200)	3.020	4.355	3.242	4.626	3.657	3.679	6.604
(30, 300)	3.818	5.581	4.155	5.666	4.651	4.450	8.253

DGP 2. In this DGP, \mathbf{x}_t is generated via a vector MA(1) model:

$$\mathbf{x}_t = \delta_t + 0.8\mathbf{B}\delta_{t-1},$$

where \mathbf{B} is a randomly drawn $p \times p$ matrix of rank r_{DGP} . The coefficients $\beta_1 = (\beta_{1,1}, \dots, \beta_{p,1})^\top$ and $\beta_2 = (\beta_{1,2}, \dots, \beta_{p,2})^\top$ are randomly generated via $\beta_{j,1} \sim U(1.0, 3.0)$ and $\beta_{j,2} \sim U(-2.5, -0.5)$. But, for a set of random indices J with cardinality equal to s (i.e., $\#(J) = s$), we set $\beta_{k,1} = \beta_{k,2} = 0$ for $k \notin J$. In this way, β_1 and β_2 share the same nonzero coordinates. Finally,

$$y_t = 0.6y_{t-1} + 0.2y_{t-2} + \beta_1^\top \mathbf{x}_{t-1} + \beta_2^\top \mathbf{x}_{t-2} + \epsilon_t.$$

In this example, the relevant predictors have a direct impact on the response, instead of through any common factors. Additionally, when s is large, it is very difficult to recover the regression coefficients because of the lack of sparsity. Therefore, DGP 2 fits neither the factor model nor the sparse linear model frameworks. Nevertheless, the covariance matrix of \mathbf{x}_t has a special structure. Observe that

$$\mathbb{E}(\mathbf{x}_t \mathbf{x}_t^\top) = \mathbf{I}_p + 0.64\mathbf{B}\mathbf{B}^\top,$$

which shows that the covariance matrix of \mathbf{x}_t has the form of a spiked matrix.

Table 2 reports the RMSFEs of the competing methods considered. The sdPCA fares the best, as it yields the smallest RMSFE, especially when s is small. This reflects that the sdPCA can better utilize the spiked covariance structure by selecting contributions from relevant predictors. However, the performance of GsP* and sdPCA converge when $s = 0.75n$, and both of them compare favorably against the other alternatives. As expected, for non-sparse models, Lasso and random forest do not work well.

Table 2. Root mean squared forecast error of the competing methods, divided by 1000. Data are generated from DGP 2, and the results are averaged over 500 Monte Carlo simulations. n and p stand for the sample size and number of observed predictors. r_{DGP} is defined in DGP 2, r is the number of factors extracted using various methods, and s is a sparsity parameter.

(r_{DGP}, s)	GsP*	GsP	sdPCA	SW	LYB	Lasso	RF
$(n, p, r) = (200, 1000, 10)$							
(5, 50)	0.857	0.883	0.695	1.017	0.966	0.870	2.017
(10, 100)	2.833	3.137	2.834	3.436	3.436	3.802	6.971
(15, 150)	6.559	7.068	6.379	7.801	7.761	9.532	16.264
$(n, p, r) = (400, 2000, 30)$							
(10, 100)	4.091	4.547	2.259	5.224	4.917	5.118	8.977
(20, 200)	15.891	17.357	8.411	20.012	19.411	20.961	35.132
(30, 300)	34.557	37.956	34.543	42.904	42.905	50.996	70.180

DGP 3. In this example, the predictors follow a VAR model:

$$\mathbf{x}_t = \mathbf{B}\mathbf{x}_{t-1} + \delta_t,$$

where $\{\delta_t\}$ are independent p -dimensional standard Gaussian processes. Let $\tilde{\mathbf{B}}$ be a randomly generated $p \times p$ matrix of rank r_{DGP} . Then, the AR coefficient matrix is constructed as

$$\mathbf{B} = \frac{\tilde{\mathbf{B}}}{1.05\|\tilde{\mathbf{B}}\|},$$

where $\|\cdot\|$ denotes the operator norm. The target variable is generated via

$$y_t = 0.5y_{t-1} + \boldsymbol{\beta}^\top \mathbf{x}_{t-1} + \epsilon_t,$$

where $\{\epsilon_t\}$ are independent standard Gaussian variates and $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)^\top$ with

$$\beta_j = \begin{cases} (-1)^j u_j, & 1 \leq j \leq s \\ 0, & \text{otherwise} \end{cases},$$

in which $u_j \sim_{\text{i.i.d.}} \text{U}(0.1, 3.0)$.

In this DGP, the predictors, following a high-dimensional VAR(1) model, exhibit complex dynamics and correlations. Without simplifying structures such as latent factors, spiked covariance matrices, and sparsity, forecasting becomes difficult. Table 3 reports the RMSFEs of various competing methods. With small s , which corresponds to sparse models, Lasso can suitably choose the predictors and yield relatively accurate forecasts. However, as s increases, its RMSFE quickly increases and becomes similar to those of the factor-based approaches. The factor-based approaches, as well as the RF, performed similarly under this DGP. This example also demonstrates that the sdPCA method may encounter loss in prediction accuracy if the number of selected common factors is under-specified; see the case of $r_{\text{DGP}} = 15$ and $r = 10$.

Overall, our simulation studies show that no forecasting method always dominates the competing methods used in the study, but the proposed GsP* procedure can be effective in some cases.

Table 3. Root mean squared forecast error of various competing methods. Data are generated from DGP 3 and the results are averaged over 500 Monte Carlo simulations. n and p stand for the sample size and number of observed predictors. r_{DGP} is defined in DGP 3, and r is the number of factors extracted using various methods.

(r_{DGP}, s)	GsP*	GsP	sdPCA	SW	LYB	Lasso	RF
$(n, p, r) = (200, 1000, 10)$							
(5, 50)	12.255	12.166	12.347	12.565	12.469	7.537	12.813
(10, 100)	17.818	18.857	17.608	17.371	17.701	16.670	18.427
(15, 150)	23.142	24.417	23.041	21.777	22.027	21.125	22.660
$(n, p, r) = (400, 1000, 30)$							
(10, 100)	16.523	17.540	17.767	18.647	18.968	14.833	18.602
(20, 200)	27.266	29.110	26.063	25.291	25.717	25.953	26.503
(30, 300)	33.763	35.635	32.438	31.672	31.827	31.340	32.427

6. Empirical Examples

In this section, we apply the proposed method to two real data sets. The first data set is the U.S. macroeconomic data, and the other consists of Taiwan particulate matter (PM_{2.5}) measurements. Forecasting plays an essential role in the applications pertaining to these two data sets, despite both of them containing high-dimensional predictors. Moreover, these two datasets have different characteristics, which enable us to examine the forecasting performance of various forecasting methods available in the literature. In addition to the proposed GO-sdPCA approach, the competing methods used in the simulation studies, such as sdPCA, SW, LYB, Lasso, and RF, are also employed in this section as benchmarks.

For both datasets, we consider the rolling-window h -step-ahead forecasting. Let $\{y_t\}$ be the target variable of interest. For predicting y_{t+h} , the factor-based methods use the predictive equation

$$\hat{y}_{t+h} = \sum_{k=1}^q \alpha_k y_{t-q+1} + \beta^\top \hat{\mathbf{f}}_t + \epsilon_{t+h},$$

where $\hat{\mathbf{f}}_t$ is the vector of estimated common factors by different methods. Note that in constructing the factors, the methods are applied to the exogenous predictors $\mathbf{x}_t, \dots, \mathbf{x}_{t-q+1}$. The autoregressive variables $y_{t-1}, \dots, y_{t-q+1}$ only enter the predictive equation after factor extraction. For Lasso and RF, $y_t, y_{t-1}, \dots, y_{t-q+1}, \mathbf{x}_t, \dots, \mathbf{x}_{t-q+1}$ are used as potential predictors.

6.1. U.S. Macroeconomic Data

First, we consider the U.S. monthly macroeconomic data from January 1973 to June 2019. The data are from the FRED-MD database maintained by St. Louis Federal Reserve at <https://research.stlouisfed.org/econ/mccracken/fred-databases/> (accessed on 1 June 2024). We transform the time series to stationarity according to McCracken and Ng [9], and, after discarding some variables containing missing values, there are 125 macroeconomic time series. Among them, we focus on predicting (1) the industrial production index, (2) the unemployment rate, (3) CPI-All, and (4) real manufacturing and trade industries sales (M&T sales). Time plots of these four target series after transformations are depicted in Figure 1.

For these data sets, we consider $h = 1$ and different combinations of q and r (the number of factors extracted). Since q lags of the predictors are used, there are $125q$ total predictors, which exceeds the sample size in each window. The last twenty years of data (240 time periods) are reserved for testing. In addition to the root mean squared forecasting errors, we also report the p -values of the Diebold–Mariano test [26,27] against the alternative hypothesis that the proposed GsP* procedure is more accurate.

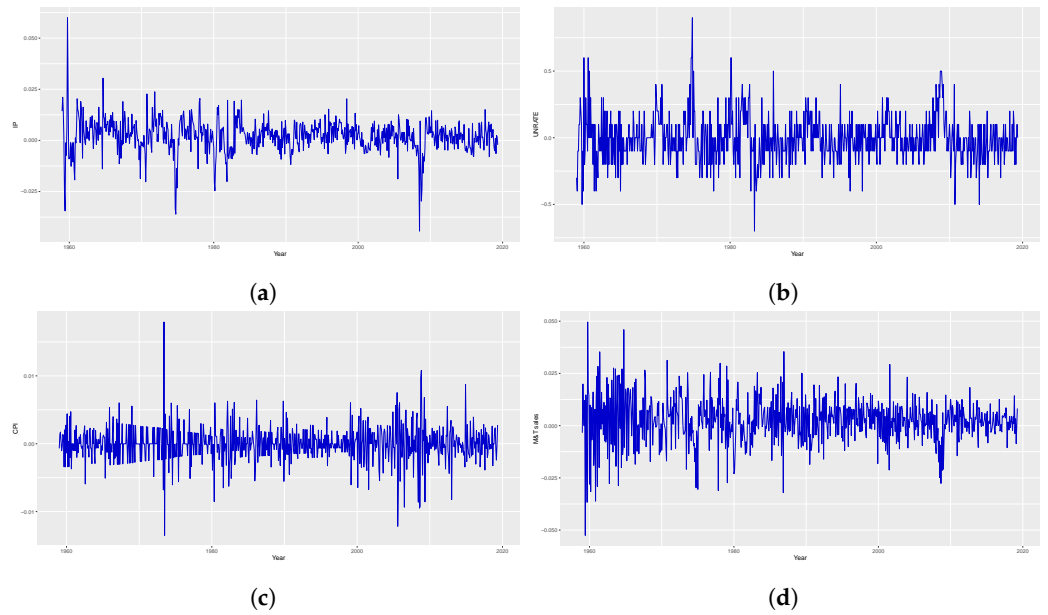


Figure 1. Time plots of the (transformed) macroeconomic time series of interest. (a) Industrial production; (b) unemployment rate; (c) CPI; (d) M&T sales.

Tables 4–7 present the results. The proposed GO-sdPCA achieved the lowest RMSFE for three of the targeted series: industrial production index, unemployment rate, and CPI. It notably outperformed the factor-based alternatives sdPCA, SW, and LYB with high confidence in forecasting these three series. In addition, no other methods demonstrated such consistent effectiveness across the targeted series.

Table 4. Root mean squared forecast errors $\times 100$ for predicting industrial production index. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

IP (RMSE $\times 100$); $h = 1$									
q	2			3			4		
Lasso	0.626			0.625 (0.053)			0.628		
RF	0.621			0.614 (0.046)			0.619		
r	2	4	6	2	4	6	2	4	6
GsP*	0.603	0.590	0.585	0.580	0.578	0.570	0.574	0.570	0.571
sdPCA	0.663	0.792	0.727	0.603	0.676	0.683	0.601 (0.024)	0.832	0.868
SW	0.631	0.636	0.640	0.623	0.629	0.634	0.617 (0.017)	0.624	0.629
LYB	0.631	0.634	0.636	0.622	0.621	0.623	0.615	0.609 (0.037)	0.612

Table 5. Root mean squared forecast errors for predicting unemployment rate. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

UNRATE (RMSE); $h = 1$									
q	2			3			4		
Lasso	0.138			0.138 (0.108)			0.139		
RF	0.135			0.134 (0.280)			0.135		
r	2	4	6	2	4	6	2	4	6
GsP*	0.134	0.135	0.133	0.133	0.134	0.132	0.133	0.134	0.132
sdPCA	0.182	0.156	0.136	0.183	0.135 (0.080)	0.144	0.229	0.181	0.146
SW	0.147	0.145	0.144	0.146	0.144 (0.002)	0.144	0.147	0.145	0.145
LYB	0.146	0.145 (0.008)	0.152	0.145	0.146	0.147	0.147	0.146	0.149

Table 6. Root mean squared forecast errors $\times 100$ for predicting CPI. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

CPI (RMSE $\times 100$); $h = 1$									
q	2			3			4		
Lasso	0.278 (0.162)			0.281			0.282		
RF	0.302 (0.014)			0.303			0.304		
r	2	4	6	2	4	6	2	4	6
GsP*	0.286	0.281	0.273	0.277	0.270	0.268	0.284	0.277	0.269
sdPCA	0.334	0.287 (0.039)	0.319	0.301	0.366	0.368	0.506	0.464	0.304
SW	0.300	0.301	0.294	0.291	0.292	0.282 (0.079)	0.294	0.293	0.283
LYB	0.300	0.300	0.304	0.293 (0.059)	0.294	0.296	0.293	0.294	0.299

Table 7. Root mean squared forecast errors $\times 100$ for predicting MT sales. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

MT Sales (RMSE $\times 100$); $h = 1$									
q	2			3			4		
Lasso	0.783			0.773 (0.203)			0.778		
RF	0.746 (0.561)			0.748			0.758		
r	2	4	6	2	4	6	2	4	6
GsP*	0.777	0.806	0.815	0.750	0.775	0.775	0.759	0.784	0.801
sdPCA	2.104	2.092	2.100	1.893 (0.113)	1.952	1.962	2.166	2.270	2.156
SW	0.761	0.745	0.774	0.750	0.737 (0.736)	0.764	0.758	0.745	0.772
LYB	0.796	0.792	0.799	0.768 (0.215)	0.784	0.786	0.772	0.773	0.797

6.2. Particulate Matters in Taiwan

We next consider the data of hourly $PM_{2.5}$ measurements in Taiwan during March of 2017. The data are sourced from the SLBDD package [28] in R. Each series in the data set represents measurements (in micrograms per cubic meters, $\mu\text{g}/\text{m}^3$) taken by a novel device known as the AirBox. After an initial examination of these series, we remove series 29 and 70, because these series are near-identically zero, except at a few time points, offering no useful variations. Among the 516 series in the data set, we choose series 101, 201, 301, 401, as target series. Figure 2 depicts their time plots.

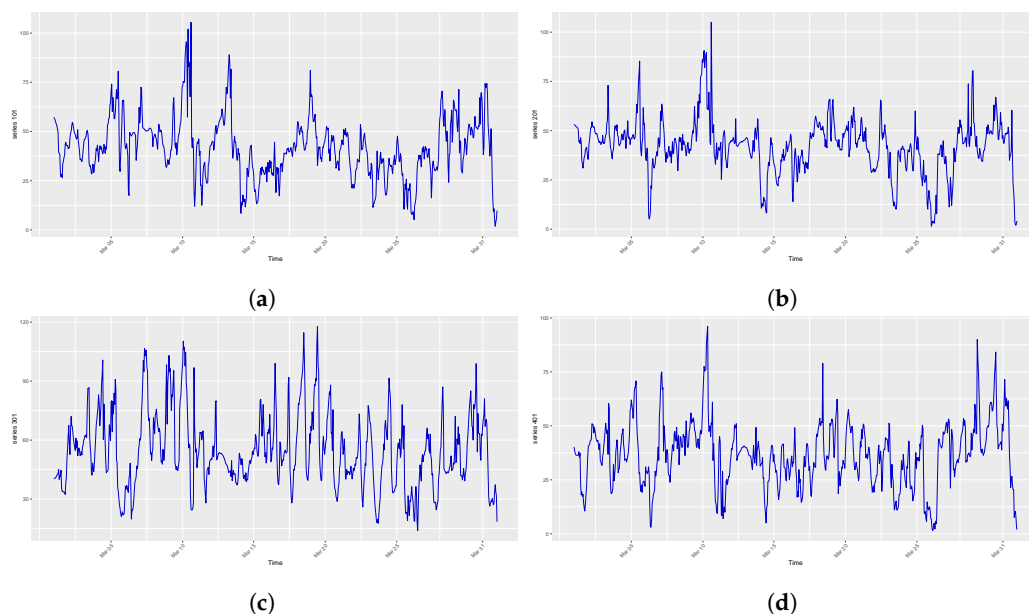


Figure 2. Time plots of the four targeted hourly $PM_{2.5}$ measurements on Taiwan in March 2017. (a) Series 101; (b) series 201; (c) series 301; (d) series 401.

We consider both one-step-ahead ($h = 1$) and two-step-ahead ($h = 2$) forecasts, and employ $q \in \{2, 3\}$ lags and $r \in \{2, 4, 6\}$ factors in forecasting. The last ten days of data (240 time periods) are reserved for out-of-sample testing. Tables 8–11 present the results for $h = 1$. The proposed GO-sdPCA ranks as the most predictive method in terms of RMSFE for two of the four targeted series: series 101 and 201. It outperforms the Lasso in forecasting all four targeted series. The performance of LYB is also noteworthy. It ranks among the best two methods for all four targeted series. The DM test, on the other hand, indicates the difference in forecasting accuracy is not statistically significant. Contrary to the case in the macroeconomic data, the proposed GsP* only outperformed the factor-based methods, including the sdPCA, with some weak confidence. For the two-step-ahead forecasts, for which the results are reported in Tables 12–15, the performance of various methods is more entangled, with the RF consistently ranked among the top two methods for three of the four targeted series. With some weaker confidence, GsP* is the best performing factor-based method for two of the four targeted series. Again, the DM test fails to separate significantly various forecasting methods. We believe this result might be caused by the substantial uncertainty in the $PM_{2.5}$ measurements.

In sum, for $h = 1$, GsP* is effective in forecasting $PM_{2.5}$ data, as well as the U.S. macroeconomic data. This implies the proposed procedure is able to capture highly predictive factors across various applications. On the other hand, for $h = 2$, the dynamic nature of the data may be much more involved. The performance of the proposed method becomes similar to those of the other forecasting methods.

Table 8. Root mean squared forecast errors for predicting series 101 in Taiwan PM_{2.5} data. The forecast horizon h is 1. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 101 (RMSE); $h = 1$						
q	2			3		
Lasso	6.363			6.327 (0.015)		
RF	6.871 (0.004)			6.910		
r	2	4	6	2	4	6
GsP*	6.047	6.193	6.173	6.021	6.162	6.091
sdPCA	6.079 (0.298)	6.151	6.221	6.079	6.154	6.167
SW	6.098 (0.269)	6.160	6.104	6.103	6.170	6.110
LYB	6.055 (0.392)	6.125	6.117	6.060	6.133	6.146

Table 9. Root mean squared forecast errors for predicting series 201 in Taiwan PM_{2.5} data. The forecast horizon h is 1. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 201 (RMSE); $h = 1$						
q	2			3		
Lasso	5.990			5.973 (0.346)		
RF	5.938 (0.399)			6.109		
r	2	4	6	2	4	6
GsP*	5.897	6.079	6.128	5.931	6.197	6.146
sdPCA	6.012	5.982 (0.201)	6.065	6.057	6.019	6.093
SW	6.113	5.956	5.952 (0.298)	6.135	5.989	5.984
LYB	6.129	5.967	5.934 (0.358)	6.154	5.994	5.972

Table 10. Root mean squared forecast errors for predicting series 301 in Taiwan PM_{2.5} data. The forecast horizon h is 1. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 301 (RMSE); $h = 1$						
q	2			3		
Lasso	6.824 (0.367)			6.854		
RF	6.653 (0.601)			6.698		
r	2	4	6	2	4	6
GsP*	6.756	6.765	6.724	6.800	6.813	6.781
sdPCA	6.733 (0.483)	6.977	6.865	6.757	7.021	6.879
SW	6.794	7.029	7.040	6.743 (0.476)	7.013	7.009
LYB	6.749	6.966	7.018	6.710 (0.516)	6.958	7.047

Table 11. Root mean squared forecast errors for predicting series 401 in Taiwan PM_{2.5} data. The forecast horizon h is 1. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 401 (RMSE); $h = 1$						
q	2			3		
Lasso	6.392			6.349 (0.125)		
RF	7.122 (0.008)			7.196		
r	2	4	6	2	4	6
GsP*	6.170	6.235	6.251	6.238	6.236	6.243
sdPCA	6.200	6.026	6.111	6.272	6.016 (0.870)	6.112
SW	6.171	5.908 (0.996)	5.964	6.227	5.953	5.999
LYB	6.139	5.928 (0.992)	5.994	6.193	5.967	6.023

Table 12. Root mean squared forecast errors for predicting series 101 in Taiwan PM_{2.5} data. The forecast horizon h is 2. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 101 (RMSE); $h = 2$						
q	2			3		
Lasso	9.584 (0.818)			9.618		
RF	9.544			9.415 (0.803)		
r	2	4	6	2	4	6
GsP*	9.788	9.969	9.915	9.758	9.941	9.893
sdPCA	9.982	9.970	9.804	9.980	9.930	9.715 (0.597)
SW	10.073	9.946	9.868 (0.326)	10.069	9.951	9.875
LYB	10.000	9.888	9.883 (0.282)	9.994	9.888	9.907

Table 13. Root mean squared forecast errors for predicting series 201 in Taiwan PM_{2.5} data. The forecast horizon h is 2. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 201 (RMSE); $h = 2$						
q	2			3		
Lasso	9.868 (0.731)			10.005		
RF	9.594 (0.987)			9.614		
r	2	4	6	2	4	6
GsP*	9.998	10.276	10.209	10.052	10.292	10.219
sdPCA	10.164	10.070 (0.339)	10.112	10.228	10.154	10.150
SW	10.233	10.151 (0.244)	10.151	10.270	10.196	10.207
LYB	10.230	10.186	10.176 (0.212)	10.267	10.222	10.269

Table 14. Root mean squared forecast errors for predicting series 301 in Taiwan PM_{2.5} data. The forecast horizon h is 2. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 301 (RMSE); $h = 2$						
q	2			3		
Lasso	11.189			10.968 (0.314)		
RF	10.727 (0.542)			10.734		
r	2	4	6	2	4	6
GsP*	10.895	10.899	10.890	10.828	10.838	10.767
sdPCA	10.930	11.308	11.220	10.794 (0.460)	11.164	11.140
SW	11.265	11.675	11.598	11.038 (0.280)	11.528	11.456
LYB	11.130	11.448	11.442	10.930 (0.362)	11.325	11.325

Table 15. Root mean squared forecast errors for predicting series 401 in Taiwan PM_{2.5} data. The forecast horizon h is 2. The lowest RMSFE achieved by each method is in boldface. Among these boldfaced values, the lowest two values are marked in red. q denotes the number of lags used in estimation, and r is the number of factors extracted. GsP* denotes the proposed GO-sdPCA. The figures in the parentheses are the p -values of the Diebold–Mariano test of whether GsP* is more accurate.

Series 401 (RMSE); $h = 2$						
q	2			3		
Lasso	9.862 (0.100)			9.965		
RF	9.694			9.676 (0.323)		
r	2	4	6	2	4	6
GsP*	9.768	9.572	9.668	9.745	9.491	9.626
sdPCA	10.391	9.317	9.427	10.388	9.213 (0.891)	9.371
SW	10.405	9.570	9.200	10.401	9.508	9.172 (0.798)
LYB	10.293	9.600	9.285	10.278	9.534	9.279 (0.747)

7. Discussion and Concluding Remarks

In this paper, we proposed a novel method for time series forecasting when many predictors are available. The rationale behind our method is to mine the possibly many (compared to the sample size) factor-relevant predictors while reducing the effect of the irrelevant variables in the high-dimensional data. The results in the simulation studies and the empirical applications suggest that the proposed method is useful for improving upon both the factor-based methods and methods for sparse linear models, such as the Lasso. Finally, we remark that the theoretical investigation of the peeling technique, a key ingredient in our method, would be an interesting topic for further research.

Author Contributions: Conceptualization, R.S.T.; Methodology, S.-C.H. and R.S.T.; Validation, R.S.T.; Formal analysis, S.-C.H.; Writing—original draft, S.-C.H.; Writing—review & editing, S.-C.H. and R.S.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Sensitivity to Tuning Parameters

In this section, we conduct additional simulations to examine the sensitivity of GO-sdPCA to the tuning parameters q_1, q_2 , and q_3 . First, we examine how the choice of q_1 affects the variable selection results of GOGA combined with the peeling technique. Then, various combinations of q_1, q_2 , and q_3 are employed to study how they impact the forecasting performance of GO-sdPCA. Throughout this section, we focus on DGP 1 introduced in Section 5 with $n = 200, p = 500, r_{DGP} = 10$, and $s = 100$.

To understand the effect of q_1 on the variable selection results, we apply GOGA with $q_1 \in \{2, 5, 10, 20\}$ and study its variable selection quality along the peeling iterations. In the i -th peeling iteration, we denote the number of true positives by TP_i . That is, TP_i is the number of variables selected by GOGA that are in the relevant set J in the i -th peeling iteration. Analogously, the false positives FP_i are the number of variables selected by GOGA which are not in J . The following two metrics are of interest.

$$R_m = \frac{\sum_{i=1}^m TP_i}{\sum_{i=1}^m (TP_i + FP_i)}, \quad \text{cumTP}_m = \sum_{i=1}^m TP_i. \tag{A1}$$

Clearly, R_m is the ratio of the cumulative true positives (TP) to the number of selected variables, which reflects the percentage of “good” predictors in the selected variables, and cumTP_m is the number of true positives selected up until the m -th peeling iteration.

Figure A1 plots R_m and cumTP_m with various q_1 . Observe first that, when $q_1 = 2, R_m$ remains above 0.5 for most m , implying that few irrelevant variables are allowed to enter the model. On the other hand, for $q_1 \in \{5, 10\}$, R_m values vary over the range $[0.2, 0.27]$, which is the ratio of the number of relevant variables to the number of candidate variables (s/p). This means the variable selection quality is only slightly better than random selection. (For $q_1 = 20$, the GOGA iteration K_n is set to 7 to avoid collinearity, and this seems to help suppress the effect of irrelevant predictors and R_m is higher than 0.3. However, R_m is also decreasing in this case.) In addition, for $q_1 = 2$, cumulative true positives plateau near 80 quickly after 10 peeling iterations, which implies about 80% of relevant predictors are captured by GOGA. Therefore, tuning a smaller q_1 appears preferable for variable selection performance.

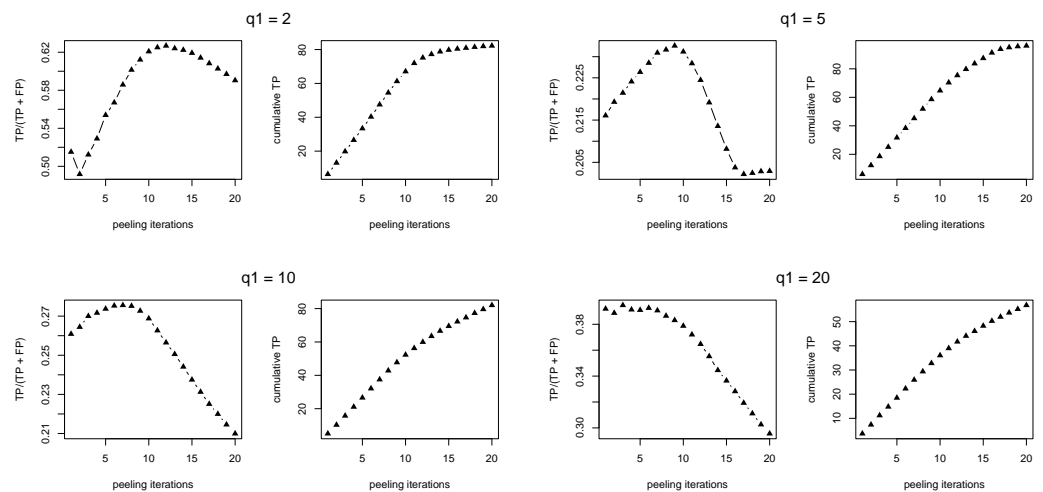


Figure A1. Variable selection performance of GOGA along peeling iterations, with $q_1 \in \{2, 5, 10, 20\}$. Figures on the left plot the ratio of the cumulative true positives (TPs) to the number of selected variables ($TP + FP$, where FP stands for false positive), which is exactly R_m defined in (A1). Figures on the right shows the cumulative true positives. The values are average across 100 Monte Carlo simulations.

Next, we examine the forecasting performance with several combinations of (q_1, q_2, q_3) . In particular, we run GO-sdPCA with q_1, q_2 , and q_3 varying across $\{2, 3, 5, 10\}$. The number

of peeling iterations is set to 10, as in Section 5. The RMSFEs are reported in Table A1. The results show that the forecasting capability of GO-sdPCA is quite stable over most combinations of (q_1, q_2, q_3) . In particular, the RMSFEs are not very sensitive to q_1 and q_3 . This highlights that the sdPCA step is very effective for mitigating the effect of irrelevant variables in the selected predictors. However, a very large q_2 may hurt the forecasting performance of GO-sdPCA. In this case, the RMSFE becomes significantly larger for $q_2 = 10$, regardless of q_1 and q_3 . We conclude that GO-sdPCA is, in general, not sensitive to reasonable choices of the tuning parameters, but q_2 should be tuned with extra care.

Table A1. Root mean square forecasting errors (RMSFE) of GO-sdPCA applied to DGP 1 with $n = 200$, $p = 500$, $r_{DGP} = 10$, and $s = 100$, with q_1, q_2 , and q_3 varying across $\{2, 3, 5, 10\}$ used in the implementation. The results are averaged over 300 Monte Carlo simulations.

q_1	$q_3 = 2$				$q_3 = 3$			
	q_2				q_2			
	2	3	5	10	2	3	5	10
2	2.30	2.33	2.76	3.11	2.43	2.43	2.63	2.90
3	2.51	2.43	2.77	3.61	2.24	2.72	2.56	3.65
5	2.39	2.77	2.65	3.70	2.41	2.61	2.59	3.66
10	2.57	2.60	2.64	3.54	2.39	2.32	2.70	3.27

q_1	$q_3 = 5$				$q_3 = 10$			
	q_2				q_2			
	2	3	5	10	2	3	5	10
2	2.31	2.44	2.53	2.88	2.29	2.40	2.58	3.11
3	2.18	2.57	2.58	3.49	2.33	2.50	2.53	3.26
5	2.29	2.28	2.65	3.49	2.39	2.48	2.60	3.32
10	2.15	2.21	2.52	3.39	2.39	2.53	2.55	3.11

References

- Peña, D.; Box, G.E. Identifying a simplifying structure in time series. *J. Am. Stat. Assoc.* **1987**, *82*, 836–843. [\[CrossRef\]](#)
- Stock, J.H.; Watson, M.W. Forecasting using principal components from a large number of predictors. *J. Am. Stat. Assoc.* **2002**, *97*, 1167–1179. [\[CrossRef\]](#)
- Stock, J.H.; Watson, M.W. Macroeconomic forecasting using diffusion indexes. *J. Bus. Econ. Stat.* **2002**, *20*, 147–162. [\[CrossRef\]](#)
- Lam, C.; Yao, Q.; Bathia, N. Estimation of latent factors for high-dimensional time series. *Biometrika* **2011**, *98*, 901–918. [\[CrossRef\]](#)
- Tsay, R.S. *Multivariate Time Series Analysis: With R and Financial Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
- Bernanke, B.S.; Boivin, J. Monetary policy in a data-rich environment. *J. Monet. Econ.* **2003**, *50*, 525–546. [\[CrossRef\]](#)
- Bernanke, B.S.; Boivin, J.; Eliasziw, P. Measuring the Effects of Monetary Policy: A Factor-Augmented Vector Autoregressive (FAVAR) Approach. *Q. J. Econ.* **2005**, *120*, 387–422.
- Jurado, K.; Ludvigson, S.C.; Ng, S. Measuring uncertainty. *Am. Econ. Rev.* **2015**, *105*, 1177–1216. [\[CrossRef\]](#)
- McCracken, M.W.; Ng, S. Fred-md: A monthly database for macroeconomic research. *J. Bus. Econ. Stat.* **2016**, *34*, 574–589. [\[CrossRef\]](#)
- Boivin, J.; Ng, S. Are more data always better for factor analysis? *J. Econom.* **2006**, *132*, 169–194. [\[CrossRef\]](#)
- Kim, H.H.; Swanson, N.R. Forecasting financial and macroeconomic variables using data reduction methods: New empirical evidence. *J. Econom.* **2014**, *178*, 352–367. [\[CrossRef\]](#)
- Huang, D.; Jiang, F.; Li, K.; Tong, G.; Zhou, G. Scaled pca: A new approach to dimension reduction. *Manag. Sci.* **2022**, *68*, 1678–1695. [\[CrossRef\]](#)
- Gao, Z.; Tsay, R.S. Supervised dynamic pca: Linear dynamic forecasting with many predictors. *J. Am. Stat. Assoc.* **2024**, *accepted*. [\[CrossRef\]](#)
- Chan, N.H.; Ing, C.-K.; Li, Y.; Yau, C.Y. Threshold estimation via group orthogonal greedy algorithm. *J. Bus. Econ. Stat.* **2017**, *35*, 334–345. [\[CrossRef\]](#)
- Ing, C.-K.; Lai, T.L. A stepwise regression method and consistent model selection for high-dimensional sparse linear models. *Stat. Sin.* **2011**, *21*, 1473–1513. [\[CrossRef\]](#)
- Ing, C.-K. Model selection for high-dimensional linear regression with dependent observations. *Ann. Stat.* **2020**, *48*, 1959–1980. [\[CrossRef\]](#)
- Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Society. Ser. B (Methodol.)* **1996**, *58*, 267–288. [\[CrossRef\]](#)
- Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [\[CrossRef\]](#)
- Temlyakov, V. *Greedy Approximation*; Cambridge Monographs on Applied and Computational Mathematics; Cambridge University Press: Cambridge, UK, 2011.

20. Barron, A.R.; Cohen, A.; Dahmen, W.; DeVore, R.A. Approximation and learning by greedy algorithms. *Ann. Stat.* **2008**, *36*, 64–94. [[CrossRef](#)]
21. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning with Applications in R*, 2nd ed.; Springer: New York, NY, USA, 2021.
22. Chi, C.-M.; Vossler, P.; Fan, Y.; Lv, J. Asymptotic properties of high-dimensional random forests. *Ann. Stat.* **2022**, *50*, 3415–3438. [[CrossRef](#)]
23. Saha, A.; Basu, S.; Datta, A. Random forests for spatially dependent data. *J. Am. Stat. Assoc.* **2023**, *118*, 665–683. [[CrossRef](#)]
24. Medeiros, M.C.; Mendes, E.F. ℓ_1 -regularization of high-dimensional time-series models with non-gaussian and heteroskedastic errors. *J. Econom.* **2016**, *191*, 255–271. [[CrossRef](#)]
25. Liaw, A.; Wiener, M. Classification and regression by randomforest. *R News* **2002**, *2*, 18–22.
26. Diebold, F.X.; Mariano, R.S. Comparing predictive accuracy. *J. Bus. Econ. Stat.* **1995**, *13*, 253–263. [[CrossRef](#)]
27. Harvey, D.; Leybourne, S.; Newbold, P. Testing the equality of prediction mean squared errors. *Int. J. Forecast.* **1997**, *13*, 281–291. [[CrossRef](#)]
28. Caro, A.; Elias, A.; Peña, D.; Tsay, R.S. *SLBDD: Statistical Learning for Big Dependent Data*. R Package Version 0.0.4. 2022. Available online: <https://cran.r-project.org/web/packages/SLBDD/index.html> (accessed on 1 June 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.