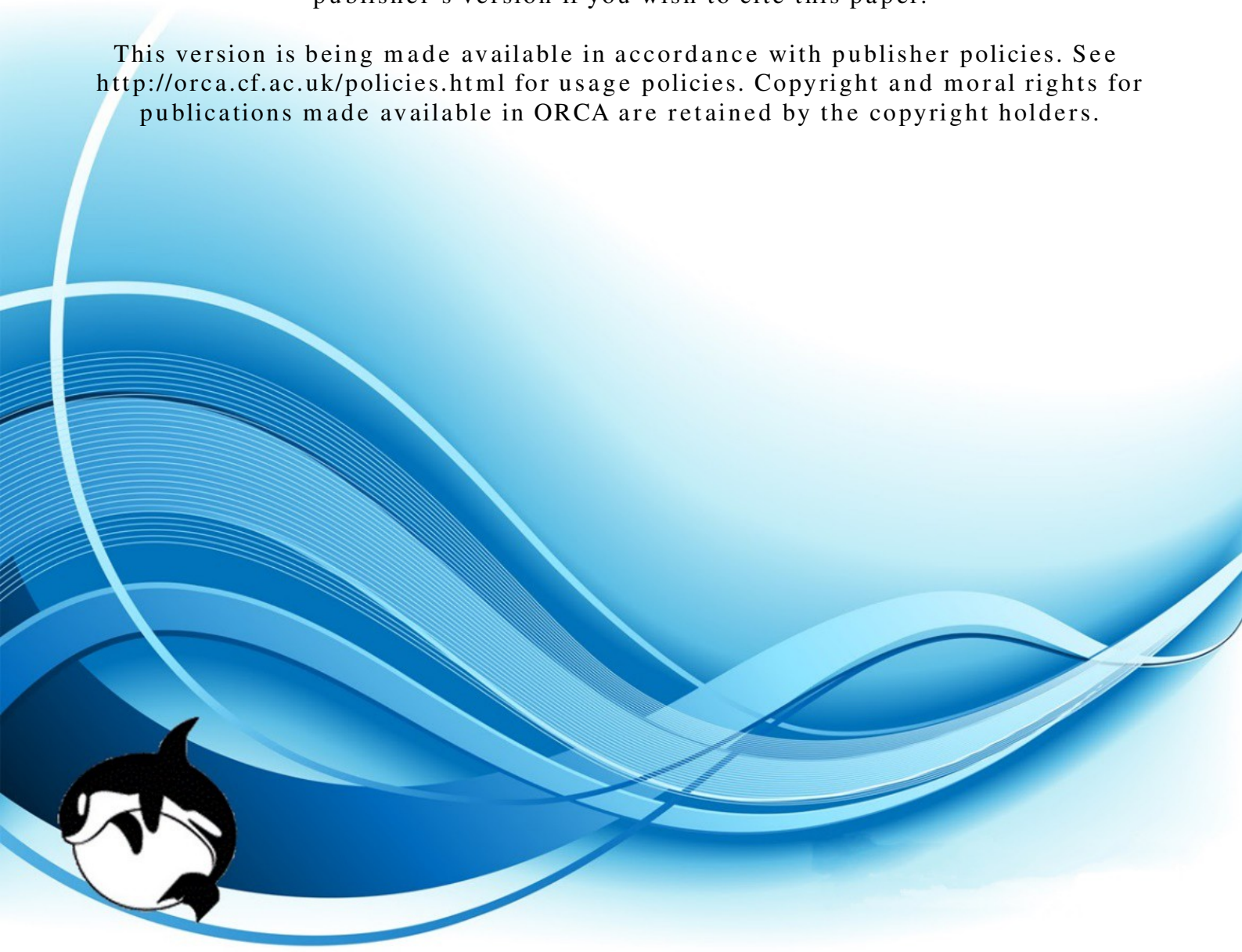


Highlights

Real-time sufficient dimension reduction through principal least squares support vector machines

Andreas Artemiou, Yuexiao Dong and Seung Jun Shin

- First approach to real time SVM-based sufficient dimension (SDR)
- Computationally very fast and more accurate SDR than other methods.

Real-time sufficient dimension reduction through principal least squares support vector machines

Andreas Artemiou, Yuexiao Dong and Seung Jun Shin
Cardiff University, Temple University and Korea University

Abstract

We propose a real-time approach for sufficient dimension reduction. Compared with popular sufficient dimension reduction methods including sliced inverse regression and principal support vector machines, the proposed principal least squares support vector machines approach enjoys better estimation of the central subspace. Furthermore, this new proposal can be used in the presence of streamed data for quick real-time updates. It is demonstrated through simulations and real data applications that our proposal performs better and faster than existing algorithms in the literature.

Keywords: central subspace, ladle estimator, online sliced inverse regression, principal support vector machines, streamed data

1. Introduction

With the increase in computing power, the decrease in cost of computing storage, and the improvement of measuring tools across different scientific fields, data nowadays is collected in a continuous fashion. For example, patients are monitored without interruption in medical studies, and meteorological data is routinely collected with very small time interval between observations. Due to the sheer volume of these massive datasets, it becomes

challenging to apply many classical statistical methods as their computation time becomes prohibitive. Real-time algorithms are thus developed as computationally efficient alternatives. After getting an initial estimator with the currently available data, the basic idea of real-time method is to update the estimator efficiently as new data are collected. In the early 1970s, real-time singular value decomposition was discussed in [1], and quick updates of linear regression was introduced in [2]. Recent developments of real-time regression include fast robust linear regression [3], real-time semiparametric regression [4], online orthogonal regression [5], and fast regression for large data [6]. In terms of applications, real-time algorithms have been applied to head pose estimation [7], stock market analysis [8], and human action recognition [9].

Aside from the volume of the data, the dimensionality of the data poses another challenge for modern data analysis. A popular tool in pattern recognition is dimension reduction. For example, [10] used dimension reduction for face recognition, [11] applied dimension reduction for text classification, and [12] proposed dimension reduction for feature selection. Sufficient dimension reduction [13] is a class of supervised dimension reduction techniques with the objective of performing feature extraction in regression without losing any regression information between the response and the predictor. Existing sufficient dimension reduction methods in the literature include, but are not limited to, sliced inverse regression (SIR) [14], sliced average variance estimation [15], principal Hessian directions [16], kernel dimension reduction [17], cumulative mean estimation [18], and principal support vector machines (PSVM) [19]. For a recent review, please see [20].

Existing SDR algorithms use all available data to perform feature extrac-

tion. In stream data, where we need to constantly update the estimation as new data are collected, the use of all available data can create computational challenges even for computationally efficient algorithms. Therefore it is important to develop real time SDR algorithms that work efficiently in the case that there are data streams. The idea is that, if we perform dimension reduction at a time point t and we have an estimate initial estimate, then at time point $t+1$ we can perform dimension reduction using the data collected between t and $t+1$ to update the estimate we had at time point t. Then at time point $t+2$ we use the data collected between $t+1$ and $t+2$ to update the estimate at time point $t+1$ and so on. This leads to a computationally efficient way of dealing with massive datasets as we work only with the data collected since the last update.

A novel real-time sufficient dimension reduction approach is introduced in this paper. Our contribution is two-fold: first we propose principal least squares support vector machines (PLSSVM) as a classical sufficient dimension reduction method; then real-time algorithms based on PLSSVM are developed. As a classical sufficient dimension reduction estimator, PLSSVM significantly improves the estimation accuracy of the original SVM-based sufficient dimension reduction method PSVM. Compared with an existing real-time sufficient dimension reduction approach [21] based on SIR, our proposal is much more efficient when the data is large. The rest of the paper is structured as follows. PLSSVM for classical sufficient dimension reduction is discussed in section 2 and we present the real-time PLSSVM algorithms in section 3. Numerical studies are provided in section 4 and we conclude the paper with some discussions in section 5. Without interrupting the main

text, the asymptotic analysis is provided in the Appendix.

2. Principal least squares SVM for sufficient dimension reduction

We introduce PLSSVM for sufficient dimension reduction in this section. After reviewing the concept of the central subspace, the population level development and the sample level algorithm of PLSSVM are provided. Estimating the dimensionality of the central subspace is also discussed.

2.1. Central subspace

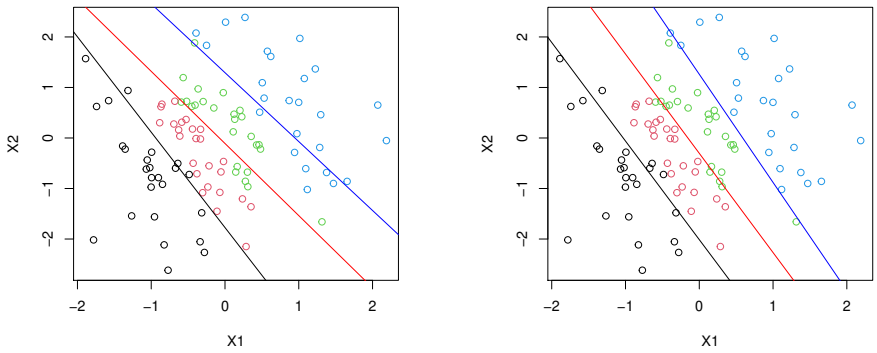
For univariate response Y and predictor vector $\boldsymbol{X} \in \mathbb{R}^p$. Sufficient dimension reduction [13] aims to find a $p \times d$ ($d \leq p$) matrix $\boldsymbol{\eta}$ such that

$$Y \perp\!\!\!\perp \boldsymbol{X} | \boldsymbol{\eta}^\top \boldsymbol{X}, \tag{1}$$

where $\perp\!\!\!\perp$ denotes statistical independence. Dimension reduction is achieved if $d < p$. The space spanned by the column vectors of $\boldsymbol{\eta}$ is called a dimension reduction subspace. Central subspace is the minimum dimension reduction subspace in the sense that it has the smallest dimension among all possible dimension reduction subspaces. We denote the central subspace between Y and $\boldsymbol{X}$ as $\mathcal{S}_{Y|\boldsymbol{X}}$. Conditions for existence of the central subspace were discussed in [22]. Throughout the paper we assume that the central subspace exists and is unique. The dimension of $\mathcal{S}_{Y|\boldsymbol{X}}$ is referred to as the structural dimension, and we denote it by d.

2.2. Population level development of principal least squares SVM

Least squares SVM (LSSVM) for classification was proposed in [23]. To motivate the use of LSSVM in the dimension reduction framework, in Figure



(a) SVM (b) LSSVM

Figure 1: Motivating example for using least squares SVM. We use 100 data points from the model $Y = 2X_1 + X_2 + 0.2\epsilon$ where $\epsilon \sim N(0, 1)$ with 4 slices (colored in black, red, green and blue). Black slices are the points that have the lowest value of Y and green the ones with the highest value of Y. The left figure shows the hyperplanes produced from SVM and the right the hyperplanes produced by LSSVM.

1 we demonstrate the different hyperplanes produced by SVM and LSSVM in a simple example where we have $Y = 2X_1 + X_2 + 0.2\epsilon$ where $\epsilon \sim N(0, 1)$, $\boldsymbol{X}$ is simulated from bivariate standard normal and we split the range of Y in 4 slices. As one can see there is better alignment of the hyperplanes in LSSVM for the different of slices. Also, since it has a closed-form solution, LSSVM achieves faster computation than the classical soft-margin SVM [24]. Given an i.i.d. sample $\{(\boldsymbol{X}_i, Y_i) : i = 1, \ldots, n\}$, LSSVM finds the optimal

separating hyperplane by solving the following minimization problem

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{\psi}^{\mathsf{T}}\boldsymbol{\psi} + \frac{\lambda}{n}\sum_{i=1}^{n}\xi_i^2 \ \text{ among } (\boldsymbol{\psi}, t, \boldsymbol{\xi}) \in \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}^n \\ &\text{subject to} \quad \xi_i = 1 - \ Y_i[\boldsymbol{\psi}^{\mathsf{T}}(\boldsymbol{X}_i - \bar{\boldsymbol{X}}) - t], \ \ i = 1, \ldots, n. \end{aligned} \tag{2}$$

Here $Y_i \in \{-1, 1\}$ for $i = 1, \ldots, n$, $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^{\mathsf{T}}$, $\bar{\boldsymbol{X}} = n^{-1}\sum_{i=1}^{n}\boldsymbol{X}_i$, and λ is a positive constant known as the misclassification penalty. A hyperplane can be defined through equation $\boldsymbol{\psi}^{\mathsf{T}}\boldsymbol{X} - t = 0$, and the solution to this minimization problem provides the optimal hyperplane.

It is easy to see that the population version of (2) is

$$\boldsymbol{\psi}^{\mathsf{T}}\boldsymbol{\psi} + \lambda E\left\{(1 - Y[\boldsymbol{\psi}^{\mathsf{T}}(\boldsymbol{X} - E(\boldsymbol{X})) - t])^2\right\}. \tag{3}$$

Here $E(\boldsymbol{X})$ denotes the expectation of $\boldsymbol{X}$, and $E(\cdot)$ is the expectation with respect to the joint distribution of $(\boldsymbol{X}, Y)$. To adapt LSSVM for sufficient dimension reduction, we follow [19] and make two modifications to (3). First, $\boldsymbol{\Sigma} = \text{var}(\boldsymbol{X})$ is added in the first term. Also, the binary response Y is replaced with $\tilde{Y} = I(Y \in A_1) - I(Y \in A_2)$. Here $I(\cdot)$ is the indicator function. A_1 and A_2 are subsets of Ω_Y, the domain of a continuous response Y, and they satisfy $A_1 \cap A_2 = \emptyset$. These modifications lead to the population version of the principal least squares SVM objective function

$$L(\boldsymbol{\psi}, t) = \boldsymbol{\psi}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{\psi} + \lambda E\left\{(1 - \tilde{Y}[\boldsymbol{\psi}^{\mathsf{T}}(\boldsymbol{X} - E(\boldsymbol{X})) - t])^2\right\}. \tag{4}$$

The following result provides the essential link between PLSSVM and sufficient dimension reduction. Its proof is similar to the proof of Theorem 1 in [19], and is thus omitted.

Theorem 1. *Suppose* $E(\boldsymbol{X}|\boldsymbol{\eta}^\tau\boldsymbol{X})$ *is a linear function of* $\boldsymbol{\eta}^\tau\boldsymbol{X}$*, where* $\boldsymbol{\eta}$ *is defined in (1). If* $(\boldsymbol{\psi}^*, t^*)$ *minimizes the objective function* $L(\boldsymbol{\psi}, t)$ *in (4) among all* $(\boldsymbol{\psi}, t) \in \mathbb{R}^p \times \mathbb{R}$*, then* $\boldsymbol{\psi}^* \in \mathcal{S}_{Y|\boldsymbol{X}}$.

We remark that the linearity assumption about the conditional expectation $E(\boldsymbol{X}|\boldsymbol{\eta}^\tau\boldsymbol{X})$ is common in the sufficient dimension reduction literature, and it is satisfied if $\boldsymbol{X}$ has an elliptically-contoured distribution. The idea of combining SVM with sufficient dimension reduction is not new. Since the first attempt by PSVM to adapt the soft-margin SVM [19], principal weighted SVM was considered in [25], principal minimax SVM was introduced in [26], and principal Lq SVM was proposed in [27]. Furthermore, the hinge loss in [19] was replaced with the logistic loss in [28], and different reweighting schemes to reduce slice imbalance bias were studied in [29] and [30]. Recently [31] proposed the use of Distance-Weighted Discrimination, another large margin classifier, for accurate and computationally faster dimension reduction especially when p is close to n.

2.3. Sample level algorithm of principal least squares SVM

Given an i.i.d. sample $\{(\boldsymbol{X}_i, Y_i) : i = 1, \ldots, n\}$, we discuss the PLSSVM algorithm in this section. Without loss of generality, assume $\sum_{i=1}^n \boldsymbol{X}_i = \boldsymbol{0}$. For fixed sets A_1 and A_2, we have $\tilde{Y}_i = I(Y_i \in A_1) - I(Y_i \in A_2)$, $i = 1, \ldots, n$. Let $\boldsymbol{1}_n$ be the n-dimensional vector with all entries equal to 1, $\boldsymbol{r} = (\boldsymbol{\psi}^\tau, t)^\tau$, $\tilde{\boldsymbol{Y}} = (\tilde{Y}_1, \ldots, \tilde{Y}_n)^\tau$, and $\boldsymbol{D}_{\tilde{Y}}$ denotes the diagonal matrix with the diagonal being entries of $\tilde{\boldsymbol{Y}}$. Let $\mathbb{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)^\tau$ and $\mathbb{X}^* = (\mathbb{X}, -\boldsymbol{1}_n)$. Moreover, let $\hat{\boldsymbol{\Sigma}}^*$ be a block diagonal matrix with the diagonal elements being $\hat{\boldsymbol{\Sigma}} =$

$n^{-1}\sum_{i=1}^{n}\boldsymbol{X}_i\boldsymbol{X}_i^{\mathsf{T}}$ and 0. The sample version of (4) then becomes

$$\boldsymbol{r}^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}^{*}\boldsymbol{r}+\frac{\lambda}{n}(\mathbf{1}_n-\boldsymbol{D}_{\tilde{Y}}\mathbb{X}^{*}\boldsymbol{r})^{\mathsf{T}}(\mathbf{1}_n-\boldsymbol{D}_{\tilde{Y}}\mathbb{X}^{*}\boldsymbol{r}). \tag{5}$$

After taking the derivative of (5) with respect to $\boldsymbol{r}$ and setting it equal to 0, the solution of $\boldsymbol{r}$ leads to

$$\hat{\boldsymbol{r}}=\hat{\boldsymbol{M}}\tilde{\boldsymbol{Y}},\ \text{where}\ \hat{\boldsymbol{M}}=\left(\frac{n\hat{\boldsymbol{\Sigma}}^{*}}{\lambda}+(\mathbb{X}^{*})^{\mathsf{T}}\mathbb{X}^{*}\right)^{-1}(\mathbb{X}^{*})^{\mathsf{T}}. \tag{6}$$

Denote the submatrix that consists of the first p rows of $\hat{\boldsymbol{M}}$ as $[\hat{\boldsymbol{M}}]_p$, and the first p elements of $\hat{\boldsymbol{r}}=(\hat{\boldsymbol{\psi}}^{\mathsf{T}},\hat{t})^{\mathsf{T}}$ becomes $\hat{\boldsymbol{\psi}}=[\hat{\boldsymbol{M}}]_p\tilde{\boldsymbol{Y}}$.

Now we describe the PLSSVM algorithm to estimate the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$. The misclassification penalty λ, the number of slices H, and the structural dimension d for $\mathcal{S}_{Y|\boldsymbol{X}}$ are assumed to be known.

1. Center the predictor and calculate $\hat{\boldsymbol{M}}$ in (6).
2. Let $\{q_1,\ldots,q_{H-1}\}$ be equally spaced sample percentiles of $\{Y_1,\ldots,Y_n\}$. Calculate $\tilde{\boldsymbol{Y}}^{k}=(\tilde{Y}_1^{k},\ldots,\tilde{Y}_n^{k})^{\mathsf{T}}$, where $\tilde{Y}_i^{k}=I(Y_i>q_k)-I(Y_i\le q_k)$, $i=1,\ldots,n$, $k=1,\ldots,H-1$.
3. Perform eigenvalue decomposition of

$$\hat{\boldsymbol{V}}=[\hat{\boldsymbol{M}}]_p\sum_{k=1}^{H-1}\tilde{\boldsymbol{Y}}^{k}\left(\tilde{\boldsymbol{Y}}^{k}\right)^{\mathsf{T}}[\hat{\boldsymbol{M}}]_p^{\mathsf{T}}. \tag{7}$$

   Denote $\hat{\boldsymbol{u}}_1,\ldots,\hat{\boldsymbol{u}}_d$ as the eigenvectors corresponding to the largest d eigenvalues of $\hat{\boldsymbol{V}}$. The subspace spanned by $\hat{\boldsymbol{u}}_1,\ldots,\hat{\boldsymbol{u}}_d$ is the estimator of $\mathcal{S}_{Y|\boldsymbol{X}}$.

It can be shown that $\hat{\boldsymbol{U}}=(\hat{u}_1,\ldots,\hat{u}_d)$ is asymptotically normal, and the details are provided in the Appendix. According to [32], the computational

complexity of LSSVM is $O(n^3)$ if a direct approach is used to solve the linear system of equations. In step 2 of the algorithm above, each choice of k leads to a different set of $[\hat{\boldsymbol{M}}]_p\tilde{\boldsymbol{Y}}^k$ in step 3. Since we have $H-1$ choices of k and the complexity of eigenvalue decomposition is $O(p^3)$, a direct approach of PLSSVM has the computational complexity of $O((H-1)n^3)$ if we assume $n > p$. If a conjugate gradient method is used to get an approximate solution, the complexity can be reduced to $O((H-1)rn^2)$, where r denotes the rank of $\lambda^{-1}n\hat{\boldsymbol{\Sigma}}^* + (\mathbb{X}^*)^\intercal\mathbb{X}^*$.

2.4. Determination of d

In step 3 of the PLSSVM sample algorithm, we need to know the structural dimension d, which has to be estimated in practice. For classical method SIR, sequential test approaches have been studied in [14] and [33], which depend on the asymptotic distribution of the SIR estimator. A BIC type criterion for SIR is proposed in [34]. Existing SVM-based sufficient dimension reduction estimators have complex asymptotic distributions, and all rely on a cross-validated BIC type criteria. See, for example, PSVM in [19] and principal Lq SVM in [27]. A novel ladle estimator was proposed in [35], and can be used directly for determination of d in the PLSSVM algorithm.

Recall that $\hat{\boldsymbol{u}}_\ell$ is the eigenvector corresponding to $\hat{\lambda}_\ell$, $\ell = 1, \ldots, p$, where $\hat{\lambda}_1 > \hat{\lambda}_2 > \ldots > \hat{\lambda}_p$ are the ordered eigenvalues of $\hat{\boldsymbol{V}}$ in (7). For $k = 1, \ldots, p-1$, denote $\hat{\boldsymbol{U}}_k = (\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_k)$. Take n bootstrap samples from the original data and denote the jth bootstrap sample as $\{(\boldsymbol{X}_i^{(j)}, Y_i^{(j)}) : i = 1, \ldots, n\}$ for $j = 1, \ldots, n$. Recalculate $\hat{\boldsymbol{V}}^{(j)}$ in (7) based on the jth bootstrap sample. Let $\hat{\boldsymbol{U}}_k^{(j)} = (\hat{\boldsymbol{u}}_1^{(j)}, \ldots, \hat{\boldsymbol{u}}_k^{(j)})$, which consists of the eigenvectors corresponding to the k leading eigenvalues of $\hat{\boldsymbol{V}}^{(j)}$. For $k = 1, \ldots, p-1$, set $f_n^0(k) =$

$n^{-1}\sum_{j=1}^{n}\{1-|det(\hat{\boldsymbol{U}}_k^{\intercal}\hat{\boldsymbol{U}}_k^{(j)})|\}$. Let $f_n^0(k)=0$ if $k=0$. Define

$$g_n(k)=\frac{f_n^0(k)}{1+\sum_{\ell=0}^{p-1}f_n^0(\ell)}+\frac{\hat{\lambda}_{k+1}}{1+\sum_{\ell=0}^{p-1}\hat{\lambda}_{\ell+1}},\quad k=0,\ldots,p-1.$$

The final ladle estimator of d is

$$\hat{d}=\arg\min g_n(k), k\in\mathcal{D}_k, \tag{8}$$

where $\mathcal{D}_k$ denotes the domain of k. Although $\mathcal{D}_k$ in the definition of $g_n(k)$ is $\{0,1,\ldots,p-1\}$, we follow [35] and use $\mathcal{D}_k=\{0,1\ldots,\lfloor p/\log p\rfloor\}$, where $\lfloor\cdot\rfloor$ denotes the integer part.

3. PLSSVM for real-time sufficient dimension reduction

Real-time sufficient dimension reduction through PLSSVM is considered in this section. Efficient algorithms for either adding new data or removing old data are provided.

As mentioned in Section 1 real-time SDR is a new concept with the first algorithm being introduced only recently in [21]. The authors introduced their algorithm, called online SIR, which can perform real-time update on sufficient dimension reduction relatively quickly and accurately. There are some shortcomings to their method though. First, it is only an approximation at each step and does not have an exact solution. Second, it can only update a single new observation at each update. Third, is based on inverse moment which have been shown to be less accurate than SVM-based methods. Finally, it can only update when one is adding data while the method we propose here, gives the opportunity to perform real-time updates by removing old data points if they were erroneously included or measured.

3.1. Real-time procedure to add new data

In addition to the currently available data $\{(\boldsymbol{X}_i, Y_i) : i = 1, \ldots, n\}$, suppose we now collect m new data points $\{(\boldsymbol{X}_i, Y_i) : i = n+1, \ldots, n+m\}$, and we want to update the estimation of the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$. Let $\mathbb{X}_N^* = (\mathbb{X}_N, -\mathbf{1}_m)$, where $\mathbb{X}_N = (\boldsymbol{X}_{n+1}, \ldots, \boldsymbol{X}_{n+m})^\intercal$ denotes the new predictors. Let $\hat{\boldsymbol{\Sigma}}_N^*$ be a block diagonal matrix with the diagonal elements being $\hat{\boldsymbol{\Sigma}}_N = m^{-1} \sum_{i=n+1}^{n+m} \boldsymbol{X}_i \boldsymbol{X}_i^\intercal$ and 0. For fixed sets A_1 and A_2, we have $\tilde{Y}_i = I(Y_i \in A_1) - I(Y_i \in A_2)$, $i = 1, \ldots, n+m$. Denote $\tilde{\boldsymbol{Y}}_N = (\tilde{Y}_{n+1}, \ldots, \tilde{Y}_{n+m})^\intercal$, $\tilde{\boldsymbol{Y}}_W = (\tilde{Y}_1, \ldots, \tilde{Y}_{n+m})^\intercal$, and $\mathbb{X}_W = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{n+m})^\intercal$. Parallel to the definitions of $\mathbb{X}_N^*$, and $\hat{\boldsymbol{\Sigma}}_N^*$, define $\mathbb{X}_W^*$ and $\hat{\boldsymbol{\Sigma}}_W^*$.

If we implement PLSSVM based on the whole data according to (6), $\boldsymbol{r}$ is estimated by

$$\hat{\boldsymbol{r}}^W = \left(\frac{(n+m)\hat{\boldsymbol{\Sigma}}_W^*}{\lambda} + (\mathbb{X}_W^*)^\intercal \mathbb{X}_W^* \right)^{-1} (\mathbb{X}_W^*)^\intercal \tilde{\boldsymbol{Y}}_W. \tag{9}$$

It is easy to see that

$$\begin{aligned} (\mathbb{X}_W^*)^\intercal \mathbb{X}_W^* &= (\mathbb{X}^*)^\intercal \mathbb{X}^* + (\mathbb{X}_N^*)^\intercal (\mathbb{X}_N^*), \\ (\mathbb{X}_W^*)^\intercal \tilde{\boldsymbol{Y}}_W &= (\mathbb{X}^*)^\intercal \tilde{\boldsymbol{Y}} + (\mathbb{X}_N^*)^\intercal \tilde{\boldsymbol{Y}}_N, \text{ and} \\ (n+m)\hat{\boldsymbol{\Sigma}}_W^* &= n\hat{\boldsymbol{\Sigma}}^* + m\hat{\boldsymbol{\Sigma}}_N^*. \end{aligned} \tag{10}$$

Denote $\boldsymbol{A} = \lambda^{-1} n \hat{\boldsymbol{\Sigma}}^* + (\mathbb{X}^*)^\intercal \mathbb{X}^*$, $\boldsymbol{B}_1 = \lambda^{-1} m \hat{\boldsymbol{\Sigma}}_N^* + (\mathbb{X}_N^*)^\intercal \mathbb{X}_N^*$, $\boldsymbol{C}_1 = (\mathbb{X}^*)^\intercal \tilde{\boldsymbol{Y}}$, and $\boldsymbol{C}_2 = (\mathbb{X}_N^*)^\intercal \tilde{\boldsymbol{Y}}_N$. (9) and (10) together lead to

$$\hat{\boldsymbol{r}}^W = (\boldsymbol{A} + \boldsymbol{B}_1)^{-1} (\boldsymbol{C}_1 + \boldsymbol{C}_2). \tag{11}$$

Note that $(\boldsymbol{A} + \boldsymbol{B}_1)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}_1(\boldsymbol{I}_{p+1} + \boldsymbol{A}^{-1}\boldsymbol{B}_1)^{-1}\boldsymbol{A}^{-1}$ from formula

25 in [36]. Together with (6), it can be shown that (11) becomes

$$\begin{aligned} \hat{\boldsymbol{r}}^W &= \boldsymbol{s}^W \left(\hat{\boldsymbol{r}} + \boldsymbol{A}^{-1}\boldsymbol{C}_2\right), \text{ where} \\ \boldsymbol{s}^W &= \boldsymbol{I}_{p+1} - \boldsymbol{A}^{-1}\boldsymbol{B}_1(\boldsymbol{I}_{p+1} + \boldsymbol{A}^{-1}\boldsymbol{B}_1)^{-1}. \end{aligned} \tag{12}$$

Note that $\boldsymbol{B}_1$ and $\boldsymbol{C}_2$ depend only on the new data. We do not need access to individual data points in the old data as long as we have $\boldsymbol{A}^{-1}$ and $\hat{\boldsymbol{r}}$.

Now we describe the real-time PLSSVM algorithm to update the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$ estimation when new data are added. The misclassification penalty λ, the number of slices H, and the structural dimension d for $\mathcal{S}_{Y|\boldsymbol{X}}$ are assumed to be known. Assume the predictors are centered such that $\sum_{i=1}^n \boldsymbol{X}_i = \boldsymbol{0}$, $i = 1, \ldots, n+m$. Recall that $\{q_1, \ldots, q_{H-1}\}$ are equally spaced sample percentiles of $\{Y_1, \ldots, Y_n\}$. We also have $\tilde{\boldsymbol{Y}}^k = (\tilde{Y}_1^k, \ldots, \tilde{Y}_n^k)^\intercal$, where $\tilde{Y}_i^k = I(Y_i > q_k) - I(Y_i \leq q_k)$ for $i = 1, \ldots, n$.

1. From the PLSSVM estimation based on the current data, extract $\boldsymbol{A}^{-1} = \left(\lambda^{-1} n \hat{\boldsymbol{\Sigma}}^* + (\mathbb{X}^*)^\intercal \mathbb{X}^*\right)^{-1}$ and $\hat{\boldsymbol{r}}_k = \hat{\boldsymbol{M}}\tilde{\boldsymbol{Y}}_k$, $k = 1, \ldots, H-1$. Here $\hat{\boldsymbol{M}}$ is defined in (6).
2. Based on the additional data, calculate $\boldsymbol{B}_1 = \lambda^{-1} m \hat{\boldsymbol{\Sigma}}_N^* + (\mathbb{X}_N^*)^\intercal \mathbb{X}_N^*$ and $\boldsymbol{C}_{2,k} = (\mathbb{X}_N^*)^\intercal \tilde{\boldsymbol{Y}}_N^k$, $k = 1, \ldots, H-1$. Here $\tilde{\boldsymbol{Y}}_N^k = (\tilde{Y}_{n+1}^k, \ldots, \tilde{Y}_{n+m}^k)^\intercal$ with $\tilde{Y}_i^k = I(Y_i > q_k) - I(Y_i \leq q_k)$ for $i = n+1, \ldots, n+m$.
3. For $k = 1, \ldots, H-1$, update $\hat{\boldsymbol{r}}_k$ to $\hat{\boldsymbol{r}}_k^W = \boldsymbol{s}^W(\hat{\boldsymbol{r}}_k + \boldsymbol{A}^{-1}\boldsymbol{C}_{2,k})$, where $\boldsymbol{s}^W$ is defined in (12). Denote $[\boldsymbol{r}_k^W]_p$ as the vector that contains the first p elements of $\boldsymbol{r}_k^W$.
4. Perform eigenvalue decomposition of

$$\hat{\boldsymbol{V}}^W = \sum_{k=1}^{H-1} [\boldsymbol{r}_k^W]_p [\boldsymbol{r}_k^W]_p^\intercal.$$

Denote $\hat{\boldsymbol{u}}_1^W, \ldots, \hat{\boldsymbol{u}}_d^W$ as the eigenvectors corresponding to the largest d eigenvalues of $\hat{\boldsymbol{V}}^W$. The subspace spanned by $\hat{\boldsymbol{u}}_1^W, \ldots, \hat{\boldsymbol{u}}_d^W$ is the updated estimator of $\mathcal{S}_{Y|\boldsymbol{X}}$.

5. Update $\boldsymbol{A}^{-1}$ to $(\boldsymbol{A}^W)^{-1} = s^W \boldsymbol{A}^{-1}$.

The updated central subspace estimator is provided in step 4. When there is a contious stream of incoming data and more updates are needed, $\hat{\boldsymbol{r}}_k^W$ in step 3 and $(\boldsymbol{A}^W)^{-1}$ in step 5 can be used to replace $\hat{\boldsymbol{r}}_k$ and $\boldsymbol{A}^{-1}$ in step 1, and we repeat the above algorithm in an iterative fashion. In terms of storing the current data for the computer memory, the algorithm above needs $\boldsymbol{A}^{-1} \in \mathbb{R}^{(p+1)\times(p+1)}$, $H-1$ vectors $\hat{\boldsymbol{r}}_k \in \mathbb{R}^{p+1}$, and $H-1$ cutoff points q_k. This is significantly less than the original $n \times (p+1)$ dimensional data. An online sufficient dimension reduction method based on SIR was proposed in [21]. Their approach updates one additional observation at a time, and becomes computationally costly when a large number of updates are needed.

3.2. Real-time procedure to remove old data

In many applications, we may discard old data points for a variety of reasons. For example, measurements may become outdated due to usage of new recording instruments, or there may be changes in some regulations and data recorded before the regulatory changes are obsolete. Given the current data $\{(\boldsymbol{X}_i, Y_i) : i = 1, \ldots, n\}$, suppose without loss of generality that the first l ($l < n$) data points are deleted, and we want to update the estimation of the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$. Let $\mathbb{X}_D^* = (\mathbb{X}_D, \mathbf{1}_l)$, where $\mathbb{X}_D = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_l)^\intercal$ denotes the predictors to be dropped. Let $\hat{\boldsymbol{\Sigma}}_D^*$ be a block diagonal matrix with the diagonal elements being $\hat{\boldsymbol{\Sigma}}_D = l^{-1} \sum_{i=1}^l \boldsymbol{X}_i \boldsymbol{X}_i^\intercal$ and 0. For fixed sets

A_1 and A_2, recall that $\tilde{Y}_i = I(Y_i \in A_1) - I(Y_i \in A_2)$, $i = 1, \ldots, n$. Denote $\tilde{\boldsymbol{Y}}_D = (\tilde{Y}_1, \ldots, \tilde{Y}_l)^\intercal$.

Now we implement PLSSVM after removing the first l observations. Parallel to (11), it can be shown that $\boldsymbol{r}$ is estimated by

$$\hat{\boldsymbol{r}}^\omega = (\boldsymbol{A} - \boldsymbol{B}_2)^{-1}(\boldsymbol{C}_1 - \boldsymbol{C}_3), \tag{13}$$

where $\boldsymbol{A} = \lambda^{-1} n \hat{\boldsymbol{\Sigma}}^* + (\mathbb{X}^*)^\intercal \mathbb{X}^*$ and $\boldsymbol{C}_1 = (\mathbb{X}^*)^\intercal \tilde{\boldsymbol{Y}}$ are defined as before, $\boldsymbol{B}_2 = \lambda^{-1} l \hat{\boldsymbol{\Sigma}}^*_D + (\mathbb{X}^*_D)^\intercal \mathbb{X}^*_D$ and $\boldsymbol{C}_3 = (\mathbb{X}^*_D)^\intercal \tilde{\boldsymbol{Y}}_D$ are computed from the deleted data. Note that $(\boldsymbol{A} - \boldsymbol{B}_2)^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}_2(\boldsymbol{I}_{p+1} - \boldsymbol{A}^{-1}\boldsymbol{B}_2)^{-1}\boldsymbol{A}^{-1}$. Thus (13) leads to

$$\begin{aligned} &\hat{\boldsymbol{r}}^\omega = \boldsymbol{s}^\omega \left(\hat{\boldsymbol{r}} - \boldsymbol{A}^{-1}\boldsymbol{C}_3 \right), \text{ where} \\ &\boldsymbol{s}^\omega = \boldsymbol{I}_{p+1} + \boldsymbol{A}^{-1}\boldsymbol{B}_2(\boldsymbol{I}_{p+1} - \boldsymbol{A}^{-1}\boldsymbol{B}_2)^{-1}. \end{aligned} \tag{14}$$

Next we describe the real-time PLSSVM algorithm to estimate $\mathcal{S}_{Y|\boldsymbol{X}}$ when old data are deleted. Assume λ, H and d are known, and the predictors are centered. Recall that $\{q_1, \ldots, q_{H-1}\}$ are sample percentiles of $\{Y_1, \ldots, Y_n\}$ and $\tilde{\boldsymbol{Y}}^k = (\tilde{Y}^k_1, \ldots, \tilde{Y}^k_n)^\intercal$.

1. From the PLSSVM estimation based on the current data, extract $\boldsymbol{A}^{-1} = \left(\lambda^{-1} n \hat{\boldsymbol{\Sigma}}^* + (\mathbb{X}^*)^\intercal \mathbb{X}^* \right)^{-1}$ and $\hat{\boldsymbol{r}}_k = \hat{\boldsymbol{M}} \tilde{\boldsymbol{Y}}_k$, $k = 1, \ldots, H-1$. Here $\hat{\boldsymbol{M}}$ is defined in (6).
2. Based on the data to be deleted, calculate $\boldsymbol{B}_2 = \lambda^{-1} l \hat{\boldsymbol{\Sigma}}^*_D + (\mathbb{X}^*_D)^\intercal \mathbb{X}^*_D$ and $\boldsymbol{C}_{3,k} = (\mathbb{X}^*_D)^\intercal \tilde{\boldsymbol{Y}}^k_D$, $k = 1, \ldots, H-1$. Here $\tilde{\boldsymbol{Y}}^k_D = (\tilde{Y}^k_1, \ldots, \tilde{Y}^k_l)^\intercal$ with $\tilde{Y}^k_i = I(Y_i > q_k) - I(Y_i \le q_k)$ for $i = 1, \ldots, l$.
3. For $k = 1, \ldots, H-1$, update $\hat{\boldsymbol{r}}_k$ to $\hat{\boldsymbol{r}}^\omega_k = \boldsymbol{s}^\omega(\hat{\boldsymbol{r}}_k - \boldsymbol{A}^{-1}\boldsymbol{C}_{3,k})$, where $\boldsymbol{s}^\omega$ is defined in (14). Denote $[\boldsymbol{r}^\omega_k]_p$ as the vector that contains the first p elements of $\boldsymbol{r}^\omega_k$.

4. Perform eigenvalue decomposition of

$$\hat{\boldsymbol{V}}^{\omega} = \sum_{k=1}^{H-1} [\boldsymbol{r}_k^{\omega}]_p [\boldsymbol{r}_k^{\omega}]_p^{\intercal}.$$

Denote $\hat{\boldsymbol{u}}_1^{\omega}, \ldots, \hat{\boldsymbol{u}}_d^{\omega}$ as the eigenvectors corresponding to the largest d eigenvalues of $\hat{\boldsymbol{V}}^{\omega}$. The subspace spanned by $\hat{\boldsymbol{u}}_1^{\omega}, \ldots, \hat{\boldsymbol{u}}_d^{\omega}$ is the updated estimator of $\mathcal{S}_{Y|\boldsymbol{X}}$.

We remark that the online SIR approach in [21] did not discuss removal of old data.

4. Simulations

In this section we evaluate the performance of the proposed methods through simulations. We consider the following models:

$$\begin{aligned}
&\text{model I:} \;\; Y = X_1 + X_2 + \sigma\varepsilon, \\
&\text{model II:} \;\; Y = X_1/[0.5 + (X_2 + 1)^2] + \sigma\varepsilon, \\
&\text{model III:} \;\; Y = X_1(X_1 + X_2 + 1) + \sigma\varepsilon.
\end{aligned}$$

Here $\boldsymbol{X} = (X_1, \ldots, X_p)^{\intercal}$ is $N(\boldsymbol{0}, \boldsymbol{I}_p)$, $\varepsilon \sim N(0, 1)$ is independent of $\boldsymbol{X}$, and σ is fixed to be 0.2. In section 4.1, we demonstrate the superior performance of the PLSSVM estimator from section 2.3 over SIR [14] and PSVM [19]. In section 4.2, we evaluate the performance of the real-time PLSSVM algorithm from section 3.1. Compared to the online SIR approach in [21], our proposal enjoys better estimation accuracy and is significantly faster with streamed batch data.

4.1. Sufficient dimension reduction with static data

Denote $\hat{\mathcal{S}}$ as the estimator of the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$. Let $\mathbf{P}(\mathcal{S})$ be the orthogonal projection onto the subspace $\mathcal{S}$. We measure the performance of estimator $\hat{\mathcal{S}}$ by

$$\Delta = \|\mathbf{P}(\hat{\mathcal{S}}) - \mathbf{P}(\mathcal{S}_{Y|\boldsymbol{X}})\|, \tag{15}$$

where $\|\cdot\|$ is the Frobenius norm. Smaller values of Δ mean better estimation. With $n = 100$, $H = 20$, $\lambda = 1$ and $p = 10, 20, 30$, we report the mean of Δ and its standard error in Table 1. While the classical SIR serves as a benchmark, PSVM is considered as a state-of-the-art approach for sufficient dimension reduction. PLSSVM consistently outperforms SIR and PSVM across all three models. Additional simulation results, although not shown here, demonstrate that the performance of PLSSVM is not sensitive to the number of slices H or the misclassification penalty λ.

Recall that step 3 of the PLSSVM algorithm from section 2.3 requires the knowledge of the structural dimension d of the central space $\mathcal{S}_{Y|\boldsymbol{X}}$, which has to be estimated in practice. We summarize the performance of the ladle estimator $\hat{d}$ in (8) in Table 2. The proportions of correct estimation, or $\hat{d} = d$, are reported based on 100 repetitions. Note that $d = 1$ for model I and $d = 2$ for model Ⅱ and model Ⅲ. For model I with a simpler structure, the ladle estimator is not very sensitive to the change in H with fixed (n, p), becomes better when n increases with fixed (p, H), and deteriorates with increasing p and fixed (n, H). The trend is less obvious for the more complex model Ⅱ and model Ⅲ. For $(n, p) = (100, 30)$, the ladle estimator becomes worse when H increases. For model Ⅱ and model Ⅲ with $n = 400$ and fixed p, the ladle estimator becomes better when H increases.

Table 1: Estimation of $\mathcal{S}_{Y|\boldsymbol{X}}$ based on $(n, H) = (100, 20)$. The mean of Δ in (15) and its standard error (in parentheses) are reported based on 100 repetitions.

model	p	SIR	PSVM	PLSSVM
I	10	0.19 (0.050)	0.22 (0.058)	0.15 (0.043)
	20	0.30 (0.051)	0.33 (0.059)	0.24 (0.048)
	30	0.38 (0.061)	0.43 (0.075)	0.32 (0.064)
II	10	0.83 (0.214)	0.92 (0.224)	0.73 (0.177)
	20	1.14 (0.156)	1.21 (0.148)	1.04 (0.140)
	30	1.31 (0.120)	1.36 (0.112)	1.23 (0.121)
III	10	1.21 (0.210)	1.20 (0.232)	1.11 (0.243)
	20	1.56 (0.167)	1.47 (0.181)	1.43 (0.188)
	30	1.70 (0.135)	1.63 (0.133)	1.59 (0.143)

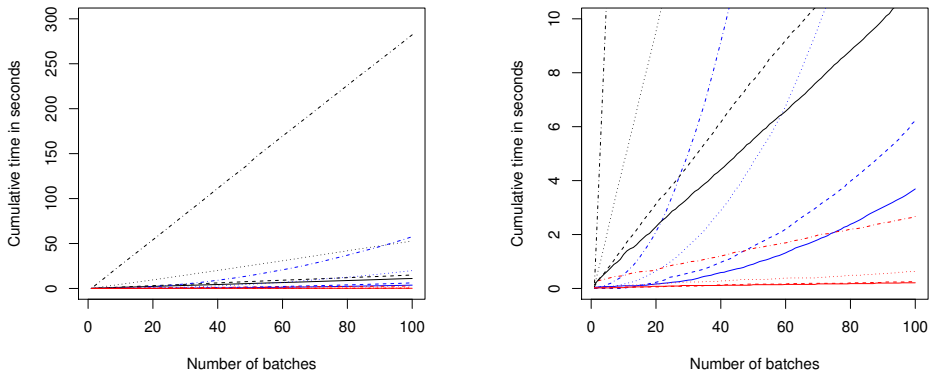
4.2. Sufficient dimension reduction with streamed data

First we generate a million observations from model I. These observations are split into 100 batches with 1000 observations in each batch. We assume that the data arrives in batches continuously. Each time a new batch arrives, we update the estimation of the central subspace. We fix $H = 20$, $\lambda = 1$, and consider $p = 10, 20, 50, 100$. We compare the computation time of our proposed real-time procedure with online SIR. [21] proposed two alternative online SIR algorithms: online SIR via perturbation and online SIR via gradient descent optimization. We only include online SIR via gradient descent optimization, which is faster than its perturbation counterpart. The original SIR is also included as a benchmark. PSVM is known to be slower than SIR [19], and is not included in this comparison. From panel (a) of Figure

Table 2: Proportions of $\hat{d}$ in (8) equal to true d are reported based on 100 repetitions.

model	p	$H = 10$			$H = 20$			$H = 50$		
		n=100	n=200	n=400	n=100	n=200	n=400	n=100	n=200	n=400
	10	0.93	0.99	0.98	0.91	0.99	0.96	0.89	0.96	0.98
I	20	0.73	0.96	0.95	0.70	0.94	0.98	0.65	0.85	0.96
	30	0.17	0.93	0.95	0.02	0.78	0.93	0.00	0.60	0.88
	10	0.16	0.06	0.05	0.30	0.21	0.20	0.47	0.36	0.63
II	20	0.62	0.29	0.19	0.76	0.47	0.48	0.78	0.69	0.62
	30	0.42	0.60	0.45	0.31	0.85	0.73	0.10	0.81	0.90
	10	0.22	0.05	0.02	0.45	0.30	0.16	0.78	0.57	0.66
III	20	0.55	0.36	0.21	0.57	0.67	0.48	0.32	0.83	0.86
	30	0.14	0.71	0.42	0.06	0.71	0.88	0.00	0.51	0.82

1, we see that both real-time PLSSVM and the original SIR are faster than online SIR when the number of batches is close to 100. While our proposed real-time algorithm can deal with batch data directly, the online SIR method in [21] updates the estimator each time a single observation arrives. When a new batch with 1000 observations arrives, online SIR has to make 1000 updates in comparison to a single updating step for real-time PLSSVM. Although online SIR is more comparable to SIR and real-time PLSSVM for $p = 10, 20, 50$, it becomes much worse for $p = 100$. In panel (b) of Figure 1, we zoom in to take a closer look. We see that the computation time of online SIR and real-time PLSSVM is a linear function of the number of batches, and the computation time of SIR is curved. If we focus on the comparison with $p = 100$ across three methods, we see that SIR is faster than real-time PLSSVM at first, and becomes slower than real-time PLSSVM after about 10 batches. Moreover, the gap grows quickly as number of batches increases



(a) (b)

Figure 2: Computation time for model I with streamed batch data. Each batch has 1000 observations. SIR (blue), online SIR (black) and real-time PLSSVM (red) are compared for $p = 10$ (solid), $p = 20$ (dashed), $p = 50$ (dotted) and $p = 100$ (dot-dashed). Panel (a): the entire picture. Panel (b): a closer look.

due to the curvature of the blue dot-dashed line. We can foresee that with enough number of batches, SIR can eventually be slower than online SIR.

Next we consider a setting with streamed single data. In each repetition, we generate 1000 observations from model Ⅱ and model Ⅲ. The first 100 observations are used to get an initial central subspace estimator. For the additional 900 observations, we assume they arrive one by one in a continuous fashion, and we update the estimation each time a new observation arrives. We compare the estimation accuracy between real-time PLSSVM and online SIR. The initial estimator for real-time PLSSVM is the PLSSVM estimator from section 2.3, while the initial estimator of online SIR is described in [21]. We fix $H = 20$, $\lambda = 1$, and consider $p = 10, 20, 30$. In Figure 2, we plot the

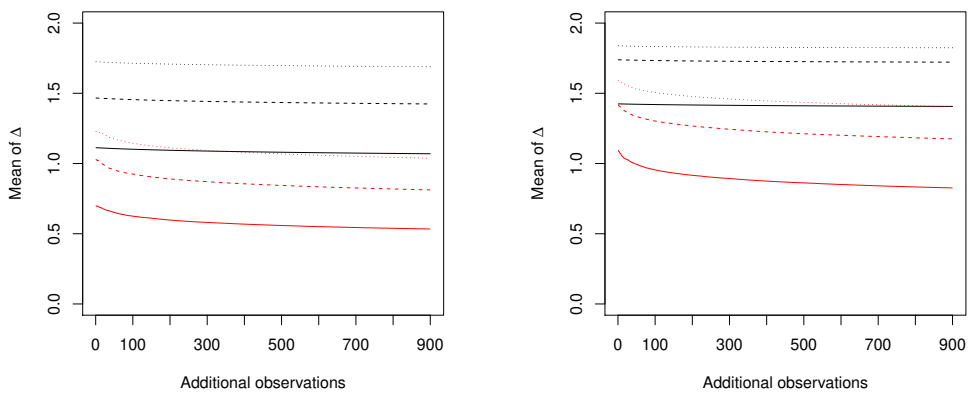
mean of Δ in (15) based on 100 repetitions versus the number of additional observations. Both panels show the superiority of real-time PLSSVM for all p, and the improvement of PLSSVM over online SIR grows as more additional data come in. While the online SIR estimator does not improve much over its initial estimator with additional data points, the real-time PLSSVM curves dip significantly with the first 200 or so additional data points, and gradually level off after about 400 additional data points. In terms of computation time for model $\mathbb{I}$ with 100 repetitions, online SIR takes 49.64 seconds, 53.49 seconds and 60.06 seconds for $p = 10, 20, 30$, respectively. Real-time PLSVM, on the other hand, takes 35.15 seconds, 48.04 seconds and 71.21 seconds for $p = 10, 20, 30$, respectively. The computation time for model $\mathbb{III}$ is basically the same as model $\mathbb{II}$, and is thus omitted. We see that even online SIR is designed specifically for streamed single data, the computation time of real-time PLSSVM is comparable.

5. Real data applications

We evaluate the performance of the proposed algorithms with two real data sets. The bike sharing data and the facial expression data are studied in section 5.1 and section 5.2, respectively. Both data sets can be downloaded from the UCI machine learning repository at $http://archive.ics.uci.edu/ml$.

5.1. Bike sharing data

The bike sharing data [37] contains the hourly and daily counts of rental bikes in a bikeshare system between 2011 and 2012 together with weather information. We use number of casual users as the response, and use three continuous weather-related variables as the predictor: normalized feeling



(a) (b)

Figure 3: Estimation of $\mathcal{S}_{Y|\boldsymbol{X}}$ with streamed single data. The mean of Δ in (15) is plotted based on 100 repetitions. Real-time PLSSVM (red) and online SIR (black) are compared for $p = 10$ (solid), $p = 20$ (dashed) and $p = 30$ (dotted). Panel (a): model Ⅱ. Panel (b): model Ⅲ.

temperature in Celsius, normalized humidity, and normalized wind speed. While the original data set has 731 days, we remove the observations from weekends and holidays to focus on 500 working days.

First we compare the performances of five different methods to estimate the central subspace using the daily counts. SIR, PSVM and PLSSVM are included as three static methods. We also include two online estimators: real-time PLSSVM and online SIR. For the static methods, we use all 500 working days to get the final estimator directly. For the online methods, we use the first 20 working days to get an initial estimator, and then update the estimator each time a new daily observation comes in. Ladle estimator indicates that $\hat{d} = 1$ for this data set, and the estimated basis of the central

Table 3: Estimated basis coefficients for the bike sharing data. Daily data are used based on 500 working days.

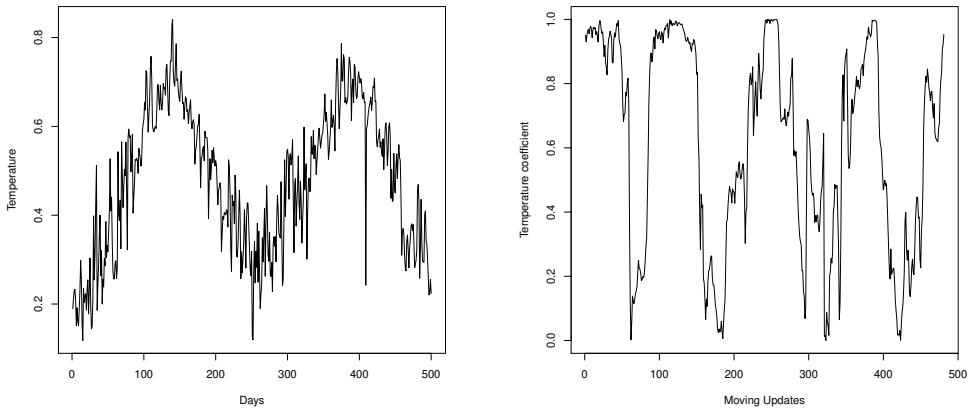
method	temperature	humidity	wind speed
SIR	0.934	-0.298	-0.199
PSVM	0.952	-0.266	-0.154
PLSSVM	0.912	-0.331	-0.241
online SIR	0.784	-0.449	-0.429
real-time PLSSVM	0.928	-0.309	-0.207

subspace is a three-dimensional vector. We refer to each component of this vector as the basis coefficient, and report the estimated coefficients in Table 3. For online SIR and real-time PLSSVM, only the final estimator is reported. We note that the signs for a fixed predictor variable are the same across five methods. It is also noticeable that the temperature coefficient has the largest magnitude, indicating that it has the largest effect on the count of casual users.

All the estimators are close to each other with the exception of online SIR. Due to the rounding error of each update, the real-time PLSSVM estimator and PLSSVM estimator are not exactly the same but fairly close. Upon closer examination of the online SIR algorithm in [21], we realize that the initial estimator used in the algorithm is based on $\boldsymbol{X}$-scaled SIR. On the other hand, the classical SIR algorithm is based on $\boldsymbol{Z}$-scaled SIR, which finds the central subspace $\mathcal{S}_{Y|\boldsymbol{Z}}$ between Y and standardized predictor $\boldsymbol{Z} = \boldsymbol{\Sigma}^{-1/2}(\boldsymbol{X} - E(\boldsymbol{X}))$ before transforming $\mathcal{S}_{Y|\boldsymbol{Z}}$ back to $\mathcal{S}_{Y|\boldsymbol{X}}$. Although they are equivalent at the population level, it is generally believed that $\boldsymbol{Z}$-scaled SIR performs better at the sample level. If we compare the initial estimators from Figure 2 with

the corresponding results in Table 1, one can clearly see that the initial values for the PLSSVM estimators match those reported in Table 1, while the initial values for online SIR estimators are worse than the corresponding SIR estimators in Table 1. The difference in the predictor scale together with the cumulative rounding errors in multiple updating steps explain the discrepancy between SIR and online SIR in Table 3.

Next we combine the real-time PLSSVM algorithms in section 3.1 and section 3.2 to consider a moving window estimator based on the daily counts. We start with the first 20 working days to get an initial PLSSVM estimator. As a new working day observation comes in, we add this data point (day 21) to update the initial estimator following the algorithm in section 3.1, and then we drop the oldest data point (day 1) following the algorithm in section 3.2. We continue this procedure each time a new data point is collected. From 500 working days, we get 481 estimators, where each estimator is based on a moving window of 20 days. In panel (a) of Figure 3, we plot the normalized feeling temperature versus the day. We see a clear seasonal trend, as the valleys correspond to days in winter and the peaks correspond to the summer days. In panel (b) of Figure 3, we plot the estimated coefficient for temperature versus the number of the moving updates. Note that the estimated central subspace basis vector always has unit length, which implies that the estimated coefficients are always between -1 and 1. We always set the estimated temperature coefficient to be positive for easy comparison. We see from Table 3 that the temperature coefficient has the largest magnitude compared to humility and wind speed, suggesting that temperature is the most important factor to affect the number of casual users. However, this



(a) (b)

Figure 4: Daily bike sharing data with 500 working days. Panel (a): temperature v.s. number of days. Panel (b): estimated temperature coefficient of moving window estimator v.s. number of moving updates.

is only true when we consider the whole data set. Based on the moving window estimators in panel (b), we see that the magnitude of the temperature coefficient is only close to 1 when there is extreme temperature, or the x-axis locations of the peaks in panel (b) roughly correspond to the x-axis locations of the peaks and the valleys in panel (a). When the estimated temperature coefficient is close to 0, the corresponding temperature is mild. This suggests that the effect of temperature on number of casual users is heightened with extreme temperature, and is dampened with mild temperature.

Last but not least, we use the hourly counts of the 500 working days to compare the computation efficiency in the presence of streamed batch data. Here we assume the data come in batches every day, with each batch containing 20 to 24 hourly counts. Due to some missing observations, we

do not always have 24 hourly counts for each day. After we get the initial estimator based on the first batch, we update the estimator every day with a new batch coming in. We get 500 estimators from 500 working days, where each estimator is based on all the hourly observations up to that day. Four methods are considered: real-time PLSSVM, SIR, online SIR, and PSVM. The computation time is 0.24 seconds for real-time PLSSVM, 1.81 seconds for SIR, 0.33 seconds for online SIR, and 9274.1 seconds for PSVM. This confirms what we have seen in Figure 1 of section 4.2, where it shows that the difference in computation time between SIR, online SIR and real-time PLSSVM is not significant for small p and reasonable n. PSVM is time-consuming because it involves quadratic programming.

5.2. Facial expression data

The facial expression data [38] is composed by 18 videos. In each video, a user performs grammatical facial expressions from Brazilian Sign Language. The video is then divided into frames identified by a timestamp. The image of each frame is manually labeled by a specialist. The (x, y, z) coordinates for 100 preselected points on the face are also recorded for each frame. For example, suppose the preselected point is the tip of the nose. Then (x, y) coordinates give the pixel position of the nose tip, and z measures the depth in millimeters for the nose tip. This can be considered as a classification problem, and the response is whether a given facial expression is present or not. Here we perform sufficient dimension reduction as a feature extraction step instead of carrying out the classification. This data can be viewed as streamed batch data, with each video being a batch with multiple frames as the individual data points. One can extract an initial feature based on

the first video, and then update the extracted feature after a new video is recorded. There will be 17 updates if all 18 videos are used. All videos have 300 (100×3) predictors, and the average number of frames in each video is 1550. We compare the computation time for feature extraction with updates. Four methods are included for the comparison: real-time PLSSVM, SIR, online SIR and PSVM. It takes 7.48 seconds for real-time PLSSVM to get all 17 updates, 88.12 seconds for SIR to get all 17 updates, 1372.39 seconds for online SIR to get all 17 updates, and 3767.28 seconds for PSVM to get the first 3 updates. As we have seen from Figure 1, online SIR is computationally expensive with streamed batch data when both p and n are large.

6. Discussion

In this paper, we propose a novel SVM-based approach for sufficient dimension reduction. PSVM [19] leverages the prowess of soft-margin SVM in classification for sufficient dimension reduction, and motivates our proposal of PLSSVM as a modified version of least squares SVM. While soft-margin SVM and least squares SVM are used for classification, PSVM and PLSSVM are designed for recovering the central subspace. Compared to PSVM, PLSSVM achieves more accurate estimation of the central subspace and is computationally more efficient. Moreover, PSVM is only applicable with static data, while PLSSVM handles both static and streamed data in a unified framework.

As an online approach, real-time PLSSVM algorithms are proposed to update the current dimension reduction estimator efficiently either when new

data points are added or when old data points are deleted. Online SIR was recently proposed by [21], and is considered as a main competitor to real-time PLSSVM for online sufficient dimension reduction. Online SIR has the following limitations as a real-time approach. First, online SIR only deals with streamed single data and can not deal with streamed batch data directly. Second, online SIR only deals with the case when a new data point is added, but does not discuss the case when old data points are deleted. Last but not least, online SIR is not consistent with the classical SIR. Namely, for the same data set, the estimator of SIR with the whole data set can be very different from the final estimator of online SIR if the data is treated as streamed observations. Real-time PLSSVM successfully addresses all the limitations. It is efficient with both streamed batch data and streamed single data. It can be easily updated either with additional data or deleted data. For the same data, static PLSSVM based on all the observations is consistent with the real-time PLSSVM when the data is treated as streamed observations. In terms of estimation accuracy, real-time PLSSVM is consistently better than online SIR in a wide range of simulation settings.

Nonlinear sufficient dimension reduction aims to find mapping $\phi : \mathbb{R}^p \to \mathbb{R}^d$ such that the conditional distribution of Y given $\phi(\boldsymbol{X})$ is the same as the conditional distribution of Y given $\boldsymbol{X}$. In the case when $\phi(\boldsymbol{X}) = \boldsymbol{\eta}^\intercal \boldsymbol{X}$ for $\boldsymbol{\eta} \in \mathbb{R}^{p\times d}$, nonlinear sufficient dimension reduction reduces to linear sufficient dimension defined through (1). Following nonlinear PSVM [19], one can easily extend the PLSSVM for nonlinear sufficient dimension reduction. Development of real-time nonlinear sufficient dimension reduction is worth future investigation. Similar to the connections between least squares SVM

and least squares regression, regularized least squares regression is linked to a regularized least squares SVM approach known as the proximal SVM [39]. A future research direction is to develop new real-time sufficient dimension reduction methods based on proximal SVM.

Acknowledgements

The authors would like to thank the associate editor and three anonymous reviewers whose comments greatly improved this article. Thanks also go to Zhanrui Cai for sharing the online SIR code.

Appendix: Asymptotic analysis of principal least squares SVM

We derive the asymptotic distribution of the PLSSVM estimator in section 2.3. Some notations are needed first. Let

$$m(\boldsymbol{\theta}, \boldsymbol{Z}) = \boldsymbol{\theta}^\intercal \boldsymbol{\Sigma}^* \boldsymbol{\theta} - \lambda(1 - \boldsymbol{\theta}^\intercal \boldsymbol{X}^* \tilde{Y})^2,$$

where $\boldsymbol{\theta} = (\boldsymbol{\psi}^\intercal, t)^\intercal, \boldsymbol{Z} = (\boldsymbol{X}^\intercal, \tilde{Y})^\intercal, \boldsymbol{X}^* = (\boldsymbol{X}^\intercal, -1)^\intercal$ and $\boldsymbol{\Sigma}^* = \mathrm{diag}(\boldsymbol{\Sigma}, 0)$. Also, let $D_{\boldsymbol{\theta}}$ denote the $(p+1)$-dimensional column vector of differential operators $(\partial/\partial\theta_1, \ldots, \partial/\partial\theta_{p+1})^\intercal$.

Then $L(\boldsymbol{\psi}, t)$ in (4) becomes $E[m(\boldsymbol{\theta}, \boldsymbol{Z})]$, and its gradient is given by

$$D_{\boldsymbol{\theta}} E[m(\boldsymbol{\theta}, \boldsymbol{Z})] = (2\boldsymbol{\psi}^\intercal \boldsymbol{\Sigma}, 0)^\intercal - 2\lambda E[\mathbf{X}^* \tilde{Y}(1 - \boldsymbol{\theta}^\intercal \mathbf{X}^* \tilde{Y})]. \tag{16}$$

If the gradient function (16) is differentiable with respect to $\boldsymbol{\theta}$, the Hessian matrix becomes

$$\boldsymbol{H} = 2\mathrm{diag}(\boldsymbol{\Sigma}, 0) + 2\lambda E(\mathbf{X}^* \tilde{Y} \tilde{Y}^\intercal (\mathbf{X}^*)^\intercal).$$

Applying Theorem 5.23 of [40], the corresponding influence function of $\hat{\boldsymbol{\theta}}$ is given by

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1}\{(2\boldsymbol{\psi}_0^\intercal\boldsymbol{\Sigma}, 0)^\intercal - 2\lambda^* E_n[\mathbf{X}^*\tilde{Y}(1 - \tilde{Y}\boldsymbol{\theta}_0^\intercal\mathbf{X}^*)]\} + o_P(n^{-\frac{1}{2}}). \quad (17)$$

Consider a fixed dividing point q_k, where $k \in \{1, \ldots, H-1\}$. Let $\boldsymbol{Z}^k = (\boldsymbol{X}^\intercal, \tilde{Y}^k)^\intercal$, where $\tilde{Y}_k = I(Y > q_k) - I(Y \leq q_k)$. Let $\boldsymbol{\theta}_{0k} = (\boldsymbol{\psi}_{0k}^\intercal, t_{0k})^\intercal$ be the minimizer of $E(m(\boldsymbol{\theta}, \boldsymbol{Z}^k))$. Similarly, $\hat{\boldsymbol{\theta}}_{0k} = (\hat{\boldsymbol{\psi}}_{0k}^\intercal, \hat{t}_{0k})^\intercal$ denotes the minimizer of $E_n(m(\boldsymbol{\theta}, \boldsymbol{Z}^k))$. Denote

$$s_k(\boldsymbol{\theta}, \boldsymbol{Z}^k) = -[\boldsymbol{H}^{-1}]_p\{(2\boldsymbol{\psi}_{0k}^\intercal\boldsymbol{\Sigma}, 0)^\intercal - 2\lambda E_n[\mathbf{X}^*\tilde{Y}^k(1 - \tilde{Y}^k\boldsymbol{\theta}_{0k}^\intercal\mathbf{X}^*)]\}, \quad (18)$$

where $[\boldsymbol{H}^{-1}]_p$ consists of the first p rows of the inverse Hessian matrix $\boldsymbol{H}^{-1}$. From (17), we have

$$\hat{\boldsymbol{\psi}}_k = \boldsymbol{\psi}_{0k} + s_k(\boldsymbol{\theta}, \boldsymbol{Z}^k) + o_P(n^{-\frac{1}{2}}).$$

For $\hat{\boldsymbol{V}} = \sum_{k=1}^{H-1} \hat{\boldsymbol{\psi}}_{0k}\hat{\boldsymbol{\psi}}_{0k}^\intercal$ in (7), its corresponding population counterpart is $\boldsymbol{V} = \sum_{k=1}^{H-1} \boldsymbol{\psi}_{0k}\boldsymbol{\psi}_{0k}^\intercal$. Before we give the asymptotic distribution for $\hat{\boldsymbol{V}}$, we introduce the notion of a commutation matrix [41]. For a $p \times q$ matrix $\boldsymbol{A}$, a commutation matrix $\boldsymbol{K}_{p,q}$ is the unique $pq \times pq$ matrix such that $\boldsymbol{K}_{p,q}\text{vec}(\boldsymbol{A}) = \text{vec}(\boldsymbol{A}^\intercal)$. Let $\boldsymbol{\Lambda}_1 = \boldsymbol{I}_{p^2} + \boldsymbol{K}_{p,p}$ and

$$\boldsymbol{\Lambda}_2 = \sum_{r=1}^{H-1}\sum_{k=1}^{H-1}(\boldsymbol{\psi}_{0r}\boldsymbol{\psi}_{0k}^\intercal \otimes E[s_r(\boldsymbol{\theta}_{0r}, \boldsymbol{Z}^r)s_k^\intercal(\boldsymbol{\theta}_{0k}, \boldsymbol{Z}^k)]),$$

where $s_k(\boldsymbol{\theta}, \boldsymbol{Z}^k)$ is defined in (18). We have the following theorem

Theorem 2. *Suppose $L(\boldsymbol{\psi}, t)$ in (4) is everywhere twice differentiable with respect to $\boldsymbol{\theta}$. Then $\sqrt{n}\text{vec}(\hat{\boldsymbol{V}} - \boldsymbol{V}) \xrightarrow{\mathcal{D}} N(\mathbf{0}, \boldsymbol{\Lambda}_1\boldsymbol{\Lambda}_2\boldsymbol{\Lambda}_1)$ as $n \to \infty$, where $\xrightarrow{\mathcal{D}}$ means converge in distribution.*

The proof is similar to Theorem 7 in [19] and is omitted.

Recall that $\hat{\boldsymbol{U}} = (\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_d)$, where $\hat{\boldsymbol{u}}_\ell$ is the eigenvector corresponding to the ℓth largest eigenvalue of $\hat{\boldsymbol{V}}$. Let $\boldsymbol{U}$ be the population counterpart of $\hat{\boldsymbol{U}}$. Denote $\boldsymbol{D}$ as a diagonal matrix with diagonal elements being the nonzero eigenvalues of $\boldsymbol{V}$. Using Corollary 1 of [42], we have

Corollary 1. *Suppose $L(\boldsymbol{\psi}, t)$ in (4) is everywhere twice differentiable with respect to $\boldsymbol{\theta}$ and rank$(\boldsymbol{V}) = d$. Then*

$$\sqrt{n}\,\mathrm{vec}(\hat{\boldsymbol{U}} - \boldsymbol{U}) \xrightarrow{\mathcal{D}} N(\boldsymbol{0}, (\boldsymbol{D}^{-1}\boldsymbol{U}^\top \otimes \boldsymbol{I}_p)\boldsymbol{\Lambda}_1\boldsymbol{\Lambda}_2\boldsymbol{\Lambda}_1(\boldsymbol{D}^{-1}\boldsymbol{U}^\top \otimes \boldsymbol{I}_p)),$$

as $n \to \infty$.

References

[1] P. A. Bussinger, Updating a singular value decomposition, Nordisk Tidskr. Informations behandling (BIT) 10 (3) (1970) 376–397.

[2] M. Chambers, John, Regression updating, Journal of the American Statistical Association 66 (336) (1971) 744–748.

[3] K. L. Clarkson, P. Drineas, M. Magdon-Ismail, M. W. Mahoney, X. Meng, D. P. Woodruff, The fast cauchy transform and faster robust linear regression, SIAM Journal on Computing 45 (3) (2016) 763–810.

[4] J. Luts, T. Broderick, M. P. Wand, Real-time semiparametric regression, Journal of Computational and Graphical Statistics 23 (3) (2014) 589–615.

[5] R. C. Souza, S. C. Leite, C. Carlos, Borges, R. F. Neto, Online algorithm based on support vectors for orthogonal regression, Pattern Recognition Letters 34 1394–1404.

[6] D. Lin, P. Foster, Dean, H. Ungar, Lyle, Vif regression: A fast regression algorithm for large sata, Ninth IEEE International Conference on Data Mining, Miami, FL (2009).

[7] G. Fanelli, J. Gall, L. Van Gool, Real time head pose estimation with random regression forests, in: CVPR 2011, Providence, RI, 2011, pp. 617–624. doi:10.1109/CVPR.2011.5995458.

[8] L. Xiaolin, Real time regression analysis in internet of stock market cycles, Cognitive Systems Research 52 (2018) 371–379.

[9] V. Bloom, V. Argyriou, D. Makris, Linear latent low dimensional space for online early action recognition and prediction, Pattern Recognition 72 (2017) 532–547.

[10] T. Mandal, Q. M. J. Wu, Y. Yuan, Curvelet based face recognition via dimension reduction, Signal processing 89 (2009) 2345–2353.

[11] H. Kim, P. Howland, H. Park, Dimension reduction in text classification with support vector machines, Journal of Machine Learning Research 6 (2005) 37–53.

[12] S. Gunal, R. Edizkan, Subspace based feature selection for pattern recognition, Information scienes 178 (2008) 3716–3726.

[13] R. D. Cook, Regression graphics: Ideas for studying regressions through graphics, John Wiley & Sons, 1998.

[14] K.-C. Li, Sliced inverse regression for dimension reduction, Journal of the American Statistical Association 86 (414) (1991) 316–327.

[15] R. D. Cook, S. Weisberg, Comments: Sliced inverse regression for dimension reduction, Journal of the American Statistical Association 86 (414) (1991) 328–332.

[16] K.-C. Li, On principal hessian directions for data visualization and dimension reduction: Another application of stein's lemma, Journal of the American Statistical Association 87 (420) (1992) 1025–1039.

[17] K. Fukumizu, F. R. Bach, M. I. Jordan, Kernel dimension reduction in regression, The Annals of Statistics 37 (4) (2009) 1871–1905.

[18] L.-P. Zhu, L. Zhu, Z.-H. Feng, Dimension reduction in regressions through cumulative slicing estimation, Journal of the American Statistical Association 105 (492) (2010) 1455–1466.

[19] B. Li, A. Artemiou, L. Li, Principal support vector machines for linear and nonlinear sufficient dimension reduction, The Annals of Statistics 39 (6) (2011) 3182–3210.

[20] B. Li, Sufficient Dimension Reduction: Methods and Applications with R, CRC Press, 2018.

[21] Z. Cai, R. Li, L. Zhu, Online sufficient dimension reduction through

sliced inverse regression, Journal of Machine Learning Research 21 (2020) 1–25.

[22] X. Yin, B. Li, R. D. Cook., Successive direction extraction for estimating the central subspace in a multiple-index regression, Journal of Multivariate Analysis 99 (2008) 1733–1757.

[23] J. Suykens, L. Lukas, P. Van Dooren, B. De Moor, J. Vandewalle, Least squares support vector machine classifiers: a large scale algorithm, in: European Conference on Circuit Theory and Design, ECCTD, Vol. 99, Citeseer, 1999, pp. 839–842.

[24] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[25] S. J. Shin, Y. Wu, H. H. Zhang, Y. Liu, Principal weighted support vector machines for sufficient dimension reduction in binary classification, Biometrika 104 (1) (2017) 67–81.

[26] J. Zhou, L. Zhu, Principal minimax support vector machine for sufficient dimension reduction with contaminated data, Computational Statistics and Data Analysis 94 (2016) 33–48.

[27] A. Artemiou, Y. Dong, Sufficient dimension reduction via principal Lq support vector machine, Electronic Journal of Statistics 10 (1) (2016) 783–805.

[28] S. J. Shin, A. Artemiou, Penalized principal logistic regression for sparse sufficient dimension reduction, Computational Statistics & Data Analysis 111 (2017) 48–58.

[29] A. Artemiou, M. Shu, A cost based reweighted scheme of principal support vector machine, in: Topics in Nonparametric Statistics, Springer, 2014, pp. 1–12.

[30] L. Smallman, A. Artemiou, A study on imbalance support vector machine algorithms for sufficient dimension reduction, Communications in Statistics-Theory and Methods 46 (6) (2017) 2751–2763.

[31] H. Randall, A. S. Artemiou, X. Qiao, Sufficient dimension reduction based on distance-weighted discrimination, Scandinavian Journal of Statistics (2020) accepted.

[32] T. Van Gestel, J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, J. Vandewalle, Benchmarking least squares support vector machines classifiers, Machine Learning 54 (2004) 5–32.

[33] J. R. Schott, Determining the dimensionality in sliced inverse regression, Journal of the American Statistical Association 89 (425) (1994) 141–148.

[34] L. Zhu, B. Miao, H. Peng, On sliced inverse regression with large dimensional covariates, Journal of the American Statistical Association 101 (474) (2006) 630–643.

[35] W. Luo, B. Li, Combining eigenvalues and variation of eigenvectors for order determination, Biometrika 103 (4) (2016) 875–887.

[36] H. V. Henderson, S. R. Searle, On deriving the inverse of a sum of matrices, Siam Review 23 (1) (1981) 53–60.

[37] F.-T. Hadi, J. Gama, Event labeling combining ensemble detectors and background knowledge, Progress in Artificial Intelligence (2013) 1–15.

[38] F. A. Freitas, S. M. Peres, C. A. M. Lima, F. V. Barbosa, Grammatical facial expressions recognition with machine learning, 27th Florida Artificial Intelligence Research Society Conference (FLAIRS) (2014) 180–185.

[39] G. Fung, O. L. Mangasarian, Proximal support vector machine classifiers, in: Proceedings KDD-2001: Knowledge Discovery and Data Mining, Citeseer, 2001, pp. 77–86.

[40] A. W. Van der Vaart, Asymptotic statistics, Vol. 3, Cambridge university press, 1998.

[41] J. R. Magnus, H. Neudecker, The commutation matrix: some properties and applications, The Annals of Statistics (1979) 381–394.

[42] E. Bura, R. Pfeiffer, On the distribution of the left singular vectors of a random matrix and its applications, Statistics and Probability Letters 78 (15) (2008) 2275–2280.