Published in final edited form as:
Stat Interface. 2018 ; 11(3): 401–420. doi:10.4310/SII.2018.v11.n3.a1.

Double Sparsity Kernel Learning with Automatic Variable Selection and Data Extraction

Jingxiang Chen,
Department of Biostatistics, University of North Carolina at Chapel Hill

Chong Zhang,
Department of Statistics and Actuarial Science, University of Waterloo

Michael R. Kosorok, and
Department of Biostatistics, University of North Carolina at Chapel Hill

Yufeng Liu[*]
Department of Statistics and Operations Research, University of North Carolina at Chapel Hill

Abstract

Learning in the Reproducing Kernel Hilbert Space (RKHS) has been widely used in many scientific disciplines. Because a RKHS can be very flexible, it is common to impose a regularization term in the optimization to prevent overfitting. Standard RKHS learning employs the squared norm penalty of the learning function. Despite its success, many challenges remain. In particular, one cannot directly use the squared norm penalty for variable selection or data extraction. Therefore, when there exists noise predictors, or the underlying function has a sparse representation in the dual space, the performance of standard RKHS learning can be suboptimal. In the literature, work has been proposed on how to perform variable selection in RKHS learning, and a data sparsity constraint was considered for data extraction. However, how to learn in a RKHS with both variable selection and data extraction simultaneously remains unclear. In this paper, we propose a unified RKHS learning method, namely, DOuble Sparsity Kernel (DOSK) learning, to overcome this challenge. An efficient algorithm is provided to solve the corresponding optimization problem. We prove that under certain conditions, our new method can asymptotically achieve variable selection consistency. Simulated and real data results demonstrate that DOSK is highly competitive among existing approaches for RKHS learning.

Keywords and phrases

Data extraction; Kernel classification; Kernel regression; Reproducing kernel Hilbert space; Selection consistency; Variable selection

1. INTRODUCTION

Recent advances in technology have enabled scientists to collect massive datasets with high dimensions. For example, in online movie evaluation systems, the data sets can contain

[*]Corresponding author: Yufeng Liu yfliu@email.unc.edu.

rating information from millions of users on thousands of movies. Extracting knowledge from such large data sets poses unprecedented challenges to existing learning techniques. To overcome new difficulties in mining big data sets, in the last few decades, many methodologies have been proposed in the machine learning literature. In this paper, we focus on supervised learning with one response variable. In particular, the learning goal is often to train a function using a training data set, such that for new observations, one can use this function to predict the unobserved responses. See Hastie *et al.* [20] for a comprehensive review of supervised learning techniques.

For many applications in supervised learning, appropriate variable selection is very important to the prediction performance of the estimated function. In particular, for real data sets, many predictors do not contain useful information with respect to the response. Hence, these redundant predictors should be excluded when we make further prediction. For instance, in classification problems, Fan and Lv [15] showed that prediction using all variables may behave similarly to random guessing, due to the noise accumulation. How to perform variable selection has drawn much attention in the literature. Traditional methods for variable selection include forward and backward selections, among others. Recently, model fitting using sparse regularization has become very popular in the learning framework. The corresponding optimization problems of these techniques are equivalent to minimizing objective functions in the $loss + penalty$ form. The loss term measures the goodness of fit of the estimated function, and the penalty term aims to select important variables in the learning problem, which further controls the complexity of the function space to prevent overfitting.

For different learning tasks, one uses different loss functions. For example, in least squares regression, one uses the squared error loss, and in standard Support Vector Machines [SVM, 9], we use the hinge loss. For the penalty term, the choice depends on the corresponding functional space. In particular, if the response depends on the predictors linearly, linear learning should be used. Otherwise, one can employ various nonlinear learning methods such as splines [12] in regression. In this paper, we focus on learning in the Reproducing Kernel Hilbert Space [RKHS, 4, 21]. This is a very general setting, and many nonlinear learning techniques can be regarded as special cases of RKHS learning. For example, it covers penalized linear regression, additive spline models with or without interactions, and the entire family of smoothing splines. RKHS learning has been extensively used in the literature, and has achieved great successes. See, for example, Schölkopf and Smola [32], Shawe-Taylor and Cristianini [33], and Hastie *et al.* [20].

For linear learning, variable selection with sparse regularization has been extensively studied. See, for example, Tibshirani [36], Fan and Li [14], Zou and Hastie [49], Wu *et al.* [42], Zhang [45], Fan and Lv [16], and the references therein. For RKHS learning, however, the problem of variable selection has received much less attention. In the literature, Guyon *et al.* [19] suggested an extension of variable selection from linear learning to kernel learning using the Recursive Feature Elimination (RFE) approach. Lin and Zhang [25] developed the Component Selection and Smoothing (COSSO), and proposed to use the sum of component norms as the sparse penalty, instead of the squared norm penalty in standard RKHS learning. Zhang *et al.* [46] proposed a structure selection method that can automatically determine whether the signal for one predictor is linear or nonlinear. Recently, Allen [1] developed an interesting framework of variable selection in RKHS learning. In particular, Allen [1] imposed a weight on each predictor, and proposed to train the model with a sparse penalty on the weight vector. When a fitted weight is zero, the corresponding predictor is regarded as unimportant in the learning problem, and is removed from further analysis. Allen [1] provided the Kernel Iterative Feature Extraction (KNIFE) algorithm to solve the corresponding optimization.

Despite the current progress in variable selection for RKHS learning, many challenges remain. First, theoretical properties of sparse penalties in linear learning have been well studied in the literature. For example, Fan and Li [14] and Zou [48] proved the oracle property of their proposed methods, and Zhao and Yu [47] showed selection consistency for LASSO problems. In contrast, theoretical properties of existing variable selection approaches for RKHS learning are much less developed. In particular, it is desirable to explore conditions under which one can have consistency for kernel variable selection. Moreover, Allen [1] proposed to use the standard squared norm penalty on the learning function to avoid over-fitting, besides the sparse penalty on the variable weight vector. However, as Zhang *et al.* [44] pointed out, this approach uses all observations to represent the fitted function. This can lead to suboptimal prediction performance as the underlying function can be well approximated by a data sparse representation in the dual space [see 44, and Section 2.2 for more details]. Therefore, it can be beneficial to have a regularization method that can automatically select data points for RKHS learning. To circumvent this difficulty, Zhang *et al.* [44] proposed a data sparsity constraint for data extraction. However, Zhang *et al.* [44] did not consider the problem of kernel variable selection, and the data sparsity method can have suboptimal performance when there are noise covariates. Therefore, it is desirable to design a new method that can perform variable selection and data extraction simultaneously.

In this paper, we propose a new DOuble Sparsity Kernel (DOSK) learning method to fill this gap. We provide an efficient algorithm to solve the corresponding optimization problem. Through numerical examples, we show that our DOSK method can often select useful predictors accurately, and the sparsely represented functions can have very good prediction performance. Moreover, under some conditions, we prove that our DOSK method can enjoy many desirable statistical properties, including variable selection consistency.

The rest of the paper is organized as follows. In Section 2, we briefly introduce standard kernel learning methods, and discuss variable selection and data extraction for learning in a RKHS. Then, we propose our DOSK method, and develop our algorithm for the corresponding optimization problem. We establish some theoretical properties of DOSK, such as selection consistency, in Section 3. Simulated and real data examples are used to demonstrate the effectiveness of our new method in Section 4. We provide some discussions in Section 5. All technical proofs are collected in the appendix.

2. METHODOLOGY

We first give a brief review of standard kernel learning in Section 2.1. Then we propose our DOSK method in Section 2.2. We discuss how to solve the corresponding optimization problem in Section 2.3.

2.1 Standard Learning in RKHS

Suppose each observation in the training data set $(\boldsymbol{x}_i, y_i)$; $i = 1,\ldots, n$, is obtained from a fixed but unknown distribution $P(\boldsymbol{X}, Y)$, where $\boldsymbol{X} \in \mathbb{R}^p$ is a vector of predictors, and Y is the response. The learning goal is to find $f(\cdot)$ based on the training data set, so that for a new observation with only $\boldsymbol{x}$ available, the prediction of Y based on $f(\boldsymbol{x})$ can be accurate. For example, in regression, one often uses $f(\boldsymbol{x})$ to estimate the response Y, and in binary margin-based classification where $Y \in \{+1, -1\}$, one can let sign$\{f(\boldsymbol{x})\}$ be the predicted label for Y. For many learning problems, the goodness of fit of f can be measured by a loss function $L\{Y, f(\boldsymbol{X})\}$. For different learning tasks, one uses different loss functions. For instance, in standard regression problems where the goal is to estimate the conditional mean of Y with given $\boldsymbol{x}$, it is common to use the squared error loss $L\{Y, f(\boldsymbol{X})\} = \{Y - f(\boldsymbol{X})\}^2$. In classification problems, one can use the hinge loss $L\{Y, f(\boldsymbol{X})\} = \{1 - Yf(\boldsymbol{X})\}_+$ for support vector machines [SVM, 9], and the deviance loss $L\{Y, f(\boldsymbol{X})\} = \log[1 + \exp\{-Yf(\boldsymbol{X})\}]$ for logistic regression [24].

The optimization problem of a learning technique typically involves minimizing an objective function in the form of *loss* + *penalty*. In particular, the objective function can be written as

$$\min_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} L\{y_i, f(\boldsymbol{x}_i)\} + \lambda J(f), \quad (1)$$

where $\mathcal{F}$ is the function space for learning. Here the penalty term $J(f)$ regularizes $f(\cdot)$ in order to prevent overfitting, and the tuning parameter λ balances $L(\cdot,\cdot)$ and $J(f)$ with the aim to achieve a good prediction performance. The choice of the penalty term varies with the choice of $\mathcal{F}$. For example, in standard linear regression, one often assumes that the conditional mean of Y is a linear function of $\boldsymbol{x}$, and it is common to use $\mathcal{F} = \{f: f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta} + \boldsymbol{\beta}_0; \boldsymbol{\beta} \in \mathbb{R}^p, \beta_0 \in \mathbb{R}\}$. There are many popular choices for $J(f)$ in the linear learning literature. See, for example, Tibshirani [36], Fan and Li [14], Zou and Hastie [49], Zhang [45], among others. If a linear function cannot estimate the response well, one often considers a nonlinear function space $\mathcal{F}$. In this paper, we focus on learning in RKHS. For more details about RKHS, we refer the readers to [38], Shawe-Taylor and Cristianini [33], and the references therein.

For learning in a RKHS $\mathcal{H}$, it is common to use the squared norm penalty $J(f) = \|f\|_{\mathcal{H}}^2$, where $\|f\|_{\mathcal{H}}$ is the norm of f in $\mathcal{H}$. In other words, (1) can be written as

$$\min_{f \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} L\{y_i, f(\boldsymbol{x}_i)\} + \lambda \|f\|_{\mathscr{H}}^2. \quad (2)$$

Kimeldorf and Wahba [21] showed that under mild conditions on L, the estimated function $\hat{f}$ from (2) has the $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_i K(\boldsymbol{x}_i, \boldsymbol{x})$, where $K(\cdot,\cdot)$ is the kernel function associated with $\mathscr{H}$, $\boldsymbol{x}_i$'s are the observed predictor vectors in the training data set, and α_i's are the parameters to estimate. Moreover, define $\boldsymbol{K}$ to be the gram matrix with the (i, j)th element $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$; $i, j = 1,\ldots, n$, and $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_n)^T$. One can verify that the penalty $\|f\|_{\mathscr{H}}$ in (2) can be written as $\hat{\boldsymbol{\alpha}}^T \boldsymbol{K} \hat{\boldsymbol{\alpha}}$. Consequently, (2) is equivalent to the following problem,

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} L\{y_i, f(\boldsymbol{x}_i)\} + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha}.$$

In practice, however, many commonly used kernel spaces, for example the well known Gaussian RKHS, do not include offsets or intercepts [29]. This can lead to suboptimal results for some learning problems. For instance, in quantile regression, if one is interested in estimating the 100τ% quantile of the response with τ close to 0 or 1, a regression function without an intercept can have inferior performance. Therefore, in this paper, we consider learning in RKHS with intercepts. In particular, in (1), we assume that $f = \tilde{f} + b \in \mathscr{H} \oplus \mathbb{R}$, and let $J(f)$ be the squared norm of $\tilde{f}$, where $\tilde{f}$ is the projection of f onto $\mathscr{H}$. The Representer's Theorem [21] shows that under mild conditions, $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_i K(\boldsymbol{x}_i, \boldsymbol{x}) + \hat{b}$, where b is the intercept term, and $J(\hat{f}) = \hat{\boldsymbol{\alpha}}^T \boldsymbol{K} \hat{\boldsymbol{\alpha}}$. Hence, for standard RKHS learning, the optimization problem (2) with an intercept in f can be written as

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L\{y_i, \sum_{j=1}^{n} \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b\} + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha}. \quad (3)$$

2.2 Double Sparsity Kernel Learning

Despite the success of standard kernel learning methods, many challenges remain. First, the standard squared norm penalty cannot perform automatic variable selection. When the underlying signal depends only on a small fraction of the predictors (note that the corresponding relationship can be nonlinear), learning with all predictors can lead to overfitting, and consequently unsatisfactory results. Second, the standard kernel learning method may lead to suboptimal performance when all the data observations are used without extraction. In this section, we first discuss some literatures on variable selection in Section 2.2.1, and data extraction in Section 2.2.2. In Section 2.2.3, we present the proposed method which can perform variable selection and data extraction simulatenously.

2.2.1 Kernel Variable Selection—In the literature, Zhang *et al.* [46] and Allen [1], among others, proposed different methods for variable section in RKHS learning. In particular, to perform variable selection in kernel learning, Allen [1] proposed the idea of variable weighted kernel learning as follows. For a weight vector $\mathbf{w} \in \mathbb{R}^p$ and any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^p$, we define the variable weighted kernel function $K_{\mathbf{w}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = K(\mathbf{w} \odot \boldsymbol{x}_1, \mathbf{w} \odot \boldsymbol{x}_2)$, where $\mathbf{w} \odot \boldsymbol{x}$ denotes the element-wise product of vectors. In other words, the jth element of $\mathbf{w}$, w_j, represents the weight of the jth predictor of $\boldsymbol{X}$ in the kernel function. For any positive definite kernel function K, one can verify by Mercer's Theorem that the newly defined variable weighted kernel $K_{\mathbf{w}}(\cdot,\cdot)$ naturally introduces a RKHS over the domain of $\boldsymbol{X}$. For identifiability, we impose the constraint that $w_j \in [0, 1]$ for all j. In the variable weighted kernel function, if $w_j = 0$, then the jth predictor of $\boldsymbol{X}$ has no impact on f or the prediction. Therefore, one can impose an L_1 type penalty on the vector $\mathbf{w}$ to achieve variable selection in RKHS learning. In particular, Allen [1] proposed KNIFE for learning in a RKHS with variable selection, with the following optimization

$$\min_{\alpha, b, w} \left[\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \sum_{j=1}^{n} K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j + b\} + \lambda_1 \left\|\mathbf{w}\right\|_1 + \lambda_2 \boldsymbol{\alpha}^T K_{\mathbf{w}} \boldsymbol{\alpha}\right], \quad (4)$$

where λ_1 and λ_2 are tuning parameters, and $\mathbf{w} \in [0, 1]^p$. To better illustrate the variable weighted kernel function, we consider several commonly used RKHSs as examples. Define x_{ik} to be the kth element of $\boldsymbol{x}_i$. The linear variable weighted kernel is $K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{p} w_k^2 x_{ik} x_{jk}$, the polynomial variable weighted kernel is $K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \{c + \sum_{k=1}^{p} w_k^2 (x_{ik} x_{jk})\}^d$, with $c \in \mathbb{R}$ and $d \in \mathbb{N}$, the Gaussian variable weighted kernel is $K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\{-\gamma \sum_{k=1}^{p} (w_k x_{ik} - w_k x_{jk})^2\}$ with $\boldsymbol{\gamma} \in \mathbb{R}^+$, and the Laplacian variable weighted kernel is $K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \sum_{k=1}^{p} |w_k x_{ik} - w_k x_{jk}|)$ with $\boldsymbol{\gamma} \in \mathbb{R}^+$.

2.2.2 Kernel Data Extraction—Generally speaking, data extraction can have two different goals. One is to improve the model performance. The other is to downsize the data volume and reduce computational burden when the dataset is massive. In this paper, the major goal of data extraction is to improve model performance. In particular, this paper concentrates on data extraction under the kernel learning framework. The motivation of such a specific data extraction is based on the following observation. The kernel evaluation representation of the regression function is similar to the knot structure in smoothing splines, in the sense that each observation in the training data can be regarded as a "knot" in a multidimensional space. In particular, when we restrict the RKHS regression to the smoothing splines, the kernel evaluation representation is equivalent to the piecewise nonlinear function representation. With the regular squared norm penalty, the resulting estimator involves all the kernel evaluation functions. In some problems, having too many knots may yield suboptimal performance. Hence it is desirable to have a regularization method that can select the kernel evaluation functions automatically.

Recently, Zhang *et al.* [44] showed that in some cases, using the squared norm penalty $\left\|\cdot\right\|_{\mathscr{H}}^{2}$ for learning in RKHS can lead to suboptimal results. In particular, in a given learning problem, let $f^{*}(\boldsymbol{x})$ be the minimizer of the conditional expected loss. In other words, $f^{*}(\boldsymbol{x}) = E[L\{Y, f(\boldsymbol{X})\} \mid \boldsymbol{X} = \boldsymbol{x}]$ for any $\boldsymbol{x}$ (e.g., $f^{*}(\boldsymbol{x})$ is the conditional mean of $Y(\boldsymbol{x})$ in standard regression). Zhang *et al.* [44] observed that if $f^{*}(\boldsymbol{x})$ can be well approximated by a function with a sparse representation in the RKHS (in other words, $f^{*}(\cdot)$ can be well approximated by $\sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \cdot) + b$ for only some nonzero α_i), learning with the squared norm penalty can have the potential danger of overfitting. To overcome this difficulty, one can apply an L_1 penalty on the vector $\boldsymbol{\alpha}$ for data selection of the estimated function. For RKHS learning problems, Zhang *et al.* [44] proposed the data sparsity constraint with the following optimization

$$\min_{\alpha, b}\left[\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \sum_{j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j + b\} + \lambda\left\|\boldsymbol{\alpha}\right\|_1\right], \quad (5)$$

where $K(\cdot,\cdot)$ is the standard kernel function and $\left\|\boldsymbol{\alpha}\right\|_1 = \sum_{i=1}^{n} |\alpha_i|$. Using the quantile regression as an example, Zhang *et al.* [44] showed that, in certain cases, learning with the data sparsity constraint in (5) can improve the prediction performance.

Besides the work mentioned above, there are some other works in the literature on data extraction to cope with big data. Specifically, methods have been proposed to implement the data extraction idea under the linear model framework to save the computational cost [27, 39]. The main data extraction idea used in these papers is to carefully select a subset of data for modeling without losing much efficiency. These methods can boost the efficiency when compared to certain traditional sampling techniques (see [27] for an example under linear regression, and [39] under logistic regression). The final model is built using only a subset of the data. Thus, these methods are different from our proposed method below, which is trained using the whole dataset.

2.2.3 Kernel Double Sparsity—

Although data extraction was used in Zhang *et al.* [44], their method does not consider variable selection. Hence, when there are noise predictors in $\boldsymbol{x}$, the proposed approach can be suboptimal. We would like to point out that modeling data sparsity can be challenging for high-dimension data, especially when there are many noisy variables. One reason is that noisy variables can mislead the importance of each observation in the modeling process. Thus, it is desirable to develop a tool that can handle these two sparsities simultaneously. To our knowledge, not much work has been done on simultaneous data extraction and variable selection in the literature. To fill this gap, we propose our DOuble Sparsity Kernel learning (DOSK) method as follows

$$\min_{\boldsymbol{\alpha}, b, w} [\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \sum_{j=1}^{n} K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j + b\} + \lambda_1 \left\|\boldsymbol{\alpha}\right\|_1 + \lambda_2 \left\|\mathbf{w}\right\|_1 + \lambda_3 \boldsymbol{\alpha}^T K_{\mathbf{w}} \alpha], \quad (6)$$

with $\lambda_i \geq 0$; $i = 1, 2, 3$, $K_{\mathbf{w}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = K(\mathbf{w} \odot \boldsymbol{x}_1, \mathbf{w} \odot \boldsymbol{x}_2)$ as defined earlier with $\mathbf{w} \in [0, 1]^p$.

The framework of our DOSK (6) is very general, in the sense that it includes many existing approaches as special cases. In particular, when $\lambda_1 = \lambda_2 = 0$, (6) reduces to the standard squared norm penalized kernel learning (3). When $\lambda_1 = 0$, (6) reduces to the KNIFE approach (4) proposed by Allen [1]. If $\lambda_2 = \lambda_3 = 0$, (6) becomes the data sparsity learning (5) in Zhang *et al.* [44]. Because DOSK is a general framework of RKHS learning, one can use various approaches to solve the optimization problem (6), based on the choice of the loss function $L(\cdot,\cdot)$, $\mathbf{w}$ and λ_l; $l = 1, 2, 3$. For example, in linear kernel learning with $\lambda_2 \neq 0$, one can verify that (6) is a biconvex problem with respect to $(\boldsymbol{\alpha}^T, b)^T$ and $\mathbf{w}$, and can be solved by the alternate convex search algorithm [18]. For more general DOSK problems, we propose a unified algorithm to solve (6) in Section 2.3.

Note that our method makes use of the whole dataset to build a model which only involves a small subset of data (those have non-zero α_j). With many α's being zero, our proposed model can save some computational burden during the prediction stage since the kernel evaluation for a new point is only needed for those selected training points with nonzero α's.

Although we impose multiple penalties in (6), our DOSK method can circumvent the difficulty of over-penalization by choosing $(\lambda_1, \lambda_2, \lambda_3)$ carefully. In particular, in Section 3, we show that if the tuning parameters are chosen appropriately, our DOSK method can enjoy desirable theoretical properties.

2.3 Computational Algorithm for DOSK

The major difficulty of solving the optimization (6) is that even L is convex, the composite loss function $L\{y, \sum_{j=1}^{n} K_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{x}_j)\alpha_j + b\}$ may not be convex with respect to $(\mathbf{w}^T, \boldsymbol{\alpha}^T, b)^T$. Consequently, many existing algorithms for convex optimizations [10] cannot be used directly. On the other hand, one can verify that if the loss function L is convex, the optimization (6) is convex respect to $(\boldsymbol{\alpha}^T, b)^T$ for a fixed $\mathbf{w}$. Hence, a natural way to circumvent the difficulty of non-convex optimization is to update $\mathbf{w}$ and $(\boldsymbol{\alpha}^T, b)^T$ recursively. This, however, cannot be done directly, as for a general kernel function $K(\cdot,\cdot)$, $L\{y, \sum_{j=1}^{n} K_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{x}_j)\alpha_j + b\}$ is not biconvex with respect to $\mathbf{w}$ and $(\boldsymbol{\alpha}^T, b)^T$. One way to tackle this problem is that for fixed $(\boldsymbol{\alpha}^T, b)^T$, we can find a linear approximation of the variable weighted kernel function $K_{\mathbf{w}}$ in a small neighbourhood of $(\mathbf{w}^T, \boldsymbol{\alpha}^T, b)^T$ [1]. Thus, to update $\mathbf{w}$, one can employ the linear approximation of $K_{\mathbf{w}}$ to make the corresponding objective function convex. Note that in the literature, the idea of local linear approximation has been widely used to solve optimizations for many learning problems. See, for example, An and Tao [3], Zou and Li [50], Lee *et al.* [22], among others.

To introduce our algorithm for DOSK, we need some further notation. Let the objective function in (6) be $\phi(\boldsymbol{\alpha}, b, \mathbf{w})$. Define an $n \times p$ matrix $A(\mathbf{w})$, whose ith row is $\sum_{j=1}^{n} \alpha_j \nabla_{\mathbf{w}} K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j)^T$, and an $n \times n$ matrix $B(\mathbf{w})$ with the (i, j)th element $B(i, j) = K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \nabla K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j)^T \mathbf{w}$. Here $\nabla_{\mathbf{w}} K_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the gradient vector of $K_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j)$ with respect to $\mathbf{w}$. By Taylor's expansion, one can verify that for $\mathbf{w}_1$ and $\mathbf{w}_2$, we have

$$K_{\mathbf{w}}\boldsymbol{\alpha} = A(\mathbf{w}_2)\mathbf{w}_1 + B(\mathbf{w}_2)\boldsymbol{\alpha} + o(\|\mathbf{w}_1 - \mathbf{w}_2\|_2). \quad (7)$$

Define $\boldsymbol{c}_{\mathbf{w}_2}(\mathbf{w}_1)\mathbf{w}_1 = A(\mathbf{w}_2)\mathbf{w}_1 + B(\mathbf{w}_2)\boldsymbol{\alpha}$, which is a linear function of $\mathbf{w}_1$. When $\mathbf{w}_1$ and $\mathbf{w}_2$ are close, we can use $\boldsymbol{c}$ as the local linear approximation of $K_{\mathbf{w}}\boldsymbol{\alpha}$ in our DOSK optimization algorithm. In particular, we outline the general algorithm to solve (6) in Algorithm 1 below.

In the $\boldsymbol{\alpha}$ and b steps in Algorithm 1, the corresponding objective functions are convex, therefore after updating the parameters, the value of ϕ decreases. On the other hand, in the $\mathbf{w}$ step, we replace the original objective function ϕ by its local linear approximation, and solve a quadratic programming problem. Denote the solution to this quadratic programming problem by $\mathbf{w}^{(QP)}$. In Algorithm 1, the updated $\mathbf{w}^{(t)} = \mathbf{w}^{(QP)}$ can have some distance from $\mathbf{w}^{(t-1)}$, hence the original ϕ function is not guaranteed to decrease. One possible way to overcome this difficulty is that in the $\mathbf{w}$ step, instead of having $\mathbf{w}^{(t)} = \mathbf{w}^{(QP)}$, we can treat $\mathbf{w}^{(QP)} - \mathbf{w}^{(t-1)}$ as a direction in which ϕ tends to decrease, and determine the appropriate step size by conducting a line search. In particular, we present the revised algorithm in Algorithm 2.

In Algorithm 2, one can verify that after updating the parameters, the ϕ function value would not increase. This helps to guarantee that we can obtain a stationary point of the objective function using Algorithm 2. In particular, we have the following theorem.

Theorem 1—Suppose that the loss function L in (6) is a convex and continuously differentiable function, and the variable weighted kernel $K_{\mathbf{w}}$ is a convex or concave and continuously differentiable function of $\mathbf{w}$. Then the solution from Algorithm 2 is a stationary point of the objective function.

Remark 1—Theorem 1 is valid for many loss functions, e.g., the squared error loss in standard regression, and the deviance loss in logistic regression. For many other loss functions that are not differentiable, such as the hinge loss in SVM, or the check loss function in quantile regression, one can consider an alternative continuous approximation to the loss function. For example, Wang *et al.* [40] proposed the hybrid huberized hinge loss for SVM. One can verify that the hybrid huberized loss meets the condition in Theorem 1, and the corresponding solution is a stationary point. Moreover, for many commonly used kernel functions, the assumptions on $K_{\mathbf{w}}$ in Theorem 1 are satisfied. For example, one can verify that the variable weighted kernel introduced by the Laplacian RKHS, or by the linear kernel when all elements in $\boldsymbol{x}$ are non-negative, is convex with respect to $\mathbf{w}$.

Algorithm 1

```
1. Initialize $\mathbf{w}^{(0)}$, $\boldsymbol{\alpha}^{(0)}$ and $b^{(0)}$ with $w_j \in [0, 1]$ for $1 \le j \le p$.
2. The $\boldsymbol{\alpha}$ step: fix $\mathbf{w}^{(t-1)}$ and $b^{(t-1)}$, and find $\boldsymbol{\alpha}^{(t)} = \mathrm{argmin}_{\boldsymbol{\alpha}}\, \phi(\boldsymbol{\alpha}, b^{(t-1)}, \mathbf{w}^{(t-1)})$. The optimization problem is convex,
   and independent of the $\lambda_2\|\mathbf{w}\|_1$. term in (6).
3. The $b$ step: fix $\mathbf{w}^{(t-1)}$ and $\boldsymbol{\alpha}^{(t)}$, and find $b^{(t)} = \mathrm{argmin}_b \frac{1}{n}\sum_{i=1}^n L\{y_i, \sum_{j=1}^n K_{\mathbf{w}^{(t-1)}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j^{(t)} + b\}$.
   This is a convex optimization with one parameter, and can be solved by standard methods.
4. The $\mathbf{w}$ step: fix $b^{(t)}$ and $\boldsymbol{\alpha}^{(t)}$, and define $\boldsymbol{c}_{\mathbf{w}^{(t-1)}}(\mathbf{w}) = A(\mathbf{w}^{(t-1)})\mathbf{w} + B(\mathbf{w}^{(t-1)})\boldsymbol{\alpha}^{(t)}$. Let $\{\boldsymbol{c}_{\mathbf{w}^{(t-1)}}(\mathbf{w})\}_i$ be
   the $i$th element of $\boldsymbol{c}_{\mathbf{w}^{(t-1)}}(\mathbf{w})$. Under the constraint $\mathbf{w}^{(t)} \in [0, 1]^p$, solve the following standard quadratic
   programming problem
   $\mathbf{w}^{(t)} = \mathrm{argmin}_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^n L[y_i, \{\boldsymbol{c}_{\mathbf{w}^{(t-1)}}(\mathbf{w})\}_i + b^{(t)}] + \lambda_2\|\mathbf{w}\|_1 + \lambda_3 \mathbf{w}^T A(\mathbf{w}^{(t-1)})\boldsymbol{\alpha}^{(t)}$.
5. Repeat steps 2–4 until convergence.
```

Algorithm 2

```
1. Initialize $\mathbf{w}^{(0)}$, $\boldsymbol{\alpha}^{(0)}$ and $b^{(0)}$ with $w_j \in [0, 1]$ for $1 \le j \le p$.
2. The $\boldsymbol{\alpha}$ step: fix $\mathbf{w}^{(t-1)}$ and $b^{(t-1)}$, and find $\alpha^{(t)} = \mathrm{argmin}_{\boldsymbol{\alpha}}\, \phi(\boldsymbol{\alpha}, b^{(t-1)}, \mathbf{w}^{(t-1)})$. The optimization problem is convex, and
   independent of the $\lambda_2\|\mathbf{w}\|_1$ term in (6).
3. The $b$ step: fix $\mathbf{w}^{(t-1)}$ and $\boldsymbol{\alpha}^{(t)}$, and find $b^{(t)} = \mathrm{argmin}_b \sum_{i=1}^n L\{y_i, \sum_{j=1}^n K_{\mathbf{w}^{(t-1)}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j^{(t)} + b\}$. This
   is a convex optimization with one parameter, and can be solved by standard methods.
4. The $\mathbf{w}$ step: fix $b^{(t)}$ and $\boldsymbol{\alpha}^{(t)}$, and define $\mathbf{w}^{(\mathrm{temp})} = \mathbf{w}^{(t-1)}$.
   (a) Define $\boldsymbol{c}_{\mathbf{w}^{(\mathrm{temp})}}(\mathbf{w}) = A(\mathbf{w}^{(\mathrm{temp})})\mathbf{w} + B(\mathbf{w}^{(\mathrm{temp})})\boldsymbol{\alpha}^{(t)}$.
   Let $\{\boldsymbol{c}_{\mathbf{w}^{(\mathrm{temp})}}(\mathbf{w})\}_i$ be the $i$th element of $\boldsymbol{c}_{\mathbf{w}^{(\mathrm{Temp})}}(\mathbf{w})$. Under the constraint $\mathbf{w} \in [0, 1]^p$, find
   $\mathbf{w}^{(QP)} = \mathrm{argmin}_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^n L[y_i, \{\boldsymbol{c}_{\mathbf{w}^{(\mathrm{temp})}}(\mathbf{w})\}_i + b^{(t)}] + \lambda_2\|\mathbf{w}\|_1 + \lambda_3 \mathbf{w}^T A(\mathbf{w}^{(\mathrm{temp})})\boldsymbol{\alpha}^{(t)}$
   (b) Define $\Delta\mathbf{w} = \mathbf{w}^{(QP)} - \mathbf{w}^{(\mathrm{temp})}$. Find the best step size $s$ by
   $s = \mathrm{argmin}_{u \ge 0}\, \phi(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \mathbf{w}^{(\mathrm{temp})} + u\Delta\mathbf{w})$.
   (c) Set $\mathbf{w}^{(\mathrm{temp})} = \mathbf{w}^{(\mathrm{temp})} + s\Delta\mathbf{w}$.
   (d) Repeat steps (a)–(c) until convergence, and set $\mathbf{w}^{(t)} = \mathbf{w}^{(\mathrm{temp})}$.
5. Repeat steps 2-4 until convergence.
```

Remark 2—Algorithm 2 replaces the quadratic programming step in Algorithm 1 by the descent direction and line search method. This approach is guaranteed to decrease the objective function value at each iteration step, at the cost of a more complex computation. On the other hand, our numerical experience shows that Algorithm 1 almost always decreases the objective for commonly used kernels and loss functions. Therefore, we use Algorithm 1 in the numerical examples, whereas in each step we check if the objective function decreases. If not, we then employ the line search approach as in Algorithm 2 instead.

Remark 3—Since the objective function can be non-convex, it is possible that the numerical solution is just a stationary point, not the global minimum. To increase the chance of finding the optimal solution, we suggest to use multiple different starting points, compare the corresponding results, and choose the fitted model with the smallest objective function value.

3. STATISTICAL LEARNING THEORY

In this section, we explore the theoretical properties of the proposed DOSK method. In particular, we first study the convergence rate of the excess risk for various learning problems under certain conditions, and then show that DOSK can enjoy selection consistency for high dimensional learning problems. Moreover, we show that the expected loss using the estimated function $\hat{f}$, $E[L\{y, \hat{f}(\boldsymbol{X})\}]$ can be well approximated by the empirical loss on the training data, in the sense that the corresponding difference converges to zero with a fast convergence rate.

Several standard assumptions on the data, kernel functions, and loss functions are required for the main theorem to hold. Details of these assumptions are left in Appendix 1. Now we present our main theorem, which studies the convergence rate of $\hat{f}$ to f_0, variable selection consistency, and the risk bound. Denote $a \vee \mathrm{b} = \max(a, b)$ for $a, b \in \mathbb{R}$.

Theorem 2

Suppose Assumptions 1-7 (see Appendix 1 for details) hold, and $\log(p)/\sqrt{n} \to 0$ as $n \to \infty$. If we choose $\lambda_1 = O\{\log(n)^{-1}\}$, $\lambda_2 = O[\{\log(p) \vee \log(n)\}/\sqrt{n}]$, and $\lambda_3 = o(\lambda_1)$ in (6), we have that the corresponding global solution $(\hat{\mathbf{w}}^T, \hat{\boldsymbol{\alpha}}^T, \hat{b})^T$ to (6) satisfies that

- Parametric Rate: $\left\|\hat{f} - f_0\right\|_2 = O_P\{\log(n)/\sqrt{n}\}$, where $\hat{f}(\boldsymbol{x}) = \sum_{j=1}^n \hat{\alpha}_j K_{\hat{\mathbf{w}}}(\boldsymbol{x}, \boldsymbol{x}_j) + \hat{b}$;
- Selection Consistency: with probability tending to 1 as $n \to \infty$, $sign(\hat{w}_j) = sign(w_j^*)$ for $j = 1,\ldots, p$, where w_j^* is the jth element of $\mathbf{w}^*$;
- Risk Bound: $|E[L\{Y, \hat{f}(\boldsymbol{X})\}] - n^{-1}\sum_{i=1}^n [L\{y_i, \hat{f}(\boldsymbol{x}_i)\}]| = O_P[\{\log(p) \vee \log(n)\}/\sqrt{n}]$, where $\hat{f}(\boldsymbol{x}) = \sum_{j=1}^n \hat{\alpha}_j K_{\hat{\mathbf{w}}}(\boldsymbol{x}, \boldsymbol{x}_j) + \hat{b}$.

Theorem 2 contains three parts. The first part suggests that $\hat{f}$ converges to f_0 at a rate very close to the “parametric rate” $O_P(n^{-1/2})$. Comparing Theorem 2 with the theoretical results in Zhang *et al.* [44], one can see that the multiple penalties in (6) do not affect the performance of $\hat{f}$, as long as the corresponding λ’s are appropriately selected. This helps to justify that our DOSK method can avoid the issue of over-penalization by carefully choosing the tuning parameters.

The second part of Theorem 2 shows that our DOSK method can enjoy the desirable asymptotic selection consistency at the global solution. In other words, if the sample size is large, one can often correctly identify the important and unimportant variables in the learning problem. This can help researchers to obtain a better understanding of the relationship between predictors and the response, and provide a more interpretable model for future prediction.

The third part of Theorem 2 studies the prediction performance of the obtained $\hat{f}$. In particular, since one uses the loss function L to measure the goodness of fit of $\hat{f}$, it is desirable to obtain a bound for the expected loss $E[L\left\{Y, \hat{f}(X)\right\}]$. For example, in regression problems, $E[L\left\{Y, \hat{f}(X)\right\}]$ indicates the average prediction error using $\hat{f}$. In margin-based classification where the loss function L dominates the $0 - 1$ loss function (which is further equivalent to the prediction error rate), $E[L\{Y, \hat{f}(\boldsymbol{X})\}]$ can be regarded as an upper bound of the future misclassification rate. The third part of Theorem 2 shows that under the assumptions specified above, the empirical measurement $n^{-1}\sum_{i=1}^{n}[L\{y_i, \hat{f}(\boldsymbol{x}_i)\}]$ converges to its expectation $E[L\{Y, \hat{f}(\boldsymbol{X})\}]$ at the rate $O_P[\{\log(p) \vee \log(n)\}/\sqrt{n}]$. This empirical loss can provide valuable information on the prediction performance of $\hat{f}$.

As a remark, we would like to point out that our theorem can be generalized to the case of local solutions, provided that similar conditions about the underlying relationship and loss function (those in Assumptions 4-6 of Appendix 1) are met. For example, the convexity of local solutions can be stated in an analogous manner as in Assumption 5, and the corresponding signal strength can be measured by the partial derivatives as in Assumption 6.

4. NUMERICAL ANALYSIS

In this section, we use regression and classification as examples of learning techniques, and explore the numerical performance of our proposed DOSK method using simulated and real data sets. In Section 4.1, we study the empirical prediction behavior of DOSK using synthetic data sets, and in Section 4.2, we examine the performance of DOSK in real data applications. We compare our method with some existing approaches in the literature. In particular, for regression problems, we compare our DOSK method with the standard linear ridge regression, LASSO, standard L_2 kernel learning as in (3), COSSO and KNIFE. Moreover, we implement the Sure Independence Screening (SIS) and Recursive Feature Elimination (RFE) methods with L_2 kernel learning. Notice here the generalization of SIS from linear learning to kernel learning is analogous to the approach discussed in Guyon *et al.* [19]. We employ the squared error loss function for all regression techniques. For classification methods, we use the SVM hinge loss for DOSK, and compare with the standard kernel SVM, kernel SIS SVM, kernel RFE SVM and KNIFE SVM.

In all numerical examples, we select the tuning parameters as follows. For our DOSK method, because there are three tuning parameters λ_1-λ_3 and potential kernel parameters (such as the γ parameter in the Gaussian kernel), we fix $\lambda_3 = 0.5$, and let other parameters be selected from a set of candidates. In particular, we let λ_1 vary in $\{0, 0.25, 0.5\}$, and let λ_2 vary in $\{2^i; i = -3, -2, \ldots, 2, 3\}$. As we will show in Section 4.1 that the selection of λ_3, the tuning parameter for the quadratic kernel regularization term, does not appear to play an essential role in maximizing the prediction accuracy of DOSK as long as its value is taken within a certain range. For the kernel parameters, because we use the Gaussian and

Laplacian kernels (whose kernel functions are discussed in Section 2.2) in our analysis, we let the parameter γ vary in {0.1, 0.2, …, 0.9, 1}, a candidate set whose range always covers $1/2\hat{\sigma}^2$ where $\hat{\sigma}$ is the median of the Euclidean distances between each pair of the observations. In our experience, this tuning procedure works reasonably well for the numerical examples in this paper. For real applications, one can perform finer tuning procedures using a larger candidate set of tuning parameters. For other existing approaches except SIS and RFE, the tuning parameters are chosen in an analogous manner. The best set of tuning parameters that minimizes the prediction error in five fold cross validations on the training data set is then selected, and we report the corresponding prediction errors on a separate testing data set. Here the prediction error for regression examples is measured by the Mean Prediction Error [MPE, 20], $\frac{1}{n}\sum_{i=1}^{n}\{\hat{f}(\boldsymbol{x}_i) - y_i\}^2$. The error measure for classification problems is the misclassification rate (MCR), $\frac{1}{n}\sum_{i=1}^{n} I[y_i \neq \text{sign}\{\hat{f}(\boldsymbol{x}_i)\}]$, where $I(\cdot)$ is the indicator function.

4.1 Simulated Examples

In this section, we conduct four simulated examples to demonstrate the performance of our DOSK method. The first two examples are regression problems, and the last two are classification problems. In each example, we let the responses depend only on several predictors, and we add noise covariates in the date sets. We denote by p_0 the number of noise predictors. To assess various methods, we repeat each example 50 times and report the average prediction errors on the training and testing data sets. Furthermore, for all the methods that perform variable selection, we report the True Positive (TP) rates and False Negative (FN) rates of predictors to compare the corresponding performance on variable selection.

Regression Example 1—For this example, the response depends only on one predictor. In particular, we have $y_i = 10\sin(x_{i1})I(0 < x_{i1} < 2\pi) + \varepsilon_i$, where x_{i1} is the first predictor of the ith observation. Here x_{ij} follows a uniform distribution within $[-2\pi, 4\pi]$ for $j = 1, \cdots, 1 + p_0$, and the error term ε is generated from the standard normal distribution. In this example, we let $p_0 = 2$ and $p_0 = 8$, and choose the size of the training data set to be 50 and 100. The size of the testing set is 10 times larger than that of the training set. We use the Laplacian kernel in this example.

The numerical results for Regression Example 1 are reported in Table 1. One can see that the ridge regression and LASSO perform poorly using linear learning, as the underlying function f_0 is highly nonlinear. Note that the standard kernel learning method with the L_2 penalty has very small prediction error rates on the training data sets. This shows that the corresponding models can fit the training observations very well. However, the errors on the testing data set are very large. This suggests that without appropriate variable selection, the performance of standard kernel learning can be greatly undermined by overfitting. Moreover, the SIS and RFE approaches can also have overfitting issues, which are partly due to their large FN rates. Compared to these methods, KNIFE and our DOSK work competitively. Note that the prediction error of COSSO is also good with a large sample size ($n = 100$). However, the corresponding variation is significantly larger than that of KNIFE or

DOSK. This suggests that decomposing the nonlinear function into a sum of orthogonal components can be instable for some kernels. Furthermore, as the underlying function can be well approximated by functions that have sparse presentations, our DOSK method works better than KNIFE. This is similar to the findings in Zhang *et al.* [44]. To demonstrate the effect of data selection, in Figure 1, we plot the fitted regression function $\hat{f}$ from our DOSK method in a typical replicate, and the underlying function f_0 as a comparison. Moreover, we plot all the training observations, and highlight the selected ones, whose corresponding $\hat{\alpha}_j$'s are non-zero. One can see that because we are using the Laplacian kernel which has a singularity at 0 and smooth elsewhere, the data sparsity penalty tends to choose the observations that are closer to the "sharp turns" of f_0 for representation. This helps to build a model that is smooth when the curvature of f_0 is small, thus prevents overfitting from using all observations in the kernel function representation.

Regression Example 2—In this example, the response Y depends on 4 predictors. In particular,

$$y_i = 10\sum_{j=1}^{4}\exp(-x_{ij}^2) + \varepsilon_i,$$

where the error term follows standard normal distribution, and x_{ij} follows a uniform distribution in [−6, 6] for $j = 1,\dots, 4$. The number of noise covariates and sizes of the training and testing data sets are the same as in Regression Example 1. We use the Gaussian kernel in this example. The prediction performance and variable selection results for Regression Example 2 are reported in Table 2, and one can draw similar conclusions as in Regression Example 1.

As to the tuning parameter selection, we fix $\lambda_3 = 0.5$ to save the computational time. Note that there are three tuning parameters λ_1, λ_2, λ_3 in (6) for the proposed DOSK. Based on our numerical experiments, the performance of DOSK is not sensitive to the choice of λ_3, the tuning parameter for the quadratic penalty term. For illustration, we draw four contour plots of the mean prediction errors for Regression Example 2 when $p_0 = 8$ in Figure 2. In particular, we set λ_3 as {0, 0.25, 0.5, 1} respectively for each plot and calculate the optimal prediction error among all combinations of λ_1 and λ_2 with τ being $1/2\hat{\sigma}^2$, where $\hat{\sigma}$ is the median of the pairwise Euclidean distances for the simulated samples. From the results, one can observe that the best (λ_1, λ_2) combination is almost always near the coordinate (0.5, 0.5) for all these λ_3 values. Because we fix λ_3 to be 0.5 in DOSK, KNIFE and DOSK have the identical number of parameters to be tuned in practice. This choice appears to work well in all the experiments we tried. As a consequence, these two methods need similar time in finding the best λ's.

Classification Example 1—In this example, we consider a binary classification problem, where the prior probabilities pr($Y = +1$) = pr($Y = -1$) = 1/2. The posterior probabilities pr($Y = +1 \mid \boldsymbol{X} = \boldsymbol{x}$) depend on two predictors. In particular, the distribution of $x_{\cdot 1}$ and $x_{\cdot 2}$ for the first class is $N\{(0, 0)^T, I_2\}$, where $x_{\cdot j}$ represents the jth predictor, and I_2 is the 2 × 2 identity matrix. For the second class, the distribution of $x_{\cdot 1}$ and $x_{\cdot 2}$ is proportional to the restricted joint normal distribution $N\{(0,0)^T, I_2\} | 9 < (x_{\cdot 1}^2 + x_{\cdot 2}^2) < 16$. To illustrate the marginal distribution of $x_{\cdot 1}$ and $x_{\cdot 2}$, we plot the first two covariates for a typical sample in Figure 3. In this example, we let $p_0 = 0, 4, 8$, and add independent noise variables following $N(0, 0.1)$ in the data set. The number of observations in the training data set is 200, and in the testing 2000. Note that a similar example was previously used in Hastie *et al.* [20]. The Gaussian kernel is used.

The simulation results are reported in Table 3. One can see that when there are no noise predictors, all the methods can provide similar classification performance, with our DOSK method being slightly better. When the number of noise covariates increases, the prediction performance of L_2 kernel SVM, SIS and RFE deteriorates. On the other hand, the KNIFE method and our DOSK work competitively. Moreover, in this example, the classification boundary $(x_{\cdot 1}^2 + x_{\cdot 2}^2 = 9)$ is relatively simple (see Figure 3 for an illustration). Hence, functions with sparse representations in the dual space can separate the two classes well. Consequently, our DOSK method works better than the KNIFE approach. In terms of variable selection, KNIFE and DOSK both perform very well, and are significantly better than the other methods.

Classification Example 2—We consider a similar example as in Classification Example 1. In particular, we let the classification signal depend on 4 predictors. For the first class, the distribution of $x_{\cdot 1}$ to $x_{\cdot 4}$ is $N\{(0, 0,0,0)^T, I_4\}$. The corresponding distribution of the second class is proportional to $N\{(0,0, 0, 0)^T, I_4\} \mid 9 < \sum_{j=1}^{4} x_{\cdot j}^2 < 16$. We let $p_0 = 0,4, 8$ in this example. The classification results are reported in Table 4, and one can draw a similar conclusion as that of Classification Example 1.

Next, we would like to discuss the computational complexity and the compare the runtime of DOSK with other methods. According to Algorithm 1, the linear approximation in the **w** step simplifies the original non-convex optimization problem into a quadratic programming program with linear constraints. Similar to KNIFE, the order of the computational cost per iteration of DOSK should be equivalent to that of the kernel regression using the quadratic loss. Similarly, the computational cost of DOSK would perform the same as the standard SVM using the hinge loss. In practice, the actual runtime of DOSK can depend on the number of iterations used before convergence. Therefore, a proper starting point $\mathbf{w}^{(0)}$ can save the computational time significantly.

In order to assess the actual runtime performance of DOSK, we use the same four simulated examples above and fix the noise dimension as $p_0 = 8$. We also include two real data applications: the CPUs and Ecoli datasets. To have a general idea of the runtime in finding the best tuning parameters, we record the average time (in seconds) that each method takes for each tuning parameter value combination. For regression examples, the linear ridge and LASSO are implemented by the R package glmnet. The L_2 Kernel method is also implemented by glmnet but includes some extra kernel matrix calculation. SIS, RFE and COSSO are implemented by the corresponding R packages SIS, caret, and COSSO respectively. KNIFE and DOSK are implemented using R entirely. For classification examples, L_2 Kernel, SIS and RFE are all primarily fitted by the R package e1071 with some extra matrix calculation. KNIFE and DOSK are implemented by a R wrapper of the Matlab package CVX to conduct the two quadratic programmings in each iteration. As to the stopping criterion, we always use the default settings when there is a corresponding R package. For KNIFE and DOSK, we set the maximum iteration number to be 300 and the stopping rule as when the L_2-norm of the objective function change is less than 0.001. The average runtime of all the methods for each tuning parameter set is listed in Table 6.

Based on the results in Table 6, it is not surprising to see that the linear ridge and LASSO take much less time than all the other methods since the core of the package glmnet contains a set of Fortran subroutines, which is much faster than the corresponding R code. The L_2 kernel method, SIS, and RFE are slower not only because they have higher complexity but also due to the extra matrix calculation in R. Similar arguments can also be made for these methods in classification, which are implemented by the libsvm C++ code. The results of COSSO heavily depend on the selection of the knots number. As to KNIFE and DOSK, they perform almost equivalently in terms of computational time under both the regression and classification examples. This comparison result is consistent with our previous discussion on the comparable computational complexity.

To assess how the computational time increases as the sample size and variable dimension become larger, we extend the classification Example 2 by ranging the sample size $n \in \{200, 400, 800, 1000\}$ and variable dimension $p \in \{10, 50, 200, 500, 1000, 2000\}$. The average run time results are summarized in Table 5. Although the computational time increases when n and p increase as expected, the proposed DOSK can still be implemented relatively efficiently when n and p are not too large. For very large datasets such as the case of $n = 1600$, and $p = 2000$, it can still take a long time to train the DOSK model. However, the computing efficiency can be further improved by replacing some core parts of our R code with C++ implementation.

4.2 Real Data Applications

In this section, we apply our DOSK method to four real data sets and explore the corresponding prediction performance. In particular, the first two real data sets are about regression problems, and the last two are for classification applications. In addition, we apply the method to a microarray dataset to assess its performance under high-dimensional settings.

4.2.1 Regression Examples: Ozone and CPUs Data—We consider the ozone pollution data in Los Angels [11], and the Central Processing Units (CPUs) performance prediction data [13] as our regression applications. The ozone data set includes 330 observations, and each observation contains the daily measurement of ozone reading (the response) in 1976. Furthermore, 8 predictors that have potential impact on the ozone readings are also available, such as temperature, inversion base height, etc. The CPUs performance data set can be found in the UCI machine learning Repository [5]. The corresponding response variable contains 209 different CPUs' published relative performance on a benchmark mix. The data set also includes 7 predictors, such as the cache size, minimum main memory, and cycle time, among others, which may be useful in predicting a computer's performance.

Before the analysis, we standardize the data sets, such that the range of each predictor is in [0, 1]. Because we do not have separate training and testing data sets, for each replicate we randomly split the data into two equal parts, and use one for training and the other for testing. We choose the best tuning parameters in a similar way as in the simulated examples, by 5-fold cross validations on the training sets. The Laplacian kernel is used for both examples. We compare our DOSK method with LASSO, standard L_2 kernel learning, SIS regression with L_2 kernel learning, RFE with L_2 kernel learning, COSSO and KNIFE.

The average prediction errors in 50 replicates are summarized in Table 7. For the ozone data, the DOSK method performs better than the existing approaches in terms of the average prediction error. For the CPUs data, one can see that the standard L_2 kernel learning may have a potential overfitting issue, which is similar to the simulation results. In terms of variable selection, we report the predictors that are selected more than 45 times out of the 50 replicates. In the CPUs data set, each method selects a small subset of the predictors in the models. In particular, SIS tends to fit a model with minimum main memory and maximum main memory. The RFE and LASSO approaches select maximum main memory, cache size, and maximum number of channels as the important variables. For COSSO, KNIFE and our DOSK methods, the maximum main memory and cache size are the selected variables. This is consistent with the insights given in Ein-Dor and Feldmesser [13]. In other words, to specify the performance of a computer, only a few components are necessary. Interestingly, LASSO works slightly better than SIS, RFE, or the COSSO methods in prediction. One possible explanation is that the response is not highly nonlinear in this example, and kernel learning methods without stable variable selection can lead to suboptimal results. In contrast, KNIFE performs competitively, while our DOSK enjoys the best accuracy. This suggests that variable weighted kernel learning can provide stable selection performance for real applications.

4.2.2 Classification Examples: Wisconsin Breast Cancer Data and Ecoli Data—For classification applications, we use the diagnostic Wisconsin breast cancer data set [35] and the Ecoli data set [31] for illustrations. These two data sets can also be found in the UCI machine learning Repository. The breast cancer data set has diagnosis results (malignant or benign) for 569 patients. The data also contain 30 predictors computed from a digitized image of a fine needle aspirate of a breast bass, such as mean distances from center to points on the perimeter, standard deviation of gray-scale values, etc. The Ecoli data set has 8 categories of proteins, and we use two categories, namely, cytoplasmic proteins and inner membrane proteins without signal sequence, for a demonstration in our analysis. The total number of samples of these two classes is 220, and the data set includes 7 predictors, such as different measures of signal protein sequence recognition, consensus sequence score, amino acid content in certain outer proteins, among others.

We use DOSK with the SVM hinge loss, and compare our method with standard L_2 kernel SVM, SIS, RFE and KNIFE. Similar to the regression examples, we standardize all the predictors before our analysis. Furthermore, we randomly split the data sets into two equal parts, and use one for training (5 fold cross validations to select the best tuning parameters) and the other for testing. We report the average prediction error rates for various methods in Table 8, and one can see that the standard kernel SVM with the L_2 norm penalty can have a potential overfitting issue on these two data sets, which is consistent with the simulation results. Compared with other methods, our DOSK performs competitively.

4.2.3 A High-dimensional Classification Example: Microarray Data—We use a microarray example [2] to show the performance of DOSK under a high dimensional setting. This dataset was also studied in [1]. The microarray data contain 62 tissue samples, with 22 normal and 40 tumor samples, as well as 2000 genes that have the highest variances across the samples. We applied the standard linear SVM with the $L_2 -$ norm penalty, logistic regression with the L_1−norm penalty, KNIFE, and DOSK for comparison. Both KNIFE and DOSK are fit with the linear kernel.

To assess the impact of variable dimension on the selected methods, we pick six subsets of the 2000 genes and let the number of genes vary within {10, 50, 200, 500, 1000, 2000}. To obtain these subsets, we sort the genes using recursive feature elimination (RFE), and choose the corresponding number of the top genes at each time. To evaluate the model performance, we conduct a 5-fold crossvalidation with 100 replications for each subset, and report the validation misclassification rates across all methods. The results are presented in Table 9.

From Table 9, DOSK and KNIFE perform similarly when the number of genes is small (such as $p = 10$), and both methods keep all the variables in the final model. In this case, DOSK tends to retain all the observations in the model possibly due to the lack of information to distinguish the important ones. When the subset has an intermediate number of genes (such as $p = 200$), the advantage of DOSK becomes more clear as the data set contains more information for the data sparsity to further improve the model performance. The advantage of double sparsity, when compared to KNIFE, indicates that the variable importance can be measured more precisely when noisy data points are removed or significantly underweighted. When the number of genes becomes very large (such as $p = 1000$), KNIFE and DOSK perform similarly again. In contrast to KNIFE and DOSK, the results of standard SVM with the L_2−norm penalty do not improve as much when the number of genes increases. This can indicate the importance of variable selection. The logistic regression with the L_1−norm penalty can produce competitive results when the number of genes is relatively large.

5. DISCUSSION

In this paper, we propose a new DOSK method in kernel learning that can perform variable selection and data extraction simultaneously. We show that under certain conditions, the new DOSK method can achieve selection consistency, and the estimated function can converge to the underlying function with a fast rate. We also develop an efficient algorithm to solve the corresponding optimization, which is guaranteed to converge to a local optimum. Numerical results show that our DOSK method is highly competitive among existing approaches.

As a remark, our DOSK method can be generalized to alleviate the computational burden for applications with massive data sets. Without loss of generality, take regression as an example. Suppose one needs to estimate a nonlinear underlying function, and the data set contains many observations and predictors. To perform kernel regression with such big data can be computationally inefficient. One way to circumvent this difficulty is to split the predictors into several parts or divide the observations into several subsets, learn on each part individually, and then combine the results. In particular, each time one can perform our DOSK method on one piece of the data set. Because our DOSK method can have double sparsity in predictors and dual variables, for each sub-regression, it is possible to find a sparsely represented function that only involves a subset of observations and predictors. Then we can combine the selected observations and predictors to train for a global estimator. One can see that this approach can greatly reduce the computational time for problems with massive data sets. Further research can be pursued in this direction.

Acknowledgments

The authors would like to thank the editor Professor Heping Zhang, the associate editor, and referees for their helpful comments and suggestions. The authors were supported in part by NIH grants R01GM126550, P01CA142538, and NSF grants DMS-1407241, IIS-1632951.

APPENDIX 1: TECHNICAL ASSUMPTIONS FOR THEOREM 2

To state our theory, we first introduce some technical assumptions, and provide detailed discussions on why these conditions are needed. We also discuss some cases where these conditions are met. We would like to point out that most of the assumptions in this paper are mild and reasonable, which can be satisfied or checked for various real applications.

To begin with, we need to present some further notation. Let $\mathbf{w}^* = (\mathbf{w}_{(1)}^T, \mathbf{w}_{(0)}^T)^T$ be the underlying variable weight vector, where elements in $\mathbf{w}_{(1)}$ are non-zero, and elements in $\mathbf{w}_{(0)}$ are zero. In other words, the predictors in $\boldsymbol{x}$ that correspond to $\mathbf{w}_{(0)}$ are noise covariates. Accordingly, one can define $\boldsymbol{x}^* = (\boldsymbol{x}_{(1)}^T, \boldsymbol{x}_{(0)}^T)^T$, such that predictors in $\boldsymbol{x}_{(1)}$ contain useful information for the learning problem. In this paper, we focus on the case that the number of useful predictors is finite (i.e., $|\mathbf{w}_{(1)}| < \infty$). Furthermore, with a little abuse of notation, we let $\|f\|_{\mathscr{H}} = \|\tilde{f}\|_{\mathscr{H}}$, where $\tilde{f}$ is the projection of f onto $\mathscr{H}$.

We impose our first assumption on the distribution of $\boldsymbol{X}$ and $\boldsymbol{X}_{(1)}$, where $\boldsymbol{X}$ and $\boldsymbol{X}_{(1)}$ correspond to the p dimension random vector and the vector containing important variables.

Assumption 1

Every element in $\boldsymbol{X}$ ranges in [0, 1]. Furthermore, the distribution of $\boldsymbol{X}_{(1)}$ is absolutely continuous with respect to the Lebesgue measure, where the corresponding Radon-Nikodym derivative is bounded away from 0.

In Assumption 1, we restrict our consideration on $\boldsymbol{X} \in [0, 1]^p$. One can verify that our theory can be naturally generalized to the case where the elements in $\boldsymbol{X}$ are uniformly bounded. We defer the discussion on the second part of Assumption 1 until after Assumption 4.

In the next assumption, we impose some constraints on the kernel function $K(\cdot,\cdot)$.

Assumption 2

The kernel function $K(\cdot,\cdot)$ is separable and sup $K(\cdot,\cdot) < \infty$. Furthermore, the kernel function $K_{\mathbf{w}^*}(\boldsymbol{x}, \cdot)$ is Lipshcitz with respect to $\boldsymbol{x}_{(1)}$, i.e. the useful variable vector, in terms of the L_2 norm.

The first part of Assumption 2 is very mild, and has been frequently used in the literature. See, for example, Steinwart and Scovel [34], [8], Zhang *et al.* [44], among others. It suggests that the corresponding RKHS $\mathscr{H}$ is not too complex, in the sense that its diameter would not be infinity. The second part is used to ensure that the best learning function using n observations can converge to the underlying function in a fast rate. See the proof of Lemma 2 for more details. This assumption is valid for many commonly used kernel functions such as the Gaussian kernel and the polynomial kernel.

In Assumption 3, we assume that L can be treated as a univariate function. This is a very mild condition, and is valid for many learning problems. For example, in standard least squares regression, we have $L(u) = u^2$ where $u = (f - y)$, and in logistic regression, $L(u) = \log\{1 + \exp(-u)\}$ where $u = yf$ and $y \in \{+1, -1\}$.

Assumption 3

The loss function $L(u)$ has a second order derivative with $0 < L''(u) < \infty$ for every u.

Assumption 3 is needed to ensure that the expected loss function is strictly convex around the underlying optimal solution. Moreover, the second order differentiability helps to control the convergence rate of the estimated function $\hat{f}$ to the best function. See the discussion of Assumption 5 for more details.

Next, we consider assumptions on the function $f(\boldsymbol{x})$. Recall that the learning goal is to obtain $\hat{f}(\boldsymbol{x})$ from the training data set for good prediction performance. Therefore, we consider the "best" function f_0, in the sense that its corresponding expected loss $E[L\{Y, f_0(\boldsymbol{X})\}]$ is the minimum among all possible $E[L\{Y, f(X)\}]$. Consequently, f_0 can have the best prediction performance under mild conditions. For instance, in classification, f_0 can achieve the minimal classification error rate, given that the loss function L is Fisher consistent [26]. We will prove that under certain conditions on f_0, the estimated function $\hat{f}$ would converge to f_0 with a desirable convergence rate.

Assumption 4

The underlying function f_0 has a sparse representation in the RKHS. In particular, there exist $\gamma_1, \ldots, \gamma_m, \mathbf{z}_1, \ldots, \mathbf{z}_m$, and b_0 such that $f_0(\mathbf{x}) = \sum_{j=1}^{m} \gamma_j K_{\mathbf{w}^*}(z_j, \mathbf{x}) + b_0$. Here m is a fixed integer, $\gamma_j \neq 0$, and $\mathbf{z}_j \in [0, 1]^p$ for $j = 1, \ldots, m$.

As a remark, we note that some RKHSs are very rich, in the sense that many functions can be well approximated by $f \in \mathscr{H}$. For example, Steinwart and Scovel [34] proved that all step functions can be approximated by f in the Gaussian RKHS arbitrarily well under mild conditions, and this result can be generalized to the case of continuous functions. However, if f_0 does not have a sparse representation in the RKHS, the function in $\mathscr{H}$ that approximates f_0 well may have an infinite norm. When $\hat{f}$ approaches f_0 as $n \to \infty$, $\|\hat{f}\|_{\mathscr{H}}$ would be unbounded. Consequently, the variation of $\hat{f}$ due to the randomness of the sample can be very large. In the literature, Bartlett *et al.* [6], among others, pointed out that large variation of $\hat{f}$ can lead to suboptimal prediction performance. Assumption 4 ensures that the underlying function f_0 has a finite norm in the RKHS. In the proof of Theorem 2, we show that with an appropriate λ_1, the data selection can provide a sparsely represented function $\hat{f}$ whose norm can be bounded away from infinity. This is crucial to prove the convergence of $\hat{f}$ to f_0, which further leads to the selection consistency of our DOSK method.

The next assumption ensures that in the updating scheme, $\hat{f}$ would converge to the global solution, once we are at a point that is close enough. To state this assumption, we first introduce some further notation. Define $\|\cdot\|_{*,2}$ to be the restricted L_2 norm with respect to the partition of $\mathbf{w}$. In particular, $\|\mathbf{x} - \mathbf{z}\|_{*,2} = \|\mathbf{x}_{(1)} - \mathbf{z}_{(1)}\|_2$. For any $n \gg m$, we define $(\boldsymbol{\alpha}_n^*, b_n^*)$ as follows. Notice that the empirical loss function value does not change if we switch the order of the pairs $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$ for $i \neq j$. Hence, without loss of generality, we can assume that $\mathbf{x}_j$ is the observation that is closest to $\mathbf{z}_j$ in terms of the $\|\cdot\|_{*,2}$ norm among the training data set $\{(\mathbf{x}_i, y_i);\ i = 1, \ldots, n\}$, for $j = 1, \ldots, m$. When $n \gg m$, we can assume that each $\mathbf{x}_j$ is distinct (in other words, $\mathbf{x}_j$ would not be closest to $\mathbf{z}_u$ and $\mathbf{z}_v$ simultaneously, compared to other observations). Next, define $(\boldsymbol{\alpha}_n^*, b_n^*)$ such that $\boldsymbol{\alpha}_n^* = (\gamma_1, \ldots, \gamma_m, 0, \ldots, 0)^T$ with length n, $b_n^* = b_0$, and let $f_{\boldsymbol{\alpha}_n^*, b_n^*}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_j^* K_{\mathbf{w}^*}(\mathbf{x}_i, \mathbf{x}) + b_n^*$. The definition of $(\boldsymbol{\alpha}_n^*, b_n^*)$ helps to show that the approximation error of the DOSK method under Assumption 4 converges to 0 very quickly. See the proof of Lemma 4 in the appendix for more discussions.

Before stating Assumption 5, we would like to discuss the second part of Assumption 1, which ensures that with large enough n, the underlying function can be well approximated by the sparsely represented function $f_{\boldsymbol{\alpha}_n^*, b_n^*}(\mathbf{x})$ from our training data. In particular, Assumption 1 guarantees that as $n \to \infty$, $f_{\boldsymbol{\alpha}_n^*, b_n^*}(\mathbf{x})$ can approach $f_0(\mathbf{x})$ with a rate very close to $O_P(n^{-1})$ in terms of the $\|\cdot\|_2$ norm. See Lemma 2 and the corresponding proof for more discussions.

Assumption 5

For any p and $n \gg m$, there exists a neighborhood $\mathcal{N}$ of $\left((\mathbf{w}^*)^T, (\boldsymbol{\alpha}_n^*)^T, b_n^*\right)^T$, such that in $\mathcal{N}$, the expected loss function $E\left[\sum_{i=1}^{n} L\{Y_i, f(\boldsymbol{X}_i)\}\right]$ is strictly convex with respect to $(\mathbf{w}^T, \boldsymbol{\alpha}^T, b)^T$.

Assumption 5 is necessary for our theory, because if the loss function is not strictly convex, a small perturbation in the training data set can lead to a significant change of $\hat{f}$. See, for example, the discussion on a similar issue for quantile regression using the check loss function in Li and Zhu [23]. Consequently, the convergence rate of $\hat{f}$ to f_0 can be difficult to obtain. To our knowledge, there has been no theoretical result on selection consistency that does not rely on the assumption or fact of local convexity. Notice that Assumption 3 is important to the validity of Assumption 5, because if L is not strictly convex, it is likely that the expected loss function is not convex even if the kernel function is locally convex. For instance, if we use the hinge loss $L(u) = [1-u]_+$ which is piecewise linear, Assumption 5 cannot be satisfied.

Next, we impose constraints on the signal strength in the learning problem. For variables weighted learning, the jth predictor provides useful information if and only if the weight w_j is positive. Variable selection consistency means that sign($\hat{w}_j$) = sign(w_j) for all j with a high probability, where sign(0) = 0. The next assumption is an important part of sufficient conditions for variable selection consistency.

Assumption 6

For any w_j in $\mathbf{w}_{(1)}$, $\frac{\partial E[L\{Y, f_0(\boldsymbol{X})\}]}{\partial w_j}\Big|_{w_j = 0, w_i = w_i^*, i \neq j} < 0$, and for any w_j in $\mathbf{w}_{(0)}$, $\frac{\partial E[L\{Y, f_0(\boldsymbol{X})\}]}{\partial w_j}\Big|_{w_j = 0, w_i = w_i^*, i \neq j} \geq 0$. Here w_i^* is the ith element of $\mathbf{w}^*$.

In Assumption 6, we measure the signal strength of w_j by its partial derivative with respect to the expected loss function evaluated at $\mathbf{w}^*$ (except the jth weight is at zero). In the literature, there are many existing assumptions on the signal strength that are (essentially) similar to Assumption 6. For example, one can verify that for regular linear regression with the squared error loss, Assumption 6 reduces to that the non-zero coefficients are bounded away from zero. This is analogous to the assumptions considered in Fan and Peng [17] and Fan and Lv [16], among others. Furthermore, we require the partial derivatives with respect to the noise covariates are non-negative.

In the last assumption, we focus on regression problems, where $Y = f_0(\boldsymbol{X}) + \varepsilon(\boldsymbol{X})$ with $(\boldsymbol{X})$ being the random error term. Notice that we include both the homoscedastic and the heteroscedastic cases here, as ε can have different distributions for different $\boldsymbol{X}$. If the distribution of ε has a very heavy tail, there is a large probability that we observe a y_i that is very far away from $f_0(\boldsymbol{x}_i)$. This outlier can lead to severely biased estimation $\hat{f}$. Assumption 7 aims to control the probability of an extreme y_i, which can help to bound the magnitude of the estimated $\hat{b}$. Recall that if a random variable U is sub-Gaussian with parameter s, then $\mathrm{pr}(|U| > u) \le 2\exp(-u^2/s)$ for large enough u.

Assumption 7

In a regression problem, the error term $\varepsilon(\boldsymbol{X})$ follows a sub-Gaussian distribution with a universal parameter $s < \infty$ for any $\boldsymbol{X}$.

Assumption 7 is very general, as many distributions are sub-Gaussian. For example, in linear regression, we often assume that $\varepsilon \sim N(0, \sigma^2)$ with a finite σ. This is a homoscedastic case of Assumption 7, and normal random variables are known to be sub-Gaussian. Furthermore, all random variables with bounded ranges are sub-Gaussian, and distributions with small kurtosis are sub-Gaussian.

APPENDIX 2: TECHNICAL PROOFS OF LEMMAS AND THEOREMS

Proof of Theorem 1

Because the objective function ϕ is lower bounded by zero, to prove convergence, it suffices to prove that for each step of updating, the objective function value is non-increasing. To this end, we will show that ϕ is non-increasing for Steps 2–4 in Algorithm 2. First, notice that for fixed w, the corresponding objective functions in the α step and the b step are convex. Hence, ϕ is non-increasing for Steps 2 and 3. We will focus on Step 4 next.

Without loss of generality, suppose that $\nabla_{\mathbf{w}}\phi(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \mathbf{w}^{(t-1)}) \neq \mathbf{0}$ (otherwise, the algorithm has already converged). We will prove that the directional derivative along $\Delta\mathbf{w}$ is negative, with which one can verify that after Step 4, the objective function value would decrease. To this end, observe that Step 4(a) can be regarded as to minimize $\psi(\mathbf{w}) = h\{g(\mathbf{w})\}$, where $h(\cdot)$ is a convex and continuously differentiable function and $g(\cdot)$ is a convex or concave and continuously differentiable function of $\mathbf{w}$. Since both h and g are continuously differentiable, they are locally Lipshcitz continuous, and so is ψ. Furthermore, because h and g are convex or concave, there exists an open neighborhood of $\mathbf{w}^{(t-1)}$, $\mathcal{N}(\mathbf{w}^{(t-1)})$, in which h and g are monotonic [7]. Therefore, in $\mathcal{N}(\mathbf{w}^{(t-1)})$, $\psi(\cdot)$ is monotonic.

Next, we prove that along the direction defined by $\Delta\mathbf{w}$, $\psi()$ is monotonically deceasing in $\mathcal{N}(\mathbf{w}^{(t-1)})$. To this end, first notice that Step 4 computes a descent direction of $\tilde{\psi}_{\mathbf{w}^{(t-1)}}(\mathbf{w}) = h\{g(\mathbf{w}^{(t-1)}) + \nabla g(\mathbf{w}^{(t-1)})^T(\mathbf{w} - \mathbf{w}^{(t-1)})\}$. Because the objective function of $\mathbf{w}^{(QP)}$ is quadratic, thus strictly convex, $\tilde{\psi}_{\mathbf{w}^{(t-1)}}(\mathbf{w})$ is strictly decreasing along $\Delta\mathbf{w}$ within $\mathcal{N}(\mathbf{w}^{(t-1)})$. Next, by similar arguments as in the proof of Proposition 1 in Allen [1], one can verify that $\psi(\cdot)$ is monotonically deceasing along $\Delta\mathbf{w}$ within $\mathcal{N}(\mathbf{w}^{(t-1)})$, and this completes the proof. ■

Proof of Theorem 2, Part I (Parametric Rate)

Before we present our proof, we first give some lemmas.

Lemma 1

Suppose Assumptions 1–7 are valid. With λ_1, λ_2 and λ_3 as in Theorem 2, we have that $\|\hat{\boldsymbol{\alpha}}\|_1 = O_P\{\log(n)\}$ and $\|\hat{b}\|_1 = O_P\{\log(n)\}$.

Proof of Lemma 1

With $\boldsymbol{\alpha} = \mathbf{0}$ and $b = 0$, we have $\phi(\mathbf{0}, 0, \mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, 0) \to E\{L(Y, 0)\}$ as $n \to \infty$, which is a constant. On the other hand, $\hat{\boldsymbol{\alpha}}$ and $\hat{b}$ are (part of) the solution to the objective function in (6). Hence,

$$\lambda_1\|\hat{\boldsymbol{\alpha}}\|_1 \le \frac{1}{n}\sum_{i=1}^{n} L\{y_i, \sum_{i=1}^{n} K_{\hat{\mathbf{w}}}(x_i, x_j)\hat{\alpha}_j + \hat{b}\} + \lambda_1\|\hat{\boldsymbol{\alpha}}\|_1 + \lambda_2\|\hat{\mathbf{w}}\|_1 + \lambda_3\hat{\boldsymbol{\alpha}}^T K_{\hat{\mathbf{w}}}\hat{\boldsymbol{\alpha}} \le \phi(\mathbf{0}, 0, \mathbf{w}).$$

Consequently, we have $\|\hat{\boldsymbol{\alpha}}\|_1 = O_P\{\log(n)\}$. For $|\hat{b}|$, in regression, because the fitted function $\hat{f}$ cannot be uniformly larger or smaller than the observed responses, we have that $|\hat{b}|$ is at most $O_P(\|\hat{\boldsymbol{\alpha}}\|_1)$, which is $O_P\{\log(n)\}$ (notice that we have assumed that the error term in regression are bounded for now). For classification problems, similar arguments hold ($\hat{f}$ cannot be uniformly positive or negative, otherwise the classification problem is of less interest), and $|\hat{b}| = O_P\{\log(n)\}$. This completes the proof. □

Lemma 2

Suppose Assumptions 1–7 are valid. We have that $\left\|f_{\boldsymbol{\alpha}_n^*, b_n^*} - f_0\right\|_2 = O_P\{\log(n)/n\}$.

Proof of Lemma 2

Notice that γ_j's are constants, and the kernel function $K_{\mathbf{w}^*}$ is Lipshcitz by Assumption 2.

Hence, we have

$$|f_{\boldsymbol{\alpha}_n^*, b_n^*}(\cdot) - f_0(\cdot)| = |\sum_{j=1}^{m}\gamma_j\{K_{\mathbf{w}^*}(\boldsymbol{x}_j, \cdot) - K_{\mathbf{w}^*}(z_j, \cdot)\}| = O_P(\max_j\|\boldsymbol{x}_j - z_j\|_2),$$

and the goal is to prove that $\|\mathbf{x}_j - \mathbf{z}_j\|_2 = O_P\{\log(n)/n\}$ for all j. To this end, note that $\mathrm{pr}(\|\mathbf{x}_j - \mathbf{z}_j\|_2 > d) = (1 - P_d)^n$, where d is a small positive number, and $P_d = \mathrm{pr}(\|z - z_j\|_2 \le d) = \int_{\|z - z_j\|_2 \le d} dP$. Using Assumption 1, one can verify that we can choose $d = 2\log(n)/n$, such that $\mathrm{pr}(\|\mathbf{x}_j - \mathbf{z}_j\|_2 > d) = O_P(n^{-2})$. By the Borel–Cantelli Lemma, we have $\|\mathbf{x}_j - \mathbf{z}_j\|_2 = O_P\{\log(n)/n\}$ holds. This completes the proof. □

The next lemma generalizes some theoretical results from the margin-based classifier literature to broader ranges of learning problems. In particular, in Zhang and Liu [43], it was shown that the convergence rate of excess risks for margin-based classifiers is related to the convergence rate of the estimated learning function. In Lemma 3, we extend the discussion to more general situations, in which one uses differentiable loss functions to measure the goodness of fit of $\hat{f}$.

Lemma 3

Suppose Assumptions 1–7 are valid. Moreover, consider a loss function $\ell\{u(f, y)\}$ that is second order differentiable with respect to u, where $u(f, y)$ is a function of the response y and the learning function f. Assume that u has the second order derivative with respect to f, and the two second order derivatives are both bounded. Then we have that, if the function f^* minimizes $E(\ell)$,

$$|E[\ell\{u(Y,f)\}] - E[\ell\{u(Y,f^*)\}]| = O\{(\|f - f^*\|_2)^2\},$$

and if f^* is not the minimizer of $E(\ell)$,

$$|E[\ell\{u(Y,f)\}] - E[\ell\{u(Y,f^*)\}]| = O\{(\|f - f^*\|_2)\}.$$

Proof of Lemma 3

This proof is analogous to that of Theorems 5 and 6 in Zhang and Liu [43]. Hence, for brevity, we only list the key steps. The first step is to introduce the idea of Bregman divergence. In particular, for a convex differentiable function $g(\cdot)$, its Bregman divergence d_g is defined as $d_g(f_1, f_2) = g(f_2) - g(f_1) - g'(f_1)(f_1 - f_2)$. Then, one can prove that the conditional excess risk $E[\ell\{u(Y, f)\}] - E[\ell\{u(Y, f^*)\}]\,|_{X=x}$ equals to the Bregman divergence $d_\ell\{f^*(x), f(x)\}$. See the proof of Theorem 4 in Zhang and Liu [43 for more details. Combining this result with Assumption 3, we can show, in a similar manner as in the proof of Theorems 5 and 6 in Zhang and Liu [43], that the claim of Lemma 3 holds. □

We are ready to prove Theorem 2. The proof follows a similar line as that of Theorem 1 in Zhang *et al.* [44]. Therefore, we only list out the key steps here. The first step is to decompose the excess risk into two parts, the estimation error and the approximation error. In particular, let $f_{\boldsymbol{\lambda}}$ be the best prediction function with respect to the penalized loss function for fixed $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$, i.e., $f_\lambda = \operatorname{arginf}_f[E\{L(Y, f)\} + \lambda_1\|\boldsymbol{\alpha}\|_1 + \lambda_2\|\mathbf{w}\|_1 + \lambda_3\boldsymbol{\alpha}^TK\mathbf{w}\boldsymbol{\alpha}]$. The estimation error is defined as $E\{L(Y,\hat{f})\} - E\{L(Y,f_\lambda)\}$, and the approximation error is defined to be $E\{L\}(Y, f_{\boldsymbol{\lambda}})\} - E\{L(Y, f_0)\}$.

Next, consider the function space $\hat{f}$ lies in, and denote it by $\mathscr{F}_\lambda$. Define $g_f(\cdot) = s^{-1}\{L(\cdot, f) - L(\cdot, f_{\boldsymbol{\lambda}})\}$, where s is chosen such that the L_2 diameter of $\mathscr{G} = \{g_f : f \in \mathscr{F}_\lambda\}$ is 1. Using Lemma 1, one can verify that $s = O_P\{\log(n)\}$. From Lemma 2 in Zhang *et al.* [44], we have that the upper bound of the L_2 entropy number of $\mathscr{G}$, $\log[N\{\eta, \mathscr{G}, L_2(T_X)\}]$, is of the order $O_P(\eta^{-2})$ [see, for example, 37, for introduction of the entropy numbers]. Here T_X is the empirical measure of a training set, and the L_2 norm is $\|f\|_{L_2(T_X)} = \{n^{-1}\sum_{i=1}^{n}|f(\boldsymbol{x}_i, y_i)|^2\}^{1/2}$. Consequently, one can obtain that the estimation error is of the order $O_P\{\log(n)/\sqrt{n}\}$, by similar arguments as in the proof of Theorem 1 in Zhang *et al.* [44]. Therefore, by Lemma 3, $\left\|\hat{f} - f_\lambda\right\|_2 = O_P\{\log(n)/\sqrt{n}\}$.

On the other hand, to derive the bound for the approximation error, one can use Assumption 1, Lemmas 2 and 3. In particular, we have that $E[L\{Y, f_\lambda(\boldsymbol{X})\}] - E[L\{Y, f_0(\boldsymbol{X})\}]$ converges at a rate faster than that of $\left\|f_{\boldsymbol{\alpha}_n^*, b_n^*} - f_0\right\|_2^2$, which is $O_P[\{\log(n)/n\}^2] = O_P\{\log^2(n)/(n^2)\}$. Thus, by Lemma 3, we have that $\|f_\lambda - f_0\|_2 = O_P\{\log(n)/n\}$. Consequently, one has that $\left\|\hat{f} - f_0\right\|_2 \le \left(\left\|\hat{f} - f_\lambda\right\|_2 + \left\|f_\lambda - f_0\right\|_2\right) = O_P\{\log(n)/\sqrt{n}\}$. This completes the proof.■

Proof of Theorem 2, Part II (Selection Consistency)

In the proof, we first assume that for regression problems, the distribution of the error has a bounded range. We will consider the more general case of sub-Gaussian distributions later.

The next lemma, Lemma 4, is an important intermediate step to the proof of Selection Consistency. With Lemma 4, we can prove that the difference between $\hat{f}$ and the best function f_0, in terms of the difference in their expected partial derivatives with respect to w_j, is converging at the rate at least $O_P\{\log(n)/\sqrt{n}\}$. This further leads to the fact that the proposed λ_2 in Theorem 2 can correctly select the important variables $\boldsymbol{x}_{(1)}$ and discard the noise $\boldsymbol{x}_{(0)}$. Consequently, we can have the desired selection consistency for our DOSK method.

Lemma 4

Suppose Assumptions 1-7 are valid. With λ_1, λ_2 and λ_3 *as in Theorem 2, we have that for any* $j = 1,. \ldots, p$,

$$\left|\left[\frac{\partial E[L\{Y, \hat{f}(\boldsymbol{X})\}] - \partial E[L\{Y, f_0(\boldsymbol{X})\}]}{\partial w_j}\right]\Bigg|_{w_j = 0, w_j = w_j^*, i \ne j}\right| = O_P\{\frac{\log(n)}{\sqrt{n}}\}.$$

Proof of Lemma 4

The proof follows a similar line as that of Theorem 2 and Lemma 3. □

We are ready to present the proof to Selection Consistency.

First, we prove that for any j,

$$\left| \left[\frac{\partial [\frac{1}{n} \sum_{i=1}^{n} L\{y_i, \hat{f}(\boldsymbol{x}_i)\}]}{\partial w_j} - \frac{\partial E[L\{Y, f_0(\boldsymbol{X})\}]}{\partial w_j} \right] \Bigg|_{w_j = 0, w_j = w_j^*, i \neq j} \right| = O_P\left\{ \frac{\log(n) \vee \log(p)}{\sqrt{n}} \right\}. \tag{8}$$

To this end, observe that

$$\begin{aligned} &\left| \left[\frac{\partial [\frac{1}{n} \sum_{i=1}^{n} L\{y_i, \hat{f}(\boldsymbol{x}_i)\}]}{\partial w_j} - \frac{\partial E[L\{Y, f_0(\boldsymbol{X})\}]}{\partial w_j} \right] \right| \\ &\leq \left| \left[\frac{\partial [\frac{1}{n} \sum_{i=1}^{n} L\{y_i, \hat{f}(\boldsymbol{x}_i)\}]}{\partial w_j} - \frac{\partial E[L\{Y, \hat{f}(\boldsymbol{X})\}]}{\partial w_j} \right] \right| + \left| \left[\frac{\partial E[L\{Y, \hat{f}(\boldsymbol{X})\}]}{\partial w_j} - \frac{\partial E[L\{Y, f_0(\boldsymbol{X})\}]}{\partial w_j} \right] \right|. \end{aligned} \tag{9}$$

As Lemma 4 bounds the second term on the RHS of (9), we proceed to show that the first term converges at the rate $O_P[\{\log(n) \vee \log(p)\}/\sqrt{n}]$. To this end, we need to introduce the Rademacher complexity [30]. In particular, let σ_i; $i = 1,\ldots, n$, be *i.i.d.* random variables, each taking the value 1 with probability 1/2, and −1 with probability 1/2. Let the set of training observations $(\boldsymbol{x}_i, y_i)$; $i = 1, \ldots, n$, which are *i.i.d.* from P, be denoted by S. Define the function class $\mathscr{H}_n(\lambda)$ as $\mathscr{H}_n(\lambda) = \{\hat{f} : \hat{f} = \text{argmin}_{\boldsymbol{\alpha}, b, w} \phi(\lambda)\}$, where $\boldsymbol{\phi}(\boldsymbol{\lambda})$ is the objective function in (6). With S fixed, we define the empirical Rademacher complexity of the function class $\mathscr{H}_n(\lambda)$ as

$$\hat{R}_n\{\mathscr{H}_n(\lambda)\} = E_{\boldsymbol{\sigma}}\{ \sup_{f \in \mathscr{H}_n(\lambda)} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(\boldsymbol{x}_i)\},$$

where $E_{\boldsymbol{\sigma}}$ represents the expectation with respect to $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$. Furthermore, denote the Rademacher complexity of $\mathscr{H}_n(\lambda)$ by

$$R_n\{\mathscr{H}_n(\lambda)\} = E_S \hat{R}_n\{\mathscr{H}_n(\lambda)\},$$

where E_S is the expectation with respect to the distribution of the sample S.

To bound the first term on the RHS of (9), we have the following lemma.

Lemma 5

Suppose Assumptions 1-7 are valid. With λ_1, λ_2 and λ_3 as in Theorem 2, we have that, for any $j = 1, \ldots, p$, with probability at least $1 - \delta$,

$$\left| [\frac{\partial\left[\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \hat{f}(\boldsymbol{x}_i)\}\right]}{\partial w_j} - \frac{\partial E\left[L\{Y, \hat{f}(\boldsymbol{X})\}\right]}{\partial w_j}] \right| \le C_1 R_n\{\mathscr{H}_n(\lambda)\} + T_n(\delta) \le C_1 \hat{R}_n\{\mathscr{H}_n(\lambda)\} + 3T_n(\delta/2),$$

where $T_n(\delta) = C_2\{n^{-1}\log(n)\log(1/\delta)\}^{1/2}$, and C_1, C_2 are universal constants that are independent of n.

The proof to Lemma 5 is quite standard in the literature of Rademacher complexity. To bound the LHS of (10) by $C_1 R_n\{\mathscr{H}_n(\lambda)\} + T_n(\delta)$, one can use the McDiarmid inequality [28] and the symmetrization technique [37]. To bound $C_1 R_n\{\mathscr{H}_n(\lambda)\}$ by $C_1 \hat{R}_n\{\mathscr{H}_n(\lambda)\} + 2T_n(\delta/2)$, one can again use the McDiarmid inequality. See the proof of Lemma 3 in Zhang *et al.* [44] for more details. Notice that there are two main differences between the proof of Lemma 3 in Zhang *et al.* [44] and that of Lemma 5. First, in Zhang *et al.* [44], the Rademacher complexity was defined on the function class $\{L(\cdot, f) : f \in \mathscr{H}_n(\lambda)\}$. By Talagrand's Lemma [Lemma 4.2 in 30], the Rademacher complexity of $\{L(\cdot, f) : f \in \mathscr{H}_n(\lambda)\}$ can be further bounded by that of $\mathscr{H}_n(\lambda)$, if the loss function L is Lipshcitz. Second, the maximum changes in the LHS of (10) if we replace one $\boldsymbol{x}_i$ or y_i can be bounded by $C_3 \log(n)/n$ (this is a direct result from Lemma 1) with C_3 being another constant, instead of $O(n^{-1})$ as in Zhang *et al.* [44].

The rest of the proof is analogous to that of Lemma 3 in Zhang *et al.* [44], and we omit the details here. □

The next step is to bound the empirical Rademacher complexity of $\mathcal{H}_n(\boldsymbol{\lambda})$. To this end, notice that

$$E_{\boldsymbol{\sigma}}\{\sup_{f \in \mathscr{H}_n(\lambda)} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(\boldsymbol{x}_i)\} \le E_{\boldsymbol{\sigma}}\{\sup_{f \in \mathscr{H}_n(\lambda)} \frac{1}{n}\sum_{i=1}^{n} \sigma_i \tilde{f}(\boldsymbol{x}_i)\} + E_{\boldsymbol{\sigma}}\{\sup_{f \in \mathscr{H}_n(\lambda)} \frac{1}{n}\sum_{i=1}^{n} \sigma_i b\}. \quad (10)$$

Hence, we proceed to bound the two terms on the RHS of (10). Notice that by Lemma 1, the first term is equivalent to $E_{\boldsymbol{\sigma}}\{\sup_{\|\tilde{f}\|_{\mathscr{H}} = O_P\{\log(n)\}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i \tilde{f}(\boldsymbol{x}_i)\}$ and the second term is equivalent to $E_{\boldsymbol{\sigma}}(\sup_{|b| = O_P\{\log(n)\}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i b)$. For the first term, one can use Theorem 5.5 in Mohri *et al.* [30] to obtain that, with Assumption 2 valid, the corresponding empirical Rademacher complexity is of the order $O_P\{\log(n)/\sqrt{n}\}|$. For the second term, notice that the distribution of Rademacher variables is similar to the binomial distribution. Therefore, we have that for large n, the distribution of $\sup_{|b| = O_P\{\log(n)\}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i b$ can be approximated by that of $|Z|$, where $\{C\sqrt{n}/\log(n)\}\ Z \sim N(0, 1)$, with C a universal constant. Hence, one can verify that

$$E_{\boldsymbol{\sigma}}\{\sup_{|b|O_P\{\log(n)\}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i b\} = E(|Z|) = O_P\{\log(n)/\sqrt{n}\}.$$

Then we have $E_{\boldsymbol{\sigma}}\{\sup_{f \in \mathscr{H}_n(\lambda)} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(\boldsymbol{x}_i)\} = O_P\{\log(n)/\sqrt{n}\}$.

Next, choose $\delta = 2p^{-1}n^{-2}$. One has that $T_n(\delta/2) = O_P[n^{-1}\log(n)\{\log(p) \vee \log(n)\}]^{1/2}$. Consequently, with probability at least $2n^{-2}$, (10) holds true for all the predictors. Combining this with Lemma 4 and the Borel–Cantelli Lemma, we have that (8) is proved.

We now need to show that $\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \sum_{j=1}^{n} K_{\mathbf{w}}\{x_i, x_j)\alpha_j + b\}$, as a function of $(\mathbf{w}^T, \boldsymbol{\alpha}^T, b)^T$, is strictly convex in a small neighborhood around $((\mathbf{w}^*)^T, (\boldsymbol{\alpha}_n^*)^T, (b_n^*)^T$. Because we have shown that $f_{\boldsymbol{\alpha}_n^*, b_n^*}(\boldsymbol{x})$ converges to f_0 in a rate faster than that of $\hat{f}$ to f_0, this guarantees that once we arrive at a temporary point around $((\mathbf{w}^*)^T, (\boldsymbol{\alpha}_n^*)^T, (b_n^*)^T$, the proposed algorithm in Section 2.3 would ensure that the solution $\hat{f}$ converges to the best function f_0. To this end, observe that in Assumption 5, we assume that $E[\frac{1}{n}\sum_{i=1}^{n} L\{Y_i, f(\boldsymbol{X}_i)\}]$ is strictly convex. Hence, it suffices to prove that

$$\sup_{(\mathbf{w}^T, \boldsymbol{\alpha}^T, b)^T \in \mathcal{N}} |\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \sum_{j=1}^{n} K_{\mathbf{w}}\{x_i, x_j)\alpha_j + b\} - E[\frac{1}{n}\sum_{i=1}^{n} L\{Y_i, f(\boldsymbol{X}_i)\}]| \to 0$$

almost surely. Note that when $\mathcal{N}$ is sufficiently small, we have $\sup_{f \in \mathcal{N}} |Pf| < \infty$. Moreover, by Lemma 1 and similar arguments as in the proof of Theorem 1 in Zhang *et al.* [44], one can have that the L_2 entropy of $\{f : f \in \mathcal{N}\}$ is $\log[N\{\varepsilon, \mathcal{N}, L_2(P_n)\}] = O[\log\{\log(n)\}]$, where P_n is the empirical measure of the training set. For any $M < \infty$, define $f_M = f \cdot I(f \le M)$, and $\mathcal{N}_M = \{f_M : f \in \mathcal{N}\}$. One has that $\log[N\{\varepsilon, \mathcal{N}_M, L_2(P_n)\}] = O[\log\{\log(n)\}]$. Therefore, by Theorem 6.2 in Wellner [41], we have that $\mathcal{N}$ is a P-Glivenko–Cantelli class. One can then verify that this conclusion leads to that for n large, $\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \sum_{j=1}^{n} K_{\mathbf{w}}(x_i, x_j)\alpha_j + b\}$

Now we have that, by Assumption 6, the partial derivative of the empirical L loss with respect to each w_j is such that

$$\frac{\partial[\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \hat{f}(\boldsymbol{x}_i)\}]}{\partial w_j}\Big|_{w_j = 0, w_i = w_i^*, i \ne j} \le O_P\{\frac{\{\log(p) \vee \log(n)\}}{\sqrt{n}}\},$$

for $w_j \in \mathbf{w}_{(0)}$, and

$$\left[\frac{\partial[\frac{1}{n}\sum_{i=1}^{n} L\{y_i, \hat{f}(\boldsymbol{x}_i)\}]}{\partial w_j} - \frac{\partial E[L\{Y, f_0(\boldsymbol{X})\}]}{\partial w_j}\right]\Big|_{w_j = 0, w_i = w_i^*, i \neq j} \leq O_P\left\{\frac{\{\log(p) \vee \log(n)\}}{\sqrt{n}}\right\},$$

for $w_j \in \mathbf{w}_{(1)}$. Because the objective function is locally convex, at the optimal point $(\hat{\mathbf{w}}, \hat{\boldsymbol{\alpha}}, \hat{b})$, selection consistency is equivalent to that $\lambda_2 \to 0$ at a rate no faster than $O_P\{\frac{\{\log(p) \vee \log(n)\}}{\sqrt{n}}\}$ [according to the soft thresholding rule in 36]. Hence, we have proven the selection consistency for the DOSK method under the assumption that the distribution of the error has a bounded range.

Lastly, we need to finish the proof by considering the general case that the distribution of the error in regression is sub-Gaussian. This can be done by showing that with a high probability, the actual errors would be bounded in a range. Then we can prove that the corresponding partial derivatives etc. converge at the same rate, because the probability of sub-Gaussian random variables being significantly away from 0 converges to zero very fast, as the bound increases.

Without loss of generality, we assume that $\varepsilon(\mathbf{X})$ follows a common sub-Gaussian distribution with c.d.f. Φ_ε. The generalization of this assumption to the heteroscedastic case is straightforward, because we are only concerned with the tail probability pr($\varepsilon(\boldsymbol{X})| > t$). Next, define $t^* = \Phi_\varepsilon^{-1}(0.5 + 0.5(1 - \delta/2)^{1/n})$, where δ is a small positive number. It can be verified that with probability at least $1 - \delta/2$, all the errors ε_i; $i = 1,\ldots, n$, are in $[-t^*, t^*]$. Since Φ_ε is the c.d.f. of a sub-Gaussian distribution with a fixed parameter, t^* diverges at a rate slower than $O\{\log(n)\}$. One can check that the RHS of (9) can be bounded similarly as in the corresponding proofs, and this completes the proof. ■

Proof of Theorem 2, Part III (Risk Bound)

The proof of this theorem is analogous to that of Lemma 5 and the second half of Selection Consistency (Part II of Theorem 2) (i.e., obtaining the bound on the empirical Rademacher complexity of $\mathscr{H}_n(\boldsymbol{\lambda})$, as well as the convergence rate of $T_n(\delta/2)$). Therefore we omit the details here. ■

References

1. Allen GI. Automatic Feature Selection via Weighted Kernels and Regularization. Journal of Computational and Graphical Statistics. 2012; 22(2):284–299.
2. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proceedings of the National Academy of Sciences. 1999; 96(12):6745–6750.
3. An LTH, Tao PD. Solving a Class of Linearly Constrained Indefinite Quadratic Problems by DC Algorithms. Journal of Global Optimization. 1997; 11(3):253–285.
4. Aronszajn N. Theory of Reproducing Kernels. Transactions of the American Mathematical Society. 1950; 68(3):337–404.
5. Bache K, Lichman M. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences; 2015.

6. Bartlett PL, Bousquet O, Mendelson S. Local Rademacher Complexities. Annals of Statistics. 2005; 33(4):1497–1537.
7. Bertsekas D, Nedic A, Ozdaglar AE. Convex Analysis and Optimization. Athena Scientific; 2003.
8. Blanchard G, Bousquet O, Massart P. Statistical Performance of Support Vector Machines. Annals of Statistics. 2008; 36(2):489–531.
9. Boser BE, Guyon IM, Vapnik VN. Proceedings of the fifth annual workshop on Computational learning theory, COLT '92. New York, NY, USA: ACM; 1992. A Training Algorithm for Optimal Margin Classifiers; 144–152.
10. Boyd S, Vandenberghe L. Convex Optimization. Cambridge: 2004.
11. Breiman L, Friedman JH. Estimating Optimal Transformations for Multiple Regression and Correlation. Journal of the American Statistical Association. 1985; 80(391):580–598.
12. De Boor C. A Practical Guide to Splines. Springer; New York: 2001.
13. Ein-Dor P, Feldmesser J. Attributes of the Performance of Central Processing Units: A Relative Performance Prediction Model. Communications of the ACM. 1987; 30(4):308–317.
14. Fan J, Li R. Variable Selection via Non-concave Penalized Likelihood and its Oracle Properties. Journal of the American Statistical Association. 2001; 96(456):1348–1360.
15. Fan J, Lv J. Sure Independence Screening for Ultrahigh Dimensional Feature Space. Journal of the Royal Statistical Society: Series B. 2008; 70(5):849–911.
16. Fan J, Lv J. A Selective Overview of Variable Selection in High Dimensional Feature Space. Statistica Sinica. 2010; 20(1):101. [PubMed: 21572976]
17. Fan J, Peng H. Nonconcave Penalized Likelihood with a Diverging Number of Parameters. Annals of Statistics. 2004; 32(3):928–961.
18. Gorski J, Pfeuffer F, Klamroth K. Biconvex Sets and Optimization with Biconvex Functions: A Survey and Extensions. Mathematical Methods of Operations Research. 2007; 66(3):373–407.
19. Guyon I, Weston J, Barnhill S, Vapnik VN. Gene Selection for Cancer Classification using Support Vector Machines. Machine Learning. 2002; 46(1–3):389–422.
20. Hastie TJ, Tibshirani RJ, Friedman JH. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer; 2011.
21. Kimeldorf G, Wahba G. Some Results on Tchebycheffian Spline Functions. Journal of Mathematical Analysis and Applications. 1971; 33:82–95.
22. Lee W, Du Y, Sun W, Hayes DN, Liu Y. Multiple Response Regression for Gaussian Mixture Models with Known Labels. Statistical Analysis and Data Mining. 2012; 5(6):493–508.
23. Li Y, Zhu J. L1-norm Quantile Regression. Journal of Computational and Graphical Statistics. 2008; 17:163–185.
24. Lin X, Wahba G, Xiang D, Gao F, Klein R, Klein B. Smoothing Spline Anova Models for Large Data Sets with Bernoulli Observations and the Randomized GACV. Annals of Statistics. 2000; 28(6):1570–1600.
25. Lin Y, Zhang HH. Component Selection and Smoothing in Multivariate Nonparametric Regression. The Annals of Statistics. 2006; 34(5):2272–2297.
26. Liu Y. Fisher Consistency of Multicategory Support Vector Machines. Eleventh International Conference on Artificial Intelligence and Statistics. 2007:289–296.
27. Ma P, Mahoney M, Yu B. A statistical perspective on algorithmic leveraging. In: Xing EP, Jebara T, editorsProceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research. Bejing, China: PMLR; 2014. 91–99.
28. McDiarmid C. Surveys in Combinatorics. Cambridge University Press; 1989. On the Method of Bounded Differences; 148–188.
29. Minh HQ. Some Properties of Gaussian Reproducing Kernel Hilbert Spaces and Their Implications for Function Approximation and Learning Theory. Constructive Approximation. 2010; 32(2):307–338.
30. Mohri M, Rostamizadeh A, Talwalkar A. Foundations of Machine Learning. MIT press; 2012.
31. Nakai K, Kanehisa M. Expert System for Predicting Protein Localization Sites in Gram-negative Bacteria. Proteins. 1991; 11(2):95–110. [PubMed: 1946347]

32. Schölkopf B, Smola AJ. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning). The MIT Press; 2002.
33. Shawe-Taylor J, Cristianini N. Kernel Methods for Pattern Analysis. Cambridge University Press; 2004.
34. Steinwart I, Scovel C. Fast Rates for Support Vector Machines using Gaussian Kernels. Annals of Statistics. 2007; 35(2):575–607.
35. Street WN, Wolberg WH, Mangasarian OL. Nuclear Feature Extraction for Breast Tumor Diagnosis. IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology. 1993:861–870.
36. Tibshirani RJ. Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society: Series B. 1996; 58(1):267–288.
37. Van der Vaart AW, Wellner JA. Weak Convergence and Empirical Processes with Application to Statistics. Springer; 2000.
38. Wahba G. Spline Models for Observational Data. Society for Industrial and Applied Mathematics; 1990.
39. Wang H, Zhu R, Ma P. Optimal Sub-sampling for Large Sample Logistic Regression. Journal of the American Statistical Association. 2017:0–0.
40. Wang L, Zhu J, Zou H. Proceedings of the 24th International Conference on Machine Learning. ACM; 2007. Hybrid Huberized Support Vector Machines for Microarray Classification; 983–990.
41. Wellner JA. Empirical Processes: Theory and Applications. Notes for a Course Given at Delft Univer sity of Technology. 2005
42. Wu S, Shen X, Geyer CJ. Adaptive Regularization using the Entire Solution Surface. Biometrika. 2009; 96(3):513–527. [PubMed: 23946545]
43. Zhang C, Liu Y. Multicategory Large-margin Unified Machines. Journal of Machine Learning Research. 2013; 14:1349–1386. [PubMed: 24415909]
44. Zhang C, Liu Y, Wu Y. On Quantile Regression in Reproducing Kernel Hilbert Spaces with Data Sparsity Constraint. Journal of Machine Learning Research. 2015 In press.
45. Zhang CH. Nearly Unbiased Variable Selection under Minimax Concave Penalty. The Annals of Statistics. 2010; 38(2):894–942.
46. Zhang HH, Cheng G, Liu Y. Linear or Nonlinear? Automatic Structure Discovery for Partially Linear Models. Journal of the American Statistical Association. 2011; 106(495):1099–1112. [PubMed: 22121305]
47. Zhao P, Yu B. On Model Selection Consistency of Lasso. Journal of Machine Learning Research. 2006; 7:2541–2563.
48. Zou H. The Adaptive Lasso and its Oracle Properties. Journal of the American Statistical Association. 2006; 101(476):1418–1429.
49. Zou H, Hastie TJ. Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society: Series B. 2005; 67(2):301–320.
50. Zou H, Li R. One-step Sparse Estimates in Nonconcave Penalized Likelihood Models. The Annals of statistics. 2008; 36(4):1509. [PubMed: 19823597]

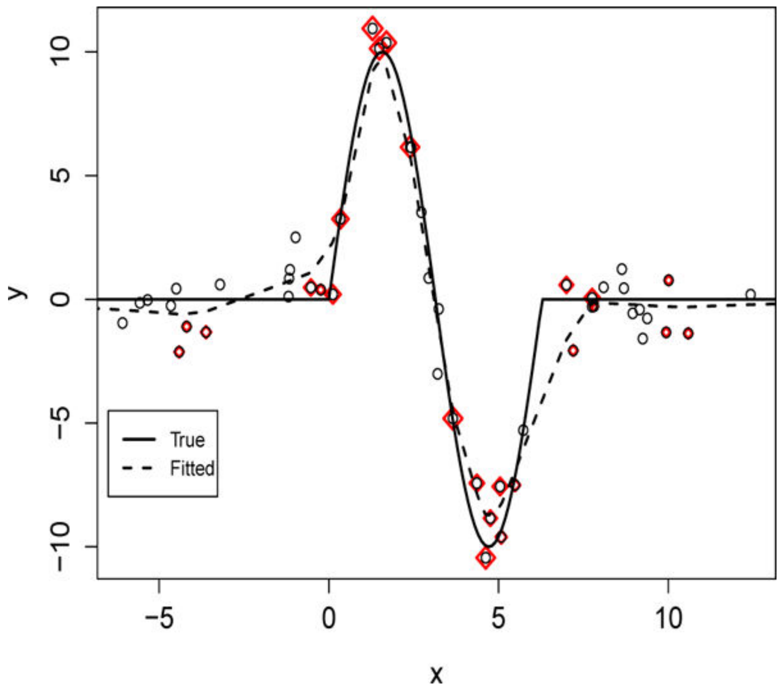

Figure 1.

Plot of the underlying f_0 (solid) and fitted $\hat{f}$ by DOSK (dashed) when $n = 100$ and $p_0 = 2$. Observations with non-zero $\hat{\alpha}_j$'s are highlighted in red. One can see that the data sparsity penalty tends to choose observations that are closer to 0, $\pi/2$, $3\pi/2$ and 2π for the function representation.

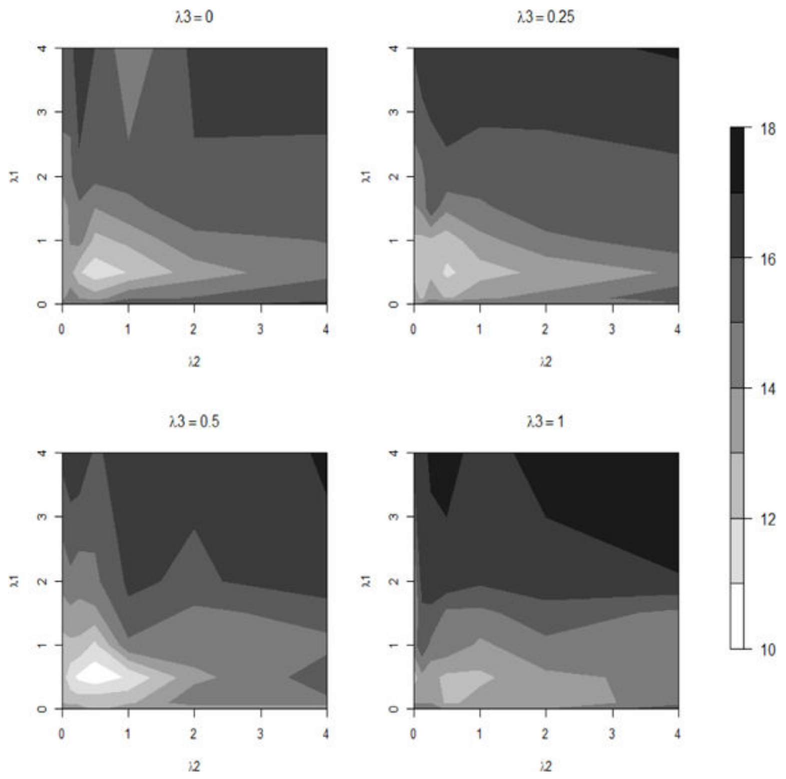

Figure 2.

Contour plots of the mean prediction errors of DOSK for Regression Example 2 where $p_0 = 8$. Here λ_3 is set as {0, 0.25, 0.5, 1} for the four panels and the kernel bandwidth $\tau = 1/2\hat{\sigma}^2$, where $\hat{\sigma}$ is the median of the pairwise Euclidean distances of the simulated samples.

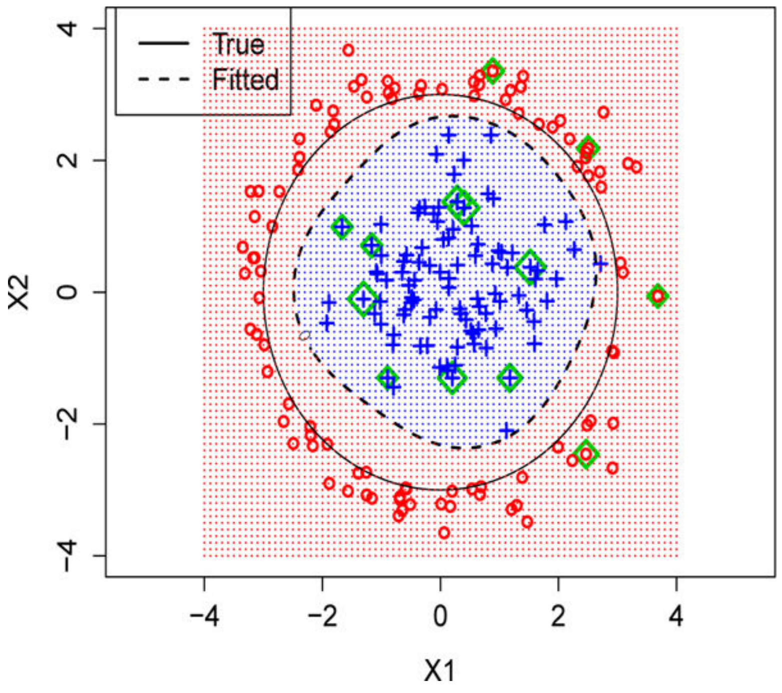

Figure 3.
Plot of the underlying classification boundary (solid circle) and estimated boundary by DOSK (dashed circle) when $n = 200$ and $p_0 = 8$. Observations with non-zero $\hat{\alpha}_j$'s are highlighted in green.

Table 1

Results of Regression Example 1. The numbers in parentheses show the corresponding standard deviations. MPE stands for mean prediction error, TP and FN represent true positive rates and false negative rates, respectively.

p_0	Method	n = 50				n = 100			
		Train MPE	**Test MPE**	**TP**	**FN**	**Train MPE**	**Test MPE**	**TP**	**FN**
2	Linear Ridge	15.89 (4.46)	17.96 (1.33)	–	–	16.29 (3.46)	17.82 (1.16)	–	–
	LASSO	15.89 (4.47)	17.96 (1.32)	1	0.49	16.29 (3.46)	17.82 (1.17)	**1**	0.5
	L_2 Kernel	2.06 (0.45)	11.17 (2.00)	–	–	2.09 (0.38)	7.36 (1.55)	–	–
	SIS	8.22 (5.50)	12.20 (7.13)	0.42	0.29	5.39 (5.85)	7.54 (7.51)	0.68	0.16
	RFE	4.77 (3.91)	10.57 (6.05)	0.44	0.30	3.10 (3.51)	5.44 (5.02)	0.7	0.16
	COSSO	7.05 (6.56)	11.99 (10.32)	0.56	0.39	0.96 (1.29)	1.99 (2.58)	0.98	0.53
	KNIFE	3.66 (0.48)	6.14 (2.00)	**1**	0.14	2.35 (0.19)	3.03 (0.57)	**1**	**0**
	DOSK	1.42 (0.21)	**3.40 (2.92)**	**1**	**0.04**	0.92 (0.13)	**1.42 (0.19)**	**1**	**0**
8	Linear Ridge	13.77 (2.89)	18.09 (1.55)	–	–	16.11 (2.78)	17.68 (1.03)	–	–
	LASSO	13.77 (2.89)	18.12 (2.15)	**1**	0.87	16.13 (2.77)	17.61 (1.02)	**1**	0.88
	L_2 Kernel	0.05 (0.01)	17.26 (1.52)	–	–	0.05 (0.01)	15.76 (1.05)	–	–
	SIS	3.94 (2.04)	16.18 (4.44)	0.46	0.31	3.07 (1.90)	9.01 (3.95)	0.86	0.26
	RFE	9.83 (4.97)	16.18 (12.30)	0.54	**0.24**	6.44 (5.73)	10.29 (6.03)	0.86	0.25
	COSSO	12.27 (40.97)	19.93 (12.30)	0.54	**0.24**	6.44 (5.73)	10.29 (8.66)	0.76	0.25
	KNIFE	2.40 (0.53)	13.89 (3.64)	**1**	0.42	1.58 (0.18)	2.69 (1.99)	**1**	0.22
	DOSK	2.70 (0.59)	**10.80 (5.59)**	0.95	0.29	1.12 (0.20)	**2.15 (2.81)**	**1**	**0.20**

Table 2

Results of Regression Example 2. The numbers in parentheses show the corresponding standard deviations. MPE stands for mean prediction error, TP and FN represent true positive rates and false negative rates, respectively.

p_0	Method	n = 50				n = 100			
		Train MPE	Test MPE	TP	FN	Training MPE	Test MPE	TP	FN
2	Linear Ridge	30.20 (7.34)	35.01 (2.54)	–	–	32.11 (5.91)	33.97 (2.09)	–	–
	LASSO	30.19 (7.34)	35.00 (2.57)	0.99	0.50	32.10 (5.91)	33.97 (2.09)	1	0.50
	L_2 Kernel	0.05 (0.02)	28.01 (2.57)	–	–	0.04 (0.01)	23.94 (2.09)	–	–
	SIS	1.07 (2.09)	30.92 (3.53)	0.34	0.31	1.92 (2.70)	29.61 (3.62)	0.29	0.41
	RFE	8.75 (8.22)	32.15 (4.05)	0.34	0.32	14.32 (9.20)	30.34 (3.53)	0.30	0.27
	COSSO	14.56 (4.60)	31.45 (11.10)	0.49	**0.17**	16.33 (8.93)	21.09 (9.62)	0.48	**0.11**
	KNIFE	6.56 (1.33)	21.26 (3.12)	**1**	0.49	5.99 (0.54)	12.99 (1.29)	**1**	0.18
	DOSK	2.14 (0.61)	**18.25 (3.70)**	**1**	0.54	2.60 (0.31)	**9.86 (1.44)**	**1**	0.12
8	Linear Ridge	26.28 (7.09)	33.95 (3.05)	–	–	30.06 (5.60)	34.21 (1.73)	–	–
	LASSO	26.26 (7.07)	33.94 (3.04)	1	0.88	29.06 (5.41)	33.17 (1.69)	1	0.88
	L_2 Kernel	0.05 (0.02)	33.97 (3.05)	–	–	0.04 (0.01)	26.23 (1.73)	–	–
	SIS	0.05 (0.03)	33.63 (2.94)	0.32	0.33	0.04 (0.01)	33.71 (1.84)	0.31	0.35
	RFE	10.54 (7.79)	32.90 (3.50)	0.33	**0.18**	13.92 (10.32)	32.25 (3.30)	0.32	0.19
	COSSO	18.36 (7.82)	35.54 (6.68)	0.31	0.25	16.41 (7.13)	27.14 (7.13)	0.51	0.18
	KNIFE	5.47 (0.78)	25.53 (4.03)	**0.99**	0.46	5.53 (0.50)	14.52 (2.41)	**1**	0.17
	DOSK	1.54 (0.33)	**23.97 (6.10)**	**0.99**	0.36	2.37 (0.28)	**10.70 (3.20)**	**1**	**0.15**

Table 3

Results of Classification Example 1. The numbers in parentheses show the corresponding standard deviations. MSC stands for Mis-Classification Rate, TP and FN represent true positive rates and false negative rates, respectively.

p_0	Method	Train MCR	Test MCR	TP	FN
	L_2 Kern	2.94 (0.93)	2.92 (0.50)		–
	SIS	2.94 (0.93)	2.92 (0.50)	**1**	**0**
0	RFE	2.94 (0.93)	2.92 (0.50)	**1**	**0**
	KNIFE	4.00 (2.92)	4.32 (3.94)	0.98	**0**
	DOSK	1.63 (0.73)	**1.72 (0.34)**	**1**	**0**
	L_2 Kern	1.63 (0.89)	6.68 (0.75)		–
	SIS	2.31 (1.22)	5.23 (1.50)	1	0.69
4	RFE	9.48 (12.84)	12.02 (12.40)	0.8	0.36
	KNIFE	3.33 (1.30)	3.31 (0.50)	1	0
	DOSK	2.07 (0.12)	**2.02 (0.56)**	**1**	**0**
	L_2 Kern	0.08 (0.21)	15.07 (1.89)		–
	SIS	0.96 (1.00)	9.53 (4.45)	1	0.66
8	RFE	5.42 (8.97)	12.18 (9.16)	0.86	0.46
	KNIFE	3.48 (1.87)	3.89 (2.97)	0.99	**0**
	DOSK	1.58 (1.63)	**1.79 (0.34)**	**1**	**0**

Table 4

Results of Classification Example 2. The numbers in parentheses show the corresponding standard deviations. MSC stands for Mis-Classification Rate, TP and FN represent true positive rates and false negative rates, respectively.

p_0	Method	Train MCR	Test MCR	TP	FN
	L_2 Kern	6.34 (0.15)	8.08 (0.80)	–	–
	SIS	6.34 (0.15)	8.08 (0.80)	1	0
0	RFE	6.34 (0.15)	8.08 (0.80)	1	0
	KNIFE	7.30 (1.70)	8.85 (0.87)	1	0
	DOSK	4.37 (1.74)	**5.81 (0.73)**	**1**	**0**
	L_2 Kern	1.58 (1.08)	14.56 (1.23)	–	–
	SIS	2.59 (1.02)	13.49 (1.87)	1.00	0.84
4	RFE	10.82 (3.96)	19.96 (6.87)	0.76	0.52
	KNIFE	7.73 (1.88)	9.41 (1.66)	1	0
	DOSK	4.94 (1.68)	**6.00 (0.84)**	**1**	**0**
	L_2 Kern	0.02 (0.01)	22.28 (1.65)	–	–
	SIS	2.02 (5.64)	19.60 (3.72)	0.96	0.72
8	RFE	8.12 (2.10)	22.93 (6.21)	0.76	0.50
	KNIFE	7.21 (1.72)	9.03 (1.20)	1	0
	DOSK	5.04 (1.75)	**5.93 (0.64)**	**1**	**0**

Table 5

Average runtime (in second) of DOSK for the extended Class-2 example with different combinations of the number of samples (#Obs) and variables (#Var) per tuning parameter combination.

#Var/#Obs	200	400	800	1600
10	12.3	53.8	103.8	518.2
50	39.6	71.3	192.5	604.3
200	54.1	189.2	398.2	1460.747
500	216.8	526.1	972.2	3351.2
1000	509.6	832.2	1851.4	8381.2
2000	1056.1	1881.3	3056.3	18972.5

Table 6

Average runtime (in second) of each method per tuning parameter combination in the selected numerical studies. Here $n = 100$ and $p_0 = 8$ for all simulated examples.

Examples	**Reg-1**	**Reg-2**	**CPUs**		**Class-1**	**Class-2**	**Ecoli**
Methods	Runtime	Runtime	Runtime	Methods	Runtime	Runtime	Runtime
Linear Ridge	0.26	0.36	0.22				
LASSO	1.12	0.87	0.57				
L_2 Kernel	13.65	13.09	11.94	L_2 Kernel	4.39	4.41	2.18
SIS	11.18	13.31	13.50	SIS	17.13	17.91	13.73
RFE	41.25	69.27	57.71	RFE	28.42	39.87	16.72
COSSO	34.23	39.37	42.84				
KNIFE	7.48	8.33	6.12	KNIFE	8.44	10.03	5.31
DOSK	7.08	7.01	6.01	DOSK	8.99	9.64	5.64

Table 7

Results in terms of the mean prediction error (MPE) for the ozone and CPUs data sets.

	Ozone		**CPUs**	
Methods	**Train MPE**	**Test MPE**	**Train MPE**	**Test MPE**
L_2 Kernel	12.51 (1.27)	17.37 (1.68)	0.01 (0.002)	0.40 (0.24)
LASSO	19.34 (1.36)	20.80 (1.69)	0.11 (0.04)	0.21 (0.09)
SIS	18.72 (1.61)	21.47 (1.78)	0.11 (0.03)	0.33 (0.21)
RFE	13.89 (1.44)	18.37 (1.73)	0.02 (0.01)	0.35 (0.20)
COSSO	17.56 (2.14)	20.45 (1.96)	0.12 (0.07)	0.28 (0.12)
KNIFE	11.03 (1.09)	17.08 (1.90)	0.10 (0.01)	0.17 (0.08)
DOSK	11.21 (1.41)	**16.92 (1.65)**	0.09 (0.02)	**0.16 (0.10)**

Table 8

Results in terms of the Mis-Classification Rate (MCR, in percentages) for the breast cancer and Ecoli data sets.

	Breast Cancer		**Ecoli**	
Methods	**Train MCR**	**Test MCR**	**Train MCR**	**Test MCR**
L_2 Kernel	0.39 (0.24)	7.78 (1.42)	0.22 (0.33)	13.24 (4.42)
SIS	1.27 (0.73)	4.20 (1.09)	0.95 (0.68)	2.13 (1.21)
RFE	1.33 (0.56)	4.26 (1.00)	0.95 (0.68)	2.13 (1.25)
KNIFE	1.77 (0.54)	4.04 (0.78)	1.69 (0.81)	2.26 (1.27)
DOSK	2.40 (0.60)	**3.97 (1.11)**	1.52 (1.02)	**1.95 (1.02)**

Table 9

Average validation misclassification rates with standard deviations (in parenthesis) on 5-fold cross validation of 100 replicates for the gene expression data. All methods use the linear kernel, and the best ones are in bold.

# Genes	SVM	L_1 Logistic	KNIFE	DOSK
10	0.116 (0.004)	0.129 (0.003)	0.092 (0.004)	**0.091** (0.004)
50	0.082 (0.005)	0.090 (0.004)	0.079 (0.007)	**0.071** (0.006)
200	0.077 (0.004)	0.069 (0.003)	0.059 (0.006)	**0.047** (0.004)
500	0.062 (0.005)	0.048 (0.003)	0.041 (0.005)	**0.036** (0.004)
1000	0.075 (0.005)	0.044 (0.003)	0.038 (0.004)	**0.035** (0.003)
2000	0.065 (0.005)	0.042 (0.004)	0.034 (0.005)	**0.032** (0.004)