Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xx.xx.xx

pp. **X–XX**

MODELING INTERACTIVE COMPONENTS BY COORDINATE KERNEL POLYNOMIAL MODELS

Xin Guo*

Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong, China

Lexin Li

Division of Biostatistics, University of California, Berkeley, Berkeley, CA 94720, USA

Qiang Wu

Department of Mathematical Sciences, Middle Tennessee State University, Murfreesboro, TN 37132, USA

(Communicated by the associate editor name)

Abstract. We proposed the use of coordinate kernel polynomials in kernel regression. This new approach, called coordinate kernel polynomial regression, can simultaneously identify active variables and effective interactive components. Reparametrization refinement is found critical to improve the modeling accuracy and prediction power. The post-training component selection allows one to identify effective interactive components. Generalization error bounds are used to explain the effectiveness of the algorithm from a learning theory perspective and simulation studies are used to show its empirical effectiveness.

1. **Introduction.** Building models and making inference from data are central tasks in machine learning and data mining. They play essential roles in revealing mechanisms of natural and social phenomena and forecasting the future. Because most phenomena are the results of interactions of a group of relevant factors, two problems are central to modern data analysis in this big data era. One is the identification of important factors that are relevant to the phenomenon under investigation. The other is the modeling of interactions between relevant factors to understand the underlying mechanism. These tasks are challenging because the existence of a large amount of irrelevant information makes many techniques less robust. While many methods have been developed for the identification of relevant factors under the context of variable selection, research in the interactive component modeling is very sparse. The main purpose of this paper is to develop a class

2010 *Mathematics Subject Classification.* Primary: 68T05, 41A25; Secondary: 68Q32.

Key words and phrases. interactive component, kernel method, coordinate kernel polynomial model, reparametrization, generalization, information criterion.

The work described in this paper is partially supported by FRCAC of Middle Tennessee State University, and is partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 25301115). All the three authors contributed equally to the paper.

* Corresponding author: Xin Guo.

of novel kernel models that are able to simultaneously identify relevant factors and model their interactions.

Variable selection algorithms usually involve a variable ranking process, either explicitly or implicitly. In filters methods [8], the ranking metrics could be correlation coefficients, statistical testing scores, mutual information, and the effective variable subset is scored by an evaluation process. A common problem of these methods is the redundant selection because highly correlated variables produce similar ranking metrics. Some algorithms integrate variable selection into the model construction or model selection process. Examples include LASSO [23], elastic net [34], and the support vector machine recursive feature eliminations [7]. These linear model based methods, however, may not be able to detect all relevant variables. For instance, in LASSO and elastic net, those variables uncorrelated to the response variable will not be selected even if they are relevant. This advocates the necessity of developing variable selection algorithms based on nonlinear models. In [17, 16] variable ranking by gradient and kernel gradient learning were proposed for both regression and binary classification problems. [12] proposed nonparametric Bayesian kernel models for variable selection. Algorithms were also proposed for simultaneous variable selection and dimension reduction, e.g. the sparse PCA, sparse ridge sliced inverse regression [11, 33], and sparse MAVE [26, 30]. The primary purpose of these methods is dimension reduction. Variable selection is used to improve the interpretability of the effective dimensions. The accuracy of variable selection usually shrinks quickly as the number of effective dimensions increases.

Compared with relevant feature selection, the interactive component modeling is even more challenging because the number of interactive components is huge in the high dimensional setting. For instance, if the dimension of the data is of thousands, the number of two-way interactive components is already of millions. Things will be even severer if multiway interactions have to be considered. The computational complexity prevents direct and independent modeling of all interactive components. In the literature we may see many measurements that are used to describe the relationship between variables in some sense. For instance, traditionally people use covariance and correlation to measure the dependence between variables. In Gauss-Markov graphic models [22, 10], the precision matrix is used to measure the conditional dependence. In [4] the sparse factor models are used to study the interactions in genes expression data analysis. These measurements help to understand the mechanisms of the underlying systems. However, without a method that could incorporate the interaction information into predictive modeling, they cannot be used to improve forecasting performance.

As far as we know, research works on building predictive models with interactive components are very sparse. The classical polynomial models are examples falling into this category where the high order terms are used to measure the effect of interactions in the forecasting. The component selection and smoothing operator (COSSO, [32, 13]) is another method that was proposed to detect nonlinear additive and interactive components in the smoothing spline ANOVA framework. Both methods, however, face difficulty in high dimension data analysis due to high computational complexity resulted by the geometrically increasing number of components.

Kernel methods are standard tools in machine learning to extend liner models to nonlinear models. The use of kernels dates back to kernel density estimators and spline models in statistics [25] and radial basis functions in computational mathematics [3]. Its resurgence in machine learning is due to the success of support vector machines [24] and kernel principal components analysis [20, 19]. Kernels have been used in a lot of real world applications including microarray data analysis, handwritten digits recognition, face recognition, and so on. The performance of kernel methods highly depends on the choice of the kernel in practical applications. It is not an easy task to find the appropriate kernel for a particular application. This has driven the research in kernel construction, kernel selection, and multiple kernel learning [18, 27, 1, 5, 28, 29, 9, 21]. A common criticism on kernel methods is that a kernel machine quite looks like a black box and is hard to interpret. For many commonly used kernels such as the Gaussian kernels, the relevance of the variables and components are mixed and hidden in the kernel functions. Although the information of the variables and their interactions are coded in the solution, it is difficult, if not impossible, to extract such information for the purpose of feature selection and interaction identification. Therefore, it would be interesting to design interpretable kernels and integrate variable selection and interactive component identification directly into kernel methods.

The purpose of this paper is to develop a new class of kernel models, which we call coordinate kernel polynomial models, for interactive component identification. The algorithm involves three stages, which are described in detail in Section 2. In Section 3 we discuss the connections between this new algorithm with some existing approaches in the literature. We provide the generalization error bounds in Section 4 and perform simulation studies in Section 5.

2. Coordinate Kernel Polynomial Models for Regression.

In this section we describe our kernel method for simultaneous variable selection and interactive component identification. The algorithm includes three stages: the primary learning based on the coordinate kernel polynomial models, the reparameterization refinement, and the post-training component selection.

2.1. Coordinate kernel polynomial.

In regression problem, a set of observations are collected for p predictors and a scalar response variable which are linked by

$$y_i = f^*(\mathbf{x}_i) + \epsilon_i, \qquad i = 1, 2, \ldots, m,$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top \in \mathbb{R}^p$, $y_i \in \mathbb{R}$, and ϵ_i is the zero-mean noise. The target is to recover the unknown true model f^* as accurate as possible to understand the impact of predictors and predict the response on unobserved data. The ordinary least square (OLS) is the most traditional and well developed method. It assumes a linear model and estimates the coefficients by minimizing the squared error between the responses and the predictions. A variety of regularizations have been applied to this linear method for different purposes. These regularized methods include ridge regression, LASSO, elastic net and many others.

Kernel techniques have been used to extend OLS so that nonlinear relationship between the predictors and response variable can be estimated. Let $K(\mathbf{x}, \mathbf{t})$ be a Mercer kernel defined for $\mathbf{x}, \mathbf{t} \in \mathbb{R}^p$, i.e. K is continuous, symmetric and positive semi-definite. There is a reproducing kernel Hilbert space $\mathcal{H}_K$ associated to K induced by the inner product $\langle K(\mathbf{x}, \cdot), K(\mathbf{t}, \cdot)\rangle_K = K(\mathbf{x}, \mathbf{t})$ and satisfying $\langle f, K(\mathbf{x}, \cdot)\rangle_K = f(\mathbf{x})$ for $f \in \mathcal{H}_K$. The regularized kernel regression estimates the true regression function by a function $\hat{f} \in \mathcal{H}_K$ minimizing the penalized squared error:

$$\hat{f} = \arg\min_{f\in\mathcal{H}_K} \left\{ \frac{1}{m}\sum_{i=1}^{m}(y_i - f(\mathbf{x}_i))^2 + \lambda\|f\|_K^2 \right\}, \tag{1}$$

where $\lambda > 0$ is a tunable parameter trading off the fitting error and the model complexity.

Designing kernels for particular domain applications or to have particular mathematical properties is critical for the success of kernel methods. To model the interactive components the kernel must be expandable so that each interactive components could be represented explicitly. At the same time, the number of parameters cannot blow up when multiway interactions enter the model.

The idea of *coordinate kernel polynomial models (CKPM)* is motivated by the smoothing spline component decomposition [13, 32] and learning polynomial combination of kernels [5]. For $\mathbf{x} = (x_1, x_2, \dots, x_p)$, $\mathbf{t} = (t_1, t_2, \dots, t_p) \in \mathbb{R}^p$, let $K_\ell(x_\ell, t_\ell)$ be a kernel that depends only on the ℓth coordinate. We call such a kernel a *coordinate kernel.* Let $d \geq 1$ be an integer. Define

$$K_{\boldsymbol{\omega}}(\mathbf{x},\mathbf{t}) = \left(\sum_{i=1}^{p} w_\ell K_\ell(x_\ell, t_\ell)\right)^d, \qquad w_\ell \geq 0. \tag{2}$$

It is a polynomial of the coordinate kernels of degree d. Therefore we call it a *coordinate kernel polynomial* (CKP). The choice of coordinate kernels can be flexible and problem dependent. Note that when $d = 1$ the kernel $K_{\boldsymbol{\omega}}$ is a linear combination of the coordinate kernels and the associated model is an additive model. When $d \geq 2$, the kernel is a nonlinear combination of the coordinate kernels. The associated model will include interactive components as explained below.

The main advantage of using coordinate kernel polynomial in kernel algorithm is its ability to explicitly represent interactive components. To see this, let $\mathbf{s} \subset \{1, 2, \dots, p\}$ and $|\mathbf{s}|$ denote the number of elements in $\mathbf{s}$. We use $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_p)$ to represent non-negative integer valued p-dimensional vectors with $\sum_{\ell=1}^{p} \alpha_\ell = d$ and $\operatorname{supp}(\boldsymbol{\alpha})$ stands for the subset of $\{1, 2, \dots, p\}$ containing the indices of nonzero elements in $\boldsymbol{\alpha}$. We can expand the coordinate kernel polynomial

$$K_{\boldsymbol{\omega}}(\mathbf{x},\mathbf{t}) = \sum_{j=1}^{d}\sum_{|\mathbf{s}|=j} K_{\mathbf{s}}(\mathbf{x},\mathbf{t}) = \sum_{j=1}^{d}\sum_{|\mathbf{s}|=j}\left(\sum_{\operatorname{supp}(\boldsymbol{\alpha})=\mathbf{s}}\binom{d}{\boldsymbol{\alpha}}\prod_{\ell=1}^{p}\left(w_\ell K_\ell(x_\ell,t_\ell)\right)^{\alpha_\ell}\right).$$

Together with the representer theorem, the solution to a kernel algorithm can be written as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} c_i K_{\boldsymbol{\omega}}(\mathbf{x},\mathbf{x}_i) = \sum_{j=1}^{d}\sum_{|\mathbf{s}|=j}\sum_{i=1}^{m} c_i K_{\mathbf{s}}(\mathbf{x},\mathbf{x}_i) = \sum_{j=1}^{d}\sum_{|\mathbf{s}|=j}\hat{f}_{\mathbf{s}}(\mathbf{x}). \tag{3}$$

When $|\mathbf{s}| = 1$, $\hat{f}_{\mathbf{s}}$ represents an additive component. When $|\mathbf{s}| = j \geq 2$, $\hat{f}_{\mathbf{s}}$ is the j-way interactive components between the variables indexed by $\mathbf{s}$.

When CKPM are used for high dimensional data analysis, a sparse weight vector $\boldsymbol{\omega} = (w_1, w_2 \dots, w_p)$ is preferred in order to facilitate variable selection. This can be implemented by introducing an ℓ_1 penalty on the weight vector $\boldsymbol{\omega}$ into the learning process. By learning a sparse vector $\boldsymbol{\omega}$, the variables with zero weights are deactivated and only those variables with nonzero weights enter the kernel model construction. As for the interactive components, firstly we see that all the interactive components involving deactivated variables are obviously inactive because

$K_{\mathbf{s}} = 0$ if $w_\ell = 0$ for some $\ell \in \mathbf{s}$. So the problem is for those interactive components involving active variables. Such a component is not necessarily active although the associated variables are active. But it is not directly seen whether it is active or inactive from the weight coefficients. For these components we will suggest a post-training selection process in Section 2.4.

In this paper we focus on the use of coordinate kernel polynomial models in regression setting. The algorithm is discussed in details in Sections 2.2, 2.3, and 2.4 below.

2.2. **Coordinate kernel polynomial regression and its solution.** When the coordinate kernel polynomial $K_{\boldsymbol{\omega}}$ is used in regression setting, the algorithm, called *coordinate kernel polynomial regression* (CKPR), is given by

$$\hat{f}_{\lambda,\mu} = \arg \min_{\substack{f \in \mathcal{H}_{K_{\boldsymbol{\omega}}} \\ \boldsymbol{\omega} \geq \mathbf{0}}} \left\{ \frac{1}{m} \sum_{i=1}^{m} (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{K_{\boldsymbol{\omega}}}^2 + \mu \sum_{\ell=1}^{p} w_\ell \right\}. \tag{4}$$

The penalty $\lambda \|f\|_{K_{\boldsymbol{\omega}}}^2$ is used to prevent overfitting and the penalty $\mu \sum_{\ell=1}^{p} w_\ell$ is used to control the sparsity of $\boldsymbol{\omega}$. The restriction $\boldsymbol{\omega} \geq \mathbf{0}$ means that all the coordinates $w_1, w_2, \ldots,$ and w_p are non-negative.

To solve the problem, let $\mathbb{K}_\ell$ be the Gram matrix of the coordinate kernel K_ℓ and $\mathbb{K}_{\boldsymbol{\omega}}$ stand for the Gram matrix of the kernel $K_{\boldsymbol{\omega}}$. Then

$$\mathbb{K}_{\boldsymbol{\omega}} = \left(\sum_{\ell=1}^{p} w_\ell \mathbb{K}_\ell \right)^{\circ d},$$

where $(\cdot)^{\circ d}$ denotes the Hadamard power of matrices. The representer theorem tells that the solution to the kernel machines takes the form

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i K_{\boldsymbol{\omega}}(\mathbf{x}, \mathbf{x}_i) = \sum_{i=1}^{m} c_i \left(\sum_{\ell=1}^{p} w_\ell K_\ell(x_\ell, x_{i\ell}) \right)^{d}. \tag{5}$$

Thus $(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_m))^\top = \mathbb{K}_{\boldsymbol{\omega}} \mathbf{c}$ where $\mathbf{c} = (c_1, c_2, \ldots, c_m)^\top$. Denote $\mathbf{y} = (y_1, y_2, \ldots, y_m)^\top$ and let $\|\cdot\|$ stand for the Euclidean norm. The target function we minimize to solve CKPR (4) has the form

$$\mathcal{E}(\mathbf{c}, \boldsymbol{\omega} | \lambda, \mu) = \frac{1}{m} \|\mathbf{y} - \mathbb{K}_{\boldsymbol{\omega}} \mathbf{c}\|^2 + \lambda \mathbf{c}^\top \mathbb{K}_{\boldsymbol{\omega}} \mathbf{c} + \mu \sum_{\ell=1}^{p} w_\ell. \tag{6}$$

It is trivial to check that

$$\mathcal{E}(\mathbf{c}, \boldsymbol{\omega} | \lambda, \mu) = \mathcal{E}(\mathbf{c} \xi^d, \boldsymbol{\omega} \xi^{-1} | \lambda \xi^{-d}, \mu \xi)$$

for any $\xi > 0$. As a result, if $(\mathbf{c}^*, \boldsymbol{\omega}^*)$ minimizes $\mathcal{E}(\mathbf{c}, \boldsymbol{\omega} | \lambda, \mu)$ and let $\xi = \sum_{\ell=1}^{p} w_\ell^*$, then

$$(\tilde{\mathbf{c}}^*, \tilde{\boldsymbol{\omega}}^*) = (\mathbf{c}^* \xi^d, \boldsymbol{\omega}^* \xi^{-1}) \tag{7}$$

minimizes

$$\mathcal{E}(\tilde{\mathbf{c}}, \tilde{\boldsymbol{\omega}} | \tilde{\lambda}, \tilde{\mu}) = \mathcal{E}(\mathbf{c} \xi^d, \boldsymbol{\omega} \xi^{-1} | \lambda \xi^{-d}, \mu \xi) \quad \text{s.t.} \sum_{\ell=1}^{p} \tilde{w}_\ell = 1, \tag{8}$$

which is equivalent to minimizing

$$\mathcal{E}(\tilde{\mathbf{c}}, \tilde{\boldsymbol{\omega}} | \tilde{\lambda}, \tilde{\mu}) = \frac{1}{m} \|\mathbf{y} - \mathbb{K}_{\tilde{\boldsymbol{\omega}}} \tilde{\mathbf{c}}\|^2 + \tilde{\lambda} \tilde{\mathbf{c}}^\top \mathbb{K}_{\tilde{\boldsymbol{\omega}}} \tilde{\mathbf{c}} + \tilde{\mu} \quad \text{s.t.} \sum_{\ell=1}^{p} \tilde{w}_\ell = 1. \tag{9}$$

We see that $\tilde{\mu}$ does not affect the solution and can be dropped. Therefore this new problem is advantageous in that $\tilde{\lambda}$ is the only parameter to be tuned.

Theoretically speaking, the solution $(\tilde{\mathbf{c}}^*, \tilde{\boldsymbol{\omega}}^*)$ to (9) can be used to recover the solution $(\mathbf{c}^*, \boldsymbol{\omega}^*)$ to (6) through their relationship (7). Though, this is impractical since we do not know ξ without solving (6). However, this is also unnecessary if we notice that both $(\tilde{\mathbf{c}}^*, \tilde{\boldsymbol{\omega}}^*)$ and $(\mathbf{c}^*, \boldsymbol{\omega}^*)$ generate the same function. To summarize, we have the following theorem.

Theorem 2.1. *If $(\tilde{\mathbf{c}}^*, \tilde{\boldsymbol{\omega}}^*)$ solves the minimization problem (9), then the solution to the CKPR (4) is given as*

$$\hat{f}_{\lambda,\mu}(\mathbf{x}) = \hat{f}_{\tilde{\lambda}}(\mathbf{x}) = \sum_{i=1}^{n} \tilde{c}_i^* K_{\tilde{\boldsymbol{\omega}}^*}(\mathbf{x}, \mathbf{x}_i)$$

provided that $\hat{f}_{\lambda,\mu}$ is not identically zero.

Next let us simply discuss the optimization of (9). Note that for fixed $\boldsymbol{\omega}$ the optimal $\mathbf{c}$ is given by

$$\tilde{\mathbf{c}}^*(\boldsymbol{\omega}) = (\mathbb{K}_{\boldsymbol{\omega}} + m\tilde{\lambda}\mathbb{I}_m)^{-1}\mathbf{y} \tag{10}$$

where and in the sequel $\mathbb{I}_m$ represents the identity matrix of size $m \times m$. Plugging it into the function $\mathcal{E}$ we obtain

$$\mathcal{E}(\tilde{\mathbf{c}}, \tilde{\boldsymbol{\omega}} | \tilde{\lambda}, \tilde{\mu}) = \tilde{\lambda}\mathbf{y}^\top (\mathbb{K}_{\boldsymbol{\omega}} + m\tilde{\lambda}\mathbb{I}_m)^{-1}\mathbf{y} + \tilde{\mu}.$$

Therefore we can solve $\tilde{\boldsymbol{\omega}}^*$ by

$$\tilde{\boldsymbol{\omega}}^* = \tilde{\boldsymbol{\omega}}^*_{\tilde{\lambda}} = \arg\min \mathbf{y}^\top (\mathbb{K}_{\boldsymbol{\omega}} + m\tilde{\lambda}\mathbb{I}_m)^{-1}\mathbf{y}, \quad \text{s.t.} \sum_{\ell=1}^{p} \tilde{w}_\ell = 1. \tag{11}$$

After $\tilde{\boldsymbol{\omega}}^*$ is obtained, $\tilde{\mathbf{c}}^* = \tilde{\mathbf{c}}^*(\tilde{\boldsymbol{\omega}}^*)$ can be computed using (10).

2.3. **Reparametrization refinement.** From Theorem 2.1, we see that the solution of CKPR depends only on one parameter $\tilde{\lambda}$. But its role is not as clear as the two parameters λ and μ in (4). Note that λ trades off the data fitting and model complexity. It is usually chosen sufficiently small to avoid introducing too much bias and at the same time not too small to result in over fitting. The parameter μ for the ℓ_1 penalty term on the weight vector $\boldsymbol{\omega}$ controls its sparsity. In case that the true model depends on only a few variables, large μ should be used to ensure $\boldsymbol{\omega}$ shrinking and sparse.

Recall that $\tilde{\lambda} = \lambda\xi^{-d}$ with ξ the ℓ_1 norm of $\boldsymbol{\omega}^*$. If a sparse $\boldsymbol{\omega}^*$ is preferable, large μ should be used, which leads to small ξ. Thus $\tilde{\lambda}$ should be large. At the same time, to fit the data well, a small choice of λ is usually preferred which requires small $\tilde{\lambda}$. Therefore, we see the sparsity and data fitting accuracy has contradictory requirements on the size of $\tilde{\lambda}$. When tuning the parameter $\tilde{\lambda}$ in practice, we observed that it is neither large enough to produce sparse $\tilde{\boldsymbol{\omega}}^*$ for variable selection nor small enough to provide good learning performance. Instead, the parameter is usually chosen at the middle. As a result, CKPR by tuning parameter $\tilde{\lambda}$ is suboptimal for both variable selection and data fitting.

To overcome this problem, we refine the result by a reparameterized training process. After learning the optimal $\tilde{\boldsymbol{\omega}}^*$ using $\tilde{\lambda}$, we introduce a new parameter γ instead of using the original parameter $\tilde{\lambda}$ when we apply (10) to compute $\tilde{\mathbf{c}}^*$. That is,

$$\hat{\mathbf{c}} = \hat{\mathbf{c}}_{\tilde{\lambda},\gamma} = (\mathbb{K}_{\tilde{\boldsymbol{\omega}}^*_{\tilde{\lambda}}} + m\gamma\mathbb{I}_m)^{-1}\mathbf{y}. \tag{12}$$

The prediction and model validation will use the function

$$\hat{f}_{\tilde{\lambda},\gamma}(\mathbf{x}) = \sum_{i=1}^{m} \hat{c}_{\tilde{\lambda},\gamma,i} K_{\tilde{\boldsymbol{\omega}}^*}(\mathbf{x}, \mathbf{x}_i). \tag{13}$$

Simulation studies have confirmed the ability of reparametrization refinement to improve both variable selection accuracy and predictive performance.

2.4. Post-training component selection.

While the interactive components associated to irrelevant variables are deactivated automatically, it is not as obvious to tell whether an interactive component associated to relevant variables are effective or not. We need a post-training selection process to identify those effective interactions. This is possible because the CKPM is able to explicitly represent each interactive components.

To identity effective components, we first rank the components associated to active variables according to their variances. Then we can select the number of components. Several methods can be used for this purpose. For instance, one can use simple threshold method, cross validation, the information criteria, or a combination of them.

We remark that, if the number of relevant variables is much smaller ($p_* \ll p$), the computational complexity of post-training component selection is low because the number of the components associated to relevant variables is only of order $O(p_*^d)$, much less than the number of components before variable selection.

3. Connections with Existing Algorithms.

The idea of parameterizing coordinates in kernel method for modeling nonlinear data structure and selecting relevant factors are not completely new. For instance, the parametrized polynomial and Gaussian kernels are used in [12]. However, this idea is conceptually different from CKPM. To see this, consider the parametrized Gaussian kernel

$$K_G(\mathbf{x}, \mathbf{t}) = \exp\left(-\sum_{\ell=1}^{p} w_\ell (x_\ell - t_\ell)^2\right).$$

Clearly, although sparse coding techniques allow to filter out the irrelevant variables, the interactive relationship between the relevant variables is still mixed together. Therefore it is impossible to explicitly represent each component as in CKPM, let alone selecting those effective interactive components.

The CKPM is closely related to COSSO [13, 32]. Both methods are motivated from the smoothing spline component decomposition model

$$f(\mathbf{x}) = b + \sum_{\ell=1}^{p} f_\ell(x_\ell) + \sum_{k,\ell=1}^{p} f_{k\ell}(x_k, x_\ell) + \text{higher order interactive terms.}$$

In COSSO each univariate function f_ℓ is assumed to be in a reproducing kernel Hilbert spaces H_ℓ of variable x_ℓ and the interactive components are in corresponding tensor product spaces $\mathcal{H}_\mathbf{s} = \bigotimes_{\ell \in \mathbf{s}} H_\ell$. COSSO searches for a function $\hat{f}$ by minimizing the risk functional

$$\hat{f}_{\text{COSSO}} = \arg\min \left\{ \frac{1}{m} \sum_{i=1}^{m} (y_i - f(\mathbf{x}_i))^2 + \lambda \sum_{|\mathbf{s}| \le d} \|P_\mathbf{s} f\|_{\mathcal{H}_\mathbf{s}} \right\},$$

where $P_{\mathbf{s}}$ is the projection onto the tensor product space. It is equivalent to

$$\begin{aligned} \hat{f}_{\text{COSSO}} = \arg\min \quad & \left\{ \frac{1}{m}\sum_{i=1}^{m}(y_i - f(\mathbf{x}_i))^2 + \lambda \sum_{|\mathbf{s}|\le d} \theta_{\mathbf{s}}^{-1}\|f_{\mathbf{s}}\|^2_{\mathcal{H}_{\mathbf{s}}} + \mu \sum_{|\mathbf{s}|\le d}\theta_{\mathbf{s}} \right\}, \\ \text{s.t.} \quad & f = b + \sum_{\mathbf{s}} f_{\mathbf{s}}, \qquad \theta_{\mathbf{s}} \ge 0. \end{aligned}$$

Let $R_{\mathbf{s}}$ stand for the kernel for $\mathcal{H}_{\mathbf{s}}$ and $R = \sum_{|\mathbf{s}|\le d}\theta_{\mathbf{s}}R_{\mathbf{s}}$. The kernel form of COSSO is

$$\begin{aligned} \hat{f}_{\text{COSSO}} = \arg\min \quad & \left\{ \frac{1}{m}\sum_{i=1}^{m}(y_i - f(\mathbf{x}_i))^2 + \lambda \|f_0\|^2_{R} + \mu \sum_{|\mathbf{s}|\le d}\theta_{\mathbf{s}} \right\}, \\ \text{s.t.} \quad & f = b + f_0, \;\; \theta_{\mathbf{s}} \ge 0. \end{aligned} \tag{14}$$

At the first glance, one may think (4) and (14) are quite similar. But this is only an illusion. The essential difference between these two algorithms lies on their different ideas for designing the kernels. In COSSO a linear combination of the kernels for all components is used. As a result, the number of parameters is equal to the number of components that is of order $O(p^d)$. It increases geometrically with d and may cause computational difficulty even in a moderate dimensional problem if modeling multiway interactions is necessary. In coordinate kernel models, however, the number of weight parameters is always equal to the dimensionality p and does not increase with d. This makes the model computationally feasible when it is necessary to consider high order interactions.

Finally, we would like to mention that learning the optimal weight vector $\boldsymbol{\omega}$ in CKPM is not only a variable selection process, but also a kernel learning process because the impact of the active variables is usually different. We would particularly mention the methods developed in [5], which, given a set of base kernels, learns an optimal polynomial combination of the base kernels. Coordinate kernel polynomials can be regarded as special cases where the coordinate kernels are selected as base kernels. In this sense the first stage of our algorithm, (4), shares great similarity with the methods in [5].

Differences exist though. The first difference is a conceptual one. Our algorithm is motivated from the component decomposition model and the purpose is to model and select the effective interactive components. In our approach the coordinate kernel polynomial serves as a tool to expand the interactive components. This is unlike the methods in [5] for which learning the nonlinear combination of base kernels is the aim. This difference becomes clear by noticing that, when the base kernels are not coordinate kernels, the resulted kernels can still apply to the methods in [5] but become useless for our purpose. A more essential difference lies in the reparametrization refinement stage, which makes our algorithm novel. Without this stage, although the algorithm also helps improve the predictive performance, the variable selection accuracy and learning performance is suboptimal.

4. Generalization Error Bounds.

In this section we provide the generalization error bound of order $O(\frac{d\log(p)}{\sqrt{m}})$ for CKPR. This explains the effectiveness of CKPR in high dimensional setting, even with dimensionality larger than the sample size.

Let $\mathrm{er}(f)$ and $\mathrm{er}_m(f)$ be the mean squared error and the sample mean squared error, respectively,

$$\mathrm{er}(f) = \mathbb{E}[(y - f(\mathbf{x}))^2], \qquad \mathrm{er}_m(f) = \frac{1}{m}\sum_{i=1}^{m}(y_i - f(\mathbf{x}_i))^2.$$

Note that $\mathrm{er}(f)$ measures the predictive power of the function f.

Assume the coordinate kernels are uniformly bounded, i.e.

$$\kappa = \sup_{\ell\in\{1,2,\ldots,p\}} \sup_{x_\ell} K_\ell(x_\ell, x_\ell) < \infty.$$

Then

$$\sup_{\mathbf{x}} K_{\boldsymbol{\omega}}(\mathbf{x},\mathbf{x}) \le \kappa^d \left(\sum_{\ell=1}^{p} w_\ell\right)^d,$$

and for any $f \in \mathcal{H}_{K_{\boldsymbol{\omega}}}$,

$$\|f\|_\infty \le \kappa^d \left(\sum_{\ell=1}^{p} w_\ell\right)^d \|f\|_{K_{\boldsymbol{\omega}}}. \tag{15}$$

Assume $|y_i| \le M$ almost surely. We have the following two theorems.

Theorem 4.1. *Let $\hat{f}_{\lambda,\mu}$ be the solution to the CKPR (4). For any $0 < \delta < 1$, there holds*

$$\mathrm{er}(\hat{f}_{\lambda,\mu}) \le \mathrm{er}_m(\hat{f}_{\lambda,\mu}) + \left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right)^2 \left(40\sqrt{\frac{e\log(p^d+1)}{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}}\right).$$

with probability $1-\delta$.

Theorem 4.2. *Let $\hat{f}_{\tilde{\lambda},\gamma}$ be the refined solution given by (13). For any $0 < \delta < 1$, there holds*

$$\mathrm{er}(\hat{f}_{\tilde{\lambda},\gamma}) \le \mathrm{er}_m(\hat{f}_{\tilde{\lambda},\gamma}) + \left(M + \frac{\kappa^d M}{\sqrt{\gamma}}\right)^2 \left(40\sqrt{\frac{e\log(p^d+1)}{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}}\right).$$

with probability $1-\delta$.

5. Simulation Studies.

The purpose of this section is to illustrate the effectiveness of our algorithm by simulations on synthetic data sets and applications on real data sets. Comparisons will be made with several state-of-the-art methods. Before we describe our simulations in detail, we remark that the two synthetic models are taken from [11] and [13] respectively, not specially designed to favor the use of our algorithm. Though, our algorithm shows to be very successful.

In our simulations we focus on the use of linear coordinate kernels $K_\ell^L(\mathbf{x},\mathbf{t}) = 1 + x_\ell t_\ell$ and Gaussian coordinate kernels $K_\ell^G(\mathbf{x},\mathbf{t}) = \exp\left(-\frac{(x_\ell - t_\ell)^2}{2\sigma_\ell^2}\right)$. We refer to our method as CKPR-L and CKPR-G respectively. We do not consider three-way or higher order interactions in this paper, so the degree in the CKP will be fixed to $d = 2$. Also, we fixed the bandwidth parameter when Gaussian coordinate kernels are used. They are set to be $\sigma_\ell = 2$ if x_ℓ is normalized to have mean zero and standard deviation 1, and $\sigma_\ell = 0.3$ if x_ℓ lies in the interval $[0,1]$. These parameters were found to work well in many problems, although they may not be optimal. The two regularization parameters $\tilde{\lambda}$ and γ are selected by five-fold cross validation.

5.1. **A synthetic example with sample size less than dimension.** This example comes from [11]. The data sets are generated from the model

$$y = x_1 + x_1 x_2 + \epsilon$$

where $\mathbf{x} \in \mathbb{R}^{200}$, all components of $\mathbf{x}$ are independent standard normal, and ϵ is zero mean normal noise with the variance set such that the signal to noise ratio equal to 20, (i.e. $\mathrm{Var}(\mathbf{E}(y|\mathbf{x}))/\mathrm{Var}(\epsilon) = 20$.) The sample size is $m = 100$, making it a typical "sample size less than dimension" problem.

As in [11], we employ two measures, the true positive rate (TPR), which is defined as the ratio of the number of correctly identified active predictors to the number of truly active predictors, and the false positive rate (FPR), which is defined as the ratio of the number of falsely identified active predictors to the total number of inactive predictors, to measure the performance of variable selection. An effective variable selection algorithm is expected to have TPR to be close to 1 and FPR to be close to 0 at the same time. We also consider the predictive power of the our algorithm, which is measured by the mean squared error (MSE) on 1000 testing points.

We replicate the simulation 100 times. Table 1 reports the average of TPR, FPR, and the mean and standard error of the MSEs. As a comparison, we also reported the performance of LASSO [23], COSSO [13], and SR-SIR [11].

We see that CKPR-L and CKPR-G perform much better than the other three methods in both variable selection accuracy and predictive power. Since the coordinate kernel polynomial with linear coordinate kernels and degree $d = 2$ is exactly a polynomial kernel of degree 2, it captures the true structure of the model and therefore performs the best. When Gaussian coordinate kernels are used, the polynomial structure could not be exactly captured, but can be well approximated. Thus, the performance of CKPR-G is also very good.

It has already been noticed in [11] that all OLS-based methods, including LASSO and elastic net, are not able to identify the second predictor x_2 successfully. Due to the loss of the active variable x_2 the predictive power of LASSO is poor.

The results of COSSO are obtained by the R package "`cosso`". It was not able to identify the variable x_2 and thus the predictive power is also poor.

The results for SR-SIR is taken from [11] directly. Theoretically SR-SIR can identify x_2 successfully. Empirically its performance varies depending on different parameter selection strategies. As a dimension reduction method, SR-SIR does not output a predictive model.

Algorithm	TPR(x_1)	TPR(x_2)	FPR	MSE
CKPR-L	1.00	1.00	0.000	0.008 (0.000)
CKPR-G	1.00	1.00	0.011	0.109 (0.015)
LASSO	1.00	0.18	0.040	1.129 (0.015)
COSSO	0.90	0.02	0.020	10.879 (8.345)
SR-SIR (AIC)	1.00	0.89	0.460	-
SR-SIR (BIC)	1.00	0.85	0.181	-
SR-SIR (RIC)	1.00	0.75	0.053	-

TABLE 1. Variable selection accuracy and average MSE for Example 1.

5.2. **A synthetic example with several two way interactions.** The second synthetic example comes from [13] which was designed to favor the use of COSSO. Let $g_1(t) = t$, $g_2(t) = (2t-1)^2$, $g_3(t) = \frac{\sin(2\pi t)}{2-\sin(2\pi t)}$, and $g_4(t) = 0.1\sin(2\pi t) + 0.2\cos(2\pi t) + 0.3\sin^2(2\pi t) + 0.4\cos^3(2\pi t) + 0.5\sin^3(2\pi t)$ be four univariate functions. Consider the 10-dimensional regression problem with several two-way interactions:

$$y = g_1(x_1) + g_2(x_2) + g_3(x_3) + g_4(x_4) + g_1(x_3x_4) + g_2(\tfrac{x_1+x_3}{2}) + g_3(x_1x_2) + \epsilon$$

where, as in [13], all components of $\mathbf{x}$ are independent uniform variables and the noise is set to be normal with standard deviation 0.2546. The measure of the predictive accuracy is the MSE on 1,000 testing points. Again, the simulations are run 100 times and we reported the average MSE in Table 2 for sample size $m = 100,\ 200,\ 400$. For comparison purposes, we included the results for COSSO and MARS [6] which were copied from [13]. All three methods are able to take two-way interactions into consideration while CKPR-G performs the best. CKPR-G also shows to be powerful in variable selection. It almost always captures the active variables while the FPR drops quickly as sample size increases (FPR=0.56 for $m = 100$, 0.24 for $m = 200$, and 0.04 for $m = 400$).

	$m = 100$	$m = 200$	$m = 400$
CKPR-G	0.119 (0.003)	0.054 (0.001)	0.025 (0.0004)
COSSO(GCV)	0.358 (0.009)	0.100 (0.003)	0.045 (0.001)
COSSO(5CV)	0.378 (0.005)	0.094 (0.004)	0.043 (0.001)
MARS	0.239 (0.008)	0.109 (0.003)	0.084 (0.001)

TABLE 2. Average and standard error of MSEs for Example 2

5.3. **UCI data sets.** We applied CKPR to the three real data sets, the Johns Hopkins University Ionosphere data, the Sonar, Mines vs. Rocks data, and the Wisconsin Breast Cancer data. They are available on the *UCI Machine Learning Repository* website (`http://archive.ics.uci.edu/ml/`). For each dataset, features were standardized. In the case of classification dataset, the labels were set to ± 1. We randomly select 50% of the data for training and tuning, and test on the remaining 50% of the data. The experiment are repeated 30 times and the root mean squared error (RMSE) was reported in Table 3. As a comparison, we also reported the best results obtained in [5]. The CKPR with reparametrization refinement and component selection shows to be competitive.

	Ionosphere	**Sonar MR**	**Wisc. BC**
n	351	208	683
p	33	60	9
CKPR-L	0.64(0.04)	0.75(0.06)	0.34(0.02)
CKPR-G	0.54(0.03)	0.77(0.06)	0.34(0.02)
Best in [5]	0.60(0.05)	0.80(0.04)	0.70(0.01)

TABLE 3. RMSE on three UCI data sets.

6. **Conclusion and Discussions.** In this paper we have proposed the use of coordinate kernel polynomials in kernel machines. By the aid of ℓ_1 penalty, the coordinate kernel polynomial regression can be used to simultaneously identify active variables and components. Reparametrization refinement is introduced to improve the modeling accuracy and learning performance. A post-training selection process by AIC or BIC is suggested to select the effective components.

The CKPR could be cast into the multiple kernel learning framework, which allows us to prove a generalization error bound of order $O(\frac{d \log p}{\sqrt{m}})$ using the idea of Rademacher chaos complexity. This provides a theoretical justification for its predictive power, even in the "sample size less than dimensionality" setting. Simulation studies are used to verify the efficiency of the algorithm empirically.

As generalization error bounds help to verify the learning performance from a predictive perspective, we are also concerned with the modeling consistency of CKPM, including both the variable selection consistency and the interactive component selection consistency. They may be interesting topics for future research.

REFERENCES

[1] F. Bach, Consistency of the group lasso and multiple kernel learning, *Journal of Machine Learning Research*, **9** (2008), 1179–1225.

[2] P. L. Bartlett and S. Mendelson, Rademacher and Gaussian complexities: risk bounds and structural results, *Journal of Machine Learning Research*, **3** (2003), 463–482.

[3] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*, Cambridge University Press, 2003.

[4] C. M. Carvalho, J. Chang, J. E. Lucas, J. R. Nevins, Q. Wang and M. West, High-dimensional sparse factor modeling: applications in gene expression genomics, *Journal of the American Statistical Association*, **103** (2008), 1438–1456.

[5] C. Cortes, M. Mohri and A. Rostamizadeh, Learning non-linear combinations of kernels, in *Advances in Neural Information Processing Systems*, 2009, 396–404.

[6] J. H. Friedman, Multivariate adaptive regression splines, *The Annals of Statistics*, **19** (1991) 1–67.

[7] I. Guyon, J. Weston, S. Barnhill and V. Vapnik, Gene selection for cancer classification using support vector machines, *Machine Learning*, **46** (2002), 389–422.

[8] R. Kohavi and G. John, Wrappers for feature selection, *Artificial Intelligence*, **97** (1997), 273–324.

[9] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui and M. I. Jordan, Learning the kernel matrix with semidefinite programming, *Journal of Machine Learning Research*, **5** (2004), 27–72 (electronic).

[10] S. L. Lauritzen, *Graphical models*, Oxford University Press, 1996.

[11] L. Li and X. Yin, Sliced inverse regression with regularizations, *Biometrics*, **64** (2008), 124–131.

[12] F. Liang, K. Mao, M. Liao, S. Mukherjee and M. West, *Nonparametric Bayesian kernel models*, Technical report, Department of Statistical Science, Duke University, Discussion Paper (2007), 07-10

[13] Y. Lin and H. Zhang, Component selection and smoothing in multivariate nonparametric regression, *The Annals of Statistics*, **34** (2006), 2272–2297.

[14] C. McDiarmid, *On the method of bounded differences*, 148–188, Cambridge University Press, 1989.

[15] R. Meir and T. Zhang, Generalization error bounds for Bayesian mixture algorithms, *Journal of Machine Learning Research*, **4** (2003), 839–860.

[16] S. Mukherjee and Q. Wu, Estimation of gradients and coordinate covariation in classification., *Journal of Machine Learning Research*, **7** (2006), 2481–2514.

[17] S. Mukherjee and D. Zhou, Learning coordinate covariances via gradients, *Journal of Machine Learning Research*, **7** (2006), 519–549.

[18] M. Pontil and C. Micchelli, Learning the kernel function via regularization, *IEEE Transcations on Neural Networks*, **15** (2004), 45–54.

[19] H. Qin and X. Guo, Semi-supervised learning with summary statistics, *Analysis and Applications*, **17** (2019), 837–851.
[20] B. Schölkopf, A. Smola and K. Müller, Kernel principal component analysis, *Artificial Neural Networks–ICANN'97*, (1997), 583–588.
[21] L. Shi, Distributed learning with indefinite kernels, *Analysis and Applications*, **17** (2019), 947–975.
[22] T. Speed and H. Kiiveri, Gaussian markov distributions over finite graphs, *The Annals of Statistics*, 138–150.
[23] R. Tibshirani, Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **58** (1996), 267–288.
[24] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.
[25] G. Wahba, *Spline models for observational data*, vol. 59, Society for Industrial Mathematics, 1990.
[26] Q. Wang and X. Yin, A nonlinear multi-dimensional variable selection method for high dimensional data: Sparse mave, *Computational Statistics & Data Analysis*, **52** (2008), 4512–4520.
[27] Q. Wu, Y. Ying and D. Zhou, Multi-kernel regularized classifiers, *Journal of Complexity*, **23** (2007), 108–134.
[28] Y. Xu and H. Zhang, Refinable kernels, *Journal of Machine Learning Research*, **8** (2007), 2083–2120.
[29] Y. Xu and H. Zhang, Refinement of reproducing kernels, *Journal of Machine Learning Research*, **10** (2009), 107–140.
[30] W. Yao and Q. Wang, Robust variable selection through mave, *Computational Statistics & Data Analysis*, **63** (2013), 42–49.
[31] Y. Ying and C. Campbell, Rademacher chaos complexities for learning the kernel problem, *Neural computation*, **22** (2010), 2858–2886.
[32] H. H. Zhang, Variable selection for support vector machines via smoothing spline anova, *Statistica Sinica*, **16** (2006), 659–674.
[33] N. Zhang, Z. Yu and Q. Wu, Overlapping sliced inverse regression for dimension reduction, *Analysis and Applications*, **17** (2019), 715–736.
[34] H. Zou and T. Hastie, Regularization and variable selection via the elastic net, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67** (2005), 301–320.

Appendix A. Proof of Theorem 4.1 and Theorem 4.2.

Proof of Theorem 4.1. Let $\hat{f}_{\lambda,\mu}$ be the solution to the CKPR and the optimal weight vector is $\boldsymbol{\omega}^*$. Recall the assumption that $|y_i| \le M$ almost surely for $1 \le i \le m$, one has

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{f}_{\lambda,\mu}(\mathbf{x}_i))^2 + \lambda\|\hat{f}_{\lambda,\mu}\|^2_{K_{\boldsymbol{\omega}^*}} + \mu\sum_{\ell=1}^{p} w^*_\ell \le M^2,$$

so

$$\hat{f}_{\lambda,\mu} \in \mathcal{F}_{\lambda,\mu} := \left\{ f : \ f \in \mathcal{H}_{K_{\boldsymbol{\omega}}}, \ \|f\|_{K_{\boldsymbol{\omega}}} \le \frac{M}{\sqrt{\lambda}}, \ \sum_{i=1}^{p} w_\ell \le \frac{M^2}{\mu} \right\}.$$

Therefore

$$\mathrm{er}(\hat{f}_{\lambda,\mu}) - \mathrm{er}_m(\hat{f}_{\lambda,\mu}) \le \sup_{f\in\mathcal{F}_{\lambda,\mu}} \mathrm{er}(f) - \mathrm{er}_m(f). \tag{16}$$

Using (15) we obtain

$$\|f\|_\infty \le \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d} \text{ for all } f \in \mathcal{F}_{\lambda,\mu}.$$

Thus

$$(y - f(\mathbf{x}))^2 \le \left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right)^2 \text{ for all } f \in \mathcal{F}_{\lambda,\mu}.$$

When an $(\mathbf{x}_i, y_i)$ pair changes, the random variable $\sup_{f\in\mathcal{F}_{\lambda,\mu}} \mathrm{er}(f) - \mathrm{er}_m(f)$ can change by no more than $\frac{1}{m}\left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right)^2$. Applying the McDiarmid's bounded difference inequality [14], we have with probability $1-\delta$

$$\sup_{f\in\mathcal{F}_{\lambda,\mu}} \mathrm{er}(f) - \mathrm{er}_m(f) \le \mathbb{E}\left[\sup_{f\in\mathcal{F}_{\lambda,\mu}} \mathrm{er}(f) - \mathrm{er}_m(f)\right] + \left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right)^2 \sqrt{\frac{\ln(1/\delta)}{2m}}. \tag{17}$$

For $i = 1, \dots, m$, let ε_i's be independent Bernoulli random variables such that $P(\varepsilon_i = 1) = P(\varepsilon = -1) = 1/2$, and that ε_i's are independent of the sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$. Using the standard symmetrization technique and the properties of Rademacher complexity [15, 2], it is not difficult to show that

$$\begin{aligned}
&\mathbb{E}\left[\sup_{f\in\mathcal{F}_{\lambda,\mu}} \mathrm{er}(f) - \mathrm{er}_m(f)\right] \\
\le\ & 2\mathbb{E}\left[\sup_{f\in\mathcal{F}_{\lambda,\mu}} \frac{1}{m}\sum_{i=1}^m \varepsilon_i (y_i - f(\mathbf{x}_i))^2\right] \\
\le\ & 8\left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right) \mathbb{E}\left[\sup_{f\in\mathcal{F}_{\lambda,\mu}} \frac{1}{m}\sum_{i=1}^m \varepsilon_i f(\mathbf{x}_i)\right] \\
=\ & 8\left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right) \mathbb{E}\left[\sup_{\sum_{\ell=1}^p w_\ell \le \frac{M^2}{\mu}} \sup_{\|f\|_{K_{\boldsymbol\omega}} \le \frac{M}{\sqrt{\lambda}}} \left\langle f, \frac{1}{m}\sum_{i=1}^m \varepsilon_i K_{\boldsymbol\omega}(\cdot, \mathbf{x}_i)\right\rangle_{K_{\boldsymbol\omega}}\right] \\
=\ & 8\left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right) \mathbb{E}\left[\sup_{\sum_{\ell=1}^p w_\ell \le \frac{M^2}{\mu}} \frac{M}{\sqrt{\lambda}} \sqrt{\frac{1}{m^2}\sum_{i,j=1}^m \varepsilon_i\varepsilon_j K_{\boldsymbol\omega}(\mathbf{x}_i, \mathbf{x}_j)}\right] \\
=\ & 8\left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu^d}\right) \frac{M}{\sqrt{\lambda}} \left(\frac{M^2}{\mu}\right)^d \mathbb{E}\left[\sup_{\sum_{\ell=1}^p w_\ell = 1} \sqrt{\frac{1}{m^2}\sum_{i,j=1}^m \varepsilon_i\varepsilon_j K_{\boldsymbol\omega}(\mathbf{x}_i, \mathbf{x}_j)}\right] \\
\le\ & 8\left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu}\right) \frac{M^{2d+1}}{\sqrt{\lambda}\mu^d} \left(\mathbb{E}\left[\sup_{\sum_{\ell=1}^p w_\ell = 1} \frac{1}{m^2}\sum_{i,j=1}^m \varepsilon_i\varepsilon_j K_{\boldsymbol\omega}(\mathbf{x}_i, \mathbf{x}_j)\right]\right)^{1/2}
\end{aligned} \tag{18}$$

The last term in (18) is closely related to the Rademacher chaos complexity which can be bounded using the idea in [31]. To this end, let $\boldsymbol\eta = (\eta_1, \eta_2, \dots, \eta_d) \in \{1, 2, \dots, p\}^d$ be a vector of indices,

$$\boldsymbol\omega_{\boldsymbol\eta} = \prod_{i=1}^d w_{\eta_i}, \text{ and } K_{\boldsymbol\eta}(\mathbf{x}, \mathbf{t}) = \prod_{i=1}^d K_{\eta_i}(x_{\eta_i}, t_{\eta_i}).$$

Let $\mathcal{K}$ be the collection of all kernels $K_{\boldsymbol{\eta}}$. Then $\mathcal{K}$ contain p^d kernels and are uniformly bounded by κ^d. By [31, Corollary 1],

$$\mathbb{E} \sup_{K\in\mathcal{K}} \frac{1}{m} \sum_{i,j=1}^{m} \varepsilon_i \varepsilon_j K(\mathbf{x}_i, \mathbf{x}_j) \le 25e\kappa^{2d} \log(p^d + 1).$$

Note that if $\sum_{\ell=1}^p w_\ell = 1$, we have

$$\sum_{\boldsymbol{\eta}} \boldsymbol{\omega}_{\boldsymbol{\eta}} = \left(\sum_{\ell=1}^p w_\ell \right)^d = 1 \quad \text{and} \quad K_{\boldsymbol{\omega}} = \sum_{\boldsymbol{\eta}} \boldsymbol{\omega}_{\boldsymbol{\eta}} K_{\boldsymbol{\eta}}.$$

Thus $\{K_{\boldsymbol{\omega}} : \sum_{\ell=1}^p w_\ell = 1\}$ is the convex hull of $\mathcal{K}$. By the discussion in [31, Section 2.1], we obtain

$$\mathbb{E} \left[\sup_{\sum_{\ell=1}^p w_\ell = 1} \frac{1}{m} \sum_{i,j=1}^{m} \varepsilon_i \varepsilon_j K_{\boldsymbol{\omega}}(\mathbf{x}_i, \mathbf{x}_j) \right] \le 25e\kappa^{2d} \log(p^d + 1).$$

Plugging this estimate into (18) we have

$$\mathbb{E} \left[\sup_{f\in\mathcal{F}_{\lambda,\mu}} \operatorname{er}(f) - \operatorname{er}_m(f) \right] \le 8 \left(M + \frac{\kappa^d M^{2d+1}}{\sqrt{\lambda}\mu} \right) \frac{M^{2d+1}}{\sqrt{\lambda}\mu} \sqrt{\frac{25e\kappa^{2d} \log(p^d+1)}{m}}. \tag{19}$$

Combining the estimates in (16), (17), and (19) proves the desired error bound. □

By the fact that the solution to the two-stage algorithm satisfies

$$\hat{f}_{\tilde{\lambda},\gamma} \in \mathcal{F}_\gamma = \left\{ f \in \mathcal{H}_{K_{\boldsymbol{\omega}}},\ \|f\|_{K_{\boldsymbol{\omega}}} \le \frac{M}{\sqrt{\gamma}},\ \sum_{\ell=1}^p w_\ell = 1 \right\}$$

Theorem 4.2 can be proved by a very similar process. We omit the details.

It is not surprising that $\hat{f}_{\tilde{\lambda},\gamma}$ depends on the parameter $\tilde{\lambda}$ while the generalization error bound is independent of $\tilde{\lambda}$ because the bound is obtained by placing $\hat{f}_{\tilde{\lambda},\gamma}$ in a very big function class $\mathcal{F}_\gamma$. Refined error bounds may be possible if we can find a smaller function class that contains $\hat{f}_{\tilde{\lambda},\gamma}$ by using the information on $\boldsymbol{\omega}^*$.

E-mail address: x.guo@polyu.edu.hk
E-mail address: lexinli@berkeley.edu
E-mail address: qiang.wu@mtsu.edu