Journal of Machine Learning Research 21 (2020) 1-39 Submitted 10/19; Revised 7/20; Published 10/20

Learning Data-adaptive Non-parametric Kernels

Fanghui Liu * FANGHUI.LIU@KULEUVEN.BE
Department of Electrical Engineering, ESAT-STADIUS, KU Leuven, B-3001, Belgium

Xiaolin Huang XIAOLINHUANG@SJTU.EDU.CN
Institute of Image Processing and Pattern Recognition
Institute of Medical Robotics, Shanghai Jiao Tong University, Shanghai, 200240, China

Chen Gong CHEN.GONG@NJUST.EDU.CN
PCA Lab, Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education
School of Computer Science and Engineering, Nanjing University of Science and Technology, 210094, China
Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR, China.

Jie Yang JIEYANG@SJTU.EDU.CN
Institute of Image Processing and Pattern Recognition
Institute of Medical Robotics, Shanghai Jiao Tong University, Shanghai, 200240, China

Li Li LI-LI@TSINGHUA.EDU.CN
Department of Automation, BNRist, Tsinghua University, 100084, China

Editor: John Shawe-Taylor

Abstract

In this paper, we propose a data-adaptive non-parametric kernel learning framework in margin based kernel methods. In model formulation, given an initial kernel matrix, a data-adaptive matrix with two constraints is imposed in an entry-wise scheme. Learning this data-adaptive matrix in a formulation-free strategy enlarges the margin between classes and thus improves the model flexibility. The introduced two constraints are imposed either exactly (on small data sets) or approximately (on large data sets) in our model, which provides a controllable trade-off between model flexibility and complexity with theoretical demonstration. In algorithm optimization, the objective function of our learning framework is proven to be gradient-Lipschitz continuous. Thereby, kernel and classifier/regressor learning can be efficiently optimized in a unified framework via Nesterov's acceleration. For the scalability issue, we study a decomposition-based approach to our model in the large sample case. The effectiveness of this approximation is illustrated by both empirical studies and theoretical guarantees. Experimental results on various classification and regression benchmark data sets demonstrate that our non-parametric kernel learning framework achieves good performance when compared with other representative kernel learning based algorithms.

Keywords: support vector machines, non-parametric kernel learning, gradient-Lipschitz continuous

1. Introduction

Kernel methods (Shawe-Taylor and Cristianini, 2000; Schölkopf and Smola, 2003; Steinwart and Andreas, 2008) have proven to be powerful in a variety of machine learning tasks, e.g., Support Vector Machines (SVM) for classification (Vapnik, 1995; Suykens et al., 2002), Support Vector Regression

∗. The bulk of this work was performed while Fanghui was a PhD student at Shanghai Jiao Tong University.

(SVR) for regression (Drucker et al., 1997), and kernel mean embedding for casual inference (Mitrovic et al., 2018). They employ a so-called *kernel* function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to compute the similarity between any two samples $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in \mathbb{R}^d$ such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}$, where $\phi : \mathcal{X} \to \mathcal{H}$ is a non-linear feature map transforming elements of input spaces $\mathcal{X}$ into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. Specifically, for a given kernel, the "kernel trick" allows for optimization in the kernel-associated hypothesis space without explicit representation of such mapping.

Generally, the performance of kernel methods largely depends on the choice of the kernel. Traditional kernel methods often adopt a classical kernel, e.g., Gaussian kernel or sigmoid kernel for characterizing the relationship among data points. Empirical studies suggest that these traditional kernels are not sufficiently flexible to depict the domain-specific characteristics of data affinities or relationships. To address such limitation, several routes have been explored. One way is to design sophisticated kernels on specific tasks such as applying optimal assignment kernels (Kriege et al., 2016) to graph classification, developing a kernel based on triplet comparisons (Kleindessner and von Luxburg, 2017), designing the class of tessellated kernels (Colbert and Peet, 2020), or even breaking the restriction of positive definiteness on kernels (Ong et al., 2004; Schleif and Tino, 2015; Loosli et al., 2016). Apart from these well-designed kernels, a series of research studies aim to automatically learn effective and flexible kernels from data, known as *learning kernels*. Algorithms for learning kernels can be roughly grouped into two categories: parametric kernel learning and non-parametric kernel learning.

1.1 Review of Kernel Learning

In parametric kernel learning, the (learned) kernel function $k(\cdot, \cdot)$ or the kernel matrix $\boldsymbol{K} = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{n \times n}$ on the training data is assumed to admit a specific parametric form, and then the relevant parameters are learned according to the given data. The earliest work is proposed by Lanckriet et al. (2004), in which they consider training SVM along with optimizing a linear combination of several pre-given positive semi-definite (PSD) matrices $\{\underline{\boldsymbol{K}}_t\}_{t=1}^s$ subject to a bounded trace constraint, i.e., $\boldsymbol{K} = \sum_{t=1}^s \mu_t \underline{\boldsymbol{K}}_t$ with $\mathrm{tr}(\boldsymbol{K}) \leq c$. Specifically, to ensure the learned kernel matrix to be PSD, one can directly use the constraint $\boldsymbol{K} \in \mathcal{S}_+^n$ (the cone of $n \times n$ positive semi-definite matrices); or consider a nonnegaitve linear combination of $\{\underline{\boldsymbol{K}}_t\}_{t=1}^s$, i.e., $\mu_t \geq 0$. Based on the above two schemes, the kernel (matrix) learning is transformed to learn the combination weights. Accordingly, the parameters in SVM and the weights in their model can be learned by solving a semi-definite programming optimization problem in a unified framework.

The above parametric kernel learning framework spawns the new field of *multiple kernel learning* (MKL) (Bach et al., 2004; Varma and Babu, 2009; Liu et al., 2019). It aims to learn a good combination of some predefined kernels (or kernel matrices) for rich representations. For example, the weight vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_s]^\top$ can be restricted by the conic sum (i.e., $\mu_t \geq 0$), the convex sum (i.e., $\mu_t \geq 0$ and $\sum_{t=1}^s \mu_t = 1$), or various regularizers such as ℓ_1 norm, mixed norm, and entropy-based formulations, see a survey (Gönen and Alpaydın, 2011). By doing so, MKL would generate a "broader" kernel to enhance the representation ability for data. Based on the idea of MKL, there are several representative approaches to learn effective kernels by exploring the data information, including: i) hierarchical kernel learning (HKL) (Bach, 2008; Jawanpuria et al., 2015) learns from a set of base kernels assumed to be embedded on a directed acyclic graph; ii) spectral mixture models (Argyriou et al., 2005; Wilson and Adams, 2013; Jean et al., 2018) aim to learn the

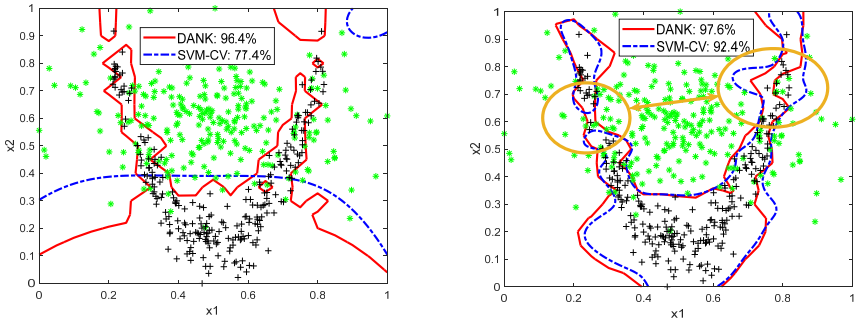
spectral density of a kernel in a parametric scheme for discovering flexible statistical representations in data; iii) kernel target alignment (Cristianini et al., 2002; Cortes et al., 2012) seeks for the "best" kernel matrix by maximizing the similarity between $\boldsymbol{K}$ and the ideal kernel $\boldsymbol{y}\boldsymbol{y}^\top$ with the label vector $\boldsymbol{y}$. Here the used ideal kernel can directly recognize the training data with 100% accuracy, and thus can be used to guide the kernel learning task. Current works on this direction often assume that the learned kernel matrix $\boldsymbol{K}$ is in a parametric way, e.g., an MKL or mixtures of spectral density form, see (Lanckriet et al., 2004; Sinha and Duchi, 2016; Li et al., 2019) and references therein.

Instead of assuming specific forms for the learned kernel in parametric kernel learning, nonparametric kernel learning is another way to acquire a positive definite kernel (matrix) in a data-specific manner. Typical examples include: Lanckriet et al. (2004) directly consider $\boldsymbol{K} \in \mathcal{S}_+^n$ without extra parametric forms in their optimization problem, which results in a nonparametric kernel learning framework. Such nonparametric kernel learning model is further explored by learning a low-rank kernel matrix (Kulis et al., 2009), imposing the pairwise constraints with side/prior information (Hoi et al., 2007), or local geometry information (Lu et al., 2009). These non-parametric kernel learning based problems are usually solved by the standard semi-definite programming or an efficient saddle-point optimization algorithm (Zhuang et al., 2011). Besides, Jain et al. (2012) investigate the equivalence between non-parametric kernel learning and Mahalanobis metric learning, and accordingly propose a non-parametric model seeking for a PSD matrix $\boldsymbol{W}$ in a learned kernel $\phi(\boldsymbol{x})^\top \boldsymbol{W} \phi(\boldsymbol{x}')$ via the LogDet divergence.

1.2 Contributions

In this paper, we propose a **D**ata-**A**daptive **N**onparametric **K**ernel (DANK) learning framework that can be seamlessly embedded to support vector machines (SVM) and support vector regression (SVR). A low-rank data-adaptive matrix in a suitable solving space is learned to enlarge the margin between classes and effectively control the model complexity. Further, by virtue of the gradient-Lipschitz continuous property of the considered objective function, the learning task can be efficiently solved by a projected gradient method with Nesterov's acceleration. Different from previous non-parametric kernel learning models optimized by semi-definite programming, the employed optimization algorithm in our DANK model is efficient in large scale cases.

To be specific, in our DANK model, a low-rank matrix $\boldsymbol{F} \in \mathbb{R}^{n\times n}$ in a bounded feasible region is imposed on a pre-given kernel matrix $\boldsymbol{K}$ in a point-wise strategy, i.e., $\boldsymbol{F} \odot \boldsymbol{K}$, where $\odot$ denotes the Hadamard product between two matrices. This *formulation-free* strategy contributes to adequate model flexibility as a result. The used low-rank constraint and the bounded constraint restrict the degree of freedom of $\boldsymbol{F}$, of which the design scheme is independent of the pre-given kernel matrix. Thereby we can restrict the flexibility of $\boldsymbol{F}$ so as to control the model complexity in a clear way. Here we take a synthetic classification data set *clowns* to illustrate this controllable trade-off between the model complexity and flexibility. The initial kernel matrix $\boldsymbol{K}$ is given by the classical Gaussian kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/2\sigma^2)$ with the width σ. Figure 1 shows that, in the left panel, the baseline SVM with an inappropriate $\sigma = 1$ lacks model flexibility and cannot precisely adapt to the data. However, based on the same $\boldsymbol{K}$, by optimizing the adaptive matrix $\boldsymbol{F}$, our DANK model shows good flexibility to fit the complex data distribution, leading to a desirable decision surface. Comparably, in the right panel, even under the condition that σ is well tuned, the baseline SVM fails to capture the local property of the data (see the brown ellipses). Instead, by learning $\boldsymbol{F}$, our DANK model still yields a more accurate boundary than SVM to fit the data points. Specifically, the



(a) $\sigma = 1$ (pre-given) (b) $\sigma = 0.07$ (cross validation)

Figure 1: Classification boundaries of SVM (in blue dash line) and our DANK model (in red solid line) with the Gaussian kernel on the *clowns* data set under different kernel width values σ.

model complexity is controlled by the introduced low-rank and bounded constraints. We find that, the generated classification boundary in Figure 1 prohibits locally linearly separable, which avoids over-fitting by an extremely flexible decision surface (Torr, 2011).

The main contributions of this paper lie in the following three folds:

- In model formulation, we propose a data-adaptive nonparametric kernel learning framework termed "DANK" to enhance the model flexibility and data adaptivity, which is then seamlessly embedded to margin based kernel methods (SVM and SVR) for classification and regression tasks. Specifically, the introduced constraints are demonstrated to be effective on a controllable trade-off between the model flexibility and complexity;

- In algorithm optimization, the DANK model and the induced classification/regression model can be formulated as a max-min optimization problem in a unified framework. The related objective function is proven to be gradient-Lipschitz continuous, and thus can be directly solved by a projected gradient method with Nesterov's acceleration;

- In scalability issue, we propose a decomposition based approach to our model by omitting the non-separable low-rank constraint in large sample case. The effectiveness of our decomposition-based scalable approach is demonstrated by both theoretical and empirical studies.

Besides, the experimental results on several classification and regression benchmark data sets demonstrate the effectiveness of the proposed DANK framework over other representative kernel learning based methods.

This paper shares the basic ideas with our previous conference work (Liu et al., 2018) but is totally different in model formulation, algorithm optimization, and scaling in large sample cases. First, in model formulation, we impose an additional low-rank constraint and a bounded constraint on $\boldsymbol{F}$, which effectively control the model flexibility with theoretical demonstration. Apart from SVM to which our DANK model is embedded for classification, the proposed DANK model is also extended to SVR for regression tasks. Second, we develop a Nesterov's smooth method to solve the designed optimization problem, which requires the discussion on its gradient-Lipschitz continuous property. Third, we conduct a decomposition-based scalable approach on DANK with theoretical guarantees

and experimental validation for large scale situations. Lastly, we provide more experimental results on popular benchmarks.

1.3 Notation

We start with notations this paper.

Matrices, vectors and elements: We take $\boldsymbol{A}$, $\boldsymbol{a}$ to be a matrix and a vector, of which the entries are A_{ij} and a_i, respectively. Denote $\boldsymbol{I}_n$ as the $n \times n$ identity matrix, $\mathbf{0}$ as a zero matrix or vector with the appropriate size, and $\mathbf{1}_n$ as the n-dimensional vector of all ones.

Sets: The set $\{1, 2, \cdots, n\}$ is written as $[n]$. We call $\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_s\}$ an s-partition of $[n]$ if $\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_s = [n]$ and $\mathcal{V}_p \cap \mathcal{V}_q = \emptyset$ for $p \neq q$. Let $|\mathcal{V}|$ denote the cardinality of the set $\mathcal{V}$. We take the notation $\mathcal{S}^n$ as the set of $n \times n$ symmetric matrices and $\mathcal{S}^n_+$ as the $n \times n$ PSD cone.

Singular value decomposition (SVD): Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and its rank $r = \text{rank}(\boldsymbol{A})$, a (compact) singular value decomposition (SVD) is defined as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^r \sigma_i(\boldsymbol{A})\boldsymbol{u}_i\boldsymbol{v}_i^\top$, where $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}$ are an $n \times r$ column-orthogonal matrix, an $r \times r$ diagonal matrix with its diagonal element $\sigma_i(\boldsymbol{A})$, and a $d \times r$ column-orthogonal matrix, respectively. If $\boldsymbol{A}$ is PSD, then $\boldsymbol{U} = \boldsymbol{V}$. Accordingly, the singular value soft-thresholding operator is defined as $\mathcal{J}_\tau(\boldsymbol{A}) = \boldsymbol{U}_{\boldsymbol{A}}\mathcal{S}_\tau(\boldsymbol{\Sigma}_{\boldsymbol{A}})\boldsymbol{V}_{\boldsymbol{A}}^\top$ with the SVD: $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and the soft-thresholding operator is $\mathcal{S}_\tau(A_{ij}) = \text{sign}(A_{ij})\max(0, |A_{ij}| - \tau)$.

Matrix norms: We use four matrix norms in this paper

Frobenius norm: $\|\boldsymbol{A}\|_{\text{F}} = \sqrt{\sum_{i,j} A_{ij}^2} = \sqrt{\sum_i \sigma_i^2(\boldsymbol{A})}$.
Spectral norm: $\|\boldsymbol{A}\|_2 = \max_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{A}\boldsymbol{x}\|_2 = \sigma_{\max}(\boldsymbol{A}) \leq \|\boldsymbol{A}\|_{\text{F}}$.
Nuclear norm: $\|\boldsymbol{A}\|_* = \sum_i \sigma_i(\boldsymbol{A})$.
Any square matrix satisfies $\text{tr}(\boldsymbol{A}) \leq \|\boldsymbol{A}\|_*$. If $\boldsymbol{A}$ is PSD, then we have $\text{tr}(\boldsymbol{A}) = \|\boldsymbol{A}\|_*$ and $\|\boldsymbol{A}\|_2 = \lambda_{\max}(\boldsymbol{A})$, where $\lambda_{\max}(\boldsymbol{A})$ denotes the largest eigenvalue of $\boldsymbol{A}$.

1.4 Paper Organization

The paper is organized as follows. In Section 2, we introduce the proposed DANK model embedded in SVM, mainly on the model formulation in Section 2.1, benefits of the introduced constraints in Section 2.2, and out-of-sample extensions in Section 2.3. The model optimization is presented in Section 3: Section 3.1 studies the gradient-Lipschitz continuous property and Section 3.2 applies Nesterov's smooth optimization method to solve our model. Scalability of our nonparametric kernel model is addressed in Section 4. Besides, in Section 5, we extend our DANK model to SVR for regression. The experimental results on popular benchmark data sets are presented in Section 6. Section 7 concludes the entire paper. Proofs are provided in the Appendices.

2. The DANK Model in SVM

In this section, we incorporate the proposed DANK model into SVM for classification, discuss benefits of the introduced constraints, and extend it to test data for out-of-sample extension. Denote $\mathcal{X} \subseteq \mathbb{R}^d$ as a compact metric space, and $\mathcal{Y} = \{-1, 1\}$ as the label space, we assume that a sample set $\mathcal{Z} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is drawn from a non-degenerate Borel probability measure ρ on $\mathcal{X} \times \mathcal{Y}$. We focus on binary classification problems for the ease of description and it can be extended to multi-classification tasks.

2.1 Model Formulation

We begin with the SVM formulation and then introduce our DANK model in SVM. The hard-margin SVM aims to learn a linear classifier $f(\boldsymbol{x}; \boldsymbol{w}, b) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b) \in \{-1, +1\}$ with $\boldsymbol{w}$ and b that determine the decision hyperplane. This is conducted by maximizing the distance between the nearest training samples of the two classes (a.k.a the *margin* $\gamma = 1/\|\boldsymbol{w}\|_2$), as this way reduces the model generalization error (Vapnik, 1995). While the data are not linearly separable in most practical settings, the hard-margin SVM is subsequently extended to a soft-margin SVM with an implicit mapping $\phi(\cdot)$ for a non-linear decision hyperplane. Mathematically, the soft-margin SVM aims to maximize the margin γ (or minimize $\|\boldsymbol{w}\|_2^2$) and minimize the slack penalty $\sum_{i=1}^n \xi_i$ with the following formulation

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^n \xi_i \\ &\text{s.t. } y_i(\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, 2, \cdots, n\,, \end{aligned} \tag{1}$$

where $\boldsymbol{\xi} = [\xi_1, \xi_2, \cdots, \xi_n]^\top$ is the slack variable and C is the balance parameter. As illustrated by Vapnik (1995), the dual form of problem (1) is given by

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \ \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha}\,, \tag{2}$$

where $\boldsymbol{Y} = \text{diag}(\boldsymbol{y})$ is the label matrix, $\boldsymbol{K} = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{n\times n}$ is the (pre-given) Gram matrix satisfying $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle_{\mathcal{H}}$, and the constraint set is given by $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^n : \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}\}$. Without loss of generality, we assume that the kernel function is bounded, i.e., $\kappa := \sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} |k(\boldsymbol{x}, \boldsymbol{x}')|$.

Theoretical results on learning kernels, mainly on multiple kernel learning (Srebro and Ben-David, 2006; Hussain and Shawe Taylor, 2011) demonstrate that, to achieve a tight bound of the estimation error (namely the gap between empirical error and expected error), a learned SVM classifier is better to admit a large margin γ, and to effectively control the complexity of the hypothesis space. Despite that the above theoretical results on multiple kernel learning cannot be directly applied to non-parametric kernel learning models as the learned kernel is sample-dependent and implicit, their results are able to motivate us to design our non-parametric model. On one hand, an adaptive matrix $\boldsymbol{F}$ is introduced into problem (2) to increase the margin γ; on the other hand, two constraints are considered to control the model complexity. Mathematically, our DANK model is

$$\begin{aligned} &\min_{\boldsymbol{F} \in \mathcal{S}_+^n} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \ \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Y}\big(\boldsymbol{F} \odot \boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha} \\ &\text{s.t. } \|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\text{F}}^2 \leq R^2,\ \text{rank}(\boldsymbol{F}) < r\,, \end{aligned} \tag{3}$$

where R refers to the bounded region size, $\text{rank}(\boldsymbol{F})$ denotes the rank of $\boldsymbol{F}$, and $r \leq n$ is a given integer. The constraint $\boldsymbol{F} \in \mathcal{S}_+^n$ is given to ensure that the learned kernel matrix $\boldsymbol{F} \odot \boldsymbol{K}$ is still a PSD one.[1] Since we do not specify the parametric form of $\boldsymbol{F}$, we can obtain a nonparametric kernel matrix $\boldsymbol{F} \odot \boldsymbol{K}$ and thus our DNAK model is nonparametric. Due to the non-convexity of the used

1. It is admitted by Schur Product Theorem (Styan, 1973) which relates positive semi-definite matrices to the Hadamard product.

rank constraint in problem (3), we consider the *nuclear norm* $\|\cdot\|_*$ instead, which is the best convex lower bound of the non-convex rank function (Recht et al., 2010) and can be minimized efficiently. Accordingly, we relax the constrained optimization problem in Eq. (3) to a unconstrained problem by absorbing the two original constraints to the objective function. Moreover, following the min-max approach (Boyd and Vandenberghe, 2004), problem (3) can be reformulated as

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \min_{\boldsymbol{F}\in\mathcal{S}_+^n} \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{Y}\big(\boldsymbol{F}\odot\boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha} + \eta\|\boldsymbol{F}-\mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2 + \tau\eta\|\boldsymbol{F}\|_*\,, \tag{4}$$

where η, τ are two regularization parameters. Here we denote $\|\boldsymbol{F}-\mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2$ as the centering regularizer. In problem (4), the inner minimization problem with respect to $\boldsymbol{F}$ is a convex conic programming, and the outer maximization problem is a point-wise minimum of concave quadratic functions of $\boldsymbol{\alpha}$. As a consequence, problem (4) is convex, and strong duality holds by Slater's condition (Boyd and Vandenberghe, 2004). The optimal values of the primal and dual form of kernel learning based SVM problems will be equal. Accordingly, when we learn the kernel matrix in problem (4), the objective function value of its primal problem would decrease, which in turn enlarges the margin γ for increasing model flexibility. The introduced two regularizers in Eq. (4) are beneficial to control the model complexity. We briefly explain this here and detail in the next subsection. Intuitively speaking, when $\boldsymbol{F}$ is chosen as the all-one matrix, the DANK model in Eq. (4) degenerates to a standard SVM problem. The considered low-rank regularizer forces $\boldsymbol{F}$ to be endowed with the low-rank structure enjoyed by the all-one matrix. The introduced centering regularizer restricts the adaptive matrix $\boldsymbol{F}$ to vary around the all-one matrix in a small range. This scheme is also able to prevent $\boldsymbol{F}$ from dropping to a trivial solution $\boldsymbol{F}=\mathbf{0}_{n\times n}$. By the above two regularizers, we can effectively restrict the complexity of $\boldsymbol{F}$, and further to control the complexity of the whole model.

2.2 Benefits of the Used Constraints

In this subsection, we aim to demonstrate two merits of the introduced constraints/regularizers. First, although the hard constraints are substituted by two regularizers in Eq. (4), the optimal solution $\boldsymbol{F}^*$ of our unconstrained optimization problem can be still restricted in a bounded set $\mathcal{F}$. This property will be beneficial to control the model complexity. Second, we elucidate that the learned kernel matrix exhibits a fast eigenvalue decay by the used constraints/regularizers, which would be helpful to achieve good generalization properties.

2.2.1 THE BOUNDEDNESS OF THE OPTIMAL SOLUTION

For notational simplicity, we denote the objective function in Eq. (4) as

$$H(\boldsymbol{\alpha},\boldsymbol{F}) = \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{Y}\big(\boldsymbol{F}\odot\boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha} + \eta\|\boldsymbol{F}-\mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2 + \tau\eta\|\boldsymbol{F}\|_*\,,$$

of which the optimal solution $(\boldsymbol{\alpha}^*,\boldsymbol{F}^*)$ is a saddle point of $H(\boldsymbol{\alpha},\boldsymbol{F})$ due to the property of the max-min problem (4). It is easy to check $H(\boldsymbol{\alpha},\boldsymbol{F}^*)\le H(\boldsymbol{\alpha}^*,\boldsymbol{F}^*)\le H(\boldsymbol{\alpha}^*,\boldsymbol{F})$ for any feasible $\boldsymbol{\alpha}$ and $\boldsymbol{F}$. Further, we define the following function

$$h(\boldsymbol{\alpha}) := H(\boldsymbol{\alpha},\boldsymbol{F}^*) = \min_{\boldsymbol{F}\in\mathcal{S}_+^n} H(\boldsymbol{\alpha},\boldsymbol{F})\,, \tag{5}$$

which is concave since $h(\cdot)$ is the minimum of a sequence of concave functions. The optimal solution $\boldsymbol{F}^*$ of problem (4) can be restricted in a bounded set $\mathcal{F}$ by the following Lemma.

Lemma 1 *Problem* (4) *admits the following equivalent formulation such that* $\boldsymbol{F}$ *can be optimized in a bounded region*

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \min_{\boldsymbol{F}\in\mathcal{S}_+^n} H(\boldsymbol{\alpha},\boldsymbol{F}) = \max_{\boldsymbol{\alpha}\in\mathcal{A}} \underbrace{\min_{\boldsymbol{F}\in\mathcal{F}} H(\boldsymbol{\alpha},\boldsymbol{F})}_{\triangleq h(\boldsymbol{\alpha})}, \tag{6}$$

where the feasible region on $\boldsymbol{F}$ *is defined by* $\mathcal{F} := \left\{\boldsymbol{F}\in\mathcal{S}_+^n : \lambda_{\max}(\boldsymbol{F}) \le n - \frac{\tau}{2} + \frac{nC^2}{4\eta}\lambda_{\max}(\boldsymbol{K})\right\}$ *as a nonempty subset of* $\mathcal{S}_+^n$.

Proof The key of the proof is to obtain the optimal solution $\boldsymbol{F}^*$ over $\mathcal{S}_+^n$, i.e., $\boldsymbol{F}^* = \operatorname{argmin}_{\boldsymbol{F}\in\mathcal{S}_+^n} H(\boldsymbol{\alpha},\boldsymbol{F})$ in problem (4). By virtue of the following expression[2]

$$\frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{Y}\big(\boldsymbol{F}\odot\boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha} = \operatorname{tr}\Big[\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\boldsymbol{K}\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\boldsymbol{F}\Big],$$

and denote

$$\boldsymbol{\Gamma}\equiv\boldsymbol{\Gamma}(\boldsymbol{\alpha}) := \frac{1}{4\eta}\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\boldsymbol{K}\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y}), \tag{7}$$

then finding $\boldsymbol{F}^*$ is equivalent to consider the following problem

$$\boldsymbol{F}^* := \operatorname*{argmin}_{\boldsymbol{F}\in\mathcal{S}_+^n} \quad -2\operatorname{tr}\Big[\eta\boldsymbol{\Gamma}(\boldsymbol{\alpha})\boldsymbol{F}\Big] + \eta\|\boldsymbol{F}-\mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2 + \tau\eta\|\boldsymbol{F}\|_*, \tag{8}$$

where we omit the irrelevant term $\mathbf{1}^\top\boldsymbol{\alpha}$ independent of the optimization variable $\boldsymbol{F}$ in problem (4). Further, due to the independence of $\boldsymbol{\Gamma}(\boldsymbol{\alpha})$ on $\boldsymbol{F}$, problem (8) can be reformulated as

$$\boldsymbol{F}^* = \operatorname*{argmin}_{\boldsymbol{F}\in\mathcal{S}_+^n} \|\boldsymbol{F}-\mathbf{1}\mathbf{1}^\top-\boldsymbol{\Gamma}(\boldsymbol{\alpha})\|_{\mathrm{F}}^2 + \tau\|\boldsymbol{F}\|_*. \tag{9}$$

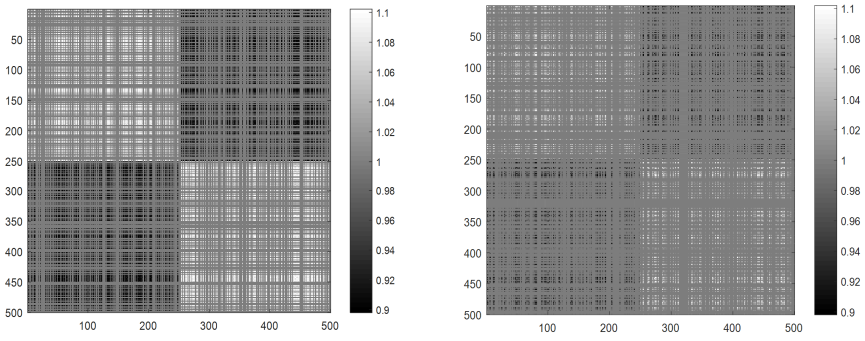
Note that the regularization parameter η is implicitly included in $\boldsymbol{\Gamma}(\boldsymbol{\alpha})$. Following (Cai et al., 2010), we can directly obtain the optimal solution of problem (9) with

$$\boldsymbol{F}^*\equiv\boldsymbol{F}(\boldsymbol{\alpha}) = \mathcal{J}_{\frac{\tau}{2}}(\mathbf{1}\mathbf{1}^\top+\boldsymbol{\Gamma}(\boldsymbol{\alpha})),$$

where we use the singular value thresholding operator $\mathcal{J}_{\frac{\tau}{2}}(\cdot)$ as the proximity operator associated with the nuclear norm, refer to Theorem 2.1 in (Cai et al., 2010) for details. Based on its closed-form, $\lambda_{\max}(\boldsymbol{F}^*)$ can be upper bounded by

$$\begin{aligned}
\lambda_{\max}(\boldsymbol{F}^*) &= \lambda_{\max}\Big(\mathcal{J}_{\frac{\tau}{2}}\big(\mathbf{1}\mathbf{1}^\top+\boldsymbol{\Gamma}(\boldsymbol{\alpha})\big)\Big) = \lambda_{\max}\Big(\mathbf{1}\mathbf{1}^\top+\boldsymbol{\Gamma}(\boldsymbol{\alpha})\Big) - \frac{\tau}{2}\\
&\le \lambda_{\max}(\mathbf{1}\mathbf{1}^\top) + \frac{1}{4\eta}\lambda_{\max}\Big(\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\boldsymbol{K}\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\Big) - \frac{\tau}{2}\\
&= n + \frac{1}{4\eta}\Big\|\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\boldsymbol{K}\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\Big\|_2 - \frac{\tau}{2}\\
&\le n + \frac{1}{4\eta}\big\|\operatorname{diag}(\boldsymbol{\alpha}^\top\boldsymbol{Y})\big\|_2^2\big\|\boldsymbol{K}\big\|_2 - \frac{\tau}{2}\\
&\le n - \frac{\tau}{2} + \frac{nC^2}{4\eta}\lambda_{\max}(\boldsymbol{K}),
\end{aligned} \tag{10}$$

2. We use the formula $\boldsymbol{x}^\top\boldsymbol{A}\odot\boldsymbol{B}\boldsymbol{y} = \operatorname{tr}(\boldsymbol{D}_x\boldsymbol{A}\boldsymbol{D}_y\boldsymbol{B}^\top)$ with $\boldsymbol{D}_x = \operatorname{diag}(\boldsymbol{x})$ and $\boldsymbol{D}_y = \operatorname{diag}(\boldsymbol{y})$.



(a) $\sigma = 1$ (b) $\sigma = 0.07$ by cross validation

Figure 2: Entries of $\boldsymbol{F}^*$ in our DANK model with the Gaussian kernel on the *clowns* data set.

where the first inequality uses the property of maximum eigenvalues, *i.e.*, $\lambda_{\max}(\boldsymbol{A}+\boldsymbol{B}) \leq \lambda_{\max}(\boldsymbol{A})+\lambda_{\max}(\boldsymbol{B})$ for any $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{S}_+^n$, and the last inequality admits by $\|\boldsymbol{\alpha}\|_2^2 \leq nC^2$.

■

Remark: The centering regularizer $\eta\|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2$ in our DANK model requires that the adaptive matrix $\boldsymbol{F} \in \mathcal{F}$ is expected to slightly vary around the all-one matrix, so each element in $[\boldsymbol{\Gamma}(\boldsymbol{\alpha})]_{ij}$ cannot be significantly larger than 1. To this end, recall Eq. (7) and problem (9), the regularizer parameter η is chosen as $\eta \in \mathcal{O}(n)$ to ensure $\mathrm{tr}[\boldsymbol{\Gamma}(\boldsymbol{\alpha})]$ to be in the same order with $\mathrm{tr}(\mathbf{1}\mathbf{1}^\top) \in \mathcal{O}(n)$.

Lemma 1 gives the upper spectral bound of $\boldsymbol{F}^*$, which demonstrates that the feasible region $\mathcal{F}$ is a subset of the PSD cone $\mathcal{S}_+^n$. In this case, we can directly solve $\boldsymbol{F}$ in the subset $\mathcal{F} \subseteq \mathcal{S}_+^n$ instead of the entire PSD cone $\mathcal{S}_+^n$, which makes it possible to seek for a good trade-off between the model flexibility and complexity. Figure 2 experimentally validates the effectiveness of the introduced constraints. We find that $\boldsymbol{F}^*$ is of low-rank and its entries range from 0.9 to 1.1 in a small region. Therefore, such small fluctuation on $\boldsymbol{F}^*$ and its low-rank structure effectively control the model complexity.

2.2.2 FAST EIGENVALUE DECAY

In the above subsection, we have theoretically and experimentally validated the boundedness of the optimal solution $\boldsymbol{F}^*$, and thus this property is beneficial to control the model complexity. In this subsection, we elucidate that the introduced constraints/regularizers result in a fast eigenvalue decay of the learned kernel (matrix), which would be helpful to achieve good prediction performance.[3]

Denote the learned kernel matrix as $\widetilde{\boldsymbol{K}} \equiv \widetilde{\boldsymbol{K}}(\boldsymbol{\alpha}) = \boldsymbol{F} \odot \boldsymbol{K}$, the learned kernel as $\tilde{k}$, and its associated RKHS as $\widetilde{\mathcal{H}}$. In fact, the eigenvalue decay of $\widetilde{\boldsymbol{K}}$ is leveraged to characterize the "size" of $\widetilde{\mathcal{H}}$. For example, a fast eigenvalue decay of a kernel matrix implies that functions in the associated RKHS are smooth, and thus achieving a good prediction performance (Bach, 2013). To this end, we need to control the complexity of $\widetilde{\mathcal{H}}$ for a fast eigenvalue decay of $\widetilde{\boldsymbol{K}}$. This can be achieved by the introduced two constraints/regularizers in our DANK model.

i) the low-rank structure: According to $\mathrm{rank}(\widetilde{\boldsymbol{K}}) \leq \mathrm{rank}(\boldsymbol{F})\mathrm{rank}(\boldsymbol{K})$, see Theorem 3.2 in (Styan,

3. It appears non-trivial to obtain rigorous analysis on generalization properties of non-parametric kernel learning as the learned kernel is sample-dependent and implicit.

1973), the rank of the learned kernel matrix $\widetilde{\boldsymbol{K}}$ mainly depends on the pre-given kernel matrix $\boldsymbol{K}$ as the introduced low-rank constraint on $\boldsymbol{F}$ yields $\mathrm{rank}(\boldsymbol{F}) \ll n$. Accordingly, we can obtain a low rank matrix $\widetilde{\boldsymbol{K}}$ if the pre-given kernel matrix $\boldsymbol{K}$ is of low rank. Here, the low rank property of $\boldsymbol{K}$, i.e., the finite non-zero eigenvalues of $\boldsymbol{K}$, implies a fast eigenvalue decay of $\boldsymbol{K}$, which holds by the following two common cases: (1) the geometric/exponential decay with $\lambda_i(\boldsymbol{K}) \propto nR_0\mathrm{e}^{-ai}$ (r_0 and a are some constants); (2) the polynomial decay with $\lambda_i(\boldsymbol{K}) \propto nR_0 i^{-b}$ with $b > 1$. Under these two cases, the learned kernel matrix $\widetilde{\boldsymbol{K}}$ is able to exhibit a fast eigenvalue decay, which would lead to a good prediction performance (Bach, 2013), or a tight estimation error bound (Liu and Liao, 2015). ii) the bounded constraint: The introduced centering regularizer $\|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2$ would restrict $\boldsymbol{F}$ to vary around the all-one matrix in a small region, so $\boldsymbol{F}$ can be decomposed into $\boldsymbol{F} := \mathbf{1}\mathbf{1}^\top + \boldsymbol{E}$, where $\boldsymbol{E}$ is regarded as a perturbation matrix with small residual error, i.e., $\|\boldsymbol{E}\|_{\mathrm{F}}$ is small. According to Hoffman-Wielandt inequality (Hoffman and Wielandt, 2003) in matrix perturbation theory, we have $\sum_{i=1}^n \left[\lambda_i(\boldsymbol{F}) - \lambda_i(\mathbf{1}\mathbf{1}^\top)\right]^2 \leq \|\boldsymbol{E}\|_{\mathrm{F}}^2$. Since the rank-one matrix $\mathbf{1}\mathbf{1}^\top$ has only one non-zero eigenvalue with $\lambda_1(\mathbf{1}\mathbf{1}^\top) = n$, and its remaining eigenvalues are zero, we have $[\lambda_1(\boldsymbol{F}) - n]^2 + \sum_{i=2}^n [\lambda_i(\boldsymbol{F})]^2 \leq \|\boldsymbol{E}\|_{\mathrm{F}}^2$. Roughly speaking, eigenvalues of $\boldsymbol{F}$ can be well approximated by a rank-one matrix $\mathbf{1}\mathbf{1}^\top$ to some extent. That means, the centering regularizer could also bring the low-rank property on $\boldsymbol{F}$, which is useful to control the complexity of the solving space. Additionally, in the light of this, we omit the exact low-rank constraint/regularizer in the large-sample case due to its inseparable property. In this case, the learned $\boldsymbol{F}$ still exhibits a relative low-rank structure, see Section 4 for details.

Based on the above analyses, we elucidate that, on one hand, the introduced low-rank regularizer and the centering regularizer ensure that $\boldsymbol{F}$ can be optimized in a bounded region in our unconstrained optimization problem. On the other hand, the used constraints/regularizers are beneficial to achieve a fast eigenvalue decay of the learned kernel (matrix) for good prediction performance.

2.3 Out-of-sample Extension

In our DANK model, the learned kernel (matrix) is non-parametric, so it is unknown for test data. This is a so-called out-of-sample extension problem (Bengio et al., 2004; Fanuel et al., 2017; Pan et al., 2017), which extensively exists in non-parametric kernel learning (Lu et al., 2009; Zhuang et al., 2011; Liu et al., 2018), metric learning (Kulis, 2013; Jain et al., 2017), and nonlinear manifold learning (Xie et al., 2013). To address this issue, we develop a simple but effective technique, i.e., the reciprocal nearest neighbor scheme, to establish the learned kernel (matrix) for test data.

Given the optimal $\boldsymbol{F}^*$ on training data, the test data $\{\boldsymbol{x}_i'\}_{i=1}^m$, and the initial kernel matrix for test data $\boldsymbol{K}' = [k(\boldsymbol{x}_i, \boldsymbol{x}_j')]_{n\times m}$, we aim to establish the adaptive matrix $\boldsymbol{F}'$ for test data by the reciprocal nearest neighbor scheme. Formally, we first construct the similarity matrix $\boldsymbol{M}$ between the training data and the test data based on the nearest neighbor scheme. The used distance to find the nearest neighbor is the standard $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ metric in the d-dimensional Euclidean space. Assuming that $\boldsymbol{x}_j'$ is the r-th nearest neighbor of $\boldsymbol{x}_i$ in the set of $\{\boldsymbol{x}_t'\}_{t=1}^m$, denoted as $\boldsymbol{x}_j' = \mathrm{NN}_r(\boldsymbol{x}_i, \{\boldsymbol{x}_t'\}_{t=1}^m)$. Meanwhile $\boldsymbol{x}_i$ is the s-th nearest neighbor of $\boldsymbol{x}_j'$ in $\{\boldsymbol{x}_t\}_{t=1}^n$, denoted as $\boldsymbol{x}_i = \mathrm{NN}_s(\boldsymbol{x}_j', \{\boldsymbol{x}_t\}_{t=1}^n)$, then the similarity matrix $\boldsymbol{M} \in \mathbb{R}^{n\times m}$ is defined by

$$M_{ij} = \frac{1}{rs}, \text{ if } \boldsymbol{x}_j' = \mathrm{NN}_r\left(\boldsymbol{x}_i, \{\boldsymbol{x}_t'\}_{t=1}^m\right) \wedge \boldsymbol{x}_i = \mathrm{NN}_s\left(\boldsymbol{x}_j', \{\boldsymbol{x}_t\}_{t=1}^n\right), \quad \forall i \in [n], j \in [m], \tag{11}$$

and thus $\boldsymbol{M}_j = [M_{1j}, M_{2j}, \cdots, M_{nj}]^\top \in \mathbb{R}^n$ describes the relationship between $\boldsymbol{x}_j'$ and the training data $\{\boldsymbol{x}_i\}_{i=1}^n$. This is a much stronger and more robust indicator of similarity than the simple and

unidirectional nearest neighborhood relationship, since it takes into account the local densities of vectors around $\boldsymbol{x}_i$ and $\boldsymbol{x}'_j$. This reciprocal nearest scheme has been extensively applied to computer vision, such as image retrieval (Qin et al., 2011) and person re-identification (Zhong et al., 2017; Zheng et al., 2012). Accordingly, $\boldsymbol{F}'$ is given by

$$\boldsymbol{F}'_j \leftarrow \boldsymbol{F}^*_{j^*}, \text{ if } j^* = \underset{t}{\operatorname{argmax}} \ \{M_{1j}, M_{2j}, \cdots, M_{tj}, \cdots, M_{nj}\} \quad \forall t \in [n]\,,$$

That is to say, if $\boldsymbol{x}_{j^*}$ is the "optimal" reciprocal nearest neighbor of $\boldsymbol{x}'_j$ among the training data set $\{\boldsymbol{x}_t\}_{t=1}^n$, the j^*-th column of $\boldsymbol{F}^*$ is assigned to the j-th column of $\boldsymbol{F}'$. By doing so, $\boldsymbol{F}^*$ results in a flexible kernel matrix $\boldsymbol{F}' \odot \boldsymbol{K}'$ for test data. Admittedly, we would be faced with the inconsistency if we directly extend the training kernel to the test kernel in this way. However, in our model, $\boldsymbol{F}$ is designed to vary in a small range with low-rank structure, and is expected to smoothly vary between any two neighboring data points in $\mathcal{F}$, so the extension to $\boldsymbol{F}'$ on test data by this scheme is reasonable. Further, we attempt to provide some theoretical justification for this scheme as follows.

Mathematically, if $\boldsymbol{x}$ and $\boldsymbol{x}'$ are the reciprocal nearest neighbor pair, we expect that $\|\varphi^\top(\boldsymbol{x})\varphi(\boldsymbol{x}'_j) - \varphi^\top(\boldsymbol{x}')\varphi(\boldsymbol{x}'_j)\|_2^2$ is small on the test data $\{\boldsymbol{x}'_j\}_{j=1}^m$, where φ is the learned implicit feature mapping such that $F_{ij}K_{ij} = \langle \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{x}_j)\rangle_{\widetilde{\mathcal{H}}}$ on the training data. Balcan et al. (2006) demonstrate that, in the presence of a large margin γ, a kernel function can also be viewed as a mapping from the input space $\mathcal{X}$ into an $\tilde{\mathcal{O}}(1/\gamma^2)$ space.

Proposition 2 *(Balcan et al., 2006) Given $0 < \epsilon \leq 1$, the margin γ and the implicit mapping $\varphi(\cdot)$ in SVM, then with probability at least $1-\delta$, let d' be a positive integer such that $d' \geq d_0 = \mathcal{O}(\frac{1}{\gamma^2}\log\frac{1}{\epsilon\delta})$, for any $\boldsymbol{x} \in \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ and $\boldsymbol{x}' \in \{\boldsymbol{x}'_i\}_{j=1}^m \subset \mathbb{R}^d$ with $d \geq d'$, we have*

$$(1-\epsilon)\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2 \leq \|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{x}')\|_2^2 \leq (1+\epsilon)\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2\,,$$

where the mapping φ is a random projection following with the Gaussian distribution or the uniform distribution.

Remark: If the learned $\tilde{k}$ in our DANK model is sub-Gaussian, the above bounds can be achieved (Shi et al., 2012). Note that the sub-Gaussian kernel assumption is mild as demonstrated by Dao et al. (2017). Regarding to d', we choose the lower bound $d = d' = d_0 = \mathcal{O}(\frac{1}{\gamma^2}\log\frac{1}{\epsilon\delta})$, which can be achieved by the margin $\gamma = 1/\|\boldsymbol{w}\|_2$ and $\|\boldsymbol{w}\|_2^2 \in \mathcal{O}(d)$.

Based on the above analysis, given the "optimal" reciprocal nearest neighbor pair $(\boldsymbol{x}, \boldsymbol{x}')$, for any test data point $\boldsymbol{x}'_j$ with $j \in [m]$, we have

$$\begin{aligned}\|\varphi^\top(\boldsymbol{x})\varphi(\boldsymbol{x}'_j) - \varphi^\top(\boldsymbol{x}')\varphi(\boldsymbol{x}'_j)\|_2^2 &\leq \|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{x}')\|_2^2 \|\varphi^\top(\boldsymbol{x}'_j)\varphi(\boldsymbol{x}'_j)\|_2 \\ &\leq \underbrace{\|\varphi^\top(\boldsymbol{x}'_j)\varphi(\boldsymbol{x}'_j)\|_2}_{\text{effected by } \boldsymbol{F}\odot\boldsymbol{K} \text{ for } \boldsymbol{x}'_j} \quad (1+\epsilon)\underbrace{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}_{\text{small}} \\ &\leq \widetilde{C}(1+\epsilon)\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2\,,\end{aligned} \tag{12}$$

where $\widetilde{C}$ is some constant as the learned kernel is bounded. As a result, we can obtain a small $\|\varphi^\top(\boldsymbol{x})\varphi(\boldsymbol{x}'_j) - \varphi^\top(\boldsymbol{x}')\varphi(\boldsymbol{x}'_j)\|_2^2$ if $\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2$ is small, which is beneficial to out-of-sample extensions.

2.4 Connections to Other Models

Our non-parametric kernel learning framework in fact covers several typical models.

Kernel with multiple layers: A kernel with l layers in deep architectures (Cho and Saul, 2009) is defined as

$$k^{(l)}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\langle \phi^{(l)}\big(\cdots \phi^{(1)}(\boldsymbol{x}_i)\big), \phi^{(l)}\big(\cdots \phi^{(1)}(\boldsymbol{x}_j)\big) \right\rangle ,$$

which computes the inner product between two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ after l successive applications of the nonlinear mapping $\phi(\cdot)$. For example, the two layer composition of Gaussian kernel can be formulated as

$$k^{(2)}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi^{(2)}(\phi^{(1)}(\boldsymbol{x}_i)), \phi^{(2)}(\phi^{(1)}(\boldsymbol{x}_j)) \rangle = e^{-\frac{1}{\sigma^2}} \exp\left(\frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sigma^2}\right). \tag{13}$$

We observe that the nested kernel in Eq. (13) can be decomposed into a fixed Gaussian kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and a nonlinear pairwise function $g(\boldsymbol{x}_i, \boldsymbol{x}_j)$ such that $k^{(2)}(\boldsymbol{x}_i, \boldsymbol{x}_j) := g(\boldsymbol{x}_i, \boldsymbol{x}_j)k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which is actually associated with the data-adaptive matrix $\boldsymbol{F} = [g(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{n\times n}$. That means, using a single kernel as well as a nonlinear pairwise layer could achieve a comparable and even better model flexibility when compared to the two-layer kernel framework. Learning the nonlinear pairwise function $g(\cdot, \cdot)$ by the matrix $\boldsymbol{F}$ is an interpolation problem, which is also related to interpolation learning (Hastie et al., 2019; Bartlett et al., 2020). Further, recent deep kernel architectures (Wilson et al., 2016; Mairal, 2016; Chen et al., 2020) bridge neural networks and kernel methods and achieve promising performance on various tasks. This will motivate us to design a multi-layer version of our DANK model, in which the parameters can be optimized in a layer-by-layer way in the training process. We leave this to future work.

Hyper-parameter learning: In our model, the entry in the kernel matrix is learned from the data, and thus Gaussian kernels with flexible variances (Ying and Zhou, 2007) can be linked to our framework. Besides, based on the Schönberg's representation theorem (Wendland, 2004), we consider our DANK model in a distribution view (Khuzani et al., 2020)

$$\tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \int_0^\infty e^{-\xi\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2} \mu(\mathrm{d}\xi) \approx \frac{1}{D}\sum_{r=1}^{D} e^{-\xi_r\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}, \tag{14}$$

where μ is an implicit non-negative Borel measure with $\{\xi_r\}_{r=1}^D \sim \mu(\cdot)$ that corresponds to the data-adaptive matrix $\boldsymbol{F}$.

3. Algorithm for DANK Model in SVM

Extra-gradient based methods can be directly applied to solve the max-min problem (4), and have been shown to exhibit an $\mathcal{O}(1/t)$ convergence rate (Nemirovski, 2004), where t is the iteration number. Further, to accelerate the convergence rate, this section investigates the gradient-Lipschitz continuity of $h(\boldsymbol{\alpha})$ in Eq. (5). Based on this, we introduce the Nesterov's smooth optimization method (Nesterov, 2005) that requires $\nabla h(\boldsymbol{\alpha})$ Lipschitz continuous to solve problem (4), that is shown to achieve $\mathcal{O}(1/t^2)$ convergence rate.

3.1 Gradient-Lipschitz Continuity of DANK in SVM

To prove the gradient-Lipschitz continuity of $h(\boldsymbol{\alpha})$, we need the following lemma.

Lemma 3 *Under the same condition in Lemma 1, for any $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2 \in \mathcal{A}$, we have*

$$\big\|\boldsymbol{F}(\boldsymbol{\alpha}_1) - \boldsymbol{F}(\boldsymbol{\alpha}_2)\big\|_{\mathrm{F}} \leq \frac{\|\boldsymbol{K}\|_{\mathrm{F}}\big(\|\boldsymbol{\alpha}_1\|_2 + \|\boldsymbol{\alpha}_2\|_2\big)}{4\eta}\big\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\big\|_2\,,$$

where $\boldsymbol{F}(\boldsymbol{\alpha}_1) = \mathcal{J}_{\frac{\tau}{2}}\big(\mathbf{1}\mathbf{1}^\top + \boldsymbol{\Gamma}(\boldsymbol{\alpha}_1)\big)$ and $\boldsymbol{F}(\boldsymbol{\alpha}_2) = \mathcal{J}_{\frac{\tau}{2}}\big(\mathbf{1}\mathbf{1}^\top + \boldsymbol{\Gamma}(\boldsymbol{\alpha}_2)\big)$.

Proof The proofs can be found in Appendix A.1. ■

Formally, based on Lemmas 1 and 3, we present the following theorem.

Theorem 4 *The function $h(\boldsymbol{\alpha})$ in Eq.* (5) *is gradient-Lipschitz continuous, i.e.*

$$\|\nabla h(\boldsymbol{\alpha}_1) - \nabla h(\boldsymbol{\alpha}_2)\|_2 \leq L\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2, \quad \forall \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{A}\,,$$

where the Lipschitz constant is $L = \kappa\left(n + 3nC^2\|\boldsymbol{K}\|_{\mathrm{F}}/4\eta\right)$ with $\kappa := \sup_{\boldsymbol{x},\boldsymbol{x}'\in\mathcal{X}}|k(\boldsymbol{x},\boldsymbol{x}')|$.

Proof The proofs can be found in Appendix A.2. ■

The above theoretical analyses demonstrate that $\nabla h(\boldsymbol{\alpha})$ is Lipschitz continuous, which provides a justification for utilizing a smooth optimization Nesterov's acceleration method to solve problem (4) with faster convergence.

3.2 Nesterov's Smooth Optimization Method

Here we introduce a projected gradient algorithm with Nesterov's acceleration to solve the optimization problem (4). Nesterov (2005) proposes an optimal scheme for smooth optimization $\min_{\boldsymbol{x}\in\mathcal{Q}} g(\boldsymbol{x})$, where $g(\cdot)$ is a convex gradient-Lipschitz continuous function over a closed convex set $\mathcal{Q}$. Introducing a continuous and strongly convex function denoted as *proxy-function* $d(\boldsymbol{x})$ on $\mathcal{Q}$, the first-order projected gradient method with Nesterov's acceleration can then be used to solve this problem. In our model, we aim to solve the following convex problem

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \; h(\boldsymbol{\alpha})\,, \tag{15}$$

where $h(\boldsymbol{\alpha})$ is concave and gradient-Lipschitz continuous with the Lipschitz constant L in Theorem 4. Here the proxy-function is defined as $d(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_0\|_2^2$ with $\boldsymbol{\alpha}_0 \in \mathcal{A}$. The first-order Nesterov's smooth optimization method for solving problem (4) is summarized in Algorithm 1.

The key steps of Nesterov's acceleration are characterized by Lines 6, 7, and 8 in Algorithm 1. To be specific, according to Nesterov (2005), at the t-th iteration, we need to solve the following problem

$$\boldsymbol{\beta}^{(t)} = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathcal{A}} \frac{L}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_0\|_2^2 + \sum_{i=0}^{t}\frac{i+1}{2}\Big[h(\boldsymbol{\alpha}^{(i)}) + \big\langle\nabla h(\boldsymbol{\alpha}^{(i)}), \boldsymbol{\alpha} - \boldsymbol{\alpha}^{(i)}\big\rangle\Big]\,, \tag{16}$$

which is equivalent to

$$\boldsymbol{\beta}^{(t)} = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathcal{A}} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_0\|_2^2 + \frac{2}{L}\sum_{i=0}^{t}\frac{i+1}{2}\nabla h^\top(\boldsymbol{\alpha}^{(i)})\boldsymbol{\alpha}\,,$$

Algorithm 1: Projected gradient method with Nesterov's acceleration for problem (4)

Input: The kernel matrix $\boldsymbol{K}$, the label matrix $\boldsymbol{Y}$, and the Lipschitz constant in Theorem 4
Output: The optimal $\boldsymbol{\alpha}^*$

1 Set the stopping criteria $t_{\max} = 2000$ and $\epsilon = 10^{-4}$.
2 Initialize $t = 0$ and $\boldsymbol{\alpha}^{(0)} \in \mathcal{A} := \mathbf{0}$.
3 **Repeat**
4 Compute $\boldsymbol{F}(\boldsymbol{\alpha}^{(t)}) = \mathcal{J}_{\frac{\tau}{2}}\big(\mathbf{1}\mathbf{1}^\top + \boldsymbol{\Gamma}(\boldsymbol{\alpha}^{(t)})\big)$;
5 Compute $\nabla h(\boldsymbol{\alpha}^{(t)}) = \mathbf{1} - \boldsymbol{Y}\big(\boldsymbol{F}(\boldsymbol{\alpha}^{(t)}) \odot \boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha}^{(t)}$;
6 Compute $\boldsymbol{\theta}^{(t)} = \mathcal{P}_{\mathcal{A}}\Big(\boldsymbol{\alpha}^{(t)} + \frac{1}{L}\nabla h(\boldsymbol{\alpha}^{(t)})\Big)$;
7 Compute $\boldsymbol{\beta}^{(t)} = \mathcal{P}_{\mathcal{A}}\Big(\boldsymbol{\alpha}^{(0)} - \frac{1}{2L}\sum_{i=0}^{t}(i+1)\nabla h(\boldsymbol{\alpha}^{(i)})\Big)$;
8 Set $\boldsymbol{\alpha}^{(t+1)} = \frac{t+1}{t+3}\boldsymbol{\theta}^{(t)} + \frac{2}{t+3}\boldsymbol{\beta}^{(t)}$;
9 $t := t + 1$;
10 **Until** $t \geq t_{\max}$ *or* $\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)}\|_2 \leq \epsilon$;

where we omit the irrelevant terms $h(\boldsymbol{\alpha}^{(i)})$ and $\nabla h^\top(\boldsymbol{\alpha}^{(i)})\boldsymbol{\alpha}^{(i)}$ that are independent of the optimization variable $\boldsymbol{\alpha}$ in Eq. (16). Accordingly, the above problem is further reformulated as

$$\boldsymbol{\beta}^{(t)} = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathcal{A}} \ \Big\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_0 + \frac{1}{2L}\sum_{i=0}^{t}(i+1)\nabla h(\boldsymbol{\alpha}^{(i)})\Big\|_2^2,$$

of which the optimal solution is $\boldsymbol{\beta}^{(t)} = \mathcal{P}_{\mathcal{A}}\Big(\boldsymbol{\alpha}_0 - \frac{1}{2L}\sum_{i=0}^{t}(i+1)\nabla h(\boldsymbol{\alpha}^{(i)})\Big)$ as outlined in Line 7 in Algorithm 1, where $\mathcal{P}_{\mathcal{A}}(\boldsymbol{\alpha})$ is a projection operator that projects $\boldsymbol{\alpha}$ over the set $\mathcal{A}$. A quick note on projection onto the feasible set $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^n : \boldsymbol{\alpha}^\top\boldsymbol{y} = 0,\ \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}\}$: it typically suffices in practice to use the alternating projection algorithm (Von Neumann, 1949). Since the feasible set $\mathcal{A}$ is the intersection of a hyperplane and a hypercube, both of them admit a simple projection step. To be specific, first clip $\boldsymbol{\alpha}$ to $[0, C]$, and then project on the hyperplane $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \frac{\boldsymbol{y}^\top\boldsymbol{\alpha}}{n}\boldsymbol{y}$. The convergence rate of the alternating projection algorithm is shown to be linear (Von Neumann, 1949) and thus it is very efficient.

It can be noticed that when Lines 6, 7, and 8 in Algorithm 1 are replaced by

$$\boldsymbol{\alpha}^{(t+1)} = \mathcal{P}_{\mathcal{A}}\Big(\boldsymbol{\alpha}^{(t)} + \frac{1}{L}\nabla h(\boldsymbol{\alpha}^{(t)})\Big) \tag{17}$$

with the Lipschitz constant $L = n - \frac{\tau}{2} + \frac{nC^2}{4\eta}\lambda_{\max}(\boldsymbol{K})$ derived from Lemma 1, the Nesterov's smooth method degenerates to a standard projected gradient method. The convergence of the Nesterov smoothing optimization algorithm is pointed out by Theorem 2 in (Nesterov, 2005)

$$h(\boldsymbol{\alpha}^*) - h(\boldsymbol{\beta}^{(t)}) \leq \frac{8L\|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}^*\|^2}{(t+1)(t+2)},$$

where $\boldsymbol{\alpha}^*$ is the optimal solution of Eq. (15). Note that, in general, Algorithm 1 cannot guarantee $\{h(\boldsymbol{\alpha}^{(t)}) : t \in \mathbb{N}\}$ and $\{h(\boldsymbol{\beta}^{(t)}) : t \in \mathbb{N}\}$ to be monotonely increasing during the maximization

process. Nevertheless, such algorithm can be modified to obtain a monotone sequence with replacing Line 6 in Algorithm 1 by

$$\begin{cases} \tilde{\boldsymbol{\theta}}^{(t)} = \mathcal{P}_{\mathcal{A}}\Big(\boldsymbol{\alpha}^{(t)} + \dfrac{1}{L}\nabla h(\boldsymbol{\alpha}^{(t)})\Big)\,, \\ \boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}}\, h(\boldsymbol{\alpha}),\ \boldsymbol{\alpha} \in \{\boldsymbol{\theta}^{(t-1)}, \tilde{\boldsymbol{\theta}}^{(t)}, \boldsymbol{\alpha}^{(t)}\}\,. \end{cases}$$

The Nesterov's smooth optimization method takes $\mathcal{O}(\sqrt{L/\epsilon})$ to find an ϵ-optimal solution, which is better than the standard projected gradient method with the complexity $\mathcal{O}(L/\epsilon)$.

4. DANK in Large Scale Case

Scalability in kernel methods is a vital issue which often limits their applications in large data sets (Rahimi and Recht, 2007; Wang et al., 2016; Liu et al., 2020), especially for nonparametric kernel learning optimized by semi-definite programming. Hence, in this section, we take our DANK model embedded in SVM as an example to study our kernel approximation approach. The presented results in this section are also suitable to other nonparametric kernel learning based algorithms.

To consider the scalability of our DANK model embedded in SVM in large-scale situations, problem (4) is reformulate as

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{F}\in\mathcal{S}_+^n} \ H(\boldsymbol{\alpha}, \boldsymbol{F}) = \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Y}\big(\boldsymbol{F}\odot\boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha} + \eta\|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2 \\ &\text{s.t. } \ \mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{1}\,. \end{aligned} \tag{18}$$

where the bias term b is usually omitted in the large scale issue (Keerthi et al., 2006; Hsieh et al., 2014; Lian and Fan, 2017). Besides, we have to omit the low-rank regularizer on $\boldsymbol{F}$ due to its inseparable property, which is reasonable based on the rapid decaying spectra of the kernel matrix (Smola and Schölkopf, 2000). Specifically, in Section 2.2.2, we have demonstrated that $\boldsymbol{F}$ would exhibit the low-rank property as well by the centering regularizer $\|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2$. Furthermore, we will experimentally verify that dropping the low-rank term in large-scale problems has no much sacrifice for the accuracy in Section 6.2.2.

In our decomposition-based scalable approach, we divide the data into small subsets by k-means, and then solve each subset independently and efficiently. Such similar scheme also exists in (Hsieh et al., 2014; Zhang et al., 2013; Si et al., 2017). To be specific, we firstly partition the data into v subsets $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_v\}$, and then solve the respective sub-problems independently with the following formulation

$$\begin{aligned} &\max_{\boldsymbol{\alpha}^{(c)}} \min_{\boldsymbol{F}^{(c,c)}\in\mathcal{S}_+^{|\mathcal{V}_c|}} \mathbf{1}^\top\boldsymbol{\alpha}^{(c)} + \eta\|\boldsymbol{F}^{(c,c)} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2 - \frac{1}{2}{\boldsymbol{\alpha}^{(c)}}^\top \boldsymbol{Y}^{(c,c)}\big(\boldsymbol{F}^{(c,c)}\odot\boldsymbol{K}^{(c,c)}\big)\boldsymbol{Y}^{(c,c)}\boldsymbol{\alpha}^{(c)} \\ &\text{s.t. } \ \mathbf{0} \le \boldsymbol{\alpha}^{(c)} \le C\mathbf{1}, \ \ \forall\, c = 1, 2, \ldots, v\,, \end{aligned} \tag{19}$$

where $|\mathcal{V}_c|$ denotes the number of data points in $\mathcal{V}_c$. Suppose that $(\bar{\boldsymbol{\alpha}}^{(c)}, \bar{\boldsymbol{F}}^{(c,c)})$ is the optimal solution of the c-th subproblem, the approximation solution $(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$ to the whole problem is concatenated by $\bar{\boldsymbol{\alpha}} = [\bar{\boldsymbol{\alpha}}^{(1)}, \bar{\boldsymbol{\alpha}}^{(2)}, \ldots, \bar{\boldsymbol{\alpha}}^{(v)}]$ and $\bar{\boldsymbol{F}} = \operatorname{diag}(\bar{\boldsymbol{F}}^{(1,1)}, \bar{\boldsymbol{F}}^{(2,2)}, \ldots, \bar{\boldsymbol{F}}^{(v,v)})$, where $\bar{\boldsymbol{F}}$ is a block-diagonal matrix. In the next, we study the decomposition-based scalable approach in the following two aspects.

First, the objective function value $H(\bar{\boldsymbol{\alpha}},\bar{\boldsymbol{F}})$ in Eq. (18) is close to $H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)$. Second, if $\boldsymbol{x}_i$ is not a support vector of the subproblem, it will also be a non-support vector of the whole problem under some conditions. To prove the above three propositions, we need the following lemma that links the subproblems to the whole problem.

Lemma 5 *Given the optimal solution $(\bar{\boldsymbol{\alpha}}^{(c)}, \bar{\boldsymbol{F}}^{(c,c)})$ of problem* (19) *with $c \in \{1, 2, \cdots, v\}$, by concatenating $\bar{\boldsymbol{\alpha}} = [\bar{\boldsymbol{\alpha}}^{(1)}, \bar{\boldsymbol{\alpha}}^{(2)}, \ldots, \bar{\boldsymbol{\alpha}}^{(v)}]$ and $\bar{\boldsymbol{F}} = \operatorname{diag}(\bar{\boldsymbol{F}}^{(1,1)}, \bar{\boldsymbol{F}}^{(2,2)}, \ldots, \bar{\boldsymbol{F}}^{(v,v)})$, the approximation solution $(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$ to the whole problem is the optimal solution of the following problem*

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{F} \in \mathcal{S}_+^n} \quad \bar{H}(\boldsymbol{\alpha}, \boldsymbol{F}) \triangleq \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{Y} \big(\boldsymbol{F} \odot \bar{\boldsymbol{K}}\big) \boldsymbol{Y} \boldsymbol{\alpha} + \eta \|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2 \\ &\text{s.t. } \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}\,, \end{aligned} \tag{20}$$

with the kernel $\bar{\boldsymbol{K}}$ defined by

$$\bar{K}_{ij} = I(\pi(\boldsymbol{x}_i), \pi(\boldsymbol{x}_j)) K_{ij}\,,$$

where $\pi(\boldsymbol{x}_i)$ is the cluster that $\boldsymbol{x}_i$ belongs to, and $I(a, b) = 1$ iff $a = b$, and $I(a, b) = 0$ otherwise.

Proof The proofs can be found in Appendix B.1. ■

Based on the above lemma, we are ready to investigate the difference between $H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)$ and $H(\bar{\boldsymbol{\alpha}},\bar{\boldsymbol{F}})$ as follows.

Theorem 6 *Denote $(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)$ and $(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$ as the optimal solutions of problem* (4) *and problem* (20)*, respectively. Suppose that each element in $\boldsymbol{F}^*$ and $\bar{\boldsymbol{F}}$ satisfies $0 < B_1 \leq \max\{F_{ij}^*, \bar{F}_{ij}\} \leq B_2$, with $B = B_2 - B_1$, we have*

$$\big|H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) - H(\bar{\boldsymbol{\alpha}},\bar{\boldsymbol{F}})\big| \leq \frac{1}{2} B C^2 Q(\pi)\,,$$

with $Q(\pi) = \sum_{i,j:\pi(\boldsymbol{x}_i) \neq \pi(\boldsymbol{x}_j)}^n |k(\boldsymbol{x}_i, \boldsymbol{x}_j)|$, where $\{\pi(\boldsymbol{x}_1), \pi(\boldsymbol{x}_2), \cdots, \pi(\boldsymbol{x}_n)\}$ is the partition indicator and C is the balance parameter in SVM.

Proof The proofs can be found in Appendix B.2. ■

Remark: $Q(\pi)$ actually consists of the off-diagonal values of the kernel matrix $\boldsymbol{K}$ if we rearrange the training data in a clustering order. It depends on the data distribution, the number of clusters v, and the kernel type. Intuitively, if the clusters are nicely shaped (e.g. Gaussian) and well-separated, then the kernel matrix may be approximately block-diagonal. In this case, $Q(\pi)$ would be small. Let we examine two extreme cases of v. If $v = 1$, i.e., only one cluster, we have $Q(\pi) = 0$; If $v = n$, i.e., each data point is grouped into a cluster, then $Q(\pi)$ can be upper bounded by $Q(\pi) \leq \sum_{i,j}^n |k(\boldsymbol{x}_i, \boldsymbol{x}_j)|$. In practical clustering algorithms, v is often chosen to be much smaller than n, i.e., $v \ll n$, and thus we can obtain a small $Q(\pi)$. In fact, it appears non-trivial to quantitatively analyze the relationship between $Q(\pi)$ and v, so we experimentally study this relationship in Section 6.2.4.

Besides, in SVM, we also concern about the relationship of support/non-support vectors between the subproblems and the whole problem. Accordingly, we present the following theorem to explain this issue.

Theorem 7 *Under the same condition of Theorem 6 with an additional bounded assumption* $\kappa := \sup_{\boldsymbol{x},\boldsymbol{x}'\in\mathcal{X}} |k(\boldsymbol{x},\boldsymbol{x}')|$*, suppose that* $\boldsymbol{x}_i$ *is not a support vector of the subproblem, i.e.,* $\bar{\alpha}_i = 0$*,* $\boldsymbol{x}_i$ *will also not be a support vector of the whole problem i.e.,* $\alpha_i = 0$*, under the following condition*

$$\left(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\right)_i \leq -(B+B_2)C\left(\|\bar{\boldsymbol{K}}_i\|_1+\kappa\right) \leq -(B+B_2)C\left(\|\boldsymbol{K}\|_1+\kappa\right), \tag{21}$$

where $\bar{\boldsymbol{K}}_i$ *denotes the* i*-th column of the kernel matrix* $\bar{\boldsymbol{K}}$*.*

Proof The proofs can be found in Appendix B.3. ■

Remark: Eq. (21) is a sufficient condition and can be expressed as

$$1 - nB_2\kappa C \leq \left(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\right)_i \leq -(B+B_2)C\left(\|\bar{\boldsymbol{K}}_i\|_1+\kappa\right),$$

where the first inequality admits $\inf\left(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\right)_i = \inf\left(1 - \sum_{j=1}^n y_i y_j \bar{F}_{ij} \bar{K}_{ij} \bar{\alpha}_j\right) = 1 - nB_2\kappa C$. So if we assume that $\left(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\right)_i$ of non-support vectors is uniformly distributed over the range $[1 - n\kappa B_2 C, 0]$, then nearly $1 - \frac{(B+B_2)C\left(\|\bar{\boldsymbol{K}}_i\|_1+\kappa\right)}{nB_2\kappa C-1} \approx 1 - \frac{c}{\sqrt{n}}$ of the total non-support vectors can be directly recognized, where c is some constant. That means, the screening proportion of non-support vectors is $(1 - c/\sqrt{n}) * 100\%$, at a certain $\mathcal{O}(n^{-1/2})$ rate. If $\left(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\right)_i$ follows with some heavy-tailed distributions over the range $[1 - n\kappa B_2 C, 0]$, the recognized rate would decrease. And specifically, we will experimentally check that our screening condition is reasonable in Section 6.2.4.

5. DANK Model in SVR

In this section, we incorporate the DANK model into SVR and also develop the Nesterov's smooth optimization algorithm to solve it. Here the label space is $\mathcal{Y} \subseteq \mathbb{R}$ for regression.

Similar to DANK in SVM revealed by problem (4), we incorporate the DANK model into SVR with the ε-insensitive loss, namely

$$\begin{aligned} \max_{\hat{\boldsymbol{\alpha}},\check{\boldsymbol{\alpha}}} \min_{\boldsymbol{F}\in\mathcal{S}_+^n} & -\frac{1}{2}(\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}})^\top \big(\boldsymbol{F} \odot \boldsymbol{K}\big)(\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}}) + (\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}})^\top \boldsymbol{y} \\ & - \varepsilon(\hat{\boldsymbol{\alpha}} + \check{\boldsymbol{\alpha}})^\top \mathbf{1} + \eta\|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}^2 + \tau\eta\|\boldsymbol{F}\|_* \\ \text{s.t. } & 0 \leq \hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}} \leq C,\ (\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}})^\top \boldsymbol{y} = 0\,, \end{aligned} \tag{22}$$

where the dual variable is $\boldsymbol{\alpha} = \hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}}$. The objective function in problem (22) is denoted as $H(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}, \boldsymbol{F})$. Further, we define the following function

$$h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}) \triangleq H(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}, \boldsymbol{F}^*) = \min_{\boldsymbol{F}\in\mathcal{S}_+^n} H(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}, \boldsymbol{F})\,, \tag{23}$$

where $h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})$ can be obtained by solving the following problem

$$\min_{\boldsymbol{F}\in\mathcal{S}_+^n} \|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top - \boldsymbol{\Gamma}(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})\|_{\mathrm{F}}^2 + \tau\|\boldsymbol{F}\|_*\,, \tag{24}$$

with $\mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}) = \frac{1}{4\eta}\operatorname{diag}(\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}})^\top \boldsymbol{K} \operatorname{diag}(\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}})$. The optimal solution of Eq. (24) is $\boldsymbol{F}^* = \mathcal{J}_{\frac{\tau}{2}}(\mathbf{1}\mathbf{1}^\top + \mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}))$. We can easily check that Lemma 1 is also applicable to problem (23)

$$h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}) = \min_{\boldsymbol{F} \in \mathcal{B}} H(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}, \boldsymbol{F})\,.$$

Similar to Lemma 3, in our DANK model embedded in SVR, $\|\mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1) - \mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2)\|_2$ can be bounded by the following lemma.

Lemma 8 *For any $\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1, \hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2 \in \mathcal{A}$, we have*

$$\begin{aligned}&\big\|\boldsymbol{F}(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1) - \boldsymbol{F}(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2)\big\|_{\mathrm{F}} \le \|\mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1) - \mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2)\|_{\mathrm{F}} \\ &\le \frac{\|\boldsymbol{K}\|}{4\eta}\big\|\hat{\boldsymbol{\alpha}}_1 - \check{\boldsymbol{\alpha}}_1 + \hat{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_2\big\|_2\big\|\hat{\boldsymbol{\alpha}}_1 - \check{\boldsymbol{\alpha}}_1 - \hat{\boldsymbol{\alpha}}_2 + \check{\boldsymbol{\alpha}}_2\big\|_2\,,\end{aligned}$$

where $\boldsymbol{F}(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1)) = \mathcal{J}_{\frac{\tau}{2}}\big(\mathbf{1}\mathbf{1}^\top + \mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1)\big)$ and $\boldsymbol{F}(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2) = \mathcal{J}_{\frac{\tau}{2}}\big(\mathbf{1}\mathbf{1}^\top + \mathbf{\Gamma}(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2)\big)$.

The proof of Lemma 8 is similar to that of Lemma 3, and here we omit the detailed proof. Next we present the partial derivative of $h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})$ regarding to $\hat{\boldsymbol{\alpha}}$ and $\check{\boldsymbol{\alpha}}$.

Proposition 9 *The objective function $h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})$ with two variables defined by Eq.* (23) *is differentiable and its partial derivatives are given by*

$$\begin{cases} \dfrac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})}{\partial \hat{\boldsymbol{\alpha}}} = -\varepsilon \boldsymbol{I} - (\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}})\boldsymbol{F} \odot \boldsymbol{K} + \boldsymbol{y}\,, \\ \dfrac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})}{\partial \check{\boldsymbol{\alpha}}} = -\varepsilon \boldsymbol{I} - (\hat{\boldsymbol{\alpha}} - \check{\boldsymbol{\alpha}})\boldsymbol{F} \odot \boldsymbol{K} - \boldsymbol{y}\,. \end{cases} \tag{25}$$

Formally, $h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})$ is proven to be gradient-Lipschitz continuous by the following theorem.

Theorem 10 *The function $h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})$ with its partial derivatives in Eq.* (25) *is gradient-Lipschitz continuous, i.e., for any $\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1, \hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2 \in \mathcal{A}$, let the concentration vectors be $\tilde{\boldsymbol{\alpha}}_1 = [\hat{\boldsymbol{\alpha}}_1^\top, \check{\boldsymbol{\alpha}}_1^\top]^\top$ and $\tilde{\boldsymbol{\alpha}}_2 = [\hat{\boldsymbol{\alpha}}_2^\top, \check{\boldsymbol{\alpha}}_2^\top]^\top$, and the partial derivatives be*

$$\begin{aligned}\nabla_{\tilde{\boldsymbol{\alpha}}_1} h(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1) &= \left[\Big(\frac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}_1)}{\partial \hat{\boldsymbol{\alpha}}}\Big|_{\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}}_1}\Big)^\top, \Big(\frac{\partial h(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}})}{\partial \check{\boldsymbol{\alpha}}}\Big|_{\check{\boldsymbol{\alpha}} = \check{\boldsymbol{\alpha}}_1}\Big)^\top\right]^\top, \\ \nabla_{\tilde{\boldsymbol{\alpha}}_2} h(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2) &= \left[\Big(\frac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}_2)}{\partial \hat{\boldsymbol{\alpha}}}\Big|_{\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}}_2}\Big)^\top, \Big(\frac{\partial h(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}})}{\partial \check{\boldsymbol{\alpha}}}\Big|_{\check{\boldsymbol{\alpha}} = \check{\boldsymbol{\alpha}}_2}\Big)^\top\right]^\top,\end{aligned}$$

we have

$$\|\nabla_{\tilde{\boldsymbol{\alpha}}_1} h(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1) - \nabla_{\tilde{\boldsymbol{\alpha}}_2} h(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2)\|_2 \le 2L\Big(\|\hat{\boldsymbol{\alpha}}_2 - \hat{\boldsymbol{\alpha}}_1\|_2 + \|\check{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_1\|_2\Big)\,.$$

where the Lipschitz constant is $L = 2\kappa\Big(n + \frac{9nC^2\|\boldsymbol{K}\|_{\mathrm{F}}}{4\eta}\Big)$.

Proof The proofs can be found in Appendix A.3. ■

Based on the gradient-Lipschitz continuity of $h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})$ demonstrated by Theorem 10, we are ready to present the first-order Nesterov's smooth optimization method for problem (22). The smooth optimization algorithm is summarized in Algorithm 2.

Algorithm 2: Projected gradient method with Nesterov's acceleration for problem (22)

Input: The kernel matrix $\boldsymbol{K}$, the label matrix $\boldsymbol{Y}$, and the Lipschitz constant L derved in Theorem 10

Output: The optimal $\boldsymbol{\alpha}^*$

Set the stopping criteria $t_{\max} = 2000$ and $\epsilon = 10^{-4}$.

Initialize $t = 0$ and $\hat{\boldsymbol{\alpha}}^{(0)}, \check{\boldsymbol{\alpha}}^{(0)} \in \mathcal{A} := \mathbf{0}$.

Repeat

  Compute $\boldsymbol{F}(\hat{\boldsymbol{\alpha}}^{(t)}, \check{\boldsymbol{\alpha}}^{(t)}) = \mathcal{J}_{\frac{\tau}{2}}\left(\mathbf{1}\mathbf{1}^\top + \boldsymbol{\Gamma}(\hat{\boldsymbol{\alpha}}^{(t)}, \check{\boldsymbol{\alpha}}^{(t)})\right)$;

  Compute $\partial h/\partial \hat{\boldsymbol{\alpha}}$ and $\partial h/\partial \check{\boldsymbol{\alpha}}$ by Eq. (25), and concentrate them as $\nabla h(\hat{\boldsymbol{\alpha}}^{(t)}, \check{\boldsymbol{\alpha}}^{(t)}) = [(\partial h/\partial \hat{\boldsymbol{\alpha}})^\top, (\partial h/\partial \check{\boldsymbol{\alpha}})^\top]^\top$;

  Compute $\boldsymbol{\theta}^{(t)} = \mathcal{P}_{\mathcal{A}}\Big([\hat{\boldsymbol{\alpha}}^{(t)\top}, \check{\boldsymbol{\alpha}}^{(t)\top}]^\top + \frac{1}{2L}\nabla h(\hat{\boldsymbol{\alpha}}^{(t)}, \check{\boldsymbol{\alpha}}^{(t)})\Big)$;

  Compute $\boldsymbol{\beta}^{(t)} = \mathcal{P}_{\mathcal{A}}\Big([\hat{\boldsymbol{\alpha}}^{(0)\top}, \check{\boldsymbol{\alpha}}^{(0)\top}]^\top - \frac{1}{4L}\sum_{i=0}^{t}(i+1)\nabla h(\hat{\boldsymbol{\alpha}}^{(i)}, \check{\boldsymbol{\alpha}}^{(i)})\Big)$;

  Set $[\hat{\boldsymbol{\alpha}}^{(t+1)\top}, \check{\boldsymbol{\alpha}}^{(t+1)\top}]^\top = \frac{t+1}{t+3}\boldsymbol{\theta}^{(t)} + \frac{2}{t+3}\boldsymbol{\beta}^{(t)}$;

  Set $\boldsymbol{\alpha}^{(t+1)} = \hat{\boldsymbol{\alpha}}^{(t+1)} - \check{\boldsymbol{\alpha}}^{(t+1)}$ and $t := t + 1$;

Until $t \geq t_{\max}$ *or* $\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)}\|_2 \leq \epsilon$;

6. Experimental Results

This section evaluates the performance of our DANK model in comparison with several representative kernel learning algorithms on classification and regression benchmark data sets. All the experiments implemented in MATLAB are conducted on a Workstation with an Intel® Xeon® E5-2695 CPU (2.30 GHz) and 64GB RAM. The source code of our DANK model in Algorithm 1 can be found in `http://www.lfhsgre.org`.

6.1 Classification Tasks

We conduct experiments on the UCI Machine Learning Repository with small scale data sets, [4] and three large data sets including *EEG*, *ijcnn1* and *covtype*.[5] Besides, we also compare these methods on the *CIFAR-10* database for image classification.[6]

6.1.1 CLASSIFICATION RESULTS ON UCI DATABASE

Ten small data sets from the UCI database are used to evaluate our DANK model embedded in SVM. Here we describe experimental settings and the compared algorithms as follows.

Experimental Settings: Table 2 lists a brief description of these ten data sets including the number of training data n and the feature dimension d. After normalizing the data to $[0, 1]^d$ by a min-max scaler, we randomly pick half of the data for training and the rest for test except for *monks1*, *monks2*, and *monks3*. In these three data sets, both training and test data have been provided. The Gaussian kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/2\sigma^2)$ is chosen as the initial kernel in our model. The kernel width σ and the balance parameter C are tuned by 5-fold cross validation on a grid of

4. `https://archive.ics.uci.edu/ml/datasets.html`
5. All datasets are available at `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
6. `https://www.cs.toronto.edu/~kriz/cifar.html`

Table 1: Comparison results with two baselines in terms of classification accuracy (mean±std. deviation %) on ten UCI data sets. The best performance is highlighted in bold. The classification accuracy on the training data is presented by italic, and does not participate in ranking. Notation "•" indicates that our DANK method is significantly better than other baseline methods via paired t-test at the 5% significance level.

Data set	(d, n)	MKL-uniform		SVM-CV		DANK	
		Training	Test	Training	Test	Training	Test
diabetic	(19, 1151)	*83.5±14.2*	61.9±8.5	*80.7±3.9*	73.0±1.7	*87.0±1.9*	81.2±1.4•
heart	(13, 270)	*95.8±4.3*	82.1±2.9	*88.9±3.0*	81.9±2.4	*94.3±1.7*	**87.9**±2.9•
monks1	(6, 124)	*99.3±1.3*	79.0±2.8	*90.3±0.0*	81.4±0.0	*100.0±0.0*	**83.6**±1.5•
monks2	(6, 169)	*100.0±0.0*	84.4±0.1	*100.0±0.0*	85.8±1.4	*100.0±0.0*	**86.7**±0.9
monks3	(6, 122)	*99.6±1.3*	90.8±0.6	*96.2±1.5*	**93.0**±1.2	*97.2±1.8*	**93.0**±0.9
sonar	(60, 208)	*100.0±0.0*	80.8±3.7	*99.9±0.3*	85.3±3.1	*100.0±0.0*	**87.0**±2.7•
spect	(21, 80)	*93.8±0.0*	**79.8**±0.3	*87.0±3.7*	73.1±3.2	*93.4±4.1*	78.9±3.2
glass	(9, 214)	*93.4±6.0*	53.0±7.0	*77.1±6.9*	69.8±2.0	*89.7±6.1*	**74.5**±1.3•
fertility	(9, 100)	*99.0±1.4*	85.6±3.0	*94.4±5.3*	85.2±1.7	*97.3±3.3*	**87.6**±2.3•
wine	(13, 178)	*100.0±0.0*	96.0±3.8	*99.5±1.0*	94.7±1.5	*99.5±1.0*	**96.4**±2.0

points, i.e., $\sigma = [2^{-5}, 2^{-4}, \ldots, 2^5]$ and $C = [2^{-5}, 2^{-4}, \ldots, 2^5]$. To avoid additional cross validation, we manually set the penalty parameter τ to 0.01. The regularization parameter η is fixed to $\|\boldsymbol{\alpha}\|_2^2$ obtained by SVM. The experiments are conducted 10 times on these ten data sets.

Compared Methods: We include the following kernel learning based algorithms:

- BMKL (Gonen, 2012): A multiple kernel learning algorithm uses Bayesian approach to ensemble the Gaussian kernels with ten different kernel widths and the polynomial kernels with three different degrees.

- LogDet (Jain et al., 2012): A nonparametric kernel learning approach aims to learn a PSD matrix $\boldsymbol{W}$ in a learned kernel $\phi(\boldsymbol{x})^\top \boldsymbol{W} \phi(\boldsymbol{x}')$ with the LogDet divergence.

- RF (Sinha and Duchi, 2016): A kernel alignment based learning framework creates randomized features, and then solves a simple optimization problem to select a subset. Finally, the kernel is learned from the optimized features by target alignment.

- KNPL (Liu et al., 2018): This nonparametric kernel learning framework is given by our conference version, which shares the initial ideas about learning in a data-adaptive scheme via a pair-wise way. But this work does not consider the bounded constraint and the low-rank structure on $\boldsymbol{F}$, and utilizes an alternating iterative algorithm to solve the corresponding semi-definite programming.

- MKL-uniform: It is a multiple kernel learning algorithm with uniform weights, and serves as a baseline. It uses 11 equal-weighted Gaussian kernels with the kernel width $\sigma = [2^{-5}, 2^{-4}, \ldots, 2^5]$, respectively. This setting avoids tuning the kernel width σ but the balance parameter C is still tuned by 5-fold cross validation.

- SVM-CV: The SVM classifier with cross validation serves as a baseline.

Table 2: Comparison results of several representative kernel learning based algorithms in terms of test accuracy on ten UCI dataaset. The best performance is highlighted in bold. Notation "•" indicates that the data-adaptive based algorithm (KNPL or DANK) is significantly better than other representative kernel learning methods via paired t-test at the 5% significance level.

Data set	LogDet	BMKL	RF	KNPL	DANK
diabetic	78.7±1.8	74.9±0.4	72.3±0.8	**81.9**±1.7•	81.2±1.4•
heart	80.6±3.5	85.6±0.8	79.1±2.4	87.4±3.9•	**87.9**±2.9•
monks1	**86.6**±0.2	78.9±2.5	84.4±0.9	83.3±3.3	83.6±1.5
monks2	85.9±1.2	82.1±1.3	73.6±1.1	83.3±1.6	**86.7**±0.9
monks3	**94.0**±1.3	**94.0**±1.0	93.7±0.6	88.7±1.2	93.0±0.9
sonar	84.1±2.2	84.8±0.6	80.5±3.1	85.8±2.8	**87.0**±2.7•
spect	79.6±3.7	78.8±0.8	76.0±2.7	**79.7**±4.8	78.9±3.2
glass	71.2±1.0	68.2±4.6	68.2±2.2	72.4±2.3	**74.5**±1.3•
fertility	86.2±1.1	84.4±1.6	84.4±4.3	85.6±3.8	**87.6**±2.3•
wine	96.1±1.8	95.0±2.8	95.1±1.1	96.1±2.0	**96.4**±2.0

Experimental Results: We first evaluate the proposed DANK model with two baselines: MKL-uniform and SVM-CV in terms of classification accuracy on the training and test data in Table 1, and then compare DANK with other representative kernel learning based algorithms in Table 2.

In Table 1, MKL-uniform sometimes performs the best on the training data, but fails to generalize on the test data in most cases. In general, it is inferior to SVM-CV and our DANK model in terms of the test accuracy. Directly enlarging the solving space without any constraint would increase model flexibility but is easy to be over-fitting. Compared with the baseline SVM-CV, the proposed DANK model achieves a good trade-off between model flexibility and complexity. Training accuracy on *diabetic*, *heart*, *monks1*, *spect*, and *glass* indicates the effectiveness of our data adaptive scheme on increasing the model flexibility. Accordingly, this strategy is helpful for our model to achieve noticeable improvements on the test data. On the *monks2*, *sonar*, and *wine* data sets, SVM-CV has already obtained nearly 100% accuracy on the training data, which indicates that the model flexibility is sufficient. In this case, it is difficult for our DANK method to achieve a huge improvement on these data sets, and accordingly the performance margins are about 0%$\sim$2%. Table 2 reports the test accuracy of typical kernel learning based algorithms. We also apply the paired t-test at the 5% significance level to investigate whether the data-adaptive approaches (KNPL and DANK) are significantly better than other methods. It can be found that, compared with representative kernel learning based algorithms including LogDet, BMKL, and RF, the proposed DANK model yields favorable performance.

In general, the improvements on the classification accuracy demonstrate the effectiveness of our data adaptive scheme, and accordingly our model has good adaptivity to the training and test data.

6.1.2 RESULTS ON LARGE-SCALE DATA SETS

To validate our decomposition scheme on large scale situations, we consider three large data sets including *EEG*, *ijcnn1*, and *covtype* for comparisons. Table 3 reports the data set statistics (i.e., the feature dimension d, the number of training samples, and the number of test data) and parameter settings including the balance parameter C, the kernel width σ, and the number of clusters. Table 4 presents the test accuracy and training time of various compared algorithms including LogDet,

Table 3: Large-sample data set statistics and parameter settings.

data sets	d	#training	#test	C	$1/2\sigma^2$	#clusters
EEG	14	7,490	7,490	32	100	5
ijcnn1	22	49,990	91,701	32	2	50
covtype	54	464,810	116,202	32	32	200

Table 4: Comparison of test accuracy and training time of all the compared algorithms on several large data sets. The rank of $\boldsymbol{F}$ in our DANK model is also given in bold.

Method		SVM-SMO	LogDet		BMKL		RF	DANK	
Data set	Setting	exact	exact	scalable	exact	scalable	exact	exact(rank.)	scalable(rank.)
EEG	acc.(%)	95.9	95.9	95.2	94.5	94.1	80.1	96.7(**36**)	96.3(**142**)
	time(sec.)	8.6	211.6	24.6	1426.3	124.5	2.4	473.6	39.8
ijcnn1	acc.(%)	96.5	97.9	97.4	98.5	97.7	93.0	98.9(**342**)	98.4(**1788**)
	time(sec.)	112.4	8472.3	78.1	109967	1916.9	25.0	28548	571.7
covtype	acc.(%)	96.1	×[1]	96.4	×	91.2	79.1	×	97.1
	time(sec.)	3972.5	×	5021.4	×	94632	364.5	×	7534.2

[1] These methods attempt to directly solve the optimization problem on *covtype* but fail due to the memory limit.

BMKL, our DANK method, SVM-SMO (Platt, 1998) (the cache is set to 5000) and RF conducted in the following two settings.

In the first setting ("exact"), we attempt to directly test these algorithms over the entire training data. Experimental results indicate that, without the decomposition-based scalable approach, our DANK method achieves the best test accuracy with 96.7% sand 98.9% on *EEG* and *ijcnn1*, respectively. However, under this setting, LogDet, BMKL, and our method fail to deal with an extreme large data set *covtype* due to the memory limit except SVM-SMO and RF.

In the second setting ("scalable"), we incorporate our kernel approximation scheme into LogDet, BMKL, and DANK evaluated on the three large data sets. Experimental results show that, by such decomposition-based scalable approach, we can speed up the above three kernel learning methods. For example, when compared with the direct solution of the optimization problem in the "exact" setting, LogDet, BMKL, and DANK equipped with kernel approximation speed up about 100x, 50x, and 50x on *ijcnn1*, respectively. More importantly, on these three data sets, our DANK method using the approximation scheme still performs better than SVM-SMO on the test accuracy, which demonstrates the effectiveness of the proposed non-parametric kernel learning framework.

Results in above two settings show that, our DANK method achieves promising test accuracy no matter whether the kernel approximation scheme is incorporated or not. Moreover, such approximation scheme makes BMKL, LogDet, and our DANK method feasible to large data sets with huge speedup in terms of computational efficiency.

6.1.3 RESULTS ON CIFAR-10 DATA SET

In this section, we test our model on a representative data set *CIFAR-10* (Krizhevsky and Hinton, 2009) for natural image classification task. This data set contains 60,000 color images with the size of $32 \times 32 \times 3$ in 10 categories, of which 50,000 images are used for training and the rest are for testing.

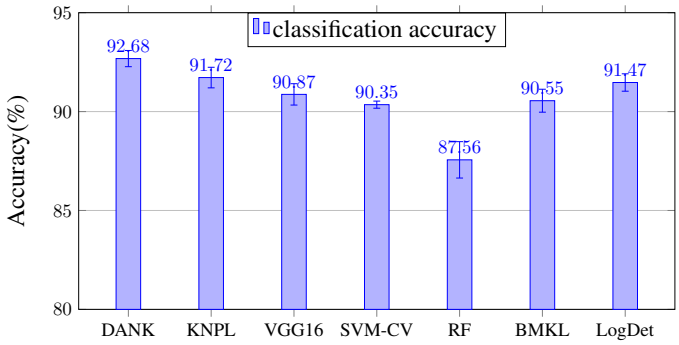


Figure 3: Performance of the compared algorithms on *CIFAR-10* data set.

In our experiment, each color image is represented by the feature extracted from a convolutional neural network, i.e., VGG16 with batch normalization (Ioffe and Szegedy, 2015) pre-trained on ImageNet (Deng et al., 2009). Then we fine-tune this pre-trained VGG16 model on the CIFAR10 data set with 240 epochs and a min-batch size of 64. The learning rate starts from 0.1 and then is divided by 10 at the 120-th, 160-th, and 200-th epoch. After that, for each image, a 4096 dimensional feature vector is obtained according to the output of the first fully-connected layer in this fine-tuned neural network.

Figure 3 shows the test accuracy (mean$\pm$std. deviation %) of the compared algorithms averaged in ten trials. The original VGG16 model with the softmax classifier achieves 90.87$\pm$0.54% on the test accuracy. Using the extracted 4096-D feature vectors, kernel learning based algorithms equipped with the initial Gaussian kernel are tuned by 5-fold cross validation on a grid of points, i.e., $\sigma = [0.001, 0.01, 0.1, 1]$ and $C = [1, 10, 20, 30, 40, 50, 80, 100]$. In terms of classification performance, SVM-CV, BMKL, LogDet, and KNPL obtain 90.35$\pm$0.18%, 90.55$\pm$0.58%, 91.47$\pm$0.44%, and 91.72$\pm$0.52% accuracy on the test data, respectively. Comparably, our DANK model achieves promising classification accuracy with 92.68$\pm$0.41%. More importantly, it outperforms SVM-CV with an accuracy margin of 2.33%, and is *statistically significant* better than the other methods via paired t-test at the 5% significance level. The improvement over SVM-CV on the test accuracy demonstrates that our DANK method equipped with the introduced kernel adjustment strategy is able to enhance the model flexibility, and thus achieves good performance.

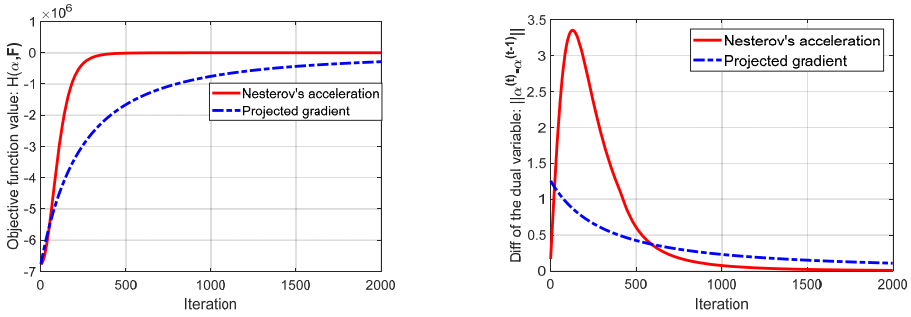
6.2 Analysis and Validation for Theoretical Results

In this subsection, we experimentally validate the effectiveness of the used Nesterov's smooth optimization method, the rationality of dropping out the low-rank regularizer in the large sample case, the robustness of τ via parameter sensitivity analysis, and the tight bound of our theoretical results.

6.2.1 CONVERGENCE EXPERIMENTS

To investigate the effectiveness of the used Nesterov's smooth optimization method, we conduct a convergence experiment on the *heart* data set.

In Figure 4(a), we plot the objective function value $H(\boldsymbol{\alpha}, \boldsymbol{F})$ versus iteration by the standard projected gradient method (in blue dashed line) and its Nesterov's acceleration (in red solid line),



(a) $H(\boldsymbol{\alpha}, \boldsymbol{F})$ (b) $\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)}\|_2$

Figure 4: Comparison between projected gradient method and our Nesterov's acceleration on the *heart* data set.

Table 5: Influence of the low-rank constraint with different values of τ on test classification accuracy. Notation "•" indicates that test accuracy by this setting τ is statistical different from that of the current setting $\tau = 0.01$ via paired t-test at the 5% significance level. The rank of $\boldsymbol{F}$ in our DANK model is also given in bold.

τ	0	0.001	0.01	0.1	1
diabetic	78.1±1.2 (**340**)•	81.1±1.2 (**10**)	81.2±1.4 (**5**)	81.2±1.2 (**2**)	81.1±1.2 (**2**)
heart	84.9±2.2 (**109**)•	87.3±2.2 (**5**)	87.9±2.9 (**3**)	86.5±2.5 (**2**)•	85.4±2.8 (**1**)•
monks1	82.1±1.3 (**68**)•	82.5±1.4 (**15**)•	83.6±1.5 (**6**)	83.2±1.4 (**2**)	81.7±1.3 (**1**)•
monks2	85.5±0.8 (**68**)•	86.2±0.7 (**13**)•	86.7±0.9 (**4**)	86.4±0.8 (**2**)	86.0±0.9 (**2**)•
monks3	90.2±1.3 (**61**)•	92.8±1.0 (**8**)	93.0±0.9 (**4**)	93.3±1.1 (**2**)	92.8±1.1 (**1**)
sonar	84.0±2.4 (**81**)•	85.2±2.4 (**32**)•	87.0±2.7 (**17**)	87.4±2.6 (**7**)	86.8±2.3 (**3**)•
spect	78.2±2.6 (**15**)	78.4±2.8 (**8**)	78.9±3.2 (**6**)	78.4±2.8 (**3**)	76.4±2.5 (**2**)•
glass	72.3±1.1 (**34**)•	73.7±1.4 (**14**)•	74.5±1.3 (**8**)	73.3±1.2 (**3**)•	71.4±1.5 (**2**)•
fertility	86.8±2.1 (**29**)•	87.1±2.2 (**11**)	87.6±2.3 (**5**)	87.3±2.1 (**3**)	87.2±2.2 (**2**)
wine	96.1±2.0 (**5**)	96.2±1.9 (**3**)	96.4±2.0 (**2**)	96.3±2.0 (**2**)	96.3±2.0 (**2**)

respectively. One can see that the developed first-order Nesterov's smooth optimization method converges faster than the projected gradient method, so the feasibility of employing Nesterov's acceleration for solving problem (4) is verified. Besides, to further illustrate the convergence of $\{\boldsymbol{\alpha}^{(t)}\}_{t=0}^{\infty}$, we plot $\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)}\|_2$ versus iteration in Figure 4(b). We find that the sequence $\{\boldsymbol{\alpha}^{(t)}\}_{t=0}^{\infty}$ yielded by the Nesterov's acceleration algorithm significantly decays in the first 500 iterations, which leads to a fast convergence to the optimal solution. Hence, compared with projected gradient method, the Nesterov's smooth optimization method is able to efficiently solve the targeted convex optimization problem in this paper.

6.2.2 VALIDATION FOR THE LOW-RANK CONSTRAINT

Due to the inseparable property of the low-rank constraint, we omit the low-rank regularizer on $\boldsymbol{F}$ for efficient optimization in the large sample case. Here we experimentally specialize in the influence of $\|\boldsymbol{F}\|_*$ on the test classification accuracy in both small and large data sets.

Regarding to the small sample case, Table 5 reports the classification accuracy and the rank of $\boldsymbol{F}$ under different values of τ on these ten data sets appeared in Section 6.1.1. Compared to our model

with $\tau = 0.01$, the setting without the low-rank regularizer (i.e., $\tau = 0$) loses about 1%$\sim$3% accuracy on most data sets with statistical difference except for the *spect* and *wine* data sets. On these two data sets, we find that, although the low-rank regularizer is not considered, the learned $\boldsymbol{F}$ still exhibits the low-rank structure, which effectively controls the model flexibility and complexity. Accordingly, our model without the low-rank constraint performs well on the two data sets. However, in the remaining data sets, the rank of $\boldsymbol{F}$ significantly increases if we drop out the low-rank regularizer. In this case, the solving space of our model could be extreme large, which would lead to over-fitting. Besides, the learned $\boldsymbol{F}$ might be sophisticated, therefore, it is not easily extended to $\boldsymbol{F}'$ for test data by the simple nearest neighbor scheme. Hence, results in Table 5 indicates that the low-rank constraint is important in small-sample data sets.

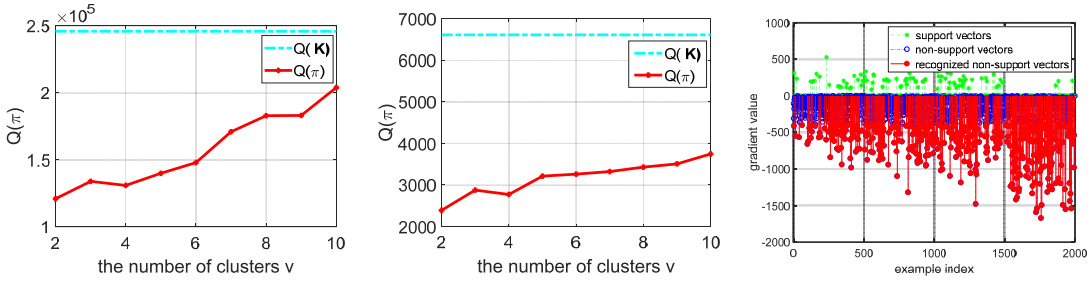
In terms of large sample case, according to Table 4, our DANK model with the "exact" solution achieves 96.7% and 98.9% accuracy on the *EEG* and *ijcnn1* data set, respectively. In contrast, after omitting the low-rank regularizer $\|\boldsymbol{F}\|_*$, the test accuracy of our model in the "scalable" setting decreases to 96.3% and 98.4%, respectively. More importantly, we find that, without the low-rank constraint, the rank of $\boldsymbol{F}$ increases from 36 (in the "exact" setting) to 142 (in the "scalable" setting) on the *EEG* data set with 7,490 training samples. This tendency also exhibits on the *ijcnn1* data set with 49,900 training samples. The rank of $\boldsymbol{F}$ on this data set increases from $\text{rank}(\boldsymbol{F}) = 342$ to $\text{rank}(\boldsymbol{F}) = 1788$. So the above results indicate that dropping the low-rank constraint leads to a slight decrease on the test accuracy; while the rank of $\boldsymbol{F}$ is indeed raising but is still much smaller than n. In the next, we explain the reason why our model still obtains a relative low-rank matrix $\boldsymbol{F}$ in the large sample case. On one hand, the regularization parameter η is chosen as $\eta := \|\boldsymbol{\alpha}\|_2^2 \in \mathcal{O}(n)$ in our experiment. This would be an extreme large value in large sample data sets, resulting in a strong regularization term $\eta\|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}$. Therefore, $\boldsymbol{F}$ varies around the all-one matrix in a small bounded region and thus shows a low-rank effect to some extent. On the other hand, the used Gaussian kernel inherits the rapid decaying spectra (Smola and Schölkopf, 2000), e.g., the exponential decay $\lambda_i \propto ne^{-ai}$ with $a > 0$ as illustrated by Bach (2013). As discussed in Section 2.2.2, when the Gaussian kernel is chosen as the initial one, the used centering regularizer is helpful to obtain a relative low-rank kernel matrix $\widetilde{\boldsymbol{K}}$. Based on the above analysis, in large sample case, using the centering regularizer $\eta\|\boldsymbol{F} - \mathbf{1}\mathbf{1}^\top\|_{\mathrm{F}}$ could be an alternative choice for the low-rank property if we have to drop $\|\boldsymbol{F}\|_*$.

From above observations and analyses, we conclude that the low-rank constraint in our model is important for small-sample cases. In large-sample data sets, due to the separability requirement, our DANK model has to drop the low-rank constraint, and thus achieving a slight decrease on the final classification accuracy. Nevertheless, the employed centering regularizer restricts $\boldsymbol{F}$ to a small bounded region around the all-one matrix, which would be an alternative choice to seek for a low-rank matrix $\bar{\boldsymbol{F}}$.

6.2.3 PARAMETER SENSITIVITY ANALYSIS

The above subsection demonstrates the importance of the low-rank regularizer in our DNAK model on small sample case, so here we need to investigate the parameter sensitivity of τ to the test accuracy on these data sets.

Table 5 reports the classification accuracy with τ in $\{0, 0.001, 0.01, 0.1, 1\}$. We find that, when τ is changed from 0 to 0.01, the rank of $\boldsymbol{F}$ rapidly decreases, which implies that the model flexibility is effectively controlled. The test accuracy yielded by our DANK model is accordingly improved



(a) *ijcnn1* (b) *covtype* (c) non-support vectors

Figure 5: Experimental validation of the derived bounds in Theorem 6 and 7.

under this range. When τ further increases to $\tau = 0.1$, there is no *statistical significance* found between $\tau = 0.01$ and $\tau = 0.1$ except on the *heart* and *glass* data sets in terms of classification accuracy. Furthermore, if τ increases to 1, $\boldsymbol{F}$ exhibits an extremely low-ranking structure. In this case, our model with $\tau = 1$ is not flexible enough to fit the data, and thus is inferior to the setting of $\tau = 0.01$ on most data sets.

Based on the above observations, our DANK model is robust to the variations of τ ranging from 0.01 to 0.1, so it can be easily tuned and we suggest $\tau = 0.01$ for practical use.

6.2.4 VALIDATION FOR OUR DERIVED BOUNDS

Here we show that the derived bounds in Theorem 6 and 7 are tight in practice.

We firstly study the relationship between $Q(\pi)$ and v in our model as shown in Figure 5. In our experiments, we randomly select 2,000 samples from the *ijcnn1* and *covtype* data set, and then group them into $v = 2, 3, \cdots, 10$ clusters, respectively. It can be observed that $Q(\pi)$ almost increases linearly with v on these two data sets, which shows consistency with theoretical results in (Giraud and Verzelen, 2019) to some extent. Further, if we consider more clusters, such as $v = 100, 200, \cdots$ (in practice, this situation is rare in clustering algorithms with only 2,000 samples but over 100 clusters), $Q(\pi)$ would increase slowly to approach to the upper bound $Q(\boldsymbol{K}) := \sum_{i,j=1}^{n} |K_{ij}|$.

In Figure 5(c), we validate that the derived condition in Theorem 7 is useful for screening non-support vectors. We follow with the above experimental setting with $v = 10$, and obtain the approximation solution $(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$. Then we compute the gradient $\left(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\right)_i$ associated with each sample and plot the support vectors (in green) and non-support vectors (in blue) as shown in Figure 5(c). Under the condition of the condition (21) in Theorem 7, the non-support vectors that satisfy $\left(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\right)_i \leq (B+B_2)C\left(\|\bar{\boldsymbol{K}}_i\|_1+\kappa\right) = 442.3$ can be directly recognized as non-support vectors (marked in red) without solving the optimization problem. We find that in the total of #SVs=214 (the number of support vectors) and #non-SVs=1786 (the number of non-support vectors), there are 723 non-support vectors recognized by our Theorem 7. That means, over 40% non-support vectors can be picked out, which demonstrates the effectiveness of our derived bound/condition in Theorem 7.

6.3 Regression

This section focuses on the proposed DANK model embedded in SVR for regression tasks. We firstly conduct the experiments on several synthetic data sets, to examine the performance of our method on recovering 1-D and 2-D test functions. Then we evaluate our model with representative regression algorithms on real-world UCI data sets.[7] The used evaluation metric here is relative mean square error (RMSE) between the learned regression function $\hat{g}(\boldsymbol{x})$ and the target label $\boldsymbol{y}$ over n data points

$$\text{RMSE} = \frac{\sum_{i=1}^{n} \big(\hat{g}(\boldsymbol{x}_i) - y_i\big)^2}{\sum_{i=1}^{n} \big(y_i - \mathbb{E}(\boldsymbol{y})\big)^2}\,.$$

6.3.1 SYNTHETIC DATA

Here we test the approximation performance of our method on 1-D and 2-D test functions compared with a baseline, the SVR with Gaussian kernel. The representative 1-D step function is defined by

$$g(s, w, a, x) = \left(\frac{\tanh\left(\frac{ax}{w} - a\lfloor \frac{x}{w} \rfloor - \frac{a}{2}\right)}{2\tanh\left(\frac{a}{2}\right)} + \frac{1}{2} + \left\lfloor \frac{x}{w} \right\rfloor \right) s\,,$$

where s is the step hight, w is the period, and a controls the smoothness of the function g. In our experiment, s, w and a are set to 3, 2 and 0.05, respectively. We plot the step function on $[-5, 5]$ as shown in Figure 6(a). One can see that the approximation function generated by SVR-CV (blue dashed line) yields a larger deviation than that of our DANK model (red solid line). To be specific, the RMSE of SVR is 0.013, while our DANK model achieves a promising approximation error with a value of 0.004.

Apart from the 1-D function, we use a 2-D test function to test SVR-CV and the proposed DANK model. The 2-D test function (Cherkassky et al., 1996) $g(u, v) \in [-0.5, 0.5] \times [-0.5, 0.5]$ is established as

$$g(u, v) = 42.659\Big(0.1 + (u - 0.5)\big(g_1(u, v) + 0.05\big)\Big)\,,$$

where $g_1(u, v)$ is defined by

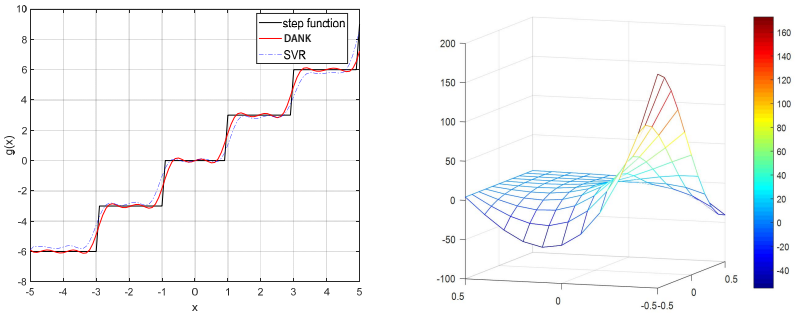
$$g_1(u, v) = (u - 0.5)^4 - 10(u - 0.5)^2(v - 0.5)^2 + 5(v - 0.5)^4\,.$$

We uniformly sample 400 data points by $g(u, v)$ as shown in Figure 6(b), and then use SVR-CV and DANK to learn a regression function from the sampled data. The regression results by SVR-CV and our DANK model are shown in Figure 6(c) and Figure 6(d), respectively. Intuitively, when we focus on the upwarp of the original function, our method is more similar to the test function than SVR-CV. In terms of RMSE, the error of our method for regressing the test function is 0.007, which is lower than that of SVR-CV with 0.042.

6.3.2 REGRESSION RESULTS ON UCI DATA SETS

We compare the proposed DANK model with other representative regression algorithms on eight data sets from the UCI database. Gonen (2012) extend BMKL to regression tasks, and thus we include it for comparisons. Apart from SVR-CV and BMKL, the Nadaraya-Watson (NW) estimator

7. The data sets are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`



(a) recovering a step function

(b) a 2-D function

(c) recovered by SVR-CV

(d) recovered by DANK

Figure 6: Approximation for 1-D and 2-D functions by SVR-CV and our DANK model.

Table 6: Comparison results of various methods on UCI regression data sets in terms of RMSE (mean±std. deviation). The best performance is highlighted in bold. The RMSE on the training data is presented by italic, and does not participate in ranking. Notation "•" indicates that the method is significantly better than other methods via paired t-test at the 5% significance level.

Data set	(d, n)	BMKL	NW	SVR-CV		DANK	
		Test	Test	Training	Test	Training	Test
bodyfat	(14, 252)	0.101±0.065	0.097±0.010	*0.007±0.000*	0.101±0.002	*0.008±0.000*	**0.091**±0.013
pyrim	(27, 74)	0.512±0.115	0.604±0.115	*0.013±0.010*	0.678±0.255	*0.007±0.004*	**0.457**±0.143•
space	(6, 3107)	0.248±0.107	0.308±0.004	*0.226±0.144*	0.261±0.112	*0.106±0.067*	**0.202**±0.114•
triazines	(60, 186)	**0.725**±0.124	0.885±0.039	*0.113±0.084*	0.815±0.206	*0.009±0.012*	0.734±0.128
cpusmall	(12, 8912)	0.133±0.004	**0.037**±0.009	*0.037±0.004*	0.104±0.002	*0.001±0.000*	0.114±0.003
housing	(13, 506)	0.286±0.027	**0.177**±0.015•	*0.085±0.049*	0.267±0.023	*0.069±0.013*	0.218±0.013
mg	(6, 1385)	0.297±0.013	**0.294**±0.010	*0.163±0.076*	0.407±0.118	*0.134±0.055*	0.303±0.058
mpg	(7, 392)	0.187±0.014	0.182±0.012	*0.095±0.087*	0.193±0.006	*0.061±0.010*	**0.173**±0.008•

with metric learning (Noh et al., 2017) is also taken into comparison. The remaining experimental settings follow with the classification tasks on the UCI database illustrated in Section 6.1.1.

Table 6 lists a brief statistics of these eight data sets, and reports the average prediction accuracy and standard deviation of every compared algorithm. Moreover, we present the regression performance of our DANK model and SVR-CV on the training data to show their respective model flexibilities. We find that, the proposed DANK model exhibits more encouraging performance than SVR-CV in terms of the RMSE on the training and test data. From the results on four data sets

including *bodyfat*, *pyrim*, *space*, and *mpg*, we observe that our method statistically achieves the best prediction performance. On the remaining data sets, our DANK model achieves comparable performance with BMKL and NW.

7. Conclusion

In this work, we propose an effective data-adaptive strategy to enhance the model flexibility for nonparametric kernel learning based algorithms. In our DANK model, a multiplicative scale factor for each entry of the Gram matrix can be learned from the data, leading to an improved data-adaptive kernel. As a result of such data-driven scheme, the model flexibility of our DANK model embedded in SVM (for classification) and SVR (for regression) is demonstrated by the experiments on synthetic and real data sets. Besides, by introducing the low-rank constraint/regularizer and the bounded constraint/regularizer into our model, the learned kernel (matrix) exhibits a fast eigenvalue decay, which would be helpful to obtain good generalization properties. Accordingly, the used constraints/regularizers are able to provide a controllable trade-off between model flexibility and complexity. Furthermore, our DANK model can be learned in an one-stage process with $\mathcal{O}(1/t^2)$ convergence rate due to the studied gradient-Lipschitz continuous property, where t is the iteration number. That means, we can simultaneously train the SVM or SVR along with optimizing the data-adaptive matrix by a projected gradient method with Nesterov's acceleration. In addition, we develop a decomposed-based scalable approach to make our DANK model feasible to large data sets, of which the effectiveness has been verified by both experimental results and theoretical demonstration.

Extending our work to the case of deep kernel framework is an exciting avenue for future research, not limited to the "shallow" kernel. When extending to a multi-layer framework, we could optimize the parameters in a layer-by-layer way during the training process.

Acknowledgments

This work was partly supported by National Key Research Development Project (No.2018AAA0100702), the National Natural Science Foundation of China (NSFC, No.61876107, U1803261, 61977046, 61973162), Committee of Science and Technology, Shanghai, China (No. 19510711200), the Fundamental Research Funds for the Central Universities (No: 30920032202), CCF-Tencent Open Fund (No: RAGR20200101), and the "Young Elite Scientists Sponsorship Program" by CAST (No: 2018QNRC001).
We thank Prof. Johan A.K. Suykens for some helpful suggestions on this paper. Jie Yang and Xiaolin Huang are corresponding authors.

A. Proofs for Gradient-Lipschitz Continuity

A.1 Proofs of Lemma 3

Proof Since $\mathcal{J}_{\frac{\tau}{2}}$ is non-expansive (Ma et al., 2011), i.e., for any $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$

$$\|\mathcal{J}_{\frac{\tau}{2}}(\mathbf{\Omega}_1) - \mathcal{J}_{\frac{\tau}{2}}(\mathbf{\Omega}_2)\|_{\mathrm{F}} \leq \|\mathbf{\Omega}_1 - \mathbf{\Omega}_2\|_{\mathrm{F}}\,,$$

with $\boldsymbol{\Omega}_1 := \mathbf{1}\mathbf{1}^\top + \boldsymbol{\Gamma}(\boldsymbol{\alpha}_1)$ and $\boldsymbol{\Omega}_2 := \mathbf{1}\mathbf{1}^\top + \boldsymbol{\Gamma}(\boldsymbol{\alpha}_2)$. Thereby, we have

$$\begin{aligned}
&\big\|\boldsymbol{F}(\boldsymbol{\alpha}_1) - \boldsymbol{F}(\boldsymbol{\alpha}_2)\big\|_{\mathrm{F}} \leq \big\|\boldsymbol{\Gamma}(\boldsymbol{\alpha}_1) - \boldsymbol{\Gamma}(\boldsymbol{\alpha}_2)\big\|_{\mathrm{F}} \\
&= \frac{1}{4\eta}\Big\|\mathrm{diag}\big(\boldsymbol{\alpha}_1^\top \boldsymbol{Y}\big)\boldsymbol{K}\mathrm{diag}\big(\boldsymbol{\alpha}_1^\top \boldsymbol{Y}\big) - \mathrm{diag}\big(\boldsymbol{\alpha}_2^\top \boldsymbol{Y}\big)\boldsymbol{K}\mathrm{diag}\big(\boldsymbol{\alpha}_2^\top \boldsymbol{Y}\big)\Big\|_{\mathrm{F}} \\
&= \frac{1}{4\eta}\Big\|\,\mathrm{diag}\left(\boldsymbol{\alpha}_1{}^\top \boldsymbol{Y} + \boldsymbol{\alpha}_2{}^\top \boldsymbol{Y}\right)\boldsymbol{K}\,\mathrm{diag}\left(\boldsymbol{\alpha}_1{}^\top \boldsymbol{Y} - \boldsymbol{\alpha}_2{}^\top \boldsymbol{Y}\right)\Big\|_{\mathrm{F}} \\
&\leq \frac{1}{4\eta}\|\boldsymbol{K}\|_{\mathrm{F}}\Big\|\,\mathrm{diag}\big(\boldsymbol{\alpha}_1{}^\top \boldsymbol{Y} + \boldsymbol{\alpha}_2{}^\top \boldsymbol{Y}\big)\Big\|_{\mathrm{F}}\Big\|\,\mathrm{diag}\left(\boldsymbol{\alpha}_1{}^\top \boldsymbol{Y} - \boldsymbol{\alpha}_2{}^\top \boldsymbol{Y}\right)\Big\|_{\mathrm{F}} \\
&\leq \frac{\|\boldsymbol{K}\|_{\mathrm{F}}}{4\eta}\big\|\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2\big\|_2\big\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\big\|_2\,,
\end{aligned}$$

which yields the desired result. ■

A.2 Proofs of Theorem 4

For notational simplicity, denote the shortcut $\boldsymbol{F}_1$ for $\boldsymbol{F}(\boldsymbol{\alpha}_1)$ and $\boldsymbol{F}_2$ for $\boldsymbol{F}(\boldsymbol{\alpha}_2)$. We will use them in the subsequent proof.

Proof The gradient of $h(\boldsymbol{\alpha})$ in Eq. (5) is computed as

$$\nabla h(\boldsymbol{\alpha}) = \mathbf{1} - \boldsymbol{Y}\big(\boldsymbol{F}(\boldsymbol{\alpha}) \odot \boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha}\,. \tag{26}$$

It is obvious that $\boldsymbol{F}(\boldsymbol{\alpha})$ is unique over the compact set $\mathcal{F}$. Hence, according to the differentiable property of the optimal value function (Penot, 2004), for any $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{A}$, from the representation of $\nabla h(\boldsymbol{\alpha})$ in Eq. (26), the function $h(\boldsymbol{\alpha})$ is proven to be gradient-Lipschitz continuous

$$\begin{aligned}
\|\nabla h(\boldsymbol{\alpha}_1) - \nabla h(\boldsymbol{\alpha}_2)\|_2 &= \big\|\boldsymbol{Y}\big(\boldsymbol{F}_1 \odot \boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha}_1 - \boldsymbol{Y}\big(\boldsymbol{F}_2 \odot \boldsymbol{K}\big)\boldsymbol{Y}\boldsymbol{\alpha}_2\big\|_2 \\
&= \big\|\boldsymbol{Y}\big(\boldsymbol{F}_1 - \boldsymbol{F}_2\big) \odot \boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha}_1 - \boldsymbol{Y}\big(\boldsymbol{F}_2 \odot \boldsymbol{K}\big)\boldsymbol{Y}\big(\boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_1\big)\big\|_2 \\
&\leq \big\|\boldsymbol{Y}\big(\boldsymbol{F}_1 - \boldsymbol{F}_2\big) \odot \boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha}_1\big\|_2 + \big\|\boldsymbol{Y}\big(\boldsymbol{F}_2 \odot \boldsymbol{K}\big)\boldsymbol{Y}\big(\boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_1\big)\big\|_2 \\
&\leq \big\|\big(\boldsymbol{F}_1 - \boldsymbol{F}_2\big) \odot \boldsymbol{K}\big\|_2\|\boldsymbol{\alpha}_1\|_2 + \big\|\boldsymbol{F}_2 \odot \boldsymbol{K}\big\|_2\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2 \\
&\leq \kappa\big\|\boldsymbol{F}_1 - \boldsymbol{F}_2\big\|_{\mathrm{F}}\|\boldsymbol{\alpha}_1\|_2 + \kappa\|\boldsymbol{F}_2\|_2\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2 \\
&\leq \frac{\kappa\|\boldsymbol{K}\|_{\mathrm{F}}}{4\eta}\|\boldsymbol{\alpha}_1\|_2\big(\|\boldsymbol{\alpha}_1\|_2 + \|\boldsymbol{\alpha}_2\|_2\big)\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2 + \kappa\lambda_{\max}(\boldsymbol{F}_2)\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2 \\
&\leq \kappa\Big(n + \frac{3nC^2\|\boldsymbol{K}\|_{\mathrm{F}}}{4\eta}\Big)\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2\,,
\end{aligned} \tag{27}$$

where the third inequality holds by $\|\boldsymbol{A} \odot \boldsymbol{B}\|_2 \leq \|\boldsymbol{A}\|_2 \max_{ij}|B_{ij}|$ when $\boldsymbol{B}$ is PSD (Johnson, 1990). The fourth equality admits due to Lemma 3 and $\boldsymbol{F}_2 \in \mathcal{F}$. The last equality is achieved by $0 \leq \boldsymbol{\alpha} \leq C,\ \|\boldsymbol{\alpha}\|_2^2 \leq nC^2$ and Lemma 1. Hence, we conclude the proof. ■

A.3 Proofs of Theorem 10

Proof For any $\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1, \hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2 \in \mathcal{A}$, from the representation of $\nabla_{\hat{\boldsymbol{\alpha}}} h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}})$ in Proposition 9, we have

$$\begin{aligned}
&\Big\| \frac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}_1)}{\partial \hat{\boldsymbol{\alpha}}}\Big|_{\hat{\boldsymbol{\alpha}}=\hat{\boldsymbol{\alpha}}_1} - \frac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}_2)}{\partial \hat{\boldsymbol{\alpha}}}\Big|_{\hat{\boldsymbol{\alpha}}=\hat{\boldsymbol{\alpha}}_2} \Big\|_2 = \big\| \big(\boldsymbol{F}_1 - \boldsymbol{F}_2\big) \odot \boldsymbol{K}\big(\hat{\boldsymbol{\alpha}}_1 - \check{\boldsymbol{\alpha}}_1\big) - \big(\boldsymbol{F}_2 \odot \boldsymbol{K}\big)\big(\hat{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_2 - \hat{\boldsymbol{\alpha}}_1 + \check{\boldsymbol{\alpha}}_1\big) \big\|_2 \\
&\quad \le \kappa \big\| \boldsymbol{F}_1 - \boldsymbol{F}_2 \big\|_{\mathrm{F}} \|\hat{\boldsymbol{\alpha}}_1 - \check{\boldsymbol{\alpha}}_1\|_2 + \kappa \|\boldsymbol{F}_2\|_2 \|\hat{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_2 - \hat{\boldsymbol{\alpha}}_1 + \check{\boldsymbol{\alpha}}_1\|_2 \\
&\quad \le \kappa \bigg(\frac{\|\boldsymbol{K}\|_{\mathrm{F}}}{4\eta} \|\hat{\boldsymbol{\alpha}}_1 - \check{\boldsymbol{\alpha}}_1\|_2 \|\hat{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_2 + \hat{\boldsymbol{\alpha}}_1 - \check{\boldsymbol{\alpha}}_1\|_2 + \lambda_{\max}(\boldsymbol{F}_2) \bigg) \|\hat{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_2 - \hat{\boldsymbol{\alpha}}_1 + \check{\boldsymbol{\alpha}}_1\|_2 \\
&\quad \le \kappa \bigg(\frac{8nC^2\|\boldsymbol{K}\|_{\mathrm{F}}}{4\eta} + n + \frac{nC^2}{4\eta} \lambda_{\max}(\boldsymbol{K}) \bigg) \bigg(\|\hat{\boldsymbol{\alpha}}_2 - \hat{\boldsymbol{\alpha}}_1\|_2 + \|\check{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_1\|_2 \bigg) \\
&\quad \le \kappa \Big(n + \frac{9nC^2\|\boldsymbol{K}\|_{\mathrm{F}}}{4\eta} \Big) \bigg(\|\hat{\boldsymbol{\alpha}}_2 - \hat{\boldsymbol{\alpha}}_1\|_2 + \|\check{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_1\|_2 \bigg) .
\end{aligned} \tag{28}$$

Similarly, $\Big\| \frac{\partial h(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}})}{\partial \check{\boldsymbol{\alpha}}}\Big|_{\check{\boldsymbol{\alpha}}=\check{\boldsymbol{\alpha}}_1} - \frac{\partial h(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}})}{\partial \check{\boldsymbol{\alpha}}}\Big|_{\check{\boldsymbol{\alpha}}=\check{\boldsymbol{\alpha}}_2} \Big\|_2$ can also be bounded. Combining these two inequalities, we have

$$\begin{aligned}
\|\nabla_{\tilde{\boldsymbol{\alpha}}_1} h(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}}_1) - \nabla_{\tilde{\boldsymbol{\alpha}}_2} h(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}}_2)\|_2 &\le \Big\| \frac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}_1)}{\partial \hat{\boldsymbol{\alpha}}}\Big|_{\hat{\boldsymbol{\alpha}}=\hat{\boldsymbol{\alpha}}_1} - \frac{\partial h(\hat{\boldsymbol{\alpha}}, \check{\boldsymbol{\alpha}}_2)}{\partial \hat{\boldsymbol{\alpha}}}\Big|_{\hat{\boldsymbol{\alpha}}=\hat{\boldsymbol{\alpha}}_2} \Big\|_2 \\
&\quad + \Big\| \frac{\partial h(\hat{\boldsymbol{\alpha}}_1, \check{\boldsymbol{\alpha}})}{\partial \check{\boldsymbol{\alpha}}}\Big|_{\check{\boldsymbol{\alpha}}=\check{\boldsymbol{\alpha}}_1} - \frac{\partial h(\hat{\boldsymbol{\alpha}}_2, \check{\boldsymbol{\alpha}})}{\partial \check{\boldsymbol{\alpha}}}\Big|_{\check{\boldsymbol{\alpha}}=\check{\boldsymbol{\alpha}}_2} \Big\|_2 \\
&\le 2L \Big(\|\hat{\boldsymbol{\alpha}}_2 - \hat{\boldsymbol{\alpha}}_1\|_2 + \|\check{\boldsymbol{\alpha}}_2 - \check{\boldsymbol{\alpha}}_1\|_2 \Big) ,
\end{aligned} \tag{29}$$

which completes the proof. ■

B. Proofs in Large Scale Situations

B.1 Proofs of Lemma 5

Proof The quadratic term with respect to $\boldsymbol{\alpha}$ in Eq. (18) can be decomposed into

$$\boldsymbol{\alpha}^\top \boldsymbol{Y} \big(\boldsymbol{F} \odot \bar{\boldsymbol{K}} \big) \boldsymbol{Y} \boldsymbol{\alpha} = \sum_{c=1}^{v} \boldsymbol{\alpha}^{(c)^\top} \boldsymbol{Y}^{(c,c)} \big(\boldsymbol{F}^{(c,c)} \odot \boldsymbol{K}^{(c,c)} \big) \boldsymbol{Y}^{(c,c)} \boldsymbol{\alpha}^{(c)} ,$$

and $\|\boldsymbol{F} - \boldsymbol{1}\boldsymbol{1}^\top\|_{\mathrm{F}}^2$ can be expressed as

$$\|\boldsymbol{F} - \boldsymbol{1}\boldsymbol{1}^\top\|_{\mathrm{F}}^2 = \sum_{i,j=1}^{n} (F_{ij} - 1)^2 = \sum_{c=1}^{v} \|\boldsymbol{F}^{(c,c)} - \boldsymbol{1}\boldsymbol{1}^\top\|_{\mathrm{F}}^2 + \text{Const} ,$$

where the constant is the sum of non-block-diagonal elements of $\bar{\boldsymbol{F}}$, and it does not affect the solution of Eq. (20). Specifically, the positive semi-definiteness on $\bar{\boldsymbol{F}}^{(c,c)}$ with $c = 1, 2, \ldots, v$ still guarantees that the whole matrix $\bar{\boldsymbol{F}}$ is PSD. Besides, the constraint in Eq. (20) is also decomposable, so the subproblems are separable and independent. As a result, the concatenation of their optimal solutions yields the optimal solution of Eq. (20). ■

B.2 Proofs of Theorem 6

Proof Denote the objective function in Eq. (20) with the Gram matrix $\bar{\boldsymbol{K}}$ as $\bar{H}(\boldsymbol{\alpha}, \boldsymbol{F})$, its optimal solution $(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$ is indeed a saddle point due to the max-min problem in Eq. (20). It is easy to check $\bar{H}(\boldsymbol{\alpha}, \bar{\boldsymbol{F}}) \leq \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}}) \leq \bar{H}(\bar{\boldsymbol{\alpha}}, \boldsymbol{F})$ for any feasible $\boldsymbol{\alpha}$ and $\boldsymbol{F}$.

Defining $H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)$ and $\bar{H}(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)$, we can easily obtain

$$\bar{H}(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) - H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) = \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \alpha_i^* \alpha_j^* y_i y_j F_{ij}^* K_{ij} \,, \tag{30}$$

with $F_{ij}^* := \boldsymbol{F}^*(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Similarly, we have

$$\bar{H}(\bar{\boldsymbol{\alpha}}, \boldsymbol{F}^*) - H(\bar{\boldsymbol{\alpha}}, \boldsymbol{F}^*) = \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \bar{\alpha}_i \bar{\alpha}_j y_i y_j F_{ij}^* K_{ij} \,.$$

Therefore, combining above equations, the upper bound of $H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) - H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$ can be derived by

$$\begin{aligned}
H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) &\leq \bar{H}(\bar{\boldsymbol{\alpha}}, \boldsymbol{F}^*) - \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \alpha_i^* \alpha_j^* y_i y_j F_{ij}^* K_{ij} \\
&= H(\bar{\boldsymbol{\alpha}}, \boldsymbol{F}^*) - \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \Big(\alpha_i^* \alpha_j^* - \bar{\alpha}_i \bar{\alpha}_j\Big) y_i y_j F_{ij}^* K_{ij} \\
&\leq H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}}) - \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \Big(\alpha_i^* \alpha_j^* - \bar{\alpha}_i \bar{\alpha}_j\Big) y_i y_j F_{ij}^* K_{ij} \\
&\leq H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}}) + \frac{1}{2} BC^2 Q(\pi) \,,
\end{aligned}$$

where the first inequality holds by $\bar{H}(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) \leq \bar{H}(\bar{\boldsymbol{\alpha}}, \boldsymbol{F}^*)$ and Eq. (30). The second inequality admits by $H(\bar{\boldsymbol{\alpha}}, \boldsymbol{F}^*) \leq H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$. Similarly, $H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) - H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})$ is lower bounded by

$$\begin{aligned}
H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) &= \bar{H}(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) - \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \alpha_i^* \alpha_j^* y_i y_j F_{ij}^* K_{ij} \\
&\geq \bar{H}(\boldsymbol{\alpha}^*, \bar{\boldsymbol{F}}) - \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \alpha_i^* \alpha_j^* y_i y_j F_{ij}^* K_{ij} \\
&= H(\boldsymbol{\alpha}^*, \bar{\boldsymbol{F}}) + \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \alpha_i^* \alpha_j^* y_i y_j (\bar{F}_{ij} - F_{ij}^*) K_{ij} \\
&\geq H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}}) + \frac{1}{2} \sum_{i,j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)}^{n} \alpha_i^* \alpha_j^* y_i y_j (\bar{F}_{ij} - F_{ij}^*) K_{ij} \\
&\geq H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}}) - \frac{1}{2} BC^2 Q(\pi) \,,
\end{aligned}$$

which concludes the proof that $\big|H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*) - H(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\big| \leq \frac{1}{2} BC^2 Q(\pi)$. ■

B.3 Proofs of Theorem 7

Proof We decompose $\nabla_{\boldsymbol{\alpha}} H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)$ into

$$\begin{aligned}\Big(\nabla_{\boldsymbol{\alpha}} H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)\Big)_i &= \Big(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\Big)_i + \Big(\boldsymbol{Y}(\boldsymbol{F}^* \odot \bar{\boldsymbol{K}})\boldsymbol{Y}\Delta\boldsymbol{\alpha}\Big)_i + \Big(\boldsymbol{Y}(\Delta\boldsymbol{F} \odot \bar{\boldsymbol{K}})\boldsymbol{Y}\bar{\boldsymbol{\alpha}}\Big)_i \\ &\quad - \sum_{j:\pi(\boldsymbol{x}_i)\neq\pi(\boldsymbol{x}_j)} \alpha_j^* y_i y_j F_{ij}^* K_{ij} \\ &\leq \Big(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\Big)_i + B_2\kappa C + (B_2 - B_1)\kappa C + B_2 C\|\bar{\boldsymbol{K}}_i\|_1 \\ &\leq \Big(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\Big)_i + B_2 C\|\bar{\boldsymbol{K}}_i\|_1 + (B + B_2)\kappa C \\ &\leq \Big(\nabla_{\boldsymbol{\alpha}} \bar{H}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{F}})\Big)_i + (B + B_2)C(\|\boldsymbol{K}\|_1 + \kappa)\,,\end{aligned}$$

where $\boldsymbol{K}_i$ denotes the i-th column of the kernel matrix $\boldsymbol{K}$. We require $\big(\nabla_{\boldsymbol{\alpha}} H(\boldsymbol{\alpha}^*, \boldsymbol{F}^*)\big)_i \leq 0$ when $\bar{\boldsymbol{\alpha}}_i = 0$. To this end, we need the right-hand of the above inequality is smaller than zero. As a result, we can conclude that $\boldsymbol{\alpha}_i = 0$ from the optimality condition of problem (18). ■

References

Andreas Argyriou, Charles A Micchelli, and Massimiliano Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceedings of International Conference on Computational Learning Theory*, pages 338–352. Springer, 2005.

Francis Bach. Sharp analysis of low-rank kernel matrix approximations. In *Proceedings of Conference on Learning Theory*, pages 185–209, 2013.

Francis R Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Proceedings of Advances in Neural Information Processing Systems*, pages 105–112, 2008.

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the International Conference on Machine Learning*, pages 1–8, 2004.

Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79–94, 2006.

Peter L. Bartlett, Philip M. Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 2020.

Yoshua Bengio, Jean Francois Paiement, and Pascal Vincent. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Proceedings of Advances in Neural Information Processing Systems*, pages 177–184, 2004.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. pages 1–11, 2020.

Vladimir Cherkassky, Don Gehring, and Filip Mulier. Comparison of adaptive methods for function estimation from samples. *IEEE Transactions on Neural Networks*, 7(4):969–984, 1996.

Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pages 342–350, 2009.

Brendon K Colbert and Matthew M Peet. A convex parametrization of a new class of universal kernel functions. *Journal of Machine Learning Research*, 21(45):1–29, 2020.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(2):795–828, 2012.

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. On kernel-target alignment. In *Proceedings of Advances in Neural Information Processing Systems*, pages 367–373, 2002.

Tri Dao, Christopher M. De Sa, and Christopher Ré. Gaussian quadrature for kernel features. In *Proceedings of Advances in neural information processing systems*, pages 6107–6117, 2017.

Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li, and Fei Fei Li. Imagenet: A large-scale hierarchical image database. In *Proceedins of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Harris Drucker, Christopher J.C. Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik. Support vector regression machines. In *Proceedings of Advances in Neural Information Processing Systems*, pages 155–161, 1997.

Michaël Fanuel, Antoine Aspeel, Jean-Charles Delvennes, and Johan A.K. Suykens. Positive semi-definite embedding for dimensionality reduction and out-of-sample extensions. *arXiv preprint arXiv:1711.07271*, 2017.

Christophe Giraud and Nicolas Verzelen. Partial recovery bounds for clustering with the relaxed k-means. *Mathematical Statistics and Learning*, 1(3):317–374, 2019.

Mehmet Gonen. Bayesian efficient multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, pages 1–8, 2012.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

Alan J. Hoffman and Helmut W. Wielandt. The variation of the spectrum of a normal matrix. In *Selected Papers Of Alan J Hoffman: With Commentary*, pages 118–120. World Scientific, 2003.

Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of the International Conference on Machine Learning*, pages 361–368, 2007.

Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. A divide-and-conquer solver for kernel support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 566–574, 2014.

Zakria Hussain and John Shawe Taylor. Improved loss bounds for multiple kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 370–377, 2011.

Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, pages 448–456, 2015.

Lalit Jain, Blake Mason, and Robert Nowak. Learning low-dimensional metrics. In *Proceedins of Advances in Neural Information Processing Systems*, pages 4142–4150, 2017.

Prateek Jain, Brian Kulis, Jason V. Davis, and Inderjit S. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13(1):519–547, 2012.

Pratik Jawanpuria, Jagarlapudi Saketha Nath, and Ganesh Ramakrishnan. Generalized hierarchical kernel learning. *Journal of Machine Learning Research*, 16(1):617–652, 2015.

Neal Jean, Sang Michael Xie, and Stefano Ermon. Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5322–5333, 2018.

Charles R Johnson. *Matrix theory and applications*, volume 40. American Mathematical Soc., 1990.

S Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7(Jul):1493–1515, 2006.

Masoud Badiei Khuzani, Yinyu Ye, Sandy Napel, and Lei Xing. A mean-field theory for learning the schönberg measure of radial basis functions. *arXiv preprint arXiv:2006.13330*, 2020.

Matthäus Kleindessner and Ulrike von Luxburg. Kernel functions based on triplet comparisons. In *Proceedings of Advances in Neural Information Processing Systems*, pages 6810–6820, 2017.

Nils M. Kriege, Pierre Louis Giscard, and Richard C. Wilson. On valid optimal assignment kernels and applications to graph classification. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1623–1631, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

Brian Kulis. Metric learning: a survey. *Foundations and Trends in Machine Learning*, 5(4), 2013.

Brian Kulis, Mátyás A Sustik, and Inderjit S Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10(Feb):341–376, 2009.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5(1):27–72, 2004.

Chun-Liang Li, Wei-Cheng Chang, Youssef Mroueh, Yiming Yang, and Barnabas Poczos. Implicit kernel learning. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2007–2016, 2019.

Heng Lian and Zengyan Fan. Divide-and-conquer for debiased ℓ_1-norm support vector machine in ultra-high dimensions. *Journal of Machine Learning Research*, 18(1):6691–6716, 2017.

Fanghui Liu, Xiaolin Huang, Chen Gong, Jie Yang, and Li Li. Nonlinear pairwise layer and its training for kernel learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3659–3666, 2018.

Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan A.K. Suykens. Random features for kernel approximation: A survey in algorithms, theory, and beyond. *arXiv preprint arXiv:2004.11154*, 2020.

Xinwang Liu, Lei Wang, Xinzhong Zhu, Miaomiao Li, En Zhu, Tongliang Liu, Li Liu, Yong Dou, and Jianping Yin. Absent multiple kernel learning algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1303–1316, 2019.

Yong Liu and Shizhong Liao. Eigenvalues ratio for kernel selection of kernel methods. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 2814–2820, 2015.

Gaëlle Loosli, Stéphane Canu, and Soon Ong Cheng. Learning SVM in Kreĭn spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1204–1216, 2016.

Zhengdong Lu, Prateek Jain, and Inderjit S. Dhillon. Geometry-aware metric learning. In *Proceedings of the International Conference on Machine Learning*, pages 673–680, 2009.

Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

Julien Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Proceedings of Advances in neural information processing systems*, pages 1399–1407, 2016.

Jovana Mitrovic, Dino Sejdinovic, and Yee Whye Teh. Causal inference via kernel deviance measures. In *Proceedings of Advances in Neural Information Processing Systems*, pages 6986–6994, 2018.

Arkadi Nemirovski. Prox-method with rate of convergence $\mathcal{O}(1/t)$ for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1): 127–152, 2005.

Yung-Kyun Noh, Masashi Sugiyama, Kee-Eung Kim, Frank Park, and Daniel D Lee. Generative local metric learning for kernel regression. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2449–2459, 2017.

Cheng Soon Ong, Xavier Mary, and Alexander J. Smola. Learning with non-positive kernels. In *Proceedings of the International Conference on Machine Learning*, pages 81–89, 2004.

Binbin Pan, Wen Sheng Chen, Bo Chen, Chen Xu, and Jianhuang Lai. Out-of-sample extensions for non-parametric kernel methods. *IEEE Transactions on Neural Networks and Learning Systems*, 28(2):334–345, 2017.

Jean-Paul Penot. Differentiability properties of optimal value functions. *Canadian Journal of Mathematics*, 56(4):825–842, 2004.

John C. Platt. Sequential minimal optimization: a fast algorithm for training support vector machines. In *Advances in Kernel Methods-Support Vector Learning*, pages 212–223, 1998.

Danfeng Qin, Stephan Gammeter, Lukas Bossard, and Till Quack. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 777–784, 2011.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

Frank Michael Schleif and Peter Tino. Indefinite proximity learning: a review. *Neural Computation*, 27(10):2039–2096, 2015.

Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2003.

John Shawe-Taylor and Nello Cristianini. *Support vector machines*, volume 2. Cambridge University Press Cambridge, 2000.

Qinfeng Shi, Chunhua Shen, Rhys Hill, and Anton Van Den Hengel. Is margin preserved after random projection? In *Proceedings of the International Coference on International Conference on Machine Learning*, pages 643–650, 2012.

Si Si, Cho-Jui Hsieh, and Inderjit S Dhillon. Memory efficient kernel approximation. *Journal of Machine Learning Research*, 18:1–32, 2017.

Aman Sinha and John C. Duchi. Learning kernels with random features. In *Proceedins of Advances in Neural Information Processing Systems*, pages 1298–1306, 2016.

Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning*, pages 911–918, 2000.

Nathan Srebro and Shai Ben-David. Learning bounds for support vector machines with learned kernels. In *Proceedings of the International Conference on Computational Learning Theory*, pages 169–183. Springer, 2006.

Ingo Steinwart and Christmann Andreas. *Support Vector Machines*. Springer Science and Business Media, 2008.

George P. H. Styan. Hadamard products and multivariate statistical analysis. *Linear Algebra and Its Applications*, 6:217–240, 1973.

Johan A.K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

Philip HS Torr. Locally linear support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 1–8, 2011.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, pages 1065–1072, 2009.

John Von Neumann. On rings of operators. reduction theory. *Annals of Mathematics*, pages 401–485, 1949.

Shusen Wang, Luo Luo, and Zhihua Zhang. SPSD matrix approximation vis column selection: theories, algorithms, and extensions. *Journal of Machine Learning Research*, 17(1):1697–1745, 2016.

Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the International Conference on Machine Learning*, pages 1067–1075, 2013.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016.

Y. Xie, J Ho, and B Vemuri. On a nonlinear generalization of sparse coding and dictionary learning. In *Proceedings of the International Conference on Machine Learning*, pages 1480–1488, 2013.

Yiming Ying and Ding-Xuan Zhou. Learnability of gaussians with flexible variances. *Journal of Machine Learning Research*, 8(Feb):249–276, 2007.

Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression. In *Proceedings of Conference on Learning Theory*, pages 592–617, 2013.

Wei-Shi Zheng, Shaogang Gong, and Tao Xiang. Reidentification by relative distance comparison. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 35(3):653–668, 2012.

Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking person re-identification with k-reciprocal encoding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1318–1327, 2017.

Jinfeng Zhuang, Ivor W Tsang, and Steven C. H Hoi. A family of simple non-parametric kernel learning algorithms. *Journal of Machine Learning Research*, 12(2):1313–1347, 2011.