The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Nonlinear Pairwise Layer and Its Training for Kernel Learning

Fanghui Liu,[♭] Xiaolin Huang,[♭] Chen Gong,[♮] Jie Yang,[♭*] Li Li[♯]
[♭]Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University
[♮]School of Computer Science and Engineering, Nanjing University of Science and Technology
[♯]Department of Automation, Tsinghua University

Abstract

Kernel learning is a fundamental technique that has been intensively studied in the past decades. For the complicated practical tasks, the traditional "shallow" kernels (*e.g.*, Gaussian kernel and sigmoid kernel) are not flexible enough to produce satisfactory performance. To address this shortcoming, this paper introduces a nonlinear layer in kernel learning to enhance the model flexibility. This layer is pairwise, which fully considers the coupling information among examples. So our model contains a fixed single mapping layer (*i.e.* a Gaussian kernel) as well as a nonlinear pairwise layer, thereby achieving better flexibility than the existing kernel structures. Moreover, the proposed structure can be seamlessly embedded to Support Vector Machines (SVM), of which the training process can be formulated as a joint optimization problem including nonlinear function learning and standard SVM optimization. We theoretically prove that the objective function is gradient-Lipschitz continuous, which further guides us how to accelerate the optimization process in a deep kernel architecture. Experimentally, we find that the proposed structure outperforms other state-of-the-art kernel-based algorithms on various benchmark datasets, and thus the effectiveness of the incorporated pairwise layer with its training approach is demonstrated.

Introduction

Kernel learning (Schölkopf and Smola 2003) is one of the enduring topics in machine learning community. The family of kernel-based methods have been extensively studied over the past decade, such as support vector machines (SVM) (Vapnik 2000), kernel logistic regression (KLR) (Zhu and Hastie 2002), and kernel PCA (Zhang et al. 2016). These methods work by mapping the data from the original input space into a high-dimensional (possibly infinite-dimensional) feature space. The mapping is implicitly defined through a *kernel*, namely a function that defines an inner product between any two examples in the reproducing kernel Hilbert space (RKHS) (Evgeniou, Pontil, and Poggio 2000).

Although these kernel-based methods obtain good performance to some extent, many empirical studies (Zhuang, Tsang, and Hoi 2011; Huang et al. 2017a) suggest that they are not sufficiently flexible to accurately describe the data distribution due to their shallow architecture. Accordingly, some recent approaches (Lloyd et al. 2014; Zhong et al. 2014) target to develop more expressive kernels than the traditional kernels to discover rich structure embedded in the data. Existing advanced kernel-based algorithms can be roughly grouped into two categories: multiple kernel learning (MKL) and deep kernel learning. MKL (Nen, Alpayd, and Ethem 2011; Varma and Babu 2009) aims at learning a combination of a set of predefined kernels to obtain a good integrated kernel, which would yield a "broader" kernel. The heuristic approaches, Bayesian approaches, and boosting approaches are usually adopted by the representative models in implement MKL (see a survey in (Nen, Alpayd, and Ethem 2011)). Recently, motivated by the successes of deep learning framework, some emerging studies (Cho 2012; Zhuang, Tsang, and Hoi 2011; Wilson et al. 2016) have attempted to incorporate deep architecture into kernel learning to further enhance the flexibility of regular MKL and other traditional kernel-based methods. As a result, deep kernel architecture substantially increases the "richness" of representations when compared to a shallow architecture.

However, there are two main drawbacks of these deep kernel based methods. First, there has so far been no satisfied or proper optimization algorithms for these nested kernels in deep architecture during the training process. Second, such deep kernel based methods often use a fixed or regular scheme to generate a nested kernel. The model flexibility is generally limited to such insufficient scheme, and thus these methods can hardly yield a promising performance as we expected. To address such limitations, we introduce a nonlinear pairwise layer in kernel learning as shown in Fig. 1. One can see that the kernel learning framework in SVM can be analogous to the structure of a neural network. In our model, we use a fixed single mapping layer but add a nonlinear pairwise layer which is highly coupled with training examples to capture the different local statistics of the input data. Such pairwise layer can be interpreted as a nonlinear function $g(\mathbf{x}_i, \mathbf{x}_j)$ that we need to learn, where $\mathbf{x}_i$ is the i^{th} example of the input $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$. By doing so, one hand, we do not limit the specific formulation for the function $g(\cdot)$, which provides more flexibility when compared with other deep nested kernels. On the other hand, we only need to learn the function $g(\cdot)$ instead of learning a multi-layer kernel in deep architecture to simplify the training process. As a result,

*Corresponding Author.

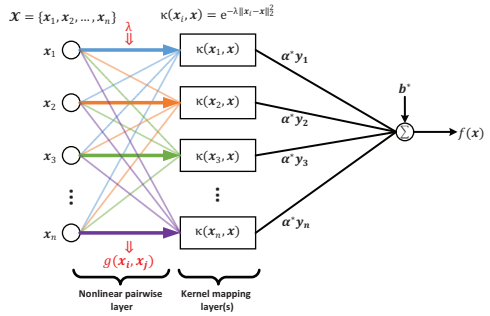


Figure 1: The structure of our KNPL model in SVM.

our model not only can capture the different local statistics of the input data, but also is much easier to learn a nested kernel in the network without too many hyper-parameters.

Formally, in this paper, we propose a flexible **k**ernel learning framework with a **n**onlinear **p**airwise **l**ayer in SVM, termed as KNPL, to enhance the model flexibility. We formulate the learning of such nonlinear function $g(\cdot)$ and the induced classification model as a joint optimization problem using an efficient iterative algorithm. Specifically, training such nonlinear function is conducted by least squares semi-definite programming via alternating direction methods. Moreover, the objective function is theoretically proved to be gradient-Lipschitz continuous, which makes it possible to accelerate the optimization process in a deep kernel architecture. Numerous experiments on various data sets demonstrate that the proposed KNPL model with the nonlinear pairwise layer can effectively enhance the model flexibility, and thus achieving superior classification performance to other state-of-the-art kernel learning based algorithms.

KNPL Model

In this section, we firstly investigate the relationship between a deep kernel and our KNPL model, and then introduce the KNPL model in SVM.

In deep kernel architecture, the kernel with l layers (Cho 2012) is defined by:

$$\mathcal{K}^{(l)}(\mathbf{x}_i, \mathbf{x}_j) = \phi^{(l)}\big(\cdots\phi^{(1)}(\mathbf{x}_i)\big) \cdot \phi^{(l)}\big(\cdots\phi^{(1)}(\mathbf{x}_j)\big)\,,$$

which computes the inner product between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ after l successive applications of the nonlinear mapping $\phi(\cdot)$. For example, the two layer composition of Gaussian kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\lambda\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ with the kernel width λ is formulated as:

$$\begin{aligned}\mathcal{K}^{(2)}(\mathbf{x}_i, \mathbf{x}_j) &= \phi^{(2)}(\phi^{(1)}(\mathbf{x}_i)) \cdot \phi^{(2)}(\phi^{(1)}(\mathbf{x}_j)) \\ &= e^{-2\lambda}\exp(-2\lambda\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j))\end{aligned}\,. \tag{1}$$

This formulation demonstrates that an iterated mapping $\phi(\phi(\mathbf{x}))$ essentially changes the input, and would generate a more comprehensive or complex representation than the single mapping $\phi(\mathbf{x})$. Here we observe that the nested kernel in Eq. (1) can be decomposed into a fixed Gaussian kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ and a nonlinear pairwise function $g(\mathbf{x}_i, \mathbf{x}_j)$, namely:

$$\mathcal{K}^{(2)}(\mathbf{x}_i, \mathbf{x}_j) \triangleq g(\mathbf{x}_i, \mathbf{x}_j) \cdot \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\,. \tag{2}$$

That is to say, using a single kernel as well as a nonlinear pairwise layer is able to achieve a comparable and even better model flexibility when compared to the deep kernel architecture as demonstrated by our KNPL model. Note that $g(\cdot)$ can be in an arbitrary type, and thus one can always find a proper function to satisfy such decomposition in Eq. (2) when any two examples $(\mathbf{x}_i, \mathbf{x}_j)$ are given.

To intuitively illustrate the superiority of the introduced nonlinear layer $g(\cdot)$, here we give a two-dimensional example in the KNPL model. In Fig. 2, data of class $+1$ (marked by green stars) are a mixture of Gaussian which takes in sandwiches of the class -1 (marked by red crosses). We compare the proposed KNPL model with a baseline method, *i.e.*, SVM with Gaussian kernel, in which the kernel width λ and the trade-off parameter C are respectively tuned via a five-fold cross validation (termed as "SVM-CV"). The classification accuracy on the training data ranges from 86.87% by "SVM-CV" to 94.93% yielded by the learned KNPL model. One can see that KNPL provides a more complex and accurate boundary, and thus is more flexible to capture the different local statistics of the training data. Specifically, the induced boundary of KNPL on some outliers is isolated, which means that it just works on a neighborhood of such outliers and has little unfavorable effect on other data. Further, the right of Fig. 2 indicates that the nonlinear function values ranges from 0.95 to 1.05. Such small and steadily fluctuation indicates that such sophisticated classification boundary is not generated by a too large kernel width λ, which makes the learned model not easy to be over-fitting.

In the next, we introduce the formulation of KNPL model. Let $\mathcal{S}^n$ be the space of $n \times n$ symmetric matrices, and $\mathcal{S}^n_+$ be the cone of positive semi-definite matrices in $\mathcal{S}^n$. A set of training examples are given by the input $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ with its label $y_i \in \{\pm 1\}$, and the output $\mathbf{y} = \{y_i\}_{i=1}^n$ forms the label matrix $\mathbf{Y} = \mathrm{diag}(\mathbf{y})$. Let $\mathbf{1}$ be a n-dimensional vector of all ones, and C is a positive trade-off parameter. Accordingly, the dual formulation of soft margin SVM is given by:

$$\begin{aligned}&\max_{\boldsymbol{\alpha}}\ \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{Y}\mathbf{K}\mathbf{Y}\boldsymbol{\alpha} \\ &s.t.\ \ \mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{1},\ \boldsymbol{\alpha}^\top\mathbf{y} = 0\end{aligned}\,, \tag{3}$$

where the dual variable is $\boldsymbol{\alpha} \in \mathbb{R}^n$ and the kernel matrix $\mathbf{K} \in \mathcal{S}^n_+$ is derived by a kernel $\mathcal{K}$, namely $\mathbf{K}_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$. When $\mathbf{K}$ is positive semi-definite, this problem is a convex quadratic programming.

As presented in Introduction, we design a nonlinear pairwise layer and need to learn the nonlinear function $g(\mathbf{x}_i, \mathbf{x}_j)$. Here, we seek to estimate its function value instead of directly learning the function $g(\cdot)$ for simplicity. They are totally equivalent when the training data are given. Formally, we introduce a pairwise matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ where $\mathbf{G}_{ij} = g(\mathbf{x}_i, \mathbf{x}_j)$ into problem (3). By Slater's condition (Boyd and Vandenberghe 2004), strong duality holds, so the optimal values of the primal and dual soft-margin SVM problems will be equal.

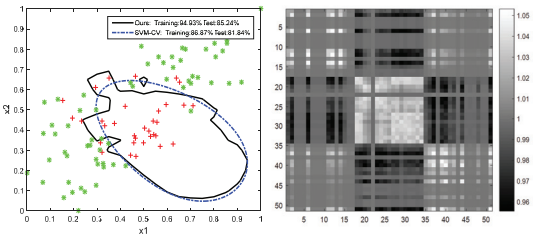


Figure 2: Left figure: Data of two classes are marked by green stars and red crosses. The solid and dashed lines illustrate the boundary of our method and SVM-CV, respectively. Right figure: The values of the pairwise matrix $\mathbf{G}$.

Hence, as we expect, the loss of the primal problem would decrease by optimizing $\mathbf{G}$. And then, following max-min approach (see details in (Boyd and Vandenberghe 2004)), we have:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \min_{\mathbf{G}\in\mathcal{S}_+^n} \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{Y}\big(\mathbf{G}\odot\mathbf{K}\big)\mathbf{Y}\boldsymbol{\alpha} + \gamma\|\mathbf{G}\|_F^2 \\ &s.t.\ \ \boldsymbol{\alpha}^\top\mathbf{y} = 0,\ \mathbf{0}\le\boldsymbol{\alpha}\le C\mathbf{1}, \mathbf{G}\mathbf{1} = n\mathbf{1} \end{aligned}, \tag{4}$$

where $\gamma > 0$ is a regularization parameter and the operator $\odot$ denotes Hadamard product. The inner minimization problem in Eq. (4) is a convex conic program on $\mathbf{G}$. The outer maximization problem is a point-wise minimum of a family of concave quadratic functions of $\boldsymbol{\alpha}$, and hence the outer optimization problem is also convex. For the pairwise matrix $\mathbf{G}$, we restrict it to be a positive semi-definite matrix, so the learned kernel matrix $\tilde{\mathbf{K}} = \mathbf{G}\odot\mathbf{K}$ is guaranteed to be a positive semi-definite one[1]. Besides, the constraint $\mathbf{G}\mathbf{1} = n\mathbf{1}$ is incorporated into Eq. (4) for scaling invariant. And also, such shift-invariant constraint is able to avoid a trivial solution $\mathbf{G} = \mathbf{0}_{n\times n}$.

Accordingly, the proposed nonlinear layer is embedded to SVM, of which the training process is formulated as a joint optimization problem including nonlinear function learning and standard SVM optimization as demonstrated by Eq. (4). Before we introduce an optimization algorithm to solve the KNPL model, we need to investigate the differentiable property of Eq. (4), which would help us to optimize this problem.

Differentiability of Problem (4)

For notational simplicity, let $\mathcal{A} = \{\boldsymbol{\alpha}\in\mathbb{R}^n : \boldsymbol{\alpha}^\top\mathbf{y} = 0,\ 0\le\boldsymbol{\alpha}\le C\mathbf{1}\}$ be the constraint for a standard SVM, $\mathcal{B}$ be a convex polyhedron defined by the affine equality $\mathbf{G}\mathbf{1} = n\mathbf{1}$ and then we introduce a function:

$$H(\boldsymbol{\alpha},\mathbf{G}) = \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{Y}\big(\mathbf{G}\odot\mathbf{K}\big)\mathbf{Y}\boldsymbol{\alpha} + \gamma\|\mathbf{G}\|_F^2\,.$$

Therefore, problem (4) is equivalent to:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \min_{\mathbf{G}\in\mathcal{S}_+^n\bigcap\mathcal{B}} H(\boldsymbol{\alpha},\mathbf{G})\,. \tag{5}$$

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

For simplicity, we investigate the following function:

$$h(\boldsymbol{\alpha}) = \min_{\mathbf{G}\in\mathcal{S}_+^n\bigcap\mathcal{B}} H(\boldsymbol{\alpha},\mathbf{G})\,. \tag{6}$$

It is obviously concave since h is the minimum of a sequence of concave functions. We term the associated function $H(\boldsymbol{\alpha},\mathbf{G})$ as the saddle representation of the objective function $h(\boldsymbol{\alpha})$.

In (Bonnans and Shapiro 1998), the authors outline a useful characterization of differentiable properties of the optimal value function, and thus we have:

Proposition 1. *The objective function $h(\boldsymbol{\alpha})$ defined by Eq. (6) is differentiable and its gradient is given by:*

$$\nabla h(\boldsymbol{\alpha}) = \mathbf{1} - \mathbf{Y}\big(\mathbf{G}^*\odot\mathbf{K}\big)\mathbf{Y}\boldsymbol{\alpha}\,, \tag{7}$$

where $\mathbf{G}^* = \operatorname{argmin}_{\mathbf{G}} H(\boldsymbol{\alpha},\mathbf{G})$.

Based on the strongly convex of $H(\boldsymbol{\alpha},\cdot)$, the uniqueness of $\mathbf{G}^*$ holds and thus this proposition is easily obtained.

In the next, we analyze the differentiable property of $\nabla h(\boldsymbol{\alpha})$. To this end, we first establish two useful lemmas.

Lemma 1. *For any $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2\in\mathcal{A}$, we have:*

$$\|\Gamma(\boldsymbol{\alpha}_1) - \Gamma(\boldsymbol{\alpha}_2)\| \le \frac{\|\mathbf{K}\|\big(\|\boldsymbol{\alpha}_1\| + \|\boldsymbol{\alpha}_2\|\big)}{4\gamma}\big\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\big\|\,,$$

where $\Gamma = \frac{1}{4\gamma}\operatorname{diag}(\boldsymbol{\alpha}^\top\mathbf{Y})\mathbf{K}\operatorname{diag}(\boldsymbol{\alpha}^\top\mathbf{Y})$.

Proof. The proof is presented in Eq. (8) (see in the next page). □

Lemma 2. *For any $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2\in\mathcal{A}$, suppose that $\mathbf{G}_1^* = \operatorname{argmin}_{\mathbf{G}} H(\boldsymbol{\alpha}_1,\mathbf{G})$ and $\mathbf{G}_2^* = \operatorname{argmin}_{\mathbf{G}} H(\boldsymbol{\alpha}_2,\mathbf{G})$, there holds:*

$$\|\mathbf{G}_1^* - \mathbf{G}_2^*\| \le \frac{\|\mathbf{K}\|\big(\|\boldsymbol{\alpha}_1\| + \|\boldsymbol{\alpha}_2\|\big)}{4\gamma}\big\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\big\|\,.$$

Proof. Let $\partial_{\mathbf{G}}H(\boldsymbol{\alpha},\cdot)$ denote the gradient w.r.t. $\mathbf{G}$. Now consider the minimization problem $\operatorname{argmin}_{\mathbf{G}} H(\boldsymbol{\alpha},\mathbf{G})$, by the first order optimality conditions, we have:

$$\begin{cases} \langle\partial_{\mathbf{G}}H(\boldsymbol{\alpha}_1,\mathbf{G}_1^*), \mathbf{G}_2^* - \mathbf{G}_1^*\rangle \ge 0 \\ \langle\partial_{\mathbf{G}}H(\boldsymbol{\alpha}_2,\mathbf{G}_2^*), \mathbf{G}_1^* - \mathbf{G}_2^*\rangle \ge 0 \end{cases}. \tag{10}$$

Consequently, we have:

$$\langle\partial_{\mathbf{G}}H(\boldsymbol{\alpha}_1,\mathbf{G}_1^*) - \partial_{\mathbf{G}}H(\boldsymbol{\alpha}_2,\mathbf{G}_2^*), \mathbf{G}_2^* - \mathbf{G}_1^*\rangle \ge 0\,.$$

Substituting the fact that:

$$\partial_{\mathbf{G}}H(\boldsymbol{\alpha},\mathbf{G}) = 2\gamma\mathbf{G} - 2\gamma\Gamma(\boldsymbol{\alpha})\,. \tag{11}$$

And then:

$$\big\langle 2\gamma\mathbf{G}_1^* - 2\gamma\Gamma(\boldsymbol{\alpha}_1) - 2\gamma\mathbf{G}_2^* + 2\gamma\Gamma(\boldsymbol{\alpha}_2), \mathbf{G}_2^* - \mathbf{G}_1^*\big\rangle \ge 0\,.$$

Accordingly, we have:

$$\begin{aligned} \big\|\mathbf{G}_1^* - \mathbf{G}_2^*\big\|^2 &\le \big\langle\Gamma(\boldsymbol{\alpha}_2) - \Gamma(\boldsymbol{\alpha}_1), \mathbf{G}_2^* - \mathbf{G}_1^*\big\rangle \\ &\le \big\|\Gamma(\boldsymbol{\alpha}_2) - \Gamma(\boldsymbol{\alpha}_1)\big\|\big\|\mathbf{G}_1^* - \mathbf{G}_2^*\big\|\,. \end{aligned}$$

By Lemma 1, we conclude the proof. □

$$
\begin{aligned}
\|\Gamma(\boldsymbol{\alpha}_1) - \Gamma(\boldsymbol{\alpha}_2)\| &= \frac{1}{4\gamma}\Big\| \operatorname{diag}\big(\boldsymbol{\alpha}_1{}^\top \mathbf{Y}\big)\mathbf{K} \operatorname{diag}\big(\boldsymbol{\alpha}_1{}^\top \mathbf{Y}\big) - \operatorname{diag}\big(\boldsymbol{\alpha}_2{}^\top \mathbf{Y}\big)\mathbf{K} \operatorname{diag}\big(\boldsymbol{\alpha}_2{}^\top \mathbf{Y}\big)\Big\| \\
&= \frac{1}{4\gamma}\Big\| \operatorname{diag}\big(\boldsymbol{\alpha}_1{}^\top \mathbf{Y} + \boldsymbol{\alpha}_2{}^\top \mathbf{Y}\big)\mathbf{K} \operatorname{diag}\big(\boldsymbol{\alpha}_1{}^\top \mathbf{Y} - \boldsymbol{\alpha}_2{}^\top \mathbf{Y}\big)\Big\| \\
&\leq \frac{1}{4\gamma}\|\mathbf{K}\|\Big\| \operatorname{diag}\big(\boldsymbol{\alpha}_1{}^\top \mathbf{Y} + \boldsymbol{\alpha}_2{}^\top \mathbf{Y}\big)\Big\|\Big\| \operatorname{diag}\big(\boldsymbol{\alpha}_1{}^\top \mathbf{Y} - \boldsymbol{\alpha}_2{}^\top \mathbf{Y}\big)\Big\| \\
&\leq \frac{\|\mathbf{K}\|}{4\gamma}\|\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2\|\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \quad (\mathbf{Y} \text{ is an orthogonal matrix})
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
\|\nabla h(\boldsymbol{\alpha}_1) - \nabla h(\boldsymbol{\alpha}_2)\| &= \big\|\mathbf{Y}\big(\mathbf{G}_1^* \odot \mathbf{K}\big)\mathbf{Y}\boldsymbol{\alpha}_1 - \mathbf{Y}\big(\mathbf{G}_2^* \odot \mathbf{K}\big)\mathbf{Y}\boldsymbol{\alpha}_2\big\| = \big\|\mathbf{Y}\big(\mathbf{G}_1^* - \mathbf{G}_2^*\big) \odot \mathbf{K}\mathbf{Y}\boldsymbol{\alpha}_1 - \mathbf{Y}\big(\mathbf{G}_2^* \odot \mathbf{K}\big)\mathbf{Y}\big(\boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_1\big)\big\| \\
&\leq \big\|\mathbf{Y}\big(\mathbf{G}_1^* - \mathbf{G}_2^*\big) \odot \mathbf{K}\mathbf{Y}\boldsymbol{\alpha}_1\big\| + \big\|\mathbf{Y}\big(\mathbf{G}_2^* \odot \mathbf{K}\big)\mathbf{Y}\big(\boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_1\big)\big\| \\
&\leq \big\|\big(\mathbf{G}_1^* - \mathbf{G}_2^*\big) \odot \mathbf{K}\big\|\|\boldsymbol{\alpha}_1\| + \big\|\mathbf{G}_2^* \odot \mathbf{K}\big\|\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \quad (\mathbf{Y} \text{ is an orthogonal matrix}) \\
&\leq \big\|\mathbf{G}_1^* - \mathbf{G}_2^*\big\|\|\mathbf{K}\|\|\boldsymbol{\alpha}_1\| + \|\mathbf{G}_2^*\|\|\mathbf{K}\|\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \quad (\text{Using } \|\mathbf{A} \odot \mathbf{B}\|_F \leq \operatorname{Tr}(\mathbf{A}\mathbf{B}^\top) \leq \|\mathbf{A}\|_F\|\mathbf{B}\|_F) \\
&\leq \frac{\|\mathbf{K}\|^2}{4\gamma}\|\boldsymbol{\alpha}_1\|\big(\|\boldsymbol{\alpha}_1\| + \|\boldsymbol{\alpha}_2\|\big)\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| + \|\mathbf{G}_2^*\|\|\mathbf{K}\|\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \quad (\text{Using Lemma 2}) \\
&\leq \frac{\|\mathbf{K}\|^2}{4\gamma}\Big(\|\boldsymbol{\alpha}_1\|\big(\|\boldsymbol{\alpha}_1\| + \|\boldsymbol{\alpha}_2\|\big) + \|\boldsymbol{\alpha}_2\|^2\Big)\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \quad \big(\text{Using } \mathbf{G}_2^* = \Gamma(\boldsymbol{\alpha}_2) \text{ by Eq. (11)}\big) \\
&\leq \frac{3nC^2\|\mathbf{K}\|^2}{4\gamma}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \quad (\text{Using } 0 \leq \boldsymbol{\alpha} \leq C,\ \|\boldsymbol{\alpha}\|^2 \leq nC^2)
\end{aligned} \tag{9}
$$

Based on the above lemmas, we can establish the gradient-Lipschitz continuity of the objective function $h(\boldsymbol{\alpha})$. Formally, we present the following theorem.

Theorem 1. *The gradient of the objective function given by Eq.* (7) *is Lipschitz continuous with Lipschitz constant* $L = \frac{3nC^2\|\mathbf{K}\|^2}{4\gamma}$ i.e. *for any* $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2 \in \mathcal{A}$, *the following inequality holds* $\|\nabla h(\boldsymbol{\alpha}_1) - \nabla h(\boldsymbol{\alpha}_2)\| \leq L\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|$.

Proof. For any $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{A}$, from representation of $\nabla h(\boldsymbol{\alpha})$ in Proposition 1, the term $\|\nabla h(\boldsymbol{\alpha}_1) - \nabla h(\boldsymbol{\alpha}_2)\|$ can be bounded by Eq. (9), which concludes the proof. □

The theoretical analysis above, mainly Theorem 1, provides a justification for utilizing the backward propagation algorithm with Nesterov's acceleration method (Flammarion and Bach 2015) to train a deep kernel network. While in our model, we just use a single layer kernel to emphasize the effectiveness of the introduced nonlinear pairwise layer. Such problem is well formulated as a joint optimization model, and thus we develop an iterative algorithm to solve the optimization problem (4).

Optimization for Problem (4)

In this section, we present an alternate iterative algorithm to solve the optimization problem (4). In each iteration of the algorithm, $\boldsymbol{\alpha}$ and $\mathbf{G}$ are alternatively optimized. When $\mathbf{G}$ is fixed, the standard SVM can be solved by the SMO algorithm (Platt 1998); When $\boldsymbol{\alpha}$ is fixed, the inner optimization problem for $\mathbf{G}$ is semi-definite least squares, which can be solved by the fast Alternating Direction Method of Multipliers (ADMM) approach (Boyd et al. 2011).

Optimization for $\mathbf{G}$ via ADMM

Just consider the variable $\mathbf{G}$ in Eq. (4) with some algebraic manipulations[2], we have:

$$
\min_{\mathbf{G} \in \mathcal{S}_+^n \bigcap \mathcal{B}} -2\operatorname{Tr}\big(\gamma\Gamma\mathbf{G}\big) + \gamma\|\mathbf{G}\|_F^2 . \tag{12}
$$

where $\Gamma = \frac{1}{4\gamma}\operatorname{diag}(\boldsymbol{\alpha}^\top\mathbf{Y})\mathbf{K}\operatorname{diag}(\boldsymbol{\alpha}^\top\mathbf{Y})$. Further, Eq. (12) is equivalent to solve the following problem:

$$
\min_{\mathbf{G} \in \mathcal{S}_+^n \bigcap \mathcal{B}} \|\mathbf{G} - \Gamma\|_F^2 . \tag{13}
$$

Such problem aims to seek the projection of Γ onto the intersection of two spaces $\mathcal{S}_+^n$ and $\mathcal{B}$ (the projection is well defined since both $\mathcal{S}_+^n$ and $\mathcal{B}$ are convex). Throughout, we assume that the solution set of Eq. (13) is not empty. Hence this issue is a typical semi-definite least-squares problem with several dedicated methods (see a survey in (Henrion and Malick 2012)). Here we choose the alternating directions based algorithm to solve this problem.

Following (He, Xu, and Yuan 2011), we reformulate Eq. (13) as:

$$
\begin{aligned}
&\min_{\mathbf{G} \in \mathcal{S}_+^n} \frac{1}{2}\|\mathbf{G} - \Gamma\|_F^2 + \frac{1}{2}\|\mathbf{Q} - \Gamma\|_F^2 \\
&s.t.\ \ \mathbf{G} - \mathbf{Q} = \mathbf{0}_{n\times n}, \mathbf{Q}\mathbf{1} = n\mathbf{1}
\end{aligned} \tag{14}
$$

By introducing augmented Lagrange multipliers to incorporate the equality constraints into the objective function Eq. (14), we obtain the augmented Lagrangian function as

[2] $\mathbf{x}^\top\mathbf{A} \odot \mathbf{B}\mathbf{y} = \operatorname{Tr}(\mathbf{D}_x\mathbf{A}\mathbf{D}_y\mathbf{B}^\top)$, where $\mathbf{D}_x = \operatorname{diag}(\mathbf{x})$ and $\mathbf{D}_y = \operatorname{diag}(\mathbf{y})$.

follows:

$$\mathcal{L}(\mathbf{G},\mathbf{Q},\mathbf{Z},\beta) = \frac{1}{2}\|\mathbf{G}-\Gamma\|_F^2+\frac{1}{2}\|\mathbf{Q}-\Gamma\|_F^2-\mathbf{Z}^\top(\mathbf{G}-\mathbf{Q})+\frac{\beta}{2}\|\mathbf{G}-\mathbf{Q}\|_F^2, \tag{15}$$

where $\mathbf{Z} \in \mathcal{S}^n$ is the Lagrangian multiplier and $\beta \geq 0$ is the penalty parameter. The ADMM algorithm is to update the variables $\mathbf{G}$, $\mathbf{Q}$, and $\mathbf{Z}$ alternately, by minimizing $\mathcal{L}$ with other variables fixed. Consequently, we have three update steps corresponding to all the variables as follows.

Step 1: Update $\mathbf{G}$: The pairwise matrix $\mathbf{G}$ is updated by solving the following optimization problem:

$$\mathbf{G}_{k+1} = \operatorname*{argmin}_{\mathbf{G}} \mathcal{P}_{\mathcal{S}_+^n}\Big(\frac{1}{1+\beta_k}\big(\beta_k\mathbf{Q}_k+\mathbf{Z}_k+\Gamma\big)\Big),$$

where $\mathcal{P}_{\mathcal{S}_+^n}$ denotes the projection onto $\mathcal{S}_+^n$. The optimal solution to this problem is then given by:

$$\mathbf{G}_{k+1} = \Big(\frac{1}{1+\beta_k}\big(\beta_k\mathbf{Q}_k+\mathbf{Z}_k+\Gamma\big)\Big)_+, \tag{16}$$

where the notation $\mathbf{V}_+$ denotes the positive part of the matrix $\mathbf{V}$, *i.e.*, $\mathbf{V}_+ = \sum_i \max(0,\lambda_i)\mathbf{v}_i\mathbf{v}_i^\top$ where λ_i and $\mathbf{v}_i$ are the i^{th} eigenvalue and eigenvector of $\mathbf{V}$, respectively.

Step 2: Update $\mathbf{Q}$: The minimization problem (15) with respect to $\mathbf{Q}$ is:

$$\begin{aligned}\mathbf{Q}_{k+1} &= \operatorname*{argmin}_{\mathbf{Q}\in\mathcal{B}} \frac{1}{2}\|\mathbf{Q}-\Gamma\|_F^2-\mathbf{Z}^\top(\mathbf{G}-\mathbf{Q})+\frac{\beta}{2}\|\mathbf{G}-\mathbf{Q}\|_F^2 \\ &= \mathcal{P}_{\mathcal{B}}\Big(\underbrace{\frac{1}{1+\beta_k}\big(\beta_k\mathbf{G}_{k+1}-\mathbf{Z}_k+\Gamma\big)}_{\triangleq\Xi}\Big).\end{aligned} \tag{17}$$

Here computing the projection $\mathcal{P}_{\mathcal{B}}(\Xi)$ requires to solve a standard quadratic programming, which arrives at:

$$\min_{\mathbf{Q}} \|\mathbf{Q}-\Xi\|_F^2, \;\; s.t. \;\; \mathbf{Q}\mathbf{1} = n\mathbf{1}.$$

Note that the above problem with the shift-invariant constraint is separable, and thus we can obtain its closed form solution. For example, we take an arbitrary column of $\mathbf{Q}$, termed as $\mathbf{q}$, and the corresponding column vector $\boldsymbol{\xi}$ in Ξ, namely:

$$\min_{\mathbf{q}} \|\mathbf{q}-\boldsymbol{\xi}\|_2^2, \;\; s.t. \;\; \mathbf{1}^\top\mathbf{q} = n. \tag{18}$$

Let τ be the Lagrange multiplier for the shift-variant constraint $\mathbf{1}^\top\mathbf{q} = n$, then the related Lagrangian function is $\mathcal{L}(\mathbf{q},\tau) = \|\mathbf{q}-\boldsymbol{\xi}\|_2^2 + \tau(\mathbf{1}^\top\mathbf{q}-n)$. The partial derivative of $\mathcal{L}(\mathbf{q},\tau)$ with respect to $\mathbf{q}$ is:

$$\frac{\partial\mathcal{L}(\mathbf{q},\tau)}{\partial\mathbf{q}} = 2\mathbf{q}-2\boldsymbol{\xi}+\tau\mathbf{1}.$$

By the shift-variant constraint and the Karush-Kuhn-Tucker condition, we have:

$$2n\mathbf{q}-2\boldsymbol{\xi}\mathbf{1}^\top\mathbf{q}+\tau\mathbf{1}n = 0.$$

Accordingly, the analytic solution is:

$$\mathbf{q}^* = \tau(2\boldsymbol{\xi}\mathbf{1}^\top-2n\mathbf{I}_n)^{-1}\mathbf{1}n, \tag{19}$$

where $\mathbf{I}_n$ is an identity matrix, and τ can be further obtained by the shift-invariant constraint. Specifically, the matrix inversion in Eq. (19) can be avoided by the Sherman-Morrison formula, and hence the solution $\mathbf{q}^*$ can be further simplified to:

$$\mathbf{q}^* = -2\tau\big(\mathbf{I}_n+\frac{\boldsymbol{\xi}\mathbf{1}^\top}{n-\mathbf{1}^\top\boldsymbol{\xi}}\big)\mathbf{1}. \tag{20}$$

Accordingly, we directly obtain the closed-form solution $\mathbf{Q} = [\mathbf{q}_1^*, \mathbf{q}_2^*, \cdots, \mathbf{q}_n^*]$ without employing or designing any inefficient iterative method for this particular purpose.

Step 3: Update Multipliers $\mathbf{Z}$ and β: The Lagrange multiplier $\mathbf{Z}$ and the penalty parameter β are updated as follows:

$$\mathbf{Z}_{k+1} = \mathbf{Z}_k-\beta_k\big(\mathbf{G}_{k+1}-\mathbf{Q}_{k+1}\big), \tag{21}$$

$$\beta_{k+1} = \min\big\{\rho\beta_k, 10^8\big\}, \tag{22}$$

where $\rho = 1.1$ is the parameter that makes β gradually increase in each loop so that the normalization constraint can be finally satisfied.

The entire iterative process for solving Eq. (15) is summarized in Algorithm 1. Its convergence has been theoretically proved in (He, Xu, and Yuan 2011) and will be empirically illustrated by the experiments.

Algorithm 1: Optimization for (15) via ADMM

Input: A given $\boldsymbol{\alpha}$, the kernel matrix $\mathbf{K}$, the label matrix $\mathbf{Y}$; Initialization of $\mathbf{G}$, $\mathbf{Q}$, $\mathbf{Z}$ by all-one matrices, and β; Stopping criteria $k_{\max} = 15$ and $\epsilon = 10^{-4}$.

Output: The optimal $\mathbf{G}^*$ that minimizes Eq. (15).

Set $k = 0$.
Repeat
  Update $\mathbf{G}$ via Eq. (16);
  Update $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n]$ via Eq. (20);
  Update Lagrange multipliers $\mathbf{Z}$ and β as in Eq. (21) and Eq. (22), respectively;
Until $k = k_{\max}$ *or*
Diff $= \max\big\{\|\mathbf{Q}_{k+1}-\mathbf{Q}_k\|_F, \|\mathbf{Z}_{k+1}-\mathbf{Z}_k\|_F\big\} \leq \epsilon$;

Optimization for α in Problem (4)

When obtaining the optimal $\mathbf{G}^*$, the learned flexible kernel $\tilde{\mathbf{K}}^* = \mathbf{G}^* \odot \mathbf{K}$ can be used for solving the dual variable $\boldsymbol{\alpha}$ in SVM iteratively. Therefore, the algorithm for solving the KNPL model is summarized in Algorithm 2. Specifically, such iteration algorithm converges very fast, usually within 10 iterations.

Predict for the Test Data

By Algorithm 1, we learn the pairwise matrix $\mathbf{G}^*$ and the induced flexible kernel matrix $\tilde{\mathbf{K}}^* = \mathbf{G}^* \odot \mathbf{K}$. Thereby, the nonlinear pairwise function $g(\mathbf{x}_i, \mathbf{x}_j)$ can be faultlessly approximated by $\mathbf{G}^*$ when the training data are given. However, such function $g(\cdot)$ is not unknown on the test data and

Algorithm 2: Algorithm for the KNPL model.

Input: The training set label $\mathbf{Y}$, the kernel matrix $\mathbf{K}$, and the pairwise matrix $\mathbf{G}$
Output: The optimal $\boldsymbol{\alpha}^*$

Set the maximum iteration number $T = 10$.
Initialize $i = 0$ and $\boldsymbol{\alpha}$.
Repeat
  Obtain $\mathbf{G}^{(i+1)}$ by Algorithm 1;
  Solve $\boldsymbol{\alpha}$ with the learned kernel $\mathbf{G}^{(i+1)} \odot \mathbf{K}$ by SMO algorithm (Platt 1998);
  $i := i + 1$;
Until $i \geq T$;

must be approximated by the learned $\mathbf{G}^*$. Suppose that the pairwise matrix $\mathbf{G}' \in \mathbb{R}^{n \times m}$ is conducted on the test data $\mathcal{X}' = \{\mathbf{x}'_i\}_{i=1}^m$. The corresponding flexible kernel matrix on the test data is $\tilde{\mathbf{K}}' = \mathbf{G}' \odot \mathbf{K}'$, where $\mathbf{K}'_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}'_j)$. Here, we build a mapping between $\mathbf{G}^*$ and $\mathbf{G}'$ by exploiting the nearest relationship between the training and test data, that is:

$$\mathbf{G}'_i \leftarrow \mathbf{G}^*_j, \text{ if } \mathbf{x}_j = \mathcal{N}(\mathbf{x}'_i, \mathcal{X}) \,. \tag{23}$$

This formulation suggests that if $\mathbf{x}_j$ is the nearest neighbor of $\mathbf{x}'_i$ among the training data $\mathcal{X}$, the j^{th} column of $\mathbf{G}^*$ is assigned to the i^{th} column of $\mathbf{G}'$. By doing so, the corresponding flexible kernel matrix $\tilde{\mathbf{K}}'$ can be obtained. Finally, the labels of test data are predicted by $\tilde{\mathbf{K}}'$ and the optimal $\boldsymbol{\alpha}^*$. Actually, there is a gap between the learned $\tilde{\mathbf{K}}'$ and $\tilde{\mathbf{K}}^*$ due to a simple nearest neighbor approximation scheme. We would like to develop sophisticated learning schemes to learn the test kernel matrix $\tilde{\mathbf{K}}'$.

Algorithm Complexity

The proposed algorithm contains two parts: solving $\mathbf{G}$ by Algorithm 1 and solving $\boldsymbol{\alpha}$ in Algorithm 2. For the optimization of $\mathbf{G}$, we use alternating directions to solve this problem with the following three steps. Herein, updating $\mathbf{G}$ (step 1) needs to solve an eigenvalue problem by a k-step Arnoldi process. Thereby, the complexity of a straightforward implementation of the Arnoldi process is at least $\mathcal{O}(kn^2 + k^2 n)$, where n is the number of training data, and k is a constant irrelevant to n, see details in (Lee et al. 2009). Updating $\mathbf{Q}$ in step 2 is related to matrix addition and multiplication operations, and thus the complexity of step 2 reaches to $O(n)$. In step 3, the computation cost for updating $\mathbf{Z}$ and $\boldsymbol{\beta}$ can be ignored. Finally, the computational complexity of Algorithm 1 is $\mathcal{O}(t(kn^2 + k^2 n + n))$, where t is the number of iterations. In Algorithm 2, the computational complexity of solving $\boldsymbol{\alpha}$ by SMO is about $O(dn^2)$ where d is the feature dimension. Finally, the total computational complexity of our algorithm is $\mathcal{O}(T(dn^2 + t(kn^2 + k^2 n)))$, where T is the number of iterations in Algorithm 2. It can be observed that the computation load of Algorithm 1 is large and we would like to further accelerate the solving process for $\mathbf{G}$.

Experiments

In this section, we compare the KNPL model with other representative kernel-based methods on the benchmark with a collection of several datasets. In addition, the convergence analyse of the optimization algorithm in KNPL is also provided.

Experimental Setup

In the experiments, fifteen real-word datasets from UCI Machine Learning Repository (Blake and Merz 1998) are used to evaluate the performance of KNPL with other kernel learning algorithms. The dataset description including the number of training set n and the problem dimension d is presented in Table 1. All data are normalized to $[0, 1]$ in advance. For some datasets, there are both training and test data (e.g., *monks1*). Otherwise, we randomly pick half of the data for training and the rest for test. In our model, the regularization parameter γ is tuned by 5-fold cross validation. That is, we randomly partition the training data into 5 subsets, one of which is used for validation in turn and the remaining ones for training. We compare the proposed KNPL model in SVM with several state-of-the-art kernel-based algorithms including **SVM-CV**, **BMKL** (Gonen 2012), **DMKL** (Strobl and Visweswaran 2014), and **EasyMKL** (Aiolli and Donini 2015).

Experimental Results

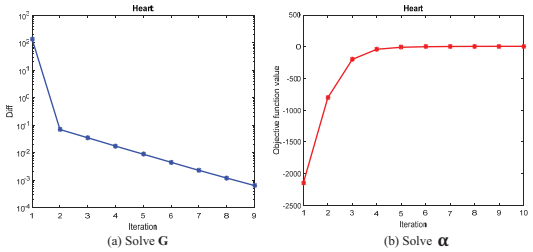
We test the above algorithms on fifteen datasets, where the procedure is repeated 10 times, and then the average classification accuracy and its standard deviation on test data are reported in Table 1. Specifically, the classification accuracies of "SVM-CV" and our method on the training data are also presented in order to show their respective model flexibilities.

Compared with DMKL, BMKL, and EasyMKL, the proposed KNPL model provides a favorable performance, *i.e.* ranking first on eight datasets and second on four datasets. The promising performance effectively demonstrates the superiority of the introduced nonlinear pairwise layer that conveys richer information than other methods. Accordingly, our model is able to have good adaptiveness to the training and test data.

In terms of the results in Table 1 when comparing to "SVM-CV", we firstly analyze four datasets in which the classification accuracy on the training data is not satisfactory, namely: *diabetic*, *haberman*, *heart*, and *SPECT*. It can be observed that KNPL significantly improves the flexibility of SVM on the training data, and thus works well for the test data with a favorable generation ability. Specifically, in *diabetic* and *haberman* datasets, the proposed KNPL model outperforms with the respective margins about 9% and 12% than "SVM-CV", which verifies the effectiveness of the learned pairwise function. Besides, in *monks*, *spambase*, and *EEG* datasets, the proposed KNPL model often slightly outperforms than "SVM-CV" with the margin about 1%$\sim$3%. And also, in the remaining eight datasets, the training accuracy yielded by"SVM-CV" indicates that the flexibility is enough. Therefore, our model can hardly achieve a huge promotion on these datasets to some extent, and even is inferior than "SVM-CV" on *brest-cancer* and *monks2*.

Table 1: Comparison results in terms of classification accuracies (mean±std. deviation %) on UCI datasets. The best performance is highlighted in **bold**. The classification accuracy on the training data is presented by *italic*, and does not participate in ranking.

Dataset	Description	DMKL	BMKL	EasyMKL	SVM-CV		KNPL	
	(d, n)	Test	Test	Test	Training	Test	Training	Test
breast-cancer	(10, 699)	96.56±1.04	98.95±1.32	**99.52±0.13**	*97.42±1.24*	96.53±0.84	*100.0±0.00*	97.35±0.74
climate	(20, 540)	94.00±1.69	93.11±0.46	90.30±1.12	*99.22±1.27*	94.74±0.83	*100.0±0.00*	**95.14±0.98**
diabetic	(19, 1151)	72.95±1.03	74.97±0.49	76.72±1.50	*80.71±3.98*	73.00±1.74	*91.23±0.56*	**81.98±1.72**
guide1-t	(4, 4000)	96.94±0.62	97.00±0.01	94.75±0.92	*97.44±0.43*	96.86±0.31	*98.25±0.27*	**97.12±0.19**
haberman	(3, 306)	69.93±3.01	69.86±1.86	74.56±4.23	*77.38±4.51*	73.26±3.89	*88.81±3.21*	**85.81±4.28**
heart	(13, 270)	80.29±2.70	87.33±0.23	86.67±1.23	*88.96±3.07*	81.92±2.47	*92.59±4.97*	**87.47±3.90**
ionosphere	(33, 351)	90.51±1.90	91.08±1.08	**97.41±0.56**	*98.75±0.99*	90.85±1.37	*100.0±0.00*	93.17±1.19
monks1	(6, 124)	**84.12±3.62**	78.91±2.55	76.73±1.25	*90.32±0.00*	81.48±0.00	*100.0±0.00*	83.38±3.35
monks2	(6, 169)	77.24±5.72	82.12±1.31	76.68±1.44	*100.0±0.00*	**85.81±1.40**	*100.0±0.00*	83.33±1.68
monks3	(6, 122)	90.69±3.20	**94.00±1.09**	81.00±0.67	*96.22±1.50*	93.07±1.24	*100.0±0.00*	88.78±1.23
sonar	(60, 208)	80.57±5.06	84.80±0.60	81.73±3.11	*99.90±0.30*	85.36±3.17	*100.0±0.00*	**85.86±2.86**
SPECT	(21, 80)	78.75±4.47	78.88±0.84	**82.50±3.24**	*87.00±3.78*	73.10±3.23	*92.75±6.99*	79.73±4.82
splice	(60, 1000)	83.87±0.82	**85.80±0.63**	83.97±1.33	*99.56±1.25*	84.96±1.52	*99.96±0.08*	84.90±0.81
spambase	(57,2300)	88.45±2.27	92.91±0.53	95.43±0.53	*95.41±0.72*	92.95±0.48	*98.12±0.33*	**95.52±0.46**
EEG	(14,14980)	73.02±6.13	78.79±4.53	75.22±3.32	*86.32±1.81*	78.34±1.48	*98.77±0.64*	**80.07±4.06**



(a) Solve $\mathbf{G}$ (b) Solve $\boldsymbol{\alpha}$

Figure 3: (a) Residual error Diff versus iteration in Algorithm 1 to solve $\mathbf{G}$; (b) Objective function values versus iteration in Algorithm 2 to solve $\boldsymbol{\alpha}$.

As a result, KNPL not only makes the baseline SVM algorithm more flexible on the training data, but also outperforms several kernel based methods a statistically significant evidence in terms of test accuracy. Moreover, the experimental results indicate that designing an advanced and delicate nonlinear pari-wise layer is rich enough to achieve promising performance when compared to a deep kernel based method.

Convergence Experiments

The experiments about the convergence of our KNPL model are conducted on *Heart* as shown in Fig. 3. We plot the residual error Diff in Algorithm 1 on a logarithmic scale versus iterations as shown in Fig. 3(a). One can see that only a few iterations (about 3) are taken to obtain the optimal solution $\mathbf{G}^*$, so the feasibility of employing ADMM for solving Eq. (12) is verified. In Fig. 3(b), we plot the objective values versus iteration to solve $\boldsymbol{\alpha}$. We can see that the objective values of the maximization sub-problem converges quickly and generally terminates within 10 iterations on *Heart*. This suggests the proposed iterative algorithm can effectively solve the target convex optimization.

Discussion

We have explored a nonlinear pairwise layer, which combines the structural properties of deep architectures with the flexibility of kernel methods, termed "Kernel with Nonlinear Pairwise Layer" (KNPL). As a result of learning the coupling information between any two examples by such layer, the different local statistics of the data is effectively captured. The introduced structure in KNPL consequently enhances the model flexibility of the standard SVM, and also is effectively solved by an iterative algorithm. Thorough experimental results demonstrate that our KNPL model is superior to several state-of-the-art kernel learning methods on classification tasks. However, several issues would be further considered, and we list them as follows: 1) the experimental results on classification accuracy gap suggest we may impose more appropriate constraints on $\mathbf{G}$ to achieve a good trade-off between the model flexibility and over-fitting. In this case, the learned kernel associated with the kernel matrix $\mathbf{G} \odot \mathbf{K}$ might be an indefinite kernel (Ying, Campbell, and Girolami 2009; Huang et al. 2017b) and even asymmetric, which would be further investigated because of both practical and theoretical need. 2) the inconsistency between the training kernel and the test kernel learning prevents our model achieving excited performance on the test data, which suggest us to employ out-of-sample extension (Pan et al. 2017) for the kernel learning.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 61572315, 6151101179, 61603248, 61602246, 61703077), in part by the Natural Science Foundation of Jiangsu Province under Grant BK20171430, in part by 973 Plan, China (No. 2015CB856004).

References

Aiolli, F., and Donini, M. 2015. Easymkl: a scalable multiple kernel learning algorithm. *Neurocomputing* 169:215–224.

Blake, C., and Merz, C. J. 1998. UCI Repository of Machine Learning Databases.

Bonnans, J. F., and Shapiro, A. 1998. Optimization problems with perturbations: A guided tour. *SIAM Review* 40(2):228–264.

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge university press.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3(1):1–122.

Cho, Y. 2012. Kernel methods for deep learning. In *Proceedings of Neural Information Processing Systems*, 342–350.

Evgeniou, T.; Pontil, M.; and Poggio, T. 2000. Regularization networks and support vector machines. *Advances in Computational Mathematics* 13(1):1–50.

Flammarion, N., and Bach, F. 2015. From averaging to acceleration, there is only a step-size. In *Proceedings of the Conference on Learning Theory*, 658–695.

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

He, B.; Xu, M.; and Yuan, X. 2011. Solving large-scale least squares semidefinite programming by alternating direction methods. *SIAM Journal on Matrix Analysis and Applications* 32(1):136–152.

Henrion, D., and Malick, J. 2012. *Projection Methods in Conic Optimization*. Springer US.

Huang, X.; Maier, A.; Hornegger, J.; and Suykens, J. A. K. 2017a. Indefinite kernels in least squares support vector machines and principal component analysis. *Applied and Computational Harmonic Analysis* 43(1):162–172.

Huang, X.; Suykens, J. A.; Wang, S.; Hornegger, J.; and Maier, A. 2017b. Classification with truncated ℓ_1 distance kernel. *IEEE Transactions on Neural Networks and Learning Systems*.

Lee, J.; Balakrishnan, V.; Koh, C. K.; and Jiao, D. 2009. From $\mathcal{O}(k^2N)$ to $\mathcal{O}(N)$: A fast complex-valued eigenvalue solver for large-scale on-chip interconnect analysis. In *Microwave Symposium Digest, 2009. MTT '09. IEEE MTT-S International*, 181–184.

Lloyd, J. R.; Duvenaud, D.; Grosse, R.; Tenenbaum, J. B.; and Ghahramani, Z. 2014. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of AAAI Conference on Artificial Intelligence*, 1242–1250.

Nen, M.; Alpayd; and Ethem, N. 2011. Multiple kernel learning algorithms. *Journal of Machine Learning Research* 12:2211–2268.

Pan, B.; Chen, W. S.; Chen, B.; Xu, C.; and Lai, J. 2017. Out-of-sample extensions for non-parametric kernel methods. *IEEE Transactions on Neural Networks and Learning Systems* 28(2):334–345.

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

Schölkopf, B., and Smola, A. J. 2003. *Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Strobl, E. V., and Visweswaran, S. 2014. Deep multiple kernel learning. In *Proceedings of International Conference on Machine Learning and Applications*, 414–417.

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

Vapnik, V. N. 2000. *The Nature of Statistical Learning Theory*. Springer.

Varma, M., and Babu, B. R. 2009. More generality in efficient multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, 1065–1072.

Wilson, A. G.; Hu, Z.; Salakhutdinov, R.; and Xing, E. P. 2016. Deep kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 370–378.

Ying, Y.; Campbell, C.; and Girolami, M. 2009. Analysis of SVM with indefinite kernels. In *Proceedings of Neural Information Processing Systems*, 2205–2213.

Zhang, L.; Yang, T.; Jin, R.; Jin, R.; and Zhou, Z. H. 2016. Stochastic optimization for kernel PCA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2316–2322.

Zhong, S.; Chen, T.; He, F.; and Niu, Y. 2014. Fast gaussian kernel learning for classification tasks based on specially structured global optimization. *Neural Network* 57(9):51–62.

Zhu, J., and Hastie, T. 2002. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics* 14(1):185–205.

Zhuang, J.; Tsang, I. W.; and Hoi, S. C. H. 2011. Two-layer multiple kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 15, 909–917.