Journal of Machine Learning Research 12 (2011) 1313-1347 Submitted 3/10; Revised 10/10; Published 4/11

A Family of Simple Non-Parametric Kernel Learning Algorithms

Jinfeng Zhuang ZHUA0016@NTU.EDU.SG
Ivor W. Tsang IVORTSANG@NTU.EDU.SG
Steven C.H. Hoi CHHOI@NTU.EDU.SG
School of Computer Engineering
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798

Editor: Tony Jebara

Abstract

Previous studies of *Non-Parametric Kernel Learning* (NPKL) usually formulate the learning task as a Semi-Definite Programming (SDP) problem that is often solved by some general purpose SDP solvers. However, for N data examples, the time complexity of NPKL using a standard interior-point SDP solver could be as high as $O(N^{6.5})$, which prohibits NPKL methods applicable to real applications, even for data sets of moderate size. In this paper, we present a family of efficient NPKL algorithms, termed "**SimpleNPKL**", which can learn non-parametric kernels from a large set of pairwise constraints efficiently. In particular, we propose two efficient SimpleNPKL algorithms. One is SimpleNPKL algorithm with linear loss, which enjoys a *closed-form* solution that can be efficiently computed by the *Lanczos* sparse eigen decomposition technique. Another one is SimpleNPKL algorithm with other loss functions (including square hinge loss, hinge loss, square loss) that can be re-formulated as a saddle-point optimization problem, which can be further resolved by a fast iterative algorithm. In contrast to the previous NPKL approaches, our empirical results show that the proposed new technique, maintaining the same accuracy, is significantly more efficient and scalable. Finally, we also demonstrate that the proposed new technique is also applicable to speed up many kernel learning tasks, including *colored maximum variance unfolding*, *minimum volume embedding*, and *structure preserving embedding*.

Keywords: non-parametric kernel learning, semi-definite programming, semi-supervised learning, side information, pairwise constraints

1. Introduction

Kernel methods have been successfully applied in various real applications ranging from data mining, computer vision and bioinformatics, and often show the state-of-the-art performance (refer to Hofmann, Schölkopf, and Smola, 2008 and references therein). Empirical evidences show that the generalization performance of kernel methods is often dominated by the chosen kernel function. Inappropriate kernels could lead to sub-optimal or even poor results. Therefore, the choice of an effective kernel plays a crucial role in many kernel based machine learning methods. Typically, traditional kernel methods, for example, Support Vector Machines (SVMs), often adopt a predefined kernel function that is empirically chosen from a pool of parametric kernel functions, such as polynomial and Gaussian kernels. One major limitation of such an approach is that choosing an appropriate kernel function manually may require a certain level of expert knowledge, which may be difficult in some situations. Another limitation lies in the difficulty of tuning optimal parameters for the predefined parametric kernel functions.

To address these limitations, a bunch of research on learning effective kernels from data automatically has been actively explored recently. An example technique is *Multiple Kernel Learning* (MKL) (Lanckriet et al., 2004; Bach et al., 2004), which aims at learning a convex combination of several predefined parametric kernels in order to identify a good target kernel for the applications. MKL has been actively studied in many applications, including bio-informatics (Sonnenburg et al., 2006a,b), computer vision (Duan et al., 2009; Sun et al., 2009; Vedaldi et al., 2009), and natural language processing (Mao and Tsang, 2011), etc. Despite some encouraging results reported, these techniques often assume the target kernel function is of some *parametric* forms, which limits their capacity of fitting diverse patterns in real complex applications.

Instead of assuming some parametric forms for the target kernel, an emerging group of kernel learning studies are devoted to *Non-Parametric Kernel Learning* (NPKL) methods, which aim to learn a Positive Semi-Definite (PSD) kernel matrix directly from the data. Example techniques include Cristianini et al. (2002), Lanckriet et al. (2004), Zhu et al. (2005), Zhang and Ando (2006), Kulis et al. (2006), Hoi et al. (2007), Kulis et al. (2009) and Li et al. (2009); Mao and Tsang (2010). NPKL provides a flexible learning scheme of incorporating prior/side information into the known similarity measures such that the learned kernel can exhibit better ability to characterize the data similarity. However, due to the PSD constraint, the resulting optimization task of NPKL is often in the form of Semi-Definite Programing (SDP). Many existing studies have simply solved such SDP problems by some general purpose SDP solvers, which often have the time complexity of $O(N^{6.5})$, making the NPKL solution infeasible to real world large-scale applications.

In this paper, we aim at addressing the efficiency and scalability issues related to the NPKL techniques proposed by Hoi et al. (2007) and Zhuang et al. (2009), which have shown the state-of-the-art empirical performance in several applications (Zhuang and Hoi, 2010). In particular, the main contributions of this paper include:

1. We propose a family of Simple Non-Parametric Kernel Learning (SimpleNPKL) algorithms for efficient and scalable non-parametric kernel learning.

2. We present the first SimpleNPKL algorithm with linear loss function to learn non-parametric kernels from pairwise constraints. The algorithm enjoys a *closed-form* solution that can be computed efficiently by sparse eigen-decomposition techniques, for example, the *Lanczos* algorithm.

3. To achieve more robust performance, we propose the second SimpleNPKL algorithm that has other loss functions (including square hinge loss, hinge loss and square loss), which can be re-formulated as a *mini-max optimization* problem. This optimization can be solved by an efficient iterative projection algorithm that mainly involves the computation of sparse eigen decomposition.

4. To further speed up the SimpleNPKL algorithm of other loss functions, we investigate some active constraint selection techniques to reduce the computation cost at each iteration step.

5. We conducted extensive experiments, which show that SimpleNPKL is significantly more efficient than existing NPKL methods. With the same linear loss function, SimpleNPKL is

on average 40 times faster than the original NPKL using a standard SDP solver. This makes the NPK learning techniques practical to large-scale applications.

6. We extend the proposed SimpleNPKL scheme to resolve other non-parametric kernel learning problems, including *colored maximum variance unfolding* (Song et al., 2008), *minimum volume embedding* (Shaw and Jebara, 2007), and *structure preserving embedding* (Shaw and Jebara, 2009). The encouraging results show that our technique is able to speed up the existing non-parametric kernel learning solutions significantly for several real-world applications.

The rest of this paper is organized as follows. Section 2 presents some background of kernel learning, briefly reviews some representative work on kernel learning research, and indicates the motivations of our work. Section 3 introduces a framework of Non-parametric Kernel Learning (NPKL) from pairwise constraints proposed by Hoi et al. (2007). Section 4 describes our proposed SimpleNPKL algorithms, which aim to resolve the NPKL task efficiently. Section 5 discusses some implementation issues for developing a fast solver in practice. Section 6 extends our technique to speed up other kernel learning methods. Section 7 gives our empirical results and Section 8 concludes this work.

2. Background Review and Related Work

In this Section, we review some backgrounds of kernel methods, and related work on kernel learning research.

2.1 Notations

For the notation throughout the paper, we adopt bold upper case letter to denote a matrix, for example, $\mathbf{A} \in \mathbb{R}^{m\times n}$, and A_{ij} to denote the entry at the ith row and jth column of the matrix $\mathbf{A}$, and bold lower case letter to denote a vector, for example, $\mathbf{x} \in \mathbb{R}^d$. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the column vectors with all zeros and all ones, respectively, and $\mathbf{I}$ to denote an identity matrix. For some algebraic operations:

- $\mathbf{x}'$ denotes the transpose of $\mathbf{x}$;
- $[\mathbf{x}]_i$ denotes the ith element of $\mathbf{x}$;
- $\mathbf{x}^p$ denotes the element-wise power of $\mathbf{x}$ with degree p;
- $|\mathbf{x}|$ denotes the vector with entries equal to the absolute value of the entries of $\mathbf{x}$;
- $\|\mathbf{x}\|_p$ denotes p-norm of $\mathbf{x}$, that is, $\sqrt[p]{\sum_i [\mathbf{x}^p]_i}$;
- $\mathbf{x}_i \circ \mathbf{x}_j$ denotes the element-wise multiplication between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$;
- $\mathbf{x} \geq \mathbf{0}$ means all entries in $\mathbf{x}$ is larger than or equal to 0;
- $\mathbf{K} \succeq \mathbf{0}$ denotes a matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ that is symmetric and positive semi-definite;
- $\mathbf{K}^p$ denotes the power of a symmetric matrix $\mathbf{K}$ with degree p;
- $\operatorname{tr} \mathbf{K} = \sum_i K_{ii}$ denotes the trace of a matrix $\mathbf{K}$;
- $\langle \mathbf{A}, \mathbf{B} \rangle = \operatorname{tr} \mathbf{AB} = \sum_{ij} A_{ij} B_{ij}$ computes the inner product between two square matrices $\mathbf{A}$ and $\mathbf{B}$. We also use it to denote general inner product of two square matrices.
- $\|\mathbf{K}\|_F = \sqrt{\sum_{ij} K_{ij}^2} = \sqrt{\operatorname{tr} \mathbf{KK}}$ denotes the Frobenius norm of a matrix $\mathbf{K}$;
- $\mathbf{A} \circ \mathbf{B}$ denotes the element-wise multiplication between two matrices $\mathbf{A}$ and $\mathbf{B}$.

2.2 Kernel Methods

In general, kernel methods work by embedding data in some Hilbert spaces, and searching for linear relations in the Hilbert spaces. The embedding is often done implicitly by only specifying inner products between any pair of examples (Hofmann et al., 2008). More formally, given an input space $\mathcal{X}$, and an embedding space $\mathcal{F}$, we can define a mapping $\Phi : \mathcal{X} \to \mathcal{F}$. For any two examples $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{x}_j \in \mathcal{X}$, the function k that returns the inner product between the two embedded examples in the space $\mathcal{F}$ is known as the kernel function, that is,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle.$$

Given the kernel function k, a matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ is called a *kernel matrix*, also known as *gram matrix*, if $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for a collection of examples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$. Note that the choice of kernel plays a central role for the success of kernel methods. However, the selection of proper kernels is nontrivial. An inappropriate kernel could result in sub-optimal or even poor performances. Therefore, learning kernels from data has become an active research topic.

2.3 Kernel Learning

We refer the term *kernel learning* to the problem of learning a kernel function or a kernel matrix from given data, corresponding to the inductive and transductive learning setting, respectively. Due to the large volume of works on this topic, we do not intend to make this Section encyclopedic. Instead, we summarize some key ideas behind representative kernel learning schemes. We discuss the strengths and limitations of existing NPKL methods, which motivates our efficient SimpleNPKL solution.

2.3.1 MULTIPLE KERNEL LEARNING AND BEYOND

Multiple kernel learning (MKL), initiated by Lanckriet et al. (2004), has been widely studied in classical supervised learning tasks. The goal is to learn both the associated kernel of a Reproducing Kernel Hilbert Space (RKHS) and the classifier in this space simultaneously:

$$\begin{array}{ll} \min_{\mathbf{K} \in \mathcal{K}} \max_{\alpha} & \alpha'\mathbf{1} - \frac{1}{2}\langle (\alpha \circ \mathbf{y})(\alpha \circ \mathbf{y})', \mathbf{K} \rangle \\ \text{s.t.} & \alpha'\mathbf{y} = 0, \;\; 0 \le \alpha_i \le C, \end{array} \tag{1}$$

where the solution space $\mathcal{K}$ is assumed to be in a convex hull spanned from m basic kernels: $\mathbf{K} = \{\sum_i p_i \mathbf{K}_i : 0 \le p_i \le 1, i = 1, \ldots, m\}$. Thus the optimization over $\mathbf{K}$ is reduced to optimizing the weight vector $\mathbf{p}$. Many studies have been focused on how to efficiently solve the optimization in (1) (Bach et al., 2004; Sonnenburg et al., 2006b; Rakotomamonjy et al., 2008; Xu et al., 2008).

The assumption of MKL on the target kernel $\mathbf{K} = \sum_i p_i \mathbf{K}_i$ implies to concatenate the mapped feature spaces. Therefore, MKL is a natural choice where the data has multiple views or heterogeneous representations. Apparently, there is “no free lunch” for kernel selection. Based on different assumptions about the optimization domain $\mathcal{K}$, one can propose different objective functions. For example, generating a series of base kernels by varying the free kernel parameters could make the cardinality $|\mathcal{K}|$ arbitrarily large. Argyriou et al. (2005) and Gehler and Nowozin (2008) discussed some interesting techniques for such situation. Other variants of MKL techniques can also be found in Lewis et al. (2006), Gönen and Alpaydin (2008), Varma and Babu (2009) and Zhuang et al. (2011).

One basic motivation of kernel learning is to further relax the optimization domain $\mathcal{K}$ such that the learned kernel can be as flexible as possible to fit the complex data. This motivates *Non-Parametric Kernel Learning* (NPKL) methods, which do not assume any parametric form of the target kernel functions/matrices.

2.3.2 NON-PARAMETRIC KERNEL LEARNING

By simply relaxing the optimization domain $\mathcal{K}$, one can turn a regular kernel learning scheme to some NPKL formulations. For example, a recent approach, called *indefinite kernel learning* (Luss and d'Aspremont, 2008; Chen and Ye, 2008; Ying et al., 2010), extends the MKL formulation to learn non-parametric kernels from an indefinite kernel $\mathbf{K}_0$, which does not assume the convex hull assumption. The indefinite kernel learning rewrites the objective function of (1) as follows:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \max_{\alpha} \quad \alpha'\mathbf{1} - \frac{1}{2}\langle(\alpha\circ\mathbf{y})(\alpha\circ\mathbf{y})', \mathbf{K}\rangle + \gamma\|\mathbf{K}-\mathbf{K}_0\|_F^2, \tag{2}$$

where $\mathbf{K}_0$ is an initial kernel, which could be indefinite.

The kernel learning formulation discussed above aims to optimize both the classifier and the kernel matrix simultaneously. Some theoretical foundations, such as existence and uniqueness of the target kernel, were given in Micchelli and Pontil (2005) and Argyriou et al. (2005).

Another line of kernel learning research mainly focuses on optimizing the kernel only with respect to some criteria under some prior constraints or heuristics. An important technique is the kernel target alignment criterion proposed in Cristianini et al. (2002), which guides the kernel learning task to optimize the kernel by maximizing the alignment between the training data examples and the class labels of the training examples:

$$\max_{\mathbf{K}\succeq\mathbf{0}} \quad \frac{\langle\mathbf{K}_{N_l}, \mathbf{T}\rangle}{\sqrt{\langle\mathbf{K}_{N_l}, \mathbf{K}_{N_l}\rangle\langle\mathbf{T}, \mathbf{T}\rangle}}, \tag{3}$$

where $\mathbf{T} = \mathbf{y}\mathbf{y}'$ is the outer product of labels, $\mathbf{K}_{N_l}$ is the sub-matrix of which the entry values are kernel evaluation on N_l labeled data examples. Note that $\mathbf{T}$ could be obtained by empirical experiments and more general than class labels. The objective (3) only involves the labeled data. A popular assumption is to treat $\mathbf{K}$ to be spanned by the eigen-vectors of some known kernel defined over both the labeled and unlabeled data (Chapelle et al., 2003; Zhu et al., 2005; Hoi et al., 2006; Zhang and Ando, 2006; Johnson and Zhang, 2008): $\mathcal{K} = \{\sum_i \lambda_i \mathbf{v}\mathbf{v}' : \lambda_i \geq 0\}$. Thus the optimization variables are reduced from the entire kernel matrix $\mathbf{K}$ to the kernel spectrum λ.

Recently Hoi et al. (2007) proposed an NPKL technique that aims to learn a fully non-parametric kernel matrix from pairwise constraints. The target kernel is maximally aligned to the constraint matrix $\mathbf{T}$ and minimally aligned to the graph Laplacian. The objective can be deemed as a form of kernel target alignment without normalization. Since our proposed family of SimpleNPKL algorithms follows this framework, we will discuss the details of this formulation in Section 3.

Besides the normalized inner product, which measures the similarity between $\mathbf{K}$ and the target $\mathbf{T}$, researchers have also proposed dissimilarity based criteria. In fact, the preceding indefinite kernel learning (2) employs the Euclidean distance to measure the dissimilarity between kernels. Besides, another example in Kulis et al. (2006) employed the Bregman divergence to measure distance between $\mathbf{K}$ and a known kernel $\mathbf{K}_0$:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \quad D_\phi(\mathbf{K}, \mathbf{K}_0) \triangleq \operatorname{tr} \mathbf{K}\mathbf{K}_0^{-1} - \log\det(\mathbf{K}\mathbf{K}_0^{-1}) - N, \tag{4}$$

where D_ϕ is a Bregman divergence (Kulis et al., 2006).

The optimization over the above learning objective function (3) or (4) will simply return the trivial solution $\mathbf{K}_0$ without additional constraints, which would make NPKL meaningless. In practice, some prior knowledge about the target kernel will be added to constrain the solution space $\mathcal{K}$. Most of the existing constraints over the entries of $\mathbf{K}$ could be expressed by $\operatorname{tr} \mathbf{KT} \leq b$. For example, as discussed in Kwok and Tsang (2003), the square of distance between two data examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space can be expressed by $\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|_2^2 = K_{ii} + K_{jj} - 2K_{ij} = \operatorname{tr} \mathbf{KT}_{ij}$, where $\mathbf{T}_{ij}$ is a matrix of $N \times N$ only taking non-zeros at $T_{ii} = T_{jj} = 1, T_{ij} = T_{ji} = -1$. Moreover, one can introduce slack variables for soft constraints.

Besides, some regularization terms over kernel $\mathbf{K}$ are often included during the optimization phase. For example, fixing the trace $\operatorname{tr} \mathbf{K} = 1$ is rather common in SDP solvers.

At last, we summarize the typical schemes of existing NPKL methods:

- To encourage the similarity (e.g., kernel target alignment) or penalize the distance (e.g., Bregman divergence) to some prior similarity information;
- To enforce some constraints to the kernel $\mathbf{K}$ with prior heuristics, such as distance constraint $K_{ii} + K_{jj} - 2K_{ij} = d_{ij}^2$, or side information, etc; and
- To include regularization terms over $\mathbf{K}$ to control capacity, such as $\operatorname{tr} \mathbf{K} = 1$.

By the above steps, NPKL provides a flexible scheme to incorporate more prior information into the target kernel. Due to the non-parametric nature, the solution space $\mathcal{K}$ is capable of fitting diverse empirical data such that the learned kernel $\mathbf{K}$ can be more effective and powerful to achieve better empirical performance than traditional parametric kernel functions.

2.3.3 OPTIMIZATION ASPECTS

Despite the powerful capacity achieved by NPKL, one key challenge with NPKL is the difficulty of the resulting optimization problem, in which

- the whole gram matrix $\mathbf{K}$ is treated as the optimization variable, that is, $O(N^2)$ variables;
- the kernel matrix $\mathbf{K}$ must be positive semi-definite.

As a result, NPKL is often turned into a Semi-Definite Programming (SDP) problem. For instance, a NPKL problem to learn a kernel matrix $\mathbf{K}$ with m linear constraints is written as follows:

$$\max_{\mathbf{K} \succeq \mathbf{0}} \operatorname{tr} \mathbf{CK} \quad : \quad \operatorname{tr} \mathbf{T}_{ij}\mathbf{K} = b_{ij}, \tag{5}$$

where $\mathbf{C}$ and $\mathbf{T}_{ij}$'s are $N \times N$ symmetric matrices and b_{ij}'s are scalars, and its dual problem can be rewritten as:

$$\min_{\mathbf{y}} \mathbf{b}'\mathbf{y} \quad : \quad \mathbf{C} - \sum_{(i,j)} \mathbf{T}_{ij} y_{ij} \succeq \mathbf{0}, \tag{6}$$

where $\mathbf{y}$ is the vector of dual variables y_{ij}'s for the linear constraints in (5) and $\mathbf{b}$ is the vector of b_{ij}'s.

Typically, this SDP problem of NPKL is usually solved by applying a general purpose SDP solver. Among various SDP solvers, the *interior-point algorithm* is one of the state-of-the-art solutions (Boyd and Vandenberghe, 2004). From Lobo et al. (1998), the time complexity per iteration of the SDP problem (6) is $O(m^2N^2)$. Using the primal-dual method for solving this SDP, the accuracy of a given solution can be improved by an absolute constant factor in $O(\sqrt{N})$ iterations (Nesterov and Nemirovskii, 1994). When m approaches to $O(N^2)$, the overall computation complexity is often as high as $O(N^{6.5})$, which makes NPKL inapplicable to real applications.

In this work, we focus on solving the efficiency and scalability issues of NPKL (Zhuang et al., 2009). In particular, we propose a family of efficient SimpleNPKL algorithms that can solve large-scale NPKL problems efficiently. Moreover, we also show that the proposed algorithms are rather general, which can be easily extended to solving other kernel learning applications, including dimensionality reduction and data embedding applications.

3. A Framework of Non-Parametric Kernel Learning from Pairwise Constraints

In this Section, we introduce the framework of Non-Parametric Kernel Learning (NPKL) (Hoi et al., 2007; Zhuang et al., 2009), which aims to learn non-parametric kernels from side information, which is presented in a collection of must-link and cannot-link pairs.

3.1 Side / Label Information

Let $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ denote the entire data collection, where each data point $\mathbf{x}_i \in \mathcal{X}$. Consider a set of N_l labeled data examples, $\mathcal{L} = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_{N_l}, y_{N_l})\}$, one can use $y_i y_j$ as the similarity measurement for any two patterns $\mathbf{x}_i$ and $\mathbf{x}_j$. Sometimes, it is possible that the class label information is not readily available, while it is easier to obtain a collection of similar (positive) pairwise constraints $\mathcal{S}$ (known as "must-links", that is, the data pairs share the same class) and a collection of dissimilar (negative) pairwise constraints $\mathcal{D}$ (known as "cannot-links", that is, the data pairs have different classes). These pairwise constraints are often referred to as *side information*.

In general, kernel learning with labeled data can be viewed as a special case of kernel learning with side information (Kwok and Tsang, 2003; Kulis et al., 2006; Hoi et al., 2007), that is, one can construct the sets of pairwise constraints $\mathcal{S}$ and $\mathcal{D}$ from $\mathcal{L}$. In real applications, it is often easier to detect pairwise constraint while the class label is difficult to obtain. For example, in bioinformatics, the interaction between two proteins can be identified by empirical experiments. These interactions are expressed naturally by pairwise constraints. However, it could be very difficult to judge the protein function, which corresponds to class labels. In the sequel, we focus on learning kernels from pairwise constraints.

Given $\mathcal{S}$ and $\mathcal{D}$, we construct a similarity matrix $\mathbf{T} \in \mathbb{R}^{N \times N}$ to represent the pairwise constraints, that is,

$$T_{ij} = \begin{cases} +1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ -1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

A straightforward and intuitive principle for kernel learning is that the kernel entry K_{ij} should be aligned with the side information T_{ij} as much as possible (Cristianini et al., 2002), that is, the alignment $T_{ij}K_{ij}$ of each kernel entry is maximized.

3.2 Locality Preserving Regularization

In addition to side/label information, preserving the intrinsic geometric structure of the data have also been explored to improve the performance of kernel learning. Typically, most existing kernel learning approaches (Kulis et al., 2006; Hoi et al., 2007; Hoi and Jin, 2008) adopt the low dimensional data manifold (Sindhwani et al., 2005) for preserving the locality of the data in kernel learning. The following reviews an approach for exploring low dimensional data manifold in kernel learning (Hoi et al., 2007).

Let us denote by $f(\mathbf{x},\mathbf{x}')$ a similarity function that measures the similarity between any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathbf{S} \in \mathbb{R}^{N\times N}$ is a similarity matrix where each element $S_{ij} = f(\mathbf{x}_i,\mathbf{x}_j) \geq 0$. Note that $f(\cdot,\cdot)$ does not have to be a kernel function that satisfies the Mercer's condition. For a given N data examples, a kernel matrix $\mathbf{K}$ can be expressed as $\mathbf{K} = \mathbf{V}'\mathbf{V} \succeq \mathbf{0}$, where $\mathbf{V} = [\mathbf{v}_1,\ldots,\mathbf{v}_N]$ is the matrix of the embedding of the N data examples. The regularizer of the kernel matrix $\mathbf{K}$, which captures the local dependency between the embedding of $\mathbf{v}_i$ and $\mathbf{v}_j$ (i.e., the low dimensional embedding of similar data examples should be similar w.r.t. the similarity S_{ij}), can be defined as:

$$\begin{aligned}\Omega(\mathbf{V},\mathbf{S}) &= \frac{1}{2}\sum_{i,j=1}^{N} S_{ij}\left\|\frac{\mathbf{v}_i}{\sqrt{D_i}} - \frac{\mathbf{v}_j}{\sqrt{D_j}}\right\|_2^2 \\ &= \text{tr}\,(\mathbf{V}\mathbf{L}\mathbf{V}^{'}) = \text{tr}\,(\mathbf{L}\mathbf{K}), \qquad (8)\end{aligned}$$

where $\mathbf{L}$ is the graph Laplacian matrix defined as:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}, \qquad (9)$$

where $\mathbf{D} = \text{diag}(D_1, D_2, \ldots, D_N)$ is a diagonal matrix with the diagonal elements defined as $D_i = \sum_{j=1}^{N} S_{ij}$.

3.3 Formulation of Non-Parametric Kernel Learning

Taking into consideration of both the side information in (7) and the regularizer in (8), the NPKL problem is then formulated into the *loss + regularization* framework (Hoi et al., 2007) as follows:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \quad \text{tr}\,\mathbf{L}\mathbf{K} + C\textstyle\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \ell\big(T_{ij}K_{ij}\big), \qquad (10)$$

which generally belongs to a Semi-Definite Programming (SDP) problem (Boyd and Vandenberghe, 2004). Here, $C > 0$ is a tradeoff parameter to control the empirical loss[1] $\ell(\cdot)$ of the alignment $T_{ij}K_{ij}$ of the target kernel and the dependency among data examples with respect to the intrinsic data structure.

4. SimpleNPKL: Simple Non-Parametric Kernel Learning

In this Section, we present a family of efficient algorithms for solving the NPKL problem in (10). We refer to the proposed efficient algorithms as "SimpleNPKL" for short.

4.1 Regularization on K

As aforementioned, the solution space of NPKL has been relaxed to boost its flexibility and capacity of fitting diverse patterns. However, arbitrarily relaxing the solution space $\mathcal{K}$ could result in over-fitting. To alleviate this problem, we introduce a regularization term:

$$\text{tr}\,(\mathbf{K}^p), \qquad (11)$$

1. The common choice of the loss function $\ell(\cdot)$ can be hinge loss, square hinge loss or linear loss.

where $p \geq 1$ is a parameter to regulate the capacity of $\mathbf{K}$. Similar regularization terms on $\mathbf{K}$ have also been adopted in previous kernel learning studies. For example, Cristianini et al. (2002) used $\|\mathbf{K}\|_F = \sqrt{\langle \mathbf{K}, \mathbf{K} \rangle} = \sqrt{\mathrm{tr}\,(\mathbf{K}^2)}$ in the objective to penalize the complexity of $\mathbf{K}$; while Lanckriet et al. (2004) proposed to adopt a hard constraint $\mathrm{tr}\,(\mathbf{K}) \leq B$, where $B > 0$ a constant, to control the capacity of $\mathbf{K}$.

We refer the modified NPK learning problem with the regularization term (11) either in the objective or in the constraint to as Simple Non-Parametric Kernel Learning (SimpleNPKL), which can be solved efficiently without engaging any standard SDP solvers. Next we present two SimpleNPKL algorithms that adopt several different types of loss functions.

4.2 SimpleNPKL with Linear Loss

First of all, we consider a linear loss function $\ell(f) = -f$, and rewrite the formulation of (10) as the SimpleNPKL formulation:

$$\min_{\mathbf{K}} \mathrm{tr}\left(\left(\mathbf{L} - C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \mathbf{T}_{ij}\right)\mathbf{K}\right) \;:\; \mathbf{K} \succeq \mathbf{0},\ \mathrm{tr}\,\mathbf{K}^p \leq B, \tag{12}$$

where $\mathbf{T}_{ij}$ is the matrix of setting the (i, j)-th entry to T_{ij} and other entries to 0. To solve this problem, we first present a proposition below.

Proposition 1 *Given* $\mathbf{A}$ *is any symmetric matrix such that* $\mathbf{A} = \mathbf{P}diag(\sigma)\mathbf{P}'$, *where* $\mathbf{P}$ *contains columns of orthonormal eigenvectors of* $\mathbf{A}$ *and* σ *is a vector of the corresponding eigenvalues, and B is any positive constant, the optimal solution* $\mathbf{K}^*$ *to the following SDP problem for* $p > 1$*:*

$$\max_{\mathbf{K}} tr\,\mathbf{AK} \;:\; \mathbf{K} \succeq \mathbf{0},\ tr\,\mathbf{K}^p \leq B, \tag{13}$$

can be expressed as the following closed-form solution:

$$\mathbf{K}^* = \left(\frac{B}{tr\,\mathbf{A}_+^{\frac{p}{p-1}}}\right)^{\frac{1}{p}} \mathbf{A}_+^{\frac{1}{p-1}} \tag{14}$$

where $\mathbf{A}_+ = \mathbf{P}diag(\sigma_+)\mathbf{P}'$, *and* σ_+ *is a vector with entries equal to* $\max(0, [\sigma]_i)$.

For $p = 1$, *the optimal solution* $\mathbf{K}^*$ *can be expressed as the following closed-form solution:*

$$\mathbf{K}^* = B\mathbf{A}_1$$

where $\mathbf{A}_1 = \mathbf{P}diag(\sigma_1)\mathbf{P}'$, *and* σ_1 *is a vector with entries equal to* $\frac{1}{\sum_{i:[\sigma]_i=\max_i[\sigma]_i} 1}$ *for all i that* $[\sigma]_i = \max_i[\sigma]_i$*; otherwise, the entries are zeros.*

Proof By introducing a dual variable $\gamma \geq 0$ for the constraint $\mathrm{tr}\,\mathbf{K}^p \leq B$, and $\mathbf{Z} \in \mathcal{S}_+^n$ ($\mathcal{S}_+^n$ is self-dual) for the constraint $\mathbf{K} \succeq \mathbf{0}$, we have the Lagrangian of (13):

$$\mathcal{L}(\mathbf{K}; \gamma, \mathbf{Z}) = \mathrm{tr}\,\mathbf{AK} + \gamma(B - \mathrm{tr}\,\mathbf{K}^p) + \mathrm{tr}\,\mathbf{KZ}.$$

By the Karush-Kuhn-Tucker (KKT) conditions, we have:

$$\mathbf{A} - \gamma p \mathbf{K}^{p-1} + \mathbf{Z} = \mathbf{0} \quad \text{and} \quad \mathrm{tr}\,\mathbf{KZ} = 0.$$

First, we show that tr $(\mathbf{KZ}) = 0$ is equivalent to $\mathbf{KZ} = \mathbf{ZK} = \mathbf{0}$. Since $\mathbf{K} \succeq \mathbf{0}, \mathbf{Z} \succeq \mathbf{0}$, we have tr $(\mathbf{KZ}) =$ tr $(\mathbf{K}^{1/2}\mathbf{K}^{1/2}\mathbf{Z}^{1/2}\mathbf{Z}^{1/2}) = \|\mathbf{K}^{1/2}\mathbf{Z}^{1/2}\|_F^2$. Thus, tr $(\mathbf{KZ}) = 0$ follows that $\mathbf{K}^{1/2}\mathbf{Z}^{1/2} = \mathbf{0}$. Pre-multiplying by $\mathbf{K}^{1/2}$ and post-multiplying by $\mathbf{Z}^{1/2}$ yields $\mathbf{KZ} = \mathbf{0}$, which in turn implies $\mathbf{KZ} = \mathbf{0} = (\mathbf{KZ})' = \mathbf{ZK}$. Hence, $\mathbf{K}$ and $\mathbf{Z}$ can be simultaneously diagonalized by the same set of orthonormal eigenvectors (Alizadeh et al., 1997). From the first KKT condition we have $\mathbf{A} = \gamma p \mathbf{K}^{p-1} - \mathbf{Z}$. Consequently, $\mathbf{A}$ can also be diagonalized with the same eigenvectors as $\mathbf{K}$ and $\mathbf{Z}$.

Assume $\mathbf{A} = \mathbf{P}\text{diag}(\sigma)\mathbf{P}'$, where $\mathbf{P}$ contains columns of orthonormal eigenvectors of $\mathbf{A}$, and σ is the vector of the corresponding eigenvalues. Then, $\mathbf{K} = \mathbf{P}\text{diag}(\lambda)\mathbf{P}'$ and $\mathbf{Z} = \mathbf{P}\text{diag}(\mu)\mathbf{P}'$, where $\lambda \geq \mathbf{0}$ and $\mu \geq \mathbf{0}$ denote the vector of the eigenvalues of $\mathbf{K}$ and $\mathbf{Z}$ respectively. Therefore, we have

$$\begin{aligned} \text{tr}\,\mathbf{K}^p &= \|\lambda\|_p^p \leq B, && (15)\\ \text{tr}\,\mathbf{AK} &= \lambda'\sigma, && (16)\\ \sigma &= \gamma p \lambda^{p-1} - \mu, && (17)\\ \lambda'\mu &= 0. && (18) \end{aligned}$$

Together with $\lambda \geq \mathbf{0}$ and $\mu \geq \mathbf{0}$, and from (18), $[\lambda]_i$ and $[\mu]_i$ cannot be both non-zeros. Hence, from (17), we know $\sigma_+ = \gamma p \lambda^{p-1}$ contains all positive components of σ. Moreover, from (16) and $\lambda \geq \mathbf{0}$, together with the constraint (15), the SDP problem (13) is reduced to

$$\max_{\lambda} \lambda'\sigma_+ \ : \ \|\lambda\|_p^p \leq B.$$

By Hölder inequality, we have $\lambda'\sigma_+ \leq \|\lambda\|_p \|\sigma_+\|_q$, where it holds for $1/p + 1/q = 1$. The equality is achieved if and only if $|\lambda|^p$ and $|\sigma_+|^q$ are linearly dependent. Thus we can scale $\mathbf{K}$ satisfying (15) to arrive at the closed-form solution of $\mathbf{K}$ in (14) for $p > 1$.

For $p = 1$, from Equations (15) and (16), the optimization task is simplified as $\max \lambda'\sigma$: $\lambda \geq 0, \|\lambda\|_1 \leq B$. Due to the linearity, the maximum objective value is obtained by choosing $[\lambda]_i = B / \sum_{i:[\sigma]_i = \max_i [\sigma]_i} 1$ for all i that $[\sigma]_i = \max_i [\sigma]_i$; otherwise, $[\lambda]_i = 0$. ■

Based on Proposition 1, we can easily solve the SimpleNPKL problem. In particular, by setting $\mathbf{A} = C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \mathbf{T}_{ij} - \mathbf{L}$, we can directly compute the optimal $\mathbf{K}^*$ to SimpleNPKL of (12) using sparse eigen-decomposition as in (14). Thus the computation cost of SimpleNPKL with linear loss is dominated by eigen-decomposition. It is clear that this can significantly reduce the time cost for the NPKL tasks. Alternatively, we add tr $(\mathbf{K}^p)$ directly into the objective, and arrive at the following formulation:

$$\min_{\mathbf{K}} \text{tr} \left(\left(\mathbf{L} - C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \mathbf{T}_{ij} \right) \mathbf{K} \right) + \frac{G}{p} \text{tr}\,\mathbf{K}^p \ : \ \mathbf{K} \succeq \mathbf{0},$$

where $G > 0$ is a tradeoff parameter. To solve this problem, we first present a proposition below.

Proposition 2 *Given* $\mathbf{A}$ *is any symmetric matrix such that* $\mathbf{A} = \mathbf{P}\textit{diag}(\sigma)\mathbf{P}'$, *where* $\mathbf{P}$ *contains columns of orthonormal eigenvectors of* $\mathbf{A}$ *and* σ *is a vector of the corresponding eigenvalues, and* B *is any positive constant, the optimal solution* $\mathbf{K}^*$ *to the following SDP problem for* $p > 1$*:*

$$\max_{\mathbf{K}} tr\,\mathbf{AK} - \frac{G}{p} tr\,\mathbf{K}^p \ : \ \mathbf{K} \succeq \mathbf{0}, \tag{19}$$

can be expressed as the following closed-form solution:

$$\mathbf{K}^* = \left(\frac{1}{G}\mathbf{A}_+\right)^{\frac{1}{p-1}} \tag{20}$$

where $\mathbf{A}_+ = \mathbf{P}diag(\sigma_+)\mathbf{P}'$*, and* σ_+ *is a vector with entries equal to* $\max(0, [\sigma]_i)$*.*

Following the techniques in the proof of Proposition 1, we obtain (20) immediately. If we set $G = \left(\frac{1}{B}\text{tr}\,\mathbf{A}_+^{\frac{p}{p-1}}\right)^{\frac{p-1}{p}}$, these two formulations result in exactly the same solution. Moreover, if we set $B = \text{tr}\,\mathbf{A}_+^{\frac{p}{p-1}}$, it means we just use the projection $\mathbf{A}_+$ as $\mathbf{K}$. No re-scaling of $\mathbf{A}_+$ is performed. In the sequel, we consider the regularization $\text{tr}\,\mathbf{K}^p$ with $p = 2$ for its simplicity and smoothness.

4.3 SimpleNPKL with Square Hinge Loss

Although the formulation with linear loss in (12) gives rise to a closed-form solution for the NPKL, one limitation of the NPKL formulation with linear loss is that it may be sensitive to noisy data due to the employment of the linear loss function. To address this issue, in this section, we present another NPKL formulation that uses (square) hinge loss $\ell(f) = (\max(0, 1-f))^d/d$, which sometimes can be more robust, where $d = 1$ (hinge loss) or 2 (square hinge loss). We first focus on the NPKL formulation with square hinge loss, which can be written into the following constrained optimization:

$$\min_{\mathbf{K},\varepsilon_{ij}} \quad \text{tr}\,\mathbf{LK} + \frac{C}{2}\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \varepsilon_{ij}^2 \tag{21}$$

$$\text{s.t.} \quad \forall (i,j)\in(\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} \geq 1-\varepsilon_{ij}, \tag{22}$$
$$\mathbf{K}\succeq \mathbf{0}, \text{tr}\,\mathbf{K}^p \leq B.$$

Note that we ignore the constraints $\varepsilon_{ij} \geq 0$ since they can be satisfied automatically. However, (21) is not in the form of (13), and thus there is no longer a closed-form solution for $\mathbf{K}$.

4.3.1 DUAL FORMULATION: THE SADDLE-POINT MINIMAX PROBLEM

By Lagrangian theory, we introduce dual variables α_{ij}'s ($\alpha_{ij} \geq 0$) for the constraints in (22), and derive a partial Lagrangian of (21):

$$\text{tr}\,\mathbf{LK} + \frac{C}{2}\sum_{(i,j)} \varepsilon_{ij}^2 - \sum_{(i,j)} \alpha_{ij}(T_{ij}K_{ij} - 1 + \varepsilon_{ij}). \tag{23}$$

For simplicity, we use $\sum_{(i,j)}$ to replace $\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})}$ in the sequel. By setting the derivatives of (23) w.r.t. the primal variables ε_{ij}'s to zeros, we have

$$\forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ C\varepsilon_{ij} = \alpha_{ij} \geq 0$$

and substituting them back into (23), we arrive at the following saddle-point minimax problem $J(\mathbf{K},\alpha)$:

$$\max_{\alpha}\min_{\mathbf{K}} \ \text{tr}\left(\left(\mathbf{L} - \sum_{(i,j)} \alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K}\right) - \frac{1}{2C}\sum_{(i,j)} \alpha_{ij}^2 + \sum_{(i,j)} \alpha_{ij} \tag{24}$$
$$\text{s.t.} \quad \mathbf{K}\succeq \mathbf{0},\ \text{tr}\,\mathbf{K}^p \leq B,\ \forall (i,j) \in \mathcal{S}\cup\mathcal{D},\ \alpha_{ij} \geq 0,$$

where $\alpha = [a_{ij}]$ denotes a matrix of dual variables α_{ij}'s for $(i,j) \in \mathcal{S} \cup \mathcal{D}$, and other entries are zeros. This problem is similar to the optimization problem of DIFFRAC (Bach and Harchaoui, 2008), in which K and α can be solved by an iterative manner.

4.3.2 ITERATIVE ALGORITHM

In this subsection, we present an iterative algorithm which follows the similar update strategy in Boyd and Xiao (2005): 1) For a fixed α_{t-1}, we can let $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^{t-1} \mathbf{T}_{ij} - \mathbf{L}$. Based on Proposition 1, we can compute the closed form solution $\mathbf{K}_t$ to (24) using (14); 2) For a fixed $\mathbf{K}_t$, we can update α_t using $\alpha_t = (\alpha_{t-1} + \eta_t \nabla J_t)_+$; 3) Step 1) and 2) are iterated until convergence. Here J denotes the objective function (24), ∇J_t abbreviates the derivative of J at α_t, and $\eta_t > 0$ is a step size parameter. The following Lemma guarantees the differentiable properties of the optimal value function (Bonnans and Shapiro, 1996; Ying et al., 2010):

Lemma 3 *Let $\mathcal{X}$ be a metric space and $\mathcal{U}$ be a normed space. Suppose that for all $x \in \mathcal{X}$ the function $f(x,\cdot)$ is differentiable and that $f(x,u)$ and $\nabla_u f(x,u)$ are continuous on $\mathcal{X} \times \mathcal{U}$, and Q be a compact subset of $\mathcal{X}$. Then the optimal value function $f(u) := \inf_{x \in Q} f(x,u)$ is differentiable. When the minimizer $x(u)$ of $f(\cdot,u)$ is unique, the gradient is given by $\nabla f(u) = \nabla_u f(u, x(u))$.*

From Proposition 1, we see that the minimizer $\mathbf{K}(\alpha)$ is unique for some fixed α. Together with the above lemma, we compute the gradient at the point α by:

$$\nabla J_{ij} = 1 - \mathrm{tr}\, \mathbf{T}_{ij} \mathbf{K} - \frac{1}{C} \alpha_{ij}, \tag{25}$$

where $\mathbf{K} = \left(\frac{B}{\mathrm{tr}\, \mathbf{A}_+^{\frac{p}{p-1}}} \right)^{\frac{1}{p}} \mathbf{A}_+^{\frac{1}{p-1}}$, $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^t \mathbf{T}_{ij} - \mathbf{L}$.

Similarly, for the another formulation:

$$\begin{aligned} \min_{\mathbf{K},\varepsilon_{ij}} \quad & \mathrm{tr}\, \mathbf{L}\mathbf{K} + \frac{C}{2} \sum_{(i,j) \in (\mathcal{S} \cup \mathcal{D})} \varepsilon_{ij}^2 + \frac{G}{p} \mathrm{tr}\, \mathbf{K}^p \\ \text{s.t.} \quad & \forall (i,j) \in (\mathcal{S} \cup \mathcal{D}),\ T_{ij} K_{ij} \geq 1 - \varepsilon_{ij}, \end{aligned} \tag{26}$$

we can derive the corresponding saddle-point minimax problem of (26):

$$\begin{aligned} \max_{\alpha} \min_{\mathbf{K}} \quad & \mathrm{tr} \left(\left(\mathbf{L} - \sum_{(i,j)} \alpha_{ij} \mathbf{T}_{ij} \right) \mathbf{K} \right) - \frac{1}{2C} \sum_{(i,j)} \alpha_{ij}^2 + \sum_{(i,j)} \alpha_{ij} + \frac{G}{p} \mathrm{tr}\, \mathbf{K}^p \\ \text{s.t.} \quad & \mathbf{K} \succeq \mathbf{0},\ \forall (i,j) \in \mathcal{S} \cup \mathcal{D},\ \alpha_{ij} \geq 0. \end{aligned}$$

Again, from the Proposition 2, we observe that the minimizer $\mathbf{K}(\alpha)$ is unique for some fixed α. Together with Lemma 3, we compute the gradient at the point α^t in the same way as in (25) by setting $\mathbf{K} = \left(\frac{1}{G} \mathbf{A}_+\right)^{\frac{1}{p-1}}$, $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^t \mathbf{T}_{ij} - \mathbf{L}$. The alternative optimization algorithm is summarized in Algorithm 1.

4.3.3 ESTIMATING THE RANK OF K

According to Proposition 1 or Proposition 2, we are required to locate the positive spectrums of $\mathbf{A}$, which can be achieved by full eigen-decomposition of $\mathbf{A}$. However, this can be computationally

Algorithm 1 SimpleNPKL with (square) hinge loss.

```
Input: Pairwise constraint matrix $\mathbf{T}$, parameters $C$ and $B$ (or G), $k$
Output: $\alpha$ and $\mathbf{K}$.
1: Construct graph Laplacian $\mathbf{L}$ using $k$ nearest neighbors;
2: Initialize $\alpha^0$;
3: repeat
4:    Set $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^{t-1}\mathbf{T}_{ij} - \mathbf{L}$;
5:    Compute the closed-form solution $\mathbf{K}_t = \left(B/\mathrm{tr}\,\mathbf{A}_+^{p/(p-1)}\right)^{1/p}\mathbf{A}_+^{1/(p-1)}$
      //For the formulation (19), use $\mathbf{K}_t = \left(\mathbf{A}_+/G\right)^{1/(p-1)}$ instead;
6:    Compute the gradient $\nabla J_{ij} = 1 - \mathrm{tr}\,\mathbf{T}_{ij}\mathbf{K}_t - \frac{1}{C}\alpha_{ij}$;
7:    Determine a step size $\eta_t$, update $\alpha_{ij}^t$ using $\alpha_{ij}^t = \left(\alpha_{ij}^{t-1} + \eta_t \nabla J_{ij}\right)_+$;
8: until convergence
```

prohibitive for large scale data sets. Moreover, the computation on the negative eigen-vectors of $\mathbf{A}$ should be avoided. The following proposition (Pataki, 1995) bounds the rank of matrix $\mathbf{K}$ in a general SDP setting.

Proposition 4 *The rank r of $\mathbf{K}$ in the SDP problem:* $\max_{\mathbf{K}\succeq\mathbf{0}}$ *tr* $(\mathbf{A}_0\mathbf{K})$ *with m linear constraints on K, follows the bound* $\begin{pmatrix} r+1 \\ 2 \end{pmatrix} \leq m$.

Moreover, from the empirical study in Alizadeh et al. (1997), the rank r is usually much smaller than this bound. This implies that the full decomposition of matrix $\mathbf{A}_0$ is not required. Our formulation (21) has an additional constraint: $\mathrm{tr}\,\mathbf{K}^2 \leq B$ for $p = 2$. This condition equivalently constraints $\mathrm{tr}\,(\mathbf{K})$, which is a common assumption in SDP problems (Krishnan and Mitchell, 2006). To show this, we have $B \geq \mathrm{tr}\,\mathbf{KK} = \frac{1}{N}\sum_i \lambda_i^2 N \geq \frac{1}{N}(\sum_i \lambda_i \cdot 1)^2 = \frac{1}{N}(\mathrm{tr}\,\mathbf{K})^2$, where the second inequality is resulted from the Cauchy inequality. Hence, we have $\mathrm{tr}\,\mathbf{K} \leq \sqrt{BN}$. Therefore, we can make use of the r estimated from Proposition 4 as a suggestion to estimate the rank of $\mathbf{K}$.

4.3.4 DETERMINING THE CONVERGENCE PROPERTIES

When the η_t is small enough or a universal choice of $\eta_t = O(1/t)$ is used, the whole optimization problem is guaranteed to converge (Boyd and Xiao, 2005). Practically, the value of η plays an important role for the convergence speed. Therefore, it is worth studying the influence of η on the convergence rate, which requires to lower bound the increment of J_{α_t} at each step. We first establish the Lipschitz property of $\nabla J(\alpha)$.

Lemma 5 *Assume we use the formulation of Proposition 2 at each iteration of Algorithm 1, then the gradient of the objective function given by (25) is Lipschitz continuous with Lipschitz constant $L = \frac{m}{G} + \frac{1}{C}$, where $m = |\mathcal{S} \cup \mathcal{D}|$ is the number of nonzeros in $\mathbf{T}$. That is,*

$$\|\nabla J(\alpha_1) - \nabla J(\alpha_2)\|_F \leq \left(\frac{m}{G} + \frac{1}{C}\right)\|\alpha_1 - \alpha_2\|_F.$$

Proof For an α_t, we use $\mathbf{K}_t$ denote the corresponding minimizer of J computed by (14). For a spectral function λ defined on $\mathbb{S}_+$, which is Lipschitz continuous with Lipschitz constant κ, we have

$$\|\lambda(\mathbf{K}_1) - \lambda(\mathbf{K}_2)\|_F \leq \kappa\|\mathbf{K}_1 - \mathbf{K}_2\|_F.$$

For our case, the p.s.d. projection is defined by $\lambda(\mathbf{K}) = \sum_i \max(0, \lambda_i)^2$. The Lipschitz constant κ of this function is 1. Therefore, for any $\mathbf{K}_1$ and $\mathbf{K}_2$ given by (14), we have

$$\begin{aligned}
\|\mathbf{K}_1 - \mathbf{K}_2\|_F &= \|\mathbf{A}_{1+} - \mathbf{A}_{2+}\|_F \\
&\leq \Big\| \frac{1}{G}\Big(\sum_{(i,j)} \alpha_{ij}^{(1)} \mathbf{T}_{ij} - \mathbf{L}\Big) - \frac{1}{G}\Big(\sum_{(i,j)} \alpha_{ij}^{(2)} \mathbf{T}_{ij} - \mathbf{L}\Big) \Big\|_F \\
&= \frac{1}{G} \Big\| \sum_{(i,j)} \big(\alpha_{ij}^{(1)} - \alpha_{ij}^{(2)}\big) \mathbf{T}_{ij} \Big\|_F \\
&\leq \frac{1}{G} \|\alpha_1 - \alpha_2\|_F \|\mathbf{T}\|_F = \frac{\sqrt{m}}{G} \|\alpha_1 - \alpha_2\|_F.
\end{aligned}$$

Consequently, we have,

$$\begin{aligned}
\|\nabla J(\alpha_1) - \nabla J(\alpha_2)\|_F &= \sqrt{\sum_{(i,j)} \Big(\big(1 - \operatorname{tr} \mathbf{T}_{ij}\mathbf{K}_1 - \frac{1}{C}\alpha_{ij}^{(1)}\big) - \big(1 - \operatorname{tr} \mathbf{T}_{ij}\mathbf{K}_2 - \frac{1}{C}\alpha_{ij}^{(2)}\big)\Big)^2} \\
&= \sqrt{\sum_{(i,j)} \Big(\operatorname{tr} \mathbf{T}_{ij}\big(\mathbf{K}_2 - \mathbf{K}_1\big) + \frac{1}{C}\big(\alpha_{ij}^{(2)} - \alpha_{ij}^{(1)}\big)\Big)^2} \\
&\leq \|\mathbf{T}\|_F \|\mathbf{K}_1 - \mathbf{K}_2\|_F + \frac{1}{C}\|\alpha_1 - \alpha_2\|_F \\
&\leq \big(\frac{m}{G} + \frac{1}{C}\big) \|\alpha_1 - \alpha_2\|_F.
\end{aligned}$$

■

With the Lipschitz property of ∇J, we can further show each iteration of Algorithm 1 makes progress towards the optimal solution. Interestingly, we are aware that the proof is very similar to the analysis of indefinite kernel learning, which is proposed very recently by Ying et al. (2010). This result is developed based on non-smooth optimization algorithm of Nesterov (2005). To make the paper complete, we expose the detailed proof in the following proposition.

Proposition 6 *Assume we use the formulation of Proposition 2, and* $\eta \geq \frac{m}{G} + \frac{1}{C}$ *at each iteration of Algorithm 1. The iteration sequence* $\{\alpha_t\}$ *generated in Algorithm 1 satisfy:*

$$J(\alpha_{t+1}) \geq J(\alpha_t) + \frac{\eta}{2}\|\alpha_{t+1} - \alpha_t\|_F^2,$$

and

$$\max_{\alpha} J(\alpha) - J(\alpha_t) \leq \frac{\eta}{2t}\|\alpha_0 - \alpha^*\|_F^2,$$

where α^* *is the optimal solution of* $\max_\alpha J(\alpha)$.

Proof Let $L = \frac{m}{G} + \frac{1}{C}$ abbreviate the Lipschitz constant of $\nabla J(\alpha)$, then we have

$$\begin{aligned}
J(\alpha) - J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle &= \int_{\alpha_t}^{\alpha} \nabla J(\alpha) d\alpha - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle \\
&= \int_0^1 \langle \nabla J(\theta\alpha + (1-\theta)\alpha_t) - \nabla J(\alpha_t), \alpha - \alpha_t \rangle d\theta \\
&\geq -\int_0^1 \|\nabla J(\theta\alpha + (1-\theta)\alpha_t) - \nabla J(\alpha_t)\| \|\alpha - \alpha_t\|_F d\theta \\
&\geq -L \int_0^1 \theta \|\alpha - \alpha_t\|_F^2 d\theta \\
&\geq -\frac{\eta}{2} \|\alpha - \alpha_t\|_F^2.
\end{aligned}$$

Applying this inequality with $\alpha = \alpha_{t+1}$, we have

$$-J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha_{t+1} - \alpha_t \rangle \geq -J(\alpha_{t+1}) - \frac{\eta}{2} \|\alpha_{t+1} - \alpha_t\|_F^2. \tag{27}$$

From step 5 in Algorithm 1, it is easy to verify that

$$\begin{aligned}
\alpha_{t+1} &= \arg\min_{\alpha} \|(\alpha - \alpha_t) - \nabla J(\alpha_t)/\eta\|_F^2 \\
&= \arg\min_{\alpha} -2\langle \alpha - \alpha_t, \nabla J(\alpha_t)/\eta \rangle + \|\alpha - \alpha_t\|_F^2 \\
&= \arg\min_{\alpha} -\nabla J(\alpha_t) - \langle \alpha - \alpha_t, \nabla J(\alpha_t) \rangle + \frac{\eta}{2} \|\alpha - \alpha_t\|_F^2.
\end{aligned} \tag{28}$$

Let $f(\alpha)$ denote the right side of (28). From the first-order optimality condition over α_{t+1}, for any α we have $\langle \nabla f(\alpha_{t+1}), \alpha - \alpha_{t+1} \rangle \geq 0$, that is,

$$-\langle \nabla J(\alpha_t), \alpha - \alpha_{t+1} \rangle \geq \eta \langle \alpha_{t+1} - \alpha_t, \alpha_{t+1} - \alpha \rangle. \tag{29}$$

Adding (27) and (29) together yields that $-J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle \geq -J(\alpha_{t+1}) + \eta \langle \alpha_t - \alpha_{t+1}, \alpha - \alpha_t \rangle + \frac{\eta}{2} \|\alpha_t - \alpha_{t+1}\|_F^2$. Note that $-J$ is convex, $-J(\alpha) \geq -J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle$. Thus we have

$$J(\alpha_{t+1}) \geq J(\alpha) + \eta \langle \alpha_t - \alpha_{t+1}, \alpha - \alpha_t \rangle + \frac{\eta}{2} \|\alpha_t - \alpha_{t+1}\|_F^2.$$

Applying $\alpha = \alpha_t$, we have that

$$J(\alpha_{t+1}) \geq J(\alpha_t) + \frac{\eta}{2} \|\alpha_{t+1} - \alpha_t\|_F^2.$$

Applying $\alpha = \alpha^*$, we have that

$$J(\alpha^*) - J(\alpha_{i+1}) \leq -\eta \langle \alpha_i - \alpha_{i+1}, \alpha^* - \alpha_i \rangle - \frac{\eta}{2} \|\alpha_i - \alpha_{i+1}\|_F^2 = \frac{\eta}{2} \|\alpha^* - \alpha_i\|_F^2 - \frac{\eta}{2} \|\alpha^* - \alpha_{i+1}\|_F^2. \tag{30}$$

Taking summation over i from 0 to $t-1$, we have

$$\sum_{i=0}^{t-1} (J(\alpha^*) - J(\alpha_{i+1})) \leq \frac{\eta}{2} \|\alpha^* - \alpha_0\|_F^2.$$

From (30), we see that the sequence $\{J(\alpha_t)\}$ increase monotonically. Thus we obtain

$$t(J(\alpha^*) - J(\alpha_t)) \leq \frac{\eta}{2}\|\alpha^* - \alpha_0\|_F^2,$$

which completes the proof. ■

4.4 SimpleNPKL with Square Loss

In this subsection, we consider square alignment loss for the SimpleNPKL framework:

$$\begin{aligned}\min_{\mathbf{K},\varepsilon_{ij}} \quad & \mathrm{tr}\,\mathbf{L}\mathbf{K} + \frac{C}{2}\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \varepsilon_{ij}^2 \\ \text{s.t.} \quad & \forall(i,j)\in(\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} = 1-\varepsilon_{ij}, \\ & \mathbf{K}\succeq \mathbf{0}, \mathrm{tr}\,\mathbf{K}^p \leq B.\end{aligned}$$

Here we need not to enforce $\varepsilon \geq 0$. With the standard techniques of Section 4.3, we derive the following min-max problem:

$$\max_{\alpha}\min_{\mathbf{K}} \mathrm{tr}\left(\mathbf{L} - \sum_{ij}\alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K} + \sum_{ij}\alpha_{ij} - \frac{1}{2C}\sum_{ij}\alpha_{ij}^2 \ \ : \ \ \mathbf{K}\succeq \mathbf{0}, \mathrm{tr}\,\mathbf{K}^p \leq B.$$

Therefore, we can compute the gradient of J w.r.t. α:

$$\nabla J_{ij} = 1 - \mathrm{tr}\,\mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}.$$

The whole analysis of Section 4.3 still holds. The difference just lies in the way of computing gradient ∇J. We will show an application of square loss in Section 6.

4.5 SimpleNPKL with Hinge Loss

In this subsection, we consider hinge loss for the SimpleNPKL framework:

$$\begin{aligned}\min_{K,\varepsilon_{ij}} \quad & \mathrm{tr}\,\mathbf{L}\mathbf{K} + C\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \varepsilon_{ij} \\ \text{s.t.} \quad & \forall(i,j)\in(\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} \geq 1-\varepsilon_{ij}, \varepsilon_{ij} \geq 0 \\ & \mathbf{K}\succeq \mathbf{0}, \mathrm{tr}\,\mathbf{K}^p \leq B.\end{aligned}$$

Following the standard techniques of Lagrangian dual, we arrive at the min-max problem:

$$\max_{\alpha}\min_{\mathbf{K}} \mathrm{tr}\left(\mathbf{L} - \sum_{ij}\alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K} + \sum_{ij}\alpha_{ij} \ \ : \ \ \mathbf{K}\succeq \mathbf{0},\ \mathrm{tr}\,\mathbf{K}^p \leq B,\ 0 \leq \alpha_{ij} \leq C.$$

Therefore, we can compute the gradient of J w.r.t. α:

$$\nabla J_{ij} = 1 - \mathrm{tr}\,\mathbf{T}_{ij}\mathbf{K}$$

The whole analysis of Section 4.3 still holds. The difference just lies in the way of computing gradient ∇J. Note that the gradient updating $\alpha = \alpha + \eta\nabla J$ may jump out of the range $[0, C]$. We need to project α into this region at each iteration. We will also show an example of Hinge loss in Section 6.

5. Implementation Issues

In this Section, we discuss some implementation issues that are important to the success of the proposed SimpleNPKL algorithms.

5.1 Building a Sparse Graph Laplacian

Recall that the graph Laplacian $\mathbf{L}$ in (9) is often sparse, in particular, which is usually computed by finding k-nearest neighbors for the purpose of constructing the similarity matrix $\mathbf{S}$. Specifically, an entry $S(i,j) = 1$ if and only if data examples i and j are among each other's k-nearest neighbors; otherwise, it is set to 0. So, there are at most k nonzero entries on each row of $\mathbf{L}$.

A naïve implementation of finding k-nearest neighbors often takes $O(N^2 \log N)$ time. To enforce the data examples i and j are among each other's k-nearest neighbors, one can use B-matching algorithm (Jebara and Shchogolev, 2006) to find the k-nearest neighbors. However, when the data set is very large, the construction of $\mathbf{L}$ becomes non-trivial and very expensive. To address this challenge, we suggest to first construct the *cover tree* structure (Beygelzimer et al., 2006), which takes $O(N \log N)$ time. The similar idea to construct a tree structure for distance metric learning was discussed in Weinberger and Saul (2008). With the aid of this data structure, the batch query of finding k-nearest neighbors on the whole data set can be done within $O(N)$ time. Hence, the graph Laplacian $\mathbf{L}$ can be constructed efficiently for large-scale problems.

5.2 Fast Eigendecomposition by Lanczos Algorithm

Among various existing SDP approaches (Boyd and Vandenberghe, 2004), the interior-point method is often deemed as the most efficient one. However, as discussed in previous subsection, the graph Laplacian $\mathbf{L}$ is often sparse. In addition, the number of pairwise constraints is usually small due to expensive cost of human labels. Therefore, $\mathbf{L} - \sum_{(i,j)} \alpha_{ij} \mathbf{T}_{ij}$ is also sparse. Such sparse structure is not yet exploited in such general algorithms. According to Proposition 1, the time cost of each iteration in Algorithm 1 is dominated by eigen-decomposition. Moreover, from Proposition 4, the rank r of the kernel matrix $\mathbf{K}$ is upper bounded by the number of active constraints. Therefore, we can estimate the rank for sparse eigen-decomposition, which can be solved efficiently using the so-called *Implicitly Restarted Lanczos Algorithm* (IRLA) (Lehoucq et al., 1998). Its computational cost is dominated by matrix-vector multiplication. Specifically, the time cost of IRLA is linear with the number of non-zeros in $\mathbf{A}$. Assume k nearest neighbors are used to construct the graph Laplacian $\mathbf{L}$, then the number of non-zeros in $\mathbf{A}$ is at most $Nk + m$, where m is the number of nonzeros in $\mathbf{T}$, and $\mathbf{A}$ is very sparse. Moreover, the time cost of computing gradient is $O(m)$. Therefore, the time complexity per iteration of SimpleNPKL is $O(Nk + m)$.

5.3 Active Constraint Selection

As shown in Algorithm 1, the computational cost of the update procedure is highly depends on the number of pairwise constraints. However, some less informative constraints often do not contribute much to the learning of the kernel matrix $\mathbf{K}$, and fitting some noisy pairwise constraints may also lead to the poor generalization. Moreover, as discussed in Section 4.3.3, the rank of $\mathbf{K}$ is lower when there are fewer active constraints in (22). Therefore, selecting pairwise constraints for SimpleNPKL may improve both the efficiency and the generalization of the NPK learning.

To speed up the eigen-decomposition process, instead of engaging all pairwise constraints, we propose to sample a subset of T_{ij}'s for SimpleNPKL. Instead of acquiring class label information for kernel learning; here, we consider another simple active constraint selection scheme. Recall that a general principle in active learning is to request the label of the data points that are most uncertain for their predictions. Following this idea, we adopt the margin criterion to measure the uncertainty of the prediction value on a data point. In particular, given a data point $\mathbf{x}_i$, assume that we have the prediction function in the form:

$$f(\mathbf{x}_i) = \sum_j y_j K(\mathbf{x}_i, \mathbf{x}_j).$$

We can use $|y_i f(\mathbf{x}_i)|$ to measure the uncertainty of prediction, where $y_i \in \{-1, +1\}$ is the class label of data point $\mathbf{x}_i$. As a result, for a data point $\mathbf{x}_i$, we choose the constraints involving point i:

$$\begin{aligned} i^* &= \arg\min_i \left| \frac{1}{l_i} \sum_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right| \\ &= \arg\min_i \left| \frac{1}{l_i} \sum_{j, T_{ij} \neq 0} T_{ij} K(\mathbf{x}_i, \mathbf{x}_j) \right|, \end{aligned}$$

where we deem T_{ij} as an entry of yy', and $l_i = |\{j : (i,j) \in \mathcal{S} \cup \mathcal{D}\}, T_{ij} \neq 0\}|$ is used as a normalization of the margin value. Based on the above formula, we choose a subset of k data points $\mathcal{S}_k$ that are most uncertain according to the margin measure. Then, we choose all the T_{ij}'s that involve any point $i \in \mathcal{S}_k$ as pairwise constraints to form a new set of constraints. Finally, we run SimpleNPKL based on this new set of constraints.

5.4 Low Rank Approximation of K

Since the rank r of $\mathbf{K}$ often satisfies $r < n$, we may express $\mathbf{K}$ as $\mathbf{K} = \mathbf{VEV}'$, where the columns of $\mathbf{V}_{n \times r}$ are eigenvectors of $\mathbf{K}$. If we fix the base $\mathbf{V}$, the number of variables is reduced from n^2 to r^2. With this approximation scheme, the $\mathbf{A}$ matrix in Algorithm 1 becomes $\mathbf{A} = \mathbf{V}'(\mathbf{L} - \sum \alpha_{ij} \mathbf{T}_{ij})\mathbf{V}$. Note $\mathbf{V}'\mathbf{LV}$ can be pre-computed and $\mathbf{V}' \sum \alpha_{ij} \mathbf{T}_{ij} \mathbf{V}$ can be computed efficiently by virtue of the sparseness. Therefore, SimpleNPKL can be significantly faster with this approximation.

6. Applications of SimpleNPKL

In this Section, we extend the proposed SimpleNPKL technique to other similar machine learning problems where the goal of the optimization is to find an optimal matrix such that its inner product with another matrix is maximized or minimized. In particular, we consider the data embedding problems, where the goal is to find a new data representation that preserves some similarity/distance constraints between pairs of data points. These problems typically can be implemented by constraining the alignment of the target kernel matrix to some prior affinity or distance structures. As a result, the kernel matrix $\mathbf{K} = \mathbf{V}'\mathbf{V}$ implies a data embedding with a natural interpretation, in which the column vector of $\mathbf{V}$ corresponds to the new data representation. We discuss several important data embedding methods below.

6.1 Colored Maximum Variance Unfolding

Colored MVU (Song et al., 2008) is an improvement of Maximum Variance Unfolding (MVU) (Weinberger et al., 2004), which produces a low-dimensional representation of the data by maximizing the trace of a matrix **K** subject to some positive definiteness, centering and distance-preserving constraints, that is:

$$\min_{\mathbf{K}} \quad -\mathrm{tr}\,\mathbf{K} \;\;:\;\; \mathbf{K} \succeq \mathbf{0}, \sum_{ij} \mathbf{K}_{ij} = 0,\; \mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = D_{ij},\; \forall (i,j) \in \mathcal{N}.$$

where $\mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = K_{ii} + K_{jj} - 2K_{ij}$ is the square distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

CMVU interprets MVU from a statistical perspective. It maximizes the dependence between the domain of input pattern **x** and the domain of label y, which is measured by the *Hilbert- Schmidt Independence Criterion* (Gretton et al., 2005; Song et al., 2008). Here we introduce slack variables ξ to measure the violations of distance constraints and penalize the corresponding square loss. Consequently the optimization task of colored MVU is reformulated as:

$$\min_{\mathbf{K},\xi} \quad -\mathrm{tr}\,\mathbf{HKHY} + \frac{C}{2}\sum \xi_{ij}^2, \;\;:\;\; \mathbf{K} \succeq \mathbf{0},\; \mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = D_{ij} - \xi_{ij},\; \forall (i,j) \in \mathcal{N}$$

where $H_{ij} = \delta_{ij} - N^{-1}$ such that **HKH** centers **K**, $\mathbf{Y} = \mathbf{y}\mathbf{y}'$ is the kernel matrix over labels. Apparently this belongs to an SDP problem.

Following the SimpleNPKL algorithms, we derive the minimax optimization problem by introducing dual variables for the inequality constraints:

$$\max_{\alpha} \min_{\mathbf{K}} \quad \mathrm{tr}\left(-\mathbf{HYH} - \sum_{ij} \alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K} + \sum_{ij} \alpha_{ij} D_{ij} - \frac{1}{2C}\sum_{ij} \alpha_{ij}^2 \;\;:\;\; \mathbf{K} \succeq \mathbf{0},\; \mathrm{tr}\,\mathbf{KK} \le B. \tag{31}$$

By substituting the following results

$$\mathbf{A} = \mathbf{HYH} + \sum_{ij} \alpha_{ij}\mathbf{T}_{ij} \;\text{ and }\; \nabla J_{ij}^t = D_{ij} - \mathrm{tr}\,\mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}^t$$

back into Algorithm 1, the problem of (31) can be solved immediately.

6.2 Minimum Volume Embedding

Minimum Volume Embedding (MVE) is another improvement of MVU (Shaw and Jebara, 2007). One limitation of MVU is that it simply maximizes the trace of **K**, which may result in the solution that engages considerably more dimensions than necessity. To address this problem, Shaw and Jebara (2007) proposed to grow the top few eigenvalues of **K** while shrinking the remaining ones. In particular, let $\mathbf{K} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i'$, $\lambda_1 \ge, \ldots, \ge \lambda_n$, and $\mathbf{K}_0 = \sum_{i=1}^d \mathbf{v}_i\mathbf{v}_i' - \sum_{i=d+1}^n \mathbf{v}_i\mathbf{v}_i'$. When the intrinsic dimensionality d is available, MVE formulates the data embedding problem as follows:

$$\min_{\mathbf{K}} \quad -\mathrm{tr}\,\mathbf{KK}_0 \;\;:\;\; \text{the same set of constraints of MVU.} \tag{32}$$

After obtaining the solution $\mathbf{K}^t$ at each step, MVE proceeds by substituting $\mathbf{K}_0 = \mathbf{K}^t$ back to the optimization of (32) and repeatedly solving the optimization. Hence, MVE improves MVU by decreasing the energy of the small eigen components of **K**. To find the solution, every $\mathbf{K}^t$ is computed by applying a general SDP solver in Shaw and Jebara (2007).

To speed up the solution, following the similar derivation in the above CMVU, we can solve (32) by eigen-decomposition in an iterative manner. Specifically, we make the following modifications:

$$A = \mathbf{K}_0 + \sum_{ij} \alpha_{ij} \mathbf{T}_{ij} \text{ and } \nabla J_{ij}^t = D_{ij} - \operatorname{tr} \mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}^t$$

By substitute the above results back into Algorithm 1, we can solve the MVE problem efficiently.

6.3 Structure Preserving Embedding

Structure Preserving Embedding (SPE) (Shaw and Jebara, 2009) is a machine learning technique that embeds graphs in low-dimensional Euclidean space such that the embedding preserves the global topological properties of the input graph. Suppose we have a connectivity matrix $\mathbf{W}$, where $W_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected and $W_{ij} = 0$ otherwise. SPE learns a kernel matrix $\mathbf{K}$ such that the similarity $\operatorname{tr} \mathbf{KW}$ is maximized while the global topological properties of the input graph are preserved. More formally, the SPE problem is formulated into the following SDP optimization:

$$\min_{\mathbf{K}} \quad -\operatorname{tr} \mathbf{KW} + C\xi \ : \ D_{ij} > (1 - W_{ij}) \max_m (W_{im} D_{im}) - \xi, \ \xi \geq 0$$

where $D_{ij} = K_{ii} + K_{jj} - 2K_{ij} = \operatorname{tr} \mathbf{KT}_{ij}$ is the squared distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Let $[n] = \{1, \dots, n\}$ and $\mathcal{N}_i$ denote the set of indices of points which are among the nearest neighbors of $\mathbf{x}_i$. Then for each point $\mathbf{x}_i$, SPE essentially generates $(n - |\mathcal{N}_i|) \times |\mathcal{N}_i|$ constraints:

$$\operatorname{tr} \mathbf{KT}_{ij} > \operatorname{tr} \mathbf{KT}_{ik} - \xi, \ \ \forall i \in [n], j \in [n] - \mathcal{N}_i, k \in \mathcal{N}_i.$$

In order to speed up the SPE algorithm, we apply the SimpleNPKL technique to turn the SPE optimization into the following minimax optimization problem:

$$\max_{\alpha} \min_{\mathbf{K}} \quad \operatorname{tr} \left(\sum_i \sum_{k \in \mathcal{N}_i} \sum_{j \notin \mathcal{N}_i} \alpha_{ijk} (\mathbf{T}_{ik} - \mathbf{T}_{ij}) - \mathbf{W} \right) \mathbf{K} \ : \ \mathbf{K} \succeq \mathbf{0}, \operatorname{tr} \mathbf{KK} \leq B, \sum \alpha_{ijk} \in [0, C].$$

Similarly, we can derive the following results:

$$\mathbf{A} = \mathbf{W} - \sum_{ijk} \alpha_{ijk} (\mathbf{T}_{ik} - \mathbf{T}_{ij}) \text{ and } \nabla J_{ijk}^t = \operatorname{tr} \mathbf{K} (\mathbf{T}_{ik} - \mathbf{T}_{ij}).$$

Substituting them back into Algorithm 1 leads to an efficient solution for the SPE problem.

7. Experiments

In this Section, we conduct extensive experiments to examine the efficacy and efficiency of the proposed SimpleNPKL algorithms.

7.1 Experimental Setup

We examine both efficacy and efficiency of the proposed SimpleNPKL using side information to learn a kernel matrix for kernel k-means clustering. As shown in Hoi et al. (2007), the learned kernel matrix of the Non-Parametric Kernel Learning (NPKL) outperforms other kernel learning methods in the task of clustering using side information. For simplicity, we only compare our proposed SimpleNPKL algorithms with the NPKL method in Hoi et al. (2007) for kernel k-means clustering. The results of k-means clustering and constrained k-means clustering using Euclidean metric are also reported as the performance of the baseline methods. The abbreviations of different approaches are described as follows:

- ***k*-means**: k-means clustering using Euclidean metric;
- **c*k*-means**: The constrained k-means clustering algorithm using Euclidean metric and side information;
- **SimpleNPKL+LL**: The proposed SimpleNPKL with linear loss defined in (12);
- **SimpleNPKL+SHL**: The proposed SimpleNPKL with squared hinge loss defined in (21);
- **NPKL+LL**: NPKL in (10) using linear loss;
- **NPKL+HL**: NPKL in (10) using hinge loss.

To construct the graph Laplacian matrix $\mathbf{L}$ in NPKL, we adopt the cover tree data structure.[2] The sparse eigen-decomposition used in SimpleNPKL is implemented by the popular *Arpack* toolkit.[3] We also adopt the standard SDP solver, SDPT3,[4] as the baseline solution for NPKL. The pair-wise constraint is assigned for randomly generated pairs of points according to their ground truth labels. The number of constraints is controlled by the resulted amount of connected components as defined in previous studies (Xing et al., 2003; Hoi et al., 2007). Note that typically the larger the number of constraints, the smaller the number of connected components.

Several parameters are involved in both NPKL and SimpleNPKL. Their notation and settings are given as follows:

- k : The number of nearest neighbors for constructing the graph Laplacian matrix $\mathbf{L}$, we set it to 5 for small data sets in Table 1, and 50 for Adult database in Table 6;
- r : The ratio of the number of connected components compared with the data set size N. In our experiments, we set $r \approx 70\%N$ which follows the setting of Hoi et al. (2007);
- B : The parameter that controls the capacity of the learned kernel in (11). We fix $B = N$ for the adult data sets and fix $B = 1$ for the data sets in Table 1 and;
- C : The regularization parameter for the loss term in NPKL and SimpleNPKL. We fix $C = 1$ for the adult data sets and several constant values in the range (0, 1] for the data sets in Table 1.

In our experiments, all clustering results were obtained by averaging the results from 20 different random repetitions. All experiments were conducted on a 32bit Windows PC with 3.4GHz CPU and 3GB RAM.

7.2 Comparisons on Benchmark Data Sets

To evaluate the clustering performance, we adopt the clustering accuracy used in Hoi et al. (2007):

$$\text{Cluster Accuracy} = \sum_{i>j} \frac{\mathbf{1}\{c_i = c_j\} = \mathbf{1}\{\hat{c}_i = \hat{c}_j\}}{0.5n(n-1)}.$$

2. The cover tree data structure is described at `http://hunch.net/~jl/projects/cover_tree/cover_tree.html`.
3. The *Arpack* toolkit can be found at `http://www.caam.rice.edu/software/ARPACK/`.
4. SDPT3 can be found at `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`.

Data Set	#Classes	#Instances	#Features
Chessboard	2	100	2
Glass	6	214	9
Heart	2	270	13
Iris	3	150	4
Protein	6	116	20
Sonar	2	208	60
Soybean	4	47	35
Spiral	2	100	3
Wine	3	178	12

Table 1: The statistics of the data sets used in our experiments.

This metric measures the percentage of data pairs that are correctly clustered together. We compare the proposed SimpleNPKL algorithms with NPKL on the nine data sets from UCI machine learning repositories,[5] as summarized in Table 1. The same data sets were also adopted in the NPKL study of Hoi et al. (2007).

The clustering accuracy and CPU time cost (the clustering time was excluded) of different NPKL methods are reported in Table 2 and 3. As can be observed from Table 2, all NPKL methods outperform the baseline k-means clustering and the constrained k-means clustering methods, which use Euclidean metric for k-means clustering. The proposed SimpleNPKL with square hinge loss produces very competitive clustering performance to the results of NPKL with hinge loss (as reported in Hoi et al., 2007). SimpleNPKL with square hinge loss and NPKL with hinge loss often perform better than the NPKL methods using linear loss.

For the CPU time cost, the time costs of SimpleNPKL and NPKL using linear loss are usually lower than those of their counterparts with (square) hinge loss. Regarding the efficiency evaluation in Table 3, our SimpleNPKL with linear loss or squared hinge loss is about 5 to 10 times faster than NPKL using the SDPT3 solver. For some cases of linear loss, SimpleNPKL can be even 100 times faster.

Recall that our key Proposition 1 provides a closed-form solution to the learned kernel matrix $\mathbf{K}$ for $p \geq 1$, in which the capacity parameter B can be omitted for SimpleNPKL+linear loss. To show the influence of the capacity parameter B for SimpleNPKL + square hinge loss, we present some results in Table 4 with a fixed $p = 2$. To clearly show the influence on convergence, we present the number of iterations instead of elapsed CPU time. We observe that SimpleNPKL + square hinge loss is not sensitive to B on the both *Iris* and *Protein* data sets. It even produces the identical accuracy on the *Iris* data set for $B \in \{2.5, 3, 3.5, 4\}$. However, it affects the number of steps it takes to converge. Similar phenomena can be observed on other data sets.

We also study the clustering performance of varying p in Table 5. We fixed $B = 1$ in this experiment. From Table 5, we can observe that SimpleNPKL+square hinge loss produces the best clustering accuracy for the *Iris* data set when $p = 4$, but the improvement is not significant comparing with $p = 2$. For the *Protein* data set, our algorithm achieves the best results when $p = 2$. In general, when $p < 2$, the clustering performance drops significantly.

5. The data sets are available at `http://archive.ics.uci.edu/ml/`.

Data Set	*k*-means	c*k*-means	NPKL		SimpleNPKL	
			LL	HL	LL	SHL
Chessboard	49.8 ±0.2	50.1±0.3	**61.1± 6.9**	56.3± 6.1	60.2± 0.0	58.8± 0.8
Glass	69.7 ±1.9	69.2±1.7	74.4± 3.7	**79.1± 4.9**	73.0± 2.5	73.5± 2.9
Heart	51.5 ±0.1	52.3±3.7	86.0± 0.3	86.2± 0.0	86.8± 0.0	**89.4± 0.1**
Iris	84.5 ±6.5	89.4±8.5	96.0± 6.1	**97.4± 0.0**	**97.4± 0.0**	**97.4± 0.0**
Protein	76.2 ±2.0	80.7±3.1	78.2± 3.2	**86.4± 3.8**	81.8± 1.8	75.9± 2.0
Sonar	50.2 ±0.1	50.8±0.2	76.8± 0.3	64.5± 6.8	70.2± 10	**78.0± 0.0**
Soybean	82.1 ±6.1	83.8±8.3	90.2± 7.7	**100.0± 0.0**	95.3± 5.1	95.4± 4.9
Spiral	50.1 ±0.6	50.6±1.3	86.5± 0.0	**94.1± 0.0**	92.2± 0.0	**94.1± 0.0**
Wine	71.2 ±1.2	76.1±2.8	78.1± 1.7	**85.5± 5.3**	83.7± 4.8	85.0± 2.6

Table 2: Clustering accuracy of SimpleNPKL, compared with the results of NPKL in (10) using a standard SDP solver, and *k*-means.

Data Set	NPKL		SimpleNPKL		Speedup
	LL	HL	LL	SHL	
Chessboard	1.38±0.07	5.23±0.06	**0.05±0.00**	0.13±0.00	27.6
Glass	1.85±0.04	32.36±0.37	**0.11±0.00**	2.95±0.00	16.8
Heart	2.64±0.10	63.84±0.68	**0.17±0.01**	13.15±0.08	15.5
Iris	1.36±0.03	1.65±0.04	**0.04±0.00**	3.45±0.01	34.0
Protein	1.80±0.06	8.16±0.11	**0.05±0.00**	1.32±0.00	36.0
Sonar	1.77±0.08	30.38±0.24	**0.11±0.00**	3.91±0.03	16.1
Soybean	1.51±0.05	3.25±0.04	**0.01±0.00**	0.16±0.00	151.0
Spiral	1.78±0.10	6.23±0.08	**0.05±0.00**	1.95±0.00	36.6
Wine	2.54±0.04	30.91±1.30	**0.09±0.00**	1.53±0.01	28.2

Table 3: CPU time of SimpleNPKL, compared with the results of NPKL in (10) using a standard SDP solver. (The best results are in bold and the last "Speedup" column is listed only for the linear loss case.)

7.3 Scalability Study on Adult Data Set

In this Section, we evaluate our SimpleNPKL algorithms on another larger data set to examine the efficiency and scalability. We adopt the *Adult* database, which is available at the website of LibSVM.[6] The database has a series of partitions: A1a, A2a, $\cdots$, A5a (see Table 6). Since the training time complexity of NPKL using standard SDP solvers is $O(N^{6.5})$, which cannot be applied on this database for comparison. We only report the results of both *k*-means and constrained *k*-means clustering as the baseline comparison.

Table 7 shows the clustering performance and CPU time cost (the clustering time was excluded) of SimpleNPKL on the *Adult* database. From the results, we can draw several observations. First of all, we can see that by learning better kernels from pairwise constraints, both SimpleNPKL algorithms produce better clustering performance than that of *k*-means clustering and constrained

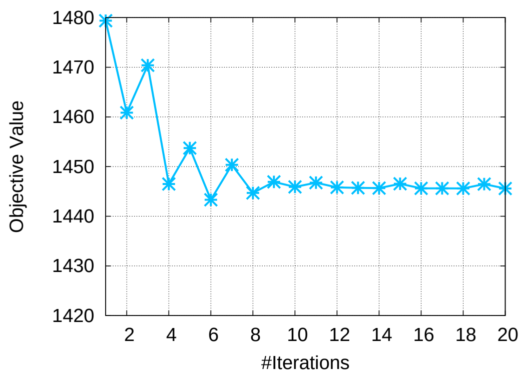
6. LibSVM can be found at `http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/`.

DataSet	B	1	1.5	2	2.5	3	3.5	4
Iris	Accur.(%)	94.8±0.0	94.8±0.0	95.2±0.4	**95.7±0.0**	95.7±0.0	95.7±0.0	95.7±0.0
	#Iterations	11	13	14	**10**	10	31	26
Protein	Accur.(%)	**74.5±0.8**	73.6±1.6	74.4±0.8	74.3±0.9	74.1±1.0	73.7±1.1	73.7±1.0
	#Iterations	51	32	51	**11**	14	27	19

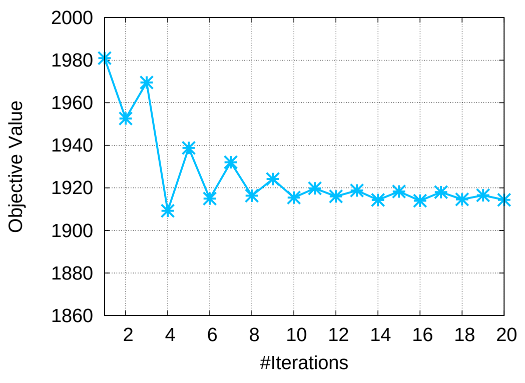
Table 4: Results of varying capacity parameter B with fixed $p = 2$ and $C = 1$ on *Iris* and *protein* data sets.

Data Set	p	1	1.5	2	2.5	3	3.5	4
Iris	Accur.(%)	61.6±3.5	58.6±4.0	94.8±0.0	94.8±0.0	95.1±0.4	94.8±0.0	**95.6±0.2**
	#Iterations	51	**6**	11	9	19	10	9
Protein	Accur.(%)	72.3±1.3	72.8±2.2	**74.5±0.8**	73.6±1.5	73.6±1.6	73.5±1.6	73.5±1.6
	#Iterations	32	35	51	40	**11**	**11**	21

Table 5: Results of varying p in the p-norm regularization over $\mathbf{K}$ with fixed $B = 1$ and $C = 1$ on *Iris* and *protein* data sets.



(a) A1a



(b) A2a

Figure 1: Convergence of SimpleNPKL using square hinge loss on *A1a* and *A2a*. The parameters are $C = 1$, $B = N$.

k-means clustering methods using Euclidean metric. Further, comparing the two algorithms themselves, in terms of clustering accuracy, they perform quite comparably, in which SimpleNPKL+SHL outperforms slightly. However, in terms of CPU time cost, SimpleNPKL+LL with linear loss is considerably lower than SimpleNPKL+SHL using square hinge loss.

We also plot the objective value $J(K,\alpha)$ of SimpleNPKL on two data sets *A1a* and *A2a* in Figure 1. We observe that SimpleNPKL with square hinge loss often converges quickly within 10 iterations. Similar results can be observed from the other data sets.

Data Set†	A1a	A2a	A3a	A4a	A5a
#Instances	1,605	2,265	3,185	4,781	6,414

†: #Classes=2, #Features=123

Table 6: The statistics of the *Adult* database.

Data Set	#Constraints	Accuracy(%)				CPU Time(s)	
		k-means	c*k*-means	SimpleNPKL		SimpleNPKL	
				LL	SHL	LL	SHL
A1a	4,104	56.4±3.5	59.0±2.3	**61.4±1.7**	60.7±2.7	**8.5**	322.9
A2a	5,443	57.3±3.6	60.2±0.1	61.1±1.3	**61.4±1.2**	**15.3**	637.2
A3a	7,773	57.8±3.5	59.2±3.0	61.1±1.7	**61.5±2.0**	**28.8**	1,160.8
A4a	12,465	58.8±1.6	59.3±3.9	**61.6±1.3**	61.4±1.5	**66.3**	2,341.3
A5a	16,161	57.7±3.1	59.8±2.2	60.8±3.1	**61.9±1.7**	**79.6**	3,692.1

Table 7: Evaluation results on *Adult* data set. (The best results are in bold.)

7.4 Comparisons on Constraint Selection

In this Section, we study the active constraint selection scheme for SimpleNPKL. Figure 2 shows the clustering performance of active constraint selection by the approach described in Section 5.3.

Several observations can be drawn from the results: 1) Comparing with the original approach using all constraints, the computation time is reduced by choosing a small amount of pairwise constraints. This is because the Lanczos algorithm can perform the sparse eigen-decomposition faster on a sparse matrix with fewer nonzero entries; 2) Though the active constraint selection costs more time than random selection, the former usually achieves better clustering (accuracy) performance than the latter with the same amount of constraints; 3) Using the proposed active constraint selection method to choose about half of the pairwise constraints for SimpleNPKL can often produce comparable or even better clustering performance than that using all constraints.

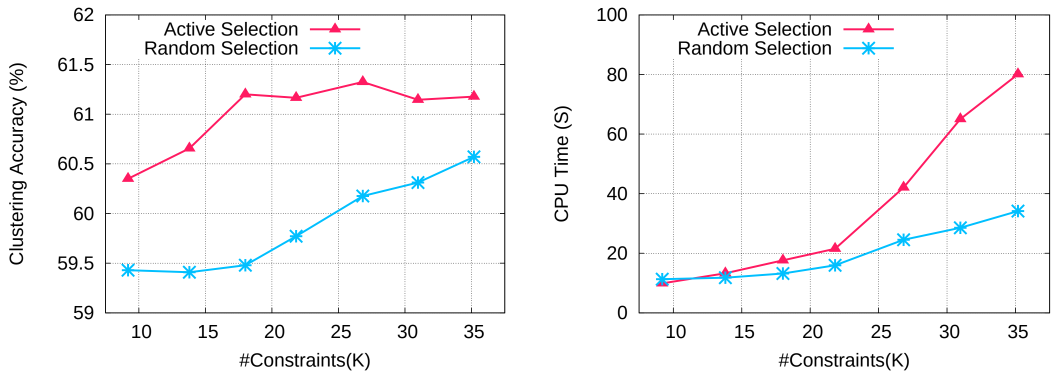
7.5 Evaluations on Data Embedding Applications

In this Section, we evaluate the performance of the proposed SimpleNPKL algorithms with applications to speed up three data embedding techniques, that is, CMVU, MVE, and SPE, respectively. Our goal is to show that SimpleNPKL is capable of producing similar empirical results to the baseline counterpart with significant efficiency gain. All the data sets are publicly available in the UCI machine learning repository. In all the experiments, we simply fix $C = 1$ for all the three methods, and set $B = m \times N$, $m \in \{0.1, 1, 2, 10\}$.

7.5.1 COLORED MAXIMUM VARIANCE UNFOLDING

The first experiment is to examine the efficiency by applying the proposed SimpleNPKL technique to solve the CMVU problem. In particular, we examine the CMVU task for learning low-dimensional embedding on three data sets which were used in Song et al. (2008). Two approaches are compared:

- **CMVU**: An approximate efficient method employed by Song et al. (2008). Suppose $\mathbf{K} = \mathbf{VAV}'$, where $\mathbf{V}$ (of size $n \times d$, $d < n$) is fixed to be the bottom d eigenvectors of the graph



(a) Clustering Accuracy

(b) CPU Time

Figure 2: Comparisons of clustering accuracy and CPU time by active constraint selection and random selection (constraint selection time is included) on A1a with parameters: $B = N, C = 1, k = 20, r = 0.6$. Using all 3.9$K$ constraints directly, the accuracy is 60.8 ± 2.9 and the CPU time is 81.6 seconds.

Laplacian of the neighborhood graph via $\mathcal{N}$. Thus the number of variables is reduced from n^2 to d^2.

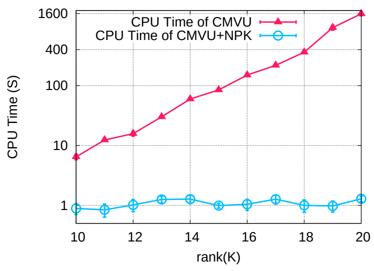
- **CMVU+NPKL**: Our SimpleNPKL method introduced in Section 6.1. Unlike the above CMVU algorithm by approximation, our method is able to obtain the global optimal solution using the SimpleNPKL scheme without approximation.

Figure 3, 4 and 5 show the experimental results of visualizing the embedding results in a 2D space and the CPU time cost of CMVU. The time costs of CMVU+NPKL were also indicated in the captions of those figures. As we can observe from the visualization results, the proposed CMVU+NPKL is able to produce comparable embedding results as those by the original CMVU in most cases. Further, by examining the time cost, we found that the time cost of CMVU increases with dimensionality d exponentially due to the intensive computational cost of solving the SDP problem. In contrast, the proposed CMVU+NPKL is able to find the global optima efficiently, which is much faster than CMVU when d is large. Although CMVU could be faster than CMVU+NPKL for very small d values, it is important to note that the optimal d value is often unknown for many applications. The proposed CMVU+NPKL approach can efficiently and directly resolve the CMVU problem without soliciting the approximation step.

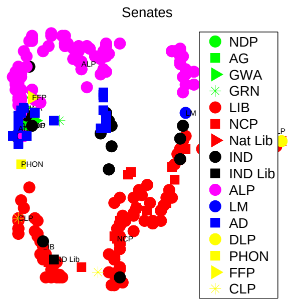
7.5.2 MINIMUM VOLUME EMBEDDING AND STRUCTURE PRESERVING EMBEDDING

This experiment is to examine the embedding performance of the SimpleNPKL technique with applications to MVE (Shaw and Jebara, 2007) and SPE (Shaw and Jebara, 2009) tasks. In particular, five approaches are compared:

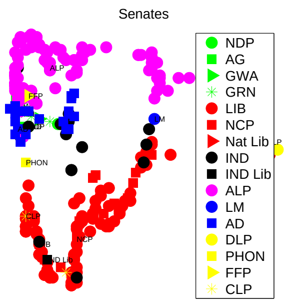
- **KPCA**: The classical Kernel Principle Component Analysis algorithm;
- **MVE**: The algorithm summarized in Table 1 in Shaw and Jebara (2007). Pay attention to the SDP solver in Step 5 and 6, which is the key for the success of MVE.
- **MVE+NPKL**: The embedding algorithm based on our SimpleNPKL algorithm. Refer to Section 6.2 for detailed discussion.



(a) Time cost of CMVU with the rank.

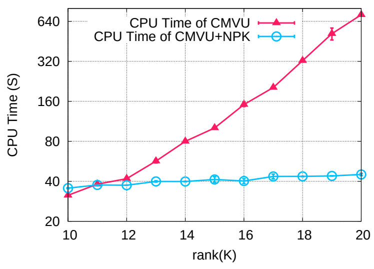


(b) Embedding of CMVU.

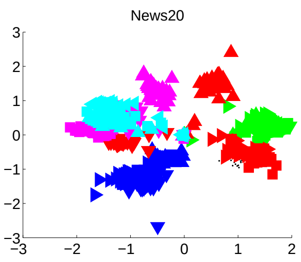


(c) Embedding of CMVU+NPKL.

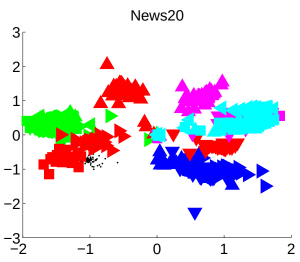
Figure 3: Comparisons of CMVU and CMVU+NPKL on *senate* data set. **Time cost of CMVU+NPK is** 1.50 ± 0.06 **seconds.**



(a) Time cost of CMVU with the rank.

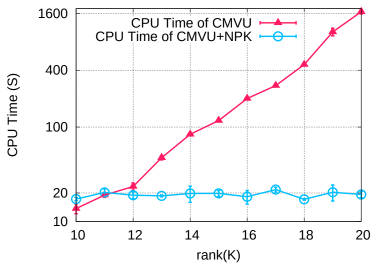


(b) Embedding of CMVU.

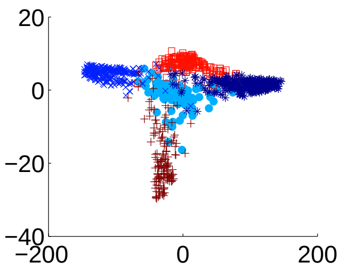


(c) Embedding of CMVU+NPKL.

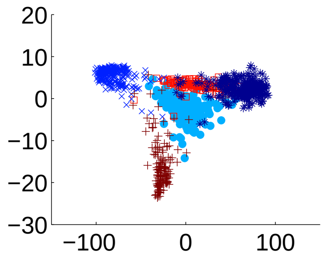
Figure 4: Comparisons of CMVU and CMVU+NPK on *news20* data set. **Time cost of CMVU+NPKL is** 120.4 ± 1.7 **seconds.**



(a) Time cost of CMVU with the rank.



(b) Embedding of CMVU.



(c) Embedding of CMVU+NPKL.

Figure 5: Comparisons of CMVU and CMVU+NPKL on *usps* data set. **Time cost of CMVU+NPKL is** 28.95 ± 1.8 **seconds.**

- **SPE**: The algorithm summarized in Table 1 of Shaw and Jebara (2009).
- **SPE+NPKL**: The embedding algorithm based on the proposed SimpleNPKL algorithm. Refer to Section 6.3;

To examine the embedding results quantitatively, we follow the previous studies (Shaw and Jebara, 2007, 2009) to evaluate the classification performance on the embedding data by performing k-nearest neighbor classification. Similar to the settings in Shaw and Jebara (2009), we randomly choose 100 points from the largest two classes for each data set, and then divide the data examples into training/validation/test sets at the ratios of 60:20:20. The validation set is used to find the best parameter of k for k-NN classification.

Table 8 shows the comparison of the classification results by five different approaches. From the results, we can see that the two proposed algorithms, MVE+NPKL and SPE+NPKL, are generally able to achieve the competitive classification results that are comparable to the other two original algorithms using a standard SDP solver. Among all compared algorithms, MVE+NPKL tends to achieve slightly better performance than the other approaches. All these results show that the proposed algorithms are effective to produce comparable embedding performance.

Data Set	KPCA	MVE	MVE+NPKL	SPE	SPE+NPKL
Wine	90.5 ±5.6	**91.9 ±6.6**	90.9±5.8	75.2 ±0.09	87.1 ±7.9
Ionosphere	79.8±7.3	**86.3 ±7.3**	84.2±8.5	80.4 ±10.4	83.6 ±7.8
Heart	**65.6 ±8.4**	62.4 ±9.8	62.9 ±9.8	54.9 ±10.2	62.2 ±11.1
Sonar	58.2 ±12.4	59.2 ±10.2	**59.8 ±12.2**	57.4 ±11.1	59.4 ±11.4
Glass	70.7 ±9.8	73.5 ±7.8	**74.5 ±10.4**	61.7 ±9.7	69.4 ±8.7
Spiral	98.7 ±2.4	69.1 ±9.8	**98.8 ±2.4**	76.7 ±0.07	82.9 ±8.4
Australian	63.2 ±9.8	61.3 ±8.2	**63.8 ±9.3**	60.1 ± 0.10	59.5 ±10.1
Breast cancer	91.9 ±5.4	92.9 ±4.6	92.4 ±5.8	93.4 ±0.07	**94.4 ±5.5**

Table 8: k-NN classification accuracy on the 2D embedded results. (The best results are bolded.)

Next we compare the computational cost of the proposed algorithms against their original methods, respectively. Table 9 shows the summary of average CPU time cost of the compared algorithms. From the results, it is clear to see that the two proposed algorithms, MVE+NPKL and SPE+NPKL, are significantly more efficient than the other two original algorithms, respectively. By comparing MVE and MVE+NPKL, we found that MVE+NPKL achieves about 10 to 30 times speedups over the original MVE algorithm; the speedup values are even more significant for the SPE problem, where the proposed SPE+NPKL algorithm attains about 50 to 90 times speedup over the original SPE algorithm. These promising results again validate the proposed SimpleNPKL is effective for improving the efficiency and scalability of the three data embedding tasks.

To further illustrate the scalability of SPE+NPKL, we propose to solve a real-world embedding task on a large data set. In particular, we crawled a Flickr[7] data set, which consists of 3,660 Flickr user profiles and a collection of 3.7 million photos uploaded by these users. Each photo was annotated with a set of textual tags by users. Accordingly the photos for a particular Flickr user are described by tiling these tags. In total, our data set has 359,832 tags and 93,692 unique tags. Each Flickr user has a contact list, which is a collection of Flickr users who may share similar tastes / interests in their photo sharing. In our data set, every user has 19.1 contacts on average. We thus set $|\mathcal{N}|$ to 20 in both MVE and SPE. Moreover, there are $97,212$ interest groups, and each Flickr user could belong to one or more interest groups. We compute *tf-idf* weight for the tags to represent a Flickr user (here the document frequency for a tag is actually the number of users annotated with

7. Flickr can be found at `http://www.flickr.com/`.

Data Set	MVE	MVE+NPKL	SpeedUp	SPE	SPE+NPKL	SpeedUp
Wine	2.92 $\pm$0.06	**0.34 $\pm$0.04**	8.5	47.72 $\pm$0.29	0.51 $\pm$0.01	93.6
Ionosphere	16.98 $\pm$0.16	1.14 $\pm$0.01	14.9	30.07 $\pm$1.25	**0.60 $\pm$0.01**	50.1
Heart	9.64 $\pm$0.00	**0.38 $\pm$0.02**	25.3	48.18 $\pm$0.31	0.51 $\pm$0.11	94.5
Sonar	7.50 $\pm$0.13	**0.46 $\pm$0.01**	16.3	30.40 $\pm$1.16	0.61 $\pm$0.02	49.8
Glass	11.08 $\pm$0.26	**0.39 $\pm$0.01**	28.2	29.10 $\pm$0.12	0.53 $\pm$0.01	54.9
Spiral	18.28 $\pm$0.28	**0.46 $\pm$0.00**	39.7	47.91 $\pm$0.91	0.48 $\pm$0.01	99.8
Australian	4.61 $\pm$0.03	**0.30 $\pm$0.02**	15.4	28.94 $\pm$0.11	0.53 $\pm$0.01	54.6
Breast cancer	16.59 $\pm$0.10	**0.49 $\pm$0.02**	33.9	48.72 $\pm$0.26	0.56 $\pm$0.01	87.0

Table 9: The evaluation of CPU time cost of different algorithms and the speedup of the SimpleNPKL method over the standard SDP solver. (The best results are bolded.)

that tag, that is, one or more photos of this user annotated with the tag). The k-nearest neighbor graph for MVE is constructed using cosine similarity between Flickr users. For SPE, we further constrain that the intra-group distance is smaller than the inter-group distance. In general, people who are friends or similar to each other tend to join the same interest group. Our goal is to apply the proposed MVE+NPKL and SPE+NPKL algorithms on these Flickr users in order to draw the 2D embedding results of the Flickr users exclusively belonging to two different interest groups: *B&W*[8] and *Catchy Colors*[9] as shown in Figure 7.

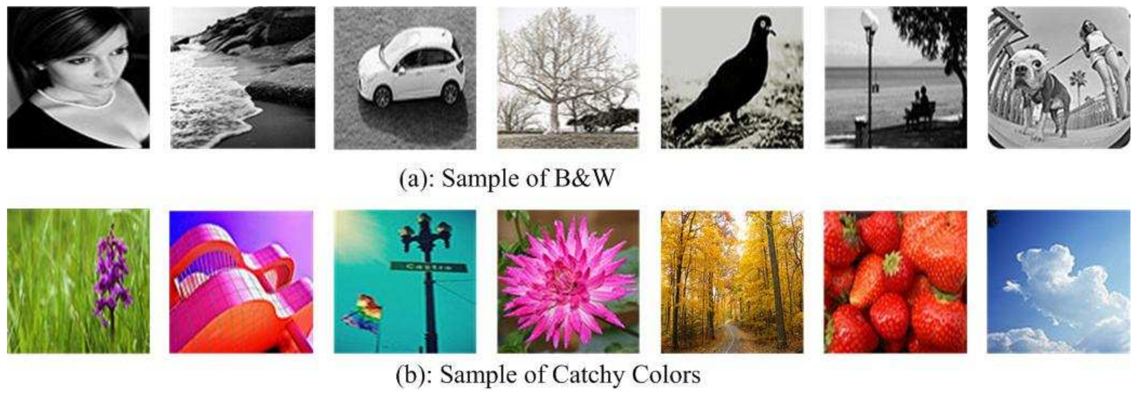
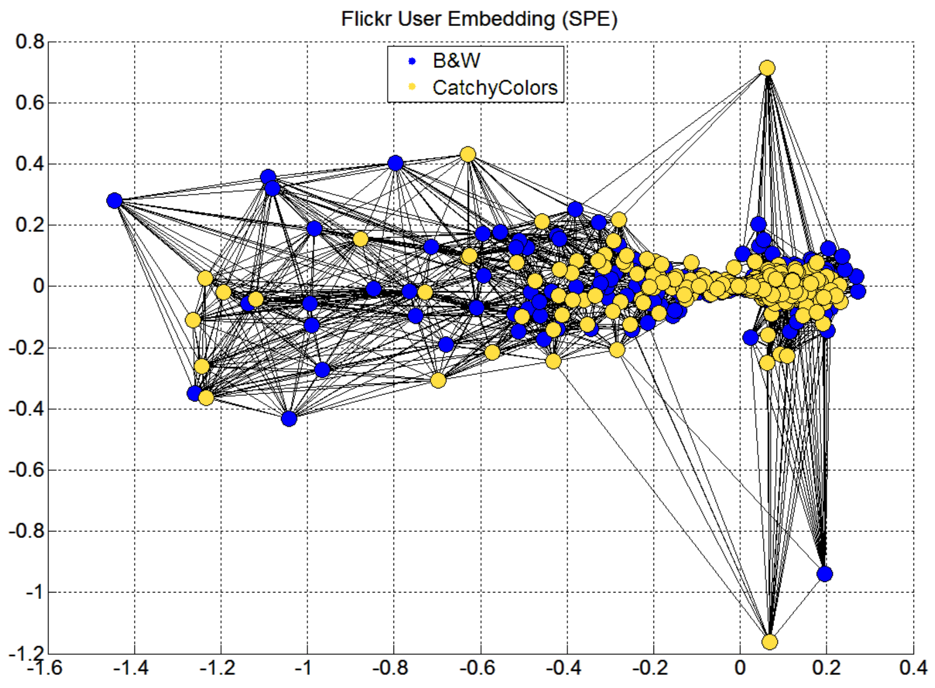


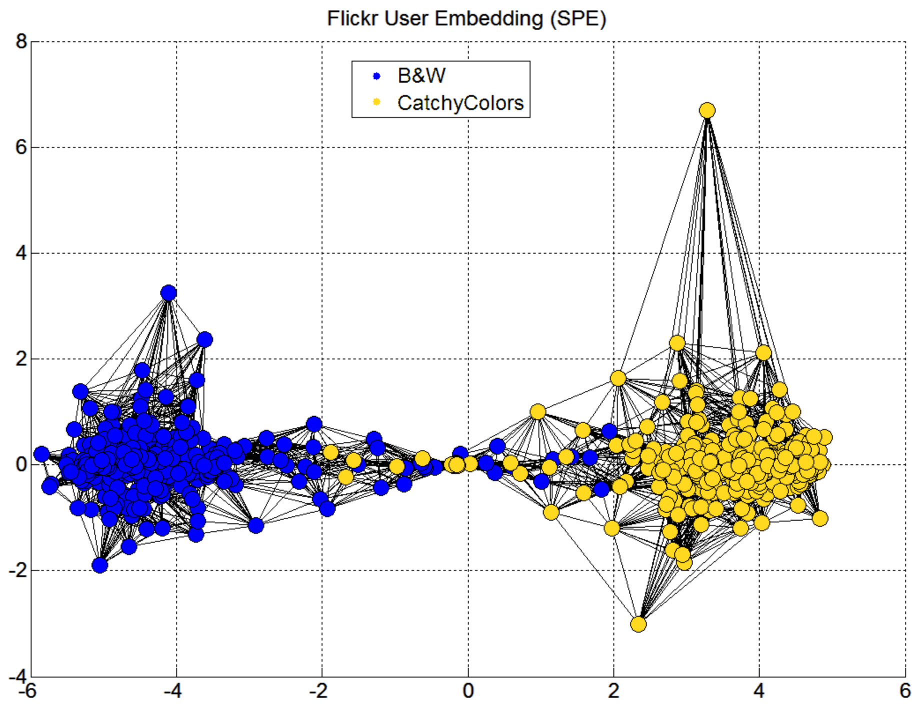
Figure 6: Sample photos from two Flickr interest groups: *B&W* and *Catchy Colors*.

Specifically, the theme of the group B&W is related to a collection of photos with black and white color only. The corresponding top 5 annotated tags for B&W are {*bw, black and white, black, white, portrait*}. In contrast, the top 5 tags for CatchyColors include {*red, blue, green, flower, yellow*}. Therefore, photos in the latter group are more colorful than the ones in B&W. An illustration of photos belonging to these two groups are depicted in Figure 6. However, the semantics of photos of these two groups are highly overlapping. Accordingly, the embedding results of MVE are highly overlapped as shown in Figure 7 (a), though it drives the spectral information into the top

8. *B&W* can be found at `http://www.flickr.com/groups/blackwhite/`.
9. *Catchy Colors* can be found at `http://www.flickr.com/groups/catchy/`.



(a) MVE+NPKL



(b) SPE+NPKL

Figure 7: The 2D embedding result of Flickr users exclusively belonging to the interest group *B&W* (blue points) and *Catchy Colors* (red points). The CPU time cost of MVE+NPKL and SPE+NPKL are 27.2 minutes and 196.4 minutes, respectively.

eigenvectors of the learned kernel matrix. On the other hand, by constraining intra-group distance less that inter-group distance, SPE can preserve the topology structure of these two groups as shown in Figure 7 (b). The 2D embedding shows the cluster structure rather clearly.

Note that the original algorithms using general SDP solvers cannot directly apply on the above large real data set. The proposed SimpleNPKL framework makes it feasible to analyze the emerging social networking problems using kernel learning methods. We hope our preliminary attempt in this paper could shed a light on a series of important applications in the future, including: 1) **Visualization:** as illustrated in Figure 7, we are able to obtain an intuitive understanding about the distribution of the entities in a social networking community. From Figure 7 (b), one can also observe the abnormal entities (e.g., the red dot on the right upper corner) and prototypes (the ones located at the centers of clusters). This may also benefit spam user detection and important user identification applications; 2) **Friend suggestion:** Given a Flickr user U_i, we can rank the other users U_j according to their similarity to U_i computed by the learned non-parametric kernel K_{ij}. With such information, a user can quickly find the other users of similar interests/tastes in photo sharing so as to facilitate the social communication between the users; 3) **Interest group recommendation:** It is interesting and beneficial to develop an intelligent scheme for recommending a Flickr user some interest groups. By applying the proposed kernel learning techniques to find similarity between Flickr users, it is possible for us to develop some recommendation scheme that suggests a Flickr user some interest groups that received the highest numbers of votes from its neighbors.

8. Conclusion

In this paper, we investigated a family of SimpleNPKL algorithms for improving the efficiency and scalability of the Non-Parametric Kernel Learning (NPKL) from large sets of pairwise constraints. We demonstrated that the proposed SimpleNPKL algorithm with linear loss for the pairwise constraints enjoys a closed-form solution, which can be simply computed by efficient sparse eigen-decomposition, such as the Lanczos algorithm. Moreover, our SimpleNPKL algorithm using other loss functions (including square hinge loss, hinge loss, and square loss) can be transformed into a saddle-point minimax optimization problem, which can be solved by an efficient iterative optimization procedure that only involves sparse eigen-decomposition computation. In contrast to the previous standard SDP solution, empirical results show that our approach achieved the same/comparable accuracy, but is significantly more efficient and scalable for large-scale data sets. We also explore some active constraint selection scheme to reduce the pairwise constraints in SimpleNPKL, which can further improve both computational efficiency and the clustering performance. Finally, we also demonstrate that the proposed family of SimpleNPKL algorithms can be applicable to other similar machine learning problems, in which we studied three example applications on data embedding problems. In the future, we will extend our technique for solving other SDP related machine learning problems.

Acknowledgments

We thank Dale Schuurmans' valuable comments and discussions. We also thank the valuable comments by the Action Editor and the anonymous reviewers. This research was in part supported by

Singapore MOE Tier-1 Grant (RG67/07), Tier-2 Grant (T208B2203), Tier-1 Grant (RG15/08), and A* SERC Grant (102 158 0034).

References

F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton. Complementarity and nondegeneracy in semidefinite programming. *Mathematical Programming*, 77:111–128, 1997.

A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceedings of Annual Conference on Learning Theory*, pages 338–352, 2005.

F. R. Bach and Z. Harchaoui. DIFFRAC: a discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems*, pages 49–56, 2008.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of International Conference on Machine Learning*, 2004.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of International Conference on Machine Learning*, pages 97–104, 2006.

J. F. Bonnans and E. Shapiro. Optimization problems with perturbations, a guided tour. *SIAM Review*, 40:228–264, 1996.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd and L. Xiao. Least-squares covariance matrix adjustment. *SIAM Journal on Matrix Analysis and Applications*, 27(2):532–546, 2005.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 15*, pages 585–592, 2003.

J. Chen and J. Ye. Training SVM with indefinite kernels. In *International Conference on Machine Learning*, pages 136–143, 2008.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373, 2002.

L. Duan, I.W. Tsang, D. Xu, and S.J. Maybank. Domain transfer SVM for video concept detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

P. V. Gehler and S. Nowozin. Infinite kernel learning. In *TECHNICAL REPORT NO. TR-178,Max Planck Institute for Biological Cybernetics*, 2008.

M. Gönen and E. Alpaydin. Localized multiple kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 352–359, 2008.

A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Proceedings of International Conference on Algorithmic Learning Theory*, pages 63–77, 2005.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008.

S. C. H. Hoi and R. Jin. Active kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 400–407, 2008.

S. C. H. Hoi, M. R. Lyu, and E. Y. Chang. Learning the unified kernel machines for classification. In *Proceedings of ACM SIGKDD conference on Knowledge Discovery and Data Mining*, pages 187–196, 2006.

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

T. Jebara and V. Shchogolev. B-matching for spectral clustering. In *European Conference on Machine Learning*, pages 679–686, September 2006.

R. Johnson and T. Zhang. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288, 2008.

K. Krishnan and J. E. Mitchell. A unifying framework for several cutting plane methods for semidefinite programming. *Optimization Methods and Software*, 21(1):57–74, 2006.

B. Kulis, M. Sustik, and I. S. Dhillon. Learning low-rank kernel matrices. In *Proceedings of International Conference on Machine Learning*, pages 505–512, 2006.

B. Kulis, M. A. Sustik, and I. S. Dhillon. Low-rank kernel learning with bregman matrix divergences. *Journal of Machine Learning Research*, 10:341–376, 2009.

J. Kwok and I. W. Tsang. Learning with idealized kernels. In *Proceedings of International Conference on Machine Learning*, pages 400–407, 2003.

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

R. B. Lehoucq, D. C. Sorensen, and C. Yang. ARPACK users guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods. Technical report, 1998.

D. P. Lewis, T. Jebara, and W. S. Noble. Nonstationary kernel combination. In *Proceedings of International Conference on Machine Learning*, pages 553–560, 2006.

F. Li, Y. Fu, Y.-H. Dai, C. Sminchisescu, and J. Wang. Kernel learning by unconstrained optimization. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2009.

M.S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra Applications*, 284:193–228, 1998.

R. Luss and A. d'Aspremont. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems 20*, 2008.

Q. Mao and I. W. Tsang. Parameter-free spectral kernel learning. In *Conference on Uncertainty in Artificial Intelligence*, 2010.

Q. Mao and I. W. Tsang. Multiple template learning for structured prediction. arXiv CoRR 1103.0890, Mar 2011.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103: 127–152, 2005.

Y. Nesterov and A. Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.

G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. Technical Report MSRR-604, Carnegie Mellon University, August 1995.

F. R. Rakotomamonjy, A.and Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

B. Shaw and T. Jebara. Minimum volume embedding. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2007.

B. Shaw and T. Jebara. Structure preserving embedding. In *Proceedings of Interational Conference on Machine Learning*, page 118, 2009.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of International Conference on Machine Learning*, pages 824–831, 2005.

L. Song, A. J. Smola, K. M. Borgwardt, and A. Gretton. Colored maximum variance unfolding. In *Advances in Neural Information Processing Systems 20*, 2008.

S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient multiple kernel learning algorithm. In *Advances in Neural Information Processing Systems 18*, 2006a.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006b.

J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *Proceedings of Interational Conference on Machine Learning*, page 134, 2009.

A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2009.

K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of International Conference on Machine Learning*, pages 1160–1167, 2008.

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of International Conference on Machine Learning*, 2004.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, 2003.

Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 1825–1832, 2008.

Y. Ying, C. Campbell, and M. Girolami. Analysis of SVM with indefinite kernels. In *Advances in Neural Information Processing Systems*, 2010.

T. Zhang and R. Ando. Analysis of spectral kernel design based semi-supervised learning. In *Advances in Neural Information Processing Systems 18*, 2006.

X. Zhu, J. S. Kandola, Z. Ghahramani, and J. D. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2005.

J. Zhuang and S. C. H. Hoi. Non-parametric kernel ranking approach for social image retrieval. In *Proceedings of the 9th ACM International Conference on Image and Video Retrieval*, pages 26–33, 2010.

J. Zhuang, I. W. Tsang, and S. C. H. Hoi. SimpleNPKL: simple non-parametric kernel learning. In *Proceedings of Interational Conference on Machine Learning*, 2009.

J. Zhuang, I. W. Tsang, and S. C. H. Hoi. Two-layer multiple kernel learning. In *Proceedings of Interational Conference on Artificial Intelligence and Statistics*, 2011.