SimpleNPKL : Simple Non-Parametric Kernel Learning

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

Abstract

Previous studies of Non-Parametric Kernel (NPK) learning usually reduce to solving some Semi-Definite Programming (SDP) problem by a standard SDP solver. However, time complexity of standard interior-point SDP solvers could be as high as $O(n^{6.5})$. Such intensive computation cost prohibits NPK learning applicable to real applications, even for data sets of moderate size. In this paper, we propose an efficient approach to NPK learning from side information, referred to as SimpleNPKL, which can efficiently learn non-parametric kernels from large sets of pairwise constraints. In particular, we show that the proposed SimpleNPKL with linear loss has a closed-form solution that can be simply computed by the *Lanczos* algorithm. Moreover, we show that the SimpleNPKL with square hinge loss can be re-formulated as a saddle-point optimization task, which can be further solved by a fast iterative algorithm. In contrast to the previous approaches, our empirical results show that our new technique achieves the same accuracy, but is significantly more efficient and scalable.

1. Introduction

Kernel methods have been widely applied in many real applications and usually shown the state-of-the-art performance. The choice of an effective kernel plays a crucial role in many kernel based machine learning techniques. Kernel methods, such as support vector machines (SVM), often adopt a predefined kernel empirically chosen from a pool of parametric kernel functions, such as polynomial and gaussian kernels. One major limitation of standard kernel methods is that choosing an appropriate kernel function manually may require certain domain knowledge, which may be difficult in some situations. Another limitation lies in the difficulty of tuning optimal parameters for the selected parametric kernel functions.

To address the above limitations, learning effective kernels from data automatically has been actively explored in recent years. One group of recent studies focuses on semi-supervised learning settings where the kernels are learned from mixtures of labeled and unlabeled data (Kondor & Lafferty, 2002; Chapelle et al., 2002; Zhang & Ando, 2005). Example techniques include diffusion kernels (Kondor & Lafferty, 2002), cluster kernels (Chapelle et al., 2002), and gaussian random field techniques (Zhu et al., 2003). These techniques often assume certain *parametric* forms of the target kernel functions and thus limit their capacity of fitting diverse patterns in real applications.

Instead of assuming parametric forms for target kernels, an emerging group of kernel learning studies are devoted to non-parametric kernel learning (Cristianini et al., 2001; Lanckriet et al., 2004; Zhu et al., 2004; Kulis et al., 2006). These include multiple kernel learning for learning a linear combination of multiple kernels from labeled data, and semi-supervised kernel learning for learning non-parametric kernel matrices from both labeled and unlabeled data (Zhu et al., 2004; Hoi et al., 2006). Although these techniques do not assume explicit parametric forms for the target kernels, they attempt to find the optimal kernels by linearly combining multiple kernels that are often parametric, which again may limit their performance.

Recently, Hoi et al. (2007) proposed a Non-Parametric Kernel (NPK) learning technique that aims to learn a fully non-parametric kernel matrix from pairwise constraints. In their approach, an effective regularizer is first introduced to exploit the local geometry of unlabeled data, and the non-parametric kernel learning problem is then formulated as a semi-definite program (SDP) that can be solved with a global solution by existing convex optimization techniques (Boyd & Vandenberghe, 2004). Unlike (Kulis et al., 2006), the performance of this NPK learning does not depend on the initial choice of the kernel matrix. The state-of-the-art performance for clustering tasks has been shown in (Hoi et al., 2007). Despite the promising performance, one

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009.

major limitation of the previous work lies in the difficulty of solving an SDP problem, which prohibits the technique applicable to large applications.

In this paper, we aim at addressing the efficiency and scalability issues related to the NPK learning technique (Hoi et al., 2007). Our main contributions include:

1) We propose a Simple NPK learning (SimpleNPKL) algorithm with linear loss from pairwise constraints. We show that the SimpleNPKL algorithm has a **closed-form solution**, which can be computed efficiently by sparse eigen-decomposition techniques, such as the *Lanczos* algorithm.

2) To have more robust performance, we propose another SimpleNPKL algorithm with square hinge loss, which can be re-formulated as a minimax optimization task. This optimization can be solved by an efficient iterative projection algorithm that involves mainly the computation of sparse eigen decomposition.

3) To further speedup our solutions, we also investigate some active constraint selection technique to reduce the computation cost in each iteration step.

4) Extensive experiments show that SimpleNPKL is significantly more efficient than the existing NPKL method. With the same linear loss function, SimpleNPKL is on average 75 times faster than the NPKL with a standard SDP solver. This makes the NPK learning techniques feasible to large applications.

The rest of this paper is organized as follows. Section 2 reviews the basics of NPK learning. Section 3 presents our proposed SimpleNPKL for the NPK learning. Implementation issues are discussed in Section 4. Section 5 gives discussions with related work. Section 6 shows our experimental results. Section 7 concludes this work.

2. Non-Parametric Kernel Learning

In the sequel, $\mathbf{0}$ and $\mathbf{1}$, respectively, denote the column vectors with all zeros and all ones, and I denotes an identity matrix. $A \succ \mathbf{0}$ (resp. $A \succeq \mathbf{0}$) means matrix A is symmetric and positive definite (pd) (resp. positive semidefinite (psd)). Moreover, $^\top$ denotes the transpose of vector/matrix, and $\mathrm{tr}(A)$ denotes the trace of matrix A.

2.1. 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 some collection of similar (positive) pairwise constraints $\mathcal{S}$ (known as "must-links", i.e., the data pairs share the same class) and a collection of dissimilar (negative) pairwise constraints $\mathcal{D}$ (known as "cannot-links", i.e., the data pairs have different classes), which is 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 & Tsang, 2003; Kulis et al., 2006; Hoi et al., 2007), i.e., one can construct the sets of pairwise constraints $\mathcal{S}$ and $\mathcal{D}$ from $\mathcal{L}$. In the sequel, we focus on learning kernels from pairwise constraints.

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

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

An intuitive principle for kernel earning is that the kernel entry K_{ij} should be aligned with the side information T_{ij} as much as possible (Cristianini et al., 2001), i.e., the alignment $T_{ij}K_{ij}$ of each kernel entry is maximized.

2.2. Locality Preserving

In addition to side/label information, preserving of the intrinsic geometric structure of the data can also be explored to improve the performance of kernel learning. Typically, most existing kernel learning approaches (Kulis et al., 2006; Hoi et al., 2007; Hoi & Jin, 2008) adopt the data manifold (Sindhwani et al., 2005) for preserving the locality in kernel learning. Below reviews an approach for exploring 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 $\mathbf{x}$ and $\mathbf{x}'$, and $S \in \mathbb{R}^{N \times N}$ the similarity matrix where each element $S_{i,j} = f(\mathbf{x}_i, \mathbf{x}_j) \geq 0$. Note that $f(\cdot, \cdot)$ does not have to be a kernel function that satisfies the Mercer's condition. Generally, a Non-Parametric Kernel (NPK) matrix K with respect to N patterns can be expressed as $K = V^\top V \succeq \mathbf{0}$,where $V = [\mathbf{v}_1, \ldots, \mathbf{v}_N]^\top$ is the matrix of the embedding of data points. The regularizer of the kernel matrix K, which captures the local dependency between the embedding of $\mathbf{v}_i$ and $\mathbf{v}_j$, can be defined as:

$$\begin{aligned} \Omega(V, S) &= \sum_{i,j=1}^{N} S_{i,j} \left\| \frac{\mathbf{v}_i}{\sqrt{d_i}} - \frac{\mathbf{v}_j}{\sqrt{d_j}} \right\|_2^2 \\ &= \mathrm{tr}(VLV^\top) = \mathrm{tr}(LK), \end{aligned} \quad (2)$$

where L is the graph Laplacian matrix defined as:

$$L = I - D^{-1/2} S D^{-1/2}, \quad (3)$$

where $D = \mathrm{diag}(d_1, d_2, \ldots, d_N)$ is a diagonal matrix with the diagonal element defined as $d_i = \sum_{j=1}^{N} S_{ij}$.

2.3. SDP Formulation of NPK Learning

Following (Hoi et al., 2007), taking into consideration of both the side information in (1) and the regularizer in (2),

the NPK learning can be formulated as follows:

$$\min_{K\succeq\mathbf{0}} \quad \text{tr}(LK) + C\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \ell(T_{i,j}K_{i,j}), \qquad (4)$$

which generally belongs to a semi-definite programming (SDP) problem (Boyd & Vandenberghe, 2004). Here, $C > 0$ is a tradeoff parameter to control the empirical loss $\ell(\cdot)$ [1] of the alignment of the learned kernel and the dependency among patterns w.r.t. the intrinsic data manifold. The above optimization can be solved with global optima by standard interior-point SDP solvers. However, the time complexity of these SDP solvers can be as high as $O(N^{6.5})$, which prohibits the NPK learning technique for real applications, even for medium-size problems.

3. Simple Non-Parametric Kernel Learning

In this section, we present our fast algorithms to solving the NPK learning problem. First of all, we introduce the following important constraint:

$$\text{tr}(KK) \le B \qquad (5)$$

into the NPK learning problem (4) to control the capacity of the kernel matrix K, where $B > 0$ is a constant. Similar constraints were also adopted in previous kernel learning studies (Kwok & Tsang, 2003; Lanckriet et al., 2004). We refer to the modified NPK learning problem with (5) as Simple NPK Learning (SimpleNPKL), which can be solved efficiently without engaging any standard SDP solvers. Next we present two SimpleNPKL approaches using two types of loss functions.

3.1. SimpleNPKL with Linear Loss

Consider a linear loss function $\ell(f) = -f$, we can rewrite the formulation of (4) as the SimpleNPKL form:

$$\min_K \text{tr}\left(\left(L - C\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})}\mathbf{T}_{i,j}\right)K\right) \;:\; K\succeq\mathbf{0},\ \text{tr}(KK)\le B, \quad (6)$$

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 show a proposition below.

Proposition 1. *Given A is any symmetric matrix such that $A = P\text{diag}(\Sigma)P^\top$, where P contains columns of orthonormal eigenvectors of A and Σ is a vector of the corresponding eigenvalues, and B is any positive constant, the optimal solution K^* to the following SDP problem:*

$$\max_K tr(AK) \;:\; K\succeq\mathbf{0},\ tr(KK)\le B, \qquad (7)$$

can be expressed as the following closed-form solution:

$$K^* = A_+\sqrt{\frac{B}{tr(A_+A_+)}} \qquad (8)$$

where $A_+ = P\text{diag}(\Sigma_+)P^\top$, and Σ_+ is a vector with entries equal to $\max(0,[\Sigma]_i)$.

Proof. By introducing a dual variable $\lambda \ge 0$ for the constraint $\text{tr}(KK) \le B$, and $Z \in \mathcal{S}_+^n$($\mathcal{S}_+^n$ is self-dual) for the constraint $K \succeq \mathbf{0}$, we have the Lagrangian equation of (7):

$$\mathcal{L}(K;\lambda,Z) = \text{tr}(AK) + \lambda(B - \text{tr}(KK)) + \text{tr}(KZ).$$

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

$$\frac{\partial\mathcal{L}}{\partial K} = A - 2\lambda K + Z = 0 \quad \text{and} \quad \text{tr}(KZ) = 0.$$

First, we show that $\text{tr}(KZ) = 0$ is equivalent to $KZ = ZK = \mathbf{0}$. Since $K\succeq\mathbf{0}, Z\succeq\mathbf{0}$, we have $\text{tr}(KZ) = \text{tr}(K^{\frac{1}{2}}K^{\frac{1}{2}}Z^{\frac{1}{2}}Z^{\frac{1}{2}}) = \|K^{\frac{1}{2}}Z^{\frac{1}{2}}\|_F^2$.Thus, $\text{tr}(KZ) = 0$ follows that $K^{\frac{1}{2}}Z^{\frac{1}{2}} = \mathbf{0}$. Pre-multiplying by $K^{\frac{1}{2}}$ and post-multiplying by $Z^{\frac{1}{2}}$ yields $KZ = 0$, which in turn implies $KZ = \mathbf{0} = (KZ)^\top = ZK$. Hence, K and Z, or equivalently A, can be simultaneously diagonalized by the same set of orthonormal eigenvectors (Alizadeh et al., 1997).

Assume $A = P\text{diag}(\Sigma)P^\top$, where P contains columns of orthonormal eigenvectors of A, and Σ is the vector of the corresponding eigenvalues. Then, $K = P\text{diag}(\Lambda)P^\top$, where Λ denotes the vector of the eigenvalues of K. Therefore, $\text{tr}(KK) = \Lambda^\top\Lambda \le B$ and $\text{tr}(AK) = \Lambda^\top\Sigma \le \sqrt{(\Lambda^\top\Lambda)(\Sigma^\top\Sigma)} \le \sqrt{B\Sigma^\top\Sigma}$. The inequality is obtained by Cauchy-Schwarz inequality. Together with $\text{diag}(\Lambda)\succeq\mathbf{0}$, we obtain the closed-form solution of K in (8). □

Based on Proposition 1, we can easily solve the SimpleNPKL problem. In particular, by setting $A = C\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})}\mathbf{T}_{i,j} - L$, we can directly compute the optimal K^* to SimpleNPKL of (6) using sparse eigen-decomposition as in (8). This apparently significantly reduces the time cost for the NPK learning tasks.

3.2. SimpleNPKL using Square Hinge Loss

Though the formulation with linear loss in (6) gives rise to a closed-form solution for the NPK learning, the formulation of the NPK learning using (square) hinge loss $\ell(f) = (\max(0, 1-f))^d/d$ sometimes can be more robust, where $d = 1$(hinge loss) or 2(square hinge loss). Next, we focus on the formulation with the square hinge loss, which can be written into the following constrained optimization:

$$\min_{K,\epsilon_{i,j}} \quad \text{tr}(LK) + \frac{C}{2}\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})}\epsilon_{i,j}^2 \qquad (9)$$

$$\text{s.t.} \quad \forall(i,j)\in(\mathcal{S}\cup\mathcal{D}),\ T_{i,j}K_{i,j}\ge 1-\epsilon_{i,j}, \qquad (10)$$

$$K\succeq\mathbf{0}, \text{tr}(KK)\le B.$$

Note that we ignore the constraints $\epsilon_{ij} \ge 0$ since they can be satisfied automatically. However, (9) is not in form of (7), and so, there is no longer a closed-form solution for K.

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

3.2.1. SADDLE-POINT MINIMAX PROBLEM

By introducing dual variables α_{ij}'s ($\alpha_{ij} \geq 0$) for the constraints in (10), we can derive a partial Lagrangian:

$$\text{tr}(LK) + \frac{C}{2}\sum_{(i,j)} \epsilon_{i,j}^2 - \sum_{(i,j)} \alpha_{ij}(T_{i,j}K_{i,j} - 1 + \epsilon_{i,j}). \quad (11)$$

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 (11) w.r.t. the primal variables ϵ_{ij}'s to zeros, we have $\forall(i,j) \in (\mathcal{S} \cup \mathcal{D})$, $C\epsilon_{ij} = \alpha_{ij} \geq 0$, and substituting them back into (11), we arrive at the following saddle-point minimax problem $\mathcal{J}(K, \boldsymbol{\alpha})$:

$$\max_{\alpha_{ij}} \min_{K} \text{tr}\left(\left(L - \sum_{(i,j)} \alpha_{ij}\mathbf{T}_{i,j}\right)K\right) - \frac{1}{2C}\sum_{(i,j)} \alpha_{ij}^2 + \sum_{(i,j)} \alpha_{ij} \quad (12)$$
$$\text{s.t.} \qquad K \succeq \mathbf{0},\ \text{tr}(KK) \leq B,\ \forall(i,j) \in \mathcal{S}\cup\mathcal{D},\ \alpha_{ij} \geq 0,$$

where $\boldsymbol{\alpha} = [a_{ij}]$ is the vector of dual variables α_{ij}'s. This problem is similar to the optimization problem of DIFFRAC (Bach & Harchaoui, 2008), in which K and $\boldsymbol{\alpha}$ can be solved by an iterative manner.

3.2.2. ITERATIVE ALGORITHM

Based on Proposition 1, for fixed α_{ij}'s, we can let $A = \sum_{(i,j)} \alpha_{ij}\mathbf{T}_{i,j} - L$, and find the optimal K to (12) by a closed-formed solution as in (8). On the other hand, with a fixed K, we have the following optimization problem:

$$\max_{\alpha_{ij}\geq 0} \sum_{(i,j)} \alpha_{ij}(1 - T_{i,j}K_{ij}) - \frac{1}{2C}\sum_{(i,j)} \alpha_{ij}^2, \quad (13)$$

which is a quadratic programming (QP) problem. It is not difficult to show that the optimal α^* of (13) is given by:

$$\alpha_{ij}^* = C\max(0, 1 - T_{ij}K_{ij}). \quad (14)$$

Let $\mathcal{J}(K, \boldsymbol{\alpha})$ denote the objective function in (12), for the best K_t at the t-th step, we have

$$\mathcal{J}_{K_t} = \min_K \mathcal{J}(K, \boldsymbol{\alpha}_{t-1}) \leq \min_K \max_{\boldsymbol{\alpha}} \mathcal{J}(K, \boldsymbol{\alpha}) = \mathcal{J}^*$$

where $\min_K$ and $\max_{\boldsymbol{\alpha}}$ commute due to the strong duality. On the other hand, we also have

$$\mathcal{J}_{\boldsymbol{\alpha}_t} = \max_{\boldsymbol{\alpha}} \mathcal{J}(K_t, \boldsymbol{\alpha}) \geq \max_{\boldsymbol{\alpha}} \min_K \mathcal{J}(K, \boldsymbol{\alpha}) = \mathcal{J}^*.$$

Thus, at each step, we have $\mathcal{J}_{K_t} \leq \mathcal{J}(K^*, \boldsymbol{\alpha}^*) \leq \mathcal{J}_{\boldsymbol{\alpha}_t}$. However, there is no guarantee that K_t and $\boldsymbol{\alpha}_t$ would always make progress toward the optimal solution $(K^*, \boldsymbol{\alpha}^*)$, i.e., it may occur that $\mathcal{J}_{\boldsymbol{\alpha}_{t+1}} > \mathcal{J}_{\boldsymbol{\alpha}_t}$ or $\mathcal{J}_{K_{t+1}} < \mathcal{J}_{K_t}$. This can be explained by a zero-sum game between two players, K and $\boldsymbol{\alpha}$ in the saddle-point minimax problem (Taskar et al., 2006). In each iteration, each player makes its best-response improvement without considering the effect of the change from the opponent's strategy. This usually leads to fluctuation on $\mathcal{J}$ and makes slow progress for convergence.

To alleviate this problem, we follow the similar update strategy in (Boyd & Xiao, 2005): 1) Compute the closed-form solution K_t using (8) for a fixed $\boldsymbol{\alpha}_{t-1}$; 2) Update $\boldsymbol{\alpha}_t$ using $\alpha_{ij}^t = \left(\alpha_{ij}^{t-1} + \eta_t\left(\frac{\partial\mathcal{J}}{\partial\alpha_{ij}}\right)\right)_+$; 3) Step 1) and 2) are iterated until convergence. Note that $\eta_t > 0$ is a step-size parameter. 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 & Xiao, 2005). In our approach, we initialize $\boldsymbol{\alpha}^0$ as $\mathbf{1}$, the corresponding K_1 is the solution of SimpleNPKL with linear loss. The whole algorithm is summarized in Algorithm 1.

Algorithm 1 SimpleNPKL with (square) hinge loss.

Input: Pairwise constraint matrix T, parameters C and B;
Output: α and K.

1: Construct graph Laplacian L.
2: Initialize α_{ij}^0.
3: Set $A = \sum_{(i,j)} \alpha_{ij}^t\mathbf{T}_{i,j} - L$.
4: Compute the closed-form solution of K_t using (8).
5: Update α_{ij}^t using $\alpha_{ij}^t = \left(\alpha_{ij}^{t-1} + \eta_t\left(\frac{\partial\mathcal{J}}{\partial\alpha_{ij}}\right)\right)_+$.
6: Repeat Steps 3-5 until convergence.

3.2.3. ESTIMATING THE RANK OF K

According to Proposition 1, we need to locate the positive spectrums of A, which can be achieved by full eigen-decomposition of A. However, this can be computationally prohibitive for large scale data sets. Moreover, the computation on the negative eigen-vectors of A should be avoided. The following proposition (Pataki, 1995) bounds the rank of matrix K in a general SDP setting.

Proposition 2. *The rank r of K in the SDP problem: $\max_{K\succeq\mathbf{0}} tr(A_0K)$ with m linear constraints on K, follows the bound $\binom{r+1}{2} \leq m$.*

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 A_0 is not required. Our formulation (9) has an additional constraint: $\text{tr}(KK) \leq B$. This condition equivalently constraints $\text{tr}(K)$, which is a common assumption in SDP problems (Kartik Krishnan, 2006). To show this, we have $B \geq \text{tr}(KK) = \frac{1}{N}\sum_i \Lambda_i^2 N \geq \frac{1}{N}(\sum_i \Lambda_i \cdot 1)^2 = \frac{1}{N}\text{tr}(K)$, where the second inequality is resulted from the Cauchy inequality. Hence, we have $\text{tr}(K) \leq \sqrt{BN}$. Therefore, we make use of the r estimated from Proposition (2) to estimate the rank of K.

3.2.4. ACTIVE CONSTRAINT SELECTION

We notice that the computation cost of the update procedure as shown in Algorithm 1 heavily depends on the number of pairwise constraints. However, some less informa-

tive constraints often do not contribute much to the learning of K, and some noisy constraints may lead to the poor generalization. Moreover, as discussed in Section 3.2.3, the rank of K is lower when there are fewer active constraints in (10). 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, one can sample a subset of T_{ij}'s for SimpleNPKL. Recently, Hoi and Jin (2008) have proposed a min-max active kernel learning framework (for acquiring class label information). 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 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, 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:

$$i^* = \operatorname*{argmin}_i \left| \frac{1}{l_i} \sum_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \right| = \operatorname*{argmin}_i \left| \frac{1}{l_i} \sum_{j, T_{ij} \neq 0} T_{ij} K(\mathbf{x_i}, \mathbf{x_j}) \right|,$$

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.

4. Implementation Issues

4.1. Building a Sparse Graph Laplacian

The graph Laplacian L in (3) is often sparse, which can be computed by finding k-nearest neighbors for the purpose of constructing the similarity matrix S. Specifically, an entry $S(i,j) = 1$ *iff* 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 L. Moreover, the number of pairwise constraints is usually small due to expensive cost of human labels. Thus, $L - \sum_{(i,j)} \alpha_{ij} T_{ij}$ is also sparse.

A naive implementation of finding k-nearest neighbors often takes $O(N^2 \log N)$ time. When the data set is large, the construction of L becomes non-trivial. To address this challenge, we suggest to first construct the *cover tree* structure (Beygelzimer et al., 2006), which takes $O(N \log N)$ time. 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 L can be constructed efficiently for large-scale problems.

4.2. Fast Eigendecomposition by Lanczos Algorithm

Among various existing SDP approaches (Boyd & Vandenberghe, 2004), the interior-point method is often deemed as the most efficient one. However, the sparse structure of the scalar matrix $\boldsymbol{\alpha} \circ T - L$ is not 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 2, the rank r of the kernel matrix 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. When the input matrix A is sparse, IRLA can be very efficient.

5. Discussions with Related Work

Our work is related to some studies of distance metric learning (DML) (Weinberger & Saul, 2008). Similarly, SimpleNPKL also learns an effective data representation from pairwise constraints. However, the similarity between points $\mathbf{x}_i$ and $\mathbf{x}_j$ defined in DML is in some parametric form of $\mathbf{x}_i^\top M \mathbf{x}_j$ where $M \succeq \mathbf{0}$, the dimension of M is $d \times d$ and d is the input dimension. Due to the iterative manner, the update procedure in (Weinberger & Saul, 2008) is analogous to Algorithm 1. In contrast to Weinberger and Saul (2008), the update of K in SimpleNPKL can be computed in closed-form with (8), while Weinberger and Saul (2008) formulated DML as an unconstrained optimization problem, which first used gradient descent to update M in the primal, and then adopted a projection method to enforce M satisfying the psd constraint. Moreover, the projection of M is expensive when the input dimension d is high. Also, the procedure in (Weinberger & Saul, 2008) cannot exploit the sparse structure of graph Laplacian for further speedup.

Unlike the non-convex gradient descent procedure used in the low-rank SDP (Kulis et al., 2007), SimpleNPKL uses convex optimization procedures and thus global optimum is guaranteed. Moreover, SimpleNPKL employs several different loss functions. In particular, the update in SimpleNPKL with linear loss can be computed in closed-form with (8), which can be obtained using sparse eigen-decomposition without any iterative process.

6. Experiments

6.1. Experimental Setup

We examine both efficacy and efficiency of the proposed SimpleNPKL for clustering. As shown in (Hoi et al., 2007),

the Non-Parametric Kernel Learning (NPKL) outperforms other kernel learning methods. For simplicity, we only compare our proposed SimpleNPKL with NPKL. The results of k-means clustering is also reported as the baseline method. The abbreviations of different approaches are described as follows: 1) **k-means**: k-means clustering; 2) **SimpleNPKL+LL**: The proposed SimpleNPKL with linear loss defined in (6); 3) **SimpleNPKL+SHL**: The proposed SimpleNPKL with squared hinge loss defined in (9); 4) **NPKL+LL**: NPKL in (4) using linear loss; 5) **NPKL+HL**: NPKL in (4) using hinge loss. To construct the graph Laplacian matrix 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, SeDuMi[4] for the NPKL. The pair-wise constraint is assigned for randomly generated pairs of points according to their labels. The number of constraints is controlled by the resulted amount of connected components as defined in (Xing et al., 2002). Note that typically the larger the number of constraints, the smaller the number of connected components.

Several parameters are involved in NPKL and SimpleNPKL. Their notation and settings are given as follows: 1) k : The number of nearest neighbors for constructing the graph Laplacian matrix L, we set it to 5 for small data sets in Table 1, and 50 for Adult database in Table 3; 2) r : The ratio of the number of connected components compared with the data set size N. In our experiments, we set $r = 70\%N$ which follows the setting of (Hoi et al., 2007); 3) B : The parameter that controls the capacity of the learned kernel in (5). We fix $B = N$ for the adult data sets and fix $B = 1$ for the data sets in Table (1) and; 4) 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 are obtained by averaging the results from 20 different repetitions. All experiments were conducted on a Windows PC with 3.4GHz 32bit CPU and 3GB RAM.

6.2. Comparisons on Benchmark Data Sets

To evaluate the clustering performance, we adopt the clustering accuracy used in (Hoi et al., 2007): ClusterAccuracy $= \sum_{i>j} \frac{\mathbf{1}\{c_i=c_j\}=\mathbf{1}\{\hat{c}_i=\hat{c}_j\}}{0.5n(n-1)}$.This metric measures the percentage of data pairs that are correctly clustered together. We compare the proposed SimpleNPKL with NPKL on the nine UCI repository data sets[5] (summarized in Table 1), which are also used in (Hoi et al., 2007).

[2]http://hunch.net/ jl/projects/cover_tree/cover_tree.html

[3]http://www.caam.rice.edu/software/ARPACK/

[4]http://sedumi.ie.lehigh.edu/

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

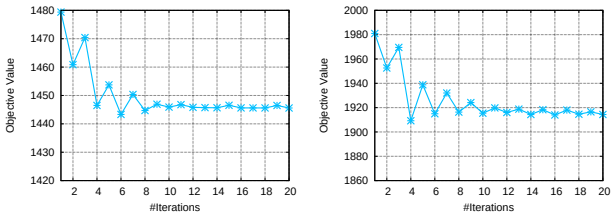
Table 1. The statistics of the data sets used in our experiments.

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
Wine	3	178	12
Double-Spiral	2	100	3

Table 3. The statistics of the *Adult* database.

Data Set†	a1a	a2a	a3a	a4a	a5a
#Instances	1,605	2,265	3,185	4,781	6,414

†: #Classes=2, #Features=123



(a) a1a (b) a2a

Figure 1. Convergence of SimpleNPKL using square hinge loss on *a1a* and *a2a*. The parameters are $C = 1$, $B = N$.

The clustering accuracy and CPU time (the time for clustering is excluded) of different NPKL methods are reported in Table 2. As can be seen from Table 2, SimpleNPKL using square hinge loss produces very competitive clustering performance to the results of NPKL with hinge loss (reported in (Hoi et al., 2007)). All NPKL methods outperform k-means clustering. SimpleNPKL with square hinge loss and NPKL with hinge loss often perform better than the NPK learning methods using linear loss. While the CPU time for SimpleNPKL and NPKL using linear loss are usually lower than that of their counterparts with (square) hinge loss.Regarding the efficiency, our SimpleNPKL using squared hinge loss is consistently 10 times faster than NPKL using a standard SDP solver. For the case of linear loss, SimpleNPKL can be even 100 times faster.

6.3. Scalability Study on Adult Data Set

In this section, we evaluate our SimpleNPKL on another larger dataset to examine the efficiency and scalability of SimpleNKPL. We adopt the *Adult* database, which is available at the website of LibSVM[6]. The database has

[6]http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/.

Table 2. Clustering accuracy and CPU time of SimpleNPKL, compared with the results of NPKL in (4) using a standard SDP solver, and k-means. (The best results are in bold and the last "Speedup" column is listed only for the linear loss case.)

Data Set	Accuracy(%)					CPU Time(s)				
	k-means	NPKL		SimpleNPKL		NPKL		SimpleNPKL		Speedup
		LL	HL	LL	SHL	LL	HL	LL	SHL	LL
Iris	84.5±6.5	96.0± 6.1	**97.4± 0.0**	**97.4± 0.0**	**97.4± 0.0**	3.39	29.73	**0.07**	3.35	48.4
Sonar	50.2±0.1	76.8± 0.3	64.5± 6.8	70.2± 10	**78.0± 0.0**	10.62	30.30	**0.10**	2.78	106.2
Chboard	49.8±0.2	**61.1± 6.9**	56.3± 6.1	58.1± 7.2	58.8± 0.8	2.44	4.87	**0.05**	0.12	48.8
Glass	69.7±1.9	74.4± 3.7	**79.1± 4.9**	74.3± 2.4	73.5± 2.9	9.25	32.09	**0.11**	2.95	84.1
Heart	51.5±0.1	86.0± 0.3	86.2± 0.0	86.8± 0.0	**89.4± 0.1**	22.33	65.15	**0.17**	12.87	131.4
Protein	76.2±2.0	78.2± 3.2	**86.4± 3.8**	81.8± 1.8	75.9± 2.0	1.96	7.58	**0.05**	0.48	39.2
Soybean	82.1±6.1	90.2± 7.7	**100.0± 0.0**	95.3± 5.1	95.4± 4.9	0.95	2.52	**0.01**	0.27	95.0
Wine	71.2±1.2	78.1± 1.7	**85.5± 5.3**	83.7± 4.8	85.0± 2.6	6.07	28.85	**0.08**	4.41	75.9
Spriral	50.1±0.6	86.5± 0.0	**94.1± 0.0**	92.2± 0.0	**94.1± 0.0**	2.30	4.40	**0.05**	1.81	46.0

Table 4. Clustering accuracy and CPU time of SimpleNPKL on *Adult* data set. (The best results are in bold.)

Data Set	#Cons.	Accuracy(%)			CPU Time(s)	
		k-means	SimpleNPKL		SimpleNPKL	
			LL	SHL	LL	SHL
a1a	4,104	56.4±3.5	**61.4±1.7**	60.7±2.7	**8.5**	322.9
a2a	5,443	57.3±3.6	61.1±1.3	**61.4±1.2**	**15.3**	637.2
a3a	7,773	57.8±3.5	61.1±1.7	**61.5±2.0**	**28.8**	1,160.8
a4a	12,465	58.8±1.6	**61.6±1.3**	61.4±1.5	**66.3**	2,341.3
a5a	16,161	57.7±3.1	60.8±3.1	**61.9±1.7**	**79.6**	3,692.1

a series of partitions: a1a, a2a, $\cdots$, a5a (see Table 3). 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 k-means clustering as the baseline comparison.

Table 4 shows the clustering performance and CPU time cost (the time for clustering is excluded) of SimpleNPKL on the Adult database. From the results, we have 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 simple k-means clustering. Further, comparing the two algorithms themselves, in terms of clustering accuracy, they perform 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 $\mathcal{J}(K, \boldsymbol{\alpha})$ of SimpleNPKL on two data sets *a1a* and *a2a* in Figure 1. We observe that SimpleNPKL with square hinge loss often converges within 10 iterations. Similar results can be observed from other data sets.

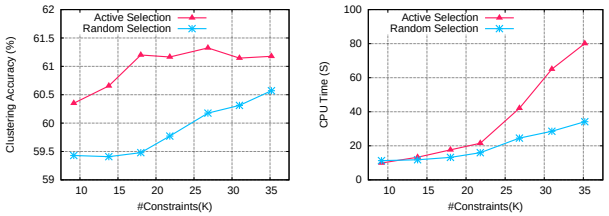
6.4. Comparisons on Constraint Selection

In this section, we study the active constraint scheme for SimpleNPKL. Figure 2 shows the clustering performance of active constraint selection using the simple approach described in Section 3.2.4. Several observations can be drawn: 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 matrix with fewer nonzero entries; 2) Though the active constraint selection costs more time than random selection, the former usually achieves better clustering performance than the latter with the same amount of constraints; 3) Using the proposed active constraint selection method to choose about 1/2 of the pairwise constraints for SimpleNPKL can often produce comparable or even better clustering performance than that using all constraints.

7. Conclusion

In this paper, we investigate fast algorithms for the NPK learning from pairwise constraints. We demonstrate that our proposed SimpleNPKL using linear loss for the pairwise constraints enjoys a closed-form solution, which can be computed efficiently by sparse eigen-decomposition, such as Lanczos algorithm. Moreover, our SimpleNPKL using square hinge loss can be transformed into a saddle-point minimax optimization problem, which is solved by an efficient iterative procedure also only involving sparse eigen-decomposition. In contrast to the previous SDP solution, empirical results show that our approach achieved the same accuracy, but is significantly more efficient and scalable for large-scale data sets. We also explore an active constraint selection scheme to reduce the pairwise constraints in SimpleNPKL. Both computational efficiency and clustering performance could be further improved. In

the future, we will extend our algorithm for other SDP related machine learning problems.



(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 $3.9K$ constraints directly, the accuracy is 60.8 ± 2.9 and the CPU time is 81.6 seconds.

Acknowledgments

This research was in part supported by Singapore MOE AcRF Tier-1 Research Grant (RG15/08) and Research Grant (RG67/07).

References

Alizadeh, F., Haeberly, J.-P. A., & Overton, M. L. (1997). Complementarity and nondegeneracy in semidefinite programming. *Math. Program.*, *77*, 111–128.

Bach, F., & Harchaoui, Z. (2008). Diffrac: a discriminative and flexible framework for clustering. *Neural Information Processing Systems* (pp. 49–56).

Beygelzimer, A., Kakade, S., & Langford, J. (2006). Cover trees for nearest neighbor. *International Conference on Machine Learning* (pp. 97–104).

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Boyd, S., & Xiao, L. (2005). Least-squaures covariance matrix adjustment. *SIAM Journal of Matrix Anal. Appl.*, *27*, 532–546.

Chapelle, O., Weston, J., & Schölkopf, B. (2002). Cluster kernels for semi-supervised learning. *Neural Information Processing Systems* (pp. 585–592).

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. S. (2001). On kernel-target alignment. *Neural Information Processing Systems* (pp. 367–373).

Hoi, S. C. H., & Jin, R. (2008). Active kernel learning. *International Conference on Machine Learning* (pp. 400–407).

Hoi, S. C. H., Jin, R., & Lyu, M. R. (2007). Learning nonparametric kernel matrices from pairwise constraints. *International Conference on Machine Learning* (pp. 361–368).

Hoi, S. C. H., Lyu, M. R., & Chang, E. Y. (2006). Learning the unified kernel machines for classification. *Knowledge Discovery and Data Mining* (pp. 187–196).

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

Kondor, R. I., & Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces. *International Conference on Machine Learning* (pp. 315–322).

Kulis, B., Surendran, A., & Platt, J. (2007). Fast low-rank semidefinite programming for embedding and clustering. *International Conference on Artificial Intelligence and Statistics*.

Kulis, B., Sustik, M., & Dhillon, I. S. (2006). Learning low-rank kernel matrices. *International Conference on Machine Learning* (pp. 505–512).

Kwok, J., & Tsang, I. (2003). Learning with idealized kernels. *International Conference on Machine Learning* (pp. 400–407).

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

Lehoucq, R. B., Sorensen, D. C., & Yang, C. (1998). *Arpack users guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods* (Technical Report).

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

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. *International Conference on Machine Learning* (pp. 824–831).

Taskar, B., Lacoste-Julien, S., & Jordan, M. I. (2006). Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research*, *7*, 1627–1653.

Weinberger, K. Q., & Saul, L. K. (2008). Fast solvers and efficient implementations for distance metric learning. *International Conference on Machine Learning* (pp. 1160–1167).

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2002). Distance metric learning with application to clustering with side-information. *Neural Information Processing Systems* (pp.505–512).

Zhang, T., & Ando, R. K. (2005). Analysis of spectral kernel design based semi-supervised learning. *Neural Information Processing Systems*.

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *International Conference on Machine Learning* (pp. 912–919).

Zhu, X., Kandola, J. S., Ghahramani, Z., & Lafferty, J. D. (2004). Nonparametric transforms of graph kernels for semi-supervised learning. *Neural Information Processing Systems*.