Learning the Optimal Neighborhood Kernel for Classification

Jun Liu[1,2], **Jianhui Chen**[1], **Songcan Chen**[2], and **Jieping Ye**[1]
[1]Department of Computer Science and Engineering
Arizona State University
[2]Department of Computer Science and Engineering
Nanjing University of Aeronautics and Astronautics
{J.Liu, Jianhui.Chen, Jieping.Ye}@asu.edu, S.Chen@nuaa.edu.cn

Abstract

Kernel methods have been applied successfully in many applications. The kernel matrix plays an important role in kernel-based learning methods, but the "ideal" kernel matrix is usually unknown in practice and needs to be estimated. In this paper, we propose to directly learn the "ideal" kernel matrix (called the optimal neighborhood kernel matrix) from a pre-specified kernel matrix for improved classification performance. We assume that the pre-specified kernel matrix generated from the specific application is a noisy observation of the ideal one. The resulting optimal neighborhood kernel matrix is shown to be the summation of the pre-specified kernel matrix and a rank-one matrix. We formulate the problem of learning the optimal neighborhood kernel as a constrained quartic problem, and propose to solve it using two methods: level method and constrained gradient descent. Empirical results on several benchmark data sets demonstrate the efficiency and effectiveness of the proposed algorithms.

1 Introduction

Kernel methods [Scholköpf and Smola, 2002] work by embedding the data into a high-dimensional (possibly infinite-dimensional) feature space, where the embedding is implicitly specified by a kernel. They have been applied successfully in many areas such as computer vision and bioinformatics [Scholköpf and Smola, 2002; Lanckriet *et al.*, 2004a; 2004b]. The so-called kernel matrix, specified by the inner product of data points in the feature space is symmetric and positive semidefinite. The kernel matrix specifies the geometry of the feature space, and induces a notion of similarity in the sample space. The learning of a good kernel matrix is essential for the successful application of kernel methods. However, it is commonly the case in practice that the "ideal" kernel matrix is unknown and needs to be learnt.

The problem of learning the optimal kernel matrix has been addressed by many researchers. The pioneer work by [Chapelle *et al.*, 2002] learned the parameters of some pre-specified kernel function using gradient descent method. More recent works focus on learning the kernel itself in a nonparametric manner. Lanckriet et al. [2004a; 2004b] proposed the framework of Multiple Kernel Learning (MKL) by representing the learnt kernel matrix as a linear combination of a set of pre-specified kernel matrices, and casted the problem as a semidefinite program or a quadratically-constrained quadratic program. Several efficient methods for accelerating the computation of MKL have been proposed in [Bach *et al.*, 2004; Sonnenburg *et al.*, 2006]. Bousquet and Herrmann [2002] proposed to learn the spectral classes of kernel matrices, and analyzed the complexities of the learnt kernel matrices. Kulis et al. [2006] learned low rank kernel matrices from the given distance and similarity constraints on the data. Luss and Aspremont [2007] proposed to learn a proxy kernel matrix by treating the indefinite kernel matrix as a noisy observation of the positive semidefinite kernel matrix, with the attractive property that the support vectors and the proxy kernel can be simultaneously computed.

In many applications, the kernel matrix generated can be treated as a noisy observation of the ideal one. In this paper, we focus on learning an optimal kernel matrix, called the optimal neighborhood kernel (ONK) matrix, from the pre-specified positive semidefinite kernel matrix for improved classification performance. The ONK matrix is shown to be the summation of the pre-specified kernel matrix and a rank-one matrix. We formulate the problem of learning ONK as a convex quartic problem. Our proposed ONK formulation is different from the neighborhood mismatch kernel proposed in [Weston *et al.*, 2003], where the kernel matrix was constructed by utilizing the neighborhood samples.

We propose two iterative methods for solving the resulting convex quartic problem: level method and constrained gradient descent. Level method is a memory-based method from the cutting plane family [Nemirovski, 1994]. Recently, the cutting plane methods have been successfully applied to a number of convex optimization problems in machine learning, e.g., the bundle method employed in [Teo *et al.*, 2007] for solving regularized risk minimization, and the analytic center cutting plane method [Luss and Aspremont, 2007] for solving the robust classification problem with indefinite kernels. Nemirovski [1994] showed that, level method has a better convergence rate than the Kelley bundle method for solving the general nonsmooth convex optimizations, and achieves better practical performance than methods such as subgradient descent. However, since the level method keeps all

the "prehistory" information, its computational cost can be high. In contrast, gradient descent is a non-memory-based method. To solve the constrained optimizations, we can extend gradient descent to its constrained case using gradient mapping [Nemirovski, 1994]. We show that one of the key steps in constrained gradient descent is the Euclidean projection onto a particular domain, which is a special case of the singly linearly constrained quadratic programs [Dai and Fletcher, 2006]. We cast the Euclidean projection as a zero finding problem, which can be solved in linear time. Empirical results on several benchmark data sets demonstrate the efficiency and effectiveness of the proposed algorithms.

The rest of the paper is organized as follows: we formulate the learning of the optimal neighborhood kernel in Section 2, present the algorithms for solving the convex quartic problem in Sections 3 & 4, report empirical results in Section 5, and conclude the paper in Section 6.

Notations Vectors are represented in bold notation, e.g., $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{x} \leq \mathbf{0}$ is understood componentwise. Matrices are represented in italic script, e.g., $X \in \mathbb{R}^{m \times n}$, and $X \succeq 0$ means that X is square, symmetric, and positive semidefinite. $\|X\|_F$ denotes the *Frobenius* norm of the matrix X, and $\|\mathbf{x}\|_2$ denotes the *Euclidean* norm of the vector $\mathbf{x}$.

2 Learning the Optimal Neighborhood Kernel

Assume that we are given a collection of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$, and $y_i \in \{\pm 1\}$. We consider the SVM formulation in the primal form:

$$\begin{array}{rl} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\limits_{i=1}^n \xi_i \\ \text{subject to} & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, 2, \ldots, n, \end{array} \tag{1}$$

where $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_n]^T$ denotes the vector of slack variables, $\phi(.)$ is a kernel feature mapping, C is the regularization parameter, and $(\mathbf{w}, b)$ determines the hyperplane in the feature space. The dual problem of (1) is given by

$$\begin{array}{rl} \max\limits_{\boldsymbol{\alpha}} & 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T Y K Y \boldsymbol{\alpha} \\ \text{subject to} & \boldsymbol{\alpha} \in P = \{\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{e}, \boldsymbol{\alpha}^T\mathbf{y} = 0\}, \end{array} \tag{2}$$

where $\mathbf{e}$ is a vector containing n 1's, $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$, $Y = \text{diag}(\mathbf{y})$, K is the kernel matrix specified by the kernel $\kappa(.,.)$ as $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$, and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T \in \mathbb{R}^n$ can be solved by quadratic programming. For an unknown test sample $\mathbf{x}$, we compute

$$g(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) + b,$$

and $\mathbf{x}$ is classified as $+1$ if $g(\mathbf{x})$ is positive, and -1 otherwise.

2.1 The Optimal Neighborhood Kernel Matrix

We assume that the pre-specified kernel matrix K is the noisy observation of the "ideal" kernel matrix G. We thus propose to learn the kernel matrix G in the neighborhood of K by solving the following optimization problem:

$$\min_{G \succeq 0} \quad \max_{\boldsymbol{\alpha} \in P} \quad 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T Y G Y \boldsymbol{\alpha} + \rho\|G - K\|_F^2 \tag{3}$$

where the parameter $\rho > 0$ controls the magnitude of the penalty on the square distance between the optimal neighborhood kernel matrix G and the pre-specified kernel matrix K.

It is easy to verify that, the objective function in (3) is convex in G and concave in $\boldsymbol{\alpha}$. Therefore, this is a convex concave optimization problem, and the existence of a saddle point is given by the well-known von Neumann Lemma [Nemirovski, 1994]. Moreover, since the optimization is strictly feasible, the minimization and maximization can be exchanged, resulting in the following optimization problem:

$$\max_{\boldsymbol{\alpha} \in P} \quad \min_{G \succeq 0} \quad 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T Y G Y \boldsymbol{\alpha} + \rho\|G - K\|_F^2. \tag{4}$$

We first show that the optimal neighborhood kernel matrix G that solves the optimization problem in (4) is the summation of the pre-specified kernel matrix K and a rank-one matrix, as summarized in the following theorem:

Theorem 1 *The optimal G^* to the inner minimization problem of (4) can be analytically solved as*

$$G^* = K + \frac{1}{2\rho}(Y\boldsymbol{\alpha})(Y\boldsymbol{\alpha})^T. \tag{5}$$

Proof: The inner minimization problem of (4) can be written as

$$\min_{G \succeq 0} \quad 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T Y G Y \boldsymbol{\alpha} + \rho\|G - K\|_F^2, \tag{6}$$

which is convex in G. We first remove the constraint $G \succeq 0$, and compute the optimal G^* for (6). It is easy to show that the optimal G^* satisfies

$$-(Y\boldsymbol{\alpha})(Y\boldsymbol{\alpha})^T + 2\rho G^* - 2\rho K = 0,$$

which leads to (5). Meanwhile, it is clear that $G^* \succeq 0$. Thus, the resulting G^* is exactly the optimal solution to (6). □

Based on Theorem 1, the optimization problem in (4) can be reformulated as

$$\min_{\boldsymbol{\alpha} \in P} \quad f(\boldsymbol{\alpha}) = -2\boldsymbol{\alpha}^T\mathbf{e} + \boldsymbol{\alpha}^T Y K Y \boldsymbol{\alpha} + \frac{1}{4\rho}(\boldsymbol{\alpha}^T\boldsymbol{\alpha})^2, \tag{7}$$

which is a convex quartic problem. We propose to solve this convex quartic problem by level method in Section 3, and constrained gradient descent in Section 4.

2.2 Discussion

It follows from (5) that when ρ tends to $+\infty$, G^* approaches K, that is, the learnt optimal neighborhood kernel matrix reduces to the pre-specified kernel matrix.

The difference between the optimal neighborhood kernel matrix and the pre-specified kernel matrix is a rank-one matrix $E = \frac{1}{2\rho}(Y\boldsymbol{\alpha})(Y\boldsymbol{\alpha})^T$. Unlike K that is determined by the sample features and the pre-specified kernel $\kappa(.,.)$, E is specified by the computed $\boldsymbol{\alpha}$ and the labels of the samples. In addition, denoting $\tilde{\phi}(\mathbf{x}_i) = (\phi(\mathbf{x}_i), \frac{y_i\alpha_i}{\sqrt{2\rho}})$ for the training samples and $\tilde{\phi}(\mathbf{x}) = (\phi(\mathbf{x}), 0)$ for the sample with unknown label, we can write the learnt optimal neighborhood kernel as $\tilde{\kappa}(\mathbf{x}, \mathbf{z}) = \tilde{\phi}(\mathbf{x})^T\tilde{\phi}(\mathbf{z})$.

3 Optimization by the Level Method

The problem in (7) is convex in $\boldsymbol{\alpha}$ so that a global optimum can be found. In this section, we show how this problem can be solved by the level method, which is a cutting plane method from the bundle family [Nemirovski, 1994].

Given a convex function $f(\boldsymbol{\alpha})$, the basic idea behind a cutting plane method is that, $f(\boldsymbol{\alpha})$ can be lower-bounded by its first-order Taylor approximation, i.e.,

$$f(\boldsymbol{\alpha}) \geq f(\tilde{\boldsymbol{\alpha}}) + (\boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}})^T f'(\tilde{\boldsymbol{\alpha}}), \forall \tilde{\boldsymbol{\alpha}},$$

where $f'(\tilde{\boldsymbol{\alpha}})$ is the derivative of $f(\boldsymbol{\alpha})$ at point $\tilde{\boldsymbol{\alpha}}$. Assume we have obtained a set of $\boldsymbol{\alpha}$'s in the set $\Omega = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_i\}$. We define the following model:

$$f_i(\boldsymbol{\alpha}) = \max_{1 \leq j \leq i} f(\boldsymbol{\alpha}_j) + (\boldsymbol{\alpha} - \boldsymbol{\alpha}_j)^T f'(\boldsymbol{\alpha}_j),$$

which is a piecewise linear convex function, and underestimates the objective as $f_i(\boldsymbol{\alpha}) \leq f(\boldsymbol{\alpha})$.

Given the computed $\boldsymbol{\alpha}$'s in $\Omega = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_i\}$, in the i-th step, the level method calculates $\boldsymbol{\alpha}_{i+1}$ by sequentially performing the following three steps:

1) Compute the upper-bound $\overline{f}_i = \min_{1 \leq j \leq i} f(\boldsymbol{\alpha}_j)$, and the lower-bound $\underline{f}_i = f_i(\mathbf{u}_i)$, where

$$\mathbf{u}_i = \arg\min_{\boldsymbol{\alpha} \in P} f_i(\boldsymbol{\alpha}) \tag{8}$$

can be solved by linear programming. Obviously, $\overline{f}_i$ and $\underline{f}_i$ respectively constitute the upper- and lower- bounds of $\min_{\boldsymbol{\alpha} \in P} f(\boldsymbol{\alpha})$, with the gap given by $\Delta_i = \overline{f}_i - \underline{f}_i$.

2) Calculate the level

$$l_i = (1 - \tau)\underline{f}_i + \tau \overline{f}_i = \underline{f}_i + \tau \Delta_i, \tag{9}$$

whose value is between $\overline{f}_i$ and $\underline{f}_i$, with $0 < \tau < 1$ (usually set to 0.5) controlling the tradeoff. Moreover, we can form the level set $\mathcal{G}_i = \{\boldsymbol{\alpha} | \boldsymbol{\alpha} \in P, f_i(\boldsymbol{\alpha}) \leq l_i\}$.

3) Project $\boldsymbol{\alpha}_i$ to the level set $\mathcal{G}_i$ to get the new solution:

$$\boldsymbol{\alpha}_{i+1} = \pi_{\mathcal{G}_i}(\boldsymbol{\alpha}_i) = \arg\min_{\hat{\boldsymbol{\alpha}} \in \mathcal{G}_i} \frac{1}{2} \|\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}_i\|_2^2, \tag{10}$$

which can be solved by quadratic programming [Boyd and Vandenberghe, 2004].

Algorithm 1 Level Method

Input: $f(.), \varepsilon, \tau, \alpha_0 \in P$
Output: $\boldsymbol{\alpha}$

```
Initialize α_1 = α_0
for i = 1 to ... do
    Compute the lower-bound f_i, the upper-bound f̄_i, and the gap Δ_i
    Calculate the level l_i, and form the level set G_i
    Compute the new solution α_{i+1} = π_{G_i}(α_i)
    if Δ_i ≤ ε then
        α = arg min_{1≤j≤i+1} f(α_j), break
    end if
end for
```

The pseudo code for solving the convex quartic problem by the level method is summarized in Algorithm 1. We have the following convergence property for the level method:

Theorem 2 *[Nemirovski, 1994, Chapter 8, pages 135-137] Let $L(f) < \infty$ be the Lipschitz constant of the convex and Lipschitz continuous function $f(\boldsymbol{\alpha})$ on the convex domain P, i.e., $|f(\boldsymbol{\alpha}) - f(\boldsymbol{\beta})| \leq L(f)\|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2, \boldsymbol{\alpha}, \boldsymbol{\beta} \in P$. Let $D(P)$ be the diameter of the bounded convex domain P, and $c(\tau) = \frac{1}{(1-\tau)^2\tau(2-\tau)}$. Then the gaps Δ_i's converge to zero, and for any $\varepsilon > 0$, if $N \geq c(\tau)L^2(f)D^2(P)\varepsilon^{-2}$, we have $\Delta_N \leq \varepsilon$.*

Theorem 2 shows that the level method has a theoretical complexity estimate $O(\varepsilon^{-2})$, which is better than Kelly bundle method's $O(\varepsilon^{-3})$. Moreover, as pointed out in [Nemirovski, 1994], the level method has a good practical performance, with an empirical $O(n \log(\frac{1}{\varepsilon}))$ complexity.

4 Constrained Gradient Descent with Efficient Euclidean Projections

Although the level method has a good practical convergence property, it needs to solve at each iteration the linear programming problem (8) and the quadratic programming problem (10), whose computational costs increase with an increasing number of constraints during the iterations. In this section, we propose a non-memory-based method, "constrained gradient descent", equipped with an efficient Euclidean projection onto the domain P, with the favorable property that the Euclidean projection can be solved in linear time. Specifically, we introduce the gradient mapping for solving the constrained optimization problem in Section 4.1, cast the Euclidean projection onto the domain P as a zero finding problem in Section 4.2, and present the constrained gradient descent method for solving the problem (7) in Section 4.3.

4.1 Gradient Mapping

To deal with constrained optimization problems, we construct the gradient mapping, which acts a similar role as the gradient in unconstrained optimization. Let $\gamma > 0$. We define

$$f_{\gamma,\mathbf{x}}(\mathbf{y}) = f(\mathbf{x}) + \langle f'(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\gamma}{2}\|\mathbf{y} - \mathbf{x}\|^2.$$

which is the tangent line of $f(.)$ at $\mathbf{x}$, regularized by the squared distance between $\mathbf{y}$ and $\mathbf{x}$. Minimizing $f_{\gamma,\mathbf{x}}(\mathbf{y})$ in the domain P is the problem of Euclidean projections onto P:

$$\begin{aligned} \pi_P(\mathbf{x} - \frac{1}{\gamma}f'(\mathbf{x})) &= \arg\min_{\mathbf{y} \in P} \frac{1}{2}\|\mathbf{y} - (\mathbf{x} - \frac{1}{\gamma}f'(\mathbf{x}))\|^2 \\ &= \arg\min_{\mathbf{y} \in P} f_{\gamma,\mathbf{x}}(\mathbf{y}). \end{aligned} \tag{11}$$

We shall discuss the efficient computation of (11) in the next subsection. We call

$$p(\gamma, \mathbf{x}) = \gamma(\mathbf{x} - \pi_P(\mathbf{x} - \frac{1}{\gamma}f'(\mathbf{x}))) \tag{12}$$

the "gradient mapping" of $f(.)$ on P. From (12), we have

$$\pi_P(\mathbf{x} - \frac{1}{\gamma}f'(\mathbf{x})) = \mathbf{x} - \frac{1}{\gamma}p(\gamma, \mathbf{x}),$$

which shows that, $\pi_P(\mathbf{x} - \frac{1}{\gamma}f'(\mathbf{x}))$ can be viewed as the result of the "gradient" step in the anti-direction of the gradient mapping $p(\gamma, \mathbf{x})$ with stepsize $\frac{1}{\gamma}$.

We say that γ is "appropriate" for $\mathbf{x}$, if

$$f(\pi_P(\mathbf{x} - \frac{1}{\gamma}f'(\mathbf{x}))) \leq f_{\gamma,\mathbf{x}}(\pi_P(\mathbf{x} - \frac{1}{\gamma}f'(\mathbf{x}))). \quad (13)$$

It has been shown in [Nemirovski, 1994] that, γ is always appropriate for any $\mathbf{x} \in P$, if $\gamma \geq L_f$, the Lipschitz gradient of $f(.)$. Moreover, when γ is appropriate for $\mathbf{x}$, $\forall \mathbf{y} \in P$, we have [Nemirovski, 1994, chapter 11, page 174]:

$$f(\mathbf{y}) \geq f(\pi_P(\mathbf{x}-\frac{1}{\gamma}f'(\mathbf{x})))+\langle p(\gamma,\mathbf{x}), \mathbf{y}-\mathbf{x}\rangle+\frac{1}{2\gamma}\|p(\gamma,\mathbf{x})\|^2, \quad (14)$$

which is key to the convergence rate proof.

4.2 Casting Euclidean Projection onto the Domain P as a Zero Finding Problem

In this subsection, we discuss the efficient computation of the Euclidean projection onto $P = \{\mathbf{x}|\mathbf{0} \leq \mathbf{x} \leq C\mathbf{e}, \mathbf{x}^T\mathbf{y} = 0\}$:

$$\pi_P(\mathbf{v}) = \arg\min_{\mathbf{x}\in P} \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2. \quad (15)$$

Our methodology is to cast (15) as a zero finding problem.

Introducing the Lagrangian variables λ, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ respectively for the constraints $\mathbf{x}^T\mathbf{y} = 0$, $\mathbf{0} \leq \mathbf{x}$ and $\mathbf{x} \leq C\mathbf{e}$, we can write the Lagrangian of (15) as

$$L(\mathbf{x}, \lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) = \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2 + \lambda\mathbf{x}^T\mathbf{y} - \boldsymbol{\mu}^T\mathbf{x} + \boldsymbol{\nu}^T(\mathbf{x} - C\mathbf{e}).$$

Let $\mathbf{x}^\star$ be the primal optimal point, and $\lambda^\star$, $\boldsymbol{\mu}^\star$ and $\boldsymbol{\nu}^\star$ be the dual optimal points. We have the following results according to the KKT conditions [Boyd and Vandenberghe, 2004]:

$$0 \leq x_i^\star \leq C, \quad (16)$$

$$x_i^\star = (v_i - \lambda^\star y_i) + (\mu_i^\star - \nu_i^\star), \quad (17)$$

$$\mu_i^\star \geq 0, \nu_i^\star \geq 0, \quad (18)$$

$$\mu_i^\star x_i^\star = 0, \nu_i^\star(x_i^\star - C) = 0, \quad (19)$$

$$\sum_i y_i x_i^\star = 0. \quad (20)$$

From (16)-(19), we have

$$x_i^\star = \max(\min(v_i - \lambda^\star y_i, C), 0). \quad (21)$$

An analysis is given as follows. When $0 < v_i - \lambda^\star y_i < C$, we have $\mu_i^\star = \nu_i^\star = 0$ (which leads to $x_i^\star = v_i - \lambda^\star y_i$), because if $\mu_i^\star \neq 0$, we have $x_i^\star = 0$ from (19), $\nu_i^\star > 0$ from (17) and $x_i^\star = C$ from (19), leading to a contradiction; similar contradiction can be constructed by assuming $\nu_i^\star \neq 0$. When $v_i - \lambda^\star y_i \geq C$, we have $x_i^\star = C$, because if $x_i^\star < C$, we have $\nu_i^\star > 0$ from (17), and $x_i^\star = C$ from (19). Similarly, when $v_i - \lambda^\star y_i \leq 0$, we have $x_i^\star = 0$.

Combining (20) and (21), we have

$$h(\lambda^\star) = \sum_i y_i \max(\min(v_i - \lambda^\star y_i, C), 0) = 0, \quad (22)$$

which shows that $\lambda^\star$ is the root of the function $h(.)$. The following theorem shows that $\lambda^\star$ exists and is unique.

Theorem 3 *Let $y_i \in \{\pm 1\}, i = 1, 2, \ldots, n$ and $\mathbf{y}$ has both negative and positive elements. The auxiliary function $h(.)$ is a continuous and monotonically decreasing function, and it has a unique root.*

Proof: It is easy to verify that, $\forall i$, $h_i(\lambda) = y_i \max(\min(v_i - \lambda y_i, C)$ is continuous and monotonically decreasing in the whole domain $(-\infty, +\infty)$. $\forall i \in \{i|y_i = 1\}$, $h_i(.)$ is strictly decreasing in the interval $[v_i - C, v_i]$; and $\forall i \in \{i|y_i = -1\}$, $h_i(.)$ is strictly decreasing in the interval $[-v_i, C - v_i]$. Denote $\lambda_{11} = \min_{i\in\{i|y_i=1\}} v_i - C$, $\lambda_{12} = \max_{i\in\{i|y_i=1\}} v_i$, $\lambda_{21} = \min_{i\in\{i|y_i=-1\}} -v_i$, $\lambda_{22} = \max_{i\in\{i|y_i=-1\}} -v_i + C$, $\lambda_1 = \min(\lambda_{11}, \lambda_{21})$, and $\lambda_2 = \max(\lambda_{12}, \lambda_{22})$. It is easy to get that $h(.)$ is strictly decreasing in the interval $[\lambda_1, \lambda_2]$. Moreover, $\forall i \in \{i|y_i = 1\}$, we have that, if $\lambda \leq \lambda_{11}$, $h_i(\lambda) = C$; and if $\lambda \geq \lambda_{12}$, $h_i(\lambda_{12}) = 0$. Similarly, $\forall i \in \{i|y_i = -1\}$, we have that, if $\lambda \leq \lambda_{21}$, $h_i(\lambda) = 0$; and if $\lambda \geq \lambda_{22}$, $h_i(\lambda_{12}) = -C$. Therefore, we have $h(\lambda_1) > 0$ and $h(\lambda_2) < 0$. According to the Intermediate Value Theorem, $h(.)$ has a unique root lying in the interval $[\lambda_1, \lambda_2]$. □

To solve the zero finding problem (22), we can make use of bisection, which produces a sequence of intervals of uncertainty with strictly decreasing lengths (decreased by a factor of two). For given $\mathbf{v}$, $\mathbf{y}$ and C, λ_1 and λ_2 in Theorem 3 are fixed, and the number of bisection iterations is at most $\log_2(\frac{\lambda_2-\lambda_1}{\delta})$ under the machine precision δ. Therefore, we have a linear complexity for solving the Euclidean projection problem (15) by bisection. However, the efficiency of bisection is independent of the function $h(.)$, and it cannot be improved even when $h(.)$ is "well-behaved", since bisection only utilizes the signs of $h(.)$ at the two boundary points, but not their values. To improve the efficiency, we follow the work of [Dekker, 1969] in using the interpolation methods that have a better local convergence rate than bisection.

4.3 Constrained Gradient Descent

Constrained Gradient Descent operates in a similar way to the classical gradient descent, and we present the pseudo code in Algorithm 2. In Step 3, we perform a line search for finding γ_k that is appropriate for $\mathbf{x}_k$, according to the Armijo-Goldstein rule [Nemirovski, 1994]. In Step 4, we compute the new approximate solution $\mathbf{x}_{k+1} = \pi_P(\mathbf{x}_k - \frac{1}{\gamma_k}f'(\mathbf{x}_k))$ by the zero finding method proposed in Section 4.2. In Step 5, we terminate the algorithm, once the relative change in the approximate solution $\mathbf{x}_k$ is not larger than the pre-specified parameter tol. Making use of (14) and following the similar technique as the one given in [Nemirovski, 1994, Chapter 10], we can establish the following convergence rate for constrained gradient descent.

Theorem 4 *Let L_f be the Lipschitz gradient of $f(.)$, i.e. $\|f'(\mathbf{x}) - f'(\mathbf{y})\| \leq L_f\|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}, \mathbf{y} \in P$, and let R_f be the Euclidean distance from the starting point $\mathbf{x}_0$ to the optimal solution set of $f(.)$. After N iterations, we have*

$$f(\mathbf{x}_N) - \min_{\mathbf{x}\in P} f(\mathbf{x}) \leq O(1)\frac{L_f R_f^2}{N}. \quad (23)$$

5 Experiments

We conduct experiments on the following UCI data sets[1]: Liver Disorder, Ionosphere, Pima Indians Diabetes, Hill Valley, a2a, and a4a. Table 1 summarizes the characteris-

[1] http://archive.ics.uci.edu/ml/

Algorithm 2 Constrained Gradient Descent

Input: $f(.)$, $\mathbf{x}_0 \in P$, tol
Output: $\mathbf{x}$

1: Initialize $\mathbf{x}_1 = \mathbf{x}_0$
2: **for** $k = 1$ to $\ldots$ **do**
3: Find γ_k that is appropriate for $\mathbf{x}_k$, i.e., satisfying (13)
4: Compute $\mathbf{x}_{k+1} = \pi_P(\mathbf{x}_k - \frac{1}{\gamma_k} f'(\mathbf{x}_k))$
5: **if** $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \leq \text{tol} \times \max(\|\mathbf{x}_k\|, 1)$ **then**
6: $\mathbf{x} = \mathbf{x}_{k+1}$, break
7: **end if**
8: **end for**

Table 1: Summary of benchmark data sets.

data set	dimensionality	sample size
Liver Disorder	6	345
Ionosphere	34	351
Pima Indians Diabetes	8	768
Hill Valley	100	1,212
a2a	123	2,265
a4a	123	4,781

tics of the data sets. In this study, we generate the pre-specified kernel by the radius basis kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\sigma)$, and try three different values for σ: $\sigma_k = 4^{-1+k} \sum_{i=1}^n \sum_{j=1}^n \|\mathbf{x}_i - \mathbf{x}_j\|_2^2/n^2$, $k = 1, 2, 3$. We implement the level method, by using the general solver MOSEK[2] for solving (8) and (10); and implement the constrained gradient descent with the efficient Euclidean projection proposed in Section 4.2. All experiments were carried out on an Intel (R) Core (TM)2 Duo 3.00GHZ processor.

Convergence Evaluation We set $C = 1$, $\rho = 100$, $\sigma = \sigma_2$, and explore the convergence of the proposed level method (Algorithm 1) and constrained gradient descent (Algorithm 2) for learning the optimal neighborhood kernel matrix on data sets: Liver Disorder and a4a. We use all the samples in this task, and thus the learnt kernel matrices are of sizes 345×345 and 4781×4781, respectively. We report the results in Fig. 1, from which we can observe that: (1) both level method and constrained gradient descent converge fast; and (2) level method consumes fewer iterations than constrained gradient descent, due to the usage of the whole "prehistory" for generating the approximate solution.

Time Efficiency We report the computational time for solving (7) on the six data sets (including all the samples) in the left figure of Fig. 2, from which we can observe that: (1) level method works relatively well; and (2) constrained gradient descent is quite efficient. It takes about 0.5 seconds to learn a kernel matrix of size 1212×1212. Compared to the level method, the constrained gradient descent is much more efficient, since each of its iteration takes less time. To explore the efficiency of the Euclidean projection method proposed in Section 4.2, we compare it with the quadratic programming provided in MOSEK, and report the results in the right figure

[2] http://www.mosek.com/

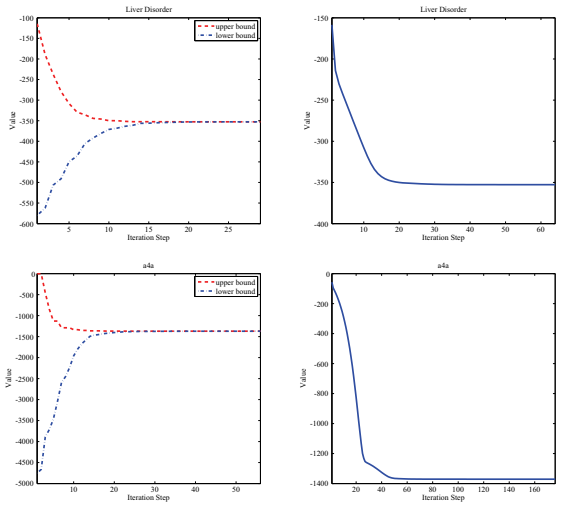

Figure 1: Convergence plots. Figures in the first column correspond to the level method, and figures in the second column correspond to the constrained gradient descent method.

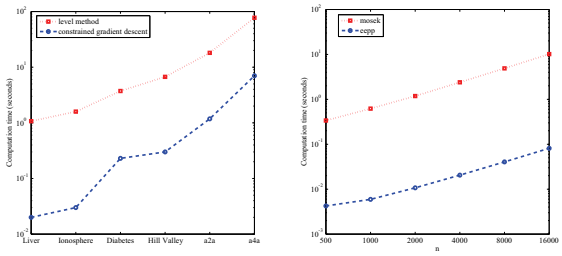

Figure 2: Computational time. The left figure reports the computational time for learning the kernel matrix on the six data sets; and the right figure shows the total computational time for solving 100 independent Euclidean projection problems by the method proposed in Section 4.2 and the quadratic programming provided by MOSEK.

of Fig. 2, where the x-axis corresponds to n, the problem size, and the y-axis denotes the total computational time for solving 100 independent problems. We can observe from the figure that, the proposed Euclidean projection method is about 100 times faster than the quadratic programming of MOSEK for this task.

Classification Performance In exploring the classification performance of the proposed method, we randomly choose 20% from each data set for training, and the remaining for testing, and report the classification performance by averaging over 20 runs. We choose the values of the parameters C and ρ through cross-validation, and report the recognition results in Table 2. We can observe from the table that the learnt optimal neighborhood kernel can yield better classification performance than the pre-specified kernel. In many cases, the improvement is significant. This demonstrates the effectiveness of the proposed optimal neighborhood kernel.

Sensitivity Study We perform the sensitivity study in terms of ρ. Fig. 3 shows the classification performance of the learnt

Table 2: Comparison of the optimal neighborhood kernel (G) and the pre-specified kernel (K) in terms of classification accuracy (%).

	σ_1		σ_2		σ_3	
	G	K	G	K	G	K
Liver	68.5	60.8	69.4	62.8	69.8	67.0
Diabetes	74.1	66.6	75.2	70.6	76.0	73.6
Ionosphere	93.0	91.5	92.1	87.8	89.7	85.8
Hill Valley	63.4	60.1	63.6	58.8	63.0	57.0
a2a	79.6	79.0	81.3	78.0	81.2	76.8
a4a	81.5	80.6	83.2	78.7	83.3	78.1

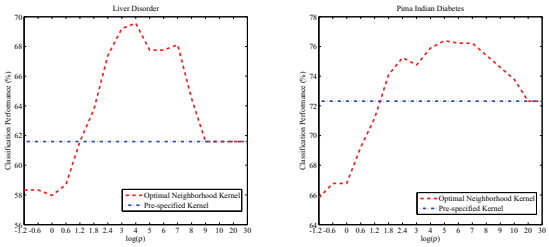

Figure 3: Performance of the learnt optimal neighborhood kernel under different values of ρ.

optimal neighborhood kernel under different ρ's on two data sets: Liver Disorder and Pima Indians Diabetes. We can observe from this figure that: (1) when ρ is quite small, the classification performance is in general not very good, since the rank-one matrix $E = \frac{1}{2\rho}(Y\boldsymbol{\alpha})(Y\boldsymbol{\alpha})^T$ dominates the learnt kernel matrix $G = K + E$; (2) for a relatively wide range of values for ρ, the learnt optimal neighborhood kernel achieves better classification performance than the pre-specified kernel; and (3) when ρ is large, the classification performance of the optimal neighborhood kernel is almost identical to that of the pre-specified kernel, since the effect of E is diminished with a large value of ρ.

6 Conclusion

In this paper, we propose to learn the optimal neighborhood kernel matrix from the pre-specified positive semidefinite kernel matrix for improved classification performance. The optimal neighborhood kernel matrix is shown to be the summation of the pre-specified kernel matrix and a rank-one matrix. We formulate the problem of learning the optimal neighborhood kernel as a convex quartic problem, and propose to solve it by the level method and the constrained gradient descent. To solve the constrained problem, we propose an efficient Euclidean projection onto the domain P. Empirical results on a collection of benchmark data sets demonstrate the efficiency and effectiveness of the proposed algorithms.

We have focused on the learning from a single kernel matrix in this paper, and we plan to extend the current work to multiple kernel learning, where an optimal kernel matrix is learnt from a given collection of kernel matrices. We plan to accelerate the proposed constrained gradient descent by using the Nesterov's method [Nemirovski, 1994], and apply the proposed efficient Euclidean projection to solving other related constrained optimization problems. We also plan to extend the proposed algorithm to the transductive setting.

Acknowledgments

This work was supported by NSF IIS-0612069, IIS-0812551, CCF-0811790, NIH R01-HG002516, NGA HM1582-08-1-0016, NSF of China 60773061, and NSF of Jiangsu Province BK2008381.

References

[Bach *et al.*, 2004] F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Twenty-first International Conference on Machine Learning*. ACM, 2004.

[Bousquet and Herrmann, 2002] O. Bousquet and D. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems*, pages 399–406, 2002.

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

[Chapelle *et al.*, 2002] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46:131–159, 2002.

[Dai and Fletcher, 2006] Y.-H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming*, 106(3):403–421, 2006.

[Dekker, 1969] T.J. Dekker. Finding a zero by means of successive linear interpolation. In B. Dejon and P. Henrici, editors, *Constructive Aspects of the Fundamental Theorem of Algebra*. Wiley Interscience, London, 1969.

[Kulis *et al.*, 2006] B. Kulis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. In *Twentythird International Conference on Machine Learning*, 2006.

[Lanckriet *et al.*, 2004a] G. Lanckriet, T. Bie, N. Christianini, M. Jordan, and W. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[Lanckriet *et al.*, 2004b] G. Lanckriet, N. Christianini, P. Bartlett, L. Ghaoui, and M. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[Luss and Aspremont, 2007] R. Luss and A. Aspremont. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems*, pages 953–960. MIT Press, 2007.

[Nemirovski, 1994] A. Nemirovski. *Efficient methods in convex programming*. Lecture Notes, 1994.

[Scholköpf and Smola, 2002] B. Scholköpf and A. Smola. *Learning with Kernels*. MIT press, 2002.

[Sonnenburg *et al.*, 2006] S. Sonnenburg, G. Ratsch, C. Schafer, and B. Scholkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[Teo *et al.*, 2007] C. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2007.

[Weston *et al.*, 2003] J. Weston, C. Leslie, D. Zhou, A. Elisseeff, and W.S. Noble. Semi-supervised protein classification using cluster kernels. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.