A review of optimization methodologies in support vector machines

John Shawe-Taylor [a] , Shiliang Sun [a,b,*]

[a] *Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom*

[b] *Department of Computer Science and Technology, East China Normal University, 500 Dongchuan Road, Shanghai 200241, China*

Abstract

Support vector machines (SVMs) are theoretically well-justified machine learning techniques, which have also been successfully applied to many real-world domains. The use of optimization methodologies plays a central role in finding solutions of SVMs. This paper reviews representative and state-of-the-art techniques for optimizing the training of SVMs, especially SVMs for classification. The objective of this paper is to provide readers an overview of the basic elements and recent advances for training SVMs and enable them to develop and implement new optimization strategies for SVM-related research at their disposal.

Key words: Decision support systems; Duality; Optimization methodology; Pattern classification; Support vector machine (SVM)

1 Introduction

Support vector machines (SVMs) are state-of-the-art machine learning techniques with their root in structural risk minimization [51,60]. Roughly speaking, by the theory of structural risk minimization, a function from a function class tends to have a low expectation of risk on all the data from an underlying distribution if it has a low empirical risk on a certain data set that is sampled from this distribution and simultaneously the function class has a low complexity, for example, measured by the VC-dimension [61]. The well-known large margin principle in SVMs [8] essentially restricts the complexity of the

* Corresponding author. Tel.: +86-21-54345186; fax: +86-21-54345119.
E-mail address: shiliangsun@gmail.com, slsun@cs.ecnu.edu.cn (S. Sun).

function class, making use of the fact that for some function classes a larger margin corresponds to a lower fat-shattering dimension [51].

Since their invention [8], research on SVMs has exploded both in theory and applications. Recent theory advances in characterizing their generalization errors include Rademacher complexity theory [3,52] and PAC-Bayesian theory [33,39]. In practice, SVMs have been successfully applied to many real-world domains [11]. Moreover, SVMs have been combined with some other research directions, for example, multitask learning, active learning, multi-view learning and semi-supervised learning, and brought forward fruitful approaches [4,15,16,55,56].

The use of optimization methodologies plays an important role in training SVMs. Due to different requirements on the training speed, memory constraint and accuracy of optimization variables, practitioners should choose different optimization methods. It is thus instructive to review the optimization techniques used to train SVMs. Given that SVMs have been blended into the developments of other learning mechanisms, this would also be beneficial to facilitate readers to formulate and solve their own optimization objectives arising from SVM-related research.

Since developments on optimization techniques for SVMs are still active, and SVMs have multiple variants depending on different purposes such as classification, regression and density estimation, it is impractical and unnecessary to include every optimization technique used so far. Therefore, this paper mainly focuses on reviewing representative optimization methodologies used for SVM classification. Being familiar with these techniques is, however, helpful to understand other related optimization methods and implement optimization for other SVM variants.

The rest of this paper is organized as follows. Section 2 introduces the theory of convex optimization, which is closely related to the optimization of SVMs. Section 3 gives the formulation of SVMs for pattern classification, where the kernel trick is also involved. Also, in Section 3, we derive the dual optimization problems and provide optimality conditions for SVMs. They constitute a foundation for the various optimization methodologies detailed in Section 4. Then we briefly describe SVMs for density estimation and regression in Section 5, and introduce optimization techniques used for learning kernels in Section 6. Finally, concluding remarks are given in Section 7.

2 Convex optimization theory

An optimization problem has the form

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\text{s.t. } f_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

Here, the vector $\mathbf{x} \in \mathbb{R}^m$ is the optimization variable, the function $f : \mathbb{R}^m \to \mathbb{R}$ is the objective function, and the functions $f_i : \mathbb{R}^m \to \mathbb{R}$ $(i = 1, \ldots, n)$ are the inequality constraint functions. The domain of this problem is $\mathcal{D} = \mathrm{dom} f \cap_{i=1}^{n} \mathrm{dom} f_i$. A vector from the domain is said to be one element of the feasible set $\mathcal{D}_f$ if and only if the constraints hold on this point. A vector $\mathbf{x}^*$ is called optimal, or the solution of the optimization problem, if its objective value is the smallest among all vectors satisfying the constraints [10]. For now we do not assume the above optimization problem is convex.

The Lagrangian associated with the problem (1) is defined as

$$L(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i f_i(\mathbf{x}), \quad \alpha_i \geq 0, \tag{2}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top$. The scalar α_i is referred to as the Lagrange multiplier associated with the ith constraint. The vector $\boldsymbol{\alpha}$ is called a dual variable or Lagrange multiplier vector. The Lagrange dual function is defined as the infimum of the Lagrangian

$$g(\boldsymbol{\alpha}) = \inf_{\mathbf{x} \in \mathcal{D}_f} L(\mathbf{x}, \boldsymbol{\alpha}). \tag{3}$$

Denote the optimal value of problem (1) by f^*. It can be easily shown that $g(\boldsymbol{\alpha}) \leq f^*$ [10].

With the aim to approach f^*, it is natural to define the following Lagrange dual optimization problem

$$\begin{aligned} &\max\ g(\boldsymbol{\alpha}) \\ &\text{s.t. } \boldsymbol{\alpha} \succeq 0, \end{aligned} \tag{4}$$

where $\boldsymbol{\alpha} \succeq 0$ means $\alpha_i \geq 0$ $(i = 1, \ldots, n)$.

2.1 Optimality conditions for generic optimization problems

KKT (Karush-Kuhn-Tucker) conditions are among the most important sufficient criteria for solving generic optimization problems, which are given in the following theorem [26,31,36,49].

Theorem 1 (KKT saddle point conditions) *Consider a generic optimization problem given by (1) where f and f_i are arbitrary functions. If a pair of variables $(\mathbf{x}^*, \boldsymbol{\alpha}^*)$ exists where $\mathbf{x}^* \in \mathbb{R}^m \cap \mathcal{D}$ and $\boldsymbol{\alpha}^* \succeq 0$, such that for* all *$\mathbf{x} \in \mathbb{R}^m \cap \mathcal{D}$ and $\boldsymbol{\alpha} \in [0, \infty)^n$,*

$$L(\mathbf{x}^*, \boldsymbol{\alpha}) \leq L(\mathbf{x}^*, \boldsymbol{\alpha}^*) \leq L(\mathbf{x}, \boldsymbol{\alpha}^*) \,, \tag{5}$$

then $\mathbf{x}^*$ is a solution to the optimization problem.

One of the important insights derived from (5) is

$$\alpha_i^* f_i(\mathbf{x}^*) = 0, \quad i = 1, \ldots, n, \tag{6}$$

which is usually called complementary slackness conditions. Furthermore, under the assumption that (5) holds, it is straightforward to prove that $L(\mathbf{x}^*, \boldsymbol{\alpha}^*) = f^* = g^*$ with g^* being the optimal value of the dual problem, and $\boldsymbol{\alpha}^*$ is the associated optimal solution. The important step is to show that

$$\max_{\boldsymbol{\alpha} \succeq 0} g(\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha} \succeq 0} \inf_{\mathbf{x} \in \mathcal{D}_f} L(\mathbf{x}, \boldsymbol{\alpha}) \geq \inf_{\mathbf{x} \in \mathcal{D}_f} L(\mathbf{x}, \boldsymbol{\alpha}^*) = L(\mathbf{x}^*, \boldsymbol{\alpha}^*) = f^*. \tag{7}$$

Combining this with $g(\boldsymbol{\alpha}) \leq f^*$ results in the assertion. Now, there is no gap between the optimal values of the primal and dual problems. The KKT-gap $\sum_{i=1}^n \alpha_i f_i(\mathbf{x})$ vanishes when the optimal pair of solutions satisfying (5) are found.

If equality constraints are included in an optimization problem, such as $h(\mathbf{x}) = 0$, we can split it into two inequality constraints $h(\mathbf{x}) \leq 0$ and $-h(\mathbf{x}) \leq 0$ [49]. Suppose the optimization problem is

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\text{s.t. } f_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, n, \\ &\qquad h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, e. \end{aligned} \tag{8}$$

The Lagrangian associated with it is defined as

$$L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i f_i(\mathbf{x}) + \sum_{j=1}^{e} \beta_j h_j(\mathbf{x}), \quad \alpha_i \geq 0,\ \beta_j \in \mathbb{R}. \tag{9}$$

The corresponding KKT saddle point conditions for problem (8) are given by the following theorem [49].

Theorem 2 (KKT saddle point conditions with equality constraints) *Consider a generic optimization problem given by (8) where f, f_i and h_j are*

arbitrary. If a triplet of variables $(\mathbf{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ *exists where* $\mathbf{x}^* \in \mathbb{R}^m \cap \mathcal{D}$, $\boldsymbol{\alpha}^* \succeq 0$, *and* $\boldsymbol{\beta}^* \in \mathbb{R}^e$, *such that for* all $\mathbf{x} \in \mathbb{R}^m \cap \mathcal{D}$, $\boldsymbol{\alpha} \in [0, \infty)^n$ *and* $\boldsymbol{\beta} \in \mathbb{R}^e$,

$$L(\mathbf{x}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) \leq L(\mathbf{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) \leq L(\mathbf{x}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) \,, \tag{10}$$

then $\mathbf{x}^*$ *is a solution to the optimization problem, where* $\mathcal{D}$ *is temporarily reused to represent the domain of problem (8).*

As equality constraints can be equivalently addressed by splitting them to two inequality constraints, we will mainly use the formulation given in (1) for optimization treatment in the sequel.

2.2 Optimality conditions for convex optimization problems

The KKT saddle point conditions are sufficient for any optimization problems. However, it is more desirable to disclose necessary conditions when implementing optimization methodologies. Although it is difficult to provide necessary conditions for generic optimization problems, this is quite possible for some convex optimization problems having nice properties.

Definition 3 (Convex function) *A function* $f : \mathbb{R}^m \to \mathbb{R}$ *is convex if its domain* $domf$ *is a convex set and if for* $0 \leq \theta \leq 1$ *and all* $\mathbf{x}, \mathbf{y} \in domf$, *the following holds*

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}). \tag{11}$$

If the functions f and f_i are all convex functions, then the problem (1) is a convex optimization problem. For a convex optimization problem shown in (1), both the domain $\mathcal{D}$ and the feasible region

$$\mathcal{D}_f = \{\mathbf{x} \in \mathcal{D} \,\text{and}\, f_i(\mathbf{x}) \leq 0 \,(i = 1, \ldots, n)\} \tag{12}$$

can be proved to be convex sets.

The following theorem [36,49] states the nice properties required for the constraints in order to obtain the necessary optimality conditions for convex optimization problems.

Theorem 4 (Constraint qualifications) *Suppose* $\mathcal{D} \in \mathbb{R}^m$ *is a convex domain, and the feasible region* $\mathcal{D}_f$ *is defined by (12) where functions* $f_i : \mathcal{D} \to \mathbb{R}$ *are convex* $(i = 1, \ldots, n)$. *Then the following three qualifications on constraint*

functions f_i are connected by $(i) \Leftrightarrow (ii)$ and $(iii) \Rightarrow (i)$:
(i) There exists an $\mathbf{x} \in \mathcal{D}$ s.t. $f_i(\mathbf{x}) < 0$ for all $i = 1, \ldots, n$ (Slater's condition [54]).
(ii) For all nonzero $\boldsymbol{\alpha} \in [0, \infty)^n$ there exists an $\mathbf{x} \in \mathcal{D}$ s.t. $\sum_{i=1}^{n} \alpha_i f_i(\mathbf{x}) \leq 0$ (Karlin's condition [25]).
(iii) The feasible region $\mathcal{D}_f$ contains at least two distinct points, and there exists an $\mathbf{x} \in \mathcal{D}_f$ s.t. all f_i are strictly convex at $\mathbf{x}$ with respect to $\mathcal{D}_f$ (Strict constraint qualification).

Now we reach the theorem [31,25,49] stating the necessity of KKT conditions.

Theorem 5 (Necessity of KKT conditions) *For a* convex *optimization problem defined by (1), if all the inequality constraints f_is satisfy one of the constraint qualifications of Theorem 4, then the KKT saddle point conditions (5) given in Theorem 1 are necessary for optimality.*

A convex optimization problem is defined as the minimization of a convex function over a convex set. It can be represented as

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\text{s.t. } f_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, n, \\ &\quad a_j^\top \mathbf{x} = b_j, \quad j = 1, \ldots, e, \end{aligned} \tag{13}$$

where the functions f and f_i are convex, and the equality constraints $h_j(\mathbf{x}) = a_j^\top \mathbf{x} - b_j$ are affine [10]. In this case, we can eliminate the equality constraints by updating the domain $\mathcal{D} \cap_{j=1}^{e} (a_j^\top \mathbf{x} = b_j) \rightarrow \mathcal{D}$ and then apply Theorem 5 on a convex problem with no equality constraints.

Slater's condition can be further refined when some of the inequality constraints in the convex problem defined by (1) are affine. The strict feasibility in Slater's condition for these constraints can be relaxed to feasibility [10].

2.3 Optimality conditions for differentiable convex problems

For SVMs, we often encounter convex problems whose objective and constraint functions are further differentiable. Suppose a convex problem (1) satisfies the conditions of Theorem 5. Then we can find its solutions making use of the KKT conditions (5). Under the assumption that the objective and constraint functions are differentiable, the KKT conditions can be simplified as in the following theorem [31,49]. Many optimization methodologies frequently use these conditions.

Theorem 6 (KKT for differentiable convex problems) *Consider a con-*

vex problem (1) whose objective and constraint functions are differentiable at solutions. The vector $\mathbf{x}^*$ *is a solution, if there exists some* $\boldsymbol{\alpha}^* \in \mathbb{R}^n$ *s.t. the following conditions hold:*

$$\boldsymbol{\alpha}^* \succeq 0 \text{ (Required by the Lagrangian)}, \tag{14}$$

$$\partial_{\mathbf{x}} L(\mathbf{x}^*, \boldsymbol{\alpha}^*) = \partial_{\mathbf{x}} f(\mathbf{x}^*) + \sum_{i=1}^{n} \boldsymbol{\alpha}_i^* \partial_{\mathbf{x}} f_i(\mathbf{x}^*) = 0 \text{ (Saddle point at } \mathbf{x}^*), \tag{15}$$

$$\partial_{\alpha_i} L(\mathbf{x}^*, \boldsymbol{\alpha}^*) = f_i(\mathbf{x}^*) \leq 0,\ i = 1, \ldots, n \text{ (Saddle point at } \boldsymbol{\alpha}^*), \tag{16}$$

$$\alpha_i^* f_i(\mathbf{x}^*) = 0,\ i = 1, \ldots, n \text{ (Zero KKT-gap)}. \tag{17}$$

Note that the sign '$\leq$' in (16) is correct as a result of the domain of α_i ($i = 1, \ldots, n$) in the Lagrangian being $[0, \infty)$.

3 SVMs and kernels

In this section, we describe the optimization formulations, dual optimization problems, optimality conditions and related concepts for SVMs for binary classification, which will be shown very useful for various SVM optimization methodologies introduced in later sections. Here after addressing linear classifiers for linearly separable and nonseparable problems, respectively, we then describe nonlinear classifiers with kernels.

Henceforth, we use boldface $\mathbf{x}$ to denote an example vector, and the sign 'n' will be used to denote the number of training examples for consistency with most literature on SVMs. Given a training set $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, -1\}$, the aim of SVMs is to induce a classifier which has good classification performance on future unseen examples.

3.1 Linear classifier for linearly separable problems

When the data set $\mathcal{S}$ is linearly separable, SVMs solve the problem

$$\begin{aligned} &\min_{\mathbf{w},b}\ \Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{s.t. } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n, \end{aligned} \tag{18}$$

where the constraints represent the linearly separable property. The large margin principle is reflected by minimizing $\frac{1}{2}\|\mathbf{w}\|^2$ with $2/\|\mathbf{w}\|$ being the margin

between two lines $\mathbf{w}^\top \mathbf{x} + b = 1$ and $\mathbf{w}^\top \mathbf{x} + b = -1$. The SVM classifier would be

$$c_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b). \tag{19}$$

Problem (18) is a differentiable convex problem with affine constraints, and therefore the constraint qualification is satisfied by the refined Slater's condition. Theorem 6 is suitable to solve the current optimization problem.

The Lagrangian is constructed as

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \lambda_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right], \quad \lambda_i \geq 0, \tag{20}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^\top$ are the associated Lagrange multipliers. Using the superscript star to denote the solutions of the optimization problem, we have

$$\partial_{\mathbf{w}} L(\mathbf{w}^*, b^*, \boldsymbol{\lambda}^*) = \mathbf{w}^* - \sum_{i=1}^{n} \lambda_i^* y_i \mathbf{x}_i = 0, \tag{21}$$

$$\partial_b L(\mathbf{w}^*, b^*, \boldsymbol{\lambda}^*) = -\sum_{i=1}^{n} \lambda_i^* y_i = 0. \tag{22}$$

From (21), the solution $\mathbf{w}^*$ has the form

$$\mathbf{w}^* = \sum_{i=1}^{n} \lambda_i^* y_i \mathbf{x}_i. \tag{23}$$

The training examples for which $\lambda_i^* > 0$ are called support vectors, because other examples with $\lambda_i^* = 0$ can be omitted from the expression.

Substituting (21) and (22) into the Lagrangian results in

$$\begin{aligned} g(\boldsymbol{\lambda}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \lambda_i y_i \mathbf{w}^\top \mathbf{x}_i + \sum_{i=1}^{n} \lambda_i \\ &= \sum_{i=1}^{n} \lambda_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ &= \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j. \end{aligned} \tag{24}$$

Thus the dual optimization problem is

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \; & g(\boldsymbol{\lambda}) = \boldsymbol{\lambda}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\lambda}^\top D \boldsymbol{\lambda} \\ \text{s.t.} \; & \boldsymbol{\lambda}^\top \mathbf{y} = 0, \\ & \boldsymbol{\lambda} \succeq 0, \end{aligned} \tag{25}$$

where $\mathbf{y} = [y_1, \ldots, y_n]^\top$ and D is a symmetric $n \times n$ matrix with entries $D_{ij} = y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$ [43,52].

The zero KKT-gap requirement (also called the complementary slackness condition) implies that

$$\lambda_i^* \left[y_i(\mathbf{x}_i^\top \mathbf{w}^* + b^*) - 1 \right] = 0, \quad i = 1, \ldots, n. \tag{26}$$

Therefore, for all support vectors the constraints are active with equality. The bias b^* can thus be calculated as

$$b^* = y_i - \mathbf{x}_i^\top \mathbf{w}^* \tag{27}$$

using any support vector $\mathbf{x}_i$. With $\boldsymbol{\lambda}^*$ and b^* calculated, the SVM decision function can be represented as

$$c^*(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} y_i \lambda_i^* \mathbf{x}^\top \mathbf{x}_i + b^* \right). \tag{28}$$

3.2 Linear classifier for linearly nonseparable problems

In the case that the data set is not linearly separable but we would still like to learn a linear classifier, a loss on the violation of the linearly separable constraints has to be introduced. A common choice is the hinge loss

$$\max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right), \tag{29}$$

which can be represented using a slack variable ξ_i.

The optimization problem is formulated as

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \; & \Phi(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \; & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{30}$$

where the scalar C controls the balance between the margin and empirical loss. This problem is also a differentiable convex problem with affine constraints. The constraint qualification is satisfied by the refined Slater's condition.

The Lagrangian of problem (30) is

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\gamma}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i\left[y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\gamma_i\xi_i, \quad \lambda_i \geq 0, \gamma_i \geq 0, \tag{31}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^\top$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]^\top$ are the associated Lagrange multipliers. Making use of Theorem 6, we obtain

$$\partial_{\mathbf{w}} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\gamma}^*) = \mathbf{w}^* - \sum_{i=1}^{n}\lambda_i^* y_i \mathbf{x}_i = 0, \tag{32}$$

$$\partial_b L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\gamma}^*) = -\sum_{i=1}^{n}\lambda_i^* y_i = 0, \tag{33}$$

$$\partial_{\xi_i} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\gamma}^*) = C - \lambda_i^* - \gamma_i^* = 0, \quad i = 1, \ldots, n, \tag{34}$$

where (32) and (33) are respectively identical to (21) and (22) for the separable case.

The dual optimization problem is derived as

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \; & g(\boldsymbol{\lambda}) = \boldsymbol{\lambda}^\top\mathbf{1} - \frac{1}{2}\boldsymbol{\lambda}^\top D\boldsymbol{\lambda} \\ \text{s.t.} \; & \boldsymbol{\lambda}^\top\mathbf{y} = 0, \\ & \boldsymbol{\lambda} \succeq 0, \\ & \boldsymbol{\lambda} \preceq C\mathbf{1}, \end{aligned} \tag{35}$$

where $\mathbf{y}$ and D have the same meanings as in problem (25).

The complementary slackness condition requires

$$\begin{aligned} & \lambda_i^*\left[y_i(\mathbf{x}_i^\top\mathbf{w}^* + b^*) - 1 + \xi_i^*\right] = 0, \quad i = 1, \ldots, n, \\ & \gamma_i^*\xi_i^* = 0, \quad i = 1, \ldots, n. \end{aligned} \tag{36}$$

Combining (34) and (36), we can solve $b^* = y_i - \mathbf{x}_i^\top\mathbf{w}^*$ for any support vector $\mathbf{x}_i$ with $0 < \lambda_i^* < C$. The existence of $0 < \lambda_i^* < C$ is equivalent to the assumption that there is at least one support vector with functional output

$y_i(\mathbf{x}_i^\top \mathbf{w}^* + b^*) = 1$ [43]. Usually, this is true. If this assumption is violated, however, other user-defined techniques have to be used to determine the b^*. Once $\boldsymbol{\lambda}^*$ and b^* are solved, the SVM decision function is given by

$$c^*(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} y_i \lambda_i^* \mathbf{x}^\top \mathbf{x}_i + b^*\right). \tag{37}$$

3.3 Nonlinear classifier with kernels

The use of kernel functions (kernels for short) provides a powerful and principled approach to detecting nonlinear relations with a linear method in an appropriate feature space [52]. The design of kernels and linear methods can be decoupled, which largely facilitates modularity of machine learning methods including SVMs.

Definition 7 (Kernel function) *A kernel is a function κ that for all $\mathbf{x}$, $\mathbf{z}$ from a space $\mathscr{X}$ (which need not be a vector space) satisfies*

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \tag{38}$$

where ϕ is a mapping from the space $\mathscr{X}$ to a Hilbert space F that is usually called the feature space

$$\phi : \mathbf{x} \in \mathscr{X} \mapsto \phi(\mathbf{x}) \in F. \tag{39}$$

To verify a function is a kernel, one approach is to construct a feature space for which the function between two inputs corresponds to first performing an explicit feature mapping and then computing the inner product between the images of the inputs. An alternative approach, which is more widely used, is to investigate the finitely positive semidefinite property [52].

Definition 8 (Finitely positive semidefinite function) *A function $\kappa : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ satisfies the finitely positive semidefinite property if it is a* symmetric *function for which the kernel matrices K with $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ formed by restriction to any finite subset of the space $\mathscr{X}$ are positive semidefinite.*

The following theorem [52] justifies the use of the above property for characterizing kernels [2,49].

Theorem 9 (Characterization of kernels) *A function $\kappa : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ which is either continuous or has a countable domain, can be decomposed as an inner product in a Hilbert space F by a feature map ϕ applied to both its*

arguments

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \tag{40}$$

if and only if it satisfies the finitely positive semidefinite property.

Normally, a Hilbert space is defined as an inner product space that is complete. If the separability property is further added to the definition of a Hilbert space, the 'kernels are continuous or the input space is countable' in Theorem 9 is then necessary to address this issue. The Hilbert space constructed in proving Theorem 9 is called the reproducing kernel Hilbert space (RKHS) because of the following reproducing property of the kernel resulting from the defined inner product

$$\langle f_F, \kappa(\mathbf{x}, \cdot) \rangle = f_F(\mathbf{x}), \tag{41}$$

where f_F is a function in the space F of functions, and function $\kappa(\mathbf{x}, \cdot)$ is the mapping $\phi(\mathbf{x})$.

Besides the linear kernel $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$, other commonly used kernels are the polynomial kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\top \mathbf{z})^{deg} \tag{42}$$

where *deg* is the degree of the polynomial, and the Gaussian radial basis function (RBF) kernel (Gaussian kernel for short)

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right). \tag{43}$$

Using the kernel trick, the optimization problem (35) for SVMs becomes

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \; & g(\boldsymbol{\lambda}) = \boldsymbol{\lambda}^\top \mathbf{1} - \frac{1}{2} \boldsymbol{\lambda}^\top D \boldsymbol{\lambda} \\ \text{s.t.} \; & \boldsymbol{\lambda}^\top \mathbf{y} = 0, \\ & \boldsymbol{\lambda} \succeq 0, \\ & \boldsymbol{\lambda} \preceq C\mathbf{1}, \end{aligned} \tag{44}$$

where the entries of D are $D_{ij} = y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The solution for the SVM classifier is formulated as

$$c^*(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} y_i \lambda_i^* \kappa(\mathbf{x}_i, \mathbf{x}) + b^*\right). \tag{45}$$

4 Optimization methodologies for SVMs

For small and moderately sized problems (say with less than $5,000$ examples), interior point algorithms are reliable and accurate optimization methods to choose [49]. For large-scale problems, methods that exploit the sparsity of the dual variables must be adopted; if further limitation on the storage is required, compact representation should be considered, for example, using an approximation for the kernel matrix.

4.1 Interior point algorithms

An interior point is a pair of primal and dual variables which satisfy the primal and dual constraints, respectively. Below we provide the main procedure of interior point algorithms for SVM optimization, given the fundamental importance of this optimization technique.

Suppose now we are focusing on solving an equivalent problem of problem (44)

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \ & \frac{1}{2}\boldsymbol{\lambda}^\top D\boldsymbol{\lambda} - \boldsymbol{\lambda}^\top \mathbf{1} \\ \text{s.t. } & \boldsymbol{\lambda}^\top \mathbf{y} = 0, \\ & \boldsymbol{\lambda} \preceq C\mathbf{1}, \\ & \boldsymbol{\lambda} \succeq 0. \end{aligned} \tag{46}$$

Introducing Lagrange multipliers h, $\mathbf{s}$ and $\mathbf{z}$ where $\mathbf{s}, \mathbf{z} \succeq 0$ and h is free, we can get the Lagrangian

$$\frac{1}{2}\boldsymbol{\lambda}^\top D\boldsymbol{\lambda} - \boldsymbol{\lambda}^\top \mathbf{1} + \mathbf{s}^\top(\boldsymbol{\lambda} - C\mathbf{1}) - \mathbf{z}^\top\boldsymbol{\lambda} + h\boldsymbol{\lambda}^\top \mathbf{y}. \tag{47}$$

The KKT conditions from Theorem 6 are instantiated as

$$\begin{aligned} & D\boldsymbol{\lambda} - \mathbf{1} + \mathbf{s} - \mathbf{z} + h\mathbf{y} = 0 \text{ (Dual feasibility)}, \\ & \boldsymbol{\lambda}^\top \mathbf{y} = 0 \text{ (Primal feasibility)}, \\ & \boldsymbol{\lambda} \preceq C\mathbf{1} \text{ (Primal feasibility)}, \\ & \boldsymbol{\lambda} \succeq 0 \text{ (Primal feasibility)}, \\ & \mathbf{s}^\top(\boldsymbol{\lambda} - C\mathbf{1}) = 0 \text{ (Zero KKT-gap)}, \\ & \mathbf{z}^\top\boldsymbol{\lambda} = 0 \text{ (Zero KKT-gap)}, \\ & \mathbf{s} \succeq 0, \ \mathbf{z} \succeq 0. \end{aligned} \tag{48}$$

Usually, the conditions $\boldsymbol{\lambda} \preceq C\mathbf{1}$ and $\mathbf{s}^\top(\boldsymbol{\lambda} - C\mathbf{1}) = 0$ are transformed to

$$
\begin{aligned}
&\boldsymbol{\lambda} + \boldsymbol{\gamma} = C\mathbf{1}, \\
&\mathbf{s}^\top \boldsymbol{\gamma} = 0, \\
&\boldsymbol{\gamma} \succeq 0.
\end{aligned} \tag{49}
$$

Instead of requiring $\mathbf{z}^\top \boldsymbol{\lambda} = 0$ and $\mathbf{s}^\top \boldsymbol{\gamma} = 0$, according to the primal-dual path-following algorithm [59], they are modified as $z_i \lambda_i = \mu$ and $s_i \gamma_i = \mu$ $(i = 1, \ldots, n)$ for $\mu > 0$. After solving the variables for a certain μ, we then decrease μ to 0. The advantage is that this can facilitate the use of a Newton-type predictor corrector algorithm to update $\boldsymbol{\lambda}, \boldsymbol{\gamma}, h, \mathbf{s}, \mathbf{z}$ [49].

Suppose we already have appropriate initialization for $\boldsymbol{\lambda}, \boldsymbol{\gamma}, h, \mathbf{s}, \mathbf{z}$, and would like to calculate their increments. For this purpose, expanding the equalities in (48) and (49), e.g., substituting $\boldsymbol{\lambda}$ with $\boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}$, yields

$$
\begin{aligned}
&D\Delta\boldsymbol{\lambda} + \Delta\mathbf{s} - \Delta\mathbf{z} + \mathbf{y}\Delta h = -D\boldsymbol{\lambda} + \mathbf{1} - \mathbf{s} + \mathbf{z} - h\mathbf{y} \triangleq \rho_1, \\
&\mathbf{y}^\top \Delta\boldsymbol{\lambda} = -\mathbf{y}^\top \boldsymbol{\lambda}, \\
&\Delta\boldsymbol{\lambda} + \Delta\boldsymbol{\gamma} = C\mathbf{1} - \boldsymbol{\lambda} - \boldsymbol{\gamma} = 0, \\
&\gamma_i^{-1} s_i \Delta\gamma_i + \Delta s_i = \mu\gamma_i^{-1} - s_i - \gamma_i^{-1}\Delta\gamma_i\Delta s_i \triangleq \rho_{kkt1_i},\ i = 1, \ldots, n, \\
&\lambda_i^{-1} z_i \Delta\lambda_i + \Delta z_i = \mu\lambda_i^{-1} - z_i - \lambda_i^{-1}\Delta\lambda_i\Delta z_i \triangleq \rho_{kkt2_i},\ i = 1, \ldots, n.
\end{aligned} \tag{50}
$$

As it is clear that

$$
\begin{aligned}
&\Delta s_i = \rho_{kkt1_i} + \gamma_i^{-1} s_i \Delta\lambda_i, \\
&\Delta z_i = \rho_{kkt2_i} - \lambda_i^{-1} z_i \Delta\lambda_i, \\
&\Delta\boldsymbol{\gamma} = -\Delta\boldsymbol{\lambda},
\end{aligned} \tag{51}
$$

we only need to further solve for $\Delta\boldsymbol{\lambda}$ and Δh. Substituting (51) into the first two equalities of (50), we have

$$
\begin{bmatrix} D + \operatorname{diag}(\boldsymbol{\gamma}^{-1}\mathbf{s}^\top + \boldsymbol{\lambda}^{-1}\mathbf{z}^\top) & \mathbf{y} \\ \mathbf{y}^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\lambda} \\ \Delta h \end{bmatrix} = \begin{bmatrix} \rho_1 - \rho_{kkt1} + \rho_{kkt2} \\ -\mathbf{y}^\top \boldsymbol{\lambda} \end{bmatrix}, \tag{52}
$$

where

$$
\begin{aligned}
&\boldsymbol{\gamma}^{-1} = [1/\gamma_1, \ldots, 1/\gamma_n]^\top, \\
&\boldsymbol{\lambda}^{-1} = [1/\lambda_1, \ldots, 1/\lambda_n]^\top, \\
&\rho_{kkt1} = [\rho_{kkt1_1}, \ldots, \rho_{kkt1_n}]^\top, \\
&\rho_{kkt2} = [\rho_{kkt2_1}, \ldots, \rho_{kkt2_n}]^\top.
\end{aligned} \tag{53}
$$

Then, the Newton-type predictor corrector algorithm, which aims to get a performance similar to higher order methods but without implementing them, is adopted to solve (52) and other variables.

By the blockwise matrix inversion formula with the Schur complement technique, the above linear equation can be readily solved if we can invert $D + \mathrm{diag}(\boldsymbol{\gamma}^{-1}\mathbf{s}^\top + \boldsymbol{\lambda}^{-1}\mathbf{z}^\top)$. The standard Cholesky decomposition, which is numerically extremely stable, is suitable to solve this problem with computational complexity $n^3/6$ [45].

Generally, interior point algorithms are a good choice for small and moderately sized problems with high reliability and precision. Because it involves Cholesky decomposition for a matrix scaled by the number of training examples, the computation is expensive for large-scale data. One way to overcome this is to combine interior point algorithms with sparse greedy matrix approximation to provide a feasible computation of the matrix inverse [49].

4.2 Chunking, sequential minimal optimization

The chunking algorithm [60] relies on the fact that the value of the quadratic form in (46) is unchanged if we remove the rows and columns of D that associate with zero entries of $\boldsymbol{\lambda}$. That is, we need only employ support vectors to represent the final classifier. Thus, the large quadratic program problem can be divided into a series of smaller subproblems.

Chunking starts with a subset (chunk) of training data and solves a small quadratic program (QP) subproblem. Then it retains the support vectors and replace the other data in the chunk with a certain number of data violating KKT conditions, and solves the second subproblem for a new classifier. Each subproblem is initialized with the solutions of the previous subproblem. At the last step, all nonzero entries of $\boldsymbol{\lambda}$ can be identified.

Osuna et al. [42] gave a theorem on the solution convergence of breaking down a large QP problem into a series of smaller QP subproblems. As long as at least one example violating the KKT conditions is added to the examples for the previous subproblem, each step will decrease the objective function and has a feasible point satisfying all constraints [44]. According to this theorem, the chunking algorithm will converge to the globally optimal solution.

The sequential minimal optimization (SMO) algorithm, proposed by Platt [44], is a special case of chunking, which iteratively solves the smallest possible optimization problem each time with two examples

$$\begin{aligned}
\min_{\lambda_i,\lambda_j} \; & \frac{1}{2}\left[\lambda_i^2 D_{ii} + 2\lambda_i\lambda_j D_{ij} + \lambda_j^2 D_{jj}\right] - \lambda_i - \lambda_j \\
\text{s.t. } & \lambda_i y_i + \lambda_j y_j = r_{ij}\,, \\
& 0 \le \lambda_i \le C, \\
& 0 \le \lambda_j \le C,
\end{aligned} \tag{54}$$

where the scaler r_{ij} is computed by fulfilling all the constraints of the full problem $\boldsymbol{\lambda}^\top \mathbf{y} = 0$.

The advantage of SMO is that a QP problem with two example can be solved analytically, and thus a numerical QP solver is avoided. It can be more than 1000 times faster and exhibit better scaling (up to one order better) in the training set size than the classical chunking algorithm given in [60]. Convergence is guaranteed as long as at least one of the two selected examples violates the KKT conditions before solving the subproblem. However, it should be noted that the SMO algorithm will not converge as quickly if solutions with a high accuracy are needed [44]. A more recent approach is the LASVM algorithm [6], which optimizes the cost function of SVMs with a reorganization of the SMO direction searches and can converge to the SVM solution.

There are other decomposition techniques based on the same idea of working set selection where more than two examples are selected. The fast convergence and inexpensive computational cost during each iteration are two conflicting goals. Two widely used decomposition packages are LIBSVM [13] and SVM^{light} [22]. LIBSVM is implemented for working sets of two examples, and exploits the second order information for working set selection. Different from LIBSVM, SVM^{light} can exploit working sets of more than two examples (up to $O(10^3)$) and thus needs a QP solver to solve the subproblems. Recently, following the SVM^{light} framework, Zanni et al. [62] proposed a parallel SVM software with gradient projection QP solvers. Experiments on data sets sized $O(10^6)$ show that training nonlinear SVMs with $O(10^5)$ support vectors can be completed in a few hours with some tens of processors [62].

4.3 Coordinate descent

Coordinate descent is a popular optimization technique that updates one variable at a time by optimizing a subproblem involving only a single variable [21]. It has been discussed and applied to the SVM optimization, e.g., in the work of [7], [18] and [37]. Here we introduce the coordinate descent method used by Hsieh et al. [21] for training large-scale linear SVMs in the dual form, though coordinate descent is surely not confined to linear SVMs.

In some applications where features of examples are high dimensional, SVMs with the linear kernel or nonlinear kernels have similar performances. Further-

more, if the linear kernel is used, much larger data sets can be adopted to train SVMs [21]. Linear SVMs are among the most popular tools to deal with this kind of applications.

By appending each example with a constant feature 1, Hsieh et al. [21] absorbed the bias b into the weight vector $\mathbf{w}$

$$\mathbf{x}^\top = [\mathbf{x}^\top, 1], \quad \mathbf{w}^\top = [\mathbf{w}^\top, b], \tag{55}$$

and solve the following dual problem

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \ & f(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^\top D\boldsymbol{\lambda} - \boldsymbol{\lambda}^\top \mathbf{1} \\ \text{s.t.} \ & \boldsymbol{\lambda} \preceq C\mathbf{1}, \\ & \boldsymbol{\lambda} \succeq 0, \end{aligned} \tag{56}$$

where parameters have the same meaning as in (35).

The optimization process begins with an initial value $\boldsymbol{\lambda}^0 \in \mathbb{R}^n$ and iteratively updates $\boldsymbol{\lambda}$ to generate $\boldsymbol{\lambda}^0, \boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^\infty$. The process from $\boldsymbol{\lambda}^k$ to $\boldsymbol{\lambda}^{k+1}$ is referred to as an outer iteration. During each outer iteration there are n inner iterations which sequentially update $\lambda_1, \ldots, \lambda_n$. Each outer iteration generates multiplier vectors $\boldsymbol{\lambda}^{k,i} \in \mathbb{R}^n$ $(i = 1, \ldots, n+1)$ such that

$$\begin{aligned} &\boldsymbol{\lambda}^{k,1} = \boldsymbol{\lambda}^k, \\ &\boldsymbol{\lambda}^{k,n+1} = \boldsymbol{\lambda}^{k+1}, \\ &\boldsymbol{\lambda}^{k,i} = [\lambda_1^{k+1}, \ldots, \lambda_{i-1}^{k+1}, \lambda_i^k, \ldots, \lambda_n^k]^\top, \ i = 2, \ldots, n. \end{aligned} \tag{57}$$

When we update $\boldsymbol{\lambda}^{k,i}$ to $\boldsymbol{\lambda}^{k,i+1}$, the following one-variable subproblem [21] is solved

$$\begin{aligned} \min_{\delta} \ & f(\boldsymbol{\lambda}^{k,i} + \delta \mathbf{e}_i) \\ \text{s.t.} \ & 0 \le \lambda_i^k + \delta \le C, \end{aligned} \tag{58}$$

where the unit vector $\mathbf{e}_i = [0, \ldots, 0, 1, 0, \ldots, 0]^\top$. The objective function is a quadratic function of the scalar δ

$$f(\boldsymbol{\lambda}^{k,i} + \delta \mathbf{e}_i) = \frac{1}{2} D_{ii}\delta^2 + \nabla_i f(\boldsymbol{\lambda}^{k,i})\delta + \text{const.}, \tag{59}$$

where $\nabla_i f$ is the ith entry of the gradient vector ∇f. For linear SVMs, $\nabla_i f$ can be efficiently calculated and thus speedy convergence of the whole optimization

process can be expected. It was shown that the above coordinate descent method is faster than some state-of-the-art solvers for SVM optimization [21]. Note that, form the point of view of coordinate descent, the SMO algorithm also implements a coordinate descent procedure, though on coordinate pairs.

4.4 Active set methods

Active set methods are among typical approaches to solving QP problems [41]. In active set methods, constraints are divided into the set of active constraints (the active set) and the set of inactive constraints. The algorithm iteratively solves the optimization problem, updating the active set accordingly until the optimal solution is found. For a certain iteration, an appropriate subproblem is solved for the variables not actively constrained.

Active set methods have also been applied to solve the SVM optimization, e.g., by [12], [19], [46], and [47]. In this subsection, we mainly outline the methods used by the two recent papers [46] and [47], which both have theoretical guarantees to converge in a finite time.

The active set method adopted by Scheinberg [46] to solve (46) is to fix variables λ_i in the active set at their current values (0 or C), and then solve the reduced subproblem for each iteration

$$\begin{aligned} \min_{\boldsymbol{\lambda}_s} \ & \frac{1}{2}\boldsymbol{\lambda}_s^\top D_{ss}\boldsymbol{\lambda}_s + \boldsymbol{\lambda}_a^\top D_{as}\boldsymbol{\lambda}_s - \boldsymbol{\lambda}_s^\top \mathbf{1} \\ \text{s.t.} \ & \boldsymbol{\lambda}_s^\top \mathbf{y}_s = -\boldsymbol{\lambda}_a^\top \mathbf{y}_a, \\ & \boldsymbol{\lambda}_s \preceq C\mathbf{1}, \\ & \boldsymbol{\lambda}_s \succeq 0, \end{aligned} \tag{60}$$

where $\boldsymbol{\lambda}_a$ and $\boldsymbol{\lambda}_s$ correspond to variables in and not in the active set, respectively, D_{ss} and D_{as} are composed of the associated rows and columns of D, and the meanings of $\mathbf{y}_a$ and $\mathbf{y}_s$ are similarly defined. Note that the cardinality of the free vector $\boldsymbol{\lambda}_s$ is the number of support vectors on the margin with $0 \prec \boldsymbol{\lambda}_s \prec C\mathbf{1}$ after the last iteration.

Solving the above reduced subproblem can be expensive if the number of the above support vectors is large. Scheinberg [46] proposed to make one step move toward its solution at each iteration. That is, each time either one variable enters or one leaves the active set. The rank-one update of the Cholesky factorization of D_{ss} is therefore efficient for this variant.

The active set method used by Shilton et al. [47] is similar but it tries to solve

a different objective function, which is a partially dual form

$$\max_{b} \min_{0 \preceq \boldsymbol{\lambda} \preceq C\mathbf{1}} \frac{1}{2} \begin{bmatrix} b \\ \boldsymbol{\lambda} \end{bmatrix}^\top \begin{bmatrix} 0 & \mathbf{y}^\top \\ \mathbf{y} & D \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\lambda} \end{bmatrix} - \begin{bmatrix} b \\ \boldsymbol{\lambda} \end{bmatrix}^\top \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \tag{61}$$

where D is the same as in (44). The advantage of using this partially dual representation is that the constraint $\boldsymbol{\lambda}^\top \mathbf{y} = 0$ whose presence complicates the implementation of the active set approach is eliminated [47].

4.5 Solving the primal with Newton's method

Although most literature on SVMs deals with the dual optimization problem, there is also some work concentrating on the optimization of the primal because primal and dual are indeed two sides of the same coin. For example, [27] and [38] studied the primal optimization of linear SVMs, and [14] and [34] studied the primal optimization of both linear and nonlinear SVMs. These works mainly use Newton-type methods. Below we introduce the optimization problem and the approach adopted in [14].

Consider a kernel function κ and an associated reproducing kernel Hilbert space $\mathcal{H}$. The optimization problem is

$$\min_{f \in \mathcal{H}} \; r\|f\|_{\mathcal{H}}^2 + \sum_{i=1}^{n} R(y_i, f(\mathbf{x}_i)), \tag{62}$$

where r is a regularization coefficient, and $R(y, t)$ is a loss function that is differentiable with respect to its second argument [14].

Suppose the optimal solution is f^*. The gradient of the objective function at f^* should vanish. We have

$$2rf^* + \sum_{i=1}^{n} \frac{\partial R}{\partial f}\left(y_i, f^*(\mathbf{x}_i)\right) \kappa(\mathbf{x}_i, \cdot) = 0, \tag{63}$$

where the reproducing property $f(\mathbf{x}_i) = \langle f, \kappa(\mathbf{x}_i, \cdot)\rangle_{\mathcal{H}}$ is used. This indicates that the optimal solution can be represented as a linear combination of kernel functions evaluated at the n training data, which is known as the representer theorem [29].

From (63), we can write $f(\mathbf{x})$ in the form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i \kappa(\mathbf{x}_i, \mathbf{x}), \tag{64}$$

and then solve for $\boldsymbol{\beta} = [\beta_1, \cdots, \beta_n]^\top$. The entries of $\boldsymbol{\beta}$ should not be interpreted as Lagrange multipliers [14].

Recall the definition of the kernel matrix K with $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The optimization problem (62) can be rewritten as the following unconstrained optimization problem

$$\min_{\boldsymbol{\beta}} \; r\boldsymbol{\beta}^\top K \boldsymbol{\beta} + \sum_{i=1}^{n} R(y_i, K_i^\top \boldsymbol{\beta}), \tag{65}$$

where K_i is the ith column of K. As the common hinge loss $R(y_i, f(x_i)) = \max(0, 1 - y_i f(\mathbf{x}_i))$ is not differentiable, Chapelle [14] adopted the Huber loss as a differentiable approximation of the hinge loss. The Huber loss is formulated as

$$R(y,t) = \begin{cases} 0, & \text{if } \; yt > 1 + h \\ \frac{(1+h-yt)^2}{4h}, & \text{if } \; |1 - yt| \leq h \\ 1 - yt, & \text{if } \; yt < 1 - h \end{cases}, \tag{66}$$

where parameter h typically varies between 0.01 and 0.5 [14]. The Newton step of variable update is

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - r_{step} H^{-1} \nabla, \tag{67}$$

where H and ∇ are the Hessian and gradient of the objective function in (65), respectively, and step size r_{step} can be found by line search along the direction of $H^{-1}\nabla$.

The term $-H^{-1}\nabla$ is provided by the standard Newton's method that aims to find an increment $\boldsymbol{\delta}$ to maximize or minimize a function say $L(\boldsymbol{\theta} + \boldsymbol{\delta})$. To show this, setting the derivative of the following second-order Taylor series approximation

$$L(\boldsymbol{\theta} + \boldsymbol{\delta}) = L(\boldsymbol{\theta}) + \boldsymbol{\delta}^\top \nabla(\boldsymbol{\theta}) + \frac{1}{2} \boldsymbol{\delta}^\top H(\boldsymbol{\theta}) \boldsymbol{\delta} \tag{68}$$

to zero, we obtain

$$\boldsymbol{\delta} = -H^{-1}(\boldsymbol{\theta})\nabla(\boldsymbol{\theta}). \tag{69}$$

It was shown that primal and dual optimization are equivalent in terms of the solution and time complexity [14]. However, as primal optimization is more focused on minimizing the desired objective function, perhaps it is superior to the dual optimization when people are interested in finding approximate solutions [14].

Here, we would like to point out an alternative solution approach called the augmented Lagrangian method [5], which has approximately the same convergence speed as the Newton's method.

4.6 Stochastic subgradient with projection

The subgradient method is a simple iterative algorithm for minimizing a convex objective function that can be non-differentiable. The method seems very much like the ordinary gradient method applied to differentiable functions, but with several subtle differences. For instance, the subgradient method is not truly a descent method: the function value often increases. Furthermore, the step lengths in the subgradient method are fixed in advance, instead of found by a line search as in the gradient method [9].

A subgradient of function $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ at $\mathbf{x}$ is any vector $\mathbf{g}$ that satisfies

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^\top(\mathbf{y} - \mathbf{x}) \tag{70}$$

for all $\mathbf{y}$ [53]. When f is differentiable, a very useful choice of $\mathbf{g}$ is $\nabla f(\mathbf{x})$, and then the subgradient method uses the same search direction as the gradient descent method.

Suppose we are minimizing the above function $f(\mathbf{x})$ that is convex. The subgradient method uses the iteration

$$\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - a_k \mathbf{g}^{(k)}, \tag{71}$$

where $\mathbf{g}^{(k)}$ is any subgradient of f at $\mathbf{x}^{(k)}$, and $a_k > 0$ is the step length.

Recently, there has been a lot of interest in studying gradient methods for SVM training [30,63]. Shalev-Shwartz et al. [50] proposed an algorithm alternating between stochastic subgradient descent and projection steps to solve

the primal problem

$$\min_{\mathbf{w}} \frac{r}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} R\left(\mathbf{w}, (\mathbf{x}_i, y_i)\right) \tag{72}$$

where hinge loss $R(\mathbf{w}, (\mathbf{x}, y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}$ with classifier function $f(\mathbf{w}) = \langle \mathbf{w}, \mathbf{x}\rangle$.

The algorithm receives two parameters as input: T - the total number of iterations; k - the number of examples used to calculate subgradients. Initially, weight vector $\mathbf{w}_1$ is set to any vector with norm no larger than $1/\sqrt{r}$. For the tth iteration, a set A_t sized k is chosen from the original n training examples to approximate the objective function as

$$f(\mathbf{w}, A_t) = \frac{r}{2}\|\mathbf{w}\|^2 + \frac{1}{k}\sum_{(\mathbf{x},y)\in A_t} R\left(\mathbf{w}, (\mathbf{x}, y)\right). \tag{73}$$

In general, A_t includes k examples sampled i.i.d from the training set. Define A_t^+ to be the set of examples in A_t for which the hinge loss is nonzero. Then, a two-step update is performed. First, update $\mathbf{w}_t$ to $\mathbf{w}_{t+\frac{1}{2}}$

$$\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \eta_t \nabla_t, \tag{74}$$

where step length $\eta_t = 1/(rt)$, and $\nabla_t = r\mathbf{w}_t - \frac{1}{k}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x}$ is a subgradient of $f(\mathbf{w}, A_t)$ at $\mathbf{w}_t$. Last, $\mathbf{w}_{t+\frac{1}{2}}$ is updated to $\mathbf{w}_{t+1}$ by projecting onto the set

$$\{\mathbf{w} : \|\mathbf{w}\| \leq 1/\sqrt{r}\}, \tag{75}$$

which is shown to contain the optimal solution of the SVM. The algorithm finally outputs $\mathbf{w}_{T+1}$.

If $k = 1$, the algorithm is a variant of the stochastic gradient method; if k is equal to the total number of training examples, the algorithm is the subgradient projection method. Beside simplicity, the algorithm is guaranteed to obtain a solution of accuracy ϵ with $\tilde{O}(1/\epsilon)$ iterations, where an ϵ-accurate solution $\hat{\mathbf{w}}$ is defined as $f(\hat{\mathbf{w}}) \leq \min_{\mathbf{w}} f(\mathbf{w}) + \epsilon$ [50].

4.7 Cutting plane algorithms

The essence of the cutting plane algorithms is to approximate a risk function by one or multiple linear under-estimator (the so-called cutting plane) [28].

Recently, some large-scale SVM solvers using cutting plane algorithms have been proposed. For example, Joachims [23] solved a solution-equivalent structural variant of the following problem

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} R\left(\mathbf{w}, (\mathbf{x}_i, y_i)\right) \tag{76}$$

where hinge loss $R(\mathbf{w}, (\mathbf{x}, y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}$ with classifier function $f(\mathbf{w}) = \langle \mathbf{w}, \mathbf{x}\rangle$. Teo et al. [57] considered a wider range of losses and regularization terms applicable to many pattern analysis problems. Franc [17] improved the cutting plane algorithm used in [23] with line search and monotonicity techniques to solve problem (76) more efficiently. Below we provide a flavor of what the basic cutting plane algorithm does in order to solve problem (76).

At iteration t, a reduced problem of (76) is solved for $\mathbf{w}_t$

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} R_t\left(\mathbf{w}, (\mathbf{x}_i, y_i)\right), \tag{77}$$

where R_t is a linear approximation of R. R_t is derived using the following property. As a result of convexity, for any $(\mathbf{x}_i, y_i)$ function R can be approximated at any point $\mathbf{w}_{t-1}$ by a linear under-estimator [17]

$$R(\mathbf{w}) \geq R(\mathbf{w}_{t-1}) + \langle \mathbf{g}_{t-1}, \mathbf{w} - \mathbf{w}_{t-1}\rangle, \ \forall \mathbf{w} \in \mathbb{R}^d, \tag{78}$$

where $\mathbf{g}_{t-1}$ is any subgradient of R at $\mathbf{w}_{t-1}$. Note that (78) resembles (70), which indicates there can be some similarity between the methods mentioned here and those in Section 4.6. This inequality can be abbreviated as $R(\mathbf{w}) \geq \langle \mathbf{g}_{t-1}, \mathbf{w}\rangle + b_{t-1}$ with b_{t-1} defined as $b_{t-1} = R(\mathbf{w}_{t-1}) - \langle \mathbf{g}_{t-1}, \mathbf{w}_{t-1}\rangle$. The equality $\langle \mathbf{g}_{t-1}, \mathbf{w}\rangle + b_{t-1} = 0$ is called a cutting plane.

For SVM optimization, a subgradient of $R(\mathbf{w}, (\mathbf{x}_i, y_i))$ at $\mathbf{w}_{t-1}$ can be given as $\mathbf{g}_{t-1} = -\pi_i y_i \mathbf{x}_i$ where $\pi_i = 1$ if $y_i\langle \mathbf{w}_{t-1}, \mathbf{x}_i\rangle < 1$ and $\pi_i = 0$ otherwise. Consequently, the term $R_t\left(\mathbf{w}, (\mathbf{x}_i, y_i)\right)$ in (77) can be replaced by a linear relaxation $\langle \mathbf{g}_{t-1}, \mathbf{w}\rangle + b_{t-1}$, and thus a standard QP is recovered which is straightforward to solve. To get a better approximation of the original risk function, more than one cutting planes at different points can be incorporated [17].

5 SVMs for density estimation and regression

In this section, we slightly touch the problem of density estimation and regression using SVMs. Many optimization methodologies introduced in the last sec-

tion can be adopted with some adaptation to deal with the density estimation and regression problems. Thereby, here we do not delve into the optimization details of these problems.

The one-class SVM, an unsupervised learning machine proposed by Schölkopf et al. [48] for density estimation, aims to capture the support of a high-dimensional distribution. It outputs a binary function which is $+1$ in a region containing most of the data, and -1 in the remaining region. The corresponding optimization problem, which can be interpreted as separating the data set from the origin in a feature space, is formulated as

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},\rho} \ & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \\ \text{s.t.} \ & \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle \geq \rho - \xi_i, \quad i = 1, \ldots, n, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{79}$$

where the scalar $\nu \in (0, 1]$ is a balance parameter. The decision function is

$$c(\mathbf{x}) = \operatorname{sign}\left(\langle \mathbf{w}, \phi(\mathbf{x})\rangle - \rho\right). \tag{80}$$

For support vector regression, the labels in the training data are real numbers, that is $y_i \in \mathbb{R}$ $(i = 1, \ldots, n)$. In order to introduce a sparse representation for the decision function, Vapnik [61] devised the following ϵ-insensitive function [49]

$$|y - f(\mathbf{x})|_\epsilon = \max\{0, |y - f(\mathbf{x})| - \epsilon\} \tag{81}$$

with $\epsilon \geq 0$, and applied it to support vector regression.

The standard form of support vector regression is to minimize the objective

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}|y_i - f(\mathbf{x}_i)|_\epsilon, \tag{82}$$

where the positive scalar C reflects the trade-off between function complexity and the empirical loss. An equivalent optimization problem commonly used is given by

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + C\sum_{i=1}^{n}\xi_i^* \\ \text{s.t.} \ & \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b - y_i \leq \epsilon + \xi_i, \\ & y_i - \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle - b \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{83}$$

The decision output for support vector regression is

$$c(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b. \tag{84}$$

6 Optimization methods for kernel learning

SVMs are one of the most famous kernel-based algorithms which encode the similarity information between examples largely by the chosen kernel function or kernel matrix. Determining the kernel functions or kernel matrices is essentially a model selection problem. This section introduces several optimization techniques used for learning kernels in SVM-type applications.

Semidefinite programming is a recent popular class of optimizations with wide engineering applications including kernel learning for SVMs [32,35]. For solving semidefinite programming, there exist interior-point algorithms with good theoretical properties and computational efficiency [58].

A semidefinite program (SDP) is a special convex optimization problem with a linear objective function, and linear matrix inequality and affine equality constraints.

Definition 10 (SDP) *An SDP is an optimization problem of the form*

$$\begin{aligned} &\min_{\mathbf{u}} \ \mathbf{c}^\top \mathbf{u} \\ &s.t. \ F^j(\mathbf{u}) := F_0^j + u_1 F_1^j + \ldots + u_q F_q^j \preceq 0, \quad j = 1, \ldots, L, \\ &\qquad A\mathbf{u} = \mathbf{b}, \end{aligned} \tag{85}$$

where $\mathbf{u} := [u_1, \ldots, u_q]^\top \in \mathbb{R}^q$ *is the decision vector,* $\mathbf{c}$ *is a constant vector defining the linear objective,* F_i^j $(i = 0, \ldots, q)$ *are given symmetric matrices and* $F^j(\mathbf{u}) \preceq 0$ *means that the symmetric matrix* $F^j(\mathbf{u})$ *is negative semidefinite.*

Lanckriet et al. [32] proposed an SDP framework for learning the kernel matrix, which is readily applicable to induction, transduction and even the multiple kernel learning cases. The idea is to wrap the original optimization problems with the following constraints on the symmetric kernel matrix K

$$\begin{aligned} &K = \sum_{i=1}^{m} \mu_i K_i, \\ &K \succeq 0, \\ &\text{trace}(K) \leq c, \end{aligned} \tag{86}$$

or with constraints

$$
\begin{aligned}
&K = \sum_{i=1}^{m} \mu_i K_i, \\
&\mu_i \geq 0, \quad i = 1, \ldots, m, \\
&K \succeq 0, \\
&\text{trace}(K) \leq c,
\end{aligned} \tag{87}
$$

and then formulate the problem in terms of an SDP which is usually solved by making use of interior-point methods. The advantages of the second set of constraints include reducing computational complexity and facilitating the study of the statistical properties of a class of kernel matrices [32]. In particular, for the 1-norm soft margin SVM, the SDP for kernel learning reduces to a quadratically constrained quadratic programming (QCQP) problem. Tools for second-order cone programming (SOCP) can be applied to efficiently solve this problem.

Argyriou et al. [1] proposed a DC (difference of convex functions) programming algorithm for supervised kernel learning. An important difference with other kernel selection methods is that they consider the convex hull of continuously parameterized basic kernels. That is, the number of basic kernels can be infinite. In their formulation, kernels are selected from the set

$$
\mathcal{K}(\mathcal{G}) := \left\{ \int G(\omega) dp(\omega) : p \in \mathcal{P}(\Omega), \omega \in \Omega \right\}, \tag{88}
$$

where Ω is the kernel parameter space, $G(\omega)$ is a basic kernel, and $\mathcal{P}(\Omega)$ is the set of all probability measures on Ω.

Making use of the conjugate function and von Neumann minimax theorem [40], Argyriou et al. [1] formulated an optimization problem which was then addressed by a greedy algorithm. As a subroutine of the algorithm involves optimizing a function which is not convex, they converted it to a difference of two convex functions. With the DC programming techniques [20], the necessary and sufficient condition for finding a solution is obtained, which is then solved by a cutting plane algorithm.

7 Conclusion

We have presented a review of optimization techniques used for training SVMs. The KKT optimality conditions are a milestone in optimization theory. The optimization of many real problems relies heavily on the application of the

conditions. We have shown how to instantiate the KKT conditions for SVMs. Along with the introduction of the SVM algorithms, the characterization of effective kernels has also been presented, which would be helpful to understand the SVMs with nonlinear classifiers.

For the optimization methodologies applied to SVMs, we have reviewed interior point algorithms, chunking and SMO, coordinate descent, active set methods, Newton's method for solving the primal, stochastic subgradient with projection, and cutting plane algorithms. The first four methods focus on the optimization of dual problems, while the last three directly deal with the optimization of the primal. This reflects current developments in SVM training.

Since their invention, SVMs have been extended to multiple variants in order to solve different applications such as the briefly introduced density estimation and regression problems. For kernel learning in SVM-type applications, we further introduced SDP and DC programming, which can be very useful for optimizing problems more complex than the traditional SVMs with fixed kernels.

We believe the optimization techniques introduced in this paper can be applied to other SVM-related research as well. Based on these techniques, people can implement their own solvers for different purposes, such as contributing to one future line of research for training SVMs by developing fast and accurate approximation methods for the challenging problem of large-scale applications. Another direction can be investigating recent promising optimization methods (e.g., [24] and its references) and applying them to solving variants of SVMs for certain machine learning applications.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Project 61075005, the Fundamental Research Funds for the Central Universities, and the PASCAL2 Network of Excellence. This publication only reflects the authors' views.

References

[1] A. Argyriou, R. Hauser, C. Micchelli, M. Pontil, A DC-programming algorithm for kernel selection, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 41–48.

[2] N. Aronszajn, Theory of reproducing kernels, Transactions of the American Mathematical Society 68 (1950) 337–404.

[3] P. Bartlett, S. Mendelson, Rademacher and Gaussian complexities: Risk bounds and structural results, Journal of Machine Learning Research 3 (2002) 463–482.

[4] K. Bennett, A. Demiriz, Semi-supervised support vector machines, Advances in Neural Information Processing Systems 11 (1999) 368–374.

[5] D. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Academic Press, New York, 1982.

[6] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, Journal of Machine Learning Research 6 (2005) 1579–1619.

[7] A. Bordes, L. Bottou, P. Gallinari, J. Weston, Solving multiclass support vector machines with LaRank, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 89–96.

[8] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifier, in: Proceedings of the 5th ACM Worksop on Computational Learning Theory, 1992, pp. 144–152.

[9] S. Boyd, L. Xiao, A. Mutapcic, Subgradient methods, available at: `http://www.stanford.edu/class/ee392o/subgrad_method.pdf`, 2003.

[10] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, England, 2004.

[11] C. Campbell, Kernel methods: A survey of current techniques, Neurocomputing 48 (2002) 63–84.

[12] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, Advances in Neural Information Processing Systems 13 (2001) 409–415.

[13] C. Chang, C. Lin, LIBSVM: A library for support vector machines, available at: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[14] O. Chapelle, Training a support vector machines in the primal, Neural Computation 19 (2007) 1155–1178.

[15] T. Evgeniou, M. Pontil, Regularized multi-task learning, in: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 109–117.

[16] J. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, S. Szedmak, Two view learning: SVM-2K, theory and practice, Advances in Neural Information Processing Systems 18 (2006) 355–362.

[17] V. Franc, S. Sonnenburg, Optimized cutting plane algorithm for support vector machines, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 320–327.

[18] T. Friess, N. Cristianini, C. Campbell, The kernel adatron algorithm: A fast and simple learning procedure for support vector machines, in: Proceedings of the 15th International Conference on Machine Learning, 1998, pp. 188–196.

[19] T. Hastie, S. Rosset, R. Tibshirani, J. Zhu, The entire regularization path, Journal of Machine Learning Research 5 (2004) 1391–1415.

[20] R. Horst, V. Thoai, DC programming: Overview. Journal of Optimization Theory and Applications 103 (1999) 1–41.

[21] C. Hsieh, K. Chang, C. Lin, S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear SVM, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 408–415.

[22] T. Joachims, Making large-scale SVM learning practical, Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, 1998.

[23] T. Joachims, Training linear SVMs in linear time, in: Proceedings of the 12th ACM Conference on Knowledge Discovery and Data Mining, 2006, pp. 217–226.

[24] A. Juditsky, A. Nemirovski, C. Tauvel, Solving variational inequalities with stochastic mirror-prox algorithm, PASCAL EPrints, 2008.

[25] S. Karlin, Mathematical Methods and Theory in Games, Programming, and Economics, Addison Wesley, Reading, MA, 1959.

[26] W. Karush, Minima of Functions of Several Variables with Inequalities as Side Constraints, Master's thesis, Department of Mathematics, University of Chicago, 1939.

[27] S. Keerthi, D. DeCoste, A modified finite Newton method for fast solution of large scale linear SVMs, Journal of Machine Learning Research 6 (2005) 341–361.

[28] J. Kelley, The cutting-plane method for solving convex programs, Journal of the Society for Industrial Applied Mathematics 8 (1960) 703–712.

[29] G. Kimeldorf, G. Wahba, A correspondence between Bayesian estimation on stochastic processes and smoothing by splines, Annals of Mathematical Statistics 41 (1970) 495–502.

[30] J. Kivinen, A. Smola, R. Williamson, Online learning with kernels, IEEE Transactions on Signal Processing 52 (2004) 2165–2176.

[31] H. Kuhn, A. Tucker, Nonlinear programming, in: Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probabilitics, 1951, pp. 481–492.

[32] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M. Jordan, Learning the kernel matrix with semidefinite programming, Journal of Machine Learning Research 5 (2004) 27–72.

[33] J. Langford, J. Shawe-Taylor, PAC-Bayes and Margins, Advances in Neural Information Processing Systems 15 (2002) 439–446.

[34] Y. Lee, O. Mangasarian, SSVM: A smooth support vector machine for classification, Computational Optimization and Applications 20 (2001) 5–22.

[35] J. Li, S. Sun, Nonlinear combination of multiple kernels for support vector machines, in: Proceedings of the 20th International Conference on Pattern Recognition, 2010, pp. 1–4.

[36] O. Mangasarian, Nonlinear Programming, McGraw-Hill, New York, 1969.

[37] O. Mangasarian, D. Musicant, Successive overrelaxation for support vector machines, IEEE Transactions on Neural Networks 10 (1999) 1032–1037.

[38] O. Mangasarian, A finite Newton method for classification, Optimization Methods and Software 17 (2002) 913–929.

[39] D. McAllester, Simplified PAC-Bayesian margin bounds, in: Proceedings of the 16th Annual Conference on Computational Learning Theory, 2003, pp. 203–215.

[40] A. Micchelli, M. Pontil, Learning the kernel function via regularization, Journal of Machine Learning Research 6 (2005) 1099–1125.

[41] J. Nocedal, S. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.

[42] E. Osuna, R. Freund, F. Girosi, An improved training algorithm for support vector machines, in: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, 1997, pp. 276–285.

[43] E. Osuna, R. Freund, F. Girosi, Support vector machines: Training and applications, Technical Report AIM-1602, Massachusetts Institute of Technology, MA, 1997.

[44] J. Platt, Sequential minimal optimization: A fast algorithm for training support vector machines, Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[45] C. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, MIT Press, Cambridge, MA, 2006.

[46] K. Scheinberg, An efficient implementation of an active set method for SVMs, Journal of Machine Learning Research 7 (2006) 2237–2257.

[47] A. Schilton, M. Palaniswami, D. Ralph, A. Tsoi, Incremental training of support vector machines, IEEE Transactions on Neural Networks 16 (2005) 114–131.

[48] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, R. Williamson, Estimating the support of a high-dimensional distribution, Neural Computation 13 (2001) 1443–1471.

[49] B. Schölkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.

[50] S. Shalev-Shwartz, Y. Singer, N. Srebro, Pegasos: Primal estimated sub-gradient solver for SVM, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 807–814.

[51] J. Shawe-Taylor, P. Bartlett, R. Williamson, M. Anthony, Structural risk minimization over data-dependent hieraichies, IEEE Transactions on Information Theory 44 (1998) 1926–1940.

[52] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, UK, 2004.

[53] N. Shor, Minimization Methods for Non-differentiable Functions, Springer-Verlag, Berlin, 1985.

[54] M. Slater, A note on Motzkin's transposition theorem, Econometrica 19 (1951) 185–186.

[55] S. Sun, D. Hardoon, Active learning with extremely sparse labeled examples, Neurocomputing 73 (2010) 2980–2988.

[56] S. Sun, J. Shawe-Taylor, Sparse semi-supervised learning using conjugate functions, Journal of Machine Learning Research 11 (2010) 2423–2455.

[57] C. Teo, Q. Le, A. Smola, S. Vishwanathan, A scalable modular convex solver for regularized risk minimization, in: Proceedings of the 13th ACM Conference on Knowledge Discovery and Data Mining, 2007, pp. 727–736.

[58] L. Vandenberghe, S. Boyd, Semidefinite programming, SIAM Review 38 (1996) 49–95.

[59] R. Vanderbei, LOQO user's manual—version 3.10, Technical Report SOR-97-08, Statistics and Operations Research, Princeton University, 1997.

[60] V. Vapnik, Estimation of Dependences Based on Empirical Data, Springer-Verlag, New York, 1982.

[61] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

[62] L. Zanni, T. Serafini, G. Zanghirati, Parallel software for training large sclae support vector machiens on multiprocessor systems, Journal of Machine Learning Research 7 (2006) 1467–1492.

[63] T. Zhang, Solving large scale linear prediction problems using stochastic gradient descent algorithms, in: Proceedings of the 21st International Conference on Machine Learning, 2004, pp. 919–926.

John Shawe-Taylor is a professor at Department of Computer Science, University College London (UK). His main research area is Sta-

tistical Learning Theory, but his contributions range from Neural Networks, to Machine Learning, to Graph Theory. He obtained a PhD in Mathematics at Royal Holloway, University of London in 1986. He subsequently completed an MSc in the Foundations of Advanced Information Technology at Imperial College. He was promoted to Professor of Computing Science in 1996. He has published over 150 research papers. He moved to the University of Southampton in 2003 to lead the ISIS research group. He has been appointed the Director of the Centre for Computational Statistics and Machine Learning at University College, London from July 2006. He has coordinated a number of European wide projects investigating the theory and practice of Machine Learning, including the NeuroCOLT projects. He is currently the scientific coordinator of a Framework VI Network of Excellence in Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL) involving 57 partners.

Shiliang Sun received the B.E. degree with honors in automatic control from the Department of Automatic Control, Beijing University of Aeronautics and Astronautics in 2002, and the Ph.D. degree with honors in pattern recognition and intelligent systems from the State Key Laboratory of Intelligent Technology and Systems, Department of Automation, Tsinghua University, Beijing, China, in 2007. In 2004, he was entitled Microsoft Fellow. Currently, he is an associate professor at the Department of Computer Science and Technology and the founding director of the Pattern Recognition and Machine Learning Research Group, East China Normal University. From 2009 to 2010, he was a visiting researcher at the Department of Computer Science, University College London, working within the Centre for Computational Statistics and Machine Learning. He is a member of the IEEE, a member of the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) network of excellence, and on the editorial boards of multiple international journals. His research interests include machine learning, pattern recognition, computer vision, natural language processing and intelligent transportation systems.