Reading Report: Binary Separation and Training Support Vector Machines

Peng Wang, Hao Wang
Dept. of Mathematical Sciences, Tsinghua Univ., China
Email: {wang-p12, haowang12}@mails.tsinghua.edu.cn

Abstract

This article acts as a reading report of the paper by Fletcher et al. on binary separating support vector machines. The problem is formulated in linear separating and non-linear cases, and difficulties caused by non-convexity are identified. Two different approaches are described: L1QP (quadratic programming) employs $\mathcal{L}_1$-penalties and incorporate kernel functions naturally; the SQP-like algorithm with factorization techniques solves the primal problem directly and show some merits. Experiments are conducted in comparison, and pros and cons of methods are discussed. The authors include detailed rationale and proof from other materials, and add in their own views, in the hope of providing a user-friendly and self-consistent survey of the topic.

Keywords: support vector machine; binary separation; L1QP method; SQP-like algorithm; quadratic programming

1 Introduction

Support vector machines (SVMs) have been widely utilized in solving a variety of classification problems. In the basic form, it is devised to address the *binary separation problem*, *i.e.* to distinguish two classes of samples in a space from each other. Explicitly, we are required to tune the parameters in the SVM model so that its *decision function* can reliably provide positive and negative values for points on both sides respectively. Once this is done with the *training data-set*, whose labels have been previously designated, the model is utilized in *predicting* the classification of new points.

To begin with, we assume that data points can be perfectly separated with a hyperplane, and the question emerges concerning how to determine an *optimal separating hyperplane* (OSH) that supplies the largest "margin" between the two clusters. When this cannot be made possible, we seek to minimize the distance those *outliers* have to move back towards the OSH. Superficially this does not constitute a novel problem, but due to non-convexity, it can no more be readily solved by convex *quadratic programming*. Some

remedies have been proposed resorting to the dual problem or introduce penalty factors and slack variables, while others attempt to determine the primal solutions directly. Finally, the ingenious application of *kernel tricks* enables us to fit the separating contour with a larger selection of functions, while still employing previous models to solve fundamental problems.

From a historical perspective, the invention of SVM is accredited to Vapnik and Chervonenkis [21] in 1964. It was not until the year of 1995 that the current formulation was proposed by Cortes and Vapnik [3], where kernel tricks [1] were put into use. Intensive inquires with respect to SVM and its variants have been conducted ever since. More prevalent in the field is the *L1QP method* that concentrates on the dual problem and introduces $\mathcal{L}_1$-penalties, yet much effort has also been done to solve the primal directly; for instance, see [13]. Fletcher and Zanghirati [8] proposed an SQP-like algorithm to better solve the *standard problem* in its original form, and applied *partial Cholesky factorization* to work with kernel functions while avoiding some undesirable numerical issues. In addition, myriad contributions have been made concerning other properties of SVMs and their application in practice, which this report just cannot begin to enumerate.

This article is intended to serve as a reading report of the paper *Binary Separation and Training Support Vector Machines* by Fletcher and Zanghirate [8]. While the majority of content in the original paper is covered by presenting interpretations by the authors of this paper, we have also included an abundant selection of relevant materials from various sources. This is done for the purpose of giving an all-round demonstration of the cutting-edge work in the area. Meanwhile, for the purpose of *getting a Tsinghua third-year student comprehending it without additional references*, a narrative style akin to textbooks is adopted. Although the text might have hence been made elongated, and sometimes verbose, this report is fairly informative and inspiring, since we have added our *own* understanding throughout. We described the rationale, albeit somewhat personal, behind how people dealt with many issues, which is rather scarce in the literature. Hopefully this justifies the length of this report.

The rest of the report is organized as follows. The problem gets formulated and fundamental concepts be defined in section 2. Sections 3 and 4 carefully survey the two divergent directions of addressing the problem from both rationalizing and mathematically deducing aspects. In section 5, numerical experiments demonstrate how these algorithms perform in practice and provide further evaluations. Finally, section 6 concludes the report.

2 Problem Formulation and Fundamentals

In this section, we first outline the *binary separation problem*. When the data points given are separable by a hyperplane in the space where they are

situated, a *linear SVM in linearly separable case* is described in section 2.1. Otherwise, in the scenario where a hyperplane is capable of approximately, if not adequately, separating the samples, a generalized *linear SVM* in section 2.2 is employed to address the problem. However, a majority of cases in practice require separation with a hypersurface. Despite this intricate nature, the *non-linear SVM* presented in section 2.3, which involves the apt use of *kernel functions*, reduces them to previous settings.

Suppose we are given a set of points $\boldsymbol{v}_i \in \mathbb{R}^n,\ i = 1, \ldots, m$, each assigned a label $a_i \in \{1, -1\}$[1], which we will later refer to as 'plus' and 'minus' points (whose subscripts are stored in sets $\mathbf{P}$ and $\mathbf{M}$ respectively). This set takes the name of *training set*, in the sense that we are supposed to utilize these pre-labeled points to find a hyperplane, or in the case of non-linear SVM, a hypersurface, such that points with contrary labels lie on different sides of it. This explains the basic problem of *binary separation*. If exact separation is not possible or not desirable, we will search for an optimal solution that minimizes classification errors to a certain degree.

When we say we *train SVMs*, what we are actually doing is tuning parameters for a given model so that they best fit the training sets (and then used for predicting the classification of new point(s) $\boldsymbol{v}$). In the context of SVMs, it is a process of optimization. The main challenge confronting us is, therefore, reducing the problem to a class that we may readily find the global optimum, and this composes the main thread of all the work below.

See section 5 for illustrations of the aforesaid concepts.

2.1 Linear SVM in Linear Separable Case

Definition 1. Suppose the training set is *separable with a hyperplane* $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v} + b = 0,\ \|\boldsymbol{w}\| = 1$, that is, $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v_i} + b \geq 0$ for all $i \in \mathbf{P}$ and $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v_i} + b \leq 0$ for all $i \in \mathbf{N}$. Alternatively, $a_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v_i} + b) \geq 0,\ \forall i$. Let the *functional margin* be the value $h = \min_i a_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v_i} + b)$. When h reaches its maximum over $\forall \boldsymbol{w} \in \mathbb{R}^n,\ b \in \mathbb{R}^n$, the solutions $\boldsymbol{w}_*,\ b_*,\ h_*$ constitute the *optimal separating hyperplane* (OSH). □

To understand the functional margin (our main optimizing goal) geometrically, the following explanation is in place. First, the separating hyperplane lies between the two classes. Then, we move all points along $-a_i\boldsymbol{w}$ (*i.e.* towards the hyperplane and parallel to its normal vector). The least distance moved is exactly h. Of course, we have $h_* \geq 0$.

It is intuitively clear that the magnitude of the functional margin characterizes how "well" the points are separated, so our major concern is max-

[1]Using other labels is totally viable. This is just for convenience of notation, as will be revealed very soon.

imizing h. The following *standard problem* (SP) is formed as a result:

$$\underset{\boldsymbol{w},b,h}{\text{maximize}} \quad h \tag{1}$$

$$\text{subject to} \quad a_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v_i} + b) \geq h, \ \forall i, \tag{2}$$

$$\text{and} \quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} = 1. \tag{3}$$

Equivalently, it can be written in matrix form [8]

$$AV^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{a}b \geq \boldsymbol{e}h, \tag{4}$$

where

$$\boldsymbol{a} = (a_1, \ldots, a_m)^{\mathrm{T}}, \ A = \mathrm{diag}(\boldsymbol{a}), \ V = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_m], \ \boldsymbol{e} = (1, \ldots, 1)^{\mathrm{T}}.$$

Definition 2. The points corresponding to the active constraints in (2), or those closest to the OSH, are named *support vectors*.[2]

2.2 Linear SVM: a General Formulation

If exact separation is not possible, from the geometric perspective, it is natural to obtain the solution by minimizing the distance *outliers* (points that are on the wrong side of separating hyperplane) have to move backwards. Written out, the problem takes the same form as SP, with the exception $h_* < 0$.

Definition 3. Function $f(x) = \mathrm{sign}(\boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{v} + b_*)$ is called the *decision function*, since we may attempt to classify any new points with it, given trained parameters $\boldsymbol{w}_*$ and b_*. □

One might well be tempted to think this is adequate to conclude the formulation of *Linear SVMs*. However, the internal problem is far worse than it appears to be. For the linearly separable case, it will not be abrupt to include the following theorem here.

Theorem 1. Assume $\{\boldsymbol{v}_i\}$ is linearly separable, then SP can be reduced to a convex quadratic programming (QP) problem, for which a global optimum can be readily found [17, p.286].
Proof. We allow $\boldsymbol{w}$ take any non-zero values. Then it is equivalent to rewrite SP as

$$\max_{\boldsymbol{w},b,h} \quad \frac{h}{\|\boldsymbol{w}\|} \tag{5}$$

$$\text{s.t.} \quad AV\frac{\boldsymbol{w}^{\mathrm{T}}}{\|\boldsymbol{w}\|} + \frac{\boldsymbol{a}}{\|\boldsymbol{w}\|}b \geq \frac{\boldsymbol{e}}{\|\boldsymbol{w}\|}h. \tag{6}$$

[2] And (together with classification functions) "they're called *machines* because they generate a binary decision; they're decision machines." [12, p.127]

Notice that the selection of h does not virtually affect the solution of the above equations, since we can multiply all variables without changes in objective (5) or constraint (6). Therefore, it is possible to set $h = 1$. Meanwhile, (5) attains maximum concurrently with the minimum of its reciprocal squared. We have now arrived at a convex QP problem (CQP)

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \tag{7}$$

$$\text{s.t.} \quad AV^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{a}b \geq \boldsymbol{e}. \tag{8}$$

□

Unfortunately, this is not true for settings where $h_* < 0$. Easy to verify, an additional negative sign is inserted in the objective function along the above proof, leaving us with a non-convex QP. In the following sections, some traditional remedies will be demonstrated (section 3), along with the approach adopted by Fletcher and Zanghirati [8] (section 4).

2.3 Non-Linear SVM and Kernel Functions

As is stated above, we have to account for the possibility where a hypersurface is required for satisfactory separation. For example, finding whether a point lies in black or white squares of a chessboard (section 5.2) ideally necessitates separation with non-smooth functions. In fact, linear separation, as literature [8] puts it, is very rare in practice. We do not seek a whole bunch of special functions to fit the data; instead, a non-linear transformation of certain virtue is introduced to convert the problem into a linear one.

Definition 4. If a vector-valued function $\boldsymbol{\phi}(\boldsymbol{v})$ maps $\mathbb{R}^n$ to another (usually of higher dimension) space, referred to as the *feature space*, such that for all $\boldsymbol{t}, \boldsymbol{y} \in \mathbb{R}^n$, and the real function $\mathcal{K}$ satisfies $\mathcal{K}(\boldsymbol{t}, \boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{t})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{y})$, then $\mathcal{K}(\boldsymbol{t}, \boldsymbol{y})$ is called a *kernel function.* □

Definition 5. The decision function of non-linear SVMs is $f(x) = \mathrm{sign}(\boldsymbol{w}_*^{\mathrm{T}}$ $\boldsymbol{\phi}(\boldsymbol{v}) + b_*)$. □

It is illuminating to envision a sphere in the $\mathbb{R}^p$ space with its center at origin (and also in section 5.1). Points inside and outside it are labeled oppositely. If we choose $\boldsymbol{\phi}(\boldsymbol{v}) = \|\boldsymbol{v}\|$, then they are mapped into $\mathbb{R}$ and perfectly separated. However, in practice, most people prefer the direct use of kernel functions $\mathcal{K}$ of their choice for at least three reasons [1, 8, 14]: a) The kernel functions with desired properties can be implicitly factored into dot products but the expression is not yet known. b) The mapping $\boldsymbol{\phi}$ may be infinite in dimension. c) None of the objective and constraints of the *dual problem* (which is a popular approach; see section 3.1) requires more than the dot products of $\boldsymbol{\phi}$, *i.e.* $\mathcal{K}$ is enough.

There has been some classifications on kernel functions. For example, positive-definite kernels and Mercer's kernels [18]. They are well beyond the scope of our main topic, and for now it is sufficient to apply the *Gaussian kernel*

$$\mathcal{K}(\boldsymbol{t}, \boldsymbol{y}) = \exp(-\|\boldsymbol{t} - \boldsymbol{y}\|^2/(2\sigma^2)), \tag{9}$$

where σ impacts the "radius of influence of the Gaussian Kernel" [8], and which is semi-positive definite.

We will defer how kernel functions are used to their appearance in specific methods described below.

3 The Traditional Approach: L1QP Method and Duality

In view of the discussion subsequent to Theorem 1 depicting the difficulties in solving SP in the non-linearly separable case, which are due to non-convexity, the L_1-penalized quadratic programming (L1QP) method was proposed. This is not a novel idea in the field of optimization; see Hesterberg et al. [13] for instance. Although there has been some dissent voices [8], blaming the choice of penalty factor (*a.k.a* the *box constraint*, reason of which uncovered later) for being "artificial" and "a difficult task", it is nonetheless a notably popular remedy for non-convexity.

Exploiting duality is a preferred strategy. Not only does it facilitate computation, but it is indispensable when applying L1QP method to non-linear settings. In section 3.1, the dual problems of SP and L1QP formula are introduced, and we derive KKT conditions to express OSH elegantly (section 3.2). Finally, we examine how alternations should be made in coordination with the kernel method in section 3.3.

3.1 L1QP Formulation and Dual Problem

Where the non-convexity stems is the possibilities of $h_* < 0$ because of incapability of linear separation. This signifies that in (8), the inequality cannot be satisfied for some $\boldsymbol{v}_i$, provided that we plug into (7) and (8) points that are not separable. The basic idea of L1QP method, consequently, is to introduce a slack variable $\boldsymbol{\xi} \geq \boldsymbol{0}$, so that (8) is converted to

$$AV^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{a}b \geq \boldsymbol{e} - \boldsymbol{\xi},\ \boldsymbol{\xi} \geq \boldsymbol{0}. \tag{10}$$

Since $\boldsymbol{\xi}$ is allowed to be arbitrarily large, the implicit h is always non-negative, and we can safely express non-separable cases with the same formula.

The penalty parameter $c > 0$ comes into light here. Intuitively, the larger in magnitude $\boldsymbol{\xi}$, the worse is separation, and L1QP utilizes the $\mathcal{L}_1$

norm for measurement[3]. These two get multiplied ($\mathcal{L}_1$ norm written in dot-product form) and added to the objective, where c represents a sense of trade-off between maximizing margin ($h \propto 1/\|\boldsymbol{w}\|$) and minimizing penalties for slacks. It is a significant parameter in L1QP method that necessitate fine-tuning. Hence the objective function

$$\min_{\boldsymbol{\xi},\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + c\boldsymbol{e}^{\mathrm{T}}\boldsymbol{\xi}. \tag{11}$$

(11) and (10) constitutes the L1QP problem. Next we find out its dual, namely L1QD.

Theorem 2. The dual of the $\mathcal{L}_1$-penalized primal (11) and (10) is

$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}Q\boldsymbol{x} - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} \tag{12}$$

$$\text{s.t.} \quad \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = 0,\ \mathbf{0} \le \boldsymbol{x} \le c\boldsymbol{e}, \tag{13}$$

where $Q = AV^{\mathrm{T}}VA$, and $\boldsymbol{x}$ is multipliers of (10).
Proof. The Lagrangian function is

$$\mathcal{L}(\boldsymbol{w}, b, h, \boldsymbol{x}, \boldsymbol{\pi}) = \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + c\boldsymbol{e}^{\mathrm{T}}\boldsymbol{\xi} - \boldsymbol{x}^{\mathrm{T}}(AV^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e} + \boldsymbol{\xi}) - \boldsymbol{\pi}^{\mathrm{T}}\boldsymbol{\xi}, \tag{14}$$

where $\boldsymbol{x}, \boldsymbol{\pi} \ge \mathbf{0}$. Setting the partial derivatives of $\mathcal{L}$ to zero(s) with respect to primal variables gives

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \mathbf{0} \ \Rightarrow\ \boldsymbol{w} = VA\boldsymbol{x}, \tag{15}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \ \Rightarrow\ \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = 0, \tag{16}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} = \mathbf{0} \ \Rightarrow\ \boldsymbol{x} + \boldsymbol{\pi} = \boldsymbol{e}. \tag{17}$$

The dual is max-min of Lagrangian, and we put stationary conditions (15)-(17) into $\mathcal{L}$ to get its minimum. Note (17) suggests we may drop π. Thus the dual objective (with min replacing max) is expressed as (12). Meanwhile, the non-negativity of multipliers, together with (17) gives the range of $\boldsymbol{x}$: it is confined in a "box" dictated by c. □

In a similar manner, one may easily find the dual, CQD, for the CQP problem.

Theorem 3. The dual of the CQP (7) and (8) is

$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}Q\boldsymbol{x} - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} \tag{18}$$

$$\text{s.t.} \quad \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = 0,\ \boldsymbol{x} \ge \mathbf{0}. \tag{19}$$

□

[3]Other norms are also acceptable ([19] even combines a selection of $\mathcal{L}_p$ norms) but not as much popular.

3.2 KKT Conditions and L1QP Algorithm

KKT conditions for L1QP consist of

1. Stationarity of $\mathcal{L}$, *i.e.* (15)-(17),
2. Feasibility in constraints (10),
3. Complementary slackness[4] $x_i(AV^{\mathrm{T}}\boldsymbol{w}+\boldsymbol{a}b-\boldsymbol{e}+\boldsymbol{\xi})_i=0,\ \pi_i\xi_i=0$, and
4. Non-negative multipliers $\mathbf{0}\leq\boldsymbol{x}\leq c\boldsymbol{e}$.

Theorem 4. Assume a solution $\boldsymbol{x}_*$ is obtained for L1QD (12) and (13), while $\exists j$, s.t. $0<x_j<c$, then the solution to L1QP is obtained with $\boldsymbol{w}_*=VA\boldsymbol{x}_*$ and $b_*=a_j-\boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{v}_j$.
Proof. The first part is obvious. If such a stated j exists, complementary slackness gives $\pi_j>0,\ \xi_j=0$, and then $a_j(\boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{v}_j+b_*)=1$. With the hypothesis $a_j\in\{-1,1\}$, this completes the proof. □

But (15) implies that the condition in this theorem is always satisfied; otherwise, $\boldsymbol{w}_*$ would be an all-zero vector, a contradiction.

All of our efforts so far amount to the *L1QP algorithm*, whose formal description is redundant in this place. In short, we can guarantee to solve L1QD for given data points, which can be realized with any convex-QP solvers, and then we obtain L1QP solutions that defines OSH in the $\mathcal{L}_1$-penalty sense.

Theorem 5. The decision function of L1QP method may be expressed as $f(\boldsymbol{v})=\mathrm{sign}(\sum_{i=1}^{m}(\boldsymbol{x}_*)_i a_i\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v})$, since it is merely definition 5 with substitution and expressed in vector form. □

3.3 Dual-based Non-linear SVMs

It is established in section 2.3 that, as long as only the dot product of data points emerge in our formula, a Gaussian kernel could be employed in the reduction to linear settings. Loosely speaking, what we are actually doing is placing a filter, which is capable of mapping functions to linear ones, before the lens of SVMs: they are unaware of the non-linear transformation. Thus the L1QP algorithm for linear separation might be adequate already[5].

For the training process, we compute a matrix $K=(\mathcal{K}(\boldsymbol{v}_i,\boldsymbol{v}_j))_{m\times m}$, which is positive definite for the Gaussian kernel when all $\boldsymbol{v}_i$'s are distinct.

[4] It seems Fletcher and Zanghirati [8] made a mistake here by setting the dot product to zero.

[5] From a mathematician's perspective, the lingering question is how can we prove the given kernels can possibly approximate arbitrary functions. However, there is still much ambiguity in literature; for instance, see [2, 22].

Then for the objective (12) of L1QD, we may write $Q = AKA$ and the formula remains valid. In the meantime, the decision function is now

$$f(\boldsymbol{v}) = \text{sign}(\sum_{i=1}^{m} (\boldsymbol{x}_*)_i a_i K(\boldsymbol{v}_i, \boldsymbol{v})). \tag{20}$$

Note if we take ϕ as the identity mapping (*i.e.* no non-linear mapping is performed), the kernel function K is exactly dot product, which of course accord with previous notations.

There is not much novelty here, partly due to our lack of knowledge about the properties of kernel functions. Theoretical analysis of their precision or asymptotic behavior is missing, albeit sufficient real-world trials have piled up to suggest that they have rather good properties. However, it is also argued [8] that with the increase of m, the determination of Q is challenging computationally and might incur numerical instability. This is where new approaches, such as the factorization method portrayed below, get their inspiration.

4 Solving the Primal: SQP-like Algorithm and Factorization

While the L1QP and other penalty-based methods provide rapid and relatively simple approximations of what is deemed optimal, it is still concerned that the solutions "may be significantly sub-optimal in comparison to the SP solution" [8]. The choice of the penalty factor c requires repeated tuning. As will be demonstrated in section 5.2, a bad c leads to fairly peculiar behavior due to insufficient or excessive emphasis on the degree of misclassification. Therefore, it might be more desirable to attempt to directly solve the SP.

A new SQP-like algorithm is proposed by Fletcher and Zanghirati [8] for computing the solution of SP, which will be described in section 4.1. It is asserted that, while the non-convexity in cases where linear separation is not feasible cannot be perfectly eschewed, this method guarantees convergence to local optimum, and can at least ensure feasibility of the iterates. Moreover, its authors *conjecture* that it terminates finitely. After that, we will explain how factorization techniques are used in solving the primal in conjunction with kernel methods in section 4.2.

Per recommendation of its authors [8], we enumerate KKT conditions for SP in preparation for the algorithm. The SP is rewritten as

$$\min_{\boldsymbol{w},b,h} \quad -h \tag{21}$$

$$\text{s.t.} \quad AV^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{a}b \geq \boldsymbol{e}h, \tag{22}$$

$$\text{and} \quad \frac{1}{2}(1 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}) = 0. \tag{23}$$

Note the quadratic equation constraint (23) is what disqualifies it from being a (convex) QP. The rationale is analogous to section 3.2 that we may arrive at equations[6]

$$VA\boldsymbol{x} = \pi\boldsymbol{w},\ \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = 0,\ \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} = 1, \tag{24}$$

$$x_i(AV^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}h)_i = 0,\ \forall i,\ \boldsymbol{x} \geq \boldsymbol{0}, \qquad \pi = h. \tag{25}$$

together with the Lagrangian

$$\mathcal{L}(\boldsymbol{w}, b, h, \boldsymbol{x}, \pi) = -h - \boldsymbol{x}^{\mathrm{T}}(\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}h) - \frac{\pi}{2}(1 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}). \tag{26}$$

4.1 The SQP-like Algorithm

For a start, it is allowed to arbitrarily initialize the variable $\boldsymbol{w}_0$, from which other variables, *viz* b_0 and h_0, are calculated to get a feasible solution. This computation comprises a linear programming

$$\min_{b,h} \quad -h \tag{27}$$

$$\text{s.t.} \quad AV^{\mathrm{T}}\boldsymbol{w}_0 + \boldsymbol{a}b \geq \boldsymbol{e}h. \tag{28}$$

Suppose that we have obtained through iteration $\boldsymbol{s}_k = (\boldsymbol{w_k}, b_k, h_k)^{\mathrm{T}}$, the variables, or to say, the *delta*, in the k-th QP, are denoted by $\boldsymbol{d} = (\boldsymbol{d}_{\boldsymbol{w}}^{\mathrm{T}}, d_b, d_h)^{\mathrm{T}}$. The formulation of this QP sub-problem, QP_k, is chiefly similar to the standard SQP method; see [20, chap.18] for details of the latter. The only concern is that QP_k may become unbounded, so we limit it by $\|\boldsymbol{d}_{\boldsymbol{w}}\|_\infty \leq \Delta$ where $\Delta > 0$. Thus QP_k is

$$\min_{\boldsymbol{d}} \quad \frac{1}{2}\pi_k \boldsymbol{d}_w^{\mathrm{T}}\boldsymbol{d}_w - d_h \tag{29}$$

$$\text{s.t.} \quad AV^{\mathrm{T}}(\boldsymbol{w}_k + \boldsymbol{d}_{\boldsymbol{w}}) + \boldsymbol{a}(b_k + d_b) \geq \boldsymbol{e}(h_k + d_h), \tag{30}$$

$$\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{d}_{\boldsymbol{w}} = 0, \tag{31}$$

$$\text{and} \quad \boldsymbol{d}_{\boldsymbol{w}} \leq \boldsymbol{e}\Delta, \tag{32}$$

where $\pi_k = \max\{h_k, 0\}$ to avoid non-convexity in circumstances of $h_k < 0$, while h_k is the standard choice because of (25).

The additional step to the standard SQP method is normalization of $\boldsymbol{s}_k + \boldsymbol{d}$. It is quite (or more than just) a coincidence that, as is well-shown above, scalar product of $\boldsymbol{s}_k$ does not virtually affect anything. As a result, when we set

$$\begin{pmatrix} \boldsymbol{w}_{k+1} \\ b_{k+1} \\ h_{k+1} \end{pmatrix} = \frac{1}{\|\boldsymbol{w}_k + \boldsymbol{d}_{\boldsymbol{w}}\|} \begin{pmatrix} \boldsymbol{w}_k + \boldsymbol{d}_{\boldsymbol{w}} \\ b_k + d_b \\ h_k + d_h \end{pmatrix}, \tag{33}$$

[6] As mentioned above, one of the equations differs from the original.

the new iterates remain feasible: we have still imposed the quadratic equality constraint $\|\boldsymbol{w}\| = 1$ in a sense.

At last, we check $\|\boldsymbol{d}\| < \tau_d$ for some fixed threshold τ_d as a criterion of termination. This concludes the description of a simple form of the algorithm, which the author of *this* article will refer to as the *naïve SQP-like algorithm*.

Fletcher et al. [8] have taken a step further to make some improvements: When QP_k reduces to a linear programming due to $\pi_k = 0$, direct calculation may reduce computational efforts, but this does not contribute to its asymptotic time complexity. In addition, it is proposed a normalized $\boldsymbol{w}_0$ that join the two nearest points of opposite classes should be chosen to initialize the algorithm. This might constitute a better initial estimate.

A more fundamental consideration is whether the algorithm converges (at least to a local optimum).

Theorem 6. The SQP-like algorithm either terminates at the k-th iteration or finds $h_{k+1} > h_k$. □

The detailed proof can be found in the original paper [8, Theorem 4.1], which is too lengthy to elaborate on in this report. However, we *highlight* the fact that even without Theorem 6, the availability of optimum solutions is apparent. Since $\boldsymbol{d} = 0$ is always feasible for QP_k, we can always get a non-negative d_h, hence $h_{k+1} \geq h_k$. This leads to either the convergence of $\{h_k\}$ or the sequence going unbounded, which shall not happen in practice (and which also "converges" to infinity).

Although a conjecture has been made that the SQP-like terminates definitely, no proof or counter-example has been found [8], and from the computational perspective, the use of a small threshold τ_d is still in place. Finally, there are two means of relief to the non-convex scenario[7]:

1. While the algorithm can possibly get trapped in a local minimum providing $\boldsymbol{w}_0$ runs too far, the heuristic initialization mentioned above provides inclination towards the correct solution.[8]

2. Given SP is eventually used in non-linear settings, one could almost always expect linear separation, provided that the kernel function is *expressive*. By this we mean the kernel is capable of finding contours, albeit distorted especially with the presence of data errors, to realize perfect separation. Although this might give rise to more issues (see section 5.4), a convex QP and hence the global solution is secured at any rate.

[7] This indeed is the authors' mere conjecture, too.

[8] But what if one of the nearest points is mislabelled? A multi-start scheme (*i.e.* choose several pairs) might act as a patch.

4.2 Primal Algorithms with Kernel Functions

The motivation of solving the primal based on kernel tricks has been outlined previously. Despite those promising aspects, the main difficulties lie in (5), where function $\boldsymbol{\phi}$ is generally unknown but irreplaceable. Then what shall be done to at least get an approximation? One way is resorting to the *Cholesky factorization*.

It is well known that a (semi)positive-definite matrix K has factors $K = U^{\mathrm{T}}U$, where, for the moment, we assume U is a square matrix. Every real-valued symmetric positive-definite matrix has a unique Cholesky decomposition[11, p.143]. If this is the case, the column vectors of U can be seen as one representation of $\boldsymbol{\phi}(\boldsymbol{v}_i)$'s, namely

$$U \approx [\boldsymbol{\phi}(\boldsymbol{v}_1) \ \cdots \ \boldsymbol{\phi}(\boldsymbol{v}_m)]\,. \tag{34}$$

Then direct solving SP with U replacing V is plausible.

Still to be addressed is how to calculate f in definition 5. If (34) is considered equal, we have[9]

$$U^{\mathrm{T}}\boldsymbol{\theta} = (\mathcal{K}(\boldsymbol{v}_1, \boldsymbol{v}), \ldots, \mathcal{K}(\boldsymbol{v}_m, \boldsymbol{v}))^{\mathrm{T}}, \tag{35}$$

where $\boldsymbol{\theta} = \boldsymbol{\phi}(\boldsymbol{v})$. However, when U cannot accurately restore the representation[10] of the actual $\boldsymbol{\phi}$, it is natural to look for the least-square solution of $\boldsymbol{\theta}$.

Definition 6. The classification function when using Cholesky factorization is $f(\boldsymbol{v}) = \boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{\theta} + b_*$, whose variables are derived above. □

An immediate observation is that $f(\boldsymbol{v}_j)$ is the same as putting it into (20) for all j. For a good K, the above rationale leads to a more inspiring result.

Theorem 7. When K is positive-definite and U non-singular, the value of $f(\boldsymbol{v})$ in definition 6 is identical to that in (20) for all $\boldsymbol{v}$. □

However, the near-singularity of K (for instance, K is deemed singular in MATLAB when $m \sim 100$ in our experiment) as well as computational costs ($O(m^3)$ in time) requires simplification through approximation. The use of *partial Cholesky factorization* [10] traces back to Fine et al. [7]. To understand this, we first consider a variant $K = \bar{U}^{\mathrm{T}}D\bar{U}$, where U is diagonal. Then $U = D^{1/2}\bar{U}$ and will not cause any difficulties. If we ignore the elements in D that are too small, or below a threshold *tol*, and remove the corresponding rows from $\bar{U}$ (and U), we obtain the *partial Cholesky factorization* of K. That is, $K \approx U^{\mathrm{T}}U$ with U full-rank and sized $N \times m$,

[9] Fletcher et al. [8] have also made a typo in this equation, which took the authors of this report several hours of discouraged debugging.

[10] Examples are abundant where a single $\mathcal{K}$ has numerous possible factors $\boldsymbol{\phi}$.

where N depends on *tol*. Such an approximation avoids conditioning issues and accelerates computation ($O(N^2m)$ in time).

Fletcher et al [8] moved on to propose a new algorithm that selects *pivots* (largest elements in D in Gaussian elimination) through iterations. It is neither described nor adopted in this report because a) its merits are found in *speed* more than *accuracy*, b) for larger problems it might lead to ill-conditioning, so the ordinary partial decomposition is adopted again, and c) the detailed formula in their paper has errors in notation unrecoverable for the authors of this report presently.

To sum up, here is a sketch of the primal algorithm based on kernel functions:

1. Specify tolerances τ_d for SQP and *tol* for factorization, together with trust-region-like radius Δ. Choose a kernel function.
2. Calculate K and get its partial Cholesky factorization U with *tol*.
3. Solve SP with $\boldsymbol{a},\ U,\ \tau_d,\ \Delta$ and obtain the trained model $(\boldsymbol{w}_*, b_*)$.
4. Use the trained model, (35), and definition 6 to classify a vector $\boldsymbol{v}$.

5 Numerical Experiments

With all the algorithms investigated closely, we are now able to test the performance of different methods in practical settings. We will set about with a couple of very simple linear and non-linear separation problems. For geometrical convenience, it is preferable that we choose data points scattered in the $\mathbb{R}^2$ plane. As the scheme with which we arrange the data points gets more complicated, comparison is attempted to made among the above methods, while we also play with different parameters to see their effects. After this, we will present results of primitive trials with real-world data.

Algorithms in sections 3 and 4 are implemented in MATLAB. Since the core procedures of the L1QP algorithm consist of essentially no more than a QP, for which built-in solvers are readily available, we have chosen to invoke the fitcsvm and predict functions from the Statistics Toolbox directly. On the other hand, the SQP-like algorithm is coded in detail. Function solve_sp sticks to the description in section 4.1, returning the trained linear SVM model for given training data. This routine is then utilized by solve_nlp_chol and solve_nlp_pchol to work with a kernel function of the user's choice, such as the gaussian_kernel. Recall the latter involves *partial Cholesky factorization*, for which we decide to use the code partchol from Forsgren et al. [9]. In addition, the classifying classify_nlp completes the experiment settings.

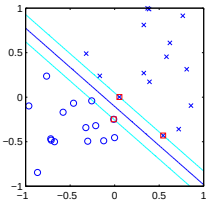

Fig. 1: line separated, SQP-like L-SVM

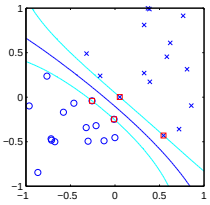

Fig. 2: line separated, NL-SVM with fac.

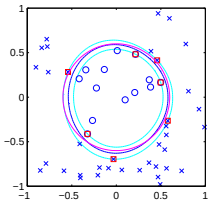

Fig. 3: circle separated, NL-SVM with fac.

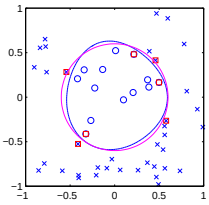

Fig. 4: circle separated, NL-SVM with L1QP.

5.1 A Pilot Study

We begin with some very fundamental examples, where the distribution of data points follows basic patterns. Not only did these sets of experiments assist in the development of our codes, but they ideally enable us to depict some rudimental concepts visually as well.

The simplest case is separable with a straight line, in which points lie in range $[-1, 1]^2$ and on different sides of $x + y = 0$. Figure 1 shows how the SQP-like algorithm for the linear SVM model performs. As can be seen in the plot, the *blue* diagonal line in the middle represents the OSH, alone with *cyan* lines marking the distance of separation, *i.e.* the $\pm h$ contour. We notice that due to absence of points from some corners, the OSH found shall be viewed as fairly good. Three supporting vectors are identified with *red* square markers. In figure 2, the non-linear SVM based on full Cholesky factorization is also applied and, without doubt, separates the points as finely.

In terms of non-linear separation, we visualize the example in section 2.3 where the circle is centered at origin and has a radius of 0.6 (drawn in pink). Figures 3 and 4 depict the resulting separation using factorization-based SVM and L1QP method[11], respectively. Since we have chosen the Gaussian kernel analogous to an exponential function, the separation attained is rather satisfactory and takes only a few supporting vectors.

It is more worthy to point out that in the experiment, we observe that when $m \sim 100$, the built-in function chol for full Cholesky factorization often reports singularity of K. We conjecture that this happens because of ill-conditioning, which again justifies the use of partial decomposition.

5.2 Chessboards

To accord with the original work [8], we have reproduced the chessboard sample problem to demonstrate some properties and comparison between the two methods.

[11] Due to limitations of the SVM package, we are currently unable to plot the $\pm h$ contours of L1QP.

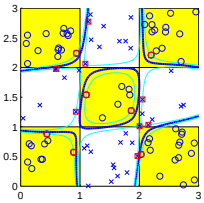

Fig. 5: full chol. fac.

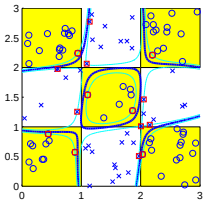

Fig. 6: partchol, w. $\sigma = 1,\ tol = 10^{-5}$

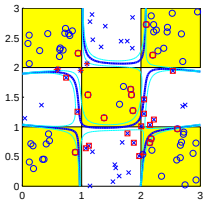

Fig. 7: partchol, w. $\sigma = 4,\ tol = 10^{-5}$

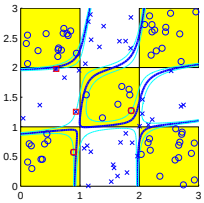

Fig. 8: partchol, w. $\sigma = 1,\ tol = 10^{-2}$

Figure 5, which is derived using full Cholesky decomposition and the SQP-like algorithm, shows a satisfying separation. While the absence of data points in some places might confuse the SVM, the blue contour matches considerably well with the squares. Precisely speaking, the SQP-like algorithm takes 9 iterations to converge below $\tau_d = 10^{-5}$ and 15 support vectors are identified[12]. In addition, to measure the *generalization ability*, which means the efficiency in classifying *unseen* points, a Monte-Carlo-like test is practiced. We evenly select 100×100 points on the plane for classification, and this *test error* in this case is merely 8.68%.

The same data is used to train an SVM with partchol, and figure 6 shows what it produces. After close inspection, it is still indistinguishable from figure 5. We notice that $\text{rank}(U) = 45$ which halves the size of K and it still produces 15 support vectors. The test error is 8.71%, proving that essentially no precision is compromised, if parameters are chosen properly.

However, figures 7 and 8 suggest that this is not always the case. First, we modify the parameter σ in the Gaussian kernel. It is discerned that when σ seem to mismatch the magnitude of the average "distance" among data, strange behaviors tend to emerge. For instance, the test error becomes 9.48% in figure 7 where $\sigma = 4$, and when $\sigma \geq 5$, the zero contours look like two circles. Yet the algorithms are not to blame since it is a kernel's issue. Second, if we enlarge *tol* as in figure 8, leading to less approximation to K, the resulting classification may neither be as good. Conversely, too small a *tol* brings about "large" and ill-conditioning U, which is also troublesome.

Table 1 lists numerical results in figures 5-8, but more experiments were done in reality yet omitted in this report.

Now we turn to analyzing the L1QP algorithm. As stated, the penalty c deserves careful choice, which is perfectly illustrated in figures 9-12.

Data points lying on the wrong side of the zero contour, or the *misclassified*, are marked with a cross. It can be seen in figure 9 that too small a c does not exert enough penalty on misclassifications and therefore encourages maximizing "separation" excessively. In fact, when $c \leq 1$ the zero

[12]It remains a minor problem for us how to reliably find support vectors when solving the primal and where there are truncation errors.

[13]See below.

fac.	m	σ	*tol*	rank(U)	iter.	#SV	miscl.[13]	err.	fig.
full	100	1	10^{-5}	n/a	9	15	0	8.68%	5
part	100	1	10^{-5}	45	8	15	0	8.71%	6
part	100	4	10^{-5}	12	2	25	5	9.48%	7
part	100	1	10^{-2}	20	8	10	0	9.14%	8

Table 1: Results of the SVM with full/partial factorization on the 3×3 chessboard problem.

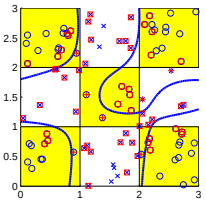

Fig. 9: L1QP, $c = 2$

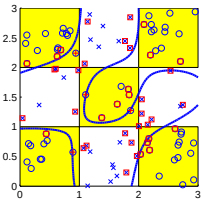

Fig. 10: L1QP, $c = 10$

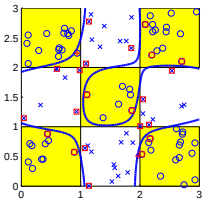

Fig. 11: L1QP, $c = 100$

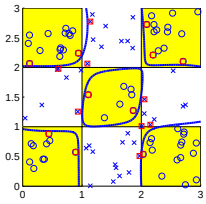

Fig. 12: L1QP, $c = 10^3$

contours make no sense at all. On the other hand, "increasing c gives fewer multipliers meeting their upper bound" [8] (*i.e.* fewer support vectors), so classification is more accurate. Eventually a good separation is attained when $c = 1000$ (see figure 12). We summarize the numerical results in table 2.

c	#SV	miscl.	err.	fig.
2	59	7	19.55%	9
10	39	4	12.66%	10
100	22	2	8.71%	11
1000	18	0	8.25%	12

Table 2: Results of the SVM with L1QP on the 3×3 chessboard problem. In all settings, $m = 100$, $\sigma = 1$.

Next to investigate is how well solve_nlp_pchol deals with larger amounts of (training) data. A much larger 8×8 chessboard is utilized and the number of points m increases to 1000. The parameters are still $\sigma = 1$, $\tau_d = 10^{-5}$, $tol = 10^{-5}$. The resulting separation is visualized[14] in figure 13. Numerically, we observe rank(U) = 213, with approximately 21 iterations required for the SQP-like algorithm, and the test error is 10.45%. Since we did not intend to perform tests with respect to various parameters, and m is not too large compared to the number of squares, this is a fairly desirable outcome. Figure 14 shows tests of the L1QP method for comparison. Its test error is 8.92%.

[14]Not all support vectors are marked.

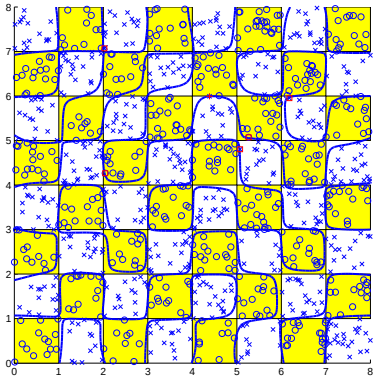

Fig. 13: partchol with 1000 samples

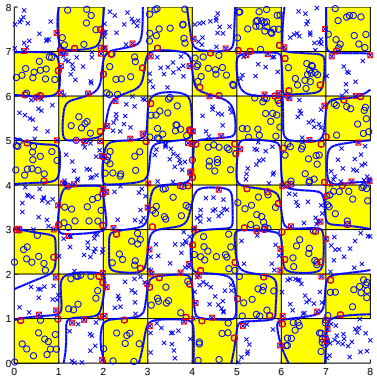

Fig. 14: L1QP with the same data set

It occurred to the authors, at the time of composing this report, that the tolerance of the QP solver quadprog is also of some significance. In Fletcher's paper [8], for instance, his own BQPD code is adopted. We plan to address this aspect in the future.

5.3 MNIST Handwritten Numbers

Some preliminary tests have also been done with *The MNIST database of handwritten digits* [16]. This is a set of 28×28 pixels grey-scale images of digits 0 to 9 written by different people. Since our SVMs are binary classifiers, we simplify the task to distinguishing 1's from 9's.

We pick $m = 400$ images from *each* category, and an additional $t = 160$ *unseen* images to test generalization. As recommended [8], let $\sigma = 1800$ (since the vectors have $28 \times 28 = 784$ dimensions) and $tol = 10^{-3}$. It is observed that matrix U has 312 rows, SQP takes 12 iterations to converge, no misclassification happens and test error is only 1/160. If σ equals 100 and 50000, the test error becomes 100% and 12.5%, respectively.

Since our implements of these SVM variants are for demonstrational purposes, we defer tests on other large-scale data sets. To apply our binary separating SVMs to multi-class classification problems, many different approaches have been developed. See [15, 5] for detail.

5.4 Mislabel and Improvements

What if some samples are *mislabelled* in the training set? This is actually ubiquitous in real-world problems. For example, in most visual object databases (*e.g.* PASCAL VOC [6] and ImageNet [4]) classification was done by humans, so errors are inevitable. However, we will see that the SQP-like algorithm at this stage tends to struggle in order to get perfect separation,

which in turn brings about strange behaviors and reduces generalization abilities.

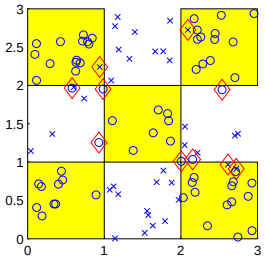

Fig. 15: 10 points are intentionally mislabelled

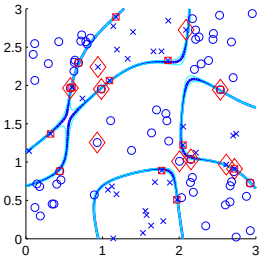

Fig. 16: partchol solution

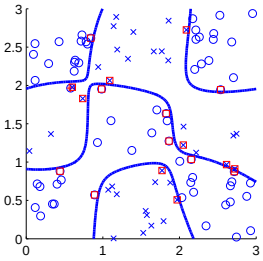

Fig. 17: L1QP solution with $c = 1000$

Some 10 points near the border lines have had their labels intentionally inverted and are marked with red diamonds in figure 15. This simulates mislabeling due to observation errors. With the same parameters as in figure 6, partchol now produces less satisfactory contours in figure 16. Despite its efforts to avoid any misclassification, the curve itself gets rather distorted and separation (h_*) is marginal, since the cyan $\pm h$ contours overlay with the blue zero contours. The test error soars towards 21.20%. In comparison, we deployed the L1QP-SVM as in figure 12. The results in figure 17 appear to be relatively better with 12.34% test error. Only a couple of misclassifications occur as a token to pay, and it could possibly be further tuned with different choices of c.

As a remedy, Fletcher et al. [8] have suggested an auxiliary problem to SP. When too small an h_* is detected, signifying poor separation (with possibly too many support vectors), they propose a *desired margin* $\hat{h} > h_*$ and an $\mathcal{L}_1$ slack variable $\boldsymbol{\xi}$. They then derive the following NLP problem

$$\underset{\boldsymbol{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad \boldsymbol{e}^{\mathrm{T}}\boldsymbol{\xi} \tag{36}$$

$$\text{subject to} \quad AV^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{a}b + \boldsymbol{\xi} \geq \boldsymbol{e}\hat{h}, \tag{37}$$

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} = 1, \tag{38}$$

$$\text{and} \quad \boldsymbol{\xi} \geq \boldsymbol{0}, \tag{39}$$

which is the most elegant solution we can currently devise.

6 Conclusion

In this report, we give a close examination over the problem of binary separation with support vector machines. The standard formulation of separation in linearly separable, non-separable, and non-linearly separating settings are

derived. Once exact separation is not possible or not preferable, the problem instant becomes a non-convex quadratic programming which is difficult to solve. The traditional approach introduces the use of $\mathcal{L}_1$-penalties and relies on its dual problem to construct a convex QP, also known as the L1QP method. However, this treatment has incurred some issues, and direct solution of the primal draws the interest of many. We describe an SQP-like algorithm proposed by Fletcher et al. [8] that facilitates the attainment of solution. For non-linear problems, kernel functions are used to readily reduce it to a previous one. It can be naturally incorporated into dual-based methods without difficulties, but primal ones require further consideration. For this purpose, (partial) Cholesky factorization is utilized and some work [8] focus on further improving computational costs and conditioning.

After these detailed description, we implement the two approaches in MATLAB to examine their performance through a series of experiments. While the chessboard and MNIST problem both demonstrate that the SQP-NLSVM with partchol often outperform L1QP, it is noticed that in the presence of mislabelled samples, the former may run into difficulties. The reason of this phenomenon is tentatively analysed, and we present the recommendations of Fletcher et al. [8] for remedy.

What distinguishes this reading report is the detailed rationale and proof added to the original paper, in addition to some valuable insights from the authors' own experience. It is our hope that the Tsinghua student have by now grasped the rudimental theory of SVM in this report.

References

[1] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[2] B Caputo, K Sim, F Furesjo, and A Smola. Appearance-based object recognition using svms: Which kernel should i use? In *Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler*, volume 2002, 2002.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[5] Kai-Bo Duan and S Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *Multiple Classifier Systems*, pages 278–285. Springer, 2005.

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[7] Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2:243–264, 2002.

[8] Roger Fletcher and Gaetano Zanghirati. Binary separation and training support vector machines. *Acta Numerica*, 19:121–158, 2010.

[9] Anders Forsgren, Philip E Gill, and Walter Murray. Computing modified newton directions using a partial cholesky factorization. *SIAM Journal on Scientific Computing*, 16(1):139–150, 1995.

[10] Donald Goldfarb and Katya Scheinberg. A product-form cholesky factorization method for handling dense columns in interior point methods for linear programming. *Mathematical Programming*, 99(1):1–34, 2004.

[11] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU Press, 1996.

[12] Peter Harrington. *Machine learning in action*. Manning, 2012.

[13] Tim Hesterberg, Nam Hee Choi, Lukas Meier, Chris Fraley, et al. Least angle and l1 penalized regression: A review. *Statistics Surveys*, 2:61–93, 2008.

[14] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[15] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[16] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[17] David G Luenberger and Yinyu Ye. *Linear and nonlinear programming*, volume 116. Springer Science & Business Media, 2008.

[18] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions*

of the royal society of London. Series A, containing papers of a mathematical or physical character, pages 415–446, 1909.

[19] Julia Neumann, Christoph Schnörr, and Gabriele Steidl. Combined svm-based feature selection and classification. *Machine learning*, 61(1-3):129–150, 2005.

[20] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[21] Vladimir Vapnik and Alexey Chervonenkis. A note on one class of perceptrons. *Automation and remote control*, 25(1), 1964.

[22] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004.