MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

A Fast Unified Classification Algorithm Based on Accelerated Proximal Gradient Method

Naoki ITO, Akiko TAKEDA and Kim-Chuan TOH

METR 2015–18 May 2015

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.keisu.t.u-tokyo.ac.jp/research/techrep/index.html

A Fast Unified Classification Algorithm Based on Accelerated Proximal Gradient Method

Naoki ITO and Akiko TAKEDA

Department of Mathematical Informatics
Graduate School of Information Science and Technology
The University of Tokyo
{naoki_ito,takeda}@mist.i.u-tokyo.ac.jp

Kim-Chuan TOH

Department of Mathematics
National University of Singapore
mattohkc@nus.edu.sg

May 2015

The binary classification is one of the most important problem in machine learning. A wide variety of binary classification models have been known including support vector machines (SVMs). To achieve a good prediction performance, it is important to find a suitable model for a given dataset. However, it is often time consuming and impractical for practitioners to try various classification models because each model employs a different formulation and algorithm. If we have a unified formulation for various classification models and an algorithm which solves the unified formulation, it may become easier to compare the performance of different classification models for a given dataset. Though several unified formulations have been proposed, e.g. unified classification model (UCM) proposed by Takeda et al., there are no unified algorithms which can deal with such unified formulations to the best of our knowledge.

In this paper, we develop a general optimization algorithm based on an accelerated proximal gradient method for UCM. Our algorithm can deal with various classification models without changing its framework. In addition, we also show that our UCM includes three existing unified formulations of binary classification: coherent risk minimization model, generalized ν-SVM, and coherent classification loss function (CCLF) minimization model. This illustrates the generality of UCM and our algorithm. Numerical experiments show that our algorithm is stable and highly competitive to specialized algorithms designed for specific models (e.g., sequential minimal optimization (SMO) for SVM).

1 Introduction

In the field of machine learning, the binary classification is one of the most important problem. A wide variety of binary classification models have been known including support vector machines (SVMs) [Cortes and Vapnik, 1995, Schölkopf et al., 2000]. To achieve a good prediction performance, it is important to find a suitable model for a given dataset. In general, each classification model employs a different formulation and algorithm. Therefore, for practitioners who are looking for a suitable classification model for their dataset, it is often time consuming and impractical to try various classification models; they might have to change not only the optimization algorithms but also solvers/software in order to solve different kind of optimization problems. If we have a unified formulation including various classification models and an algorithm which solves the unified formulation, then it will speed up the process of finding the best classification model for a given dataset.

For the purpose, we can use a unified classification model (UCM) proposed by [Takeda et al., 2013]. It is formulated as a robust optimization problem of a linear function: a minimization problem of the worst case coefficient over an assumed uncertainty set $\mathcal{U}$ (i.e., min-max problem). The differences in the models lie in the uncertainty set $\mathcal{U}$ used, which determines the feasible region of the inner max problem. The min-max problem is reduced to a single-level optimization problem (e.g., quadratic optimization problem and second-order cone problem) by using an appropriate uncertainty set $\mathcal{U}$ and taking the dual for the inner max problem. For example, when $\mathcal{U}$ is given by the reduced convex hull [Bennett and Bredensteiner, 2000, Crisp and Burges, 2000] of samples, UCM would lead to the well-known ν-SVM [Schölkopf et al., 2000]. We also can construct a new classification model by defining a new type of $\mathcal{U}$ for a given dataset. There are individual optimization methods proposed for UCM with specific $\mathcal{U}$, e.g. [Takeda et al., 2013, Iwata et al., 2014, Bertsimas and Takeda, 2014], but to the best of our knowledge, there are currently no unified algorithms which are applicable to UCM with various $\mathcal{U}$.

In this paper, we propose a fast unified classification algorithm for UCM. We first derive a simple unified formulation of classification models by focusing on a convex variant of UCM. Secondly, we design an efficient optimization algorithm for solving the resulting unified model. The key idea is to adapt the accelerated proximal gradient

(APG) method [Beck and Teboulle, 2009] to solve a dual formulation of UCM, as well as to develop efficient algorithms for projections onto the associated feasible set $\mathcal{U}$. Thirdly, in order to make our APG method practically efficient, we employed various techniques such as backtracking line search and adaptive restarting strategies to speed up the convergence of the algorithm.

Our unified algorithm using the APG method makes it easy to select a suitable model for given datasets, because we can use the same algorithmic framework while comparing various binary classification models; by only changing the computation of projections, it can solve UCMs with different uncertainty sets $\mathcal{U}$.

The APG method has typically been used for unconstrained optimization problems, especially where the objective function has both differentiable and simple non-differentiable terms. For example, Beck and Teboulle [2009] applied the APG method to unconstrained ℓ_1-regularized least square regression. Zhou et al. [2010] used another variant of APG method [Nesterov, 2005] to solve primal SVMs (i.e., unconstrained hinge loss minimization problems.) The APG method has demonstrated good performance for those problems in practice. On the other hand, UCM is formulated as an optimization problem constrained over $\mathcal{U}$. For such a constrained problem, the APG method requires the computation of the projection of a point onto the feasible region. Thus the practical efficiency of the APG method would depend on the projection.

Although the computation of the projection is not always easy in general, popular classification models typically lead to simple feasible sets $\mathcal{U}$ for which projection onto such sets can be computed efficiently. Even if a classification model has a complicated set $\mathcal{U}$, we can still take advantage of the structure of $\mathcal{U}$ in the APG method by designing an augmented Lagrangian method to compute the projection. Our work here has thus widen the application range of the APG method from individually discussed classification models to UCM.

From the viewpoint of computational complexity, our algorithm can reduce the gap from the optimal objective value at the quadratic order of $O(1/k^2)$, where k is the number iterations. The iteration complexity is proven to be optimal for first-order algorithms in the sense of [Nemirovsky and Yudin, 1983]. Indeed, it is superior to other common algorithms designed in machine learning, e.g. Frank-Wolfe algorithm [e.g. Frank and Wolfe, 1956, Jaggi, 2013] and the alternating direction method of multipliers (ADMM) [e.g. Gabay and Mercier, 1976, Boyd et al., 2010] which have the iteration complexity of $O(1/k)$. There is also an accelerated version of ADMM [Goldstein et al., 2012] for strongly convex problems, but UCM is not strongly convex in general.

We also show that UCM includes three unified formulations of classification models: the coherent risk minimization model [Bertsimas and Takeda, 2014] which is a simplified version of [Gotoh et al., 2014], the generalized ν-SVM [Kanamori et al., 2013] and the model based on the coherent classification loss function (CCLF) [Yang et al., 2014], though the relations among these models are not discussed in those papers. Figure 1 illustrates the relations among the classification models. The coherent risk measure is a class of measure functions which has been extensively studied in the area of financial engineering. Gotoh et al. [2014] brings the concept of the coherent risk into the binary classification, and constructs a unified binary classification model using a powerful repre-

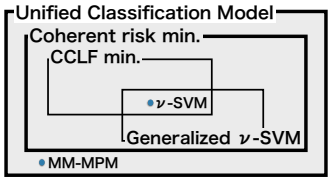

Figure 1: Relations among Classification Models

sentation theorem known for the coherent risk measure. The generalized ν-SVM extends the hinge loss function of ν-SVM to proper, closed, and convex functions. The CCLF is a generic term for quasi-convex functions satisfying five specific properties which are extracted from the empirical 0–1 loss function (i.e., rate of misclassification). Yang et al. [2014] considers the classification model based on the CCLF minimization. Though the three models are very general, UCM can cover them. This also implies the generality of our optimization algorithm.

While our method has extensive generality, numerical experiments show that it performs stably and is highly competitive to specialized algorithms designed for specific classification models (such as SMO [Platt, 1998] for ν-SVM). Indeed, our method solved classification models with a linear kernel substantially faster than SMO and the interior-point methods for large datasets.

The rest of this paper is organized as follows. Section 2 outlines a binary classification problem and UCM. Section 3 shows the APG method for UCM and introduces the backtracking rule. In Section 4, we develop efficient algorithms for computing projections used in the APG method. Section 5 shows that UCM includes three generalized classification models: the coherent risk minimization model, generalized ν-SVM, and the CCLF minimization model. Numerical experiments are presented in Section 6.

Notation. We define the plus operator as $[u]_+ := \max\{0, u\}$. The number of the elements in a set A is denoted by $|A|$. Column vectors are denoted by boldface letters, e.g. $\boldsymbol{x}$, and its i-th element is denoted as x_i. $\boldsymbol{e}$ denotes the all-one vector. $\|\boldsymbol{x}\|$ denotes the Euclidean norm (ℓ_2-norm) of a vector $\boldsymbol{x}$. The convex hull of a set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$ is denoted by $\mathrm{conv}(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\})$, i.e.

$$\mathrm{conv}(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}) = \Big\{ \sum_{i=1}^{k} q_i \boldsymbol{x}_i \;\Big|\; \boldsymbol{q}^\top \boldsymbol{e} = 1,\ \boldsymbol{q} \geq \boldsymbol{0} \Big\}.$$

The projection $P_A(\bar{\boldsymbol{x}})$ of a point $\bar{\boldsymbol{x}}$ onto A is defined as follows:

$$P_A(\bar{\boldsymbol{x}}) = \operatorname*{argmin}_{\boldsymbol{x} \in A} \frac{1}{2} \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|^2. \tag{1}$$

Other notation needed for binary classification models will be introduced in Section 2.

2 Binary Classification Models

Let $\mathcal{X} \subset \mathbb{R}^n$ be the input domain and $\{+1, -1\}$ be the set of the binary labels. Suppose that we have samples,

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \in \mathcal{X} \times \{+1, -1\}.$$

Let $M := \{1, \ldots, m\}$, $M_+ := \{i \in M \mid y_i = +1\}$, and $M_- := \{i \in M \mid y_i = -1\}$. Let $m_+ = |M_+|$ and $m_- = |M_-|$.

We compute $(\boldsymbol{w}, b)$ for a decision function $h(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ using these samples and use $h(\boldsymbol{x})$ to predict the label for a new input point $\hat{\boldsymbol{x}} \in \mathcal{X}$. If $h(\hat{\boldsymbol{x}})$ is positive (resp. negative), then the label of $\hat{\boldsymbol{x}}$ is predicted to be $+1$ (resp. -1). Here we focus on linear learning models using linear functions $h(\boldsymbol{x})$, but the discussions in this paper can be directly applied to non-linear kernel models [Schölkopf and Smola, 2002] using nonlinear maps $\phi(\boldsymbol{x})$ mapping $\boldsymbol{x}$ from the original space to a high dimensional space.

2.1 Unified Classification Model

There are various of binary classification models to compute $(\boldsymbol{w}, b)$ such as the support vector machines (SVMs), e.g. [Cortes and Vapnik, 1995, Schölkopf et al., 2000, Schölkopf and Smola, 2002], margin maximized minimax probability machine (MM-MPM) [Nath and Bhattacharyya, 2007], and a model based on Fisher's discriminant analysis (MM-FDA) [Bhattacharyya, 2004, Takeda et al., 2013]. In this paper, we deal with a unified classification model (UCM) proposed by Takeda et al. [2013], which provides a unified formulation for these classification models. UCM is formulated as the following robust optimization problem:

$$\min_{\|\boldsymbol{w}\|^2=1} \max_{\boldsymbol{x} \in \mathcal{U}} -\boldsymbol{w}^\top \boldsymbol{x}, \tag{2}$$

where $\mathcal{U} \subseteq \mathbb{R}^n$ is a closed convex set which is called as an *uncertainty* set. The uncertainty set $\mathcal{U}$, for example, is typically represented as the Minkowski difference of two closed convex sets $\mathcal{U}_+$ and $\mathcal{U}_-$:

$$\mathcal{U}_+ \ominus \mathcal{U}_- = \{\boldsymbol{x}_+ - \boldsymbol{x}_- \mid \boldsymbol{x}_+ \in \mathcal{U}_+,\ \boldsymbol{x}_- \in \mathcal{U}_-\},$$

where the sets $\mathcal{U}_+$ and $\mathcal{U}_-$ typically represent uncertainty of mean vectors $\bar{\boldsymbol{x}}_+$ and $\bar{\boldsymbol{x}}_-$ of the classes $+1$ and -1, respectively. See Figure 2 for examples of uncertainty sets. It is known that if $\mathbf{0} \notin \mathcal{U}$ (i.e., $\mathcal{U}_+ \cap \mathcal{U}_- = \phi$), the non-convex constraint $\|\boldsymbol{w}\| = 1$ can be replaced by the convex one $\|\boldsymbol{w}\| \leq 1$ without changing the optimal solutions. On the other hand, if $\mathbf{0} \in \mathcal{U}$ (i.e., $\mathcal{U}_+ \cap \mathcal{U}_- \neq \phi$), this replacement leads UCM to an obvious and worthless optimal solution $\boldsymbol{w}^* = \mathbf{0}$.

In this paper, we assume $\mathbf{0} \notin \mathcal{U}$ (i.e., $\mathcal{U}_+ \cap \mathcal{U}_- = \phi$) and focus on the convex model although UCM (2) also includes non-convex models, e.g. Eν-SVM [Perez-Cruz et al., 2003]. The convex UCM (2) can be equivalently formulated as follows:

$$\max_{\boldsymbol{x} \in \mathcal{U}} \min_{\|\boldsymbol{w}\| \leq 1} -\boldsymbol{w}^\top \boldsymbol{x} = -\min_{\boldsymbol{x} \in \mathcal{U}} \|\boldsymbol{x}\|. \tag{3}$$

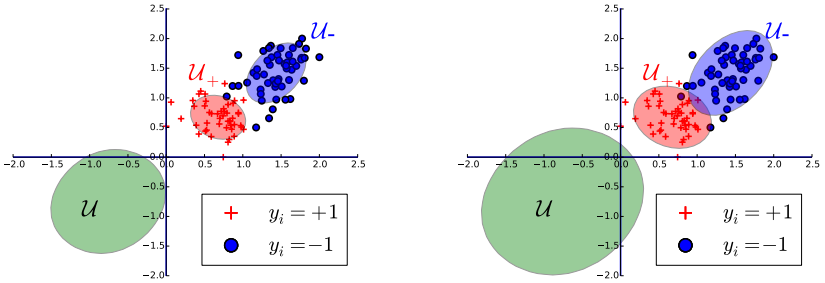

Figure 2: Examples of Uncertainty Sets $\mathcal{U}$, $\mathcal{U}_+$, and $\mathcal{U}_-$ for UCM. The left panel illustrates the case where $\mathbf{0} \notin \mathcal{U}$ (i.e., $\mathcal{U}_+ \cap \mathcal{U}_- = \phi$). In this case, the non-convex constraint $\|\boldsymbol{w}\| = 1$ can be replaced by $\|\boldsymbol{w}\| \leq 1$ without changing the optimal solutions. On the other hand, if $\mathbf{0} \in \mathcal{U}$ (i.e., $\mathcal{U}_+ \cap \mathcal{U}_- \neq \phi$) as illustrated in the right panel, UCM is essentially non-convex; the convex relaxation $\|\boldsymbol{w}\| \leq 1$ of the non-convex constraint $\|\boldsymbol{w}\| = 1$ leads UCM to a worthless solution $\boldsymbol{w}^* = \mathbf{0}$. In this paper, we assume $\mathbf{0} \notin \mathcal{U}$ (as in the left panel) and focus on the convex model.

Note that the optimal value of (3) is 0 if and only if $\mathbf{0} \in \mathcal{U}$ (non-convex case). An optimal solution $\boldsymbol{w}^*$ of (2) can be obtained by the optimality condition for $\boldsymbol{w}$, i.e. $\boldsymbol{w}^* = \boldsymbol{x}^*/\|\boldsymbol{x}^*\|$, where $\boldsymbol{x}^*$ is an optimal solution of (3). There are various ways to compute the bias term b of the decision function h as illustrated in Takeda et al. [2013]; an example is to use the value b^* minimizing the empirical loss, i.e.,

$$b^* = \operatorname*{argmin}_b \frac{1}{m} \sum_{i \in M} I\big(y_i(\boldsymbol{x}_i^\top \boldsymbol{w}^* + b) < 0\big),$$

where

$$I(u) = \begin{cases} 1 & \text{if } u < 0 \\ 0 & \text{otherwise.} \end{cases}$$

Various existing classification models can be reduced to (2) (and (3)) by defining an appropriate $\mathcal{U}$. Conversely, a properly chosen $\mathcal{U}$ would lead to a new classification model.

The elements of $\mathcal{U}$ are often represented by using some parameter variables $\boldsymbol{q} \in \mathbb{R}^d$ as follows:

$$\mathcal{U} := \{\boldsymbol{x}(\boldsymbol{q}) \mid \boldsymbol{q} \in \mathcal{U}'\},$$

where $\mathcal{U}' \subseteq \mathbb{R}^d$ is a convex set, and $\boldsymbol{x}(\boldsymbol{q}) : \mathbb{R}^d \to \mathbb{R}^n$ is a continuous map. Then, we can equivalently formulate the UCM (3) as follows:

$$\min_{\boldsymbol{q} \in \mathcal{U}'} \; f(\boldsymbol{q}) := \frac{1}{2}\big\|\boldsymbol{x}(\boldsymbol{q})\big\|^2. \tag{4}$$

In the followings, we consider to solve UCM of the form (4) because it can keep the feasible region $\mathcal{U}'$ simple so that the projection $P_{\mathcal{U}'}(\bar{\boldsymbol{q}})$ of a point $\bar{\boldsymbol{q}}$ onto $\mathcal{U}'$ can be computed efficiently as shown in Section 4. A few examples of $\mathcal{U}$, $\boldsymbol{x}(\boldsymbol{q})$, $\mathcal{U}'$, and related models to UCM (e.g., ν-SVM, MM-MPM, and MM-FDA) are given in the Appendix.

3 Accelerated Proximal Gradient Method

In this section, we provide a general optimization algorithm for the UCM of the form (4) based on the accelerated proximal gradient (APG) method [Beck and Teboulle, 2009]. Our algorithm is applicable to various $\mathcal{U}'$, unlike specialized algorithms designed for a particular classification model.

The APG method requires the following assumptions for its convergence.

Assumption 1. *The following conditions hold:*

1. *$f(\boldsymbol{q})$ is a proper, closed, convex and differentiable function.*
2. *The gradient $\nabla f(\boldsymbol{q})$ is Lipschitz continuous, i.e. there exists a constant $L > 0$ such that*
$$\|\nabla f(\boldsymbol{q}) - \nabla f(\boldsymbol{p})\| \leq L\|\boldsymbol{q} - \boldsymbol{p}\| \quad \forall \boldsymbol{q}, \boldsymbol{p} \in \mathbb{R}^d.^*$$
We refer to the minimum value of such L as the Lipschitz constant L_f of the gradient $\nabla f(\boldsymbol{q})$.
3. *The feasible region $\mathcal{U}'$ is closed and convex.*

Indeed, the models in Examples 1–5 in the Appendix satisfy Assumption 1. In addition to this, we make the following assumption for the purpose of computational tractability.

Assumption 2. *The projection $P_{\mathcal{U}'}(\bar{\boldsymbol{q}})$ of a point $\bar{\boldsymbol{q}}$ onto the feasible region $\mathcal{U}'$ (see (1) for the definition) can be computed efficiently.*

Simple projection algorithms for some $\mathcal{U}'$ are developed in Section 4. We will later discuss the situation where Assumption 2 does not hold.

3.1 Accelerated Proximal Gradient Method

Let L be an estimation of the Lipschitz constant L_f of the gradient $\nabla f(\boldsymbol{q})$. We define an approximate function $g_L : \mathbb{R}^d \to \mathbb{R}$ of $f(\boldsymbol{q})$ around $\boldsymbol{p}$ and a mapping $T_L(\boldsymbol{q}) : \mathbb{R}^d \to \mathcal{U}'$ as follows:

$$g_L(\boldsymbol{q}; \boldsymbol{p}) = f(\boldsymbol{p}) + \langle \nabla f(\boldsymbol{p}), \boldsymbol{q} - \boldsymbol{p} \rangle + \frac{L}{2}\|\boldsymbol{q} - \boldsymbol{p}\|^2$$
$$T_L(\boldsymbol{p}) = \operatorname*{argmin}_{\boldsymbol{q} \in \mathcal{U}'} g_L(\boldsymbol{q}; \boldsymbol{p}).$$

*It is not sufficient to consider the Lipschitz constant over $\mathcal{U}'$ because the APG method generates a point ($\boldsymbol{p}^{k+1}$ at Step 3 shown later) which may not be in $\mathcal{U}'$.

The basic proximal gradient (PG) method generates a sequence $\{\boldsymbol{q}^k\}_{k=0}^{\infty}$ by

$$\begin{aligned}\boldsymbol{q}^{k+1} = T_L(\boldsymbol{q}^k) \;&=\; \operatorname*{argmin}_{\boldsymbol{q}\in\mathcal{U}'}\Big\{\frac{L}{2}\Big\|\boldsymbol{q}-\Big(\boldsymbol{q}^k-\frac{1}{L}\nabla f(\boldsymbol{q}^k)\Big)\Big\|^2\Big\}\\ &= P_{\mathcal{U}'}\Big(\boldsymbol{q}^k-\frac{1}{L}\nabla f(\boldsymbol{q}^k)\Big).\end{aligned}$$

The above PG method coincides with the gradient projection method in this case. It is known that the PG method has the iteration complexity that $f(\boldsymbol{q}^k)-f(\boldsymbol{q}^*)=O(1/k)$, where $\boldsymbol{q}^*$ is an optimal solution of (4).

The recently proposed APG method [Beck and Teboulle, 2009] is an acceleration of the PG method. It generates two sequences $\{\boldsymbol{p}^k\}_{k=1}^{\infty}$ and $\{\boldsymbol{q}^k\}_{k=0}^{\infty}$. For an arbitrary initial point $\boldsymbol{p}^1=\boldsymbol{q}^0\in\mathcal{U}$ and $t_1=1$, the APG method solves (4) through the following steps ($k=1,2,\ldots$):

Accelerated Proximal Gradient Method

Step 1. Compute

$$\boldsymbol{q}^k = T_L(\boldsymbol{p}^k) = P_{\mathcal{U}'}\Big(\boldsymbol{p}^k-\frac{1}{L}\nabla f(\boldsymbol{p}^k)\Big).$$

Step 2. Compute

$$t_{k+1}=\frac{1+\sqrt{1+4t_k^2}}{2}.$$

Step 3. Compute

$$\boldsymbol{p}^{k+1}=\boldsymbol{q}^k+\frac{t_k-1}{t_{k+1}}(\boldsymbol{q}^k-\boldsymbol{q}^{k-1}).$$

Note that $\boldsymbol{p}^{k+1}\in\mathcal{U}'$ is not always true while $\boldsymbol{q}^k\in\mathcal{U}'$ holds for all $k=1,2,\ldots$. For the APG method, the following convergence result is known.

Theorem 1 (Theorem 4.4 of [Beck and Teboulle, 2009])**.** *Suppose that L is the Lipschitz constant L_f of the gradient $\nabla f(\boldsymbol{q})$. Let the sequence $\{\boldsymbol{q}^k\}_{k=0}^{\infty}$ be generated by the APG method, and let $\boldsymbol{q}^*$ be an optimal solution of (4). For any $k\geq 0$, we have*

$$f(\boldsymbol{q}^k)-f(\boldsymbol{q}^*)\leq\frac{2L\|\boldsymbol{q}^0-\boldsymbol{q}^*\|^2}{(k+1)^2}.$$

As the optimal solution $\boldsymbol{q}^*$ and the optimal value $f(\boldsymbol{q}^*)$ are not known a priori, the iteration complexity in Theorem 1 does not provide us with a good stopping condition for the APG method. As the upper bound on the right-hand side is only a worst-case bound, estimating the number of iterations needed to ensure that it is less than a certain desired accuracy is often exceedingly conservative, even if one is able to find an upper bound on $\|\boldsymbol{q}^0-\boldsymbol{q}^*\|^2$, say in the case when $\mathcal{U}'$ is a bounded set. The next lemma provides a possible way to check the optimality of the generated iterate $\boldsymbol{q}^k$.

Lemma 1. *A point $\boldsymbol{q}^*$ is an optimal solution of (4) if and only if $T_L(\boldsymbol{q}^*)=\boldsymbol{q}^*$.*

Proof. From Proposition 4.7.2 of [Bertsekas et al., 2003], $\boldsymbol{q}^*$ is an optimal solution of (4) if and only if

$$\langle \nabla f(\boldsymbol{q}^*),\ \boldsymbol{q} - \boldsymbol{q}^* \rangle \geq 0, \quad \forall \boldsymbol{q} \in \mathcal{U}'. \tag{5}$$

Recall that

$$T_L(\boldsymbol{q}^*) = \operatorname*{argmin}_{\boldsymbol{q} \in \mathcal{U}'} \Big\{ f(\boldsymbol{q}^*) + \langle \nabla f(\boldsymbol{q}^*),\ \boldsymbol{q} - \boldsymbol{q}^* \rangle + \frac{L}{2} \|\boldsymbol{q} - \boldsymbol{q}^*\|^2 \Big\}. \tag{6}$$

If $\boldsymbol{q}^*$ satisfies (5), it is obvious that $T_L(\boldsymbol{q}^*) = \boldsymbol{q}^*$. Next, we show the if part. Since $T_L(\boldsymbol{q}^*)$ is an optimal solution of (6), we have

$$\begin{aligned} 0 &\leq \langle \nabla g_L(T_L(\boldsymbol{q}^*); \boldsymbol{q}^*), \boldsymbol{q} - T_L(\boldsymbol{q}^*) \rangle \\ &= \langle \nabla f(\boldsymbol{q}^*) + L(T_L(\boldsymbol{q}^*) - \boldsymbol{q}^*),\ \boldsymbol{q} - T_L(\boldsymbol{q}^*) \rangle, \quad \forall \boldsymbol{q} \in \mathcal{U}'. \end{aligned} \tag{7}$$

If $\boldsymbol{q}^* = T_L(\boldsymbol{q}^*)$ holds, then (7) leads to (5). □

Note that Lemma 1 holds for any convex set $\mathcal{U}'$. The term $L(T_L(\boldsymbol{q}^*) - \boldsymbol{q}^*)$ of (7) can be seen as a violation of the optimality condition (5). Thus, it is natural to employ the following stopping criteria:

- If
$$\|L(\boldsymbol{q}^k - \boldsymbol{p}^k)\| < \epsilon \quad \text{or} \quad \|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\| < \epsilon$$
for sufficiently small $\epsilon > 0$, then terminate the algorithm.

Note that $\boldsymbol{q}^k = T_L(\boldsymbol{p}^k)$. While the condition $\|L(\boldsymbol{q}^k - \boldsymbol{p}^k)\| < \epsilon$ is easy to check at each iteration, it measures the necessary and sufficient optimality for $\boldsymbol{p}^k$ but not $\boldsymbol{q}^k$. On the other hand, the condition $\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\| < \epsilon$ check for the necessary and sufficient optimality of $\boldsymbol{q}^k$, but computing $T_L(\boldsymbol{q}^k)$ involves the extra computation of the gradient $\nabla f(\boldsymbol{q}^k)$ for which the cost can be high (see Tables 3 and 7 in Section 6). Thus we check the condition, $\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\| < \epsilon$, only in every 100 iterations. Despite the difference, as one can see from Figure 6 in Section 6, the two quantities $\left\|L(\boldsymbol{q}^k - \boldsymbol{p}^k)\right\|$ and $\left\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\right\|$ do not differ much in practice. Hence it is safe to use the first condition as the surrogate termination criterion, and the second condition is used as an extra check only after the first condition is satisfied.

Remark 1. *It is sometimes difficult to compute the projection $P_{\mathcal{U}'}$ in Step 1 exactly. Jiang et al. [2012] showed that the APG method still has the iteration complexity $O(1/k^2)$ if the projection $P_{\mathcal{U}'}$ is computed approximately with a sufficiently small tolerance ϵ_k at the k-th iteration, where $\sum_{k=1}^{\infty} \epsilon_k < \infty$. Hence, we can use a numerical method, such as a bisection method in Section 4, to compute the projection $P_{\mathcal{U}'}$ without losing the iteration complexity of $O(1/k^2)$.*

3.2 Strategies to Speed-up Convergence

In order to make our APG method practically efficient, we employ several strategies to speed-up the APG method.

Backtracking Strategy. In Theorem 1, we assumed that L is the Lipschitz constant L_f of $\nabla f(\boldsymbol{q})$. In order to speed up the convergence of the APG method, however, it is advantageous to use a smaller value for L whenever possible since the constant L plays the role of a step size as in a gradient descent method; fixing L to be the Lipschitz constant L_f of $\nabla f(\boldsymbol{q})$ is usually too conservative (see Table 6 in Section 6). Thus we adopt the following backtracking strategy at the beginning of each iteration with arbitrary constants $\eta > 1$ and $L > 0$ shown in [Beck and Teboulle, 2009].

Step 0. Find the smallest integer $i_k \in \{0, 1, \ldots\}$ such that

$$f(T_{\bar{L}}(\boldsymbol{p}^k)) \leq g_{\bar{L}}(T_{\bar{L}}(\boldsymbol{p}^k); \boldsymbol{p}^k), \tag{8}$$

with $\bar{L} = \eta^{i_k} L$. Then update $L \leftarrow \bar{L}$.

The inequality (8) in Step 0 ensures that the complexity result in Theorem 1 still holds. We note that if $\bar{L} \geq L_f$, then the inequality in Step 0 is satisfied. We computes the Step 0 in every 10 iterations because Step 0 involves extra computations of the function value $f(T_{\bar{L}}(\boldsymbol{p}^k))$ whose cost can be significant (see Tables 3 and 7 in Section 6).

Decreasing Strategy for L. Beck and Teboulle [2009] designed the backtracking algorithm (Step 0) so that the values of L is non-decreasing, i.e. $\bar{L} = \eta^{i_k} L$ ($\geq L$) would be the next L. However, smaller $\bar{L}$ than L may satisfy (8) at some $\boldsymbol{p}^k$, $k \in \{1, 2, \ldots\}$. As noted above, it is advantageous to use a smaller value for L whenever possible since the constant $\frac{1}{L}$ gives a larger step size. Thus, we re-design Step 0 so that i_k can take a value in $\{-1, 0, 1, \ldots\}$ to decrease the value of L if the condition (8) permits.

Restarting Strategy. The value $\frac{t_k - 1}{t_{k+1}} \in [0, 1)$ in Step 3 determines the amount of *momentum* $\frac{t_k - 1}{t_{k+1}}(\boldsymbol{q}^k - \boldsymbol{q}^{k-1})$. The sequence of the values $\{\frac{t_k - 1}{t_{k+1}}\}_{k=1}^{\infty}$ is monotonically increasing and is bounded by 1. When the value $\frac{t_k - 1}{t_{k+1}}$ is close to 1, i.e., the momentum is high, the sequences of the solutions $\{\boldsymbol{q}^k\}_{k=0}^{\infty}$ and $\{\boldsymbol{p}^k\}_{k=1}^{\infty}$ would overshoot and oscillate around the optimal solution while Theorem 1 ensures their convergence. In order to avoid the oscillation and further speed up the convergence, we introduce an adaptive restarting strategy shown in [O'Donoghue and Candés, 2013].

Step 4. If $\nabla f(\boldsymbol{p}^k)^\top (\boldsymbol{q}^k - \boldsymbol{q}^{k-1}) > 0$, then restart the APG method (Algorithm 1) with $\boldsymbol{p}^1 = \boldsymbol{q}^0 = \boldsymbol{q}^{k-1}$.

Roughly, the APG method restarts to reset the momentum back to zero if the direction of motion $\boldsymbol{q}^k - \boldsymbol{q}^{k-1}$ seems to cause the value of f to increase, which may be a sign of oscillation. Note that the computational cost of Step 4 is inexpensive since $\nabla f(\boldsymbol{p}^k)$ is already computed at Step 1.

Decreasing Strategy for η. Since a smaller value of L (larger step size) tends to enlarge the distance of $\boldsymbol{q}^k - \boldsymbol{q}^{k-1}$ as illustrated in Figure 3, the decreasing strategy for L induces high momentum inherently. Hence, it sometimes triggers the restart step (at Step 4)

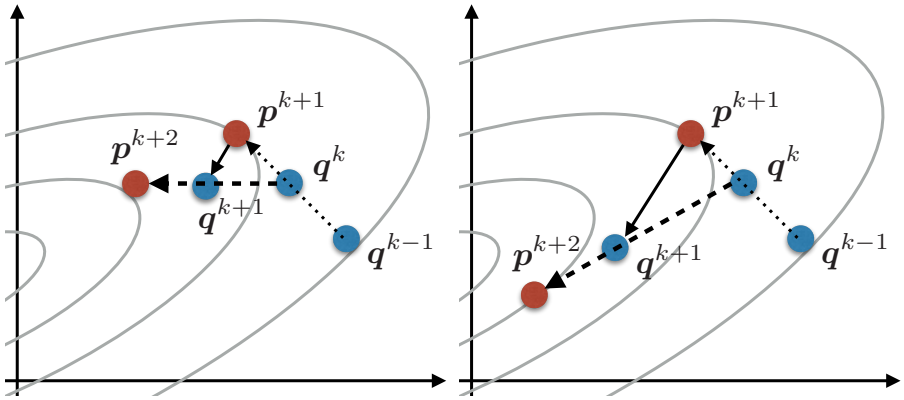

Figure 3: Effects of the value of L to the momentum $\frac{t_k-1}{t_{k+1}}(\boldsymbol{q}^k - \boldsymbol{q}^{k-1})$ (left: L is large, right: L is small). A smaller value of L gives a larger step size at Step 1 as illustrated by the solid lines. This results in high momentum at Step 3 as illustrated by the dashed lines.

frequently and/or makes the APG method unstable. In order to avoid the instability, it would be necessary to reduce the rate of decreasing L. Here we take a strategy to reduce the value of η as $\eta \leftarrow \delta \cdot \eta + (1-\delta)\cdot 1$ for arbitrary $\delta \in (0,1)$ when the restart occurs.

As a consequence, our practical APG method is described as Algorithm 1.

3.3 Remedy for Difficult Projection

For the situation where Assumption 2 does not hold, we can apply an augmented Lagrangian method to (4) in order to replace (4) with a sequence of subproblems that satisfy Assumption 2. Suppose that $\mathcal{U}' = S \cap T$, where S is a simple closed convex set for which projection onto such a set can be computed efficiently, $T = \{\boldsymbol{q} \mid g_i(\boldsymbol{q}) \le 0,\ i \in \{1,\dots,l_g\},\ h_j(\boldsymbol{q}) = 0,\ j \in \{1,\dots,l_h\}\}$, where $g_i : \mathbb{R}^d \to \mathbb{R}$ is a proper closed convex function, and $h_j : \mathbb{R}^d \to \mathbb{R}$ is an affine function. We consider the following augmented Lagrangian (see e.g. [Rockafellar, 1978]):

$$\mathcal{L}_\sigma(\boldsymbol{q},\boldsymbol{\lambda},\boldsymbol{\mu}) := f(\boldsymbol{q}) + \frac{\sigma}{2}\sum_{i=1}^{l_g}\left\{\left[g_i(\boldsymbol{q}) + \frac{\lambda_i}{\sigma}\right]_+^2 - \left(\frac{\lambda_i}{\sigma}\right)^2\right\} + \underbrace{\frac{\sigma}{2}\sum_{j=1}^{l_h}\left\{\left(h_j(\boldsymbol{q}) + \frac{\mu_j}{\sigma}\right)^2 - \left(\frac{\mu_j}{\sigma}\right)^2\right\}}_{\sum_{j=1}^{l_h}\left(\mu_j h_j(\boldsymbol{q}) + \frac{\sigma}{2}(h_j(\boldsymbol{q}))^2\right)},$$

where $\lambda_i \ge 0$ (and $\mu_i \in \mathbb{R}$) is the Lagrange multiplier associated with the constraint $g_i(\boldsymbol{q}) \le 0$ (and $h_j(\boldsymbol{q}) = 0$, respectively), and $\sigma > 0$ is a penalty parameter to penalize any constraint violation $[g_i(\boldsymbol{q})]_+^2$ and $(h_j(\boldsymbol{q}))^2$. Let $\mathbb{R}^{l_g}_+$ be the positive orthant. We then apply the augmented Lagrangian method to solve (4).

Algorithm 1 A Practical Accelerate Proximal Gradient Method for (4)

INPUT: $\mathcal{U}'$, $\epsilon > 0$, $L > 0$, $\eta > 1$, $\delta \in (0,1)$, $k_{max} > 0$, $\boldsymbol{p}^1 = \boldsymbol{q}^0, t_1 = 1$ OUTPUT: $\boldsymbol{q}^k$

for $k = 1, \ldots, k_{max}$ **do**
  if k mod 10 == 1 **then** # Do Steps 0 and 1
    $L = L/\eta$; $\boldsymbol{q}^k = T_L(\boldsymbol{p}^k) = P_{\mathcal{U}'}\left(\boldsymbol{p}^k - \frac{1}{L}\nabla f(\boldsymbol{p}^k)\right)$
    while $f(\boldsymbol{q}^k) > g_L(\boldsymbol{q}^k; \boldsymbol{p}^k)$ **do**
      $L = \eta L$; $\boldsymbol{q}^k = T_L(\boldsymbol{p}^k) = P_{\mathcal{U}'}\left(\boldsymbol{p}^k - \frac{1}{L}\nabla f(\boldsymbol{p}^k)\right)$
    end while
  else # Do Step 1
    $\boldsymbol{q}^k = T_L(\boldsymbol{p}^k) = P_{\mathcal{U}'}\left(\boldsymbol{p}^k - \frac{1}{L}\nabla f(\boldsymbol{p}^k)\right)$
  end if
  if $\|L(\boldsymbol{q}^k - \boldsymbol{p}^k)\| < \epsilon$ or ((k mod 100 == 1) and ($\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\| < \epsilon$)) **then**
    break
  end if
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$ # Step 2
  $\boldsymbol{p}^{k+1} = \boldsymbol{q}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{q}^k - \boldsymbol{q}^{k-1})$ # Step 3
  if $\langle \nabla f(\boldsymbol{p}^k), \boldsymbol{q}^k - \boldsymbol{q}^{k-1} \rangle > 0$ **then** # Step 4
    restart Algorithm 1 with $\boldsymbol{p}^1 = \boldsymbol{q}^0 = \boldsymbol{q}^{k-1}$, $\eta \leftarrow \delta \cdot \eta + (1-\delta) \cdot 1$.
  end if
end for

Augmented Lagrangian Method

Step 1. Compute

$$\boldsymbol{q}^{k+1} = \operatorname*{argmin}_{\boldsymbol{q} \in S} \mathcal{L}_\sigma(\boldsymbol{q}, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k). \tag{9}$$

Step 2. Update

$$\boldsymbol{\lambda}^{k+1} = P_{\mathbb{R}_+^{lg}}\left(\boldsymbol{\lambda}^k + \sigma \nabla_{\boldsymbol{\lambda}} \mathcal{L}_\sigma(\boldsymbol{q}^{k+1}, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)\right)$$
$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \sigma \nabla_{\boldsymbol{\mu}} \mathcal{L}_\sigma(\boldsymbol{q}^{k+1}, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k).$$

The APG method can be applied to the subproblem (9) since the projection onto S is easy to compute, i.e. Assumption 2 holds. Although the APG method is executed at each iteration of the augmented Lagrangian method, the computational cost could be reduced by setting $\boldsymbol{q}^k$ to the initial point of the APG method (i.e., using *hot start* strategy); it can be expected that the optimal solution of (9) may not change too much at each iteration since the structure of the problem (9) does not change except for the values of multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$.

4 Vector Projection Computation

In the APG method (Algorithm 1), the projection $P_{\mathcal{U}'}$ onto $\mathcal{U}'$ appears at Step 1. We need to change the computation of the projection $P_{\mathcal{U}'}$ depending on the definition of

$\mathcal{U}'$. In this section, we introduce some examples of $\mathcal{U}'$ where $P_{\mathcal{U}'}$ is easy to compute, i.e. Assumption 2 holds, and provide efficient projection computations for them.

4.1 Projection onto Balls

Let $\bar{\boldsymbol{x}}_o$ and Σ_o, $o \in \{+,-\}$, be the mean vectors and the positive definite covariance matrices, respectively, of $\boldsymbol{x}_i$, $i \in M_o$. Let the parameter variable $\boldsymbol{q} = (\boldsymbol{u}_+, \boldsymbol{u}_-)$, where $\boldsymbol{u}_o \in \mathbb{R}^n$, $o \in \{+,-\}$. As shown in Example 1 (in the Appendix), MM-MPM [Nath and Bhattacharyya, 2007] can be formulated in the form (4) of UCM as follows:

$$\min_{\boldsymbol{q}:=(\boldsymbol{u}_+,\boldsymbol{u}_-)\in\mathcal{U}'} \frac{1}{2}\big\|\big(\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2}\boldsymbol{u}_+\big) - \big(\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2}\boldsymbol{u}_-\big)\big\|^2, \tag{10}$$

where $\mathcal{U}' := \big\{(\boldsymbol{u}_+, \boldsymbol{u}_-) \mid \|\boldsymbol{u}_o\| \le \kappa,\ o \in \{+,-\}\big\} \subseteq \mathbb{R}^{2n}$ with a parameter $\kappa \in [0,\infty)$. There exists the supremum $\kappa_{max} \in [0,\infty)$ of κ such that the optimal value of (10) is not 0 (see Example 1 in the Appendix). Then, the projection $P_{\mathcal{U}'}(\bar{\boldsymbol{q}})$ of a point $\bar{\boldsymbol{q}}$ onto the feasible set $\mathcal{U}'$ is clearly easy to compute. Letting $\bar{\boldsymbol{q}} = (\bar{\boldsymbol{u}}_+, \bar{\boldsymbol{u}}_-)$ and $\hat{\boldsymbol{q}} = (\hat{\boldsymbol{u}}_+, \hat{\boldsymbol{u}}_-) = P_{\mathcal{U}'}(\bar{\boldsymbol{q}})$, we have

$$\hat{\boldsymbol{u}}_o = \min\Big\{1,\ \frac{\kappa}{\|\bar{\boldsymbol{u}}_o\|}\Big\}\bar{\boldsymbol{u}}_o, \quad o \in \{+,-\}.$$

Hence, the APG method would work for MM-MPM (10), efficiently.

Similarly, the following problem is known to be equivalent to MM-FDA [Bhattacharyya, 2004, Takeda et al., 2013]:

$$\min_{\boldsymbol{q}:=\boldsymbol{u}\in\mathcal{U}'} \frac{1}{2}\big\|\big(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-\big) + \big(\Sigma_+ + \Sigma_-\big)^{1/2}\boldsymbol{u}\big\|^2, \tag{11}$$

where $\mathcal{U}' := \big\{\boldsymbol{u} \mid \|\boldsymbol{u}\| \le \kappa\big\} \subseteq \mathbb{R}^n$ with a parameter $\kappa \in [0,\infty)$. There exists the supremum $\kappa_{max} \in [0,\infty)$ of κ such that the optimal value of (11) is not 0. In applying the APG method to MM-FDA (11), we can compute the projection $P_{\mathcal{U}'}(\bar{\boldsymbol{q}})$ as follows:

$$P_{\mathcal{U}'}(\bar{\boldsymbol{q}}) = \min\Big\{1,\ \frac{\kappa}{\|\bar{\boldsymbol{q}}\|}\Big\}\bar{\boldsymbol{q}}.$$

4.2 Projection onto a Simplex with Upper Bounds

Let us consider a vector $\boldsymbol{q} \in \mathbb{R}^m$. We denote by $\boldsymbol{q}_+$ and $\boldsymbol{q}_-$ the subvectors of $\boldsymbol{q}$ corresponding to the label $+1$ and -1, respectively. $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$ are subvectors of $\boldsymbol{e}$ with size m_+ and m_-. ν-SVM [Schölkopf et al., 2000] can be formulated in the form (4) of UCM as follows (see Example 2 in the Appendix):

$$\min_{\boldsymbol{q}\in\mathcal{U}'} \frac{1}{2}\Big\|\sum_{i\in M_+} q_i\boldsymbol{x}_i - \sum_{i\in M_-} q_i\boldsymbol{x}_i\Big\|^2, \tag{12}$$

where $\mathcal{U}' := \big\{\boldsymbol{q} \mid \boldsymbol{q}_o^\top\boldsymbol{e}_o = \frac{1}{2},\ o \in \{+,-\},\ \boldsymbol{0} \le \boldsymbol{q} \le \frac{1}{m\nu}\boldsymbol{e}\big\} \subseteq \mathbb{R}^m$ with a parameter $\nu \in (0,1]$. There exists the infimum ν_{min} (and the maximum $\nu_{max} = \frac{2\min\{m_+,m_-\}}{m}$) of

$\nu \in (0, 1]$ such that the optimal value of (12) is not 0 (and the problem (12) is feasible, respectively).

In applying the APG method to ν-SVM (12), we shall be concerned with the projection $P_{\mathcal{U}'}$:

$$\min_{\boldsymbol{q}} \Big\{ \frac{1}{2}\|\boldsymbol{q} - \bar{\boldsymbol{q}}\|^2 \mid \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+,-\},\ \boldsymbol{0} \le \boldsymbol{q} \le \frac{1}{m\nu}\boldsymbol{e} \Big\}. \tag{13}$$

The problem (13) can be decomposed into the following two problems:

$$\min_{\boldsymbol{q}_o} \Big\{ \frac{1}{2}\|\boldsymbol{q}_o - \bar{\boldsymbol{q}}_o\|^2 \mid \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \boldsymbol{0} \le \boldsymbol{q}_o \le \frac{1}{m\nu}\boldsymbol{e}_o \Big\},\ o \in \{+,-\}. \tag{14}$$

We should mention that there are several algorithms, e.g. in [Kiwiel, 2008, Pardalos and Kovoor, 1990] and references therein, that can be applied to (14). If we employ the breakpoint search algorithm [Kiwiel, 2008, Algorithm 3.1], for example, the exact solution $\boldsymbol{q}_o^*$ of (14) can be obtained with the computational complexity of $O(m_o)$, $o \in \{+,-\}$. According to our observation, however, the bisection algorithm which we will describe later requires less computation time to compute a solution $\hat{\boldsymbol{q}}_o$ such that $|\hat{q}_i - q_i^*|$, $i \in M_o$, are less than the machine epsilon $\epsilon' \approx 2.22 \times 10^{-16}$ (i.e., IEEE 754 double precision).

For the problems (14), the following properties are well-known.

Lemma 2 (e.g. [Helgason et al., 1980])**.** *Suppose that the problems* (14) *are feasible, i.e.* $\nu \in \big(0, \frac{2\min\{m_+, m_-\}}{m}\big]$. *Let*

$$q_i(\theta_o) = \min\Big\{ [\bar{q}_i - \theta_o]_+, \frac{1}{m\nu} \Big\}, \quad i \in M_o, \quad o \in \{+,-\},$$

and let

$$h_o(\theta_o) = \sum_{i \in M_o} q_i(\theta_o), \quad o \in \{+,-\}.$$

The following statements hold for $o \in \{+,-\}$*:*

1. $h_o(\theta_o)$ *is a continuous and non-increasing function.*
2. *There exists* $\theta_o^* \in (\bar{q}_o^{min} - \frac{1}{2m_o}, \bar{q}_o^{max} - \frac{1}{2m_o})$ *such that* $h_o(\theta_o^*) = \frac{1}{2}$, *where* $\bar{q}_o^{max} = \max\{\bar{q}_i \mid i \in M_o\}$ *and* $\bar{q}_o^{min} = \min\{\bar{q}_i \mid i \in M_o\}$.
3. *Let* $\hat{q}_i = q_i(\theta_o^*)$, $i \in M_o$. $\hat{\boldsymbol{q}}_o$ *is an optimal solution of* (14).

Therefore, the problem (14) can be reduced to the problem finding θ_o^* such that $h_o(\theta_o^*) = \frac{1}{2}$, $o \in \{+,-\}$. Since $h_o(\theta_o)$ is non-increasing (as illustrated in Figure 4), the bisection algorithm can be applied to solving the equation $h_o(\theta_o) = \frac{1}{2}$ numerically with any desired accuracy $\epsilon' > 0$. The steps are as follows.

Bisection Algorithm for Projection

Step 1. Set $\theta_o^u = \bar{q}_o^{max} - \frac{1}{2m_o}$ and $\theta_o^l = \bar{q}_o^{min} - \frac{1}{2m_o}$.

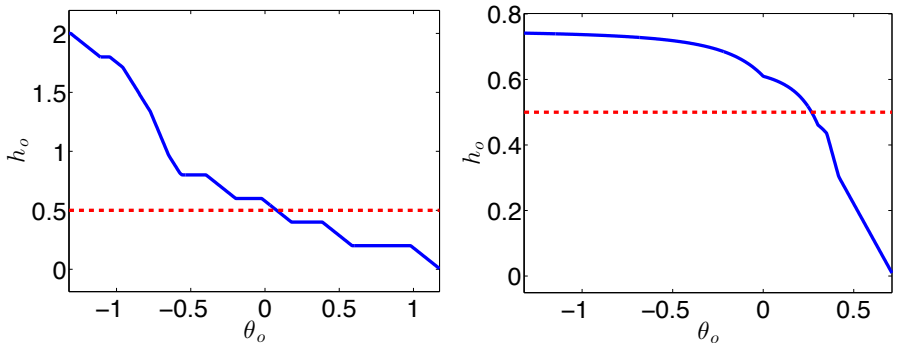

Figure 4: Illustration of the function $h_o(\theta_o) := \sum_{i\in M_o} q_i(\theta_o), \quad o \in \{+,-\}$. The left panel illustrates the case where $q_i(\theta_o) := \min\left\{[\bar{q}_i - \theta_o]_+, \frac{1}{m\nu}\right\}$ (as in Section 4.2). In this case, the function h_o is piecewise linear. The right panel illustrates the case where $q_i(\theta_o) = \min\left\{1, \kappa/(\sum_{i\in M_o}[\bar{q}_i - \bar{\theta}]_+^2)^{\frac{1}{2}}\right\}[\bar{q}_i - \bar{\theta}]_+$ (as in Section 4.3). In both cases, $h_o(\theta_o)$ is a non-increasing function.

Step 2. Set $\hat{\theta}_o = (\theta_o^u + \theta_o^l)/2$

Step 3. Compute $h_o(\hat{\theta}_o)$.

Step 4. If $h_o(\hat{\theta}_o) = \frac{1}{2}$, then terminate with $\hat{\theta}_o^* = \hat{\theta}_o$.
Else if $h_o(\hat{\theta}_o) < \frac{1}{2}$, then set $\theta_o^u = \hat{\theta}_o$.
Else if $h_o(\hat{\theta}_o) > \frac{1}{2}$, then set $\theta_o^l = \hat{\theta}_o$.

Step 5. If $|\theta_o^u - \theta_o^l| < \epsilon'$, then terminate with $\hat{\theta}_o^* = \hat{\theta}_o$. Else, go to Step 2.

All steps require at most $O(m_o)$ operations. Thus, the computational complexity of the bisection algorithm is at most $O\big(m_o \log(\frac{\bar{q}_o^{max} - \bar{q}_o^{min}}{\epsilon'})\big)$.

As the bisection method narrows the interval (θ_o^l, θ_o^u), some of the terms $q_i(\theta_o)$, $i \in M_o$, can be fixed to 0, $\bar{q}_i - \theta_o$, or $\frac{1}{m\nu}$ for $\theta_o \in (\theta_o^l, \theta_o^u)$. Thus we can refine the computation of $h_o(\hat{\theta}_o)$ at Step 3 by avoiding recalculating certain sums. We divide M_o into the following three disjoint sets for given θ_o^l and θ_o^u satisfying $\theta_o^l < \theta_o^u$:

$$
\begin{aligned}
I_o^E &:= \left\{ i \in M_o \mid \bar{q}_i \geq \theta_o^u + \frac{1}{m\nu} \right\}, \\
I_o^L &:= \{ i \in M_o \mid \bar{q}_i < \theta_o^l \}, \\
U_o &:= M_o \backslash (I_o^E \cup I_o^L).
\end{aligned}
$$

Then we have

$$
h_o(\theta_o) = \sum_{i\in M_o} q_i(\theta_o) = \frac{1}{m\nu}\rho_o' + \sum_{i\in U_o} q_i(\theta_o), \quad \theta_o \in (\theta_o^l, \theta_o^u),
$$

where $\rho'_o = |I_o^E|$. When $\hat{\theta}_o \in (\theta_o^l, \theta_o^u)$ is fixed at Step 2, we also partition the set U_o into the following two disjoint sets:

$$\begin{aligned} E_o &:= \Big\{ i \in U_o \mid \bar{q}_i \geq \hat{\theta}_o + \frac{1}{m\nu} \Big\}, \\ G_o &:= \{ i \in U_o \mid \hat{\theta}_o + \frac{1}{m\nu} > \bar{q}_i \geq \hat{\theta}_o \}, \\ L_o &:= \{ i \in U_o \mid \hat{\theta}_o > \bar{q}_i \}. \end{aligned}$$

Then we have

$$\begin{aligned} \sum_{i \in U_o} q_i(\hat{\theta}_o) &= \sum_{i \in E_o} q_i(\hat{\theta}_o) + \sum_{i \in G_o} q_i(\hat{\theta}_o) \\ &= \frac{1}{m\nu} \Delta\rho'_o + (\Delta s_o - \Delta\rho_o \hat{\theta}_o), \quad \hat{\theta}_o \in (\theta_o^l, \theta_o^u), \end{aligned}$$

where $\Delta\rho'_o = |E_o|$, $\Delta s_o = \sum_{i \in G_o} \bar{q}_i$ and $\Delta\rho_o = |G_o|$. Hence, we have

$$h_o(\hat{\theta}_o) = (\rho'_o + \Delta\rho'_o) \frac{1}{m\nu} + (\Delta s_o - \Delta\rho_o \hat{\theta}_o), \quad \hat{\theta}_o \in (\theta_o^l, \theta_o^u).$$

By leveraging on the structure of h_o, the bisection method for (14) can be described as Algorithm 2.

Algorithm 2 Bisection Algorithm for (14)

INPUT: $\bar{\boldsymbol{q}}_o,\ \epsilon' > 0$ OUTPUT: $\boldsymbol{q}_o$
INITIALIZE: $U_o = M_o,\ s_o = 0,\ \rho_o = 0,\ \rho'_o = 0,\ \theta_o^u = \bar{q}_o^{max} - \frac{1}{2m_o},\ \theta_o^l = \bar{q}_o^{min} - \frac{1}{2m_o}$
while $|\theta_o^u - \theta_o^l| > \epsilon'$ **do**
  $\hat{\theta}_o = \frac{\theta_o^u + \theta_o^l}{2}$ # Step 2
  $L_o = \{ i \in U_o \mid \bar{q}_i < \hat{\theta}_o \}$, $\hat{G}_o = \{ i \in U_o \mid \bar{q}_i \geq \hat{\theta}_o \}$ # Step 3
  $E_o = \{ i \in \hat{G}_o \mid \bar{q}_i - \hat{\theta}_o \geq \frac{1}{m\nu} \}$, $G_o = \hat{G}_o \backslash E_o$
  $\Delta\rho'_o = |E_o|,\ \Delta\rho_o = |G_o|$
  $\Delta s_o = \sum_{i \in G_o} \bar{q}_i$
  $val = (\rho'_o + \Delta\rho'_o) \frac{1}{m\nu} + (\Delta s_o - \Delta\rho_o \hat{\theta}_o)$
  if $val < \frac{1}{2}$ **then** # Step 4
    $\rho'_o = \rho'_o + \Delta\rho'_o$
    $\theta_o^u = \hat{\theta}_o,\ U_o = L_o$
  else if $val > \frac{1}{2}$ **then**
    $\theta_o^l = \hat{\theta}_o,\ U_o = G_o$
  else
    break
  end if
end while
$q_i = [\min\{\bar{q}_i - \hat{\theta}_o, \frac{1}{m\nu}\}]_+,\ \forall i \in M_o$ # Step 5

4.3 Projection onto a Simplex with Separable Norm Constraints

As in the former section, we denote by $\boldsymbol{q}_+$ and $\boldsymbol{q}_-$ the subvectors of $\boldsymbol{q} \in \mathbb{R}^m$ corresponding to the label $+1$ and -1, respectively. $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$ are similarly defined subvectors of $\boldsymbol{e}$. Let us consider the projection $P_{\mathcal{U}'}$ onto the following set:

$$\mathcal{U}' := \left\{ \boldsymbol{q} \;\middle|\; \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \|\boldsymbol{q}_o\|^2 \le \kappa_o^2, o \in \{+,-\},\ \boldsymbol{q} \ge \boldsymbol{0} \right\},$$

where $\kappa_o \in \left(0, \sqrt{\frac{1}{2}}\right]$, $o \in \{+,-\}$, are positive parameters. Such $\mathcal{U}'$ appears in a biased variant of ℓ_2-loss ν-SVM (see the equation (43) in Example 3 in the Appendix).

We can construct an efficient algorithm based on the bisection method for the projection $P_{\mathcal{U}'}$:

$$\min_{\boldsymbol{q}} \left\{ \frac{1}{2}\|\boldsymbol{q} - \bar{\boldsymbol{q}}\|^2 \;\middle|\; \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \|\boldsymbol{q}_o\|^2 \le \kappa_o^2,\ o \in \{+,-\},\ \boldsymbol{q} \ge \boldsymbol{0} \right\}. \tag{15}$$

The problem (15) can be decomposed into the following two problems.

$$\min_{\boldsymbol{q}_o} \left\{ \frac{1}{2}\|\boldsymbol{q}_o - \bar{\boldsymbol{q}}_o\|^2 \;\middle|\; \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \|\boldsymbol{q}_o\|^2 \le \kappa_o^2,\ \boldsymbol{q}_o \ge \boldsymbol{0} \right\}, \quad o \in \{+,-\}. \tag{16}$$

Here we assume that $\kappa_o \in (\kappa_o^{min}, \sqrt{\frac{1}{2}})$, where $\kappa_o^{min} = \frac{1}{2\sqrt{m_o}}$. Note that if $\kappa_o^{min} > \kappa_o$, then the problem (16) is infeasible; if $\kappa_o^{min} = \kappa_o$, then $\frac{1}{2m_o}\boldsymbol{e}$ is the unique feasible (and optimal) solution; and if $\kappa_o \ge \sqrt{\frac{1}{2}}$, then the constraint $\|\boldsymbol{q}_o\|^2 \le \kappa_o^2$ is redundant. We have the following lemma by invoking the KKT conditions for (16) as in [Helgason et al., 1980].

Lemma 3. *Suppose that $\kappa_o \in (\kappa_o^{min}, \sqrt{\frac{1}{2}})$. Let*

$$q_i(\theta) = \min\left\{1, \frac{\kappa_o}{(\sum_{i \in M_o} [\bar{q}_i - \theta_o]_+^2)^{1/2}}\right\}[\bar{q}_i - \theta_o]_+, \quad \forall i \in M_o, \quad o \in \{+,-\}$$

and

$$h_o(\theta_o) = \sum_{i \in M_o} q_i(\theta_o), \quad o \in \{+,-\}.$$

For $o \in \{+,-\}$, the following statements hold.

1. *$h_o(\theta_o)$ is continuous and non-increasing.*
2. *There exists $\theta_o^* \in (\bar{q}_o^{min} - \frac{0.5}{m_o}, \bar{q}_o^{max} - \max\{\frac{0.5}{m_o}, \frac{\kappa_o}{\sqrt{m_o}}\})$ such that $h_o(\theta^*) = \frac{1}{2}$, where $\bar{q}_o^{max} = \max\{\bar{q}_i \mid i \in M_o\}$ and $\bar{q}_o^{min} = \min\{\bar{q}_i \mid i \in M_o\}$.*
3. *Let $\hat{q}_i = q_i(\theta_o^*),\ \forall i \in M_o$. $\hat{\boldsymbol{q}}_o$ is an optimal solution of (16).*

Proof. If $1 < \kappa_o/(\sum_{i\in M_o}[\bar{q}_i - \theta_o]_+^2)^{\frac{1}{2}}$, it is obvious that $h_o(\theta_o) = \sum_{i\in M_o}[\bar{q}_i - \theta_o]_+$ is non-increasing. Consider the case where $1 \geq \kappa_o/(\sum_{i\in M_o}[\bar{q}_i - \theta]_+^2)^{\frac{1}{2}}$. Let $E_{\theta_o} = \{i \in M_o \mid \bar{q}_i > \theta_o\}$. Let us denote by $\bar{\boldsymbol{q}}_{E_{\theta_o}}$ the subvectors of $\bar{\boldsymbol{q}}_o$ corresponding to the index $i \in E_{\theta_o}$. Then, we have

$$h_o(\theta_o) = \frac{\kappa_o \sum_{i\in M_o}[\bar{q}_i - \theta_o]_+}{(\sum_{i\in M_o}[\bar{q}_i - \theta_o]_+^2)^{1/2}} = \frac{\kappa_o \sum_{i\in E_{\theta_o}}(\bar{q}_i - \theta_o)}{(\sum_{i\in E_{\theta_o}}(\bar{q}_i - \theta_o)^2)^{1/2}} = \frac{\kappa_o \boldsymbol{e}_{E_{\theta_o}}^\top(\bar{\boldsymbol{q}}_{E_{\theta_o}} - \theta_o \boldsymbol{e}_{E_{\theta_o}})}{\|\bar{\boldsymbol{q}}_{E_{\theta_o}} - \theta_o \boldsymbol{e}_{E_{\theta_o}}\|}.$$

For notational simplicity, we omit the subscript E_{θ_o} in the following. Then, we have the derivative $h_o'(\theta_o)$ as

$$h_o'(\theta_o)/\kappa_o = \frac{-\|\boldsymbol{e}\|^2\|\bar{\boldsymbol{q}} - \theta_o\boldsymbol{e}\| + \left\{\boldsymbol{e}^\top(\bar{\boldsymbol{q}} - \theta_o\boldsymbol{e})\right\}^2/\|\bar{\boldsymbol{q}} - \theta_o\boldsymbol{e}\|}{\|\bar{\boldsymbol{q}} - \theta_o\boldsymbol{e}\|^2} = \frac{-\|\boldsymbol{e}\|^2\|\bar{\boldsymbol{q}}\|^2 + (\bar{\boldsymbol{q}}^\top\boldsymbol{e})^2}{\|\bar{\boldsymbol{q}} - \theta_o\boldsymbol{e}\|^3}.$$

From the Cauchy-Schwarz inequality:

$$(\bar{\boldsymbol{q}}^\top\boldsymbol{e})^2 \leq \|\bar{\boldsymbol{q}}\|^2\|\boldsymbol{e}\|^2,$$

we have $h_o'(\theta_o) \leq 0$. The second part of the lemma holds since $h_o(\bar{q}_o^{min} - \frac{1}{2m_o}) \geq \frac{1}{2}$, $h_o(\bar{q}_o^{max} - \frac{1}{2m_o}) \leq \frac{1}{2}$, and $h_o(\theta)$ is continuous. Next, we show the third part of the lemma. If $\kappa_o > \frac{1}{2\sqrt{m_o}}$, the problem (16) satisfies the Slater condition, i.e. there exists a feasible relative interior point $\frac{1}{2m_o}\boldsymbol{e}_o$. Under the Slater condition, $\hat{\boldsymbol{q}}_o$ is an optimal solution of (16) if and only if $\hat{\boldsymbol{q}}_o$ satisfies the KKT conditions, i.e., there exists $(\theta_o, \boldsymbol{\zeta}_o, \lambda_o)$ such that

$$\begin{cases} (1+\lambda_o)\hat{\boldsymbol{q}}_o - \bar{\boldsymbol{q}}_o + \theta_o\boldsymbol{e}_o - \boldsymbol{\zeta}_o = \boldsymbol{0}, \\ \boldsymbol{\zeta}_o^\top\hat{\boldsymbol{q}}_o = 0, \quad \lambda_o(\|\hat{\boldsymbol{q}}_o\|^2 - \kappa_o^2) = 0, \\ \hat{\boldsymbol{q}}_o \geq \boldsymbol{0}, \quad \hat{\boldsymbol{q}}_o^\top\boldsymbol{e}_o = \dfrac{1}{2}, \quad \|\hat{\boldsymbol{q}}_o\|^2 \leq \kappa_o^2, \\ \boldsymbol{\zeta}_o \geq \boldsymbol{0}, \quad \lambda_o \geq 0. \end{cases}$$

$\hat{q}_i = q_i(\theta_o^*)$ satisfies the KKT conditions by

$$\theta = \theta_o^*, \qquad \zeta_i = [\theta_o^* - \bar{q}_i]_+, \ (i \in M_o)$$
$$\lambda_o = \min\left\{1, \frac{\kappa_o}{(\sum_{i\in M_o}[\bar{q}_i - \theta_o^*]_+^2)^{1/2}}\right\}^{-1} - 1.$$

This completes the proof. □

Therefore, the problem (16) can be reduced to the problem of finding a solution θ_o^* of the equation $h_o(\theta_o^*) = \frac{1}{2}$. It can be computed numerically by the bisection method since $h_o(\theta_o)$ is non-increasing (as illustrated in Figure 4). Algorithm 3 shows the bisection algorithm for (16). It requires $O(m_o)$ operations at each iteration. Since $\kappa_o \leq \sqrt{\frac{1}{2}}$, Algorithm 3 iterates at most $\lceil \log_2(\frac{d}{\epsilon'}) \rceil$ steps, where $d = \bar{q}_o^{max} - \bar{q}_o^{min} + \sqrt{\frac{1}{2}}$. Thus the computational complexity of Algorithm 3 is at most $O\left(m_o \log\left(\frac{d}{\epsilon'}\right)\right)$.

Algorithm 3 Bisection Algorithm for (16)

INPUT: $\bar{\boldsymbol{q}}_o,\ \epsilon' > 0$ OUTPUT: $\boldsymbol{q}_o$

INITIALIZE: $U_o = M_o,\ \theta_o^u = q_{max} - \frac{1}{2m_o},\ \theta_o^l = q_{min} - \max\{\frac{1}{2m_o}, \frac{\kappa_o}{\sqrt{m_o}}\}$

while $|\theta_o^u - \theta_o^l| > \epsilon'$ **do**

  $\hat{\theta}_o = \frac{\theta_o^u + \theta_o^l}{2}$ # Step 2

  $G_o = \{i \in U_o \mid \bar{q}_i \geq \hat{\theta}_o\},\ L_o = \{i \in U_o \mid \bar{q}_i < \hat{\theta}_o\}$ # Step 3

  $val = h_o(\hat{\theta}_o)$

  if $val < \frac{1}{2}$ **then** # Step 4

    $U_o = L_o,\ \theta_o^u = \hat{\theta}_o$

  else if $val > \frac{1}{2}$ **then**

    $U_o = G_o,\ \theta_o^l = \hat{\theta}_o$

  else

    break

  end if

end while

$q_i = \min\left\{1, \kappa_o / (\sum_{i \in M_o} [\bar{q}_i - \bar{\theta}]_+^2)^{\frac{1}{2}}\right\} [\bar{q}_i - \bar{\theta}]_+,\ \forall i \in M_o$ # Step 5

4.4 Complex Uncertainty Sets

For the problem (3) or (4) with complex uncertainty sets which may not satisfy Assumption 2, we can replace (3) or (4) with a sequence of subproblems that meet Assumption 2 by applying the augmented Lagrangian method. For instance, let us consider the uncertainty set

$$\mathcal{U}' = \{\boldsymbol{q} \mid \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+, -\},\ \|\boldsymbol{q}\|^2 \leq \kappa^2,\ \boldsymbol{q} \geq \boldsymbol{0}\}.$$

Such $\mathcal{U}'$ appears in ℓ_2-loss ν-SVM (see the equation (39) in Example 3 in the Appendix). Letting $S = \{\boldsymbol{q} \mid \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+, -\},\ \boldsymbol{q} \geq \boldsymbol{0}\}$ and $T = \{\boldsymbol{q} \mid \|\boldsymbol{q}\|^2 \leq \kappa^2\}$, we have $\mathcal{U}' = S \cap T$. As shown in Section 4.2, the projection P_S onto S can be computed by the bisection method. Hence, the subproblem (9) in Step 1 of the augmented Lagrangian method in Section 3 can be solved efficiently by the APG method.

5 Relation to Existing Unified Classification Models

Besides UCM (2), there are several unified formulations of binary classification models. In this section, we show that UCM (2) includes three unified formulations: the model minimizing coherent risk measure [Bertsimas and Takeda, 2014], the generalized ν-SVM [Kanamori et al., 2013], and the model of minimizing coherent classification loss function (CCLF)[†] [Yang et al., 2014].

[†]Note that coherent risk and coherent loss are derived from different ideas although their names are similar.

5.1 Classification Models based on Coherent Risk Measure Minimization

Define

$$\boldsymbol{v} = \begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix}, \quad \boldsymbol{z}_i = \begin{pmatrix} y_i \boldsymbol{x}_i \\ y_i \end{pmatrix}, \quad i \in M.$$

Then $\boldsymbol{v}^\top \boldsymbol{z}_i = y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 0$ means that the label y_i of the sample $\boldsymbol{x}_i$ is correctly predicted. Assume that the samples $\boldsymbol{x}_i,\ i \in M$, are distributed with an unknown distribution. The purpose here is to compute $\boldsymbol{v}$ such that the values $\boldsymbol{v}^\top \boldsymbol{z}_i,\ i \in M$, are large, and the resulting decision function $h(\boldsymbol{x})$ can correctly predict the label for a new input point $\hat{\boldsymbol{x}} \in \mathcal{X}$ which is distributed with the same unknown distribution. The possibility of misclassification (i.e., $\boldsymbol{v}^\top \boldsymbol{z} < 0$) can be regarded as a risk. Bertsimas and Takeda [2014] measured the risk of $\boldsymbol{v}$ by the coherent risk measure [Artzner et al., 1999], which is a class of measure functions and extensively studied in the field of financial engineering.

5.1.1 Coherent Risk Measure

Let $\mathcal{V}$ be a linear space of random variables defined on an appropriate probability space. The coherent risk measure is a class of measure functions defined as follows.

Definition 1. *[Artzner et al., 1999] A function $\mu : \mathcal{V} \to \mathbb{R}$ that satisfies the following four conditions for all random variables $\tilde{v}, \tilde{w} \in \mathcal{V}$ is called a coherent risk measure.*

- *Monotonicity: If $\tilde{v} \geq \tilde{w}$, then $\mu(\tilde{v}) \leq \mu(\tilde{w})$.*
- *Translation invariance: $\mu(\tilde{v} + a) = \mu(\tilde{v}) - a,\ \forall a \in \mathbb{R}$.*
- *Subadditivity: $\mu(\tilde{v} + \tilde{w}) \leq \mu(\tilde{v}) + \mu(\tilde{w})$.*
- *Positive homogeneity: $\mu(\lambda \tilde{v}) = \lambda \mu(\tilde{v}), \quad \forall \lambda \geq 0$.*

Note that subadditivity and positive homogeneity imply convexity. For a coherent risk measure μ and a $\mathbb{R}^n$-valued random variable $\tilde{\boldsymbol{z}}$, we consider minimizing coherent risk measures:

$$\inf_{\boldsymbol{v} \in V} \mu(\boldsymbol{v}^\top \tilde{\boldsymbol{z}}), \tag{17}$$

where V is a closed convex set.

It is well-known that any coherent risk measure can be equivalently described in terms of the worst-case expectation over a family of distributions $\mathcal{Q}$, and any family of distributions $\mathcal{Q}$ defines a coherent risk measure. Therefore, the coherent risk minimization problem (17) can be written as follows (see, e.g., representation theorem for coherent risk measures in [Artzner et al., 1999]):

$$\inf_{\boldsymbol{v} \in V} \sup_{q \in \mathcal{Q}} \mathbb{E}_q(-\boldsymbol{v}^\top \tilde{\boldsymbol{z}}), \tag{18}$$

where $\mathbb{E}_q(\tilde{v})$ denotes the expectation of the random variable $\tilde{v}$ under a distribution q.

Here we assume finite probability space for $\tilde{\boldsymbol{z}}$ by following [Bertsimas and Brown, 2009]. Denote the support of $\tilde{\boldsymbol{z}}$ by $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ and let $Z = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m]$. Then (18) is equivalent to

$$\inf_{\boldsymbol{v} \in V} \max_{\boldsymbol{q} \in \mathcal{Q}} \ -\boldsymbol{v}^\top Z \boldsymbol{q}. \tag{19}$$

Note that $\mathcal{Q} \subseteq \Delta := \{\boldsymbol{q} \in \mathbb{R}^m \mid \boldsymbol{q}^\top \boldsymbol{e} = 1,\ \boldsymbol{q} \geq \boldsymbol{0}\}$.

5.1.2 Binary Classification Model

Now we consider a machine learning model which minimizes a coherent risk measure by assuming an appropriate $\mathcal{Q} \subseteq \Delta$. Suppose that $V = \{\boldsymbol{v} = (\boldsymbol{w}, b) \mid \|\boldsymbol{w}\| \leq 1\}$. Then (19) is equivalent to

$$\min_{b, \|\boldsymbol{w}\| \leq 1} \ \max_{\boldsymbol{q} \in \mathcal{Q}} \ -\boldsymbol{w}^\top \widetilde{Z} \boldsymbol{q} - b \boldsymbol{y}^\top \boldsymbol{q} \Big(= -\boldsymbol{v}^\top Z \boldsymbol{q} \Big), \tag{20}$$

where $\widetilde{Z} := [y_1 \boldsymbol{x}_1, \ldots, y_m \boldsymbol{x}_m]$. The problem (20) is investigated in [Gotoh et al., 2014] as a relaxed counterpart of the coherent risk based-classification. We further reformulate it as follows:

$$\begin{aligned} &\max_{\boldsymbol{q} \in \mathcal{Q}} \ \min_{b, \|\boldsymbol{w}\| \leq 1} -\boldsymbol{w}^\top \widetilde{Z} \boldsymbol{q} - b \boldsymbol{y}^\top \boldsymbol{q} \\ &= -\min \Big\{ \|\widetilde{Z} \boldsymbol{q}\| \ \Big| \ \boldsymbol{q} \in \widetilde{\mathcal{Q}} \Big\}, \end{aligned} \tag{21}$$

where $\widetilde{\mathcal{Q}} = \{\boldsymbol{q} \in \mathcal{Q} \mid \boldsymbol{y}^\top \boldsymbol{q} = 0\}$. Note that

$$\begin{aligned} \widetilde{\mathcal{Q}} \subseteq \widetilde{\Delta} :=& \{\boldsymbol{q} \mid \boldsymbol{e}^\top \boldsymbol{q} = 1,\ \boldsymbol{q} \geq \boldsymbol{0},\ \boldsymbol{y}^\top \boldsymbol{q} = 0\} \\ =& \{\boldsymbol{q} \mid \boldsymbol{q} \geq \boldsymbol{0},\ \boldsymbol{e}_o^\top \boldsymbol{q}_o = \frac{1}{2},\ o \in \{+, -\}\}. \end{aligned}$$

Here, $\boldsymbol{q}_+$ and $\boldsymbol{q}_-$ denote the subvectors of $\boldsymbol{q}$ corresponding to the label $+1$ and -1, respectively. $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$ are similarly defined subvectors of $\boldsymbol{e}$. The second form of $\widetilde{\Delta}$ comes from the fact that $\boldsymbol{y}^\top \boldsymbol{q} = 0$ is equivalent to $\boldsymbol{q}_+^\top \boldsymbol{e}_+ = \boldsymbol{q}_-^\top \boldsymbol{e}_-$. An optimal solution $\boldsymbol{w}^*$ of (20) can be obtained by the optimality condition for $\boldsymbol{w}$: $\boldsymbol{w}^* = \widetilde{Z} \boldsymbol{q}^* / \|\widetilde{Z} \boldsymbol{q}^*\|$, where $\boldsymbol{q}^*$ is an optimal solution of (21). b is derived from the Lagrange multiplier corresponding to $\boldsymbol{y}^\top \boldsymbol{q} = 0$.

Let $\mathcal{C}$ be the Minkovski difference of the convex hulls:

$$\begin{aligned} \mathcal{C} :&= \frac{1}{2} \Big\{ \mathrm{conv}\big(\{\boldsymbol{x}_i \mid i \in M_+\}\big) \ominus \mathrm{conv}\big(\{\boldsymbol{x}_i \mid i \in M_-\}\big) \Big\} \\ &= \Big\{ \sum_{i \in M_+} q_i \boldsymbol{x}_i \mid \boldsymbol{q}_+^\top \boldsymbol{e}_+ = \frac{1}{2},\ \boldsymbol{q}_+ \geq \boldsymbol{0} \Big\} \ominus \Big\{ \sum_{i \in M_-} q_i \boldsymbol{x}_i \mid \boldsymbol{q}_-^\top \boldsymbol{e}_- = \frac{1}{2},\ \boldsymbol{q}_- \geq \boldsymbol{0} \Big\} \\ &= \Big\{ \widetilde{Z} \boldsymbol{q} \mid \boldsymbol{q} \in \widetilde{\Delta} \Big\}. \end{aligned}$$

Now we have the next proposition.

Proposition 1. *For any $\widetilde{\mathcal{Q}}$, the coherent risk minimization problem* (21) *is equivalent to UCM* (3) *with $\mathcal{U} = \{\widetilde{Z}\boldsymbol{q} \mid \boldsymbol{q} \in \widetilde{\mathcal{Q}}\}$. Conversely, for any $\mathcal{U} \subseteq \mathcal{C}$, UCM* (3) *is equivalent to the coherent risk minimization problem* (21) *with $\widetilde{\mathcal{Q}} = \{\boldsymbol{q} \in \widetilde{\Delta} \mid \widetilde{Z}\boldsymbol{q} \in \mathcal{U}\}$.*

Proof. The proposition can be proven by mimicking the proof of [Bertsimas and Brown, 2009, Theorem 3.1]. For any $\widetilde{\mathcal{Q}}$, we have

$$\begin{aligned}\min_{\boldsymbol{q}} \{\|\widetilde{Z}\boldsymbol{q}\| \mid \boldsymbol{q} \in \widetilde{\mathcal{Q}}\} &= \min_{\boldsymbol{x},\boldsymbol{q}} \{\|\boldsymbol{x}\| \mid \boldsymbol{x} = \widetilde{Z}\boldsymbol{q},\ \boldsymbol{q} \in \widetilde{\mathcal{Q}}\} \\ &= \min_{\boldsymbol{x}} \{\|\boldsymbol{x}\| \mid \boldsymbol{x} \in \{\widetilde{Z}\boldsymbol{q} \mid \boldsymbol{q} \in \widetilde{\mathcal{Q}}\}\} \\ &= \min_{\boldsymbol{x}} \{\|\boldsymbol{x}\| \mid \boldsymbol{x} \in \mathcal{U}\}.\end{aligned}$$

Conversely, for any $\mathcal{U} \subseteq \mathcal{C}$, we have

$$\begin{aligned}\min_{\boldsymbol{x}} \{\|\boldsymbol{x}\| \mid \boldsymbol{x} \in \mathcal{U}\} &= \min_{\boldsymbol{q},\boldsymbol{x}} \{\|\boldsymbol{x}\| \mid \boldsymbol{q} \in \widetilde{\Delta},\ \widetilde{Z}\boldsymbol{q} = \boldsymbol{x} \in \mathcal{U}\} \\ &= \min_{\boldsymbol{q}} \{\|\widetilde{Z}\boldsymbol{q}\| \mid \boldsymbol{q} \in \{\boldsymbol{q} \in \widetilde{\Delta} \mid \widetilde{Z}\boldsymbol{q} \in \mathcal{U}\}\} \\ &= \min_{\boldsymbol{q}} \{\|\widetilde{Z}\boldsymbol{q}\| \mid \boldsymbol{q} \in \widetilde{\mathcal{Q}}\}.\end{aligned}$$

□

Proposition 1 states that any coherent risk minimization problem can be reduced to UCM, and UCM minimizes a coherent risk measure if $\mathcal{U} \subseteq \mathcal{C}$. The models in Examples 2–4 actually satisfy $\mathcal{U} \subseteq \mathcal{C}$ and minimize coherent risk measures. For instance, Takeda and Sugiyama [2008] showed that ν-SVM (Example 2) minimizes the coherent risk measure known as conditional value-at-risk (CVaR).

On the other hand, UCM with $\mathcal{U} \not\subseteq \mathcal{C}$ does not minimize the coherent risk measure. For such a situation, we can make a coherent version of the UCM by restricting $\mathcal{U}$ to a subset of $\mathcal{C}$. See Example 5 in the Appendix.

5.2 Generalized ν-SVM

Let $l : \mathbb{R} \to \mathbb{R}$ be a closed, convex and proper loss function. The generalized ν-SVM [Kanamori et al., 2013] is formulated as follows:

$$\inf_{\|\boldsymbol{w}\|^2 \leq \lambda^2, b, \rho} \Big\{ -\rho + \frac{1}{m} \sum_{i=1}^{m} l(\rho - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)) \Big\}, \tag{22}$$

where $\lambda > 0$ is a parameter. Note that (22) is equivalent to ν-SVM if $l(u) = [u/\nu]_+$, where $\nu \in (\nu_{min}, \nu_{max}]$ is a parameter. The consistent estimator of the conditional probability $P(y \mid \boldsymbol{x})$ of the label y is available for the model (22). See [Kanamori et al., 2013] for details.

The Lagrangian function $\mathcal{L}$ associated with (22) is given by:

$$\begin{aligned}&\mathcal{L}(\boldsymbol{w}, b, \rho, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mu)\\ &\quad = -\rho + \frac{1}{m}\sum_{i\in M} l(\xi_i) + \sum_{i\in M}\alpha_i\big(\rho - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - \xi_i\big) + \mu(\|\boldsymbol{w}\|^2 - \lambda^2),\\ &\qquad\qquad (\boldsymbol{\alpha}\geq \mathbf{0},\ \mu \geq 0).\end{aligned}$$

Then, (22) is equivalent to

$$\min_{\boldsymbol{w},b,\rho,\boldsymbol{\xi}} \max_{\alpha\geq\mathbf{0},\mu\geq 0} \mathcal{L}(\boldsymbol{w}, b, \rho, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mu),$$

and its dual problem is given by

$$\begin{aligned}&\max_{\alpha\geq\mathbf{0},\mu\geq 0}\min_{\boldsymbol{w},b,\rho,\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{w}, b, \rho, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mu)\\ &\quad = \max_{\boldsymbol{\alpha}\geq\mathbf{0},\mu\geq 0}\min_{\boldsymbol{w},b,\rho,\boldsymbol{\xi}} \Big\{\Big(\sum_{i\in M}\alpha_i - 1\Big)\rho + \frac{1}{m}\sum_{i\in M}\big(l(\xi_i) - m\alpha_i\xi_i\big)\\ &\qquad\qquad - \sum_{i\in M}\alpha_i y_i(\boldsymbol{w}^\top\boldsymbol{x}_i + b) + \mu(\|\boldsymbol{w}\|^2 - \lambda^2)\Big\}. \quad (23)\end{aligned}$$

Let μ^* be the optimal solution of (23) with respect to μ. Here we detect whether $\mu^* = 0$ or $\mu^* > 0$. Fix $\mu = 0$. Then, the dual problem (23) is reduced to

$$\begin{aligned}&\max_{\alpha\geq\mathbf{0}}\min_{\boldsymbol{w},b,\rho,\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{w}, b, \rho, \boldsymbol{\xi}, \boldsymbol{\alpha}, 0)\\ &= -\min_{\boldsymbol{\alpha}\in\tilde{\Delta}}\Big\{\frac{1}{m}\sum_{i\in M} l^*(m\alpha_i)\ \Big|\ \sum_{i\in M}\alpha_i y_i\boldsymbol{x}_i = \mathbf{0}\Big\}\\ &= -\min_{\boldsymbol{\alpha}\in\tilde{\Delta}}\Big\{\frac{1}{m}\sum_{i\in M} l^*(m\alpha_i) + \lambda\Big\|\sum_{i\in M}\alpha_i y_i\boldsymbol{x}_i\Big\|\ \Big|\ \sum_{i\in M}\alpha_i y_i\boldsymbol{x}_i = \mathbf{0}\Big\},\end{aligned} \quad (24)$$

where $l^*(q)$ be the convex conjugate of $l(u)$, i.e. $l^*(q) = \sup_u\{uq - l(u)\}$. On the contrary, assuming that $\mu > 0$, the dual problem (23) is reduced to

$$\begin{aligned}&\max_{\alpha\geq\mathbf{0},\mu>0}\min_{\boldsymbol{w},b,\rho,\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{w}, b, \rho, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mu)\\ &= \max_{\boldsymbol{\alpha}\geq\mathbf{0},\mu>0}\Big\{-\frac{1}{m}\sum_{i\in M} l^*(m\alpha_i) - \frac{1}{4\mu}\Big\|\sum_{i\in M}\alpha_i y_i\boldsymbol{x}_i\Big\|^2 - \mu\lambda^2\ \Big|\ \sum_{i\in M}\alpha_i = 1,\ \sum_{i\in M}\alpha_i y_i = 0\Big\}\\ &= -\min_{\boldsymbol{\alpha}\in\tilde{\Delta}}\Big\{\frac{1}{m}\sum_{i\in M} l^*(m\alpha_i) + \lambda\Big\|\sum_{i\in M}\alpha_i y_i\boldsymbol{x}_i\Big\|\Big\}.\end{aligned} \quad (25)$$

Since (25) has a less constraint than (24), the value of (25) is larger than or equals to (24). Hence, we can assume $\mu^* > 0$, and (25) is actually the dual problem of the generalized ν-SVM. An optimal solution $\boldsymbol{w}^*$ of (20) can be obtained by the optimality condition: $\boldsymbol{w}^* = \lambda\sum_{i\in M}\alpha_i^* y_i\boldsymbol{x}_i/\|\sum_{i\in M}\alpha_i^* y_i\boldsymbol{x}_i\|$, where $\boldsymbol{\alpha}^*$ is an optimal solution of (25). The optimal b can be derived from the Lagrange multiplier corresponding to $\boldsymbol{y}^\top\boldsymbol{\alpha} = 0$.

From these results, we have the following proposition.

Proposition 2. *Let $\boldsymbol{\alpha}^*$ be an optimal solution of the dual* (25) *of generalized ν-SVM. Assume that we have*

$$\widetilde{\mathcal{Q}} = \Big\{ \boldsymbol{q} \in \widetilde{\Delta} \;\Big|\; \frac{1}{m\lambda} \sum_{i \in M} l^*(mq_i) \leq \kappa^* \Big\},$$

a priori, where $\kappa^* = \frac{1}{m\lambda} \sum_{i \in M} l^*(m\alpha_i^*)$. Then, $\boldsymbol{q} = \boldsymbol{\alpha}^*$ is also an optimal solution of the coherent risk minimization problem (21)*, i.e.* (21) *and* (25) *lead to the same decision function.*

Proof. For any $\lambda > 0$, the dual problem (25) of the generalized ν-SVM is equivalent to the following problem:

$$\min_{\boldsymbol{\alpha} \in \widetilde{\Delta}} \Big\{ \frac{1}{m\lambda} \sum_{i \in M} l^*(m\alpha_i) + \Big\| \sum_{i \in M} \alpha_i y_i \boldsymbol{x}_i \Big\| \Big\}.$$

Then, the statement holds from Lemma 5 in the Appendix by letting

$$f(\boldsymbol{\alpha}) = \Big\| \sum_{i \in M} \alpha_i y_i \boldsymbol{x}_i \Big\|, \quad g(\boldsymbol{\alpha}) = \frac{1}{m\lambda} \sum_{i \in M} l^*(m\alpha_i).$$

□

In the sense of Proposition 2, the coherent risk minimization model can cover the generalized ν-SVM.

5.3 Coherent Classification Loss Function

Define the 0–1 loss function:

$$I(u) = \begin{cases} 1 & \text{if } u < 0 \\ 0 & \text{otherwise.} \end{cases}$$

Here we consider the following classification model:

$$\min_{\boldsymbol{v} \in V} \; L(Z^\top \boldsymbol{v}) := \frac{1}{m} \sum_{i=1}^{m} I(\boldsymbol{z}_i^\top \boldsymbol{v}). \tag{26}$$

The problem (26) finds the decision hyperplane which minimizes the number of misclassifications for the training data. However, minimizing the non-convex loss function $L(\boldsymbol{u})$ is typically computationally intractable. Recently, Yang et al. [2014] proposed the classification model that replaces $L(\boldsymbol{u})$ by the coherent classification loss function (CCLF) having five salient properties derived from $L(\boldsymbol{u})$.

Definition 2 (Yang et al. [2014])**.** *Let $\rho : \mathbb{R}^m \to \mathbb{R}$ be a quasi-convex and lower semi-continuous function. The function ρ which satisfies the following five conditions for all $\boldsymbol{u} \in \mathbb{R}^m$ is called a CCLF.*

- *Complete classification:* $\rho(\boldsymbol{u}) = 0$ *if and only if* $\boldsymbol{u} \geq \boldsymbol{0}$.
- *Misclassification avoidance: If* $\boldsymbol{u} < \boldsymbol{0}$, *then* $\rho(\boldsymbol{u}) = 1$.
- *Monotonicity: If* $\boldsymbol{u}_1 \geq \boldsymbol{u}_2$, *then* $\rho(\boldsymbol{u}_1) \leq \rho(\boldsymbol{u}_2)$.
- *Order invariance:* $\rho(\boldsymbol{u}) = \rho(P\boldsymbol{u})$ *for all permutation matrices* P.
- *Scale invariance:* $\rho(\boldsymbol{u}) = \rho(a\boldsymbol{u})$ *for all* $a > 0$.

Here, quasi-convexity is introduced for computational tractability. Note that $L(\boldsymbol{u})$ itself is not CCLF because it does not satisfy quasi-convexity. Definition 1 (coherent risk measure) and Definition 2 (CCLF) have clearly different conditions, e.g. the coherent risk measure is convex, but CCLF is quasi-convex. However, a vector $\boldsymbol{v} = (\boldsymbol{w}, b)$ that minimizes a CCLF $\rho(Z^\top \boldsymbol{v})$ also minimizes a coherent risk measure. To show this, we use the following characterization of the CCLF.

Lemma 4. *Suppose that* $\rho(\boldsymbol{u})$ *is a CCLF. Then there exists a set of probability measures* $\mathcal{Q}_k \subseteq \Delta$ *parameterized by* $k \in [0, 1]$ *such that*

$$\rho(\boldsymbol{u}) = 1 - \sup\bigl\{k \in [0,1] \mid \sup_{\boldsymbol{q} \in \mathcal{Q}_k} (-\boldsymbol{u}^\top \boldsymbol{q}) \leq 0\bigr\},$$

where $\mathcal{Q}_0 = \{\frac{1}{m}\boldsymbol{e}\}$, $\mathcal{Q}_1 = \Delta$, *and* $\mathcal{Q}_k \subseteq \mathcal{Q}_{k'}$ *if* $0 \leq k \leq k' \leq 1$.

The lemma can be verified by the proof of Lemma A-1 in the supplementary material of [Yang et al., 2014]. Now we have the following result.

Proposition 3. *Consider the CCLF minimization problem:*

$$\min_{\boldsymbol{v} \in V} \rho(Z^\top \boldsymbol{v}). \tag{27}$$

The problem (27) *can be reduced to the coherent risk minimization problem* (20). *Moreover, if the interior of* $\{\widetilde{Z}\boldsymbol{q} \mid \boldsymbol{q} \in \widetilde{\Delta}\}$ *contains the origin* $\boldsymbol{0}$, *which implies that the datasets* $\{\boldsymbol{x}_i \mid i \in M_+\}$ *and* $\{\boldsymbol{x}_j \mid j \in M_-\}$ *are linearly non-separable, then* (27) *and* (20) *are equivalent.*

Proof. From Lemma 4, (27) is equivalent to

$$\sup\bigl\{k \in [0,1] \;\big|\; \min_{\boldsymbol{v} \in V} \sup_{\boldsymbol{q} \in \mathcal{Q}_k} (-\boldsymbol{v}^\top Z\boldsymbol{q}) \leq 0\bigr\}. \tag{28}$$

Let k^* be an optimal solution of (28). Clearly, (28) can be reduced to the coherent risk minimization problem (20) by setting $\mathcal{Q} = \mathcal{Q}_{k^*}$. If $\{\widetilde{Z}\boldsymbol{q} \mid \boldsymbol{q} \in \widetilde{\Delta}\}$ contains the origin $\boldsymbol{0}$ in its interior, then we have $\sup_{\boldsymbol{q} \in \Delta}(-\boldsymbol{v}^\top Z\boldsymbol{q}) = \sup_{\boldsymbol{q} \in \widetilde{\Delta}}(-\boldsymbol{w}^\top \widetilde{Z}\boldsymbol{q}) > 0$. Thus (20) can be reduced to (28) by setting

$$\mathcal{Q}_k = \begin{cases} \mathcal{Q} & \text{if } k \leq k^* \\ \Delta & \text{otherwise} \end{cases}.$$

□

Yang et al. [2014] showed that the CCLF:

$$\bar{\rho}(\boldsymbol{u}) = \max\{t \in \{0, 1, \ldots, m\} \mid \textstyle\sum_{i=1}^{t} u_{(i)} < 0\}/m,$$

where $\{u_{(i)}\}_{i=1}^{m}$ is the permutation of $\{u_i\}_{i=1}^{m}$ in ascending order $u_{(1)} \leq \ldots \leq u_{(m)}$, is the tightest upper bound of $L(\boldsymbol{u})$ in the CCLF. The classifier that minimizes the tightest CCLF $\bar{\rho}(Z^\top \boldsymbol{v})$ is computationally tractable and shows a good prediction performance. It can be verified that the classifier minimizing the tightest CCLF $\bar{\rho}(Z^\top \boldsymbol{v})$ is equivalent to ν-SVM with a specific parameter ν.

Proposition 4. *Suppose that*

$$\bar{\rho}(Z^\top \boldsymbol{v}) = \max\left\{t \in \{0, 1, \ldots, m\} \mid \textstyle\sum_{i=1}^{t} \boldsymbol{z}_{(i)}^\top \boldsymbol{v} < 0\right\}/m,$$

where $\{\boldsymbol{z}_{(i)}\}_{i=1}^{m}$ *is the permutation of* $\{\boldsymbol{z}_i\}_{i=1}^{m}$ *such that* $\boldsymbol{z}_{(1)}^\top \boldsymbol{v} \leq \ldots \leq \boldsymbol{z}_{(m)}^\top \boldsymbol{v}$. *Let* $\overline{\mathcal{Q}}_k$ *satisfy the property that* $\overline{\mathcal{Q}}_0 = \{\boldsymbol{q} \mid \boldsymbol{q}^\top \boldsymbol{e} = 1,\ 0 \leq q_i \leq \frac{1}{m}\}$ *if* $k = 0$*; and*

$$\overline{\mathcal{Q}}_k = \left\{\boldsymbol{q} \mid \boldsymbol{q}^\top \boldsymbol{e} = 1,\ 0 \leq q_i \leq \frac{1}{m - (\lceil mk \rceil - 1)}\right\}$$

if $0 < k \leq 1$. *Then we have*

$$\bar{\rho}(Z^\top \boldsymbol{v}) = 1 - \sup\{k \in [0, 1] \mid \sup_{\boldsymbol{q} \in \overline{\mathcal{Q}}_k} (-\boldsymbol{v}^\top Z \boldsymbol{q}) \leq 0\}. \tag{29}$$

Proof. Fix $\boldsymbol{v} \in V$. If $Z^\top \boldsymbol{v} \geq \boldsymbol{0}$, then we have $\bar{\rho}(Z^\top \boldsymbol{v}) = 0$, as well as $\sup_{\boldsymbol{q} \in \overline{\mathcal{Q}}_1} (-\boldsymbol{v}^\top Z \boldsymbol{q}) \leq 0$. Hence the equivalence (29) holds. Next, we suppose $Z^\top \boldsymbol{v} \not\geq 0$, and let $t^0 := \max\{t \mid \sum_{i=1}^{t} \boldsymbol{z}_{(i)}^\top \boldsymbol{v} < 0\}$ (i.e., $\bar{\rho}(Z^\top \boldsymbol{v}) = \frac{t^0}{m}$). For $k^0 = 1 - \frac{t^0}{m}$,

$$\mathcal{Q}_{k^0} = \{\boldsymbol{q} \mid \boldsymbol{q}^\top \boldsymbol{e} = 1,\ 0 \leq q_i \leq \tfrac{1}{t^0 + 1},\ \forall i \in M\}.$$

Noting that $\sum_{i=1}^{t^0+1} \boldsymbol{z}_{(i)}^\top \boldsymbol{v} \geq 0$ holds from the definition of t^0, we obtain

$$\sup_{\boldsymbol{q} \in \overline{\mathcal{Q}}_{k^0}} -\boldsymbol{v}^\top Z \boldsymbol{q} = \sup_{\boldsymbol{q} \in \overline{\mathcal{Q}}_{k^0}} -\sum_{i=1}^{m} q_i \boldsymbol{z}_{(i)}^\top \boldsymbol{v} = -\frac{1}{t^0 + 1} \sum_{i=1}^{t^0+1} \boldsymbol{z}_{(i)}^\top \boldsymbol{v} \leq 0.$$

On the other hand for arbitrarily small $\epsilon \in \left(0, \frac{1}{m}\right)$,

$$\mathcal{Q}_{k^0+\epsilon} = \{\boldsymbol{q} \mid \boldsymbol{q}^\top \boldsymbol{e} = 1,\ 0 \leq q_i \leq \tfrac{1}{t^0},\ \forall i \in M\}.$$

Then we have

$$\sup_{\boldsymbol{q} \in \overline{\mathcal{Q}}_{k^0+\epsilon}} -\boldsymbol{v}^\top Z \boldsymbol{q} = \sup_{\boldsymbol{q} \in \overline{\mathcal{Q}}_{k^0+\epsilon}} -\sum_{i=1}^{m} q_i \boldsymbol{z}_{(i)}^\top \boldsymbol{v} = -\frac{1}{t^0} \sum_{i=1}^{t^0} \boldsymbol{z}_{(i)}^\top \boldsymbol{v} > 0.$$

This implies the right hand side of (29) is $\frac{t^0}{m}$. This completes the proof. □

Let k^* be an optimal solution of (29). The classifier minimizing the tightest CCLF $\bar{\rho}(Z^\top \boldsymbol{v})$ is equivalent to ν-SVM with $\nu = 1 - \frac{(\lceil mk^* \rceil - 1)}{m}$.

6 Numerical Experiment

In this section we demonstrate the performance of our algorithm. We run the numerical experiments on a Red Hat Enterprise Linux Server release 6.4 (Santiago) with Intel Xeon Processor E5-2680 (2.7GHz) and 64 GB of physical memory. We implemented the practical APG method (Algorithm 1) by MATLAB R2013a and the bisection methods by C++. The C++ code was called from MATLAB via MEX files.

We conducted the experiments using artificial datasets and benchmark datasets from LIBSVM Data [Chang and Lin, 2011]. The artificial datasets were generated as follows. Positive samples $\{\boldsymbol{x}_i \in \mathbb{R}^n \mid i \in M_+\}$ and negative samples $\{\boldsymbol{x}_i \in \mathbb{R}^n \mid i \in M_-\}$ were distributed with n-dimensional standard normal distributions $\mathcal{N}_n(\boldsymbol{0}, I_n)$ and $\mathcal{N}_n(\frac{10}{\sqrt{n}}\boldsymbol{e}, SS^\top)$, respectively, where the elements of the n by n matrix S are i.i.d. random variables following the standard normal distribution $\mathcal{N}(0,1)$. The marginal probability of the label was assumed to be same, i.e. $P(y=+1) = P(y=-1) = \frac{1}{2}$. After generating samples, we scaled them so that each input vector $\boldsymbol{x}_i$ $(\forall i \in M)$ was in $[-1,1]^n$ for the purpose of computational stability, following LIBSVM [Chang and Lin, 2011]. On the other hand, we scaled the benchmark datasets, that are not scaled by Chang and Lin [2011], so that $\boldsymbol{x}_i \in [0,1]^n$, $(\forall i \in [m])$ in order to leverage their sparsity. The details of benchmark datasets are shown in Table 1.

6.1 Projection Algorithms

Before we present the performance of our practical APG, we compare the performance of our projection algorithm (Algorithm 2) based on bisection in Section 4.2 against the breakpoint search algorithm [Kiwiel, 2008, Algorithm 3.1] with random pivoting. Both algorithms are implemented by C++. We generated $\mathbb{R}^n$-valued random vectors $\tilde{\boldsymbol{q}}$ with uniformly distributed elements and computed the projections $P_{\mathcal{U}'}(\tilde{\boldsymbol{q}})$ of $\tilde{\boldsymbol{q}}$ onto $\mathcal{U}'$, where $\mathcal{U}' := \left\{\boldsymbol{q} \mid \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+,-\},\ \boldsymbol{0} \leq \boldsymbol{q} \leq \frac{1}{m\nu}\boldsymbol{e}\right\}$ with $\nu = 0.5$. The bisection algorithm used the accuracy of $\epsilon' \approx 2.22 \times 10^{-16}$ (i.e., IEEE 754 double precision). Table 2 reports the average and standard deviation of computation times over 20 trials. As we can see from Table 2, the bisection method is faster and more stable (in the sense that the variance in the computation time is smaller) than the breakpoint search algorithm. This explains why we have chosen to use the bisection methods in Section 4.2 to perform the projection steps in Algorithm 1.

6.2 APG for ν-SVM

We solved the ν-SVM (12) in the form of UCM via the APG method, SeDuMi [Sturm, 1999], and LIBSVM [Chang and Lin, 2011]. SeDuMi is a general purpose optimization solver implementing the interior point method for large-scale second-order cone problems such as (12). LIBSVM implements the sequential minimal optimization (SMO) [Platt, 1998] which is specialized for learning ν-SVM. For reference, we also compared the APG method with LIBLINEAR [Fan et al., 2008] which implements a coordinate descent

Table 1: Details of Datasets. We have scaled the datasets that are highlighted in boldface type.

data	m (	m_+,	m_-)	n	range	density
a8a	22,696 (	5,506,	17,190)	123	$[0,1]^n$	0.113
a9a	32,561 (	7,841,	24,720)	123	$[0,1]^n$	0.113
australian	690 (	307,	383)	14	$[-1,1]^n$	0.874
breast-cancer	683 (	444,	239)	10	$[-1,1]^n$	1.000
cod-rna	59,535 (	39,690,	19,845)	8	$[0,1]^n$	0.999
colon-cancer	62 (	40,	22)	2,000	$[0,1]^n$	0.984
covtype	581,012 (	297,711,	283,301)	54	$[0,1]^n$	0.221
diabetes	768 (	500,	268)	8	$[-1,1]^n$	0.999
duke	44 (	21,	23)	7,129	$[0,1]^n$	0.977
epsilon-normalized	400,000 (	199,823,	200,177)	2,000	$[-0.15,0.16]^n$	1.000
fourclass	862 (	307,	555)	2	$[-1,1]^n$	0.996
german.numer	1,000 (	300,	700)	24	$[-1,1]^n$	0.958
gisette	6,000 (	3,000,	3,000)	5,000	$[-1,1]^n$	0.991
heart	270 (	120,	150)	13	$[-1,1]^n$	0.962
ijcnn1	35,000 (	3,415,	31,585)	22	$[-0.93,1]^n$	0.591
ionosphere	351 (	225,	126)	34	$[-1,1]^n$	0.884
leu	38 (	11,	27)	7,129	$[0,1]^n$	0.974
liver-disorders	345 (	145,	200)	6	$[-1,1]^n$	0.991
madelon	2,000 (	1,000,	1,000)	500	$[0,1]^n$	0.999
mushrooms	8,124 (	3,916,	4,208)	112	$[0,1]^n$	0.188
news20.binary	19,996 (	9,999,	9,997)	1,355,191	$[0,1]^n$	3.36E-04
rcv1-origin	20,242 (	10,491,	9,751)	47,236	$[0,0.87]^n$	0.002
real-sim	72,309 (	22,238,	50,071)	20,958	$[0,1]^n$	0.002
skin-nonskin	245,057 (	50,859,	194,198)	3	$[0,1]^n$	0.983
sonar	208 (	97,	111)	60	$[-1,1]^n$	1.000
splice	1,000 (	517,	483)	60	$[-1,1]^n$	1.000
svmguide1	3,089 (	1,089,	2,000)	4	$[0,1]^n$	0.997
svmguide3	1,243 (	947,	296)	22	$[0,1]^n$	0.805
url-combined	2,396,130 (	1,603,985,	792,145)	3,231,961	$[0,1]^n$	3.54E-05
w7a	24,692 (	740,	23,952)	300	$[0,1]^n$	0.039
w8a	49,749 (	1,479,	48,270)	300	$[0,1]^n$	0.039

Table 2: Runtime of Projection Algorithms (msec.)

vector		Breakpoint		Bisection	
dim. n	range	ave.	std.	ave.	std.
100,000	$[0,10]^n$	8.3	2.0	4.9	0.6
100,000	$[0,1000]^n$	9.0	1.4	5.5	1.2
1,000,000	$[0,10]^n$	94.2	16.2	49.9	3.4
1,000,000	$[0,1000]^n$	99.1	15.6	53.4	4.4

method for C-SVM[†] [Cortes and Vapnik, 1995] and is known to be a quite efficient method; we note that it may not be a fair comparison because LIBLINEAR omits the bias term b of C-SVM from the calculations, i.e. solves a different model which is less complex than the ν-SVM (12), in order to speed up the computation. Although LIBLINEAR can virtually deal with the bias term b by augmenting the dimension of the samples, the best performance of the resulting model tends to be lower than the one of ν-SVM as reported in [Kitamura et al., 2014].

We used the error tolerance $\tilde{\epsilon} = 10^{-8}$ (default) in SeDuMi. LIBSVM, LIBLINEAR, and the APG method share the same stopping criteria: they terminate if the violation of the KKT optimality condition is less than ϵ. In this experiments, the tolerance $\epsilon = 10^{-6}$ was chosen so that the objective value of the APG method will be lower than the one of SeDuMi. The heuristic option in LIBSVM was set to "off" in order to speed up its convergence for large datasets. In the APG method, η and δ were set to 1.1 and 0.8, respectively. L was initially set to the maximum value in the diagonal elements of $\widetilde{Z}^\top \widetilde{Z}$ (i.e., the coefficient matrix of the quadratic form), where $\widetilde{Z} = [y_1 \boldsymbol{x}_1, \ldots, y_m \boldsymbol{x}_m]$. The initial point $\boldsymbol{q}^0$ was set to the *center* $\boldsymbol{q}^c$ of $\mathcal{U}'$, i.e. $q_i^c = \frac{1}{2m_o}$, $i \in M_o$, $o \in \{+, -\}$.

6.2.1 Artificial Datasets

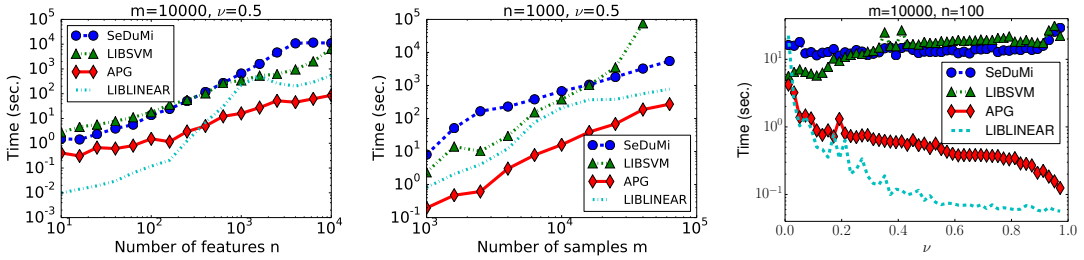

Figure 5: Computation Time for ν-SVM

Computation Time First, we measured the computation time with respect to the size of the datasets and parameter using artificial datasets. The results (for the linear kernel) are shown in Figure 5. The left panel shows the computation time with respect to the dimension n of the features for $m = 10000$ and $\nu = 0.5$. The APG method has a clear advantage when the dimension n is high, say $n \geq 10^3$. The middle panel shows the computation time with respect to the number m of the samples for $n = 1000$ and $\nu = 0.5$. LIBSVM did not converge within a week for $m = 63000$. SeDuMi, LIBLINEAR, and the APG method were scalable for the increased number of samples m. The right panel illustrates the computation time with respect to the parameter ν for $m = 10000$ and $n = 100$. We may observe that the APG method (and LIBLINEAR) is very efficient when ν is larger than 0.1, but converges more slowly when ν is small. The latter can be

[†] C-SVM (with the bias term b) is known to lead to the same decision function as ν-SVM if ν and C are set properly [Schölkopf et al., 2000]. The value of C corresponding to ν can be computed by LIBSVM.

attributed to the fact that smaller ν enlarges the feasible region $\mathcal{U}'$ and lengthens the distance between the initial point $\boldsymbol{q}^0 = \boldsymbol{q}^c$ and the optimal solution $\boldsymbol{q}^*$.

Table 3: Runtime Breakdown of the APG Method (sec.)

Function	% Time	Time	# Evals.	Time/Eval.
$\nabla f(\boldsymbol{q})$	78.1%	14.909	1375	0.0108
$f(\boldsymbol{q})$	10.8%	2.053	369	0.0056
$P_{\mathcal{U}}(\boldsymbol{q})$	6.9%	1.307	1453	0.0009
Total Runtime:		19.080		

Runtime Breakdown Table 3 shows the runtime breakdown of the APG method for $(m, n, \nu) = (10000, 1000, 0.5)$. The computation of the gradient $\nabla f(\boldsymbol{q})$ and the function $f(\boldsymbol{q})$ was the most time-consuming parts in the APG method. Since the computations of Step 0 and $\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\|$ involve the extra evaluation of $f(\boldsymbol{q})$ and/or $\nabla f(\boldsymbol{q}^k)$, computing them only every 100 and 10 iteration, respectively, as in Algorithm 1 would be effective to reduce the total runtime. Our projection algorithm was efficient enough in the sense that its runtime was marginal compared to the runtime of the other parts.

Running History Figure 6 shows the runtime history of the APG method for $(m, n, \nu) = (10000, 1000, 0.5)$. The left panel depicts the violations of the optimality, i.e. the values of

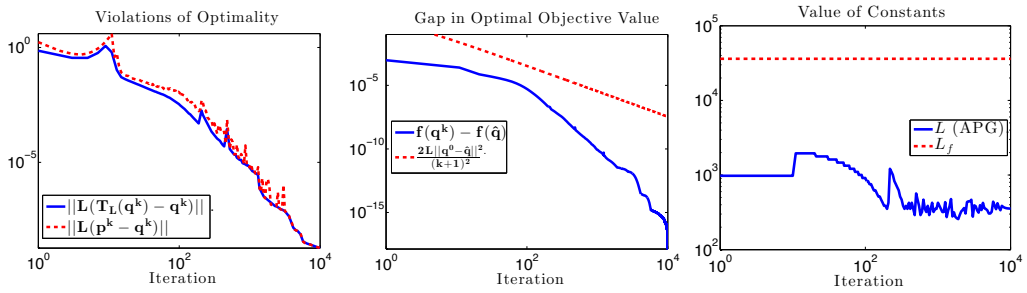

Figure 6: Running History of the APG Method for ν-SVM

$\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\|$ (necessary and sufficient optimality condition for $\boldsymbol{q}^k$) and $\|L(\boldsymbol{q}^k - \boldsymbol{p}^k)\|$ (sufficient optimality condition for $\boldsymbol{q}^k$), at each iteration. One can observe that their values do not differ much. Thus, it would not matter if we check the value of $\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\|$ in only every 100 iteration as done in Algorithm 1. The middle panel illustrated the objective value of $f(\boldsymbol{q}^k) - f(\hat{\boldsymbol{q}})$ and its theoretical upper bound $\frac{2L\|\boldsymbol{q}^0 - \hat{\boldsymbol{q}}\|^2}{(k+1)^2}$ which is derived from Theorem 1, where we regarded $\hat{\boldsymbol{q}} = \boldsymbol{q}^{10000}$ as an optimal solution. At $k = 1000$, the objective value $f(\boldsymbol{q}^k)$ reached $f(\hat{\boldsymbol{q}})$ within a relative error 0.0003% though the violation of optimality was greater than $\epsilon = 10^{-6}$. Thus, a larger tolerance, say $\epsilon = 10^{-5}$, may also lead to a reasonable solution in practice. The right panel illustrated the value of constant L of the APG at each iteration and the value of Lipschitz constant

L_f; L_f is known to be the largest eigenvalue of $\widetilde{Z}^\top \widetilde{Z}$, where $\widetilde{Z} = [y_1 \boldsymbol{x}_1, \ldots, y_m \boldsymbol{x}_m]$. While $L_f = 3.61 \times 10^4$, Step 0 of the APG method leads to a much smaller average value of 3.68×10^2 for L, while the maximum value for L is 1.95×10^3.

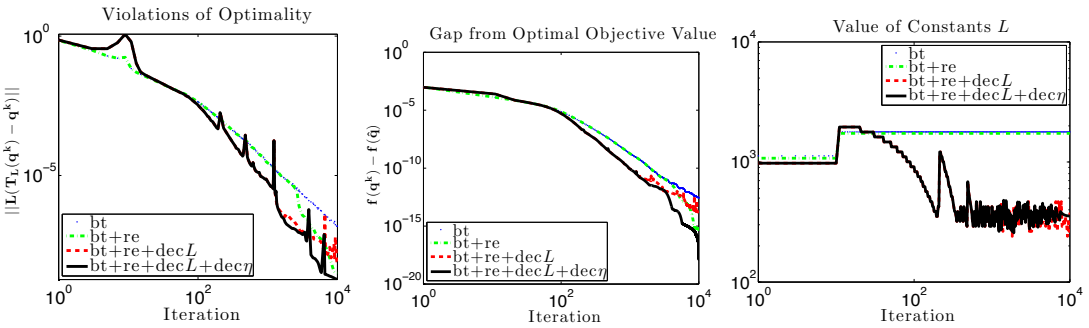

Figure 7: Effect of various acceleration strategies for the APG Method. 'bt' refers to the backtracking strategy of [Beck and Teboulle, 2009], 're' to the restarting strategy, 'dec$L$' to decreasing strategy for L, and 'decη' to decreasing strategy for η.

Effect of Each Acceleration Strategy Figure 7 illustrates the effect of various heuristic acceleration strategies described in Section 3.2 for the APG method. 'bt' refers to the backtracking strategy of [Beck and Teboulle, 2009], 're' to the restarting strategy, 'dec$L$' to decreasing strategy for L, and 'decη' to decreasing strategy for η. The APG method with 'bt+re' restarted at $k = 2594$, where the sharp decrease in the values of $\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\|$ (the left panel) and $f(\boldsymbol{q}^k) - f(\hat{\boldsymbol{q}})$ (the middle panel) occurred. 'dec$L$' was effective to reduce $L\|T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k\|$ and $f(\boldsymbol{q}^k) - f(\hat{\boldsymbol{q}})$ in the early iterations. The APG method with 'bt+re+dec$L$' seems to be unstable near the optimum, but the one with 'bt+re+dec$L$+decη' converged stably.

Computation Time for RBF Kernel The kernel method [Schölkopf and Smola, 2002] can be applied to ν-SVM by setting the objective function as $f(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{q}^\top Y K Y \boldsymbol{q}$, where K is a square kernel matrix defined as $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ using a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and Y is a diagonal matrix whose diagonal is the label vector $\boldsymbol{y}$. Figure 8

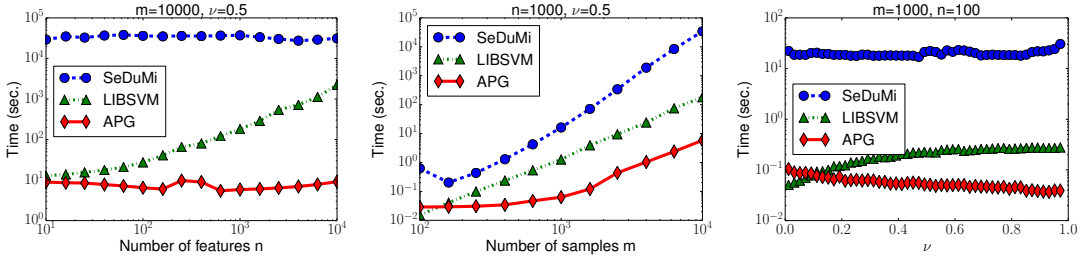

Figure 8: Computation Time for ν-SVM with RBF Kernel

shows the results for the RBF kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/2\gamma^2)$. The parameter γ was set to $\sqrt{n/2}$ which is the default parameter of LIBSVM. Since both K and Y are m by m matrices, the dimension n of features had little effect on the problem size and the computation time of SeDuMi and the APG method as shown in the left panel of Figure 8. LIBSVM computes only elements of K that are necessary for updating the solution. This might result in the increase in the computation time of LIBSVM with respect to n. On the other hand, the number m of samples affects the computation time significantly because some computational cost involving K grow quadratically with respect to the number m of samples. This detracts the scalability of SeDuMi with respect to m, as shown in the middle panel of Figure 8, because SeDuMi solves an linear equation $A\boldsymbol{x} = \boldsymbol{b}$ at each iteration where the coefficient matrix A contains K as a submatrix. On the other hand, the APG method did not lose its scalability so much; the most time-consuming part of the APG method is the computation of $\nabla f(\boldsymbol{q})$, but it requires just matrix-vector multiplications. These results show the advantage of the APG method for large datasets. The right panel of Figure 8 is the computation time with respect to ν for $m = 1000$ and $n = 100$. It shows similar trends observed in the results for linear kernel (Figure 5).

6.2.2 Benchmark Datasets

We also conducted experiments using the benchmark datasets. Practically, ν is often set to a small value because a smaller ν tends to show a better performance empirically (see Table 9). Thus, we set ν to $0.9\nu_{min} + 0.1\nu_{max}$, where the values of ν_{min} and ν_{max} for each dataset are shown in Table 4, except the large datasets 'epsilon-normalized', 'news20.binary', and 'url-combined'; for them, ν was set to $0.5\nu_{max}$ since we could not compute ν_{min} within 36000 seconds using SeDuMi.

Computation Time The experimental results are shown in Table 5. When using linear kernel, the APG method outperformed LIBSVM and SeDuMi for large datasets such that $m \geq 50000$. LIBLINEAR generally showed quite better performance than others. (Again, note that it solves a different type of SVM, i.e. C-SVM without the bias term b.) However, the APG method had an advantage over LIBLINEAR for many datasets such that $n \geq 2000$.

When using the RBF kernel, SeDuMi broke down for datasets such that $m \geq 10000$ as in the case of the artificial datasets. The APG method had run out of memory for $m \geq 50000$ since it requires the m by m dense kernel matrix K to compute the gradient $\nabla f(\boldsymbol{q})$. In order to decrease the memory usage and the computation time, a little ingenuity is required such as a coordinate-wise update of $\boldsymbol{q}$ or an approximation of K using random sample of the training set. However, our practical APG method was still competitive with LIBSVM and had stable and good performance for datasets when n is large.

We should remark that the number of iterations taken by the APG method with the RBF kernel tends to be smaller than the one with the linear kernel. However, when using the RBF kernel, the computational complexity of $\nabla f(\boldsymbol{q})$ changes from $O(mn)$ to

Table 4: Details of Each Dataset with Respect to ν-SVM.

data	ν_{min}	ν_{max}
a8a	0.355	0.485
a9a	0.352	0.482
australian	0.288	0.890
breast-cancer	0.064	0.700
cod-rna	0.174	0.667
colon-cancer	0.008	0.710
covtype	0.580	0.975
diabetes	0.515	0.698
duke	0.012	0.955
epsilon-normalized	–	0.999
fourclass	0.524	0.712
german.numer	0.517	0.600
gisette	0	1.000
heart	0.333	0.889
ijcnn1	0.185	0.195
ionosphere	0.145	0.718
leu	0.013	0.579
liver-disorders	0.718	0.841
madelon	0.559	1.000
mushrooms	0	0.964
news20.binary	–	1.000
rcv1-origin	0.001	0.963
real-sim	0.004	0.615
skin-nonskin	0.213	0.415
sonar	0.026	0.933
splice	0.373	0.966
svmguide1	0.122	0.705
svmguide3	0.401	0.476
url-combined	–	0.661
w7a	0.028	0.060
w8a	0.028	0.059

Table 5: Computation Time for Benchmark Datasets (sec.). '–' means that the algorithm did not converge with in 36000 seconds. '**' means that it had run out of memory. The best results are indicated by boldface. The underlined results are better than LIBLINEAR.

				Linear				RBF		
data	m	n	ν	SeDuMi	LIBSVM	APG (iter)	LIBLINEAR	SeDuMi	LIBSVM	APG(iter)
a8a	22,696	123	0.368	16.82	32.31	**1.70** (563)	0.059	–	**68.24**	72.88(343)
a9a	32,561	123	0.365	25.42	66.11	**4.84** (665)	0.071	–	**138.89**	170.63(390)
australian	690	14	0.348	0.34	**0.12**	0.86 (4056)	0.003	8.03	**0.049**	0.076(243)
breast-cancer	683	10	0.128	0.21	**0.004**	0.05 (253)	0.001	6.72	**0.015**	0.050(144)
cod-rna	59,535	8	0.223	6.71	53.73	**3.39** (588)	0.170	**	267.56	**(**)
colon-cancer	62	2,000	0.078	0.18	**0.01**	0.09 (210)	0.153	0.17	**0.013**	**0.013**(109)
covtype	581,012	54	0.620	288.71	16164.88	**60.75** (701)	1.547	**	–	**(**)
diabetes	768	8	0.533	0.14	**0.02**	0.06 (306)	0.003	8.53	**0.060**	0.093(296)
duke	44	7,129	0.106	0.38	**0.04**	0.11 (317)	0.302	0.09	0.041	**0.017**(120)
epsilon-normalized	400,000	2,000	0.500	–	–	**1685.33** (2643)	–	**	–	**(**)
fourclass	862	2	0.543	0.12	**0.01**	0.07 (356)	0.000	12.34	0.069	**0.068**(150)
german.numer	1,000	24	0.525	**0.26**	**0.26**	0.29 (1107)	0.035	17.13	0.14	**0.10**(236)
gisette	6,000	5,000	0.100	4572.24	57.48	**9.31** (590)	0.411	5586.74	59.07	**4.93**(235)
heart	270	13	0.388	0.14	**0.005**	0.04 (232)	0.001	0.47	**0.008**	0.040(196)
ijcnn1	35,000	22	0.186	7.31	53.57	**5.67** (2000)	0.286	–	**73.83**	236.15(511)
ionosphere	351	34	0.202	0.25	**0.02**	0.22 (1064)	0.339	0.85	**0.012**	0.051(234)
leu	38	7,129	0.070	0.40	**0.03**	0.13 (175)	0.168	0.08	0.034	**0.013**(103)
liver-disorders	345	6	0.731	0.11	**0.01**	0.11 (736)	0.007	0.81	**0.015**	0.037(168)
madelon	2,000	500	0.603	29.15	11.57	**0.32** (510)	87.431	133.85	3.77	**0.24**(108)
mushrooms	8,124	112	0.096	8.74	1.34	**0.56** (435)	0.060	28505.46	**5.74**	19.02(661)
news20.binary	19,996	1,355,191	0.100	–	1333.26	**22.29** (321)	1.572	–	929.85	**51.86**(11)
rcv1-origin	20,242	47,236	0.097	–	587.10	**7.82** (485)	8.963	–	192.16	**97.68**(189)
real-sim	72,309	20,958	0.065	–	6351.62	**12.71** (384)	4.333	**	**1879.90**	**(**)
skin-nonskin	245,057	3	0.233	62.70	726.20	**8.02** (609)	0.092	**	**4425.59**	**(**)
sonar	208	60	0.117	0.29	**0.12**	0.33 (1922)	3.809	0.27	**0.009**	0.060(279)
splice	1,000	60	0.432	0.56	0.25	**0.11** (331)	0.022	15.27	0.18	**0.067**(120)
svmguide1	3,089	4	0.180	0.35	**0.08**	0.15 (394)	0.003	1188.65	**0.42**	1.45(323)
svmguide3	1,243	22	0.408	**0.36**	1.07	0.93 (3248)	1.695	30.17	0.23	**0.21**(430)
url-combined	2,396,130	3,231,961	0.500	–	–	**4853.91** (2521)	–	**	**	**(**)
w7a	24,692	300	0.031	69.90	135.91	**14.13** (4280)	0.806	–	**17.89**	286.27(1208)
w8a	49,749	300	0.031	189.40	143.42	**38.69** (5960)	0.465	–	**124.55**	1423.42(2100)

Table 6: Constants for Benchmark Datasets

	Linear			RBF		
	L(APG)			L(APG)		
	ave.	max.	L_f	ave.	max.	L_f
a8a	2.99.E+03	1.54.E+04	1.43.E+05	7.00.E+01	2.27.E+02	2.00.E+04
a9a	5.59.E+03	3.38.E+04	2.05.E+05	1.37.E+02	4.99.E+02	2.88.E+04
australian	1.97.E+02	1.21.E+03	2.91.E+03	1.92.E+01	4.69.E+01	3.56.E+02
breast-cancer	9.05.E+01	2.27.E+02	1.68.E+04	1.21.E+01	2.13.E+01	4.52.E+02
cod-rna	1.05.E+03	5.65.E+03	1.24.E+05	–	–	–
colon-cancer	4.02.E+02	6.60.E+02	3.80.E+04	5.91.E-01	9.09.E-01	5.65.E+01
covtype	3.71.E+04	2.37.E+05	4.58.E+06	–	–	–
diabetes	5.21.E+01	1.53.E+02	1.76.E+03	1.64.E+01	4.69.E+01	6.34.E+02
duke	2.82.E+03	4.55.E+03	6.00.E+04	6.89.E-01	9.09.E-01	4.09.E+01
epsilon-normalized	2.09.E+04	1.85.E+05	1.40.E+05	–	–	–
fourclass	5.61.E+01	1.87.E+02	2.80.E+02	5.72.E+01	1.03.E+02	5.36.E+02
german.numer	2.16.E+02	1.03.E+03	8.44.E+03	1.95.E+01	4.69.E+01	4.50.E+02
gisette	1.84.E+04	5.71.E+04	2.02.E+07	9.17.E+00	2.13.E+01	3.61.E+03
heart	1.07.E+02	2.53.E+02	7.49.E+02	1.01.E+01	2.13.E+01	1.19.E+02
ijcnn1	1.20.E+03	1.05.E+04	5.89.E+03	1.93.E+02	8.84.E+02	3.12.E+04
ionosphere	2.46.E+02	9.83.E+02	2.14.E+03	1.19.E+01	2.83.E+01	2.24.E+02
leu	1.70.E+03	2.75.E+03	6.05.E+04	6.07.E-01	9.09.E-01	3.46.E+01
liver-disorders	1.27.E+01	3.85.E+01	1.84.E+03	5.04.E+00	9.68.E+00	8.23.E+02
madelon	9.54.E+01	2.87.E+02	2.44.E+05	6.54.E-01	1.00.E+00	1.92.E+03
mushrooms	1.84.E+03	6.34.E+03	2.30.E+05	2.31.E+01	1.03.E+02	1.74.E+04
news20.binary	4.63.E+01	1.03.E+02	1.17.E+03	9.09.E-01	9.09.E-01	2.00.E+04
rcv1-origin	3.20.E+01	8.30.E+01	4.49.E+02	4.42.E-01	9.09.E-01	2.02.E+04
real-sim	6.75.E+01	2.27.E+02	9.21.E+02	–	–	–
skin-nonskin	5.08.E+03	2.82.E+04	2.19.E+05	–	–	–
sonar	2.27.E+02	1.12.E+03	2.68.E+03	4.88.E+00	9.68.E+00	1.51.E+02
splice	3.59.E+02	1.11.E+03	1.74.E+03	6.63.E+00	1.06.E+01	3.57.E+02
svmguide1	3.07.E+01	1.13.E+02	2.47.E+03	1.52.E+01	4.69.E+01	2.92.E+03
svmguide3	1.53.E+02	6.68.E+02	4.92.E+03	1.22.E+01	4.69.E+01	1.15.E+03
url-combined	9.94.E+05	1.10.E+07	1.57.E+08	–	–	–
w7a	5.53.E+03	4.62.E+04	6.52.E+04	4.80.E+01	1.83.E+02	2.31.E+04
w8a	1.04.E+04	1.06.E+05	1.32.E+05	1.14.E+02	4.51.E+02	4.65.E+04

$O(m^2)$. Hence the total runtime tended to increase except for "gisette" whose n and m have the same order of magnitude.

In summary, the APG method showed better performance than specialized algorithms designed for learning SVM, such as LIBSVM and LIBLINEAR, in many datasets. Taking into account the generality (i.e., applicability to other models) of APG, one could argue that it is a very efficient method.

Value of Constant Table 6 shows the values taken by the parameter L. We can see that our practical APG method (with backtracking strategy and decreasing strategy for L) keep the values of L to be much smaller than the Lipschitz constant L_f.

6.3 APG for MM-MPM

Next, we conducted experiments on MM-MPM (10) in the form of UCM. To the best of our knowledge, there are no specialized method for MM-MPM. Thus, we compared the APG method only to SeDuMi [Sturm, 1999] which implements the interior point method for the large-scale second-order cone problems such as (10). We used the error tolerance $\epsilon = 10^{-8}$ (default) in SeDuMi and the setting $\epsilon = 10^{-6}$ in the APG method so that their objective values of the solutions are within a relative error of 0.001%. In the APG method, η and δ were set to 1.1 and 0.8. The value of L was initially set to the maximum value in the diagonal elements of $\widetilde{\Sigma}^\top \widetilde{\Sigma}$ (i.e., the coefficient matrix of the quadratic form), where $\widetilde{\Sigma} = [\Sigma_+^{1/2}, -\Sigma_-^{1/2}]$. The initial point $\boldsymbol{q}^0 = (\boldsymbol{u}_+^0, \boldsymbol{u}_-^0)$ was set to the origin $\mathbf{0}$ (i.e., the center of $\mathcal{U}'$).

6.3.1 Artificial Datasets

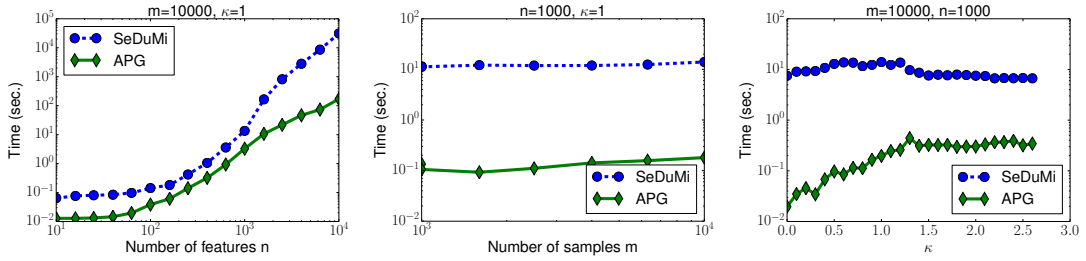

Figure 9: Computation Time for MM-MPM

Computation Time The computation time for the artificial datasets are shown in Figure 9. The left panel shows the results with respect to the number n of features for $m = 10000$ and $\kappa = 1$. The APG method has a clear advantage over SeDuMi for large n, say $n \geq 10^3$. The middle panel illustrates the computation time with respect to the number m of samples for $n = 2000$ and $\kappa = 1$. The computation time is nearly independent of the number m of samples because the sizes of matrices $\Sigma_o^{1/2}$ ($o \in \{+, -\}$), which are used for computing the function $f(\boldsymbol{q})$ and the gradient $\nabla f(\boldsymbol{q})$, are n by n. The right panel shows the computation time with respect to the parameter κ for $m = 10000$ and $n = 2000$. We can observe that a larger value of κ leads to more computation time for the APG method although it is still far more efficient than SeDuMi. The effect of a larger κ on the APG method could be because it gives a larger feasible region $\mathcal{U}'$, which in turns lead to a larger distance between the initial point $\boldsymbol{q}^0$ (the center $\mathbf{0}$ of $\mathcal{U}'$) and the optimal solution $\boldsymbol{q}^*$ as in the case of ν-SVM (the right panel of Figure 5).

Runtime Breakdown Table 7 shows the runtime breakdown of the APG method for $(m, n, \kappa) = (10000, 1000, 1)$. As in the case of ν-SVM (Table 3), the computations of the gradient $\nabla f(\boldsymbol{q})$ and the function value $f(\boldsymbol{q})$ are the most time-consuming parts. Thus, computing Step 0 and $L\|T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k\|$ periodically, which involve the computations of

Table 7: Runtime Breakdown of APG Method for MM-MPM (sec.)

Function	% Time	Time	# Evals.	Time/Eval.
$\nabla f(\boldsymbol{q})$	75.9%	0.836	339	2.47.E-03
$f(\boldsymbol{q})$	14.2%	0.157	119	1.32.E-03
$P_{\mathcal{U}'}(\boldsymbol{q})$	3.1%	0.034	383	8.88.E-05
Total Runtime:		1.102		

$f(\boldsymbol{q}^k)$ and/or $\nabla f(\boldsymbol{q}^k)$, is effective to reduce total runtime. The projection $P_{\mathcal{U}'}(\boldsymbol{q})$ for MM-MPM shown in Section 4.1 can be computed highly efficiently.

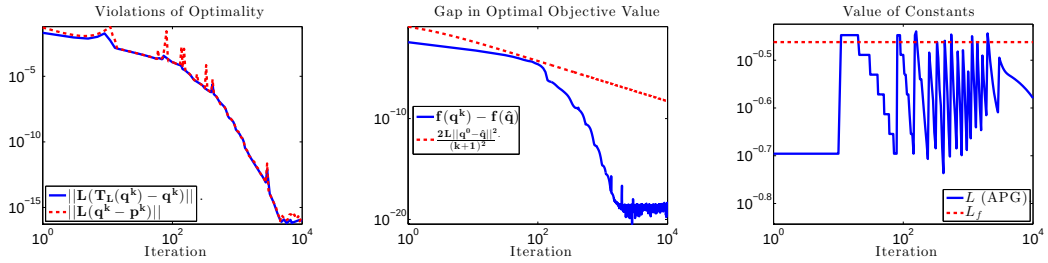

Figure 10: Running History of APG Method for MM-MPM with $(m, n, \kappa) = (10000, 1000, 1)$

Running History Figure 10 shows the running history of the APG method for $(m, n, \kappa) = (10000, 1000, 1)$. In the left panel, $\|L(T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k)\|$ (violation of necessary and sufficient optimality condition for $\boldsymbol{q}^k$) and $L\|\boldsymbol{q}^k - \boldsymbol{p}^k\|$ (violation of sufficient optimality condition for $\boldsymbol{q}^k$) take almost the same values. Thus, it would not matter if we check the value of $L\|T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k\|$ in only every 100 iteration as is done in Algorithm 1. The middle panel shows the values $f(\boldsymbol{q}^k) - f(\hat{\boldsymbol{q}})$ and its theoretical upper bound, which is derived from Theorem 1, where we regarded $\hat{\boldsymbol{q}} = \boldsymbol{q}^{10000}$ is an optimal solution. Though the strategies described in Section 3.2 are heuristic, the value of $f(\boldsymbol{q}^k) - f(\hat{\boldsymbol{q}})$ did not exceed the theoretical upper bound and decreased much faster than the bound. In the right panel, we can see that the APG method uses values smaller than L_f for L in most iterations, where the Lipschitz constant L_f of the gradient $\nabla f(\boldsymbol{q}) = \widetilde{\Sigma}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_- + \widetilde{\Sigma}\boldsymbol{q})$ is known to be the largest eigenvalue of the matrix $\widetilde{\Sigma}^\top\widetilde{\Sigma}$ (recall that $\widetilde{\Sigma} = [\Sigma_+^{1/2}, -\Sigma_-^{1/2}]$ and $\boldsymbol{q} = (\boldsymbol{u}_+, \boldsymbol{u}_-)$).

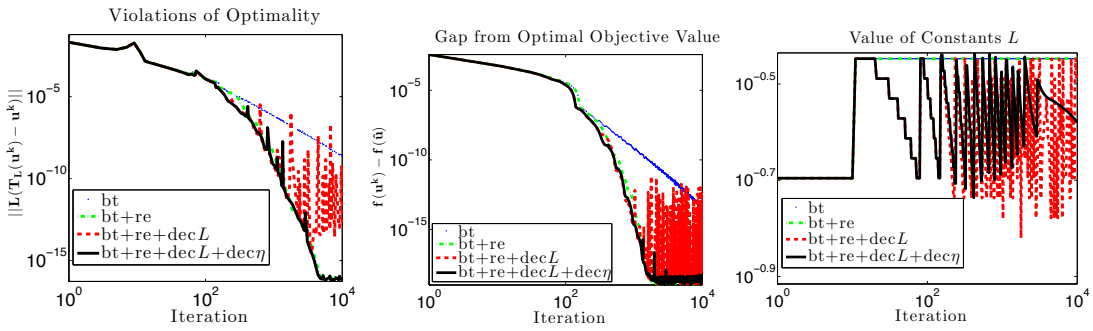

Figure 11: Effect of Each Strategy for the APG Method. 'bt' refers to the backtracking strategy of [Beck and Teboulle, 2009], 're' to the restarting strategy, 'dec$L$' to decreasing strategy for L, and 'decη' to decreasing strategy for η.

Effect of Each Strategy Figure 11 illustrates the effect of each strategy in Section 3.2. 'bt' refers to the backtracking strategy of [Beck and Teboulle, 2009], 're' to the restarting strategy, 'dec$L$' to decreasing strategy for L, and 'decη' to decreasing strategy for η. As in the case of ν-SVM, 're' is effective in reducing the violation of optimality $L\|T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k\|$ and the value of $f(\boldsymbol{q}^k) - f(\hat{\boldsymbol{q}})$. 'dec$L$' seems to make the APG method to be unstable, but 'decη' can stabilize it. 'bt+re+dec$L$+decη' decreased $L\|T_L(\boldsymbol{q}^k) - \boldsymbol{q}^k\|$ and $f(\boldsymbol{q}^k) - f(\hat{\boldsymbol{q}})$ slightly faster than 'bt+re'.

Kernel Method The kernel method can be applied to MM-MPM as shown in [Nath and Bhattacharyya, 2007]. When using a nonlinear kernel, however, the covariance matrix Σ_o grows to an m by m matrix. SeDuMi would break down for such a large matrix (see the column of 'RBF' in Table 5 for the results dealing with m by m matrix K). Hence, we omit experiments for a nonlinear kernel.

6.3.2 Benchmark Datasets

Table 8 shows the computational results for the benchmark datasets. We did the experiments by setting $\kappa = \kappa_{max}/2$, but MM-MPM could not be solved for $n \geq 20000$ because the sizes of the n by n matrice $\Sigma_o^{1/2}$ $(o \in \{+, -\})$ are extremely large.

The APG method was much faster than SeDuMi especially when the dimension is high, say $n \geq 2000$. Unlike for ν-SVM (Table 6), the APG method for MM-MPM sometimes led larger values of L than the Lipschitz constant L_f. However, the average of the values of L is still smaller than L_f.

6.4 Classification Ability

Using the benchmark datasets, we compared the classification ability of classification models: ν-SVM, MM-MPM, and MM-FDA. Each dataset was randomly partitioned into 10 disjoint sets. We investigated the averages of the test accuracy using cross-validation

Table 8: Computational Results for MM-MPM with Linear Kernel. The best results are indicated by boldface. '**' means that the algorithm could not be computed due to out of memory.

					Computation Time		Values		
							L(APG)		
data	m	n	κ_{max}	κ	SeDuMi	APG (iter)	ave.	max.	L_f
a8a	22,696	123	9.52.E-01	4.76.E-01	0.664	**0.034** (25)	1.12.E+00	1.21.E+00	1.34.E+00
a9a	32,561	123	9.60.E-01	4.80.E-01	0.166	**0.010** (25)	1.12.E+00	1.21.E+00	1.34.E+00
australian	690	14	1.23.E+00	6.17.E-01	0.054	**0.006** (25)	1.85.E+00	2.00.E+00	2.06.E+00
breast-cancer	683	10	2.32.E+00	1.16.E+00	0.038	**0.005** (30)	1.01.E+00	1.10.E+00	1.17.E+00
cod-rna	59,535	8	1.41.E+00	7.05.E-01	0.068	**0.015** (115)	1.69.E-01	2.29.E-01	2.34.E-01
colon-cancer	62	2,000	1.85.E+00	9.24.E-01	311.329	**0.178** (157)	2.56.E+01	4.22.E+01	2.81.E+01
covtype	581,012	54	6.59.E-01	3.29.E-01	0.086	**0.016** (107)	8.31.E-01	1.10.E+00	1.07.E+00
diabetes	768	8	6.88.E-01	3.44.E-01	0.061	**0.003** (18)	3.70.E-01	3.86.E-01	4.79.E-01
duke	44	7,129	1.98.E+00	9.89.E-01	13459.0	**0.660** (291)	1.55.E+02	2.15.E+02	1.97.E+02
epsilon-normalized	400,000	2,000	1.16.E+00	5.82.E-01	361.0	**1.186** (217)	3.64.E-01	6.07.E-01	4.77.E-01
fourclass	862	2	7.22.E-01	3.61.E-01	0.069	**0.003** (12)	4.68.E-01	5.47.E-01	6.04.E-01
german.numer	1,000	24	6.52.E-01	3.26.E-01	0.063	**0.005** (33)	2.45.E+00	2.84.E+00	2.89.E+00
gisette	6,000	5,000	8.53.E+00	4.27.E+00	5438.2	**56.292** (1640)	9.40.E+01	2.29.E+02	1.14.E+02
heart	270	13	1.10.E+00	5.48.E-01	0.049	**0.004** (24)	1.85.E+00	1.98.E+00	2.08.E+00
ijcnn1	35,000	22	9.00.E-01	4.50.E-01	0.068	**0.013** (114)	4.06.E-01	6.52.E-01	3.04.E-01
ionosphere	351	34	1.30.E+00	6.48.E-01	0.084	**0.016** (110)	3.49.E+00	4.49.E+00	5.15.E+00
leu	38	7,129	2.11.E+00	1.05.E+00	12971.377	**0.514** (220)	1.37.E+02	2.68.E+02	1.46.E+02
liver-disorders	345	6	4.09.E-01	2.04.E-01	0.075	**0.009** (71)	3.05.E-01	4.08.E-01	3.78.E-01
madelon	2,000	500	6.50.E-01	3.25.E-01	1.964	**0.017** (43)	2.81.E-01	3.28.E-01	3.31.E-01
mushrooms	8,124	112	1.53.E+01	7.66.E+00	0.170	**0.166** (840)	2.38.E+00	3.63.E+00	2.92.E+00
news20.binary	19,996	1,355,191	**	**	**	** (**)	**	**	**
rcv1-origin	20,242	47,236	**	**	**	** (**)	**	**	**
real-sim	72,309	20,958	**	**	**	** (**)	**	**	**
skin-nonskin	245,057	3	1.63.E+00	8.14.E-01	0.066	**0.006** (50)	1.65.E-01	1.98.E-01	2.25.E-01
sonar	208	60	1.29.E+00	6.44.E-01	0.096	**0.020** (116)	3.47.E+00	4.60.E+00	5.07.E+00
splice	1,000	60	1.02.E+00	5.09.E-01	0.071	**0.009** (39)	2.39.E+00	2.84.E+00	2.92.E+00
svmguide1	3,089	4	1.26.E+00	6.29.E-01	0.070	**0.003** (28)	1.06.E-01	1.15.E-01	8.61.E-02
svmguide3	1,243	21	6.24.E-01	3.12.E-01	0.083	**0.012** (105)	5.90.E-01	8.91.E-01	5.82.E-01
url-combined	2,396,130	3,231,961	**	**	**	** (**)	**	**	**
w7a	24,692	300	1.41.E+00	7.03.E-01	0.588	**0.036** (112)	1.42.E+00	2.21.E+00	1.82.E+00
w8a	49,749	300	1.39.E+00	6.97.E-01	0.592	**0.031** (108)	1.45.E+00	2.21.E+00	1.88.E+00

Table 9: Average Performance of Each Classification Model. The best results are indicated by boldface. '–' means that the cross-validation could not be done within 36000 sec. '**' means that it had run out of memory.

dataset	ν-SVM (ν)	MM-MPM (κ)	MM-FDA (κ)
a8a	**84.4%** (0.365)	80.7% (0.942)	**84.4%** (1.320)
a9a	**84.7%** (0.362)	80.7% (0.950)	**84.7%** (1.331)
australian	85.7% (0.830)	86.1% (0.245)	**87.5%** (1.390)
breast-cancer	97.1% (0.074)	**97.5%** (2.081)	97.4% (0.313)
cod-rna	**93.9%** (0.280)	93.5% (0.420)	93.7% (0.399)
colon-cancer	**88.8%** (0.294)	87.1% (0.184)	87.1% (1.218)
covtype	**76.3%** (0.592)	75.6% (0.649)	75.5% (0.918)
diabetes	**77.3%** (0.542)	74.9% (0.610)	76.8% (0.875)
duke	**88.5%** (0.022)	**88.5%** (0.984)	**88.5%** (1.364)
epsilon-normalized	–	89.6% (1.038)	**89.7%** (1.481)
fourclass	77.7% (0.676)	72.7% (0.569)	**78.6%** (0.102)
german.numer	76.7% (0.541)	71.9% (0.385)	**77.3%** (0.552)
gisette	97.4% (0.109)	**97.9%** (3.409)	**97.9%** (4.825)
heart	**84.1%** (0.397)	**84.1%** (0.760)	**84.1%** (0.154)
ijcnn1	74.7% (0.194)	85.9% (0.891)	**91.0%** (0.636)
ionosphere	**88.3%** (0.211)	86.9% (0.386)	87.8% (0.846)
leu	**94.2%** (0.023)	**94.2%** (2.096)	**94.2%** (0.209)
liver-disorders	**68.1%** (0.762)	63.8% (0.199)	66.4% (0.516)
madelon	59.3% (0.913)	59.7% (0.128)	**60.0%** (0.09)
mushrooms	**100.0%** (0.010)	**100.0%** (9.192)	**100.0%** (6.208)
news20.binary	**66.4%** (0.999)	**	**
rcv1-origin	**97.0%** (0.106)	**	**
real-sim	**97.5%** (0.134)	**	**
skin-nonskin	93.7% (0.319)	93.5% (1.612)	**93.8%** (2.095)
sonar	**79.8%** (0.395)	**79.8%** (0.639)	77.9% (0.909)
splice	80.9% (0.499)	80.6% (0.605)	**81.0%** (0.195)
svmguide1	**95.4%** (0.132)	94.4% (1.124)	91.6% (1.561)
svmguide3	**82.5%** (0.411)	74.3% (0.552)	81.9% (0.692)
url-combined	–	**	**
w7a	**98.5%** (0.038)	96.0% (1.256)	98.2% (1.756)
w8a	**98.6%** (0.038)	96.1% (1.246)	98.3% (1.725)

over the 10 disjoint sets. We found the best parameter of the each classification model using grid search with cross-validation. The results are reported in Table 9. Though ν-SVM tends to show the high prediction performance, the model that shows the best performance varies with datasets. These results show the importance of finding a suitable classification model to a dataset in order to achieve a high prediction performance. Our algorithm is useful for the purpose; it will speed up the process of finding a suitable model.

7 Conclusion

In this work, we provided a general algorithm for the unified classification model (UCM) proposed by Takeda et al. [2013]. We applied the accelerated proximal gradient (APG)

method [Beck and Teboulle, 2009] to UCM by devising efficient projection computations and effective heuristic acceleration strategies. Our unified algorithm makes it easy to compare various models, because we can use the same algorithmic framework for the models by only changing the computation of projections. Thus, it might be practical and useful for practitioners who are looking for the best model for a given dataset.

We also showed that UCM is a versatile model which encompasses the coherent risk minimization model [Gotoh et al., 2014, Bertsimas and Takeda, 2014], generalized ν-SVM [Kanamori et al., 2013], and the coherent classification loss function (CCLF) minimization model [Yang et al., 2014], as special cases. This illustrates the generality of the unified model and our algorithm.

Numerical experiments demonstrate the efficiency of our algorithm for large datasets. As a future work, we would like to investigate the efficiency of the APG method when applied to the UCM having complicated uncertainty set $\mathcal{U}$. We also would like to find a better way of choosing an initial point for the APG method to further improve its efficiency.

References

P. Artzner, F. Delbaen, J. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, July 1999. doi: 10.1111/1467-9965.00068.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

K. P. Bennett and E. J. Bredensteiner. Duality and geometry in SVM classifiers. In *ICML*, pages 57–64, 2000.

D. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Optimization and Computation Series. Athena Scientific, 2003.

D. Bertsimas and D. B. Brown. Constructing uncertainty sets for robust linear optimization. *Operations Research*, 57(6):1483–1495, Dec. 2009. doi: 10.1287/opre.1080.0646.

D. Bertsimas and A. Takeda. Optimizing over coherent risk measures and non-convexities : A robust mixed integer optimization approach. Technical Report July, Department of Mathematical Engineering, The University of Tokyo, Japan, 2014. URL `http://www.keisu.t.u-tokyo.ac.jp/research/techrep/data/2014/METR14-18.pdf`.

C. Bhattacharyya. Second order cone programming formulations for feature selection. *The Journal of Machine Learning Research*, 5:1417–1433, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010. doi: 10.1561/2200000016.

C. Chang and C. Lin. Libsvm : A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi: 10.1007/BF00994018.

D. J. Crisp and C. J. C. Burges. A geometric interpretation of ν-SVM classifiers. In *NIPS 12*, pages 244–250. MIT Press, 2000.

R. Fan, K. Chang, H. Cho-Jui, X. Wang, and C. Lin. Liblinear : A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. Software available at `http://www.csie.ntu.edu.tw/ cjlin/liblinear`.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. doi: 10.1002/nav.3800030109.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40, 1976. doi: 10.1016/0898-1221(76)90003-1.

T. Goldstein, B. O. Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. Technical report, Group in Computational and Applied Mathematics, Department of Mathematics, University of California, Los Angeles, California, 2012.

J. Gotoh, A. Takeda, and R. Yamamoto. Interaction between financial risk measures and machine learning methods. *Computational Management Science*, 11:365–402, 2014.

K. Helgason, J. Kennington, and H. Lall. A polynomially bounded algorithm for a singly constrained quadratic program. *Mathematical Programming*, 18:338–343, 1980.

S. Iwata, Y. Nakatsukasa, and A. Takeda. Global optimization methods for extended fisher discriminant analysis. In *The 7th international conference on Artificial Intelligence and Statistics (AISTATS 2014)*, volume 33, pages 411—-419, Reykjavik, Iceland, 2014.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 427–435. JMLR Workshop and Conference Proceedings, 2013.

K. Jiang, D. Sun, and K. Toh. An inexact accelerated proximal gradient method for large scale linearly constrained convex sdp. *SIAM Journal on Optimization*, 22:1042–1064, 2012.

T. Kanamori, A. Takeda, and T. Suzuki. Conjugate relation between loss functions and uncertainty sets in classification problems. *The Journal of Machine Learning Research*, 14:1461–1504, 2013.

M. Kitamura, A. Takeda, and S. Iwata. Exact svm training by wolfe's minimum norm point algorithm. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6, Sept 2014.

K. Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming*, 112:473–491, 2008.

K. Natarajan, D. Pachamanova, and M. Sim. Constructing risk measures from uncertainty sets. *Operations Research*, 57(5):1129–1141, Oct. 2009. doi: 10.1287/opre.1080.0683.

J. S. Nath and C. Bhattacharyya. Maximum margin classifiers with specified false positive and false negative error rates. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 35–46. SIAM, 2007. doi: 10.1137/1.9781611972771.4.

A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, 1983.

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

B. O'Donoghue and E. Candés. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, pages 1–18, 2013.

P. Pardalos and N. Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46(1-3): 321–328, 1990. doi: 10.1007/BF01585748.

F. Perez-Cruz, J. Weston, D. J. L. Hermann, and B. Schölkopf. Extension of the ν-SVM range for classification. In *Advances in Learning Theory: Methods, Models and Applications*, pages 179–196. IOS Press, 2003.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, January 1998.

R. T. Rockafellar. Monotone operators and augmented lagrangian methods in nonlinear programming. *Nonlinear programming 3*, 1978.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, May 2000.

J. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999. Version 1.05 available from `http://fewcal.kub.nl/sturm`.

A. Takeda and M. Sugiyama. Nu-support vector machine as conditional value-at-risk minimization. In *Proceedings of International Conference on Machine Learning*, pages 1056–1063, 2008.

A. Takeda, H. Mitsugi, and T. Kanamori. A unified classification model based on robust optimization. *Neural computation*, 25(3):759–804, Mar. 2013. doi: 10.1162/NECO_a_00412.

W. Yang, M. Sim, and H. Xu. The coherent loss function for classification. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, volume 32, pages 37–45. JMLR Workshop and Conference Proceedings, 2014.

T. Zhou, D. Tao, and X. Wu. Nesvm: A fast gradient method for support vector machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 679–688, Dec 2010. doi: 10.1109/ICDM.2010.135.

Appendix

Here we show a few examples of $\mathcal{U}$ (and $\mathcal{U}'$) for UCM and relate the resulting models to UCM. The following lemma is helpful to show equivalences of UCM and existing models.

Lemma 5. *Let $\boldsymbol{q}^*$ be an optimal solution of the following problem:*

$$\min_{\boldsymbol{q}} \{ f(\boldsymbol{q}) + g(\boldsymbol{q}) \mid \boldsymbol{q} \in \mathcal{U}' \}, \tag{30}$$

where $\mathcal{U}' \subset \mathbb{R}^d$ is a compact set and $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ are proper closed functions. Then, the following problem:

$$\min_{\boldsymbol{q}} \{ f(\boldsymbol{q}) \mid \boldsymbol{q} \in \mathcal{U}', \quad g(\boldsymbol{q}) \leq g(\boldsymbol{q}^*) \} \tag{31}$$

has $\boldsymbol{q}^*$ as an optimal solution.

Proof. Fix $\bar{\boldsymbol{q}} \in \mathcal{U}'$. If $f(\bar{\boldsymbol{q}}) < f(\boldsymbol{q}^*)$, then we have $g(\bar{\boldsymbol{q}}) > g(\boldsymbol{q}^*)$ since $f(\bar{\boldsymbol{q}}) + g(\bar{\boldsymbol{q}}) \geq f(\boldsymbol{q}^*) + g(\boldsymbol{q}^*)$. In other word, the value of $f(\boldsymbol{q}^*)$ cannot be decreased over $\mathcal{U}'$ anymore without increasing the value of $g(\boldsymbol{q}^*)$. Hence, the problem (31) has an optimal solution $\boldsymbol{q}^*$. □

Example 1 (Ellipsoidal uncertainty sets and related classifiers)**.** *Here we introduce two examples of ellipsoidal uncertainty sets from [Takeda et al., 2013]. Let $\bar{\boldsymbol{x}}_o$ and Σ_o, $o \in \{+, -\}$, be the mean vectors and the positive definite covariance matrices, respectively, of $\boldsymbol{x}_i$, $i \in M_o$. First, let*

$$\mathcal{U} := \mathcal{U}_+ \ominus \mathcal{U}_-, \quad \textit{where} \quad \mathcal{U}_o := \{ \bar{\boldsymbol{x}}_o + \Sigma_o^{1/2} \boldsymbol{u}_o \mid \|\boldsymbol{u}_o\| \leq \kappa \}, \quad o \in \{+, -\} \tag{32}$$

with a parameter $\kappa \in [0, \kappa_{max}) \subseteq [0, \infty)$*, where* κ_{max} *is the supremum of* κ *such that* $\mathbf{0} \notin \mathcal{U}$*. Then UCM of the form* (4) *can be formulated as follows:*

$$\min_{\boldsymbol{q}=(\boldsymbol{u}_+,\boldsymbol{u}_-)\in\mathcal{U}'} \frac{1}{2}\big\|\big(\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2}\boldsymbol{u}_+\big) - \big(\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2}\boldsymbol{u}_-\big)\big\|^2 \tag{10}$$

where $\mathcal{U}' := \big\{(\boldsymbol{u}_+, \boldsymbol{u}_-) \mid \|\boldsymbol{u}_o\| \le \kappa,\ o \in \{+,-\}\big\}$ $(\subseteq \mathbb{R}^{2n})$*. Meanwhile, UCM* (2) *with the uncertainty set* (32) *is equivalent to*

$$\min_{\|\boldsymbol{w}\|\le 1} \Big(-\boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + \kappa \sum_{o\in\{+,-\}} \sqrt{\boldsymbol{w}^\top \Sigma_o \boldsymbol{w}}\ \Big). \tag{33}$$

This classification model is known as the margin maximized minimax probability machine (MM-MPM) [Nath and Bhattacharyya, 2007], and it is proved that the MM-MPM maximizes the margin for given acceptable false-positive and false-negative rates.

Another example is not represented as a Minkowski difference. Let us consider the following uncertainty set:

$$\mathcal{U} := \big\{(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{u} \ \big|\ \|\boldsymbol{u}\| \le \kappa\big\}, \tag{34}$$

with a parameter $\kappa \in [0, \kappa_{max}) \subseteq [0, \infty)$*, where* κ_{max} *is the supremum of* κ *such that* $\mathbf{0} \notin \mathcal{U}$*. Then the UCM of the form* (4) *can be formulated as follows:*

$$\min_{\boldsymbol{q}=\boldsymbol{u}\in\mathcal{U}'} \frac{1}{2}\big\|\big(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-\big) + \big(\Sigma_+ + \Sigma_-\big)^{1/2}\boldsymbol{u}\big\|^2 \tag{11}$$

where $\mathcal{U}' := \{\boldsymbol{u} \mid \|\boldsymbol{u}\| \le \kappa\}$ $(\subseteq \mathbb{R}^n)$*. Meanwhile, UCM* (2) *with the uncertainty set* (34) *is equivalent to*

$$\min_{\|\boldsymbol{w}\|\le 1} -\boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + \kappa\sqrt{\boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)\boldsymbol{w}}. \tag{35}$$

The model replacing the Euclidean norm $\|\boldsymbol{w}\|$ *in* (35) *with the* ℓ_1*-norm* $\|\boldsymbol{w}\|_1$ *is equivalent to a sparse feature selection model [Bhattacharyya, 2004] based on Fisher's discriminant analysis (FDA). According to [Takeda et al., 2013], we refer to the model* (35) *as MM-FDA.*

Let us consider a vector $\boldsymbol{q} \in \mathbb{R}^m$. We denote by $\boldsymbol{q}_+$ and $\boldsymbol{q}_-$ the subvectors of $\boldsymbol{q}$ corresponding to the label $+1$ and -1, respectively. $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$ are subvectors of $\boldsymbol{e}$ with size m_+ and m_-.

Example 2 (ν-SVM)**.** *Let* $\mathcal{U}$ *be the Minkowski difference of the reduced convex hulls [Bennett and Bredensteiner, 2000, Crisp and Burges, 2000] of samples in each class, i.e.*

$$\begin{aligned} &\mathcal{U} := \mathcal{U}_+ \ominus \mathcal{U}_-, \\ &\quad \textit{where}\quad \mathcal{U}_o := \Big\{ \sum_{i\in M_o} q_i \boldsymbol{x}_i \ \Big|\ \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \mathbf{0} \le q_i \le \frac{1}{m\nu},\ i \in M_o \Big\}, \quad o \in \{+,-\}, \end{aligned} \tag{36}$$

where $\nu \in (\nu_{min}, \nu_{max}] \subseteq (0, 1]$ *is a parameter,* ν_{min} *is the infimum of* $\nu > 0$ *such that* $\mathbf{0} \notin \mathcal{U}$*, and* $\nu_{max} := \frac{2\max\{m_+, m_-\}}{m}$ *is the maximum of* $\nu \leq 1$ *such that* $\mathcal{U} \neq \phi$*. Then, UCM of the form* (4) *can be formulated as follows:*

$$\min_{\boldsymbol{q} \in \mathcal{U}'} \frac{1}{2} \Big\| \sum_{i \in M_+} q_i \boldsymbol{x}_i - \sum_{i \in M_-} q_i \boldsymbol{x}_i \Big\|^2 \tag{12}$$

where $\mathcal{U}' = \{\boldsymbol{q} \mid \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+, -\},\ \mathbf{0} \leq \boldsymbol{q} \leq \frac{1}{m\nu}\boldsymbol{e}\}$*. Meanwhile, UCM* (2) *is equivalent to* ν*-SVM [Schölkopf et al., 2000]*[‡]*:*

$$\begin{array}{ll} \min\limits_{\boldsymbol{w}, b, \rho, \boldsymbol{\xi}} & -\rho + \dfrac{1}{m\nu} \displaystyle\sum_{i \in M} \xi_i \\ \text{s.t.} & \rho - y_i(\boldsymbol{x}_i^\top \boldsymbol{w} + b) \leq \xi_i,\ i \in M, \\ & \boldsymbol{\xi} \geq \mathbf{0},\ \|\boldsymbol{w}\|^2 \leq 1. \end{array} \tag{37}$$

Indeed, the dual of the inner maximization problem in UCM (2) *coincides with* (37) *(up to a scaling of variables) and the resulting classifiers are the same.*

We note that ν*-SVM is known to be equivalent to the standard SVM [Cortes and Vapnik, 1995] whose parameter is* $C \in (0, \infty)$*. Since the parameter* ν *of* ν*-SVM takes a value in the finite range* $(0, 1]$*, it is numerically advantageous to the standard SVM formulation.*

Example 3 (ℓ_2-loss ν-SVM)**.** *Let*

$$\mathcal{U} := \left\{ \sum_{i \in M_+} q_i \boldsymbol{x}_i - \sum_{i \in M_-} q_i \boldsymbol{x}_i \;\middle|\; \begin{array}{c} \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+, -\},\ \boldsymbol{q} \geq \mathbf{0}, \\ \|\boldsymbol{q}\|^2 \leq \kappa^2 \end{array} \right\}, \tag{38}$$

with a parameter $\kappa \in [\kappa_{min}, \kappa_{max}) \subseteq \left[0, \sqrt{\frac{1}{2}}\right]$*, where* $\kappa_{min} := \frac{1}{2}\sqrt{\frac{1}{m_+} + \frac{1}{m_-}}$ *is the minimum of* $\kappa \geq 0$ *such that* $\mathcal{U} \neq \phi$*, and* κ_{max} *is the supremum of* $\kappa \leq \sqrt{\frac{1}{2}}$ *such that* $\mathbf{0} \notin \mathcal{U}$*; note that* $\mathcal{U}$ *is identical for all* $\kappa \geq \sqrt{\frac{1}{2}}$ *because the constraint* $\|\boldsymbol{q}\|^2 \leq \kappa^2$ *is to be redundant. Then, UCM of the form* (4) *can be formulated as follows:*

$$\min_{\boldsymbol{q} \in \mathcal{U}'} \frac{1}{2} \Big\| \sum_{i \in M_+} q_i \boldsymbol{x}_i - \sum_{i \in M_-} q_i \boldsymbol{x}_i \Big\|^2 \tag{39}$$

where $\mathcal{U}' = \{\boldsymbol{q} \mid \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+, -\},\ \|\boldsymbol{q}\|^2 \leq \kappa^2,\ \boldsymbol{q} \geq \mathbf{0}\}$*. Let us consider the following* ℓ_2*-loss* ν*-SVM:*

$$\begin{array}{ll} \min\limits_{\boldsymbol{w}, b, \boldsymbol{\xi}, \rho} & -\rho + \dfrac{1}{m\nu} \displaystyle\sum_{i \in M} \xi_i^2 \\ \text{s.t.} & \rho - y_i(\boldsymbol{x}_i^\top \boldsymbol{w} + b) \leq \xi_i,\ i \in M \\ & \boldsymbol{\xi} \geq \mathbf{0},\ \|\boldsymbol{w}\|^2 \leq 1, \end{array} \tag{40}$$

[‡]Schölkopf et al. [2000] defines the objective function of ν-SVM by $\frac{1}{2}\|\boldsymbol{w}\|^2 - \nu\rho + \frac{1}{m}\sum_{i \in M} \xi_i$ without the constraint $\|\boldsymbol{w}\|^2 \leq 1$, but we can easily show that the both problems, (37) and the original ν-SVM [Schölkopf et al., 2000], are equivalent because they have the same dual problem.

where $\nu \in (\nu_{min}, \infty) \subseteq (0, \infty)$ is a parameter value, and ν_{min} is the minimum value of $\nu > 0$ such that $\boldsymbol{w}^* \neq \mathbf{0}$ is the optimal solution of (40)*. If κ and ν are set properly, UCM leads to the same decision function as ℓ_2-loss ν-SVM* (40)*. Indeed, for all $\nu \in (0, \infty)$, the dual problem of* (40)*:*

$$\begin{array}{ll} \min\limits_{\boldsymbol{q}} & \Big\| \sum\limits_{i \in M_+} q_i \boldsymbol{x}_i - \sum\limits_{i \in M_-} q_i \boldsymbol{x}_i \Big\| + \dfrac{m\nu}{4} \|\boldsymbol{q}\|^2 \\ \text{s.t.} & \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+, -\},\ \boldsymbol{q} \geq \mathbf{0}, \end{array} \tag{41}$$

has an optimal solution $\boldsymbol{q}^*$, and there exists $\kappa = \|\boldsymbol{q}^*\| \in \left[0, \sqrt{\frac{1}{2}}\right]$ such that UCM (39) *has the same optimal solution $\boldsymbol{q}^*$. The proof follows from Lemma 5 by letting*

$$f(\boldsymbol{q}) = \Big\| \sum_{i \in M_+} q_i \boldsymbol{x}_i - \sum_{i \in M_-} q_i \boldsymbol{x}_i \Big\|, \quad g(\boldsymbol{q}) = \frac{m\nu}{4} \|\boldsymbol{q}\|^2.$$

There are advantages to deal with the formulation (39) *of UCM rather than* (41)*. Since the parameter ν of* (41) *takes a value in the infinite range $(0, \infty)$, it may be hard for practitioners to find the best ν, and more unfavorably, optimization algorithms would be numerically unstable for very large ν. Such issues do not appear for UCM* (39) *since the parameter κ takes a value in the finite range $\left[0, \sqrt{\frac{1}{2}}\right)$.*

We also can construct a biased variant of ℓ_2-loss ν-SVM. Let

$$\mathcal{U} := \left\{ \sum_{i \in M_+} q_i \boldsymbol{x}_i - \sum_{i \in M_-} q_i \boldsymbol{x}_i \;\middle|\; \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \|\boldsymbol{q}_o\|^2 \leq \kappa_o^2,\ o \in \{+, -\},\ \boldsymbol{q} \geq \mathbf{0} \right\}, \tag{42}$$

with different parameters $\kappa_o \in \left[0, \sqrt{\frac{1}{2}}\right]$ for $o \in \{+, -\}$. There exists the maximum range $[\kappa_o^{min}, \kappa_o^{max}) \subseteq \left[0, \sqrt{\frac{1}{2}}\right]$ of κ_o such that $\mathbf{0} \notin \mathcal{U}$ and $\mathcal{U} \neq \phi$, where $\kappa_o^{min} = \frac{1}{2}\sqrt{\frac{1}{m_o}}$. Then, UCM of the form (4) *can be formulated as follows:*

$$\min_{\boldsymbol{q} \in \mathcal{U}'} \frac{1}{2} \Big\| \sum_{i \in M_+} q_i \boldsymbol{x}_i - \sum_{i \in M_-} q_i \boldsymbol{x}_i \Big\|^2 \tag{43}$$

where

$$\mathcal{U}' = \left\{ \boldsymbol{q} \;\middle|\; \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \|\boldsymbol{q}_o\|^2 \leq \kappa_o^2,\ o \in \{+, -\},\ \boldsymbol{q} \geq \mathbf{0} \right\}.$$

The biased variant of ℓ_2-loss ν-SVM is formulated as follows:

$$\begin{array}{ll} \min\limits_{\boldsymbol{w}, b, \boldsymbol{\xi}, \rho} & -\rho + \sum\limits_{o \in \{+, -\}} \Big(\dfrac{1}{m\nu_o} \sum\limits_{i \in M_o} \xi_i^2 \Big) \\ \text{s.t.} & \rho - y_i(\boldsymbol{x}_i^\top \boldsymbol{w} + b) \leq \xi_i,\ i \in M \\ & \boldsymbol{\xi} \geq \mathbf{0},\ \|\boldsymbol{w}\|^2 \leq 1, \end{array} \tag{44}$$

where $\nu_o \in (0,\infty)$, $o \in \{+,-\}$, *are parameters. The dual problem of* (44)*:*

$$\begin{array}{ll} \min\limits_{\boldsymbol{q}} & \Big\| \sum\limits_{i\in M_+} q_i \boldsymbol{x}_i - \sum\limits_{i\in M_-} q_i \boldsymbol{x}_i \Big\| + \sum\limits_{o\in\{+,-\}} \Big(\frac{m\nu_o}{4} \|\boldsymbol{q}_o\|^2 \Big) \\ \text{s.t.} & \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+,-\},\ \boldsymbol{q} \geq \boldsymbol{0}, \end{array} \tag{45}$$

has an optimal solution $\boldsymbol{q}^*$*, and there exists* $\kappa_o = \|\boldsymbol{q}_o^*\| \in \big[0, \sqrt{\frac{1}{2}}\big]$ *such that UCM* (43) *has the same optimal solution* $\boldsymbol{q}^*$*.*

Example 4. *Let*

$$\mathcal{U} := \left\{ \sum_{i\in M_+} q_i \boldsymbol{x}_i - \sum_{i\in M_-} q_i \boldsymbol{x}_i \;\middle|\; \begin{array}{l} \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+,-\},\ \boldsymbol{q} \geq 0 \\ \sum_{i\in M} q_i \log(\frac{q_i}{1/m}) \leq -\log(\alpha) \end{array} \right\}, \tag{46}$$

where $\alpha \in (0,1)$ *is a parameter; there exists the maximum range* $(\alpha_{min}, \alpha_{max}) \subseteq (0,1)$ *of* α *such that* $\boldsymbol{0} \notin \mathcal{U}$ *and* $\mathcal{U} \neq \phi$*. In* $\mathcal{U}$*, the Kullback-Leibler divergence of* $\frac{1}{m}\boldsymbol{e}$ *from* $\boldsymbol{q}$ *is bounded. Then, UCM of the form* (4) *can be formulated as follows:*

$$\min_{\boldsymbol{q}\in\mathcal{U}'} \frac{1}{2} \Big\| \sum_{i\in M_+} q_i \boldsymbol{x}_i - \sum_{i\in M_-} q_i \boldsymbol{x}_i \Big\|^2 \tag{47}$$

where

$$\mathcal{U}' = \left\{ \boldsymbol{q} \;\middle|\; \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+,-\},\ \boldsymbol{q} \geq 0,\ \sum_{i\in M} q_i \log(\frac{q_i}{1/m}) \leq -\log(\alpha) \right\}.$$

Let us consider the following generalized ν*-SVM [Kanamori et al., 2013] whose loss function is* $\exp(\zeta)$*:*

$$\begin{array}{ll} \min\limits_{\boldsymbol{w},b,\boldsymbol{\zeta},\rho} & -\rho + \frac{1}{m} \sum\limits_{i\in M} \exp(\zeta_i) \\ \text{s.t.} & \rho - y_i(\boldsymbol{x}_i^\top \boldsymbol{w} + b) \leq \zeta_i,\ i \in M \\ & \|\boldsymbol{w}\|^2 \leq \lambda^2, \end{array} \tag{48}$$

where $\lambda \in (0,\infty)$ *is a parameter. There exists the supremum* $\lambda_{max} \in (0,\infty)$ *of* λ *such that* $\boldsymbol{w}^* \neq \boldsymbol{0}$ *is the optimal solution of* (48)*. If* α *and* λ *are set properly, UCM leads to the same decision function as the problem* (48)*. Indeed, for all* $\lambda \in (0,\infty)$*, the dual problem of* (48)*:*

$$\begin{array}{ll} \min\limits_{\boldsymbol{q}} & \Big\| \sum\limits_{i\in M_+} q_i \boldsymbol{x}_i - \sum\limits_{i\in M_-} q_i \boldsymbol{x}_i \Big\| + \frac{1}{\lambda} \left(\sum\limits_{i\in M} q_i \left(\log \frac{q_i}{1/m} \right) \right) \\ \text{s.t.} & \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ o \in \{+,-\},\ \boldsymbol{q} \geq \boldsymbol{0}, \end{array} \tag{49}$$

has an optimal solution $\boldsymbol{q}^*$*, and there exists* $\alpha = \exp\Big(-\sum_{i\in M} q_i^* \Big(\log \frac{q_i^*}{1/m} \Big) \Big) \in (0,1)$ *such that UCM* (47) *has the same optimal solution* $\boldsymbol{q}^*$*. The proof follows from Lemma 5*

by letting

$$f(\boldsymbol{q}) = \Big\| \sum_{i \in M_+} q_i \boldsymbol{x}_i - \sum_{i \in M_-} q_i \boldsymbol{x}_i \Big\|, \quad g(\boldsymbol{q}) = \frac{1}{\lambda} \left(\sum_{i \in M} q_i \left(\log \frac{q_i}{1/m} \right) \right).$$

There are advantages to deal with the formulation (47) *of UCM rather than* (49)*. Since the parameter* λ *of* (49) *takes a value in the infinite range* $(0, \infty)$*, it may be hard for practitioners to find the best* ν*, and more unfavorably, optimization algorithms would be numerically unstable for very large* λ*. Such issues do not appear for UCM* (47) *since the parameter* α *takes a value in the finite range* $(0, 1)$*.*

Example 5 (Bertsimas and Takeda [2014])**.** *Let us consider the MM-MPM* (33) *(Example 1). The objective function of MM-MPM* (33) *is known as a classical mean-standard deviation, where* $-\boldsymbol{w}^\top \bar{\boldsymbol{x}}_o$ *and* $\sqrt{\boldsymbol{w}^\top \Sigma_o \boldsymbol{w}}$*,* $o \in \{+, -\}$*, are the sample expected value and the sample standard deviation of the values* $-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$*,* $i \in M_o$*, but the constant term* $-y_i b$ *is omitted. In general, the mean-standard deviation risk measure is not a coherent risk measure (due to lack of monotonicity). Here we modify the uncertainty set* (32) *as*

$$\mathcal{U} := \mathcal{U}_+ \ominus \mathcal{U}_-,$$
$$\text{where} \quad \mathcal{U}_o := \left\{ \bar{\boldsymbol{x}}_o + \Sigma_o^{1/2} \boldsymbol{u}_o \;\middle|\; \begin{array}{c} \|\boldsymbol{u}_o\| \le \kappa_o,\ \boldsymbol{q}_o^\top \boldsymbol{e}_o = \frac{1}{2},\ \boldsymbol{q}_o \ge \boldsymbol{0} \\ \bar{\boldsymbol{x}}_o + \Sigma_o^{1/2} \boldsymbol{u}_o = \sum_{i \in M_o} q_i \boldsymbol{x}_i \end{array} \right\}, \quad o \in \{+, -\},$$

so that $\mathcal{U} \subseteq \mathcal{C}$*. This modification makes the mean-standard deviation coherent (see [Natarajan et al., 2009]). Similarly, while the objective function of MM-FDA* (35) *is not a coherent risk measure, the following modification of the uncertainty set* (34)*:*

$$\mathcal{U} := \left\{ \begin{array}{l} (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) \\ \qquad + (\Sigma_+ + \Sigma_-)^{1/2} \boldsymbol{u} \end{array} \;\middle|\; \begin{array}{c} \|\boldsymbol{u}\| \le \kappa,\ \boldsymbol{q}_+^\top \boldsymbol{e}_+ = \frac{1}{2},\ \boldsymbol{q}_-^\top \boldsymbol{e}_- = \frac{1}{2},\ \boldsymbol{q} \ge \boldsymbol{0} \\ (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2} \boldsymbol{u} = \widetilde{Z} \boldsymbol{q} \end{array} \right\} \tag{50}$$

leads UCM (2) *to a coherent risk measure minimization problem.*