Journal of Machine Learning Research 18 (2017) 1-49 Submitted 5/16; Revised 1/17; Published 3/17

A Unified Formulation and Fast Accelerated Proximal Gradient Method for Classification

Naoki Ito NAOKI_ITO@MIST.I.U-TOKYO.AC.JP
Department of Mathematical Informatics
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan

Akiko Takeda ATAKEDA@ISM.AC.JP
Department of Mathematical Analysis and Statistical Inference
The Institute of Statistical Mathematics
10-3 Midori-cho, Tachikawa, Tokyo 190-8562 Japan

Kim-Chuan Toh MATTOHKC@NUS.EDU.SG
Department of Mathematics
National University of Singapore
Blk S17, 10 Lower Kent Ridge Road, Singapore 119076, Singapore

Editor: Moritz Hardt

Abstract

Binary classification is the problem of predicting the class a given sample belongs to. 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. The difficulty can be mitigated if we have a unified formulation and an efficient universal algorithmic framework for various classification models to expedite the comparison of performance of different models for a given dataset. In this paper, we present a unified formulation of various classification models (including C-SVM, ℓ_2-SVM, ν-SVM, MM-FDA, MM-MPM, logistic regression, distance weighted discrimination) and develop a general optimization algorithm based on an accelerated proximal gradient (APG) method for the formulation. We design various techniques such as backtracking line search and adaptive restarting strategy in order to speed up the practical convergence of our method. We also give a theoretical convergence guarantee for the proposed fast APG 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).

Keywords: restarted accelerated proximal gradient method, binary classification, minimum norm problem, vector projection computation, support vector machine

1. Introduction

Binary classification is one of the most important problems in machine learning. Among the wide variety of binary classification models which have been proposed to date, the most popular ones include support vector machines (SVMs) (Cortes and Vapnik, 1995; Schölkopf et al., 2000) and logistic regression (Cox, 1958). To achieve a good prediction performance,

it is often important for the user to find a suitable model for a given dataset. However, the task of finding a suitable model is often time consuming and tedious as different classification models generally employ different formulations and algorithms. Moreover, the user might have to change not only the optimization algorithms but also solvers/software in order to solve different models. The goal of this paper is to present a unified formulation for various classification models and also to design a fast universal algorithmic framework for solving different models. By doing so, one can simplify and speed up the process of finding the best classification model for a given dataset. We can also compare various classification methods in terms of computation time and prediction performance in the same platform.

In this paper, we first propose a unified classification model which can express various models including C-SVM (Cortes and Vapnik, 1995), ν-SVM (Schölkopf et al., 2000), ℓ_2-SVM, logistic regression (Cox, 1958), MM-FDA, MM-MPM (Nath and Bhattacharyya, 2007), distance weighted discrimination (Marron et al., 2007). The unified model is first formulated as an unconstrained ℓ_2-regularized loss minimization problem, which is further transformed into the problem of minimizing a convex objective function (quadratic function, plus additional terms only for logistic regression) over a simple feasible region such as the intersection of a box and a hyperplane, truncated simplex, unit ball, and so on. Taking different loss functions (correspondingly different feasible regions in the transformed problem) will lead to different classification models. For example, when the feasible region is given by the intersection of a box and a hyperplane, the unified formulation coincides with the well-known C-SVM or logistic regression, and when the region is given by a truncated simplex, the problem is the same as the ν-SVM.

It is commonly acknowledged that there is "no free lunch" in supervised learning in the sense that no single algorithm can outperform all other algorithms in all cases. Therefore, there can be no single "best" software for binary classification. However, by taking advantage of the above-mentioned unified formulation, we can design an efficient algorithm which is applicable to the various existing models mentioned in the last paragraph. Our proposed algorithm is based on the accelerated proximal gradient (APG) method (Beck and Teboulle, 2009; Nesterov, 2005) and during the algorithm, only the procedure for computing the projection onto the associated feasible region differs for different models. In other words, by changing the computation of the projection, our algorithm can be applied to arbitrary classification models (i.e., arbitrary feasible region) without changing the optimization algorithmic framework. The great advantage of our algorithm is that most existing models have simple feasible regions, which make the computation of the projection easy and efficient.

The APG method is known to be one of the most efficient first-order methods theoretically. In order to make our APG based method practically efficient, we further employ various techniques such as backtracking line search (Beck and Teboulle, 2009; Scheinberg et al., 2014) and adaptive restarting strategies (O'Donoghue and Candès, 2015) to speed up the convergence of the algorithm. Our method incorporates several speed-up strategies while guaranteeing its convergence even for a general non-strongly convex function. Since our method is a further improvement of the standard APG method, we will call it as fast accelerated proximal gradient (FAPG) method. Our FAPG method improves several other restarting schemes (Nesterov, 2013; Lin and Xiao, 2015; Su et al., 2014) which have guaranteed convergence in the case when the objective function is strongly convex. To make our method even more efficient in practice, we simplify some steps in the implementation

of our FAPG algorithm, though we can no longer ensure its theoretical convergence. To summarize, 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. Indeed, our method solved SVMs with a linear kernel substantially faster than LIBSVM (Chang and Lin, 2011) which implemented the SMO (Platt, 1998) and SeDuMi (Sturm, 1999) which implemented an interior-point method. Moreover, our FAPG method often run faster than the highly optimized LIBLINEAR (Fan et al., 2008) especially for large-scale datasets with feature dimension $n > 2000$. The FAPG method can be applied not only to the unified classification model but also to the general convex composite optimization problem such as ℓ_1-regularized classification models, which are solved faster than LIBLINEAR in most cases. It may be better to think of a stochastic variant of our method for further improvement and we leave it as a future research topic. We focus here on the deterministic one as the first trial to provide an efficient unified algorithm which is applicable to all well-known existing classification models.

The rest of this paper is organized as follows. Section 2 introduces some preliminary definitions and results on binary classification models and the APG method. Section 3 presents a unified formulation of binary classification models. In Section 4, we provide solution methods for the unified formulation. We develop efficient algorithms for computing projections which are used in the APG method. Then we design our FAPG method combined with various techniques to speed-up its practical convergence. The iteration complexity of $O\big((\log k/k)^2\big)$ of our algorithm is also established, where k is the iteration counter. Numerical experiments are presented in Section 5.

2. Preliminaries

In this section, we introduce some preliminary definitions on binary classification models and the APG method.

2.1 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,

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

Define $M := \{1, \ldots, m\}$, $M_+ := \{i \in M \mid y_i = +1\}$, 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 by 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.

There are various 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), the margin maximized minimax probability machine (MM-MPM) (Nath

and Bhattacharyya, 2007), the model based on Fisher's discriminant analysis (MM-FDA) (Bhattacharyya, 2004; Takeda et al., 2013), and the logistic regression (Cox, 1958).

Many binary classification models have the following formulation which consists of the sum of loss of each sample and a regularization term:

$$\min_{\boldsymbol{w},b} \sum_{i=1}^{m} \ell\big(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - b)\big) + \frac{1}{C}\|\boldsymbol{w}\|_p^p, \tag{1}$$

where $\ell : \mathbb{R} \to \mathbb{R}$ is a proper, closed, and convex function; $C > 0$ is a parameter; and $\|\cdot\|_p$ is the p-norm with $p \in [1, \infty]$.

2.2 Accelerated Proximal Gradient Method

In this section, we first introduce the accelerated proximal gradient (APG) method (Beck and Teboulle, 2009) which is designed for the following problem:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^d} F(\boldsymbol{\alpha}) := f(\boldsymbol{\alpha}) + g(\boldsymbol{\alpha}). \tag{2}$$

Note that one can express a minimization problem constrained over a set S in the form of (2) by setting $g = \delta_S$, where

$$\delta_S(\boldsymbol{\alpha}) = \begin{cases} 0 & (\boldsymbol{\alpha} \in S) \\ +\infty & (\boldsymbol{\alpha} \notin S) \end{cases}$$

is the indicator function of the set S.

To apply the APG method to (2), we need to assume the following conditions:

1. $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed, and convex function which is possibly nonsmooth. Its effective domain $\mathrm{dom}(g) = \{\boldsymbol{\alpha} \in \mathbb{R}^d \mid g(\boldsymbol{\alpha}) < +\infty\}$ is closed and convex.

2. $f : \mathbb{R}^d \to \mathbb{R}$ is a proper, closed, convex, and continuously differentiable function, and its gradient $\nabla f(\cdot)$ is Lipschitz continuous on $\mathbb{R}^d$,[1] i.e., there exists a constant $L > 0$ such that
$$\|\nabla f(\boldsymbol{\alpha}) - \nabla f(\boldsymbol{\beta})\|_2 \le L\|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2 \quad \forall \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^d. \tag{3}$$
The minimum value of such L is referred to as the Lipschitz constant L_f of $\nabla f(\cdot)$.

3. The problem (2) is solvable, i.e., the optimal value is finite and an optimal solution $\boldsymbol{\alpha}^*$ exists.

Let $L \ge L_f$. We define an approximate function $Q_L : \mathbb{R}^d \to \mathbb{R}$ of $f(\boldsymbol{\alpha})$ around $\boldsymbol{\beta}$ and a mapping $T_L(\boldsymbol{\alpha}) : \mathbb{R}^d \to \mathbb{R}^d$ as follows:

$$Q_L(\boldsymbol{\alpha}; \boldsymbol{\beta}) = f(\boldsymbol{\beta}) + \langle \nabla f(\boldsymbol{\beta}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle + \frac{L}{2}\|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2^2 + g(\boldsymbol{\alpha})$$
$$T_L(\boldsymbol{\beta}) = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^d} Q_L(\boldsymbol{\alpha}; \boldsymbol{\beta}).$$

1. It is sufficient if $\nabla f(\cdot)$ is Lipschitz continuous on a neighborhood of $\mathrm{dom}(g)$: the convex hull of $\mathrm{dom}(g) \cup \{\boldsymbol{\beta}^k \mid k = 1, 2, \ldots\}$, where $\boldsymbol{\beta}^k$ $(k = 1, 2, \ldots)$ are points generated at Step 3 of the APG method. Obviously, however, it is not possible to know the points a priori.

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

$$\begin{aligned}\boldsymbol{\alpha}^{k+1} = T_L(\boldsymbol{\alpha}^k) \;&=\; \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^d}\Big\{g(\boldsymbol{\alpha}) + \frac{L}{2}\Big\|\boldsymbol{\alpha} - \Big(\boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k)\Big)\Big\|_2^2\Big\}\\ &= \operatorname{prox}_{g,L}\Big(\boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k)\Big),\end{aligned}$$

where $\operatorname{prox}_{g,L}(\bar{\boldsymbol{\alpha}}) := \operatorname{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^d}\Big\{g(\boldsymbol{\alpha}) + \frac{L}{2}\|\boldsymbol{\alpha}-\bar{\boldsymbol{\alpha}}\|_2^2\Big\}$ is the proximal operator of g. If $g = \delta_S$, then $\operatorname{prox}_{g,L_k} = P_S$, where $P_S(\bar{\boldsymbol{\alpha}}) = \operatorname{argmin}_{\boldsymbol{\alpha}\in S}\|\boldsymbol{\alpha}-\bar{\boldsymbol{\alpha}}\|_2$ is the orthogonal projection of $\bar{\boldsymbol{\alpha}}$ onto the set S. In this case, the above PG method coincides with the gradient projection method. If $g(\cdot) = \|\cdot\|_1$, then $\big(\operatorname{prox}_{g,L}(\boldsymbol{\alpha})\big)_i = \operatorname{sign}(\alpha_i)\max\{0, |\alpha_i| - L\}\quad (i = 1, 2, \ldots, d)$ which is known as the soft-thresholding operator. Other analytical computations of the proximal operators of various g can be found in (Parikh and Boyd, 2014, Section 6).

It is known that the PG method has the iteration complexity such that $F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \le O(1/k)$, where $\boldsymbol{\alpha}^*$ is an optimal solution of (2). The APG method (Beck and Teboulle, 2009), which is also known as *FISTA*, is an acceleration of the PG method. It generates two sequences $\{\boldsymbol{\beta}^k\}_{k=1}^{\infty}$ and $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$. For an arbitrary initial point $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0 \in \mathbb{R}^d$ and $t_1 = 1$, the APG method solves (2) through the following steps ($k = 1, 2, \ldots$):

Accelerated Proximal Gradient Method

Step 1. Compute

$$\boldsymbol{\alpha}^k = T_L(\boldsymbol{\beta}^k) = \operatorname{prox}_{g,L}\Big(\boldsymbol{\beta}^k - \frac{1}{L}\nabla f(\boldsymbol{\beta}^k)\Big).$$

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

Step 3. Compute $\boldsymbol{\beta}^{k+1} = \boldsymbol{\alpha}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$.

For the APG method, the iteration complexity result such that $F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \le O(1/k^2)$ is known (see (Beck and Teboulle, 2009, Theorem 4.4)). The second term $\frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ in Step 3 can be seen as the *momentum* of the sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$. It enlarges the moving distance of the sequences $\{\boldsymbol{\alpha}\}_{k=0}^{\infty}, \{\boldsymbol{\beta}\}_{k=1}^{\infty}$ which may lead them closer to the optimum $\boldsymbol{\alpha}^*$ more quickly. While various other APG methods (e.g, (Nesterov, 2013; Monteiro et al., 2016; Su et al., 2014)) are proposed, the above APG (namely, FISTA) is used in many applications because it is simpler to implement.

It is known that $\boldsymbol{\alpha}^*$ is an optimal solution of (2) if and only if $\boldsymbol{\alpha}^* = T_L(\boldsymbol{\alpha}^*)$. More specifically, the necessary and sufficient optimality condition for $\boldsymbol{\alpha}^*$ to be an optimal solution of (2) is

$$\exists \gamma_\alpha \in \partial g(\boldsymbol{\alpha}^*) \quad \text{s.t.} \quad \langle \nabla f(\boldsymbol{\alpha}^*) + \gamma_\alpha,\ \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \ge 0, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^d. \tag{4}$$

On the other hand, from the definition of $T_L(\boldsymbol{\beta})$, we have

$$\exists \gamma_\beta \in \partial g(T_L(\boldsymbol{\beta})) \quad \text{s.t.} \quad \langle \nabla f(T_L(\boldsymbol{\beta})) + L\big(T_L(\boldsymbol{\beta}) - \boldsymbol{\beta}\big) + \gamma_\beta,\ \boldsymbol{\alpha} - T_L(\boldsymbol{\beta})\rangle \ge 0, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^d, \tag{5}$$

for any $\boldsymbol{\beta} \in \mathbb{R}^d$. The term $L\big(T_L(\boldsymbol{\beta}) - \boldsymbol{\beta}\big)$ in (5) can be seen as the residual of the optimality condition (4). Thus it would be a natural criterion to terminate the APG if $L\|T_L(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|_2 < \epsilon$ with a small constant $\epsilon > 0$.

Despite having a strong iteration complexity result, the APG method may still not be efficient enough for practical purpose. In the following, we describe several well-known strategies to make the APG method practically efficient.

2.2.1 Backtracking Strategy

We assume that L is greater than or equals to the Lipschitz constant L_f of $\nabla f(\boldsymbol{\alpha})$. 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 is usually too conservative (see Table 5 in Section 5). Thus we adopt the following backtracking strategy (Beck and Teboulle, 2009) after Step 1 with arbitrary given constants $\eta_u > 1$ and $L_0 > 0$:

'bt': While

$$F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k), \tag{6}$$

update $L_k \leftarrow \eta_u L_k$ and $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k)$. Set $L_{k+1} \leftarrow L_k$.

The inequality (6) in 'bt' ensures that the complexity result $F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \leq O(1/k^2)$ still holds. We note that the inequality $F(\boldsymbol{\alpha}^k) \leq Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ is satisfied if $L_k \geq L_f$, i.e., it is a weaker condition than (3).

2.2.2 Decreasing Strategy for L_k

Beck and Teboulle (2009) designed the backtracking strategy 'bt' so that the values of L_k is non-decreasing. In fact, the convergence analysis of (Beck and Teboulle, 2009) requires the value of L_k to be non-decreasing. However, it is advantageous to decrease the value of L_k whenever possible since the constant $\frac{1}{L_k}$ gives a larger step size.

To allow L_k to decrease, Scheinberg et al. (2014) modifies the APG method so that $\{t_k\}_{k=1}^{\infty}$ satisfies $t_k/L_k \geq t_{k+1}(t_{k+1}-1)/L_{k+1}$ $(\forall k \geq 1)$ and the sequences $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ and $\{\boldsymbol{\beta}^k\}_{k=1}^{\infty}$ are generated along with $\{t_k\}_{k=1}^{\infty}$. To be specific, let us introduce the following simple step to decrease the value of L_k, where $\eta_d > 1$ is a given constant.

'dec': Set $L_{k+1} \leftarrow L_k/\eta_d$.

The modified APG of (Scheinberg et al., 2014) can be described as in Algorithm 1. It differs from the original APG in that the updates of t_k and $\boldsymbol{\beta}^k$ are added to 'bt' and Step 2 is modified. The convergence of Algorithm 1 is shown as follows.

Proposition 1 (from Scheinberg et al., 2014) *Let S^* be the set of optimal solutions of (2). For any $\boldsymbol{\alpha}^* \in S^*$, the sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ generated by Algorithm 1 satisfies the following inequality:*

$$F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \leq \frac{2\eta_u L_f \|\boldsymbol{\alpha}^0 - \boldsymbol{\alpha}^*\|_2^2}{k^2}, \quad \forall k \geq 1.$$

Algorithm 1 An Accelerated Proximal Gradient Method with Non-Monotonic Backtracking

Input: f, ∇f, g, $\text{prox}_{g,L}$, $\epsilon > 0$, $L_1 = L_0 > 0$, $\eta_u > 1$, $\eta_d > 1$, $k_{max} > 0$, $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0$
Output: $\boldsymbol{\alpha}^k$
Initialize: $t_1 \leftarrow 1,\ t_0 \leftarrow 0$
for $k = 1, \ldots, k_{max}$ **do**
  $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # Step 1
  while $F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ **do**
    $L_k \leftarrow \eta_u L_k$ # 'bt'
    $t_k \leftarrow \frac{1+\sqrt{1+4(L_k/L_{k-1})t_{k-1}^2}}{2}$
    $\boldsymbol{\beta}^k \leftarrow \boldsymbol{\alpha}^{k-1} + \frac{t_{k-1}-1}{t_k}(\boldsymbol{\alpha}^{k-1} - \boldsymbol{\alpha}^{k-2})$
    $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$
  end while
  if $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\| < \epsilon$ **then**
    break
  end if
  $L_{k+1} \leftarrow L_k/\eta_d$ # 'dec'
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4(L_{k+1}/L_k)t_k^2}}{2}$ # Step 2'
  $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^k + \frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ # Step 3
end for

2.2.3 Restarting Strategy

The value $\frac{t_k-1}{t_{k+1}} \in [0, 1)$ in Step 3 determines the amount of *momentum* in $\frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$. When the value $\frac{t_k-1}{t_{k+1}}$ is close to 1, i.e., the momentum is high, the sequences of solutions $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ and $\{\boldsymbol{\beta}^k\}_{k=1}^{\infty}$ would overshoot and oscillate around the optimal solution $\boldsymbol{\alpha}^*$. In order to avoid the oscillation and further speed up the convergence, O'Donoghue and Candès (2015) introduced an adaptive restarting strategy:

're': If $\nabla f(\boldsymbol{\beta}^k)^\top(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}) + g(\boldsymbol{\alpha}^k) - g(\boldsymbol{\alpha}^{k-1}) > 0$,
then update $t_{k+1} \leftarrow 1$, $t_k \leftarrow 0$, $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^{k-1}$, and $\boldsymbol{\alpha}^k \leftarrow \boldsymbol{\alpha}^{k-1}$.

Roughly, the APG method resets the momentum back to zero and restarts from the previous point $\boldsymbol{\alpha}^{k-1}$ if the direction of motion $\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}$ seems to cause the (approximated) objective value to increase, which may be a sign of overshooting. Note that the computational cost of 're' is inexpensive since $\nabla f(\boldsymbol{\beta}^k)$ has already been computed at Step 1. O'Donoghue and Candès (2015) also provided a heuristic convergence analysis for their restarting scheme ('re') when f is a strongly convex (i.e., there exists a constant $\mu > 0$ such that $f(\boldsymbol{\alpha}) - \frac{\mu}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_2^2$ is convex) quadratic function and $g = 0$. However, the convergence of 're' for a general strongly convex objective function is unknown (not to mention for the general non-strongly convex function in our problem).

There are several other restarting schemes (Nesterov, 2013; Lin and Xiao, 2015; Su et al., 2014). They have guaranteed convergence in the case that f is strongly convex.

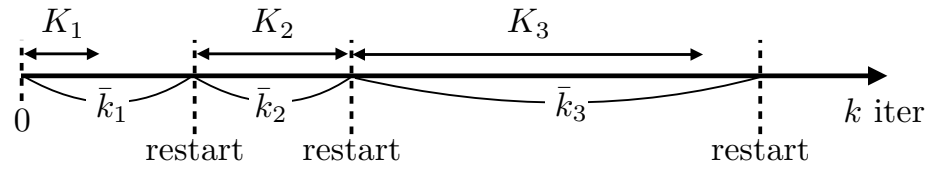

Figure 1: Illustration of Maintaining Top-Speed Strategy ('mt'). A prohibition period of restart for K_i iteration are imposed after the i-th restart occurs. If the condition $\nabla f(\boldsymbol{\beta}^k)^\top(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}) < 0$ is satisfied after the period passed, then the $(i+1)$-th restart occurs. The next prohibition period is doubled, i.e., $K_{i+1} = 2K_i$. $\bar{k}_i (\geq K_i)$ denotes the number of iteration taken between the $(i-1)$-th and i-th restart, which will be used to convergence analysis in Section 4.2.

Nesterov (2013) proposed a restarting method, which has asymptotic linear convergence if g is strongly convex. Lin and Xiao (2015) showed that a similar restart technique can achieve the same convergence rate as Nesterov's scheme if f is strongly convex. Su et al. (2014) modeled the APG method as a second-order ordinary differential equation (ODE). They provided a *speed restarting* framework that ensures a linear convergence of the ODE with respect to time in the case that f is strongly convex and $g = 0$. To the best of our knowledge, none of the restarting schemes have convergence guarantees for a general non-strongly convex function.

2.2.4 Maintaining Top-Speed Strategy

The restarting strategy cancels out high momentum and prevents overshooting. In the neighborhood of an optimum, however, maintaining high momentum may be effective rather than restarting APG (see Figures 6 and 8 in Section 5), because large overshooting may not occur. Thus we put a prohibition period of restart for K_i iteration after the i-th restart occurs, where K_i increases as $K_i = 2K_{i-1}$ $(i = 2, 3, \ldots)$ and $K_1 \geq 2$. See Figure 1 for illustration. Precisely, letting $k = k_{re}$ be the iteration count at which the last restart occurs, we introduce the following step:

'mt': If $k \leq k_{re} + K_i$, then skip 're'. If restart occurs, then update $k_{re} \leftarrow k$, $K_{i+1} \leftarrow 2K_i$, and $i \leftarrow i + 1$.

While a similar strategy is taken in the experiment of (Monteiro et al., 2016), its convergence analysis is not provided. One of our contributions in this paper is to show that in fact the modification 'mt' of the restarting strategy can ensure the convergence rate of $O\big((\log k/k)^2\big)$ under a mild assumption. We will elaborate it in Section 4.2.

3. New Unified Formulation

In this paper, we focus on the following ℓ_2-regularized loss minimization problem:

$$\min_{\boldsymbol{w},b} \ \mathcal{L}(\widetilde{A}^\top \boldsymbol{w} - \boldsymbol{a}b) + \frac{1}{2C}\|\boldsymbol{w}\|_2^2, \tag{7}$$

where $\mathcal{L} : \mathbb{R}^m \to \mathbb{R}$ is a proper, closed, and convex function; $\widetilde{A} \in \mathbb{R}^{n\times l}$, $\boldsymbol{a} \in \mathbb{R}^l$, and $C > 0$ is a parameter. We will later show examples such that

- $l = m, \quad \widetilde{A} = \widetilde{X} := [y_1\boldsymbol{x}_1, y_2\boldsymbol{x}_2, \ldots, y_m\boldsymbol{x}_m], \quad \boldsymbol{a} = \boldsymbol{y}, \quad$ and
- $l = n, \quad \widetilde{A} = I, \quad \boldsymbol{a} = \boldsymbol{0}.$

Our algorithm to be described later can be applied to a more general loss-regularized model with $p \in (1, \infty)$:

$$\min_{\boldsymbol{w},b} \big\{\mathcal{L}(\widetilde{A}^\top \boldsymbol{w} - \boldsymbol{a}b) \mid \|\boldsymbol{w}\|_p \le \lambda\big\}, \tag{8}$$

where $\lambda > 0$ is a parameter. We note that the condition $p \in (1, \infty)$ is required for our unified classification algorithm because the smoothness of the dual norm of $\|\cdot\|_p$ is required to apply the APG method to the dual problem of (8). However, if $p \in \{1, \infty\}$ and $\mathcal{L}$ is smooth, the APG method can be applied to the primal problem (8). See Appendix D for ℓ_1-regularized problems, i.e., the case of $p = 1$.

3.1 Dual Formulation

The dual problems of (7) is given by

$$\min_{\boldsymbol{\alpha}} \ \Big\{\mathcal{L}^*(-\boldsymbol{\alpha}) + \frac{C}{2}\|\widetilde{A}\boldsymbol{\alpha}\|_2^2 \mid \boldsymbol{\alpha}^\top \boldsymbol{a} = 0\Big\}, \tag{9}$$

where $\mathcal{L}^*(\boldsymbol{\alpha}) = \sup_{\boldsymbol{z}}\{\boldsymbol{\alpha}^\top \boldsymbol{z} - \mathcal{L}(\boldsymbol{z})\}$ is the convex conjugate of $\mathcal{L}$.

The dual problem of (8) is

$$\min_{\boldsymbol{\alpha}} \ \big\{\mathcal{L}^*(-\boldsymbol{\alpha}) + \lambda\|\widetilde{A}\boldsymbol{\alpha}\|_p^* \mid \boldsymbol{\alpha}^\top \boldsymbol{a} = 0\big\}, \tag{10}$$

where $\|\boldsymbol{z}\|_p^* = \sup_{\|\boldsymbol{w}\|_p \le 1}\{\boldsymbol{z}^\top \boldsymbol{w}\}$ is the dual norm of $\|\cdot\|_p$. It is known that $\|\cdot\|_p^* = \|\cdot\|_q$, where $q = p/(p-1)$. The problem (10) can be seen as the Fenchel dual of (8), whose derivation is shown in Appendix A. We note that the norms $\|\cdot\|_p$ and $\|\cdot\|_q^*$ are smooth for $p \in (1, \infty)$.

As shown later, in all classification models, $\mathcal{L}^*$ is the sum of a smooth function and an indicator function of a simple set S for which the projection P_S can be computed efficiently. Thus we can apply the APG method as an acceleration of the gradient projection method to the dual problems (9) and (10). By this construction, we can ensure the fast convergence rate of $F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \le O(1/k^2)$ for various classification models in a unified way. In other word, the APG method can obtain a solution with desired accuracy $\epsilon > 0$ in $O(1/\sqrt{\epsilon})$ iteration.

There is an existing work (Zhou et al., 2010) which applies the APG method to the primal C-SVM, i.e., (7) with the hinge loss function. However, their method requires $O(1/\epsilon)$ iteration to obtain a solution with the desired accuracy $\epsilon > 0$ because it approximates the hinge loss by a smoothed function and making the approximation tighter requires extra iterations (Nesterov, 2005).

3.2 Relation to Existing Binary Classification Models

In this section, we show some specific examples of $\mathcal{L}$ and $\mathcal{L}^*$.

3.2.1 C-SVM

Let $\widetilde{A} = \widetilde{X}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (7). Let $\ell(z) = \max\{0, 1 - z\}$ be the hinge loss and $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^{m} \ell(z_i)$. Then the primal problem (7) is known as the standard C-SVM (Cortes and Vapnik, 1995). Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^{m} \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} -\alpha & \text{if } \alpha \in [0,1] \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (9) can be reduced to

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2}\|\widetilde{X}\boldsymbol{\alpha}\|_2^2 - \boldsymbol{\alpha}^\top \boldsymbol{e} + \delta_{S_C}(\boldsymbol{\alpha}), \tag{11}$$

where $S_C = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{0} \le \boldsymbol{\alpha} \le \boldsymbol{e}\}$.

3.2.2 ℓ_2-SVM

Let $\widetilde{A} = \widetilde{X}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (7). Let $\ell(z) = (\max\{0, 1 - z\})^2$ be the truncated squared loss and $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^{m} \ell(z_i)$. Then the primal problem (7) is known as the ℓ_2-SVM. Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^{m} \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} \frac{\alpha^2}{4} & \text{if } \alpha \ge 0 \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (9) can be reduced to

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2}\|\widetilde{X}\boldsymbol{\alpha}\|_2^2 + \frac{\|\boldsymbol{\alpha}\|_2^2}{4} + \delta_{S_{\ell_2}}(\boldsymbol{\alpha}), \tag{12}$$

where $S_{\ell_2} = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{\alpha} \ge \boldsymbol{0}\}$.

3.2.3 Logistic regression

Let $\widetilde{A} = \widetilde{X}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (7). Let $\ell(z) = \log(1 + \exp(-z))$ be the logistic loss and $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^{m} \ell(z_i)$. Then the primal problem (7) is the logistic regression (Cox, 1958). Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^{m} \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} \alpha\log(\alpha) + (1-\alpha)\log(1-\alpha) & (0 < \alpha < 1) \\ 0 & (\alpha = 0, 1) \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (9) can be reduced to

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2}\|\widetilde{X}\boldsymbol{\alpha}\|_2^2 + \sum_{i:\alpha_i > 0} \alpha_i \log(\alpha_i) + \sum_{i:\alpha_i < 1} (1-\alpha_i)\log(1-\alpha_i) + \delta_{S_{\mathrm{LR}}}(\boldsymbol{\alpha}), \tag{13}$$

where $S_{\mathrm{LR}} = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{0} \le \boldsymbol{\alpha} \le \boldsymbol{e}\}$.

We should note that the gradient of the second and third terms in (13) is not Lipschitz continuous on S_{LR} and not defined at $\alpha_i = 0, 1$ $(i \in M)$. However, since it is known that

its optimal solution $\boldsymbol{\alpha}^*$ satisfies $\mathbf{0} < \boldsymbol{\alpha}^* < \boldsymbol{e}$ (Yu et al., 2011), we can instead solve (13) by replacing $S_{\rm LR}$ by $S_{\rm LR}(\xi) := \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi \boldsymbol{e} \le \boldsymbol{\alpha} \le (1-\xi)\boldsymbol{e}\}$, where $\xi > 0$ is a small constant. The second and third terms in (13) are differentiable over $S_{\rm LR}(\xi)$. We will later show numerically that the resulting decision hyperplane is robust to a small deviation of ξ (see Table 6 in Section 5).

3.2.4 ν-SVM

Let $\widetilde{A} = \widetilde{X}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (7). Let

$$\mathcal{L}(\boldsymbol{z}) = \min_{\rho} \Big\{ -\rho + \frac{1}{m\nu} \sum_{i=1}^{m} \max\{\rho - z_i, 0\} \Big\},$$

where $\nu \in (0, 1]$ is a given parameter. The function $\mathcal{L}(\boldsymbol{z})$, known as the conditional value-at-risk (CVaR) (Rockafellar and Uryasev, 2000), is the expected value of the upper ν-tail distribution. Gotoh and Takeda (2005) pointed out that minimizing CVaR is equivalent to ν-SVM (Schölkopf et al., 2000), which is also known to be equivalent to C-SVM.

Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \begin{cases} 0 & \Big(\boldsymbol{e}^\top \boldsymbol{\alpha} = 1 \text{ and } \boldsymbol{\alpha} \in \Big[0, \frac{1}{m\nu}\Big]^m\Big) \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem is given by

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2} \|\widetilde{X}\boldsymbol{\alpha}\|_2^2 + \delta_{S_\nu}(\boldsymbol{\alpha}), \tag{14}$$

where $S_\nu = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{\alpha}^\top \boldsymbol{e} = 1,\ \mathbf{0} \le \boldsymbol{\alpha} \le \frac{1}{m\nu}\boldsymbol{e}\}$ is the intersection of the hyperplane $\{\boldsymbol{\alpha} \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0\}$ and the truncated simplex $\{\boldsymbol{\alpha} \mid \boldsymbol{\alpha}^\top \boldsymbol{e} = 1,\ \mathbf{0} \le \boldsymbol{\alpha} \le \frac{1}{m\nu}\boldsymbol{e}\}$. Note that any positive value for C does not change the optimal solution of (14), and therefore, we can set $C = 1$. There exists a valid range $(\nu_{min}, \nu_{max}] \subseteq (0, 1]$ for ν, where ν_{min} is the infimum of $\nu > 0$ such that $\widetilde{X}\boldsymbol{\alpha}^* \neq \mathbf{0}$, and $\nu_{max} := \frac{2\max\{m_+, m_-\}}{m}$ is the maximum of $\nu \le 1$ for which $S_\nu \neq \phi$. Since the parameter ν takes a value in the finite range $(\nu_{min}, \nu_{max}]$, choosing the parameter value for ν-SVM is often easier than that for C-SVM.

3.2.5 Distance Weighted Discrimination (DWD)

Let $\widetilde{A} = \widetilde{X}$, $\boldsymbol{a} = \boldsymbol{y}$, $\lambda = 1$ and $p = 2$ in (8). For a positive parameter ν, let $\ell(z)$ be the function defined by

$$\begin{aligned} \ell(z) &= \min_{\rho} \Big\{ \frac{1}{\rho} + \frac{1}{m\nu}(\rho - z) \mid \rho \ge z,\ \rho \ge \sqrt{m\nu} \Big\} \\ &= \begin{cases} \frac{1}{z} & \text{if } z \ge \sqrt{m\nu} \\ \frac{2\sqrt{m\nu} - z}{m\nu} & \text{otherwise} \end{cases}. \end{aligned}$$

Consider $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^m \ell(z_i)$. This loss function is used in the distance weighted discrimination (DWD) (Marron et al., 2007) which is proposed as an alternative to ν-SVM. Since we

have

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^{m} \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} -2\sqrt{\alpha} & \left(0 \le \alpha \le \frac{1}{m\nu}\right) \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (9) can be equivalently reduced to

$$\min_{\boldsymbol{\alpha}} \Big\{ \|\widetilde{X}\boldsymbol{\alpha}\|_2 - 2\sum_{i=1}^{m} \sqrt{\alpha_i} + \delta_{S_{\mathrm{DWD}}}(\boldsymbol{\alpha}) \Big\}, \tag{15}$$

where $S_{\mathrm{DWD}} = \big\{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \mathbf{0} \le \boldsymbol{\alpha} \le \frac{1}{m\nu}\boldsymbol{e}\big\}$.

As in the case of the logistic regression, the second term in (15) is not differentiable at $\alpha_i = 0$ ($i \in M$). However, since it is known that there exists an optimal solution $\boldsymbol{\alpha}^*$ such that $\alpha_i^* > 0$ ($i \in M$), we can solve (15) by replacing S_{DWD} by $S_{\mathrm{DWD}}(\xi) = \big\{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi\boldsymbol{e} \le \boldsymbol{\alpha} \le \frac{1}{m\nu}\boldsymbol{e}\big\}$ where $\xi > 0$ is a small constant. The second term in (15) is differentiable over $S_{\mathrm{DWD}}(\xi)$.

3.2.6 Extended Fisher's discriminant analysis (MM-FDA)

The well-known Fisher's discriminant analysis (FDA) uses a decision function to maximize the ratio of the variance between the classes to the variance within the classes. Let $\bar{\boldsymbol{x}}_o$ and Σ_o, $o \in \{+,-\}$, be the mean vectors and the positive definite covariance matrices of $\boldsymbol{x}_i$, $i \in M_o$, respectively. Then FDA is formulated as follows:

$$\max_{\boldsymbol{w}} \ \frac{\big(\boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-)\big)^2}{\boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)\boldsymbol{w}}.$$

Its optimal solution is given by

$$\boldsymbol{w}^* = (\Sigma_+ + \Sigma_-)^{-1}(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-).$$

Let $\widetilde{A} = I$ and $\boldsymbol{a} = \mathbf{0}$ in (7). Here we consider the mean-standard deviation type of risk corresponding to FDA:

$$\mathcal{L}(\boldsymbol{w}) = -\boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + \kappa\sqrt{\boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)\boldsymbol{w}},$$

where $\kappa > 0$ is a parameter. Since

$$\begin{aligned} \mathcal{L}^*(-\boldsymbol{\alpha}) &= \sup_{\boldsymbol{w}} \Big\{ -\boldsymbol{\alpha}^\top\boldsymbol{w} + \boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) - \kappa\sqrt{\boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)\boldsymbol{w}}\Big) \Big\} \\ &= \sup_{\boldsymbol{w}} \min_{\|\boldsymbol{u}\|_2 \le \kappa} \Big\{ -\boldsymbol{\alpha}^\top\boldsymbol{w} + \boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + \boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{u} \Big\} \\ &= \min_{\boldsymbol{u}} \big\{ \delta_{S_{\mathrm{FDA}}}(\boldsymbol{u}) \mid \boldsymbol{\alpha} = (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{u} \big\}, \end{aligned}$$

where $S_{\mathrm{FDA}} = \big\{\boldsymbol{u} \in \mathbb{R}^n \mid \|\boldsymbol{u}\|_2 \le \kappa\big\}$, the dual problem (9) can be reduced to

$$\min_{\boldsymbol{\alpha},\boldsymbol{u}} \Big\{ \delta_{S_{\mathrm{FDA}}}(\boldsymbol{u}) + \frac{C}{2}\|\boldsymbol{\alpha}\|_2^2 \mid \boldsymbol{\alpha} = (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{u} \Big\},$$

which is equivalent to

$$\min_{\boldsymbol{u}} \left\{ \delta_{S_{\mathrm{FDA}}}(\boldsymbol{u}) + \left\| (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{u} \right\|_2^2 \right\}. \tag{16}$$

Takeda et al. (2013) showed that

$$\kappa_{\max} := \left| \begin{array}{ll} \inf_{\boldsymbol{u},\kappa} & \kappa \\ \text{s.t.} & \min_{\boldsymbol{u}} \left\{ \left\| (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{u} \right\|_2 + \delta_{S_{\mathrm{FDA}}}(\boldsymbol{u}) \right\} = 0 \end{array} \right.$$

is equivalent to FDA. Hence (16) can be seen as an extension of FDA. We will refer it as MM-FDA in this paper.

3.2.7 Maximum Margin Minimax Probability Machine (MM-MPM)

Let $\widetilde{A} = I$ and $\boldsymbol{a} = \boldsymbol{0}$ in (7). We consider a class-wise mean-standard deviation type of risk:

$$\mathcal{L}(\boldsymbol{w}) = \Big(-\boldsymbol{w}^\top \bar{\boldsymbol{x}}_+ + \kappa\sqrt{\boldsymbol{w}^\top \Sigma_+ \boldsymbol{w}} \Big) - \Big(-\boldsymbol{w}^\top \bar{\boldsymbol{x}}_- + \kappa\sqrt{\boldsymbol{w}^\top \Sigma_- \boldsymbol{w}} \Big).$$

Similar to MM-FDA, we have

$$\begin{aligned} \mathcal{L}^*(\boldsymbol{w}) &= \sup_{\boldsymbol{w}} \Big\{ -\boldsymbol{\alpha}^\top \boldsymbol{w} + \Big(\boldsymbol{w}^\top \bar{\boldsymbol{x}}_+ - \kappa\sqrt{\boldsymbol{w}^\top \Sigma_+ \boldsymbol{w}} \Big) - \Big(\boldsymbol{w}^\top \bar{\boldsymbol{x}}_- - \kappa\sqrt{\boldsymbol{w}^\top \Sigma_- \boldsymbol{w}} \Big) \Big\} \\ &= \sup_{\boldsymbol{w}} \min_{\substack{\|\boldsymbol{u}_+\|_2 \le \kappa \\ \|\boldsymbol{u}_-\|_2 \le \kappa}} \Big\{ -\boldsymbol{\alpha}^\top \boldsymbol{w} + \big(\boldsymbol{w}^\top \bar{\boldsymbol{x}}_+ + \boldsymbol{w}^\top \Sigma_+^{1/2} \boldsymbol{u}_+ \big) - \big(\boldsymbol{w}^\top \bar{\boldsymbol{x}}_- + \boldsymbol{w}^\top \Sigma_-^{1/2} \boldsymbol{u}_- \big) \Big\} \\ &= \min_{\boldsymbol{u}_+, \boldsymbol{u}_-} \big\{ \delta_{S_{\mathrm{MPM}}}(\boldsymbol{u}_+, \boldsymbol{u}_-) \mid \boldsymbol{\alpha} = \big(\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2} \boldsymbol{u}_+ \big) - \big(\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2} \boldsymbol{u}_- \big) \big\}, \end{aligned}$$

where $S_{\mathrm{MPM}} = \big\{ (\boldsymbol{u}_+, \boldsymbol{u}_-) \in \mathbb{R}^n \times \mathbb{R}^n \mid \|\boldsymbol{u}_+\|_2 \le \kappa,\ \|\boldsymbol{u}_-\|_2 \le \kappa \big\}$. Thus the dual problem (9) can be reduced to

$$\min_{\boldsymbol{\alpha}, \boldsymbol{u}_+, \boldsymbol{u}_-} \Big\{ \delta_{S_{\mathrm{MPM}}}(\boldsymbol{u}_+, \boldsymbol{u}_-) + \frac{C}{2} \|\boldsymbol{\alpha}\|_2^2 \mid \boldsymbol{\alpha} = \big(\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2} \boldsymbol{u}_+ \big) - \big(\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2} \boldsymbol{u}_- \big) \Big\},$$

which is equivalent to

$$\min_{\boldsymbol{u}_+, \boldsymbol{u}_-} \big\{ \delta_{S_{\mathrm{MPM}}}(\boldsymbol{u}_+, \boldsymbol{u}_-) + \big\| \big(\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2} \boldsymbol{u}_+ \big) - \big(\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2} \boldsymbol{u}_- \big) \big\|_2^2 \big\}. \tag{17}$$

The last problem (17) is equivalent to the dual of the maximum margin minimax probability machine (MM-MPM) (Nath and Bhattacharyya, 2007).

4. A Fast APG method for Unified Binary Classification

In this section, we first provide an efficient vector projection computation in order to apply the APG method to the unified formulation of the binary classification models. After that, we develop a fast APG (FAPG) method and show its convergence.

4.1 Vector Projection Computation

When applying the APG method to a constrained optimization problem over S, the projection P_S onto S appears at Step 1. We need to change the computation of the projection P_S depending on the definition of S. In this section, we provide efficient projection computations for S_C, S_{ℓ_2}, $S_{\mathrm{LR}}(\xi)$, S_ν, $S_{\mathrm{DWD}}(\xi)$, S_{MPM}, and S_{FDA}.

4.1.1 Bisection Method for Projection

Many models shown in Section 3 have a linear equality and box constraints. Here we consider the set $S = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{e} = r,\ l \le \alpha_i \le u\ (\forall i \in M)\}$ and provide an efficient algorithm for computing the projection $P_S(\bar{\boldsymbol{\alpha}})$:

$$\min_{\boldsymbol{\alpha}} \Big\{ \frac{1}{2}\|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|_2^2 \mid \boldsymbol{\alpha}^\top \boldsymbol{e} = r, \quad l \le \alpha_i \le u, \quad (i \in M) \Big\}. \tag{18}$$

We assume that (18) is feasible. One way to solve (18) is to use breakpoint search algorithms (e.g., (Helgason et al., 1980; Kiwiel, 2008; Duchi et al., 2008)). They are exact algorithms and have linear time complexity of $O(m)$. Helgason et al. (1980); Kiwiel (2008) developed the breakpoint search algorithms for the continuous quadratic knapsack problem (CQKP) which involves the projection (18) as a special case. By integrating the breakpoint search algorithm and the red-black tree data structure, Duchi et al. (2008) developed an efficient update algorithm of the gradient projection method when the gradient is sparse.

In this paper, we provide a numerical algorithm for (18). Although its worst case complexity is $O\Big(m \log\big(\frac{\bar{\alpha}_{\max} - \bar{\alpha}_{\min}}{\epsilon'}\big)\Big)$, where $\bar{\alpha}_{\max} = \max\{\bar{\alpha}_i \mid i \in M\}$ and $\bar{\alpha}_{\min} = \min\{\bar{\alpha}_i \mid i \in M\}$, it often requires less time to compute a solution $\hat{\boldsymbol{\alpha}}$ such that $\|\hat{\boldsymbol{\alpha}} - P_S(\bar{\boldsymbol{\alpha}})\|_\infty < \epsilon' = u$ in practice, where $u \approx 2.22 \times 10^{-16}$ is the IEEE 754 double precision. Note that the error u can occur even when using the breakpoint search algorithm because u is the supremum of the relative error due to rounding in the floating point number system. Thus, it is sufficient to choose $\epsilon' = u$ as the stopping tolerance for our algorithm in practice. [2]

The difference between the algorithms is that the breakpoint search algorithms use binary search, whereas our algorithm uses bisection to solve an equation. It is known that the projection (18) can be reduced to finding the root of an one dimensional monotone function.

Lemma 2 (e.g. Helgason et al., 1980) *Suppose that the problem* (18) *is feasible. Let*

$$\alpha_i(\theta) = \min\Big\{ \max\{l, \bar{\alpha}_i - \theta\}, u \Big\}, \quad i \in M$$

and let

$$h(\theta) = \sum_{i \in M} \alpha_i(\theta).$$

The following statements hold:

2. Another practical way is to decrease ϵ', i.e., to improve the accuracy of the projection progressively, as APG iterates. The APG method with the inexact computation also share the same iteration complexity $O(1/k^2)$ as the exact counterpart, as shown in (Jiang et al., 2012).

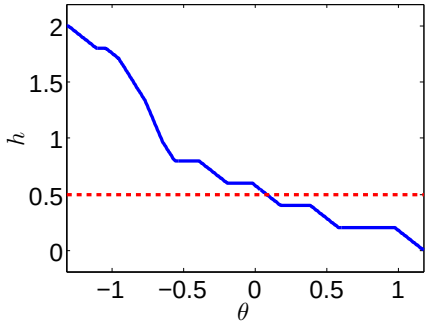

Figure 2: Illustration of the function $h(\theta) := \sum_{i \in M} \alpha_i(\theta)$. The function h is piecewise linear.

1. *$h(\theta)$ is a continuous, non-increasing, and piecewise linear function which has breakpoints at $\bar{\alpha}_i - u$ and $\bar{\alpha}_i - l$ $(i \in M)$.*

2. *There exists $\theta^* \in (\bar{\alpha}_{\min} - \frac{r}{m}, \bar{\alpha}_{\max} - \frac{r}{m})$ such that $h(\theta^*) = r$.*

3. *Let $\hat{\alpha}_i = \alpha_i(\theta^*)$, $i \in M$. Then $\hat{\boldsymbol{\alpha}}$ is an optimal solution of* (18).

An example of $h(\theta)$ is illustrated in Figure 2. To solve $h(\theta) = r$, the breakpoint search uses the binary search to find two adjacent breakpoints that contain a solution θ^* between them. Then the exact solution θ^* can be obtained by linear interpolation of the two breakpoints.

Instead of binary search, we employ the bisection method to solve the equation $h(\theta) = r$.

Bisection Algorithm for Projection

Step 1. Set $\theta^u = \bar{\alpha}_{\max} - \frac{r}{m}$ and $\theta^l = \bar{\alpha}_{\min} - \frac{r}{m}$.

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

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

Step 4. If $h(\hat{\theta}) = r$, then terminate with $\hat{\theta}^* = \hat{\theta}$.
Else if $h(\hat{\theta}) < r$, then set $\theta^u = \hat{\theta}$.
Else if $h(\hat{\theta}) > r$, then set $\theta^l = \hat{\theta}$.

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

All steps require at most $O(m)$ operations and the maximum number of iteration is $\lceil \log(\frac{\bar{\alpha}_{\max} - \bar{\alpha}_{\min}}{\epsilon'}) \rceil$. Thus, the computational complexity of the bisection algorithm is at most $O\big(m \log(\frac{\bar{\alpha}_{\max} - \bar{\alpha}_{\min}}{\epsilon'})\big)$.

As the bisection method narrows the interval (θ^l, θ^u), some of the terms $\alpha_i(\theta)$, $i \in M$, can be set to l, $\bar{\alpha}_i - \theta$, or u for $\theta \in (\theta^l, \theta^u)$. By exploiting the property of $\alpha_i(\theta)$, we can refine the computation of $h(\hat{\theta})$ in Step 3 by avoiding recalculation of certain sums. Moreover, we can often obtain an exact solution. Let us divide M into the following three disjoint sets

for given θ^l and θ^u satisfying $\theta^l < \theta^u$:

$$\begin{aligned} U &:= \{i \in M \mid \bar{\alpha}_i - \theta^u \geq u \quad (\text{i.e., } \alpha_i(\theta) = u, \quad \forall \theta \in [\theta^l, \theta^u])\} \\ L &:= \{i \in M \mid \bar{\alpha}_i - \theta^l \leq l \quad (\text{i.e., } \alpha_i(\theta) = l, \quad \forall \theta \in [\theta^l, \theta^u])\} \\ I &:= M \backslash (U \cup L). \end{aligned}$$

If $\bar{\alpha}_i - \theta^u \geq l$ and $\bar{\alpha}_i - \theta^l \leq u \ \forall i \in I$, then

$$\theta = \Big(|U|u + |L|l + \sum_{i \in I} \alpha_i - r \Big) / |I| \tag{19}$$

is an exact solution. If $I \neq \phi$, then we also divide I into the following three disjoint sets for given $\hat{\theta} \in (\theta^l, \theta^u)$:

$$\begin{aligned} I^U &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \geq u, \quad (\text{i.e., } \alpha_i(\theta) = u \quad \forall \theta \in [\theta^l, \hat{\theta}])\} \\ I^L &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \leq l, \quad (\text{i.e., } \alpha_i(\theta) = l \quad \forall \theta \in [\hat{\theta}, \theta^u])\} \\ I^C &= I \backslash (I^U \cup I^L) \end{aligned}$$

Then we have

$$h(\hat{\theta}) = \sum_{i \in M} \alpha_i(\hat{\theta}) = \underbrace{|U|u}_{s_u} + \underbrace{|I^U|u}_{\Delta s_u} + \underbrace{|L|l}_{s_l} + \underbrace{|I^L|l}_{\Delta s_l} + \underbrace{\sum_{i \in I^C} \alpha_i}_{s_c} - |I^C|\hat{\theta}.$$

By storing the value of each terms, we can reduce the number of additions. The resulting algorithm can be described as Algorithm 2.

Compared to the existing breakpoint search algorithms, our bisection method can avoid the computation of breakpoints and the comparison between them. In many cases, our algorithm can obtain the exact solution by (19). It is often faster than the breakpoint search algorithms in practice (see Table 2 in Section 5).

Remark 3 *Mimicking (Kiwiel, 2008), we can divide the set M more finely and reduce the recalculation of certain sums as shown in Appendix B. However, Algorithm 2 allows for simpler implementation and runs faster on our computer systems.*

In the followings, we use Algorithm 2 to compute the projection onto S_C, S_{ℓ_2}, $S_{\mathrm{LR}}(\xi)$, $S_{\mathrm{DWD}}(\xi)$ and S_ν.

4.1.2 Projection for C-SVM, Logistic Regression, ℓ_2-SVM, DWD

The projection for C-SVM, logistic regression, ℓ_2-SVM, and DWD can be formulated as follows:

$$\min_{\boldsymbol{\alpha}} \Big\{ \frac{1}{2} \|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|_2^2 \mid \boldsymbol{y}^\top \boldsymbol{\alpha} = 0, \quad l \leq \alpha_i \leq u \quad (i \in M) \Big\}, \tag{20}$$

where $(l, u) = (0, 1)$ for C-SVM, $(l, u) = (0, \infty)$ for ℓ_2-SVM, $(l, u) = (\xi, 1 - \xi)$ for logistic regression, and $(l, u) = \big(\xi, \frac{1}{m\nu}\big)$ for DWD. By transforming the variable $\boldsymbol{\alpha}$ and the vector $\bar{\boldsymbol{\alpha}}$ to

$$\beta_i = \begin{cases} \alpha_i & \text{if } y_i = 1 \\ -\alpha_i + l + u & \text{otherwise} \end{cases} \quad \text{and} \quad \bar{\beta}_i = \begin{cases} \bar{\alpha}_i & \text{if } y_i = 1 \\ -\bar{\alpha}_i + l + u & \text{otherwise,} \end{cases}$$

Algorithm 2 Bisection Algorithm for (18)

INPUT: $\bar{\boldsymbol{\alpha}},\ r,\ \boldsymbol{l},\ \boldsymbol{u},\ \epsilon' > 0$ OUTPUT: $\boldsymbol{\alpha}$

INITIALIZE: $I \leftarrow M,\ s_u \leftarrow s_l \leftarrow s_c \leftarrow 0,\ \theta^u \leftarrow \bar{\alpha}_{\max} - \frac{r}{m},\ \theta^l \leftarrow \bar{\alpha}_{\min} - \frac{r}{m}$ # Step 1

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

$\hat{\theta} \leftarrow \frac{\theta^u + \theta^l}{2}$ # Step 2

$\hat{u} \leftarrow u + \hat{\theta}, \quad \hat{l} \leftarrow l + \hat{\theta}$

$I^U \leftarrow \{i \in I \mid \bar{\alpha}_i \geq \hat{u}\}, \quad I^L \leftarrow \{i \in I \mid \bar{\alpha}_i \leq \hat{l}\}$ # Step 3

$I^C \leftarrow I \backslash (I^U \cup I^L)$

$\Delta s_u \leftarrow |I^U| u, \quad \Delta s_l \leftarrow |I^L| l, \quad s_c \leftarrow \sum_{i \in I^C} \alpha_i$

$val \leftarrow s_u + \Delta s_u + s_l + \Delta s_l + s_c - |I^C| \hat{\theta}$

if $val < r$ **then** # Step 4

$\theta^u \leftarrow \hat{\theta}, \quad I \leftarrow I^C \cup I^L$

$s_u \leftarrow s_u + \Delta s_u$

else if $val > r$ **then**

$\theta^l \leftarrow \hat{\theta}, \quad I \leftarrow I^C \cup I^U$

$s_l \leftarrow s_l + \Delta s_l$

else

break

end if

if $\bar{\alpha}_i - \theta^u \geq l$ and $\bar{\alpha}_i - \theta^l \leq u \quad \forall i \in I$ **then**

$\hat{\theta} \leftarrow (s_u + s_l + s_c - r)/|I|$

break

end if

end while

$\alpha_i \leftarrow \alpha_i(\hat{\theta}),\ \forall i \in M$ # Step 5

respectively, the problem (20) can be reduced to

$$\min_{\boldsymbol{\beta}} \Big\{ \frac{1}{2}\|\boldsymbol{\beta} - \bar{\boldsymbol{\beta}}\|_2^2 \mid \boldsymbol{e}^\top \boldsymbol{\beta} = (l+u)m_-, \quad l \le \beta_i \le u \quad (i \in M) \Big\}.$$

The last problem can be solved by Algorithm 2.

4.1.3 Projection for ν-SVM

In applying the APG method to ν-SVM, we shall be concerned with the projection $P_{S_\nu}(\bar{\boldsymbol{\alpha}})$:

$$\min_{\boldsymbol{\alpha}} \Big\{ \frac{1}{2}\|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|_2^2 \;\Big|\; \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{\alpha}^\top \boldsymbol{e} = 1,\ \boldsymbol{0} \le \boldsymbol{\alpha} \le \frac{1}{m\nu}\boldsymbol{e} \Big\}. \tag{21}$$

Let $\boldsymbol{\alpha}_+$ and $\boldsymbol{\alpha}_-$ be the subvectors of $\boldsymbol{\alpha}$ corresponding to the label $+1$ and -1, respectively. Let $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$ be subvectors of $\boldsymbol{e}$ with size m_+ and m_-. From the fact that the conditions $\boldsymbol{\alpha}^\top \boldsymbol{y} = 0$ and $\boldsymbol{\alpha}^\top \boldsymbol{e} = 1$ are equivalent to $\boldsymbol{\alpha}_o^\top \boldsymbol{e}_o = \frac{1}{2}$ $(o \in \{+,-\})$, the problem (21) can be decomposed into the following two problems:

$$\min_{\boldsymbol{\alpha}_o} \Big\{ \frac{1}{2}\|\boldsymbol{\alpha}_o - \bar{\boldsymbol{\alpha}}_o\|_2^2 \;\Big|\; \boldsymbol{e}_o^\top \boldsymbol{\alpha} = \frac{1}{2},\ \boldsymbol{0} \le \boldsymbol{\alpha}_o \le \frac{1}{m\nu}\boldsymbol{e}_o \Big\},\ o \in \{+,-\}. \tag{22}$$

Then, Algorithm 2 can be applied to solve (22).

4.1.4 Projection for MM-MPM and MM-FDA

The projection $P_{S_{\rm FDA}}(\bar{\boldsymbol{\alpha}})$ of $\bar{\boldsymbol{\alpha}}$ onto $S_{\rm FDA} = \big\{\boldsymbol{\alpha} \in \mathbb{R}^n \mid \|\boldsymbol{\alpha}\|_2 \le \kappa\big\}$ is easy to compute; We have

$$P_{S_{\rm FDA}}(\bar{\boldsymbol{\alpha}}) = \min\Big\{1, \frac{\kappa}{\|\bar{\boldsymbol{\alpha}}\|_2}\Big\}\bar{\boldsymbol{\alpha}}.$$

Similarly, the projection $P_{S_{\rm MPM}}(\bar{\boldsymbol{\alpha}}_+, \bar{\boldsymbol{\alpha}}_-)$ for MM-MPM can be computed as follows:

$$P_{S_{\rm MPM}}(\bar{\boldsymbol{\alpha}}_+, \bar{\boldsymbol{\alpha}}_-) = \Big(\min\Big\{1, \frac{\kappa}{\|\bar{\boldsymbol{\alpha}}_+\|_2}\Big\}\bar{\boldsymbol{\alpha}}_+, \min\Big\{1, \frac{\kappa}{\|\bar{\boldsymbol{\alpha}}_-\|_2}\Big\}\bar{\boldsymbol{\alpha}}_- \Big).$$

4.2 Convergence Analysis of a Fast APG Method with Speed-up Strategies

We have shown several strategies to speed-up the APG method in Section 2. While the convergence proof has been shown for the backtracking 'bt' and decreasing 'dec' strategy, the convergence for the restart 're' and maintaining top-speed 'mt' strategy is unknown. In this section, we develop a fast APG (FAPG) method by combining all these techniques and show its convergence.

4.2.1 A Fast APG with Stabilization

In addition to the techniques in Section 2, we also employ the following practical stabilization strategy, where it does *not* affect our convergence analysis.

Decreasing L ('dec') and restarting strategies ('re') shown in Section 2 are effective to speed-up the practical convergence of the APG method, especially in early iterations and near the optimal solution, respectively (see Figures 6 and 10 in Section 5). However, they have opposing effects. Decreasing L_k enlarges the stepsize $\frac{1}{L_k}$, which extends $\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}$.

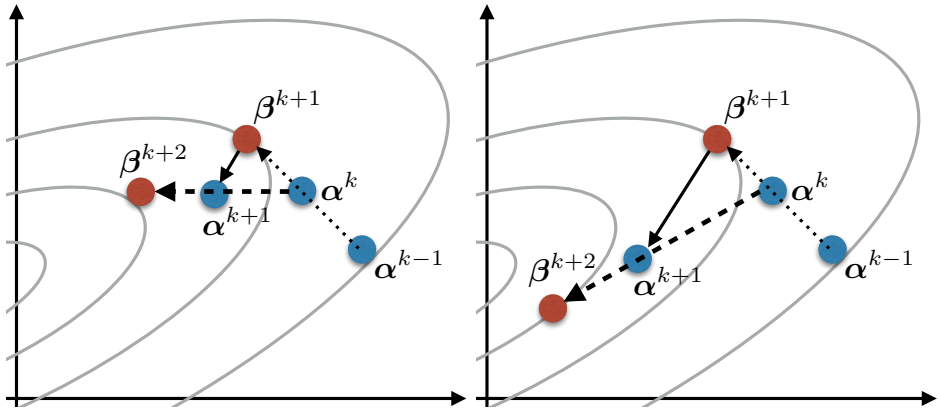

Figure 3: Effects of the value of L_k to the momentum $\frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ (left: L_k is large, right: L_k is small). A smaller value of L_k gives a larger step size at Step 1 as illustrated by the solid lines. This results in high momentum at Step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

This inherently induces high momentum (as illustrated in Figure 3). On the other hand, the restarting strategy cancels out high momentum. Hence combining them triggers the restart frequently and makes the APG method unstable. In order to avoid the instability, it would be necessary to reduce the η_d (rate of decreasing L_k) near the optimum. The restart would occur when the sequences approach the optimum. Therefore we take the following strategy to reduce the value of η_d with a constant $\delta \in (0, 1)$:

'st': Update $\eta_d \leftarrow \delta \cdot \eta_d + (1 - \delta) \cdot 1$ when the restart occurs.

Algorithm 3 is our FAPG method which combines these strategies. The difference from Algorithm 1 is the "if block" after Step 3.

4.2.2 Convergence Analysis

Let $\bar{k}_i$ denotes the number of iteration taken between the $(i-1)$-th and the i-th restart (see Figure 1 for illustration). In the followings, we denote $\boldsymbol{\alpha}^k$ as $\boldsymbol{\alpha}^{(j,k_{j+1})}$ if $k = \sum_{i=1}^{j} \bar{k}_i + k_{j+1}$ $(\bar{k}_{j+1} \geq k_{j+1} \geq 0)$. In other word, $\boldsymbol{\alpha}^{(i,k_{i+1})}$ denotes a point obtained at the k_{i+1}-th iteration counting from the i-th restart. Note that $\boldsymbol{\alpha}^{(i+1,0)} = \boldsymbol{\alpha}^{(i,\bar{k}_{i+1}-1)}$ $(\forall i \geq 0)$. We will frequently switch the notations $\boldsymbol{\alpha}^k$ and $\boldsymbol{\alpha}^{(i,k_{i+1})}$ for notational convenience.

The following lemma is useful to evaluate the function F.

Lemma 4 (from Beck and Teboulle, 2009) *If* $F(T_L(\boldsymbol{\beta})) \leq Q_L(T_L(\boldsymbol{\beta}); \boldsymbol{\beta})$, *then*

$$F(T_L(\boldsymbol{\beta})) - F(\boldsymbol{\alpha}) \leq \frac{L}{2}\{\|\boldsymbol{\beta} - \boldsymbol{\alpha}\|_2^2 - \|T_L(\boldsymbol{\beta}) - \boldsymbol{\alpha}\|_2^2\}, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^d.$$

Using Lemma 4, we obtain the following key lemma.

Algorithm 3 A Fast Accelerated Proximal Gradient (FAPG) Method with Speeding-Up Strategies

Input: f, ∇f, g, $\mathrm{prox}_{g,L}$, $\epsilon > 0$, $L_1 = L_0 > 0$, $\eta_u > 1$, $\eta_d > 1$, $k_{max} > 0$, $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0 = \boldsymbol{\alpha}^{-1}$, $K_1 \geq 2$, $\delta \in (0,1)$
Output: $\boldsymbol{\alpha}^k$
Initialize: $t_1 \leftarrow 1,\ t_0 \leftarrow 0,\ i \leftarrow 1,\ k_{re} \leftarrow 0$
for $k = 1, \ldots, k_{max}$ **do**
  $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \mathrm{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # Step 1
  while $F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ **do**
    $L_k \leftarrow \eta_u L_k$ # 'bt'
    $t_k \leftarrow \frac{1+\sqrt{1+4(L_k/L_{k-1})t_{k-1}^2}}{2}$
    $\boldsymbol{\beta}^k \leftarrow \boldsymbol{\alpha}^{k-1} + \frac{t_{k-1}-1}{t_k}(\boldsymbol{\alpha}^{k-1} - \boldsymbol{\alpha}^{k-2})$
    $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \mathrm{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$
  end while
  if $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\| < \epsilon$ **then**
    break
  end if
  $L_{k+1} \leftarrow L_k/\eta_d$ # 'dec'
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4(L_{k+1}/L_k)t_k^2}}{2}$ # Step 2'
  $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^k + \frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ # Step 3
  if $k > k_{re} + K_i$ and $\langle \nabla f(\boldsymbol{\beta}^k),\ \boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}\rangle + g(\boldsymbol{\alpha}^k) - g(\boldsymbol{\alpha}^{k-1}) > 0$ **then**
    $k_{re} \leftarrow k, \quad K_{i+1} \leftarrow 2K_i, \quad i \leftarrow i+1$ # 'mt'
    $\eta_d \leftarrow \delta \cdot \eta_d + (1-\delta)\cdot 1$ # 'st'
    $t_{k+1} \leftarrow 1, \quad t_k \leftarrow 0, \quad \boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^{k-1}, \quad \boldsymbol{\alpha}^k \leftarrow \boldsymbol{\alpha}^{k-1}$ # 're'
  end if
end for

Lemma 5 *Let the sequence* $\{\boldsymbol{\alpha}^k\}_{k=1}^{\infty}(\equiv\{\boldsymbol{\alpha}^{(i,k_{i+1})}\})$ *be generated by Algorithm 3. The following inequality holds:*

$$F(\boldsymbol{\alpha}^k)\le F(\boldsymbol{\alpha}^0),\quad \forall k\ge 1.$$

Moreover,

$$F(\boldsymbol{\alpha}^{(i,k_{i+1})})\le F(\boldsymbol{\alpha}^{(i,0)}),\quad \forall i\ge 0\ \ \textit{and}\ \ \forall k_{i+1}\in\{0,1,2,\ldots,\bar{k}_{i+1}\},$$

and

$$F(\boldsymbol{\alpha}^{(i+1,0)})\le F(\boldsymbol{\alpha}^{(i,0)}),\quad \forall i\ge 0.$$

Proof First, assume that the restart does not occur (i.e., the steps 're', 'st', and 'mt' are not executed) until the k-th iteration. From Lemma 4, we have

$$F(\boldsymbol{\alpha}^1)-F(\boldsymbol{\alpha}^0)\le\frac{L_1}{2}\{\|\boldsymbol{\beta}^1-\boldsymbol{\alpha}^0\|_2^2-\|\boldsymbol{\alpha}^1-\boldsymbol{\alpha}^0\|_2^2\}\ =\ -\frac{L_1}{2}\|\boldsymbol{\alpha}^1-\boldsymbol{\alpha}^0\|_2^2\le 0. \tag{23}$$

For all $n=1,2,\ldots$, we also have

$$\begin{aligned}F(\boldsymbol{\alpha}^{n+1})-F(\boldsymbol{\alpha}^n)&\le\frac{L_{n+1}}{2}\{\|\boldsymbol{\beta}^{n+1}-\boldsymbol{\alpha}^n\|_2^2-\|\boldsymbol{\alpha}^{n+1}-\boldsymbol{\alpha}^n\|_2^2\}\\&=\frac{L_{n+1}}{2}\Big\{\Big(\frac{t_n-1}{t_{n+1}}\Big)^2\|\boldsymbol{\alpha}^n-\boldsymbol{\alpha}^{n-1}\|_2^2-\|\boldsymbol{\alpha}^{n+1}-\boldsymbol{\alpha}^n\|_2^2\Big\}.\end{aligned}$$

Summing over $n=1,2,\ldots,k-1$, we obtain

$$\begin{aligned}&F(\boldsymbol{\alpha}^k)-F(\boldsymbol{\alpha}^1)\\&\le\frac{1}{2}\left\{L_2\Big(\frac{t_1-1}{t_2}\Big)^2\|\boldsymbol{\alpha}^1-\boldsymbol{\alpha}^0\|_2^2+\sum_{n=2}^{k-1}\Big(L_{n+1}\Big(\frac{t_n-1}{t_{n+1}}\Big)^2-L_n\Big)\|\boldsymbol{\alpha}^n-\boldsymbol{\alpha}^{n-1}\|_2^2-L_k\|\boldsymbol{\alpha}^k-\boldsymbol{\alpha}^{k-1}\|_2^2\right\}.\end{aligned}$$

Note that $t_1-1=0$ and $\Big(L_{n+1}\Big(\frac{t_n-1}{t_{n+1}}\Big)^2-L_n\Big)\le 0$ for all $n\ge 1$ because

$$\frac{t_n-1}{t_{n+1}}=\frac{2(t_n-1)}{1+\sqrt{1+4(L_{n+1}/L_n)t_n^2}}\le\frac{2(t_n-1)}{\sqrt{4(L_{n+1}/L_n)t_n^2}}=\sqrt{\frac{L_n}{L_{n+1}}}\frac{t_n-1}{t_n}.$$

Thus we have

$$F(\boldsymbol{\alpha}^k)-F(\boldsymbol{\alpha}^1)\le 0.$$

Similarly, since the restart does not occur from $\boldsymbol{\alpha}^{(i,0)}$ to $\boldsymbol{\alpha}^{(i,\bar{k}_{i+1})}$ $(\forall i\ge 0)$, we have

$$F(\boldsymbol{\alpha}^{(i,k_{i+1})})\le F(\boldsymbol{\alpha}^{(i,0)}),\quad \forall k_{i+1}\in\{0,1,\ldots,\bar{k}_{i+1}\},$$

and hence by definition,

$$F(\boldsymbol{\alpha}^{(i+1,0)})=F(\boldsymbol{\alpha}^{(i,\bar{k}_{i+1}-1)})\le F(\boldsymbol{\alpha}^{(i,0)}).$$

■

Lemma 5 states that the initial function value $F(\boldsymbol{\alpha}^{(i,0)})$ of each outer iteration gives an upper bound of the subsequent function values and hence the sequence $\{F(\boldsymbol{\alpha}^{(i,0)})\}_{i=1}^{\infty}$ is non-increasing as illustrated in Figure 4.

Now we are ready to show the convergence result of Algorithm 3.

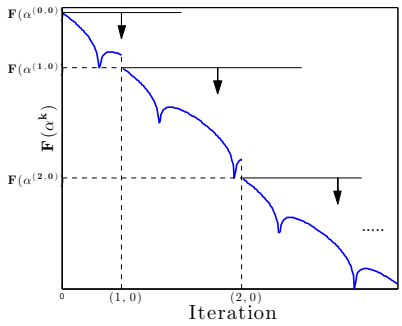

Figure 4: Illustration of the sequence $\{F(\boldsymbol{\alpha}^k)\}_{k=0}^{\infty}$ generated by Algorithm 3. Lemma 5 ensures that the function values $F(\boldsymbol{\alpha}^{(i,0)})$ at restarted points are non-increasing and gives upper bounds of the subsequent function values.

Theorem 6 *Consider the sequence* $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}(\equiv\{\boldsymbol{\alpha}^{(i,k_{i+1})}\})$ *generated by Algorithm 3. Let* S^* *be the set of optimal solutions and* $B := \{\boldsymbol{\alpha} \mid F(\boldsymbol{\alpha}) \le F(\boldsymbol{\alpha}^0)\}$ *be the level set. Assume that there exists a finite* R *such that*

$$R \ge \sup_{\boldsymbol{\alpha}^* \in S^*} \sup_{\boldsymbol{\alpha} \in B} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_2.$$

Then we have

$$F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \le 2\eta_u L_f R^2 \Big(\frac{\log_2(k+2)}{k - \log_2(k+2)}\Big)^2, \quad \forall k \ge 3.$$

Proof From Lemma 5, we have $\boldsymbol{\alpha}^k \in B$ for all $k \ge 1$. Let $\boldsymbol{\alpha}^k = \boldsymbol{\alpha}^{(j,k_{j+1})}$. We assume $k_{j+1} \ge 1$ without loss of generality. From Proposition 1, we have

$$F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F(\boldsymbol{\alpha}^*) \le \frac{2\eta_u L_f \|\boldsymbol{\alpha}^{(j,0)} - \boldsymbol{\alpha}^*\|_2^2}{k_{j+1}^2}$$

Moreover, for all $1 \le i \le j$, Lemma 5 leads to

$$\begin{aligned} F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F(\boldsymbol{\alpha}^*) &\le F(\boldsymbol{\alpha}^{(j,0)}) - F(\boldsymbol{\alpha}^*) \\ &\le F(\boldsymbol{\alpha}^{(i,0)}) - F(\boldsymbol{\alpha}^*) \\ &= F(\boldsymbol{\alpha}^{(i-1,\bar{k}_i-1)}) - F(\boldsymbol{\alpha}^*) \\ &\le \frac{2\eta_u L_f \|\boldsymbol{\alpha}^{(i-1,0)} - \boldsymbol{\alpha}^*\|_2^2}{(\bar{k}_i - 1)^2} \\ &\le \frac{2\eta_u L_f R^2}{(\bar{k}_i - 1)^2}. \end{aligned}$$

Note that $\bar{k}_i \geq K_i \geq 2$. Thus we obtain

$$F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F(\boldsymbol{\alpha}^*) \leq \frac{2\eta_u L_f R^2}{(\max\{\bar{k}_1 - 1, \bar{k}_2 - 1, \ldots, \bar{k}_j - 1, k_{j+1}\})^2} \leq \frac{2\eta_u L_f R^2}{(\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\} - 1)^2}$$

From

$$k \geq \sum_{i=1}^{j} K_i = \frac{K_1(2^j - 1)}{2 - 1} \geq 2^{j+1} - 2,$$

the number of restart j is at most $\log_2(k+2) - 1$. Hence we have

$$\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\} \geq \frac{k}{j+1} \geq \frac{k}{\log_2(k+2)}, \tag{24}$$

which leads to

$$F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F(\boldsymbol{\alpha}^*) \leq 2\eta_u L_f R^2 \Big(\frac{\log_2 k}{k - \log_2(k+2)}\Big)^2, \quad \forall k \geq 3.$$

■

The above theorem ensures the convergence of Algorithm 3 for C-SVM, ν-SVM, MM-MPM, MM-FDA, and ℓ_2-SVM because the level set B is bounded for any initial point $\boldsymbol{\alpha}^0 \in \mathrm{dom}(g)$ since S_C, S_ν, S_{FDA}, and S_{MPM} are bounded and the objective function of ℓ_2-SVM is strongly convex. Similarly, the algorithm is convergent for the logistic regression problem (13) if S_{LR} is replaced by $\{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi \boldsymbol{e} \leq \boldsymbol{\alpha} \leq \boldsymbol{e}\}$ for some small constant $\xi > 0$. To guarantee the convergence of Algorithm 3 for the distance weighted discrimination problem (15), we need to replace S_{DWD} by $S_{\mathrm{DWD}}(\xi) := \big\{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi \boldsymbol{e} \leq \boldsymbol{\alpha} \leq \frac{1}{m\nu}\boldsymbol{e}\big\}$ for some small constant $\xi > 0$ and make the assumption that $\|\widetilde{X}\boldsymbol{\alpha}\|_2 > 0$ for all $\boldsymbol{\alpha} \in S_{\mathrm{DWD}}(\xi)$.

The iteration complexity $O\big((\log(k)/k)^2\big)$ shown in Theorem 6 can be improved to $O\big((W(k)/k)^2\big)$ by finding a better lower bound of (24), where the Lambert's W function $W(\cdot)$ (Corless et al., 1996) is the inverse function of $x\exp(x)$ and diverges slower than $\log(\cdot)$.

Theorem 7 *Consider the sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty} (\equiv \{\boldsymbol{\alpha}^{(i,k_{i+1})}\})$ generated by Algorithm 3. Let S^* be the set of optimal solutions and $B := \{\boldsymbol{\alpha} \mid F(\boldsymbol{\alpha}) \leq F(\boldsymbol{\alpha}^0)\}$ be the level set. Assume that there exists a finite R such that*

$$R \geq \sup_{\boldsymbol{\alpha}^* \in S^*} \sup_{\boldsymbol{\alpha} \in B} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_2.$$

Then we have

$$F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \leq 2\eta_u L_f R^2 \Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2, \quad \forall k \geq 3.$$

Proof See Appendix C. ■

4.3 A Practical FAPG

In this section, we develop a practical version of the FAPG method which reduces the computation cost of FAPG at each iterations.

4.3.1 Heuristic Backtracking Procedure

Although the APG method of (Scheinberg et al., 2014), which is employed in Algorithm 1 and modified in Algorithm 3, can guarantee the convergence, its principal drawback is the extra computational cost at the backtracking step. Recall that

$$Q_{L_k}(\boldsymbol{\alpha}^k;\boldsymbol{\beta}^k) = f(\boldsymbol{\beta}^k) + \langle \nabla f(\boldsymbol{\beta}^k),\ \boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\rangle + \frac{L_k}{2}\|\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\|_2^2 + g(\boldsymbol{\alpha}^k).$$

To check the condition $F(\boldsymbol{\alpha}^k) \le Q_{L_k}(\boldsymbol{\alpha}^k;\boldsymbol{\beta}^k)$, we need to recompute $f(\boldsymbol{\beta}^k)$ and $\nabla f(\boldsymbol{\beta}^k)$ since $\boldsymbol{\beta}^k$ is updated at each loop of the backtracking.

On the other hand, since the original APG of (Beck and Teboulle, 2009) does not update $\boldsymbol{\beta}^k$ during the backtracking step, we can reduce the computation cost by storing the value of $f(\boldsymbol{\beta}^k)$ and $\nabla f(\boldsymbol{\beta}^k)$. Hence we follow the strategy of (Beck and Teboulle, 2009), i.e., we remove the steps for updating t_k and $\boldsymbol{\beta}^k$ subsequent to 'bt' from Algorithm 3 and restore Step 2 as $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$.

4.3.2 Skipping Extra Computations

The backtracking step ('bt') involves extra computation of the function value $F(\boldsymbol{\alpha}^k)$. Checking the termination criteria $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|_2 < \epsilon$ also involves the extra computation of $\nabla f(\boldsymbol{\alpha}^k)$ which is needed for $T_{L_k}(\boldsymbol{\alpha}^k)$. These computation costs can be significant (see Tables 3 and 9 in Section 5). Thus we compute the 'bt' and $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|_2$ in every 10 and 100 iterations, respectively.

Instead of checking the condition $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|_2 < \epsilon$, we check whether $L_k\|\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\|_2 < \epsilon$ at each iteration. The reason for doing so is that computing $L_k\|\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\|_2$ is much cheaper than computing $T_{L_k}(\boldsymbol{\alpha}^k)$. It follows from (4) and (5) that if $\boldsymbol{\alpha}^k = \boldsymbol{\beta}^k$, then $\boldsymbol{\alpha}^k$ is an optimal solution. Moreover, $L_k(\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k)$ represents a residual of a sufficient optimality condition for $\boldsymbol{\alpha}^k$ (and necessary and sufficient optimality condition for $\boldsymbol{\beta}^k$).

Finally, our practical FAPG is described as in Algorithm 4. We note that FAPG can be applied not only to the unified formulation shown in Section 3, but also to the optimization problem of the form (2). See Appendix D in which we demonstrate the performance of FAPG for ℓ_1-regularized classification models.

4.4 Computation of Primal Solution from Dual Solution

Various classification models can be solved in a unified way by applying the vector projection method and the FAPG method to the dual formulation (9) or (10). However, the primal solution $(\boldsymbol{w}, b)$ of (7) or (8) is still required to do classification tasks. In this section, we provide a method to compute the primal solution $(\hat{\boldsymbol{w}}, \hat{b})$ which corresponds to the dual solution $\hat{\boldsymbol{\alpha}}$.

Algorithm 4 A Practical Fast Accelerated Proximal Gradient Method

Input: f, ∇f, g, $\text{prox}_{g,L}$, $\epsilon > 0$, $L > 0$, $\eta_u > 1$, $\eta_d > 1$ $\delta \in (0,1)$, $k_{max} > 0$, $K_1 \geq 2$, $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0$

Output: $\boldsymbol{\alpha}^k$

Initialize: $t_1 \leftarrow 1$, $i \leftarrow 1$, $k_{re} \leftarrow 0$

for $k = 1, \ldots, k_{max}$ **do**

  $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L}\nabla f(\boldsymbol{\beta}^k)\right)$ # Step 1

  if $k \bmod 10 == 1$ **then**

    while $F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ **do**

      $L_k \leftarrow \eta L_k$; $\quad \boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) \leftarrow \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # 'bt'

    end while

  end if

  if $\|L(\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k)\| < \epsilon$ or (($k \bmod 100 == 1$) and ($\|L(T_L(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\| < \epsilon$)) **then**

    break

  end if

  $L_{k+1} \leftarrow L_k/\eta_d$ # 'dec'

  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$ # Step 2

  $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ # Step 3

  if $k > k_{re} + K_i$ and $\langle \nabla f(\boldsymbol{\beta}^k),\ \boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}\rangle + g(\boldsymbol{\alpha}^k) - g(\boldsymbol{\alpha}^{k-1}) > 0$ **then**

    $k_{re} \leftarrow k$, $K_{i+1} \leftarrow 2K_i$, $i \leftarrow i + 1$. # 'mt'

    $\eta_d \leftarrow \delta \cdot \eta_d + (1 - \delta) \cdot 1$. # 'st'

    $t_{k+1} \leftarrow 1$, $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^{k-1}$, $\boldsymbol{\alpha}^k \leftarrow \boldsymbol{\alpha}^{k-1}$ # 're'

  end if

end for

4.4.1 Computation of $\boldsymbol{w}$

The primal vector $\hat{\boldsymbol{w}}$ in (8) corresponding to the dual solution $\hat{\boldsymbol{\alpha}}$ (and $\hat{\boldsymbol{u}}$ for MM-FDA and MM-MPM) can be obtained as

$$\hat{\boldsymbol{w}} = \operatorname*{argmax}_{\|\boldsymbol{w}\|_p \le 1} \boldsymbol{w}^\top \boldsymbol{z}, \quad \text{with} \quad \hat{w}_i = \frac{\operatorname{sign}(z_i)|z_i|^{q-1}}{\|\boldsymbol{z}\|_q^{q-1}} \quad \forall\, i \in M, \tag{25}$$

where $q = p/(p-1)$ and $\boldsymbol{z} = \widetilde{A}\hat{\boldsymbol{\alpha}}$ (i.e., $\boldsymbol{z} = \widetilde{X}\hat{\boldsymbol{\alpha}}$ for C-SVM, ℓ_2-SVM, ν-SVM, logistic regression, and DWD; $\boldsymbol{z} = \hat{\boldsymbol{\alpha}} = (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\hat{\boldsymbol{u}}$ for MM-FDA; and $\boldsymbol{z} = \hat{\boldsymbol{\alpha}} = (\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2}\hat{\boldsymbol{u}}_+) - (\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2}\hat{\boldsymbol{u}}_-)$ for MM-MPM).

4.4.2 Computation of b

There are some issues on computing an optimal bias term b. For C-SVM, ℓ_2-SVM, ν-SVM, DWD, and logistic regression, the bias term b is derived from the Lagrange multiplier corresponding to the constraint $\boldsymbol{\alpha}^\top \boldsymbol{y} = 0$. However, it is often difficult to compute the corresponding Lagrange multiplier. In addition, MM-FDA and MM-MPM does not provide a specific way to compute the bias term. Thus we need to estimate an appropriate value of b.

One of the efficient ways is to estimate b as the minimum solution of training error under a given $\hat{\boldsymbol{w}}$, i.e.,

$$\hat{b} \in \operatorname*{argmin}_b \ \rho(b) := \sum_{i=1}^m \ell\big(y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}} - b)\big), \quad \text{where} \quad \ell(z) = \begin{cases} 1 & (z < 0) \\ 0 & (\text{otherwise}). \end{cases} \tag{26}$$

Let $\zeta_i = \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}$ $(i = 1, 2, \ldots, m)$. Let σ be the permutation such that $\zeta_{\sigma(1)} \le \zeta_{\sigma(2)} \le \ldots \le \zeta_{\sigma(m)}$. If $b < \zeta_{\sigma(1)}$, then $\boldsymbol{x}_i^\top \hat{\boldsymbol{w}} - b > 0$ $(\forall i \in M)$, i.e., all samples are predicted as positive $\hat{y} = +1$. Then we have m_- misclassified samples and thus $\rho(b) = m_-$. If $b \in (\zeta_{\sigma(1)}, \zeta_{\sigma(2)})$, then only $\boldsymbol{x}_{\sigma(1)}$ is predicted as negative $\hat{y} = -1$. Thus we have $\rho(b) = m_- + y_{\sigma(1)}$. Similarly, if $b \in (\zeta_{\sigma(k)}, \zeta_{\sigma(k+1)})$, then we have $\rho(b) = m_- + \sum_{i=1}^k y_{\sigma(i)}$. Thus, by letting $k^* \in \operatorname{argmin}_k \sum_{i=1}^k y_{\sigma(i)}$, an arbitrary constant $\hat{b} \in (\zeta_{\sigma(k^*)}, \zeta_{\sigma(k^*+1)})$ is an optimal solution of the problem (26). The minimization algorithm is as follows:

Step 1. Compute $\zeta_i = \hat{\boldsymbol{w}}^\top \boldsymbol{x}_i$ $(i = 1, 2, \ldots, m)$.

Step 2. Sort ζ_i as $\zeta_{\sigma(1)} \le \zeta_{\sigma(2)} \le \ldots \le \zeta_{\sigma(m)}$.

Step 3. Find $k^* = \operatorname{argmin}_k \sum_{i=1}^k y_{\sigma(i)}$.

Step 4. Compute $\hat{b} = (\zeta_{(k^*)} + \zeta_{(k^*+1)})/2$.

Its computational complexity is $O\big(m(n + \log m)\big)$.

4.4.3 Duality Gap

The component b obtained above is not necessarily the primal optimal solution. Moreover, $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{u}}$ are close to optimal, but may not be exactly optimal due to e.g., numeric overflow.

We still obtain a primal solution $(\hat{\boldsymbol{w}}, \hat{b})$ with some optimality gap guarantee. Since $(\hat{\boldsymbol{w}}, \hat{b})$ and $\hat{\boldsymbol{\alpha}}$ are the primal and dual feasible solutions, respectively, we can obtain the duality gap by

$$\left\{\mathcal{L}(\widetilde{A}^\top \hat{\boldsymbol{w}} - \boldsymbol{a}\hat{b}) + \frac{1}{2C}\|\hat{\boldsymbol{w}}\|_2^2)\right\} - \left\{\mathcal{L}^*(-\hat{\boldsymbol{\alpha}}) + \frac{C}{2}\|\widetilde{A}\hat{\boldsymbol{\alpha}}\|_2^2\right\}$$

for the formulations of (7) and (9); and

$$\mathcal{L}(\widetilde{A}^\top \hat{\boldsymbol{w}} - \boldsymbol{a}\hat{b}) - \left\{\mathcal{L}^*(-\hat{\boldsymbol{\alpha}}) + \lambda\|\widetilde{A}\hat{\boldsymbol{\alpha}}\|_p^*\right\}$$

for the formulations of (8) and (10). Therefore, the obtained primal solution $(\hat{\boldsymbol{w}}, \hat{b})$ has a guarantee that the gap between its objective value and the optimal value is at most the duality gap.

5. Numerical Experiment

In this section we demonstrate the performance of our proposed FAPG 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 FAPG method (Algorithm 4) in MATLAB R2013a and the bisection method (Algorithm 2) in 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 $n \times n$ 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 the 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.

5.1 Projection Algorithms

Before presenting the performance of our practical FAPG algorithm, we compared the performance of our bisection algorithm (Algorithm 2) against the breakpoint search algorithm (Kiwiel, 2008, Algorithm 3.1) with random pivoting. Both algorithms were implemented in C++. We generated $\mathbb{R}^n$-valued random vectors $\tilde{\boldsymbol{\alpha}}$ with uniformly distributed elements and computed the projections $P_{S_\nu}(\tilde{\boldsymbol{\alpha}})$ of $\tilde{\boldsymbol{\alpha}}$ onto S_ν, where $S_\nu := \left\{\boldsymbol{\alpha} \mid \boldsymbol{e}_o^\top \boldsymbol{\alpha}_o = \frac{1}{2},\ o \in \{+, -\},\ \boldsymbol{0} \leq \boldsymbol{\alpha} \leq \frac{1}{m\nu}\boldsymbol{e}\right\}$ 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 the computation times and the number of iterations of 20 trials. As we can see from Table 2, the bisection method was faster and more stable (in the sense that the deviations are smaller) than the breakpoint search algorithm. This explains why we

data	m (	m_+,	m_-)	n	range	density	source
a8a	22,696 (	5,506,	17,190)	123	$[0,1]^n$	0.113	(Bache and Lichman, 2013)
a9a	32,561 (	7,841,	24,720)	123	$[0,1]^n$	0.113	(Bache and Lichman, 2013)
australian	690 (	307,	383)	14	$[-1,1]^n$	0.874	(Bache and Lichman, 2013)
breast-cancer	683 (	444,	239)	10	$[-1,1]^n$	1.000	(Bache and Lichman, 2013)
cod-rna	59,535 (	39,690,	19,845)	8	$[0,1]^n$	0.999	(Uzilov et al., 2006)
colon-cancer	62 (	40,	22)	2,000	$[0,1]^n$	0.984	(Alon et al., 1999)
covtype	581,012 (	297,711,	283,301)	54	$[0,1]^n$	0.221	(Bache and Lichman, 2013)
diabetes	768 (	500,	268)	8	$[-1,1]^n$	0.999	(Bache and Lichman, 2013)
duke	44 (	21,	23)	7,129	$[0,1]^n$	0.977	(West et al., 2001)
epsilon	400,000 (	199,823,	200,177)	2,000	$[-0.15,0.16]^n$	1.000	(Sonnenburg et al., 2008)
fourclass	862 (	307,	555)	2	$[-1,1]^n$	0.996	(Ho and Kleinberg, 1996)
german.numer	1,000 (	300,	700)	24	$[-1,1]^n$	0.958	(Bache and Lichman, 2013)
gisette	6,000 (	3,000,	3,000)	5,000	$[-1,1]^n$	0.991	(Guyon et al., 2005)
heart	270 (	120,	150)	13	$[-1,1]^n$	0.962	(Bache and Lichman, 2013)
ijcnn1	35,000 (	3,415,	31,585)	22	$[-0.93,1]^n$	0.591	(Prokhorov, 2001)
ionosphere	351 (	225,	126)	34	$[-1,1]^n$	0.884	(Bache and Lichman, 2013)
leu	38 (	11,	27)	7,129	$[0,1]^n$	0.974	(Golub et al., 1999)
liver-disorders	345 (	145,	200)	6	$[-1,1]^n$	0.991	(Bache and Lichman, 2013)
madelon	2,000 (	1,000,	1,000)	500	$[0,1]^n$	0.999	(Guyon et al., 2005)
mushrooms	8,124 (	3,916,	4,208)	112	$[0,1]^n$	0.188	(Bache and Lichman, 2013)
news20.binary	19,996 (	9,999,	9,997)	1,355,191	$[0,1]^n$	3.36E-04	(Keerthi and DeCoste, 2005)
rcv1-origin	20,242 (	10,491,	9,751)	47,236	$[0,0.87]^n$	0.002	(Lewis et al., 2004)
real-sim	72,309 (	22,238,	50,071)	20,958	$[0,1]^n$	0.002	(McCallum)
skin-nonskin	245,057 (	50,859,	194,198)	3	$[0,1]^n$	0.983	(Bache and Lichman, 2013)
sonar	208 (	97,	111)	60	$[-1,1]^n$	1.000	(Bache and Lichman, 2013)
splice	1,000 (	517,	483)	60	$[-1,1]^n$	1.000	(Bache and Lichman, 2013)
svmguide1	3,089 (	1,089,	2,000)	4	$[0,1]^n$	0.997	(Hsu et al., 2003)
svmguide3	1,243 (	947,	296)	22	$[0,1]^n$	0.805	(Hsu et al., 2003)
url	2,396,130 (	1,603,985,	792,145)	3,231,961	$[0,1]^n$	3.54E-05	(Ma et al., 2009)
w7a	24,692 (	740,	23,952)	300	$[0,1]^n$	0.039	(Platt, 1998)
w8a	49,749 (	1,479,	48,270)	300	$[0,1]^n$	0.039	(Platt, 1998)

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

		msec.(#iter.)			
		Breakpoint		Bisection	
dim. n	range	ave.	std.	ave.	std.
100,000	$[0,10]^n$	8.3 (25.7)	2.0 (5.4)	4.9 (31.1)	0.6 (4.7)
100,000	$[0,1000]^n$	9.0 (27.3)	1.4 (4.1)	5.5 (27.1)	1.2 (3.2)
1,000,000	$[0,10]^n$	94.2 (22.6)	16.2 (4.3)	49.9 (32.0)	3.4 (3.1)
1,000,000	$[0,1000]^n$	99.1 (26.5)	15.6 (3.2)	53.4 (30.0)	4.4 (1.5)

Table 2: Runtime of Projection Algorithms

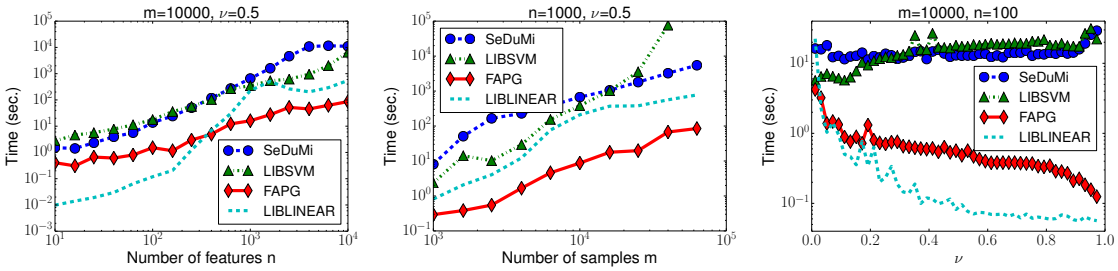

Figure 5: Computation Time for ν-SVM

have chosen to use the bisection methods in Section 4.1 to perform the projection steps in Algorithm 4.

5.2 ν-SVM

As mentioned in Section 3.2, the standard C-SVM (11) and ν-SVM (14) are equivalent. Here we chose ν-SVM to solve because choosing the parameter of ν-SVM is easier than that of C-SVM. We solved the ν-SVM (14) via our FAPG method, SeDuMi (Sturm, 1999), and LIBSVM (Chang and Lin, 2011). SeDuMi is a general purpose optimization solver implementing an interior point method for large-scale second-order cone problems such as (14). LIBSVM implements the sequential minimal optimization (SMO) (Platt, 1998) which is specialized for learning ν-SVM. For reference, we also compared the FAPG method with LIBLINEAR (Fan et al., 2008) which implements a highly optimized stochastic dual coordinate descent method (Hsieh et al., 2008) for C-SVM[3] (Cortes and Vapnik, 1995) and is known to be quite an 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, [4] i.e., it solves a less complex model than the ν-SVM (14) in order to speed up the computation.

We terminate the algorithms if the violation of the KKT optimality condition is less than $\epsilon = 10^{-6}$. The heuristic option in LIBSVM was set to "off" in order to speed up its convergence for large datasets. In the FAPG method, (η_u, η_d, δ) were set to $(1.1, 1.1, 0.8)$. L_0 was set to the maximum value in the diagonal elements of $\widetilde{X}^\top \widetilde{X}$ (i.e., the coefficient matrix of the quadratic form). The initial point $\boldsymbol{\alpha}^0$ was set to the *center* $\boldsymbol{\alpha}^c$ of S_ν, i.e. $\alpha_i^c = \frac{1}{2m_o}$, $i \in M_o$, $o \in \{+, -\}$.

5.2.1 Scalability

First, we compare the computation time with respect to the size of the datasets and parameters using artificial datasets. The results 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 FAPG method has a clear advantage when the dimension n is high, say

3. 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.
4. 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).

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

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

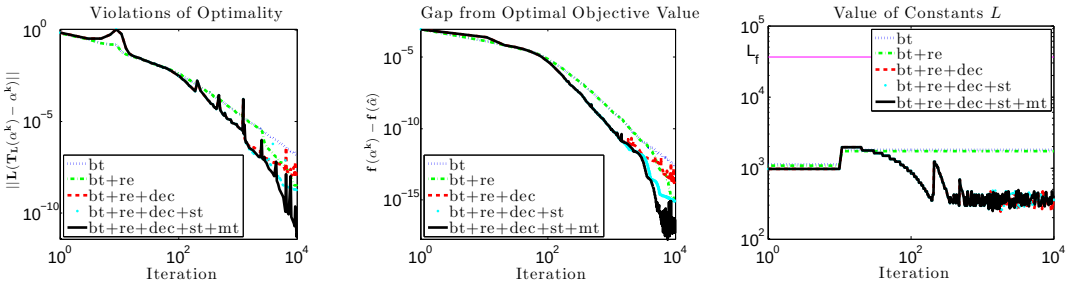

Figure 6: Effect of various acceleration strategies for the APG Method.

$n \geq 10^3$. 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 FAPG method (and LIBLINEAR) is very efficient when ν is larger than 0.1. This can be attributed to the fact that larger ν shrinks the feasible region S_ν and shorten the distance between the initial point $\boldsymbol{\alpha}^0 = \boldsymbol{\alpha}^c$ and the optimal solution $\boldsymbol{\alpha}^*$.

5.2.2 Runtime Breakdown

Table 3 shows the runtime breakdown of the FAPG method for the artificial dataset with $(m, n, \nu) = (10000, 1000, 0.5)$. The computation of the gradient $\nabla f(\boldsymbol{\alpha})$ and the function $f(\boldsymbol{\alpha})$ was the most time-consuming parts in the FAPG method. Since the computations of 'bt' (and $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$) involve the extra evaluation of $f(\boldsymbol{\alpha}^k)$ (and/or $\nabla f(\boldsymbol{\alpha}^k)$), computing them only every 100 iteration (and 10 iteration, respectively) as in Algorithm 4 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.

5.2.3 Effect of Each Acceleration Strategy

Figure 6 shows the running history of the FAPG method with various acceleration strategies for the artificial dataset with $(m, n, \nu) = (10000, 1000, 0.5)$.

The left panel depicts the violations of the optimality, i.e. the values of $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$. The FAPG method with 'bt+re' restarted at $k = 2594$, where the sharp decrease occurred. 'dec' was effective to reduce $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|$ in the early iterations. The FAPG method with 'bt+re+dec' seems to be unstable near the optimum (say $k \geq 10^3$), but

the one with 'bt+re+dec+st' converged stably. Moreover, 'bt+re+dec+st+mt' obtained much more accurate solution than others. We note that several spikes of the values occurred when using 'dec' because it sometimes leads a small value of L_k that violates the condition (6). However, the violation was swiftly recovered in every 10 iterations by 'bt' and did not affect the speed of convergence so much.

The middle panel illustrated the gap in objective value $f(\boldsymbol{\alpha}^k) - f(\hat{\boldsymbol{\alpha}})$ where we regarded $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{10000}$ of 'bt+re+dec+st+mt' as an optimal solution. The gap behaved similar to the violations of the optimality $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$. At $k = 1000$, the objective value $f(\boldsymbol{\alpha}^k)$ reached $f(\hat{\boldsymbol{\alpha}})$ 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_k of the FAPG at each iteration and the value of Lipschitz constant L_f; L_f is known to be the largest eigenvalue of $\widetilde{X}^\top \widetilde{X}$. While $L_f = 3.61 \times 10^4$, the FAPG method with 'bt+re+dec+st+mt' leads to a much smaller average value of 3.68×10^2 for L_k, while the maximum value for L_k is 1.95×10^3.

5.2.4 Benchmark Results

We also conducted experiments using the benchmark datasets. C was set to 10. The computation time is shown in Table 4. When using linear kernel, the FAPG method outperformed LIBSVM and SeDuMi for large datasets where $m \geq 50000$. We indicated the smallest computation time among the three methods except for LIBLINEAR by boldface because the optimal solutions of LIBLINEAR are different from those of the other three methods (note that LIBLINEAR solves a simpler type of SVM, i.e., C-SVM without the bias term b). However, the FAPG method had an advantage over LIBLINEAR for many datasets where $n \geq 2000$.

When using the RBF kernel, SeDuMi broke down for datasets with $m \geq 10000$ as in the case of the artificial datasets. The FAPG method run out of memory for $m \geq 50000$ since it requires the $m \times m$ dense kernel matrix K to compute the gradient $\nabla f(\boldsymbol{\alpha})$. The FAPG can avoid the memory shortage by computing the elements of K on demand as LIBSVM does for large datasets. Although the current implementation has room for improvement, our practical FAPG method was still competitive with LIBSVM and had stable and good performance for datasets with large n.

We should remark that the number of iterations taken by the FAPG 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 $\nabla f(\boldsymbol{\alpha})$ changes from $O(mn)$ to $O(m^2)$. Hence the total runtime tended to increase except for "gisette" whose n and m have the same order of magnitude.

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

Table 5 shows the values taken by the parameter L_k. We can see that our practical FAPG method (with backtracking strategy and decreasing strategy for L_k) keep the values of L_k to be much smaller than the Lipschitz constant L_f.

				Linear				RBF			
data	m	n	ν	SeDuMi	LIBSVM	FAPG (iter)	LIBLIN	SeDuMi	LIBSVM	FAPG (iter)	
a8a	22,696	123	0.368	16.82	32.31	**1.70** (563)	0.06	–	**68.24**	72.88(343)	
a9a	32,561	123	0.365	25.42	66.11	**4.84** (665)	0.07	–	**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)	0.17	**	267.56	**(**)	
colon-cancer	62	2,000	0.078	0.18	**0.01**	0.09 (210)	0.15	0.17	**0.013**	**0.013**(109)	
covtype	581,012	54	0.620	288.71	16164.88	**60.75** (701)	1.55	**	–	**(**)	
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)	0.30	0.09	0.041	**0.017**(120)	
epsilon	400,000	2,000	0.500	–	–	**1685.33** (2643)	–	**	–	**(**)	
fourclass	862	2	0.543	0.12	**0.01**	0.07 (356)	0.0004	12.34	0.069	**0.068**(150)	
german.numer	1,000	24	0.525	**0.26**	**0.26**	0.29 (1107)	0.04	17.13	0.14	**0.10**(236)	
gisette	6,000	5,000	0.100	4572.24	57.48	**9.31** (590)	0.41	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)	0.29	–	**73.83**	236.15(511)	
ionosphere	351	34	0.202	0.25	**0.02**	0.22 (1064)	0.34	0.85	**0.012**	0.051(234)	
leu	38	7,129	0.070	0.40	**0.03**	0.13 (175)	0.17	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)	87.43	133.85	3.77	**0.24**(108)	
mushrooms	8,124	112	0.096	8.74	1.34	**0.56** (435)	0.06	28505.46	**5.74**	19.02(661)	
news20.binary	19,996	1,355,191	0.100	–	1333.26	**22.29** (321)	1.57	–	929.85	**51.86**(11)	
rcv1-origin	20,242	47,236	0.097	–	587.10	**7.82** (485)	8.96	–	192.16	**97.68**(189)	
real-sim	72,309	20,958	0.065	–	6351.62	**12.71** (384)	4.33	**	**1879.90**	**(**)	
skin-nonskin	245,057	3	0.233	62.70	726.20	**8.02** (609)	0.09	**	**4425.59**	**(**)	
sonar	208	60	0.117	0.29	**0.12**	0.33 (1922)	3.81	0.27	**0.009**	0.060(279)	
splice	1,000	60	0.432	0.56	0.25	**0.11** (331)	0.02	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)	1.70	30.17	0.23	**0.21**(430)	
url	2,396,130	3,231,961	0.500	–	–	**4853.91** (2521)	–	**	**	**(**)	
w7a	24,692	300	0.031	69.90	135.91	**14.13** (4280)	0.81	–	**17.89**	286.27(1208)	
w8a	49,749	300	0.031	189.40	143.42	**38.69** (5960)	0.47	–	**124.55**	1423.42(2100)	

Table 4: 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 except for LIBLINEAR are indicated by boldface. The underlined results are better than LIBLINEAR.

	Linear			RBF		
	L_k(FAPG)			L_k(FAPG)		
	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	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	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

Table 5: Constants for Benchmark Datasets

	Artificial	svmguide1	mushrooms	a8a
$\xi = 10^{-2}$	99.8%	95.5%	99.4%	84.7%
$\xi = 10^{-3}$	100.0%	95.5%	100.0%	84.8%
$\xi = 10^{-4}$	100.0%	95.5%	100.0%	84.8%
$\xi = 10^{-5}$	100.0%	95.5%	100.0%	84.8%
LIBLINEAR	85.2%	90.7%	100.0%	84.7%

Table 6: Classification Accuracies with Varied ξ from 10^{-2} to 10^{-5}.

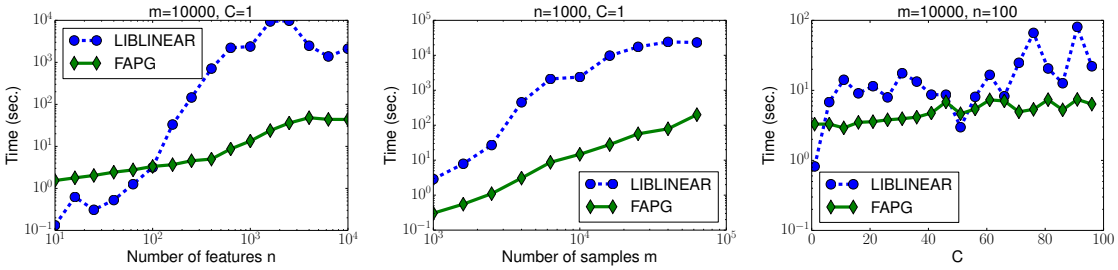

Figure 7: Computation Time for Logistic Regression.

5.3 Logistic Regression

We solved the dual logistic regression (13) via the FAPG method and LIBLINEAR (Fan et al., 2008). LIBLINEAR implements a highly optimized stochastic dual coordinate descent method (Yu et al., 2011) whose subproblems are solved by the Newton method. Note that, as in the case of SVM, LIBLINEAR omits the bias term b from the calculations, i.e., it solves a less complex model than (13). We terminate both algorithms if the violation of the KKT optimality condition is less than $\epsilon = 10^{-6}$. In the FAPG method, (η_u, η_d, δ) were set to $(1.1, 1.1, 0.8)$. L_0 was set to 1. The initial point $\boldsymbol{\alpha}^0$ was set to $\xi \boldsymbol{e}$.

5.3.1 Sensitivity to ξ

Since the gradient ∇f is not defined at $\alpha_i = 0, 1$ $(i \in M)$, we approximately solve the problem by using the constraints $\xi \leq \alpha_i \leq 1 - \xi$ $(i \in M)$, where $\xi > 0$. Table 6 shows the classification accuracy for $\xi = 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}. For all cases, the approximated logistic regression performed better than LIBLINEAR. The classification accuracy of FAPG was identical for $\xi \leq 10^{-3}$, and better than the accuracy with $\xi = 10^{-2}$. We employ $\xi = 10^{-4}$ in the following experiments.

5.3.2 Scalability

First, we compare 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 7. The left panel shows the computation time with respect to the dimension n of the features for $m = 10000$ and $C = 10$. The FAPG method has a clear advantage when the dimension n is high, say $n \geq 10^2$. The middle panel shows the computation time with respect to the number m of the samples for $n = 1000$ and $C = 10$. The computation time of FAPG grew

Function	% Time	Time	# Evals.	Time/Eval.
$\nabla f(\boldsymbol{\alpha})$	85.4%	10.005	1452	6.89.E-03
$f(\boldsymbol{\alpha})$	12.3%	1.443	340	4.24.E-03
$P_{S_{\mathrm{LR}}}(\boldsymbol{\alpha})$	0.5%	0.058	1473	3.94.E-05
Total Runtime:		11.720		

Table 7: Runtime Breakdown of the FAPG Method (sec.)

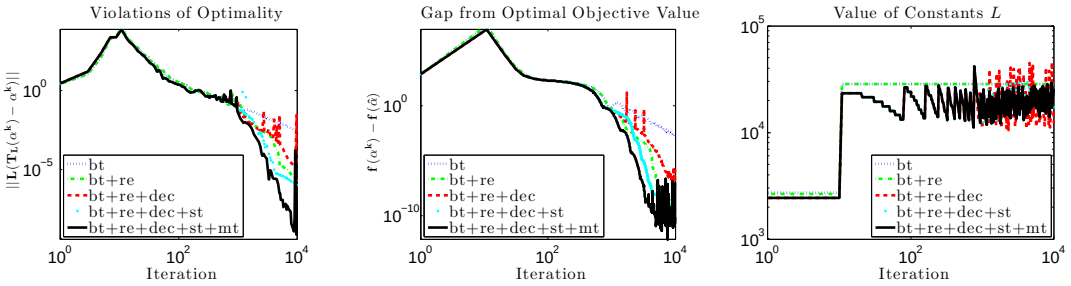

Figure 8: Effect of various acceleration strategies for the FAPG Method.

slower than the one of LIBLINEAR. The right panel illustrates the computation time with respect to the parameter C for $m = 10000$ and $n = 100$. Unlike ν-SVM, the computation time of the FAPG did not change so much depending on the parameter C because it does not affect the area of feasible region. We can observe that the FAPG method is numerically more stable than LIBLINEAR with respect to the changes of the parameter C.

5.3.3 Runtime Breakdown

Table 7 shows the runtime breakdown of the FAPG method for the artificial dataset with $(m, n, C) = (10000, 1000, 10)$. The computation of the gradient $\nabla f(\boldsymbol{\alpha})$ and the function $f(\boldsymbol{\alpha})$ was the most time-consuming parts as in the case of ν-SVM. Thus skipping backtracking step and evaluation of the optimality $\|L_k(\boldsymbol{\alpha}^k - T_{L_k}(\boldsymbol{\alpha}^k))\|$ reduce the computation time significantly because it can avoid the extra computation of f and ∇f.

5.3.4 Effect of Each Acceleration Strategy

Figure 8 shows the running history of the FAPG method with various acceleration strategies for the dataset 'a8a'. C was set to 10.

The left panel depicts the violations of the optimality, i.e. the values of $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$. 're' had effect on decreasing the values after $k = 762$ at which the first restart occur. 'bt+re+dec' just became unstable compared to 'bt+re' because the gradient of the logistic regression is intrinsically large near the boundary of the feasible region S_{LR}. However, 'bt+re+dec+st' could recover from the instability. It is remarkable that only 'bt+re+dec+st+mt' fell below 10^{-6} in 10000 iterations. Thus, one could argue that 'mt' had a significant effect on reducing the violation. The middle panel illustrated the gap in the objective value $f(\boldsymbol{\alpha}^k) - f(\hat{\boldsymbol{\alpha}})$ where we regarded $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{10000}$ of 'bt+re+dec+st+mt' as an optimal solution. 'bt+re+dec+st+mt' decreased the function value faster than others

	m	n	C	FAPG (iter)	LIBLIN
a8a	22696	123	10	19.82 (4939)	9.82
a9a	32561	123	10	<u>35.32</u> (6155)	57.62
australian	690	14	10	0.45 (1738)	0.01
breast-cancer	683	10	10	0.49 (1904)	0.004
cod-rna	59535	8	10	31.40 (3718)	2.03
colon-cancer	62	2000	10	<u>0.11</u> (269)	0.16
covtype	581012	54	10	2039.19 (20143)	73.74
diabetes	768	8	10	0.20 (794)	0.01
duke	44	7129	10	<u>0.27</u> (327)	0.31
epsilon	400000	2000	10	27791.25 (10873)	5225.43
fourclass	862	2	10	0.21 (851)	0.001
german.numer	1000	24	10	0.56 (1741)	0.31
gisette	6000	5000	10	20.30 (1289)	0.54
heart	270	13	10	0.15 (817)	0.02
ijcnn1	35000	22	10	14.57 (2478)	0.51
ionosphere	351	34	10	0.39 (1639)	0.31
leu	38	7129	10	<u>0.11</u> (148)	0.19
liver-disorders	345	6	10	0.16 (828)	0.01
madelon	2000	500	10	<u>2.23</u> (910)	1035.26
mushrooms	8124	112	10	2.26 (1224)	0.09
news20.binary	19996	1355191	10	96.24 (1263)	22.15
rcv1-origin	20242	47236	10	14.41 (1293)	11.41
real-sim	72309	20958	10	142.79 (5630)	7.80
skin-nonskin	245057	3	10	247.11 (8373)	3.59
sonar	208	60	10	0.22 (1150)	0.10
splice	1000	60	10	<u>0.66</u> (1689)	6.02
svmguide1	3089	4	10	0.91 (1313)	0.16
svmguide3	1243	22	10	0.49 (1263)	0.29
url	2396130	3231961	10	– (–)	–
w7a	24692	300	10	17.00 (3400)	10.69
w8a	49749	300	10	51.13 (5279)	42.77

Table 8: 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 results indicated by underline are better than LIBLINEAR.

and found the minimum solution while oscillation occurred at the end of iteration. The right panel illustrated the value of constant L_k of the FAPG at each iteration.

5.3.5 Benchmark Results

We measured computation time for the benchmark datasets. The parameter C was set to 10 throughout this experiment. The experimental results are shown in Table 8. LIBLINEAR was efficient because it solves a less complex model (without the bias term) than (13). In some cases, however, FAPG solved (13) faster than LIBLINEAR. The computation time and iteration count of FAPG was nearly proportional to the number of samples m. On the other hand, LIBLINEAR took much longer time for 'madeon' and 'splice' although they are relatively small datasets.

Taking these results and Figure 7 into account, one could argue that the FAPG method exhibits stable convergence empirically.

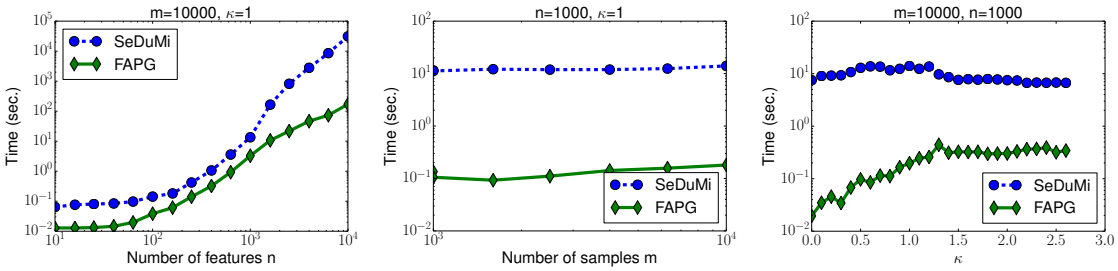

Figure 9: Computation Time for MM-MPM

5.4 MM-MPM

Next, we conducted experiments on MM-MPM (17). To the best of our knowledge, there are no specialized methods for MM-MPM. Thus, we compared the FAPG method only to SeDuMi (Sturm, 1999) which implements an interior point method for the large-scale second-order cone problems such as (17). We terminated the computations if the violation of KKT optimality is less than $\epsilon = 10^{-6}$. In the FAPG method, (η_u, η_d, δ) were set to $(1.1, 1.1, 0.8)$. The value of L_0 was 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 $\boldsymbol{u}^0 = (\boldsymbol{u}_+^0, \boldsymbol{u}_-^0)$ was set to the origin $\mathbf{0}$.

5.4.1 Scalability

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 FAPG 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 $f(\boldsymbol{u})$ and the gradient $\nabla f(\boldsymbol{u})$, are $n \times n$. 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 FAPG method although it is still far more efficient than SeDuMi. The effect of a larger κ on the FAPG method could be because it gives a larger feasible region S_{MPM}, which in turns leads to a larger distance between the initial point $\boldsymbol{u}^0$ and the optimal solution $\boldsymbol{u}^*$ as in the case of ν-SVM (the right panel of Figure 5).

5.4.2 Runtime Breakdown

Table 9 shows the runtime breakdown of the FAPG method for the artificial dataset with $(m, n, \kappa) = (10000, 1000, 1)$. As in the case of ν-SVM (Table 3), the computations of the gradient $\nabla f(\boldsymbol{u})$ and the function value $f(\boldsymbol{u})$ are the most time-consuming parts. Thus, computing 'bt' and $L_k \|T_{L_k}(\boldsymbol{u}^k) - \boldsymbol{u}^k\|$ periodically, which involves the computations of $f(\boldsymbol{u}^k)$ and/or $\nabla f(\boldsymbol{u}^k)$, is effective to reduce the total runtime. The projection $P_{S_{\mathrm{MPM}}}(\boldsymbol{u})$ for MM-MPM shown in Section 4.1.4 can be computed highly efficiently.

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

Table 9: Runtime Breakdown of FAPG Method for MM-MPM (sec.)

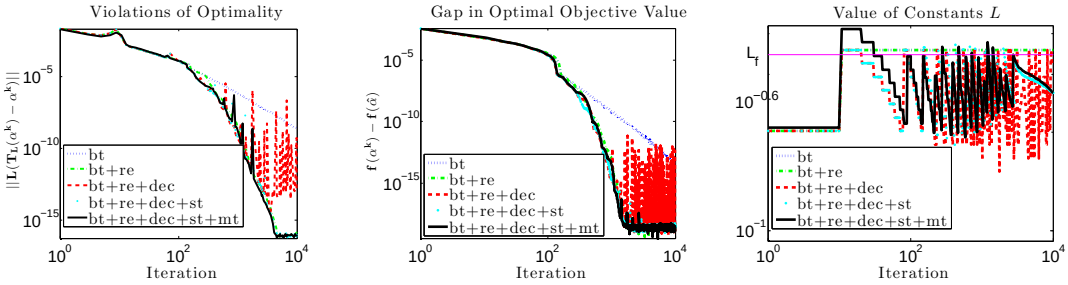

Figure 10: Effect of Each Strategy for the APG Method.

5.4.3 Effect of Each Strategy

Figure 10 illustrates the running history of FAPG with various practical strategies for the artificial dataset with $(m, n, \kappa) = (10000, 1000, 1)$. As in the case of ν-SVM, 're' is effective in reducing the violation of optimality $L_k\|T_{L_k}(\boldsymbol{u}^k)-\boldsymbol{u}^k\|$ and the value of $f(\boldsymbol{u}^k)-f(\hat{\boldsymbol{\alpha}})$. 'dec' seems to make the FAPG method to be unstable, but 'st' can stabilize it. 'bt+re+dec+st' and 'bt+re+dec+st+mt' decreased $L\|T_L(\boldsymbol{u}^k) - \boldsymbol{u}^k\|$ and $f(\boldsymbol{u}^k) - f(\hat{\boldsymbol{\alpha}})$ slightly faster than 'bt+re'.

In the right panel, we can see that the APG method uses values smaller than L_f for L_k in most iterations, where the Lipschitz constant L_f of the gradient $\nabla f(\boldsymbol{u}) = \widetilde{\Sigma}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_- + \widetilde{\Sigma}\boldsymbol{u})$ 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 $\boldsymbol{u} = (\boldsymbol{u}_+, \boldsymbol{u}_-)$).

5.4.4 Benchmark Results

Table 10 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 $n \times n$ matrices $\Sigma_o^{1/2}$ $(o \in \{+, -\})$ are extremely large.

The FAPG method was much faster than SeDuMi especially when the dimension is high, say $n \geq 2000$. Unlike for ν-SVM (Table 5), the FAPG method for MM-MPM sometimes led to larger values of L_k than the Lipschitz constant L_f. However, the average of the values of L_k is still smaller than L_f.

5.5 Classification Ability

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

					Computation Time		Values		
							L(FAPG)		
data	m	n	κ_{max}	κ	SeDuMi	FAPG (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	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	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 10: 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.

dataset	ν-SVM (ν)	Logistic (C)	MM-MPM (κ)	MM-FDA (κ)
a8a	84.4% (0.37)	**84.6%** (0.5)	80.7% (0.94)	84.4% (1.32)
a9a	84.7% (0.36)	**84.8%** (0.1)	80.7% (0.95)	84.7% (1.33)
australian	85.7% (0.83)	**87.7%** (0.1)	86.1% (0.25)	87.5% (1.39)
breast-cancer	97.1% (0.07)	97.4% (0.5)	**97.5%** (2.08)	97.4% (0.31)
cod-rna	**93.9%** (0.28)	**93.9%** (4.5)	93.5% (0.42)	93.7% (0.40)
colon-cancer	**88.8%** (0.29)	87.1% (6.0)	87.1% (0.18)	87.1% (1.22)
covtype	**76.3%** (0.59)	75.6% (1.0)	75.6% (0.65)	75.5% (0.92)
diabetes	**77.3%** (0.54)	**77.3%** (3.5)	74.9% (0.61)	76.8% (0.88)
duke	**88.5%** (0.02)	86.0% (5.5)	**88.5%** (0.98)	**88.5%** (1.36)
epsilon	–	–	89.6% (1.04)	**89.7%** (1.48)
fourclass	77.7% (0.68)	78.5% (0.1)	72.7% (0.57)	**78.6%** (0.10)
german.numer	76.7% (0.54)	**77.4%** (0.1)	71.9% (0.39)	77.3% (0.55)
gisette	97.4% (0.11)	97.4% (1.0)	**97.9%** (3.41)	**97.9%** (4.83)
heart	84.1% (0.40)	**84.4%** (0.1)	84.1% (0.76)	84.1% (0.15)
ijcnn1	74.7% (0.19)	91.8% (20.0)	85.9% (0.89)	**91.0%** (0.64)
ionosphere	88.3% (0.21)	**90.0%** (5.0)	86.9% (0.39)	87.8% (0.85)
leu	**94.2%** (0.02)	**94.2%** (5.0)	**94.2%** (2.10)	**94.2%** (0.21)
liver-disorders	**68.1%** (0.76)	67.9% (6.0)	63.8% (0.20)	66.4% (0.52)
madelon	59.3% (0.91)	57.7% (0.1)	59.7% (0.13)	**60.0%** (0.09)
mushrooms	**100.0%** (0.01)	**100.0%** (6.0)	**100.0%** (9.19)	**100.0%** (6.21)
news20.binary	**97.1%** (0.21)	96.5% (20.0)	**	**
rcv1-origin	97.0% (0.11)	**97.1%** (20.0)	**	**
real-sim	97.5% (0.13)	**97.6%** (15.0)	**	**
skin-nonskin	93.7% (0.32)	92.4% (6.0)	93.5% (1.61)	**93.8%** (2.10)
sonar	**79.8%** (0.40)	78.8% (0.5)	**79.8%** (0.64)	77.9% (0.91)
splice	80.9% (0.50)	80.2% (3.0)	80.6% (0.61)	**81.0%** (0.20)
svmguide1	**95.4%** (0.13)	**95.4%** (20.0)	94.4% (1.12)	91.6% (1.56)
svmguide3	**82.5%** (0.41)	82.1% (15.0)	74.3% (0.55)	81.9% (0.69)
url	–	–	**	**
w7a	**98.5%** (0.04)	**98.5%** (4.0)	96.0% (1.26)	98.2% (1.76)
w8a	98.6% (0.04)	**98.7%** (17.5)	96.1% (1.25)	98.3% (1.73)

Table 11: 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.

cross-validation 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 11. The model that shows the best performance varies with datasets. This implies 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 provides a unified and efficient framework for solving various classification models.

6. Conclusion

In this work, we presented a unified classification model and provided a general algorithm for the model. We designed a fast accelerated proximal gradient (FAPG) method based on the original APG method in (Beck and Teboulle, 2009) to the model 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 provides a practical and useful tool for practitioners who are looking for the best model for a given dataset. Numerical experiments demonstrate the efficiency of our algorithm for large datasets. Indeed, our method often run faster than LIBLINEAR (Fan et al., 2008) especially for large-scale datasets such that $n > 2000$.

In this paper we showed a convergence property of our method for a general non-strongly convex function because our examples are mostly non-strongly convex. As a future work, we would like to see if the linear convergence for FAPG can be established for μ-strongly convex function and provide an adaptive parameter selection scheme even when μ is positive but unknown. In addition, we would like to investigate stochastic variants of our method. Recently, large scale learning has motivated researchers to develop stochastic gradient descent (SGD) methods. SGD approximates the gradient $\nabla\mathcal{L}(\cdot)$ of a loss function by a computationally inexpensive random variable whose expectation coincides with $\nabla\mathcal{L}(\cdot)$. More precisely, it assumes that the loss function is defined as the sum of loss of each sample, i.e., $\mathcal{L}(\widetilde{X}^\top \boldsymbol{w} - \boldsymbol{y}b) = \sum_{i=1}^m \ell(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - b))$, and approximates the *full* gradient $\nabla\mathcal{L}(\widetilde{X}^\top \boldsymbol{w} - \boldsymbol{y}b)$ by the gradient $\nabla\ell(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - b))$ of loss of a randomly chosen sample. PG and APG methods can be adapted in SGD, e.g., as (Nitanda, 2014; Defazio et al., 2014) and references there in. In particular, the stochastic dual coordinate ascent (SDCA) method (Shalev-Shwartz and Zhang, 2013, 2016) is a stochastic APG applied to the dual formulation, which is closely related to our unified classification model. While its framework would be useful to develop a stochastic variant of our FAPG presented in this paper, it cannot be directly applied to the unified classification model because ν-SVM, MM-MPM, and MM-FDA do not satisfy the assumption on $\mathcal{L}(\cdot)$: the loss function is defined as the sum of loss of each sample. It is an important future work to relax the assumption for stochastic methods and extend our FAPG to stochastic ones in order to solve even larger scale problems.

Acknowledgments

We would like to express our warm thanks to the reviewers, whose insightful comments have led to many substantial improvements.

Appendix

Appendix A. Derivation of the Dual Problem (10)

The primal problem (8) can be transformed as follows:

$$\begin{aligned}
&\min_{\boldsymbol{w},b}\{\mathcal{L}(\widetilde{A}^\top \boldsymbol{w} - b\boldsymbol{a}) \mid \|\boldsymbol{w}\|_p \le \lambda\} \\
&= \min_{\boldsymbol{w},b,\boldsymbol{z}}\{\mathcal{L}(\boldsymbol{z}) \mid \|\boldsymbol{w}\|_p \le \lambda,\ \boldsymbol{z} = \widetilde{A}^\top \boldsymbol{w} - b\boldsymbol{a}\} \\
&= \min_{\|\boldsymbol{w}\|_p \le \lambda, b, \boldsymbol{z}} \max_{\boldsymbol{\alpha}}\{\mathcal{L}(\boldsymbol{z}) + \boldsymbol{\alpha}^\top(\boldsymbol{z} - \widetilde{A}^\top \boldsymbol{w} + b\boldsymbol{a})\}.
\end{aligned}$$

Its dual problem (10) is derived as follows:

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} \min_{\|\boldsymbol{w}\|_p \le \lambda, b, \boldsymbol{z}} \{\mathcal{L}(\boldsymbol{z}) + \boldsymbol{\alpha}^\top (\boldsymbol{z} - \widetilde{A}^\top \boldsymbol{w} + b\boldsymbol{a})\} \\
&= -\min_{\boldsymbol{\alpha}} \Big\{ \max_{\boldsymbol{z}} \big\{ -\boldsymbol{\alpha}^\top \boldsymbol{z} - \mathcal{L}(\boldsymbol{z}) \big\} + \max_{\|\boldsymbol{w}\|_p \le \lambda} \big\{ \boldsymbol{\alpha}^\top \widetilde{A}^\top \boldsymbol{w} \big\} + \max_{b} \big\{ \boldsymbol{\alpha}^\top \boldsymbol{a} b \big\} \Big\} \\
&= -\min_{\boldsymbol{\alpha}} \{\mathcal{L}^*(-\boldsymbol{\alpha}) + \lambda \|\widetilde{A}\boldsymbol{\alpha}\|_p^* \mid \boldsymbol{\alpha}^\top \boldsymbol{a} = 0\}.
\end{aligned}$$

Appendix B. Refined Bisection Algorithm

The computational cost of $h(\hat{\theta})$ in Algorithm 2 can be reduced by dividing the indices set M more finely as in (Kiwiel, 2008). Here we divide M into the following four disjoint sets for given θ^l and θ^u satisfying $\theta^l < \theta^u$:

$$\begin{aligned}
U &:= \{i \in M \mid \bar{\alpha}_i - \theta^u \ge u && (\text{i.e., } \alpha_i(\theta) = u, && \forall \theta \in [\theta^l, \theta^u])\} \\
L &:= \{i \in M \mid \bar{\alpha}_i - \theta^l \le l && (\text{i.e., } \alpha_i(\theta) = l, && \forall \theta \in [\theta^l, \theta^u])\} \\
C &:= \{i \in M \mid \bar{\alpha}_i - \theta^l < u,\ \bar{\alpha}_i - \theta^u > l && (\text{i.e., } \alpha_i(\theta) = \bar{\alpha}_i - \theta, && \forall \theta \in [\theta^l, \theta^u])\} \\
I &:= M \backslash (U \cup C \cup L) && ([\theta^l, \theta^u] \text{ contains a breakpoint of } \alpha_i(\theta)) \ .
\end{aligned}$$

If $I = \phi$, then solving

$$\theta = \Big(|U| u + |L| l + \sum_{i \in C} \bar{\alpha}_i - r \Big) / |C|$$

obtains an exact solution. If $I \neq \phi$, then we also divide I into the following five disjoint sets for given $\hat{\theta} \in (\theta^l, \theta^u)$:

$$\begin{aligned}
I^U &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \ge u, && (\text{i.e., } \alpha_i(\theta) = u && \forall \theta \in [\theta^l, \hat{\theta}])\} \\
I^L &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \le l, && (\text{i.e., } \alpha_i(\theta) = l && \forall \theta \in [\hat{\theta}, \theta^u])\} \\
I^{C_u} &= \{i \in I \mid \bar{\alpha}_i - \theta^l < u,\ \bar{\alpha}_i - \hat{\theta} > l, && (\text{i.e., } \alpha_i(\theta) = \bar{\alpha}_i - \theta && \forall \theta \in [\theta^l, \hat{\theta}])\} \\
I^{C_l} &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} < u,\ \bar{\alpha}_i - \theta^u > l, && (\text{i.e., } \alpha_i(\theta) = \bar{\alpha}_i - \theta && \forall \theta \in [\hat{\theta}, \theta^u])\} \\
I^I &= I \backslash (I^U \cup I^{C_u} \cup I^{C_l} \cup I^L)
\end{aligned}$$

Then we have

$$\begin{aligned}
h(\hat{\theta}) = \sum_{i \in M} \alpha_i(\hat{\theta}) = \underbrace{|U| u}_{s_u} + \underbrace{|I^U| u}_{\Delta s_u} + \underbrace{|L| l}_{s_l} + \underbrace{|I^L| l}_{\Delta s_l} \\
+ \underbrace{\sum_{i \in I^C} \alpha_i}_{s_c} + \underbrace{\sum_{i \in I^{C_u}} \alpha_i}_{\Delta s_{c_u}} + \underbrace{\sum_{i \in I^{C_l}} \alpha_i}_{\Delta s_{c_l}} + \sum_{i \in I^I} \alpha_i - |C \cup I^{C_u} \cup I^{C_l} \cup I^I| \hat{\theta}.
\end{aligned}$$

By leveraging on the structure of h, the refined bisection method for (18) can be described as Algorithm 5.

Algorithm 5 Refined Bisection Algorithm for (18)

INPUT: $\bar{\boldsymbol{\alpha}},\ r,\ \boldsymbol{l},\ \boldsymbol{u},\ \epsilon' > 0$ OUTPUT: $\boldsymbol{\alpha}$

INITIALIZE: $I \leftarrow M,\ s_u \leftarrow s_l \leftarrow s_c \leftarrow 0,\ \theta^u \leftarrow \bar{\alpha}_{\max} - \frac{r}{m},\ \theta^l \leftarrow \bar{\alpha}_{\min} - \frac{r}{m}$ # Step 1

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

$\hat{\theta} \leftarrow \frac{\theta^u + \theta^l}{2}$ # Step 2

$\hat{u} \leftarrow u + \hat{\theta},\quad \hat{l} \leftarrow l + \hat{\theta}$

$I^U \leftarrow \{i \in I \mid \bar{\alpha}_i \geq \hat{u}\},\quad I^L \leftarrow \{i \in I \mid \bar{\alpha}_i \leq \hat{l}\}$ # Step 3

$u^u \leftarrow u + \theta^u,\quad l^l \leftarrow l + \theta^l$

$I^{C_u} \leftarrow \{i \in I \mid \bar{\alpha}_i > l^l,\ \bar{\alpha}_i < \hat{u}\},\quad I^{C_l} \leftarrow \{i \in I \mid \bar{\alpha}_i < u^u,\ \bar{\alpha}_i > \hat{l}\}$

$I^I \leftarrow I \backslash (I^U \cup I^{C_u} \cup I^{C_l} \cup I^L)$

$\Delta s_u \leftarrow |I^U| u,\quad \Delta s_l \leftarrow |I^L| l$

$\Delta s_{c_u} \leftarrow \sum_{i \in I^{C_u}} \alpha_i,\quad \Delta s_{c_l} \leftarrow \sum_{i \in I^{C_l}} \alpha_i$

$val \leftarrow s_u + \Delta s_u + s_l + \Delta s_l + s_c + \Delta s_{c_u} + \Delta s_{c_l} + \sum_{i \in I^I} \alpha_i - |C \cup I^{C_u} \cup I^{C_l} \cup I^I| \hat{\theta}$

if $val < r$ **then** # Step 4

$\theta^u \leftarrow \hat{\theta},\quad I \leftarrow I^I \cup I^{C_l} \cup I^L$

$s_u \leftarrow s_u + \Delta s_u,\quad s_c \leftarrow s_c + \Delta s_{c_u}$

else if $val > r$ **then**

$\theta^l \leftarrow \hat{\theta},\quad I \leftarrow I^I \cup I^{C_u} \cup I^U$

$s_l \leftarrow s_l + \Delta s_l,\quad s_c \leftarrow s_c + \Delta s_{c_l}$

else

break

end if

if $I == \phi$ **then**

$\hat{\theta} \leftarrow (s_u + s_l + s_c - r)/|C|$

break

end if

end while

$\alpha_i \leftarrow \alpha_i(\hat{\theta}),\ \forall i \in M$ # Step 5

Appendix C. Proof of Theorem 7

We show the iteration complexity $O\big((W(k)/k)^2\big)$ of Algorithm 3, where $W(z)$ is the inverse function of $f(x) = x\exp(x)$ and called Lambert's W function (Corless et al., 1996). $W(z)$ is single-valued for $z \geq 0$, but multi-valued for $z < 0$. For $z \geq 0$, $W(z)$ is monotonically increasing, but diverges slower than $\log(z)$. It is known that $W(z)$ cannot be expressed in terms of elementary functions.

Here we improve the lower bound (24) of

$$\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\}.$$

Since $\bar{k}_i \geq K_i \geq 2^i$ for all $i \in \{1, 2, \ldots, j\}$, we have

$$\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\} \geq \max\Big\{\frac{k}{j+1}, 2^j\Big\}.$$

$\frac{k}{j+1}$ is monotonically decreasing and 2^j is monotonically increasing with respect to j. Thus the right-hand side is minimized at $j = \hat{j}$ such that

$$\frac{k}{\hat{j}+1} = 2^{\hat{j}}.$$

The above equation implies that

$$\begin{aligned}
&\frac{2k}{\hat{j}+1} = \exp\big((\hat{j}+1)\ln 2\big)\\
&\Rightarrow 2k\ln 2 = (\hat{j}+1)\ln 2\exp\big((\hat{j}+1)\ln 2\big)\\
&\Rightarrow W(2k\ln 2) = (\hat{j}+1)\ln 2\\
&\Rightarrow \hat{j} = \frac{W(2k\ln 2)}{\ln 2} - 1,
\end{aligned}$$

which yields

$$\max\Big\{\frac{k}{j+1}, 2^j\Big\} \geq \frac{k\ln 2}{W(2k\ln 2)}.$$

This leads to

$$F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F(\boldsymbol{\alpha}^*) \leq 2\eta_u L_f R^2\Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2, \quad \forall k \geq 3.$$

Appendix D. Application of FAPG to ℓ_1-regularized models

The ℓ_1-regularized logistic regression:

$$\min_{\boldsymbol{w}} C\sum_{i=1}^m \log(1+\exp(-y_i\boldsymbol{w}^\top\boldsymbol{x}_i)) + \|\boldsymbol{w}\|_1 \tag{27}$$

and the ℓ_1-regularized ℓ_2-loss SVM:

$$\min_{\boldsymbol{w}} C\sum_{i=1}^m \big(\max\{0, 1-y_i\boldsymbol{w}^\top\boldsymbol{x}_i\}\big)^2 + \|\boldsymbol{w}\|_1 \tag{28}$$

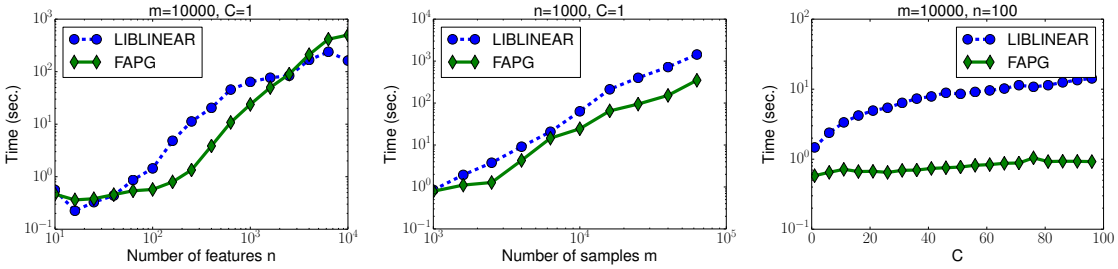

Figure 11: Computation Time for ℓ_1-regularized Logistic Regression (27).

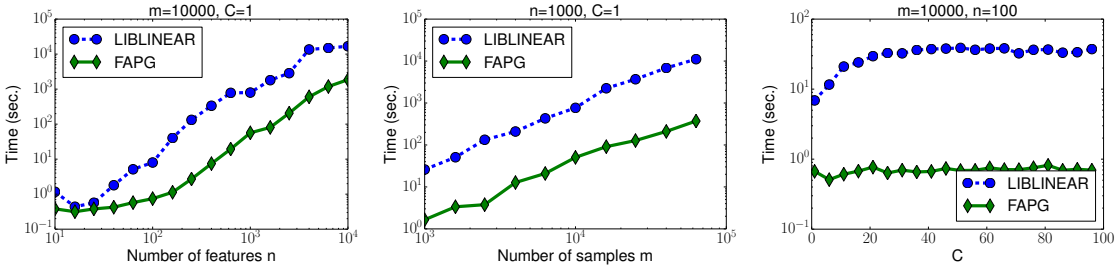

Figure 12: Computation Time for ℓ_1-regularized ℓ_2-loss SVM (28).

have the form of (2), where $C > 0$ is a hyperparameter. Letting $g(\cdot) = \|\cdot\|_1$, we have $\left(\mathrm{prox}_{g,L}(\boldsymbol{w})\right)_i = \mathrm{sign}(w_i)\max\{0, |w_i| - L\}$ $(i = 1, 2, \ldots, m)$ which is known as the soft-thresholding operator.

We applied the practical FAPG (Algorithm 4) and LIBLINEAR (Fan et al., 2008) to (27) and (28) using the artificial datasets which is generated as described in Section 5. We set the initial point $\boldsymbol{w}^0$ to the origin $\mathbf{0}$ and the tolerance ϵ to 10^{-6}. For all datasets, FAPG found solutions with better objective value than LIBLINEAR does. Figures 11 and 12 show the computation time of FAPG and LIBLINEAR. FAPG was highly competitive to and numerically more stable than LIBLINEAR with respect to the changes of the hyperparameter C.

References

U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`. Accessed 6 February 2016.

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.

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

C.-C. Chang and C.-J. 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`.

R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, 1996.

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

D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958.

A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, pages 1646–1654. Curran Associates, Inc., 2014.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279, Helsinki, Finland, 2008.

R.-E. Fan, K.-W. Chang, C.-J Hsieh, X.-R. Wang, and C.-J. 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`.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

J. Gotoh and A. Takeda. A linear classification model based on conditional geometric score. *Pacific Journal of Optimization*, 1(2):277–296, 2005.

I. Guyon, S. Gunn, A. B. Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in Neural Information Processing Systems 17*, pages 545–552, 2005.

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

T.-K. Ho and E. M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 2, pages 880–885. IEEE, 1996.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine learning*, pages 408–415. ACM, 2008.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

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

S. S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

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

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

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Q. Lin and L. Xiao. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. *Computational Optimization and Applications*, 60 (3):633–674, April 2015.

J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious URLs: an application of large-scale online learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 681–688. ACM, 2009.

J. S. Marron, M. J. Todd, and J. Ahn. Distance-weighted discrimination. *Journal of the American Statistical Association*, 102(480):1267–1271, 2007.

A. McCallum. SRAA: simulated/real/aviation/auto UseNet data. URL `http://people.cs.umass.edu/~mccallum/data.html`. Accessed 6 February 2016.

R. D. C. Monteiro, C. Ortiz, and B. F. Svaiter. An adaptive accelerated first-order method for convex optimization. *Computational Optimization and Applications*, 64:31–73, 2016.

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.

Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

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

A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1574–1582. Curran Associates, Inc., 2014.

B. O'Donoghue and E. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15:715–732, 2015.

N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1 (3):127–239, 2014.

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.

D. Prokhorov. IJCNN 2001 neural network competition. *Slide presentation in IJCNN'01*, 2001. URL `http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf`. Accessed 6 February 2016.

R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2(3):21–41, 2000.

K. Scheinberg, D. Goldfarb, and X. Bai. Fast first-order methods for composite convex optimization with backtracking. *Foundations of Computational Mathematics*, 14(3):389–417, Jun 2014.

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.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14:567–599, 2013.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1):105–145, 2016.

S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge, 2008. URL `http://largescale.ml.tu-berlin.de`. Accessed 6 February 2016.

J.F. 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`.

W. Su, S. Boyd, and E. J. Candès. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, Mar 2014.

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

A. V. Uzilov, J. M. Keegan, and D. H. Mathews. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7 (1):173, 2006.

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olson, J. R. Marks, and J. R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences*, 98(20): 11462–11467, 2001.

H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1):41–75, 2011.

T. Zhou, D. Tao, and X. Wu. NESVM: A fast gradient method for support vector machines. In *2010 IEEE 10th International Conference on Data Mining*, pages 679–688, December 2010.