Mach Learn (2015) 98:369–406
DOI 10.1007/s10994-014-5436-1

An efficient primal dual prox method for non-smooth optimization

Tianbao Yang · Mehrdad Mahdavi · Rong Jin · Shenghuo Zhu

Received: 16 November 2012 / Accepted: 17 February 2014 / Published online: 21 March 2014

Abstract We study the non-smooth optimization problems in machine learning, where both the loss function and the regularizer are non-smooth functions. Previous studies on efficient empirical loss minimization assume either a smooth loss function or a strongly convex regularizer, making them unsuitable for non-smooth optimization. We develop a simple yet efficient method for a family of non-smooth optimization problems where the dual form of the loss function is bilinear in primal and dual variables. We cast a non-smooth optimization problem into a minimax optimization problem, and develop a primal dual prox method that solves the minimax optimization problem at a rate of $O(1/T)$ assuming that the proximal step can be efficiently solved, significantly faster than a standard subgradient descent method that has an $O(1/\sqrt{T})$ convergence rate. Our empirical studies verify the efficiency of the proposed method for various non-smooth optimization problems that arise ubiquitously in machine learning by comparing it to the state-of-the-art first order methods.

Keywords Non-smooth optimization · Primal dual method · Convergence rate · Sparsity · Efficiency

Editor: Shai Shalev-Shwartz.

T. Yang (✉) · S. Zhu
NEC Laboratories America, Inc., Cupertino, CA, USA
e-mail: yangtia1@gmail.com

S. Zhu
e-mail: zsh@sv.nec-labs.com

M. Mahdavi · R. Jin
Department of Computer Science and Engineering, Michigan State University,
East Lansing, MI, USA
e-mail: mahdavim@cse.msu.edu

R. Jin
e-mail: rongjin@cse.msu.edu

1 Introduction

Formulating machine learning tasks as a regularized empirical loss minimization problem makes an intimate connection between machine learning and mathematical optimization. In regularized empirical loss minimization, one tries to jointly minimize an empirical loss over training samples plus a regularization term of the model. This formulation includes support vector machine (SVM) (Hastie et al. 2008), support vector regression (Smola and Schölkopf 2004), Lasso (Zhu et al. 2003), logistic regression, and ridge regression (Hastie et al. 2008) among many others. Therefore, optimization methods play a central role in solving machine learning problems and challenges exist in machine learning applications demand the development of new optimization algorithms.

Depending on the application at hand, various types of loss and regularization functions have been introduced in the literature. The efficiency of different optimization algorithms crucially depends on the specific structures of the loss and the regularization functions. Recently, there have been significant interests on gradient descent based methods due to their simplicity and scalability to large datasets. A well-known example is the Pegasos algorithm (Shalev-Shwartz et al. 2011) which minimizes the ℓ_2^2 regularized hinge loss (i.e., SVM) and achieves a convergence rate of $O(1/T)$, where T is the number of iterations, by exploiting the strong convexity of the regularizer. Several other first order algorithms (Ji and Ye 2009; Chen et al. 2009) are also proposed for smooth loss functions (e.g., squared loss and logistic loss) and non-smooth regularizers (i.e., $\ell_{1,\infty}$ and group lasso). They achieve a convergence rate of $O(1/T^2)$ by exploiting the smoothness of the loss functions.

In this paper, we focus on a more challenging case where both the loss function and the regularizer are non-smooth, to which we refer as non-smooth optimization. Non-smooth optimization of regularized empirical loss has found applications in many machine learning problems. Examples of non-smooth loss functions include hinge loss (Vapnik 1998), generalized hinge loss (Bartlett and Wegkamp 2008), absolute loss (Hastie et al. 2008), and ϵ-insensitive loss (Rosasco et al. 2004); examples of non-smooth regularizers include lasso (Zhu et al. 2003), group lasso (Yuan and Lin 2006), sparse group lasso (Yang et al. 2010), exclusive lasso (Zhou et al. 2010), $\ell_{1,\infty}$ regularizer (Quattoni et al. 2009), and trace norm regularizer (Rennie and Srebro 2005).

Although there are already many existing studies on tackling smooth loss functions (e.g., square loss for regression, logistic loss for classification), or smooth regularizers (e.g., ℓ_2^2 norm), there are serious challenges in developing efficient algorithms for non-smooth optimization. In particular, common tricks, such as smoothing non-smooth objective functions (Nesterov 2005a, b), can not be applied to non-smooth optimization to improve convergence rate. This is because they require both the loss functions and regularizers be written in the maximization form of bilinear functions, which unfortunately are often violated, as we will discuss later. In this work, we focus on optimization problems in machine learning where both the loss function and the regularizer are non-smooth. Our goal is to develop an efficient gradient based algorithm that has a convergence rate of $O(1/T)$ for a wide family of non-smooth loss functions and general non-smooth regularizers.

It is noticeable that according to the information based complexity theory (Traub et al. 1988), it is impossible to derive an efficient first order algorithm that generally works for all non-smooth objective functions. As a result, we focus on a family of non-smooth optimization problems, where the dual form of the non-smooth loss function is bilinear in both primal and dual variables. Additionally, we show that many non-smooth loss functions have this bilinear dual form. We derive an efficient gradient based method, with a convergence rate of $O(1/T)$, that explicitly updates both the primal and dual variables. The proposed method is referred to

as **Primal Dual Prox (Pdprox)** method. Besides its capability of dealing with non-smooth optimization, the proposed method is effective in handling the learning problems where additional constraints are introduced for dual variables.

The rest of this paper is organized as follows. Section 2 reviews the related work on minimizing regularized empirical loss especially the first order methods for large-scale optimization. Section 3 presents some notations and definitions. Section 4 presents the proposed primal dual prox method, its convergence analysis, and several extensions of the proposed method. Section 5 presents the empirical studies, and Sect. 6 concludes this work.

2 Related work

Our work is closely related to the previous studies on regularized empirical loss minimization. In the following discussion, we mostly focus on non-smooth loss functions and non-smooth regularizers.

2.1 Non-smooth loss functions

Hinge loss is probably the most commonly used non-smooth loss function for classification. It is closely related to the max-margin criterion. A number of algorithms have been proposed to minimize the ℓ_2^2 regularized hinge loss (Platt 1998; Joachims 1999, 2006; Hsieh et al. 2008; Shalev-Shwartz et al. 2011), and the ℓ_1 regularized hinge loss (Cai et al. 2010; Zhu et al. 2003; Fung and Mangasarian 2002). Besides the hinge loss, recently a generalized hinge loss function (Bartlett and Wegkamp 2008) has been proposed for cost sensitive learning. For regression, square loss is commonly used due to its smoothness. However, non-smooth loss functions such as absolute loss (Hastie et al. 2008) and ϵ-insensitive loss (Rosasco et al. 2004) are useful for robust regression. The Bayes optimal predictor of square loss is the mean of the predictive distribution, while the Bayes optimal predictor of absolute loss is the median of the predictive distribution. Therefore absolute loss is more robust for long-tailed error distributions and outliers (Hastie et al. 2008). Rosasco et al. (2004) also proved that the estimation error bound for absolute loss and ϵ-insensitive loss converges faster than that of square loss. Non-smooth piecewise linear loss function has been used in quantile regression (Koenker 2005; Gneiting 2008). Unlike the absolute loss, the piecewise linear loss function can model non-symmetric error in reality.

2.2 Non-smooth regularizers

Besides the simple non-smooth regularizers such as ℓ_1, ℓ_2, and ℓ_∞ norms (Duchi and Singer 2009), many other non-smooth regularizers have been employed in machine learning tasks. Yuan and Lin (2006) introduced group lasso for selecting important explanatory factors in group manner. The $\ell_{1,\infty}$ norm regularizer has been used for multi-task learning (Argyriou et al. 2008). In addition, several recent works (Hou et al. 2011; Nie et al. 2010; Liu et al. 2009) considered mixed $\ell_{2,1}$ regularizer for feature selection. Zhou et al. (2010) introduced exclusive lasso for multi-task feature selection to model the scenario where variables within a single group compete with each other. Trace norm regularizer is another non-smooth regularizer, which has found applications in matrix completion (Recht et al. 2010; Candès and Recht 2008), matrix factorization (Rennie and Srebro 2005; Srebro et al. 2005), and multi-task learning (Argyriou et al. 2008; Ji and Ye 2009). The optimization algorithms presented in these works are usually limited: either the convergence rate is not guaranteed (Argyriou

et al. 2008; Recht et al. 2010; Hou et al. 2011; Nie et al. 2010; Rennie and Srebro 2005; Srebro et al. 2005) or the loss functions are assumed to be smooth (e.g., the square loss or the logistic loss) (Liu et al. 2009; Ji and Ye 2009). Despite the significant efforts in developing algorithms for minimizing regularized empirical losses, it remains a challenge to design a first order algorithm that is able to efficiently solve non-smooth optimization problems at a rate of $O(1/T)$ when both the loss function and the regularizer are non-smooth.

2.3 Gradient based optimization

Our work is closely related to (sub)gradient based optimization methods. The convergence rate of gradient based methods usually depends on the properties of the objective function to be optimized. When the objective function is strongly convex and smooth, it is well known that gradient descent methods can achieve a geometric convergence rate (Boyd and Vandenberghe 2004). When the objective function is smooth but not strongly convex, the optimal convergence rate of a gradient descent method is $O(1/T^2)$, and is achieved by the Nesterov's methods (Nesterov 2007). For the objective function which is strongly convex but not smooth, the convergence rate becomes $O(1/T)$ (Shalev-Shwartz et al. 2011). For general non-smooth objective functions, the optimal rate of any first order method is $O(1/\sqrt{T})$. Although it is not improvable in general, recent studies are able to improve this rate to $O(1/T)$ by exploring the special structure of the objective function (Nesterov 2005a, b). In addition, several methods are developed for composite optimization, where the objective function is written as a sum of a smooth and a non-smooth function (Lan 2010; Nesterov 2007; Lin 2010). Recently, these optimization techniques have been successfully applied to various machine learning problems, such as SVM (Zhou et al. 2010), general regularized empirical loss minimization (Duchi and Singer 2009; Hu et al. 2009), trace norm minimization (Ji and Ye 2009), and multi-task sparse learning (Chen et al. 2009). Despite these efforts, one major limitation of the existing (sub)gradient based algorithms is that in order to achieve a convergence rate better than $O(1/\sqrt{T})$, they have to assume that the loss function is smooth or the regularizer is strongly convex, making them unsuitable for non-smooth optimization.

2.4 Convex–concave optimization

The present work is also related to convex–concave minimization. Tseng (2008) and Nemirovski (2005) developed prox methods that have a convergence rate of $O(1/T)$, provided the gradients are Lipschitz continuous and have been applied to machine learning problems (Sun et al. 2009). In contrast, our method achieves a rate of $O(1/T)$ without requiring the whole gradient but part of the gradient to be Lipschitz continuous. Several other primal-dual algorithms have been developed for regularized empirical loss minimization that update both primal and dual variables. Zhu and Chan (2008) proposed a primal-dual method based on gradient descent, which only achieves a rate of $O(1/\sqrt{T})$. It was generalized in Esser et al. (2010), which shares the similar spirit of the proposed algorithm. However, the explicit convergence rate was not established even though the convergence is proved. Mosci et al. (2010) presented a primal-dual algorithm for group sparse regularization, which updates the primal variable by a prox method and the dual variable by a Newton's method. In contrast, the proposed algorithm is a first order method that does not require computing the Hessian matrix as the Newton's method does, and is therefore more scalable to large datasets. Combettes and Pesquet (2011) and Radu loan Bot Ernö Robert Csetnek (2012) proposed primal-dual splitting algorithms for finding zeros of maximal monotone operators of special types. Lan et al. (2011) considered the primal-dual convex formulations for general cone programming

and apply Nesterov's optimal first order method (Nesterov 2007), Nesterov's smoothing technique (Nesterov 2005b), and Nemirovski's prox method (Nemirovski 2005). Nesterov (2005a) proposed a primal dual gradient method for a special class of structured non-smooth optimization problems by exploring an excessive gap technique.

2.5 Optimizing non-smooth functions

We note that Nesterov's smoothing technique (Nesterov 2005b) and excessive gap technique (Nesterov 2005a) can be applied to non-smooth optimization and both achieve $O(1/T)$ convergence rate for a special class of non-smooth optimization problems. However, the limitation of these approaches is that they require all the non-smooth terms (i.e., the loss and the regularizer) to be written as an explicit max structure that consists of a bilinear function in primal and dual variables, thus limits their applications to many machine learning problems. In addition, Nesterov's algorithms need to solve additional maximizations problem at each iteration. In contrast, the proposed algorithm only requires mild condition on the non-smooth loss functions (Sect. 4), and allows for any commonly used non-smooth regularizers, without having to solve an additional optimization problem at each iteration. Compared to Nesterov's algorithms, the proposed algorithm is applicable to a large class of non-smooth optimization problems, is easier to implement, its convergence analysis is much simpler, and its empirical performance is usually comparably favorable. Finally we noticed that, as we are preparing our manuscript, a related work (Chambolle and Pock 2011) has recently been published in the Journal of Mathematical Imaging and Vision that shares a similar idea as this work. Both works maintain and update the primal and dual variables for solving a non-smooth optimization problem, and achieve the same convergence rate (i.e., $O(1/T)$). However, our work distinguishes from Chambolle and Pock (2011) in the following aspects: (i) We propose and analyze two primal dual prox methods: one gives an extra gradient updating to dual variables and the other gives an extra gradient updating to primal variables. Depending on the nature of applications, one method may be more efficient than the others; (ii) In Sect. 4.6, we discuss how to efficiently solve the interim projection problems for updating both primal variable and dual variable, a critical issue for making the proposed algorithm practically efficient. In contrast, Chambolle and Pock (2011) simply assumes that the interim projection problems can be solved efficiently; (iii) We focus our analysis and empirical studies on the optimization problems that are closely related to machine learning. We demonstrate the effectiveness of the proposed algorithm on various classification, regression, and matrix completion tasks with non-smooth loss functions and non-smooth regularizers; (iv) We also conduct analysis and experiments on the convergence of the proposed methods when dealing with the ℓ_1 constraint on the dual variable, an approach that is commonly used in robust optimization, and observe that the proposed methods converge much faster when the bound of the ℓ_1 constraint is small and the obtained solution is more robust in terms of prediction in the presence of noise in labels. In contrast, the study (Chambolle and Pock 2011) only considers the application in image problems.

We also note that the proposed algorithm is closely related to proximal point algorithm (Rockafellar 1976) as shown in He and Yuan (2012), and many variants including the modified Arrow–Hurwicz method (Popov 1980), the Douglas–Rachford (DR) splitting algorithm (Lions and Mercier 1979), the alternating method of multipliers (ADMM) (Boyd et al. 2011), the forward–backward splitting algorithm (Bredies 2009), the FISTA algorithm (Beck and Teboulle 2009). For a detailed comparison with some of these algorithms, one can refer to Chambolle and Pock (2011).

3 Notations and definitions

In this section we provide the basic setup, some preliminary definitions and notations used throughout this paper.

We denote by $[n]$ the set of integers $\{1, \cdots, n\}$. We denote by $(\mathbf{x}_i, y_i), i \in [n]$ the training examples, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and y_i is the assigned class label, which is discrete for classification and continuous for regression. We assume $\|\mathbf{x}_i\|_2 \leq R, \ \forall i \in [n]$. We denote by $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)^\top$ and $\mathbf{y} = (y_1, \cdots, y_n)^\top$. Let $\mathbf{w} \in \mathbb{R}^d$ denote the linear hypothesis, $\ell(\mathbf{w}; \mathbf{x}, y)$ denote a loss of prediction made by the hypothesis $\mathbf{w}$ on example $(\mathbf{x}, y)$, which is a convex function in terms of $\mathbf{w}$. Examples of convex loss function are hinge loss $\ell(\mathbf{w}; \mathbf{x}, y) = \max(1 - y\mathbf{w}^\top\mathbf{x}, 0)$, and absolute loss $\ell(\mathbf{w}; \mathbf{x}, y) = |\mathbf{w}^\top\mathbf{x} - y|$. To characterize a function, we introduce the following definitions

Definition 1 A function $\ell(\mathbf{z}) : \mathcal{Z} \to \mathbb{R}$ is a G-Lipschitz continuous if

$$|\ell(\mathbf{z}_1) - \ell(\mathbf{z}_2)| \leq G\|\mathbf{z}_1 - \mathbf{z}_2\|_2, \forall \mathbf{z}_1, \mathbf{z}_2 \in \mathcal{Z}.$$

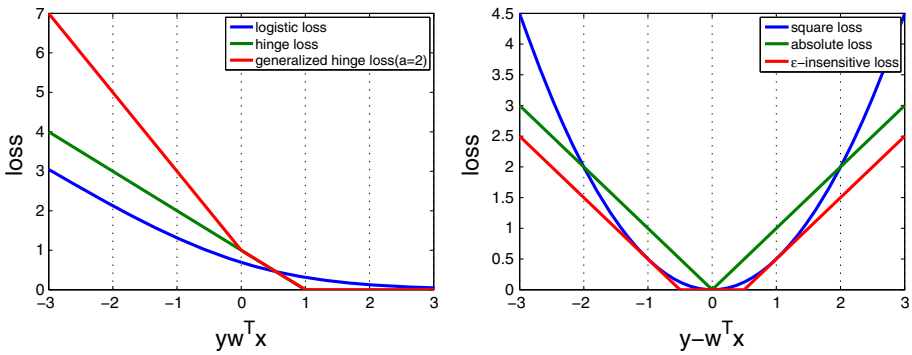
Definition 2 A function $\ell(\mathbf{z}) : \mathcal{Z} \to \mathbb{R}$ is a ρ-smooth function if its gradient is ρ-Lipschitz continuous

$$\|\nabla\ell(\mathbf{z}_1) - \nabla\ell(\mathbf{z}_2)\|_2 \leq \rho\|\mathbf{z}_1 - \mathbf{z}_2\|_2, \forall \mathbf{z}_1, \mathbf{z}_2 \in \mathcal{Z}.$$

A function is non-smooth if either its gradient is not well defined or its gradient is not Lipschtiz continuous. Examples of smooth loss functions are logistic loss $\ell(\mathbf{w}; \mathbf{x}, y) = \log(1 + \exp(-y\mathbf{w}^\top\mathbf{x}))$, square loss $\ell(\mathbf{w}; \mathbf{x}, y) = \frac{1}{2}(\mathbf{w}^\top\mathbf{x} - y)^2$, and examples of non-smooth loss functions are hinge loss, and absolute loss. The difference between logistic loss and hinge loss, square loss and absolute loss can be seen in Fig. 1. Examples of non-smooth regularizer include $R(\mathbf{w}) = \|\mathbf{w}\|_1$, i.e. ℓ_1 norm, $R(\mathbf{w}) = \|\mathbf{w}\|_\infty$, i.e. ℓ_∞ norm. More examples can be found in Sect. 4.1.

In this paper, we aim to solve the following optimization problem, which occurs in many machine learning problems,

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}; \mathbf{x}_i, y_i) + \lambda R(\mathbf{w}), \tag{1}$$



(a) classification **(b)** regression

Fig. 1 Loss functions **a** classification **b** regression

where $\ell(\mathbf{w}; \mathbf{x}, y)$ is a non-smooth loss function, $R(\mathbf{w})$ is a non-smooth regularizer on $\mathbf{w}$, and λ is a regularization parameter.

We denote by $\Pi_{\mathcal{Q}}[\widehat{\mathbf{z}}] = \arg\min_{\mathbf{z}\in\mathcal{Q}} \frac{1}{2}\|\mathbf{z} - \widehat{\mathbf{z}}\|_2^2$ the projection of $\widehat{\mathbf{z}}$ into domain $\mathcal{Q}$, and by $\Pi_{\mathcal{Q}_1,\mathcal{Q}_2}\begin{pmatrix}\widehat{\mathbf{z}}_1\\ \widehat{\mathbf{z}}_2\end{pmatrix}$ the joint projection of $\widehat{\mathbf{z}}_1$ and $\widehat{\mathbf{z}}_2$ into domains $\mathcal{Q}_1$ and $\mathcal{Q}_2$, respectively. Finally, we use $[s]_{[0,a]}$ to denote the projection of s into $[0, a]$, where $a > 0$.

4 Pdprox: a primal dual prox method for non-smooth optimization

We first describe the non-smooth optimization problems that the proposed algorithm can be applied to, and then present the primal dual prox method for non-smooth optimization. We then prove the convergence rate of the proposed algorithms and discuss several extensions. Proofs for technical lemmas are deferred to the appendix.

4.1 Non-smooth optimization

We first focus our analysis on linear classifiers and denote by $\mathbf{w} \in \mathbb{R}^d$ a linear model. The extension to nonlinear models is discussed in Sect. 4.7. Also, extension to a collection of linear models $\mathbf{W} \in \mathbb{R}^{d\times K}$ can be done in a straightforward way. We consider the following general *non-smooth* optimization problem:

$$\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} \left[\mathcal{L}(\mathbf{w}) = \max_{\boldsymbol{\alpha}\in\mathcal{Q}_{\boldsymbol{\alpha}}} L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) + \lambda R(\mathbf{w})\right]. \tag{2}$$

The parameters $\mathbf{w}$ in domain $\mathcal{Q}_{\mathbf{w}}$ and $\boldsymbol{\alpha}$ in domain $\mathcal{Q}_{\boldsymbol{\alpha}}$ are referred to as primal and dual variables, respectively. Since it is impossible to develop an efficient first order method for general non-smooth optimization, we focus on the family of non-smooth loss functions that can be characterized by bilinear function $L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y})$, i.e.

$$L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) = c_0(\mathbf{X}, \mathbf{y}) + \boldsymbol{\alpha}^\top \mathbf{a}(\mathbf{X}, \mathbf{y}) + \mathbf{w}^\top \mathbf{b}(\mathbf{X}, \mathbf{y}) + \mathbf{w}^\top \mathbf{H}(\mathbf{X}, \mathbf{y})\boldsymbol{\alpha}, \tag{3}$$

where $c_0(\mathbf{X}, \mathbf{y})$, $\mathbf{a}(\mathbf{X}, \mathbf{y})$, $\mathbf{b}(\mathbf{X}, \mathbf{y})$, and $\mathbf{H}(\mathbf{X}, \mathbf{y})$ are the parameters depending on the training examples $(\mathbf{X}, \mathbf{y})$ with consistent sizes. In the sequel, we denote by $L(\mathbf{w}, \boldsymbol{\alpha}) = L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y})$ for simplicity, and by $G_{\mathbf{w}}(\mathbf{w}, \boldsymbol{\alpha}) = \nabla_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha})$ and $G_{\alpha}(\mathbf{w}, \boldsymbol{\alpha}) = \nabla_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha})$ the partial gradients of $L(\mathbf{w}, \boldsymbol{\alpha})$ in terms of $\mathbf{w}$ and $\boldsymbol{\alpha}$, respectively.

Remark 1 One direct consequence of assumption in (3) is that the partial gradient $G_{\mathbf{w}}(\mathbf{w}, \boldsymbol{\alpha})$ is independent of $\mathbf{w}$, and $G_{\boldsymbol{\alpha}}(\mathbf{w}, \boldsymbol{\alpha})$ is independent of $\boldsymbol{\alpha}$, since $L(\mathbf{w}, \boldsymbol{\alpha})$ is bilinear in $\mathbf{w}$ and $\boldsymbol{\alpha}$. We will explicitly exploit this property in developing the efficient optimization algorithms. We also note that no explicit assumption is made for the regularizer $R(\mathbf{w})$. This is in contrast to the smoothing techniques used in Nesterov (2005a, b).

To efficiently solve the optimization problem in (1), we need first turn it into the form (2). To this end, we assume that the loss function can be written into a dual form, which is bilinear in the primal and the dual variables, i.e.

$$\ell(\mathbf{w}; \mathbf{x}_i, y_i) = \max_{\alpha_i\in\Delta_\alpha} f(\mathbf{w}, \alpha_i; \mathbf{x}_i, y_i), \tag{4}$$

where $f(\mathbf{w}, \alpha; \mathbf{x}, y)$ is a bilinear function in $\mathbf{w}$ and α, and Δ_α is the domain of variable α. Using (4), we cast problem (1) into (2) with $L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y})$ given by

$$L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}, \alpha_i; \mathbf{x}_i, y_i), \tag{5}$$

with $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^\top$ defined in the domain $\mathcal{Q}_{\boldsymbol{\alpha}} = \{\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^\top, \alpha_i \in \Delta_\alpha\}$.

Before delving into the description of the proposed algorithms and their analysis, we give a few examples that show many non-smooth loss functions can be written in the form of (4):

– Hinge loss (Vapnik 1998):

$$\ell(\mathbf{w}; \mathbf{x}, y) = \max(0, 1 - y\mathbf{w}^\top\mathbf{x}) = \max_{\alpha\in[0,1]} \alpha(1 - y\mathbf{w}^\top\mathbf{x}).$$

– Generalized hinge loss (Bartlett and Wegkamp 2008):

$$\begin{aligned}\ell(\mathbf{w}; \mathbf{x}, y) &= \begin{cases} 1 - ay\mathbf{w}^\top\mathbf{x} & \text{if } y\mathbf{w}^\top\mathbf{x} \le 0 \\ 1 - y\mathbf{w}^\top\mathbf{x} & \text{if } 0 < y\mathbf{w}^\top\mathbf{x} < 1 \\ 0 & \text{if } y\mathbf{w}^\top\mathbf{x} \ge 1 \end{cases} \\ &= \max_{\substack{\alpha_1\ge 0, \alpha_2\ge 0 \\ \alpha_1+\alpha_2\le 1}} \alpha_1(1 - ay\mathbf{w}^\top\mathbf{x}) + \alpha_2(1 - y\mathbf{w}^\top\mathbf{x}),\end{aligned}$$

where $a > 1$.

– Absolute loss (Hastie et al. 2008):

$$\ell(\mathbf{w}; \mathbf{x}, y) = |\mathbf{w}^\top\mathbf{x} - y| = \max_{\alpha\in[-1,1]} \alpha(\mathbf{w}^\top\mathbf{x} - y).$$

– ϵ-insensitive loss (Rosasco et al. 2004):

$$\ell(\mathbf{w}; \mathbf{x}, y) = \max(|\mathbf{w}^\top\mathbf{x} - y| - \epsilon, 0) = \max_{\substack{\alpha_1\ge 0, \alpha_2\ge 0 \\ \alpha_1+\alpha_2\le 1}} \Big[(\mathbf{w}^\top\mathbf{x} - y)(\alpha_1 - \alpha_2) - \epsilon(\alpha_1 + \alpha_2)\Big].$$

– Piecewise linear loss (Koenker 2005):

$$\begin{aligned}\ell(\mathbf{w}; \mathbf{x}, y) &= \begin{cases} a|\mathbf{w}^\top\mathbf{x} - y| & \text{if } \mathbf{w}^\top\mathbf{x} \le y \\ (1 - a)|\mathbf{w}^\top\mathbf{x} - y| & \text{if } \mathbf{w}^\top\mathbf{x} \ge y \end{cases} \\ &= \max_{\substack{\alpha_1\ge 0, \alpha_2\ge 0 \\ \alpha_1+\alpha_2\le 1}} \alpha_1 a(y - \mathbf{w}^\top\mathbf{x}) + \alpha_2(1 - a)(\mathbf{w}^\top\mathbf{x} - y).\end{aligned}$$

– ℓ_2 loss (Nie et al. 2010):

$$\ell(\mathbf{W}; \mathbf{x}, \mathbf{y}) = \|\mathbf{W}^\top\mathbf{x} - \mathbf{y}\|_2 = \max_{\|\alpha\|_2\le 1} \alpha^\top(\mathbf{W}^\top\mathbf{x} - \mathbf{y}),$$

where $\mathbf{y} \in \mathbb{R}^K$ is multiple class label vector and $\mathbf{W} = (\mathbf{w}_1, \cdots, \mathbf{w}_K)$.

Besides the non-smooth loss function $\ell(\mathbf{w}; \mathbf{x}, y)$, we also assume that the regularizer $R(\mathbf{w})$ is a non-smooth function. Many non-smooth regularizers are used in machine learning problems. We list a few of them in the following, where $\mathbf{W} = (\mathbf{w}_1, \cdots, \mathbf{w}_K)$, $\mathbf{w}_k \in \mathbb{R}^d$ and $\mathbf{w}^j$ is the jth row of $\mathbf{W}$.

– lasso: $R(\mathbf{w}) = \|\mathbf{w}\|_1$, ℓ_2 norm: $R(\mathbf{w}) = \|\mathbf{w}\|_2$, and ℓ_∞ norm: $R(\mathbf{w}) = \|\mathbf{w}\|_\infty$.
– group lasso: $R(\mathbf{w}) = \sum_{g=1}^{K} \sqrt{d_g}\|\mathbf{w}_g\|_2$, where $\mathbf{w}_g \in \mathbb{R}^{d_g}$.
– exclusive lasso: $R(\mathbf{W}) = \sum_{j=1}^{d} \|\mathbf{w}^j\|_1^2$.

- $\ell_{2,1}$ norm: $R(\mathbf{W}) = \sum_{j=1}^{d} \|\mathbf{w}^j\|_2$.
- $\ell_{1,\infty}$ norm: $R(\mathbf{W}) = \sum_{j=1}^{d} \|\mathbf{w}^j\|_\infty$.
- trace norm: $R(\mathbf{W}) = \|\mathbf{W}\|_1$, the summation of singular values of $\mathbf{W}$.
- other regularizers: $R(\mathbf{W}) = \left(\sum_{k=1}^{K} \|\mathbf{w}_k\|_2\right)^2$.

Note that unlike Nesterov (2005a, b), we do not further require the non-smooth regularizer to be written into a bilinear dual form, which could be violated by many non-smooth regularizers, e.g. $R(\mathbf{W}) = \left(\sum_{k=1}^{K} \|\mathbf{w}_k\|_2\right)^2$ or more generally $R(\mathbf{w}) = V(\|\mathbf{w}\|)$, where $V(z)$ is a monotonically increasing function.

We close this section by presenting a lemma showing an important property of the bilinear function $L(\mathbf{w}, \boldsymbol{\alpha})$.

Lemma 1 *Let $L(\mathbf{w}, \boldsymbol{\alpha})$ be bilinear in $\mathbf{w}$ and $\boldsymbol{\alpha}$ as in (3). Given fixed $\mathbf{X}$, $\mathbf{y}$ there exists $c > 0$ such that $\|H(\mathbf{X}, \mathbf{y})\|_2^2 \leq c$, then for any $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{Q}_{\boldsymbol{\alpha}}$, and $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{Q}_{\mathbf{w}}$ we have*

$$\|G_\alpha(\mathbf{w}_1, \boldsymbol{\alpha}_1) - G_\alpha(\mathbf{w}_2, \boldsymbol{\alpha}_2)\|_2^2 \leq c\|\mathbf{w}_1 - \mathbf{w}_2\|_2^2, \tag{6}$$

$$\|G_{\mathbf{w}}(\mathbf{w}_1, \boldsymbol{\alpha}_1) - G_{\mathbf{w}}(\mathbf{w}_2, \boldsymbol{\alpha}_2)\|_2^2 \leq c\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2^2. \tag{7}$$

Remark 2 The value of constant c in Lemma 1 is an input to our algorithms used to set the step size. In Appendix 1, we show how to estimate constant c for certain loss functions. In addition the constant c in bounds (6) and (7) do not have to be the same as shown by the the example of generalized hinge loss in Appendix 1. It should be noticed that the inequalities in Lemma 1 indicate $L(\mathbf{w}, \boldsymbol{\alpha})$ has Liptschitz continuos gradients, however, the gradient of the whole objective with respect to $\mathbf{w}$, i.e., $G_{\mathbf{w}}(\mathbf{w}, \boldsymbol{\alpha}) + \lambda \partial R(\mathbf{w})$ is not Lipschitz continuous due to the general non-smooth term $R(\mathbf{w})$, which prevents previous convex-concave minimization scheme (Tseng 2008; Nemirovski 2005) not applicable.

4.2 The proposed primal-dual prox methods

In this subsection, we present two variants of Primal Dual Prox (Pdprox) method for solving the non-smooth optimization problem in (2). The common feature shared by the two algorithms is that they update both the primal and the dual variables at each iteration. In contrast, most first order methods only update the primal variables. The key advantages of the proposed algorithms is that they are able to capture the **sparsity structures of both primal and dual variables**, which is usually the case when both the regularizer and the loss functions are both non-smooth. The two algorithms differ from each other in the number of copies for the dual or the primal variables, and the specific order for updating those. Although our analysis shows that the two algorithms share the same convergence rate; however, our empirical studies show that the one algorithm is more preferable than the other depending on the nature of the applications.

4.3 Pdprox-dual algorithm

Algorithm 1 shows the first primal dual prox algorithm for optimizing the problem in (2). Compared to the other gradient based algorithms, Algorithm 1 has several interesting features:

(i) it updates both the dual variable $\boldsymbol{\alpha}$ and the primal variable $\mathbf{w}$. This is useful when additional constraints are introduced for the dual variables, as we will discuss later.
(ii) it introduces an extra dual variable $\boldsymbol{\beta}$ in addition to $\boldsymbol{\alpha}$, and updates both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ at each iteration by a gradient mapping. The gradient mapping on the dual variables into a sparse

Algorithm 1 The Pdprox-dual Algorithm for Non-Smooth Optimization

1: **Input**: step size $\gamma = \sqrt{1/(2c)}$, where c is specified in (6).
2: **Initialization**: $\mathbf{w}_0 = \mathbf{0}$, $\boldsymbol{\beta}_0 = \mathbf{0}$
3: **for** $t = 1, 2, \ldots$ **do**
4: $\quad \boldsymbol{\alpha}_t = \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}}\left[\boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_{t-1}, \boldsymbol{\beta}_{t-1})\right]$
5: $\quad \mathbf{w}_t = \arg\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} \frac{1}{2}\left\|\mathbf{w} - \left(\mathbf{w}_{t-1} - \gamma G_{\mathbf{w}}(\mathbf{w}_{t-1}, \boldsymbol{\alpha}_t)\right)\right\|_2^2 + \gamma\lambda R(\mathbf{w})$
6: $\quad \boldsymbol{\beta}_t = \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}}\left[\boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t)\right]$
7: **end for**
8: **Output** $\widehat{\mathbf{w}}_T = \sum_{t=1}^T \mathbf{w}_t/T$ and $\widehat{\boldsymbol{\alpha}}_T = \sum_{t=1}^T \boldsymbol{\alpha}_t/T$.

Algorithm 2 The Pdprox-primal Algorithm for Non-Smooth Optimization

1: **Input**: step size $\gamma = \sqrt{1/(2c)}$, where c is specified in (7).
2: **Initialization**: $\mathbf{u}_0 = \mathbf{0}$, $\boldsymbol{\alpha}_0 = \mathbf{0}$
3: **for** $t = 1, 2, \ldots$ **do**
4: $\quad \mathbf{w}_t = \arg\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} \frac{1}{2}\left\|\mathbf{w} - \left(\mathbf{u}_{t-1} - \gamma G_{\mathbf{w}}(\mathbf{u}_{t-1}, \boldsymbol{\alpha}_{t-1})\right)\right\|_2^2 + \gamma\lambda R(\mathbf{w})$
5: $\quad \boldsymbol{\alpha}_t = \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}}\left[\boldsymbol{\alpha}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_{t-1})\right]$
6: $\quad \mathbf{u}_t = \mathbf{w}_t + \gamma(G_{\mathbf{w}}(\mathbf{u}_{t-1}, \boldsymbol{\alpha}_{t-1}) - G_{\mathbf{w}}(\mathbf{w}_t, \boldsymbol{\alpha}_t))$
7: **end for**
8: **Output** $\widehat{\mathbf{w}}_T = \sum_{t=1}^T \mathbf{w}_t/T$ and $\widehat{\boldsymbol{\alpha}}_T = \sum_{t=1}^T \boldsymbol{\alpha}_t/T$.

domain allows the proposed algorithm to capture the sparsity of the dual variables (more discussion on how the sparse constraint on the dual variable affects the convergence is presented in Sect. 4.7). Compared to the second algorithm presented below, we refer to Algorithm 1 as **Pdprox-dual** algorithm since it introduces an extra dual variable in updating.

(iii) the primal variable $\mathbf{w}$ is updated by a composite gradient mapping (Nesterov 2007) in step 5. Solving a composite gradient mapping in this step allows the proposed algorithm to capture the sparsity of the primal variable. Similar to many other approaches for composite optimization (Duchi and Singer 2009; Hu et al. 2009), we assume that the mapping in step 5 can be solved efficiently. (This is the only assumption we made on the non-smooth regularizer. The discussion in Sect. 4.6 shows that the proposed algorithm can be applied to a large family of non-smooth regularizers).

(iv) the step size γ is fixed to $\sqrt{1/(2c)}$, where c is the constant specified in Lemma 1. This is in contrast to most gradient based methods where the step size depends on T and/or λ. This feature is particularly useful in implementation as we often observe that the performance of a gradient method is sensitive to the choice of the step size.

4.4 Pdprox-primal algorithm

In Algorithm 1, we maintain two copies of the dual variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and update them by two gradient mappings[1]. We can actually save one gradient mapping on the dual variable by first updating the primal variable $\mathbf{w}_t$, and then updating $\boldsymbol{\alpha}_t$ using partial gradient computed with $\mathbf{w}_t$. As a tradeoff, we add an extra primal variable $\mathbf{u}$, and update it by a simple calculation. The detailed steps are shown in Algorithm 2. Similar to Algorithm 1, Algorithm 2 also needs to compute two partial gradients (except for the initial partial gradient on the primal variable), i.e., $G_{\mathbf{w}}(\cdot, \boldsymbol{\alpha}_t)$ and $G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \cdot)$. Different from Algorithm 1, Algorithm 2 (i) maintains $(\mathbf{w}_t, \boldsymbol{\alpha}_t, \mathbf{u}_t)$ at each iteration with $O(2d + n)$ memory, while Algorithm 1 maintains

[1] The extra gradient mapping on $\boldsymbol{\beta}$ can also be replaced with a simple calculation, as discussed in Sect. 4.6.

$(\boldsymbol{\alpha}_t, \mathbf{w}_t, \boldsymbol{\beta}_t)$ at each iteration with $O(2n+d)$ memory; (ii) and replaces one gradient mapping on an extra dual variable $\boldsymbol{\beta}_t$ with a simple update on an extra primal variable $\mathbf{u}_t$. Depending on the nature of applications, one method may be more efficient than the other. For example, if the dimension d is much larger than the number of examples n, then Algorithm 1 would be more preferable than Algorithm 2. When the number of examples n is much larger than the dimension d, Algorithm 2 could save the memory and the computational cost. However, as shown by our analysis in Sect. 4.5, the convergence rate of two algorithms are the same. Because it introduces an extra primal variable, we refer to Algorithm 2 as the **Pdprox-primal** algorithm.

Remark 1 It should be noted that although Algorithm 1 uses a similar strategy for updating the dual variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, but it is significantly different from the mirror prox method (Nemirovski 2005). First, unlike the mirror prox method that introduces an auxiliary variable for $\mathbf{w}$, Algorithm 1 introduces a composite gradient mapping for updating $\mathbf{w}$. Second, Algorithm 1 updates $\mathbf{w}_t$ using the partial gradient computed from the updated dual variable $\boldsymbol{\alpha}_t$ rather than $\boldsymbol{\beta}_{t-1}$. Third, Algorithm 1 does not assume that the overall objective function has Lipschitz continuous gradients, a key assumption that limits the application of the mirror prox method.

Remark 2 A similar algorithm with an extra primal variable is also proposed in a recent work (Chambolle and Pock 2011). It is slightly different from Algorithm 2 in the order of updating on the primal variable and the dual variable, and the gradients used in the updating. We discuss the differences between the Pdprox method and the algorithm in Chambolle and Pock (2011) with our notations in Appendix 3.

4.5 Convergence analysis

This section establishes bounds on the convergence rate of the proposed algorithms. We begin by presenting a theorem about the convergence rate of Algorithms 1 and 2. For ease of analysis, we first write (2) into the following equivalent minimax formulation

$$\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} \max_{\boldsymbol{\alpha}\in\mathcal{Q}_{\boldsymbol{\alpha}}} F(\mathbf{w}, \boldsymbol{\alpha}) = L(\mathbf{w}, \boldsymbol{\alpha}) + \lambda R(\mathbf{w}). \tag{8}$$

Our main result is stated in the following theorem.

Theorem 1 *By running Algorithm 1 or Algorithm 2 with T steps, we have*

$$F(\widehat{\mathbf{w}}_T, \boldsymbol{\alpha}) - F(\mathbf{w}, \widehat{\boldsymbol{\alpha}}_T) \leq \frac{\|\mathbf{w}\|_2^2 + \|\boldsymbol{\alpha}\|_2^2}{\sqrt{(2/c)T}},$$

for any $\mathbf{w} \in \mathcal{Q}_{\mathbf{w}}$ and $\boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}}$. In particular,

$$\mathcal{L}(\widehat{\mathbf{w}}_T) - \mathcal{D}(\widehat{\boldsymbol{\alpha}}_T) \leq \frac{\|\widetilde{\mathbf{w}}_T\|_2^2 + \|\widetilde{\boldsymbol{\alpha}}_T\|_2^2}{\sqrt{(2/c)T}}$$

where $\mathcal{D}(\boldsymbol{\alpha}) = \min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} F(\mathbf{w}, \boldsymbol{\alpha})$ is the dual objective, and $\widetilde{\mathbf{w}}_T, \widetilde{\boldsymbol{\alpha}}_T$ are given by $\widetilde{\mathbf{w}}_T = \arg\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} F(\mathbf{w}, \widehat{\boldsymbol{\alpha}}_T)$, $\widetilde{\boldsymbol{\alpha}}_T = \arg\max_{\boldsymbol{\alpha}\in\mathcal{Q}_{\alpha}} F(\widehat{\mathbf{w}}_T, \boldsymbol{\alpha})$.

Remark 3 It is worth mentioning that in contrast to most previous studies whose convergence rates are derived for the optimality of either the primal objective or the dual objective, the convergence result in Theorem 1 is on the duality gap, which can serve a certificate of the

convergence for the proposed algorithm. It is not difficult to show that when $\mathcal{Q}_{\mathbf{w}} = \mathbb{R}^d$ the dual objective can be computed by

$$\mathcal{D}(\boldsymbol{\alpha}) = c_0(\mathbf{X}, \mathbf{y}) + \boldsymbol{\alpha}^\top \mathbf{a}(\mathbf{X}, \mathbf{y}) - \lambda R^* \left(\frac{-\mathbf{b}(\mathbf{X}, \mathbf{y}) - H(\mathbf{X}, \mathbf{y})\boldsymbol{\alpha}}{\lambda} \right)$$

where $R^*(\mathbf{u})$ is the convex conjugate of $R(\mathbf{w})$. For example, if $R(\mathbf{w}) = 1/2\|\mathbf{w}\|_2^2$, $R^*(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|_2^2$; if $R(\mathbf{w}) = \|\mathbf{w}\|_p$, $R^*(\mathbf{u}) = I(\|\mathbf{u}\|_q \leq 1)$, where $I(\cdot)$ is an indicator function, $p = 1, 2, \infty$ and $1/p + 1/q = 1$.

Before proceeding to the proof of Theorem 1, we present the following Corollary that follows immediately from Theorem 1 and states the convergence bound for the objective $\mathcal{L}(\mathbf{w})$ in (2).

Corollary 1 *Let $\mathbf{w}^*$ be the optimal solution to (2), bounded by $\|\mathbf{w}^*\|_2^2 \leq D_1$, and $\|\boldsymbol{\alpha}\|_2^2 \leq D_2, \forall \boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}}$. By running Algorithm 1 or 2 with T iterations, we have*

$$\mathcal{L}(\widehat{\mathbf{w}}_T) - \mathcal{L}(\mathbf{w}^*) \leq \frac{D_1 + D_2}{\sqrt{(2/c)T}}.$$

Proof Let $\mathbf{w} = \mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathcal{Q}_{\mathbf{w}}} \mathcal{L}(\mathbf{w})$ and $\widetilde{\boldsymbol{\alpha}}_T = \arg\max_{\boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}}} F(\widehat{\mathbf{w}}_T, \boldsymbol{\alpha})$ in Theorem 1, then we have

$$\max_{\boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}}} F(\widehat{\mathbf{w}}_T, \boldsymbol{\alpha}) - F(\mathbf{w}^*, \widehat{\boldsymbol{\alpha}}_T) \leq \frac{\|\mathbf{w}^*\|_2^2 + \|\widetilde{\boldsymbol{\alpha}}_T\|_2^2}{\sqrt{(2/c)T}},$$

Since $\mathcal{L}(\mathbf{w}) = \max_{\boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}}} F(\mathbf{w}, \boldsymbol{\alpha}) \geq F(\mathbf{w}, \widehat{\boldsymbol{\alpha}}_T)$, then we have

$$\mathcal{L}(\widehat{\mathbf{w}}_T) - \mathcal{L}(\mathbf{w}^*) \leq \frac{D_1 + D_2}{\sqrt{(2/c)T}}.$$

□

In order to aid understanding, we present the proof of Theorem 1 for each algorithm separately in the following subsections.

4.5.1 Convergence analysis of Algorithm 1

For the simplicity of analysis, we assume $\mathcal{Q}_{\mathbf{w}} = \mathbb{R}^d$ is the whole Euclidean space. We discuss how to generalize the analysis to a convex domain $Q_{\mathbf{w}}$ in Sect. 4.7. In order to prove Theorem 1 for Algorithm 1, we present a series of lemmas to pave the path for the proof. We first restate the key updates in Algorithm 1 as follows:

$$\boldsymbol{\alpha}_t = \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}} \left[\boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_{t-1}, \boldsymbol{\beta}_{t-1}) \right], \tag{9}$$

$$\mathbf{w}_t = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{2}\|\mathbf{w} - (\mathbf{w}_{t-1} - \gamma G_{\mathbf{w}}(\mathbf{w}_{t-1}, \boldsymbol{\alpha}_t))\|_2^2 + \gamma\lambda R(\mathbf{w}), \tag{10}$$

$$\boldsymbol{\beta}_t = \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}} \left[\boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) \right]. \tag{11}$$

Lemma 2 *The updates in Algorithm 1 are equivalent to the following gradient mappings,*

$$\begin{pmatrix} \boldsymbol{\alpha}_t \\ \mathbf{w}_t \end{pmatrix} = \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}, \mathbb{R}^d} \begin{pmatrix} \boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{u}_{t-1}, \boldsymbol{\beta}_{t-1}) \\ \mathbf{u}_{t-1} - \gamma (G_{\mathbf{w}}(\mathbf{u}_{t-1}, \boldsymbol{\alpha}_t) + \lambda \mathbf{v}_t) \end{pmatrix},$$

and

$$\begin{pmatrix}\boldsymbol{\beta}_t\\ \mathbf{u}_t\end{pmatrix} = \Pi_{\mathcal{Q}_\alpha,\mathbb{R}^d}\begin{pmatrix}\boldsymbol{\beta}_{t-1}+\gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)\\ \mathbf{u}_{t-1}-\gamma(G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)+\lambda\mathbf{v}_t)\end{pmatrix},$$

with initialization $\mathbf{u}_0 = \mathbf{w}_0$*, where* $\mathbf{v}_t \in \partial R(\mathbf{w}_t)$ *is a partial gradient of the regularizer on* $\mathbf{w}_t$.

Proof First, we argue that there exists a fixed (sub)gradient $\mathbf{v}_t \in \partial R(\mathbf{w}_t)$ such that the composite gradient mapping (10) is equivalent to the following gradient mapping,

$$\mathbf{w}_t = \Pi_{\mathbb{R}^d}\left[\mathbf{w}_{t-1} - \gamma\left(G_{\mathbf{w}}(\mathbf{w}_{t-1},\boldsymbol{\alpha}_t)+\lambda\mathbf{v}_t\right)\right]. \tag{12}$$

To see this, since $\mathbf{w}_t$ is the optimal solution to (10), by first order optimality condition, there exists a subgradient $\mathbf{v}_t = \partial R(\mathbf{w}_t)$ such that $\mathbf{w}_t - \mathbf{w}_{t-1} + \gamma G_{\mathbf{w}}(\mathbf{w}_{t-1},\boldsymbol{\alpha}_t) + \gamma\lambda\mathbf{v}_t = \mathbf{0}$, i.e.

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \gamma G_{\mathbf{w}}(\mathbf{w}_{t-1},\boldsymbol{\alpha}_t) - \gamma\lambda\mathbf{v}_t,$$

which is equivalent to (12) since the projection $\Pi_{\mathbb{R}^d}$ is an identical mapping.

Second, the updates in Algorithm 1 for $(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{w})$ are equivalent to the following updates for $(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{w},\mathbf{u})$

$$\begin{aligned}
\boldsymbol{\alpha}_t &= \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}}\left[\boldsymbol{\beta}_{t-1}+\gamma G_{\boldsymbol{\alpha}}(\mathbf{u}_{t-1},\boldsymbol{\beta}_{t-1})\right],\\
\mathbf{w}_t &= \Pi_{\mathbb{R}^d}\left[\mathbf{u}_{t-1} - \gamma\left(G_{\mathbf{w}}(\mathbf{u}_{t-1},\boldsymbol{\alpha}_t)+\lambda\mathbf{v}_t\right)\right], \qquad (13)\\
\boldsymbol{\beta}_t &= \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}}\left[\boldsymbol{\beta}_{t-1}+\gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)\right],\\
\mathbf{u}_t &= \mathbf{w}_t - \gamma(G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t) - G_{\mathbf{w}}(\mathbf{u}_{t-1},\boldsymbol{\alpha}_t)), \qquad (14)
\end{aligned}$$

with initialization $\mathbf{u}_0 = \mathbf{w}_0$. The reason is because $\mathbf{u}_t = \mathbf{w}_t$, $t = 1,\cdots$ due to $G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t) = G_{\mathbf{w}}(\mathbf{u}_{t-1},\boldsymbol{\alpha}_t)$, where we use the fact that $L(\mathbf{w},\boldsymbol{\alpha})$ is linear in $\mathbf{w}$.

Finally, by plugging (13) for $\mathbf{w}_t$ into the update for $\mathbf{u}_t$ in (14), we complete the proof of Lemma 2. □

The reason that we translate the updates for $(\boldsymbol{\alpha}_t,\mathbf{w}_t,\boldsymbol{\beta}_t)$ in Algorithm 1 into the updates for $(\boldsymbol{\alpha}_t,\mathbf{w}_t,\boldsymbol{\beta}_t,\mathbf{u}_t)$ in Lemma 2 is because it allows us to fit the updates for $(\boldsymbol{\alpha}_t,\mathbf{w}_t,\boldsymbol{\beta}_t,\mathbf{u}_t)$ into Lemma 8 as presented in Appendix 4, which leads us to a key inequality as stated in Lemma 3 to prove Theorem 1.

Lemma 3 *For all* $t = 1, 2, \cdots$*, and any* $\mathbf{w}\in\mathbb{R}^d, \boldsymbol{\alpha}\in\mathcal{Q}_{\boldsymbol{\alpha}}$*, we have*

$$\begin{aligned}
&\gamma\begin{pmatrix}G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)+\lambda\mathbf{v}_t\\ -G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)\end{pmatrix}^\top\begin{pmatrix}\mathbf{w}_t-\mathbf{w}\\ \boldsymbol{\alpha}_t-\boldsymbol{\alpha}\end{pmatrix} \le \frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2 - \frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2\\
&\quad+\gamma^2\left\|G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t) - G_{\boldsymbol{\alpha}}(\mathbf{u}_{t-1},\boldsymbol{\beta}_{t-1})\right\|_2^2 - \frac{1}{2}\|\mathbf{w}_t-\mathbf{u}_{t-1}\|_2^2.
\end{aligned}$$

The proof of Lemma 3 is deferred to Appendix 4. We are now ready to prove the main theorem for Algorithm 1.

Proof (of Theorem 1 for Algorithm 1) Since $F(\mathbf{w},\boldsymbol{\alpha})$ is convex in $\mathbf{w}$ and concave in $\boldsymbol{\alpha}$, we have

$$\begin{aligned}
F(\mathbf{w}_t,\boldsymbol{\alpha}_t) - F(\mathbf{w},\boldsymbol{\alpha}_t) &\le (G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)+\lambda\mathbf{v}_t)^\top(\mathbf{w}_t-\mathbf{w}),\\
F(\mathbf{w}_t,\boldsymbol{\alpha}) - F(\mathbf{w}_t,\boldsymbol{\alpha}_t) &\le -G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)^\top(\boldsymbol{\alpha}_t-\boldsymbol{\alpha}),
\end{aligned}$$

where $\mathbf{v}_t \in \partial R(\mathbf{w}_t)$ is the partial gradient of $R(\mathbf{w})$ on $\mathbf{w}_t$ stated in Lemma 2. Combining the above inequalities with Lemma 3, we have

$$\begin{aligned}
&\gamma\left(F(\mathbf{w}_t,\boldsymbol{\alpha}_t)-F(\mathbf{w},\boldsymbol{\alpha}_t)+F(\mathbf{w}_t,\boldsymbol{\alpha})-F(\mathbf{w}_t,\boldsymbol{\alpha}_t)\right)\\
&\quad\le \frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2-\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2+\gamma^2\|G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)-G_{\boldsymbol{\alpha}}(\mathbf{u}_{t-1},\boldsymbol{\beta}_{t-1})\|_2^2\\
&\qquad-\frac{1}{2}\|\mathbf{w}_t-\mathbf{u}_{t-1}\|_2^2\\
&\quad\le \frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2-\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2+\gamma^2 c\|\mathbf{w}_t-\mathbf{u}_{t-1}\|_2^2-\frac{1}{2}\|\mathbf{w}_t-\mathbf{u}_{t-1}\|_2^2\\
&\quad\le \frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2-\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2,
\end{aligned}$$

where the second inequality follows the inequality (6) in Lemma 1 and the fact $\gamma=\sqrt{1/(2c)}$. By adding the inequalities of all iterations and dividing both sides by T, we have

$$\frac{1}{T}\sum_{t=1}^{T}\left(F(\mathbf{w}_t,\boldsymbol{\alpha})-F(\mathbf{w},\boldsymbol{\alpha}_t)\right)\le\frac{\|\mathbf{w}\|_2^2+\|\boldsymbol{\alpha}\|_2^2}{\sqrt{(2/c)}\,T}. \tag{15}$$

We complete the proof by using the definitions of $\widehat{\mathbf{w}}_T, \widehat{\boldsymbol{\alpha}}_T$, and the convexity–concavity of $F(\mathbf{w},\boldsymbol{\alpha})$ with respect to $\mathbf{w}$ and $\boldsymbol{\alpha}$, respectively. □

4.5.2 Convergence analysis of Algorithm 2

We can prove the convergence bound for Algorithm 2 by following the same path. In the following we present the key lemmas similar to Lemmas 2 and 3, with proofs omitted.

Lemma 4 *There exists a fixed partial gradient $\mathbf{v}_t \in \partial R(\mathbf{w}_t)$ such that the updates in Algorithm 2 are equivalent to the following gradient mappings,*

$$\begin{pmatrix}\mathbf{w}_t\\ \boldsymbol{\alpha}_t\end{pmatrix}=\Pi_{\mathbb{R}^d,\mathcal{Q}_{\boldsymbol{\alpha}}}\begin{pmatrix}\mathbf{u}_{t-1}-\gamma\left(G_{\mathbf{w}}(\mathbf{u}_{t-1},\boldsymbol{\beta}_{t-1})+\lambda\mathbf{v}_t\right)\\ \boldsymbol{\beta}_{t-1}+\gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\beta}_{t-1})\end{pmatrix}$$

and

$$\begin{pmatrix}\mathbf{u}_t\\ \boldsymbol{\beta}_t\end{pmatrix}=\Pi_{\mathbb{R}^d,\mathcal{Q}_{\boldsymbol{\alpha}}}\begin{pmatrix}\mathbf{u}_{t-1}-\gamma\left(G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)+\lambda\mathbf{v}_t\right)\\ \boldsymbol{\beta}_{t-1}+\gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)\end{pmatrix},$$

with initialization $\boldsymbol{\beta}_0=\boldsymbol{\alpha}_0$.

Lemma 5 *For all $t=1,2,\cdots$, and any $\mathbf{w}\in\mathbb{R}^d, \boldsymbol{\alpha}\in\mathcal{Q}_\alpha$, we have*

$$\begin{aligned}
\gamma\begin{pmatrix}G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)+\lambda\mathbf{v}_t\\ -G_{\boldsymbol{\alpha}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)\end{pmatrix}^{\top}\begin{pmatrix}\mathbf{w}_t-\mathbf{w}\\ \boldsymbol{\alpha}_t-\boldsymbol{\alpha}\end{pmatrix}&\le\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2-\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2\\
&+\gamma^2\left\|G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)-G_{\mathbf{w}}(\mathbf{u}_{t-1},\boldsymbol{\beta}_{t-1})\right\|_2^2-\frac{1}{2}\|\boldsymbol{\alpha}_t-\boldsymbol{\beta}_{t-1}\|_2^2.
\end{aligned}$$

Proof (*of Theorem* 1 *for Algorithm 2*) Similar to proof of Theorem 1 for Algorithm 1, we have

$$\begin{aligned}
&\gamma\left(F(\mathbf{w}_t,\boldsymbol{\alpha}_t)-F(\mathbf{w},\boldsymbol{\alpha}_t)+F(\mathbf{w}_t,\boldsymbol{\alpha})-F(\mathbf{w}_t,\boldsymbol{\alpha}_t)\right)\\
&\quad\le\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2-\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2+\gamma^2\left\|G_{\mathbf{w}}(\mathbf{w}_t,\boldsymbol{\alpha}_t)-G_{\mathbf{w}}(\mathbf{u}_{t-1},\boldsymbol{\beta}_{t-1})\right\|_2^2\\
&\qquad-\frac{1}{2}\|\boldsymbol{\alpha}_t-\boldsymbol{\beta}_{t-1}\|_2^2\\
&\quad\le\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2-\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2+\gamma^2c\|\boldsymbol{\alpha}_t-\boldsymbol{\beta}_{t-1}\|_2^2-\frac{1}{2}\|\boldsymbol{\alpha}_t-\boldsymbol{\beta}_{t-1}\|_2^2\\
&\quad\le\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_{t-1}\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_{t-1}\end{pmatrix}\right\|_2^2-\frac{1}{2}\left\|\begin{pmatrix}\mathbf{w}-\mathbf{u}_t\\ \boldsymbol{\alpha}-\boldsymbol{\beta}_t\end{pmatrix}\right\|_2^2,
\end{aligned}$$

where the last step follows the inequality (7) in Lemma 1 and the fact $\gamma=\sqrt{1/(2c)}$. By adding the inequalities of all iterations and dividing both sides by T, we have

$$\frac{1}{T}\sum_{t=1}^{T}\left(F(\mathbf{w}_t,\boldsymbol{\alpha})-F(\mathbf{w},\boldsymbol{\alpha}_t)\right)\le\frac{\|\mathbf{w}\|_2^2+\|\boldsymbol{\alpha}\|_2^2}{\sqrt{(2/c)}\,T}. \tag{16}$$

We complete the proof by using the definitions of $\widehat{\mathbf{w}}_T,\widehat{\boldsymbol{\alpha}}_T$, and the convexity–concavity of $F(\mathbf{w},\boldsymbol{\alpha})$ with respect to $\mathbf{w}$ and $\boldsymbol{\alpha}$, respectively. □

Comparison with Pegasos on ℓ_2^2 regularizer We compare the proposed algorithm to the Pegasos algorithm (Shalev-Shwartz et al. 2011)[2] for minimizing the ℓ_2^2 regularized hinge loss. Although in this case both algorithms achieve a convergence rate of $O(1/T)$, their dependence on the regularization parameter λ is very different. In particular, the convergence rate of the proposed algorithm is $O\left(\frac{(1+n\lambda)R}{\sqrt{2n}\lambda T}\right)$ by noting that $\|\mathbf{w}^*\|_2^2=O(1/\lambda)$, $\|\boldsymbol{\alpha}^*\|_2^2\le\|\boldsymbol{\alpha}^*\|_1\le n$, and $c=R^2/n$, while the Pegasos algorithm has a convergence rate of $\widetilde{O}\left(\frac{(\sqrt{\lambda}+R)^2}{\lambda T}\right)$ (Corollary 1 in Shalev-Shwartz et al. 2011), where $\widetilde{O}(\cdot)$ suppresses a logarithmic term $\ln(T)$. According to the common assumption of learning theory (Wu and Zhou 2005; Smale and Zhou 2003), the optimal λ is $O(n^{-1/(\tau+1)})$ if the probability measure can be approximated by the closure of RKHS $\mathcal{H}_\kappa$ with exponent $0<\tau\le 1$. As a result, the convergence rate of the proposed algorithm is $O(\sqrt{n}R/T)$ while the convergence rate of Pegasos is $O(n^{1/(1+\tau)}R^2/T)$. Since $\tau\in(0,1]$, the proposed algorithm could be more efficient than the Pegasos algorithm, particularly when λ is sufficiently small. This is verified by our empirical studies in Sect. 5.7 (see Fig. 8). It is also interesting to note that the convergence rate of Pdprox has a better dependence on R, the ℓ_2 norm bound of examples $\|\mathbf{x}\|_2\le R$, compared to R^2 in the convergence rate of Pegasos. Finally, we mention that the proposed algorithm is a deterministic algorithm that requires a full pass of all training examples at each iteration, while Pegasos can be purely stochastic by sampling one example for computing the sub-gradient, which maintains the same convergence rate. It remains an interesting and open problem to extend the Pdprox algorithm to its stochastic or randomized version with a similar convergence rate.

[2] We compare to the deterministic Pegasos that computes the gradient using all examples at each iteration. It would be criticized that it is not fair to compare with Pegasos since it is a stochastic algorithm, however, such a comparison (both theoretically and empirically) would provide a formal evidence that solving the min–max problem by a primal dual method with an extra-gradient may yield better convergence than solving the primal problem.

4.6 Implementation issues

In this subsection, we discuss some implementation issues: (1) how to efficiently solve the optimization problems for updating the primal and dual variables in Algorithms 1 and 2; (2) how to set a good step size; and (3) how to implement the algorithms efficiently.

Both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are updated by a gradient mapping that requires computing the projection into the domain $\mathcal{Q}_{\boldsymbol{\alpha}}$. When $\mathcal{Q}_{\boldsymbol{\alpha}}$ is only consisted of box constraints (e.g., hinge loss, absolute loss, and ϵ-insensitive loss), the projection $\prod_{\mathcal{Q}_{\boldsymbol{\alpha}}}[\widehat{\alpha}]$ can be computed by thresholding. When $\mathcal{Q}_{\boldsymbol{\alpha}}$ is comprised of both box constraints and a linear constraint (e.g., generalized hinge loss), the following lemma gives an efficient algorithm for computing $\prod_{\mathcal{Q}_{\boldsymbol{\alpha}}}[\widehat{\boldsymbol{\alpha}}]$.

Lemma 6 *For $\mathcal{Q}_{\boldsymbol{\alpha}} = \{\boldsymbol{\alpha} : \boldsymbol{\alpha} \in [0, s]^n, \boldsymbol{\alpha}^\top \mathbf{v} \leq \rho\}$, the optimal solution $\boldsymbol{\alpha}^*$ to projection $\prod_{\mathcal{Q}_{\boldsymbol{\alpha}}}[\widehat{\boldsymbol{\alpha}}]$ is computed by*

$$\alpha_i^* = [\widehat{\alpha}_i - \eta v_i]_{[0,s]}, \forall i \in [n],$$

where $\eta = 0$ if $\sum_i [\widehat{\alpha}_i]_{[0,s]} v_i \leq \rho$ and otherwise is the solution to the following equation

$$\sum_i [\widehat{\alpha}_i - \eta v_i]_{[0,s]} v_i - \rho = 0. \tag{17}$$

Since $\sum_i [\widehat{\alpha}_i - \eta v_i]_{[0,s]} v_i - 1$ is monotonically decreasing in η, we can solve η in (17) by a bi-section search.

Remark 4 It is notable that when the domain is a simplex type domain, i.e. $\sum_i \alpha_i \leq \rho$, Duchi et al. (2008) has proposed more efficient algorithms for solving the projection problem.

Moreover, we can further improve the efficiency of Algorithm 1 by removing the gradient mapping on $\boldsymbol{\beta}$. The key idea is similar to the analysis provided in Sect. 4.7 for arguing that the convergence rate presented in Theorem 1 for Algorithm 2 holds for any convex domain $\mathcal{Q}_{\mathbf{w}}$. Actually, the update on $\boldsymbol{\alpha}$ is equivalent to

$$\boldsymbol{\alpha}_t = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2} \|\boldsymbol{\alpha} - \left(\boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}\left(\mathbf{w}_{t-1}, \boldsymbol{\beta}_{t-1}\right)\right)\|_2^2 + \gamma Q(\boldsymbol{\alpha}),$$

which together with the first order optimality condition implies

$$\boldsymbol{\alpha}_t = \boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_{t-1}, \boldsymbol{\beta}_{t-1}) - \gamma \partial Q(\boldsymbol{\alpha}_t),$$

where

$$Q(\boldsymbol{\alpha}) = \begin{cases} 0 & \boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}} \\ +\infty & \text{otherwise} \end{cases},$$

is the indicator function of the domain $\mathcal{Q}_{\boldsymbol{\alpha}}$. Then we can update the β_t by

$$\begin{aligned} \boldsymbol{\beta}_t &= \arg\min_{\boldsymbol{\alpha}} \frac{1}{2} \|\boldsymbol{\alpha} - \left(\boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) - \partial Q(\boldsymbol{\alpha}_t)\right)\|_2^2, \\ &= \boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) - \partial Q(\boldsymbol{\alpha}_t) \end{aligned}$$

which can be computed simply by

$$\boldsymbol{\beta}_t = \boldsymbol{\alpha}_t + \gamma \left(G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) - G_{\boldsymbol{\alpha}}(\mathbf{w}_{t-1}, \boldsymbol{\beta}_{t-1})\right).$$

The new Pdprox-dual algorithm is presented in Algorithm 3. To prove the convergence rate of Algorithm 3, we can follow the same analysis to first prove the duality gap for $L(\mathbf{w}, \boldsymbol{\alpha}) + \lambda R(\mathbf{w}) - Q(\boldsymbol{\alpha})$ and then absorb $Q(\boldsymbol{\alpha})$ into the domain constraint of $\boldsymbol{\alpha}$. The convergence result presented in Theorem 1 holds the same for Algorithm 3.

Algorithm 3 The Pdprox-dual Algorithm for Non-Smooth Optimization

Input: step size $\gamma = \sqrt{1/(2c)}$, where c is specified in (6).
Initialization: $\mathbf{w}_0 = \mathbf{0}$, $\boldsymbol{\beta}_0 = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\boldsymbol{\alpha}_t = \Pi_{\mathcal{Q}_{\boldsymbol{\alpha}}}\left[\boldsymbol{\beta}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_{t-1}, \boldsymbol{\beta}_{t-1})\right]$
  $\mathbf{w}_t = \arg\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} \frac{1}{2}\left\|\mathbf{w} - \left(\mathbf{w}_{t-1} - \gamma G_{\mathbf{w}}(\mathbf{w}_{t-1}, \boldsymbol{\alpha}_t)\right)\right\|_2^2 + \gamma\lambda R(\mathbf{w})$
  $\boldsymbol{\beta}_t = \boldsymbol{\alpha}_t + \gamma(G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) - G_{\boldsymbol{\alpha}}(\mathbf{w}_{t-1}, \boldsymbol{\beta}_{t-1}))$
end for
Output $\widehat{\mathbf{w}}_T = \sum_{t=1}^T \mathbf{w}_t/T$ and $\widehat{\boldsymbol{\alpha}}_T = \sum_{t=1}^T \boldsymbol{\alpha}_t/T$.

Remark 5 In Appendix 3, we show that the updates on $(\mathbf{w}_t, \boldsymbol{\alpha}_t)$ of Algorithm 3 are essentially the same to the Algorithm 1 in Chambolle and Pock (2011), if we remove the extra dual variable in Algorithm 3 and the extra primal variable in Algorithm 1 in Chambolle and Pock (2011). However, the difference is that in Algorithm 3, we maintain two dual variables and one primal variable at each iteration, while the Algorithm 1 in Chambolle and Pock (2011) maintains two primal variables and one dual variable at each iteration.

For the composite gradient mapping for $\mathbf{w} \in \mathcal{Q}_{\mathbf{w}} = \mathbb{R}^d$, there is a closed form solution for simple regularizers (e.g., ℓ_1, ℓ_2) and decomposable regularizers (e.g., $\ell_{1,2}$). Efficient algorithms are available for composite gradient mapping when the regularizer is the ℓ_∞ and $\ell_{1,\infty}$, or trace norm. More details can be found in Duchi and Singer (2009) and Ji and Ye (2009). Here we present an efficient solution to a general regularizer $V(\|\mathbf{w}\|)$, where $\|\mathbf{w}\|$ is either a simple regularizer (e.g., ℓ_1, ℓ_2, and ℓ_∞) or a decomposable regularizer (e.g., $\ell_{1,2}$ and $\ell_{1,\infty}$), and $V(z)$ is convex and monotonically increasing for $z \geq 0$. An example is $V(\|\mathbf{w}\|) = (\sum_k \|\mathbf{w}_k\|_2)^2$, where $\mathbf{w}_1, \ldots, \mathbf{w}_K$ forms a partition of $\mathbf{w}$.

Lemma 7 *Let $V_*(\eta)$ be the convex conjugate of $V(z)$, i.e. $V(z) = \max_\eta \eta z - V_*(\eta)$. Then the solution to the composite mapping*

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} \frac{1}{2}\|\mathbf{w} - \widehat{\mathbf{w}}\|_2^2 + \lambda V(\|\mathbf{w}\|),$$

can be computed by

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{Q}_{\mathbf{w}}} \frac{1}{2}\|\mathbf{w} - \widehat{\mathbf{w}}\|_2^2 + \lambda\eta\|\mathbf{w}\|,$$

where η satisfies $\|\mathbf{w}^*\| - V_*'(\eta) = 0$. Since both $\|\mathbf{w}^*\|$ and $-V_*'(\eta)$ are non-increasing functions in η, we can efficiently compute η by a bi-section search.

The value of the step size γ in Algorithms 2 and 3 depends on the value of c, a constant that upper bounds the spectral norm square of the matrix $H(\mathbf{X}, \mathbf{y})$. In many machine learning applications, by assuming a bound on the data (e.g., $\|\mathbf{x}\|_2 \leq R$), one can easily compute an estimate of c. We present derivations of the constant c for hinge loss and generalized hinge loss in Appendix 1. However, the computed value of c might be overestimated, thus the step size γ is underestimated. Therefore, to improve the empirical performances, one can scale up the estimated value of γ by a factor larger than one and choose the best factor by tuning among a set of values. In addition, the authors in Chambolle and Pock (2011) suggested a two step sizes scheme with τ for updating the primal variable and σ for updating the dual variable. Depending on the nature of applications, one may observe better performances by carefully choosing the ratio between the two step sizes provided that σ and τ satisfy $\sigma\tau \leq 1/c$. In the last subsection, we observe the improved performance for solving SVM by using the two

step sizes scheme and by carefully tuning the ratio between the two step sizes. Furthermore, Pock and Chambolle (2011) presents a technique for computing diagonal preconditioners in the cases when estimating the value of c is difficult for complex problems, and applies it to general linear programing problems and some computer vision problems.

Finally, we discuss the two implementation schemes for Algorithms 2 and 3. Note that in Algorithm 2, we maintain and update two primal variables $\mathbf{w}_t, \mathbf{u}_t \in \mathbb{R}^d$, while in Algorithm 3 we maintain and update two dual variables $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t \in \mathbb{R}^n$. We refer to the implementation with two primal variables as double-primal implementation and the one with two dual variables as double-dual implementation. In fact, we can also implement Algorithm 2 by double-dual implementation and implement Algorithm 3 by double-primal implementation. For Algorithm 2, in which the updates are

$$\begin{aligned}
\mathbf{w}_t &= \min_{\mathbf{w}\in\mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w} - (\mathbf{u}_{t-1} - \gamma G_\mathbf{w}(\boldsymbol{\alpha}_{t-1}))\|_2^2}{2} + \gamma\lambda R(\mathbf{w}) \\
\boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha}\in\mathcal{Q}_\alpha} \frac{\|\boldsymbol{\alpha} - (\boldsymbol{\alpha}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t))\|_2^2}{2} \\
\mathbf{u}_t &= \mathbf{w}_t + \gamma(G_\mathbf{w}(\boldsymbol{\alpha}_{t-1}) - G_\mathbf{w}(\boldsymbol{\alpha}_t)),
\end{aligned}$$

we can plug the expression of $\mathbf{u}_t$ into $\mathbf{w}_t$ and obtain

$$\begin{aligned}
\mathbf{w}_t &= \min_{\mathbf{w}\in\mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w} - (\mathbf{w}_{t-1} + 2\gamma G_\mathbf{w}(\boldsymbol{\alpha}_{t-2}) - 2\gamma G_\mathbf{w}(\boldsymbol{\alpha}_{t-1}))\|_2^2}{2} + \gamma\lambda R(\mathbf{w}) \\
\boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha}\in\mathcal{Q}_\alpha} \frac{\|\boldsymbol{\alpha} - (\boldsymbol{\alpha}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{w}_t))\|_2^2}{2}
\end{aligned}$$

To implement above updates, we can only maintain one primal variable and two dual variables. Depending on the nature of implementation, one may be better than the other. For example, if the number of examples n is much larger than the number of dimensions d, the double-primal implementation may be more efficient than the double-dual implementation, and vice versa. In Sect. 5.7, we provide more examples and an experiment to demonstrate this.

4.7 Extensions and discussion

4.7.1 Nonlinear model

For a nonlinear model, the min–max formulation becomes

$$\min_{g\in\mathcal{H}_\kappa} \max_{\boldsymbol{\alpha}\in\mathcal{Q}_\alpha} L(g, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) + \lambda R(g),$$

where $\mathcal{H}_\kappa$ is a Reproducing Kernel Hilbert Space (RKHS) endowed with a kernel function $\kappa(\cdot,\cdot)$. Algorithm 1 can be applied to obtain the nonlinear model by changing the primal variable to g. For example, step 5 in Algorithm 1 is modified to the following composite gradient mapping

$$g_t = \arg\min_{g\in\mathcal{H}_\kappa} \frac{1}{2}\left\|g - \hat{g}_{t-1}\right\|_{\mathcal{H}_\kappa}^2 + \gamma\lambda R(g), \tag{18}$$

where

$$\hat{g}_{t-1} = \left(g_{t-1} - \gamma\nabla_g L(g_{t-1}, \boldsymbol{\alpha}_t; \mathbf{X}, \mathbf{y})\right).$$

Similar changes can be made to Algorithm 2 for the extension to the nonlinear model. To end this discussion, we make several remarks. (1) The gradient with respect to the primal variable (i.e., the kernel predictor $g \in \mathcal{H}_\kappa$) is computed on each $g(\mathbf{x}_i) = \langle g, \kappa(\mathbf{x}_i, \cdot)\rangle$ by $\kappa(\mathbf{x}_i, \cdot)$. (2) We can perform the computation by manipulating on a finite number of parameters due to the representer theorem provided that the regularizer $R(g)$ is a monotonic norm (Bach et al. 2011). Therefore, we only need to maintain and update the coefficients $\zeta = (\zeta_1, \ldots, \zeta_n)$ in $g = \sum_{i=1}^n \zeta_i \kappa(\mathbf{x}_i, \cdot)$. (3) The primal dual prox method for optimization with nonlinear model has been adopted in our prior work (Yang et al. 2012) for multiple kernel learning where the regularizer is $R(g_1, \ldots, g_m) = \left(\sum_{k=1}^m \|g_k\|_{\mathcal{H}_k}\right)^2$. It can also be generalized to solve MKL with more general sparsity-induced norms. (Bach et al. 2011 considers how to compute the proximal mapping in (18) for more general sparsity induced norms.)

4.7.2 Incorporating the bias term

It is easy to learn a bias term w_0 in the classifier $\mathbf{w}^\top \mathbf{x} + w_0$ by Pdprox without too many changes. We can use the augmented feature vector $\widehat{\mathbf{x}}_i = \begin{pmatrix} 1 \\ \mathbf{x}_i \end{pmatrix}$ and the augmented weight vector $\widehat{\mathbf{w}} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}$, and run Algorithms 1 or 2 with no changes except that the regularizer $R(\widehat{\mathbf{w}}) = R(\mathbf{w})$ does not involve w_0 and the step size $\gamma = \sqrt{1/(2c)}$ will be a different value due to the change in the bound of the new feature vectors by $\|\widehat{\mathbf{x}}\|_2 \leq \sqrt{1 + R^2}$, which would yield a different value of c in Lemma 1 (c.f. Appendix 1).

4.7.3 Domain constraint on primal variable

Now we discuss how to generalize the convergence analysis to the case when a convex domain $\mathcal{Q}_\mathbf{w}$ is imposed on $\mathbf{w}$. We introduce $\widehat{R}(\mathbf{w}) = \lambda R(\mathbf{w}) + Q(\mathbf{w})$, where $Q(\mathbf{w})$ is an indicator function for $\mathbf{w} \in \mathcal{Q}_\mathbf{w}$, i.e.

$$Q(\mathbf{w}) = \begin{cases} 0 & \mathbf{w} \in \mathcal{Q}_\mathbf{w} \\ +\infty & \text{otherwise} \end{cases}.$$

Then we can write the domain constrained composite gradient mapping in step 5 of Algorithm 1 or step 4 of Algorithm 2 into a domain free composite gradient mapping as the following:

$$\mathbf{w}_t = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w} - (\mathbf{w}_{t-1} - \gamma G_\mathbf{w}(\mathbf{w}_{t-1}, \boldsymbol{\alpha}_t))\|_2^2 + \gamma \widehat{R}(\mathbf{w}),$$
$$\mathbf{w}_t = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w} - (\mathbf{u}_{t-1} - \gamma G_\mathbf{w}(\mathbf{u}_{t-1}, \boldsymbol{\alpha}_{t-1}))\|_2^2 + \gamma \widehat{R}(\mathbf{w}).$$

Then we have an equivalent gradient mapping,

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \gamma G_\mathbf{w}(\mathbf{w}_{t-1}, \boldsymbol{\alpha}_t) - \gamma \partial \widehat{R}(\mathbf{w}_t),$$
$$\mathbf{w}_t = \mathbf{u}_{t-1} - \gamma G_\mathbf{w}(\mathbf{u}_{t-1}, \boldsymbol{\alpha}_{t-1}) - \gamma \partial \widehat{R}(\mathbf{w}_t).$$

Then Lemmas 2 and 3, and Lemmas 4 and 5 all hold as long as we replace $\lambda \mathbf{v}_t$ with $\widehat{\mathbf{v}}_t \in \partial \widehat{R}(\mathbf{w}_t)$. Finally in proving Theorems 1 we can absorb $Q(\mathbf{w})$ in $L(\mathbf{w}, \boldsymbol{\alpha}) + \widehat{R}(\mathbf{w})$ into the domain constraint.

4.7.4 Additional constraints on dual variables

One advantage of the proposed primal dual prox method is that it provides a convenient way to handle additional constraints on the dual variables α. Several studies introduce additional constraints on the dual variables. In Dekel and Singer (2006), the authors address a budget SVM problem by introducing a $1-\infty$ interpolation norm on the empirical hinge loss, leading to a sparsity constraint $\|\boldsymbol{\alpha}\|_1 \leq m$ on the dual variables, where m is the target number of support vectors. The corresponding optimization problem is given by

$$\min_{\mathbf{w}\in\mathbb{R}^d} \max_{\boldsymbol{\alpha}\in[0,1]^n, \|\boldsymbol{\alpha}\|_1 \leq m} \frac{1}{n}\sum_{i=1}^{n} \alpha_i (1 - y_i \mathbf{w}^\top \mathbf{x}_i) + \lambda R(\mathbf{w}). \tag{19}$$

In Huang et al. (2010), a similar idea is applied to learn a distance metric from noisy training examples. We can directly apply Algorithms 1 or 2 to (19) with $\mathcal{Q}_{\boldsymbol{\alpha}}$ given by $\mathcal{Q}_{\boldsymbol{\alpha}} = \{\boldsymbol{\alpha} : \boldsymbol{\alpha} \in [0,1]^n, \|\boldsymbol{\alpha}\|_1 \leq m\}$. The prox mapping to this domain can be efficiently computed by Lemma 6. It is straightforward to show that the convergence rate is $[D_1 + m]/[\sqrt{2n}T]$ in this case.

5 Experiments

In this section we present empirical studies to verify the efficiency of the proposed algorithm. We organize our experiments as follows.

- In Sects. 5.1, 5.2, and 5.3 we compare the proposed algorithm to the state-of-the-art first order methods that directly update the primal variable at each iteration. We apply all the algorithms to three different tasks with different non-smooth loss functions and regularizers. The baseline first order methods used in this study include the gradient descent algorithm (**gd**), the forward and backward splitting algorithm (**fobos**) (Duchi and Singer 2009), the regularized dual averaging algorithm (**rda**) (Xiao 2009), the accelerated gradient descent algorithm (**agd**) (Chen et al. 2009). Since the proposed method is a non-stochastic method, we compare it to the non-stochastic variant of **gd, fobos**, and **rda**. Note that **gd, fobos, rda**, and **agd** share the same convergence rate of $O(1/\sqrt{T})$ for non-smooth problems.
- In Sect. 5.4, our algorithm is compared to the state-of-the-art primal dual gradient method (Nesterov 2005a), which employs an excessive gap technique for non-smooth optimization, updates both the primal and dual variables at each iteration, and has a convergence rate of $O(1/T)$.
- In Sect. 5.5, we test the proposed algorithm for optimizing problem in (19) with a sparsity constraint on the dual variables.
- In Sect. 5.7, we compare the two variants of the proposed method on a data set when $n \gg d$, and compare Pdprox to the Pegasos algorithm.

All the algorithms are implemented in Matlab (except otherwise mentioned) and run on a 2.4 GHZ machine. Since the performance of the baseline algorithms **gd**, **fobos** and **rda** depends heavily on the initial value of the stepsize, we generate 21 values for the initial stepsize by scaling their theoretically optimal values with factors $2^{[-10:1:10]}$, and report the best convergence among the 21 possible values. The stepsize of **agd** is changed adaptively in the optimization process, and we just give it an appropriate initial step size. Since in the first four subsections we focus on comparison with baselines, we use the Pdprox-dual algorithm

(Algorithm 1) of the proposed Pdprox method. We also use the tuning technique to select the best scale-up factor for the step size γ of Pdprox. Finally, all algorithms are initialized with a solution of all zeros.

5.1 Group lasso regularizer for grouped feature selection

In this experiment we use the group lasso for regularization, i.e., $R(\mathbf{w}) = \sum_g \sqrt{d_g}\|\mathbf{w}_g\|_2$, where $\mathbf{w}_g$ corresponds to the gth group variables and d_g is the number of variables in group g. To apply Nesterov's method, we can write $R(\mathbf{w}) = \max_{\|\mathbf{u}_g\|_2 \le 1} \sum_g \sqrt{d_g}\mathbf{w}_g^\top \mathbf{u}_g$. We use the MEMset Donar dataset (Yeo and Burge 2003) as the testbed. This dataset was originally used for splice site detection. It is divided into a training set and a test set: the training set consists of 8,415 true and 179,438 false donor sites, and the testing set has 4,208 true and 89,717 false donor sites. Each example in this dataset was originally described by a sequence of {A, C, G, T} of length 7. We follow Yang et al. (2010) and generate group features with up to three-way interactions between the 7 positions, leading to 2,604 attributes in 63 groups. We normalize the length of each example to 1. Following the experimental setup in Yang et al. (2010), we construct a balanced training dataset consisting of all 8,415 true and 8,415 false donor sites that are randomly sampled from all 179,438 false sites.

Two non-smooth loss functions are examined in this experiment: hinge loss and absolute loss. Figure 2 plots the values of the objective function versus running time (s), using two different values of regularization parameter, i.e., $\lambda = 10^{-3}, 10^{-5}$ to produce different levels of sparsity. We observe that (i) the proposed algorithm **Pdprox** clearly outperforms all the baseline algorithms in all the cases; (ii) for the absolute loss, which has a sharp curvature change at zero compared to hinge loss, the baseline algorithms of **gd, fobos, rda, agd**, especially of **agd** that is originally designed for smooth loss functions, deteriorate significantly compared to the proposed algorithm **Pdprox**. Finally, we observe that for the hinge loss and $\lambda = 10^{-3}$, the classification performance of the proposed algorithm on the testing dataset is 0.6565, measured by maximum correlation coefficient (Yeo and Burge 2003). This is almost identical to the best result reported in Yang et al. (2010) (i.e., 0.6520).

5.2 $\ell_{1,\infty}$ regularization for multi-task learning

In this experiment, we perform multi-task regression with $\ell_{1,\infty}$ regularizer (Chen et al. 2009). Let $\mathbf{W} = (\mathbf{w}_1, \cdots, \mathbf{w}_k) \in \mathbb{R}^{d\times k}$ denote the k linear hypotheses for regression. The $\ell_{1,\infty}$ regularizer is given by $R(\mathbf{W}) = \sum_{j=1}^d \|\mathbf{w}^j\|_\infty$, where $\mathbf{w}^j$ is the jth row of $\mathbf{W}$. To apply Nesterov's method, we rewrite the $\ell_{1,\infty}$ regularizer as $R(\mathbf{W}) = \max_{\|\mathbf{u}_j\|_1 \le 1} \sum_{j=1}^d \mathbf{u}_j{}^\top \mathbf{w}^j$. We use the School data set (Argyriou et al. 2008), a common dataset for multi-task learning. This data set contains the examination scores of 15,362 students from 139 secondary schools corresponding to 139 tasks, one for each school. Each student in this dataset is described by 27 attributes. We follow the setup in Argyriou et al. (2008), and generate a training data set with 75 % of the examples from each school and a testing data set with the remaining examples. We test the algorithms using both the absolute loss and the ϵ-insensitive loss with $\epsilon = 0.01$. The initial stepsize for **gd**, **fobos**, and **rda** are tuned similarly as that for the experiment of group lasso. We plot the objective versus the running time in Fig. 3, from which we observe the similar results in the group feature selection task, i.e. (i) the proposed **Pdprox** algorithm outperforms the baseline algorithms, (ii) the baseline algorithm of **agd** becomes even worse for ϵ-insensitive loss than for absolute loss. Finally, we observe that the regression performance measured by root mean square error (RMSE) on the testing data set

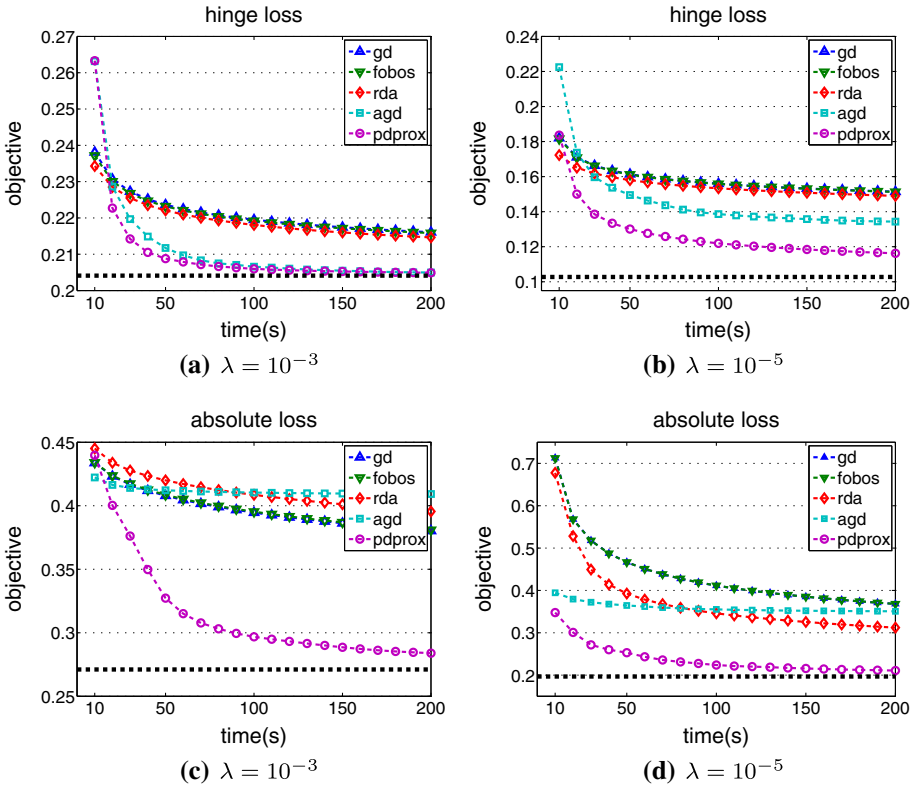


Fig. 2 Comparison of convergence speed for hinge loss (**a**, **b**) and absolute loss (**c**, **d**) with group lasso regularizer. Note that for better visualization we plot the objective starting from 10 s in all figures. The objective of all algorithms at 0 s is 1. The black bold dashed lines in all Figures show the optimal objective value by running Pdprox with a large number of iterations so that the duality gap is less than 10^{-4}

for absolute loss and ϵ-insensitive loss is 10.34 (optimized by **Pdprox**), comparable to the performance reported in Chen et al. (2009).

5.3 Trace norm regularization for max-margin matrix factorization/ matrix completion

In this experiment, we evaluate the proposed method using trace norm regularization, a regularizer often used in max-margin matrix factorization and matrix completion, where the goal is to recover a full matrix $\mathbf{X}$ from partially observed matrix $\mathbf{Y}$. The objective is composed of a loss function measuring the difference between $\mathbf{X}$ and $\mathbf{Y}$ on the observed entries and a trace norm regularizer on $\mathbf{X}$, assuming that $\mathbf{X}$ is low rank. Hinge loss function is used in max-margin matrix factorization (Rennie and Srebro 2005; Srebro et al. 2005), and absolute loss is used instead of square loss in matrix completion. We test on 100K MovieLens data set [3] that contains 1 million ratings from 943 users on 1,682 movies. Since there are five distinct ratings that can be assigned to each movie, we follow Rennie and Srebro (2005) and Srebro et al. (2005) by introducing four thresholds $\theta_{1,2,3,4}$ to measure the hinge loss between the predicted value X_{ij} and the ground truth Y_{ij}. Because our goal is to demonstrate the efficiency

[3] http://www.cs.umn.edu/Research/GroupLens/.

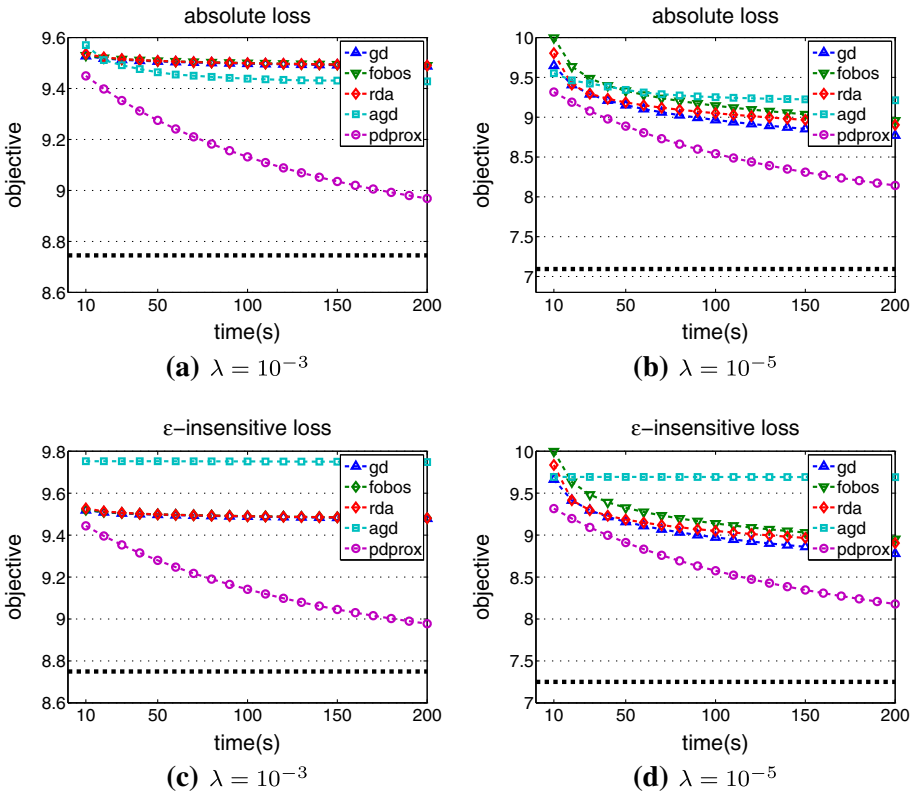


Fig. 3 Comparison of convergence speed for absolute loss (**a**, **b**) and ϵ-insensitive loss (**c**, **d**) with $\ell_{1,\infty}$ regularizer. Note that for better visualization we plot the objective starting from 10 s in all figures. The objective of all algorithms at 0 s is 20.52. The *black bold dashed lines* in all Figures show the optimal objective value by running Pdprox with a large number of iterations so that the duality gap is less than 10^{-4}

of the proposed algorithm for non-smooth optimization, therefore we simply set $\theta_{1,2,3,4} = (0, 3, 6, 9)$. Note that we did not compare to the optimization algorithm in Rennie and Srebro (2005) since it cast the problem into a non-convex problem by using explicit factorization of $\mathbf{X} = \mathbf{U}\mathbf{V}^\top$, which suffers a local minimum, and the optimization algorithm in Srebro et al. (2005) since it formulated the problem into a SDP problem, which suffers from a high computational cost. To apply Nesterov's method, we write $\|\mathbf{X}\|_1 = \max_{\|\mathbf{A}\|\leq 1} tr(\mathbf{A}^\top\mathbf{X})$, and at each iteration we need to solve a maximization problem $\max_{\|\mathbf{A}\|\leq 1} \lambda tr(\mathbf{A}^\top\mathbf{X}) - \frac{\mu}{2}\|\mathbf{A}\|_F^2$, where $\|\mathbf{A}\|$ is the spectral norm on $\mathbf{A}$. The solution of this optimization is obtained by performing SVD decomposition of $\mathbf{X}$ and thresholding the singular values appropriately. Since MovieLens data set is much larger than the data sets used in last two subsections, in this experiment, we (i) run all the algorithms for 1,000 iterations and plot the objective versus time; (ii) enlarge the range of tuning parameters to $2^{[-15:1:15]}$. The results are shown in Fig. 4, from which we observe that (i) Pdprox can quickly reduce the objective in a small amount of time, e.g., for absolute loss when setting $\lambda = 10^{-3}$ in order to obtain a solution with an accuracy of 10^{-3}, **Pdprox** needs 10^3 s, while **agd** needs 3.2×10^4 s; (ii) for absolute loss no matter how we tune the stepsizes for each baseline algorithm, Pdprox performs the best; and (iii) for hinge loss when $\lambda = 10^{-5}$, by tuning the stepsizes of baseline algorithms, **gd,**

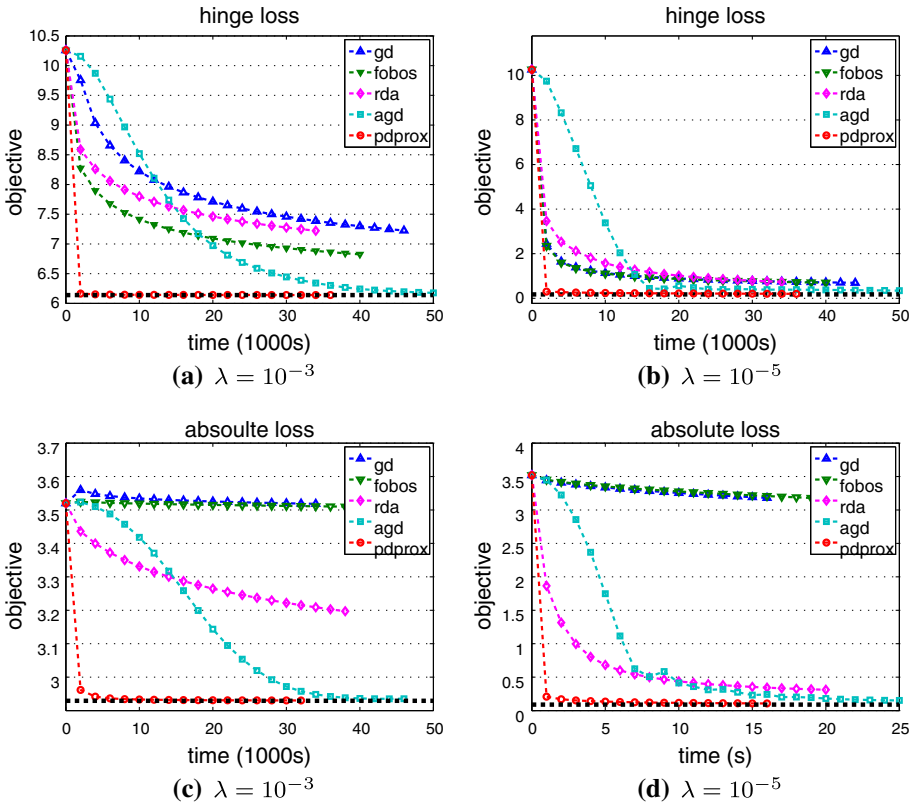


Fig. 4 Comparison of convergence speed for **a**, **b** max-margin matrix factorization with hinge loss and trace norm regularizer, and **c**, **d** matrix completion with absolute loss and trace norm regularizer. The *black bold dashed lines* in all Figures show the optimal objective value by running Pdprox with a large number of iterations so that the duality gap is less than 10^{-4}

fobos, and **rda** can achieve comparable performance to Pdprox. We note that although **agd** can achieve smaller objective value than Pdprox at the end of 1,000 iterations, however, the objective value is reduced slowly.

5.4 Comparison: Pdprox versus primal-dual method with excessive gap technique

In this section, we compare the proposed primal dual prox method to Nesterov's primal dual method (Nesterov 2005a), which is an improvement of his algorithm in Nesterov (2005b). The algorithm in Nesterov (2005b) for non-smooth optimization suffers a problem of setting the value of smoothing parameter that requires the number of iterations to be fixed in advance. Nesterov (2005a) addresses the problem by exploring an excessive gap technique and updating both the primal and dual variables, which is similar to the proposed Pdprox method. We refer to this baseline as **Pdexg**. We run both algorithms on the three tasks as in Sects. 5.1, 5.2, and 5.3, i.e., group feature selection with hinge loss and group lasso regularizer on MEMset Donar data set, multi-task learning with ϵ-insensitive loss and $\ell_{1,\infty}$ regularizer on School data set, and matrix completion with absolute loss and trace norm regularizer on 100 K

MovieLens data set. To implement the primal dual method with excessive gap technique, we need to intentionally add a domain on the optimal primal variable, which can be derived from the formulation. For example, in group feature selection problem whose objective is $1/n \sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \lambda \sum_g \sqrt{d_g} \|\mathbf{w}_g\|_2$, we can derive that the optimal primal variable $\mathbf{w}^*$ lies in $\|\mathbf{w}\|_2 \leq \sum_g \|\mathbf{w}_g\|_2 \leq \frac{1}{\lambda \sqrt{d_{\min}}}$, where $d_{\min} = \min_g d_g$. Similar techniques are applied to multi-task learning and matrix completion.

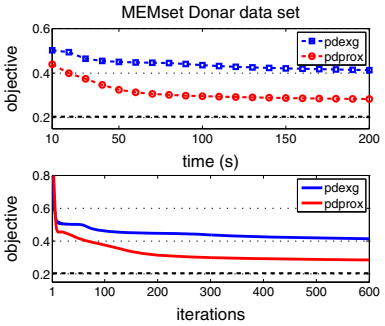
The performance of the two algorithms on the three tasks is shown in Fig. 5. Since both algorithms are in the same category, i.e. updating both primal and dual variables at each iteration and having a convergence rate in the order of $O(1/T)$, we also plot the objective versus the number of iterations in the bottom panels of each subfigure in Fig. 5.

The results show that the proposed Pdprox method converges faster than Pdexg on MEMset Donar data set for group feature selection with hinge loss and group lasso regularizer, and on 100 K MovieLens data set for matrix completion with absolute loss and trace norm regularizer. However, Pdexg performs better on School data set for multi-task learning with ϵ-insensitive loss and $\ell_{1,\infty}$ regularizer. One interesting phenomenon we can observe from Fig. 5 is that for larger values of λ (e.g., 10^{-3}), the improvement of Pdprox over Pdexg is also larger. The reason is that the proposed Pdprox captures the sparsity of primal variable at each iteration. This does not hold for Pdexg because it casts the non-smooth regularizer into a dual form and consequently does not explore the sparsity of the primal variable at each iteration. Therefore the larger of λ, the sparser of the primal variable at each iteration in Pdprox that yields to larger improvement over Pdexg. For the example of group feature selection task with hinge loss and group lasso regularizer, when setting $\lambda = 10^{-3}$, the sparsity of the primal variable (i.e., the proportion of the number of group features with zero norm) in Pdprox averaged over all iterations is 0.7886. However, by reducing λ to 10^{-5} the average sparsity of the primal variable in Pdprox is reduced to 0. In both settings the average sparsity of the primal variable in Pdexg is 0. The same argument also explains why Pdprox does not perform as well as Pdexg on School data set when setting $\lambda = 10^{-5}$, since in this case the primal variables in both algorithms are not sparse. When setting $\lambda = 10^{-3}$, the average sparsity (i.e., the proportion of the number of features with zero norm across all tasks) of the primal variable in Pdprox and Pdexg is 0.3766 and 0, respectively. Finally, we also observe similar performance of the two algorithms on the three tasks with other loss functions including absolute loss for group feature selection, absolute loss for multi-task learning, and hinge loss for max-margin matrix factorization.

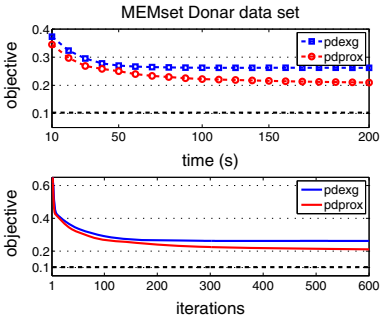
5.5 Sparsity constraint on the dual variables

In this subsection, we examine empirically the proposed algorithm for optimizing the problem in Eq. (19), in which a sparsity constraint is introduced for the dual variables. We test the algorithm on three large data sets from the UCI repository, namely, a9a, rcv1(binary) and covtye[4]. In the experiments we use ℓ_2^2 regularizer and fix $\lambda = 1/n$. First, we run the proposed algorithm 100 seconds on the three data sets with different values of $m = 100, 200, 400$ and plot the objective versus the number of iterations. The results are shown in Fig. 6, which verify that the convergence is faster with smaller m, which is consistent with the convergence bound $O([D + m]/[\sqrt{2n}\lambda])$ of the proposed algorithm for (19).

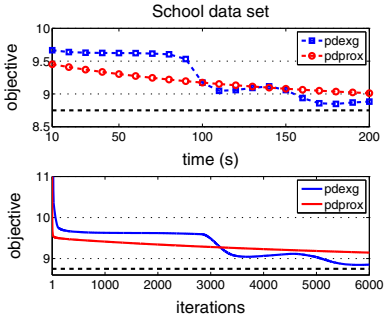
[4] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets.



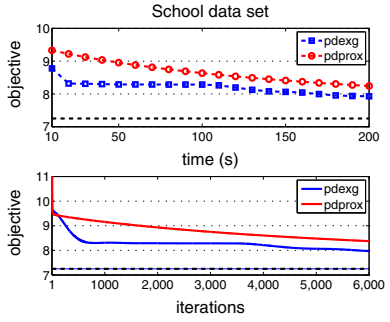
(a) Group feature selection: hinge loss and group lasso regularizer with $\lambda = 10^{-3}$.



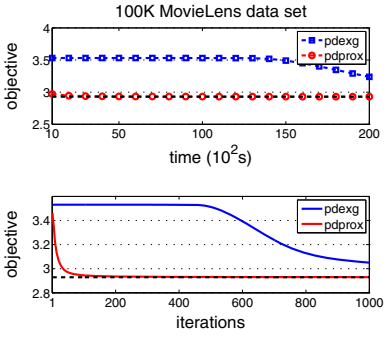
(b) Group feature selection: hinge loss and group lasso regularizer with $\lambda = 10^{-5}$.



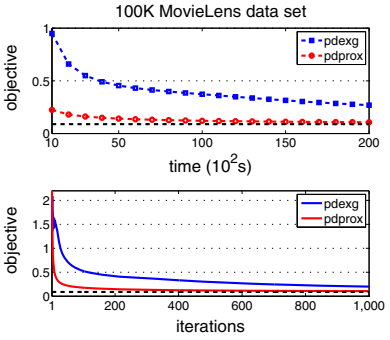
(c) Multi-task learning: ϵ-insensitive loss and $\ell_{1,\infty}$ regularizer with $\lambda = 10^{-3}$.



(d) Multi-task learning: ϵ-insensitive loss and $\ell_{1,\infty}$ regularizer with $\lambda = 10^{-5}$.



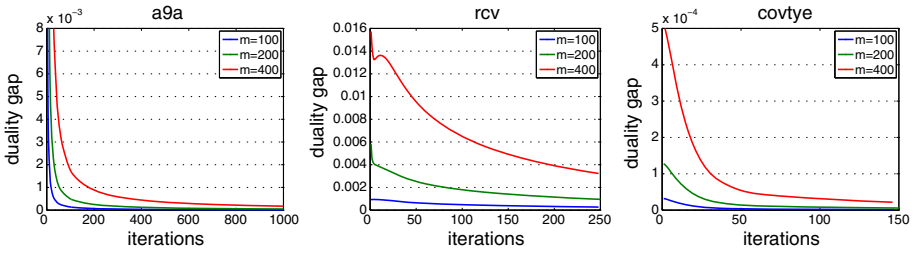
(e) Matrix completion: absolute loss and trace norm regularizer with $\lambda = 10^{-3}$.



(f) Matrix completion: absolute loss and trace norm regularizer with $\lambda = 10^{-5}$.

Fig. 5 Pdprox versus Primal-Dual method with excessive gap technique. The *black bold dashed lines* in all Figures show the optimal objective value by running Pdprox with a large number of iterations so that the duality gap is less than 10^{-4}

Second, we demonstrate that the formulation in Eq. (19) with a sparsity constraint on the dual variables is useful in the case when labels are contaminated with noise. To generate the noise in labels, we randomly flip the labels with a probability 0.2. We run both the proposed



(a) a9a **(b)** rcv1 **(c)** covtype

Fig. 6 Comparison of convergence with varied m

Table 1 Running time (forth column) and classification accuracy (fifth column) of Pdprox for (19) and of Liblinear on noisily labeled training data, where noise is added to labels by random flipping with a probability 0.2. We fix $\lambda = 1/n$ or $C = 1$ in Liblinear

Data set	(n, d)/ACC(%)	Alg.	Running time	ACC(%)
a9a	(32561, 123)	Pdprox(m = 200)	0.82s(0.01)	83.44(0.1)
	85.01	Liblinear	1.15s(0.57)	78.90(0.4)
rcv1	(20242, 47236)	Pdprox(m = 200)	1.57s(0.23)	94.05(0.2)
	96.54	Liblinear	3.30s(0.74)	93.66(0.2)
covtype	(571012, 54)	Pdprox(m = 4000)	48s(3.34)	73.58(0.01)
	75.80	Liblinear	37s(0.64)	68.66(0.001)

In the second column, we report the number of training examples (n), the number of attributes (d), and also the accuracy by training Liblinear on the original data and evaluating it on the testing data

algorithm for (19) and Liblinear[5] on the training data with noise added to the labels. The stopping criterion for the proposed algorithm is when duality gap is less than 10^{-3}, and for Liblinear is when the maximal dual violation is less than 10^{-3}. The running time and accuracy on testing data averaged over 5 random trials are reported in Table 1, which demonstrate that in the presence of noise in labels, by adding a sparsity constraint on the dual variables, we are able to obtain better performance than Liblinear trained on the noisily labeled data. Furthermore the running time of Pdprox is comparable to, if not less than, that of Liblinear.

Finally, we note that choosing a small m in Eq. (19) is different from simply training a classifier with a small number of examples. For instance, for rcv1, we have run the experiment with 200 training examples, randomly selected from the entire data set. With the same stopping criterion, the testing performance is $0.8131(\pm 0.05)$, significantly lower than that of optimizing (19) with $m = 200$.

5.6 Comparison: double-primal versus double-dual implementation

From the discussion in Sect. 4.6, we have seen that both Pdprox-primal and Pdprox-dual algorithm can be implemented either by maintaining two dual variables, to which we refer as double-dual implementation, or by maintaining two primal variables, to which we refer as double-primal implementation. One implementation could be more efficient than the other implementation, depending on the nature of applications. For example, in multi-task regression with ℓ_2 loss (Nie et al. 2010), if the number of examples is much larger than the number

[5] http://www.csie.ntu.edu.tw/~cjlin/liblinear.

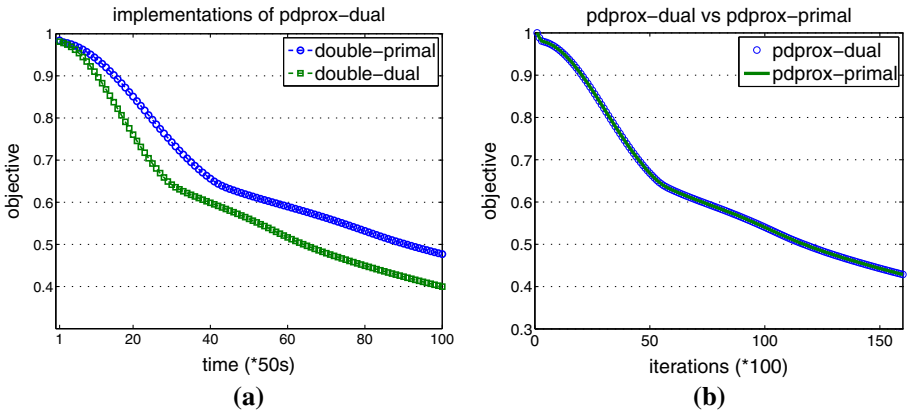


Fig. 7 **a** Comparison of double-primal implementation versus double-dual implementation of Pdprox-dual, and **b** comparison of Pdprox-dual versus Pdprox-primal both with double-dual implementation, on a subset of webspam data using trigram features. **a** indicates that the two implementation methods may affect its performance. **b** shows that the two algorithms have almost the same performance with the same implementation framework

of attributes, i.e., $n \gg d$, and the number of tasks K is large, then the size of dual variable $\alpha \in \mathbb{R}^{n\times K}$ is much larger than the size of primal variable $W \in \mathbb{R}^{d\times K}$. It would be expected that the double-primal implementation is more efficient than the double-dual implementation. In contrast, in matrix completion with absolute loss, if the number of observed entries $|\Omega|$ which corresponds to the size of dual variable is much less than the total number of entries n^2 which corresponds to the size of primal variable, then the double-dual implementation would be more efficient than the double-primal implementation.

In the following experiment, we restrict our demonstration to a binary classification problem that given a set of training examples $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i \in \mathbb{R}^d$, one aims to learn a prediction model $\mathbf{w} \in \mathbb{R}^d$. We choose web spam data set [6] as the testing bed, which contains 350000 examples, and 16609143 trigrams extracted for each example. We use hinge loss and ℓ_2^2 regularizer with $\lambda = 1/n$, where n is the number of experimented data.

We demonstrate that when $d \gg n$, the double-dual implementation is more efficient than double-primal implementation. For the purpose of demonstration, we randomly sample from the whole data a subset of $n = 1{,}000$ examples, which have a total of 8287348 features, and we solve the sub-optimization problem over the subset. It is worth noting that such kind of problem appears commonly in distributed computing on individual nodes when the number of attributes is huge. The objective value versus running time of the two implementations of Pdprox-dual are plotted in Fig. 7, which shows that double-dual implementation is more efficient than double-primal implementation is this case. As a complement, we also plot the objective of Pdprox-dual and Pdprox-primal both with double-dual implementation, which shows that Pdprox-primal and Pdprox-dual performs similarly.

5.7 Comparison for solving ℓ_2^2 regularized SVM

In this subsection, we compare the proposed Pdprox method with Pegasos for solving ℓ_2^2 regularized SVM when $\lambda = O(n^{-1/(1+\epsilon)})$, $\epsilon \in (0, 1]$. We also compare Pdprox using one step size and two step sizes, and compare them to the accelerated version proposed in Chambolle

[6] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

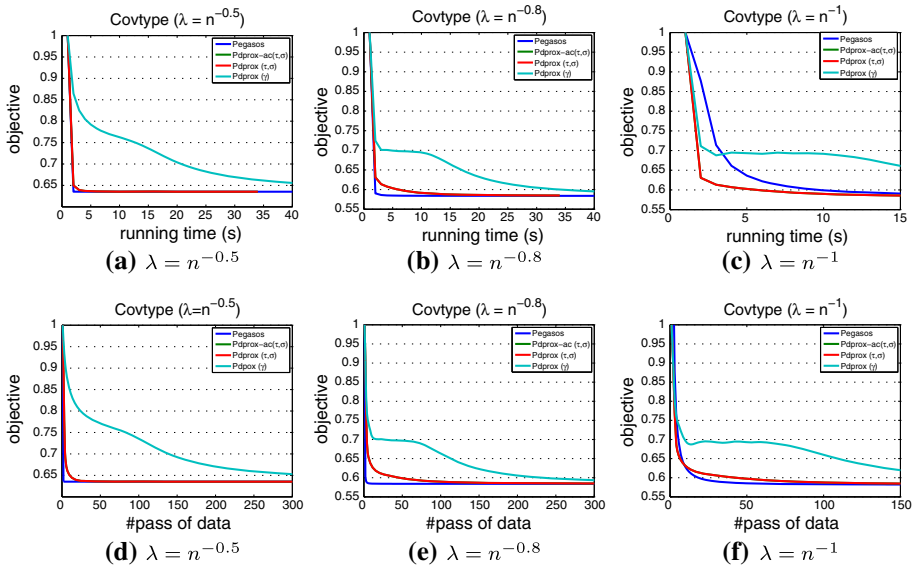


Fig. 8 Comparison of convergence speed of Pdprox versus Pegasos on covtype data set. The best ratio between the step size τ for updating $\mathbf{w}$ and the step size σ for updating $\boldsymbol{\alpha}$ is 0.01. The curves of Pdprox-ac(τ, σ) are overlapped with that of Pdprox (τ, σ)

and Pock (2011) for strongly convex functions. We implement Pdprox-dual algorithm (by double-dual implementation) in C++ using the same data structures as coded by Shai Shalev-Shwartz [7].

Figure 8 shows the comparison of **Pdprox** versus **Pegasos** on covtype data set with three different levels of $\lambda = n^{-0.5}, n^{-0.8}, n^{-1}$. We compute the objective value of Pdprox after each iteration and compute the objective value of Pegasos after one effective pass of all data (i.e., n number of iterations where n is the total number of training examples). We also compare the one step size scheme (Pdprox (γ)) with the two step sizes scheme (Pdprox (τ, σ)) and the accelerated version (Pdprox-ac(τ, σ)) proposed in Chambolle and Pock (2011) for strongly convex functions. The relative ratio between the step size τ for updating the primal variable and the step size σ for updating the dual variable is selected among a set of values $\{1000, 100, 10, 1, 0.1, 0.01, 0.001\}$.

The results demonstrate that (1) the two step sizes scheme with careful tuning of the relative ratio yields better convergences than the one step size scheme; (2) Pegasos still remains a state-of-the-art algorithm for solving the ℓ_2^2 regularized SVM; but when the problem is relatively difficult, i.e., λ is relatively small (e.g., less than $1/n$), the Pdprox algorithm with the two step sizes may converge faster in terms of running time; (3) the accelerated version for solving SVM is almost identical the basic version.

6 Conclusions

In this paper, we study non-smooth optimization in machine learning where both the loss function and the regularizer are non-smooth. We develop an efficient gradient based method

[7] http://www.cs.huji.ac.il/~shais/code/index.html.

for a family of non-smooth optimization problems in which the dual form of the loss function can be expressed as a bilinear function in primal and dual variables. We show that, assuming the proximal step can be efficiently solved, the proposed algorithm achieves a convergence rate of $O(1/T)$, faster than $O(1/\sqrt{T})$ suffered by many other first order methods for non-smooth optimization. In contrast to existing studies on non-smooth optimization, our work enjoys more simplicity in implementation and analysis, and provides a unified methodology for a diverse set of non-smooth optimization problems. Our empirical studies demonstrate the efficiency of the proposed algorithm in comparison with the state-of-the-art first order methods for solving many non-smooth machine learning problems, and the effectiveness of the proposed algorithm for optimizing the problem with a sparse constraint on the dual variables for tackling the noise in labels. In future, we plan to adapt the proposed algorithm for stochastic updating and for distributed computing environments.

Appendix 1: derivation of constant c for (generalized) hinge loss

As mentioned before, it is easy to derive the constant c in Eqs. (6) and (7) for the non-smooth loss functions listed before under the assumption that $\|\mathbf{x}\|_2 \leq R$. As an example, here we derive the constant for hinge loss and generalized hinge loss. For other non-smooth loss functions, we can derive the value of c in a similar way. For hinge loss, $L(\mathbf{w}, \boldsymbol{\alpha})$ in (5) is given by

$$L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \alpha_i (1 - y_i \mathbf{w}^\top \mathbf{x}_i),$$

and its partial gradients are

$$\begin{aligned} G_{\boldsymbol{\alpha}}(\mathbf{w}, \boldsymbol{\alpha}) &= \frac{1}{n}\mathbf{1} - \frac{1}{n}(\mathbf{x}_1 y_1, \cdots, \mathbf{x}_n y_n)^\top \mathbf{w}, \\ G_{\mathbf{w}}(\mathbf{w}, \boldsymbol{\alpha}) &= -\frac{1}{n}\mathbf{X}(\boldsymbol{\alpha} \circ \mathbf{y}), \end{aligned}$$

where $\mathbf{1}$ denotes a vector of all ones, and $\circ$ denotes the element-wise product. Then,

$$\begin{aligned} \|G_{\boldsymbol{\alpha}}(\mathbf{w}_1, \boldsymbol{\alpha}_1) - G_{\boldsymbol{\alpha}}(\mathbf{w}_2, \boldsymbol{\alpha}_2)\|_2^2 &= \frac{1}{n^2} \sum_{i=1}^{n} \left(\mathbf{w}_1^\top \mathbf{x}_i y_i - \mathbf{w}_2^\top \mathbf{x}_i y_i \right)^2 \leq \frac{R^2}{n} \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2, \\ \|G_{\mathbf{w}}(\mathbf{w}_1, \boldsymbol{\alpha}_1) - G_{\mathbf{w}}(\mathbf{w}_2, \boldsymbol{\alpha}_2)\|_2^2 &= \frac{1}{n^2} \left\| \sum_{i=1}^{n} \left(\alpha_i^1 - \alpha_i^2 \right) y_i \mathbf{x}_i \right\|_2^2 \\ &\leq \frac{R^2}{n} \sum_{i=1}^{n} \left(\alpha_i^1 - \alpha_i^2 \right)^2 = \frac{R^2}{n} \|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2^2, \end{aligned}$$

which implies $c = R^2/n$. For the example of generalized hinge loss, $L(\mathbf{w}, \boldsymbol{\alpha})$ in (5) is

$$L(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \alpha_i^1 \left(1 - a y_i \mathbf{w}^\top \mathbf{x}_i \right) + \alpha_i^2 (1 - y_i \mathbf{w}^\top \mathbf{x}_i),$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2] \in \mathcal{Q}_{\boldsymbol{\alpha}} = \{\boldsymbol{\alpha} : \boldsymbol{\alpha} \in [0,1]^{n\times 2}, \boldsymbol{\alpha}^1 + \boldsymbol{\alpha}^2 \leq \mathbf{1}\}$, and its partial gradients are

$$G_{\boldsymbol{\alpha}}(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{n}[\mathbf{1}, \mathbf{1}] - \frac{1}{n}\left[a\left(\mathbf{x}_1 y_1, \cdots, \mathbf{x}_n y_n\right)^\top \mathbf{w}, \left(\mathbf{x}_1 y_1, \cdots, \mathbf{x}_n y_n\right)^\top \mathbf{w}\right],$$
$$G_{\mathbf{w}}(\mathbf{w}, \boldsymbol{\alpha}) = -\frac{1}{n}\mathbf{X}\left(a(\boldsymbol{\alpha}^1 \circ \mathbf{y}) + \boldsymbol{\alpha}^2 \circ \mathbf{y}\right),$$

where $\mathbf{1}$ denotes a vector of all ones, and $\circ$ denotes the element-wise product. Then for any $\mathbf{w}_1, \mathbf{w}_2$ and $\boldsymbol{\alpha}_1 = (\boldsymbol{\alpha}^{1,1}, \boldsymbol{\alpha}^{2,1}), \boldsymbol{\alpha}_2 = (\boldsymbol{\alpha}^{1,2}, \boldsymbol{\alpha}^{2,2}) \in \mathcal{Q}_{\boldsymbol{\alpha}}$, given $\|\mathbf{x}\|_2 \leq R$, we have

$$\begin{aligned}
\|G_{\boldsymbol{\alpha}}(\mathbf{w}_1, \boldsymbol{\alpha}_1) - G_{\boldsymbol{\alpha}}(\mathbf{w}_2, \boldsymbol{\alpha}_2)\|_F^2 &= \frac{a^2+1}{n^2}\sum_{i=1}^n \left(\mathbf{w}_1^\top \mathbf{x}_i y_i - \mathbf{w}_2^\top \mathbf{x}_i y_i\right)^2 \\
&\leq \frac{(a^2+1)R^2}{n}\|\mathbf{w}_1 - \mathbf{w}_2\|_2^2, \\
\|G_{\mathbf{w}}(\mathbf{w}_1, \boldsymbol{\alpha}_1) - G_{\mathbf{w}}(\mathbf{w}_2, \boldsymbol{\alpha}_2)\|_2^2 &= \frac{1}{n^2}\left\|\sum_{i=1}^n a\left(\alpha_i^{1,1} - \alpha_i^{1,2}\right) y_i \mathbf{x}_i + \sum_{i=1}^n \left(\alpha_i^{2,1} - \alpha_i^{2,2}\right) y_i \mathbf{x}_i\right\|_2^2 \\
&\leq \frac{2a^2R^2}{n}\sum_{i=1}^n \left(\alpha_i^{1,1} - \alpha_i^{1,2}\right)^2 + \frac{2R^2}{n}\sum_{i=1}^n \left(\alpha_i^{2,1} - \alpha_i^{2,2}\right)^2 \\
&\leq \frac{2a^2R^2}{n}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_F^2,
\end{aligned}$$

which implies $c = (a^2+1)R^2/n$ in Eq. (6) and $c = (2a^2R^2)/n$ in Eq. (7). We can derive the value of c in (6) and (7) similarly for other non-smooth loss functions.

Appendix 2: proof of lemma 1

Since

$$G_{\boldsymbol{\alpha}}(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) = \mathbf{a}(\mathbf{X}, \mathbf{y}) + H(\mathbf{X}, \mathbf{y})^\top \mathbf{w},$$
$$G_{\mathbf{w}}(\mathbf{w}, \boldsymbol{\alpha}; \mathbf{X}, \mathbf{y}) = \mathbf{b}(\mathbf{X}, \mathbf{y}) + H(\mathbf{X}, \mathbf{y})\boldsymbol{\alpha}.$$

Then

$$\|G_{\boldsymbol{\alpha}}(\mathbf{w}_1, \boldsymbol{\alpha}_1; \mathbf{X}, \mathbf{y}) - G_{\boldsymbol{\alpha}}(\mathbf{w}_2, \boldsymbol{\alpha}_2; \mathbf{X}, \mathbf{y})\|_2^2 \leq \|H(\mathbf{X}, \mathbf{y})^\top(\mathbf{w}_1 - \mathbf{w}_2)\|_2^2 \leq c\|\mathbf{w}_1 - \mathbf{w}_2\|_2^2,$$
$$\|G_{\mathbf{w}}(\mathbf{w}_1, \boldsymbol{\alpha}_1; \mathbf{X}, \mathbf{y}) - G_{\mathbf{w}}(\mathbf{w}_2, \boldsymbol{\alpha}_2; \mathbf{X}, \mathbf{y})\|_2^2 \leq \|H(\mathbf{X}, \mathbf{y})(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)\|_2^2 \leq c\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2^2,$$

where we use the assumption $\|H(\mathbf{X}, \mathbf{y})\|_2^2 = \|H(\mathbf{X}, \mathbf{y})^\top\|_2^2 \leq c$.

Appendix 3: the differences between Algorithm 1 in Chambolle and Pock (2011) and Pdprox-primal algorithm (Algorithm 2) and Pdprox-dual algorithm (Algorithm 3)

We make the following correspondences between our notations (appearing the R.H.S of the following equalities) and the notations in Chambolle and Pock (2011) (appearing the L.H.S

of the following equalities),

$$\begin{aligned}
&\mathbf{x} = \mathbf{w}, \quad \mathbf{y} = \boldsymbol{\alpha}, \quad \bar{\mathbf{x}} = \mathbf{u} \\
&G(\mathbf{w}) = \lambda R(\mathbf{w}) + \mathbf{w}^\top \mathbf{b} + I_{\mathcal{Q}_\mathbf{w}}(\mathbf{w}) \\
&F^*(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}^\top \mathbf{a} + I_{\mathcal{Q}_{\boldsymbol{\alpha}}}(\boldsymbol{\alpha}) \\
&K = H^\top \\
&\alpha^\top H^\top \mathbf{w} + \mathbf{w}^\top \mathbf{b} + \boldsymbol{\alpha}^\top \mathbf{a} + c_0 = L(\mathbf{w}, \boldsymbol{\alpha}) \\
&\delta = \tau = \gamma \\
&\theta = 1
\end{aligned}$$

where we suppress the dependence of $\mathbf{a}$, $\mathbf{b}$, H, c_0 on $(\mathbf{X}, \mathbf{y})$, and $I_{\mathcal{Q}}(\mathbf{x})$ is an indicator function

$$I_{\mathcal{Q}}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \in \mathcal{Q} \\ +\infty, & \text{otherwise} \end{cases}$$

The problem in Chambolle and Pock (2011) is to solve

$$\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} \mathcal{O}(\mathbf{w}, \boldsymbol{\alpha}) = \alpha^\top H^\top \mathbf{w} + G(\mathbf{w}) - F^*(\alpha)$$

and the updates in Chambolle and Pock (2011) are calculated by

$$\begin{aligned}
\boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha}} \frac{\|\boldsymbol{\alpha} - (\boldsymbol{\alpha}_{t-1} + \gamma H^\top \mathbf{u}_{t-1})\|_2^2}{2\gamma} + F^*(\alpha) \\
\mathbf{w}_t &= \min_{\mathbf{w}} \frac{\|\mathbf{w} - (\mathbf{w}_{t-1} - \gamma H \boldsymbol{\alpha}_t))\|_2^2}{2\gamma} + G(\mathbf{w}) \\
\mathbf{u}_t &= \mathbf{w}_t + \theta(\mathbf{w}_t - \mathbf{w}_{t-1})
\end{aligned}$$

or equivalently

$$\begin{aligned}
\boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}}} \frac{\|\boldsymbol{\alpha} - (\boldsymbol{\alpha}_{t-1} + \gamma(H^\top \mathbf{u}_{t-1} + \mathbf{a}))\|_2^2}{2\gamma} \\
\mathbf{w}_t &= \min_{\mathbf{w} \in \mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w} - (\mathbf{w}_{t-1} - \gamma(H \boldsymbol{\alpha}_t + \mathbf{b}))\|_2^2}{2\gamma} + \lambda R(\mathbf{w}) \\
\mathbf{u}_t &= \mathbf{w}_t + \theta(\mathbf{w}_t - \mathbf{w}_{t-1})
\end{aligned}$$

Note that the partial gradients of $L(\mathbf{w}, \boldsymbol{\alpha})$ are $G_\mathbf{w}(\mathbf{w}, \boldsymbol{\alpha}) = G_\mathbf{w}(\boldsymbol{\alpha}) = H\boldsymbol{\alpha} + b$ and $G_{\boldsymbol{\alpha}}(\mathbf{w}, \boldsymbol{\alpha}) = G_{\boldsymbol{\alpha}}(\mathbf{w}) = H^\top \mathbf{w} + \mathbf{a}$[8], then we can write the above updates as

$$\begin{aligned}
\boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha} \in \mathcal{Q}_{\boldsymbol{\alpha}}} \frac{\|\boldsymbol{\alpha} - (\boldsymbol{\alpha}_{t-1} + \gamma G_{\boldsymbol{\alpha}}(\mathbf{u}_{t-1}))\|_2^2}{2} \\
\mathbf{w}_t &= \min_{\mathbf{w} \in \mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w} - (\mathbf{w}_{t-1} - \gamma G_\mathbf{w}(\boldsymbol{\alpha}_t))\|_2^2}{2} + \gamma\lambda R(\mathbf{w}) \\
\mathbf{u}_t &= \mathbf{w}_t + \theta(\mathbf{w}_t - \mathbf{w}_{t-1})
\end{aligned}$$

[8] We use $G_\mathbf{w}$ and $G_{\boldsymbol{\alpha}}$ to denote partial gradients.

However the updates of Pdprox-primal algorithm (Algorithm 2) in our paper are

$$\begin{aligned}\mathbf{w}_t &= \min_{\mathbf{w}\in\mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w}-(\mathbf{u}_{t-1}-\gamma G_\mathbf{w}(\boldsymbol{\alpha}_{t-1}))\|_2^2}{2} + \gamma\lambda R(\mathbf{w})\\ \boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha}\in\mathcal{Q}_\alpha} \frac{\|\boldsymbol{\alpha}-(\boldsymbol{\alpha}_{t-1}+\gamma G_\boldsymbol{\alpha}(\mathbf{w}_t))\|_2^2}{2}\\ \mathbf{u}_t &= \mathbf{w}_t + \gamma(G_\mathbf{w}(\boldsymbol{\alpha}_{t-1}) - G_\mathbf{w}(\boldsymbol{\alpha}_t))\end{aligned}$$

If we remove the extra primal variable $\mathbf{u}_t$, we have the following updates of Algorithm 1 in Chambolle and Pock (2011):

$$\begin{aligned}\boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha}\in\mathcal{Q}_\alpha} \frac{\|\boldsymbol{\alpha}-(\boldsymbol{\alpha}_{t-1}+\gamma G_\boldsymbol{\alpha}(2\mathbf{w}_{t-1}-\mathbf{w}_{t-2}))\|_2^2}{2}\\ \mathbf{w}_t &= \min_{\mathbf{w}\in\mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w}-(\mathbf{w}_{t-1}-\gamma G_\mathbf{w}(\boldsymbol{\alpha}_t))\|_2^2}{2} + \gamma\lambda R(\mathbf{w})\end{aligned} \tag{20}$$

and the following updates of the Pdprox-primal algorithm:

$$\begin{aligned}\mathbf{w}_t &= \min_{\mathbf{w}\in\mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w}-(\mathbf{w}_{t-1}-\gamma G_\mathbf{w}(2\boldsymbol{\alpha}_{t-1}-\boldsymbol{\alpha}_{t-2}))\|_2^2}{2} + \gamma\lambda R(\mathbf{w})\\ \boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha}\in\mathcal{Q}_\alpha} \frac{\|\boldsymbol{\alpha}-(\boldsymbol{\alpha}_{t-1}+\gamma G_\boldsymbol{\alpha}(\mathbf{w}_t))\|_2^2}{2}\end{aligned} \tag{21}$$

We can clearly see the difference between our updates and the updates of Algorithm 1 in Chambolle and Pock (2011), which lies in the order of updating on the primal variable and the dual variable, and the gradients used in the updating as well. On the other hand, if we remove the extra dual variable in Algorithm 3, the updates are the same to that of Algorithm in Chambolle and Pock (2011), i.e.,

$$\begin{aligned}\boldsymbol{\alpha}_t &= \min_{\boldsymbol{\alpha}\in\mathcal{Q}_\alpha} \frac{\|\boldsymbol{\alpha}-(\boldsymbol{\alpha}_{t-1}+\gamma(2G_\boldsymbol{\alpha}(\mathbf{w}_{t-1}) - G_\boldsymbol{\alpha}(\mathbf{w}_{t-2}))\|_2^2}{2}\\ \mathbf{w}_t &= \min_{\mathbf{w}\in\mathcal{Q}_\mathbf{w}} \frac{\|\mathbf{w}-(\mathbf{w}_{t-1}-\gamma G_\mathbf{w}(\boldsymbol{\alpha}_t))\|_2^2}{2} + \gamma\lambda R(\mathbf{w})\end{aligned} \tag{22}$$

by noting that $G_\boldsymbol{\alpha}(\mathbf{w})$ is linear in $\mathbf{w}$. It is also worth noting that Pdprox-primal can be implemented by maintaing one primal variable and two dual variables as in (20), and similarly Pdprox-dual can be implemented by maintaing two primal variables and one dual variable as in (21). Depending on the nature of applications, we can choose different implementations for Pdprox-primal or Pdprox-dual to achieve better efficiency.

Appendix 4: proof of lemma 3

In order to prove Lemma 3, we first present the following lemma with its proof.

Lemma 8 *Let Z be a convex compact set, and $U \subseteq Z$ be convex and closed, $\mathbf{z}_0 \in Z$, and $\gamma > 0$. Considering the following points with fixed η, ξ,*

$$\begin{aligned}\mathbf{z}_h &= \arg\min_{\mathbf{z}\in U} \frac{1}{2}\|\mathbf{z}-(\mathbf{z}_0-\gamma\xi)\|_2^2,\\ \mathbf{z}_1 &= \arg\min_{\mathbf{z}\in U} \frac{1}{2}\|\mathbf{z}-(\mathbf{z}_0-\gamma\eta)\|_2^2,\end{aligned}$$

then for any $\mathbf{z} \in U$, *we have*

$$\gamma \eta^\top (\mathbf{z}_h - \mathbf{z}) \le \frac{1}{2}\|\mathbf{z} - \mathbf{z}_0\|_2 - \frac{1}{2}\|\mathbf{z} - \mathbf{z}_1\|_2 + \gamma^2 \|\xi - \eta\|_2^2 - \frac{1}{2}\Big[\|\mathbf{z}_h - \mathbf{z}_0\|_2^2 + \|\mathbf{z}_h - \mathbf{z}_1\|_2^2\Big].$$

Equipped with above lemma, it is straightforward to prove Lemma 3. We note that the two updates in Lemma 2 are the same as the two updates in Lemma 8 if we make the following correspondences:

$$\begin{aligned}
&U = Z = \mathbb{R}^d \times \mathcal{Q}_{\boldsymbol{\alpha}}, \quad \mathbf{z} = \begin{pmatrix} \mathbf{w} \\ \alpha \end{pmatrix} \in U, \\
&\mathbf{z}_0 = \begin{pmatrix} \mathbf{u}_{t-1} \\ \boldsymbol{\beta}_{t-1} \end{pmatrix}, \ \mathbf{z}_h = \begin{pmatrix} \mathbf{w}_t \\ \boldsymbol{\alpha}_t \end{pmatrix}, \ \mathbf{z}_1 = \begin{pmatrix} \mathbf{u}_t \\ \boldsymbol{\beta}_t \end{pmatrix}, \\
&\xi = \begin{pmatrix} G_{\mathbf{w}}(\mathbf{u}_{t-1}, \boldsymbol{\alpha}_t) + \lambda \mathbf{v}_t \\ -G_{\boldsymbol{\alpha}}(\mathbf{u}_{t-1}, \boldsymbol{\beta}_{t-1}) \end{pmatrix}, \quad \eta = \begin{pmatrix} G_{\mathbf{w}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) + \lambda \mathbf{v}_t \\ -G_{\alpha}(\mathbf{w}_t, \boldsymbol{\alpha}_t) \end{pmatrix}.
\end{aligned}$$

Then the inequality in Lemma 3 follows immediately the inequality in Lemma 8, which is stated explicitly again:

$$\begin{aligned}
\gamma \begin{pmatrix} G_{\mathbf{w}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) + \lambda \mathbf{v}_t \\ -G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) \end{pmatrix}^\top \begin{pmatrix} \mathbf{w}_t - \mathbf{w} \\ \boldsymbol{\alpha}_t - \boldsymbol{\alpha} \end{pmatrix} &\le \frac{1}{2}\left\| \begin{pmatrix} \mathbf{w} - \mathbf{u}_{t-1} \\ \boldsymbol{\alpha} - \boldsymbol{\beta}_{t-1} \end{pmatrix} \right\|_2^2 - \frac{1}{2}\left\| \begin{pmatrix} \mathbf{w} - \mathbf{u}_t \\ \boldsymbol{\alpha} - \boldsymbol{\beta}_t \end{pmatrix} \right\|_2^2 \\
&+ \gamma^2 \left\| G_{\boldsymbol{\alpha}}(\mathbf{w}_t, \boldsymbol{\alpha}_t) - G_{\boldsymbol{\alpha}}(\mathbf{u}_{t-1}, \boldsymbol{\beta}_{t-1}) \right\|_2^2 \\
&- \frac{1}{2}\left[\|\mathbf{w}_t - \mathbf{u}_{t-1}\|_2^2 + \underbrace{\|\boldsymbol{\alpha}_t - \boldsymbol{\beta}_{t-1}\|_2^2 + \|\mathbf{w}_t - \mathbf{u}_t\|_2^2 + \|\boldsymbol{\alpha}_t - \boldsymbol{\beta}_t\|_2^2}_{\ge 0} \right].
\end{aligned}$$

Lemma 8 is a special case of Lemma 3.1 Nemirovski (2005) for Euclidean norm. A proof is provided here for completeness.

Proof (of Lemma 8) Since

$$\begin{aligned}
\mathbf{z}_h &= \arg\min_{\mathbf{z} \in U} \frac{1}{2}\|\mathbf{z} - (\mathbf{z}_0 - \gamma \xi)\|_2^2, \\
\mathbf{z}_1 &= \arg\min_{\mathbf{z} \in U} \frac{1}{2}\|\mathbf{z} - (\mathbf{z}_0 - \gamma \eta)\|_2^2,
\end{aligned}$$

by the first order optimality condition, we have

$$(\mathbf{z} - \mathbf{z}_h)^\top (\gamma \xi - \mathbf{z}_0 + \mathbf{z}_h) \ge 0, \forall \mathbf{z} \in U, \tag{23}$$

$$(\mathbf{z} - \mathbf{z}_1)^\top (\gamma \eta - \mathbf{z}_0 + \mathbf{z}_1) \ge 0, \forall \mathbf{z} \in U. \tag{24}$$

Applying (23) with $\mathbf{z} = \mathbf{z}_1$ and (24) with $\mathbf{z} = \mathbf{z}_h$, we get

$$\begin{aligned}
\gamma (\mathbf{z}_h - \mathbf{z}_1)^\top \xi &\le (\mathbf{z}_0 - \mathbf{z}_h)^\top (\mathbf{z}_h - \mathbf{z}_1), \\
\gamma (\mathbf{z}_1 - \mathbf{z}_h)^\top \eta &\le (\mathbf{z}_0 - \mathbf{z}_1)^\top (\mathbf{z}_1 - \mathbf{z}_h).
\end{aligned}$$

Summing up the two inequalities, we have

$$\gamma (\mathbf{z}_h - \mathbf{z}_1)^\top (\xi - \eta) \le (\mathbf{z}_1 - \mathbf{z}_h)^\top (\mathbf{z}_h - \mathbf{z}_1) = -\|\mathbf{z}_1 - \mathbf{z}_h\|_2^2.$$

Then

$$\gamma \|\xi - \eta\|_2 \|\mathbf{z}_h - \mathbf{z}_1\|_2 \ge -\gamma (\mathbf{z}_h - \mathbf{z}_1)^\top (\xi - \eta) \ge \|\mathbf{z}_1 - \mathbf{z}_h\|_2^2. \tag{25}$$

We continue the proof as follows:

$$\begin{aligned}
&\frac{1}{2}\|\mathbf{z}-\mathbf{z}_0\|_2^2-\frac{1}{2}\|\mathbf{z}-\mathbf{z}_1\|_2^2\\
&\quad=\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0+(\mathbf{z}-\mathbf{z}_1)^\top(\mathbf{z}_1-\mathbf{z}_0)\\
&\quad=\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0+(\mathbf{z}-\mathbf{z}_1)^\top(\gamma\eta+\mathbf{z}_1-\mathbf{z}_0)-(\mathbf{z}-\mathbf{z}_1)^\top\gamma\eta\\
&\quad\geq\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0-(\mathbf{z}-\mathbf{z}_1)^\top\gamma\eta\\
&\quad=\underbrace{\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma\eta}_{\epsilon}+(\mathbf{z}_h-\mathbf{z})^\top\gamma\eta,
\end{aligned}$$

where the inequality follows (24).

$$\begin{aligned}
\epsilon&=\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma\eta\\
&=\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma(\eta-\xi)-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma\xi\\
&=\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma(\eta-\xi)\\
&\quad+(\mathbf{z}_1-\mathbf{z}_h)^\top(\gamma\xi-\mathbf{z}_0+\mathbf{z}_h)-(\mathbf{z}_1-\mathbf{z}_h)^\top(\mathbf{z}_h-\mathbf{z}_0)\\
&\geq\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_1-\mathbf{z}_0)^\top\mathbf{z}_0-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma(\eta-\xi)-(\mathbf{z}_1-\mathbf{z}_h)^\top(\mathbf{z}_h-\mathbf{z}_0)\\
&=\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_h-\mathbf{z}_0)^\top\mathbf{z}_0-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma(\eta-\xi)-(\mathbf{z}_1-\mathbf{z}_h)^\top\mathbf{z}_h\\
&=\left[\frac{1}{2}\|\mathbf{z}_1\|_2^2-\frac{1}{2}\|\mathbf{z}_h\|_2^2-(\mathbf{z}_1-\mathbf{z}_h)^\top\mathbf{z}_h\right]+\left[\frac{1}{2}\|\mathbf{z}_h\|_2^2-\frac{1}{2}\|\mathbf{z}_0\|_2^2-(\mathbf{z}_h-\mathbf{z}_0)^\top\mathbf{z}_0\right]\\
&\quad-(\mathbf{z}_h-\mathbf{z}_1)^\top\gamma(\eta-\xi)\\
&\geq\frac{1}{2}\|\mathbf{z}_h-\mathbf{z}_1\|_2^2+\frac{1}{2}\|\mathbf{z}_h-\mathbf{z}_0\|_2^2-\gamma\|\mathbf{z}_h-\mathbf{z}_1\|_2\|\eta-\xi\|_2\\
&\geq\frac{1}{2}\left\{\|\mathbf{z}_h-\mathbf{z}_1\|^2+\|\mathbf{z}_h-\mathbf{z}_0\|^2\right\}-\gamma^2\|\eta-\xi\|_2^2,
\end{aligned}$$

where the first inequality follows (23), and the last inequality follows (25). Combining the above results, we have

$$\gamma(\mathbf{z}_h-\mathbf{z})^\top\eta\leq\frac{1}{2}\|\mathbf{z}-\mathbf{z}_0\|_2^2-\frac{1}{2}\|\mathbf{z}-\mathbf{z}_1\|_2^2+\gamma^2\|\eta-\xi\|_2^2-\frac{1}{2}\{\|\mathbf{z}_h-\mathbf{z}_1\|_2^2+\|\mathbf{z}_h-\mathbf{z}_0\|_2^2\}.$$

□

Appendix 5: proof of lemma 6

By introducing Lagrangian multiplier for constraint $\sum_i \alpha_i v_i \leq \rho$, we have the following min-max problem

$$\max_{\eta}\min_{\boldsymbol{\alpha}\in[0,1]^n}\frac{1}{2}\|\boldsymbol{\alpha}-\widehat{\boldsymbol{\alpha}}\|^2+\eta\left(\sum_i\alpha_i v_i-\rho\right).$$

The solution to $\boldsymbol{\alpha}$ is $\alpha_i = [\widehat{\alpha}_i - \eta^* v_i]_{[0,1]}$. By KKT condition, the optimal η^* is equal to 0 if $\sum_i [\widehat{\boldsymbol{\alpha}}_i]_{[0,1]} v_i < \rho$, otherwise we have

$$\sum_i [\widehat{\alpha}_i - \eta^* v_i]_{[0,1]} v_i - \rho = 0.$$

Since the left side of above equation is a monotonically decreasing function in η^*, we can compute η^* by efficient bi-section search.

Appendix 6: proof of lemma 7

Using the convex conjugate $V_*(\eta)$ of $V(z)$, the composite mapping can be written as

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \widehat{\mathbf{w}}\|_2^2 + \lambda \max_{\eta} \left(\eta\|\mathbf{w}\| - V_*(\eta)\right).$$

The problem is equivalent to maximize the following function on η,

$$\left(\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \widehat{\mathbf{w}}\|_2^2 + \lambda\eta\|\mathbf{w}\|\right) - \lambda V_*(\eta).$$

Let $\mathbf{w}(\eta)$ denote the solution to the minimization problem. Then the optimal solution of η satisfies

$$\lambda\|\mathbf{w}(\eta)\| - \lambda V_*'(\eta) = 0,$$

i.e.

$$\|\mathbf{w}(\eta)\| - V_*'(\eta) = 0.$$

It is easy to show that $\|\mathbf{w}(\eta)\|$ is a non-increasing function in η. Similarly, since $V_*(\eta)$ is a convex function, its negative gradient $-V_*'(\eta)$ is a non-increasing function. Therefore, we can compute the optimal η by bi-section search.

References

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*, 243–272.

Bach, F., Jenatton, R., & Mairal, J. (2011). *Optimization with sparsity-inducing penalties (foundations and trends(R) in machine learning)*. Hanover, MA: Now Publishers Inc.

Bartlett, P. L., & Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *JMLR*, *9*, 1823–1840.

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

Radu loan Bot Ernö Robert Csetnek, A.H. (2012). A primal-dual splitting algorithm for finding zeros of sums of maximally monotone operators. ArXiv e-prints.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundation and Trends in Machine Learning*, *3*, 1–122.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York: Cambridge University Press.

Bredies, K. (2009). A forward-backward splitting algorithm for the minimization of non-smooth convex functionals in banach space. Inverse Problems 25, Article ID 015,005, p 20.

Cai, Y., Sun, Y., Cheng, Y., Li, J., Goodison, S. (2010). Fast implementation of l1 regularized learning algorithms using gradient descent methods. In *SDM*, pp. 862–871.

Candès, E.J., Recht, B. (2008). Exact matrix completion via convex optimization. CoRR abs/0805.4471.

Chambolle, A., & Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, *40*, 120–145.
Chen, X., Pan, W., Kwok, J.T., Carbonell, J.G. (2009). Accelerated gradient method for multitasksparse learning problem. In ICDM, pp. 746–751.
Combettes, P.L., Pesquet, J.C.: Primal-dual splitting algorithm for solving inclusions with mixtures of composite, Lipschitzian, and parallel-sum monotone operators. http://hal.inria.fr/hal-00643381.
Dekel, O., Singer, Y. (2006). Support vector machines on a budget. In *NIPS*, pp. 345–352.
Duchi, J., Shalev-Shwartz, S., Singer, Y., Chandra, T. (2008). Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pp. 272–279.
Duchi, J., & Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *JMLR*, *10*, 2899–2934.
Esser, E., Zhang, X., & Chan, T. F. (2010). A general framework for a class of first order Primal-Dual algorithms for convex optimization in imaging science. *SIAM Journal of Imaging Sciences*, *3*, 1015–1046.
Fung, G., Mangasarian, O.L. (2002). A feature selection newton method for support vector machine classification. Technical report, Computational Optimization and Applications.
Gneiting, T. (2008). Quantiles as optimal point predictors. Technical report: Department of Statistics, University of Washington.
Hastie, T., Tibshirani, R., & Friedman, J. (2008). *The elements of statistical learning: Data mining, inference and prediction*. Heidelberg: Springer.
He, B., & Yuan, X. (2012). Convergence analysis of primal-dual algorithms for a saddle-point problem: From contraction perspective. *SIAM Journal on Imaging Science*, *5*, 119–149.
Hou, C., Nie, F., Yi, D., Wu, Y. (2011). Feature selection via joint embedding learning and sparse regression. In *IJCAI*, pp. 1324–1329.
Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear svm. In *ICML*, pp. 408–415.
Hu, C., Kwok, J., Pan, W. (2009). Accelerated gradient methods for stochastic optimization and online learning. In NIPS, pp. 781–789.
Huang, K., Jin, R., Xu, Z., Liu, C.L. (2010) Robust metric learning by smooth optimization. In *UAI*, pp. 244–251.
Ji, S., Ye, J. (2009). An accelerated gradient method for trace norm minimization. In *ICML*, pp. 457–464.
Joachims, T. (1999). Making large-scale support vector machine learning practical. In *Advances in Kernel methods: Support vector, learning*, pp. 169–184.
Joachims, T. (2006). Training linear svms in linear time. In *KDD*, pp. 217–226.
Koenker, R. (2005). *Quantile regression*. Cambridge: Cambridge University Press.
Lan, G. (2010). An optimal method for stochastic composite optimization. *Mathematical Programming*, *133*, 365–397.
Lan, G., Lu, Z., & Monteiro, R. D. C. (2011). Primal-dual first-order methods with 1/epsilon iteration-complexity for cone programming. *Mathematical Programming*, *126*, 1–29.
Lin, Q. (2010). A smoothing stochastic gradient method for composite optimization. ArXiv e-prints.
Lions, P. L., & Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *Siam Journal on Numerical Analysis*, *16*, 964–979.
Liu, J., Ji, S., Ye, J. (2009). Multi-task feature learning via efficient l2, 1-norm minimization. In *UAI*, pp. 339–348.
Mosci, S., Villa, S., Verri, A., Rosasco, L. (2010). A primal-dual algorithm for group sparse regularization with overlapping groups. In *NIPS*, pp. 2604–2612.
Nemirovski, A. (2005). Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, *15*, 229–251.
Nesterov, Y. (2005). Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, *16*, 235–249.
Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, *103*, 127–152.
Nesterov, Y. (2007). Gradient methods for minimizing composite objective function. Core discussion papers.
Nie, F., Huang, H., Cai, X., & Ding, C. (2010). Efficient and robust feature selection via joint 2,1-norms minimization. *Advances in Neural Information Processing Systems*, *23*, 1813–1821.
Platt, J.C. (1998). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel methods: Support vector learning*, pp. 185–208. Cambridge, MA.
Pock, T., Chambolle, A. (2011). Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *Proceedings of the 2011 International Conference on Computer Vision*, pp. 1762–1769.

Popov, L. (1980). A modification of the arrow-hurwitz method of search for saddle points. *Matematicheskie Zametki*, *28*, 777–784.
Quattoni, A., Carreras, X., Collins, M., Darrell, T. (2009). An efficient projection for l1, infinity regularization. In *ICML*, pp. 857–864.
Recht, B., Fazel, M., & Parrilo, P. A. (2010). Guaranteed Minimum-Rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, *52*, 471–501.
Rennie, J.D.M., Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pp. 713–719.
Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, *14*, 877–898.
Rosasco, L., De Vito, E., Caponnetto, A., Piana, M., & Verri, A. (2004). Are loss functions all the same? *Neural Computation*, *16*, 1063–1076.
Shalev-Shwartz, S., Singer, Y., Srebro, N., & Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, *127*(1), 3–30.
Smale, S., & Zhou, D. X. (2003). Estimating the approximation error in learning theory. *Applied Analysis (Singapore)*, *1*(1), 17–41.
Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222.
Srebro, N., Rennie, J.D.M., Jaakkola, T.S. (2005). Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pp. 1329–1336.
Sun, L., Liu, J., Chen, J., & Ye, J. (2009). Efficient recovery of jointly sparse vectors. *Advances in Neural Information Processing Systems*, *22*, 1812–1820.
Traub, J. F., Wasilkowski, G. W., & Woźniakowski, H. (1988). *Information-based complexity*. San Diego: Academic Press Professional Inc.
Tseng, P. (2008). On accelerated proximal gradient methods for convex–concave optimization. Technical report.
Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley-Interscience.
Wu, Q., & Zhou, D. X. (2005). Svm soft margin classifiers: Linear programming versus quadratic programming. *Neural Computation*, *17*, 1160–1187.
Xiao, L. (2009). Dual averaging method for regularized stochastic learning and online optimization. In: Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, A. Culotta (eds.) NIPS, pp. 2116–2124.
Yang, H., Xu, Z., King, I., Lyu, M.R. (2010) Online learning for group lasso. In *ICML*, pp. 1191–1198.
Yang, T., Mahdavi, M., Jin, R., Zhang, L., Zhou, Y. (2012). Multiple kernel learning from noisy labels by stochastic programming. In *ICML*.
Yeo, G., Burge, C.B. (2003). Maximum entropy modeling of short sequence motifs with applications to rna splicing signals. In *RECOMB*, pp. 322–331.
Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *JRSS*, *68*, 49–67.
Zhou, T., Tao, D., Wu, X. (2010). Nesvm: A fast gradient method for support vector machines. CoRR abs/1008.4000.
Zhou, Y., Jin, R., Hoi, S.C. (2010). Exclusive lasso for multi-task feature selection. In *AISTAT*, pp. 988–995.
Zhu, J., Rosset, S., Hastie, T., Tibshirani, R. (2003). 1-norm support vector machines. In *NIPS*.
Zhu, M., Chan, T. (2008). An efficient primal-dual hybrid gradient algorithm for total variation image restoration. UCLA CAM Report pp. 08–34.