Journal of Machine Learning Research 15 (2014) 1371-1429 Submitted 5/12; Revised 1/14; Published 4/14

Towards Ultrahigh Dimensional Feature Selection for Big Data

Mingkui Tan TANMINGKUI@GMAIL.COM
School of Computer Engineering
Nanyang Technological University
Blk N4, #B1a-02, Nanyang Avenue 639798, Singapore

Ivor W. Tsang IVOR.TSANG@GMAIL.COM
Center for Quantum Computation & Intelligent Systems
University of Technology Sydney, Sydney P.O. Box 123,
Broadway, NSW 2007 Sydney, Australia

Li Wang LIWANGUCSD@GMAIL.COM
Department of Mathematics, University of California
San Diego 9500 Gilman Drive La Jolla, CA 92093, USA

Editor: Sathiya Keerthi

Abstract

In this paper, we present a new adaptive feature scaling scheme for ultrahigh-dimensional feature selection on *Big Data*, and then reformulate it as a convex semi-infinite programming (SIP) problem. To address the SIP, we propose an efficient *feature generating paradigm*. Different from traditional gradient-based approaches that conduct optimization on all input features, the proposed paradigm iteratively activates a group of features, and solves a sequence of multiple kernel learning (MKL) subproblems. To further speed up the training, we propose to solve the MKL subproblems in their primal forms through a modified accelerated proximal gradient approach. Due to such optimization scheme, some efficient cache techniques are also developed. The feature generating paradigm is guaranteed to converge globally under mild conditions, and can achieve lower feature selection bias. Moreover, the proposed method can tackle two challenging tasks in feature selection: 1) group-based feature selection with complex structures, and 2) nonlinear feature selection with explicit feature mappings. Comprehensive experiments on a wide range of synthetic and real-world data sets of tens of million data points with $O(10^{14})$ features demonstrate the competitive performance of the proposed method over state-of-the-art feature selection methods in terms of generalization performance and training efficiency.

Keywords: big data, ultrahigh dimensionality, feature selection, nonlinear feature selection, multiple kernel learning, feature generation

1. Introduction

With the rapid development of the Internet, *big data* of large volume and ultrahigh dimensionality have emerged in various machine learning applications, such as text mining and information retrieval (Deng et al., 2011; Li et al., 2011, 2012). For instance, Weinberger et al. (2009) have studied a collaborative email-spam filtering task with 16 trillion (10^{13}) unique features. The ultrahigh dimensionality not only incurs unbearable memory requirements and high computational cost in training, but also deteriorates the generalization ability because of the "curse of dimensionality" issue (Duda et al., 2000.; Guyon and Elisseeff, 2003; Zhang and Lee, 2006; Dasgupta et al., 2007; Blum et al., 2007). Fortunately, for many data sets with ultrahigh dimensions, most of the features are irrelevant to the output. Accordingly, dropping the irrelevant features and selecting the most relevant features can vastly improve the generalization performance (Ng, 1998). Moreover, in many applications such as bioinformatics (Guyon and Elisseeff, 2003), a small number of features (genes) are required to interpret the results for further biological analysis. Finally, for ultrahigh-dimensional problems, a sparse classifier is important for faster predictions.

Ultrahigh dimensional data also widely appear in many nonlinear machine learning tasks. For example, to tackle the intrinsic nonlinearity of data, researchers proposed to achieve fast training and prediction through linear techniques using explicit feature mappings (Chang et al., 2010; Maji and Berg, 2009). However, most of the explicit feature mappings will dramatically expand the data dimensionality. For instance, the commonly used 2-degree polynomial kernel feature mapping has a dimensionality of $O(m^2)$, where m denotes the number of input features (Chang et al., 2010). Even with a medium m, the dimensionality of the induced feature space is very huge. Other typical feature mappings include the spectrum-based feature mapping for string kernel (Sonnenburg et al., 2007; Sculley et al., 2006), histogram intersection kernel feature expansion (Wu, 2012), and so on.

Numerous feature selection methods have been proposed for classification tasks in the last decades (Guyon et al., 2002; Chapelle and Keerthi, 2008). In general, existing feature selection methods can be classified into two categories, namely filter methods and wrapper methods (Kohavi and John, 1997; Ng, 1998; Guyon et al., 2002). Filter methods, such as the signal-to-noise ratio method (Golub et al., 1999) and the spectral feature filtering (Zhao and Liu, 2007), own the advantages of low computational cost, but they are incapable of finding an optimal feature subset w.r.t. a predictive model of interest. On the contrary, by incorporating the inductive learning rules, wrapper methods can select more relevant features (Xu et al., 2009a; Guyon and Elisseeff, 2003). However, in general, the wrapper methods are more computationally expensive than the filter methods. Accordingly, how to scale the wrapper methods to big data is an urgent and challenging issue, and is also a major focus of this paper.

One of the most famous wrapper methods is the support vector machine (SVM) based recursive feature elimination (SVM-RFE), which has shown promising performance in the Microarray data analysis, such as gene selection task (Guyon et al., 2002). Specifically, SVM-RFE applys a recursive feature elimination scheme, and obtains nested subsets of features based on the weights of SVM classifiers. Unfortunately, the nested feature selection strategy is "monotonic" and suboptimal in identifying the most informative feature subset (Xu et al., 2009a; Tan et al., 2010). To address this drawback, non-monotonic feature selection methods have gained great attention (Xu et al., 2009a; Chan et al., 2007). Basically, the non-monotonic feature selection requires the convexity of the objective in order to easily find a global solution. To this end, Chan et al. (2007) proposed two convex relaxations to an ℓ_0-norm sparse SVM, namely QSSVM and SDP-SSVM. The resultant models are convex, and can be solved by the convex quadratically constrained quadratic programming (QCQP) and the semi-definite programming (SDP), respectively. These two methods belong to the non-monotonic feature selection methods. However, they are very expensive especially for high dimensional problems. Xu *et al.* proposed another non-monotonic feature selection method, namely NMMKL (Xu et al., 2009a). Unfortunately, NMMKL is computationally infeasible for high dimensional problems since it involves a QCQP problem with many quadratic constraints.

Focusing on the logistic loss, recently, some researchers proposed to select features using greedy strategies (Tewari et al., 2011; Lozano et al., 2011), which iteratively include one feature into a feature subset. For example, Lozano et al. (2011) proposed a group orthogonal matching pursuit. Tewari et al. (2011) further introduced a general greedy scheme to solve more general sparsity constrained problems. Although promising performance has been observed, the greedy methods have several drawbacks. For example, since only one feature is involved in each iteration, these greedy methods are very expensive when there are a large number of features to be selected. More critically, due to the absence of appropriate regularizer in the objective function, the over-fitting problem may happen, which may deteriorate the generalization performance (Lozano et al., 2011; Tewari et al., 2011).

Given a set of labeled patterns $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^m$ is an instance with m features, and $y_i \in \{\pm 1\}$ is the output label. To avoid the over-fitting problem or induce sparsity, people usually introduce some regularizers to the loss function. For instance, to select features that contribute the most to the margin, we can learn a sparse decision function $d(\mathbf{x}) = \mathbf{w}'\mathbf{x}$ by solving:

$$\min_{\mathbf{w}} \quad \|\mathbf{w}\|_0 + C\sum_{i=1}^{n} l(-y_i\mathbf{w}'\mathbf{x}_i),$$

where $l(\cdot)$ is a convex loss function, $\mathbf{w} \in \mathbb{R}^m$ is the weight vector, $\|\mathbf{w}\|_0$ denotes the ℓ_0-norm that counts the number of non-zeros in $\mathbf{w}$, and $C > 0$ is a regularization parameter. Unfortunately, this problem is NP-hard due to the ℓ_0-norm regularizer. Therefore, many researchers resort to learning a sparse decision rule through an ℓ_1-convex relaxation instead (Bradley and Mangasarian, 1998; Zhu et al., 2003; Fung and Mangasarian, 2004):

$$\min_{\mathbf{w}} \quad \|\mathbf{w}\|_1 + C\sum_{i=1}^{n} l(-y_i\mathbf{w}'\mathbf{x}_i), \tag{1}$$

where $\|\mathbf{w}\|_1 = \sum_{j=1}^m |w_j|$ is the ℓ_1-norm on $\mathbf{w}$. The ℓ_1-regularized problem can be efficiently solved, and many optimization methods have been proposed to solve this problem, including Newton methods (Fung and Mangasarian, 2004), proximal gradient methods (Yuan et al., 2011), coordinate descent methods (Yuan et al., 2010, 2011), and so on. Interested readers can find more details of these methods in (Yuan et al., 2010, 2011) and references therein. Beside these methods, recently, to address the big data challenge, great attention has been paid on online learning methods and stochastic gradient descent (SGD) methods for dealing with big data challenges (Xiao, 2009; Duchi and Singer, 2009; Langford et al., 2009; Shalev-Shwartz and Zhang, 2013).

However, there are several deficiencies regarding these ℓ_1-norm regularized model and existing ℓ_1-norm methods. Firstly, since the ℓ_1-norm regularization shrinks the regressors, the feature selection bias inevitably exists in the ℓ_1-norm methods (Zhang and Huang, 2008; Zhang, 2010b; Lin et al., 2010; Zhang, 2010a). To demonstrate this, let $L(\mathbf{w}) =$

$\sum_{i=1}^{n} l(-y_i \mathbf{w}'\mathbf{x}_i)$ be the empirical loss on the training data, then $\mathbf{w}^*$ is an optimal solution to (1) if and only if it satisfies the following optimality conditions (Yuan et al., 2010):

$$\begin{cases} \nabla_j L(\mathbf{w}^*) = -1/C & \text{if } \ w_j^* > 0, \\ \nabla_j L(\mathbf{w}^*) = 1/C & \text{if } \ w_j^* < 0, \\ -1/C \leq \nabla_j L(\mathbf{w}^*) \leq 1/C & \text{if } \ w_j^* = 0. \end{cases} \quad (2)$$

According to the above conditions, one can achieve different levels of sparsity by changing the regularization parameter C. On one hand, using a small C, minimizing $\|\mathbf{w}\|_1$ in (1) would favor selecting only a few features. The sparser the solution is, the larger the predictive risk (or empirical loss) will be (Lin et al., 2010). In other words, the solution bias will happen (Figueiredo et al., 2007). In an extreme case, where C is chosen to be tiny or even close to zero, none of the features will be selected according to the condition (2), which will lead to a very poor prediction model. On the other hand, using a large C, one can learn an more fitted prediction model to to reduce the empirical loss. However, according to (2), more features will be included. In summary, the sparsity and the unbiased solutions cannot be achieved simultaneously via solving (1) by changing the tradeoff parameter C. A possible solution is to do de-biasing with the selected features using re-training. For example, we can use a large C to train an unbiased model with the selected features (Figueiredo et al., 2007; Zhang, 2010b). However, such de-biasing methods are not efficient.

Secondly, when tackling big data of ultrahigh dimensions, the ℓ_1-regularization would be inefficient or infeasible for most of the existing methods. For example, when the dimensionality is around 10^{12}, one needs about 1 TB memory to store the weight vector $\mathbf{w}$, which is intractable for existing ℓ_1-methods, including online learning methods and SGD methods (Langford et al., 2009; Shalev-Shwartz and Zhang, 2013). Thirdly, due to the scale variation of $\mathbf{w}$, it is also non-trivial to control the number of features to be selected meanwhile to regulate the decision function.

In the conference version of this paper, an ℓ_0-norm sparse SVM model is introduced (Tan et al., 2010). Its nice optimization scheme has brought significant benefits to several applications, such as image retrieval (Rastegari et al., 2011), multi-label prediction (Gu et al., 2011a), feature selection for multivariate performance measures (Mao and Tsang, 2013), feature selection for logistic regression (Tan et al., 2013), and graph-based feature selection (Gu et al., 2011b). However, several issues remain to be solved. First of all, the tightness of the convex relation remains unclear. Secondly, the adopted optimization strategy is incapable of dealing with very large-scale problems with many training instances. Thirdly, the presented feature selection strategy was limited to linear features, but in many applications, one indeed needs to tackle nonlinear features that are with complex structures.

Regarding the above issues, in this paper, we propose an adaptive feature scaling (AFS) for feature selection by introducing a continuous feature scaling vector $\mathbf{d} \in [0,1]^m$. To enforce the sparsity, we impose an explicit ℓ_1-constraint $||\mathbf{d}||_1 \leq B$, where the scalar B represents the least number of features to be selected. The solution to the resultant optimization problem is non-trivial due to the additional constraint. Fortunately, by transforming it as a convex semi-infinite programming (SIP) problem, an efficient optimization scheme can be developed. In summary, this paper makes the following extensions and improvements.

- A feature generating machine (FGM) is proposed to efficiently address the ultrahigh-dimensional feature selection task through solving the proposed SIP problem. Instead of performing the optimization on all input features, FGM iteratively infers the most informative features, and then solves a reduced multiple kernel learning (MKL) subproblem, where each base kernel is defined on a set of features.[1]

- The proposed optimization scheme mimics the re-training strategy to reduce the feature selection bias with little effort. Specifically, the feature selection bias can be effectively alleviated by separately controlling the complexity and sparsity of the decision function, which is one of the major advantages of the proposed scheme.

- To speed up the training on big data, we propose to solve the primal form of the MKL subproblem by a modified accelerated proximal gradient method. As a result, the memory requirement and computational cost can be significantly reduced. The convergence rate of the modified APG is also provided. Moreover, several cache techniques are proposed to further enhance the efficiency.

- The feature generating paradigm is also extended to group feature selection with complex group structures and nonlinear feature selection using explicit feature mappings.

The remainder of this paper is organized as follows. In Section 2, we start by presenting the adaptive feature scalings (AFS) for linear feature selection and group feature selection, and then present the convex SIP reformulations of the resultant optimization problems. After that, to solve the SIP problems, in Section 3, we propose the feature generating machine (FGM) which includes two core steps, namely the worst-case analysis step and the subproblem optimization step. In Section 4, we illustrate the worst-case analysis for a number of learning tasks, including the group feature selection with complex group structures and the nonlinear feature selection with explicit feature mappings. We introduce the subproblem optimization in Section 5 and extend FGM for multiple kernel learning w.r.t. many additive kernels in Section 6. Related studies are presented in Section 7. We conduct empirical studies in Section 8, and conclude this work in Section 9.

2. Feature Selection Through Adaptive Feature Scaling

Throughout the paper, we denote the transpose of vector/matrix by the superscript $'$, a vector with all entries equal to one as $\mathbf{1} \in \mathbb{R}^n$, and the zero vector as $\mathbf{0} \in \mathbb{R}^n$. In addition, we denote a data set by $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]' = [\mathbf{f}^1, ..., \mathbf{f}^m]$, where $\mathbf{x}_i \in \mathbb{R}^m$ represents the ith instance and $\mathbf{f}^j \in \mathbb{R}^n$ denotes the jth feature vector. We use $|\mathcal{G}|$ to denote cardinality of an index set $\mathcal{G}$ and $|v|$ to denote the absolute value of a real number v. For simplicity, we denote $\mathbf{v} \succeq \boldsymbol{\alpha}$ if $v_i \geq \alpha_i, \forall i$ and $\mathbf{v} \preceq \boldsymbol{\alpha}$ if $v_i \leq \alpha_i, \forall i$. We also denote $\|\mathbf{v}\|_p$ as the ℓ_p-norm of a vector and $\|\mathbf{v}\|$ as the ℓ_2-norm of a vector. Given a vector $\mathbf{v} = [\mathbf{v}_1', ..., \mathbf{v}_p']'$, where $\mathbf{v}_i$ denotes a sub-vector of $\mathbf{v}$, we denote $\|\mathbf{v}\|_{2,1} = \sum_{i=1}^p \|\mathbf{v}_i\|$ as the mixed ℓ_1/ℓ_2 norm (Bach et al., 2011) and $\|\mathbf{v}\|_{2,1}^2 = (\sum_{i=1}^p \|\mathbf{v}_i\|)^2$. Accordingly, we call $\|\mathbf{v}\|_{2,1}^2$ as an $\ell_{2,1}^2$ regularizer. Following Rakotomamonjy et al. (2008), we define $\frac{x_i}{0} = 0$ if $x_i = 0$ and ∞ otherwise. Finally, $\mathbf{A} \odot \mathbf{B}$ represents the element-wise product between two matrices $\mathbf{A}$ and $\mathbf{B}$.

[1] The C++ and MATLAB source codes of the proposed methods are publicly available at `http://www.tanmingkui.com/fgm.html`.

2.1 A New AFS Scheme for Feature Selection

In the standard support vector machines (SVM), one learns a linear decision function $d(\mathbf{x}) = \mathbf{w}'\mathbf{x} - b$ by solving the following ℓ_2-norm regularized problem:

$$\min_{\mathbf{w}} \ \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} l(-y_i(\mathbf{w}'\mathbf{x}_i - b)), \tag{3}$$

where $\mathbf{w} = [w_1, \ldots, w_m]' \in \mathbb{R}^m$ denotes the weight of the decision hyperplane, b denotes the shift from the origin, $C > 0$ represents the regularization parameter and $l(\cdot)$ denotes a convex loss function. In this paper, we concentrate on two kinds of loss functions, namely the squared hinge loss

$$l(-y_i(\mathbf{w}'\mathbf{x}_i - b)) = \frac{1}{2}\max(1 - y_i(\mathbf{w}'\mathbf{x}_i - b), 0)^2$$

and the logistic loss

$$l(-y_i(\mathbf{w}'\mathbf{x}_i - b)) = \log(1 + \exp(-y_i(\mathbf{w}'\mathbf{x}_i - b))).$$

For simplicity, herein we concentrate the squared hinge loss only.

In (3), the ℓ_2-regularizer $||\mathbf{w}||^2$ is used to avoid the over-fitting problem (Hsieh et al., 2008), which, however, cannot induce sparse solutions. To address this issue, we introduce a feature scaling vector $\mathbf{d} \in [0, 1]^m$ such that we can scale the importance of features. Specifically, given an instance $\mathbf{x}_i$, we impose $\sqrt{\mathbf{d}} = [\sqrt{d_1}, \ldots, \sqrt{d_m}]'$ on its features (Vishwanathan et al., 2010), resulting in a re-scaled instance

$$\widehat{\mathbf{x}}_i = (\mathbf{x}_i \odot \sqrt{\mathbf{d}}). \tag{4}$$

In this scaling scheme, the jth feature is selected if and only if $d_j > 0$.

Note that, in many real-world applications, one may intend to select a desired number of features with acceptable generalization performance. For example, in the Microarray data analysis, due to expensive bio-diagnosis and limited resources, biologists prefer to select less than 100 genes from hundreds of thousands of genes (Guyon et al., 2002; Golub et al., 1999). To incorporate such prior knowledge, we explicitly impose an ℓ_1-norm constraint on $\mathbf{d}$ to induce the sparsity:

$$\sum_{j=1}^{m} d_j = ||\mathbf{d}||_1 \leq B, \ d_j \in [0, 1], \ j = 1, \cdots, m,$$

where the integer B represents the least number of features to be selected. Similar feature scaling scheme has been used by many works (e.g., Weston et al., 2000; Chapelle et al., 2002; Grandvalet and Canu, 2002; Rakotomamonjy, 2003; Varma and Babu, 2009; Vishwanathan et al., 2010). However, different from the proposed scaling scheme, in these scaling schemes, $\mathbf{d}$ is not bounded in $[0, \ 1]^m$.

Let $\mathcal{D} = \left\{\mathbf{d} \in \mathbb{R}^m \middle| \sum_{j=1}^m d_j \leq B, \ d_j \in [0, 1], \ j = 1, \cdots, m\right\}$ be the domain of $\mathbf{d}$, the proposed AFS can be formulated as the following problem:

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{D}} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) \geq 1 - \xi_i, \quad i = 1, \cdots, n, \end{aligned} \tag{5}$$

where C is a regularization parameter that trades off between the model complexity and the fitness of the decision function, and $b/\|\mathbf{w}\|$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$. This problem is non-convex w.r.t. $\mathbf{w}$ and $\mathbf{d}$ simultaneously, and the compact domain $\mathcal{D}$ contains infinite number of elements. However, for a fixed $\mathbf{d}$, the inner minimization problem w.r.t. $\mathbf{w}$ and $\boldsymbol{\xi}$ is a standard SVM problem:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) \geq 1 - \xi_i, \quad i = 1, \cdots, n, \end{aligned} \tag{6}$$

which can be solved in its dual form. By introducing the Lagrangian multiplier $\alpha_i \geq 0$ to each constraint $y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) \geq 1 - \xi_i$, the Lagrangian function is:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}\alpha_i\left(y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) - 1 + \xi_i\right). \tag{7}$$

By setting the derivatives of $\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha})$ w.r.t. $\mathbf{w}$, $\boldsymbol{\xi}$ and b to $\mathbf{0}$, respectively, we get

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}}), \boldsymbol{\alpha} = C\boldsymbol{\xi}, \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0. \tag{8}$$

Substitute these results into (7), and we arrive at the dual form of problem (6) as:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -\frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + \boldsymbol{\alpha}'\mathbf{1},$$

where $\mathcal{A} = \{\boldsymbol{\alpha} | \sum_{i=1}^{n}\alpha_i y_i = 0, \boldsymbol{\alpha} \succeq \mathbf{0}\}$ is the domain of $\boldsymbol{\alpha}$. For convenience, let $\mathbf{c}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \in \mathbb{R}^m$, we have $\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 = \sum_{j=1}^{m} d_j[c_j(\boldsymbol{\alpha})]^2$, where the jth coordinate of $\mathbf{c}(\boldsymbol{\alpha})$, namely $c_j(\boldsymbol{\alpha})$, is a function of $\boldsymbol{\alpha}$. For simplicity, let

$$f(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}\sum_{j=1}^{m} d_j[c_j(\boldsymbol{\alpha})]^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}.$$

Apparently, $f(\boldsymbol{\alpha}, \mathbf{d})$ is linear in $\mathbf{d}$ and concave in $\boldsymbol{\alpha}$, and both $\mathcal{A}$ and $\mathcal{D}$ are compact domains. Problem (5) can be equivalently reformulated as the following problem:

$$\min_{\mathbf{d}\in\mathcal{D}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -f(\boldsymbol{\alpha}, \mathbf{d}), \tag{9}$$

However, this problem is still difficult to be addressed. Recall that both $\mathcal{A}$ and $\mathcal{D}$ are convex compact sets, according to the minimax saddle-point theorem (Sion, 1958), we immediately have the following relation.

Theorem 1 *According to the minimax saddle-point theorem (Sion, 1958), the following equality holds by interchanging the order of* $\min_{\mathbf{d}\in\mathcal{D}}$ *and* $\max_{\boldsymbol{\alpha}\in\mathcal{A}}$ *in (9),*

$$\min_{\mathbf{d}\in\mathcal{D}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -f(\boldsymbol{\alpha}, \mathbf{d}) = \max_{\boldsymbol{\alpha}\in\mathcal{A}}\min_{\mathbf{d}\in\mathcal{D}} \quad -f(\boldsymbol{\alpha}, \mathbf{d}).$$

Based on the above equivalence, rather than solving the original problem in (9), hereafter we address the following minimax problem instead:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{D}} \quad f(\boldsymbol{\alpha}, \mathbf{d}). \tag{10}$$

2.2 AFS for Group Feature Selection

The above AFS scheme for linear feature selection can be extended for group feature selections, where the features are organized into groups defined by $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$, where $\cup_{j=1}^{p}\mathcal{G}_j = \{1, ..., m\}$, $p = |\mathcal{G}|$ denotes the number of groups, and $\mathcal{G}_j \subset \{1, ..., m\}, j = 1, ..., p$ denotes the index set of feature supports belonging to the jth group. In the group feature selection, a feature in one group is selected if and only if this group is selected (Yuan and Lin, 2006; Meier et al., 2008). Let $\mathbf{w}_{\mathcal{G}_j} \in \mathbb{R}^{|\mathcal{G}_j|}$ and $\mathbf{x}_{\mathcal{G}_j} \in \mathbb{R}^{|\mathcal{G}_j|}$ be the components of $\mathbf{w}$ and $\mathbf{x}$ related to $\mathcal{G}_j$, respectively. The group feature selection can be achieved by solving the following non-smooth group lasso problem (Yuan and Lin, 2006; Meier et al., 2008):

$$\min_{\mathbf{w}} \quad \lambda \sum_{j=1}^{p} ||\mathbf{w}_{\mathcal{G}_j}||_2 + \sum_{i=1}^{n} l(-y_i \sum_{j=1}^{p} \mathbf{w}'_{\mathcal{G}_j}\mathbf{x}_{i\mathcal{G}_j}), \tag{11}$$

where λ is a trade-off parameter. Many efficient algorithms have been proposed to solve this problem, such as the accelerated proximal gradient descent methods (Liu and Ye, 2010; Jenatton et al., 2011b; Bach et al., 2011), block coordinate descent methods (Qin et al., 2010; Jenatton et al., 2011b) and active set methods (Bach, 2009; Roth and Fischer, 2008). However, the issues of the ℓ_1-regularization, namely the scalability issue for big data and the feature selection bias, will also happen when solving (11). More critically, when dealing with feature groups with complex structures, the number of groups can be exponential in the number of features m. As a result, solving (11) could be very expensive.

To extend AFS to group feature selection, we introduce a group scaling vector $\widehat{\mathbf{d}} = [\widehat{d}_1, \ldots, \widehat{d}_p]' \in \widehat{\mathcal{D}}$ to scale the groups, where $\widehat{\mathcal{D}} = \big\{\widehat{\mathbf{d}} \in \mathbb{R}^p \big| \sum_{j=1}^{p} \widehat{d}_j \leq B, \widehat{d}_j \in [0,1],\ j = 1, \cdots, p\big\}$. Here, without loss of generality, we first assume that there is no overlapping element among groups, namely, $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \forall i \neq j$. Accordingly, we have $\mathbf{w} = [\mathbf{w}'_{\mathcal{G}_1}, ..., \mathbf{w}'_{\mathcal{G}_p}]' \in \mathbb{R}^m$. By taking the shift term b into consideration, the decision function is expressed as:

$$d(\mathbf{x}) = \sum_{j=1}^{p} \sqrt{\widehat{d}_j}\mathbf{w}'_{\mathcal{G}_j}\mathbf{x}_{\mathcal{G}_j} - b,$$

By applying the squared hinge loss, the AFS based group feature selection task can be formulated as the following optimization problem:

$$\begin{aligned}
\min_{\widehat{\mathbf{d}}\in\widehat{\mathcal{D}}} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\
\text{s.t.} \quad & y_i\left(\sum_{j=1}^{p}\sqrt{\widehat{d}_j}\mathbf{w}'_{\mathcal{G}_j}\mathbf{x}_{i\mathcal{G}_j} - b\right) \geq 1 - \xi_i, \ \ \xi_i \geq 0, \ \ i = 1, \cdots, n.
\end{aligned}$$

With similar deductions in Section 2.1, this problem can be transformed into the following minimax problem:

$$\min_{\widehat{\mathbf{d}}\in\widehat{\mathcal{D}}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} -\frac{1}{2}\sum_{j=1}^{p}\widehat{d}_j\left\|\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}\right\|^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + \boldsymbol{\alpha}'\mathbf{1}.$$

This problem is reduced to the linear feature selection case if $|\mathcal{G}_j| = 1, \forall j = 1, ..., p$. For convenience, hereafter we drop the hat from $\widehat{\mathbf{d}}$ and $\widehat{\mathcal{D}}$. Let $\mathbf{c}_{\mathcal{G}_j}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}$. Moreover, we define

$$f(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}\sum_{j=1}^{p} d_j \left\|\mathbf{c}_{\mathcal{G}_j}(\boldsymbol{\alpha})\right\|^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}.$$

Finally, we arrive at a unified minimax problem for both linear and group feature selections:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}}\max_{\mathbf{d}\in\mathcal{D}} \quad f(\boldsymbol{\alpha}, \mathbf{d}), \tag{12}$$

where $\mathcal{D} = \left\{\mathbf{d} \in \mathbb{R}^p \middle| \sum_{j=1}^{p} d_j \leq B,\ d_j \in [0, 1],\ j = 1, \cdots, p\right\}$. When $|\mathcal{G}_j| = 1, \forall j = 1, ..., p$, we have $p = m$, and problem (12) is reduced to problem (10).

2.3 Group Feature Selection with Complex Structures

Now we extend the above group AFS scheme to feature groups with overlapping features or even more complex structures. When dealing with groups with overlapping features, a heuristic way is to explicitly augment $\mathbf{X} = [\mathbf{f}^1, ..., \mathbf{f}^m]$ to make the overlapping groups non-overlapping by repeating the overlapping features. For example, suppose $\mathbf{X} = [\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3]$ with groups $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2\}$, where $\mathcal{G}_1 = \{1, 2\}$ and $\mathcal{G}_2 = \{2, 3\}$, and $\mathbf{f}^2$ is an overlapping feature. To avoid the overlapping feature issue, we can repeat $\mathbf{f}^2$ to construct an augmented data set $\mathbf{X}_{au} = [\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^2, \mathbf{f}^3]$, where the group index sets become $\mathcal{G}_1 = \{1, 2\}$ and $\mathcal{G}_2 = \{3, 4\}$. This feature augmentation strategy can be extended to groups with even more complex structures, such as tree structures or graph structures (Bach, 2009). For simplicity, in this paper, we only study the tree-structured groups.

Definition 1 *Tree-structured set of groups (Jenatton et al., 2011b; Kim and Xing, 2010, 2012). A super set of groups* $\mathcal{G} \triangleq \{\mathcal{G}_h\}_{\mathcal{G}_h\in\mathcal{G}}$ *with* $|\mathcal{G}| = p$ *is said to be tree-structured in* $\{1, ..., m\}$*, if* $\cup\mathcal{G}_h = \{1, ..., m\}$ *and if for all* $\mathcal{G}_g, \mathcal{G}_h \in \mathcal{G}$, $(\mathcal{G}_g \cap \mathcal{G}_h \neq \varnothing) \Rightarrow (\mathcal{G}_g \subseteq \mathcal{G}_h \text{ or } \mathcal{G}_h \subseteq \mathcal{G}_g)$*. For such a set of groups, there exists a (non-unique) total order relation* $\preceq$ *such that:*

$$\mathcal{G}_g \preceq \mathcal{G}_h \Rightarrow \{\mathcal{G}_g \subseteq \mathcal{G}_h \textit{ or } \mathcal{G}_g \cap \mathcal{G}_h = \varnothing\}.$$

Similar to the overlapping case, we augment the overlapping elements of all groups along the tree structures, resulting in the augmented data set $\mathbf{X}_{au} = [\mathbf{X}_{\mathcal{G}_1}, ..., \mathbf{X}_{\mathcal{G}_p}]$, where $\mathbf{X}_{\mathcal{G}_i}$ represents the data columns indexed by $\mathcal{G}_i$ and p denotes the number of all possible groups. However, this simple idea may bring great challenges for optimization, particularly when there are huge number of overlapping groups (For instance, in graph-based group structures, the number of groups p can be exponential in m (Bach, 2009)).

3. Feature Generating Machine

Under the proposed AFS scheme, both linear feature selection and group feature selection can be cast as the minimax problem in (12). By bringing in an additional variable $\theta \in \mathbb{R}$, this problem can be further formulated as a semi-infinite programming (SIP) problem (Kelley, 1960; Pee and Royset, 2010):

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}, \theta \in \mathbb{R}} \theta, \quad \text{s.t.} \quad \theta \geq f(\boldsymbol{\alpha}, \mathbf{d}), \quad \forall\ \mathbf{d} \in \mathcal{D}. \tag{13}$$

In (13), each nonzero $\mathbf{d} \in \mathcal{D}$ defines a quadratic constraint w.r.t. $\boldsymbol{\alpha}$. Since there are infinite $\mathbf{d}$'s in $\mathcal{D}$, problem (13) involves infinite number of constraints, thus it is very difficult to be solved.

3.1 Optimization Strategies by Feature Generation

Before solving (13), we first discuss its optimality condition. Specifically, let $\mu_h \geq 0$ be the dual variable for each constraint $\theta \geq f(\boldsymbol{\alpha}, \mathbf{d})$, the Lagrangian function of (13) can be written as:

$$\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \theta - \sum_{\mathbf{d}_h \in \mathcal{D}} \mu_h \left(\theta - f(\boldsymbol{\alpha}, \mathbf{d}_h)\right).$$

By setting its derivative w.r.t. θ to zero, we have $\sum \mu_h = 1$. Let $\mathcal{M} = \{\boldsymbol{\mu} | \sum \mu_h = 1, \mu_h \geq 0, h = 1, ..., |\mathcal{D}|\}$ be the domain of $\boldsymbol{\mu}$ and define

$$f_{\mathrm{m}}(\boldsymbol{\alpha}) = \max_{\mathbf{d}_h \in \mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d}_h).$$

The KKT conditions of (13) can be written as:

$$\sum_{\mathbf{d}_h \in \mathcal{D}} \mu_h \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \mathbf{d}_h) = \mathbf{0}, \qquad \text{and} \qquad \sum_{\mathbf{d}_h \in \mathcal{D}} \mu_h = 1. \tag{14}$$

$$\mu_h(f(\boldsymbol{\alpha}, \mathbf{d}_h) - f_m(\boldsymbol{\alpha})) = 0, \quad \mu_h \geq 0, \quad h = 1, ..., |\mathcal{D}|. \tag{15}$$

In general, there are many constraints in problem (14). However, most of them are nonactive at the optimality if the data contain only a small number of relevant features w.r.t. the output $\mathbf{y}$. Specifically, according to condition (15), we have $\mu_h = 0$ if $f(\boldsymbol{\alpha}, \mathbf{d}_h) < f_{\mathrm{m}}(\boldsymbol{\alpha})$, which will induce the sparsity among μ_h's. Motivated by this observation, we design an efficient optimization scheme which iteratively "finds" the active constraints, and then solves a subproblem with the selected constraints only. By applying this scheme, the computational burden brought by the infinite number of constraints can be avoided. The details of the above procedure is presented in Algorithm 1, which is also known as the cutting plane algorithm (Kelley, 1960; Mutapcic and Boyd, 2009).

Algorithm 1 involves two major steps: the feature inference step (also known as the worst-case analysis) and the subproblem optimization step. Specifically, the worst-case analysis is to infer the most-violated $\mathbf{d}_t$ based on $\boldsymbol{\alpha}^{t-1}$, and add it into the active constraint set $\mathcal{C}_t$. Once an active $\mathbf{d}_t$ is identified, we update $\boldsymbol{\alpha}^t$ by solving the following subproblem with the constraints defined in $\mathcal{C}_t$:

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}, \theta \in \mathbb{R}} \theta, \qquad \text{s.t.} \qquad f(\boldsymbol{\alpha}, \mathbf{d}_h) - \theta \leq 0,\ \forall\ \mathbf{d}_h \in \mathcal{C}_t. \tag{16}$$

Algorithm 1 Cutting Plane Algorithm for Solving (13).

```
1: Initialize $\boldsymbol{\alpha}^0 = C\mathbf{1}$ and $\mathcal{C}_0 = \emptyset$. Set iteration index $t = 1$.
2: Feature Inference:
     Do worst-case analysis to *infer* the most violated $\mathbf{d}_t$ based on $\boldsymbol{\alpha}^{t-1}$.
     Set $\mathcal{C}_t = \mathcal{C}_{t-1} \bigcup \{\mathbf{d}_t\}$.
3: Subproblem Optimization:
     Solve subproblem (16), obtaining the optimal solution $\boldsymbol{\alpha}^t$ and $\mu^t$.
4: Let $t = t + 1$. Repeat step 2-3 until convergence.
```

For feature selection tasks, the optimization complexity of (13) can be greatly reduced, since there are only a small number of active constraints involved in problem (16).

The whole procedure iterates until some stopping conditions are achieved. As will be shown later, in general, each active $\mathbf{d}_t \in \mathcal{C}_t$ involves at most B **new features**. In this sense, we refer Algorithm 1 to as the *feature generating machine* (FGM). Recall that, at the beginning, there is no feature being selected, thus we have the empirical loss $\boldsymbol{\xi} = \mathbf{1}$. According to (8), we can initialize $\boldsymbol{\alpha}^0 = C\mathbf{1}$. Finally, once the optimal solution $\mathbf{d}^*$ to (16) is obtained, the selected features (or feature groups) are associated with the nonzero entries in $\mathbf{d}^*$. Note that, each $\mathbf{d} \in \mathcal{C}_t$ involves at most B features/groups, thus the number of selected features/groups is no more than tB after t iterations, namely $||\mathbf{d}^*||_0 \leq tB$.

3.2 Convergence Analysis

Before the introduction of the worst-case analysis and the solution to the subproblem, we first conduct the convergence analysis of Algorithm 1.

Without loss of generality, let $\mathcal{A} \times \mathcal{D}$ be the domain for problem (13). In the $(t+1)$th iteration, we find a new constraint $\mathbf{d}_{t+1}$ based on $\boldsymbol{\alpha}_t$ and add it into $\mathcal{C}_t$, i.e., $f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1}) = \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}_t, \mathbf{d})$. Apparently, we have $\mathcal{C}_t \subseteq \mathcal{C}_{t+1}$. For convenience, we define

$$\beta_t = \max_{1\leq i\leq t} f(\boldsymbol{\alpha}_t, \mathbf{d}_i) = \min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{1\leq i\leq t} f(\boldsymbol{\alpha}, \mathbf{d}_i).$$

and

$$\varphi_t = \min_{1\leq j\leq t} f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}) = \min_{1\leq j\leq t} (\max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}_j, \mathbf{d})),$$

First of all, we have the following lemma.

Lemma 1 *Let (α^*, θ^*) be a globally optimal solution of (13), $\{\theta_t\}$ and $\{\varphi_t\}$ as defined above, then: $\theta_t \leq \theta^* \leq \varphi_t$. With the number of iteration t increasing, $\{\theta_t\}$ is monotonically increasing and the sequence $\{\varphi_t\}$ is monotonically decreasing (Chen and Ye, 2008).*

Proof According to the definition, we have $\theta_t = \beta_t$. Moreover, θ^*=$\min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d})$. For a fixed feasible $\boldsymbol{\alpha}$, we have $\max_{\mathbf{d}\in\mathcal{C}_t} f(\boldsymbol{\alpha}, \mathbf{d}) \leq \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d})$, then

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{C}_t} f(\boldsymbol{\alpha}, \mathbf{d}) \leq \min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d}),$$

that is, $\theta_t \leq \theta^*$. On the other hand, for $\forall j = 1, \cdots, k$, $f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}) = \max_{\mathbf{d} \in \mathcal{D}} f(\boldsymbol{\alpha}_j, \mathbf{d})$, thus $(\boldsymbol{\alpha}_j, f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}))$ is a feasible solution of (13). Then $\theta^* \leq f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1})$ for $j = 1, \cdots, t$, and hence we have

$$\theta^* \leq \varphi_t = \min_{1 \leq j \leq t} f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}).$$

With increasing iteration t, the subset $\mathcal{C}_t$ is monotonically increasing, so $\{\theta_t\}$ is monotonically increasing while $\{\varphi_t\}$ is monotonically decreasing. The proof is completed. ■

The following theorem shows that FGM converges to a global solution of (13).

Theorem 2 *Assume that in Algorithm 1, the subproblem (16) and the worst-case analysis in step 2 can be solved. Let $\{(\boldsymbol{\alpha}_t, \theta_t)\}$ be the sequence generated by Algorithm 1. If Algorithm 1 terminates at iteration $(t+1)$, then $\{(\boldsymbol{\alpha}_t, \theta_t)\}$ is the global optimal solution of (13); otherwise, $(\boldsymbol{\alpha}_t, \theta_t)$ converges to a global optimal solution $(\boldsymbol{\alpha}^*, \theta^*)$ of (13).*

Proof We can measure the convergence of FGM by the gap difference of series $\{\theta_t\}$ and $\{\varphi_t\}$. Assume in tth iteration, there is no update of $\mathcal{C}_t$, i.e. $\mathbf{d}_{t+1} = \arg\max_{\mathbf{d} \in \mathcal{D}} f(\boldsymbol{\alpha}_t, \mathbf{d}) \in \mathcal{C}_t$, then $\mathcal{C}_t = \mathcal{C}_{t+1}$. In this case, $(\boldsymbol{\alpha}_t, \theta_t)$ is the globally optimal solution of (13). Actually, since $\mathcal{C}_t = \mathcal{C}_{t+1}$, in Algorithm 1, there will be no update of $\boldsymbol{\alpha}$, i.e. $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t$. Then we have

$$\begin{aligned} & f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1}) = \max_{\mathbf{d} \in \mathcal{D}} f(\boldsymbol{\alpha}_t, \mathbf{d}) = \max_{\mathbf{d} \in \mathcal{C}_t} f(\boldsymbol{\alpha}_t, \mathbf{d}) = \max_{1 \leq i \leq t} f(\boldsymbol{\alpha}_t, \mathbf{d}_i) = \theta_t \\ & \varphi_t = \min_{1 \leq j \leq t} f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}) \leq \theta_t. \end{aligned}$$

According to Lemma 1, we have $\theta_t \leq \theta^* \leq \varphi_t$, thus we have $\theta_t = \theta^* = \varphi_t$, and $(\boldsymbol{\alpha}_t, \theta_t)$ is the global optimum of (13).

Suppose the algorithm does not terminate in finite steps. Let $\mathcal{X} = \mathcal{A} \times [\theta_1, \theta^*]$, a limit point $(\bar{\boldsymbol{\alpha}}, \bar{\theta})$ exists for $(\boldsymbol{\alpha}_t, \theta_t)$, since $\mathcal{X}$ is compact. And we also have $\bar{\theta} \leq \theta^*$. For each t, let $\mathcal{X}_t$ be the feasible region of tth subproblem, which have $\mathcal{X}_t \subseteq \mathcal{X}$, and $(\bar{\boldsymbol{\alpha}}, \bar{\theta}) \in \cap_{t=1}^{\infty} \mathcal{X}_t \subseteq \mathcal{X}$. Then we have $f(\bar{\boldsymbol{\alpha}}, \mathbf{d}_t) - \bar{\theta} \leq 0$, $\mathbf{d}_t \in C_t$ for each given $t = 1, \cdots$.

To show $(\bar{\boldsymbol{\alpha}}, \bar{\theta})$ is global optimal of problem (13), we only need to show $(\bar{\boldsymbol{\alpha}}, \bar{\theta})$ is a feasible point of problem (13), i.e., $\bar{\theta} \geq f(\bar{\boldsymbol{\alpha}}, \mathbf{d})$ for all $\mathbf{d} \in \mathcal{D}$, so $\bar{\theta} \geq \theta^*$ and we must have $\bar{\theta} = \theta^*$. Let $v(\boldsymbol{\alpha}, \theta) = \min_{\mathbf{d} \in \mathcal{D}} (\theta - f(\boldsymbol{\alpha}, \mathbf{d})) = \theta - \max_{\mathbf{d} \in \mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d})$. Then $v(\boldsymbol{\alpha}, \theta)$ is continuous w.r.t. $(\boldsymbol{\alpha}, \theta)$. By applying the continuity property of $v(\boldsymbol{\alpha}, \theta)$, we have

$$\begin{aligned} v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) &= v(\boldsymbol{\alpha}_t, \theta_t) + (v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) - v(\boldsymbol{\alpha}_t, \theta_t)) \\ &= (\theta_t - f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1})) + (v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) - v(\boldsymbol{\alpha}_t, \theta_t)) \\ &\geq (\theta_t - f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1})) - (\bar{\theta} - f(\bar{\boldsymbol{\alpha}}, \mathbf{d}_t)) + (v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) - v(\boldsymbol{\alpha}_t, \theta_t)) \to 0 \ \ (\text{when } t \to \infty), \end{aligned}$$

where we use the continuity of $v(\boldsymbol{\alpha}, \theta)$. The proof is completed. ■

4. Efficient Worst-Case Analysis

According to Theorem 2, the exact solution to the worst-case analysis is necessary for the global convergence of FGM. Fortunately, for a number of feature selection tasks, the exact worst-case analysis does exist. For simplicity, hereafter we drop the superscript t from $\boldsymbol{\alpha}^t$.

4.1 Worst-Case Analysis for Linear Feature Selection

The worst-case analysis for the linear feature selection is to solve the following maximization problem:

$$\max_{\mathbf{d}} \ \frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2, \quad \text{s.t.} \ \sum_{j=1}^{m} d_j \leq B, \mathbf{0} \preceq \mathbf{d} \preceq \mathbf{1}. \tag{17}$$

This problem in general is very hard to be solved. Recall that $\mathbf{c}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \in \mathbb{R}^m$, and we have $\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\|^2 = \|\sum_{i=1}^{n}(\alpha_i y_i \mathbf{x}_i) \odot \sqrt{\mathbf{d}}\|^2 = \sum_{j=1}^{m} c_j(\boldsymbol{\alpha})^2 d_j$. Based on this relation, we define a feature score s_j to measure the importance of features as

$$s_j = [c_j(\boldsymbol{\alpha})]^2.$$

Accordingly, problem (17) can be further formulated as a linear programming problem:

$$\max_{\mathbf{d}} \ \frac{1}{2}\sum_{j=1}^{m} s_j d_j, \quad \text{s.t.} \ \sum_{j=1}^{m} d_i \leq B, \quad \mathbf{0} \preceq \mathbf{d} \preceq \mathbf{1}. \tag{18}$$

The optimal solution to this problem can be obtained without any numeric optimization solver. Specifically, we can construct a feasible solution by first finding the B features with the largest feature score s_j, and then setting the corresponding d_j to 1 and the rests to 0. It is easy to verify that such a $\mathbf{d}$ is also an optimal solution to (18). Note that, as long as there are more than B features with $s_j > 0$, we have $||\mathbf{d}||_0 = B$. In other words, in general, $\mathbf{d}$ will include B features into the optimization after each worst-case analysis.

4.2 Worst-Case Analysis for Group Feature Selection

The worst-case analysis for linear feature selection can be easily extended to group feature selection. Suppose that the features are organized into groups by $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$, and there is no overlapping features among groups, namely $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$, $\forall i \neq j$. To find the most-active groups, we just need to solve the following optimization problem:

$$\max_{\mathbf{d}\in\mathcal{D}} \ \sum_{j=1}^{p} d_j \left\|\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}\right\|^2 = \max_{\mathbf{d}\in\mathcal{D}} \sum_{j=1}^{p} d_j \mathbf{c}'_{\mathcal{G}_j}\mathbf{c}_{\mathcal{G}_j}, \tag{19}$$

where $\mathbf{c}_{\mathcal{G}_j} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}$ for group $\mathcal{G}_j$. Let $s_j = \mathbf{c}'_{\mathcal{G}_j}\mathbf{c}_{\mathcal{G}_j}$ be the score for group $\mathcal{G}_j$. The optimal solution to (19) can be obtained by first finding the B groups with the largest s_j's, and then setting their d_j's to 1 and the rests to 0. If $|\mathcal{G}_j| = 1$, $\forall j \in \{1, ..., m\}$, problem (19) is reduced to problem (18), where $\mathcal{G} = \{\{1\}, ..., \{p\}\}$ and $s_j = [c_j(\boldsymbol{\alpha})]^2$ for $j \in \mathcal{G}$. In this sense, we unify the worst-case analysis of the two feature selection tasks in Algorithm 2.

4.3 Worst-Case Analysis for Groups with Complex Structures

Algorithm 2 can be also extended to feature groups with overlapping features or with tree-structures. Recall that $p = |\mathcal{G}|$, the worst-case analysis in Algorithm 2 takes $O(mn + p\log(B))$ cost, where the $O(mn)$ cost is for computing $\mathbf{c}$, and the $O(p\log(B))$ cost is for

Algorithm 2 Algorithm for Worst-Case Analysis.

```
Given $\boldsymbol{\alpha}$, $B$, the training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and the group index set $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$.
Calculate $\mathbf{c} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$.
Calculate the feature score $\mathbf{s}$, where $s_j = \mathbf{c}'_{\mathcal{G}_j}\mathbf{c}_{\mathcal{G}_j}$.
Find the $B$ largest $s_j$'s.
Set $d_j$ corresponding to the $B$ largest $s_j$'s to 1 and the rests to 0.
Return $\mathbf{d}$.
```

sorting s_j's. The second term is negligible if $p = O(m)$. However, if p is extremely large, namely $p \gg m$, the computational cost for computing and sorting s_j will be unbearable. For instance, if the feature groups are organized into a graph or a tree structure, p can become very huge, namely $p \gg m$ (Jenatton et al., 2011b).

Since we just need to find the B groups with the largest s_j's, we can address the above computational difficulty by implementing Algorithm 2 in an incremental way. Specifically, we can maintain a cache $\mathbf{c}_B$ to store the indices and scores of the B feature groups with the largest scores among those traversed groups, and then calculate the feature score s_j for each group one by one. After computing s_j for a new group $\mathcal{G}_j$, we update $\mathbf{c}_B$ if $s_j > s_B^{min}$, where s_B^{min} denotes the smallest score in $\mathbf{c}_B$. By applying this technique, the whole computational cost of the worst-case analysis can be greatly reduced to $O(n\log(m) + B\log(p))$ if the groups follow the tree-structure defined in Definition 1.

Remark 2 *Given a set of groups $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$ that is organized as a tree structure in Definition 1, suppose $\mathcal{G}_h \subseteq \mathcal{G}_g$, then $s_h < s_g$. Furthermore, $\mathcal{G}_g$ and all its decedent $\mathcal{G}_h \subseteq \mathcal{G}_g$ will not be selected if $s_g < s_B^{min}$. Therefore, the computational cost of the worst-case analysis can be reduced to $O(n\log(m) + B\log(p))$ for a balanced tree structure.*

5. Efficient Subproblem Optimization

After updating $\mathcal{C}_t$, now we tend to solve the subproblem (16). Recall that, any $\mathbf{d}_h \in \mathcal{C}_t$ indexes a set of features. For convenience, we define $\mathbf{X}_h \triangleq [\mathbf{x}_h^1, ..., \mathbf{x}_h^n]' \in \mathbb{R}^{n\times B}$, where $\mathbf{x}_h^i$ denotes the ith instance with the features indexed by $\mathbf{d}_h$.

5.1 Subproblem Optimization via MKL

Regarding problem (16), let $\mu_h \geq 0$ be the dual variable for each constraint defined by $\mathbf{d}_h$, the Lagrangian function can be written as:

$$\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \theta - \sum_{\mathbf{d}_h \in \mathcal{C}_t} \mu_t \left(\theta - f(\boldsymbol{\alpha}, \mathbf{d}_h)\right).$$

By setting its derivative w.r.t. θ to zero, we have $\sum \mu_t = 1$. Let $\boldsymbol{\mu}$ be the vector of all μ_t's, and $\mathcal{M} = \{\boldsymbol{\mu} | \sum \mu_h = 1, \mu_h \geq 0\}$ be the domain of $\boldsymbol{\mu}$. By applying the minimax saddle-point theorem (Sion, 1958), $\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu})$ can be rewritten as:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \min_{\boldsymbol{\mu}\in\mathcal{M}} \; -\sum_{\mathbf{d}_h\in\mathcal{C}_t} \mu_h f(\boldsymbol{\alpha}, \mathbf{d}_h) = \min_{\boldsymbol{\mu}\in\mathcal{M}} \max_{\boldsymbol{\alpha}\in\mathcal{A}} \; -\frac{1}{2}(\boldsymbol{\alpha}\odot\mathbf{y})'\Big(\sum_{\mathbf{d}_h\in\mathcal{C}_t} \mu_h \mathbf{X}_h \mathbf{X}_h' + \frac{1}{C}\mathbf{I}\Big)(\boldsymbol{\alpha}\odot\mathbf{y}), \qquad (20)$$

where the equality holds since the objective function is concave in $\boldsymbol{\alpha}$ and convex in $\boldsymbol{\mu}$. Problem (20) is a multiple kernel learning (MKL) problem (Lanckriet et al., 2004; Rakotomamonjy et al., 2008) with $|\mathcal{C}_t|$ base kernel matrices $\mathbf{X}_h\mathbf{X}_h'$. Several existing MKL approaches can be adopted to solve this problem, such as SimpleMKL (Rakotomamonjy et al., 2008). Specifically, SimpleMKL solves the non-smooth optimization problem by applying a sub-gradient method (Rakotomamonjy et al., 2008; Nedic and Ozdaglar, 2009). Unfortunately, it is expensive to calculate the sub-gradient w.r.t. $\boldsymbol{\alpha}$ for large-scale problems. Moreover, the convergence speed of sub-gradient methods is limited. The minimax subproblem (20) can be also solved by the proximal gradient methods (Nemirovski, 2005; Tseng, 2008) or SQP methods (Pee and Royset, 2010) with faster convergence rates. However, these methods involve expensive subproblems, and they are very inefficient when n is large.

Based on the definition of $\mathbf{X}_h$, we have $\sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h\mathbf{X}_h\mathbf{X}_h^{'} = \sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h\mathbf{X}\text{diag}(\mathbf{d}_h)\mathbf{X}^{'} = \mathbf{X}\text{diag}(\sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h\mathbf{d}_h)\mathbf{X}^{'}$ w.r.t. the linear feature selection task. Accordingly, we have

$$\mathbf{d}^* = \sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h^*\mathbf{d}_h, \tag{21}$$

where $\boldsymbol{\mu}^* = [\mu_1^*, ..., \mu_h^*]'$ denotes the optimal solution to (20). It is easy to check that, the relation in (21) also holds for the group feature selection tasks. Since $\sum_{h=1}^{|\mathcal{C}_t|}\mu_h^* = 1$, we have $\mathbf{d}^* \in \mathcal{D} = \big\{\mathbf{d}\big|\sum_{j=1}^m d_j \leq B,\ d_j\in[0,1],\ j=1,\cdots,m\big\}$, where the nonzero entries are associated with selected features/groups.

5.2 Subproblem Optimization in the Primal

Solving the MKL problem in (20) is very expensive when n is very large. In other words, the dimension of the optimization variable $\boldsymbol{\alpha}$ in (20) is very large. Recall that, after t iterations, $\mathcal{C}_t$ includes at most tB features, where $tB \ll n$. Motivated by this observation, we propose to solve it in the primal form w.r.t. $\mathbf{w}$. Apparently, the dimension of the optimization variable $\mathbf{w}$ is much smaller than $\boldsymbol{\alpha}$, namely $||\mathbf{w}||_0 \leq tB \ll n$.

Without loss of generality, we assume that $t = |\mathcal{C}_t|$ after tth iterations. Let $\mathbf{X}_h \in \mathbb{R}^{n\times B}$ denote the data with features indexed by $\mathbf{d}_h \in \mathcal{C}_t$, $\boldsymbol{\omega}_h \in \mathbb{R}^B$ denote the weight vector w.r.t. $\mathbf{X}_h$, $\boldsymbol{\omega} = [\boldsymbol{\omega}_1', ..., \boldsymbol{\omega}_t']' \in \mathbb{R}^{tB}$ be a supervector concatenating all $\boldsymbol{\omega}_h$'s, where $tB \ll n$. For convenience, we define

$$P(\boldsymbol{\omega}, b) = \frac{C}{2}\sum_{i=1}^n \xi_i^2$$

w.r.t. the squared hinge loss, where $\xi_i = \max(1 - y_i(\sum_h \boldsymbol{\omega}_h'\mathbf{x}_{ih} - b), 0)$, and

$$P(\boldsymbol{\omega}, b) = C\sum_{i=1}^n \log(1+\exp(\xi_i)),$$

w.r.t. the logistic loss, where $\xi_i = -y_i(\sum_{h=1}^t \boldsymbol{\omega}_h'\mathbf{x}_{ih} - b)$.

Theorem 3 *Let $\mathbf{x}_{ih}$ denote the ith instance of $\mathbf{X}_h$, the MKL subproblem (20) can be equivalently addressed by solving an $\ell_{2,1}^2$-regularized problem:*

$$\min_{\boldsymbol{\omega}} \quad \frac{1}{2}\left(\sum_{h=1}^t \|\boldsymbol{\omega}_h\|\right)^2 + P(\boldsymbol{\omega}, b). \tag{22}$$

Furthermore, the dual optimal solution $\boldsymbol{\alpha}^*$ *can be recovered from the optimal solution* ξ^**. To be more specific,* $\alpha_i^* = C\xi_i^*$ *holds for the square-hinge loss and* $\alpha_i = \frac{C\exp(\xi_i^*)}{1+\exp(\xi_i^*)}$ *holds for the logistic loss.*

The proof can be found in Appendix A.

According to Theorem 3, rather than directly solving (20), we can address its primal form (22) instead, which brings great advantages for the efficient optimization. Moreover, we can recover $\boldsymbol{\alpha}^*$ by $\alpha_i^* = C\xi_i^*$ and $\alpha_i = \frac{C\exp(\xi_i^*)}{1+\exp(\xi_i^*)}$ w.r.t. the squared hinge loss and logistic loss, respectively.[2] For convenience, we define

$$F(\boldsymbol{\omega}, b) = \Omega(\boldsymbol{\omega}) + P(\boldsymbol{\omega}, b),$$

where $\Omega(\boldsymbol{\omega}) = \frac{1}{2}(\sum_{h=1}^{t} \|\boldsymbol{\omega}_h\|)^2$. $F(\boldsymbol{\omega}, b)$ is a non-smooth function w.r.t $\boldsymbol{\omega}$, and $P(\boldsymbol{\omega}, b)$ has block coordinate Lipschitz gradient w.r.t $\boldsymbol{\omega}$ and b, where $\boldsymbol{\omega}$ is deemed as a block variable. Correspondingly, let $\nabla P(\mathbf{v}) = \partial_{\mathbf{v}} P(\mathbf{v}, v_b)$ and $\nabla_b P(\mathbf{v}, v_b) = \partial_b P(\mathbf{v}, v_b)$. It is known that $F(\boldsymbol{\omega}, b)$ is at least Lipschitz continuous for both logistic loss and squared hinge loss (Yuan et al., 2011):

$$P(\boldsymbol{\omega}, b) \leq P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v}\rangle + \langle \nabla_b P(v_b), b - v_b\rangle + \frac{L}{2}||\boldsymbol{\omega} - \mathbf{v}||^2 + \frac{L_b}{2}||b - v_b||^2,$$

where L and L_b denote the Lipschitz constants regarding $\boldsymbol{\omega}$ and b, respectively.

Since $F(\boldsymbol{\omega}, b)$ is separable w.r.t $\boldsymbol{\omega}$ and b, we can minimize it regarding $\boldsymbol{\omega}$ and b in a block coordinate descent manner (Tseng, 2001). For each block variable, we update it through an accelerated proximal gradient (APG) method (Beck and Teboulle, 2009; Toh and Yun, 2009), which iteratively minimizes a quadratic approximation to $F(\boldsymbol{\omega}, b)$. Specifically, given a point $[\mathbf{v}', v_b]'$, the quadratic approximation to $F(\boldsymbol{\omega}, b)$ at this point w.r.t. $\boldsymbol{\omega}$ is:

$$\begin{aligned} Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b) &= P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v}\rangle + \Omega(\boldsymbol{\omega}) + \frac{\tau}{2}||\boldsymbol{\omega} - \mathbf{v}||^2 \\ &= \frac{\tau}{2}\|\boldsymbol{\omega} - \mathbf{u}\|^2 + \Omega(\boldsymbol{\omega}) + P(\mathbf{v}, v_b) - \frac{1}{2\tau}\|\nabla P(\mathbf{v})\|^2, \end{aligned} \tag{23}$$

where τ is a positive constant and $\mathbf{u} = \mathbf{v} - \frac{1}{\tau}\nabla P(\mathbf{v})$. To minimize $Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ w.r.t. $\boldsymbol{\omega}$, it is reduced to solve the following Moreau projection problem (Martins et al., 2010):

$$\min_{\boldsymbol{\omega}} \ \frac{\tau}{2}\|\boldsymbol{\omega} - \mathbf{u}\|^2 + \Omega(\boldsymbol{\omega}). \tag{24}$$

For convenience, let $\mathbf{u}_h$ be the corresponding component to $\boldsymbol{\omega}_h$, namely $\mathbf{u} = [\mathbf{u}_1', ..., \mathbf{u}_t']'$. Martins et al. (2010) has shown that, problem (24) has a unique closed-form solution, which is summarized in the following proposition.

Proposition 1 *Let* $S_\tau(\mathbf{u}, \mathbf{v})$ *be the optimal solution to problem (24) at point* $\mathbf{v}$*, then* $S_\tau(\mathbf{u}, \mathbf{v})$ *is unique and can be calculated as follows:*

$$[S_\tau(\mathbf{u}, \mathbf{v})]_h = \begin{cases} \frac{o_h}{\|\mathbf{u}_h\|}\mathbf{u}_h, & \textit{if } \ o_h > 0, \\ \mathbf{0}, & \textit{otherwise}, \end{cases} \tag{25}$$

[2] Here the optimal dual variable $\boldsymbol{\alpha}^*$ is required in the worst-case analysis.

where $[S_\tau(\mathbf{u},\mathbf{v})]_h \in \mathbb{R}^B$ denote the corresponding component w.r.t. $\mathbf{u}_h$ and $\mathbf{o} \in \mathbb{R}^t$ be an intermediate variable. Let $\widehat{\mathbf{o}} = [\|\mathbf{u}_1\|, ..., \|\mathbf{u}_t\|]' \in \mathbb{R}^t$, the intermediate vector $\mathbf{o}$ can be calculated via a soft-threshold operator: $o_h = [soft(\widehat{\mathbf{o}}, \varsigma)]_h = \begin{cases} \widehat{o}_h - \varsigma, & if \quad \widehat{o}_h > \varsigma, \\ 0, & Otherwise. \end{cases}$ Here the threshold value ς can be calculated in Step 4 of Algorithm 3.

Proof The proof can be adapted from the results in Appendix F in (Martins et al., 2010). ■

Algorithm 3 Moreau Projection $S_\tau(\mathbf{u},\mathbf{v})$.

```
Given an point $\mathbf{v}$, $s = \frac{1}{\tau}$ and the number of kernels $t$.
1: Calculate $\widehat{o}_h = \|\mathbf{g}_h\|$ for all $h = 1, ..., t$.
2: Sort $\widehat{\mathbf{o}}$ to obtain $\bar{\mathbf{o}}$ such that $\bar{o}_{(1)} \geq ... \geq \bar{o}_{(t)}$.
3: Find $\rho = \max \left\{ t \middle| \bar{o}_h - \frac{s}{1+hs} \sum_{i=1}^{h} \bar{o}_i > 0, h = 1, ..., t \right\}$.
4: Calculate the threshold value $\varsigma = \frac{s}{1+\rho s} \sum_{i=1}^{\rho} \bar{o}_i$.
5: Compute $\mathbf{o} = soft(\widehat{\mathbf{o}}, \varsigma)$.
6: Compute and output $S_\tau(\mathbf{u},\mathbf{v})$ via equation (25).
```

Remark 3 *For the Moreau projection in Algorithm 3, the sorting takes $O(t)$ cost. In FGM setting, t in general is very small, thus the Moreau projection can be efficiently computed.*

Now we tend to minimize $F(\boldsymbol{\omega}, b)$ regarding b. Since there is no regularizer on b, it is equivalent to minimize $P(\boldsymbol{\omega}, v_b)$ w.r.t. b. The updating can be done by $b = v_b - \frac{1}{\tau_b} \nabla_b P(\mathbf{v}, v_b)$, which is essentially the steepest descent update. We can use the Armijo line search (Nocedal and Wright, 2006) to find a step size $\frac{1}{\tau_b}$ such that,

$$P(\boldsymbol{\omega}, b) \leq P(\boldsymbol{\omega}, v_b) - \frac{1}{2\tau_b} |\nabla_b P(\mathbf{v}, v_b)|^2,$$

where $\boldsymbol{\omega}$ is the minimizer to $Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$. This line search can be efficiently performed since it is conducted on a single variable only.

With the calculation of $S_\tau(\mathbf{g})$ in Algorithm 3 and the updating rule of b above, we propose to solve (22) through a modified APG method in a block coordinate manner in Algorithm 4. In Algorithm 4, L_t and L_{bt} denote the Lipschitz constants of $P(\boldsymbol{\omega}, b)$ w.r.t. $\boldsymbol{\omega}$ and b at the t iteration of Algorithm 1, respectively. In practice, we estimate L_0 by $L_0 = 0.01nC$, which will be further adjusted by the line search. When $t > 0$, L_t is estimated by $L_t = \eta L_{t-1}$. Finally, a sublinear convergence rate of Algorithm 4 is guaranteed.[3]

Theorem 4 *Let L_t and L_{bt} be the Lipschitz constant of $P(\boldsymbol{\omega}, b)$ w.r.t. $\boldsymbol{\omega}$ and b respectively. Let $\{(\boldsymbol{\omega}^{k'}, b^k)'\}$ be the sequences generated by Algorithm 4 and $L = \max(L_{bt}, L_t)$, for any $k \geq 1$, we have:*

$$F(\boldsymbol{\omega}^k, b^k) - F(\boldsymbol{\omega}^*, b^*) \leq \frac{2L_t||\boldsymbol{\omega}^0 - \boldsymbol{\omega}^*||^2}{\eta(k+1)^2} + \frac{2L_{bt}(b^0 - b^*)^2}{\eta(k+1)^2} \leq \frac{2L||\boldsymbol{\omega}^0 - \boldsymbol{\omega}^*||^2}{\eta(k+1)^2} + \frac{2L(b^0 - b^*)^2}{\eta(k+1)^2}.$$

[3] Regarding Algorithm 4, a linear convergence rate can be attained w.r.t. the logistic loss under mild conditions. The details can be found in Appendix C.

The proof can be found in Appendix B. The internal variables L^k and L_b^k in Algorithm 3 are useful in the proof of the convergence rate.

According to Theorem 4, if L_{bt} is very different from L_t, the block coordinate updating scheme in Algorithm 4 can achieve an improved convergence speed over the batch updating w.r.t. $(\boldsymbol{\omega}', b)'$. Moreover, the warm-start for initialization of $\boldsymbol{\omega}$ and b in Algorithm 4 is useful to accelerate the convergence speed.

Algorithm 4 Accelerated Proximal Gradient for Solving Problem (22) (**Inner Iterations**).

```
Initialization: Initialize the Lipschitz constant $L_t = L_{t-1}$, set $\boldsymbol{\omega}^0 = \mathbf{v}^1 = [\boldsymbol{\omega}'_{t-1}, \mathbf{0}']'$ and
$b^0 = v_b^1 = b_{t-1}$ by **warm start**, $\tau_0 = L_t$, $\eta \in (0,1)$, parameter $\varrho^1 = 1$ and $k = 1$.
1: Set $\tau = \eta\tau_{k-1}$.
    For $j = 0, 1, ...,$
        Set $\mathbf{u} = \mathbf{v}^k - \frac{1}{\tau}\nabla p(\mathbf{v}^k)$, compute $S_\tau(\mathbf{u}, \mathbf{v}^k)$.
        If $F(S_\tau(\mathbf{u}, \mathbf{v}^k), v_b^k) \leq Q(S_\tau(\mathbf{u}, \mathbf{v}^k), \mathbf{v}^k, v_b^k)$,
            Set $\tau_k = \tau$, stop;
        Else
            $\tau = \min\{\eta^{-1}\tau, L_t\}$.
        End
    End
2: Set $\boldsymbol{\omega}^k = S_{\tau_k}(\mathbf{u}, \mathbf{v}^k)$ and $L^k = \tau_k$.
3: Set $\tau_b = \eta\tau_k$.
    For $j = 0, 1, ...$
        Set $b = v_b^k - \frac{1}{\tau_b}\nabla_b P(\mathbf{v}, v_b^k)$.
        If $P(\boldsymbol{\omega}^k, b) \leq P(\boldsymbol{\omega}^k, v_b^k) - \frac{1}{2\tau_b}|\nabla_b P(\mathbf{v}, v_b^k)|^2$,
            Stop;
        Else
            $\tau_b = \min\{\eta^{-1}\tau_b, L_t\}$.
        End
    End
4: Set $b^k = b$ and $L_b^k = \tau_b$. Go to Step 8 if the stopping condition achieves.
5: Set $\varrho^{k+1} = \frac{1+\sqrt{1+4(\varrho^k)^2}}{2}$.
6: Set $\mathbf{v}^{k+1} = \boldsymbol{\omega}^k + \frac{\varrho^k - 1}{\varrho^{k+1}}(\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k-1})$ and $v_b^{k+1} = b^k + \frac{\varrho^k - 1}{\varrho^{k+1}}(b^k - b^{k-1})$.
7: Let $k = k + 1$ and go to step 1.
8: Return and output $\boldsymbol{\omega}_t = \boldsymbol{\omega}^k$, $b_t = b^k$ and $L_t = \eta\tau_k$.
```

Warm Start: From Theorem 4, the number of iterations needed by APG to achieve an ϵ-solution is $O(\frac{||\boldsymbol{\omega}^0 - \boldsymbol{\omega}^*||}{\sqrt{\epsilon}})$. Since FGM incrementally includes a set of features into the subproblem optimization, an warm start of $\boldsymbol{\omega}^0$ can be very useful to improve its efficiency. To be more specific, when a new active constraint is added, we can use the optimal solution of the last iteration (denoted by $[\boldsymbol{\omega}_1^{*\prime}, ..., \boldsymbol{\omega}_{t-1}^{*}{}']$) as an initial guess to the next iteration. In other words, at the tth iteration, we use $\boldsymbol{\omega}^{-1} = \boldsymbol{\omega}^0 = [\boldsymbol{\omega}_1^{*\prime}, ..., \boldsymbol{\omega}_{t-1}^{*}{}', \mathbf{0}']'$ as the starting point.

5.3 De-biasing of FGM

Based on Algorithm 4, we show that FGM resembles the *re-training* process and can achieve de-biased solutions. For convenience, we first revisit the de-biasing process in the ℓ_1-minimization (Figueiredo et al., 2007).

De-biasing for ℓ_1-methods. To reduce the solution bias, a de-biasing process is often adopted in ℓ_1-methods. For example, in the sparse recovery problem (Figueiredo et al., 2007), after solving the ℓ_1-regularized problem, a **least-square problem** (which drops the ℓ_1-regularizer) is solved with the detected features (or supports). To reduce the feature selection bias, one can also apply this de-biasing technique to the ℓ_1-SVM for classification tasks. However, it is worth mentioning that, when dealing with classification tasks, due to the label noises, such as the rounding errors of labels, a regularizer is necessary and important to avoid the over-fitting issue. Alternatively, we can apply the standard SVM on the selected features to do the de-biasing using a relative large C, which is also referred to as the *re-training*. When C goes to infinity, it is equivalent to minimize the empirical loss without any regularizer, which, however, may cause the over-fitting problem.

De-biasing effect of FGM. Recall that, in FGM, the parameters B and the trade-off parameter C are adjusted separately. In the worst-case analysis, FGM includes B features/groups that violate the optimality condition the most. When B is sufficiently small, the selected B features/groups can be regarded as the most relevant features. After that, FGM addresses the $\ell_{2,1}^2$-regularized problem (22) w.r.t. the selected features only, which mimics the above re-training strategy for de-biasing. Specifically, we can use a relatively large C to penalize the empirical loss to reduce the solution bias. Accordingly, with a suitable C, each outer iteration of FGM can be deemed as the de-biasing process, and the de-biased solution will in turn help the worst-case analysis to select more discriminative features.

5.4 Stopping Conditions

Suitable stopping conditions of FGM are important to reduce the risk of over-fitting and improve the training efficiency. The stopping criteria of FGM include 1) the stopping conditions for the outer cutting plane iterations in Algorithm 1; 2) the stopping conditions for the inner APG iterations in Algorithm 4.

5.4.1 Stopping Conditions for Outer iterations

We first introduce the stopping conditions w.r.t. the outer iterations in Algorithm 1. Recall that the optimality condition for the SIP problem is $\sum_{\mathbf{d}_t \in \mathcal{D}} \mu_t \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \mathbf{d}_t) = \mathbf{0}$ and $\mu_t(f(\boldsymbol{\alpha}, \mathbf{d}_t) - f_m(\boldsymbol{\alpha})) = 0, \forall \mathbf{d}_t \in \mathcal{D}$. A direct stopping condition can be written as:

$$f(\boldsymbol{\alpha}, \mathbf{d}) \leq f_{\mathrm{m}}(\boldsymbol{\alpha}) + \epsilon, \quad \forall \mathbf{d} \in \mathcal{D}, \tag{26}$$

where $f_{\mathrm{m}}(\boldsymbol{\alpha}) = \max_{\mathbf{d}_h \in \mathcal{C}_t} f(\boldsymbol{\alpha}, \mathbf{d}_h)$ and ϵ is a small tolerance value. To check this condition, we just need to find a new $\mathbf{d}_{t+1}$ via the worst-case analysis. If $f(\boldsymbol{\alpha}, \mathbf{d}_{t+1}) \leq f_m(\boldsymbol{\alpha}) + \epsilon$, the stopping condition in (26) is achieved. In practice, due to the scale variation of $f_{\mathrm{m}}(\boldsymbol{\alpha})$ for different problems, it is non-trivial to set the tolerance ϵ. Since we perform the subproblem optimization in the primal, and the objective value $F(\boldsymbol{\omega}_t)$ monotonically decreases. Therefore, in this paper, we propose to use the relative function value difference as the stopping condition instead:

$$\frac{F(\boldsymbol{\omega}_{t-1}, b) - F(\boldsymbol{\omega}_t, b)}{F(\boldsymbol{\omega}_0, b)} \leq \epsilon_c, \tag{27}$$

where ϵ_c is a small tolerance value. In some applications, one may need to select a desired number of features. In such cases, we can terminate Algorithm 1 after a maximum number of T iterations with at most TB features being selected.

5.4.2 Stopping Conditions for Inner iterations

Exact and Inexact FGM: In each iteration of Algorithm 1, one needs to do the inner master problem minimization in (22). The optimality condition of (22) is $\nabla_{\boldsymbol{\omega}} F(\boldsymbol{\omega}) = \mathbf{0}$. In practice, to achieve a solution with high precision to meet this condition is expensive. Therefore, we usually achieve an ϵ-accurate solution instead.

Nevertheless, an inaccurate solution may affect the convergence. To demonstrate this, let $\widehat{\boldsymbol{\omega}}$ and $\widehat{\boldsymbol{\xi}}$ be the exact solution to (22). According to Theorem 3, the exact solution of $\widehat{\boldsymbol{\alpha}}$ to (20) can be obtained by $\widehat{\boldsymbol{\alpha}} = \widehat{\boldsymbol{\xi}}$. Now suppose $\boldsymbol{\omega}$ is an ϵ-accurate solution to (22) and $\boldsymbol{\xi}$ be the corresponding loss, then we have $\alpha_i = \widehat{\alpha}_i + \epsilon_i$, where ϵ_i is the gap between $\widehat{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha}$. When performing the worst-case analysis in Algorithm 2, we need to calculate

$$\mathbf{c} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i=1}^{n} (\widehat{\alpha}_i + \epsilon_i) y_i \mathbf{x}_i = \widehat{\mathbf{c}} + \sum_{i=1}^{n} \epsilon_i y_i \mathbf{x}_i = \widehat{\mathbf{c}} + \Delta\widehat{\mathbf{c}},$$

where $\widehat{\mathbf{c}}$ denotes the exact feature score w.r.t. $\widehat{\boldsymbol{\alpha}}$, and $\Delta\widehat{\mathbf{c}}$ denotes the error of $\mathbf{c}$ brought by the inexact solution. Apparently, we have

$$|\widehat{c}_j - c_j| = |\Delta\widehat{c}_j| = O(\epsilon), \ \forall j = 1, ..., m.$$

Since we only need to find those significant features with the largest $|c_j|'s$, a sufficiently small ϵ is enough such that we can find the most-active constraint. Therefore, the convergence of FGM will not be affected if ϵ is sufficiently small, but overall convergence speed of FGM can be greatly improved. Let $\{\boldsymbol{\omega}^k\}$ be the inner iteration sequence, in this paper, we set the stopping condition of the inner problem as

$$\frac{F(\boldsymbol{\omega}^{k-1}) - F(\boldsymbol{\omega}^k)}{F(\boldsymbol{\omega}^{k-1})} \leq \epsilon_{in}, \tag{28}$$

where ϵ_{in} is a small tolerance value. In practice, we set $\epsilon_{in} = 0.001$, which works well for the problems that will be studied in this paper.

5.5 Cache for Efficient Implementations

The optimization scheme of FGM allows to use some cache techniques to improve the optimization efficiency.

Cache for features. Different from the cache used in kernel SVM which caches kernel entries (Fan et al., 2005), we directly cache the features in FGM. In gradient-based methods, one needs to calculate $\mathbf{w}'\mathbf{x}_i$ for each instance to compute the gradient of the loss function, which takes $O(mn)$ cost in general. Unlike these methods, the gradient computation in the modified APG algorithm of FGM is w.r.t. the selected features only. Therefore, we can use a column-based database to store the data, and cache these features in the main memory to accelerate the feature retrieval. To cache these features, we needs $O(tBn)$ additional memory. However, the operation complexity for feature retrieval can be significantly reduced from $O(nm)$ to $O(tBn)$, where $tB \ll m$ for high dimensional problems. It is worth mentioning that, the cache for features is particularly important for the nonlinear feature selection with explicit feature mappings, where the data with expanded features can be too large to be loaded into the main memory.

Cache for inner products. The cache technique can be also used to accelerate the Algorithm 4. To make a sufficient decrease of the objective value, in Algorithm 4, a line search is performed to find a suitable step size. When doing the line search, one may need to calculate the loss function $P(\boldsymbol{\omega})$ many times, where $\boldsymbol{\omega} = S_\tau(\mathbf{g}) = [\boldsymbol{\omega}_1', ..., \boldsymbol{\omega}_t']'$. The computational cost will be very high if n is very large. However, according to equation (25), we have

$$\boldsymbol{\omega} = S_\tau(\mathbf{g}_h) = \frac{o_h}{||\mathbf{g}_h||}\mathbf{g}_h = \frac{o_h}{||\mathbf{g}_h||}(\mathbf{v}_h - \frac{1}{\tau}\nabla P(\mathbf{v}_h)),$$

where only o_h is affected by the step size. Then the calculation of $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ follows

$$\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i = \sum_{i=1}^n \left(\sum_{h=1}^t \boldsymbol{\omega}_h'\mathbf{x}_{ih}\right) = \sum_{i=1}^n \left(\sum_{h=1}^t \frac{o_h}{||\mathbf{g}_h||}\left(\boxed{\mathbf{v}_h'\mathbf{x}_{ih}} - \frac{1}{\tau}\boxed{\nabla P(\mathbf{v}_h)'\mathbf{x}_{ih}}\right)\right).$$

According to the above calculation rule, we can make a fast computation of $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ by caching $\mathbf{v}_h'\mathbf{x}_{ih}$ and $\nabla P(\mathbf{v}_h)'\mathbf{x}_{ih}$ for the h^{th} group of each instance $\mathbf{x}_i$. Accordingly, the complexity of computing $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ can be reduced from $O(ntB)$ to $O(nt)$. That is to say, no matter how many line search steps will be conducted, we only need to scan the selected features once, which can greatly reduce the computational cost.

6. Nonlinear Feature Selection Through Kernels

By applying the kernel tricks, we can extend FGM to do nonlinear feature selections. Let $\boldsymbol{\phi}(\mathbf{x})$ be a nonlinear feature mapping that maps the input features with nonlinear relations into a high-dimensional linear feature space. To select the features, we can also introduce a scaling vector $\mathbf{d} \in \mathcal{D}$ and obtain a new feature mapping $\boldsymbol{\phi}(\mathbf{x} \odot \sqrt{\mathbf{d}})$. By replacing $(\mathbf{x} \odot \sqrt{\mathbf{d}})$ in (5) with $\boldsymbol{\phi}(\mathbf{x} \odot \sqrt{\mathbf{d}})$, the kernel version of FGM can be formulated as the following semi-infinite kernel (SIK) learning problem:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A},\theta} \theta \quad : \quad \theta \geq f_{\mathbf{K}}(\boldsymbol{\alpha}, \mathbf{d}), \ \ \forall\ \mathbf{d} \in \mathcal{D},$$

where $f_{\mathbf{K}}(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}(\boldsymbol{\alpha}\odot\mathbf{y})'(\mathbf{K}_\mathbf{d}+\frac{1}{C}I)(\boldsymbol{\alpha}\odot\mathbf{y})$ and $\mathbf{K}_\mathbf{d}^{ij}$ is calculated as $\boldsymbol{\phi}(\mathbf{x}_i\odot\sqrt{\mathbf{d}})'\boldsymbol{\phi}(\mathbf{x}_j\odot\sqrt{\mathbf{d}})$. This problem can be solved by Algorithm 1. However, we need to solve the following optimization problem in the worst-case analysis:

$$\max_{\mathbf{d}\in\mathcal{D}} \ \frac{1}{2}\Big\|\sum_{i=1}^n \boldsymbol{\alpha}_i y_i \boldsymbol{\phi}(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\Big\|^2 = \max_{\mathbf{d}\in\mathcal{D}} \ \frac{1}{2}(\boldsymbol{\alpha} \odot \mathbf{y})'\mathbf{K}_\mathbf{d}(\boldsymbol{\alpha} \odot \mathbf{y}), \tag{29}$$

6.1 Worst Case Analysis for Additive Kernels

In general, solving problem (29) for general kernels (*e.g.*, Gaussian kernels) is very challenging. However, for **additive kernels**, this problem can be exactly solved. A kernel $\mathbf{K}_\mathbf{d}$ is an additive kernel if it can be linearly represented by a set of base kernels $\{\mathbf{K}_j\}_{j=1}^p$ (Maji and Berg, 2009). If each base kernel $\mathbf{K}_j$ is constructed by one feature or a subset of features, we can select the optimal subset features by choosing a small subset of kernels.

Proposition 2 *The worst-case analysis w.r.t. additive kernels can be exactly solved.*

Proof Suppose that each base kernel $\mathbf{K}_j$ in an additive kernel is constructed by one feature or a subset of features. Let $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$ be the index set of features that produce the base kernel set $\{\mathbf{K}_j\}_{j=1}^p$ and $\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})$ be the corresponding feature map to $\mathcal{G}_j$. Similar to the group feature selection, we introduce a feature scaling vector $\mathbf{d} \in \mathcal{D} \subset \mathbb{R}^p$ to scale $\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})$. The resultant model becomes:

$$\begin{aligned} \min_{\mathbf{d}\in\widehat{\mathcal{D}}} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i\left(\sum_{j=1}^p \sqrt{d_j}\mathbf{w}_{\mathcal{G}_j}'\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j}) - b\right) \geq 1 - \xi_i, \;\; \xi_i \geq 0, \;\; i = 1, \cdots, n, \end{aligned}$$

where $\mathbf{w}_{\mathcal{G}_j}$ has the same dimensionality with $\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})$. By transforming this problem to the SIP problem in (13), we can solve the kernel learning (selection) problem via FGM. The corresponding worst-case analysis is reduced to solve the following problem:

$$\max_{\mathbf{d}\in\mathcal{D}} \;\sum_{j=1}^p d_j(\boldsymbol{\alpha}\odot\mathbf{y})'\mathbf{K}_j(\boldsymbol{\alpha}\odot\mathbf{y}) = \max_{\mathbf{d}\in\mathcal{D}} \;\sum_{j=1}^p d_j s_j,$$

where $s_j = (\boldsymbol{\alpha}\odot\mathbf{y})'\mathbf{K}_j(\boldsymbol{\alpha}\odot\mathbf{y})$ and $\mathbf{K}_j^{i,k} = \boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})'\boldsymbol{\phi}_j(\mathbf{x}_{k\mathcal{G}_j})$. This problem can be exactly solved by choosing the B kernels with the largest s_j's. ■

In the past decades, many additive kernels have been proposed based on specific application contexts, such as the general intersection kernel in computer vision (Maji and Berg, 2009), string kernel in text mining and ANOVA kernels (Bach, 2009). Taking the general intersection kernel for example, it is defined as: $k(\mathbf{x}, \mathbf{z}, a) = \sum_{j=1}^p \min\{|x_j|^a, |z_j|^a\}$, where $a > 0$ is a kernel parameter. When $a = 1$, it reduces to the well-known Histogram Intersection Kernel (HIK), which has been widely used in computer vision and text classifications (Maji and Berg, 2009; Wu, 2012).

It is worth mentioning that, even though we can exactly solve the worst-case analysis for additive kernels, the subproblem optimization is still very challenging for large-scale problems because of two reasons. Firstly, storing the kernel matrices takes $O(n^2)$ space complexity, which is unbearable when n is very large. Secondly, solving the MKL problem with many training points is still computationally expensive. To address these issues, we propose to a group of approximated features, such as the random features (Vedaldi and Zisserman, 2010) and the HIK expanded features (Wu, 2012), to approximate a base kernel. As a result, the MKL problem is reduced to the **group feature selection** problem. Therefore, it is scalable to big data by avoiding the storage the base kernel matrices. Moreover, the subproblem optimization can be more efficiently solved in the primal form.

6.2 Worst-Case Analysis for Ultrahigh Dimensional Big Data

Ultrahigh dimensional big data widely exist in many application contexts. Particularly, in the nonlinear classification tasks with explicit nonlinear feature mappings, the dimensionality of the feature space can be ultrahigh. If the explicit feature mapping is available, the nontrivial nonlinear feature selection task can be cast as a linear feature selection problem in the high-dimensional feature space.

Taking the polynomial kernel $k(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i' \mathbf{x}_j + r)^\upsilon$ for example, the dimension of the feature mapping exponentially increases with υ (Chang et al., 2010), where υ is referred to as the degree. When $\upsilon = 2$, the 2-degree explicit feature mapping can be expressed as

$$\boldsymbol{\phi}(\mathbf{x}) = [r, \sqrt{2\gamma r}x_1, ..., \sqrt{2\gamma r}x_m, \gamma x_1^2, ..., \gamma x_m^2, \sqrt{2}\gamma x_1 x_2, ..., \sqrt{2}\gamma x_{m-1} x_m].$$

The second-order feature mapping can capture the feature pair dependencies, thus it has been widely applied in many applications such as text mining and natural language processing (Chang et al., 2010). Unfortunately, the dimensionality of the feature space is $(m+2)(m+1)/2$ and can be ultrahigh for a median m. For example, if $m = 10^6$, the dimensionality of the feature space is $O(10^{12})$, and around 1 TB memory is required to store the weight vector $\mathbf{w}$. As a result, most of the existing methods are not applicable (Chang et al., 2010). Fortunately, this computational bottleneck can be effectively avoided by FGM since only tB features are required to be stored in the main memory. For convenience, we store the indices and scores of the selected tB features in a structured array $\mathbf{c}_B$.

Algorithm 5 Incremental Implementation of Algorithm 2 for Ultrahigh Dimensional Data.

```
Given $\boldsymbol{\alpha}$, $B$, number of data groups $k$, feature mapping $\boldsymbol{\phi}(\mathbf{x})$ and a structured array $\mathbf{c}_B$.
1: Split $\mathbf{X}$ into $k$ subgroups $\mathbf{X} = [\mathbf{X}^1, ..., \mathbf{X}^k]$.
2: For $j = 1, ..., k$.
     Calculate the feature score $\mathbf{s}$ w.r.t. $\mathbf{X}^j$ according to $\boldsymbol{\phi}(\mathbf{x}_i)$.
     Sort $\mathbf{s}$ and update $\mathbf{c}_B$.
     For $i = j+1, ..., k$. (**Optional**)
         Calculate the cross feature score $\mathbf{s}$ w.r.t. $\mathbf{X}^j$ and $\mathbf{X}^i$.
         Sort $\mathbf{s}$ and update $\mathbf{c}_B$.
     End
   End
3: Return $\mathbf{c}_B$.
```

For ultrahigh dimensional big data, it can be too huge to be loaded into the main memory, thus the worst-case analysis is still very challenging to be addressed. Motivated by the incremental worst-case analysis for complex group feature selection in Section 4.3, we propose to address the big data challenge in an incremental manner. The general scheme for the incremental implementation is presented in Algorithm 5. Particularly, we partition $\mathbf{X}$ into k small data subset of lower dimensionality as $\mathbf{X} = [\mathbf{X}^1, ..., \mathbf{X}^k]$. For each small data subset, we can load it into memory and calculate the feature scores of the features. In Algorithm 5, the inner loop w.r.t. the iteration index i is only used for the second-order feature selection, where the calculation of feature score for the **cross-features** is required. For instance, in the nonlinear feature selection using the 2-degree polynomial mapping, we need to calculate the feature score of $x_i x_j$.

7. Connections to Related Studies

In this section, we discuss the connections of proposed methods with related studies, such as the ℓ_1-regularization (Jenatton et al., 2011a), active set methods (Roth and Fischer, 2008; Bach, 2009), SimpleMKL (Rakotomamonjy et al., 2008), ℓ_q-MKL (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012), infinite kernel learning (IKL) (Gehler and Nowozin, 2008), SMO-MKL (Vishwanathan et al., 2010), and so on.

7.1 Relation to ℓ_1-regularization

Recall that the ℓ_1-norm of a vector $\mathbf{w}$ can be expressed as a variational form (Jenatton et al., 2011a):

$$\|\mathbf{w}\|_1 = \sum_{j=1}^m |w_j| = \frac{1}{2}\min_{\mathbf{d}\succeq 0}\sum_{j=1}^m \frac{w_j^2}{d_j} + d_j. \tag{30}$$

It is not difficult to verify that, $d_j^* = |w_j|$ holds at the optimum, which indicates that the scale of d_j^* is proportional to $|w_j|$. Therefore, it is meaningless to impose an additional ℓ_1-constraint $||\mathbf{d}||_1 \leq B$ or $||\mathbf{w}||_1 \leq B$ in (30) since both $\mathbf{d}$ and $\mathbf{w}$ are scale-sensitive. As a result, it is not so easy for the ℓ_1-norm methods to control the number of features to be selected as FGM does. On the contrary, in AFS, we bound $\mathbf{d} \in [0,1]^m$.

To demonstrate the connections of AFS to the ℓ_1-norm regularization, we need to make some transformations. Let $\widehat{w}_j = w_j\sqrt{d_j}$ and $\widehat{\mathbf{w}} = [\widehat{w}_1, ..., \widehat{w}_m]'$, the variational form of the problem (5) can be equivalently written as

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{D}}\min_{\widehat{\mathbf{w}},\boldsymbol{\xi},b} \quad & \frac{1}{2}\sum_{j=1}^m \frac{\widehat{w}_j^2}{d_j} + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i(\widehat{\mathbf{w}}'\mathbf{x}_i - b) \geq 1 - \xi_i, \quad i = 1, \cdots, n. \end{aligned}$$

For simplicity, hereby we drop the hat from $\widehat{\mathbf{w}}$ and define a new regularizer $\|\mathbf{w}\|_B^2$ as

$$\|\mathbf{w}\|_B^2 = \min_{\mathbf{d}\succeq\mathbf{0}}\sum_{j=1}^m \frac{w_j^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_1 \leq B, \ \ \mathbf{d} \in [0,1]^m. \tag{31}$$

This new regularizer has the following properties.

Proposition 3 *Given a vector $\mathbf{w} \in \mathbb{R}^m$ with $\|\mathbf{w}\|_0 = \kappa > 0$, where κ denotes the number of nonzero entries in $\mathbf{w}$. Let $\mathbf{d}^*$ be the minimizer of (31), we have: (I) $d_j^* = 0$ if $|w_j| = 0$. (II) If $\kappa \leq B$, then $d_j^* = 1$ for $|w_j| > 0$; else if $\frac{\|\mathbf{w}\|_1}{\max\{|w_j|\}} \geq B$ and $\kappa > B$, then we have $\frac{|w_j|}{d_j^*} = \frac{\|\mathbf{w}\|_1}{B}$ for all $|w_j| > 0$. (III) If $\kappa \leq B$, then $\|\mathbf{w}\|_B = \|\mathbf{w}\|_2$; else if $\frac{\|\mathbf{w}\|_1}{\max\{|w_j|\}} \geq B$ and $\kappa > B$, $\|\mathbf{w}\|_B = \frac{\|\mathbf{w}\|_1}{\sqrt{B}}$.*

The proof can be found in Appendix D.

According to Proposition 3, if $B < \kappa$, $\|\mathbf{w}\|_B$ is equivalent to the ℓ_1-norm regularizer. However, no matter how large the magnitude of $|w_j|$ is, d_j in $\|\mathbf{w}\|_B^2$ is always upper bounded by 1, which lead to two advantages of $\|\mathbf{w}\|_B^2$ over the ℓ_1-norm regularizer. Firstly, by using $\|\mathbf{w}\|_B^2$, the sparsity and the over-fitting problem can be controlled separately by FGM. Specifically, one can choose a proper C to reduce the feature selection bias, and a proper stopping tolerance ϵ_c in (27) or a proper parameter B to adjust the number of features to be selected. Conversely, in the ℓ_1-norm regularized problems, the number of features is determined by the regularization parameter C, but the solution bias may happen if we intend to select a small number of features with a small C. Secondly, by transforming the resultant optimization problem into an SIP problem, a feature generating paradigm has been developed. By iteratively infer the most informative features, this scheme is particularly suitable for dealing with ultrahigh dimensional big data that are infeasible for the existing ℓ_1-norm methods, as shown in Section 6.2.

Proposition 3 can be easily extended to the group feature selection cases and multiple kernel learning cases. For instance, given a $\mathbf{w} \in \mathbb{R}^m$ with p groups $\{\mathcal{G}_1, ..., \mathcal{G}_p\}$, we have $\sum_{j=1}^p ||\mathbf{w}_{\mathcal{G}_j}||_2 = \|\mathbf{v}\|_1$, where $\mathbf{v} = [||\mathbf{w}_{\mathcal{G}_1}||, ..., ||\mathbf{w}_{\mathcal{G}_p}||]' \in \mathbb{R}^p$. Therefore, the above two advantages are also applicable to FGM for group feature selection and multiple kernel learning.

7.2 Connection to Existing AFS Schemes

The proposed AFS scheme is very different from the existing AFS schemes (e.g., Weston et al., 2000; Chapelle et al., 2002; Grandvalet and Canu, 2002; Rakotomamonjy, 2003; Varma and Babu, 2009; Vishwanathan et al., 2010). In existing works, the scaling vector $\mathbf{d} \succeq \mathbf{0}$ $\mathbf{d}$ is not upper bounded. For instance, in the SMO-MKL method (Vishwanathan et al., 2010), the AFS problem is reformulated as the following problem:

$$\min_{\mathbf{d}\succeq\mathbf{0}} \max_{\boldsymbol{\alpha}\in\mathcal{A}} \mathbf{1}'\boldsymbol{\alpha} - \frac{1}{2}\sum_{j=1}^{p} d_j(\boldsymbol{\alpha}\odot\mathbf{y})'\mathbf{K}_j(\boldsymbol{\alpha}\odot\mathbf{y}) + \frac{\lambda}{2}(\sum_j d_j^q)^{\frac{2}{q}},$$

where $\mathcal{A} = \{\boldsymbol{\alpha} | 0 \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \mathbf{y}'\boldsymbol{\alpha} = 0\}$ and $\mathbf{K}_j$ denote a sub-kernel. When $0 \leq q \leq 1$, it induces sparse solutions, but results in non-convex optimization problems. Moreover, the sparsity of the solution is still determined by the regularization parameter λ. Consequently, the solution bias inevitably exists in the SMO-MKL formulation.

A more related work is the ℓ_1-MKL (Bach et al., 2004; Sonnenburg et al., 2006) or the SimpleMKL problem (Rakotomamonjy et al., 2008), which tries to learn a linear combination of kernels. The variational regularizer of SimpleMKL can be written as:

$$\min_{\mathbf{d}\succeq 0} \sum_{j=1}^{p} \frac{||\mathbf{w}_j||^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_1 \leq 1,$$

where p denotes the number of kernels and $\mathbf{w}_j$ represents the parameter vector of the jth kernel in the context of MKL (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012). Correspondingly, the regularizer $||\mathbf{w}||_B^2$ regarding kernels can be expressed as:

$$\min_{\mathbf{d}\succeq 0} \sum_{j=1}^{p} \frac{||\mathbf{w}_j||^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_1 \leq B, \quad \mathbf{d} \in [0,1]^p. \tag{32}$$

To illustrate the difference between (32) and the ℓ_1-MKL, we divide the two constraints in (32) by B, and obtain

$$\sum_{j=1}^{p} \frac{d_j}{B} \leq 1, \quad 0 \leq \frac{d_j}{B} \leq \frac{1}{B}, \forall j \in \{1, ..., p\}.$$

Clearly, the box constraint $\frac{d_j}{B} \leq \frac{1}{B}$ makes (32) different from the variational regularizer in ℓ_1-MKL. Actually, the ℓ_1-norm MKL is only a special case of $||\mathbf{w}||_B^2$ when $B = 1$. Moreover, by extending Proposition 3, we can obtain that if $B > \kappa$, we have $||\mathbf{w}||_B^2 = \sum_{j=1}^{p} ||\mathbf{w}_j||^2$, which becomes a non-sparse regularizer. Another similar work is the ℓ_q-MKL, which generalizes the ℓ_1-MKL to ℓ_q-norm ($q > 1$) (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012). Specifically, the variational regularizer of ℓ_q-MKL can be written as

$$\min_{\mathbf{d} \succeq 0} \sum_{j=1}^{p} \frac{||\mathbf{w}_j||^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_q^2 \leq 1.$$

We can see that, the box constraint $0 \leq \frac{d_j}{B} \leq \frac{1}{B}, \forall j \in \{1, ..., p\}$ is missing in the ℓ_q-MKL. However, when $q > 1$, the ℓ_q-MKL cannot induce sparse solutions, and thus cannot discard non-important kernels or features. Therefore, the underlying assumption for ℓ_q-MKL is that, most of the kernels are relevant for the classification tasks. Finally, it is worth mentioning that, when doing multiple kernel learning, both ℓ_1-MKL and ℓ_q-MKL require to compute and involve all the base kernels. Consequently the computational cost is unbearable for large-scale problems with many kernels.

An infinite kernel learning method is introduced to deal with infinite number of kernels ($p = \infty$) (Gehler and Nowozin, 2008). Specifically, IKL adopts the ℓ_1-MKL formulation (Bach et al., 2004; Sonnenburg et al., 2006), thus it can be considered as a special case of FGM when setting $B = 1$. Due to the infinite number of possible constraints, IKL also adopts the cutting plane algorithm to address the resultant problem. However, it can only include one kernel per iteration; while FGM can include B kernels per iteration. In this sense, IKL is also analogous to the active set methods (Roth and Fischer, 2008; Bach, 2009). For both methods, the worst-case analysis for large-scale problems usually dominates the overall training complexity. For FGM, since it is able to include B kernels per iteration, it obviously reduces the number of worst-case analysis steps, and thus has great computational advantages over IKL. Finally, it is worth mentioning that, based on the IKL formulation, it is non-trivial for IKL to include B kernels per iteration.

7.3 Connection to Multiple Kernel Learning

In FGM, each subproblem is formulated as a SimpleMKL problem (Rakotomamonjy et al., 2008), and any SimpleMKL solver can be used to solve it. For instance, an approximate solution can be also efficiently obtained by a sequential minimization optimization (SMO) (Bach et al., 2004; Vishwanathan et al., 2010). Sonnenburg et al. (2006) proposed a semi-infinite linear programming formulation for MKL which allows MKL to be iteratively solved with SVM solver and linear programming. Xu et al. (2009b) proposed an extended level method to improve the convergence of MKL. More recently, an online ultra-fast MKL algorithm, called as the UFO-MKL, was proposed by Orabona and Jie (2011). However, its $O(1/\epsilon)$ convergence rate is only guaranteed when a strongly convex regularizer $\Omega(\mathbf{w})$ is added to the objective. Without the strongly convex regularizer, its convergence is unclear.

In summary, FGM is different from MKL in several aspects. At first, FGM iteratively includes B new kernels through the worst-case analysis. Particularly, these B kernels will be formed as a base kernel for the MKL subproblem of FGM. From the kernel learning view, FGM provides a new way to construct base kernels. Secondly, since FGM tends to select a subset of kernels, it is especially suitable for MKL with many kernels. Thirdly, to scale MKL to big data, we propose to use the approximated features (or explicit feature mappings) for kernels. As a result, the MKL problem is reduced to a group feature selection problem, and we can solve the subproblem in its primal form.

7.4 Connection to Active Set Methods

Active set methods have been widely applied to address the challenges of large number of features or kernels (Roth and Fischer, 2008; Bach, 2009). Basically, active set methods iteratively include a variable that violates the optimality condition of the sparsity-induced problems. In this sense, active methods can be considered as a special case of FGM with $B = 1$. However, FGM is different from active set methods. Firstly, their motivations are different: active set methods start from the Lagrangian duality of the sparsity-induced problems; while FGM starts from the proposed AFS scheme, solves an SIP problem. Secondly, active set methods only include one active feature/group/kernel at each iteration. Regarding this algorithm, when the desired number of kernels or groups becomes relatively large, active set methods will be very computationally expensive. On the contrary, FGM allows to add B new features/groups/kernels per iteration, which can greatly improve the training efficiency by reducing the number of worst-case analysis. Thirdly, a sequence of $\ell_{2,1}^2$-regularized non-smooth problems are solved in FGM, which is very different from the active set methods. Finally, the de-biasing of solutions is not investigated in the active set methods (Bach, 2009; Roth and Fischer, 2008).

8. Experiments

We compare the performance of FGM with several state-of-the-art baseline methods on three learning tasks, namely the linear feature selection, the ultrahigh dimensional nonlinear feature selection and the group feature selection.[4]

The experiments are organized as follows. Firstly, in Section 8.1, we present the general experimental settings. After that in Section 8.2, we conduct synthetic experiments to study the performance of FGM on the linear feature selection. Moreover, in Section 8.3, we study

[4] In the experiments, some aforementioned methods, such as NMMKL, QCQP-SSVM and SVM-RFE, are not included for comparison due to the high computational cost for the optimization or sub-optimality for the feature selection. Interested readers can refer to (Tan et al., 2010) for the detailed comparisons. We also do not include the ℓ_q-MKL (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012) for comparison since it cannot induce sparse solutions. Instead, we include an ℓ_q-variant, i.e., UFO-MKL (Orabona and Jie, 2011), for comparison. Finally, since IKL is a special case of FGM with $B = 1$, we study its performance through FGM with $B = 1$ instead. Since it is analogous to the active set methods, its performance can be also observed from the results of active set method.

the performance of FGM with the shift of hyperplane. In Section 8.4, we conduct real-world experiments on linear feature selection. In Section 8.5, we conduct ultrahigh dimensional nonlinear feature selection experiments with polynomial feature mappings. Finally, we demonstrate the efficacy of FGM on the group feature selection in Section 8.6.

8.1 Data Sets and General Experimental Settings

Several large-scale and high dimensional real-world data sets are used to verify the performance of different methods. General information of these data sets, such as the average nonzero features per instance, is listed in Table 1.[5] Among them, `epsilon`, `Arxiv astro-ph`, `rcv1.binary` and `kddb` data sets have been split into training set and testing set. For `real-sim`, `aut-avn` and `news20.binary`, we randomly split them into training and testing sets, as shown in Table 1.

Data set	m	n_{train}	n_{test}	# nonzeros per instance	Parameter Range		
					l1-SVM (C)	l1-LR(C)	SGD-SLR(λ_1)
`epsilon`	2,000	400,000	100,000	2,000	[5e-4, 1e-2]	[2e-3, 1e-1]	[1e-4, 8e-3]
`aut-avn`	20,707	40,000	22,581	50	–	–	–
`real-sim`	20,958	32,309	40,000	52	[5e-3, 3e-1]	[5e-3, 6e-2]	[1e-4, 8e-3]
`rcv1`	47,236	677,399	20,242	74	[1e-4, 4e-3]	[5e-5, 2e-3]	[1e-4, 8e-3]
`astro-ph`	99,757	62,369	32,487	77	[5e-3, 6e-2]	[2e-2, 3e-1]	[1e-4, 8e-3]
`news20`	1,355,191	9,996	10,000	359	[5e-3, 3e-1]	[5e-2, 2e1]	[1e-4, 8e-3]
`kddb`	29,890,095	19,264,097	748,401	29	[5e-6, 3e-4]	[3e-6, 1e-4]	[1e-4, 8e-3]

Table 1: Statistics of the data sets used in the experiments. Parameter Range lists the ranges of the parameters for various ℓ_1-methods to select different number of features. The data sets `rcv1` and `aut-avn` will be used in group feature selection tasks.

On the linear feature selection task, comparisons are conducted between FGM and the ℓ_1-regularized methods, including ℓ_1-SVM and ℓ_1-LR. For FGM, we study FGM with SimpleMKL solver (denoted by MKL-FGM)[6] (Tan et al., 2010), FGM with APG method for the squared hinge loss (denoted by PROX-FGM) and the logistic loss (denoted by PROX-SLR), respectively.

Many efficient batch training algorithms have been developed to solve ℓ_1-SVM and ℓ_1-LR, such as the interior point method, fast iterative shrinkage-threshold algorithm (FISTA), block coordinate descent (BCD), Lassplore method (Liu and Ye, 2010), generalized linear model with elastic net (GLMNET) and so on (Yuan et al., 2010, 2011). Among them, LIBLinear, which adopts the coordinate descent to solve the non-smooth optimization problem, has demonstrated state-of-the-art performance in terms of training efficiency (Yuan et al.,

[5] Among these data sets, `epsilon`, `real-sim`, `rcv1.binary`, `news20.binary` and `kddb` can be downloaded at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`, `aut-avn` can be downloaded at `http://vikas.sindhwani.org/svmlin.html` and `Arxiv astro-ph` is from Joachims (2006).

[6] For the fair comparison, we adopt the LIBLinear (e.g., CD-SVM) as the SVM solver in SimpleMKL when performing linear feature selections. The source codes of MKL-FGM are available at `http://www.tanmingkui.com/fgm.html`.

2010). In LIBLinear, by taking the advantages of data sparsity, it achieves very fast convergence speed for sparse data sets (Yuan et al., 2010, 2011). In this sense, we include the LIBLinear solver for comparison[7]. Besides, we take the standard SVM and LR classifier of LIBLinear with all features as the baselines, denoted by CD-SVM and CD-LR, respectively. We use the default stopping criteria of LIBLinear for ℓ_1-SVM, ℓ_1-LR, CD-SVM and CD-LR.

SGD methods have gained great attention for solving large-scale problems (Langford et al., 2009; Shalev-Shwartz and Zhang, 2013). In this experiment, we include the proximal stochastic dual coordinate ascent with logistic loss for comparison (which is denoted by SGD-SLR). SGD-SLR has shown the state-of-the-art performance among various SGD methods (Shalev-Shwartz and Zhang, 2013).[8] In SGD-SLR, there are three important parameters, namely λ_1 to penalize $||\mathbf{w}||_1$, λ_2 to penalize $||\mathbf{w}||_2^2$, and the stopping criterion *min.dgap*. Suggested by the package, in the following experiment, we fix λ_2 = 1e-4 and *min.dgap*=1e-5, and change λ_1 to obtain different levels of sparsity. All the methods are implemented in C++.

On group feature selection tasks, we compare FGM with four recently developed group lasso solvers: FISTA (Liu and Ye, 2010; Jenatton et al., 2011b; Bach et al., 2011), block coordinate descent method (denoted by BCD) (Qin et al., 2010), active set method (denoted by ACTIVE) (Bach, 2009; Roth and Fischer, 2008) and UFO-MKL (Orabona and Jie, 2011). Among them, FISTA has been thoroughly studied by several researchers (Liu and Ye, 2010; Jenatton et al., 2011b; Bach et al., 2011), and we adopt the implementation of SLEP package[9], where an improved line search is used (Liu and Ye, 2010). We implement the block coordinate descent method proposed by Qin et al. (2010), where each subproblem is formulated as a trust-region problem and solved by a Newton's root-finding method (Qin et al., 2010). For UFO-MKL, it is an online optimization method,[10] and we stop the training after 20 epochs. Finally, we implement ACTIVE method based on the SLEP solver. All the methods for group feature selection are implemented in MATLAB for fair comparison.

All the comparisons are performed on a 2.27GHZ Intel(R)Core(TM) 4 DUO CPU running windows sever 2003 with 24.0GB main memory.

8.2 Synthetic Experiments on Linear Feature Selection

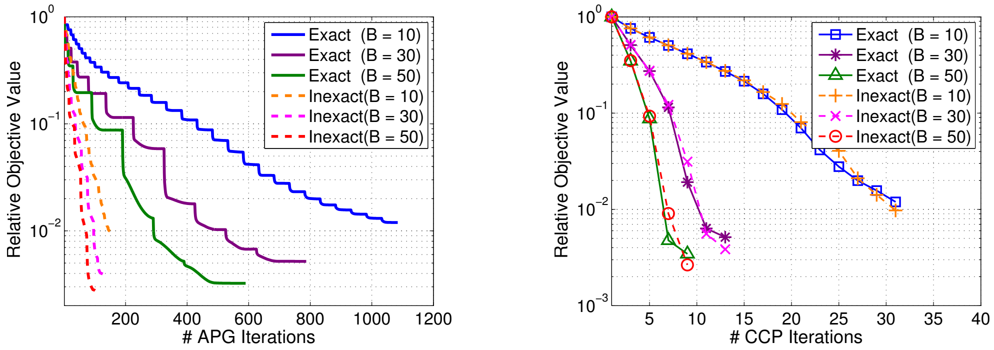
In this section, we compare the performance of different methods on two toy data sets of different scales, namely $\mathbf{X} \in \mathbb{R}^{4,096\times 4,096}$ and $\mathbf{X} \in \mathbb{R}^{8,192\times 65,536}$. Here each $\mathbf{X}$ is a Gaussian random matrix with each entry sampled from the i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. To produce the output $\mathbf{y}$, we first generate a sparse vector $\mathbf{w}$ with 300 nonzero entries, with each nonzero entry sampled from the i.i.d. Uniform distribution $\mathcal{U}(0, 1)$. After that, we produce the output by $\mathbf{y} = \text{sign}(\mathbf{X}\mathbf{w})$. Since only the nonzero w_i contributes to the output $\mathbf{y}$, we consider the corresponding feature as a relevant feature regarding $\mathbf{y}$. Similarly, we generate the testing data set $\mathbf{X}_{\text{test}}$ with output labels $\mathbf{y}_{\text{test}} = \text{sign}(\mathbf{X}_{\text{test}}\mathbf{w})$. The number of testing points for both cases is set to 4,096.

[7] Sources are available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

[8] Sources are available at `http://stat.rutgers.edu/home/tzhang/software.html`.

[9] Sources are available at `http://www.public.asu.edu/~jye02/Software/SLEP/index.htm`.

[10] Sources are available at `http://dogma.sourceforge.net/`.



(a) Relative objective values w.r.t. APG iterations.

(b) Relative objective values w.r.t. outer iterations.

Figure 1: Convergence of Inexact FGM and Exact FGM on the synthetic data set.

8.2.1 Convergence Comparison of Exact and Inexact FGM

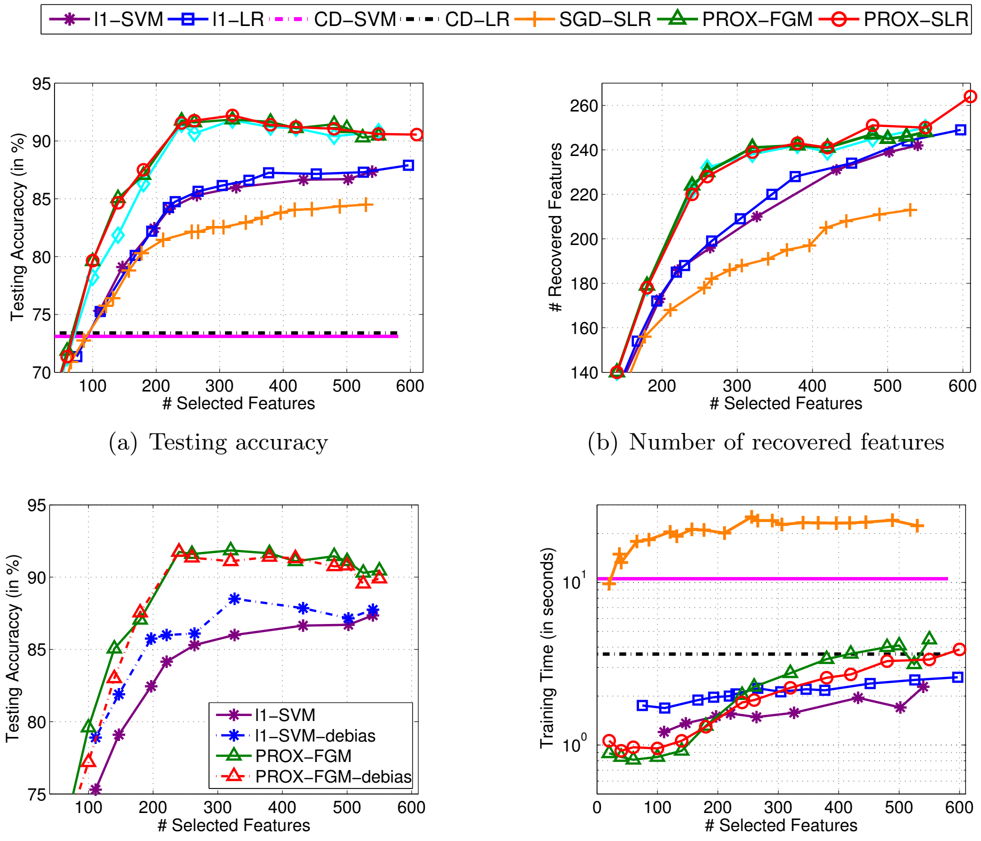
In this experiment, we study the convergence of the *Exact* and *Inexact* FGM on the small scale data set. To study the *Exact* FGM, for simplicity, we set the stopping tolerance $\epsilon_{in} = 1.0 \times 10^{-6}$ in equation (28) for APG algorithm; while for *Inexact* FGM, we set $\epsilon_{in} = 1.0 \times 10^{-3}$. We set $C = 10$ and test different B's from $\{10, 30, 50\}$. In this experiment, only the squared hinge loss is studied. In Figure 1(a), we report the relative objective values w.r.t. all the APG iterations for both methods; In Figure 1(b), we report the relative objective values w.r.t. the outer iterations. We have the following observations from Figures 1(a) and 1(b).

Firstly, from Figure 1(a), for each comparison method, the function value sharply decreases at some iterations, where an active constraint is added. For the *Exact* FGM, it requires more APG iterations under the tolerance $\epsilon_{in} = 1.0 \times 10^{-6}$, but the function value does not show significant decrease after several APG iterations. On the contrary, from Figure 1(a), the *Inexact* FGM, which uses a relatively larger tolerance $\epsilon_{in} = 1.0 \times 10^{-3}$, requires much fewer APG iterations to achieve the similar objective values to *Exact* FGM under the same parameter B. Particularly, from Figure 1(b), the *Inexact* FGM achieves the similar objective values to *Exact* FGM after each outer iteration. According to these observations, on one hand, ϵ_{in} should be small enough such that the subproblem can be sufficiently optimized. On the other hand, a relatively large tolerance (e.g. $\epsilon_{in} = 1.0 \times 10^{-3}$) can greatly accelerate the convergence speed without degrading the performance.

Moreover, according to Figure 1(b), PROX-FGM with a large B in general converges faster than that with a small B. Generally speaking, by using a large B, less number of outer iterations and worst-case analysis are required, which is critical when dealing with big data. However, if B is too large, some non-informative features may be mistakenly included, and the solution may not be exactly sparse.

8.2.2 Experiments on Small-Scale Synthetic Dataset

In this experiment, we evaluate the performance of different methods in terms of testing accuracies w.r.t. different number of selected features. Specifically, to obtain sparse solutions of different sparsities, we vary $C \in [0.001, 0.007]$ for l1-SVM, $C \in$ [5e-3, 4e-2] for l1-LR and $\lambda_1 \in$ [7.2e-4, 2.5e-3] for SGD-SLR.[11] On contrary to these methods, we fix $C = 10$ and choose even numbers in $\{2, 4, ..., 60\}$ for B to obtain different number of features. It can be seen that, it is much easier for FGM to control the number of features to be selected. Specifically, the testing accuracies and the number of recovered ground-truth features w.r.t. the number of selected features are reported in Figure 2(a) and Figure 2(b), respectively. The training time of different methods is listed in Figure 2(d).



(a) Testing accuracy

(b) Number of recovered features

(c) De-biased results

(d) Training time

Figure 2: Experimental results on the small data set, where CD-SVM and CD-LR denote the results of standard SVM and LR with all features, respectively. The training time of MKL-FGM is about 1,500 seconds, which is up to 1,000 times slower than APG solver. We did not report it in the figures due to presentation issues.

For convenience of presentation, let m_s and m_g be the number of selected features and the number of ground-truth features, respectively. From Figure 2(a) and Figure 2(b), FGM

[11] Here, we carefully choose C or λ_1 for these three ℓ_1-methods such that the numbers of selected features uniformly spread over the range [0, 600]. Since the values of C and λ_1 change a lot for different problems, hereafter we only give their ranges. Under this experimental setting, the results of ℓ_1-methods cannot be further improved through parameter tunings.

based methods demonstrate better testing accuracy than all ℓ_1-methods when $m_s > 100$. Correspondingly, from Figure 2(b), under the same number of selected features, FGM based methods include more ground-truth features than ℓ_1-methods when $m_s{\geq}100$. For SGD-SLR, it shows the worst testing accuracy among the comparison methods, and also recovers the least number of ground-truth features.

One of the possible reasons for the inferior performance of the ℓ_1-methods, as mentioned in the Introduction section, is the solution bias brought by the ℓ_1-regularization. To demonstrate this, we do re-training to reduce the bias using CD-SVM with $C = 20$ with the selected features, and then do the prediction using the de-biased models. The results are reported in Figure 2(c), where l1-SVM-debias and PROX-FGM-debias denote the de-biased counterparts for l1-SVM and PROX-FGM, respectively. In general, *if there was no feature selection bias, both FGM and l1-SVM should have the similar testing accuracy to their de-biased counterparts*. However, from Figure 2(c), l1-SVM-debias in general has much better testing accuracy than l1-SVM; while PROX-FGM has similar or even better testing accuracy than PROX-FGM-debias and l1-SVM-debias. These observations indicate that: 1) the solution bias indeed exists in the ℓ_1-methods and affects the feature selection performance; 2) FGM can reduce the feature selection bias.

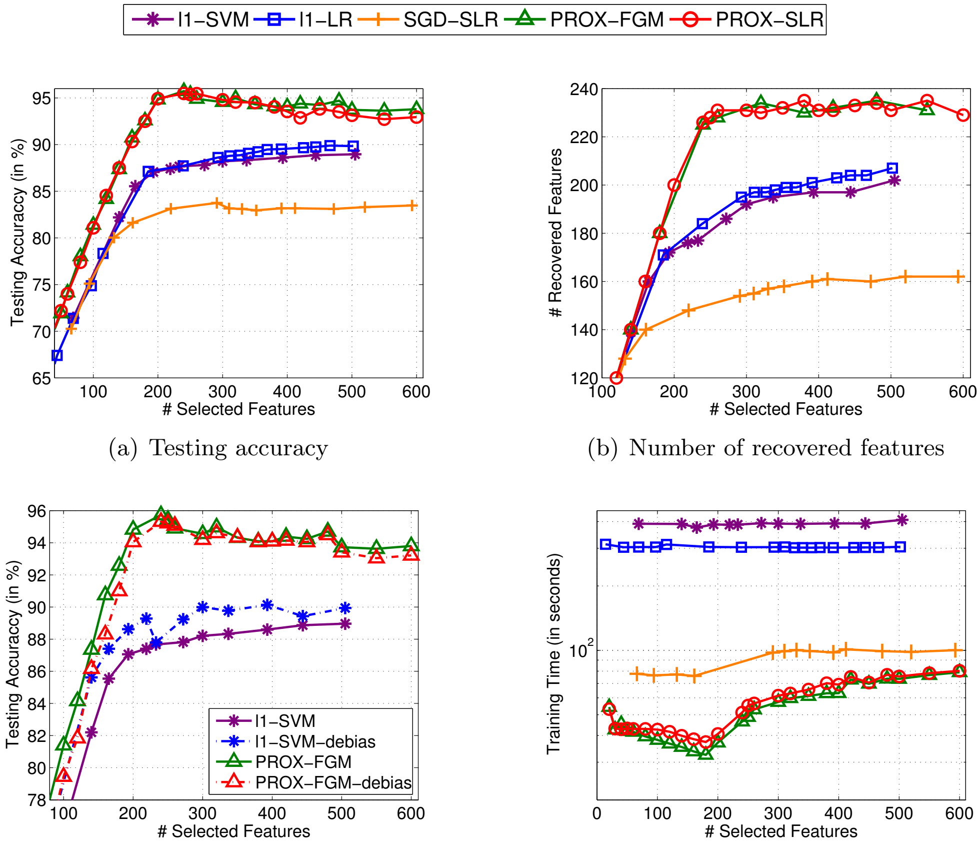
From Figure 2(d), on this small-scale data set, PROX-FGM and PROX-SLR achieve comparable efficiency with the LIBlinear solver. On the contrary, SGD-SLR, which is a typical stochastic gradient method, spends the longest training time. This observation indicates that SGD-SLR method may not be suitable for small-scale problems. Finally, as reported in the caption of Figure 2(d), PROX-FGM and PROX-SLR are up to 1,000 times faster than MKL-FGM using SimpleMKl solver. The reason is that, SimpleMKl uses the subgradient methods to address the non-smooth optimization problem with n variables; While in PROX-FGM and PROX-SLR, the subproblem is solved in the primal problem w.r.t. a small number of selected variables.

Finally, from Figure 2, if the number of selected features is small ($m_s < 100$), the testing accuracy is worse than CD-SVM and CD-LR with all features. However, if sufficient number ($m_s > 200$) of features are selected, the testing accuracy is much better than CD-SVM and CD-LR with all features, which verifies the importance of the feature selection.

8.2.3 Experiments on Large-scale Synthetic Dataset

To demonstrate the scalability of FGM, we conduct an experiment on a large-scale synthetic data set, namely $\mathbf{X} \in \mathbb{R}^{8,192\times 65,536}$. Here, we do not include the comparisons with MKL-FGM due to its high computational cost. For PROX-FGM and PROX-SLR, we follow their experimental settings above. For l1-SVM and l1-LR, we vary $C \in [0.001, 0.004]$ and $C \in [0.005, 0.015]$ to determine the number of features to be selected, respectively. The testing accuracy, the number of recovered ground-truth features, the de-biased results and the training time of the compared methods are reported in Figure 3(a), 3(b), 3(c) and 3(d), respectively.

From Figure 3(a) and 3(b) and 3(c), both PROX-FGM and PROX-SLR outperform l1-SVM, l1-LR and SGD-SLR in terms of both testing accuracy and the number of recovered ground-truth features. From Figure 3(d), PROX-FGM and PROX-SLR show better training efficiency than the coordinate based methods (namely, LIBlinear) and the SGD



(a) Testing accuracy

(b) Number of recovered features

(c) De-biased results

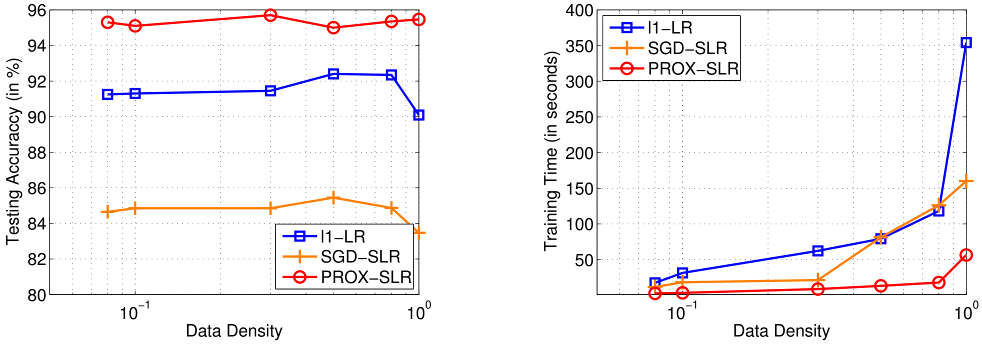
(d) Training time

Figure 3: Performance comparison on the large-scale synthetic data set.

based method (namely SGD-SLR). Basically, FGM solves a sequence of small optimization problems of $O(ntB)$ cost, and spends only a small number of iterations to do the worst-case analysis of $O(mn)$ cost. On the contrary, the ℓ_1-methods may take many iterations to converge, and each iteration takes $O(mn)$ cost. On this large-scale data set, SGD-SLR shows faster training speed than LIBlinear, but it has much inferior testing accuracy over other methods.

In LIBlinear, the efficiency has been improved by taking the advantage of the data sparsity. Considering this, we investigate the sensitivity of the referred methods to the data density. To this end, we generate data sets of different data densities by sampling the entries from $\mathbf{X}^{8,192\times 65,656}$ with different data densities in $\{0.08, 0.1, 0.3, 0.5, 0.8, 1\}$, and study the influence of the data density on different learning algorithms. For FGM, only the logistic loss is studied (e.g. PROX-SLR). We use the default experimental settings for PROX-SLR, and watchfully vary $C \in [0.008, 5]$ for l1-LR and $\lambda_1 \in$ [9.0e-4, 3e-3] for SGD-SLR. For the sake of brevity, we only report the **best accuracy** obtained over all parameters, and the corresponding training time of l1-LR, SGD-SLR and PROX-SLR in Figure 4.

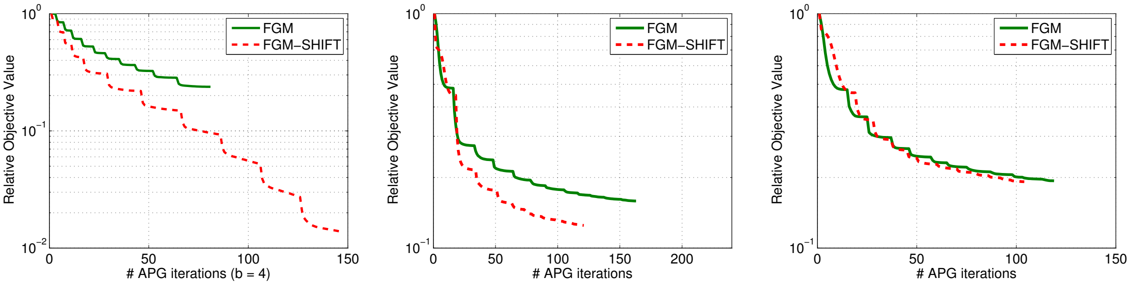
From Figure 4(a), under different data densities, PROX-SLR always outperforms l1-SVM and SGD-SLR in terms of the **best accuracy**. From Figure 4(b), l1-SVM shows



(a) Testing accuracy

(b) Training time

Figure 4: Performance comparison on the synthetic data set ($n = 8,192$, $m = 65,536$) with different data densities in $\{0.08, 0.1, 0.3, 0.5, 0.8, 1\}$.



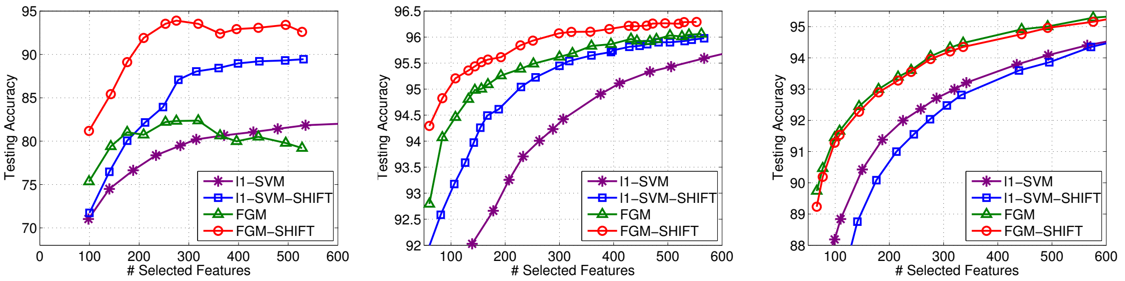
(a) Synthetic data set ($b = 4$)

(b) `astro-ph`

(c) `real-sim`

Figure 5: Relative objective values regarding each APG iteration, where $b = 4$ in the caption of Figure 5(a) denotes the ground-truth shift of the hyperplane from the origin.

comparable efficiency with PROX-SLR on data sets of low data density. However, on relative denser data sets, PROX-SLR is much more efficient than l1-SVM, which indicates that FGM has a better scalability than l1-SVM on dense data.



(a) Synthetic data set ($b = 4$)

(b) `astro-ph`

(c) `real-sim`

Figure 6: Testing accuracy of different methods on the three data data sets.

8.3 Feature Selection with Shift of Hyperplane

In this section, we study the effectiveness of the shift version of FGM (denoted by FGM-SHIFT) on a synthetic data set and two real-world data sets, namely `real-sim` and `astro-ph`. We follow the data generation in Section 7.1 to generate the synthetic data set except that we include a shift term b for the hyperplane when generating the output $\mathbf{y}$. Specifically, we produce $\mathbf{y}$ by $\mathbf{y} = \text{sign}(\mathbf{Xw} - b\mathbf{1})$, where $b = 4$. The shift version of ℓ_1-SVM by LIBlinear (denoted by l1-SVM-SHIFT) is adopted as the baseline. In Figure 5, we report the relative objective values of FGM and FGM-SHIFT w.r.t. the APG iterations on three data sets. In Figure 6, we report the testing accuracy versus different number of selected features.

From Figure 5, FGM-SHIFT indeed achieves much lower objective values than FGM on the synthetic data set and `astro-ph` data set, which demonstrates the effectiveness of FGM-SHIFT. On the `real-sim` data set, FGM and FGM-SHIFT achieve similar objective values, which indicates that the shift term on `real-sim` is not significant. As a result, FGM-SHIFT may not significantly improve the testing accuracy.

From Figure 6, on the synthetic data set and `astro-ph` data set, FGM-SHIFT shows significant better testing accuracy than the baseline methods, which coincides with the better objective values of FGM-SHIFT in Figure 5. l1-SVM-SHIFT also shows better testing accuracy than l1-SVM, which verifies the importance of shift consideration for l1-SVM. However, on the `real-sim` data set, the methods with shift show similar or even inferior performances over the methods without shift consideration, which indicates that the shift of the hyperplane from the origin is not significant on the `real-sim` data set. Finally, FGM and FGM-SHIFT are always better than the counterparts of l1-SVM.

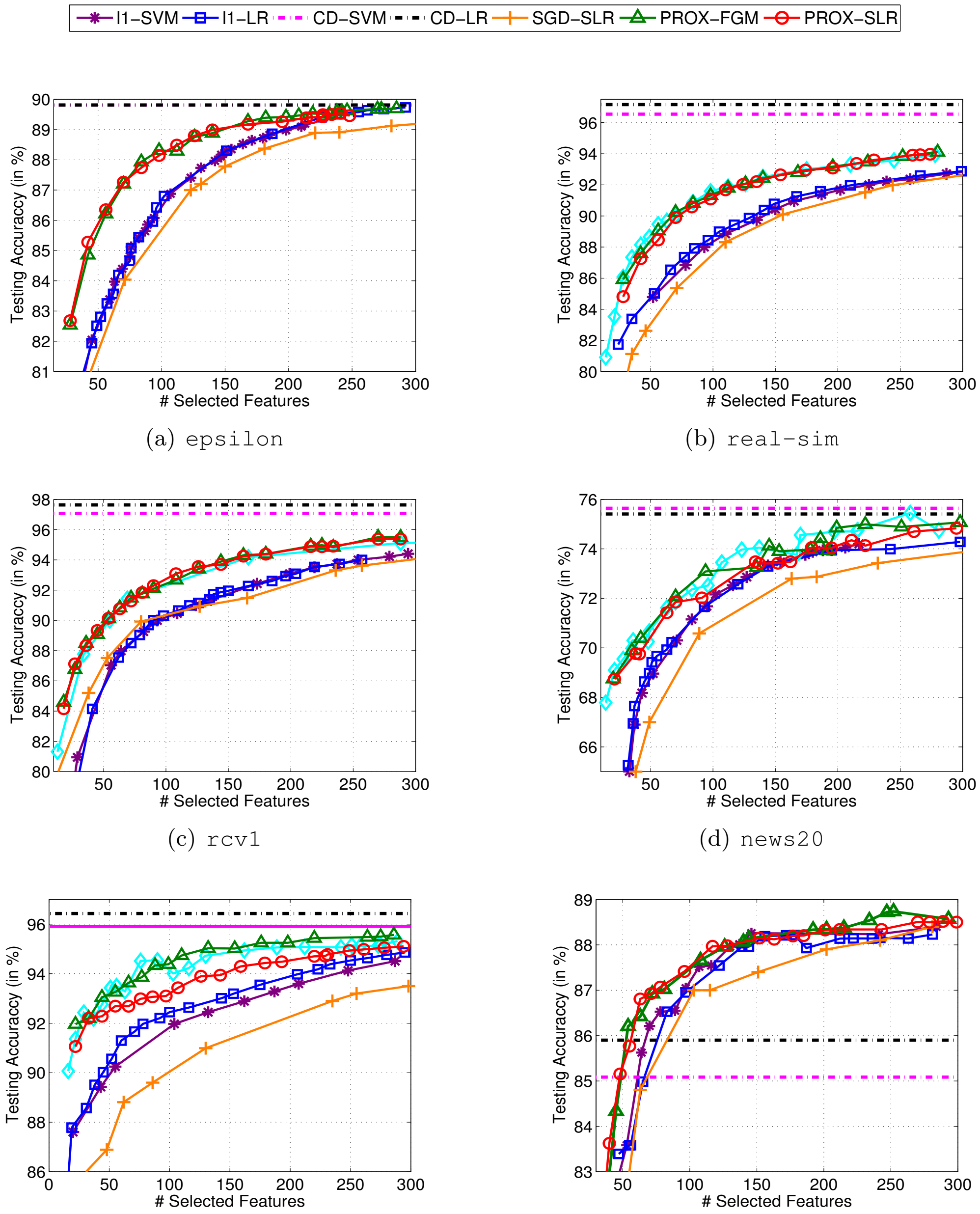
8.4 Performance Comparison on Real-World Data Sets

In this section, we conduct three experiments to compare the performance of FGM with the referred baseline methods on real-world data sets. In Section 8.4.1, we compare the performance of different methods on six real-world data sets. In Section 8.4.2, we study the feature selection bias issue. Finally, in Section 8.4.3, we conduct the sensitivity study of parameters for FGM.

8.4.1 Experimental Results on Real-World Data Sets

On real-world data sets, the number of ground-truth features is unknown. We only report the testing accuracy versus different number of selected features. For FGM, we fix $C = 10$, and vary $B \in \{2, 4, ..., 60\}$ to select different number of features. For the ℓ_1-methods, we watchfully vary the regularization parameter to select different number of features. The ranges of C and λ_1 for ℓ_1-methods are listed in Table 1.

The testing accuracy and training time of different methods against the number of selected features are reported in Figure 7 and Figure 8, respectively. From Figure 7, on all data sets, FGM (including PROX-FGM, PROX-SLR and MKL-FGM) obtains comparable or better performance than the ℓ_1-methods in terms of testing accuracy within 300 features. Particularly, FGM shows much better testing accuracy than ℓ_1-methods on five of the studied data sets, namely `epsilon`, `real-sim`, `rcv1.binary`, `Arxiv astro-ph` and `news20`.



(a) epsilon

(b) real-sim

(c) rcv1

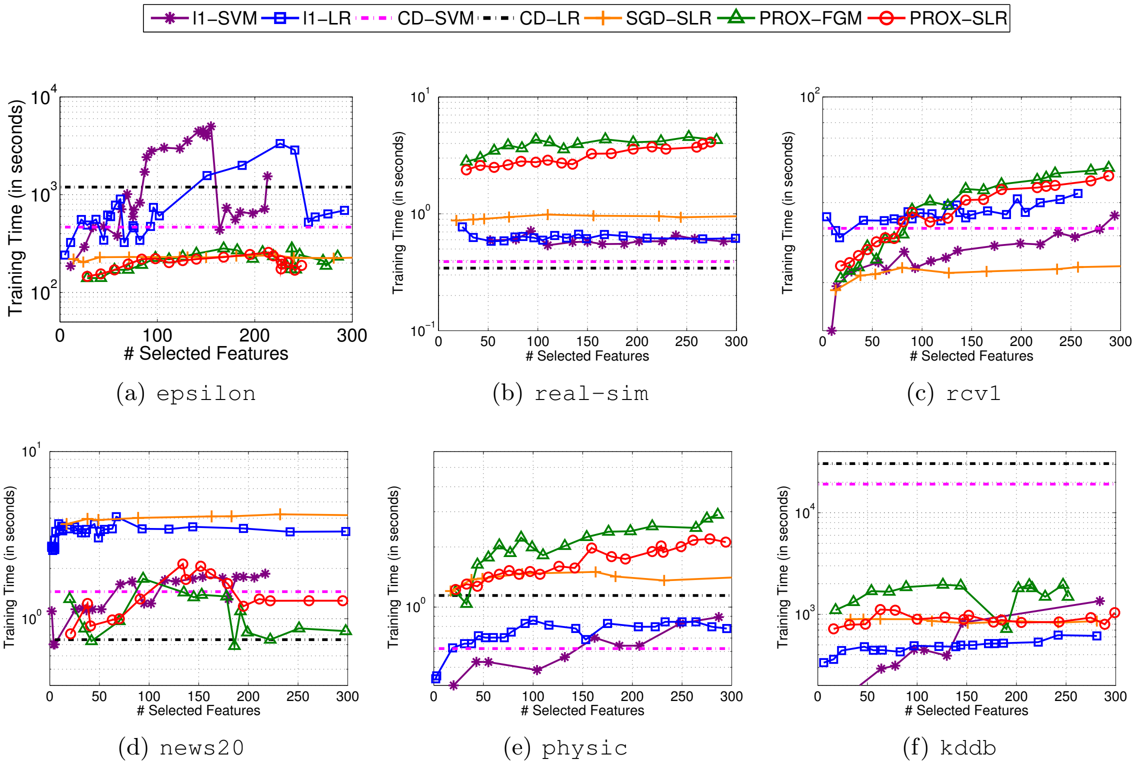
(d) news20

(e) astro-ph

(f) kddb

Figure 7: Testing accuracy on various data sets.

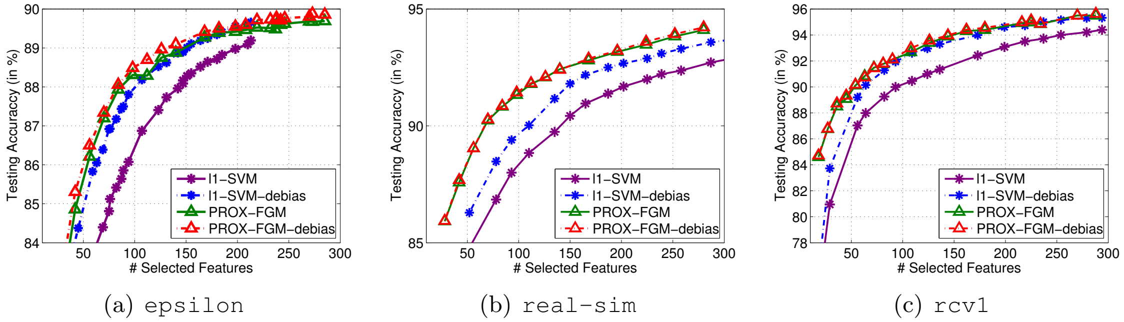
From Figure 8, PROX-FGM and PROX-SLR show competitive training efficiency with the ℓ_1-methods. Particularly, on the large-scale dense epsilon data set, PROX-FGM and PROX-SLR are much efficient than the LIBlinear ℓ_1-solvers. For SGD-SLR, although it demonstrates comparable training efficiency with PROX-FGM and PROX-SLR, it attains much worse testing accuracy. In summary, FGM based methods in general obtain better



(a) epsilon (b) real-sim (c) rcv1

(d) news20 (e) physic (f) kddb

Figure 8: Training time on various data sets.



(a) epsilon (b) real-sim (c) rcv1

Figure 9: De-biased results on real-world data sets.

feature subsets with competitive training efficiency with the considered baselines on real-world data sets.

8.4.2 De-biasing Effect of FGM

In this experiment, we demonstrate the de-biasing effect of FGM on three real-world data sets, namely epsilon, real-sim and rcv1. Here, only the squared hinge loss (namely PFOX-FGM) is studied. The de-biased results are reported in Figure 9, where PROX-

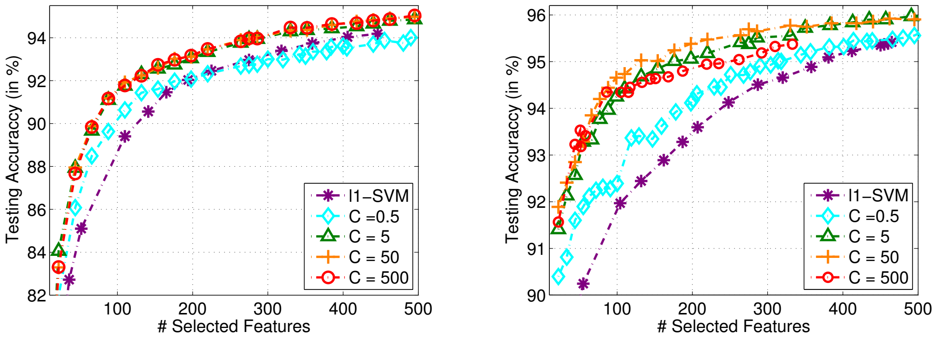
FGM-debias and l1-SVM-debias denote the de-biased results of PROX-FGM and l1-SVM, respectively.

From Figure 9, l1-SVM-debias shows much better results than l1-SVM, indicating that the feature selection bias issue exists in l1-SVM on these real-world data sets. On the contrary, PROX-FGM achieves close or even better results compared with its de-biased counterparts, which verifies that PROX-FGM itself can reduce the feature selection bias. Moreover, on these data sets, FGM shows better testing accuracy than the de-biased l1-SVM, namely l1-SVM-debias, which indicates that the features selected by FGM are more relevant than those obtained by l1-SVM due to the reduction of feature selection bias.

8.4.3 Sensitivity Study of Parameters

In this section, we conduct the sensitivity study of parameters for PROX-FGM. There are two parameters in FGM, namely the sparsity parameter B and the regularization parameter C. In this experiments, we study the sensitivity of these two parameters on `real-sim` and `astro-ph` data sets. l1-SVM is adopted as the baseline.

In the first experiment, we study the sensitivity of C. FGM with suitable C can reduce the feature selection bias. However, C is too large, the over-fitting problem may happen. To demonstrate this, we test $C \in \{0.5, 5, 50, 500\}$. The testing accuracy of FGM under different C's is reported in Figure 10. From Figure 10, the testing accuracy with small a C in general is worse than that with a large C. The reason is that, when C is small, feature selection bias may happen due to the under-fitting problem. However, when C is sufficient large, increasing C may not necessarily improve the performance. More critically, if C is too large, the over-fitting problem may happen. For example, on the `astro-ph` data set, FGM with $C = 500$ in general performs much worse than FGM with $C = 5$ and $C = 50$. Another important observation is that, on both data sets, FGM with different C's generally performs better than the l1-SVM.

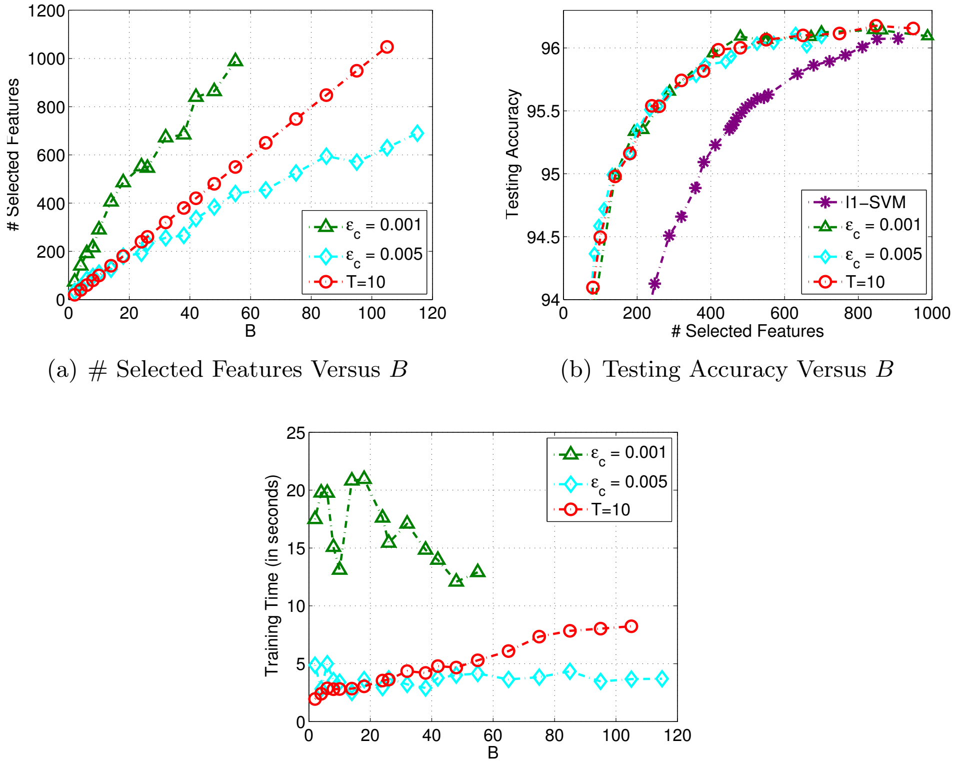


(a) `real-sim` (b) `astro-ph`

Figure 10: Sensitivity of the parameter C for FGM on `real-sim` and `astro-ph` data sets.

Recall that, a large C may lead to slower convergence speed due to the increasing of the Lipschitz constant of $F(\boldsymbol{\omega}, b)$. In practice, we suggest choosing C in the range of

$[1, 100]$. In Section 8.4, we have set $C = 10$ for all data sets. Under this setting, FGM has demonstrated superb performance over the competing methods. On the contrary, choosing the regularization parameter for ℓ_1-methods is more difficult. In other words, FGM is more convenient to do model selections.



(a) # Selected Features Versus B

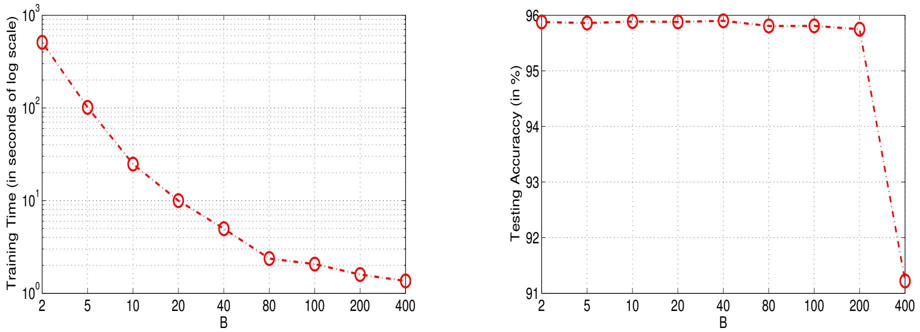
(b) Testing Accuracy Versus B

(c) Training Time (in seconds)

Figure 11: Sensitivity of the parameter B for FGM on `astro-ph` data set, where FGM is stopped once $(F(\boldsymbol{\omega}_{t-1}, b) - F(\boldsymbol{\omega}_t, b))/F(\boldsymbol{\omega}_0, b) \leq \epsilon_c$.

In the second experiment, we study the sensitivity of parameter B for FGM under two stopping conditions: (1) the condition $(F(\boldsymbol{\omega}_{t-1}, b) - F(\boldsymbol{\omega}_t, b))/F(\boldsymbol{\omega}_0, b) \leq \epsilon_c$ is achieved; (2) a maximum T iterations is achieved, where $T = 10$. Here, we test two values of ϵ_c, namely $\epsilon_c = 0.005$ and $\epsilon_c = 0.001$. The number of selected features, the testing accuracy and the training time versus different B are reported in Figure 11(a), 11(b) and 11(c), respectively.

In Figure 11(a), given the number of selected feature # features, the number of required iterations is about $\lceil \frac{\#\text{ features}}{B} \rceil$ under the first stopping criterion. In this sense, FGM with $\epsilon_c = 0.001$ takes more than 10 iterations to terminate, thus will choose more features. As a result, it needs more time for the optimization with the same B, as shown in Figure 11(c). On the contrary, FGM with $\epsilon_c = 0.005$ requires fewer number of iterations (smaller than 10 when $B > 20$). Surprisingly, as shown in Figure 11(b), FGM with fewer iterations (where $\epsilon_c = 0.005$ or $T = 10$) obtain similar testing accuracy with FGM using $\epsilon_c = 0.001$, but has much better training efficiency. This observation indicates that, we can set a small number outer iterations (for example $5 \leq T \leq 20$) to trade-off the training efficiency and the feature selection performance.



(a) Training Time (in seconds) Versus B

(b) Testing Accuracy Versus B

Figure 12: Sensitivity of the parameter B for FGM on astro-ph data set. Given a parameter B, we stop FGM once 400 features are selected.

In the third experiment, we study the influence of the parameter B on the performance of FGM on the astro-ph data set. For convenience of comparison, we stop FGM once 400 features are selected w.r.t. different B's.

The training time and testing accuracy w.r.t. different B's are shown in Figure 12(a) and 12(b), respectively. From Figure 12(a), choosing a large B in general leads to better training efficiency. Particularly, FGM with $B = 40$ is about 200 times faster than FGM with $B = 2$. Recall that, active set methods can be considered as special cases of FGM with $B = 1$ (Roth and Fischer, 2008; Bach, 2009). Accordingly, we can conclude that, FGM with a properly selected B can be much faster than active set methods. However, it should be pointed that, if B is too large, the performance may degrade. For instance, if we choose $B = 400$, the testing accuracy dramatically degrades, which indicates that the selected 400 features are not the optimal ones. In summary, choosing a suitable B (e.g. $B \leq 100$) can much improve the efficiency while maintaining promising generalization performance.

8.5 Ultrahigh Dimensional Feature Selection via Nonlinear Feature Mapping

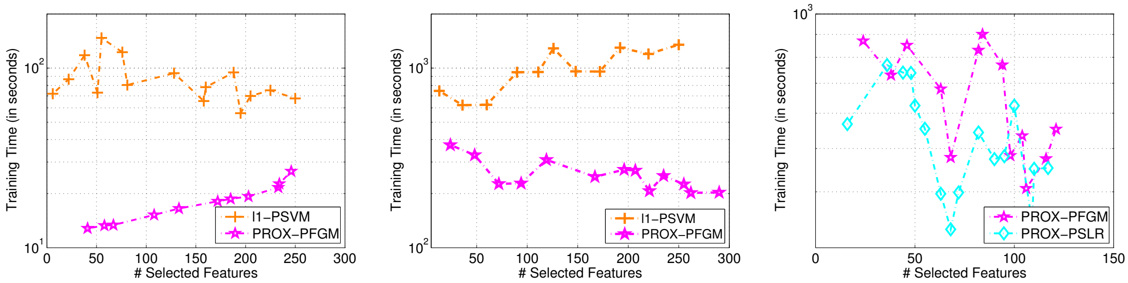
In this experiment, we compare the efficiency of FGM and ℓ_1-SVM on nonlinear feature selections using polynomial feature mappings on two medium dimensional data sets and a high dimensional data set. The comparison methods are denoted by PROX-PFGM, PROX-PSLR and l1-PSVM, respectively.[12] The details of the studied data sets are shown in Table 2, where m_{Poly} denotes the dimension of the polynomial mappings and γ is the polynomial kernel parameter used in this experiment. The mnist38 data set consists of the digital images of 3 and 8 from the mnist data set.[13] For the kddb data set, we only use the first 10^6 instances. Finally, we change the parameter C for l1-PSVM to obtain different number of features.

[12] The codes of l1-PSVM are available at http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#fast_training_testing_for_degree_2_polynomial_mappings_of_data.

[13] The data set is available from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Data set	m	m_{Poly}	n_{train}	n_{test}	γ
`mnist38`	784	$O(10^5)$	40,000	22,581	4.0
`real-sim`	20,958	$O(10^8)$	32,309	40,000	8.0
`kddb`	4,590,807	$O(10^{14})$	1000,000	748,401	4.0

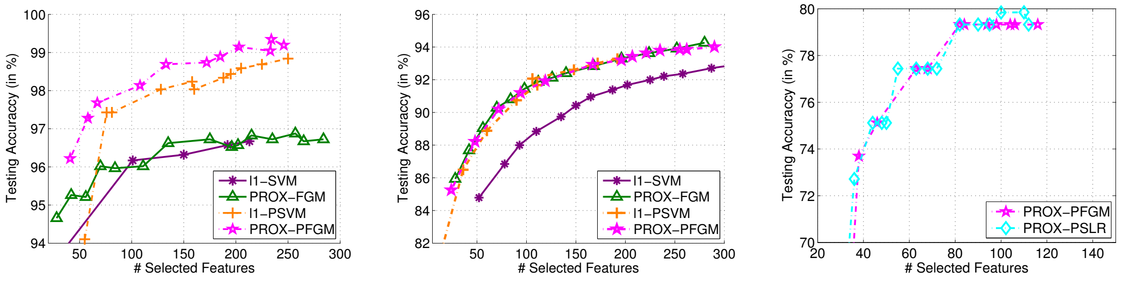
Table 2: Details of data sets using polynomial feature mappings.



(a) `mnist38` (b) `real-sim` (c) `kddb`

Figure 13: Training time of different methods on nonlinear feature selection using polynomial mappings.

The training time and testing accuracy on different data sets are reported in Figure 13 and 14, respectively. Both PROX-PFGM and l1-PSVM can address the two medium dimensional problems. However, PROX-PFGM shows much better efficiency than l1-PSVM. Moreover, l1-PSVM is infeasible on the `kddb` data set due to the ultrahigh dimensionality. Particularly, in l1-PSVM, it needs more than 1TB memory to store a dense $\mathbf{w}$, which is infeasible for a common PC. Conversely, this difficulty can be effectively addressed by FGM. Specifically, PROX-PFGM completes the training within 1000 seconds.



(a) `mnist38` (b) `real-sim` (c) `kddb`

Figure 14: Testing accuracy of different methods on nonlinear feature selection using polynomial mappings.

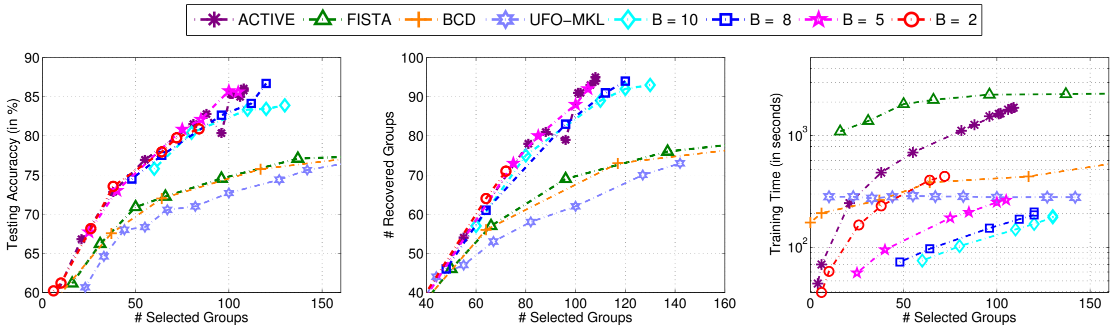
From the figures, the testing accuracy on `mnist38` data set with polynomial mapping is much better than that of linear methods, which demonstrate the usefulness of the nonlinear feature expansions. On the `real-sim` and `kddb` data sets, however, the performance with polynomial mapping does not show significant improvements. A possible reason is that these two data sets are linearly separable.

8.6 Experiments for Group Feature Selection

In this section, we study the performance of FGM for group feature selection on a synthetic data set and two real-world data sets. Here only the logistic loss is studied since it has been widely used for group feature selections on classification tasks (Roth and Fischer, 2008; Liu and Ye, 2010). To demonstrate the sensitivity of the parameter C to FGM, we vary C to select different number of groups under the stopping tolerance $\epsilon_c = 0.001$. For each C, we test $B \in \{2, 5, 8, 10\}$. The tradeoff parameter λ in SLEP is chosen from $[0, 1]$, where a larger lambda leads to more sparse solutions (Liu and Ye, 2010). Specifically, we set λ in $[0.002, 0.700]$ for FISTA and ACTIVE, and set λ in $[0.003, 0.1]$ for BCD.

8.6.1 Synthetic Experiments on Group Feature Selection

In the synthetic experiment, we generate a random matrix $\mathbf{X} \in \mathbb{R}^{4{,}096\times 400{,}000}$ with each entry sampled from the i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. After that, we directly group the 400,000 features into 40,000 groups of equal size (Jenatton et al., 2011b), namely each feature group contains 10 features. We randomly choose 100 groups of them as the ground-truth informative groups. To this end, we generate a sparse vector $\mathbf{w}$, where only the entries of the selected groups are nonzero values sampled from the i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. Finally, we produce the output labels by $\mathbf{y} = \text{sign}(\mathbf{Xw})$. We generate 2,000 testing points in the same manner.



(a) Testing accuracy (b) Recovered features (c) Training time

Figure 15: Results of group feature selection on the synthetic data set.

The testing accuracy, training time and number of recovered ground-truth groups are reported in Figure 15(a), 15(b) and 15(c), respectively. Here only the results within 150 groups are included since we only have 100 informative ground-truth groups. From Figure 15(a), FGM achieves better testing accuracy than FISTA, BCD and UFO-MKL. The reason is that, FGM can reduce the group feature selection bias. From Figure 15(c), in general, FGM is much more efficient than FISTA and BCD. Interestingly, the active set method (denoted by ACTIVE) also shows good testing accuracy compared with FISTA and BCD, but from Figure 15(c), its efficiency is limited since it only includes one element per iteration. Accordingly, when selecting a large number of groups on big data, its computational cost becomes unbearable. For UFO-MKL, although its training speed is fast, its testing accuracy is generally worse than others. Finally, with a fixed B for FGM, the number of selected groups will increase when C becomes large. This is because, with a larger C, one imposes more importance on the training errors, more groups are required to achieve lower empirical errors.

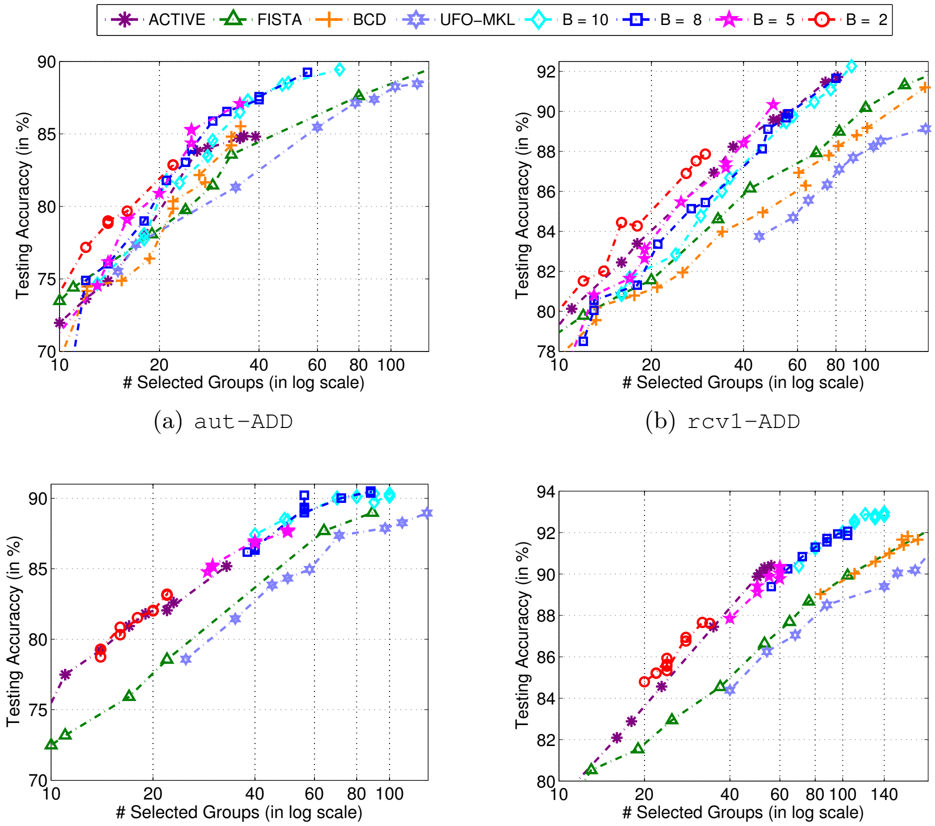
Data set	m	n_{train}	Size of training set (GB)			n_{test}	Size of testing set(GB)		
			Linear	ADD	HIK		Linear	ADD	HIK
aut	20,707	40, 000	0.027	0.320	0.408	22,581	0.016	0.191	0.269
rcv1	47,236	677,399	0.727	8.29	9.700	20,242	0.022	0.256	0.455

Table 3: Details of data sets used for HIK kernel feature expansion and Additive kernel feature expansion. For HIK kernel feature expansion, each original feature is represented by a group of 100 features; while for Additive kernel feature expansion, each original feature is represented by a group of 11 features.

8.6.2 Experiments on Real-World Data Sets

In this section, we study the effectiveness of FGM for group feature selection on two real-world data sets, namely `aut-avn` and `rcv1`. In real-world applications, the group prior of features comes in different ways. In this paper, we produce the feature groups using the explicit kernel feature expansions (Wu, 2012; Vedaldi and Zisserman, 2010), where each original feature is represented by a group of approximated features. Such expansion can vastly improve the training efficiency of kernel methods while keeping good approximation performance in many applications, such as in computer vision (Wu, 2012). For simplicity, we only study the HIK kernel expansion (Wu, 2012) and the additive Gaussian kernel expansion (Vedaldi and Zisserman, 2010). In the experiments, for fair comparisons, we pre-generate the explicit features for two data sets. The details of the original data sets and the expanded data sets are listed in Table 3. We can observe that, after the feature expansion, the storage requirements dramatically increase.

Figure 16 and 17 report the testing accuracy and training time of different methods, respectively. From Figure 16, FGM and the active set method achieve superior performance over FISTA, BCD and UFO-MKL in terms of testing accuracy. Moreover, from Figure 17, FGM gains much better efficiency than the active set method. It is worth mentioning that, due to the unbearable storage requirement, the feature expansion cannot be explicitly stored when dealing with ultrahigh dimensional big data. Accordingly, FISTA and BCD, which require the explicit presentation of data, cannot work in such cases. On the contrary, the proposed feature generating paradigm can effectively address this computational issue since it only involves a sequence of small-scale optimization problems.



(a) aut-ADD

(b) rcv1-ADD

(c) aut-HIK

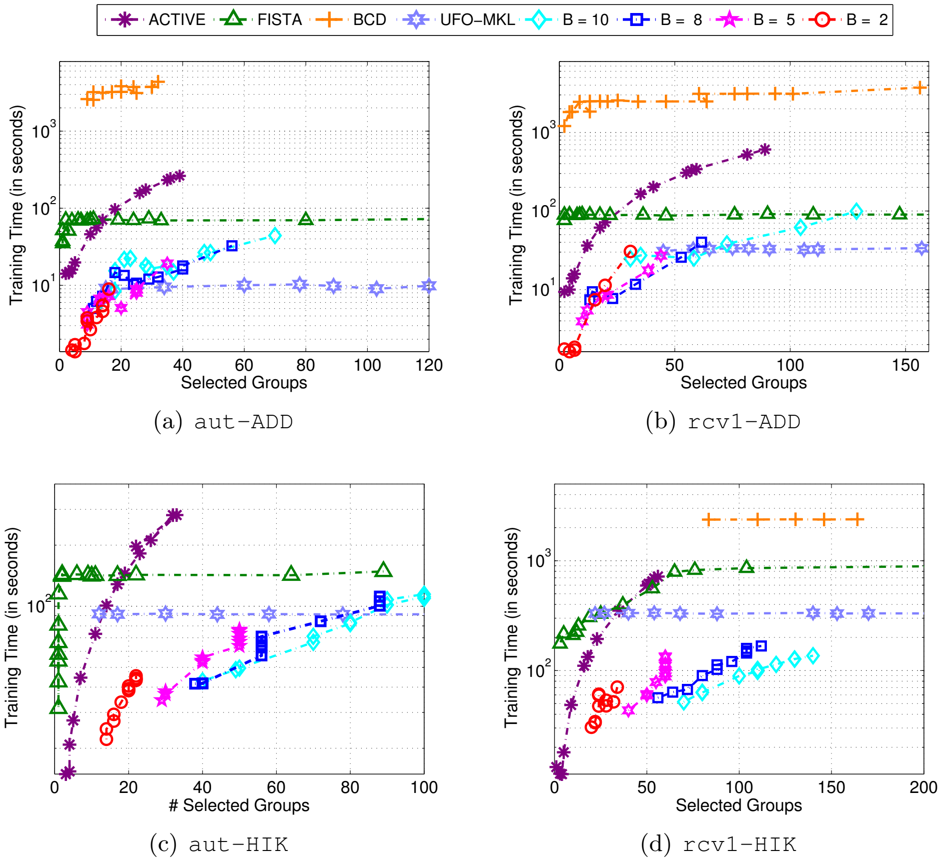
(d) rcv1-HIK

Figure 16: Testing accuracy on group feature selection tasks. The groups are generated by HIK or additive feature mappings. The results of BCD on aut-HIK is not reported due to the heavy computational cost.

9. Conclusions

In this paper, an adaptive feature scaling (AFS) scheme has been proposed to conduct feature selection tasks. Specifically, to explicitly control the number features to be selected, we first introduce a vector $\mathbf{d} \in [0,1]^m$ to scale the input features, and then impose an ℓ_1-norm constraint $||\mathbf{d}||_1 \leq B$, where B represents the least number of features to be selected. Although the resultant problem is non-convex, we can transform it into an equivalent convex SIP problem. After that, a feature generating machine (FGM) is proposed to solve the SIP problem, which essentially includes B informative features per iteration and solves a sequence of much reduced MKL subproblems. The global convergence of FGM has been verified. Moreover, to make FGM scalable to big data, we propose to solve the primal form of the MKL subproblem through a modified APG method. Some efficient cache techniques are also developed to further improve the training efficiency. Finally, FGM has been extended to perform group feature selection and multiple kernel learning w.r.t. additive kernels.

FGM has two major advantages over the ℓ_1-norm methods and other existing feature selection methods. Firstly, with a separate control of the model complexity and sparsity,



(a) aut-ADD

(b) rcv1-ADD

(c) aut-HIK

(d) rcv1-HIK

Figure 17: Training time on group feature selections.

FGM can effectively handle the feature selection bias issue. Secondly, since only a small subset of features or kernels are involved in the subproblem optimization, FGM is particularly suitable for the ultrahigh dimensional feature selection task on big data, for which most of the existing methods are infeasible. It is worth mentioning that, unlike most of the existing methods, FGM avoids the storing of all base kernels or the full explicit feature mappings. Therefore, it can vastly reduce the unbearable memory demands of MKL with many base kernels or the nonlinear feature selection with ultrahigh-dimensional feature mappings.

Comprehensive experiments have been conducted to study the performance of the proposed methods on both linear feature selection and group feature selection tasks. Extensive experiments on synthetic data sets and real-world data sets have demonstrated the superior performance of FGM over the baseline methods in terms of both training efficiency and testing accuracy.

In this paper, the proposed methods have tackled big data problems with million training examples ($O(10^7)$) and 100 trillion features ($O(10^{14})$). Recall that the subproblems of FGM can be possibly addressed through SGD methods, we will explore SGD methods in the future to further improve the training efficiency over bigger data with ultra-large sample size.

Acknowledgments

We would like to acknowledge the valuable comments and useful suggestions by the Action Editor and the four anonymous reviewers. We would like to express our gratitude to Dr. Xinxing Xu and Dr. Shijie Xiao for the proofreading and comments. This research was partially supported by the Nanyang Technological University, the ASTAR Thematic Strategic Research Programme (TSRP) Grant No. 1121720013, and the Australian Research Council Future Fellowship FT130100746.

Appendix A. Proof of Theorem 3

Proof The proof parallels the results of Bach et al. (2004), and is based on the conic duality theory. Let $\Omega(\boldsymbol{\omega}) = \frac{1}{2}\left(\|\boldsymbol{\omega}_h\|\right)^2$ and define the cone $\mathcal{Q}_B = \{(\mathbf{u}, v) \in \mathbb{R}^{B+1}, \|\mathbf{u}\|_2 \leq v\}$. Furthermore, let $z_h = \|\boldsymbol{\omega}_h\|$, we have $\Omega(\boldsymbol{\omega}) = \frac{1}{2}\left(\sum_{h=1}^t \|\boldsymbol{\omega}_t\|\right)^2 = \frac{1}{2}z^2$ with $z = \sum_{h=1}^t z_h$. Apparently, we have $z_h \geq 0$ and $z \geq 0$. Finally, problem (22) can be transformed to the following problem:

$$\min_{z,\boldsymbol{\omega}} \quad \frac{1}{2}z^2 + P(\boldsymbol{\omega}, b), \quad \text{s.t.} \sum_{h=1}^t z_h \leq z, \quad (\boldsymbol{\omega}_t, z_h) \in \mathcal{Q}_B, \tag{33}$$

where $\boldsymbol{\omega} = [\boldsymbol{\omega}'_1, ..., \boldsymbol{\omega}'_t]'$. The Lagrangian function of (33) regarding the squared hinge loss can be written as:

$$\begin{aligned}&\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi})\\ =\ & \frac{1}{2}z^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 - \sum_{i=1}^n \alpha_i \left(y_i(\sum \boldsymbol{\omega}'_h \mathbf{x}_{ih} - b) - 1 + \xi_i\right) + \gamma(\sum_{h=1}^t z_h - z) - \sum_{h=1}^t (\boldsymbol{\zeta}'_h \boldsymbol{\omega}_h + \varpi_h z_h),\end{aligned}$$

where $\boldsymbol{\alpha}$, γ, $\boldsymbol{\zeta}_t$ and ϖ_t are the Lagrangian dual variables to the corresponding constraints. The KKT condition can be expressed as

$$\begin{array}{ll}
\nabla_z \mathcal{L} = z - \gamma = 0 & \Rightarrow z = \gamma;\\
\nabla_{z_h} \mathcal{L} = \gamma - \varpi_h = 0 & \Rightarrow \varpi_h = \gamma;\\
\nabla_{\boldsymbol{\omega}_h} \mathcal{L} = -\sum_{i=1}^n \alpha_i y_i \mathbf{x}_{ih} - \boldsymbol{\zeta}_h = 0 & \Rightarrow \boldsymbol{\zeta}_h = -\sum_{i=1}^n \alpha_i y_i \mathbf{x}_{ih};\\
\nabla_{\xi_i} \mathcal{L} = C\xi_i - \alpha_i = 0 & \Rightarrow \xi_i = \frac{\alpha_i}{C};\\
\|\boldsymbol{\zeta}_h\| \leq \varpi_h & \Rightarrow \|\boldsymbol{\zeta}_h\| \leq \gamma;\\
\nabla_b \mathcal{L} = 0 & \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0.
\end{array}$$

By substituting the above equations into the Lagrangian function, we have

$$\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) = -\frac{1}{2}\gamma^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + 1'\boldsymbol{\alpha}.$$

Hence the dual problem of the $\ell_{2,1}^2$-regularized problem regarding squared hinge loss can be written as:

$$
\begin{aligned}
\max_{\gamma,\boldsymbol{\alpha}} \quad & -\frac{1}{2}\gamma^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + 1'\boldsymbol{\alpha} \\
\text{s.t} \quad & \Big\| \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih} \Big\| \leq \gamma, \quad h = 1, \cdots, t, \\
& \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0, \quad i = 1, \cdots, n.
\end{aligned}
$$

Let $\theta = \frac{1}{2}\gamma^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}$, $\boldsymbol{\omega}_h = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih}$ and $f(\boldsymbol{\alpha}, \mathbf{d}_h) = \frac{1}{2}\|\boldsymbol{\omega}_h\|^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}$, we have

$$
\begin{aligned}
\max_{\theta,\boldsymbol{\alpha}} \quad & -\theta, \\
\text{s.t} \quad & f(\boldsymbol{\alpha}, \mathbf{d}_h) \leq \theta, \quad h = 1, \cdots, t, \\
& \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0, \quad i = 1, \cdots, n.
\end{aligned}
$$

which indeed is in the form of problem (16) by letting $\mathcal{A}$ be the domain of $\boldsymbol{\alpha}$. This completes the proof and brings the connection between the primal and dual formulation.

By defining $0\log(0) = 0$, with the similar derivation above, we can obtain the dual form of (33) regarding the logistic loss. Specifically, the Lagrangian function of (33) w.r.t. the logistic loss is:

$$
\begin{aligned}
&\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) \\
&\quad = \frac{1}{2}z^2 + C\sum_{i=1}^{n} \log(1 + \exp(\xi_i)) - \sum_{i=1}^{n} \alpha_i \left(y_i(\sum \boldsymbol{\omega}'_h \mathbf{x}_{ih} - b) + \xi_i \right) + \gamma(\sum_{h=1}^{t} z_h - z) - \sum_{h=1}^{t} (\boldsymbol{\zeta}'_h \boldsymbol{\omega}_h + \varpi_h z_h),
\end{aligned}
$$

where $\boldsymbol{\alpha}$, γ, $\boldsymbol{\zeta}_t$ and ϖ_t are the Lagrangian dual variables to the corresponding constraints. The KKT condition can be expressed as

$$
\begin{array}{ll}
\nabla_z \mathcal{L} = z - \gamma = 0 & \Rightarrow z = \gamma; \\
\nabla_{z_h} \mathcal{L} = \gamma - \varpi_h = 0 & \Rightarrow \varpi_h = \gamma; \\
\nabla_{\boldsymbol{\omega}_h} \mathcal{L} = -\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih} - \boldsymbol{\zeta}_h = 0 & \Rightarrow \boldsymbol{\zeta}_h = -\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih}; \\
\nabla_{\xi_i} \mathcal{L} = \frac{C\exp(\xi_i)}{1+\exp(\xi_i)} - \alpha_i = 0 & \Rightarrow \exp(\xi_i) = \frac{\alpha_i}{C - \alpha_i}; \\
\|\boldsymbol{\zeta}_h\| \leq \varpi_h & \Rightarrow \|\boldsymbol{\zeta}_h\| \leq \gamma; \\
\nabla_b \mathcal{L} = 0 & \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0.
\end{array}
$$

By substituting all the above results into the Lagrangian function, we have

$$
\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) = -\frac{1}{2}\gamma^2 - \sum_{i=1}^{n} (C - \alpha_i)\log(C - \alpha_i) - \sum_{i=1}^{n} \alpha_i \log(\alpha_i).
$$

The dual form of the $\ell_{2,1}^2$-regularized problem regarding logistic loss can be written as:

$$\begin{aligned}\max_{\gamma,\boldsymbol{\alpha}} \quad & -\frac{1}{2}\gamma^2 - \sum_{i=1}^n (C-\alpha_i)\log(C-\alpha_i) - \sum_{i=1}^n \alpha_i \log(\alpha_i)\\ \text{s.t.} \quad & \Big\| \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{ih} \Big\| \leq \gamma, \quad h = 1, \cdots, t,\\ & \sum_{i=1}^n \alpha_i y_i = 0, \alpha_i \geq 0, \quad i = 1, \cdots, n.\end{aligned}$$

Let $\theta = \frac{1}{2}\gamma^2 + \sum_{i=1}^n (C-\alpha_i)\log(C-\alpha_i) + \sum_{i=1}^n \alpha_i \log(\alpha_i)$, $\boldsymbol{\omega}_h = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{ih}$, $f(\boldsymbol{\alpha}, \mathbf{d}_h) = \frac{1}{2}\|\boldsymbol{\omega}_h\|^2 + \sum_{i=1}^n (C-\alpha_i)\log(C-\alpha_i) + \sum_{i=1}^n \alpha_i \log(\alpha_i)$, then we have

$$\begin{aligned}\max_{\theta,\boldsymbol{\alpha}} \quad & -\theta,\\ \text{s.t.} \quad & f(\boldsymbol{\alpha}, \mathbf{d}_h) \leq \theta, \quad h = 1, \cdots, t,\\ & \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \cdots, n.\end{aligned}$$

Finally, according to the KKT condition, we can easily recover the dual variable $\boldsymbol{\alpha}$ by $\alpha_i = \frac{C\exp(\xi_i)}{1+\exp(\xi_i)}$. This completes the proof. ■

Appendix B. Proof of Theorem 4

The proof parallels the results of Beck and Teboulle (2009), and includes several lemmas. First of all, we define a one variable function $Q_{\tau_b}(\mathbf{v}, b, v_b)$ w.r.t. b as

$$Q_{\tau_b}(\mathbf{v}, b, v_b) \quad = \quad P(\mathbf{v}, v_b) + \langle \nabla_b P(\mathbf{v}, v_b), b - v_b\rangle + \frac{\tau_b}{2}\|b - v_b\|^2, \tag{34}$$

where we abuse the operators $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ for convenience.

Lemma 4 *$S_\tau(\mathbf{u}, \mathbf{v}) = \arg\min_{\boldsymbol{\omega}} Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ is the minimizer of problem (23) at point $\mathbf{v}$, if and only if there exists $g(S_\tau(\mathbf{u}, \mathbf{v})) \in \partial\Omega(S_\tau(\mathbf{u}, \mathbf{v}))$, the subgradient of $\Omega(\boldsymbol{\omega})$ at $S_\tau(\mathbf{u}, \mathbf{v})$, such that*

$$g(S_\tau(\mathbf{u}, \mathbf{v})) + \tau(S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v}) + \nabla P(\mathbf{v}) = \mathbf{0}.$$

Proof The proof can be completed by the optimality condition of $Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ w.r.t. $\boldsymbol{\omega}$. ■

Lemma 5 *Let $S_\tau(\mathbf{u}, \mathbf{v}) = \arg\min_{\boldsymbol{\omega}} Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ be the minimizer of problem (23) at point $\mathbf{v}$, and $S_{\tau_b}(b) = \arg\min_b Q_{\tau_b}(\mathbf{v}, b, v_b)$ be the minimizer of problem (34) at point v_b. Due to the line search in Algorithm 4, we have*

$$\begin{aligned}&F(S_\tau(\mathbf{u}, \mathbf{v}), v_b) \leq Q_\tau(S_\tau(\mathbf{u}, \mathbf{v}), \mathbf{v}, v_b).\\ &P(\mathbf{v}, S_{\tau_b}(v_b)) \leq Q_{\tau_b}(\mathbf{v}, S_{\tau_b}(v_b), v_b).\end{aligned}$$

and

$$F(S_\tau(\mathbf{u},\mathbf{v}), S_{\tau_b}(b)) \leq Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b) - v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2. \quad (35)$$

Furthermore, for any $(\boldsymbol{\omega}', b)'$ *we have*

$$\begin{aligned} F(\boldsymbol{\omega}, b) - F(S_\tau(\mathbf{u},\mathbf{v}), S_{\tau_b}(b)) \quad \geq \quad & \tau_b\langle S_{\tau_b}(b) - v_b, v_b - b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b) - v_b||^2 \\ & +\tau\langle S_\tau(\mathbf{u},\mathbf{v}) - \mathbf{v}, \mathbf{v} - \boldsymbol{\omega}\rangle + \frac{\tau}{2}\|S_\tau(\mathbf{u},\mathbf{v}) - \mathbf{v}||^2. \quad (36)\end{aligned}$$

Proof We only prove the inequality (35) and (36). First of all, recall that in Algorithm 4, we update $\boldsymbol{\omega}$ and b separately. It follows that

$$\begin{aligned} & F(S_\tau(\mathbf{u},\mathbf{v}), S_{\tau_b}(b)) \\ = \quad & \Omega(S_\tau(\mathbf{u},\mathbf{v})) + P(S_\tau(\mathbf{u},\mathbf{v}), S_{\tau_b}(v_b)) \\ \leq \quad & \Omega(S_\tau(\mathbf{u},\mathbf{v})) + Q_{\tau_b}(S_\tau(\mathbf{u},\mathbf{v}), S_{\tau_b}(b), v_b) \\ = \quad & \Omega(S_\tau(\mathbf{u},\mathbf{v})) + P(S_\tau(\mathbf{u},\mathbf{v}), v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b) - v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2 \\ = \quad & F(S_\tau(\mathbf{u},\mathbf{v}), v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b) - v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2 \\ \leq \quad & Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b) - v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2. \end{aligned}$$

This proves the inequality in (35).

Now we prove the inequality (36). First of all, since both $P(\boldsymbol{\omega}, b)$ and $\Omega(\boldsymbol{\omega})$ are convex functions, we have

$$\begin{aligned} & P(\boldsymbol{\omega}, b) \geq P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b - v_b\rangle, \\ & \Omega(\boldsymbol{\omega}) \geq \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega} - S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \end{aligned}$$

where $g(S_\tau(\mathbf{u},\mathbf{v}))$ be the subgradient of $\Omega(\boldsymbol{\omega})$ at point $S_\tau(\mathbf{u},\mathbf{v})$. Summing up the above inequalities, we obtain

$$\begin{aligned} & F(\boldsymbol{\omega}, b) \\ \geq \quad & P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b - v_b\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega} - S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \end{aligned}$$

In addition, we have

$$\begin{aligned} & F(\boldsymbol{\omega}, b) - \Big(Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b) - v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2\Big) \\ = \quad & P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b - v_b\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega} - S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \\ & - \Big(Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b) - v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2\Big) \\ = \quad & P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b - v_b\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega} - S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \\ & -\Big(P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), S_\tau(\mathbf{u},\mathbf{v}) - \mathbf{v}\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \frac{\tau}{2}\|S_\tau(\mathbf{u},\mathbf{v}) - \mathbf{v}||^2 + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b) - v_b\rangle \\ & \qquad + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2\Big) \\ = \quad & \langle \nabla P(\mathbf{v}) + g(S_\tau(\mathbf{g},\mathbf{v})), \boldsymbol{\omega} - S_\tau(\mathbf{u},\mathbf{v})\rangle - \frac{\tau}{2}\|S_\tau(\mathbf{u},\mathbf{v}) - \mathbf{v}||^2 \\ & + \langle \nabla_b P(\mathbf{v},v_b), b - S_{\tau_b}(b)\rangle - \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b||^2. \end{aligned}$$

With the relation $S_{\tau_b}(b) = b - \frac{\nabla_b P(\mathbf{v}, v_b)}{\tau_b}$ and Lemma 4, we obtain

$$\begin{aligned}
&F(\boldsymbol{\omega}, b) - F(S_\tau(\mathbf{u}, \mathbf{v}), S_{\tau_b}(b)) \\
&\geq \quad F(\boldsymbol{\omega}, b) - \Big(Q_\tau(S_\tau(\mathbf{u}, \mathbf{v}), \mathbf{v}, v_b) + \langle \nabla_b P(\mathbf{v}, v_b), S_{\tau_b}(b) - v_b \rangle + \frac{\tau_b}{2} \| S_{\tau_b}(b) - v_b \|^2 \Big). \\
&\geq \quad \langle \nabla P(\mathbf{v}) + g(S_\tau(\mathbf{g}, \mathbf{v})), \boldsymbol{\omega} - S_\tau(\mathbf{u}, \mathbf{v}) \rangle - \frac{\tau}{2} \| S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v} \|^2 \\
&\qquad + \langle \nabla_b P(\mathbf{v}, v_b), b - S_{\tau_b}(b) \rangle - \frac{\tau_b}{2} \| S_{\tau_b}(b) - v_b \|^2. \\
&= \quad \tau \langle \mathbf{v} - S_\tau(\mathbf{u}, \mathbf{v}), \boldsymbol{\omega} - S_\tau(\mathbf{u}, \mathbf{v}) \rangle - \frac{\tau}{2} \| S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v} \|^2 \\
&\qquad + \tau_b \langle v_b - S_{\tau_b}(b), b - S_{\tau_b}(b) \rangle - \frac{\tau_b}{2} \| S_{\tau_b}(b) - v_b \|^2. \\
&= \quad \tau \langle S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v}, \mathbf{v} - \boldsymbol{\omega} \rangle + \frac{\tau}{2} \| S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v} \|^2 \\
&\qquad + \tau_b \langle S_{\tau_b}(b) - v_b, v_b - b \rangle + \frac{\tau_b}{2} \| S_{\tau_b}(b) - v_b \|^2.
\end{aligned}$$

This completes the proof. ■

Lemma 6 *Let $L_{bt} = \sigma L_t$, where $\sigma > 0$. Furthermore, let us define*

$$\begin{aligned}
\mu^k &= F(\boldsymbol{\omega}^k, b^k) - F(\boldsymbol{\omega}^*, b^*), \\
\boldsymbol{\nu}^k &= \rho^k \boldsymbol{\omega}^k - (\rho^k - 1)\boldsymbol{\omega}^{k-1} - \boldsymbol{\omega}^*, \\
\upsilon^k &= \rho^k b^k - (\rho^k - 1) b^{k-1} - b^*,
\end{aligned}$$

and then the following relation holds:

$$\frac{2(\rho^k)^2 \mu^k}{L^k} - \frac{(\rho^{k+1})^2 \mu^{k+1}}{L^{k+1}} \geq \left(||\boldsymbol{\nu}^{k+1}||^2 - ||\boldsymbol{\nu}^k||^2 \right) + \sigma \left((\upsilon^{k+1})^2 - (\upsilon^k)^2 \right).$$

Proof Note that we have $\boldsymbol{\omega}^{k+1} = S_\tau(\mathbf{u}, \mathbf{v}^{k+1})$ and $b^{k+1} = S_{\tau_b}(v_b^{k+1})$. By applying Lemma 5, let $\boldsymbol{\omega} = \boldsymbol{\omega}^k$, $\mathbf{v} = \mathbf{v}^{k+1}, \tau = L^{k+1}$, $b = b^k$, $v_b = v_b^{k+1}, \tau_b = L_b^{k+1}$, we have

$$\begin{aligned}
2(\mu^k - \mu^{k+1}) \quad \geq \quad & L^{k+1} \left(||\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}||^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \mathbf{v}^{k+1} - \boldsymbol{\omega}^k \rangle \right) \\
& + L_b^{k+1} \left(||b^{k+1} - v_b^{k+1}||^2 + 2\langle b^{k+1} - v_b^{k+1}, v_b^{k+1} - b^k \rangle \right).
\end{aligned}$$

Multiplying both sides by $(\rho^{k+1} - 1)$, we obtain

$$\begin{aligned}
2(\rho^{k+1} - 1)(\mu^k - \mu^{k+1}) \quad \geq \quad & L^{k+1}(\rho^{k+1} - 1) \left(||\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}||^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \mathbf{v}^{k+1} - \boldsymbol{\omega}^k \rangle \right) \\
& + L_b^{k+1}(\rho^{k+1} - 1) \left(||b^{k+1} - v_b^{k+1}||^2 + 2\langle b^{k+1} - v_b^{k+1}, v_b^{k+1} - b^k \rangle \right).
\end{aligned}$$

Also, let $\boldsymbol{\omega} = \boldsymbol{\omega}^*$, $\mathbf{v} = \mathbf{v}^{k+1}, \tau = L^{k+1}$, $b = b^k$, $v_b = v_b^{k+1}$, and $\tau_b = L_b^{k+1}$, we have

$$\begin{aligned}
-2\mu^{k+1} \quad \geq \quad & L^{k+1} \left(||\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}||^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \mathbf{v}^{k+1} - \boldsymbol{\omega}^* \rangle \right) \\
& + L_b^{k+1} \left(||b^{k+1} - v_b^{k+1}||^2 + 2\langle b^{k+1} - v_b^{k+1}, v_b^{k+1} - b^* \rangle \right).
\end{aligned}$$

Summing up the above two inequalities, we get

$$\begin{aligned}
& 2\left((\rho^{k+1} - 1)\mu^k - \rho^{k+1}\mu^{k+1} \right) \\
& \quad \geq \quad L^{k+1} \left(\rho^{k+1} ||\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}||^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \rho^{k+1}\mathbf{v}^{k+1} - (\rho^{k+1} - 1)\boldsymbol{\omega}^k - \boldsymbol{\omega}^* \rangle \right) \\
& \qquad + L_b^{k+1} \left(\rho^{k+1} ||b^{k+1} - v_b^{k+1}||^2 + 2\langle b^{k+1} - v_b^{k+1}, \rho^{k+1} v_b^{k+1} - (\rho^{k+1} - 1) b^k - b^* \rangle \right).
\end{aligned}$$

Multiplying both sides by ρ^{k+1}, we obtain

$$\begin{aligned}
&2\left(\rho^{k+1}(\rho^{k+1}-1)\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\\
&\quad\geq\ \ L^{k+1}\left((\rho^{k+1})^2||\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1}||^2+2\rho^{k+1}\langle\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1},\rho^{k+1}\mathbf{v}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*\rangle\right)\\
&\qquad\ +L_b^{k+1}\left((\rho^{k+1})^2||b^{k+1}-v_b^{k+1}||^2+2\rho^{k+1}\langle b^{k+1}-v_b^{k+1},\rho^{k+1}v_b^{k+1}-(\rho^{k+1}-1)b^k-b^*\rangle\right).
\end{aligned}$$

Since $(\rho^k)^2=(\rho^{k+1})^2-\rho^{k+1}$, it follows that

$$\begin{aligned}
&2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\\
&\quad\geq\ \ L^{k+1}\left((\rho^{k+1})^2||\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1}||^2+2\rho^{k+1}\langle\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1},\rho^{k+1}\mathbf{v}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*\rangle\right)\\
&\qquad\ +L_b^{k+1}\left((\rho^{k+1})^2||b^{k+1}-v_b^{k+1}||^2+2\rho^{k+1}\langle b^{k+1}-v_b^{k+1},\rho^{k+1}v_b^{k+1}-(\rho^{k+1}-1)b^k-b^*\rangle\right).
\end{aligned}$$

By applying the equality $||\mathbf{u}-\mathbf{v}||^2+2\langle\mathbf{u}-\mathbf{v},\mathbf{v}-\mathbf{w}\rangle=||\mathbf{u}-\mathbf{w}||^2-||\mathbf{v}-\mathbf{w}||^2$, we have

$$\begin{aligned}
&2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\\
&\quad\geq\ \ L^{k+1}\left(||\rho^{k+1}\boldsymbol{\omega}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*||^2-||\rho^{k+1}\mathbf{v}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*||^2\right)\\
&\qquad\ +L_b^{k+1}\left(||\rho^{k+1}b^{k+1}-(\rho^{k+1}-1)b^k-b^*||^2-||\rho^{k+1}v_b^{k+1}-(\rho^{k+1}-1)b^k-b^*||^2\right).
\end{aligned}$$

With $\rho^{k+1}\mathbf{v}^{k+1}=\rho^{k+1}\boldsymbol{\omega}^k+(\rho^k-1)(\boldsymbol{\omega}^k-\boldsymbol{\omega}^{k-1})$, $\rho^{k+1}v_b^{k+1}=\rho^{k+1}b^k+(\rho^k-1)(b^k-b^{k-1})$ and the definition of $\boldsymbol{\nu}^k$, it follows that

$$2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\geq L^{k+1}\left(||\boldsymbol{\nu}^{k+1}||^2-||\boldsymbol{\nu}^k||^2\right)+L_b^{k+1}\left((\upsilon^{k+1})^2-(\upsilon^k)^2\right).$$

Assuming that there exists a $\sigma>0$ such that $L_b^{k+1}=\sigma L^{k+1}$, we get

$$\frac{2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)}{L^{k+1}}\geq\left(||\boldsymbol{\nu}^{k+1}||^2-||\boldsymbol{\nu}^k||^2\right)+\sigma\left((\upsilon^{k+1})^2-(\upsilon^k)^2\right).$$

Since $L^{k+1}\geq L^k$ and $L_b^{k+1}\geq L_b^k$, we have

$$\frac{2(\rho^k)^2\mu^k}{L^k}-\frac{(\rho^{k+1})^2\mu^{k+1}}{L^{k+1}}\geq\left(||\boldsymbol{\nu}^{k+1}||^2-||\boldsymbol{\nu}^k||^2\right)+\sigma\left((\upsilon^{k+1})^2-(\upsilon^k)^2\right).$$

This completes the proof. ■

Finally, with Lemma 6, following the proof of Theorem 4.4 in (Beck and Teboulle, 2009), we have

$$F(\boldsymbol{\omega}^k,b^k)-F(\boldsymbol{\omega}^*,b^*)\leq\frac{2L^k||\boldsymbol{\omega}^0-\boldsymbol{\omega}^*||^2}{(k+1)^2}+\frac{2\sigma L^k(b^0-b^*)^2}{(k+1)^2}\leq\frac{2L_t||\boldsymbol{\omega}^0-\boldsymbol{\omega}^*||^2}{\eta(k+1)^2}+\frac{2L_{bt}(b^0-b^*)^2}{\eta(k+1)^2}.$$

This completes the proof.

Appendix C: Linear Convergence of Algorithm 4 for the Logistic Loss

In Algorithm 4, by fixing $\varrho^k=1$, it is reduced to the proximal gradient method (Nesterov, 2007), and it attains a linear convergence rate for the logistic loss, if $\mathbf{X}$ satisfies the following *Restricted Eigenvalue Condition* (Zhang, 2010b):

Definition 2 *(Zhang, 2010b) Given an integer $\kappa > 0$, a design matrix $\mathbf{X}$ is said to satisfy the Restricted Eigenvalue Condition at sparsity level κ, if there exists positive constants $\gamma_-(\mathbf{X}, \kappa)$ and $\gamma_+(\mathbf{X}, \kappa)$ such that*

$$\gamma_-(\mathbf{X}, \kappa) = \inf\left\{\frac{\boldsymbol{\omega}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\omega}}{\boldsymbol{\omega}^\top\boldsymbol{\omega}}, \boldsymbol{\omega} \neq \mathbf{0}, ||\boldsymbol{\omega}||_0 \leq \kappa\right\},$$
$$\gamma_+(\mathbf{X}, \kappa) = \sup\left\{\frac{\boldsymbol{\omega}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\omega}}{\boldsymbol{\omega}^\top\boldsymbol{\omega}}, \boldsymbol{\omega} \neq \mathbf{0}, ||\boldsymbol{\omega}||_0 \leq \kappa\right\}.$$

Remark 7 *For the logistic loss, if $\gamma_-(\mathbf{X}, tB) \geq \tau > 0$, Algorithm 4 with $\varrho^k = 1$ attains a linear convergence rate.*

Proof Let $\xi_i = -y_i(\sum_{h=1}^t \boldsymbol{\omega}_h' \mathbf{x}_{ih} - b)$, the Hessian matrix for the logistic loss can be calculated by (Yuan et al., 2011):

$$\nabla^2 P(\boldsymbol{\omega}) = C\mathbf{X}'\boldsymbol{\Delta}\mathbf{X},$$

where $\boldsymbol{\Delta}$ is a diagonal matrix with diagonal element $\boldsymbol{\Delta}_{i,i} = \frac{1}{1+\exp(\xi_i)}(1 - \frac{1}{1+\exp(\xi_i)}) > 0$. Apparently, $\nabla^2 P(\boldsymbol{\omega}, b)$ is upper bounded on a compact set due to the existence of $\gamma_+(\mathbf{X}, \kappa)$. Let $\sqrt{\boldsymbol{\Delta}}$ be the square root of $\boldsymbol{\Delta}$. Then if $\gamma_-(\mathbf{X}, tB) \geq \tau > 0$, we have $\gamma_-(\sqrt{\boldsymbol{\Delta}}\mathbf{X}, tB) > 0$ due to $\boldsymbol{\Delta}_{i,i} > 0$. In other words, the logistic loss is strongly convex if $\gamma_-(\mathbf{X}, tB) > 0$. Accordingly, the linear convergence rate can be achieved (Nesterov, 2007). ■

Appendix D: Proof of Proposition 3

Proof Proof of argument (I): We prove it by contradiction. Firstly, suppose $\mathbf{d}^*$ is a minimizer and there exists an $l \in \{1 \dots m\}$, such that $w_l = 0$ but $d_l^* > 0$. Let $0 < \epsilon < d_l^*$, and choose one $j \in \{1 \dots m\}$ where $j \neq l$, such that $|w_j| > 0$. Define new solution $\widehat{\mathbf{d}}$ in the following way:

$$\widehat{d}_j = d_j^* + d_l^* - \epsilon, \quad \widehat{d}_l = \epsilon, \quad \text{and,}$$
$$\widehat{d}_k = d_k^*, \quad \forall k \in \{1 \dots m\} \backslash \{j, l\}.$$

Then it is easy to check that

$$\sum_{j=1}^m \widehat{d}_j = \sum_{j=1}^m d_j^* \leq B.$$

In other words, $\widehat{\mathbf{d}}$ is also a feasible point. However, since $\widehat{d}_j = d_j^* + d_l^* - \epsilon \geq d_j^*$, it follows that

$$\frac{w_j^2}{\widehat{d}_j} < \frac{w_j^2}{d_j^*}.$$

Therefore, we have

$$\sum_{j=1}^m \frac{w_j^2}{\widehat{d}_j} < \sum_{j=1}^m \frac{w_j^2}{d_j^*},$$

which contradict the assumption that $\mathbf{d}^*$ is the minimizer.

On the other hand, if $|w_j| > 0$ and $d_j^* = 0$, by the definition, $\frac{x_j^2}{0} = \infty$. As we expect to get the finite minimum, so if $|w_j| > 0$, we have $d_j^* > 0$.

(II): First of all, the argument holds trivially when $\|\mathbf{w}\|_0 = \kappa \leq B$.

If $\|\mathbf{w}\|_0 = \kappa > B$, without loss of generality, we assume $|w_j| > 0$ for the first κ elements. From the argument (I), we have $1 \geq d_j > 0$ for $j \in \{1 \dots \kappa\}$ and $\sum_{j=1}^{\kappa} d_j \leq B$. Note that $\sum_{j=1}^{\kappa} \frac{w_j^2}{d_j}$ is convex regarding $\mathbf{d}$. The minimization problem can be written as:

$$\min_{\mathbf{d}} \sum_{j=1}^{\kappa} \frac{w_j^2}{d_j}, \qquad \text{s.t.} \quad \sum_{j=1}^{\kappa} d_j \leq B, \;\; d_j > 0, \;\; 1 - d_j \geq 0. \tag{37}$$

The KKT condition of this problem can be written as:

$$\begin{aligned}
&-w_j^2/d_j^2 + \gamma - \zeta_j + \nu_j = 0,\\
&\zeta_j d_j = 0,\\
&\nu_j(1 - d_j) = 0, && (38)\\
&\gamma(B - \sum_{j=1}^{\kappa} d_j) = 0, && (39)\\
&\gamma \geq 0, \zeta_j \geq 0, \nu_j \geq 0, \forall\; j \in \{1 \dots \kappa\},
\end{aligned}$$

where γ, ζ_j and ν_j are the dual variables for the constraints $\sum_{j=1}^{\kappa} d_j \leq B$, $d_j > 0$ and $1 - d_j \geq 0$ respectively. For those $d_j > 0$, we have $\zeta_j = 0$ for $\forall j \in \{1 \dots \kappa\}$ due to the KKT condition. Accordingly, by the first equality in KKT condition, we must have

$$d_j = |w_j|/\sqrt{\gamma + \nu_j}, \qquad \forall j \in \{1 \dots \kappa\}.$$

Moreover, since $\sum_{j=1}^{\kappa} d_j \leq B < \kappa$, there must exist some $d_j < 1$ with $\nu_j = 0$ (otherwise $\sum_{j=1}^{\kappa} d_j$ will be greater than B). Here $\nu_j = 0$ because of the condition (38). This observation implies that $\gamma \neq 0$ since each d_j is bounded. Since $d_j \leq 1$, the condition $\sqrt{\gamma + \nu_j} \geq \max\{|w_j|\}$ must hold for $\forall j \in \{1 \dots \kappa\}$. Furthermore, by the complementary condition (39), we must have

$$\sum_{j=1}^{\kappa} d_j = B.$$

By substituting $d_j = |w_j|/\sqrt{\gamma + \nu_j}$ back to the objective function of (37), it becomes

$$\sum_{j=1}^{\kappa} |w_j|\sqrt{\gamma + \nu_j}.$$

To get the minimum of the above function, we are required to set the nonnegative ν_j as small as possible.

Now we complete the proof with the assumption $\|\mathbf{w}\|_1 / \max\{|w_j|\} \geq B$. When setting $\nu_j = 0$, we get $d_j = \frac{|w_j|}{\sqrt{\gamma}}$ and $\sum_{j=1}^{\kappa} \frac{|w_j|}{\sqrt{\gamma}} = B$. It is easy to check that $\sqrt{\gamma} = \|\mathbf{w}\|_1/B \geq \max\{|w_j|\}$ and $d_j = B|w_j|/\|\mathbf{w}\|_1 \leq 1$, which satisfy the KKT condition. Therefore, the above $\mathbf{d}$ is an optimal solution. This completes the proof of the argument (II).

(III): With the results of (II), if $\kappa \leq B$, we have $\sum_{j=1}^{m} \frac{w_j^2}{d_j} = \sum_{j=1}^{\kappa} w_j^2$. Accordingly, we have $\|\mathbf{w}\|_B = \|\mathbf{w}\|_2$. And if $\kappa > B$ and $\|\mathbf{w}\|_1 / \max\{w_j\} \geq B$, we have

$$\sum \frac{w_j^2}{d_j} = \sum \frac{|w_j|}{d_j} |w_j| = \frac{\|\mathbf{w}\|_1}{B} \sum |w_j| = \frac{(\|\mathbf{w}\|_1)^2}{B}.$$

Hence we have $\|\mathbf{w}\|_B = \sqrt{\sum \frac{w_j^2}{d_j}} = \frac{\|\mathbf{w}\|_1}{\sqrt{B}}$. This completes the proof. ■

References

F. R. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. In *Optimization for Machine Learning*. S. Sra, S. Nowozin, S. J. Wright., 2011.

F. R. Bach. High-dimensional non-linear variable selection through hierarchical kernel learning. Technical report, 2009.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

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

B . Blum, M. I. Jordan, D. E. Kim, R. Das, P. Bradley, and D. Baker. Feature selection methods for improving protein structure prediction with rosetta. In *NIPS*, 2007.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, 1998.

A. B. Chan, N. Vasconcelos, and G. R. G. Lanckriet. Direct convex relaxations of sparse SVM. In *ICML*, 2007.

Y. W. Chang, C. J. Hsieh, K. W. Chang, M. Ringgaard, and C. J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *JMLR*, 11:1471–1490, 2010.

O. Chapelle and S. S. Keerthi. Multi-class feature selection with support vector machines. In *Proceedings of the American Statistical Association*, 2008.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Mach. Learn.*, 46(1):131–159, 2002.

J. Chen and J. Ye. Training SVM with indefinite kernels. In *ICML*, 2008.

A. Dasgupta, P. Drineas, and B. Harb. Feature selection methods for text classification. In *KDD*, 2007.

J. Deng, A. C. Berg, and F. Li. Hierarchical semantic indexing for large scale image retrieval. In *CVPR*, pages 785–792. IEEE, 2011.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed.)*. Hoboken, NJ: Wiley-Interscience, 2000.

R. Fan, P. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. *JMLR*, 6:1889–1918, 2005.

M Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE J. Sel. Top. Sign. Proces.: Special Issue on Convex Optimization Methods for Signal Processing*, 1(4):586–597, 2007.

G. M. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Comput. Optim. Appl.*, 28:185–202, 2004.

P. Gehler and S. Nowozin. Infinite kernel learning. Technical report, 2008.

T. R. Golub, D. K. Slonim, and P. Tamayo. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 7:286–531, 1999.

Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. In *NIPS*, 2002.

Q. Gu, Z. Li, and J. Han. Correlated multi-label feature selection. In *CIKM*, 2011a.

Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. In *UAI*, 2011b.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3: 1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46:389–422, 2002.

C. J. Hsieh, K. W. Chang, C. J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *ICML*, 2008.

A. Jain, S.V.N. Vishwanathan, and M. Varma. SPF-GMKL: generalized multiple kernel learning with a million kernels. In *KDD*, 2012.

R. Jenatton, J. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, 2011a.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *JMLR*, 12:2297–2334, 2011b.

T. Joachims. Training linear SVMs in linear time. In *KDD*, 2006.

J. E. Kelley. The cutting plane method for solving convex programs. *J. Soc. Ind. Appl. Math.*, 8(4):703 –712, 1960.

S. Kim and E. P. Xing. Tree-guided group lasso for multi-response regression with structured sparsity, with applications to eQTL mapping. *Ann. Statist.*, Forthcoming, 2012.

S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML 2010*, 2010.

S. Kim and S. Boyd. On general minimax theorems. *Pacific J. Math.*, 1958, 8(1):171–176, 1958.

M. Kloft and G. Blanchard. On the convergence rate of ℓ_p-norm multiple kernel learning. *JMLR*, 13:2465–2501, 2012.

M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K. Müller, and A. Zien. Efficient and accurate ℓ_p-norm multiple kernel learning. *NIPS*, 22(22):997–1005, 2009.

M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. Lp-norm multiple kernel learning. *JMLR*, 12:953–997, 2011.

R. Kohavi and G. John. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, 1997.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *JMLR*, 10:777–801, 2009.

P. Li, A. Shrivastava, J. Moore, and A. C. Konig. Hashing algorithms for large-scale learning. In *NIPS*, 2011.

P. Li, A. Owen, and C. H. Zhang. One permutation hashing. In *NIPS*, 2012.

D. Lin, D. P. Foster, and L. H. Ungar. A risk ratio comparison of ℓ_0 and ℓ_1 penalized regressions. Technical report, University of Pennsylvania, 2010.

J. Liu and J. Ye. Efficient l1/lq norm regularization. Technical report, 2010.

A. C. Lozano, G. Swirszcz, and N. Abe. Group orthogonal matching pursuit for logistic regression. In *AISTATS*, 2011.

S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009.

Q. Mao and I. W. Tsang. A feature selection method for multivariate performance measures. *IEEE Trans. Pattern Anal. Mach.*, 35(9):2051–2063, 2013.

A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. Online multiple kernel learning for structured prediction. Technical report, 2010.

L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *J. Roy. Stat. Soc. B.*, 70(1):53–71, 2008.

A. Mutapcic and S. Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optim. Method Softw.*, 24(3):381?06, 2009.

A. Nedic and A. Ozdaglar. Subgradient methods for saddle-point problems. *J. Optimiz. Theory App.*, 142(1):205–228, 2009.

A. Nemirovski. 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 J. Opt.*, 15:229–251, 2005.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, 2007.

A. Y. Ng. On feature selection: Learning with exponentially many irrelevant features as training examples. In *ICML*, 1998.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 2nd edition, 2006.

F. Orabona and L. Jie. Ultra-fast optimization algorithm for sparse multi kernel learning. In *ICML*, 2011.

E. Y. Pee and J. O. Royset. On solving large-scale finite minimax problems using exponential smoothing. *J. Optimiz. Theory App.*, online, 2010.

Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. Technical report, 2010.

A. Rakotomamonjy. Variable selection using svm-based criteria. *JMLR*, 3:1357–1370, 2003.

A. Rakotomamonjy, F. R. Bach, Y. Grandvalet, and S. Canu SimpleMKL. *JMLR*, 9: 2491–2521, 2008.

M. Rastegari, C. Fang, and L. Torresani. Scalable object-class retrieval with approximate and top-ranking. In *ICCV*, 2011.

V. Roth and B. Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *ICML*, 2008.

D. Sculley, G. M. Wachman, and C. E. Brodley. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *The Fifteenth Text REtrieval Conference (TREC) Proceedings*, 2006.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *JMLR*, 14:567–599, 2013.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large Scale Multiple Kernel Learning. *JMLR*, 7:1531–1565, 2006.

S. Sonnenburg, G. Rätsch, and K. Rieck. *Large scale learning with string kernels.* MIT Press, 2007.

M. Tan, I. W. Tsang, and L. Wang. Learning sparse svm for feature selection on very high dimensional data sets. In *ICML*, 2010.

M. Tan, I. W. Tsang, and L. Wang. Minimax sparse logistic regression for very high-dimensional feature selection. *IEEE Trans. Neural Netw. Learning Syst.*, 24(10):1609–1622, 2013.

A. Tewari, P. Ravikumar, and I. S. Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In *NIPS*, 2011.

K. C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. Technical report, 2009.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Technical report, University of Washington, 2008.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optimiz. Theory App.*, 109(3):475–494, 2001.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, 2009.

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

S. V. N. Vishwanathan, Z. Sun, N. Theera-Ampornpunt, and M. Varma. Multiple kernel learning and the SMO algorithm. In *NIPS*, December 2010.

K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, 2009.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *NIPS*, 2000.

J. Wu. Efficient hik svm learning for image classification. *IEEE Trans. Image Process*, 21(10):4442–4453, 2012.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. In *NIPS*, 2009.

Z. Xu, R. Jin, Ye J., Michael R. Lyu, and King I. Non-monotonic feature selection. In *ICML*, 2009a.

Z. Xu, R. Jin, I. King, and M.R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*. 2009b.

Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, 2010.

G. X. Yuan, K. W. Chang, C. J. Hsieh, and C. J. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *JMLR*, 11:3183–3236, 2010.

G. X. Yuan, C. H. Ho, and C. J. Lin. An improved GLMNET for l1-regularized logistic regression and support vector machines. *JMLR*, 13:1999–2030, 2012.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc. B.*, 68(1):49–67, 2006.

C. H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Ann. Statist.*, 38(2):894–942, 2010a.

C. H. Zhang and J. Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Ann. Statist.*, 36(4):1567–1594, 2008.

D. Zhang and W. S. Lee. Extracting key-substring-group features for text classification. In *KDD*, 2006.

T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *JMLR*, 11: 1081–1107, 2010b.

Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *ICML*, 2007.

J. Zhu, S. Rossett, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *NIPS*, 2003.