Machine Learning DOI 10.1007/s10994-013-5367-2

Block Coordinate Descent Algorithms for Large-scale Sparse Multiclass Classification

Mathieu Blondel
Kazuhiro Seki
Kuniaki Uehara

Received: 4 November 2012 / Accepted: 23 April 2013

Abstract Over the past decade, ℓ_1 regularization has emerged as a powerful way to learn classifiers with implicit feature selection. More recently, mixed-norm (e.g., ℓ_1/ℓ_2) regularization has been utilized as a way to select entire groups of features. In this paper, we propose a novel direct multiclass formulation specifically designed for large-scale and high-dimensional problems such as document classification. Based on a multiclass extension of the squared hinge loss, our formulation employs ℓ_1/ℓ_2 regularization so as to force weights corresponding to the same features to be zero across all classes, resulting in compact and fast-to-evaluate multiclass models. For optimization, we employ two globally-convergent variants of block coordinate descent, one with line search (Tseng and Yun, 2009) and the other without (Richtárik and Takáč, 2012). We present the two variants in a unified manner and develop the core components needed to efficiently solve our formulation. The end result is a couple of block coordinate descent algorithms specifically tailored to our multiclass formulation. Experimentally, we show that block coordinate descent performs favorably to other solvers such as FOBOS, FISTA and SpaRSA. Furthermore, we show that our formulation obtains very compact multiclass models and outperforms ℓ_1/ℓ_2-regularized multiclass logistic regression in terms of training speed, while achieving comparable test accuracy.

Keywords multiclass classification · group sparsity · block coordinate descent

1 Introduction

ℓ_1-regularized loss minimization has attracted a great deal of research over the past decade [33]. ℓ_1 regularization has many advantages, including its computational efficiency, its ability to perform implicit feature selection and under certain conditions,

Mathieu Blondel · Kazuhiro Seki · Kuniaki Uehara
Graduate School of System Informatics
Kobe University
1-1 Rokkodai, Nada, Kobe 657-8501, Japan
E-mail: mathieu@mblondel.org

to recover the model's true sparsity [37]. More recently, mixed-norm (e.g., ℓ_1/ℓ_2) regularization has been proposed [2,35] as a way to select groups of features. Here, the notion of group is application-dependent and may be used to exploit prior knowledge about natural feature groups [35,20] or problem structure [21,10].

In this paper, we focus on the application of ℓ_1/ℓ_2 regularization to multiclass classification problems. Let $\boldsymbol{W}$ be a $d \times m$ matrix, where d represents the number of features and m the number of classes. We denote by $\boldsymbol{W}_{j:} \in \mathbf{R}^m$ the j^{th} row of $\boldsymbol{W}$ and by $\boldsymbol{W}_{:r} \in \mathbf{R}^d$ its r^{th} column. We consider the traditional multiclass model representation where an input vector $\boldsymbol{x} \in \mathbf{R}^d$ is classified to one of the m classes using the following rule:

$$y = \underset{r \in \{1,\ldots,m\}}{\operatorname{argmax}} \boldsymbol{W}_{:r} \cdot \boldsymbol{x}. \tag{1}$$

Each column $\boldsymbol{W}_{:r}$ of $\boldsymbol{W}$ can be thought as a prototype representing the r^{th} class and the inner product $\boldsymbol{W}_{:r} \cdot \boldsymbol{x}$ as the score of the r^{th} class with respect to $\boldsymbol{x}$. Therefore, Eq. (1) chooses the class with highest score. Given n training instances $\boldsymbol{x}_i \in \mathbf{R}^d$ and their associated labels $y_i \in \{1, \ldots, m\}$, our goal is to estimate $\boldsymbol{W}$.

In this paper, we propose a novel *direct* multiclass formulation specifically designed for large-scale and high-dimensional problems such as document classification. Based on a multiclass extension of the squared hinge loss, our formulation employs ℓ_1/ℓ_2 regularization so as to force weights corresponding to the same features to be zero across all classes, resulting in compact and fast-to-evaluate multiclass models (see Section 2). For optimization, we employ two globally-convergent variants of block coordinate descent, one with line search (Tseng and Yun [28]) and the other without (Richtárik and Takáč [23]). We present the two variants in a unified manner and develop core components needed to optimize our objective (efficient gradient computation, Lipschitz constant of the gradient, computationally cheap stopping criterion). The end result is a couple of block coordinate descent algorithms specifically tailored to our multiclass formulation. Experimentally, we show that block coordinate descent performs favorably to other solvers such as FOBOS [11], FISTA [3] and SpaRSA [32]. Furthermore, we show that our formulation obtains very compact multiclass models and outperforms ℓ_1/ℓ_2-regularized multiclass logistic regression in terms of training speed, while achieving comparable test accuracy.

2 Sparsity-inducing regularization

Fig. 1 illustrates sparsity patterns obtained by different forms of regularization and shows why ℓ_1/ℓ_2 regularization is particularly well adapted to multiclass models. With ℓ_2^2 regularization (ridge), $R_{\ell_2^2}(\boldsymbol{W}) = \frac{1}{2}\sum_{j,r} \boldsymbol{W}_{jr}^2$, sparsity is not enforced and therefore the obtained model may become completely dense. With ℓ_1 regularization (lasso), $R_{\ell_1}(\boldsymbol{W}) = \sum_{j,r} |\boldsymbol{W}_{jr}|$, the model becomes sparse at the individual weight level. A well-known problem of ℓ_1 regularization is that, if several features are correlated, it tends to select only one of them, even if other features are useful for prediction. To solve this problem, ℓ_1 regularization can be combined with ℓ_2^2 regularization. The resulting regularization, $R_{\ell_1+\ell_2^2}(\boldsymbol{W}) = \rho R_{\ell_1}(\boldsymbol{W}) + (1-\rho) R_{\ell_2^2}(\boldsymbol{W})$, where $\rho > 0$ is a hyperparameter, is known as elastic-net in the literature [38] and leads to sparsity at the individual weight level.

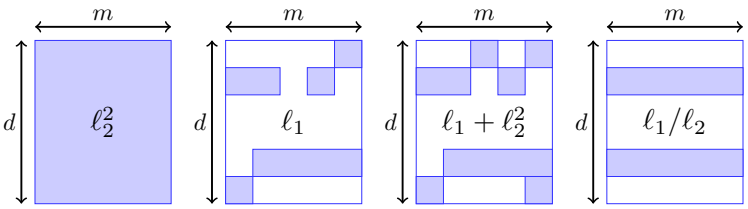


Fig. 1 Illustration of the sparsity patterns obtained by ℓ_2^2 (ridge), ℓ_1 (lasso), $\ell_1+\ell_2^2$ (elastic-net) and ℓ_1/ℓ_2 (group lasso) regularizations on the matrix $\boldsymbol{W} \in \mathbf{R}^{d\times m}$. With ℓ_1/ℓ_2 regularization, we can obtain compact and fast-to-evaluate multiclass models.

Let $R_{\ell_2}(\boldsymbol{W}_{j:}) = ||\boldsymbol{W}_{j:}||_2$ (notice that the ℓ_2 norm is not squared). With ℓ_1/ℓ_2 regularization (group lasso), $R_{\ell_1/\ell_2}(\boldsymbol{W}) = \sum_j R_{\ell_2}(\boldsymbol{W}_{j:})$, the model becomes sparse at the feature group (here, row) level. Applied to a multiclass model, ℓ_1/ℓ_2 regularization can thus force weights corresponding to the same feature to become zero across all classes. The corresponding features can therefore be safely ignored at test time, which is especially useful when features are expensive to extract. For more information on sparsity inducing penalties, see Bach et al. 's excellent survey [1].

3 Related work

3.1 Multiclass classification: direct vs. indirect formulations

Classifying an object into one of several categories is an important problem arising in many applications such as document classification and object recognition. Machine learning approaches to this problem can be roughly divided into two categories: direct and indirect approaches. While direct approaches formulate the multiclass problem directly, indirect approaches reduce the multiclass problem to multiple independent binary classification or regression problems. Because support vector machines (SVMs) [5] were originally proposed as a binary classification model, they have frequently been used in combination with indirect approaches to perform multiclass classification. Among them, one of the most popular is "one-vs-rest" [25], which consists in learning to separate one class from all the others, independently for all m possible classes. Direct multiclass SVM extensions were later proposed by Weston-Watkins [30], Lee et al. [18] and Crammer-Singer [8]. They were all formulated as constrained problems and solved in the dual. An unconstrained (non-differentiable) form of the Crammer-Singer formulation is popularly used with stochastic subgradient descent algorithms such as Pegasos [26]. Another popular direct multiclass (smooth) formulation, which is an intuitive extension of traditional logistic regression, is multiclass logistic regression. In this paper, we propose an efficient direct multiclass formulation.

3.2 Sparse multiclass classification

Recently, mixed-norm regularization has attracted much interest [35, 20, 11, 21, 10] due to its ability to impose sparsity at the feature group level. Few papers, how-

ever, have investigated its application to multiclass classification. Zhang et al. [36] extend Lee et al.'s multiclass SVM formulation [18] to employ ℓ_1/ℓ_∞ regularization and formulate the learning problem as a linear program (LP). However, they experimentally verify their method only on very small problems (both in terms of n and d). Duchi and Singer [10] propose a boosting-like algorithm specialized for ℓ_1/ℓ_2-regularized multiclass logistic regression. In another paper, Duchi and Singer [11] derive and analyze FOBOS, a stochastic subgradient descent framework based on forward-backward splitting and apply it, among other things, to ℓ_1/ℓ_2-regularized multiclass logistic regression. In this paper, we choose ℓ_1/ℓ_2 regularization, since it can be more efficiently optimized than ℓ_1/ℓ_∞ regularization (see Section 4.7).

3.3 Coordinate descent methods

Although coordinate descent methods were among the first optimization methods proposed and studied in the literature (see [4] and references therein), it is only recently that they regained popularity, thanks to several successful applications in the machine learning [17,27,14,16,33,22] and optimization [28,31,23] communities. Conceptually and algorithmically simple, (block) coordinate descent algorithms focus at each iteration on updating one block of variables while keeping the others fixed, and have been shown to be particularly well-suited for minimizing objective functions with non-smooth separable regularization such as ℓ_1 or ℓ_1/ℓ_2 [28,31,23].

Coordinate descent algorithms have different trade-offs: expensive gradient-based greedy block selection as opposed to cheap cyclic or randomized selection, use of line search [28,31] or not [23]. For large-scale linear classification, and we confirm in this paper, cyclic and randomized block selection schemes have been shown to achieve excellent performance [33,6,34,23]. The most popular loss function for ℓ_1-regularized binary classification is arguably logistic regression, due to its smoothness [33]. Binary logistic regression was also successfully combined with ℓ_1/ℓ_2 regularization in the case of user-defined feature groups [20]. However, recent works [33,6,34] using coordinate descent indicate that logistic regression is substantially slower to train than ℓ_2-loss (squared hinge) SVMs. This is because, contrary to ℓ_2-loss SVMs, logistic regression requires expensive log and exp computations (equivalent to dozens of multiplications) to compute the gradient or objective value [34]. Motivated by this background, we propose a novel efficient *direct* multiclass formulation. Compared to multiclass logistic regression, which suffers from the same problems as its binary counterpart, our formulation can be optimized very efficiently by block coordinate descent and lends itself to large-scale and high-dimensional problems such as document classification.

4 Sparse direct multiclass classification

4.1 Objective function

Given n training instances $\boldsymbol{x}_i \in \mathbf{R}^d$ and their associated labels $y_i \in \{1, \ldots, m\}$, our goal is to estimate $\boldsymbol{W}$ such that Eq. (1) produces accurate predictions and $\boldsymbol{W}$ is

Algorithm 1 Block-coordinate algorithm for minimization of $F(\boldsymbol{W})$

$\boldsymbol{W} \leftarrow \mathbf{0}_{d\times m}$
for $k = 1, 2, \ldots, K$ **do**
  for $l = 1, 2, \ldots, d$ **do**
    Choose block j [Section 4.4]
    Update $\boldsymbol{W}_{j:} \leftarrow \boldsymbol{W}_{j:} + \alpha_j \boldsymbol{\delta}_j$ [Algorithm 2]
  end for
  Stop if a suitable criterion is met [Section 4.5]
end for

row-wise sparse. To this end, we minimize the following convex objective:

$$\underset{\boldsymbol{W}\in\mathbf{R}^{d\times m}}{\text{minimize}}\ F(\boldsymbol{W}) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\sum_{r\neq y_i}\max(1-(\boldsymbol{W}_{:y_i}\cdot\boldsymbol{x}_i - \boldsymbol{W}_{:r}\cdot\boldsymbol{x}_i),0)^2}_{L(\boldsymbol{W})} + \lambda\underbrace{\sum_{j=1}^{d}||\boldsymbol{W}_{j:}||_2}_{R(\boldsymbol{W})}, \tag{2}$$

where $\lambda > 0$ is a parameter controlling the trade-off between loss and penalty minimization. We call $F(\boldsymbol{W})$ ℓ_1/ℓ_2-regularized multiclass squared hinge loss function. Intuitively, for all training instances and classes (different from the correct label), if the score is less than the score assigned to the correct label by at least 1, the model suffers zero loss. Otherwise, it suffers a loss which is quadratically proportional to the difference between the scores. Besides convexity, $F(\boldsymbol{W})$ possesses the following desirable properties:

1. It is a direct multiclass formulation and its relation with Eq. (1) is intuitive.
2. Its objective value and gradient can be computed efficiently (unlike multiclass logistic regression, which requires expensive log and exp operations).
3. It empirically performs comparably or better than other multitask and multiclass formulations.
4. It meets several conditions needed to prove global convergence of block coordinate descent algorithms (see Section 4.6).

Our objective, Eq. (2), is similar in spirit to Weston and Watkins' multiclass SVM formulation [30], in that it ensures that the correct class's score is greater than all the other classes by at least 1. However, it has the following differences: it is unconstrained (rather than constrained), it is ℓ_1/ℓ_2-regularized (rather than ℓ_2^2-regularized) and it penalizes misclassifications quadratically (rather linearly), which ensures differentiability of $L(\boldsymbol{W})$.

4.2 Optimization by block coordinate descent

A key property of $F(\boldsymbol{W})$ is the separability of its non-smooth part $R(\boldsymbol{W})$ over groups $j = 1, 2, \ldots, d$. This calls for an algorithm which minimizes $F(\boldsymbol{W})$ by updating $\boldsymbol{W}$ group by group. In this paper, to minimize $F(\boldsymbol{W})$, we thus employ block coordinate descent. We consider two variants, one with line search (Tseng and Yun [28]) and the other without (Richtárik and Takáč [23]). We present the two variants in a unified manner.

Algorithm 2 Solving the block sub-problem associated with $\boldsymbol{W}_{j:}$.

Compute $G(\boldsymbol{W})_{j:}$ [Algorithm 3]
Choose $\mathcal{L}_j$ [Section 4.4]
Compute
$\boldsymbol{V}_{j:} = \boldsymbol{W}_{j:} - \frac{1}{\mathcal{L}_j} G(\boldsymbol{W})_{j:}$
$\mu_j = \frac{\lambda}{\mathcal{L}_j}$
$\boldsymbol{W}^*_{j:} = \text{Prox}_{\mu_j||\cdot||_2}(\boldsymbol{V}_{j:}) = \max\{1 - \frac{\mu_j}{||\boldsymbol{V}_{j:}||_2}, 0\}\boldsymbol{V}_{j:}$
$\boldsymbol{\delta}_j = \boldsymbol{W}^*_{j:} - \boldsymbol{W}_{j:}$
Choose α_j [Section 4.4]
Update $\boldsymbol{W}_{j:} \leftarrow \boldsymbol{W}_{j:} + \alpha_j \boldsymbol{\delta}_j$

Algorithm 1 outlines block coordinate descent for minimizing $F(\boldsymbol{W})$. At each iteration, Algorithm 1 selects a block $\boldsymbol{W}_{j:} \in \mathrm{R}^m$ of coefficients and updates it, keeping all other blocks fixed (how to choose the block is delayed to Section 4.4). This procedure is repeated several times until a suitable stopping criterion is met or the maximum number of outer iterations K is reached. The main difficulty arising in Algorithm 1 is how to solve the sub-problem associated with each weight block $\boldsymbol{W}_{j:}$. Let $\boldsymbol{W}^t$ be the weight matrix at iteration t. The key idea of block coordinate descent frameworks for non-smooth separable minimization [28,23] is to update each block by solving the following quadratic approximation of F around $\boldsymbol{W}^t$:

$$\boldsymbol{W}^*_{j:} = \underset{\boldsymbol{W}_{j:} \in \mathbf{R}^m}{\text{argmin}} \; G(\boldsymbol{W}^t)^{\mathrm{T}}_{j:}(\boldsymbol{W}_{j:} - \boldsymbol{W}^t_{j:}) + \frac{1}{2}(\boldsymbol{W}_{j:} - \boldsymbol{W}^t_{j:})^{\mathrm{T}} \boldsymbol{H}^t (\boldsymbol{W}_{j:} - \boldsymbol{W}^t_{j:}) + \lambda ||\boldsymbol{W}_{j:}||_2, \tag{3}$$

where we used $G(\boldsymbol{W})_{j:} \in \mathbf{R}^m$ to denote the j^{th} row of the gradient of $L(\boldsymbol{W})$ and $\boldsymbol{H}^t$ is a $m \times m$ matrix. If we choose $\boldsymbol{H}^t = \mathcal{L}^t_j \boldsymbol{I}$ where $\mathcal{L}^t_j$ is a scalar (we discuss its choice in Section 4.4) and $\boldsymbol{I}$ is the identity matrix, Eq. (3) can be rewritten as:

$$\boldsymbol{W}^*_{j:} = \underset{\boldsymbol{W}_{j:} \in \mathbf{R}^m}{\text{argmin}} \; \frac{1}{2}||\boldsymbol{W}_{j:} - \boldsymbol{V}^t_{j:}||^2 + \mu^t_j ||\boldsymbol{W}_{j:}||_2$$

where we defined $\boldsymbol{V}^t_{j:} = \boldsymbol{W}^t_{j:} - \frac{1}{\mathcal{L}^t_j} G(\boldsymbol{W}^t)_{j:}$ and $\mu^t_j = \frac{\lambda}{\mathcal{L}^t_j}$. This problem takes a form which is well-known in the signal-processing literature and whose solution is called proximity operator [7]. The proximity-operator associated with the ℓ_2 norm takes a closed form (see e.g. [11] for a derivation):

$$\boldsymbol{W}^*_{j:} = \text{Prox}_{\mu^t_j||\cdot||_2}(\boldsymbol{V}^t_{j:}) = \max\{1 - \frac{\mu^t_j}{||\boldsymbol{V}^t_{j:}||_2}, 0\}\boldsymbol{V}^t_{j:}. \tag{4}$$

This operator is known as vectorial soft-thresholding operator [32], owing to the fact that $\boldsymbol{W}^*_{j:}$ becomes entirely zero when $1 - \frac{\mu^t}{||\boldsymbol{V}^t_{j:}||_2} < 0$. Summarizing, we obtain $\boldsymbol{W}^*_{j:}$ by taking a partial gradient step with step size $\frac{1}{\mathcal{L}^t_j}$ and then projecting the result by $\text{Prox}_{\mu^t_j||\cdot||_2}$. Finally, let $\boldsymbol{\delta}^t_j = \boldsymbol{W}^*_{j:} - \boldsymbol{W}^t_{j:}$. The last step consists in setting $\boldsymbol{W}^{t+1}_{j:} = \boldsymbol{W}^t_{j:} + \alpha^t_j \boldsymbol{\delta}^t_j$. We discuss the choice of α^t_j in Section 4.4. Algorithm 2 summarizes how to solve the block sub-problem associated with $\boldsymbol{W}_{j:}$ (we drop the superscript t since there is no ambiguity).

4.3 Efficient partial gradient computation

We now discuss efficient computation of the partial gradient $G(\boldsymbol{W})_{j:}$ of $L(\boldsymbol{W})$, which is crucial for the general efficiency of Algorithm 1. We first rewrite $L(\boldsymbol{W})$ as:

$$L(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{r\neq y_i}\max(A(\boldsymbol{W})_{ir}, 0)^2,$$

where $A(\boldsymbol{W})$ is a $n \times m$ matrix defined by:

$$A(\boldsymbol{W})_{ir} = 1 - (\boldsymbol{W}_{:y_i} \cdot \boldsymbol{x}_i - \boldsymbol{W}_{:r} \cdot \boldsymbol{x}_i).$$

The partial gradient of $L(\boldsymbol{W})$ can then be concisely written as:

$$G(\boldsymbol{W})_{j:} = -\frac{2}{n}\sum_{i=1}^{n}\sum_{r\neq y_i}\max(A(\boldsymbol{W})_{ir}, 0)[\boldsymbol{x}_{ij}\boldsymbol{e}_{y_i} - \boldsymbol{x}_{ij}\boldsymbol{e}_r], \tag{5}$$

where $\boldsymbol{e}_r = [\underbrace{0, \ldots, 0}_{r-1}, 1, 0, \ldots, 0]^{\mathrm{T}}$.

Since computing $A(\boldsymbol{W})_{ir}$ from scratch would be computationally prohibitive, we instead initialize $A(\boldsymbol{W})$ to $\mathbf{1}_{n\times m}$ at the beginning of Algorithm 1, then when a weight block is updated by $\boldsymbol{W}_{j:} \leftarrow \boldsymbol{W}_{j:} + \alpha_j\boldsymbol{\delta}_j$, we update $A(\boldsymbol{W})$ by $A(\boldsymbol{W})_{ir} \leftarrow A(\boldsymbol{W})_{ir} + \alpha_j(\boldsymbol{\delta}_{jr} - \boldsymbol{\delta}_{jy_i})\boldsymbol{x}_{ij}$ for all i such that $\boldsymbol{x}_{ij} \neq 0$ and all $r \neq y_i$. Thanks to this implementation technique, denoting $\hat{n}$ the average number of non-zero values per feature, the cost of computing Eq. (5) is only $O(\hat{n}(m-1))$. We summarize how to efficiently compute $G(\boldsymbol{W})_{j:}$ in Algorithm 3. When using sparse data, the compressed sparse column (CSC) format can be used for fast access to all non-zero values of feature j (inner loop in Algorithm 3).

4.4 Choice of block, $\mathcal{L}_j^t$ and α_j^t

We now discuss how to choose, at every iteration, the block $\boldsymbol{W}_{j:}$ to update, $\mathcal{L}_j^t$ and α_j^t, depending on whether a line search is used or not.

4.4.1 With line search (Tseng and Yun)

Following Tseng and Yun [28], we can choose

$$\mathcal{L}_j^t = \max(||h(\boldsymbol{W}^t)_{j:}||_\infty, \epsilon),$$

where ϵ is a small constant (e.g., 10^{-12}) to ensure positivity and:

$$h(\boldsymbol{W})_{j:} = [\frac{\partial^2 L}{\partial \boldsymbol{W}_{j1}^2}, \ldots, \frac{\partial^2 L}{\partial \boldsymbol{W}_{jm}^2}]^{\mathrm{T}}.$$

In our case, L is not twice-differentiable, since $G(\boldsymbol{W})$ is not differentiable when $A(\boldsymbol{W})_{ir} = 0$. We can however define its generalized second derivatives [19,6]:

$$h(\boldsymbol{W})_{j:} = \frac{2}{n}\sum_{i=1}^{n}\sum_{r\neq y_i}\delta_{[A(\boldsymbol{W})_{ir}>0]}(\boldsymbol{x}_{ij}^2\boldsymbol{e}_{y_i} + \boldsymbol{x}_{ij}^2\boldsymbol{e}_r), \tag{6}$$

Algorithm 3 Efficient computation of $G(\boldsymbol{W})_{j:}$ and $h(\boldsymbol{W})_{j:}$

```
G(W)_{j:} ← 0_m
h(W)_{j:} ← 0_m
for r = 1, 2, ..., m do
  for i = 1, 2, ..., n such that x_ij ≠ 0 do
    if y_i ≠ r and A(W)_ir > 0 then
      G(W)_{j y_i} ← G(W)_{j y_i} − (2/n) A(W)_ir x_ij
      G(W)_{jr} ← G(W)_{jr} + (2/n) A(W)_ir x_ij
      h(W)_{j y_i} ← h(W)_{j y_i} + (2/n) x_ij^2
      h(W)_{jr} ← h(W)_{jr} + (2/n) x_ij^2
    end if
  end for
end for
```

where $\delta[.]$ is the Kronecker delta. Since choosing $\mathcal{L}_j^t$ as above might lead to an overly large step size $\frac{1}{\mathcal{L}_j^t}$, Tseng and Yun choose α_j^t such that the following sufficient decrease condition is satisfied:

$$F(\boldsymbol{W}^{t+1}) - F(\boldsymbol{W}^t) \leq \sigma\alpha_j^t(G(\boldsymbol{W}^t)_{j:}^{\mathrm{T}}\boldsymbol{\delta}_j + \lambda||\boldsymbol{W}_{j:}^t + \boldsymbol{\delta}_j||_2 - \lambda||\boldsymbol{W}_{j:}^t||_2), \tag{7}$$

where σ is a user-defined constant such that $0 < \sigma < 1$. We can choose α_j^t by backtracking line search, that is, by sequentially trying $\alpha_j^t = 1, \omega, \omega^2, \ldots$ until Eq. (7) is satisfied. Common choices in the optimization literature for σ and ω are 0.01 and 0.5, respectively. Since we have $\boldsymbol{W}_{j:}^{t+1} = (1 - \alpha_j^t)\boldsymbol{W}_{j:}^t + \alpha_j^t\boldsymbol{W}_{j:}^*$, we see that $\boldsymbol{W}_{j:}^{t+1}$ can be interpreted as a weighted sum between the current iterate and the subproblem's solution.

Similarly to Eq. (5), the cost of computing Eq. (7) and Eq. (6) is $O(\hat{n}(m-1))$. In practice, we observe that one line search step often suffices for Eq. (7) to be satisfied. Therefore, the cost of one call to Algorithm 2 is in general $O(\hat{n}(m-1))$.

To enjoy Tseng and Yun's theoretical guarantees (see Section 4.6), we need to use cyclic block selection. That is, in Algorithm 1, at each inner iteration, we need to choose $j = l$.

4.4.2 Without line search (Richtárik and Takáč)

We show in Appendix A that $G(\boldsymbol{W})_{j:}$ is Lipschitz with constant

$$\mathcal{K}_j = \frac{4(m-1)}{n}\sum_i \boldsymbol{x}_{ij}^2.$$

Following Richtárik and Takáč [23], we can choose $\mathcal{L}_j^t = \mathcal{K}_j$. In that case, no line search is needed, i.e., $\alpha_j^t = 1$ and $\boldsymbol{W}_{j:}^{t+1} = \boldsymbol{W}_{j:}^*$. Our implementation pre-computes $\mathcal{K}_j \ \forall j \in \{1, \ldots, d\}$ and stores the results in a d-dimensional vector. Note that, Richtárik and Takáč assume that blocks are selected with uniform probability $\frac{1}{d}$.

Using a line search or not is a matter of trade-off: using a line search has higher cost per iteration but can potentially lead to greater progress due to the larger step size. We compare both strategies experimentally in Section 5.2. One advantage of Richtárik and Takáč's framework, however, is that it can be parallelized [24], potentially leading to significant speedups. In future work, we plan to compare sequential and parallel block coordinate descent when applied to our objective, Eq. (2).

4.5 Stopping criterion

We would like to develop a stopping criterion for Algorithm 1 which can be checked at almost no extra computational cost. Proposition 1 characterizes an optimal solution of Eq. (2).

Proposition 1 *$\boldsymbol{W}$ is an optimal solution of Eq. (2) if and only if $\forall j$:*

$$\begin{cases} ||G(\boldsymbol{W})_{j:}||_2 \leq \lambda & \text{if } \boldsymbol{W}_{j:} = \mathbf{0} \quad (8a) \\ G(\boldsymbol{W})_{j:} + \dfrac{\lambda \boldsymbol{W}_{j:}}{||\boldsymbol{W}_{j:}||_2} = \mathbf{0} & \text{if } \boldsymbol{W}_{j:} \neq \mathbf{0}. \quad (8b) \end{cases}$$

Proof is given in Appendix B. Using Proposition 1 and the fact that Eq. (8b) is equivalent to $||G(\boldsymbol{W})_{j:}||_2 = \lambda$ if $\boldsymbol{W}_{j:} \neq \mathbf{0}$, we define v^t, the optimality violation at the t^{th} iteration (the bigger, the stronger the violation):

$$v^t = \begin{cases} \max(||G(\boldsymbol{W}^t)_{j(t):}||_2 - \lambda, 0) & \text{if } \boldsymbol{W}^t_{j(t):} = \mathbf{0} \\ |||G(\boldsymbol{W}^t)_{j(t):}||_2 - \lambda| & \text{if } \boldsymbol{W}^t_{j(t):} \neq \mathbf{0}, \end{cases} \quad (9)$$

where $j(t)$ denotes the block selected at the t^{th} iteration. In Eq. (9), the max operator is to account for the inequality in (8a) and the absolute value for the equality in (8b). Since we already need $G(\boldsymbol{W}^t)_{j(t):}$ for solving each block sub-problem, computing v^t comes at almost no extra cost.

As indicated in Algorithm 1, we check convergence at the end of each outer iteration. Let $\mathcal{T}^k = \{(k-1)d+1, (k-1)d+2, \ldots, kd\}$ be the set of values taken by t at the k^{th} outer iteration. One possible stopping criterion is:

$$\frac{\sum_{t \in \mathcal{T}^k} v^t}{\sum_{t \in \mathcal{T}^1} v^t} < \tau, \quad (10)$$

where $0 < \tau \leq 1$ is a user-defined tolerance constant (the bigger, the faster to stop). This criterion is the most natural when cyclic block selection is used, since the sums in Eq. (10) are then over all blocks $1, \ldots, d$. Another possible stopping criterion consists in replacing the ℓ_1 norm by the ℓ_∞ norm:

$$\frac{\max_{t \in \mathcal{T}^k} v^t}{\max_{t \in \mathcal{T}^1} v^t} < \tau.$$

We use this criterion when randomized uniform block selection is used. In both cases, the denominator serves the purpose of normalization (hence, τ is not sensitive to the dataset dimensionality).

4.6 Global convergence properties

We discuss convergence properties for the two block coordinate descent variants we considered: cyclic block coordinate descent with line search (Tseng and Yun [28]) and randomized block coordinate descent without line search (Richtárik and Takáč [23]). To have finite termination of the line search, Tseng and Yun (Lemma 5.1),

require that L has Lipschitz continuous gradient, which we prove with Lemma 1 in Appendix A. For asymptotic convergence, Tseng and Yun assume that each block is cyclically visited (Equation (12)). They further assume (Assumption 1) that $\boldsymbol{H}^t$ is upper-bounded by some value and lower-bounded by 0, which is guaranteed by our choice $\boldsymbol{H}^t = \mathcal{L}_j^t \boldsymbol{I}$. Richtárik and Takáč also assume (Section 2) that the blockwise gradient is Lipschitz. They show (Theorem 4) that using their algorithm, there exists a finite iteration t such that $P(F(\boldsymbol{W}^t) - F(\boldsymbol{W}^*) \leq \epsilon) \geq 1 - \rho$, where $\epsilon > 0$ is the accuracy of the solution and $0 < \rho < 1$ is the target confidence.

4.7 Extensions

A straightforward extension of our objective, Eq. (2), is label ranking with multilabel data:

$$\underset{\boldsymbol{W} \in \mathbf{R}^{d \times m}}{\text{minimize}} \frac{1}{n} \sum_{i=1}^{n} \sum_{r \in \mathcal{Y}_i, r' \notin \mathcal{Y}_i} \max(1 - (\boldsymbol{W}_{:r} \cdot \boldsymbol{x}_i - \boldsymbol{W}_{:r'} \cdot \boldsymbol{x}_i), 0)^2 + \lambda \sum_{j=1}^{d} ||\boldsymbol{W}_{j:}||_2,$$

where $\mathcal{Y}_i$ is the set of labels assigned to $\boldsymbol{x}_i$. Intuitively, this objective attempts to assign higher score to relevant labels than to non-relevant labels. If the goal is to predict label sets rather than label rankings, threshold selection methods [13, 12] may be applied as a post-processing step.

Another possible extension consists in replacing ℓ_1/ℓ_2 regularization by ℓ_1/ℓ_∞ regularization or $\ell_1 + \ell_1/\ell_2$ regularization (sparse group lasso [15]). This requires changing the proximity operator, Eq. (4), as well as reworking the stopping criterion developed in Section 4.5. Similarly to ℓ_1/ℓ_2 regularization, ℓ_1/ℓ_∞ regularization leads to group sparsity. However, the proximity operator associated with the ℓ_∞ norm requires a projection on an ℓ_1-norm ball [1] and is thus computationally more expensive than the proximity operator associated with the ℓ_2 norm, which takes a closed form, Eq. (4). For $\ell_1 + \ell_1/\ell_2$ regularization (sparse group lasso), the groupwise proximity operator can readily be computed by applying first the proximity operator associated with the ℓ_1 norm and then the one associated with the ℓ_2 norm [1]. However, sparse group lasso regularization requires the tuning of an extra hyperparameter, which balances between ℓ_1/ℓ_2 and ℓ_1 regularizations. For this reason, we do not consider it in our experiments.

Table 1 Datasets used in Section 5.

Dataset	Instances	Features	Non-zero features	Classes
Amazon7	1,362,109	262,144	0.04%	7
RCV1	534,135	47,236	0.1%	52
MNIST	70,000	780	19%	10
News20	18,846	130,088	0.1%	20
Sector	9,619	55,197	0.3%	105

5 Experiments

We conducted two experiments. In the first experiment, we investigated the performance (in terms of speed of convergence and row sparsity) of block coordinate descent (with or without line search) for optimizing the proposed direct multiclass formulation Eq. (2), compared to other state-of-the-art solvers. In the second experiment, we compared the proposed direct multiclass formulation with other multiclass and multitask formulations in terms of test accuracy, row sparsity and training speed. Experiments were run on a Linux machine with an Intel Xeon CPU (3.47GHz) and 4GB memory.

5.1 Datasets

Table 1 summarizes the datasets we used to conduct our experiments:

- Amazon7: product-review (books, DVD, electronics, ...) classification.
- RCV1: news document classification.
- MNIST: handwritten digit classification.
- News20: newgroup message classification.
- Sector: web-page (industry sectors) classification.

We created Amazon7 using the entire data of Dredze et al. [9] (they used only a small subset). For the scale of this dataset, constructing feature vectors from raw text by conventional bag-of-words extraction exceeded the memory of our computer. For this reason, we instead used the hashing trick [29] (a popular technique for large-scale and high-dimensional linear classification problems) and set the dimensionality to $d = 2^{18}$. Amazon7 is available for download from `http://www.mblondel.org/data/`. Other datasets are available in vectorized form from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. To determine test accuracy, we used stratified selection in order to split each dataset into 4/5 training and 1/5 testing.

5.2 Comparison of block coordinate descent with other solvers

In this section, we compare different solvers:

- BCD (LS): block coordinate descent with line search and with cyclic block selection (Tseng and Yun [28]),
- BCD (CST): block coordinate descent without line search and with randomized uniform block selection (Richtárik and Takáč [23]),

– FISTA (LS): an accelerated iterative thresholding algorithm with line search (Beck and Teboulle [3]),
– FISTA (CST): same as above but with constant step size $\frac{1}{K}$ (see Appendix A),
– SpaRSA: a similar approach to ISTA [3] but with different line search (Wright et al. [32]),
– FOBOS: a projected stochastic subgradient descent framework (Duchi and Singer [11]).

All solvers are used to minimize the same objective: our proposed multiclass formulation, Eq. (2).

Fig. 2 and Fig. 3 compare the relative objective value difference $\frac{F(\boldsymbol{W})-F(\boldsymbol{W}^*)}{F(\boldsymbol{W}^*)}$ (lower is better) and test accuracy (higher is better) of the above solvers as a function of training time, when $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$, respectively. For FOBOS, we used the step size $\eta_t = \frac{\eta_0}{\sqrt{t}}$, where we chose η_0 beforehand from $10^{-3}, 10^{-2}, \ldots, 10^{3}$ with a held-out validation set.

Fig. 4 compares the number of non-zero rows of the solution (lower is better) as a function of training time for the different solvers, when $\lambda = 10^{-3}$ (left) and $\lambda = 10^{-5}$ (right).

5.2.1 Comparison of block coordinate descent with or without line search

Fig. 2 and Fig. 3 indicate that block coordinate descent (BCD) with line search was overall slightly faster to converge than without. Empirically, we observe that the sufficient decrease condition checked by the line search, Eq. (7), is usually accepted on the first try ($\alpha_j^t = 1$). In that case, the line search does not incur much extra cost, since the objective value difference $F(\boldsymbol{W}^{t+1}) - F(\boldsymbol{W}^t)$, needed for Eq. (7), can be computed in the same loop as the partial gradient. For the few times when more than one line search step is required, our formulation has the advantage that the objective value difference can be computed very efficiently (no expensive log or exp). However, similarly to other iterative solvers, BCD (both with or without line search) may suffer from slow convergence on very loosely regularized problems (very small λ).

In terms of row sparsity, Fig. 4 shows that in all datasets, BCD had a two-phase behavior: first increasing the number of non-zero rows, then rapidly decreasing it. Compared to other solvers, BCD was always the fastest to reach the sparsity level corresponding to a given λ value.

5.2.2 Comparison with a projected stochastic subgradient descent solver: FOBOS

BCD outperformed FOBOS on smaller datasets (News20, Sector) and was comparable to FOBOS on larger datasets (MNIST, RCV1, Amazon7). However, for FOBOS, we found that tuning the initial step size η_0 was crucial to obtain good convergence speed and accuracy. This additional "degree of freedom" is a major disadvantage of FOBOS over BCD, in practice. However, since it is based on stochastic subgradient descent, FOBOS can handle non-differentiable loss functions (e.g., the Crammer-Singer multiclass loss), unlike BCD.

Fig. 4 shows that FOBOS obtained much less sparse solutions than BCD. In particular, on RCV1 with $\lambda = 10^{-3}$, BCD obtained less than 5% non-zero rows whereas FOBOS obtained almost 80%.

5.2.3 Comparison with full-gradient solvers: FISTA and SpaRSA

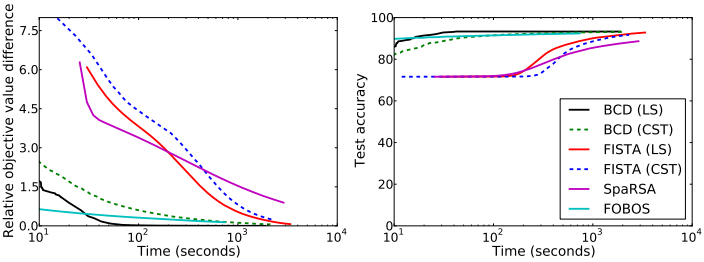
BCD outperformed FISTA and SpaRSA on all datasets, both in speed of objective value decrease and test accuracy increase. FISTA (LS) and SpaRSA achieved similar convergence speed with a slight advantage for FISTA (LS). Interestingly, FISTA (CST) was always quite worse than FISTA (LS), showing that, in the full-gradient case, doing a line search to adjust the step size at every iteration is greatly beneficial. In contrast, the difference between BCD (LS) and BCD (CST) appeared to be smaller. FISTA (CST) uses one global step size $\frac{1}{\mathcal{K}}$ whereas BCD (CST) uses a per-block step size $\frac{1}{\mathcal{K}_j}$. Therefore, BCD (CST) uses a constant step size which is more appropriate for each block.

BCD, FOBOS, FISTA and SpaRSA differ in how they make use of gradient information at each iteration. FISTA and SpaRSA use the entire gradient $G(\boldsymbol{W}) \in \mathbf{R}^{d \times m}$ averaged over all n training instances. This is expensive, especially when both n and d are large. On the other hand, FOBOS uses a stochastic approximation of the entire gradient (averaged over a single training instance) and BCD uses only the partial gradient $G(\boldsymbol{W})_{j:} \in \mathbf{R}^m$ (averaged over all training instances). FOBOS and BCD can therefore quickly start to minimize Eq. (2) and increase test accuracy, when FISTA and SpaRSA are not even done computing $G(\boldsymbol{W})$ yet. Additionally, FISTA and SpaRSA change $\boldsymbol{W}$ entirely at each iteration, which forces to recompute $G(\boldsymbol{W})$ and $F(\boldsymbol{W}^{t+1}) - F(\boldsymbol{W}^t)$ entirely. In the case of BCD, only one block $\boldsymbol{W}_{j:}$ is modified at a time, enabling the fast implementation technique described in Section 4.3.

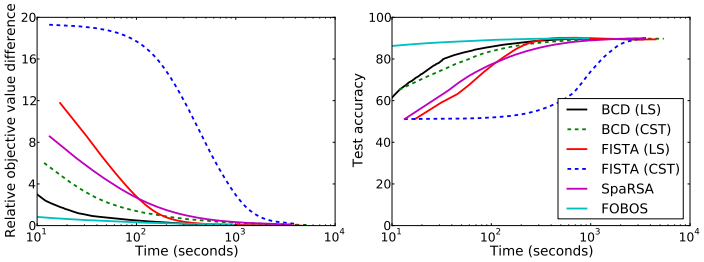
In terms of sparsity, FISTA and SpaRSA reduced the number of non-zero rows much more slowly than BCD. However, in the limit, they obtained similar row sparsity to BCD.

5.2.4 Effect of shrinking

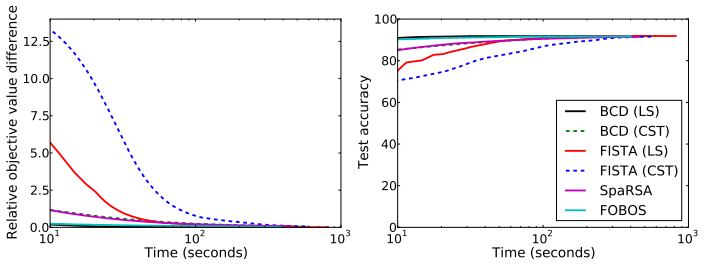
We also extended to ℓ_1/ℓ_2 regularization the shrinking method originally proposed by Yuan et al. [33] for ℓ_1-regularized binary classification. Indeed, using optimality conditions developed in Section 4.5, it is possible to discard zero blocks early if, according to the optimality conditions, they are likely to remain zero. However, we found that shrinking did not improve convergence on lower-dimensional datasets such as RCV1 and only slightly helped on higher-dimensional datasets such as Amazon7. This is in line with Yuan et al.'s experimental results on ℓ_1-regularized binary classification.



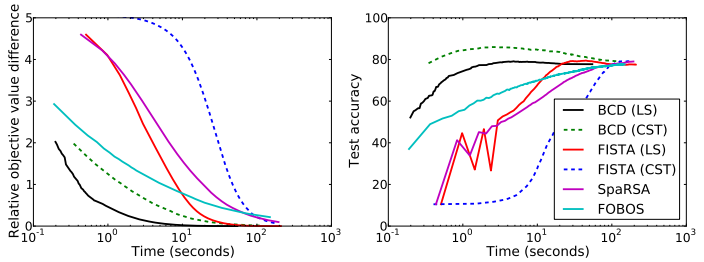
(a) Amazon7



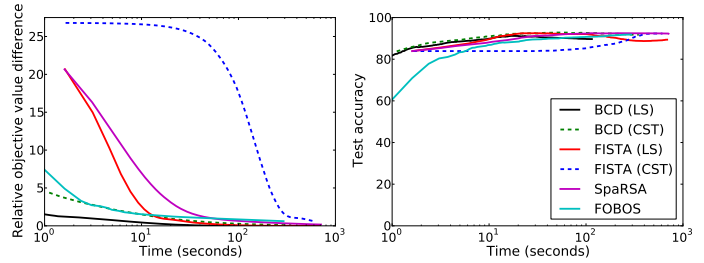
(b) RCV1



(c) MNIST

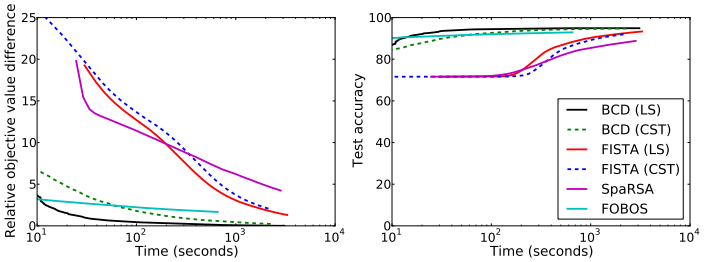


(d) News20

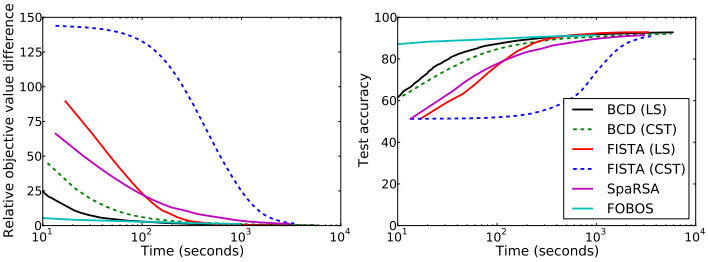


(e) Sector

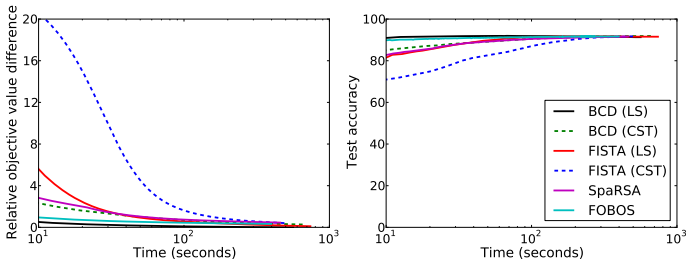
Fig. 2 Relative objective value difference (left) and test accuracy (right) as a function of training time, when $\lambda = 10^{-3}$. Time is in log-scale.



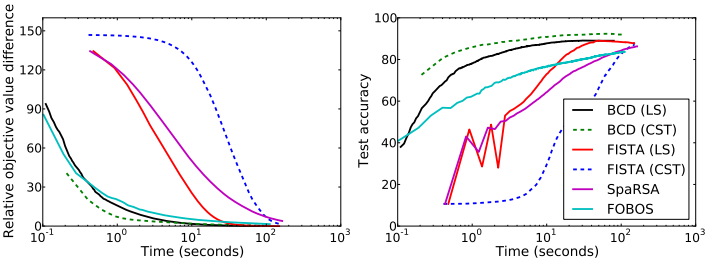
(a) Amazon7



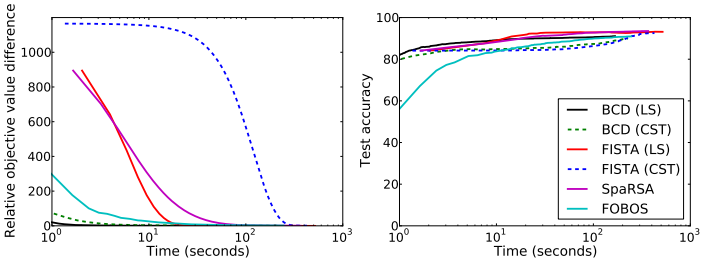
(b) RCV1



(c) MNIST

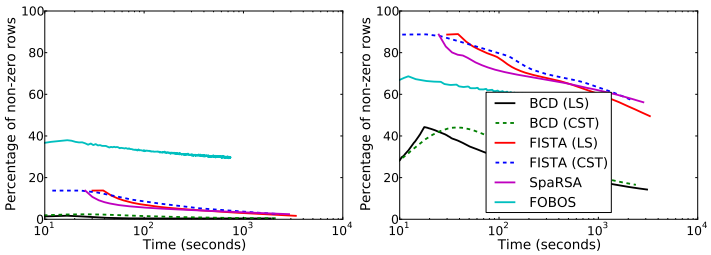


(d) News20

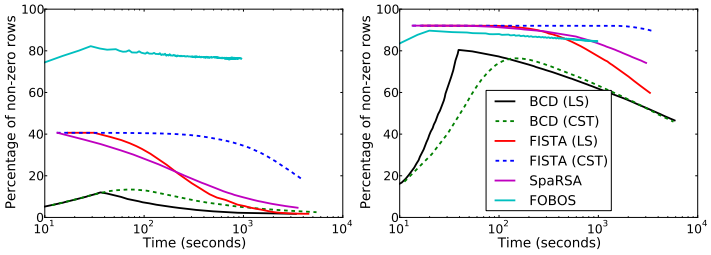


(e) Sector

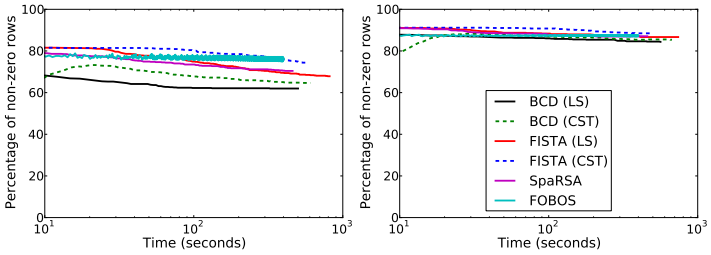
Fig. 3 Relative objective value difference (left) and test accuracy (right) as a function of training time, when $\lambda = 10^{-5}$. Time is in log-scale.



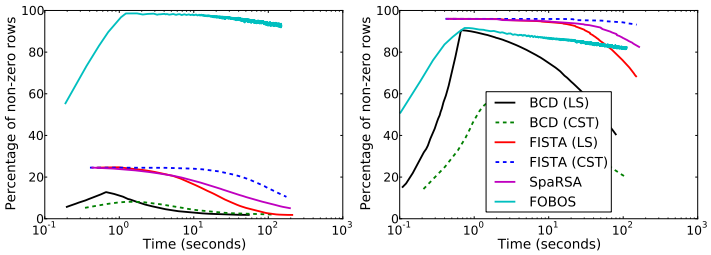
(a) Amazon7



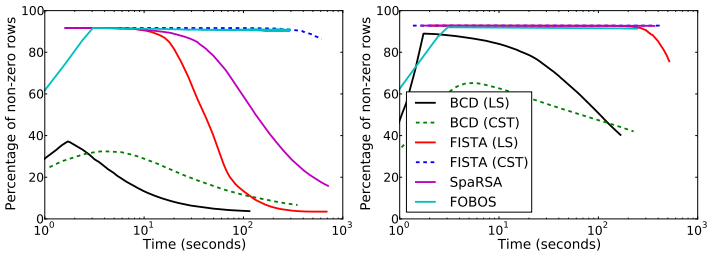
(b) RCV1



(c) MNIST



(d) News20



(e) Sector

Fig. 4 Percentage of non-zero rows as a function of training time, when $\lambda = 10^{-3}$ (left) and $\lambda = 10^{-5}$ (right). Time is in log-scale.

5.3 Comparison with other multiclass and multitask objectives

In this experiment, we used block coordinate descent to minimize and compare different ℓ_1/ℓ_2-regularized multiclass and multitask objectives:

- multiclass squared hinge (proposed, same as Eq. (2)):

$$\underset{\boldsymbol{W}\in\mathbf{R}^{d\times m}}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{n}\sum_{r\neq y_i}\max(1-(\boldsymbol{W}_{:y_i}\cdot\boldsymbol{x}_i-\boldsymbol{W}_{:r}\cdot\boldsymbol{x}_i),0)^2+\lambda\sum_{j=1}^{d}||\boldsymbol{W}_{j:}||_2.$$

- multitask squared hinge:

$$\underset{\boldsymbol{W}\in\mathbf{R}^{d\times m}}{\text{minimize}} \frac{1}{n}\sum_{r=1}^{m}\sum_{i=1}^{n}\max(1-\boldsymbol{Y}_{ir}\boldsymbol{W}_{:r}\cdot\boldsymbol{x}_i,0)^2+\lambda\sum_{j=1}^{d}||\boldsymbol{W}_{j:}||_2, \tag{11}$$

  where $\boldsymbol{Y}_{ir}=+1$ if $y_i=r$ and $\boldsymbol{Y}_{ir}=-1$ otherwise.
- multiclass logistic regression:

$$\underset{\boldsymbol{W}\in\mathbf{R}^{d\times m}}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{n}\log(1+\sum_{r\neq y_i}\exp(\boldsymbol{W}_{:r}\cdot\boldsymbol{x}_i-\boldsymbol{W}_{:y_i}\cdot\boldsymbol{x}_i))+\lambda\sum_{j=1}^{d}||\boldsymbol{W}_{j:}||_2. \tag{12}$$

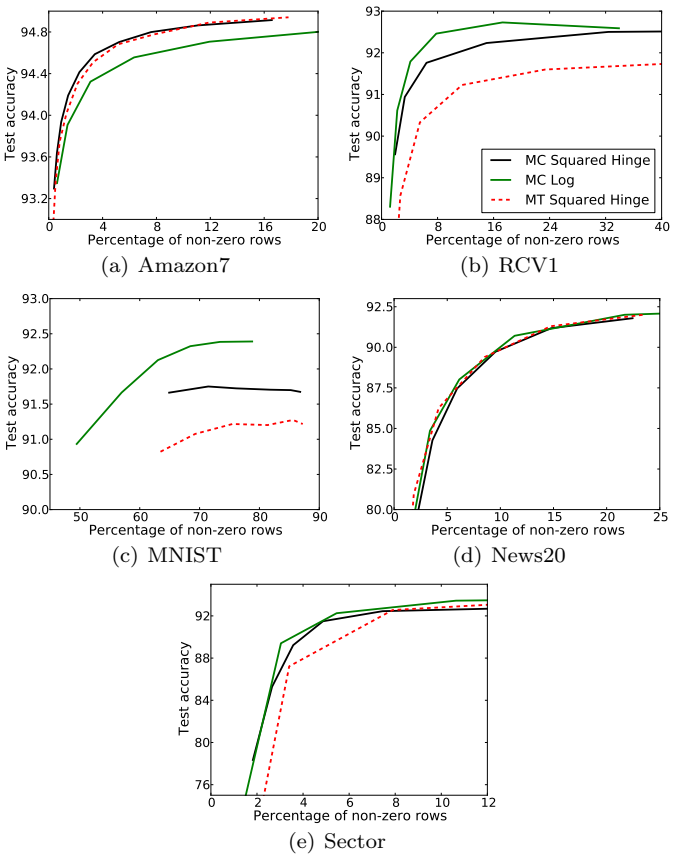
For both multitask squared hinge and multiclass logistic regression, we computed the partial gradient using an efficient implementation technique similar to the one described in Section 4.3. For the multiclass and multitask squared hinge formulations, we used BCD with line search. For the multiclass logistic regression formulation, we used BCD without line search, since we observed faster training times (see Section 5.3.1). For multiclass logistic regression, the partial gradient's Lipschitz constant is $\mathcal{K}_j=\frac{1}{2n}\sum_i \boldsymbol{x}_{ij}^2$ [10].

Fig. 5 compares the row-sparsity / test-accuracy trade-off of the above objectives. We generated 10 log-spaced values between $\lambda=10^{-2}$ and $\lambda=10^{-4}$ (Sector), and between $\lambda=10^{-3}$ and $\lambda=10^{-5}$ (other datasets). For each λ value, we computed the solution and measured the percentage of non-zero rows as well as test accuracy, so as to obtain Fig. 5. Table 2 shows the time (minimum, median and maximum) that was needed to compute the solutions of each objective. The results reported are averages obtained from 3 different train-test splits. We set $\tau=10^{-3}$ and $K=200$.

5.3.1 Comparison with multiclass logistic regression

Compared to multiclass logistic regression, Eq. (12), our objective achieved overall comparable accuracy. As indicated in Table 2, however, our objective was substantially faster to train (up to ten times in terms of median time) than multiclass logistic regression. Computationally, our objective has indeed two important advantages. First, the objective and gradient are "lazy": they iterate over instances and classes only when the score is not greater than the score assigned to the correct label by at least 1, whereas multiclass logistic regression always iterates over all n instances and $m-1$ classes. Second, they do not contain any exp or log computations, which are expensive to compute in practice (equivalent to dozens of multiplications) [34].

We also tried to use BCD with line search for optimizing the multiclass logistic regression objective. However, we found that the version without line search was



(a) Amazon7 (b) RCV1

(c) MNIST (d) News20

(e) Sector

Fig. 5 Test accuracy of multiclass and multiclass ℓ_1/ℓ_2-regularized objective functions as a function of the percentage of non-zero rows in the solution. Results were obtained by computing solutions for 10 log-spaced values between $\lambda = 10^{-2}$ and $\lambda = 10^{-4}$ (Sector), and between $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$ (other datasets).

overall faster. For example, on Amazon7, the median time with line search was 8723.28 seconds instead of 5822.83 seconds without line search. This contrasts with our results from Section 5.2 and thus suggests that a line search may not be beneficial when the objective value and gradient are expensive to compute. However, our results show that even without line search, multiclass logistic regression is much slower to train than our formulation.

5.3.2 Comparison with multitask squared hinge loss

Compared to the multitask squared hinge formulation, Eq. (11), we found that our direct multiclass formulation had overall better accuracy and training time. This multitask formulation can be thought as one-vs-rest with ℓ_1/ℓ_2 regularization. It is semantically different from our multiclass formulation, since it only attempts to correctly predict binary labels for different tasks (columns of $\boldsymbol{Y}$), not the true

Table 2 Minimum, median and maximum training times (in seconds) of different ℓ_1/ℓ_2-regularized objective functions, when computing solutions for 10 log-space values between $\lambda = 10^{-2}$ and $\lambda = 10^{-4}$ (Sector), and between $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$ (other datasets).

Dataset	MC Squared Hinge	MT Squared Hinge	MC Logistic
Amazon7	285.93	655.53	4,137.39
	486.54	909.84	5,822.83
	1,118.65	1,740.90	6,128.27
RCV1	2,962.82	5,089.91	10,413.47
	3,901.58	6,300.77	11,540.80
	4,581.33	6,556.67	15,055.57
MNIST	32.55	42.74	141.54
	55.25	79.56	260.30
	61.67	99.37	645.04
News20	50.46	71.42	172.94
	70.73	93.82	218.04
	75.08	98.15	244.62
Sector	62.01	251.57	149.09
	167.01	259.50	532.30
	191.52	273.85	743.45

multiclass labels. It is instructive to compare its gradient (without regularization term)

$$G_{MT}(\boldsymbol{W})_{j:} = -\frac{2}{n}\sum_r \sum_i \boldsymbol{Y}_{ir} \max(1 - \boldsymbol{Y}_{ir}\boldsymbol{W}_{:r} \cdot \boldsymbol{x}_i, 0)\boldsymbol{x}_{ij}\boldsymbol{e}_r \tag{13}$$

with the gradient in the case of our formulation,

$$G(\boldsymbol{W})_{j:} = -\frac{2}{n}\sum_{i=1}^{n} \sum_{r \neq y_i} \max(1 - (\boldsymbol{W}_{:y_i} \cdot \boldsymbol{x}_i - \boldsymbol{W}_{:r} \cdot \boldsymbol{x}_i), 0)[\boldsymbol{x}_{ij}\boldsymbol{e}_{y_i} - \boldsymbol{x}_{ij}\boldsymbol{e}_r]. \tag{14}$$

The main difference is that the inner sum in Eq. (13) updates only one element $G_{MT}(\boldsymbol{W})_{jr}$ of the gradient by adding $\boldsymbol{x}_{ij}$ weighted by $\boldsymbol{Y}_{ir} \max(1 - \boldsymbol{Y}_{ir}\boldsymbol{W}_{:r} \cdot \boldsymbol{x}_i, 0)$, whereas the inner sum in Eq. (14) updates two elements $G(\boldsymbol{W})_{jr}$ and $G(\boldsymbol{W})_{jy_i}$ by adding/subtracting $\boldsymbol{x}_{ij}$ weighted by $\max(A(\boldsymbol{W})_{ir}, 0) = \max(1 - (\boldsymbol{W}_{:y_i} \cdot \boldsymbol{x}_i - \boldsymbol{W}_{:r} \cdot \boldsymbol{x}_i), 0)$.

Using an efficient implementation technique similar to the one described in Section 4.3, the cost of computing Eq. (13) is $O(\hat{n}m)$ rather than $O(\hat{n}(m-1))$ for Eq. (14). We also observed that our multiclass objective typically reached the stopping criterion in fewer iterations than the multitask objective (e.g., $k = 73$ vs. $k = 108$ on the News20 dataset with $\lambda = 10^{-3}$).

6 Conclusion

In this paper, we proposed a novel direct sparse multiclass formulation, specifically designed for large-scale and high-dimensional problems. We presented two block coordinate descent variants [28,23] in a unified manner and developed the core components needed to efficiently optimize our formulation. Experimentally, we showed that block coordinate descent achieves comparable or better convergence speed than FOBOS [11], while obtaining much sparser solutions and not needing an extra hyperparameter. Furthermore, it outperformed full gradient based solvers such as FISTA [3]

and SpaRSA [32]. Compared to multiclass logistic regression, our multiclass formulation had significantly faster training times (up to ten times in terms of median time) while achieving similar test accuracy. Compared to a multitask squared hinge formulation, our formulation had overall better test accuracy and faster training times. In future work, we would like to empirically evaluate the extensions described in Section 4.7.

A Lemma 1 and its proof

Lemma 1 *L has Lipschitz continuous gradient, that is, there exists $\mathcal{K} > 0$ such that*

$$||G(\boldsymbol{W}_1) - G(\boldsymbol{W}_2)|| \leq \mathcal{K}||\boldsymbol{W}_1 - \boldsymbol{W}_2|| \ \forall \boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbf{R}^{d \times m}.$$

Proof. We rewrite $L(\boldsymbol{W})$ using a single vector [26] notation:

$$L(\boldsymbol{W}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{r \neq y_i} \max(1 - (\bar{\boldsymbol{W}} \cdot \Phi(\boldsymbol{x}_i, y_i) - \bar{\boldsymbol{W}} \cdot \Phi(\boldsymbol{x}_i, r)), 0)^2,$$

where $\bar{\boldsymbol{W}} \in \mathbf{R}^{dm}$ refers to the "flattened" version of $\boldsymbol{W}$ and $\Phi(\boldsymbol{x}, r) \in \mathbf{R}^{dm}$ is a vector which is zero everywhere except in the block corresponding to class r where it is $\boldsymbol{x}$. Using this notation, the entire "flattened" gradient can be concisely written as:

$$\bar{G}(\boldsymbol{W}) = -\frac{2}{n} \sum_{i=1}^{n} \sum_{r \neq y_i} \max(1 - \bar{\boldsymbol{W}} \cdot \Phi_{ir}, 0) \Phi_{ir},$$

where $\Phi_{ir} = \Phi(\boldsymbol{x}_i, y_i) - \Phi(\boldsymbol{x}_i, r)$. We first show that $\max(1 - \bar{\boldsymbol{W}} \cdot \Phi_{ir}, 0)\Phi_{ir}$ itself is Lipschitz, that is, there exists $\hat{\mathcal{K}}$ such that:

$$||\max(1 - \bar{\boldsymbol{W}}_1 \cdot \Phi_{ir}, 0)\Phi_{ir} - \max(1 - \bar{\boldsymbol{W}}_2 \cdot \Phi_{ir}, 0)\Phi_{ir}|| \leq \hat{\mathcal{K}}||\bar{\boldsymbol{W}}_1 - \bar{\boldsymbol{W}}_2|| \ \forall \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2 \in \mathbf{R}^{dm}.$$

We consider different cases. If both $1 - \bar{\boldsymbol{W}}_1 \cdot \Phi_{ir} < 0$ and $1 - \bar{\boldsymbol{W}}_2 \cdot \Phi_{ir} < 0$, then the left-hand side becomes zero and the inequality trivially holds for any $\hat{\mathcal{K}}$. If $1 - \bar{\boldsymbol{W}}_1 \cdot \Phi_{ir} > 0$ and $1 - \bar{\boldsymbol{W}}_2 \cdot \Phi_{ir} < 0$, then:

$$\begin{aligned} ||(1 - \bar{\boldsymbol{W}}_1 \cdot \Phi_{ir})\Phi_{ir}|| &= (1 - \bar{\boldsymbol{W}}_1 \cdot \Phi_{ir})||\Phi_{ir}|| \\ &\leq (\bar{\boldsymbol{W}}_2 \cdot \Phi_{ir} - \bar{\boldsymbol{W}}_1 \cdot \Phi_{ir})||\Phi_{ir}|| \\ &= \Phi_{ir} \cdot (\bar{\boldsymbol{W}}_2 - \bar{\boldsymbol{W}}_1)||\Phi_{ir}|| \\ &\leq ||\Phi_{ir}||^2 ||\bar{\boldsymbol{W}}_2 - \bar{\boldsymbol{W}}_1||. \end{aligned}$$

The second line uses $\bar{\boldsymbol{W}}_2 \cdot \Phi_{ir} > 1$ and the last line uses the Cauchy-Shwarz inequality. The other two cases can be handled similarly. $\max(1 - \bar{\boldsymbol{W}} \cdot \Phi_{ir}, 0)\Phi_{ir}$ is thus Lipschitz with constant $\hat{\mathcal{K}} = ||\Phi_{ir}||^2 = 2||\boldsymbol{x}_i||^2$. Finally, a sum of Lipschitz functions is Lipschitz, therefore, $G(\boldsymbol{W})$ is Lipschitz with constant $\mathcal{K} = \frac{4}{n} \sum_i \sum_{r \neq y_i} ||\boldsymbol{x}_i||^2 = \frac{4(m-1)}{n} \sum_i ||\boldsymbol{x}_i||^2$. Using the same proof technique, we can show that $G(\boldsymbol{W})_{j:}$ is Lipschitz with constant $\mathcal{K}_j = \frac{4(m-1)}{n} \sum_i \boldsymbol{x}_{ij}^2$.

B Proof of Proposition 1

From standard convex analysis, $\boldsymbol{W}$ is an optimal solution of Eq. (2) if and only if:

$$\mathbf{0} \in \partial F(\boldsymbol{W}) = \partial L(\boldsymbol{W}) + \lambda \partial R(\boldsymbol{W}) \tag{15}$$

where $\partial F(\boldsymbol{W})$ denotes the subdifferential of F. Since L is differentiable, we have $\partial L(\boldsymbol{W}) = \{G(\boldsymbol{W})\}$. Denote $\hat{R}(\boldsymbol{W}_{j:}) = ||\boldsymbol{W}_{j:}||_2$. We have $\partial R(\boldsymbol{W}) = [\partial \hat{R}(\boldsymbol{W}_{1:}), \ldots, \partial \hat{R}(\boldsymbol{W}_{d:})]$, where:

$$\partial \hat{R}(\boldsymbol{W}_{j:}) = \begin{cases} \{\boldsymbol{z} \in \boldsymbol{R}^m : \hat{R}^*(\boldsymbol{z}) \leq 1\}, & \text{if } \boldsymbol{W}_{j:} = \mathbf{0} \\ \{\boldsymbol{z} \in \mathbf{R}^m : \hat{R}^*(\boldsymbol{z}) = 1 \ \text{ and } \boldsymbol{z}^{\mathrm{T}} \boldsymbol{W}_{j:} = \hat{R}(\boldsymbol{w})\}, & \text{if } \boldsymbol{W}_{j:} \neq \mathbf{0}, \end{cases}$$

and $\hat{R}^*$ denotes the dual norm of $\hat{R}$ [1]. Since the ℓ_2 norm is dual of itself, we have:

$$\begin{cases} \partial\hat{R}(\boldsymbol{W}_{j:}) \in \{\boldsymbol{z} \in \mathbf{R}^m : ||\boldsymbol{z}||_2 \leq 1\} & \text{if } \boldsymbol{W}_{j:} = \mathbf{0} \\ \partial\hat{R}(\boldsymbol{W}_{j:}) = \frac{W_{j:}}{||W_{j:}||_2} & \text{if } \boldsymbol{W}_{j:} \neq \mathbf{0}. \end{cases}$$

Finally, applying Eq. (15), we obtain for all j:

$$\begin{cases} ||G(\boldsymbol{W})_{j:}||_2 \leq \lambda & \text{if } \boldsymbol{W}_{j:} = \mathbf{0} \\ G(\boldsymbol{W})_{j:} + \frac{\lambda \boldsymbol{W}_{j:}}{||\boldsymbol{W}_{j:}||_2} = \mathbf{0} & \text{if } \boldsymbol{W}_{j:} \neq \mathbf{0}, \end{cases}$$

which concludes the proof.

References

1. Bach, F.R., Jenatton, R., Mairal, J., Obozinski, G.: Optimization with sparsity-inducing penalties. Foundations and Trends in Machine Learning **4**(1), 1–106 (2012)
2. Bakin, S.: Adaptative regression and model selection in data mining problems. Ph.D. thesis, Australian National University (1999)
3. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences. **2**, 183–202 (2009)
4. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1999)
5. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. Proceedings of Conference on Learning Theory (COLT), pp. 144–152 (1992)
6. Chang, K.W., Hsieh, C.J., Lin, C.J.: Coordinate descent method for large-scale l2-loss linear support vector machines. Journal of Machine Learning Research pp. 1369–1398 (2008)
7. Combettes, P., Wajs, V.: Signal recovery by proximal forward-backward splitting. Multiscale Modeling & Simulation **4**, 1168–1200 (2005)
8. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research **2**, 265–292 (2002)
9. Dredze, M., Crammer, K., Pereira, F.: Confidence-weighted linear classification. In: Proceedings of International Conference on Machine Learning (ICML), pp. 264–271 (2008)
10. Duchi, J., Singer, Y.: Boosting with structural sparsity. In: Proceedings of International Conference on Machine Learning (ICML), pp. 297–304 (2009)
11. Duchi, J., Singer, Y.: Efficient online and batch learning using forward backward splitting. Journal of Machine Learning Research pp. 2899–2934 (2009)
12. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Proceedings of Neural Information Processing Systems (NIPS), pp. 681–687 (2001)
13. Fan, R.E., Lin, C.J.: A Study on Threshold Selection for Multi-label Classification. Tech. rep., National Taiwan University (2007)
14. Friedman, J., Hastie, T., Höfling, H., Tibshirani, R.: Pathwise coordinate optimization. The Annals of Applied Statistics **1**, 302–332 (2007)
15. Friedman, J., Hastie, T., Tibshirani, R.: A note on the group lasso and a sparse group lasso. Tech. Rep. arXiv:1001.0736 (2010)
16. Friedman, J.H., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software **33**, 1–22 (2010)
17. Fu, W.J.: Penalized Regressions: The Bridge versus the Lasso. Journal of Computational and Graphical Statistics **7**, 397–416 (1998)
18. Lee, Y., Lin, Y., Wahba, G.: Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. Journal of the American Statistical Association **99**, 67–81 (2004)
19. Mangasarian, O.: A finite newton method for classification. Optimization Methods and Software pp. 913–929 (2002)
20. Meier, L., Van De Geer, S., Bhlmann, P.: The group lasso for logistic regression. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **70**(1), 53–71 (2008)
21. Obozinski, G., Taskar, B., Jordan, M.I.: Joint covariate selection and joint subspace selection for multiple classification problems. Statistics and Computing **20**(2), 231–252 (2010)

22. Qin, Z., Scheinberg, K., Goldfarb, D.: Efficient Block-coordinate Descent Algorithms for the Group Lasso. Tech. rep., Columbia Uversity (2010)
23. Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Mathematical Programming pp. 1–38 (2012)
24. Richtárik, P., Takáč, M.: Parallel coordinate descent methods for big data optimization. Tech. Rep. arXiv:1212.0873 (2012)
25. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. Journal of Machine Learning Research **5**, 101–141 (2004)
26. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: primal estimated sub-gradient solver for svm. Mathematical Programming pp. 1–28 (2010)
27. Shevade, S.K., Keerthi, S.S.: A simple and efficient algorithm for gene selection using sparse logistic regression. Bioinformatics **19**(17), 2246–2253 (2003)
28. Tseng, P., Yun, S.: A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming **117**, 387–423 (2009)
29. Weinberger, K., Dasgupta, A., Langford, J., Smola, A., Attenberg, J.: Feature hashing for large scale multitask learning. In: Proceedings of International Conference on Machine Learning (ICML), pp. 1113–1120 (2009)
30. Weston, J., Watkins, C.: Support vector machines for multi-class pattern recognition. In: Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 219–224 (1999)
31. Wright, S.J.: Accelerated block-coordinate relaxation for regularized optimization. SIAM Journal on Optimization **22**, 159–186 (2012)
32. Wright, S.J., Nowak, R.D., Figueiredo, M.A.T.: Sparse reconstruction by separable approximation. Transactions on Signal Processing **57**(7), 2479–2493 (2009)
33. Yuan, G.X., Chang, K.W., Hsieh, C.J., Lin, C.J.: A comparison of optimization methods and software for large-scale l1-regularized linear classification. Journal of Machine Learning Research pp. 3183–3234 (2010)
34. Yuan, G.X., Ho, C.H., Lin, C.J.: An improved glmnet for l1-regularized logistic regression. In: Proceedings of the International Conference on Knowledge Discovery and Data mining, pp. 33–41 (2011)
35. Yuan, M., Yuan, M., Lin, Y., Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society, Series B **68**, 49–67 (2006)
36. Zhang, H.H., Liu, Y., Wu, Y., Zhu, J.: Variable selection for multicategory svm via sup-norm regularization. Electronic Journal of Statistics **2**, 149–167 (2006)
37. Zhao, P., Yu, B.: On model selection consistency of lasso. Journal of Machine Learning Research **7**, 2541–2563 (2006)
38. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society Series B **67**, 301–320 (2005)