Fast Active-set-type Algorithms for L_1-regularized Linear Regression

Jingu Kim **Haesun Park**
School of Computational Science and Engineering
College of Computing, Georgia Institute of Technology
{jingu,hpark}@cc.gatech.edu

Abstract

In this paper, we investigate new active-set-type methods for l_1-regularized linear regression that overcome some difficulties of existing active set methods. By showing a relationship between l_1-regularized linear regression and the linear complementarity problem with bounds, we present a fast active-set-type method, called *block principal pivoting*. This method accelerates computation by allowing exchanges of several variables among working sets. We further provide an improvement of this method, discuss its properties, and also explain a connection to the structure learning of Gaussian graphical models. Experimental comparisons on synthetic and real data sets show that the proposed method is significantly faster than existing active set methods and competitive against recently developed iterative methods.

1 INTRODUCTION

L_1-regularized linear regression, also known as the Lasso (Tibshirani, 1996), has been a highly successful method for various applications. By constraining the l_1-norm of the coefficient vector, this method simultaneously avoids over-fitting to training data and achieves sparsity in obtained coefficients. The sparsity has two important benefits; it improves interpretation by explicitly showing the relationship between the response and the features (Tibshirani, 1996), and it also allows computationally efficient models because only a small number of coefficients remain nonzero.

Researchers used l_1-regularization for many other learning problems including logistic regression (Lee et al., 2006), graphical model selection (Banerjee et al., 2008; Friedman et al., 2008b), principal component analysis (Zou et al., 2006), and sparse coding (Lee et al., 2007; Mairal et al., 2009). Although some researchers designed specialized algorithms for those problems, many utilized l_1-regularized linear regression as a subproblem to solve their intended sparse learning tasks. Hence, an efficient algorithm for l_1-regularized linear regression is important not only in its own right but also for those extended sparse learning problems.

Since the objective function of l_1-regularized linear regression is not differentiable, development of an efficient algorithm is not trivial. Among several approaches, one of the most influential has been the *least angle regression* (LARS) by Efron et al. (2004). In LARS, features are sequentially selected so that they remain equiangular, exploiting the fact that coefficient paths are piecewise linear with respect to the regularization parameter. Despite its ability to discover the full regularization paths, however, LARS is designed to select one feature at a time, and it can become very slow when applied to large problems. Lee et al. (2007) proposed the *feature-sign search* algorithm as a part of their study on sparse coding. This algorithm is a relaxed form of LARS so that a particular solution can be efficiently found by not following the exact coefficient path. The algorithm follows the structure of active-set methods (Nocedal and Wright, 1999) and shares the same difficulty of LARS for large problems.

Several iterative methods have been introduced to overcome the scalability issue. Recent developments in signal processing literature include an interior point method (Kim et al., 2007) and a gradient projection method (Figueiredo et al., 2007). Such methods have a particular advantage for their intended purpose, which is signal reconstruction, because they can handle very large problems. Another promising algorithm using a coordinate descent method (Friedman et al., 2008a) was recently introduced.

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9. Copyright 2010 by the authors.

In this paper, we investigate new active-set-type methods. By an active-set-type method, we mean an algorithm that directly searches for the sets of active and passive variables and computes the solution only up to numerical rounding errors. By showing a relationship between l_1-regularized linear regression and the linear complementarity problem with bounds (BLCP), we present an efficient method, called *block principal pivoting*, which overcomes the difficulty of the LARS and feature-sign search methods. Although the block principal pivoting method was proposed for linear complementarity problems (Júdice and Pires, 1994), its substantial benefit to l1-regularized linear regression has not been studied previously and is an important contribution of this paper. We further propose an improvement of this method, discuss its characteristics, and also show a connection to the structure learning of Gaussian graphical models. Experimental comparisons on both synthetic and real data sets show the proposed method significantly outperforms several existing ones.

2 ACTIVE-SET-TYPE METHODS

2.1 FORMULATIONS AND OPTIMALITY CONDITIONS

Suppose data are given as $(x_i, y_i)_{i=1}^n$ where $y_i \in \mathbb{R}$ is the response of $x_i \in \mathbb{R}^p$. We assume that y_i's are centered so that $\sum_{i=1}^n y_i = 0$. The coefficients $\beta \in \mathbb{R}^p$ are to be found by solving a minimization problem,

$$\min_{\beta \in \mathbb{R}^p} \mathcal{L}(\beta, \lambda) = \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 , \qquad (1)$$

where λ is a parameter, the i^{th} row of X is x_i^T, and the i^{th} element of y is y_i. Whereas we are concerned in Eq. (1) in this paper, an alternative formulation in which l_1-norm is used for constraints is possible (Tibshirani, 1996). The dual of Eq. (1) can be written as

$$\min_{r \in \mathbb{R}^n} \mathcal{L}(r, \lambda) = \frac{1}{2} r^T r - y^T r \text{ s.t. } \left\|X^T r\right\|_\infty \le \lambda, \quad (2)$$

using the derivation given in (Kim et al., 2007). Eq. (2) is easier to handle because the objective function is smooth and constraints are simple.

Our discussion for the derivation of new methods starts from writing down the optimality conditions for the dual problem in Eq. (2). Let r^* be the solution of Eq. (2), and let $\beta_+^*, \beta_-^* \in \mathbb{R}^p$ be corresponding Lagrange multipliers for the two inequality constraints, $X^T r \le \lambda$ and $X^T r \ge -\lambda$, respectively. Defining $\beta^* = \beta_+^* - \beta_-^*$, the Karush-Kuhn-Tucker (KKT) conditions for Eq. (2) can be written as

$$\begin{aligned} d^* = X^T r^* &= X^T y - X^T X \beta^*, && \text{(3a)} \\ -\lambda \le \; & d^* \le \lambda, && \text{(3b)} \\ -\lambda < d_i^* < \lambda &\Rightarrow \beta_i^* = 0, && \text{(3c)} \\ d_i^* = \lambda &\Rightarrow \beta_i^* \ge 0, && \text{(3d)} \\ d_i^* = -\lambda &\Rightarrow \beta_i^* \le 0. && \text{(3e)} \end{aligned}$$

Note that because the problem in Eq. (2) is convex, a solution satisfying Eqs. (3) is optimal for Eq. (2).

2.2 ACTIVE-SET METHODS AND LIMITATION

The key idea of active-set-type methods for Eqs. (1) and (2) is the following. Let $\mathcal{E}_+^*, \mathcal{E}_-^* \subseteq \{1, \cdots, p\}$ denote the active constraints of the solution r^* in Eq. (2); that is, $\mathcal{E}_+^* = \left\{i | \left(X^T r^*\right)_i = \lambda\right\}$ and $\mathcal{E}_-^* = \left\{i | \left(X^T r^*\right)_i = -\lambda\right\}$. If we know $\mathcal{E}_+^*$ and $\mathcal{E}_-^*$ in advance, then the solution β^* can be easily computed by using Eq. (3c) and solving a normal equation for Eq. (3a). The goal of active-set methods is to find the sets $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$ in some systematic way.

A standard example of active-set-type methods is the *active-set* method (Nocedal and Wright, 1999), which is a well known scheme generally applicable to quadratic programming problems. The active-set method begins with an initial feasible solution, for which a zero vector is usually used. The method maintains *working sets* $(\mathcal{E}_+, \mathcal{E}_-)$ as candidates for $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$ and iteratively exchanges a variable among $(\mathcal{E}_+, \mathcal{E}_-, \{1, \cdots, p\} - (\mathcal{E}_+ \cup \mathcal{E}_-))$ in such a way that the value of the objective function monotonically decreases. Due to the monotonic-decreasing property, the active-set method is guaranteed to finish in a finite number of steps.

In the case of l_1-regularized linear regression, the feature-sign algorithm in (Lee et al., 2007) exactly follows the structure of the active-set method. In fact, applying the standard active-set method to the dual problem in Eq. (2), the feature-sign algorithm can be directly derived although the authors did not use this derivation. LARS is a modified active-set algorithm in which the modification is elegantly designed so that the full coefficient paths can be computed.

Despite the finite termination property, the active-set methods have a major limitation: Because typically only one variable is exchanged among the working sets per iteration, they can become very slow for large problems. The number of iterations severely depends on the number of nonzero elements of the optimal solution. For large problems, an algorithm in which the number of iterations does not depend upon the problem size is needed. We now describe such an algorithm.

3 BLCP AND BLOCK PRINCIPAL PIVOTING METHODS

The optimality conditions in Eqs. (3) are indeed equivalent to the linear complementarity problem with bounds (BLCP) (Júdice and Pires, 1994). BLCP is a generalized class of problems from linear complementarity problems (LCP) (Murty, 1988), and LCP and BLCP frequently arise in quadratic programming. Among several approaches for BLCP, an efficient algorithm proposed in (Júdice and Pires, 1994) implements an intuitive way of speeding up the process of finding $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$. In this section, we summarize the algorithm and also propose its improvement.

3.1 BLOCK PRINCIPAL PIVOTING

Suppose the index set $\{1, \cdots, p\}$ is partitioned into three disjoint subsets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$; i.e., $\mathcal{F} = \mathcal{F}_+ \cup \mathcal{F}_- = \{1, \cdots, p\} - \mathcal{H}$ and $\mathcal{F}_+ \cap \mathcal{F}_- = \emptyset$. The sets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$ will be our working sets. We eventually want to make $(\mathcal{F}_+, \mathcal{F}_-)$ identical to $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$ up to exchanges of degenerate variables.[1] Let $\beta_{\mathcal{H}}$ and $d_{\mathcal{H}}$ denote the subsets of vectors β and d corresponding to index set $\mathcal{H}$. Likewise, let $X_{\mathcal{H}}$ denote a submatrix of X that consists only of the column vectors whose indices belong to $\mathcal{H}$. The subsets and submatrices for $\mathcal{F}_+, \mathcal{F}_-$, and $\mathcal{F}$ are similarly defined.

We start with an initial setting for $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$, where $\mathcal{H} = \{1, \cdots, p\}, \mathcal{F}_+ = \mathcal{F}_- = \emptyset$ is usually used. For the subsets, we assume

$$\beta_{\mathcal{H}} = 0,\ d_{\mathcal{F}_+} = \lambda,\ d_{\mathcal{F}_-} = -\lambda, \tag{4}$$

and compute the remaining elements of β and d using Eqs. (3) as follows:

$$d_{\mathcal{F}} = \left(X^T y\right)_{\mathcal{F}} - X_{\mathcal{F}}^T X_{\mathcal{F}} \beta_{\mathcal{F}}, \tag{5a}$$

$$d_{\mathcal{H}} = \left(X^T y\right)_{\mathcal{H}} - X_{\mathcal{H}}^T X_{\mathcal{F}} \beta_{\mathcal{F}}. \tag{5b}$$

Since $d_{\mathcal{F}}$ is fixed by the assumptions in Eq. (4), one can first solve for $\beta_{\mathcal{F}}$ in Eq. (5a) and substitute the result into Eq. (5b) to obtain $d_{\mathcal{H}}$.

Then, we check if the obtained values are optimal using the following conditions:

$$-\lambda \le d_{\mathcal{H}} \le \lambda,\ \beta_{\mathcal{F}_+} \ge 0,\ \beta_{\mathcal{F}_-} \le 0. \tag{6}$$

If β and d satisfy Eqs. (4)-(6), then they satisfy the optimality conditions in Eqs. (3). If β and d satisfy all these conditions, we call the pair (β, d) *feasible*; otherwise, it is called *infeasible.* If a feasible pair (β, d) is found, then it means that the current working sets $(\mathcal{F}_+, \mathcal{F}_-)$ are the same with $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$, and therefore the algorithm terminates with the solution β. Otherwise, we change the sets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$ and try the process again.

The issue is how we update the sets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$. To this end, we define *infeasible variables* to be the ones that violate at least one condition in Eq. (6), which consist of the following four cases:

$$\mathcal{G} = \cup_{k=1}^4 \mathcal{J}_k, \tag{7a}$$

$$\mathcal{J}_1 = \{i \in \mathcal{H} : d_i > \lambda\}, \tag{7b}$$

$$\mathcal{J}_2 = \{i \in \mathcal{H} : d_i < -\lambda\}, \tag{7c}$$

$$\mathcal{J}_3 = \{i \in \mathcal{F}_+ : \beta_i < 0\}, \tag{7d}$$

$$\mathcal{J}_4 = \{i \in \mathcal{F}_- : \beta_i > 0\}. \tag{7e}$$

The set $\mathcal{G}$ contains all infeasible variables. Now choose a subset $\hat{\mathcal{G}} \subseteq \mathcal{G}$ and let

$$\hat{\mathcal{J}}_1 = \hat{\mathcal{G}} \cap \mathcal{J}_1, \hat{\mathcal{J}}_2 = \hat{\mathcal{G}} \cap \mathcal{J}_2, \hat{\mathcal{J}}_3 = \hat{\mathcal{G}} \cap \mathcal{J}_3, \hat{\mathcal{J}}_4 = \hat{\mathcal{G}} \cap \mathcal{J}_4.$$

Then, the update rule is given as

$$\mathcal{H} \leftarrow (\mathcal{H} - (\hat{\mathcal{J}}_1 \cup \hat{\mathcal{J}}_2)) \cup (\hat{\mathcal{J}}_3 \cup \hat{\mathcal{J}}_4), \tag{8a}$$

$$\mathcal{F}_+ \leftarrow (\mathcal{F}_+ - \hat{\mathcal{J}}_3) \cup \hat{\mathcal{J}}_1, \tag{8b}$$

$$\mathcal{F}_- \leftarrow (\mathcal{F}_- - \hat{\mathcal{J}}_4) \cup \hat{\mathcal{J}}_2. \tag{8c}$$

The size $|\hat{\mathcal{G}}|$ represents how many variables are exchanged per iteration, and the choice of $\hat{\mathcal{G}}$ characterizes the algorithm. If $|\hat{\mathcal{G}}| > 1$, then the algorithm is called a *block* principal pivoting algorithm. If $|\hat{\mathcal{G}}| = 1$, then the algorithm is called a *single* principal pivoting algorithm. The active set method can be understood as an instance of single principal pivoting algorithms. After updating by Eqs. (8), the algorithm repeats the entire procedure until the number of infeasible variables (i.e., $|\mathcal{G}|$) becomes zero.

In order to speed up the procedure, $\hat{\mathcal{G}} = \mathcal{G}$ is used, which we call the *full exchange rule.* This exchange rule considerably improves the search procedure by reducing number of iterations. Although this property is desirable, however, using only the full exchange rule does not guarantee finite termination, and we need more refinement.

3.2 FINITE TERMINATION

In active set methods, the variable to exchange is carefully selected to reduce the objective function. However, the full exchange rule does not have this property and may lead to a cycle although it occurs rarely. To fix this problem, a backup exchange rule is used to guarantee termination in a finite number of steps.

The backup rule is to exchange the infeasible variable with the largest index:

$$\hat{\mathcal{G}} = \{i : i = \max\{j : j \in \mathcal{G}\}\}. \tag{9}$$

[1] We say β_i is degenerate if $\beta_i = 0$ and ($d_i = \lambda$ or $-\lambda$) are satisfied at the same time.

Algorithm 1 Block principal pivoting algorithm for the l_1-regularized liner regression

```
Input: $X \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$
Output: $\beta$
1: Initialize $\mathcal{H} = \{1, \cdots, p\}$, $\mathcal{F}_+ = \mathcal{F}_- = \emptyset$, $\beta = 0$,
   $d = -X^T y$, $k = K_{max}$, $t = p + 1$
2: Set Eq. (4) and compute ($\beta_\mathcal{F}$,$d_\mathcal{H}$) by Eqs. (5).
3: while $(\beta, d)$ is infeasible do
4:     Find $\mathcal{G}$ by Eqs. (7).
5:     If $|\mathcal{G}| < t$, set $t = |\mathcal{G}|$, $k = K_{max}$, and $\hat{\mathcal{G}} = \mathcal{G}$.
       If $|\mathcal{G}| \geq t$ and $k \geq 1$, set $k = k - 1$, and $\hat{\mathcal{G}} = \mathcal{G}$.
       If $|\mathcal{G}| \geq t$ and $k = 0$, set $\hat{\mathcal{G}}$ by Eq. (9).
6:     Update $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$ by Eqs. (8).
7:     Set Eq. (4) and compute ($\beta_\mathcal{F}$,$d_\mathcal{H}$) by Eqs. (5).
8: end while
```

In this case, the set $\hat{\mathcal{G}}$ contains only one variable, so it is a single principal pivoting rule. This simple rule guarantees a finite termination: Assuming that matrix $X^T X$ has full rank, the single principal pivoting rule in Eq. (9) returns the solution for Eqs. (3) in a finite number of steps (Júdice and Pires, 1992).

Combining the full exchange rule and the backup rule, the *block principal pivoting algorithm* is summarized in Algorithm 1. Because the backup rule is slower than the full exchange rule, it is used only if the full exchange rule does not work well. Variable t is used to control the number of infeasible variables, and variable k is used as a buffer on the number of the full exchange rules that may be tried. When the full exchange rule fails to decrease the number of infeasible variables within K_{max} iterations, the backup rule is used until it reduces the number of infeasible variables under the lowest value achieved so far, which is stored in t. This has to occur in a finite number of steps because the backup rule has a finite termination property. As soon as the backup rule achieves a new lowest number of infeasible variables, then we return to the full exchange rule. Since the number of infeasible variables is systematically reduced, the algorithm terminates in a finite number of steps. The choice of K_{max} has trade-offs, and $K_{max} = 3$ was shown to work well in practice (Júdice and Pires, 1994).

3.3 REDUCED BLOCK EXCHANGE

The full exchange rule is the most greedy version of exchange rules, and it can slow down if features are highly correlated. In this case, the full exchange rule tends to exchange too many variables unnecessarily and spend more iterations until termination. For large problems, this behavior can be more problematic because unnecessarily increasing the size of $\mathcal{F}$ means that we have to solve large linear equations in Eq. (5a) even though the final number of nonzero values, $\left|\mathcal{E}_+^* \cup \mathcal{E}_-^*\right|$, can be small.

Table 1: Characteristics of active-set-type methods: LARS, feature-sign (FS), and block principal pivoting (BP)

	LARS	FS	BP
variable search	equiangular	active-set	block
regularization path	✓		
monotonic decrease	✓	✓	
finiteness	✓	✓	✓
scalable			✓

To address this difficulty, we designed *reduced block exchange rule* by constraining the maximum increase of $|\mathcal{F}|$ as follows.

1. Allow full exchanges for reducing $|\mathcal{F}|$, i.e., $\hat{\mathcal{J}}_3 = \mathcal{J}_3$ and $\hat{\mathcal{J}}_4 = \mathcal{J}_4$.
2. Limit the increase of $|\mathcal{F}|$ by enforcing that $|\hat{\mathcal{J}}_1 \cup \hat{\mathcal{J}}_2| \leq \alpha p$ where $0 < \alpha < 1$ is a parameter.

The sets $\hat{\mathcal{J}}_1$ and $\hat{\mathcal{J}}_2$ can be naturally determined by sorting $\mathcal{J}_1$ and $\mathcal{J}_2$ based on the absolute values of the violation, i.e., $d_i - \lambda$ for $\mathcal{J}_1$ and $-\lambda - d_i$ for $\mathcal{J}_2$. In our experiments, $\alpha = 0.2$ generally produced good results. We modify Algorithm 1 by using this exchange rule in the first two cases of Step 5 and call the modified one *reduced block principal pivoting* method.

3.4 SUMMARY OF CHARACTERISTICS

A summary of active-set-type methods is shown in Table 1. LARS computes the entire regularization path by spending more time to compute the equiangular vector. The feature-sign algorithm maintains the monotonic-decreasing property but does not follow the regularization path. The block principal pivoting method maintains the finite termination property and becomes scalable to large problems. While LARS has benefits for computing the full regularization path, for obtaining a particular solution, block principal pivoting methods have advantage. This is the case if the parameter is estimated by prior knowledge or by theoretical analysis, or if l_1-regularized linear regression is used as a subroutine for other sparse learning tasks.

Block principal pivoting methods require that the feature matrix X has full column rank. However, since the other two methods are typically implemented using normal equations, they similarly require that features in active set have full column rank in every iteration. Furthermore, additional l_2-norm regularization such as elastic net (Zou and Hastie, 2005) can be adopted to alleviate this concern. In return, block principal pivoting methods enable significant speed-up.

In some sparse learning tasks in which l_1-regularized linear regression is used as a subroutine, the full rank assumption is always satisfied. We describe an example in the following section.

4 GAUSSIAN STRUCTURE LEARNING

For graphical models such as Markov random fields (MRFs), sparse structures often need to be learned from data. In the Gaussian case, learning structure is equivalent to identifying zero and nonzero elements of the inverse covariance matrix. We briefly describe how the proposed method can be used to accelerate the structure learning of Gaussian graphical models within the framework of (Banerjee et al., 2008).

Suppose $(x_i)_{i=1}^n$, $x_i \in \mathbb{R}^p$ are observed from multivariate Gaussian distribution, $N(\mu, \Sigma)$. Our goal is to estimate the inverse covariance matrix Σ^{-1} such that a small number of elements in Σ^{-1} are nonzero. Following Banerjee et al. (2008), a penalized maximum likelihood formulation can be written as

$$\max_Z \log\det Z - \operatorname{trace}(Z\hat{\Sigma}) - \eta \sum_{i,j=1}^{p} |Z_{ij}|, \tag{10}$$

where $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n (x_i - \hat{\mu})(x_i - \hat{\mu})^T$, $\hat{\mu} = \frac{1}{n}\sum_{i=1}^n x_i$, and $\eta > 0$ is a parameter to control the strength of penalty. The dual form for Eq. (10) is written as

$$\begin{aligned} &\min_W -\log\det(\hat{\Sigma} + W) - d \qquad (11)\\ &\text{s.t. } \hat{\Sigma} + W \succ 0, \ -\eta \le W_{ij} \le \eta, \ \forall i, j, \end{aligned}$$

where $A \succ 0$ means that A is positive definite.

Eq. (11) can be efficiently solved by block coordinate descent approach. Let $V = \hat{\Sigma} + W$, initialize V with $\hat{\Sigma} + \eta I$, and assume that $\hat{\Sigma}$ and V are partitioned as

$$V = \begin{pmatrix} V_{11} & v_{12} \\ v_{12}^T & v_{22} \end{pmatrix}, \ \hat{\Sigma} = \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{pmatrix},$$

where $v_{12}, s_{12} \in \mathbb{R}^{p-1}$ and $v_{22}, s_{22} \in \mathbb{R}$. Then, we update v_{12} (and v_{12}^T, of course) while fixing all other elements. Using Schur complement, the update problem can be simplified as

$$v_{12} = \arg\min_v v^T W_{11}^{-1} v \text{ s.t. } \|v - s_{12}\|_\infty \le \eta. \tag{12}$$

One can solve Eq. (12) for each column (and corresponding row) of V by permutations. Banerjee et al. (2008) solved Eq. (12) using a smooth optimization technique, and Friedman et al. (2008b) used a coordinate descent method for the dual of Eq. (12).

In this paper, we efficiently solve Eq. (12) by exploiting its relation to Eq. (2). As we did for Eq. (2), we can write the KKT optimality conditions for Eq. (12) as

$$\begin{aligned} k &= s_{12} - W_{11} l, && (13a)\\ -\eta \le\ & k \le \eta, && (13b)\\ -\eta < k_i < \eta &\ \Rightarrow\ l_i = 0, && (13c)\\ k_i = \eta &\ \Rightarrow\ l_i \ge 0, && (13d)\\ k_i = -\eta &\ \Rightarrow\ l_i \le 0, && (13e) \end{aligned}$$

where $v = s_{12} - k$. Hence, the proposed method presented in Section 3 can be directly used for Eqs. (13). As shown in (Banerjee et al., 2008), W_{11} remains positive definite throughout iterations, satisfying the full rank assumption.

5 EXPERIMENTAL VALIDATION

We implemented the proposed methods in MATLAB and compared with several existing ones. We first compared active-set-type methods under various conditions, and then we compared with iterative methods and tested on real data sets. We also provide results for Gaussian structure learning task. All experiments were executed on a 2.66GHz Intel Quad Core processor with Linux OS, with multi-threading option disabled. All results are the average of 10 executions.

5.1 ACTIVE-SET-TYPE METHODS

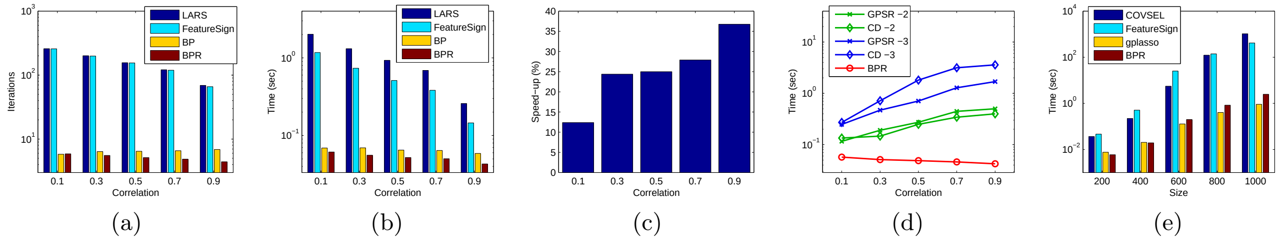
To see the behavior of active-set-type methods under various conditions, we tested them with synthetic data sets generated by a linear model: $y = X\beta + \epsilon$. We tried two types of feature matrix X, and how we generated them is described below. Each β_i was sampled from uniform distribution on $[-1, 1]$. Then, independent Gaussian noise ϵ was generated and scaled so that the average magnitude of ϵ_i is five percent of the average magnitude of $(X\beta)_i$. MATLAB implementations of the feature-sign (Lee et al., 2007) and the LARS algorithms (Sjöstrand, 2005) were used.[2]

We first tested active-set-type methods with sparse random features. We sampled each element of X from uniform distribution on $[0, 1]$ and randomly selected 70 percent of the elements to make them zero. As shown in Table 2, considerable improvements were achieved by the two new methods: the block principal pivoting (BP) and reduced block principal pivoting (BPR). The numbers of iterations required by these methods are small for various problem sizes and values of parameter λ. Accordingly, the execution time of these methods depend less severely on such variations than the LARS and feature-sign methods. One can see that the BP and BPR algorithms are up to 100 (or more) times faster than LARS.

[2] The LARS algorithm was modified to stop at the solution for given parameter λ.

Table 2: Execution results of the LARS, feature-sign (FS), block principal pivoting (BP), and reduced block principal pivoting (BPR, with $\alpha = 0.2$) methods on data sets with sparse random features (see text). The third column shows the number of nonzero elements in obtained β for the corresponding value of λ.

			# of iterations				Time (seconds)			
Problem size	λ	Nonzeros	LARS	FS	BP	BPR	LARS	FS	BP	BPR
2500 × 1000	16	70	71	71	**2**	**2**	1.07	0.367	**0.341**	**0.34**
	9.71	263	264	264	**4**	**4**	4.23	1.53	**0.357**	**0.352**
	5.89	485	486	486	**4**	**5**	12.3	6.49	**0.402**	**0.378**
	3.58	654	655	655	**4**	**8**	23.5	13.0	**0.439**	**0.459**
	2.17	769	772	770	**5**	**8**	41.0	21.2	**0.516**	**0.493**
5000 × 2000	25.9	83	84	84	**2**	**2**	5.46	**2.32**	**2.37**	**2.37**
	15.7	436	437	437	**4**	**4**	24.3	7.79	**2.41**	**2.41**
	9.56	884	887	885	**4**	**5**	91.1	45.0	**2.62**	**2.57**
	5.80	1226	1233	1228	**4**	**8**	222	118	**2.94**	**2.90**
	3.52	1493	1500	1495	**4**	**7**	462	226	**3.31**	**3.05**
10000 × 5000	35.3	269	270	270	**3**	**3**	79.7	29.6	**27.2**	**27.2**
	21.4	1134	1135	1135	**4**	**5**	394	145	**27.8**	**27.7**
	13.0	2128	2129	2129	**5**	**6**	2112	872	**31.2**	**29.6**
	7.89	3027	3030	3028	**5**	**7**	7632	2955	**36.4**	**33.1**
	4.79	3675	3696	3676	**5**	**9**	15969	5954	**42.4**	**40.8**



(a) (b) (c) (d) (e)

Figure 1: (a,b): Iteration counts and execution time of the LARS, FS, BP, and BPR methods on data sets with correlated features (see text) (c): Speed-up by BPR upon BP (d): Execution time of BPR and other iterative optimization schemes (e): Execution time for Gaussian structure learning for various sizes.

Next, we observed behaviors with correlated features. For various values of ρ, we created X by generating 1000 samples of multivariate Gaussian distribution with 500 variables where the correlation coefficient of each pair of variables is ρ. We generated ten instances of such data for each ρ, and for each instance, all algorithms were executed for several λ's obtained as follows. We computed the smallest λ that allows $\beta = 0$ as a solution, which is $\lambda_{max} = \left\| X^T y \right\|_\infty$, and then we divided the interval $(\lambda_{max}, \lambda_{max} \times 0.01)$ by six in log-scale to get the five values in the middle. Average results over ten data sets and the five parameter values are shown in Figure 1-(a,b,c). Initially, we thought that BP can much slow down with highly correlated features; however, both BP and BPR appeared faster than other methods even for the highly correlated case. As correlation increase, a smaller number of features tend to remain nonzero because all the features become more similar. This generally resulted in smaller number of iterations and execution time for the LARS, feature-sign, and BPR methods in high correlation.

BPR showed a clear advantage against BP especially for the highly correlated case. Figure 1-(a,b) show that BPR reduced both the number of iterations and the execution time of BP, and Figure 1-(c) illustrates the speed-up of BPR against BP. For high correlation, the reduced block exchange rule effectively controls the increase of $|\mathcal{F}|$ and becomes more efficient than the full exchange rule. Hence, we focused on BPR in the following experiments.

5.2 COMPARISON WITH ITERATIVE METHODS

To see relative performance against iterative optimization methods, we compared BPR with two recently proposed methods: the gradient projection method by Figueiredo et al. (2007) (GPSR) and the coordinate descent method by Friedman et al. (2008a) (CD).[3]

To compare these algorithms under the same condition, we stopped both algorithms in the following way. Suppose the solution in the k^{th} iteration is β^k, and let $d^k = X^T y - X^T X \beta^k$. Then, defining

$$g_i^k = \begin{cases} -d_i^k + \lambda & \text{if } (\beta_i^k > 0) \text{ or } (\beta_i^k = 0 \text{ and } d_i > \lambda) \\ -d_i^k - \lambda & \text{if } (\beta_i^k < 0) \text{ or } (\beta_i^k = 0 \text{ and } d_i < -\lambda) \\ 0 & \text{if } \beta_i^k = 0 \text{ and } -\lambda \le d_i \le \lambda \end{cases}$$

[3] The code of GPSR was obtained from the authors. For CD, although the authors provide an R package, we reimplemented in MATLAB/C to compare in the same environment and stopping criterion. Our code was carefully optimized by using C (i.e., MEX files) for iterative operations, which can be slow in MATLAB script.

Table 3: Execution results on data sets from UCI data repository. The results for gradient projection (GPSR) and coordinate descent (CD) methods are given for two tolerance (τ) values.

			Time (sec)							# of nonzeros (selected features)				
			Active-set-type (AS)			10^{-2}		10^{-3}		AS	10^{-2}		10^{-3}	
	size	λ	LARS	FS	BPR	GPSR	CD	GPSR	CD		GPSR	CD	GPSR	CD
Arrhy-thmia	452 ×279	0.86	0.046	0.041	**0.014**	0.028	0.015	0.043	0.023	52	54	53	53	52
		0.57	0.073	0.063	**0.022**	0.045	0.026	0.072	0.025	78	83	80	80	80
		0.16	0.224	0.144	**0.023**	0.046	0.033	0.163	0.042	140	174	155	143	140
Isolet	6238 ×617	0.74	2.61	0.75	**0.41**	4.80	0.693	36.4	0.954	116	137	117	116	116
		0.11	9.37	3.94	**0.431**	8.93	2.09	63.7	3.57	301	362	308	305	301
		0.017	24.5	13.3	**0.728**	4.49	3.76	74.2	8.73	515	604	532	529	520
Internet Ad	3279 ×1558	0.53	9.02	2.72	**1.52**	5.32	1.82	17.9	1.96	243	256	189	243	196
		0.28	17.0	6.14	**1.57**	3.79	1.86	19.3	2.54	408	448	330	424	357
		0.081	76.6	38.5	**1.85**	2.86	3.31	19.0	5.46	862	1075	755	882	771
Gisette	6000 ×5000	0.64	160	44.8	**30.4**	75.6	36.3	249	39.0	465	534	464	466	465
		0.34	448	153	**31.2**	73.5	48.5	264	58.1	1070	1330	1121	1083	1057
		0.18	1495	606	**35.0**	58.2	74.7	304	100.0	1829	2702	1964	1877	1825

and $\Delta^k = (\|g^k\|_2 / \#$ of nonzeros in $g^k)$, an algorithm was stopped if $\Delta^k \leq \tau\Delta^0$ where Δ^0 is the value using initial values and τ is a chosen tolerance. This criterion can be obtained by using the subdifferential of l_1-norm along with the criterion in (Lin and Moré, 1999). We observed that at most $\tau = 10^{-2}$ or 10^{-3} is recommended because values larger than these produced very inaccurate solutions in our repeated trials (See also Section 5.3).

Execution results of these algorithms on the correlated data set mentioned in the previous subsection are presented in Figure 1-(d).[4] The results indicate that BPR is highly competitive against these state-of-the-art iterative methods, even for a loose tolerance. Interestingly, BPR and the two iterative methods showed reverse trends with respect to the correlation coefficients: the two iterative methods became slower for highly correlated data unlike BPR.

It is worth mentioning that all the active-set-type methods, including LARS, feature-sign, and BPR, return an exact solution. In contrast, iterative methods do so only if very small tolerance is applied. The BPR method enjoys the property that it returns an exact solution without loosing scalability.

5.3 UCI DATA SETS

In Table 3, the execution results on data sets from UCI repository[5] are shown. Linear regression was considered for each data set with elastic net penalty with small l_2-norm regularization (parameter $10^{-4} \times \lambda$ when λ is given for l_1-norm). Being consistently faster than the LARS and feature-sign methods, BPR was competitive against iterative methods. For $\tau = 10^{-2}$, solutions from iterative methods were inaccurate as can be seen from the fact that the number of selected features are different from that of exact solutions obtained by active-set-type methods. When $\tau = 10^{-3}$ was applied for higher accuracy, the computation time increased. While providing exact solutions, BPR was very fast among all the methods tested.

5.4 GAUSSIAN STRUCTURE LEARNING

Figure 1-(e) and Table 4 show execution results for the structure learning of Gaussian graphical models. Parameter η was selected using the formulation in (Banerjee et al., 2008), and time was measured until the duality gap, described in the same paper, of 10^{-1} is achieved. For the results shown in Figure 1-(e), with various dimensions p, $\frac{1}{3}p$ samples of multivariate Gaussian distribution with sparse inverse covariance matrix were generated and used. BPR appeared significantly faster than the COVSEL (Banerjee et al., 2008) and feature-sign method, where the feature-sign method was used for Eq. (12). BPR showed comparable performance with graphical Lasso (gplasso) (Friedman et al., 2008b). The same trend can be observed from Table 4, which shows execution times on the gene expression data set used in (Banerjee et al., 2008).

6 RELATED WORK

In our prior work, a similar algorithm was used for solving the nonnegativity-constrained least squares (NNLS) problems (Kim and Park, 2008). The optimality conditions for the NNLS problems appear as a linear complementarity problem (LCP):

$$v = u + Mz,\ v \geq 0,\ z \geq 0,\ v^T z = 0, \qquad (14)$$

which is simpler than BLCP. A notable difference is that in NNLS, variables are exchanged between two groups (zero or positive) due to nonnegativity constraints, whereas in l_1-regularized linear regression,

[4] We report the best results from the two methods: the Barzilai-Borwein version for GPSR and the covariance updating with active-set arrangement for CD.

[5] http://archive.ics.uci.edu/ml/

Table 4: Execution time (sec) of Gaussian structure learning on Rosetta Compendium gene expression data set (6316 features and 300 samples)

COVSEL	feature-sign	BP	gplasso
53.55	225.24	4.82	4.07

variables can take any sign, and thus they are exchanged among three groups (negative, zero, or positive). Due to the difference, the exchange rules of the proposed method are more complicated, and the finite-termination proof becomes more difficult (Júdice and Pires, 1992). One might observe that l_1-regularized linear regression can be reformulated as the NNLS problems (Tibshirani, 1996; Figueiredo et al., 2007), and the algorithm in (Kim and Park, 2008) might be used. However, this approach is inefficient because the reformulation doubles the variable size. Furthermore, after reformulation, the matrix M in Eq. (14) becomes always rank deficient, making the application of (Kim and Park, 2008) infeasible.

7 DISCUSSIONS

We introduced new active-set-type algorithms for l_1-regularized linear regression and demonstrated their computational benefits through experimental comparisons. The proposed method achieved significant speed-up maintaining the accuracy of solutions.

Several questions remain to be investigated in future work. First, the performance of BP or BPR would depend on how often the full or the reduced block exchange rule fails so that the backup rule comes to play. In our extensive tests in this paper, the backup rule appearance was not observed, and this shows that these fast exchange rules in practice work well for various data sets including the highly correlated case. Still, further analysis on conditions under which the backup rule could appear and how much it affects the performance of overall algorithm will help further understanding. In addition, the current backup rule might not be the best one, and the design of alternative backup rules will be useful. Finally, there are a number of other sparse learning tasks in which applying the proposed methods can immediately provide speed-up.

Acknowledgements

This work was supported in part by the National Science Foundation grants CCF-0732318 and CCF-0808863, and the Samsung Foundation of Culture scholarship awarded to J. Kim. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4): 586–597, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. 2008a.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432, 2008b.

J. J. Júdice and F. M. Pires. Basic-set algorithm for a generalized linear complementarity problem. *Journal of Optimization Theory and Applications*, 74(3):391–411, 1992.

J. J. Júdice and F. M. Pires. A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers and Operations Research*, 21(5):587–596, 1994.

J. Kim and H. Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM)*, pages 353–362, 2008.

S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 4(1):606–617, 2007.

H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient l1 regularized logistic regression. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, pages 1–9, 2006.

C.-J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696, 2009.

K. Murty. *Linear complementarity, linear and nonlinear programming*. Heldermann Verlag, 1988.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

K. Sjöstrand. Matlab implementation of lasso, lars, the elastic net, and spca, 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?3897`. Version 2.0.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320, 2005.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.