PAPER

Stochastic Discrete First-order Algorithm for Feature Subset Selection

Kota KUDO[†], **Yuichi TAKANO**[††], ***Nonmembers*, *and* Ryo NOMURA**[†††], ***Member***

SUMMARY This paper addresses the problem of selecting a significant subset of candidate features to use for multiple linear regression. Bertsimas *et al.* [5] recently proposed the discrete first-order (DFO) algorithm to efficiently find near-optimal solutions to this problem. However, this algorithm is unable to escape from locally optimal solutions. To resolve this, we propose a stochastic discrete first-order (SDFO) algorithm for feature subset selection. In this algorithm, random perturbations are added to a sequence of candidate solutions as a means to escape from locally optimal solutions, which broadens the range of discoverable solutions. Moreover, we derive the optimal step size in the gradient-descent direction to accelerate convergence of the algorithm. We also make effective use of the L_2-regularization term to improve the predictive performance of a resultant subset regression model. The simulation results demonstrate that our algorithm substantially outperforms the original DFO algorithm. Our algorithm was superior in predictive performance to lasso and forward stepwise selection as well.

key words: *feature subset selection, optimization algorithm, linear regression, machine learning, statistics*

1. Introduction

Feature subset selection for multiple linear regression involves selecting a significant subset of candidate features available for constructing a linear regression model. Subset selection allows data collection and storage costs to be reduced and the process of estimating regression coefficients to be more efficient. It also aids in elucidating causality between selected features and the response inferred by regression models. More importantly, the predictive performance can be improved by elimination of redundant features, which are known to cause regression models to overfit noisy datasets.

The exhaustive method of finding the best subset of features is to evaluate the quality of all possible subsets of features [13]. One method of best subset selection, the mixed-integer optimization approach, was first proposed in the 1970s [2] and has recently received renewed attention [5], [6], [20], [22], [23], [30]. However, this approach is infeasible due to the heavy computational burden unless the number of candidate features is small.

A number of prior studies have focused on heuristic optimization algorithms to handle large-scale subset selection problems. The best-known algorithm of this type is stepwise selection [10], which repeats addition and elimination of one feature at a time. Various regularization techniques have been studied intensively for sparse estimation in recent years [3], [8], [11], [16], [31], [33]–[35]. Lasso [31] is the most commonly used of these methods. In lasso, the L_1-regularization term is employed to shrink unnecessary regression coefficients toward zero. Several efficient algorithms have been developed for lasso; these include least angle regression [9], coordinate descent methods [12], and alternating direction methods of multipliers [7].

Bertsimas *et al.* [5] recently proposed the discrete first-order (DFO) algorithm to efficiently find near-optimal solutions to the feature subset selection problem. This algorithm is a discrete extension of gradient-descent methods used in convex optimization, and its convergence properties have been established. However, once the DFO algorithm becomes trapped at locally optimal solutions, it cannot escape from them. For this reason, Bertsimas *et al.* [5] executed the DFO algorithm repeatedly with random initializations, and the best solution obtained from these was further refined by using optimization software.

In the current paper, a stochastic discrete first-order (SDFO) algorithm is proposed. It is an improved version of the DFO algorithm for feature subset selection. Specifically, to escape from locally optimal solutions and search for a broad range of solutions, we add random perturbations to a sequence of solutions provided by the algorithm. Similar techniques have been used effectively in gradient-descent bit-flipping algorithms for error correction [1], [29], [32]. To accelerate convergence of the algorithm, we adopt a strategy of computing the optimal step size in the gradient-descent direction. We also incorporate an L_2-regularization term [17] into our algorithm to enhance the predictive performance of the resultant subset regression model.

The efficacy of our method is assessed through simulation experiments, following the method of previous studies [5], [15]. The simulation results obtained in that way show that our algorithm decreases the out-of-sample prediction error much faster than does repeated execution of the DFO algorithm with random initializations. Moreover, our algorithm achieved better predictive performance than both the lasso and the forward stepwise selection methods.

2. Problem Formulation

Suppose that we are given n samples $(y_i; x_{i1}, x_{i2}, \dots, x_{ip})$ for $i = 1, 2, \dots, n$. Here, y_i is the value to be predicted, and x_{ij}

[†]The author is with the Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba-shi, 305–8577 Japan.

[††]The author is with the Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba-shi, 305–8577 Japan.

[†††]The author is with the Center for Data Science, Waseda University, Tokyo, 162–0042 Japan.

DOI: 10.1587/transinf.E0.D.1

is the jth feature of the ith sample. We assume, without loss of generality, that they are standardized as

$$\sum_{i=1}^{n} x_{ij} = \sum_{i=1}^{n} y_i = 0, \quad \frac{1}{n}\sum_{i=1}^{n}(x_{ij})^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i)^2 = 1$$

for all $j = 1, 2, \ldots, p$.

The multiple linear regression model is formulated as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e},$$

where

$$\boldsymbol{y} := (y_1, y_2, \ldots, y_n)^\top \in \mathbb{R}^{n\times 1},$$
$$\boldsymbol{X} := \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \in \mathbb{R}^{n\times p},$$
$$\boldsymbol{\beta} := (\beta_1, \beta_2, \ldots, \beta_p)^\top \in \mathbb{R}^{p\times 1},$$
$$\boldsymbol{e} := (e_1, e_2, \ldots, e_n)^\top \in \mathbb{R}^{n\times 1}.$$

Here, $\boldsymbol{\beta}$ is a vector of regression coefficients to be estimated, and $\boldsymbol{e}$ is a vector containing the prediction errors.

We consider selecting the subset of features with a specified size that has the minimum sum of squared errors (SSE; *i.e.*, $\|\boldsymbol{e}\|_2^2 = \sum_{i=1}^{n}(e_i)^2$). This optimization problem is posed as

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad f(\boldsymbol{\beta}) := \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \tag{1}$$
$$\text{subject to} \quad \|\boldsymbol{\beta}\|_0 \le k, \tag{2}$$

where $\|\boldsymbol{\beta}\|_0$ denotes the number of nonzero entries of $\boldsymbol{\beta}$, and k is a user-defined parameter representing the subset size, which satisfies $k < \min\{n, p\}$. This problem is of supreme importance in statistics [21] and machine learning [14]; however, obtaining an exact solution is also known to be NP-hard [25].

3. Discrete First-order Algorithm

This section contains a brief review of the DFO algorithm for feature subset selection; see Bertsimas *et al.* [5] for the full details.

Suppose that L is a Lipschitz constant for the gradient of function f, which means that

$$\|\nabla f(\boldsymbol{\eta}) - \nabla f(\boldsymbol{\beta})\|_2 \le L\|\boldsymbol{\eta} - \boldsymbol{\beta}\|_2$$

for all $\boldsymbol{\eta}, \boldsymbol{\beta} \in \mathbb{R}^{p\times 1}$. From the definition (1), we have

$$\nabla f(\boldsymbol{\beta}) = \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}^\top\boldsymbol{y}.$$

Therefore, we have $L = \lambda_{\max}(\boldsymbol{X}^\top\boldsymbol{X})$, where $\lambda_{\max}(\cdot)$ indicates the maximum eigenvalue of the matrix.

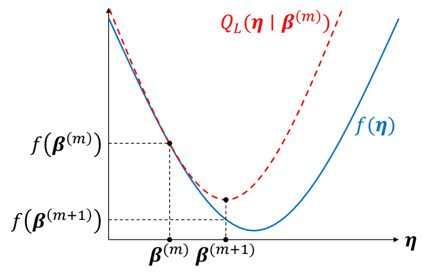

$Q_L(\boldsymbol{\eta} \mid \boldsymbol{\beta}^{(m)})$
$f(\boldsymbol{\eta})$
$f(\boldsymbol{\beta}^{(m)})$
$f(\boldsymbol{\beta}^{(m+1)})$
$\boldsymbol{\beta}^{(m)}$
$\boldsymbol{\beta}^{(m+1)}$
$\boldsymbol{\eta}$

Fig. 1 Process of the discrete first-order algorithm

As shown in Figure 1, it follows that

$$f(\boldsymbol{\eta}) \le Q_L(\boldsymbol{\eta} \mid \boldsymbol{\beta}) \tag{3}$$
$$:= f(\boldsymbol{\beta}) + \nabla f(\boldsymbol{\beta})^\top(\boldsymbol{\eta} - \boldsymbol{\beta}) + \frac{L}{2}\|\boldsymbol{\eta} - \boldsymbol{\beta}\|_2^2 \tag{4}$$

for all $\boldsymbol{\eta}, \boldsymbol{\beta} \in \mathbb{R}^{p\times 1}$. This guarantees that $Q_L(\boldsymbol{\eta} \mid \boldsymbol{\beta})$ is an upper bound of the value $f(\boldsymbol{\eta})$ to be minimized.

Now we define the operator $\mathcal{H}_k$:

$$\boldsymbol{\eta} := (\eta_1, \eta_2, \ldots, \eta_p)^\top \in \mathcal{H}_k(\boldsymbol{\beta})$$

by the following steps.

1. Sort the entries of $\boldsymbol{\beta}$ such that
$$|\beta_{\pi(1)}| \ge |\beta_{\pi(2)}| \ge \cdots \ge |\beta_{\pi(p)}|, \tag{5}$$
where π is a permutation of $\{1, 2, \ldots, p\}$.
2. Assign the k entries of $\boldsymbol{\beta}$ to $\boldsymbol{\eta}$ as
$$\eta_j := \begin{cases} \beta_j & \text{if } j \in \{\pi(1), \pi(2), \ldots, \pi(k)\}, \\ 0 & \text{otherwise.} \end{cases}$$

For example, when $\boldsymbol{\beta} = (-3, 1, 4, -2)^\top$, we obtain $\boldsymbol{\eta} = (-3, 0, 4, 0)^\top \in \mathcal{H}_k(\boldsymbol{\beta})$ for $k = 2$.

To update the solution $\boldsymbol{\beta}^{(m)} \in \mathbb{R}^{p\times 1}$ at the mth iteration, the DFO algorithm finds $\boldsymbol{\eta}$ such that $Q_L(\boldsymbol{\eta} \mid \boldsymbol{\beta}^{(m)})$ is minimized (Figure 1). This computation can be performed by using the operator $\mathcal{H}_k$ as

$$\boldsymbol{\beta}^{(m+1)} \in \underset{\boldsymbol{\eta}}{\operatorname{argmin}}\{Q_L(\boldsymbol{\eta} \mid \boldsymbol{\beta}^{(m)}) \mid \|\boldsymbol{\eta}\|_0 \le k\} \tag{6}$$
$$= \underset{\boldsymbol{\eta}}{\operatorname{argmin}}\left\{\left\|\boldsymbol{\eta} - \left(\boldsymbol{\beta}^{(m)} - \frac{1}{L}\nabla f(\boldsymbol{\beta}^{(m)})\right)\right\|_2^2 \;\middle|\; \|\boldsymbol{\eta}\|_0 \le k\right\}$$
$$= \mathcal{H}_k\left(\boldsymbol{\beta}^{(m)} - \frac{1}{L}\nabla f(\boldsymbol{\beta}^{(m)})\right). \tag{7}$$

The operation (7) moves $\boldsymbol{\beta}^{(m)}$ in the gradient-descent direction and then extracts the k most significant entries. This update is guaranteed to improve the quality of solutions in the following sense:

$$\begin{aligned} f(\boldsymbol{\beta}^{(m+1)}) &\le Q_L(\boldsymbol{\beta}^{(m+1)} \mid \boldsymbol{\beta}^{(m)}) \quad \because \text{Eq. (3)} \\ &\le Q_L(\boldsymbol{\beta}^{(m)} \mid \boldsymbol{\beta}^{(m)}) \quad \because \text{Eq. (6)} \\ &= f(\boldsymbol{\beta}^{(m)}). \quad \because \text{Eq. (4)} \end{aligned}$$

The DFO algorithm for feature subset selection is summarized as follows.

Algorithm 1 (Discrete First-order Algorithm):

Step 0: Let $\delta > 0$ be a threshold for convergence. Set $\boldsymbol{\beta}^{(1)} \in \mathbb{R}^{p\times 1}$ such that $\|\boldsymbol{\beta}^{(1)}\|_0 \le k$, and $m \leftarrow 1$.

Step 1: Compute

$$\boldsymbol{\beta}^{(m+1)} \in \mathcal{H}_k\Big(\boldsymbol{\beta}^{(m)} - \frac{1}{L}\nabla f(\boldsymbol{\beta}^{(m)})\Big).$$

Step 2: If $|f(\boldsymbol{\beta}^{(m)}) - f(\boldsymbol{\beta}^{(m+1)})| \le \delta$, then terminate the algorithm and output the solution $\boldsymbol{\beta}^{(m+1)}$.

Step 3: Set $m \leftarrow m+1$ and return to Step 1.

4. Stochastic Discrete First-order Algorithm

This section presents our SDFO algorithm for feature subset selection.

4.1 Random Perturbations

We begin by sampling a vector $\boldsymbol{\xi} := (\xi_1, \xi_2, \dots, \xi_p)^\top$ of random perturbations, where each entry follows an independent normal distribution $\mathrm{N}(0, \sigma^2)$ with standard deviation σ. We next define a new operator $\mathcal{G}_k$:

$$\boldsymbol{\eta} \in \mathcal{G}_k(\boldsymbol{\beta} \mid \boldsymbol{\xi})$$

by the following steps.

1. Sort the entries of $\boldsymbol{\beta} + \boldsymbol{\xi}$ such that

$$\begin{aligned}|\beta_{\pi(1)} + \xi_{\pi(1)}| &\ge |\beta_{\pi(2)} + \xi_{\pi(2)}| \\ &\ge \cdots \ge |\beta_{\pi(p)} + \xi_{\pi(p)}|.\end{aligned}$$

2. Assign the k entries of $\boldsymbol{\beta}$ to $\boldsymbol{\eta}$ as

$$\eta_j := \begin{cases}\beta_j & \text{if } j \in \{\pi(1), \pi(2), \dots, \pi(k)\}, \\ 0 & \text{otherwise.}\end{cases}$$

This operation partially randomizes a subset of features and makes it possible to escape from locally optimal solutions.

Our SDFO algorithm for feature subset selection is summarized as follows. Note that since random perturbations may deteriorate the quality of solutions, we retain $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{p\times 1}$ as a known solution.

Algorithm 2 (Stochastic Discrete First-order Algorithm):

Step 0: Let T be an upper limit on the number of iterations. Set $\boldsymbol{\beta}^{(1)} = \hat{\boldsymbol{\beta}} \in \mathbb{R}^{p\times 1}$ such that $\|\boldsymbol{\beta}^{(1)}\|_0 \le k$, and $m \leftarrow 1$.

Step 1: Set σ_m and sample a vector $\boldsymbol{\xi}^{(m)} \sim \mathrm{N}(\boldsymbol{0}, \sigma_m^2 \boldsymbol{I})$.

Step 2: Compute

$$\boldsymbol{\beta}^{(m+1)} \in \mathcal{G}_k\Big(\boldsymbol{\beta}^{(m)} - \frac{1}{L}\nabla f(\boldsymbol{\beta}^{(m)}) \,\Big|\, \boldsymbol{\xi}^{(m)}\Big).$$

Step 3: If $f(\boldsymbol{\beta}^{(m+1)}) < f(\hat{\boldsymbol{\beta}})$, then $\hat{\boldsymbol{\beta}} \leftarrow \boldsymbol{\beta}^{(m+1)}$.

Step 4: If $m \ge T$, then terminate the algorithm and output the solution $\hat{\boldsymbol{\beta}}$.

Step 5: Set $m \leftarrow m+1$ and return to Step 1.

4.2 Analysis of Random Perturbations

We conducted a simple analysis to determine an appropriate value of σ_m to be used in Step 1 (Algorithm 2). For analytical simplicity, it is assumed here that $\nabla f(\boldsymbol{\beta}) \approx \boldsymbol{0}$ when $\boldsymbol{\beta}$ is close to a locally optimal solution; this will occur when only a few features are related to the response.

Suppose that $c := \beta_{\pi(k)}$ according to Eq. (5). Then, if $|\xi_1| > |c + \xi_2|$ holds with $\xi_1, \xi_2 \sim \mathrm{N}(0, \sigma^2)$, the $\pi(k)$th feature will be replaced by a feature not already included. Accordingly, the probability of the $\pi(k)$th feature being replaced by some unincluded feature is given by

$$P(c, p, k) := 1 - \Big(1 - \Pr(|\xi_1| > |c + \xi_2|)\Big)^{p-k},$$

$$\Pr(|\xi_1| > |c + \xi_2|) = 2\int_0^\infty \Big(\int_{-\xi_1 - c}^{\xi_1 - c} \phi(\xi_1)\phi(\xi_2)\,\mathrm{d}\xi_2\Big)\mathrm{d}\xi_1,$$

where $\phi(\,\cdot\,)$ is the probability density function of the normal distribution $\mathrm{N}(0, \sigma^2)$.

As shown in Table 1, we calculated this replacement probability numerically with $\sigma = \gamma c$, where γ is a user-defined parameter. In this case, the value of $P(c, p, k)$ does not depend on c. When $\gamma = 0.15$, selected features are rarely replaced. When $\gamma = 0.25$, the $\pi(k)$th feature is frequently replaced, so the sequence of solutions will not converge to a locally optimal solution. For this reason, we set

$$\sigma_m = 0.2 \cdot \beta_{\pi(k)}^{(m)} \tag{8}$$

as a compromise value in Step 1 (Algorithm 2) in our simulation experiments.

Table 1 Replacement probability $P(c, p, k)$ with $\sigma = \gamma c$

		γ		
p	k	0.15	0.20	0.25
100	10	0.0002	0.0360	0.3436
500	10	0.0012	0.1808	0.8989

4.3 Optimal Step Size

To accelerate convergence of the algorithm, we adopt a strategy of computing the optimal step size in the gradient-descent direction.

Specifically, we use the following step size, which minimizes the SSE (1) at the mth iteration:

$$\alpha_m \in \operatorname*{argmin}_{\alpha} f\Big(\boldsymbol{\beta}^{(m)} - \alpha\nabla f(\boldsymbol{\beta}^{(m)})\Big). \tag{9}$$

A closed-form solution can be obtained for this step size.

Theorem 1: The optimal step size (9) is given by

$$\alpha_m = \frac{\nabla f(\boldsymbol{\beta}^{(m)})^\top \nabla f(\boldsymbol{\beta}^{(m)})}{\nabla f(\boldsymbol{\beta}^{(m)})^\top \boldsymbol{X}^\top \boldsymbol{X} \nabla f(\boldsymbol{\beta}^{(m)})}. \tag{10}$$

(*Proof*) The first-order optimality condition for Eq. (9) is written as

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}\alpha} f\left(\boldsymbol{\beta}^{(m)} - \alpha \nabla f(\boldsymbol{\beta}^{(m)})\right) \\ &= -\nabla f(\boldsymbol{\beta}^{(m)})^\top \nabla f(\boldsymbol{\beta}^{(m)}) \\ &\quad + \nabla f(\boldsymbol{\beta}^{(m)})^\top \boldsymbol{X}^\top \boldsymbol{X} \nabla f(\boldsymbol{\beta}^{(m)}) \alpha = 0,\end{aligned}$$

which provides the solution (10). □

Consequently, we can rewrite Step 2 (Algorithm 2) as follows.

Step 2 Compute

$$\boldsymbol{\beta}^{(m+1)} \in \mathcal{G}_k\left(\boldsymbol{\beta}^{(m)} - \alpha_m \nabla f(\boldsymbol{\beta}^{(m)}) \,\middle|\, \boldsymbol{\xi}^{(m)}\right),$$

where α_m is given by Eq. (10).

4.4 L_2-Regularization

To enhance the predictive performance of the subset regression model, we use an L_2-regularization term [17] in our algorithm. This amounts to solving the following optimization problem:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad f_\lambda(\boldsymbol{\beta}) := \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2 \tag{11}$$

$$\text{subject to} \quad \|\boldsymbol{\beta}\|_0 \le k, \tag{12}$$

where $\lambda \ge 0$ is a regularization parameter.

Our algorithm can be readily applied to this problem because the objective function (11) is differentiable:

$$\nabla f_\lambda(\boldsymbol{\beta}) = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})\boldsymbol{\beta} - \boldsymbol{X}^\top \boldsymbol{y}.$$

Note also that the Lipschitz constant is revised to $L = \lambda_{\max}(\boldsymbol{X}^\top \boldsymbol{X}) + \lambda$.

5. Simulation Experiments

This section describes the effectiveness of our method as evaluated by simulation experiments.

5.1 Synthetic Datasets

We considered a low-dimensional setting ($n = 500$ and $p = 100$) and a high-dimensional setting ($n = 100$ and $p = 500$), where n and p are the numbers of samples and candidate features, respectively.

Following previous studies [5], [15], we generated synthetic datasets according to the following steps.

1. First, we defined a vector of true coefficients having s nonzero entries, equally spaced, as
$$\boldsymbol{\beta}^* := (1, \boldsymbol{0}^\top, 1, \boldsymbol{0}^\top, \ldots, 1, \boldsymbol{0}^\top, 1)^\top \in \{0,1\}^{p \times 1},$$
$$s := \|\boldsymbol{\beta}^*\|_0.$$
2. Next, we obtained each row vector $\boldsymbol{x}^\top \in \mathbb{R}^{1 \times p}$ in the matrix $\boldsymbol{X}$ from a normal distribution $\boldsymbol{x} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ is the covariance matrix having $\rho^{|i-j|}$ as the (i, j)th entry.
3. Finally, we generated each entry of the response vector $\boldsymbol{y}$ as
$$y := (\boldsymbol{\beta}^*)^\top \boldsymbol{x} + \varepsilon,$$
with the error term obtained from a normal distribution $\varepsilon \sim \mathrm{N}(0, \tau^2)$.

Here, s is the true subset size, ρ is the strength of correlations between features, and τ is the standard deviation of the error term. We tested $s = 10$ and $\rho \in \{0.35, 0.70\}$, following Hastie *et al.* [15].

The *signal-to-nose ratio* (SNR) is defined as the signal variance divided by the noise variance:

$$\nu := \frac{\mathrm{Var}((\boldsymbol{\beta}^*)^\top \boldsymbol{x})}{\mathrm{Var}(\varepsilon)} = \frac{(\boldsymbol{\beta}^*)^\top \boldsymbol{\Sigma} \boldsymbol{\beta}^*}{\tau^2}. \tag{13}$$

Note that when the SNR (*i.e.,* ν) is large, the error variance (*i.e.,* τ^2) should be small, making it easy to accurately estimate $\boldsymbol{\beta}^*$.

5.2 Evaluation Metrics

Suppose that $\hat{\boldsymbol{\beta}}$ was provided by our algorithm. From this, we compute the *relative test error* [15], which characterizes the expected (out-of-sample) prediction error.

$$\begin{aligned}\textbf{Relative Test Error} &:= \frac{\mathbb{E}[(y - \hat{\boldsymbol{\beta}}^\top \boldsymbol{x})^2]}{\mathrm{Var}(\varepsilon)} \\ &= \frac{(\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}})^\top \boldsymbol{\Sigma} (\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}) + \tau^2}{\tau^2}\end{aligned}$$

Here, a perfect score is 1 (occurring when $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^*$), and the null score is $\nu + 1$ (occurring when $\hat{\boldsymbol{\beta}} = \boldsymbol{0}$).

Note that $\hat{S} := \{j \mid \hat{\beta}_j \neq 0\}$ and $S^* := \{j \mid \beta_j^* \neq 0\}$ are the index sets of the selected and true features, respectively. To evaluate the accuracy of the subset selection, we used the *F1 score* [26], which is the harmonic average of Recall $:= |\hat{S} \cap S^*|/|S^*|$ and Precision $:= |\hat{S} \cap S^*|/|\hat{S}|$; that is,

$$\textbf{F1 Score} := \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}.$$

We also examined the number of selected features,

$$\textbf{Number of nonzeros} := \|\hat{\boldsymbol{\beta}}\|_0 = |\hat{S}|.$$

The results were averaged over ten trials.

5.3 Convergence Performance

To confirm the superiority of our method over the original

DFO algorithm, we compared the convergence performance of the following methods:

DFO: the discrete first-order algorithm [5] (Algorithm 1),
DFO+OS: Algorithm 1 using the optimal step size (10),
SDFO: our stochastic discrete first-order algorithm (Algorithm 2), and
SDFO+OS: Algorithm 2 using the optimal step size (10).

All these algorithms were implemented in the Python programming language. The subset size was set as $k = s = 10$. The initial solution $\boldsymbol{\beta}^{(1)} = \mathbf{0}$ was given to the DFO algorithms; after that, the DFO algorithms were repeatedly executed by generating $\boldsymbol{\beta}^{(1)} \in \mathcal{H}_k(\boldsymbol{\xi})$ randomly with $\boldsymbol{\xi} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{I})$, where the threshold for convergence was set to $\delta = 10^{-4}$, following Bertsimas *et al.* [5]. The initial solution $\boldsymbol{\beta}^{(1)} = \mathbf{0}$ was also given to our SDFO algorithms with the standard deviation (8). All computation was performed on an Apple MacBook Air computer with an Intel Core i5-5250U CPU (1.60 GHz) and 4 GB of memory.

Figure 2 shows the results for the low-dimensional setting (*i.e.,* $n = 500$ and $p = 100$). It is clear that the performance of DFO was the worst, with the relative test error provided by DFO decreasing very slowly in many cases. Including random perturbations and using the optimal step size were both effective, as indicated by DFO+OS and SDFO performing better than DFO alone. Moreover, SDFO+OS achieved the best performance of the four algorithms. However, these algorithms performed quite similarly in the case shown in Figure 2c, where the features were weakly correlated ($\rho = 0.35$) and the SNR was very high ($\nu = 4.00$). In that case, all the algorithms found the best possible solution very quickly.

Figure 3 shows the results for the high-dimensional setting (*i.e.,* $n = 100$ and $p = 500$). In this case, because the number of samples is smaller than the number of candidate features, it is very difficult to find a good solution. Indeed, none of the algorithms reduced the relative test error when the SNR was 0.25 and 1.00. Also in this case, our SDFO algorithm (both with and without optimal step size) increased the relative test error very rapidly because of its faster convergence. However, when the SNR was 4.00, our SDFO algorithms substantially outperformed the DFO algorithms.

5.4 Predictive Performance

To demonstrate the advantage of our method over the commonly used algorithms for feature subset selection, we compared the predictive performance among the following methods:

Lasso: lasso (*i.e.,* L_1-regularized regression) [31];
Forward stepwise: forward stepwise selection [10];
SDFO+OS: Algorithm 2 using the optimal step size (10); and
SDFO+OS+L2: Algorithm 2 using the optimal step size (10) and the L_2-regularization term (11).

The lasso method was performed using the `glmnet` package [12] for the R programming language, where a sequence of 100 values of the regularization parameter λ was produced by the default configuration. Forward stepwise selection was performed using the `step` function of R, where the Akaike information criterion was used for selection. As in Section 5.3, our SDFO algorithms were implemented in the Python programming language, and the regularization parameter λ was set to each element of $\{0, 1, 10, 100, 1000\}$. Each of the SDFO computations was terminated after 0.05 s for the low-dimensional setting and after 0.10 s for the high-dimensional setting.

The regularization parameter λ and the subset size $k \in \{1, 2, \ldots, 30\}$ were tuned through hold-out validation. Following Hastie *et al.* [15], we chose the values for which the SSE (1) of the external validation set of size n was minimized. As before, all computation was performed on an Apple MacBook Air computer with an Intel Core i5-5250U CPU (1.60 GHz) and 4 GB memory.

Figure 4 shows the results for the low-dimensional setting (*i.e.,* $n = 500$ and $p = 100$). The lasso method provided the smallest relative test error when the SNR was very small, but its performance was the worst for larger SNR values. The main reason for this is that the lasso method selected too many features (Figures 4c and 4f). Forward stepwise selection and SDFO+OS delivered similar performances, whereas SDFO+OS+L2 outperformed both, especially when the SNR was small. This is because SDFO+OS+L2 tends to avoid overfitting because it includes the L_2-regularization term.

Figure 5 shows the results for the high-dimensional setting (*i.e.,* $n = 100$ and $p = 500$). In this case, SDFO+OS+L2 outperformed the other methods, particularly when the SNR was large. Note also that the lasso method performed better in the high-dimensional setting (Figure 5) than in the low-dimensional setting (Figure 4). Because of the regularization terms, both SDFO+OS+L2 and the lasso method can achieve high predictive performance in the high-dimensional setting.

6. Conclusion

We proposed the SDFO algorithm for selecting a subset of features to use for multiple linear regression. In contrast with the original DFO algorithm, our SDFO algorithm can escape from locally optimal solutions by adding random perturbations to the sequence of candidate solutions provided by the algorithm. We also derived the optimal step size in the gradient-descent direction to accelerate convergence of the algorithm. Separately, we incorporated an L_2-regularization term into the algorithm to improve the predictive performance.

The simulation results confirmed that our algorithm reduced the out-of-sample prediction error much faster than did repeated execution of the DFO algorithm with random initializations. Moreover, our algorithm delivered better accuracy in terms of both subset selection and prediction than did the lasso and forward stepwise selection methods. In addition to these successful results, it is notable that our algorithm is very simple to understand and easy to implement.

These properties seem likely to make it attractive to both academic researchers and corporate data analysts.

A future direction of study will be to further improve the efficiency of the algorithm by means of sophisticated implementation techniques. Another direction of future research is to conduct theoretical analyses of the algorithm's performance. We are now working on extending our algorithm to mixed-integer optimization models for ordinal regression and classification [4], [18], [19], [24], [27], [28].

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP17K12983.

References

[1] Al Rasheed, O., Ivaniš, P., & Vasić, B. (2014). Fault-tolerant probabilistic gradient-descent bit flipping decoder. IEEE Communications Letters, 18(9), 1487–1490.

[2] Arthanari, T.S., & Dodge, Y. (1981). Mathematical programming in statistics. Wiley, New York.

[3] Bach, F., Jenatton, R., Mairal, J., & Obozinski, G. (2012). Optimization with sparsity-inducing penalties. Foundations and Trends® in Machine Learning, 4(1), 1–106.

[4] Bertsimas, D., & King, A. (2017). Logistic regression: From art to science. Statistical Science, 32(3), 367–384.

[5] Bertsimas, D., King, A., & Mazumder, R. (2016). Best subset selection via a modern optimization lens. The Annals of Statistics, 44(2), 813–852.

[6] Bertsimas, D., Pauphilet, J., & Van Parys, B. (2019). Sparse regression: Scalable algorithms and empirical performance. arXiv preprint arXiv:1902.06547. `https://arxiv.org/abs/1902.06547`

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

[8] Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American Statistical Association, 96(456), 1348–1360.

[9] Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. The Annals of Statistics, 32(2), 407–499.

[10] Efroymson, M.A. (1960). Multiple regression analysis. In: Ralston, A., Wilf, H.S. (eds.) Mathematical Methods for Digital Computers, pp. 191–203. Wiley, New York.

[11] Frank, L. E., & Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. Technometrics, 35(2), 109–135.

[12] Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software, 33(1), 1–22.

[13] Furnival, G.M., & Wilson, R.W. (1974). Regressions by leaps and bounds. Technometrics, 16(4), 499–511.

[14] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3(Mar), 1157–1182.

[15] Hastie, T., Tibshirani, R., & Tibshirani, R. J. (2017). Extended comparisons of best subset selection, forward stepwise selection, and the lasso. arXiv preprint arXiv:1707.08692. `https://arxiv.org/abs/1707.08692`

[16] Hastie, T., Tibshirani, R., & Wainwright, M. (2015). Statistical learning with sparsity: the lasso and generalizations. Chapman and Hall/CRC.

[17] Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55–67.

[18] Kamiya, S., Miyashiro, R., & Takano, Y. (2019). Feature subset selection for the multinomial logit model via mixed-integer optimization. In the 22nd International Conference on Artificial Intelligence and Statistics (pp. 1254–1263).

[19] Kimura, K. (2019). Application of a mixed integer nonlinear programming approach to variable selection in logistic regression. Journal of the Operations Research Society of Japan, 62(1), 15–36.

[20] Konno, H., & Yamamoto, R. (2009). Choosing the best set of variables in regression analysis using integer programming. Journal of Global Optimization, 44(2), 273–282.

[21] Miller, A. (2002). Subset selection in regression. Chapman and Hall/CRC.

[22] Miyashiro, R., & Takano, Y. (2015). Subset selection by Mallows' C_p: A mixed integer programming approach. Expert Systems with Applications, 42(1), 325–331.

[23] Miyashiro, R., & Takano, Y. (2015). Mixed integer second-order cone programming formulations for variable selection in linear regression. European Journal of Operational Research, 247(3), 721–731.

[24] Naganuma, M., Takano, Y., & Miyashiro, R. (2019). Feature subset selection for ordered logit model via tangent-plane-Based approximation. IEICE Transactions on Information and Systems, 102(5), 1046–1053.

[25] Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. SIAM Journal on Computing, 24(2), 227–234.

[26] van Rijsbergen, C. J. (1979). Information Retrieval, 2nd Edition. Butterworth-Heinemann, Oxford.

[27] Sato, T., Takano, Y., & Miyashiro, R. (2017). Piecewise-linear approximation for feature subset selection in a sequential logit model. Journal of the Operations Research Society of Japan, 60(1), 1–14.

[28] Sato, T., Takano, Y., Miyashiro, R., & Yoshise, A. (2016). Feature subset selection for logistic regression via mixed integer optimization. Computational Optimization and Applications, 64(3), 865–880.

[29] Sundararajan, G., Winstead, C., & Boutillon, E. (2014). Noisy gradient descent bit-flip decoding for LDPC codes. IEEE Transactions on Communications, 62(10), 3385–3400.

[30] Takano, Y., & Miyashiro, R. (2019). Best subset selection via cross-validation criterion. Optimization Online. `http://www.optimization-online.org/DB_HTML/2019/01/7028.html`

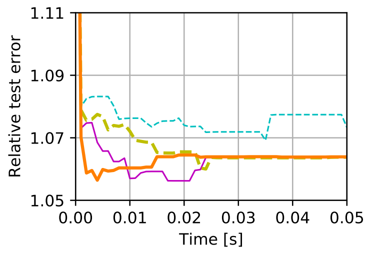
[31] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267–288.

[32] Winstead, C., Tithi, T., Boutillon, E., & Ghaffari, F. (2018). Recent advances on stochastic and noise enhanced methods in error correction decoders. In 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC) (pp. 1–5). IEEE.

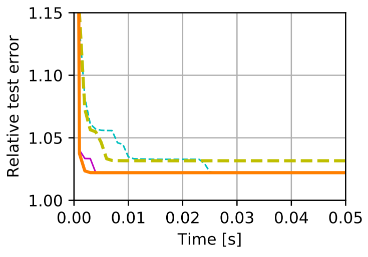
[33] Zhang, C. H. (2010). Nearly unbiased variable selection under minimax concave penalty. The Annals of Statistics, 38(2), 894–942.

[34] Zou, H. (2006). The adaptive lasso and its oracle properties. Journal of the American Statistical Association, 101(476), 1418–1429.

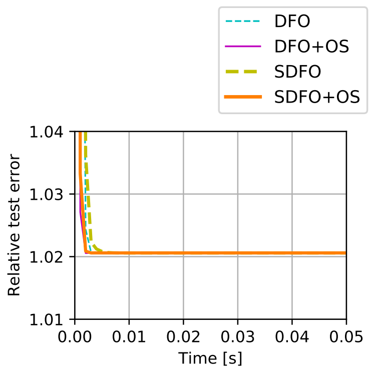
[35] Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301–320.



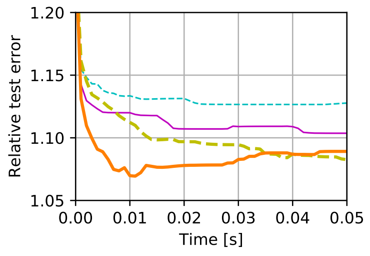
(a) $\rho = 0.35, \nu = 0.25$



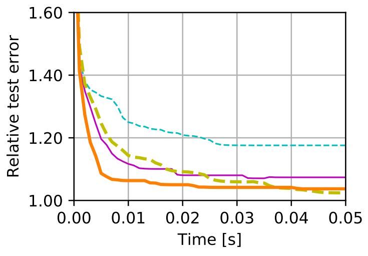
(b) $\rho = 0.35, \nu = 1.00$



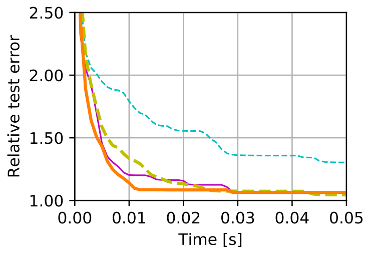
(c) $\rho = 0.35, \nu = 4.00$



(d) $\rho = 0.70, \nu = 0.25$

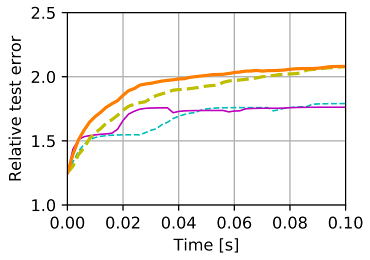


(e) $\rho = 0.70, \nu = 1.00$

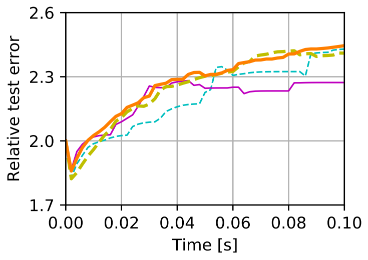


(f) $\rho = 0.70, \nu = 4.00$

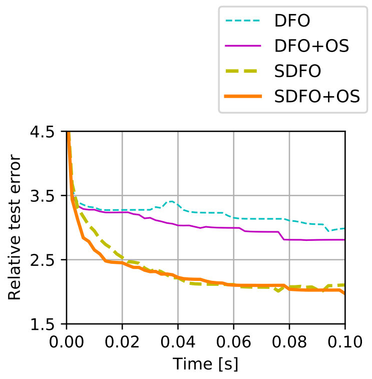
Fig. 2 Relative test error versus computation time [s]: $n = 500$, $p = 100$, and $s = 10$



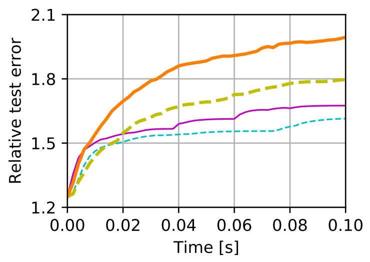
(a) $\rho = 0.35, \nu = 0.25$



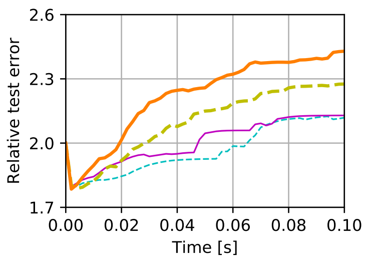
(b) $\rho = 0.35, \nu = 1.00$



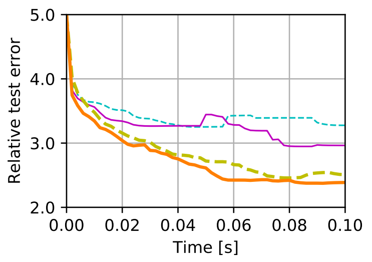
(c) $\rho = 0.35, \nu = 4.00$



(d) $\rho = 0.70, \nu = 0.25$

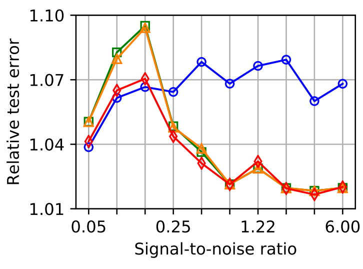


(e) $\rho = 0.70, \nu = 1.00$

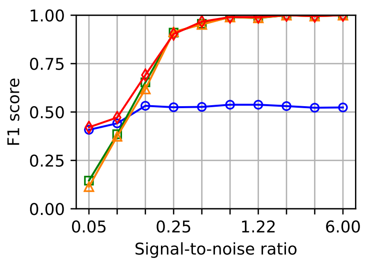


(f) $\rho = 0.70, \nu = 4.00$

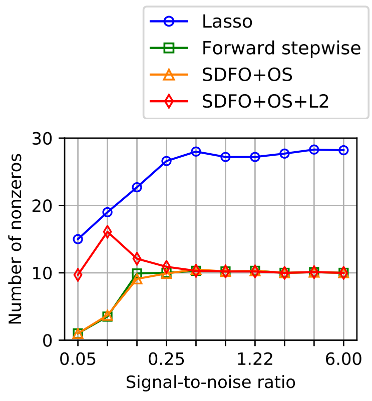
Fig. 3 Relative test error versus computation time [s]: $n = 100$, $p = 500$, and $s = 10$



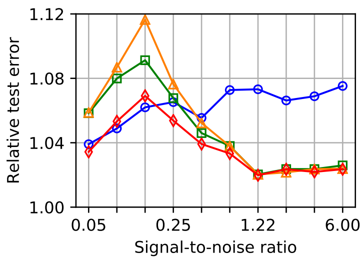
(a) Relative test error, $\rho = 0.35$



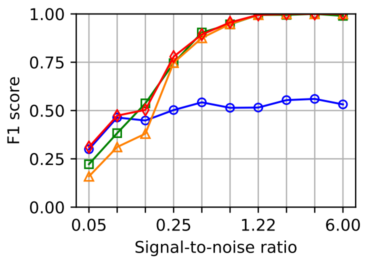
(b) F1 score, $\rho = 0.35$



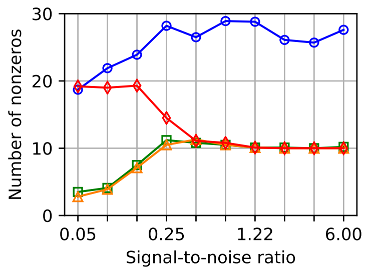
(c) Number of nonzeros, $\rho = 0.35$



(d) Relative test error, $\rho = 0.70$

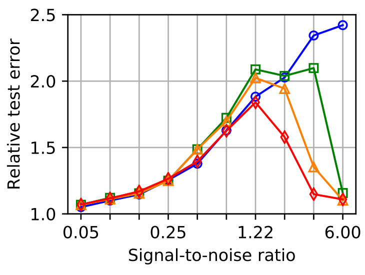


(e) F1 score, $\rho = 0.70$

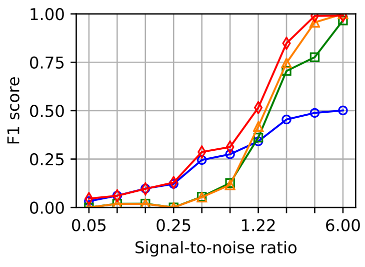


(f) Number of nonzeros, $\rho = 0.70$

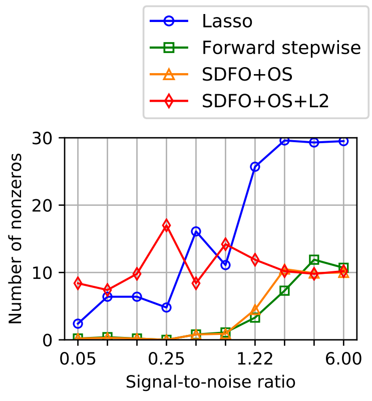
Fig. 4 Evaluation metrics for various signal-to-noise ratios: $n = 500$, $p = 100$, and $s = 10$



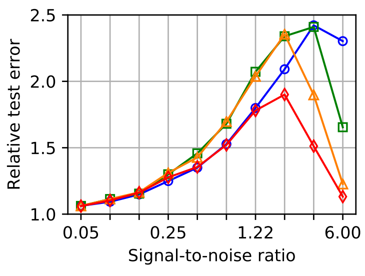
(a) Relative test error, $\rho = 0.35$



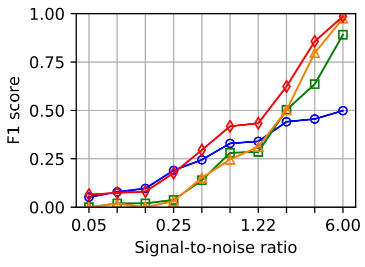
(b) F1 score, $\rho = 0.35$



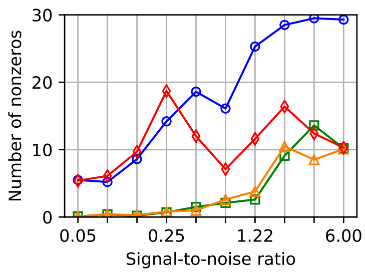
(c) Number of nonzeros, $\rho = 0.35$



(d) Relative test error, $\rho = 0.70$



(e) F1 score, $\rho = 0.70$



(f) Number of nonzeros, $\rho = 0.70$

Fig. 5 Evaluation metrics for various signal-to-noise ratios: $n = 100$, $p = 500$, and $s = 10$