Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Fast Lasso Algorithm via Selective Coordinate Descent

Yasuhiro Fujiwara[†], **Yasutoshi Ida**[†], **Hiroaki Shiokawa**[‡†], **Sotetsu Iwamura**[†]
[†]NTT Software Innovation Center, 3-9-11 Midori-cho Musashino-shi, Tokyo, 180-8585, Japan
[‡]Center for Computational Science, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
{fujiwara.yasuhiro, ida.yasutoshi, iwamura.sotetsu}@lab.ntt.co.jp, shiokawa@cs.tsukuba.ac.jp

Abstract

For the AI community, the lasso proposed by Tibshirani is an important regression approach in finding explanatory predictors in high dimensional data. The coordinate descent algorithm is a standard approach to solve the lasso which iteratively updates weights of predictors in a round-robin style until convergence. However, it has high computation cost. This paper proposes *Sling*, a fast approach to the lasso. It achieves high efficiency by skipping unnecessary updates for the predictors whose weight is zero in the iterations. Sling can obtain high prediction accuracy with fewer predictors than the standard approach. Experiments show that Sling can enhance the efficiency and the effectiveness of the lasso.

Introduction

The lasso is a popular l_1-regularized least squares regression approach for high dimensional data (Tibshirani 1996). It continues to attract more attention in the field of artificial intelligence (Zhou et al. 2015; Gong and Zhang 2011). In many practical learning problems, it is important to identify predictors that have some relationship with a response (Nakatsuji and Fujiwara 2014; Nakatsuji et al. 2011). The central requirement for good predictors is that they should have high correlations with the response but should not be correlated with each other. The main challenge is to find the smallest possible set of predictors while achieving high prediction accuracy for the response (Hastie, Tibshirani, and Friedman 2011). The most appealing property of the lasso is the sparsity of the solution by adding an l_1-norm regularization term to the squared loss term; the lasso effectively uses the l_1-norm constraint to shrink/suppress predictors in finding the sparse set of predictors. Due to its effectiveness, the lasso is used in a variety of applications such as image processing (Liu et al. 2014), topic detection (Kasiviswanathan et al. 2011), and disease diagnosis (Xin et al. 2014).

Although the lasso was developed in the mid-1990s, it did not receive much attention until the early 2000s since its computation cost is high (Tibshirani 2011). The original lasso paper used an off-the-shelf approach that did not scale well for large data. In 2002, Tibshirani et al. developed the LARS algorithm to efficiently solve the lasso. This led to many approached applying the lasso to a variety of problems such as the elastic Net (Zou and Hastie 2005) and grouped Lasso (Yuan and Lin 2006). In 2007, a research team of Tibshirani et al. proposed the coordinate descent algorithm, which is faster than the LARS algorithm as demonstrated in (Friedman et al. 2007). The coordinate descent algorithm iteratively updates weights of predictors one at a time to find the solution. The subsequent papers of 2010 and 2012 improved the efficiency of the coordinate descent algorithm (Friedman, Hastie, and Tibshirani 2010; Tibshirani et al. 2012). The coordinate descent algorithm is now regarded as the standard approach for implementing the lasso (Zhou et al. 2015)[1]; this paper refers to the coordinate descent algorithm based approach as the standard approach.

However, current applications must handle large data sets. In the proposal of mid-1990s, the lasso was applied to prostate cancer data which has at most ten predictors (Tibshirani 1996). Recent image processing applications, however, handle data containing thousands of predictors (Liu et al. 2014). Moreover, in topic detection, the number of predictors reaches the tens of thousands (Kasiviswanathan et al. 2011). In order to increase the processing speed of the lasso, many researchers focused on screening techniques (Ghaoui, Viallon, and Rabbani 2010; Tibshirani et al. 2012; Liu et al. 2014). Since screening can detect predictors that have weights of zero as a solution before the iterations, it can improve the efficiency of the coordinate descent algorithm. However, as mentioned in the previous paper, the efficiency of the coordinate descent algorithm should be improved to handle the large size of data (Tibshirani 2011).

This paper proposes *Sling* as a novel and efficient algorithm for the lasso. In the standard approach, weights of predictors are iteratively updated in a round-robin manner until convergence if they are not pruned by the screening technique. The standard approach computes a weight for each predictor by using nonzero weights of other predictors. Therefore, once a predictor has a nonzero weight in the iterations, it induces additional computation cost even if the weight of the predictor is zero after the convergence. The same as the standard approach, Sling is based on the coordinate descent algorithm. However, it updates weights

[1] The *glmnet* R language package is an implementation of the coordinate descent algorithm.

in a selective style unlike the standard approach; it updates only the predictors that must have nonzero weights in the iterations. After convergence, our approach updates the predictors that are expected to have nonzero weights. Since we avoid updating the predictors whose weights are zero in the iterations, our approach can efficiently obtain the solution for the lasso. Note that the screening technique prunes predictor whose weights are zero before entering the iterations while our approach prunes predictors in the iterations. If p is the number of predictors where each predictor has n observations, the given data is represented as a $n \times p$ matrix. When the matrix has full column rank, our approach provably guarantees to output the same result as the standard approach. Moreover, our approach achieves high prediction accuracy by using fewer predictors than the standard approach. Experiments demonstrate that Sling cuts the processing time by up to 70% from the standard approach. To the best of our knowledge, this is the first study to improve the efficiency of the standard approach for the lasso by pruning unnecessary predictors in the iterations.

Related Work

Since the lasso is an important regression approach used in a variety of applications such as image processing, topic detection, and disease diagnosis, many approaches have been developed to efficiently solve it.

By employing the sparsity in the solutions of the lasso, several screening techniques prune predictors whose weights are zero before entering the iterations (Liu et al. 2014; Tibshirani et al. 2012; Ghaoui, Viallon, and Rabbani 2010). Since recent applications must handle large numbers of predictors, screening techniques have been attracting much interest. However, these techniques do not enhance prediction accuracy unlike our approach. Since our approach is based on the standard approach, it adopts the sequential strong rule. However, our approach can also use previous other screening techniques.

To efficiently solve the lasso, several approaches transform l_1-regularized least squares as a constrained quadratic programming problem such as the interior method (Kim et al. 2007), GPSR (Figueiredo, Nowak, and Wrigh 2007), and ProjectionL1 (Schmidt, Fung, and Rosales 2007). However, these approaches double variable size, raising computational cost. DPNM proposed by Gong et al. derives another dual form of l_1-regularized least squares to apply a projected Newton method to efficiently solve the dual problem (Gong and Zhang 2011). However, their approach has a limitation in that it assumes $n \geq p$ and matrix $\mathbf{X}$ must have full column rank. Zhou et al. proposed an interesting approach that transforms the lasso regression problem into a binary SVM classification problem to allow efficient solution computation (Zhou et al. 2015). However, their approach does not guarantee to output the optimal solution for the lasso. FISTA is a scalable proximal method for l_1-regularized optimization which can be applied to various loss functions (Beck and Teboulle 2009). However, its convergence rate tends to suffer if predictors have high correlation. ADMM is another popular approach for l_1-regularized optimization problems such as the lasso (Das, Johnson, and Banerjee 2014). However, as described in (Li et al. 2015; Zhou et al. 2015), the ADMM-based approach does not effectively reduce the computation time for the lasso relative to glmnet which is based on the coordinate descent algorithm; the coordinate descent algorithm is more efficient than ADMM for the lasso as demonstrated in (Li et al. 2014).

Preliminary

This section introduces the standard approach for the lasso. In the regression scenario, we are provided with a response vector and p predictor vectors of n observations. We assume each vector is centered and normalized. Let $\mathbf{y} = (y[1], y[2], \dots, y[n])^\top$ be the response vector where $\mathbf{y} \in \mathcal{R}^n$. In addition, let $\mathbf{X} \in \mathcal{R}^{n \times p}$ be the matrix of p predictors where $\mathbf{x}_i$ is the i-th column vector in matrix $\mathbf{X}$. Note that $\mathbf{x}_i$ corresponds to each predictor. The lasso learns the following sparse linear model to predict $\mathbf{y}$ from $\mathbf{X}$ by minimizing the squared loss and an l_1-norm constraint (Tibshirani 1996):

$$\min_{\mathbf{w} \in \mathcal{R}^p} \tfrac{1}{2n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1 \tag{1}$$

where $\mathbf{w} = (w[1], w[2], \dots, w[p])^\top$ denotes the weight vector and $\lambda > 0$ is a tuning parameter; weight $w[i]$ corresponds to the i-th predictor p_i. If matrix $\mathbf{X}$ has full column rank, we have a unique solution for the optimization problem (Hastie, Tibshirani, and Friedman 2011). Otherwise, which is necessarily the case when $p > n$, there may not be a unique solution. The optimal λ is usually unknown in practical applications. Therefore, we need to solve Equation (1) for a series of tuning parameters $\lambda_1 > \lambda_2 > \ldots > \lambda_K$ where K is the number of tuning parameters, and then select the solution that is optimal in terms of a pre-specified criterion such as the Schwarz Bayesian information criterion, Akaike information criterion or cross-validation (Bishop 2007).

Tibshirani et al. proposed a coordinate descent algorithm that updates predictor weights one at a time (Friedman et al. 2007). The coordinate descent algorithm partially conducts the optimization with respect to weight $w[i]$ by supposing that it has already estimated other weights. It computes the gradient at $w[i] = \tilde{w}[i]$, which only exists if $w[i] \neq 0$. If $\tilde{w}[i] > 0$, we have the following equation for the gradient by differentiating Equation (1) with respect to $w[i]$:

$$-\tfrac{1}{n} \textstyle\sum_{j=1}^{n} \left\{ x[j,i] (y[j] - \sum_{k=1}^{p} x[j,k] \tilde{w}[k]) \right\} + \lambda \tag{2}$$

where $\tilde{\mathbf{w}} = (\tilde{w}[1], \tilde{w}[2], \dots, \tilde{w}[p])^\top$ is a weight vector and $x[j,i]$ is (j,i)-th element of matrix $\mathbf{X}$. A similar expression exists if $\tilde{w}[i] < 0$, and $\tilde{w}[i] = 0$ is treated separately. The coordinate descent algorithm updates weights as follows:

$$\tilde{w}[i] \leftarrow S(z[i], \lambda) = \begin{cases} z[i] - \lambda & (z[i] > 0 \text{ and } |z[i]| > \lambda) \\ z[i] + \lambda & (z[i] < 0 \text{ and } |z[i]| > \lambda) \\ 0 & (|z[i]| \leq \lambda) \end{cases} \tag{3}$$

In Equation (3), $S(z[i], \lambda)$ is the soft-thresholding operator (Friedman et al. 2007) and $z[i]$ is a parameter for the i-th predictor given as follows:

$$z[i] = \tfrac{1}{n} \textstyle\sum_{j=1}^{n} x[j,i] (y[j] - \tilde{y}^{(i)}[j]) \tag{4}$$

where $\tilde{y}^{(i)}[j] = \sum_{k \neq i} x[j,k] \tilde{w}[k]$. The coordinate descent algorithm iteratively updates weights in a round-robin manner; we iteratively update all predictors by using Equation (3) until convergence.

Tibshirani et al. proposed an efficient way to compute parameter $z[i]$ used in the updates (Friedman, Hastie, and Tibshirani 2010). They transformed Equation (4) as follows:

$$z[i] = \tilde{w}[i] + \frac{1}{n}\left(\langle \mathbf{x}_i, \mathbf{y}\rangle - \sum\nolimits_{j:|\tilde{w}[j]|>0} \langle \mathbf{x}_i, \mathbf{x}_j\rangle \tilde{w}[j]\right) \quad (5)$$

where $\langle \mathbf{x}_i, \mathbf{y}\rangle$ is the inner product of vector $\mathbf{x}_i$ and $\mathbf{y}$, *i.e.*, $\langle \mathbf{x}_i, \mathbf{y}\rangle = \sum_{j=1}^{n} x[j,i]y[j]$. Note that Equation (4) and (5) give the same result. If m is the number of predictors that have nonzero weights, Equation (5) requires $O(m)$ time to update a weight while Equation (4) needs $O(n)$ time. Since the lasso sparsely selects predictors, we have $m \ll n$. Therefore, we can update weights more efficiently by using Equation (5) instead of Equation (4). To exploit Equation (5), we need to initially compute the inner products of response vector $\mathbf{y}$ with each predictor vector $\mathbf{x}_i$ before the iterations. In addition, each time a predictor vector $\mathbf{x}_j$ is adopted by the model to predict the response vector, we need to compute its inner product with all remaining predictors. That is, if a predictor is additionally determined to have a nonzero weight in the solution, the inner products of the predictor with other predictors must be computed to use Equation (5).

So as to increase the efficiency, several researchers, including Tibshirani et al., proposed screening techniques to detect predictors whose weights are zero (Ghaoui, Viallon, and Rabbani 2010; Tibshirani et al. 2012; Liu et al. 2014). Let $\tilde{\mathbf{w}}_{\lambda_k}$ be the solution for tuning parameter λ_k, Tibshirani et al. proposed the sequential strong rule that discards the i-th predictor for tuning parameter λ_k if we have

$$\frac{1}{n}|\mathbf{x}_i^\top(\mathbf{y} - \mathbf{X}\tilde{\mathbf{w}}_{\lambda_{k-1}})| < 2\lambda_k - \lambda_{k-1} \quad (6)$$

Since their screening technique may erroneously discard predictors that have nonzero weights, all discarded predictors are checked by the Karush-Kuhn-Tucker (KKT) condition after convergence. The KKT condition is that $\frac{1}{n}|\mathbf{x}_i^\top(\mathbf{y} - \mathbf{X}\tilde{\mathbf{w}}_{\lambda_j})| \leq \lambda$ holds if $\tilde{w}_i = 0$. However, the efficiency of the coordinate descent algorithm must be improved to handle the large data sizes as described in (Tibshirani 2011).

Proposed Approach

We present our proposal, Sling, that efficiently computes the solution for the lasso based on the standard approach. First, we overview the ideas underlying Sling. That is followed by a full description. Note that our approach can be applied to the variants of the lasso such as the elastic Net and grouped Lasso although we focus on the lasso in this paper.

Ideas

As described in the previous section, the standard approach iteratively updates weights of all predictors by computing parameter $z[i]$ in a round-robin style. In order to increase efficiency, we do not update the weights of all predictors. Instead, our approach skips unnecessary updates; it first updates only predictors that must have nonzero weights until convergence. It then updates the predictors that are likely to have nonzero weights. Our approach dynamically determines the updated predictors by computing lower and upper bounds of parameter $z[i]$ in each iteration. We can improve the efficiency of the standard approach since we effectively prune the unnecessary predictors in the iteration (Shiokawa, Fujiwara, and Onizuka 2015; Fujiwara and Irie 2014). If matrix $\mathbf{X}$ has full column rank, our approach outputs the same result as the standard approach since there is a unique lasso solution due to its convex property (Tseng 2001). Otherwise, our approach can more accurately predict the response with fewer predictors than the standard approach since it can avoid updating predictors that give zero gradient by using the bounds in the iterations.

Upper and Lower Bounds

In each iteration, we compute the upper/lower bounds of parameter $z[i]$ for each predictor by setting a reference vector; the reference vector consists of weights of predictors set prior to commencing the iterations. In order to compute the bounds, we apply the Cauchy-Schwarz inequality (Steele 2004) to determine the difference between the reference vector and a weight vector in the iterations. Let $\tilde{\mathbf{w}}_r = (\tilde{w}_r[1], \tilde{w}_r[2], \ldots, \tilde{w}_r[p])^\top$ be the reference vector, we define the upper bound of parameter $z[i]$ as follows:

Definition 1 *Let $\overline{z}[i]$ be the upper bound of parameter $z[i]$, $\overline{z}[i]$ is given as follows:*

$$\overline{z}[i] = \tilde{w}[i] - \tilde{w}_r[i] + \frac{1}{n}\|\mathbf{v}_i\|_2\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2 + z_r[i] \quad (7)$$

In Equation (7), $\mathbf{v}_i$ is a vector of length p whose j-th element is $\langle \mathbf{x}_i, \mathbf{x}_j\rangle$, the inner product of vector $\mathbf{x}_i$ and $\mathbf{x}_j$. In addition, $z_r[i]$ is a score of parameter $z[i]$ given by reference vector $\tilde{\mathbf{w}}_r$, i.e., $z_r[i]$ is computed as follows:

$$z_r[i] = \tilde{w}_r[i] + \frac{1}{n}\left(\langle \mathbf{x}_i, \mathbf{y}\rangle - \sum\nolimits_{j:|\tilde{w}_r[j]|>0} \langle \mathbf{x}_i, \mathbf{x}_j\rangle \tilde{w}_r[j]\right) \quad (8)$$

Note that we can compute $\|\mathbf{v}_i\|_2$ and $z_r[i]$ in Equation (7) before commencing the iterations since they have constant scores throughout the iterations. On the other hand, we need $O(p)$ time to compute $\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2$ in each iteration since $\mathbf{w}$ is a vector of length p that is updated in each iteration. Similarly, the lower bound is defined as follows:

Definition 2 *If $\underline{z}[i]$ is the lower bound of parameter $z[i]$, $\underline{z}[i]$ is given by the following equation:*

$$\underline{z}[i] = \tilde{w}[i] - \tilde{w}_r[i] - \frac{1}{n}\|\mathbf{v}_i\|_2\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2 + z_r[i] \quad (9)$$

We show the following two lemmas to show that $\overline{z}[i]$ and $\underline{z}[i]$ give the upper and lower bounds, respectively;

Lemma 1 *For parameter $z[i]$ of the i-th predictor p_i, we have $\overline{z}[i] \geq z[i]$ in the iterations.*

Proof Since $\mathbf{v}_i$ is a vector whose j-th element is $\langle \mathbf{x}_i, \mathbf{x}_j\rangle$, from Equation (5) and (8), we have

$$\begin{aligned} z[i] &= \tilde{w}[i] + \tfrac{1}{n}(\langle \mathbf{x}_i, \mathbf{y}\rangle - \langle \mathbf{v}_i, \tilde{\mathbf{w}}\rangle) \\ &= \tilde{w}_r[i] + \tilde{w}[i] - \tilde{w}_r[i] + \tfrac{1}{n}(\langle \mathbf{x}_i, \mathbf{y}\rangle - \langle \mathbf{v}_i, \tilde{\mathbf{w}}_r\rangle - \langle \mathbf{v}_i, \tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\rangle) \\ &= z_r[i] + \tilde{w}[i] - \tilde{w}_r[i] - \tfrac{1}{n}\langle \mathbf{v}_i, \tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\rangle \end{aligned}$$

From the Cauchy-Schwarz inequality, we have

$$-\frac{1}{n}\langle \mathbf{v}_i, \tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\rangle \leq \frac{1}{n}\|\mathbf{v}_i\|_2\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2$$

As a result, we have

$$z[i] \leq \tilde{w}[i] - \tilde{w}_r[i] + \frac{1}{n}\|\mathbf{v}_i\|_2\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2 + z_r[i] = \overline{z}[i] \qquad \square$$

Lemma 2 *In the iterations, $\underline{z}[i] \leq z[i]$ holds for parameter $z[i]$ of the i-th predictor, p_i.*

We omit proof of Lemma 2 because of space limitations. However, it can be proved in a similar fashion to Lemma 1 by using the Cauchy-Schwarz inequality.

In terms of computation cost, $O(p)$ time is needed to compute the bounds if we naively use Equation (7) and (9) since it needs $O(p)$ time to compute $\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2$. However, we can efficiently update $\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2$. If $\tilde{w}'[i]$ is the score of $\tilde{w}[i]$ before the update and $\tilde{\mathbf{w}}'$ is the vector of weights before the update, we can update $\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2$ as follows:

$$\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2 = \sqrt{\|\tilde{\mathbf{w}}' - \tilde{\mathbf{w}}_r\|_2^2 - (\tilde{w}'[i])^2 + (\tilde{w}[i])^2} \quad (10)$$

Clearly $O(1)$ time is needed to update $\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2$ every time a weight is updated from Equation (10). Thus, we have the following property in computing the upper/lower bounds:

Lemma 3 *For each predictor, it requires $O(1)$ time to compute the upper/lower bounds in each iteration.*

Proof If a weight is updated, we can obtain $\|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_r\|_2$ at $O(1)$ time. In addition, it requires $O(1)$ time to obtain $\tilde{w}[i] - \tilde{w}_r[i]$, $\|\mathbf{v}_i\|_2$, and $z_r[i]$ in the iterations; we can precompute $\|\mathbf{v}_i\|_2$ and $z_r[i]$ before commencing the iterations. Therefore, it is clear that we can compute the upper/lower bounds at $O(1)$ time from Equation (7) and (9). □

Predictors of Nonzero Weights

We identify the predictors that must/can have nonzero weights by computing the bounds. We can improve the efficiency of the lasso since we can effectively avoid updating the unnecessary predictors (Fujiwara and Shasha 2015). Our approach is based on the properties for predictors that have nonzero weights. The first property is given as follows for predictors that must have nonzero weights in the iterations:

Lemma 4 *Predictor p_i must have a nonzero weight if $\underline{z}[i] > \lambda$ or $\overline{z}[i] < -\lambda$ holds for parameter $z[i]$.*

Proof In the case of $\underline{z} > \lambda$, we have $z[i] \geq \underline{z}[i] > \lambda > 0$ and $|z[i]| \geq |\underline{z}[i]| > \lambda$ since $\lambda > 0$ and $z[i] \geq \underline{z}[i]$ from Lemma 2. As a result, we have $w[i] = z[i] - \lambda$ from Equation (3) if $\underline{z} > \lambda$ holds for predictor p_i. If $\overline{z}[i] < -\lambda$ holds, we similarly have $z[i] < 0$ and $|z[i]| > \lambda$. Therefore, the score of weight $\tilde{w}[i]$ must be nonzero from Equation (3). □

In order to introduce the property of predictors that may have nonzero weights, we show the following property of predictors whose weights must be zero in the iterations:

Lemma 5 *If we have $\overline{z}[i] \leq \lambda$ and $\underline{z}[i] \geq -\lambda$ for parameter $z[i]$, the weight of predictor p_i must be zero in the iterations.*

Proof If $\overline{z}[i] \leq \lambda$ and $\underline{z}[i] \geq -\lambda$ hold, we have $-\lambda \leq \underline{z}[i] \leq z[i] \leq \overline{z}[i] \leq \lambda$ for parameter $z[i]$. Therefore, $|z[i]| \leq \lambda$ holds for parameter $z[i]$. As a result, we have $\tilde{w}[i] = 0$ from Equation (3) if $\overline{z}[i] \leq \lambda$ and $\underline{z}[i] \geq -\lambda$ hold. □

From Lemma 5, we have the following lemma for predictors that may have nonzero weights:

Lemma 6 *Predictor p_i can have a nonzero weight if $\overline{z}[i] > \lambda$ or $\underline{z}[i] < -\lambda$ for parameter $z[i]$.*

Algorithm 1 Sling

```
for k = 1 to K do
    λ := λ_k;
    if k = 1 then
        U := ∅;
    else
        U := P_{k-1};
    compute initial weights by Equation (11);
    repeat
        repeat
            w̃_r := w̃;
            for each p_i ∈ U do
                compute z̃_r[i] from w̃_r;
            repeat
                for each p_i ∈ U do
                    if z̲[i] > λ or z̄[i] < −λ then
                        update w̃[i] by Equation (5);
                        update ‖w̃ − w̃_r‖_2 by Equation (10);
            until w̃ reaches the convergence
            w̃_r := w̃;
            for each p_i ∈ U do
                compute z̃_r[i] from w̃_r;
            repeat
                for each p_i ∈ U do
                    if z̄[i] > λ or z̲[i] < −λ then
                        update w̃[i] by Equation (5);
                    else
                        w̃[i] = 0;
                    update ‖w̃ − w̃_r‖_2 by Equation (10);
            until w̃ reaches the convergence
            for each p_i ∈ S_k do
                if p_i violates the KKT condition then
                    add p_i to U;
        until ∀p_i ∈ S_k, p_i does not violate KKT condition
        for each p_i ∈ P do
            if p_i violates the KKT condition then
                add p_i to U;
    until ∀p_i ∈ P, p_i does not violate KKT condition
```

Proof It is clear from Lemma 5. □

Note that if a predictor meets the condition of Lemma 4, the predictor must meet the condition of Lemma 6; a set of predictors of Lemma 4 is included in a set of predictors of Lemma 6. This is because we have (1) $\overline{z}[i] > \lambda$ if $\underline{z}[i] > \lambda$ and (2) $\underline{z}[i] < -\lambda$ if $\overline{z}[i] < -\lambda$. Therefore, if a predictor must have a nonzero weight by Lemma 4, the predictor can have a nonzero weight by Lemma 6. In addition, if a predictor does not meet the condition of Lemma 6, the weight of the predictor must be zero from Lemma 5. We employ Lemma 4 and 6 to effectively update predictors that must/can have nonzero weights in the iterations.

Algorithm

Algorithm 1 gives a full description of Sling. It is based on the standard approach of the lasso, where the weights of predictors are updated by the transformed equation (Equation (5)) and unnecessary predictors are pruned by the sequential strong rule as described in the previous section. Our approach computes the solutions of the lasso for a series of tuning parameters $\lambda_1 > \lambda_2 > \ldots > \lambda_K$. In Algorithm 1, $\mathcal{U}$ is a set of predictors updated in the iterations, $\mathcal{P}$ is a set of all p predictors, $\mathcal{P}_k$ is a set of predictors that have nonzero weights as the solution for tuning parameter λ_k, and $\mathcal{S}_k$ are predictors that survive the sequential strong rule for λ_k. For λ_k, we compute initial scores of the weights based on the observation that the solution for λ_k is similar to the solution for λ_{k-1}. Specifically, if $w_{\lambda_k}[i]$ is the weight of predictor p_i in the solution for parameter λ_k, we compute the initial scores of the weight for parameter λ_k as follows:

$$\tilde{w}[i] = \begin{cases} 0 & (k=1) \\ w_{\lambda_{k-1}}[i] & (k=2) \\ w_{\lambda_{k-1}}[i] + \Delta w_{\lambda_{k-1}}[i] & (k \geq 3) \end{cases} \quad (11)$$

where $\Delta w_{\lambda_{k-1}}[i] = w_{\lambda_{k-1}}[i] - w_{\lambda_{k-2}}[i]$.

In Algorithm 1, our approach first sets $\lambda := \lambda_k$ and then initializes $\mathcal{U} := \emptyset$ if $k = 1$ and $\mathcal{U} := \mathcal{P}_{k-1}$ otherwise similar to the standard approach (Tibshirani et al. 2012) (lines 2-6). It then computes the initial scores of weights by using Equation (11) (line 7). Before iteratively updating the weights of the predictors, it sets the reference vector and computes parameter $z[i]$ for the vector in order to compute the upper/lower bounds (lines 10-12 and 19-21). In each iterative update, our approach first selects predictors that must have nonzero weights by using the condition of Lemma 4 (lines 13-18). It then performs the iterative update by employing Lemma 6 to identify predictors that may have nonzero weights (lines 22-29). After convergence, our approach checks the KKT condition for predictors p_i such that $p_i \in \mathcal{S}_k$ (lines 30-32) and $p_i \in \mathcal{P}$ (lines 34-36) in order to verify that no predictor violates the KKT condition (line 33 and 37). This process is the same as the standard approach (Tibshirani et al. 2012).

After we obtain the solutions for each tuning parameter, we can select the solution that is optimal in terms of a prespecified criterion such as Schwarz Bayesian information criterion. In terms of the computation cost, our approach has the following property:

Theorem 1 *If m_k is the number of predictors that have nonzero weights in the iterations for tuning parameter λ_k and t is the number of update computations, our approach requires $O(m_k t + np \max\{0, m_k - m_{k-1}\})$ time to compute the solution for λ_k.*

Proof As shown in Algorithm 1, our approach first initializes the set of predictor $\mathcal{U}$ and computes the initial scores of weights from Equation (11). These processes need $O(p)$ time. It then sets the reference vector in $O(m_k)$ time. In the iterations, it updates weights in $O(m_k t)$ time. This is because we can compute the updated weights as well as the KKT condition and parameter $z[i]$ for the reference vector by using Equation (5) which requires $O(m_t)$ time for each computation. Note that our approach can compute the upper/lower bounds in each iteration in $O(t)$ time since it needs $O(1)$ to compute the bounds in each iteration as shown in Lemma 3. In addition, as described in the previous section, if a predictor is additionally determined to have a nonzero weight in the solution, we need to compute its inner product with all the other predictors when using Equation (5). Since (1) we need $O(np)$ time to compute inner products of each predictor for other predictors where n is the number of observations and (2) the number of predictors is $\max\{0, m_k - m_{k-1}\}$ to additionally compute the inner products, it requires $O(np \max\{0, m_k - m_{k-1}\})$ time to compute the inner product when utilizing Equation (5). As a result, our approach needs $O(m_k t + np \max\{0, m_k - m_{k-1}\})$ time for given parameter λ_k. □

In Theorem 1, we assume that $m_k = 0$ if $k = 0$. Since our approach can effectively prune predictors by using upper/lower bounds (Fujiwara et al. 2013), it can improve the efficiency of the standard approach for implementing the lasso. In terms of the regression results, our approach has the following property for a given tuning parameter:

Theorem 2 *Our approach converges the same solution as the standard approach if matrix $\mathbf{X}$ has full column rank.*

Proof As shown in Algorithm 1, our approach first updates predictors if we have $\underline{z}[i] > \lambda$ or $\overline{z}[i] < -\lambda$ for each predictor. It then updates predictors such that $\overline{z}[i] > \lambda$ or $\underline{z}[i] < -\lambda$. Note that it is clear that we have $\overline{z}[i] > \lambda$ or $\underline{z}[i] < -\lambda$ if $\underline{z}[i] > \lambda$ or $\overline{z}[i] < -\lambda$ hold. As a result, these two processes indicate that our approach does not update predictors if $\overline{z}[i] \leq \lambda$ and $\underline{z}[i] \geq -\lambda$ hold for each parameter. For a predictor whose $\overline{z}[i] \leq \lambda$ and $\underline{z}[i] \geq -\lambda$, its weight must be zero as shown in Lemma 5. As a result, our approach cannot prune a predictor if it has a nonzero weight in any iteration. Since there is a unique lasso solution if matrix $\mathbf{X}$ has full column rank, the coordinate descent algorithm will converge to the unique solution (Tibshirani et al. 2012; Friedman, Hastie, and Tibshirani 2010). Since our approach and the standard approach are based on coordinate descent, it is clear that our approach converges the same results as the standard approach if matrix $\mathbf{X}$ has full column rank. □

If matrix $\mathbf{X}$ does not have full column rank, which is necessarily the case when $p > n$, there may not be a unique lasso solution. In that case, our approach can yield high accuracy while using fewer predictors than the standard approach. In the next section, we detail experiments that show the efficiency and effectiveness of Sling.

Experimental Evaluation

We performed experiments on the datasets of *DNA*, *Protein*, *Reuters*, *TDT2*, and *Newsgroups* to show the efficiency and effectiveness of our approach. The datasets have 600, 2871, 8293, 10212, and 18846 items, respectively. In addition, they have 180, 357, 18933, 36771, and 26214 features, respectively. Details of the datasets are shown in Chih-Jen Lin's webpage[2] and Deng Cai's webpage[3]. In the experiments, we randomly picked one feature as the response vector $\mathbf{y}$, and then set the remaining features as the matrix of predictors $\mathbf{X}$ the same as the previous paper (Liu et al. 2014). Note that the number of data items corresponds to the number of observations, n. Since we have $p > n$ in Reuters, TDT2, and Newsgroups, matrix $\mathbf{X}$ does not have full column rank while matrix $\mathbf{X}$ is full column rank for DNA and Protein.

We set $\lambda_1 = \frac{1}{n} \max_i |\langle \mathbf{x}_i, \mathbf{y} \rangle|$ and $\lambda_K = 0.001 \lambda_1$ by following the previous paper (Friedman, Hastie, and Tibshirani 2010). We constructed a sequence of K scores of tuning parameters decreasing from λ_1 to λ_K on a log scale where $K = 50$. In this section, "Sling" and "Standard" represent the results of our approach and the standard approach, respectively. The standard approach prunes predictors by the

[2]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/multiclass.html

[3]http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

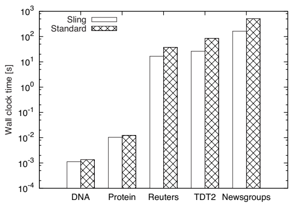

Figure 1: Processing time of each approach.

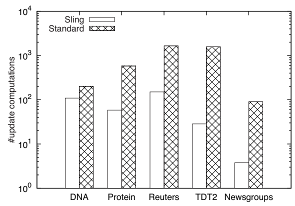

Figure 2: Number of update computations in the iterations.

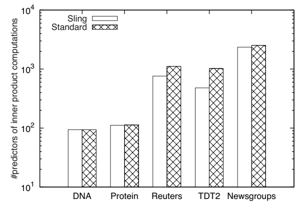

Figure 3: Number of predictors whose inner products are computed.

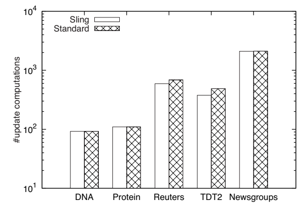

Figure 4: Number of nonzero weight predictors.

Table 1: Prediction results of each approach

Dataset	Squared loss		Object function	
	Sling	Standard	Sling	Standard
DNA	5.097×10^{-2}	5.097×10^{-2}	5.866×10^{-2}	5.866×10^{-2}
Protein	2.616×10^{-3}	2.616×10^{-3}	4.492×10^{-3}	4.492×10^{-3}
Reuters	5.003×10^{-6}	5.006×10^{-6}	7.241×10^{-6}	7.251×10^{-6}
TDT2	1.679×10^{-6}	1.691×10^{-6}	2.760×10^{-6}	2.795×10^{-6}
Newsgroups	3.463×10^{-6}	3.478×10^{-6}	4.811×10^{-6}	4.838×10^{-6}

sequence strong rule and updates the weights by transforming the equation. Since we evaluated each approach by setting a series of tuning parameters λ, we report average values of the experimental results. We conducted all experiments on a Linux 2.70 GHz Intel Xeon server. We implemented all approaches using GCC.

Efficiency

Figure 1 shows the processing time of each approach. In addition, Figure 2 shows the number of updates and Figure 3 shows the number of predictors whose inner products were computed in the iterations.

Figure 1 indicates that our approach greatly improves the efficiency of the standard approach; it cuts the processing time by up to 70% from the standard approach. As described in the previous sections, the standard approach updates all the predictors that survive the sequential screening rule. On the other hand, our approach effectively avoids updating all the survived predictors by computing the upper/lower bounds although our approach employs the sequential screening rule the same as the standard approach. Therefore, the number of update computation of our approach is fewer than that of the standard approach as shown in Figure 2. In addition, as shown in Figure 3, Sling can effectively reduce the number of inner product computations, especially if $p > n$ (Reuters, TDT2, and Newsgroups). This is because Sling reduces the number of predictors used if matrix $\mathbf{X}$ does not have full column rank as we will discuss later. Since our approach effectively reduces the number of updates and inner product computations, it offers superior efficiency over the standard approach.

Effectiveness

In this section, we evaluate the prediction error for the given responses of each approach. This experiment performed leave-one-out cross validation in evaluating the prediction error in terms of the squared loss for the response. Table 1 shows the prediction error and scores of the objective function (Equation (1)). In addition, we show the average number of nonzero weight predictors in Figure 4.

As expected, experimental results indicate that the prediction error and the number of nonzero weight predictors of our approach are the same as those of the standard approach if matrix $\mathbf{X}$ has full column rank (DNA and Protein). This is because our approach is guaranteed to yield the same solution as the standard approach as described in Theorem 2. In addition, for $p > n$ (Reuters, TDT2, and Newsgroups), our approach uses fewer predictors than the standard approach while slightly improving the prediction error compared to the standard approach. As shown in Algorithm 1, we update predictors that must/can have nonzero weights by computing the upper/lower bounds of parameter $z[i]$. Therefore, if $|z[i]|$ has a large score, our approach is likely to update predictor p_i that corresponds to $z[i]$. Since parameter $z[i]$ is introduced from the gradient at $w[i] = \tilde{w}[i]$ as shown in Equation (2), our approach can effectively identify predictors of high gradients in each iteration. As a result, our approach can effectively compute the solution for the optimization problem of Equation (1) by using the upper/lower bounds. Therefore, scores of the objective function of our approach are less than those of the standard approach as shown in Table 1. On the other hand, since the standard approach updates predictors in round-robin style, it can update ineffective predictors as the solution whose weight are zero after convergence. Table 1, along with Figure 1, indicates that our approach improves the efficiency while its prediction results match the accuracy of the standard approach. Thus, Sling is an attractive option for the research community in performing the lasso.

Conclusions

The lasso is an important l_1-regularized least squares regression approach for high dimensional data in the AI community. We proposed Sling, an efficient algorithm that improves the efficiency of applying the lasso. Our approach avoids updating predictors whose weights must/can be zero in each iteration. Experiments showed that our approach offers improved efficiency and effectiveness over the standard approach. The lasso is a fundamental approach used in variety of applications such as image processing, topic detection, and disease diagnosis. Our approach will allow many lasso-based applications to be processed more efficiently, and should improve the effectiveness of future applications.

References

Beck, A., and Teboulle, M. 2009. A Fast Iterative Shrinkage-thresholding Algorithm for Linear Inverse Problems. *SIAM J. Imaging Sciences* 2(1):183–202.

Bishop, C. M. 2007. *Pattern Recognition and Machine Learning*. Springer.

Das, P.; Johnson, N.; and Banerjee, A. 2014. Online Portfolio Selection with Group Sparsity. In *AAAI*, 1185–1191.

Figueiredo, M. A. T.; Nowak, R. D.; and Wrigh, S. J. 2007. Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems. *IEEE Journal on Selected Topics in Signal Processing* 1(4):586 – 597.

Friedman, J.; Hastie, T.; Höfling, H.; and Tibshirani, R. 2007. Pathwise Coordinate Optimization. *Annals of Applied Statistics* 1(2):302–332.

Friedman, J. H.; Hastie, T.; and Tibshirani, R. 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* 33(1):1–22.

Fujiwara, Y., and Irie, G. 2014. Efficient Label Propagation. In *ICML*, 784–792.

Fujiwara, Y., and Shasha, D. 2015. Quiet: Faster Belief Propagation for Images and Related Applications. In *IJCAI*, 3497–3503.

Fujiwara, Y.; Nakatsuji, M.; Shiokawa, H.; Mishima, T.; and Onizuka, M. 2013. Fast and Exact Top-k Algorithm for PageRank. In *AAAI*.

Ghaoui, L. E.; Viallon, V.; and Rabbani, T. 2010. Safe Feature Elimination in Sparse Supervised Learning. *CoRR* abs/1009.3515.

Gong, P., and Zhang, C. 2011. A Fast Dual Projected Newton Method for l_1-regularized Least Squares. In *IJCAI*, 1275–1280.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2011. *The Elements of Statistical Learning*. Springer.

Kasiviswanathan, S. P.; Melville, P.; Banerjee, A.; and Sindhwani, V. 2011. Emerging Topic Detection Using Dictionary Learning. In *CIKM*, 745–754.

Kim, S.-J.; Koh, K.; Lustig, M.; Boyd, S.; and Gorinevsky, D. 2007. An Interior-Point Method for Large-Scale L1-Regularized Least Squares. *IEEE Journal of Selected Topics in Signal Processing* 1(4):606–617.

Li, X.; Zhao, T.; Yuan, X.; and Liu, H. 2014. *An R Package flare for High Dimensional Linear Regression and Precision Matrix Estimation*. https://cran.r-project.org/web/packages/flare/vignettes/vignette.pdf.

Li, X.; Zhao, T.; Yuan, X.; and Liu, H. 2015. The flare Package for High Dimensional Linear Regression and Precision Matrix Estimation in R. *Journal of Machine Learning Research* 16:553–557.

Liu, J.; Zhao, Z.; Wang, J.; and Ye, J. 2014. Safe Screening with Variational Inequalities and Its Application to Lasso. In *ICML*, 289–297.

Nakatsuji, M., and Fujiwara, Y. 2014. Linked Taxonomies to Capture Users' Subjective Assessments of Items to Facilitate Accurate Collaborative Filtering. *Artif. Intell.* 207:52–68.

Nakatsuji, M.; Fujiwara, Y.; Uchiyama, T.; and Fujimura, K. 2011. User Similarity from Linked Taxonomies: Subjective Assessments of Items. In *IJCAI*, 2305–2311.

Schmidt, M. W.; Fung, G.; and Rosales, R. 2007. Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches. In *ECML*, 286–297.

Shiokawa, H.; Fujiwara, Y.; and Onizuka, M. 2015. SCAN++: Efficient Algorithm for Finding Clusters, Hubs and Outliers on Large-scale Graphs. *PVLDB* 8(11):1178–1189.

Steele, J. M. 2004. *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*. Cambridge University Press.

Tibshirani, R.; Bien, J.; Friedman, J.; Hastie, T.; Simon, N.; Taylor, J.; and Tibshirani, R. J. 2012. Strong Rules for Discarding Predictors in Lasso-type Problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 74(2):245–266.

Tibshirani, R. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society, Series B* 58:267–288.

Tibshirani, R. 2011. Regression Shrinkage and Selection via the Lasso: a Retrospective. *Journal of the Royal Statistical Society: Series B* 73(3):273–282.

Tseng, P. 2001. Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. *Journal of Optimization Theory and Applications* 109(3):475–494.

Xin, B.; Kawahara, Y.; Wang, Y.; and Gao, W. 2014. Efficient Generalized Fused Lasso and Its Application to the Diagnosis of Alzheimer's Disease. In *AAAI*, 2163–2169.

Yuan, M., and Lin, Y. 2006. Model Selection and Estimation in Rregression with Grouped Variables. *Journal of the Royal Statistical Society: Series B* 68(1):49–67.

Zhou, Q.; Chen, W.; Song, S.; Gardner, J. R.; Weinberger, K. Q.; and Chen, Y. 2015. A Reduction of the Elastic Net to Support Vector Machines with an Application to GPU Computing. In *AAAI*, 3210–3216.

Zou, H., and Hastie, T. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B* 67(2):301–320.