

Large-Scale Sparse Logistic Regression

Jun Liu
Arizona State University
Tempe, AZ 85287
j.liu@asu.edu

Jianhui Chen
Arizona State University
Tempe, AZ 85287
jianhui.chen@asu.edu

Jieping Ye
Arizona State University
Tempe, AZ 85287
jieping.ye@asu.edu

ABSTRACT

Logistic Regression is a well-known classification method that has been used widely in many applications of data mining, machine learning, computer vision, and bioinformatics. Sparse logistic regression embeds feature selection in the classification framework using the ℓ_1 -norm regularization, and is attractive in many applications involving high-dimensional data. In this paper, we propose Lassplore for solving Large-scale sparse logistic regression. Specifically, we formulate the problem as the ℓ_1 -ball constrained smooth convex optimization, and propose to solve the problem using the Nesterov's method, an optimal first-order black-box method for smooth convex optimization. One of the critical issues in the use of the Nesterov's method is the estimation of the step size at each of the optimization iterations. Previous approaches either applies the constant step size which assumes that the Lipschitz gradient is known in advance, or requires a sequence of decreasing step size which leads to slow convergence in practice. In this paper, we propose an adaptive line search scheme which allows to tune the step size adaptively and meanwhile guarantees the optimal convergence rate. Empirical comparisons with several state-of-the-art algorithms demonstrate the efficiency of the proposed Lassplore algorithm for large-scale problems.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications - Data Mining

General Terms

Algorithms

Keywords

Logistic regression, sparse learning, ℓ_1 -ball constraint, Nesterov's method, adaptive line search

1. INTRODUCTION

Logistic Regression (LR) [18, 23] is a classical classification method that has been used widely in many applications includ-

ing document classification [4, 5], computer vision [9], natural language processing [14, 22], and bioinformatics [2, 20, 29, 34, 35]. For applications with many features but limited training samples, LR is prone to overfitting. Regularization is commonly applied to reduce overfitting and obtain a robust classifier. The ℓ_2 -norm regularization has been used extensively in the LR model, leading to a smooth (differentiable) unconstrained convex optimization problem. Standard optimization algorithms such as Newton method and conjugate gradient method [3, 23] can be applied for solving such a formulation. Recently, there is a growing interest in applying the ℓ_1 -norm regularization in the LR model. The use of the ℓ_1 -norm regularization has long been recognized as a practical strategy to obtain a sparse model. The ℓ_1 -norm regularized sparse LR model is attractive in many applications involving high-dimensional data in that it performs feature selection and classification simultaneously [6, 10, 11, 16, 18, 26, 28, 30, 31].

Solving the ℓ_1 regularized logistic regression is, however, more challenging than solving the ℓ_2 regularized counterpart, since the regularization term is non-differentiable. Many algorithms have been proposed in the past for solving the ℓ_1 regularized logistic regression. The iteratively reweighted least squares least angle regression algorithm (IRLS-LARS) used a quadratic approximation for the average logistic loss function, which was subsequently solved by the LARS method [7, 18]. The Bayesian logistic regression (BBR) algorithm used a cyclic coordinate descent method for the Bayesian logistic regression [8]. Glimpath is a general solver for the ℓ_1 regularized generalized linear models using path following methods [27]; it can solve the ℓ_1 regularized logistic regression. SMLR is another general solver for various sparse linear classifiers [17]; and it can solve the sparse logistic regression. In [16], an interior-point method was proposed for solving the ℓ_1 regularized logistic regression. Recently, an algorithm based on the fixed point continuation algorithm [13] was proposed to solve the ℓ_1 regularized logistic regression [32]. An extensive comparison among twelve sparse logistic regression algorithms was given in [30].

In this paper, we propose the Lassplore algorithm for solving large-scale sparse logistic regression. Specifically, we formulate the problem as the ℓ_1 -ball constrained logistical regression formulation, in which the objective function is continuously differentiable, and the problem domain set is closed and convex. We further propose to solve this problem using the Nesterov's method, which has the optimal convergence rate among the first-order methods. One of the critical issues in the use of the Nesterov's method is the estimation of an appropriate step size in each of the optimization iterations. One simple approach is to apply a constant step sizes, which assumes that the Lipschitz gradient is given in advance. Another approach is to estimate the appropriate step size via the inexact line search scheme; however, such a scheme generates a se-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'09, June 28–July 1, 2009, Paris, France.

Copyright 2009 ACM 978-1-60558-495-9/09/06 ...\$5.00.

quence of decreasing step size and may result in slow convergence in practice. We propose to make use of the Nesterov’s estimate sequence for deriving a specialized adaptive line search scheme, in which the appropriate step size is tuned adaptively for each of the optimization iteration. The proposed line search scheme can lead to improved efficiency in practical implementations while preserving the optimal convergence rate. We have performed experimental studies using a collection of large-scale data sets. Empirical comparisons with several state-of-the-art algorithms demonstrate the efficiency of the proposed Lassplore algorithm for the large-scale sparse logistic regression problems.

Organization: We introduce sparse logistic regression in Section 2, review the Nesterov’s method in Section 3, derive an adaptive line search scheme in Section 4, present the Lassplore algorithm in Section 5, report empirical studies in Section 6, and conclude this paper in Section 7.

2. SPARSE LOGISTIC REGRESSION

Let $\mathbf{a} \in \mathbb{R}^n$ denote a sample, and $b \in \{-1, +1\}$ be the associated (binary) class label. Logistic regression model is given by:

$$\text{Prob}(b|\mathbf{a}) = \frac{1}{1 + \exp(-b(\mathbf{w}^T \mathbf{a} + c))}, \quad (1)$$

where $\text{Prob}(b|\mathbf{a})$ is the conditional probability of the label b , given the sample \mathbf{a} , $\mathbf{w} \in \mathbb{R}^n$ is the weight vector, and $c \in \mathbb{R}$ is the intercept. $\mathbf{w}^T \mathbf{a} + c = 0$ defines a hyperplane in the feature space, on which $\text{Prob}(b|\mathbf{a}) = 0.5$. The conditional probability $\text{Prob}(b|\mathbf{a})$ is larger than 0.5 if $\mathbf{w}^T \mathbf{a} + c$ has the same sign as b , and less than 0.5 otherwise.

Suppose that we are given a set of m training data $\{\mathbf{a}_i, b_i\}_{i=1}^m$, where $\mathbf{a}_i \in \mathbb{R}^n$ denotes the i -th sample and $b_i \in \{-1, +1\}$ denotes the corresponding class label. The likelihood function associated with these m samples is defined as $\prod_{i=1}^m \text{Prob}(b_i|\mathbf{a}_i)$. The negative of the log-likelihood function is called the (empirical) logistic loss, and the average logistic loss is defined as:

$$\begin{aligned} f(\mathbf{w}, c) &= -\frac{1}{m} \log \prod_{i=1}^m \text{Prob}(b_i|\mathbf{a}_i) \\ &= \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-b_i(\mathbf{w}^T \mathbf{a}_i + c))), \end{aligned} \quad (2)$$

which is a smooth and convex function. We can determine \mathbf{w} and c by minimizing the average logistic loss:

$$\min_{\mathbf{w}, c} f(\mathbf{w}, c), \quad (3)$$

leading to a smooth convex optimization problem.

When m , the number of training samples is smaller than n , the dimension of the samples, directly solving the logistic regression formulation in (3) is ill-posed and may lead to overfitting. A standard technique to avoid overfitting is regularization.

2.1 ℓ_1 -Ball Constrained Logistic Regression

When adding an ℓ_1 -norm regularization to $f(\mathbf{w}, c)$, we obtain the ℓ_1 -norm regularized logistic regression problem:

$$\min_{\mathbf{w}, c} f(\mathbf{w}, c) + \varrho \|\mathbf{w}\|_1, \quad (4)$$

where $\varrho > 0$ is a regularization parameter. The solution to the ℓ_1 -norm regularized logistic regression can be interpreted in a Bayesian framework as the maximum a posteriori probability estimate of \mathbf{w} and c , when \mathbf{w} has a Laplacian prior distribution on \mathbb{R}^n and covariance ϱI , and c has the uniform prior on \mathbb{R} .

Since the ℓ_1 regularization term is nonsmooth (non-differentiable), solving the ℓ_1 -norm regularized logistic regression is much more challenging than solving the ℓ_2 -norm case. Despite of its computational challenge, the ℓ_1 -norm regularized logistic regression has recently received great interests, due to the sparseness of the resulting model and its empirical success [11, 16, 18, 28, 30, 31]. There are several strategies for solving (4). The first strategy is to treat (4) as a nonsmooth optimization problem, and then solve it by subgradient-based algorithms [24], e.g., Gauss-Seidel [31], grafting [28], shooting [11], BBR [8]. The second strategy is to apply some smooth approximation to the ℓ_1 -norm, so that (4) can be approximated by a smooth function, which can then be solved by smooth optimization methods, e.g., the smoothl1 [30]. The third strategy is to introduce additional variables to reformulate (4) as a smooth optimization problem with smooth constraint functions, e.g., l1-logreg [16] and ProjectionL1 [30]. The fourth strategy is to convert (4) to the equivalent ℓ_1 -ball constrained optimization problem with a smooth objective function, e.g. the iteratively reweighted least squares (IRLS-LARS) [18].

The Lassplore algorithm proposed in this paper belongs to the fourth category. Specifically, we convert the problem (4) to its equivalent counterpart, the ℓ_1 -ball constrained logistic regression:

$$\begin{aligned} \min_{\mathbf{w}, c} f(\mathbf{w}, c) \\ \text{subject to } \|\mathbf{w}\|_1 \leq z, \end{aligned} \quad (5)$$

for some value of $z \geq 0$, the radius of the ℓ_1 -ball. It is known that there is a one-to-one correspondence between (4) and (5).

We can also employ two types (ℓ_1 and ℓ_2) of regularization by solving the following optimization problem

$$\begin{aligned} \min_{\mathbf{w}, c} f(\mathbf{w}, c) + \frac{\rho}{2} \|\mathbf{w}\|^2 \\ \text{subject to } \|\mathbf{w}\|_1 \leq z. \end{aligned} \quad (6)$$

The problem (6) reduces to: (i) the logistic regression (3), when setting $\rho = 0$ and z a sufficiently large value; (ii) the ℓ_2 -norm regularized logistic regression, when setting $\rho > 0$ and z a sufficiently large value; and (iii) the ℓ_1 -ball constrained logistic regression, when setting $\rho = 0$. In the following discussion, we focus on solving the more general formulation (6).

2.2 Function Value and Gradient Evaluation

Our proposed Lassplore algorithm is a first-order black-box oracle method that evaluates at each iteration the function value and gradient. In (6), the value and gradient of $\frac{\rho}{2} \|\mathbf{w}\|^2$ are easy to compute, so we focus on $f(\mathbf{w}, c)$ in the following discussion.

Let $\mathbf{b} = [b_1, b_2, \dots, b_m]^T \in \mathbb{R}^m$, $\mathbf{1} \in \mathbb{R}^n$ be the vector of all ones, $A = [b_1 \mathbf{a}_1, b_2 \mathbf{a}_2, \dots, b_m \mathbf{a}_m]^T \in \mathbb{R}^{m \times n}$, and $\mathbf{c} \in \mathbb{R}^m$ be an m -dimensional vector with all entries being c . Denote $f'(\mathbf{w}, c) = [\nabla_{\mathbf{w}} f(\mathbf{w}, c)^T, \nabla_c f(\mathbf{w}, c)^T]^T$. We have

$$\nabla_c f(\mathbf{w}, c) = -\frac{1}{m} \mathbf{b}^T (\mathbf{1} - \mathbf{p}), \quad (7)$$

$$\nabla_{\mathbf{w}} f(\mathbf{w}, c) = -\frac{1}{m} A^T (\mathbf{1} - \mathbf{p}), \quad (8)$$

$$\mathbf{p} = \mathbf{1} ./ (\mathbf{1} + \exp(-A\mathbf{w} - \mathbf{b} \odot \mathbf{c})), \quad (9)$$

where $p_i = \text{Prob}(b_i|\mathbf{a}_i)$ is the conditional probability of b_i given \mathbf{a}_i , \odot denotes the componentwise multiplication, and $./$ denotes componentwise division. We can compute $f(\mathbf{w}, c)$ as

$$f(\mathbf{w}, c) = -\frac{1}{m} \mathbf{1}^T \log(\mathbf{p}). \quad (10)$$

From (7)-(10), we see that only the matrix-vector multiplications involving A and A^T are required for computing the objective and

the gradient. Thus, when A is sparse, we can efficiently deal with sparse logistic regression problems with large m and n .

Since our proposed Lassplore algorithm is built on the Nesterov's method with an adaptive line search scheme, we first review the Nesterov's method in Section 3, and derive an adaptive line search scheme for the Nesterov's method in Section 4. Note that, the Nesterov's method can deal with constrained optimizations by the usage of gradient mapping [25]. For convenience of illustration, we first focus on the unconstrained smooth convex optimization in Sections 3 & 4, and then discuss the extension to the constrained optimization in Section 5. The discussions in Sections 3 & 4 are applicable to the general smooth convex optimization.

3. THE NESTEROV'S METHOD

Let us consider the following unconstrained smooth convex minimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^n} g(\mathbf{x}), \quad (11)$$

where the function $g(\mathbf{x})$ belongs to the class of convex and differential family $\mathcal{S}_{\mu, L}^{1,1}(\mathbb{R}^n)$, with L and μ satisfying

$$L \equiv \max_{\mathbf{x} \neq \mathbf{y}} \frac{\|g'(\mathbf{x}) - g'(\mathbf{y})\|}{\|\mathbf{x} - \mathbf{y}\|} < +\infty, \quad (12)$$

$$\mu \equiv \min_{\mathbf{x} \neq \mathbf{y}} \frac{\langle g'(\mathbf{x}) - g'(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle}{\|\mathbf{x} - \mathbf{y}\|^2} \geq 0. \quad (13)$$

L defined in (12) is called the Lipschitz gradient of the function $g(\mathbf{x})$. μ defined in (13) is nonnegative, as the gradient $g'(\cdot)$ is monotone when $g(\mathbf{x})$ is convex. Moreover, when $\mu > 0$, $g(\mathbf{x})$ is called a strongly convex function. When the function $g(\mathbf{x})$ is twice differentiable, we have

$$\mu I_n \preceq g''(\mathbf{x}) \preceq L I_n, \forall \mathbf{x}, \quad (14)$$

where I_n is an $n \times n$ identity matrix, and $A \preceq B$ indicates that the matrix $B - A$ is positive semidefinite. The inequality (14) implies that, L is the upper-bound of the largest eigenvalue of $g''(\mathbf{x})$, $\forall \mathbf{x}$; and similarly μ is the lower-bound of the smallest eigenvalue of $g''(\mathbf{x})$, $\forall \mathbf{x}$. It is clear that the relationship $\mu \leq L$ always holds. In the following discussion, we denote \mathbf{x}^* and g^* as an optimal solution and the optimal objective function value, respectively.

3.1 The Algorithm

The Nesterov's method utilizes two sequences: $\{\mathbf{x}_k\}$ and $\{\mathbf{s}_k\}$, where $\{\mathbf{x}_k\}$ is the sequence of approximate solutions, and $\{\mathbf{s}_k\}$ is the sequence of searching points. The searching point \mathbf{s}_k is the affine combination of \mathbf{x}_{k-1} and \mathbf{x}_k as

$$\mathbf{s}_k = \mathbf{x}_k + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1}), \quad (15)$$

where β_k is a tuning parameter. The approximate solution \mathbf{x}_{k+1} can be computed as a gradient step of \mathbf{s}_k as

$$\mathbf{x}_{k+1} = \mathbf{s}_k - \frac{1}{L_k} g'(\mathbf{s}_k), \quad (16)$$

where $1/L_k$ is the step size. Fig. 1 illustrates how the Nesterov's method works. Starting from an initial point \mathbf{x}_0 , we compute \mathbf{s}_k and \mathbf{x}_{k+1} recursively according to (15) and (16), and arrive at the optimal solution \mathbf{x}^* .

In the Nesterov's method, β_k and L_k are two key parameters. When they are set properly, the sequence $\{\mathbf{x}_k\}$ can converge to the optimal \mathbf{x}^* at a certain convergence rate. The Nesterov's constant scheme [25] and the Nemirovski's line search scheme [24] are two well-known ones for setting β_k and L_k .

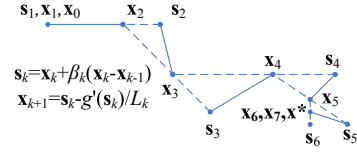


Figure 1: Illustration of the Nesterov's method. We set $\mathbf{x}_1 = \mathbf{x}_0$, and thus $\mathbf{s}_1 = \mathbf{x}_1$. The search point \mathbf{s}_k is the affine combination of \mathbf{x}_{k-1} and \mathbf{x}_k (the dashed lines), and the next approximate solution is obtained by a gradient step of \mathbf{s}_k (solid lines). We assume that $\mathbf{x}_6 = \mathbf{x}_7 = \mathbf{x}^*$, and \mathbf{x}^* is an optimal solution.

The Nesterov's constant scheme assumes that both L and μ are known in advance, and it sets $L_k = L$ and β_k according to L and μ . It has been shown that the resulting scheme can achieve the lower complexity bounds (in the same order) for the class $\mathcal{S}_{\mu, L}^{1,1}(\mathbb{R}^n)$ by first-order black-box methods, and thus the Nesterov's method is an optimal first-order black-box method.

Although the Nesterov's constant scheme can achieve the lower complexity bounds, one major limitation is that both L and μ need to be known in advance, which is however not the case in many applications (e.g., sparse logistic regression). Moreover, the exact computation of L and μ might be much more challenging than solving the problem itself.

3.2 The Nemirovski's Line Search Scheme

To address the problem that L is usually unknown in advance, Nemirovski [24] proposed a line search scheme for determining L_k (see Algorithm 1). In this scheme, we first initialize L_k with L_{k-1} (see Step 2), and then apply a line search process (Steps 4-11) for finding the L_k that satisfies the condition in Step 6. It is clear that, the sequence L_k is non-decreasing with increasing k . Moreover, L_k is upper-bounded by $2L$ since once $L_k \geq L$, the condition in Step 6 always holds [24, Chapter 10.2, page 163]. For β_k , it is computed based on the sequence $\{t_k\}$. As the sequence $\{t_k\}$ is independent of L , μ and the function $g(\mathbf{x})$, the sequence $\{\beta_k\}$ is identical for all the (smooth convex) optimization problems.

Algorithm 1 The Nemirovski's Line Search Scheme

Input: $L_0 > 0$, $\mathbf{x}_1 = \mathbf{x}_0$, N , $t_{-1} = 0$, $t_0 = 1$
Output: \mathbf{x}_N

- 1: **for** $k = 1$ to N **do**
- 2: Set $\beta_k = \frac{t_{k-2}-1}{t_{k-1}}$ and $L_k = L_{k-1}$
- 3: Compute $\mathbf{s}_k = \mathbf{x}_k + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1})$
- 4: **for** $j = 1$ to \dots **do**
- 5: Compute $\mathbf{x}_{k+1} = \mathbf{s}_k - \frac{1}{L_k} g'(\mathbf{s}_k)$
- 6: **if** $g(\mathbf{x}_{k+1}) \leq g(\mathbf{s}_k) - \frac{1}{2L_k} \|g'(\mathbf{s}_k)\|^2$ **then**
- 7: goto Step 12
- 8: **else**
- 9: $L_k = 2L_k$
- 10: **end if**
- 11: **end for**
- 12: Set $t_k = (1 + \sqrt{1 + 4t_{k-1}^2})/2$
- 13: **end for**

THEOREM 1. [24, Chapter 10.2, pages 163-165] Let $L_g = \max(2L, L_0)$ and R_g be the distance from the starting point to the optimal solution set. Then for Algorithm 1, we have

$$g(\mathbf{x}_{N+1}) - g^* \leq \frac{2L_g R_g^2}{(N+1)^2}. \quad (17)$$

Theorem 1 shows that the scheme presented in Algorithm 1 is optimal for the class $\mathcal{S}_{0,L}^{1,1}(\mathbb{R}^n)$. However, there are two limitations: (i) it cannot achieve the Q-linear rate [25] for the class $\mathcal{S}_{\mu,L}^{1,1}(\mathbb{R}^n)$ even when $\mu > 0$ is known; and (ii) the step size $1/L_k$ is only allowed to be monotonically decreasing (note, in proving the convergence rate in Theorem 1, the relationship $L_k \leq L_{k+1}$ is explicitly enforced [24, Chapter 10.2, page 165]). In the next section, we shall propose an adaptive line search scheme that can avoid these limitations.

4. AN ADAPTIVE LINE SEARCH SCHEME

In this section, we propose an adaptive line search scheme for the Nesterov's method. Our line search scheme is built upon the estimate sequence [25, Chapter 2.2], which will be reviewed in Section 4.1. In our line search scheme, we do not assume that L and μ are known in advance, but we assume that, $\tilde{\mu}$, the lower-bound of μ is known in advance. This assumption is reasonable, since 0 is always a lower-bounded of μ as shown in (13). Moreover, for the sparse logistic regression formulation in (6) of Section 2.1, we have $\mu \geq \rho$. The proofs follow similar arguments in [25], and are given in the Appendix.

4.1 Estimate Sequence

Definition 1. [25, Chapter 2.2] A pair of sequences $\{\phi_k(\mathbf{x})\}$ and $\{\lambda_k \geq 0\}$ is called an *estimate sequence* of the function $g(\mathbf{x})$ if the following two conditions hold:

$$\lim_{k \rightarrow \infty} \lambda_k = 0, \quad (18)$$

$$\phi_k(\mathbf{x}) \leq (1 - \lambda_k)g(\mathbf{x}) + \lambda_k\phi_0(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^n. \quad (19)$$

The following theorem provides a systematic way for constructing the estimate sequence:

THEOREM 2. [25, Chapter 2.2]¹ Let us assume that:

1. $g(\mathbf{x})$ is smooth and convex, with Lipschitz gradient L and strongly convexity parameter μ . Moreover, we know the value of $\tilde{\mu}$, which satisfies $\mu \geq \tilde{\mu} \geq 0$.
2. $\phi_0(\mathbf{x})$ is an arbitrary function on \mathbb{R}^n .
3. $\{\mathbf{s}_k\}$ is an arbitrary searching sequence on \mathbb{R}^n .
4. $\{\alpha_k\}$ satisfies: $\alpha_k \in (0, 1)$ and $\sum_{k=0}^{\infty} \alpha_k = \infty$.
5. $\lambda_0 = 1$.

Then $\{\phi_k(\mathbf{x}), \lambda_k\}$ defined by the recursive rules:

$$\lambda_{k+1} = (1 - \alpha_k)\lambda_k, \quad (20)$$

$$\begin{aligned} \phi_{k+1}(\mathbf{x}) &= (1 - \alpha_k)\phi_k(\mathbf{x}) + \alpha_k[g(\mathbf{s}_k) + \\ &\quad \langle g'(\mathbf{s}_k), \mathbf{x} - \mathbf{s}_k \rangle + \frac{\tilde{\mu}}{2}\|\mathbf{x} - \mathbf{s}_k\|^2] \end{aligned} \quad (21)$$

is an estimate sequence.

If we choose a simple quadratic function for $\phi_0(\mathbf{x})$ as $\phi_0(\mathbf{x}) = \phi_0^* + \frac{\gamma_0}{2}\|\mathbf{x} - \mathbf{v}_0\|^2$, then we can specify the estimation sequence defined in Theorem 2 as [25]:

$$\phi_k(\mathbf{x}) = \phi_k^* + \frac{\gamma_k}{2}\|\mathbf{x} - \mathbf{v}_k\|^2, \quad (22)$$

¹Compared to the theorem proposed in [25], we employ $\tilde{\mu}$ in (21) rather than μ , where $\tilde{\mu}$ is a known lower-bound of μ . We make such a substitution for cases where μ is unknown.

where the sequences $\{\gamma_k\}$, $\{\mathbf{v}_k\}$ and $\{\phi_k^*\}$ satisfy:

$$\mathbf{v}_{k+1} = \frac{1}{\gamma_{k+1}} \left[(1 - \alpha_k)\gamma_k \mathbf{v}_k + \tilde{\mu}\alpha_k \mathbf{s}_k - \alpha_k g'(\mathbf{s}_k) \right], \quad (23)$$

$$\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k \tilde{\mu}, \quad (24)$$

$$\begin{aligned} \phi_{k+1}^* &= (1 - \alpha_k)\phi_k^* + \alpha_k g(\mathbf{s}_k) - \frac{\alpha_k^2}{2\gamma_{k+1}} \|g'(\mathbf{s}_k)\|^2 + \\ &\quad \frac{\alpha_k(1 - \alpha_k)\gamma_k}{\gamma_{k+1}} \left(\frac{\tilde{\mu}}{2} \|\mathbf{s}_k - \mathbf{v}_k\|^2 + \langle g'(\mathbf{s}_k), \mathbf{v}_k - \mathbf{s}_k \rangle \right). \end{aligned} \quad (25)$$

The estimate sequence defined in Definition 1 has the following important property:

THEOREM 3. [25, Chapter 2.2] Let $\{\phi_k(\mathbf{x})\}$ and $\{\lambda_k \geq 0\}$ be an estimate sequence. For any sequence $\{\mathbf{x}_k\}$, if

$$g(\mathbf{x}_k) \leq \phi_k^* \equiv \min_{\mathbf{x} \in \mathbb{R}^n} \phi_k(\mathbf{x}), \quad (26)$$

we have

$$g(\mathbf{x}_k) - g^* \leq \lambda_k [\phi_0(\mathbf{x}^*) - g^*] \rightarrow 0. \quad (27)$$

4.2 Proposed Adaptive Line Search Scheme

Our proposed line search scheme is based on Theorem 3. Specifically, we aim at looking for the approximate solution sequence $\{\mathbf{x}_k\}$ (generated with the adaptive step size $\frac{1}{L_k}$) that satisfies the condition in (26), so that the convergence rate of the solution sequence can be analyzed with the sequence $\{\lambda_k\}$, according to Theorem 3.

We first show how to satisfy the condition (26) in the following Lemma:

LEMMA 1. Let $\mathbf{v}_0 = \mathbf{x}_0$ and $\phi_0^* = g(\mathbf{x}_0)$. If we compute the approximate solution \mathbf{x}_{k+1} and searching point \mathbf{s}_k by

$$\mathbf{x}_{k+1} = \mathbf{s}_k - \frac{1}{L_k} g'(\mathbf{s}_k), \quad (28)$$

$$\mathbf{s}_k = \frac{\alpha_k \gamma_k \mathbf{v}_k + \gamma_{k+1} \mathbf{x}_k}{\gamma_k + \alpha_k \tilde{\mu}}, \quad (29)$$

where \mathbf{v}_k is updated according to (23), and $\alpha_k, \gamma_k, L_k, \mathbf{s}_k$ and \mathbf{x}_{k+1} satisfy

$$L_k \alpha_k^2 = \gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k \tilde{\mu}, \quad (30)$$

$$g(\mathbf{s}_k) - \frac{1}{2L_k} \|g'(\mathbf{s}_k)\|^2 \geq g(\mathbf{x}_{k+1}), \quad (31)$$

then, the condition in (26) holds.

Next, we show in the following lemma that \mathbf{s}_k in (29) can be simplified as the combination of \mathbf{x}_{k-1} and \mathbf{x}_k :

LEMMA 2. The search point given in (29) can be computed by

$$\mathbf{s}_{k+1} = \mathbf{x}_{k+1} + \beta_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k), \quad (32)$$

where

$$\beta_{k+1} = \frac{\gamma_{k+1}(1 - \alpha_k)}{\alpha_k(\gamma_{k+1} + L_{k+1}\alpha_{k+1})}. \quad (33)$$

Based on the results in Lemmas 1 & 2, we propose our adaptive line search scheme shown in Algorithm 2. The while loop from Step 2 to Step 11 determines whether L_k should be increased, so that the condition (31) in Step 6 holds. Note that, like the Nemirovski's line search scheme, L_k is upper-bounded by $2L$, since

(31) always holds when $L_k \geq L$. In Step 12, we initialize L_{k+1} as $L_{k+1} = L_k \cdot h(\tau)$. Here, $\tau = 2L_k \frac{g(\mathbf{s}_k) - g(\mathbf{x}_{k+1})}{\|g'(\mathbf{s}_k)\|^2} \geq 1$, due to the condition in Step 6. Intuitively, when τ is large, the step size $\frac{1}{L_k}$ used in computing \mathbf{x}_{k+1} as the gradient step of \mathbf{s}_k is small. In this paper, we employ the following simple piecewise linear function:

$$h(\tau) = \begin{cases} 1, & 1 \leq \tau \leq 5 \\ 0.8, & \tau > 5. \end{cases} \quad (34)$$

Thus, L_{k+1} is reduced to $0.8L_k$ when τ is large. Our experiments show that this particular choice of $h(\cdot)$ works well. We set β_k according to (33). It is clear that β_k is dependent on L_k .

Algorithm 2 An Adaptive Line Search Scheme

Input: $\tilde{\mu}, \alpha_{-1} = 0.5, \mathbf{x}_{-1} = \mathbf{x}_0, L_0 = L_{-1}, \gamma_0 \geq \tilde{\mu}, \lambda_0 = 1$

Output: \mathbf{x}_N

```

1: for  $k = 0$  to  $N$  do
2:   while 1 do
3:     Compute  $\alpha_k \in (0, 1)$  as the root of  $L_k \alpha_k^2 = (1 - \alpha_k) \gamma_k + \alpha_k \tilde{\mu}$ ,  $\gamma_{k+1} = (1 - \alpha_k) \gamma_k + \alpha_k \tilde{\mu}$ ,  $\beta_k = \frac{\gamma_k(1 - \alpha_k - 1)}{\alpha_{k-1}(\gamma_k + L_k \alpha_k)}$ 
4:     Compute  $\mathbf{s}_k = \mathbf{x}_k + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1})$ 
5:     Set  $\mathbf{x}_{k+1} = \mathbf{s}_k - \frac{1}{L_k} g'(\mathbf{s}_k)$ 
6:     if  $g(\mathbf{x}_{k+1}) \leq g(\mathbf{s}_k) - \frac{1}{2L_k} \|g'(\mathbf{s}_k)\|^2$  then
7:       goto Step 12
8:     else
9:        $L_k = 2L_k$ 
10:    end if
11:  end while
12:  Set  $\tau = 2L_k \frac{g(\mathbf{s}_k) - g(\mathbf{x}_{k+1})}{\|g'(\mathbf{s}_k)\|^2}$ ,  $L_{k+1} = h(\tau)L_k$ 
13:  Set  $\lambda_{k+1} = (1 - \alpha_k)\lambda_k$ 
14: end for

```

Recall that the step size $\frac{1}{L_k}$ is only allowed to be monotonically decreasing, i.e., $L_k \leq L_{k+1}$, in the Nemirovski's scheme. We show that although the step size is allowed to decrease, the proposed line search scheme preserves the convergence property, as summarized in the following theorem:

THEOREM 4. For Algorithm 2, we have

$$\lambda_N \leq \min \left\{ \prod_{k=1}^N (1 - \sqrt{\frac{\tilde{\mu}}{L_k}}), \frac{1}{(1 + \sum_{k=1}^N \frac{1}{2} \sqrt{\frac{\gamma_0}{L_k}})^2} \right\}, \quad (35)$$

and

$$g(\mathbf{x}_N) - g^* \leq \lambda_N \left[g(\mathbf{x}_0) - g^* + \frac{\gamma_0}{2} \|\mathbf{x}_0 - \mathbf{x}^*\|^2 \right]. \quad (36)$$

From (35), we can observe that, the smaller L_k is, the smaller λ_N is. Therefore, by decreasing the value of L_k , we can accelerate the convergence.

Next, we compare our proposed line search scheme with existing ones. The proposed scheme is clearly different from Nesterov's constant scheme, as we assume that the Lipschitz gradient is not known in advance. It differs from the Nemirovski's scheme in the following aspects. First, L_k is allowed to decrease in our scheme, which is not the case in the Nemirovski's scheme. Second, in our scheme, β_k is dependent on L_k , while β_k in the Nemirovski's scheme is independent on L_k . Third, the Nemirovski's scheme can only achieve the convergence rate of $O(\frac{1}{\sqrt{N}})$, while our method can achieve the Q-linear convergence rate [25] in the strongly convex case, if $\tilde{\mu} (>0)$, the lower-bound of μ is known.

5. THE PROPOSED APPROACH

We are ready to present the Lassplore algorithm for solving (6). We first discuss the gradient mapping for dealing with the constrained optimization in Section 5.1, and then present the Lassplore algorithm in Section 5.2.

5.1 Gradient Mapping

The constrained optimization problem in (6) is a special case of the following constrained smooth convex optimization:

$$\min_{\mathbf{x} \in G} g(\mathbf{x}), \quad (37)$$

where G is a convex set. In (6), G is the ℓ_1 -ball.

To deal with the constrained optimization problem (37), we construct the gradient mapping, which acts a similar role as the gradient in unconstrained optimization. Let $L_{\mathbf{x}} > 0$. We define

$$g_{L_{\mathbf{x}}, \mathbf{x}}(\mathbf{y}) = g(\mathbf{x}) + \langle g'(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L_{\mathbf{x}}}{2} \|\mathbf{y} - \mathbf{x}\|^2,$$

which is the tangent line of $g(\cdot)$ at \mathbf{x} , regularized by the square distance between \mathbf{y} and \mathbf{x} . Minimizing $g_{L_{\mathbf{x}}, \mathbf{x}}(\mathbf{y})$ in the domain G is the problem of Euclidean projections onto G :

$$\begin{aligned} \pi_G(\mathbf{x} - \frac{1}{L_{\mathbf{x}}} f'(\mathbf{x})) &\equiv \arg \min_{\mathbf{y} \in G} \frac{1}{2} \|\mathbf{y} - (\mathbf{x} - \frac{1}{L_{\mathbf{x}}} f'(\mathbf{x}))\|^2 \\ &= \arg \min_{\mathbf{y} \in G} g_{L_{\mathbf{x}}, \mathbf{x}}(\mathbf{y}). \end{aligned} \quad (38)$$

We call

$$p(L_{\mathbf{x}}, \mathbf{x}) = L_{\mathbf{x}}(\mathbf{x} - \pi_G(\mathbf{x} - \frac{1}{L_{\mathbf{x}}} f'(\mathbf{x}))) \quad (39)$$

the "gradient mapping" of $g(\cdot)$ on G . From (39), we have

$$\pi_G(\mathbf{x} - \frac{1}{L_{\mathbf{x}}} f'(\mathbf{x})) = \mathbf{x} - \frac{1}{L_{\mathbf{x}}} p(L_{\mathbf{x}}, \mathbf{x}),$$

which shows that, $\pi_G(\mathbf{x} - \frac{1}{L_{\mathbf{x}}} f'(\mathbf{x}))$ can be viewed as the result of the "gradient" step in the anti-direction of the gradient mapping $p(L_{\mathbf{x}}, \mathbf{x})$ with stepsize $\frac{1}{L_{\mathbf{x}}}$.

With the gradient mapping, the discussions in Sections 3 & 4 can be extended to solve the constrained optimization problem (37). Moreover, the constrained problem has the same convergence rate as the unconstrained one [25, Chapter 2.2.3].

5.2 The Lassplore Algorithm

In this subsection, we present the proposed Lassplore algorithm for solving (6). For convenience of illustration, we denote the approximate solution $\mathbf{x}_k = [\mathbf{w}_k^T, c_k]^T$, and searching point $\mathbf{s}_k = [(\mathbf{s}_k^w)^T, s_k^c]^T$. We also note that $g(\mathbf{w}, c) = f(\mathbf{w}, c) + \frac{\rho}{2} \|\mathbf{w}\|^2$.

Algorithm 3 is an application of Algorithm 2 to solve the sparse logistic regression problem. Next, we point out the main differences. First, due to the ℓ_1 -ball constraint, in Step 5, \mathbf{w}_{k+1} is computed by the Euclidean projection $\pi_G(\cdot)$. In our problem, $G = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_1 \leq z\}$ is the ℓ_1 -ball, and thus $\pi_G(\cdot)$ is the Euclidean projection onto the ℓ_1 -ball. We make use of method proposed in [21] for computing the projection. Second, the condition in Step 6 is replaced with $g(\mathbf{x}_{k+1}) \leq g(\mathbf{s}_k) + \langle g'(\mathbf{s}_k), \mathbf{x}_{k+1} - \mathbf{s}_k \rangle + \frac{L_k}{2} \|\mathbf{x}_{k+1} - \mathbf{s}_k\|^2$. When this condition holds, we say that L_k is "appropriate" for \mathbf{s}_k [24, Chapter 11]. Due to such a change, we also revise the computation of τ in Step 12.

By the similar analysis, we can extend Algorithm 1 to solve the sparse logistic regression problem.

Algorithm 3 Lassplore: Large-Scale Sparse Logistic Regression**Input:** $\tilde{\mu} = \rho, z > 0, \alpha_{-1} = 0.5, L_0, \gamma_0 \geq \tilde{\mu}, \mathbf{x}_{-1} = \mathbf{x}_0$ **Output:** \mathbf{x}

```

1: for  $k = 0$  to  $\dots$  do
2:   while 1 do
3:     Compute  $\alpha_k \in (0, 1)$  as the root of  $L_k \alpha_k^2 = (1 - \alpha_k) \gamma_k + \alpha_k \tilde{\mu}, \gamma_{k+1} = (1 - \alpha_k) \gamma_k + \alpha_k \tilde{\mu}, \beta_k = \frac{\gamma_k (1 - \alpha_k - 1)}{\alpha_{k-1} (\gamma_k + L_k \alpha_k)}$ 
4:     Compute  $\mathbf{s}_k = \mathbf{x}_k + \beta_k (\mathbf{x}_k - \mathbf{x}_{k-1})$ 
5:     Compute  $\mathbf{w}_{k+1} = \pi_G(\mathbf{s}_k^w - \frac{1}{L_k} \nabla_{\mathbf{w}} g(\mathbf{s}_k^w, \mathbf{s}_k^c))$ 
6:      $c_{k+1} = \mathbf{s}_k^c - \frac{1}{L_k} \nabla_c g(\mathbf{s}_k^w, \mathbf{s}_k^c)$ 
7:     if  $g(\mathbf{x}_{k+1}) \leq g(\mathbf{s}_k) + \langle g'(\mathbf{s}_k), \mathbf{x}_{k+1} - \mathbf{s}_k \rangle + \frac{L_k}{2} \|\mathbf{x}_{k+1} - \mathbf{s}_k\|^2$  then
8:       goto Step 12
9:     else
10:       $L_k = 2L_k$ 
11:    end if
12:  end while
13:  Set  $\tau = \frac{L_k \|\mathbf{x}_{k+1} - \mathbf{s}_k\|^2 / 2}{g(\mathbf{x}_{k+1}) - g(\mathbf{s}_k) - \langle g'(\mathbf{s}_k), \mathbf{x}_{k+1} - \mathbf{s}_k \rangle}, L_{k+1} = h(\tau) L_k$ 
14:  if convergence criterion is satisfied then
15:     $\mathbf{x} = \mathbf{x}_{k+1}$  and terminate the algorithm
16:  end if
17: end for

```

Table 1: Statistics of the test data sets. m denotes the sample size, and n denotes the data dimensionality.

Data set	m	n	# nonzeros
colon-cancer	62	2,000	124,000
leukemia	38	7,129	270,902
duke breast-cancer	44	7,129	313,676
rcv1	20,242	47,236	1,498,952
real-sim	72,309	20,958	3,709,083
news20	19,996	1,355,191	9,097,916

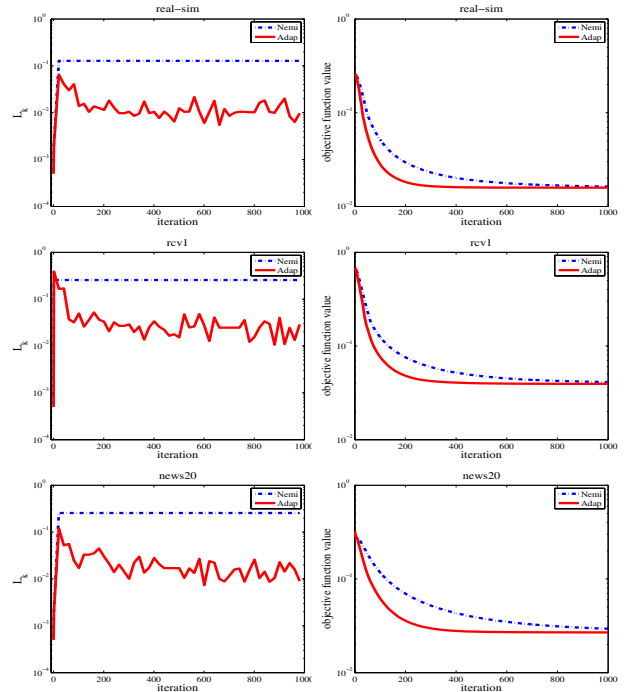
6. EMPIRICAL EVALUATIONS

We have performed experimental studies to evaluate the scalability of the proposed algorithm using the following six data sets: colon-cancer (colon) [1], leukemia (leu) [12], duke breast-cancer (duke) [33], rcv1 [19], real-sim, and news20 [15]. The statistics of the test data sets are given in Table 1 (for rcv1, real-sim, and news20, we use a total of 2,000 samples in the following experiments). All experiments were carried out on an Intel (R) (T2250) 1.73GHZ processor. The source codes are available online².

6.1 Convergence Comparison

In this experiment, we examine the convergence property of the proposed Lassplore algorithm. We conduct experiments on the three large data sets including real-sim, rcv1, and news20. We set the ℓ_1 -ball radius $z = m$, the ℓ_2 regularization parameter $\rho = 0$ and $\tilde{\mu} = \rho$. We run the algorithms for a total of 1,000 iterations, and report both L_k ($1/L_k$ is the step size) and the objective function value in (6). The results are shown in Fig. 2, where the red curve corresponds to the proposed adaptive line search scheme, denoted as ‘‘Adap’’, and the blue curve corresponds to the Nemirovski’s line search scheme, denoted as ‘‘Nemi’’.

We can observe from Fig. 2 that (i) in the Nemirovski’s scheme, the value of L_k is nondecreasing, and L_k becomes quite large after a few iterations; (ii) in the proposed scheme, the value of L_k

**Figure 2: Comparison of the proposed adaptive line search scheme (Adap) and Nemirovski’s line search scheme (Nemi) in terms of the value of L_k (left column) and the objective function value (right column).** For all the plots, the y -axis is plotted in a logarithmic scale.

varies during the iterations; (iii) in most cases, the value of L_k in the proposed scheme is about $1/10$ of the one in the Nemirovski’s scheme. As a result, the proposed scheme converges much faster, which is clear from the plots in the right column; and (iv) the proposed Lassplore algorithm converges rapidly in the first few iterations (note that the y -axis is plotted in a logarithmic scale), which is consistent with the result in Theorem 4.

6.2 Pathwise Solutions

In this experiment, we evaluate the pathwise solutions. It is often the case in practical applications that the optimal ℓ_1 constraint parameter z is unknown. One common approach for solving this problem is to compute the solutions corresponding to a sequence of values of the parameter, e.g., $z_1 < z_2 < \dots < z_s$, from which the optimal one is chosen by evaluating certain criteria. This can be done by simply applying the Lassplore algorithm to solving the s independent problems (called ‘‘cold-start’’). However, a more efficient approach is through the so-called ‘‘warm-start’’, which uses the solution of the previous problem as the warm-start of the latter. Indeed, the proposed Lassplore algorithm can benefit from the warm-start technique, as the solution corresponding to z_i is always within the ℓ_1 -ball of radius z_{i+1} , and thus is feasible to the problem corresponding to z_{i+1} .

We conduct the experiments using the colon data set. We choose 100 values of z , uniformly distributed over $[0.005m, 0.5m]$ on a logarithmic scale. The results are presented in Fig. 3 (left plot). We can observe from the figure that the warm-start approach requires a fewer number of iterations (and thus less computation time) than the cold-start approach. We show the number of nonzeros of the solution \mathbf{w} under different values of z in the right plot of Fig. 3. We can observe that the number of nonzeros usually increases when the radius z becomes larger.

²<http://www.public.asu.edu/~jye02/Software/lassplore/>

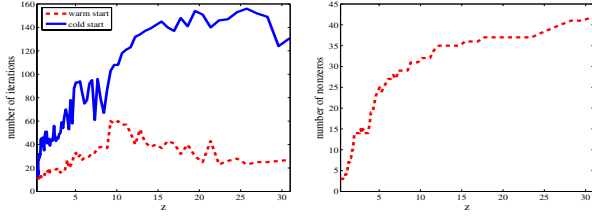


Figure 3: Comparison of cold-start and warm-start for computing the pathwise solutions using the colon data set in terms of the number of iterations required (left plot). The right plot shows the number of nonzeros of the solution w . z is uniformly distributed over $[0.005m, 0.5m]$ on a logarithmic scale.

6.3 Time Efficiency

In this experiment, we compare the proposed Lassplore algorithm with two recent solvers: ProjectionL1 [30] and l1-logreg [16] in terms of the computational time for solving the sparse logistic regression. ProjectionL1 has been shown to be one of the fastest methods among the twelve methods studied in [30], and l1-logreg is quite efficient for solving large-scale sparse logistic regression. Both ProjectionL1 and the proposed algorithms are implemented in Matlab; while l1-logreg is implemented in C, with various external supports such as BLAS, LAPACK and Intel MKL libraries. The results reported below should be interpreted with caution: “It is very difficult, if not possible, to carry out a fair comparison of solutions methods, due to the issue of implementation (which can have a great influence on the algorithm performance), the choice of algorithm parameters, and the different stopping criterion” [16].

Both ProjectionL1 and l1-logreg solve the ℓ_1 -norm regularized logistic regression, while our proposed algorithms solve the ℓ_1 -ball constrained logistic regression. To make a fair comparison, we first run the competing algorithm to obtain the solution corresponding to a given ℓ_1 -norm regularization parameter ϱ , from which we compute the corresponding radius z of the ℓ_1 -ball, and finally run our proposed algorithms with the computed z . It is known that there exists a ϱ_{\max} [16], at which the solution to the problem (4) is zero, and thus ϱ is usually set as a fractional ratio of ϱ_{\max} .

Comparison with ProjectionL1 We use the colon data set in this experiment. We terminate the proposed algorithm, once it achieves the value of $f(w, c)$ equal to or less than that obtained by ProjectionL1. To examine the scalability of the algorithms with an increasing dimensionality (under a fixed sample size), we conduct an experiment by sampling the first 100, 200, 300, 400, and 500 dimensions. We try four settings for ϱ : $10^{-1}\varrho_{\max}$, $10^{-2}\varrho_{\max}$, $10^{-3}\varrho_{\max}$, and $10^{-4}\varrho_{\max}$, and report the results in Fig. 4. We can observe from the figure that (i) the computational time of ProjectionL1 grows much faster than the proposed algorithms when the data dimensionality increases; and (ii) the proposed algorithm based on the adaptive line search scheme consumes much less time than the one based on the Nemirovski’s scheme, which is consistent with our previous study. We observe a similar trend on other two small data sets including leukemia and duke breast-cancer. We have not performed the comparison on the three large data sets, as ProjectionL1 does not scale to large data sets.

Comparison with l1-logreg We use all the six data sets in this experiment. l1-logreg employs the duality gap as the stopping criterion, and we try the following three settings: 10^{-3} , 10^{-4} and 10^{-5} , for exploring the time efficiency under different precisions. Meanwhile, we try the following five values for the ℓ_1 -norm regularization parameter ϱ : $10^{-1}\varrho_{\max}$, $10^{-2}\varrho_{\max}$, $10^{-3}\varrho_{\max}$, $10^{-4}\varrho_{\max}$

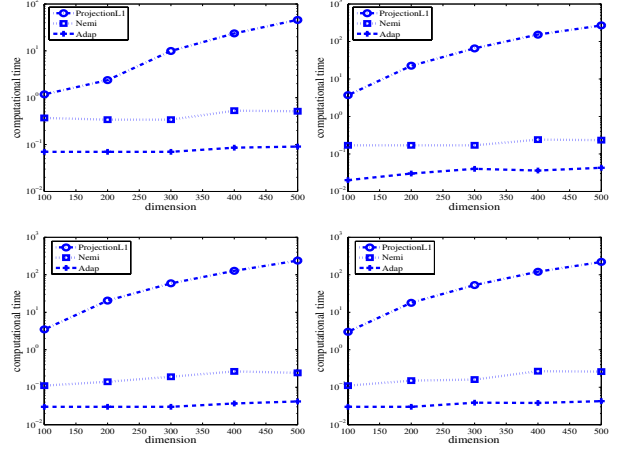


Figure 4: Comparison of the proposed algorithms and ProjectionL1 in terms of the computational time (in seconds) using the colon data set. The x -axis denotes the data dimensionality and the y -axis denotes the computational time. The four plots (from left to right and from top to bottom) correspond to the ℓ_1 -norm regularization parameter $10^{-1}\varrho_{\max}$, $10^{-2}\varrho_{\max}$, $10^{-3}\varrho_{\max}$ and $10^{-4}\varrho_{\max}$, respectively.

and $10^{-5}\varrho_{\max}$. For a given ϱ and duality gap, we first run l1-logreg to compute the solution (w_1, c_1) ; we then compute the objective value $f(w_1, c_1)$; and finally we run the proposed algorithm until the obtained objective function value is within the corresponding duality gap of $f(w_1, c_1)$.

The results are shown in Table 2. We can observe from the table that (i) all three methods are quite efficient for solving sparse logistic regression problems of high dimensionality (see Table 1); (ii) the proposed algorithm based on the adaptive line search scheme outperforms the one based on the Nemirovski’s scheme by a large margin. In most cases, Adap is over three times faster than Nemi; and (iii) Nemi is generally slower than l1-logreg, while Adap is very competitive with l1-logreg in most cases.

7. CONCLUSION

In this paper, we propose the Lassplore algorithm for solving large-scale sparse logistic regression. Specifically, we formulate the sparse logistic regression problem as the ℓ_1 -ball constrained smooth optimization problem, and propose to solve the problem by the Nesterov’s method, an optimal first-order black-box method for the smooth convex optimization. One of the critical issues in the use of the Nesterov’s method is the estimation of the step size at each of the optimization iterations. The Nesterov’s constant scheme and the Nemirovski’s line search scheme are two well-known approaches for setting the step size. The former scheme assumes that the Lipschitz gradient of the given function (to be optimized) is known in advance, which may not be the case in practice; the latter scheme requires a decreasing sequence of the step sizes which leads to a slow convergence. In this paper, we propose an adaptive line search scheme which allows to adaptively tune the step size and meanwhile guarantees an optimal convergence rate. We have conducted an extensive empirical study by comparing the proposed algorithm with several state-of-the-art algorithms. Our empirical results demonstrate the scalability of the Lassplore algorithm for solving large-scale problems.

The efficiency of the proposed algorithm depends on the choice of the function $h(\tau)$ in (34). We plan to explore other choices of

Table 2: Comparison of the computational time (in seconds). Upper part: duality gap $=10^{-3}$; Median part: duality gap $=10^{-4}$; Bottom part: duality gap $=10^{-5}$. The second to the sixth columns ($10^{-1}, 10^{-2}, \dots, 10^{-5}$) correspond to different ratios of ρ over ρ_{\max} (ρ is the regularization parameter employed in l1-logreg, and the solution is zero when $\rho = \rho_{\max}$). “Nemi” denotes the algorithm based on the Nemirovski’s line search scheme, and “Adap” denotes the algorithm based on the proposed adaptive line search scheme.

ρ/ρ_{\max}	10^{-1}			10^{-2}			10^{-3}			10^{-4}			10^{-5}		
	l1-log	Nemi	Adap	l1-log	Nemi	Adap	l1-log	Nemi	Adap	l1-log	Nemi	Adap	l1-log	Nemi	Adap
colon	0.34	0.50	0.15	0.39	1.24	0.17	0.41	2.26	0.19	0.24	1.35	0.18	0.11	0.47	0.15
leu	0.85	2.17	0.34	0.96	5.36	0.39	1.37	7.28	0.41	0.36	1.13	0.23	0.29	0.47	0.17
duke	0.90	2.18	0.33	0.96	5.28	0.39	1.40	7.52	0.43	0.34	1.06	0.22	0.32	0.51	0.18
real-sim	0.51	0.98	0.64	0.79	2.41	1.34	1.97	6.17	1.87	1.85	7.91	1.59	3.58	5.32	1.12
rcv1	1.31	2.17	1.65	2.40	10.71	4.52	6.42	24.64	5.02	6.52	24.03	3.66	3.52	9.66	1.91
news20	74.70	78.77	53.06	58.40	267.9	105.3	282.0	644.2	129.8	541.0	576.6	86.40	926.0	422.0	61.85
colon	0.36	0.82	0.16	0.45	2.10	0.20	0.58	6.21	0.23	0.47	9.38	0.24	0.27	5.43	0.22
leu	1.07	4.24	0.38	1.24	8.60	0.43	2.29	19.21	0.48	1.57	21.42	0.56	0.64	6.16	0.40
duke	1.11	4.30	0.39	1.20	8.43	0.42	2.35	19.24	0.49	1.52	21.50	0.55	0.61	6.20	0.40
real-sim	0.61	1.41	0.87	1.40	3.88	2.27	2.30	12.53	3.26	2.29	26.33	3.23	3.85	29.71	2.77
rcv1	1.61	3.33	2.47	4.00	18.07	7.18	11.80	55.14	9.19	9.96	57.76	8.09	7.72	57.27	5.22
colon	0.45	1.45	0.15	0.59	4.46	0.25	0.79	9.04	0.25	0.71	8.83	0.24	0.59	9.07	0.23
leu	1.27	7.03	0.30	1.53	18.42	0.42	2.70	21.58	0.40	2.80	21.08	0.43	2.52	21.38	0.41
duke	1.24	7.13	0.30	1.59	18.51	0.42	2.75	21.71	0.41	2.82	22.16	0.44	2.47	22.10	0.41
real-sim	0.72	2.75	1.88	1.91	7.24	4.14	3.36	23.27	5.42	3.41	32.37	5.85	4.40	33.12	6.87
rcv1	2.04	8.37	5.85	5.17	36.04	14.28	14.90	56.43	14.21	17.20	57.47	14.94	12.20	58.87	11.01

The results reported in this table should be interpreted with caution, as the issue of implementation can have a great influence on the algorithm performance. Note that, the algorithms Nemi and Adap are implemented in Matlab; while l1-logreg is implemented in C, with various external supports such as BLAS, LAPACK and Intel MKL libraries.

$h(\tau)$ to further improve the algorithm. Many real-world classification problems involve data from multiple classes. We plan to extend the Lassplore algorithm for solving large-scale sparse multinomial logistic regression.

8. ACKNOWLEDGMENTS

This work was supported by NSF IIS-0612069, IIS-0812551, CCF-0811790, NIH R01-HG002516, and NGA HM1582-08-1-0016.

9. REFERENCES

- [1] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Cell Biology*, 96:6745–6750, 1999.
- [2] M. P. Asgary, S. Jahandideh, P. Abdolmaleki, and A. Kazemnejad. Analysis and identification of β -turn types using multinomial logistic regression and artificial neural network. *Bioinformatics*, 23(23):3125–3130, 2007.
- [3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [4] J. R. Brzezinski. Logistic regression modeling for context-based classification. In *DEXA '99: Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, pages 755–759, 1999.
- [5] M. Chang, W. Yih, and C. Meek. A logistic regression model for detecting prominences. In *ACM SIGKDD International conference on Knowledge Discovery and Data Mining*, 1996.
- [6] J. Duchi, S. Shalev-Shwartz, Y. Singer, and C. Tushar. Efficient projection onto the ℓ_1 -ball for learning in high dimensions. In *International Conference on Machine Learning*, 2008.
- [7] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.
- [8] S. Eyheramendy, E. Genkin, W. Ju, D.D. Lewis, and D. Madigan. Sparse bayesian classifiers for text categorization. Technical report.
- [9] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.
- [10] J. Friedman, T. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. Technical report, Department of Statistics, Stanford University, 2008.
- [11] W. Fu. Penalized regressions: the bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7:397–416, 1998.
- [12] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
- [13] E.T. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for ℓ_1 -regularized minimization with applications to compressed sensing. Technical report, CAAM TR07-07, 2007.
- [14] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.
- [15] S. S. Keerthi, K. B. Duan, S. K. Shevade, and A. N. Poo. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61:151–165, 2005.
- [16] K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.
- [17] B. Krishnapuram and A. Hartemink. Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.
- [18] S. Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient ℓ_1 regularized logistic regression. In *The Twenty-first National Conference on Artificial Intelligence*, 2006.
- [19] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.
- [20] J. G. Liao and K. Chin. Logistic regression for disease classification using microarray data. *Bioinformatics*, 23(15):1945–1951, 2007.
- [21] J. Liu and J. Ye. Efficient euclidean projections in linear time. In *International Conference on Machine Learning*, 2009.
- [22] A. Maghbooleh. A logistic regression model for detecting prominences. In *The Fourth International Conference on Spoken Language*, 1996.
- [23] T. P. Minka. A comparison of numerical optimizers for logistic regression. Technical report, 2007.
- [24] A. Nemirovski. *Efficient methods in convex programming*. Lecture Notes, 1994.
- [25] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2003.
- [26] A.Y. Ng. Feature selection, ℓ_1 vs. ℓ_2 regularization, and rotational invariance. In *International Conference on Machine Learning*, 2004.

- [27] M.Y. Park and T. Hastie. ℓ_1 regularized path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B*, 69:659–677, 2007.
- [28] S. Perkins, K. Lacker, and J. Theiler. Grafting: fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.
- [29] M. A. Sartor, G. D. Leikauf, and M. Medvedovic. Lrpath: A logistic regression approach for identifying enriched biological groups in gene expression data. *Bioinformatics*, 25(2):211–217, 2008.
- [30] M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for ℓ_1 regularization: A comparative study and two new approaches. In *The Eighteenth European Conference on Machine Learning*, pages 286–297, 2007.
- [31] S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.
- [32] J. Shi, W. Yin, S. Osher, and P. Sajda. A fast algorithm for large scale ℓ_1 -regularized logistic regression. Technical report, CAAM TR08-07, 2008.
- [33] M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olso, J. R. Marks, and J. R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *National Academy of Sciences*, 98(20):11462–11467, 2001.
- [34] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 14(1):1081–1088, 2001.
- [35] J. Zhu and T. Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443, 2004.

APPENDIX

Proof of Theorem 2: Prove by induction. Considering $\lambda_0 = 1$, we have $\phi_0(\mathbf{x}) \leq (1 - \lambda_0)g(\mathbf{x}) + \lambda_0\phi_0(\mathbf{x}) \equiv \phi_0(\mathbf{x})$. As $g(\mathbf{x})$ belongs to the family class $\mathcal{S}_{\mu, L}^{1,1}(\mathbb{R}^n)$, we have [25]:

$$g(\mathbf{x}) \geq g(\mathbf{s}_k) + \langle g'(\mathbf{s}_k), \mathbf{x} - \mathbf{s}_k \rangle + \frac{\mu}{2} \|\mathbf{x} - \mathbf{s}_k\|^2, \forall \mathbf{x}. \quad (40)$$

Let (19) holds for some $k \geq 0$. Then from (21), we have

$$\begin{aligned} \phi_{k+1}(\mathbf{x}) &\leq (1 - \alpha_k)\phi_k(\mathbf{x}) + \alpha_k g(\mathbf{x}) \\ &= (1 - \alpha_k)(\phi_k(\mathbf{x}) - (1 - \lambda_k)g(\mathbf{x})) \\ &\quad + (1 - (1 - \alpha_k)\lambda_k)g(\mathbf{x}) \\ &\leq (1 - (1 - \alpha_k)\lambda_k)g(\mathbf{x}) + (1 - \alpha_k)\lambda_k\phi_0(\mathbf{x}) \\ &= (1 - \lambda_{k+1})g(\mathbf{x}) + \lambda_{k+1}\phi_0(\mathbf{x}), \end{aligned} \quad (41)$$

where the first inequality follows from (40) and $\mu \geq \tilde{\mu}$, and the last equality utilizes (20). Therefore, (19) holds for $k + 1$. Moreover, the condition 4 ensures that $\lambda_k \rightarrow 0$. \square

Proof of Lemma 1: Prove by induction. It is easy to verify that $g(\mathbf{x}_0) = \phi_0^*$ holds. Let $g(\mathbf{x}_k) \leq \phi_k^*$ hold for some k . From (25), we have

$$\begin{aligned} \phi_{k+1}^* &\geq (1 - \alpha_k)g(\mathbf{x}_k) + \alpha_k g(\mathbf{s}_k) - \frac{\alpha_k^2}{2\gamma_{k+1}} \|g'(\mathbf{s}_k)\|^2 \\ &\quad + \frac{\alpha_k(1 - \alpha_k)\gamma_k}{\gamma_{k+1}} \langle g'(\mathbf{s}_k), \mathbf{v}_k - \mathbf{s}_k \rangle \\ &\geq g(\mathbf{s}_k) - \frac{\alpha_k^2}{2\gamma_{k+1}} \|g'(\mathbf{s}_k)\|^2 \\ &\quad + (1 - \alpha_k) \langle g'(\mathbf{s}_k), \frac{\alpha_k\gamma_k}{\gamma_{k+1}} (\mathbf{v}_k - \mathbf{s}_k) + \mathbf{x}_k - \mathbf{s}_k \rangle \\ &\geq g(\mathbf{x}_{k+1}), \end{aligned} \quad (42)$$

where the first inequality follows from $\phi_k^* \geq g(\mathbf{x}_k)$ and $\frac{\mu}{2} \|\mathbf{s}_k - \mathbf{v}_k\|^2 \geq 0$, the second inequality utilizes the convexity of $g(\mathbf{x})$, i.e., $g(\mathbf{x}_k) \geq g(\mathbf{s}_k) + \langle g'(\mathbf{s}_k), \mathbf{x}_k - \mathbf{s}_k \rangle$, and the last inequality follows from (28-31). \square

Proof of Lemma 2: From (23), (28), (29) and (30), we can write \mathbf{v}_{k+1} as the combination of \mathbf{x}_k and \mathbf{x}_{k+1} as:

$$\begin{aligned} \mathbf{v}_{k+1} &= \frac{1}{\gamma_{k+1}} \left\{ \frac{1 - \alpha_k}{\alpha_k} [(\gamma_k + \alpha_k\tilde{\mu})\mathbf{s}_k - \gamma_{k+1}\mathbf{x}_k] \right. \\ &\quad \left. + \tilde{\mu}\alpha_k\mathbf{s}_k - \alpha_k g'(\mathbf{s}_k) \right\} \\ &= \mathbf{x}_k + \frac{1}{\alpha_k} (\mathbf{s}_k - \mathbf{x}_k) - \frac{1}{\alpha_k L_k} g'(\mathbf{s}_k) \\ &= \mathbf{x}_k + \frac{1}{\alpha_k} (\mathbf{x}_{k+1} - \mathbf{x}_k), \end{aligned} \quad (43)$$

where the first equality follows from (29), the second equality follows from (30), and the last equality follows from (28). Hence, from (29), we can write \mathbf{s}_{k+1} as:

$$\begin{aligned} \mathbf{s}_{k+1} &= \mathbf{x}_{k+1} + \frac{\alpha_{k+1}\gamma_{k+1}(\mathbf{v}_{k+1} - \mathbf{x}_{k+1})}{\gamma_{k+1} + \alpha_{k+1}\tilde{\mu}} \\ &= \mathbf{x}_{k+1} + \frac{\alpha_{k+1}\gamma_{k+1}(1 - \alpha_k)}{\alpha_k(\gamma_{k+1} + \alpha_{k+1}\tilde{\mu})} (\mathbf{x}_{k+1} - \mathbf{x}_k) \\ &= \mathbf{x}_{k+1} + \beta_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k), \end{aligned} \quad (44)$$

where the first equality follows from (30), the second equality utilizes (43), and the last equality follows from the definition of β_{k+1} in (33). \square

Proof of Theorem 4: Prove by induction. As required by the input of Algorithm 2, $\gamma_0 \geq \tilde{\mu}$. If $\gamma_k \geq \tilde{\mu}$, we have

$$\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k\tilde{\mu} \geq \tilde{\mu}. \quad (45)$$

Therefore, we conclude that $\gamma_k \geq \tilde{\mu}$ always holds. From (30), we have $\gamma_{k+1} = L_k\alpha_k^2 \geq \tilde{\mu}$, so that $\alpha_k \geq \sqrt{\frac{\tilde{\mu}}{L_k}}$. Since $\lambda_k = \prod_{i=1}^k (1 - \alpha_i)$, we can get

$$\lambda_N \leq \prod_{k=1}^N \left(1 - \sqrt{\frac{\tilde{\mu}}{L_k}} \right). \quad (46)$$

We have $\gamma_0 \geq \gamma_0\lambda_0$, since $\lambda_0 = 1$. If $\gamma_k \geq \gamma_0\lambda_k$, we have

$$\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k\tilde{\mu} \geq (1 - \alpha_k)\gamma_k \geq (1 - \alpha_k)\gamma_0\lambda_k = \gamma_0\lambda_{k+1},$$

where the first inequality utilizes $\tilde{\mu} \geq 0$, and the last equality follows from $\lambda_{k+1} = (1 - \alpha_k)\lambda_k$. Therefore, $L_k\alpha_k^2 = \gamma_{k+1} \geq \gamma_0\lambda_{k+1}$ always holds.

Since $\alpha_k \in (0, 1)$ and $\lambda_{k+1} = (1 - \alpha_k)\lambda_k$, it is clear that λ_k is strictly decreasing. Denote $a_k = \frac{1}{\sqrt{\lambda_k}}$. We have

$$\begin{aligned} a_{k+1} - a_k &= \frac{\lambda_k - \lambda_{k+1}}{\sqrt{\lambda_k\lambda_{k+1}}(\sqrt{\lambda_k} + \sqrt{\lambda_{k+1}})} \\ &\geq \frac{\lambda_k - \lambda_{k+1}}{2\lambda_k\sqrt{\lambda_{k+1}}} = \frac{\alpha_k}{2\sqrt{\lambda_{k+1}}} \geq \frac{1}{2} \sqrt{\frac{\gamma_0}{L_k}}. \end{aligned} \quad (47)$$

We have $a_0 = \frac{1}{\sqrt{\lambda_0}} = 1$, since $\lambda_0 = 1$. From (47), we have

$$a_k = \frac{1}{\sqrt{\lambda_k}} \geq 1 + \sum_{i=1}^k \frac{1}{2} \sqrt{\frac{\gamma_0}{L_i}}, \quad (48)$$

which leads to

$$\lambda_N \leq \frac{1}{(1 + \sum_{k=1}^N \frac{1}{2} \sqrt{\frac{\gamma_0}{L_k}})^2}. \quad (49)$$

Incorporating (46) and (49), we obtain (35).

In Algorithm 2, Steps 3-6 ensure that conditions (28-31) hold. According to Lemma 1, the condition (26) holds. By using Theorem 3, we obtain (36). \square