Journal of Machine Learning Research 11 (2010) 3183-3234 Submitted 11/09; Revised 7/10; Published 11/10

A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification

Guo-Xun Yuan r96042@csie.ntu.edu.tw
Kai-Wei Chang b92084@csie.ntu.edu.tw
Cho-Jui Hsieh b92085@csie.ntu.edu.tw
Chih-Jen Lin cjlin@csie.ntu.edu.tw
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Editor: Sathiya Keerthi

Abstract

Large-scale linear classification is widely used in many areas. The L1-regularized form can be applied for feature selection; however, its non-differentiability causes more difficulties in training. Although various optimization methods have been proposed in recent years, these have not yet been compared suitably. In this paper, we first broadly review existing methods. Then, we discuss state-of-the-art software packages in detail and propose two efficient implementations. Extensive comparisons indicate that carefully implemented coordinate descent methods are very suitable for training large document data.

Keywords: L1 regularization, linear classification, optimization methods, logistic regression, support vector machines, document classification

1. Introduction

Recently, L1-regularized classifiers have attracted considerable attention because they can be used to obtain a sparse model. Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, l$, $\boldsymbol{x}_i \in R^n$, $y_i \in \{-1, +1\}$, training an L1-regularized linear classifier involves the following unconstrained optimization problem:

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \equiv \|\boldsymbol{w}\|_1 + C \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i), \tag{1}$$

where $\|\cdot\|_1$ denotes the 1-norm and $\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$ is a non-negative (convex) loss function. The regularization term $\|\boldsymbol{w}\|_1$ is used to avoid overfitting the training data. The user-defined parameter $C > 0$ is used to balance the regularization and loss terms.

If $\|\boldsymbol{w}\|_2$ is used in (1), we obtain an L2-regularized classifier. Although L2 regularization is used more commonly, an L1-regularized formula often produces a sparse $\boldsymbol{w}$. Nonzero elements help to select important features; in addition, the time required to produce predictions may be reduced. Considerable literature has been published on the advantages of using L1 regularization; see, for example, Zhao and Yu (2006). However, an L1-regularized

form (1) is not differentiable regardless of its loss function. This drawback leads to greater difficulty in solving the optimization problem. Therefore, certain considerations are required to handle the non-differentiability.

Many loss functions can be used for linear classification. A commonly used one is logistic loss:

$$\xi_{\log}(\boldsymbol{w}; \boldsymbol{x}, y) = \log(1 + e^{-y\boldsymbol{w}^T\boldsymbol{x}}). \tag{2}$$

This loss function is twice differentiable. Note that minimizing logistic loss is equivalent to maximizing the likelihood, whereas minimizing the regularized loss in (1) is equivalent to maximizing the posterior with independent Laplace priors on the parameters. Two other frequently used functions are the L1- and the L2-loss functions:

$$\xi_{\text{L1}}(\boldsymbol{w}; \boldsymbol{x}, y) = \max(1 - y\boldsymbol{w}^T\boldsymbol{x},\ 0) \quad \text{and} \tag{3}$$

$$\xi_{\text{L2}}(\boldsymbol{w}; \boldsymbol{x}, y) = \max(1 - y\boldsymbol{w}^T\boldsymbol{x},\ 0)^2. \tag{4}$$

Because of the $\max(\cdot)$ operation, the L1-loss function is not differentiable. On the other hand, L2 loss is differentiable, but not twice differentiable (Mangasarian, 2002). We refer to (1) with logistic loss as L1-regularized logistic regression and (1) with L1/L2 loss as L1-regularized L1-/L2-loss support vector machines (SVMs).

In some applications, we require a bias term b (also called as an intercept) in the loss function; therefore, $\boldsymbol{w}^T\boldsymbol{x}$ in (2)–(4) is replaced with $\boldsymbol{w}^T\boldsymbol{x} + b$. For example, the logistic loss function becomes

$$\xi_{\log}(\boldsymbol{w}, b; \boldsymbol{x}, y) = \log\left(1 + e^{-y(\boldsymbol{w}^T\boldsymbol{x}+b)}\right).$$

The optimization problem then involves variables $\boldsymbol{w}$ and b:

$$\min_{\boldsymbol{w},b} \quad \|\boldsymbol{w}\|_1 + C\sum_{i=1}^{l} \xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i). \tag{5}$$

Because b does not appear in the regularization term, most optimization methods used to solve (1) can solve (5) as well. In this paper, except whenever b is required, we mainly consider the formulation (1).

Many papers have proposed optimization methods for large-scale L1-regularized logistic regression (i.e., using $\xi_{\log}$ as the loss function). These studies did not consider L1- or L2-loss functions because logistic loss has a probability interpretation and is twice differentiable. These methods differ in various aspects such as the convergence speed, ease of implementation, and practical applicability. With so many available methods, it is important to conduct a comprehensive comparison. Schmidt et al. (2007, 2009) compared optimization methods for L1-regularized classifiers; however, they did not include some state-of-the-art solvers. Moreover, their comparison is based on the number of function evaluations instead of the actual running time. In this paper, we categorize and compare existing methods for logistic regression. We also extend some methods to solve L2-loss SVMs. We exclude L1 loss from the discussion because most methods for logistic regression or L2-loss SVMs assume the differentiability of the loss functions. Readers interested in using L1-loss SVMs can refer to Bradley and Mangasarian (1998), Zhu et al. (2004), Fung and Mangasarian (2004), Mangasarian (2006), and references therein.

1.1 Basic Properties of (1)

In this paper, we assume that the loss function $\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$ is convex, differentiable, and nonnegative. The proof presented in Appendix A shows that (1) attains at least one global optimum. Unlike L2 regularization, which leads to a unique optimal solution, here, (1) may possess multiple optimal solutions. We use $\boldsymbol{w}^*$ to denote any optimal solution of (1). The convexity of $f(\boldsymbol{w})$ implies that all optimal solutions have the same function value, which is denoted as f^*. For more information about the set of optimal solutions, see, for example, Hale et al. (2008, Section 2).

From standard convex analysis, $\boldsymbol{w}^*$ is optimal for (1) if and only if $\boldsymbol{w}^*$ satisfies the following optimality conditions:

$$\begin{cases} \nabla_j L(\boldsymbol{w}^*) + 1 = 0 & \text{if } w_j^* > 0, \\ \nabla_j L(\boldsymbol{w}^*) - 1 = 0 & \text{if } w_j^* < 0, \\ -1 \le \nabla_j L(\boldsymbol{w}^*) \le 1 & \text{if } w_j^* = 0, \end{cases} \tag{6}$$

where $L(\boldsymbol{w})$ is the loss term in (1):

$$L(\boldsymbol{w}) \equiv C \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \tag{7}$$

1.2 Organization and Notation

The remainder of this paper is organized as follows. In Section 2, we briefly survey existing approaches. Section 3 lists state-of-the-art L1-regularized logistic regression packages compared in this study. Sections 4–6 discuss these packages in detail; two of these (Sections 4.1.2 and 5.1) are our proposed implementations. In Section 7, we extend several methods to train L2-loss SVMs. Section 8 describes the extensive experiments that were carried out. Comparison results indicate that decomposition methods (Section 4) are very suitable for large-scale document data. Finally, the discussions and conclusions are presented in Section 9. A supplementary file including additional details and descriptions of experiments is available at `http://www.csie.ntu.edu.tw/~cjlin/papers/l1/supplement.pdf`

In this study, we use consistent notation to describe several state-of-the-art methods that are considered. The following table lists some frequently used symbols:

l:	number of instances
n:	number of features
i:	index for a data instance
j:	index for a data feature
k:	iteration index for an optimization algorithm

We may represent training instances $(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, l$ in the following form:

$$X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_l^T \end{bmatrix} \in R^{l \times n} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_l \end{bmatrix} \in \{-1, +1\}^l.$$

For any vector $\boldsymbol{v}$, we consider the following two representations for sub-vectors:

$$\boldsymbol{v}_{t:s} \equiv [v_t, \ldots, v_s]^T \quad \text{and} \quad \boldsymbol{v}_I \equiv [v_{i_1}, \ldots, v_{i_{|I|}}]^T,$$

where $I = \{i_1, \ldots, i_{|I|}\}$ is an index set. Similarly,

$$X_{I,:} \equiv \begin{bmatrix} \boldsymbol{x}_{i_1}^T \\ \vdots \\ \boldsymbol{x}_{i_{|I|}}^T \end{bmatrix}. \tag{8}$$

The function $\tau(s)$ gives the first derivative of the logistic loss function $\log(1 + e^s)$:

$$\tau(s) \equiv \frac{1}{1 + e^{-s}}. \tag{9}$$

An indicator vector for the jth component is

$$\boldsymbol{e}_j \equiv [\underbrace{0, \ldots, 0}_{j-1}, 1, 0, \ldots, 0]^T. \tag{10}$$

We use $\|\cdot\|$ or $\|\cdot\|_2$ to denote the 2-norm and $\|\cdot\|_1$ to denote the 1-norm.

2. A Survey of Existing Methods

In this section, we survey existing methods for L1-regularized problems. In Sections 2.1–2.3, we focus on logistic regression and L2-loss SVMs for data classification. Section 2.5 briefly discusses works on regression problems using the least-square loss.

2.1 Decomposition Methods

Decomposition methods are a classical optimization approach. Because it is difficult to update all variables simultaneously, at each iteration, we can choose a subset of variables as the working set and fix all others. The resulting sub-problem contains fewer variables and is easier to solve. The various decomposition methods that have been applied to solve L1-regularized problems are categorized into two types according to the selection of the working set.

2.1.1 Cyclic Coordinate Descent Methods

A simple coordinate descent method cyclically chooses one variable at a time and solves the following one-variable sub-problem:

$$\min_z \quad g_j(z) \equiv f(\boldsymbol{w} + z\boldsymbol{e}_j) - f(\boldsymbol{w}), \tag{11}$$

where $\boldsymbol{e}_j$ is defined in (10). This function $g_j(z)$ has only one non-differentiable point at $z = -w_j$. In optimization, the cyclic method for choosing working variables is often called the Gauss-Seidel rule (Tseng and Yun, 2009).

Several works have applied coordinate descent methods to solve (1) with logistic loss. Here, a difficulty arises in that sub-problem (11) does not have a closed-form solution.

Goodman (2004) assumed nonnegative feature values (i.e., $x_{ij} \geq 0,\ \forall i, j$) and then approximated $g_j(z)$ by a function $A_j(z)$ at each iteration. $A_j(z)$ satisfies $A_j(z) \geq g_j(z), \forall z$, and $A_j(0) = g_j(0) = 0$; therefore, minimizing $A_j(z)$ may reduce the function value. Moreover, there is a closed-form solution for minimizing $A_j(z)$. Goodman (2004) actually studied a problem with additional constraints $w_j \geq 0$; however, his approach can be extended to solve the original L1-regularized logistic regression by taking the sign of w_j into consideration.

Genkin et al. (2007) implemented a cyclic coordinate descent method called BBR to solve L1-regularized logistic regression. BBR approximately minimizes sub-problem (11) in a trust region and applies one-dimensional Newton steps. Balakrishnan and Madigan (2005) reported an extension of BBR for online settings.

In Section 4.1.2, we propose a coordinate descent method by extending Chang et al.'s (2008) approach for L2-regularized classifiers. Chang et al. (2008) approximately solved the sub-problem (11) by a one-dimensional Newton direction with line search. Experiments show that their approach outperforms a BBR variant for L2-regularized classification. Therefore, for L1 regularization, this approach might be faster than BBR. Hereafter, we refer to this efficient coordinate descent method as CDN (coordinate descent using one-dimensional Newton steps).

Tseng and Yun (2009) broadly discussed decomposition methods for L1-regularized problems. One of their approaches is a cyclic coordinate descent method. They considered a general cyclic setting so that several working variables are updated at each iteration. We show that CDN is a special case of their general algorithms.

If we randomly select the working variable, then the procedure becomes a stochastic coordinate descent method. Shalev-Shwartz and Tewari (2009, Algorithm 2) recently studied this issue. Duchi and Singer (2009) proposed a similar coordinate descent method for the maximum entropy model, which is a generalization of logistic regression.

2.1.2 Variable Selection Using Gradient Information

Instead of cyclically updating one variable, we can choose working variables based on the gradient information.[1] This method for selecting variables is often referred to as the Gauss-Southwell rule (Tseng and Yun, 2009). Because of the use of gradient information, the number of iterations is fewer than those in cyclic coordinate descent methods. However, the cost per iteration is higher. Shevade and Keerthi (2003) proposed an early decomposition method with the Gauss-Southwell rule to solve (1). In their method, one variable is chosen at a time and one-dimensional Newton steps are applied; therefore, their method differs from the cyclic coordinate descent methods described in Section 2.1.1 mainly in terms of finding the working variables. Hsieh et al. (2008) showed that for L2-regularized linear classification, maintaining the gradient for selecting only one variable at a time is not cost-effective. Thus, for decomposition methods using the gradient information, a larger working set should be used at each iteration.

In the framework of decomposition methods proposed by Tseng and Yun (2009), one type of method selects a set of working variables based on the gradient information. The working set can be large, and therefore, they approximately solved the sub-problem. For the same method, Yun and Toh (2011) enhanced the theoretical results and carried out

1. Because $f(\boldsymbol{w})$ is not differentiable, $\nabla_j L(\boldsymbol{w}) \pm 1$ is used according to the sign of w_j.

experiments with document classification data. We refer to their method as CGD-GS because the method described in Tseng and Yun (2009) is called "coordinate gradient descent" and a Gauss-Southwell rule is used.

2.1.3 Active Set Methods

Active set methods are a popular optimization approach for linear-constrained problems. For problem (1), an active method becomes a special decomposition method because at each iteration, a sub-problem over a working set of variables is solved. The main difference from decomposition methods described earlier is that the working set contains all non-zero variables. Therefore, an active set method iteratively predicts what a correct split of zero and non-zero elements in $\boldsymbol{w}$ is. If the split is correct, then solving the sub-problem gives the optimal values of non-zero elements.

Perkins et al. (2003) proposed an active set method for (1) with logistic loss. This implementation uses gradient information to predict $\boldsymbol{w}$'s zero and non-zero elements.

2.2 Methods by Solving Constrained Optimization Problems

This type of method transforms (1) to equivalent constrained optimization problems. We further classify them into two groups.

2.2.1 Optimization Problems with Smooth Constraints

We can replace $\boldsymbol{w}$ in (1) with $\boldsymbol{w}^+ - \boldsymbol{w}^-$, where $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ are both nonnegative vectors. Then, problem (1) becomes equivalent to the following bound-constrained optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w}^+, \boldsymbol{w}^-} \quad & \sum_{j=1}^{n} w_j^+ + \sum_{j=1}^{n} w_j^- + C \sum_{i=1}^{l} \xi(\boldsymbol{w}^+ - \boldsymbol{w}^-; \boldsymbol{x}_i, y_i) \\ \text{subject to} \quad & w_j^+ \geq 0,\ w_j^- \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{12}$$

The objective function and constraints of (12) are smooth, and therefore, the problem can be solved by standard bound-constrained optimization techniques. Schmidt et al. (2009) proposed ProjectionL1 to solve (12). This is an extension of the "two-metric projection" method (Gafni and Bertsekas, 1984). Limited-memory quasi Newton implementations, for example, LBFGS-B by Byrd et al. (1995) and BLMVM by Benson and Moré (2001), require function/gradient evaluations and use a limited amount of memory to approximate the Hessian matrix. Kazama and Tsujii (2003) presented an example of using BLMVM for (12). Lin and Moré's (1999) trust region Newton method (TRON) can also be applied to solve (12). In addition to function/gradient evaluations, TRON needs to calculate Hessian-vector products for faster convergence. Lin et al. (2008) applied TRON to solve L2-regularized logistic regression and showed that TRON outperforms LBFGS for document data. No previous work has applied TRON to solve L1-regularized problems, and therefore, we describe this in Section 5.1.

Koh et al. (2007) proposed an interior point method to solve L1-regularized logistic regression. They transformed (1) to the following constrained problem:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{u}} \quad & \sum_{j=1}^{n} u_j + C\sum_{i=1}^{l} \xi(\boldsymbol{w};\boldsymbol{x}_i,y_i) \\ \text{subject to} \quad & -u_j \le w_j \le u_j, \quad j=1,\ldots,n. \end{aligned} \tag{13}$$

Equation (13) can be made equivalent to (12) by setting

$$w_j^+ = \frac{u_j + w_j}{2} \text{ and } w_j^- = \frac{u_j - w_j}{2}.$$

To ensure that $\boldsymbol{w}$ and $\boldsymbol{u}$ are in the interior of the feasible region set, Koh et al. (2007) added a log barrier to the objective function of (13) and applied a Newton method.

2.2.2 Optimization Problems with Non-smooth Constraints

It is well-known that for any choice of C in (1), there exists a corresponding K such that (1) is equivalent to the following problem:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \sum_{i=1}^{l} \xi(\boldsymbol{w};\boldsymbol{x}_i,y_i) \\ \text{subject to} \quad & \|\boldsymbol{w}\|_1 \le K. \end{aligned} \tag{14}$$

See the explanation in, for example, Donoho and Tsaig (2008, Section 1.2). Notice that in (14), the constraint is not smooth at $\{\boldsymbol{w} \mid w_j = 0 \text{ for some } j\}$. However, (14) contains fewer variables and constraints than (12). Lee et al. (2006) applied the LARS algorithm described in Efron et al. (2004) to find a Newton direction at each step and then used a backtracking line search to minimize the objective value. Kivinen and Warmuth's (1997) concept of exponentiated gradient (EG) can solve (14) with additional constraints $w_j \ge 0,\ \forall j$. In a manner similar to the technique for constructing (12), we can remove the nonnegative constraints by replacing $\boldsymbol{w}$ with $\boldsymbol{w}^+ - \boldsymbol{w}^-$. If k is the iteration index, EG updates $\boldsymbol{w}$ by the following rule:

$$w_j^{k+1} = \frac{w_j^k \exp\bigl(-\eta_k \nabla_j \bigl(\sum_{i=1}^{l} \xi(\boldsymbol{w}^k;\boldsymbol{x}_i,y_i)\bigr)\bigr)}{Z_k},$$

where Z_k is a normalization term for maintaining $\|\boldsymbol{w}^k\|_1 = K,\ \forall k$ and η_k is a user-defined learning rate. Duchi et al. (2008) applied a gradient projection method to solve (14). The update rule is

$$\boldsymbol{w}^{k+1} = \arg\min_{\boldsymbol{w}} \left\{ \|\bigl(\boldsymbol{w}^k - \eta_k \nabla\bigl(\sum\nolimits_{i=1}^{l} \xi(\boldsymbol{w}^k;\boldsymbol{x}_i,y_i)\bigr)\bigr) - \boldsymbol{w}\| \;\middle|\; \|\boldsymbol{w}\|_1 \le K \right\}. \tag{15}$$

They developed a fast algorithm to project a solution to the closest point satisfying the constraint. They also considered replacing the gradient in (15) with a stochastic sub-gradient. In a manner similar to (15), Liu et al. (2009) proposed a gradient projection method called

Lassplore and carefully addressed the selection of the learning rate η_k. However, they evaluated their method only on data with no more than $2,000$ instances.[2]

Kim and Kim (2004) discussed a coordinate descent method to solve (14). They used the gradient information to select an element w_j for update. However, because of the constraint $\|\boldsymbol{w}\|_1 \leq K$, the whole vector $\boldsymbol{w}$ is normalized at each step. Thus, the setting is very different from the unconstrained situations described in Section 2.1.1. Kim et al. (2008) made further improvements to realize faster convergence.

Active set methods have also been applied to solve (14). However, in contrast to the active set methods described in Section 2.1.3, here, the sub-problem at each iteration is a constrained optimization problem. Roth (2004) studied general loss functions including logistic loss.

2.3 Other Methods for L1-regularized Logistic Regression/L2-loss SVMs

We briefly review other methods in this section.

2.3.1 Expectation Maximization

Many studies have considered Expectation Maximization (EM) frameworks to solve (1) with logistic loss (e.g., Figueiredo, 2003; Krishnapuram et al., 2004, 2005). These works find an upper-bound function $\hat{f}(\boldsymbol{w}) \geq f(\boldsymbol{w}), \forall \boldsymbol{w}$, and perform Newton steps to minimize $\hat{f}(\boldsymbol{w})$. However, as pointed out by Schmidt et al. (2009, Section 3.2), the upper-bound function $\hat{f}(\boldsymbol{w})$ may not be well-defined at some $w_i = 0$ and hence certain difficulties must be addressed.

2.3.2 Stochastic Gradient Descent

Stochastic gradient descent methods have been successfully applied to solve (1). At each iteration, the solution is updated using a randomly selected instance. These types of methods are known to efficiently generate a reasonable model, although they suffer from slow local convergence. Under an online setting, Langford et al. (2009) solved L1-regularized problems by a stochastic gradient descent method. Shalev-Shwartz and Tewari (2009) combined Langford et al.'s (2009) method with other techniques to obtain a new stochastic gradient descent method for (1).

2.3.3 Quasi Newton Methods

Andrew and Gao (2007) proposed an Orthant-Wise Limited-memory quasi Newton (OWL-QN) method. This method is extended from LBFGS (Liu and Nocedal, 1989), a limited memory quasi Newton approach for unconstrained smooth optimization. At each iteration, this method finds a sub-space without considering some dimensions with $w_j = 0$ and obtains a search direction similar to LBFGS. A constrained line search on the same sub-space is then conducted and the property $w_j^{k+1} w_j^k \geq 0$ is maintained. Yu et al. (2010) proposed a quasi Newton approach to solve non-smooth convex optimization problems. Their method can be used to improve the line search procedure in OWL-QN.

2. Although Table 1 in Liu et al. (2009) shows larger numbers, in Section 6 of the same paper, they stated that "we use a total of 2,000 samples in the following experiments."

2.3.4 Hybrid Methods

Shi et al. (2010) proposed a hybrid algorithm for (1) with logistic loss. They used a fixed-point method to identify the set $\{j \mid w_j^* = 0\}$, where $\boldsymbol{w}^*$ is an optimal solution, and then applied an interior point method to achieve fast local convergence.

2.3.5 Quadratic Approximation Followed By Coordinate Descent

Krishnapuram et al. (2005) and Friedman et al. (2010) replaced the loss term with a second-order approximation at the beginning of each iteration and then applied a cyclic coordinate descent method to minimize the quadratic function. We will show that an implementation in Friedman et al. (2010) is efficient.

2.3.6 Cutting Plane Methods

Teo et al. (2010) implemented a bundle (cutting plane) method BMRM to handle non-smooth loss functions. It includes an extension for L1 regularization.

2.3.7 Approximating L1 Regularization by L2 Regularization

Kujala et al. (2009) proposed the approximation of L1-regularized SVMs by iteratively reweighting training data and solving L2-regularized SVMs. That is, using the current w_j, they adjusted the jth feature of the training data and then trained an L2-regularized SVM in the next step.

2.3.8 Solution Path

Several works have attempted to find the "solution path" of (1). The optimal solution of (1) varies according to parameter C. It is occasionally useful to find all solutions as a function of C; see, for example, Rosset (2005), Zhao and Yu (2007), Park and Hastie (2007), and Keerthi and Shevade (2007). We do not provide details of these works because this paper focuses on the case in which a single C is used.

2.4 Strengths and Weaknesses of Existing Methods

Although this paper will compare state-of-the-art software, we discuss some known strengths and weaknesses of existing methods.

2.4.1 Convergence Speed

Optimization methods using higher-order information (e.g., quasi Newton or Newton methods) usually enjoy fast local convergence. However, they involve an expensive iteration. For example, Newton methods such as TRON or IPM need to solve a large linear system related to the Hessian matrix. In contrast, methods using or partially using the gradient information (e.g., stochastic gradient descent) have slow local convergence although they can more quickly decrease the function value in the early stage.

2.4.2 Implementation Efforts

Methods using higher-order information are usually more complicated. Newton methods need to include a solver for linear systems. In contrast, methods such as coordinate descent or stochastic gradient descent methods are easy to implement. They involve only vector operations. Other methods such as expectation maximization are intermediate in this respect.

2.4.3 Handling Large-scale Data

In some methods, the Newton step requires solving a linear system of n variables. Inverting an $n \times n$ matrix is difficult for large n. Thus, one should not use direct methods (e.g., Gaussian elimination) to solve the linear system. Instead, TRON and IPM employ conjugate gradient methods and Friedman et al. (2007) use coordinate descent methods. We observe that for existing EM implementations, many consider direct inversions, and therefore, they cannot handle large-scale data.

2.4.4 Feature Correlation

Methods that work on a block of variables at a time (e.g., decomposition methods) may be more efficient if features are almost independent; however, they may be less efficient if features are highly correlated.

2.4.5 Data Type

No method is the most suitable for all data types. A method that is efficient for one application may be slow for another. This paper is focused on document classification, and a viable method must be able to easily handle large and sparse features.

2.5 Least-square Regression for Signal Processing and Image Applications

Recently, L1-regularized problems have attracted considerable attention for signal processing and image applications. However, they differ from (1) in several aspects. First, $y_i \in R$, and therefore, a regression instead of a classification problem is considered. Second, the least-square loss function is used:

$$\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i) = (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2. \tag{16}$$

Third, in many situations, $\boldsymbol{x}_i$ are not directly available. Instead, the product between the data matrix X and a vector can be efficiently obtained through certain operators. We briefly review some of the many optimization methods for least-square regression. If we use formulation (14) with non-smooth constraints, the problem reduces to LASSO proposed by Tibshirani (1996) and some early optimization studies include, for example, Fu (1998) and Osborne et al. (2000a,b). Fu (1998) applied a cyclic coordinate descent method. For least-square loss, the minimization of the one-variable sub-problem (11) has a closed-form solution. Sardy et al. (2000) also considered coordinate descent methods, although they allowed a block of variables at each iteration. Wu and Lange (2008) considered a coordinate descent method, but used the gradient information for selecting the working variable at each

iteration. Osborne et al. (2000a) reported one of the earliest active set methods for L1-regularized problems. Roth's (2004) method for general losses (see Section 2.2.2) reduces to this method if the least-square loss is used. Friedman et al. (2007) extended Fu's coordinate descent method to find a solution path. Donoho and Tsaig (2008) also obtained a solution path. Their procedure requires solving a linear system of a matrix $X_{:,J}^T X_{:,J}$, where J is a subset of $\{1, \ldots, n\}$. Figueiredo et al. (2007) transformed the regression problem to a bound-constrained formula in (12) and applied a projected gradient method. Wright et al. (2009) proposed the iterative minimization of the sum of the L1 regularization term and a quadratic approximation of the loss term. In the quadratic approximation, they used a diagonal matrix instead of the Hessian of the loss term, and therefore, the minimization can be easily carried out. Hale et al. (2008) proposed a fixed point method to solve (1) with the least-square loss (16). Their update rule is generated from a fixed-point view; however, it is very similar to a gradient projection algorithm.

The dual of LASSO and the dual of (1) have been discussed in Osborne et al. (2000b) and Kim et al. (2007), respectively. However, thus far, there have been few optimization methods for the dual problem. Tomioka and Sugiyama (2009) proposed a dual augmented Lagrangian method for L1-regularized least-square regression that theoretically converges super-linearly.

Most optimization methods for classification discussed in the earlier sections can handle general smooth loss functions, and therefore, they can be applied to the regression problem. However, data for classification applications may be very different from those for regression applications. For example, in text classification, the numbers of instances and features are both large and data are very sparse. However, in certain signal processing applications, the number of instances may be much smaller than the number of features (i.e., $l \ll n$) and the data may be dense. Moreover, in classification, the parameter C is often selected by cross validation; however, in signal processing, the selection may be application dependent. Few optimization studies have investigated both types of applications. The interior point method for logistic regression by solving (13) has been applied to the least-square regression problem (Kim et al., 2007). Duchi et al. (2008) compared their gradient projection implementation (see Section 2.2.2) with interior point methods using both logistic and least-square losses. In Section 2.1.2, we mentioned a decomposition method CGD-GS by Yun and Toh (2011). In the same paper, Yun and Toh have also investigated the performance of CGD-GS on regression problems.

In this paper, we focus on data classification, and therefore, our conclusions may not apply to regression applications. In particular, the efficient calculation between X and a vector in some signal processing applications may afford advantages to some optimization methods.

3. Methods and Software Included for Comparison

In the rest of this paper, we compare some state-of-the-art software BBR, SCD, CGD-GS, IPM, BMRM, OWL-QN, Lassplore and GLMNET. We further develop two efficient implementations. One is a coordinate descent method (CDN) and the other is a trust region Newton method (TRON). These packages are selected because of two reasons. First, they are pub-

licly available. Second, they are able to handle large and sparse data sets. We categorize these packages into three groups:

- decomposition methods,
- methods by solving constrained optimization problems,
- other methods,

and describe their algorithms in Sections 4–6. The comparison results are described in Sections 8. Note that classification (logistic regression and L2-loss SVMs) is our focus, and therefore, software for regression problems using the least-square loss (16) is not considered.

4. Decomposition Methods

This section discusses three methods sequentially or randomly choosing variables for update, and one method using gradient information for the working set selection.

4.1 Cyclic Coordinate Descent Methods

From the current solution $\boldsymbol{w}^k$, a coordinate descent method updates one variable at a time to generate $\boldsymbol{w}^{k,j} \in R^n,\ j = 1, \ldots, n+1$, such that $\boldsymbol{w}^{k,1} = \boldsymbol{w}^k$, $\boldsymbol{w}^{k,n+1} = \boldsymbol{w}^{k+1}$, and

$$\boldsymbol{w}^{k,j} = \left[w_1^{k+1}, \ldots, w_{j-1}^{k+1}, w_j^k, \ldots, w_n^k\right]^T \text{ for } j = 2, \ldots, n.$$

To update $\boldsymbol{w}^{k,j}$ to $\boldsymbol{w}^{k,j+1}$, the following one-variable optimization problem is solved:

$$\min_z \quad g_j(z) = |w_j^{k,j} + z| - |w_j^{k,j}| + L_j(z; \boldsymbol{w}^{k,j}) - L_j(0; \boldsymbol{w}^{k,j}), \tag{17}$$

where

$$L_j(z; \boldsymbol{w}^{k,j}) \equiv L(\boldsymbol{w}^{k,j} + z\boldsymbol{e}_j).$$

For simplicity, hereafter, we use $L_j(z)$ in most places. If the solution of (17) is d, then we update the jth element by

$$w_j^{k,j+1} = w_j^{k,j} + d.$$

Typically, a coordinate descent method sequentially goes through all variables and then repeats the same process; see the framework in Algorithm 1. We refer to the process of updating every n elements (i.e., from $\boldsymbol{w}^k$ to $\boldsymbol{w}^{k+1}$) as an outer iteration and the step of updating one element (i.e., from $\boldsymbol{w}^{k,j}$ to $\boldsymbol{w}^{k,j+1}$) as an inner iteration. Practically, we only approximately solve (17), where several approaches are discussed below.

Using the same way to obtain the optimality condition in (6), for the one-variable function $g_j(z)$, we have that $z = 0$ is optimal for (17) if and only if

$$\begin{cases} L_j'(0) + 1 = 0 & \text{if } w_j^{k,j} > 0, \\ L_j'(0) - 1 = 0 & \text{if } w_j^{k,j} < 0, \\ -1 \le L_j'(0) \le 1 & \text{if } w_j^{k,j} = 0. \end{cases} \tag{18}$$

The optimality condition at $z = 0$ shows if modifying $w_j^{k,j}$ may decrease the function value or not.

Algorithm 1 A framework of coordinate descent methods

1. Given $\boldsymbol{w}^1$.
2. For $k = 1, 2, 3, \ldots$ (outer iterations)
   (a) $\boldsymbol{w}^{k,1} = \boldsymbol{w}^k$.
   (b) For $j = 1, 2, \ldots, n$ (inner iterations)
      - Find d by solving the sub-problem (17) exactly or approximately.
      - $\boldsymbol{w}^{k,j+1} = \boldsymbol{w}^{k,j} + d\boldsymbol{e}_j$.
   (c) $\boldsymbol{w}^{k+1} = \boldsymbol{w}^{k,n+1}$.

4.1.1 BBR

Genkin et al. (2007) propose a coordinate descent method BBR for (1) and (5) with logistic loss. This method is extended from Zhang and Oles (2001) for L2-regularized logistic regression. At each inner iteration, BBR approximately solves the sub-problem (17) by a trust region Newton method. With the trust region Δ_j, it requires the step z to satisfy

$$|z| \le \Delta_j \quad \text{and} \quad w_j^{k,j} + z \begin{cases} \ge 0 & \text{if } w_j^{k,j} > 0, \\ \le 0 & \text{if } w_j^{k,j} < 0. \end{cases} \tag{19}$$

The first constraint confines the step size to be in the trust region and Δ_j is updated at the end of each inner iteration. Due to the non-differentiable point at $z = -w_j^{k,j}$, the second constraint ensures that the function is differentiable in the search space.

To approximately solve (17), BBR minimizes a quadratic function upper-bounding the function $g_j(z)$ in the trust region. Though $g_j(z)$ is not differentiable, by considering both cases of $w_j^{k,j} > 0$ and $w_j^{k,j} < 0$, we obtain the following form:

$$g_j(z) = g_j(0) + g_j'(0)z + \frac{1}{2}g_j''(\eta z)z^2, \tag{20}$$

where $0 < \eta < 1$,

$$g_j'(0) \equiv \begin{cases} L_j'(0) + 1 & \text{if } w_j^{k,j} > 0, \\ L_j'(0) - 1 & \text{if } w_j^{k,j} < 0, \end{cases} \quad \text{and} \quad g_j''(\eta z) \equiv L_j''(\eta z). \tag{21}$$

Notice that when $w_j^{k,j} = 0$, $g_j(z)$ is non-differentiable at $z = 0$ and $g_j'(0)$ is not well-defined. We will discuss this situation later. BBR finds an upper bound U_j of $g_j''(z)$ such that

$$U_j \ge g_j''(z), \ \forall |z| \le \Delta_j.$$

Then $\hat{g}_j(z)$ is an upper-bound function of $g_j(z)$:

$$\hat{g}_j(z) \equiv g_j(0) + g_j'(0)z + \frac{1}{2}U_j z^2.$$

Any step z satisfying $\hat{g}_j(z) < \hat{g}_j(0)$ leads to

$$g_j(z) - g_j(0) = g_j(z) - \hat{g}_j(0) \le \hat{g}_j(z) - \hat{g}_j(0) < 0,$$

Algorithm 2 BBR: Approximately solving the sub-problem by a trust region method

1. Given $\boldsymbol{w}^{k,j}$ and Δ_j.
2. Calculate U_j by (22).
3. Find a step d by (23).
4. Update Δ_j by $\Delta_j \leftarrow \max(2|d|, \Delta_j/2)$.

so the function value is decreased. If logistic loss (2) is used, BBR suggests setting U_j as

$$U_j \equiv C \sum_{i=1}^{l} x_{ij}^2 F\big(y_i(\boldsymbol{w}^{k,j})^T \boldsymbol{x}_i, \Delta_j |x_{ij}|\big), \tag{22}$$

where

$$F(r,\delta) = \begin{cases} 0.25 & \text{if } |r| \le \delta, \\ \frac{1}{2+e^{(|r|-\delta)}+e^{(\delta-|r|)}} & \text{otherwise.} \end{cases}$$

If $w_j^{k,j} \neq 0$, BBR minimizes $\hat{g}_j(z)$ under the constraints (19) to obtain

$$d = \min\left(\max\big(P(-\frac{g_j'(0)}{U_j}, w_j^{k,j}), -\Delta_j\big), \Delta_j\right), \tag{23}$$

where

$$P(z,w) \equiv \begin{cases} z & \text{if } \operatorname{sgn}(w+z) = \operatorname{sgn}(w), \\ -w & \text{otherwise.} \end{cases}$$

Now consider the case of $w_j^{k,j} = 0$, where $g_j'(0)$ is not well-defined at this point. If $L_j'(0)+1 < 0$, by defining $g_j'(0) \equiv L_j'(0)+1$, any $0 < z \le -g_j'(0)/U_j$ gives a smaller $\hat{g}_j(z)$ than $\hat{g}_j(0)$. We thus obtain the new point by mapping $-g_j'(0)/U_j$ back to the trust region. The situation for $L_j'(0) - 1 > 0$ is similar. We do not need to handle the situation $-1 \le L_j'(0) \le 1$ as (18) and $w_j^{k,j} = 0$ indicate that $z = 0$ is the minimum of $g_j(z)$.

The procedure of BBR to approximately solve (17) is described in Algorithm 2. The major cost is on calculating $g_j'(0)$ and U_j. For logistic loss, $L_j'(0)$ needed for calculating $g_j'(0)$ is

$$L_j'(0) = C \sum_{i=1}^{l} y_i x_{ij} \left(\tau(y_i(\boldsymbol{w}^{k,j})^T \boldsymbol{x}_i) - 1\right), \tag{24}$$

where $\tau(\cdot)$ is defined in (9). From (22) and (24), the most expensive operation is on obtaining $\boldsymbol{w}^T\boldsymbol{x}_i$, $\forall i$. A common trick for saving the running time is to store $\boldsymbol{w}^T\boldsymbol{x}_i$, $\forall i$ and update them according to

$$\boldsymbol{w}^T\boldsymbol{x}_i \leftarrow \boldsymbol{w}^T\boldsymbol{x}_i + d \cdot x_{ij}. \tag{25}$$

If $\boldsymbol{w}^T\boldsymbol{x}_i$, $\forall i$ are available, both (22) and (24) need $O(l)$ operations. Because maintaining $\boldsymbol{w}^T\boldsymbol{x}_i$ via (25) also takes $O(l)$, the cost per inner iteration is $O(l)$.

Unfortunately, there is no convergence proof yet for the method BBR.

4.1.2 Coordinate Descent Method Using One-dimensional Newton Directions (CDN)

BBR replaces the second derivative term in (20) with an upper bound U_j. If we keep using $g_j''(0)$ and obtain the one-dimensional Newton direction at $z = 0$, the local convergence may be faster. This issue has been studied in L2-regularized logistic regression/SVMs, where BBR reduces to the approach by Zhang and Oles (2001), and Chang et al. (2008) showed that a coordinate descent method using one-dimensional Newton directions is faster. Here, we extend Chang et al.'s method for L1-regularized problems. The new approach, referred to as CDN, is expected to be faster than BBR following a similar reason.

A Newton direction is obtained from minimizing the second-order approximation, but $g_j(z)$ is not differentiable due to the L1-regularization term. Thus, we consider only the second-order approximation of the loss term $L_j(z)$ and solve

$$\min_z \quad |w_j^{k,j} + z| - |w_j^{k,j}| + L_j'(0)z + \frac{1}{2}L_j''(0)z^2. \tag{26}$$

This problem can be reduced to a form commonly referred to as "soft-thresholding" in signal processing. We show in Appendix B that (26) has the following closed-form solution:

$$d = \begin{cases} -\frac{L_j'(0)+1}{L_j''(0)} & \text{if } L_j'(0) + 1 \le L_j''(0)w_j^{k,j}, \\ -\frac{L_j'(0)-1}{L_j''(0)} & \text{if } L_j'(0) - 1 \ge L_j''(0)w_j^{k,j}, \\ -w_j^{k,j} & \text{otherwise.} \end{cases} \tag{27}$$

Because (26) is only a quadratic approximation of $f(\boldsymbol{w}^{k,j} + z\boldsymbol{e}_j) - f(\boldsymbol{w}^{k,j})$, the direction d may not ensure the decrease of the function value. For the convergence, Chang et al. (2008) consider a line search procedure to find $\lambda \in (0, 1)$ such that the step λd satisfies the sufficient decrease condition:

$$f(\boldsymbol{w}^{k,j} + \lambda d\boldsymbol{e}_j) - f(\boldsymbol{w}^{k,j}) = g_j(\lambda d) - g_j(0) \le -\sigma(\lambda d)^2,$$

where σ is any constant in $(0, 1)$. However, as pointed out by Tseng and Yun (2009), this condition may not be suitable here due to the non-smooth regularization term $\|\boldsymbol{w}\|_1$. We follow Tseng and Yun (2009) to use a modified condition:

$$g_j(\lambda d) - g_j(0) \le \sigma\lambda \left(L_j'(0)d + |w_j^{k,j} + d| - |w_j^{k,j}| \right). \tag{28}$$

To find λ, CDN adopts a backtrack line search to sequentially check $\lambda = 1,\ \beta,\ \beta^2,\ \ldots$, where $\beta \in (0, 1)$, until λd satisfies (28). A description of how CDN approximately solves the sub-problem (17) is in Algorithm 3.

In Appendix D, we explain that Algorithm 3 falls into a class of Gauss-Seidel coordinate descent methods in Tseng and Yun (2009). By showing that all required assumptions are satisfied, we can directly enjoy some nice theoretical properties. First, following Lemma 5 in Tseng and Yun (2009), the line search procedure stops in a finite number of steps. Second, for (1) with logistic loss, any limit point of $\{\boldsymbol{w}^k\}$ is an optimum. Further, if the loss function $L(\boldsymbol{w})$ is strictly convex, the algorithm converges at least linearly. The proof is in the supplementary file.

Algorithm 3 CDN: Approximately solving the sub-problem by Newton directions with line search

1. Given $\boldsymbol{w}^{k,j}$. Choose $\beta \in (0,1)$.
2. Calculate the Newton direction d by (27).
3. Compute $\lambda = \max\{1, \beta, \beta^2, \dots\}$ such that λd satisfies (28).

We discuss the computational cost. To obtain the sub-problem (26), we must calculate $L_j'(0)$ and $L_j''(0)$. For logistic loss, $L_j'(0)$ is shown in (24) and

$$L_j''(0) = C \sum_{i=1}^{l} x_{ij}^2 \left(\tau(y_i(\boldsymbol{w}^{k,j})^T \boldsymbol{x}_i)\right) \left(1 - \tau(y_i(\boldsymbol{w}^{k,j})^T \boldsymbol{x}_i)\right). \tag{29}$$

Similar to the situation in BBR, calculating $\boldsymbol{w}^T \boldsymbol{x}_i,\ \forall i$ is the major cost here. We can apply the same trick in (25) to maintain $\boldsymbol{w}^T \boldsymbol{x}_i,\ \forall i$. In our implementation, we maintain $e^{\boldsymbol{w}^T \boldsymbol{x}_i}$ instead:

$$e^{\boldsymbol{w}^T \boldsymbol{x}_i} \leftarrow e^{\boldsymbol{w}^T \boldsymbol{x}_i} \cdot e^{\lambda d x_{ij}}. \tag{30}$$

The line search procedure needs to calculate $g_j(\lambda d)$. From (2), the main cost is still on obtaining $(\boldsymbol{w} + \lambda d \boldsymbol{e}_j)^T \boldsymbol{x}_i,\ \forall i$, so the trick in (30) is again useful. If $e^{\boldsymbol{w}^T \boldsymbol{x}_i},\ \forall i$ are available, from (24), (29) and (2), the cost per inner iteration is

$$(1 + \#\text{ line search steps}) \times O(l).$$

To reduce the cost for line search, Chang et al. (2008) obtain a function $\hat{g}_j(\cdot)$ satisfying $\hat{g}_j(\lambda d) > g_j(\lambda d)$ and check $\hat{g}_j(\lambda d) - g_j(0)$ in (28). Calculating $\hat{g}_j(\lambda d)$ is much easier than $g_j(\lambda d)$, so we may avoid calculating the last $g_j(\lambda d)$ in the line search procedure. In some iterations, $\lambda = 1$ already satisfies (28), and therefore, this trick makes the cost of the line search procedure negligible. We do not show details here; however, derivations can be found in Fan et al. (2008, Appendix G).

Next we describe two implementation techniques to improve the convergence speed. The first one is to use a random permutation of sub-problems. In Algorithm 1, we cyclically consider variables to form one-variable sub-problems. Chang et al. (2008) show that solving sub-problems in a random order may lead to faster convergence. That is, at the kth iteration, we randomly permute $\{1, 2, \dots, n\}$ to $\{\pi_k(1), \pi_k(2), \dots, \pi_k(n)\}$ and update the elements of $\boldsymbol{w}$ in the order of $\{w_{\pi_k(1)}, w_{\pi_k(2)}, \dots, w_{\pi_k(n)}\}$.

The second implementation technique is shrinking. Past works such as Hsieh et al. (2008), Joachims (1998), and Krishnapuram et al. (2005, Section 3.5) heuristically remove some variables to obtain a smaller optimization problem. If w_j has remained zero at many iterations, it is very possible that the optimal w_j^* is also zero. Therefore, we can remove such variables. Our shrinking implementation is related to that in the software LIBSVM (Chang and Lin, 2011). From the optimality condition (6),

$$-1 < \nabla_j L(\boldsymbol{w}^*) < 1 \quad \text{implies} \quad w_j^* = 0. \tag{31}$$

We prove in Appendix C the following theorem:

Theorem 1 *Assume $\{\boldsymbol{w}^k\}$ globally converges to $\boldsymbol{w}^*$. If $-1 < \nabla_j L(\boldsymbol{w}^*) < 1$, then there is K_j such that for all $k \geq K_j$,*

$$-1 < \nabla_j L(\boldsymbol{w}^{k,j}) < 1 \quad \textit{and} \quad w_j^{k,j} = 0.$$

Using this theorem, we design the following shrinking strategy. Before updating $w_j^{k,j}$ via approximately solving the sub-problem (17), if

$$w_j^{k,j} = 0 \quad \text{and} \quad -1 + M^{k-1} < \nabla_j L(\boldsymbol{w}^{k,j}) < 1 - M^{k-1}, \tag{32}$$

we conjecture that $w_j^{k,j}$ may remain zero and hence remove it for optimization. We choose

$$M^{k-1} \equiv \frac{\max_j v_j}{l},$$

where

$$v_j \equiv \begin{cases} |\nabla_j L(\boldsymbol{w}^{k-1,j}) + 1| & \text{if } w_j^{k-1,j} > 0, \\ |\nabla_j L(\boldsymbol{w}^{k-1,j}) - 1| & \text{if } w_j^{k-1,j} < 0, \\ \max\left(\nabla_j L(\boldsymbol{w}^{k-1,j}) - 1,\ -1 - \nabla_j L(\boldsymbol{w}^{k-1,j}),\ 0\right) & \text{if } w_j^{k-1,j} = 0, \end{cases}$$

From (6), $v_j, j = 1, \ldots, n$ measure the violation of the optimality condition at the $(k-1)$st iteration. The value M^{k-1} reflects how aggressive we remove variables. It is large in the beginning, but approaches zero in the end.

The shrinking implementation introduces little extra cost. When updating the jth component at the kth iteration, we calculate

$$\nabla_j L(\boldsymbol{w}^{k,j}) = L_j'(0; \boldsymbol{w}^{k,j})$$

for the direction d in (27), regardless of implementing shrinking or not. Thus, $\nabla_j L(\boldsymbol{w}^{k,j})$ needed for checking (32) is already available. Moreover, it can be used to calculate v_j and M^k, which are needed for the next iteration.

4.1.3 Stochastic Coordinate Descent Method (SCD)

Shalev-Shwartz and Tewari (2009) propose a stochastic coordinate descent method (SCD) to solve the bound-constrained problem in (12). At the kth iteration, SCD randomly chooses a working variable from $\{w_1^+, \ldots, w_n^+, w_1^-, \ldots, w_n^-\}$. The one-variable sub-problem is

$$\min_z \quad g_j(z) \equiv z + L_j(z; \boldsymbol{w}^{k,+} - \boldsymbol{w}^{k,-}) - L_j(0; \boldsymbol{w}^{k,+} - \boldsymbol{w}^{k,-}),$$

subject to the non-negativity constraint

$$w_j^{k,+} + z \geq 0 \quad \text{or} \quad w_j^{k,-} + z \geq 0,$$

according to whether w_j^+ or w_j^- is the working variable. SCD considers a second-order approximation similar to BBR:

$$\hat{g}_j(z) = g_j(0) + g_j'(0)z + \frac{1}{2}U_j z^2, \tag{33}$$

Algorithm 4 SCD for L1-regularized logistic regression

1. Given $(\boldsymbol{w}^+, \boldsymbol{w}^-)$ and $U_j > 0$.
2. While $(\boldsymbol{w}^+, \boldsymbol{w}^-)$ is not optimal for (12)
   (a) Select an element from $\{w_1^+, \ldots, w_n^+, w_1^-, \ldots, w_n^-\}$.
   (b) Update w_j^+ or w_j^- by (36)–(37).

where

$$g_j'(0) = \begin{cases} 1 + L_j'(0) & \text{for } w_j^+ \\ 1 - L_j'(0) & \text{for } w_j^- \end{cases} \quad \text{and} \quad U_j \geq g_j''(z),\ \forall z.$$

BBR considers U_j to be an upper bound of $g_j''(z)$ only in the trust region, while SCD finds a global upper bound of $g_j''(z)$. For logistic regression, following (9) and (29), we have $\tau(\cdot)\,(1 - \tau(\cdot)) \leq 0.25$ and

$$U_j = 0.25C \sum_{i=1}^{l} x_{ij}^2 \geq g_j''(z),\ \forall z. \tag{34}$$

Shalev-Shwartz and Tewari (2009) assume $-1 \leq x_{ij} \leq 1,\ \forall i, j$, so a simpler upper bound is

$$U_j = 0.25Cl. \tag{35}$$

Using the direction obtained by minimizing (33) and taking the non-negativity into consideration, SCD updates $\boldsymbol{w}$ by the following way:

If w_j^+ is selected

$$w_j^+ \leftarrow w_j^+ + \max(-w_j^+,\ -\frac{1 + L_j'(0)}{U_j}) \tag{36}$$

Else

$$w_j^- \leftarrow w_j^- + \max(-w_j^-,\ -\frac{1 - L_j'(0)}{U_j}) \tag{37}$$

A description of SCD is in Algorithm 4.

4.2 CGD-GS: a Decomposition Method Using Gradients for Selecting Variables

Instead of updating one variable at a time, some decomposition methods choose a larger working set $J \subset N \equiv \{1, \ldots, n\}$. We discuss a Gauss-Southwell method for selecting J (Tseng and Yun, 2009; Yun and Toh, 2011). This method, referred to as CGD-GS, can handle any smooth loss function including $\xi_{\log}$.

Following the principle of decomposition methods, if $\boldsymbol{w}^k$ is the current solution and J is the set of working variables, one should solve the following sub-problem:

$$\begin{aligned} \min_{\boldsymbol{d}} \quad & L(\boldsymbol{w}^k + \boldsymbol{d}) - L(\boldsymbol{w}^k) + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1 \\ \text{subject to} \quad & d_j = 0,\ \forall j \notin J. \end{aligned}$$

Because it is still difficult to solve this sub-problem, CGD-GS considers a quadratic approximation of the loss term:

$$\begin{aligned} \min_{\boldsymbol{d}} \quad & q_k(\boldsymbol{d}) \equiv \nabla L(\boldsymbol{w}^k)^T\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T H\boldsymbol{d} + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1 \\ \text{subject to} \quad & d_j = 0,\ \forall j \notin J, \end{aligned} \tag{38}$$

where H is either $\nabla^2 L(\boldsymbol{w}^k)$ or its approximation. To ensure the convergence, CGD-GS conducts a backtrack line search to find λ such that $\lambda\boldsymbol{d}$ satisfies

$$f(\boldsymbol{w}^k + \lambda\boldsymbol{d}) - f(\boldsymbol{w}^k) \le \sigma\lambda\left(\nabla L(\boldsymbol{w}^k)^T\boldsymbol{d} + \gamma\boldsymbol{d}^T H\boldsymbol{d} + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1\right), \tag{39}$$

where $0 < \sigma < 1$ and $0 \le \gamma < 1$. This condition is the same as (28) if $\gamma = 0$ and J contains only one element. Tseng and Yun (2009) used (39) for both the cyclic selection (Gauss-Seidel) or the selection using gradient information (Gauss-Southwell).

For selecting J using gradients, Tseng and Yun (2009) proposed two possible ways.[3] The first one, referred to as the Gauss-Southwell-r rule, requires J to satisfy

$$\|\boldsymbol{d}(J)\|_\infty \ge v\|\boldsymbol{d}(N)\|_\infty, \tag{40}$$

where $v \in (0,1)$ is a constant and $\boldsymbol{d}(J)$ and $\boldsymbol{d}(N)$ are the solution of (38) by considering J and N as the working set, respectively. This condition connects the directions of solving sub-problems using a subset and a full set. The other condition for selecting J is

$$q_k(\boldsymbol{d}(J)) \le v \cdot q_k(\boldsymbol{d}(N)), \tag{41}$$

where $v \in (0,1)$ and $q_k(\boldsymbol{d})$ is defined in (38). This condition is referred to as the Gauss-Southwell-q rule. Algorithm 5 summarizes the procedure.

The remaining issues are how to solve the sub-problem (38) and how to obtain J satisfying (40) or (41). The CGD-GS implementation considers a diagonal matrix with positive entries as H. For example, $H_{jj} = \max(\nabla^2_{jj} L(\boldsymbol{w}^k), \epsilon)$, where ϵ is a small positive value. Then, the sub-problem becomes $|J|$ separable one-variable problems like (26). Each one-variable problem has a simple closed-form solution. Further, it is easy to find indices satisfying (40) or (41). For example, the rule (40) becomes to find the larger elements of $\boldsymbol{d}(N)$. Tseng and Yun (2009) proved that any limit point of $\{\boldsymbol{w}^k\}$ is an optimum of (1).

We discuss the computational cost for logistic regression. If H is diagonal, then solving (38) takes only $O(|J|)$ operations. Constructing (38) is more expensive because we need to calculate

$$\nabla L(\boldsymbol{w}) = C\sum_{i=1}^{l}\big(\tau(y_i\boldsymbol{w}^T\boldsymbol{x}_i) - 1\big)y_i\boldsymbol{x}_i. \tag{42}$$

Yun and Toh (2011) apply a trick similar to (30) and maintain $e^{\boldsymbol{w}^T\boldsymbol{x}_i},\ \forall i$, but the gradient calculation still needs $O(ln)$. This high cost may not pay off, and therefore, Hsieh et al. (2008) favor decomposition methods without maintaining the gradient. Finding the working set J is another potentially expensive step, but it is cheap for a diagonal H.

3. Note that (40) indirectly uses gradient information. If the 1-norm term is not considered and H is diagonal, then $\boldsymbol{d}(N)$ is a "scaled" gradient with $d_j(N) = -\nabla_j L(\boldsymbol{w}^k)/H_{jj}$. Thus, (40) selects indices with larger scaled gradient values.

Algorithm 5 CGD-GS for L1-regularized logistic regression

1. Given $\boldsymbol{w}^1$. Choose (40) or (41) as the strategy for selecting working sets. Given $0 < \beta, \sigma < 1$ and $0 \le \gamma < 1$.
2. For $k = 1, 2, 3, \ldots$
   - Choose an H and a working set J.
   - Get $\boldsymbol{d}^k$ by solving the sub-problem (38).
   - Compute $\lambda = \max\{1, \beta, \beta^2, \ldots\}$ such that $\lambda \boldsymbol{d}^k$ satisfies (39).
   - $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \lambda \boldsymbol{d}^k$.

5. Methods by Solving Constrained Optimization Problems

Section 2.2 lists several methods for L1-regularized classification by solving the bound-constrained problems (12), (13), and (14). In this section, we discuss TRON, IPM, and Lassplore in detail.

5.1 A Trust Region Newton Method (TRON)

We apply the trust region Newton method in Lin and Moré (1999) to solve (12). A previous study of this method for L1-regularized logistic regression is by Lee (2008). For convenience, in this section, we slightly abuse the notation by redefining

$$\boldsymbol{w} \equiv \begin{bmatrix} \boldsymbol{w}^+ \\ \boldsymbol{w}^- \end{bmatrix} \in R^{2n}. \tag{43}$$

Then, problem (12) can be written in a general form of bounded-constrained problems:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \bar{f}(\boldsymbol{w}) \\ \text{subject to} \quad & \boldsymbol{w} \in \Omega \equiv \{\boldsymbol{w} \mid l_j \le w_j \le u_j, \forall j\}, \end{aligned}$$

where $\bar{f}(\boldsymbol{w})$ denotes the objective function of (12), and $\boldsymbol{l}$ and $\boldsymbol{u}$ are lower and upper bounds, respectively.

At the kth iteration of the trust region Newton method, we have an iterate $\boldsymbol{w}^k$, a size Δ_k of the trust region, and a quadratic model

$$q_k(\boldsymbol{d}) \equiv \frac{1}{2}\boldsymbol{d}^T \nabla^2 \bar{f}(\boldsymbol{w}^k)\boldsymbol{d} + \nabla \bar{f}(\boldsymbol{w}^k)^T \boldsymbol{d}$$

to approximate the value $\bar{f}(\boldsymbol{w}^k + \boldsymbol{d}) - \bar{f}(\boldsymbol{w}^k)$. Next, we find a step $\boldsymbol{d}^k$ by approximately solving the following trust region problem

$$\begin{aligned} \min_{\boldsymbol{d}} \quad & q_k(\boldsymbol{d}) \\ \text{subject to} \quad & \|\boldsymbol{d}\| \le \Delta_k,\ \boldsymbol{w}^k + \boldsymbol{d} \in \Omega. \end{aligned} \tag{44}$$

We then update $\boldsymbol{w}^k$ and Δ_k by checking the ratio

$$\rho_k = \frac{\bar{f}(\boldsymbol{w}^k + \boldsymbol{d}^k) - \bar{f}(\boldsymbol{w}^k)}{q_k(\boldsymbol{d}^k)} \tag{45}$$

Algorithm 6 A trust region method for L1-regularized logistic regression

1. Given $\boldsymbol{w}^1$.
2. For $k = 1, 2, 3, \ldots$ (outer iterations)
   - Approximately solve (44) and obtain a direction $\boldsymbol{d}^k$; see Algorithm 7.
   - Compute ρ_k via (45).
   - Update $\boldsymbol{w}^k$ to $\boldsymbol{w}^{k+1}$ according to (46) and update Δ_k to Δ_{k+1}.

Algorithm 7 TRON: Finding a direction $\boldsymbol{d}^k$ by approximately solving (44)

1. Given $0 < \epsilon < 1$. Find the Cauchy step $\boldsymbol{d}^{k,C}$ and the Cauchy point
$$\boldsymbol{w}^{k,1} = \boldsymbol{w}^k + \boldsymbol{d}^{k,C} \quad \text{and} \quad \boldsymbol{d}^{k,1} = \boldsymbol{d}^{k,C}.$$
2. For $t = 1, \ldots, 2n+1$ (inner iterations)
   - Find F_t and B_t (free/bounded sets) at $\boldsymbol{w}^{k,t}$ by (50).
   - If $F_t = \emptyset$, then stop and return $\boldsymbol{d}^k = \boldsymbol{w}^{k,t} - \boldsymbol{w}^k$.
   - Approximately solve
$$\begin{aligned} \min_{\boldsymbol{v}_{F_t}} \quad & q_k(\boldsymbol{d}^{k,t} + \boldsymbol{v}) \\ \text{subject to} \quad & \|\boldsymbol{d}^{k,t} + \boldsymbol{v}\| \le \Delta_k,\ \boldsymbol{v}_{B_t} = \boldsymbol{0}, \end{aligned}$$
     by Conjugate Gradient (CG) methods. Denote the solution as $\boldsymbol{v}^{k,t}$.
   - Projected line search on $\boldsymbol{w}^{k,t} + \lambda \boldsymbol{v}^{k,t}$ to obtain $\boldsymbol{w}^{k,t+1}$ and $\boldsymbol{d}^{k,t+1}$; see Equation (55). We ensure that $F_t \subset F_{t+1}$ and $|F_t| < |F_{t+1}|$.
   - If one of the following situations occurs:
$$\|\nabla q_k(\boldsymbol{d}^{k,t+1})_{F_t}\| \le \epsilon \|\nabla \bar{f}(\boldsymbol{w}^k)_{F_t}\|,$$
     or CG abnormally stops (explained in text), then stop and return
$$\boldsymbol{d}^k = \boldsymbol{w}^{k,t+1} - \boldsymbol{w}^k.$$

of the actual reduction in the function to the predicted reduction in the quadratic model. The direction $\boldsymbol{d}^k$ is accepted if ρ_k is large enough:

$$\boldsymbol{w}^{k+1} = \begin{cases} \boldsymbol{w}^k + \boldsymbol{d}^k & \text{if } \rho_k > \eta_0, \\ \boldsymbol{w}^k & \text{if } \rho_k \le \eta_0, \end{cases} \tag{46}$$

where $\eta_0 > 0$ is a pre-specified value. The size Δ_k of the trust region is then updated according to the reduction of the function value. If the reduction is significant, then Δ_k is enlarged. Otherwise, we reduce Δ_k. More details can be found in Lin and Moré (1999). The framework of our trust region method is given in Algorithm 6. Earlier works applying trust region methods for L2-regularized problems include, for example, Lin et al. (2008). The algorithm here is more complicated due to the bound constraints.

5.1.1 Cauchy Point

A challenge for minimizing bound-constrained problems is to quickly identify bounded components at an optimal solution (i.e., components which are upper- or lower-bounded). Because each w_j can be either bounded or free, the number of combinations is exponential. One commonly used approach takes the negative gradient direction to obtain a new point and projects it back to the feasible region Ω. With a proper line search to ensure the reduction of the quadratic model $q_k(\boldsymbol{d})$, not only do we effectively guess the set of bounded components at an optimum, but also the convergence is guaranteed. To be more precise, we find a step size $\lambda > 0$ so that

$$q_k(\boldsymbol{d}^{k,C}) \leq q_k(\mathbf{0}) + \sigma \nabla q_k(\mathbf{0})^T \boldsymbol{d}^{k,C} \quad \text{and} \quad \|\boldsymbol{d}^{k,C}\| \leq \Delta_k, \tag{47}$$

where

$$\boldsymbol{d}^{k,C} = P[\boldsymbol{w}^k - \lambda \nabla \bar{f}(\boldsymbol{w}^k)] - \boldsymbol{w}^k \tag{48}$$

is called the Cauchy step in bound-constrained optimization and $\sigma \in (0, 1/2)$ is a constant. The projection operator $P[\cdot]$ maps $\boldsymbol{w}^k - \lambda \nabla \bar{f}(\boldsymbol{w}^k)$ back to the feasible region Ω:

$$P[w_j] = \min\left(u_j,\ \max(w_j, l_j)\right), \tag{49}$$

so some components become bounded. Although the negative gradient direction is projected in (48), the resulting direction is still a descending one (i.e., $\nabla q_k(\mathbf{0})^T \boldsymbol{d}^{k,C} < 0$). Hence, one can always find a small $\lambda > 0$ such that (47) is satisfied. The point $\boldsymbol{w}^{k,C} \equiv \boldsymbol{w}^k + \boldsymbol{d}^{k,C}$ is referred to as the Cauchy point.

5.1.2 Newton Direction

Gradient descent methods suffer from slow convergence, so in (48) we should have used the Newton direction. However, the second-order information is accurate only if there are no bound constraints. Lin and Moré (1999) propose using Newton directions on the subspace of the Cauchy point's free components. Recall that we find the Cauchy point to predict the bounded elements at an optimum. We obtain the free/bounded sets at the Cauchy point

$$F \equiv F(\boldsymbol{w}^{k,C}) = \{j \mid l_j < w_j^{k,C} < u_j\} \quad \text{and} \quad B \equiv B(\boldsymbol{w}^{k,C}) = \{j \mid j \notin F\}, \tag{50}$$

and find a Newton direction on the space F by solving

$$\begin{aligned} \min_{\boldsymbol{v}_F} \quad & q_k(\boldsymbol{d}^{k,C} + \boldsymbol{v}) \\ \text{subject to} \quad & \|\boldsymbol{d}^{k,C} + \boldsymbol{v}\| \leq \Delta_k,\ \boldsymbol{v}_B = \mathbf{0}. \end{aligned} \tag{51}$$

If the free set at the Cauchy point is close to that at an optimum, using a Newton direction on this sub-space leads to fast convergence.

Because (51) does not enforce the feasibility of $\boldsymbol{w}^{k,C} + \boldsymbol{v}$, one needs a projected line search procedure similar to (47)–(48). Details are shown later in (55). The resulting point may contain more bounded components than the Cauchy point. In this situation, Lin and Moré (1999) continue to minimize a quadratic approximation on the new sub-space. They thus generate inner iterates $\boldsymbol{w}^{k,1} = \boldsymbol{w}^{k,C}, \boldsymbol{w}^{k,2}, \boldsymbol{w}^{k,3}, \ldots$, until that the free/bounded sets do

not change. If m inner iterations are taken, then the direction $\boldsymbol{d}^k$ for the kth trust region iteration in Algorithm 6 is

$$\boldsymbol{d}^k = \boldsymbol{w}^{k,m+1} - \boldsymbol{w}^k.$$

Details of our procedure are described in Algorithm 7. Because each inner iteration enlarges the bounded set, the number of inner iterations is bounded by $2n$, the number of variables. In practice, very few inner iterations (one or two) are taken. Another reason to take inner iterations is for the quadratic convergence proof in Lin and Moré (1999).

Let t be the inner-iteration index and B_t, F_t be bounded/free sets at $\boldsymbol{w}^{k,t}$. With $\boldsymbol{v}_{B_t} = \boldsymbol{0}$,

$$q_k(\boldsymbol{d}^{k,t} + \boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}_{F_t}^T \nabla^2 \bar{f}(\boldsymbol{w}^k)_{F_t,F_t} \boldsymbol{v}_{F_t} + \nabla q_k(\boldsymbol{d}^{k,t})_{F_t}^T \boldsymbol{v}_{F_t} + q_k(\boldsymbol{d}^{k,t}), \tag{52}$$

so minimizing $q_k(\boldsymbol{d}^{k,t} + \boldsymbol{v})$ is equivalent to solving the following linear system

$$\nabla^2 \bar{f}(\boldsymbol{w}^k)_{F_t,F_t} \boldsymbol{v}_{F_t} = -\nabla q_k(\boldsymbol{d}^{k,t})_{F_t}. \tag{53}$$

To solve (53), we conduct conjugate gradient (CG) iterations until

$$\|\nabla^2 \bar{f}(\boldsymbol{w}^k)_{F_t,F_t} \boldsymbol{v}_{F_t} + \nabla q_k(\boldsymbol{d}^{k,t})_{F_t}\| = \|\nabla q_k(\boldsymbol{d}^{k,t+1})_{F_t}\| \le \epsilon \|\nabla \bar{f}(\boldsymbol{w}^k)_{F_t}\| \tag{54}$$

is satisfied, where ϵ is a given positive constant. See Section 5.1.3 for reasons to choose CG. CG may stop before reaching (54) if either the iterate causes our search direction to exceed the trust region boundary or the singularity of the matrix $\nabla^2 \bar{f}(\boldsymbol{w}^k)_{F_t,F_t}$ is detected.

Once a direction $\boldsymbol{v}^{k,t}$ is identified, we conduct a projected line search to ensure the feasibility and the sufficient decrease of $q_k(\boldsymbol{d})$. This procedure is similar to (47)–(48) for the Cauchy step. We find λ (e.g., by a backtrack line search) such that

$$\begin{aligned} &\boldsymbol{w}^{k,t+1} = P[\boldsymbol{w}^{k,t} + \lambda \boldsymbol{v}^{k,t}],\ \boldsymbol{d}^{k,t+1} = \boldsymbol{w}^{k,t+1} - \boldsymbol{w}^k, \text{ and} \\ &q_k(\boldsymbol{d}^{k,t+1}) \le q_k(\boldsymbol{d}^{k,t}) + \sigma \nabla q_k(\boldsymbol{d}^{k,t})_{F_t}^T (\boldsymbol{d}^{k,t+1} - \boldsymbol{d}^{k,t})_{F_t}, \end{aligned} \tag{55}$$

where $P[\cdot]$ is defined in (49) and σ is the same as that in (47).

5.1.3 Hessian-vector Product

CG is very suitable for solving the linear system (53) as it requires only Hessian-vector products. For logistic regression,

$$\nabla^2 \bar{f}(\boldsymbol{w})_{F_t,F_t} = C \begin{bmatrix} X^T \\ -X^T \end{bmatrix}_{F_t,:} D \begin{bmatrix} X & -X \end{bmatrix}_{:,F_t}, \tag{56}$$

where D is an $l \times l$ diagonal matrix with

$$D_{ii} = \tau\left(y_i(\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i\right)\left(1 - \tau(y_i(\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i)\right). \tag{57}$$

The matrix $\nabla^2 \bar{f}(\boldsymbol{w})_{F_t,F_t}$ may be too large to be stored. If using CG, the Hessian-vector product can be conducted by a sequence of matrix-vector products:

$$\nabla^2 \bar{f}(\boldsymbol{w}^{k,t})_{F_t,F_t} \boldsymbol{v}_{F_t} = C \begin{bmatrix} X^T \\ -X^T \end{bmatrix}_{F_t,:} \left(D\left(\begin{bmatrix} X & -X \end{bmatrix}_{:,F_t} \boldsymbol{v}_{F_t}\right)\right). \tag{58}$$

Thus, the memory problem is solved.

In Lin et al. (2008) for L2-regularized problems, Hessian-vector products are only needed in CG, but here they are also used in the projected line search for calculating $q_k(\boldsymbol{d}^{k,t+1}) - q_k(\boldsymbol{d}^{k,t})$; see (52) and (55). Moreover, in (58) we use only a sub-matrix of the Hessian, so $|F_t|$ columns of $\begin{bmatrix} X & -X \end{bmatrix}$ are needed. Because in general $|F_t|$ is smaller than the number of variables, calculating (58) may be faster than the product between the whole Hessian and a vector. To quickly access X's columns, storing the data matrix X in the column format is more suitable. For a discussion between row and column formats, see Lin et al. (2008, Section 4.3).

5.1.4 Convergence

From Theorem 2.1 of Lin and Moré (1999), any limit point of $\{\boldsymbol{w}^k\}$ is an optimum of (12). For the local convergence rate, in Appendix E, we indicate that in general TRON can achieve quadratic convergence.

5.2 An Interior Point Method (IPM)

Koh et al. (2007) proposed an interior point method to solve (13) with logistic loss. In (13), we omit the bias term b, but Koh et al. (2007) included this term. They consider a log barrier function so that $(\boldsymbol{w}, \boldsymbol{u})$ is an interior point of the feasible region:

$$\phi_t(b, \boldsymbol{w}, \boldsymbol{u}) \equiv t \left(\sum_{j=1}^{n} u_j + C \sum_{i=1}^{l} \xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i) \right) - \sum_{j=1}^{n} \log(u_j^2 - w_j^2),$$

where $t > 0$ is a parameter. The unique minimizer $(b^*(t), \boldsymbol{w}^*(t), \boldsymbol{u}^*(t))$ under any given t forms a curve called the "central path," which approaches an optimal solution of (13) as $t \to \infty$. An interior point method thus alternatively minimizes $\phi_t(b, \boldsymbol{w}, \boldsymbol{u})$ and adjusts t. From a set of $(\boldsymbol{w}, b)$ on the search path, we can construct a feasible solution to the dual problem of (13) and evaluate the duality gap. The duality gap is guaranteed to converge to 0 as we walk along the central path when $t \to \infty$. Thus, we can check the duality gap for the stopping condition.

At the kth iteration, using the current t_k, interior point methods approximately minimize $\phi_{t_k}(b, \boldsymbol{w}, \boldsymbol{u})$ by finding a Newton direction. The following linear system is solved:

$$\nabla^2 \phi_{t_k}(b^k, \boldsymbol{w}^k, \boldsymbol{u}^k) \begin{bmatrix} \Delta b \\ \Delta \boldsymbol{w} \\ \Delta \boldsymbol{u} \end{bmatrix} = -\nabla \phi_{t_k}(b^k, \boldsymbol{w}^k, \boldsymbol{u}^k). \tag{59}$$

For logistic loss,

$$\nabla \phi_t(b, \boldsymbol{w}, \boldsymbol{u}) = \begin{bmatrix} tC \sum_{i=1}^{l} y_i \left(\tau(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)) - 1 \right) \\ tC \sum_{i=1}^{l} \left(\tau(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)) - 1 \right) y_i \boldsymbol{x}_i + \begin{bmatrix} 2w_1 / \left(u_1^2 - w_1^2 \right) \\ \vdots \\ 2w_n / \left(u_n^2 - w_n^2 \right) \end{bmatrix} \\ t\boldsymbol{e}_n - \begin{bmatrix} 2u_1 / \left(u_1^2 - w_1^2 \right) \\ \vdots \\ 2u_n / \left(u_n^2 - w_n^2 \right) \end{bmatrix} \end{bmatrix}$$

Algorithm 8 IPM for L1-regularized logistic regression

1. Given b^1 and an interior point $(\boldsymbol{w}^1, \boldsymbol{u}^1)$. Let $t^1 = \frac{1}{Cl}$.
2. For $k = 1, 2, 3, \ldots$
   - Obtain a Newton direction $\begin{bmatrix} \Delta b \\ \Delta \boldsymbol{w} \\ \Delta \boldsymbol{u} \end{bmatrix}$ by solving (59).
   - A backtrack line search procedure to ensure the sufficient decrease of $\phi_{t_k}(\cdot)$
   - Update
   $$\begin{bmatrix} b^{k+1} \\ \boldsymbol{w}^{k+1} \\ \boldsymbol{u}^{k+1} \end{bmatrix} = \begin{bmatrix} b^k + \lambda \Delta b \\ \boldsymbol{w}^k + \lambda \Delta \boldsymbol{w} \\ \boldsymbol{u}^k + \lambda \Delta \boldsymbol{u} \end{bmatrix}.$$
   - Construct a dual feasible point and evaluate the duality gap η.
   - Set
   $$t^{k+1} = \begin{cases} \max\left(\mu \min(2n/\eta, t^k),\ t^k\right) & \text{if } \lambda \geq s_{\min}, \\ t^k & \text{otherwise,} \end{cases}$$
   where μ and $s_{\min}$ are constants.

and

$$\nabla^2 \phi_t(b, \boldsymbol{w}, \boldsymbol{u}) = \begin{bmatrix} tC\boldsymbol{y}^T D \boldsymbol{y} & tC\boldsymbol{e}_l^T DX & \mathbf{0}^T \\ tCX^T D\boldsymbol{e}_l & tCX^T DX + D_1 & D_2 \\ \mathbf{0} & D_2 & D_1 \end{bmatrix},$$

where $\tau(\cdot)$ is defined in (9), $\boldsymbol{e}_n \in R^n$ and $\boldsymbol{e}_l \in R^l$ are the vectors of all ones, D is similar to (57) but includes b, and D_1 and D_2 are $n \times n$ diagonal matrices:

$$(D_1)_{jj} = 2\left(u_j^2 + w_j^2\right) / \left(u_j^2 - w_j^2\right)^2 \quad \text{and} \quad (D_2)_{jj} = -4u_j w_j / \left(u_j^2 - w_j^2\right)^2.$$

Koh et al. (2007) apply preconditioned conjugate gradient methods (PCG) to solve (59) with diagonal preconditioning.

For the convergence, a backtrack line search procedure is needed to ensure the sufficient decrease of $\phi_{t_k}(\cdot)$. Koh et al. (2007) did not discuss details of their method's convergence. However, because interior point methods are a type of Newton methods, they often enjoy fast local convergence.

5.3 Lassplore Method

Liu et al. (2009) apply Nesterov's method (Nesterov, 2003) to solve (14). This method can handle (14) with and without the bias term. For simplicity, we do not consider the bias term. In addition to the sequence of iterations $\{\boldsymbol{w}^k\}$, for faster convergence, Nesterov's method uses another sequence of searching points $\{\boldsymbol{s}^k\}$, where

$$\boldsymbol{s}^k = \boldsymbol{w}^k + \beta_k(\boldsymbol{w}^k - \boldsymbol{w}^{k-1}),$$

for some positive parameter β_k. From $\boldsymbol{s}^k$, we obtain $\boldsymbol{w}^{k+1}$ by taking the negative gradient direction:

$$\boldsymbol{w}^{k+1} = \boldsymbol{s}^k - \lambda_k \nabla \bar{L}(\boldsymbol{s}^k),$$

Algorithm 9 Lassplore for solving (14)

- Given $\boldsymbol{w}^0$ and $\boldsymbol{w}^1$, $\alpha_0 = 0.5$, $\lambda_1 > 0$, $\gamma_1 \geq 0$.
- For $k = 1, 2, 3, \ldots$
  1. While 1 do
     - Compute $\alpha_k \in (0, 1)$ as the root of $\eta_k(\alpha)$, β_k by (60), and γ_{k+1} by (61).
     - Compute $\boldsymbol{s}^k = \boldsymbol{x}^k + \beta_k(\boldsymbol{w}^k - \boldsymbol{w}^{k-1})$.
     - Compute $\boldsymbol{w}^{k+1}$ by (63).
     - If $\bar{L}(\boldsymbol{w}^{k+1})$ satisfies (62)
       goto Step 2.
       Else
       $\lambda_k \leftarrow \lambda_k/2$.
  2. Find the initial λ_{k+1} for the next iteration by an adaptive scheme.[4]

where

$$\bar{L}(\boldsymbol{w}) \equiv \sum_{i=1}^{l} \xi_i(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$$

is the objective function of (14) and λ_k is the step size. Note that $\bar{L}(\boldsymbol{w})$ is different from $L(\boldsymbol{w})$ defined in (7) because the penalty parameter C is not needed in (14). Liu et al. (2009) suggest to estimate β_k by

$$\beta_k = \frac{\gamma_k(1 - \alpha_{k-1})}{\alpha_{k-1}(\gamma_k + \frac{\alpha_k}{\lambda_k})}, \tag{60}$$

where $\alpha_k \in (0, 1)$ is the root of a quadratic function

$$\eta_k(\alpha) \equiv \frac{\alpha^2}{\lambda_k} + \gamma_k\alpha - \gamma_k$$

and $\gamma_k \geq 0$ satisfies

$$\gamma_{k+1} = (1 - \alpha_k)\gamma_k \text{ if } k \geq 1 \text{ and } \gamma_1 \geq 0. \tag{61}$$

We have $\alpha_k \in (0, 1)$ because $\eta_k(0) = -\gamma_k < 0$ and $\eta_k(1) = 1/\lambda_k > 0$.

Lassplore applies an adaptive line search scheme that adjusts λ_k so that $\boldsymbol{w}^{k+1}$ satisfies

$$\bar{L}(\boldsymbol{w}^{k+1}) \leq \bar{L}(\boldsymbol{s}^k) + \nabla\bar{L}(\boldsymbol{s}^k)^T(\boldsymbol{w}^{k+1} - \boldsymbol{s}^k) + \frac{1}{2\lambda_k}\|\boldsymbol{w}^{k+1} - \boldsymbol{s}^k\|_2^2. \tag{62}$$

The new solution $\boldsymbol{w}^{k+1}$ generated from the above process may not satisfy the constraint in (14), so Lassplore projects the solution to the feasible region:

$$\boldsymbol{w}^{k+1} \equiv \arg\min_{\boldsymbol{w}}\{\|(\boldsymbol{s}^k - \lambda_k\nabla L(\boldsymbol{s}^k)) - \boldsymbol{w}\| \mid \|\boldsymbol{w}\|_1 \leq K\}. \tag{63}$$

This projection is related to (15) in Section 2.2.2, but Lassplore applies the method in Liu and Ye (2009) to efficiently compute (63).

A summary of Lassplore is given in Algorithm 9.

4. See Liu et al. (2009, Algorithm 3) for details.

6. Three Other Methods for L1-regularized Logistic Regression

We describe details of three more methods because they are included in our comparison.

6.1 Orthant-Wise Limited-memory Quasi-Newton (OWL-QN)

LBFGS (Liu and Nocedal, 1989) is a limited memory quasi Newton method for unconstrained smooth optimization. It can not deal with (1) because of the non-differentiability. Andrew and Gao (2007) modified LBFGS to solve (1) and named their method as OWL-QN. Here, we discuss how they handle the non-differentiability.

From earlier discussion, we know that (1) is differentiable in a region where the sign of w_j does not change. Thus, OWL-QN restricts the search space to such a region (called orthant). At the kth iteration, it searches $\boldsymbol{w}^{k+1}$ on the space:

$$\Omega_k \equiv \{\boldsymbol{w} \in R^n \mid \mathrm{sgn}(w_j) = s_j^k, j = 1, \ldots, n\},$$

where

$$s_j^k \equiv \begin{cases} \mathrm{sgn}(w_j^k) & \text{if } w_j^k \neq 0, \\ \mathrm{sgn}(-\bar{\nabla}_j f(\boldsymbol{w}^k)) & \text{otherwise,} \end{cases} \tag{64}$$

and

$$\bar{\nabla}_j f(\boldsymbol{w}) \equiv \begin{cases} L_j'(\boldsymbol{w}) + 1 & \text{if } w_j > 0 \text{ or } (w_j = 0 \text{ and } L_j'(\boldsymbol{w}) + 1 < 0), \\ L_j'(\boldsymbol{w}) - 1 & \text{if } w_j < 0 \text{ or } (w_j = 0 \text{ and } L_j'(\boldsymbol{w}) - 1 > 0), \\ 0 & \text{otherwise} \end{cases} \tag{65}$$

is defined as the pseudo gradient of $f(\boldsymbol{w})$. In (64), if $w_j^k = 0$, we consider the space where w_j can be moved by taking the negative gradient direction. OWL-QN then approximately minimizes a quadratic approximation of (1) in the search space Ω_k:

$$\begin{aligned} \min_{\boldsymbol{d}} \quad & f(\boldsymbol{w}^k) + \bar{\nabla} f(\boldsymbol{w}^k)^T \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}^T H_k \boldsymbol{d} \\ \text{subject to} \quad & \boldsymbol{w}^k + \boldsymbol{d} \in \Omega_k, \end{aligned} \tag{66}$$

where H_k approximates the Hessian of $f(\boldsymbol{w}^k)$ by the first-order information gathered from previous iterations. Details for getting H_k can be found in Liu and Nocedal (1989). To approximately solve (66), OWL-QN finds the minimum of the quadratic objective function:

$$\boldsymbol{d}^k = -H_k^{-1} \bar{\nabla} f(\boldsymbol{w}^k), \tag{67}$$

obtains a direction $\bar{\boldsymbol{d}}^k$ by confining $\boldsymbol{d}^k$ on the same orthant as $-\bar{\nabla} f(\boldsymbol{w}^k)$:

$$\bar{d}_j^k = \begin{cases} d_j^k & \text{if } \mathrm{sgn}(d_j^k) = \mathrm{sgn}(-\bar{\nabla}_j f(\boldsymbol{w}^k)), \\ 0 & \text{otherwise,} \end{cases} \tag{68}$$

and then conducts a backtracking line search to find λ such that the sufficient decrease of the function value is satisfied. Note that $\boldsymbol{w}^{k+1}$ must be in Ω_k, so following (64),

$$w_j^{k+1} = \begin{cases} w_j^k + \lambda \bar{d}_j^k & \text{if } \mathrm{sgn}(w_j^k + \lambda \bar{d}_j^k) = s_j^k, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 10 OWL-QN: An extension of LBFGS for L1-regularized logistic regression

1. Given $\boldsymbol{w}^1$.
2. For $k = 1, 2, 3, \ldots$
   - Compute $\bar{\nabla} f(\boldsymbol{w}^k)$ by (65).
   - Obtain H^k by information gathered from previous iterations and compute $\boldsymbol{d}^k$ by (67).
   - Compute $\bar{\boldsymbol{d}}^k$ by (68).
   - Find $\boldsymbol{w}^{k+1} \in \Omega_k$ by a backtracking line search.

Algorithm 10 summarizes the procedure.

Regarding the convergence, Yu et al. (2010, Appendix D) point out that the proof by Andrew and Gao (2007) is flawed. Yu et al. (2010) give the convergence proof for a slightly modified algorithm.

6.2 Generalized Linear Model with Elastic Net

Friedman et al. (2010) proposed GLMNET to handle least-square and log-linear losses with L1/L2 regularization. Here, we discuss how GLMNET solves L1-regularized logistic regression. Although GLMNET can solve (5) with the bias term, for simplicity, we only indicate how GLMNET solves (1).

Because of the twice differentiability of the logistic loss function, the gradient of $L(\boldsymbol{w})$ is shown in (42) and the Hessian is

$$\nabla^2 L(\boldsymbol{w}) = CX^T DX, \tag{69}$$

where $D \in R^{l \times l}$ is a diagonal matrix with

$$D_{ii} = \tau(y_i \boldsymbol{w}^T \boldsymbol{x}_i)\big(1 - \tau(y_i \boldsymbol{w}^T \boldsymbol{x}_i)\big) \tag{70}$$

and $\tau(\cdot)$ is defined in (9). See similar formulations derived earlier for TRON in (56) and (57). Given the current solution $\boldsymbol{w}^k$, GLMNET considers a quadratic approximation of $L(\boldsymbol{w})$. By the second-order Taylor expansion,

$$\begin{aligned} & f(\boldsymbol{w}^k + \boldsymbol{d}) - f(\boldsymbol{w}^k) \\ = & \Big(\|\boldsymbol{w}^k + \boldsymbol{d}\|_1 + L(\boldsymbol{w}^k + \boldsymbol{d})\Big) - \Big(\|\boldsymbol{w}^k\|_1 + L(\boldsymbol{w}^k)\Big) \\ \approx & \nabla L(\boldsymbol{w}^k)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T \nabla^2 L(\boldsymbol{w}^k)\boldsymbol{d} + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1. \end{aligned}$$

Then, GLMNET solves the Newton-like system

$$\min_{\boldsymbol{d}} \quad q_k(\boldsymbol{d}) \equiv \nabla L(\boldsymbol{w}^k)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T \nabla^2 L(\boldsymbol{w}^k)\boldsymbol{d} + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1 \tag{71}$$

by a cyclic coordinate descent method. Following the framework in Algorithm 1, $\boldsymbol{d}$'s values are sequentially updated by minimizing the following one-variable function:

$$\begin{aligned} g_j(z) &\equiv q_k(\boldsymbol{d} + z\boldsymbol{e}_j) - q_k(\boldsymbol{d}) \\ &= |w_j^k + d_j + z| - |w_j^k + d_j| + Bz + \frac{1}{2}Az^2, \end{aligned}$$

Algorithm 11 GLMNET for L1-regularized logistic regression

1. Given $\boldsymbol{w}^1$.
2. For $k = 1, 2, 3, \ldots$
   - Let $\boldsymbol{d}^k \leftarrow \mathbf{0}$.
   - While $\boldsymbol{d}^k$ is not optimal for minimizing $q_k(\boldsymbol{d})$
     - For $j = 1, \ldots, n$
       - Solve the following one-variable problem by (27):

$$\bar{z} = \arg\min_z \quad q_k(\boldsymbol{d}^k + z\boldsymbol{e}_j) - q_k(\boldsymbol{d}^k).$$

       - $d_j^k \leftarrow d_j^k + \bar{z}$.
   - $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \boldsymbol{d}^k$.

where

$$B \equiv \nabla_j L(\boldsymbol{w}^k) + \sum_t \nabla^2_{jt} L(\boldsymbol{w}^k) d_t \quad \text{and} \quad A \equiv \nabla^2_{jj} L(\boldsymbol{w}^k)$$

can be calculated using (42) and (69). It is easy to minimize $g_j(z)$ by (27).[5]

Because calculating the matrix D involves many exponential operations, GLMNET also considers using an approximation of $\nabla^2 L(\boldsymbol{w})$ and minimizes

$$q_k(\boldsymbol{d}) \equiv \nabla L(\boldsymbol{w}^k)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T H \boldsymbol{d} + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1,$$

where $H \equiv 0.25 C X^T \mathcal{I} X$ and $\mathcal{I}$ is an identity matrix. That is, we use a cheaper but less accurate approximation of $f(\boldsymbol{w}^k + \boldsymbol{d}) - f(\boldsymbol{w}^k)$. A sketch of GLMNET is in Algorithm 11.

We briefly describe some GLMNET's implementation details. GLMNET applies a shrinking technique to solve a smaller optimization problem than (71); see similar techniques for decomposition methods in Section 4.1.2. Using a sparse representation of $\boldsymbol{w}$ and maintaining an index set Ω to indicate the non-zero elements of $\boldsymbol{d}$, GLMNET solves a smaller problem by a coordinate descent method:

$$\min_{\boldsymbol{d}_\Omega} \quad \nabla L(\boldsymbol{w}^k)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T H \boldsymbol{d} + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1.$$

GLMNET conducts feature-wise normalization before solving the optimization problem. That is, it solves (1) by replacing $\boldsymbol{x}_i$ with $\tilde{\boldsymbol{x}}_i$, where

$$\tilde{x}_{ij} \equiv \frac{x_{ij} - \bar{x}_j}{\sigma_j}, \forall i, j, \quad \bar{x}_j = \frac{\sum_{i=1}^l x_{ij}}{l}, \forall j, \quad \text{and } \sigma_j = \sqrt{\sum_{i=1}^l x_{ij}^2 - \sum_{i=1}^l \bar{x}_j^2}, \forall j.$$

Notice that there is no guarantee that GLMNET converges to an optimum. Furthermore, the function value may not decrease because GLMNET does not conduct a line search procedure on the direction $\boldsymbol{d}$. For minimizing the quadratic approximation $q_k(\boldsymbol{d})$, GLMNET measures the relative step change in the successive coordinate descent iterations. Deciding when to stop minimizing $q_k(\boldsymbol{d})$ is an issue because a strict stopping condition may already cause long running time for $q_1(\boldsymbol{d})$.

5. In GLMNET implementation, instead of finding z, a different but equivalent update is used to get new $w_j^k + d_j$.

Algorithm 12 BMRM for L1-regularized logistic regression

1. Given $\boldsymbol{w}^1$.
2. For $k = 1, 2, 3, \ldots$
   - Compute and store $\boldsymbol{a}_k$ and b_k by (73).
   - Obtain $\boldsymbol{w}^{k+1}$ by solving the linear program (76).

6.3 Bundle Method

Bundle method is a cutting plane approach for minimizing non-smooth convex problems. Teo et al. (2010) proposed a bundle method BMRM to handle non-differentiable loss functions (e.g., L1 loss). They provide an extension to handle L1 regularization. Interestingly, BMRM applies cutting planes only to the loss function, regardless of whether it is differentiable or not. Therefore, for problem (1) with logistic loss, BMRM uses a non-smooth method to handle the smooth loss function and some other ways for the non-smooth regularization term $\|\boldsymbol{w}\|_1$.

Let $\boldsymbol{w}^k$ be the solution at the kth iteration. Using the convexity of the loss function, BMRM builds a cutting plane (i.e., the first-order Taylor expansion) of $L(\boldsymbol{w})$ at $\boldsymbol{w} = \boldsymbol{w}^k$:

$$\begin{aligned} L(\boldsymbol{w}) &\geq \nabla L(\boldsymbol{w}^k)^T(\boldsymbol{w} - \boldsymbol{w}^k) + L(\boldsymbol{w}^k) \\ &= \boldsymbol{a}_k^T \boldsymbol{w} + b_k, \ \forall \boldsymbol{w}, \end{aligned} \tag{72}$$

where

$$\boldsymbol{a}_k \equiv \nabla L(\boldsymbol{w}^k) \quad \text{and} \quad b_k \equiv L(\boldsymbol{w}^k) - \boldsymbol{a}_k^T \boldsymbol{w}^k. \tag{73}$$

If $L(\boldsymbol{w})$ is non-differentiable, in (72), BMRM substitutes $\nabla L(\boldsymbol{w})$ with the sub-gradient of $L(\boldsymbol{w})$.

BMRM maintains all cutting planes from the earlier iterations to form a lower-bound function for $L(\boldsymbol{w})$:

$$L(\boldsymbol{w}) \geq L_k^{\text{CP}}(\boldsymbol{w}) \equiv \max_{1 \leq t \leq k} \boldsymbol{a}_t^T \boldsymbol{w} + b_t, \ \forall \boldsymbol{w}. \tag{74}$$

BMRM obtains $\boldsymbol{w}^{k+1}$ by solving the following sub-problem:

$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{w}\|_1 + L_k^{\text{CP}}(\boldsymbol{w}). \tag{75}$$

Using (74) and the splitting of $\boldsymbol{w}$ in (12) by $\boldsymbol{w} = \boldsymbol{w}^+ - \boldsymbol{w}^-$, Equation (75) can be reformulated to the following linear programming problem:

$$\begin{aligned} \min_{\boldsymbol{w}^+, \boldsymbol{w}^-, \zeta} \quad & \sum_{j=1}^{n} w_j^+ + \sum_{j=1}^{n} w_j^- + \zeta \\ \text{subject to} \quad & \boldsymbol{a}_t^T(\boldsymbol{w}^+ - \boldsymbol{w}^-) + b_t \leq \zeta, \quad t = 1, \ldots, k, \\ & w_j^+ \geq 0, \ w_j^- \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{76}$$

A summary of the BMRM approach is given in Algorithm 12. Teo et al. (2010, Appendix C) indicated that because of the L1 regularization, the convergence result has not been fully established yet.

7. L1-regularized L2-loss Support Vector Machines

Previous sections have focused on L1-regularized logistic regression. Now we consider (1) with the L2-loss function (4). The optimization problem can be rewritten as

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \equiv \|\boldsymbol{w}\|_1 + C \sum_{i \in I(\boldsymbol{w})} b_i(\boldsymbol{w})^2, \tag{77}$$

where

$$b_i(\boldsymbol{w}) \equiv 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i \quad \text{and} \quad I(\boldsymbol{w}) \equiv \{i \mid b_i(\boldsymbol{w}) > 0\}. \tag{78}$$

Therefore, the sum of losses is

$$L(\boldsymbol{w}) = C \sum_{i \in I(\boldsymbol{w})} b_i(\boldsymbol{w})^2. \tag{79}$$

In contrast to logistic loss, the L2-loss function is differentiable but not twice differentiable (Mangasarian, 2002). Thus, some methods discussed in Sections 4–6 may not be directly applicable because they use second-order information. However, as shown in Mangasarian (2002, Section 3), (79) is twice differentiable at all but $\{\boldsymbol{w} \mid b_i(\boldsymbol{w}) = 0 \text{ for some } i\}$. Moreover, $\nabla L(\boldsymbol{w})$ is globally Lipschitz continuous, so a generalized Hessian exists everywhere. Using a generalized Hessian, we may modify algorithms using the second-order information for L1-regularized L2-loss SVMs. In the following two sections, we extend CDN and TRON to solve (77).

7.1 CDN for L2-loss SVMs

To apply CDN for L2-loss SVMs, in the sub-problem (17), we have

$$L_j(z) = C \sum_{i \in I(\boldsymbol{w}^{k,j} + z\boldsymbol{e}_j)} b_i(\boldsymbol{w}^{k,j} + z\boldsymbol{e}_j)^2.$$

For a second-order approximation similar to (26), we need $L_j'(0)$ and $L_j''(0)$:

$$L_j'(0) = -2C \sum_{i \in I(\boldsymbol{w}^{k,j})} y_i x_{ij} b_i(\boldsymbol{w}^{k,j}).$$

Unfortunately, $L_j''(0)$ is not well-defined if there exists some i such that $b_i(\boldsymbol{w}^{k,j}) = 0$. Following Chang et al. (2008), we consider the generalized second derivative:

$$2C \sum_{i \in I(\boldsymbol{w}^{k,j})} x_{ij}^2. \tag{80}$$

By replacing $L_j''(0)$ in (26) with the above quantity, we can easily obtain a direction d. However, the value in (80) may be zero if $x_{ij} = 0,\ \forall i \in I(\boldsymbol{w}^{k,j})$. To apply the convergence result in Tseng and Yun (2009), we ensure the strict positivity by taking

$$\max\left(2C \sum_{i \in I(\boldsymbol{w}^{k,j})} x_{ij}^2,\ \epsilon\right), \tag{81}$$

where ϵ is a small positive value. Following the explanation in Appendix F, we have the finite termination of the line search procedure and the asymptotic convergence of the function value. If the loss function $L(\boldsymbol{w})$ is strictly convex, the algorithm converges at least linearly. The proof is in the supplementary file.

Like the situation for logistic regression, the major cost for finding the Newton direction d and for the line search procedure is to calculate $\boldsymbol{w}^T\boldsymbol{x}_i,\ \forall i \in I$. We maintain $b_i(\boldsymbol{w}),\ \forall i$ using the same trick in (25). All other implementation techniques discussed in Section 4.1.2 for logistic regression can be applied here.

7.2 TRON for L2-loss SVMs

We apply TRON to solve the bound-constrained problem (12) using L2 loss. Following the notation in Section 5.1, $\boldsymbol{w} \in R^{2n}$ is defined in (43) and $\bar{f}(\boldsymbol{w})$ is the objective function in (12).

The quadratic model $q_k(\boldsymbol{d})$ requires the gradient and the Hessian of $\bar{f}(\boldsymbol{w})$. We have

$$\nabla \bar{f}(\boldsymbol{w}) = \boldsymbol{e} + 2C\left(\bar{X}_{I,:}^T \bar{X}_{I,:}\boldsymbol{w} - \bar{X}_{I,:}^T \boldsymbol{y}_I\right),$$

where $\boldsymbol{e} \in R^{2n}$ is a vector of all ones, $\bar{X} \equiv \begin{bmatrix} X & -X \end{bmatrix}$, and I is defined in (78). $\bar{X}_{I,:}$ denotes a sub-matrix including $\bar{X}$'s rows corresponding to I; see (8). Note that $b_i(\boldsymbol{w})$ defined in (78) is now calculated by

$$1 - y_i(\boldsymbol{w}_{1:n} - \boldsymbol{w}_{(n+1):2n})^T \boldsymbol{x}_i.$$

Following Mangasarian (2002) and Fan et al. (2008, Appendix D), we consider the generalized Hessian matrix:

$$2C\bar{X}^T D \bar{X},$$

where $D \in R^{l \times l}$ is a diagonal matrix with

$$D_{ii} = \begin{cases} 1 & \text{if } b_i(\boldsymbol{w}) > 0, \\ 0 & \text{if } b_i(\boldsymbol{w}) \le 0. \end{cases}$$

We then apply Algorithm 7 to approximately minimize $q_k(\boldsymbol{d})$. For the Hessian-vector product, only instances in the index set I and variables in the free set F are considered. Thus, the Hessian-vector product in (58) becomes

$$2C\bar{X}_{I,F}^T\left(D_{I,I}\left(\bar{X}_{I,F}\boldsymbol{v}_F\right)\right),$$

where $\boldsymbol{v}$ is a vector in R^{2n}.

Regarding the convergence, from Theorem 2.1 of Lin and Moré (1999), any limit point of $\{\boldsymbol{w}^k\}$ is an optimal solution. However, without the twice differentiability, it is unclear if the local quadratic convergence holds.

8. Numerical Experiments

In Sections 4–7, we have described details of several large-scale optimization methods for L1-regularized linear classification. In this section, we conduct experiments to investigate their performances. We describe data sets and experimental settings first. Then, we comprehensively compare methods for logistic regression and L2-loss SVMs. Programs used in this paper are available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/exp.html`.

Data set	l	n	#nz
a9a	32,561	123	451,592
real-sim	72,309	20,958	3,709,083
news20	19,996	1,355,191	9,097,916
rcv1	677,399	47,236	49,556,258
yahoo-japan	176,203	832,026	23,506,415
yahoo-korea	460,554	3,052,939	156,436,656

Table 1: Statistics of data sets: l and n denote the numbers of instances and features in a data set, respectively. The column #nz indicates the number of non-zero entries.

Data set	LR without bias			LR with bias			L2-loss SVM		
	C	Density	Acc.	C	Density	Acc.	C	Density	Acc.
a9a	4.0	94.3	85.2	2.0	89.3	85.3	0.5	90.2	85.2
real-sim	4.0	16.8	97.1	4.0	16.2	97.0	1.0	18.9	97.1
news20	64.0	0.2	95.1	64.0	0.2	94.9	64.0	0.7	96.1
rcv1	4.0	23.8	97.8	4.0	23.0	97.8	1.0	25.9	97.8
yahoo-japan	4.0	1.3	91.9	4.0	1.0	93.1	1.0	1.4	92.2
yahoo-korea	4.0	1.0	87.6	4.0	0.9	87.7	1.0	1.0	87.7

Table 2: The best parameter C, the model density (%), and the testing accuracy (%). We conduct five-fold cross validation on the training set to select C in $\{2^k \mid k = -4, -3, \ldots, 6\}$. Using the selected C, we build a model to predict the testing set.

8.1 Data Sets

Table 1 lists data statistics. Most data sets include documents, where the numbers of both features and instances are large. An exception is the problem a9a from UCI "adults" data sets; it has $l \gg n$. For other document sets, real-sim includes some Usenet articles, news20 is a collection of news documents, rcv1 is an archive of manually categorized newswire stories from Reuters, and yahoo-japan/yahoo-korea are document data from Yahoo!. Except yahoo-japan and yahoo-korea, other data sets are publicly available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. For every document data set, instance-wise normalization has been conducted so that the length of each instance is one.

To estimate the testing accuracy, a stratified selection is taken to split each data set into one fifth for testing and the rest for training.

8.2 Experimental Settings

We consider the following implementations discussed in Sections 4–7.

- BBR: the cyclic coordinate descent method for logistic regression is described in Section 4.1.1. We download version 4.03 from `http://www.bayesianregression.org/`.
- CDN: the cyclic coordinate descent methods for logistic regression and L2-loss SVMs are respectively described in Sections 4.1.2 and 7.1. To check the sufficient decrease condition, we use $\sigma = 0.01$ and $\beta = 1/2$. For L2-loss SVM, we use $\epsilon = 10^{-12}$ in (81). The

implementation is the same as that included in version 1.6 of our software LIBLINEAR (`http://www.csie.ntu.edu.tw/~cjlin/liblinear/`).
In Section 4.1.2, we discuss the shrinking technique for CDN. We defer the investigation of its effectiveness to Section 8.4. In all other places of the comparison, we run CDN with the shrinking strategy.

- SCD: the stochastic coordinate descent method for logistic regression is described in Section 4.1.3. The source code is available at `http://ttic.uchicago.edu/~tewari/code/scd/`.
- CGD-GS: the block coordinate gradient descent method for logistic regression is described in Section 4.2. Following Yun and Toh's (2011) settings, we choose

$$H = \text{diag}\left[\min\big(\max(\nabla^2_{jj} L(\boldsymbol{w}^k), 10^{-10}), 10^{10}\big)\right]_{j=1,\ldots,n}$$

  and apply the Gauss-Southwell-r rule to choose the working set J; see Equation (40). Note that from Table 9 of Yun and Toh (2011), implementations using Gauss-Southwell-r and Gauss-Southwell-q rules perform similarly on document data. We use default values for all parameters; see Section 5.1 in Yun and Toh (2011). In particular, $\sigma = 0.1$, $\beta = 1/2$, and $\gamma = 0$. The source code is available at `http://www.math.nus.edu.sg/~matys/`.
- TRON: the trust region Newton methods for logistic regression and L2-loss SVMs are respectively described in Sections 5.1 and 7.2. In the projected line search (47) and (55), we use $\sigma = 0.01$ and $\beta = 0.1$. For the stopping condition of the CG procedure in Algorithm 7, we use $\epsilon = 0.1$. The parameter η_0 in (46) is 0.0001.
- IPM: the interior point method for logistic regression is described in Section 5.2. For small and dense data sets, IPM solves a linear system in Algorithm 8 by Cholesky factorization. For large and sparse data sets, it applies a preconditioned conjugate gradient method to approximately solve the linear system. In the experiment, we only consider the large and sparse setting. We use default parameters: $\sigma = 0.01$ and $\beta = 1/2$ for the line search procedure. To update t^k, we use $s_{\min} = 1/2$ and $\mu = 2$. The source code (version 0.8.2) is downloaded from `http://www.stanford.edu/~boyd/l1_logreg/`.
  As $\boldsymbol{w}$ lies in the interior of the feasible region, every w_j is non-zero after IPM stops. To gain the sparsity, following the condition (31), Koh et al. (2007) assign those w_j satisfying $|\nabla_j L(\boldsymbol{w})| \leq 0.9999$ to zero. However, if we have not run enough iterations to obtain an accurate solution, this modification of $\boldsymbol{w}$ may result in an erroneous model. We thus add another condition $|w_j| < 1$ in deciding if w_j should be assigned to zero. We will address this issue again in Section 8.3.
  We find that because of a problem of not initializing an element in an array, the previous version (0.8.1) of IPM is two or three times slower than the latest version (0.8.2). This observation indicates the difficulty in comparing software. Some minor issues may significantly affect the conclusions.
- OWL-QN: the quasi Newton method is described in Section 6.1. The source code (version 1.1.2) is available at `http://research.microsoft.com/en-us/um/people/jfgao/`.
- GLMNET: the method is described in Section 6.2 for logistic regression. Following the default setting, we use the full Hessian in (71) instead of an approximation. We gradually reduce the stopping tolerance to obtain different models. The coordinate descent method for minimizing the quadratic approximation $q_k(\boldsymbol{d})$ stops according to a tolerance the

same as the overall stopping tolerance. The source code (version 1.5) is available at `http://cran.r-project.org/web/packages/glmnet/index.html`.

- Lassplore: this method is described in Section 5.3. We check the 1-norm of the optimal solution (obtained by other solvers) to calculate the value K in (14). The source code (version 1.0) is available at `http://www.public.asu.edu/~jye02/Software/lassplore`.
- BMRM: this bundle method is described in Section 6.3. We apply it to both logistic regression and L2-loss SVMs. The linear programming problem (76) is solved via GNU Linear Programming Kit (GLPK). The source code (version 2.2) and GLPK are available at `http://users.rsise.anu.edu.au/~chteo/BMRM.html` and `http://www.gnu.org/software/glpk/`, respectively.

GLMNET is implemented in Fortran along with an R interface. CGD-GS and Lassplore are primarily implemented in MATLAB, but expensive operations are coded in C/C++. All other solvers are implemented in C/C++ with double precision.

Some implementations (SCD, TRON, OWL-QN, and BMRM) solve (1), while some (CGD-GS, IPM, GLMNET, and Lassplore) consider the bias term and solve (5). BBR[6] and our CDN implementation can handle both (1) and (5). According to whether the bias term is considered, we categorize methods into two groups for comparison.[7] Moreover, for certain solvers their formulations are scaled by a constant. We ensure that equivalent optimization problems are solved.

Software such as IPM and GLMNET supports finding a solution path of various C values. However, in our experiments, we focus on the performance of an algorithm using a fixed C. For all methods, we set the initial solution $\boldsymbol{w}^1 = \mathbf{0}$.

The parameter C is chosen by five-fold cross validation (CV) on the training set. Using models trained under the best C, we predict the testing set to obtain the testing accuracy. The best C, the model density (the number of non-zero coefficients in $\boldsymbol{w}$ divided by n), and the corresponding testing accuracy are recorded in Table 2.

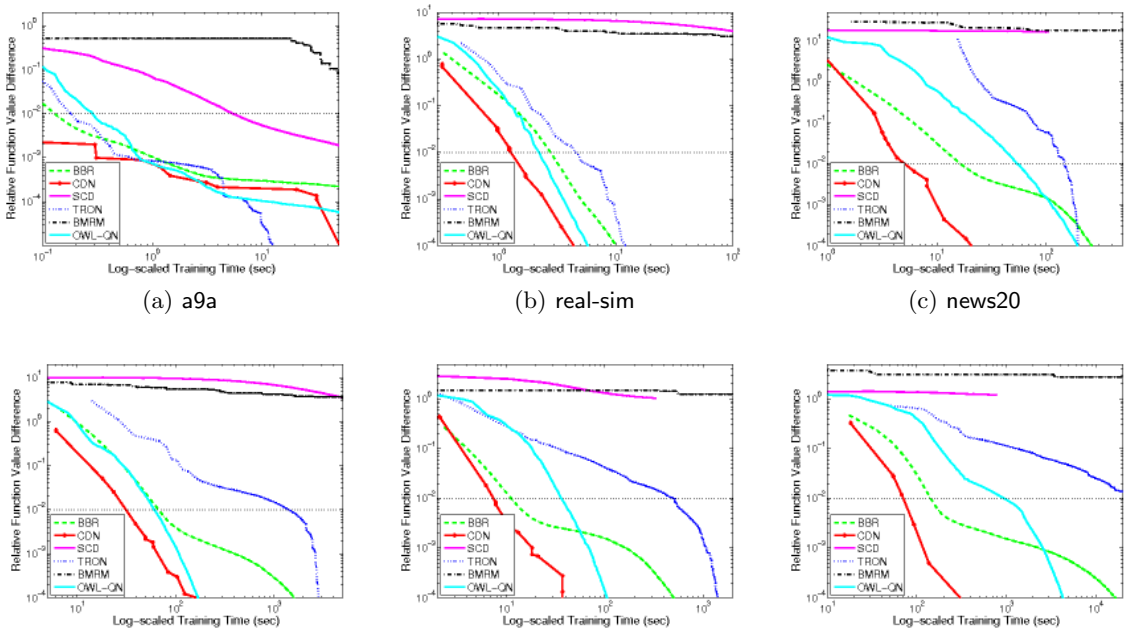
In the rest of this section, we compare the training speed of solvers for logistic regression and L2-loss SVMs by using the best parameter C of each data set. We run all experiments on a 64-bit machine with Intel Xeon 2.0GHz CPU (E5504), 4MB cache, and 32GB main memory. We use GNU C/C++/Fortran compilers (version 4.4.1) and ensure that for each package the "-O3" optimization flag is set.

8.3 Comparing Methods for L1-regularized Logistic Regression

We separately compare solvers for optimization problems without/with the bias term. The first group, which solves (1), includes BBR, CDN, SCD, TRON, OWL-QN, and BMRM. We begin at showing in Figure 1 the relation between the function value and the training time. In each figure, the x-axis is the training time and the y-axis is the relative difference to the optimal function value:

$$\frac{f(\boldsymbol{w}) - f^*}{f^*}, \tag{82}$$

6. BBR considers bias term as a feature and allows users to use an option "-I" to specify weights to individual features.
7. For simplicity, we apply BBR only to solve (1).



(a) a9a (b) real-sim (c) news20

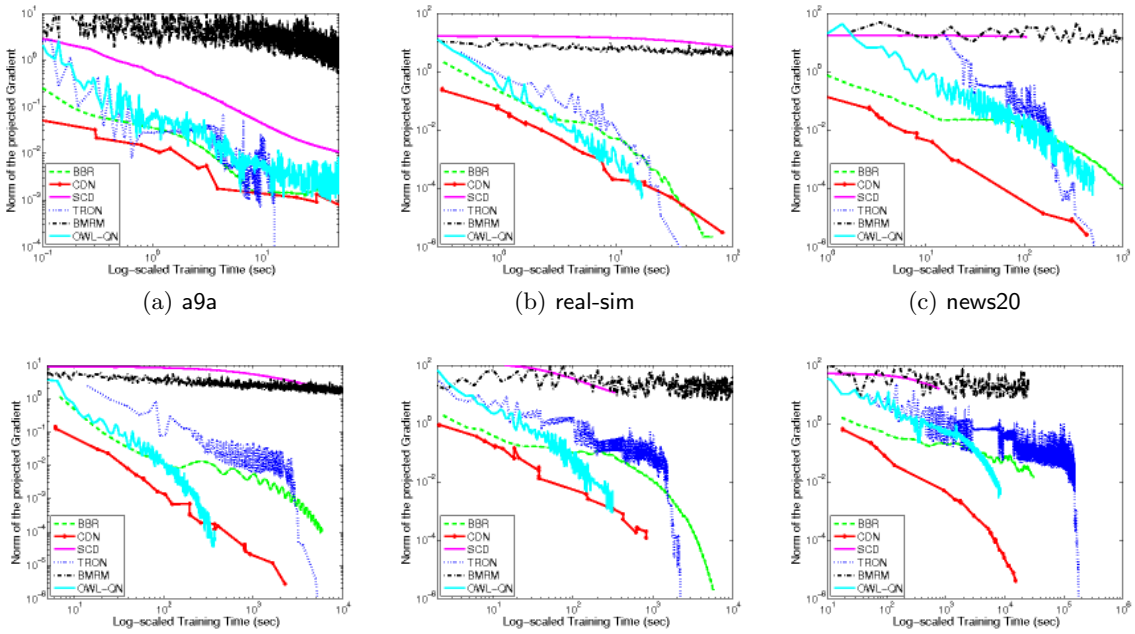
(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 1: Relative difference between the objective function and the optimum value, versus training time. Logistic regression without using the bias term is solved. Both x-axis and y-axis are log-scaled.

where f^* is obtained by running TRON with a strict stopping condition. Both x-axis and y-axis are log-scaled. We draw a dotted reference line in Figure 1 to indicate the relative error 0.1. From Figure 1, BBR and CDN can more quickly give a good solution than SCD, TRON, OWL-QN, and BMRM. However, quasi Newton and Newton methods may have faster local convergence. In Figures 1(c) and 1(d), TRON's curve is almost vertical in the end. This fast local convergence is mainly useful for problems such as a9a, for which BBR and CDN are less competitive. We note that a9a is not a document data set and its number of features is much smaller than the number of instances. Earlier studies for L2-regularized classifiers have shown that coordinate descent methods are less competitive for this type of data (Hsieh et al., 2008). The same situation seems to occur here for L1 regularization.

It is surprising that SCD is much slower than CDN and BBR because they are all coordinate descent methods. We modify CDN to randomly select working variables, a stochastic method similar to SCD; the result was still much faster than SCD, suggesting that the stochastic selection is not the culprit. It turns out that a too large upper bound of the second derivative causes the slow convergence of SCD. When $x_{ij} \in [-1, 1]$, SCD replaces $\sum_{i=1}^{l} x_{ij}^2$ in (34) with l. Then its U_j in (35) is much larger than U_j in (22) for BBR. Therefore, SCD's step size $-g_j'(0)/U_j$ is often too small.

Equation (82) cannot be used for a practical stopping condition because f^* is not easily available. Instead, we often rely on the gradient information. Now (1) is not differentiable,



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 2: The 2-norm of the minimum-norm sub-gradient (83) versus training time. Logistic regression without using the bias term is solved. Both x-axis and y-axis are log-scaled.

so we follow the optimality condition (6) to define the minimum-norm sub-gradient:

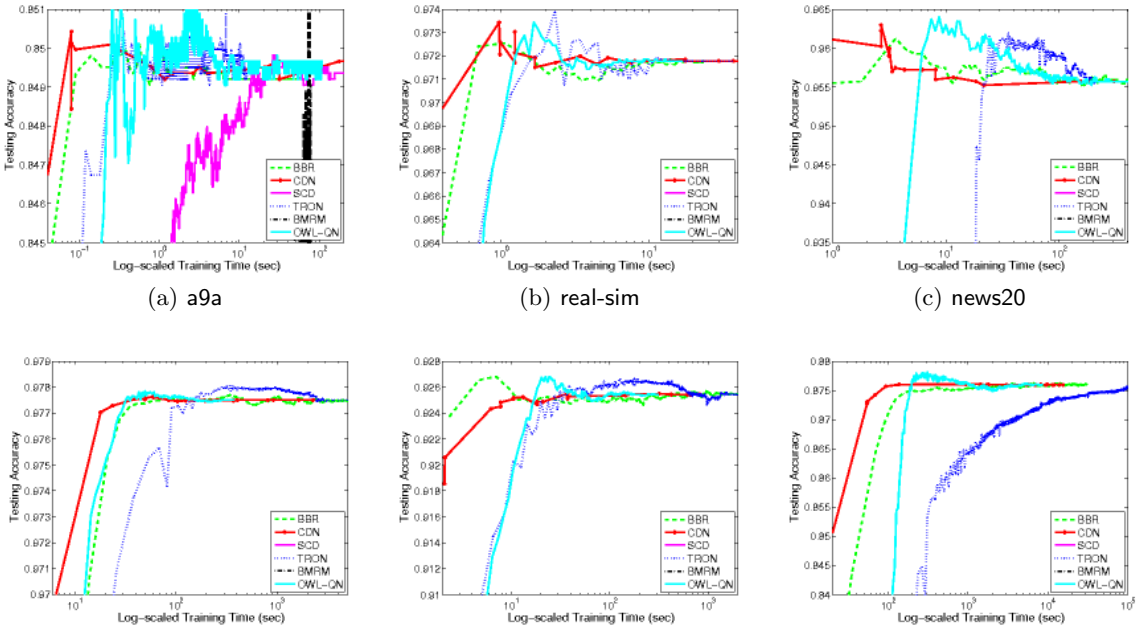
$$\nabla_j^S f(\boldsymbol{w}) \equiv \begin{cases} \nabla_j L(\boldsymbol{w}) + 1 & \text{if } w_j > 0, \\ \nabla_j L(\boldsymbol{w}) - 1 & \text{if } w_j < 0, \\ \text{sgn}(\nabla_j L(\boldsymbol{w})) \max(|\nabla_j L(\boldsymbol{w})| - 1, 0) & \text{otherwise.} \end{cases}$$

Since $\nabla^S f(\boldsymbol{w}) = \mathbf{0}$ if and only if $\boldsymbol{w}$ is optimal, we can check $\|\nabla^S f(\boldsymbol{w})\|$ for the stopping condition. Figure 2 shows the scaled 2-norm of the minimum-norm sub-gradient,

$$\frac{l}{\min(\sum_{i:y_i=1} 1, \sum_{i:y_i=-1} 1)\|\nabla^S f(\boldsymbol{w}^1)\|} \|\nabla^S f(\boldsymbol{w})\|, \tag{83}$$

along the training time. From Figure 2, CDN and BBR are the best in the early stage of the procedure, while Newton and quasi Newton methods such as TRON and OWL-QN have faster local convergence. This observation is consistent with Figure 1. Some methods (e.g., decomposition methods) do not calculate $\nabla^S f(\boldsymbol{w})$ in their procedures, so a gradient-based stopping condition may introduce extra cost. However, these methods may be able to use approximate gradient values obtained during the calculation. For example, coordinate descent methods calculate only $\nabla_j f(\boldsymbol{w}^{k,j})$, $j = 1, \ldots, n$ instead of $\nabla f(\boldsymbol{w}^{k+1})$, but these values can be directly used in a stopping condition; see details in Appendix F of Fan et al. (2008).

In Figure 3, we investigate how the testing accuracy is improved along the training time. The testing accuracy is the percentage of correct predictions on the testing set. Results show



(a) a9a (b) real-sim (c) news20

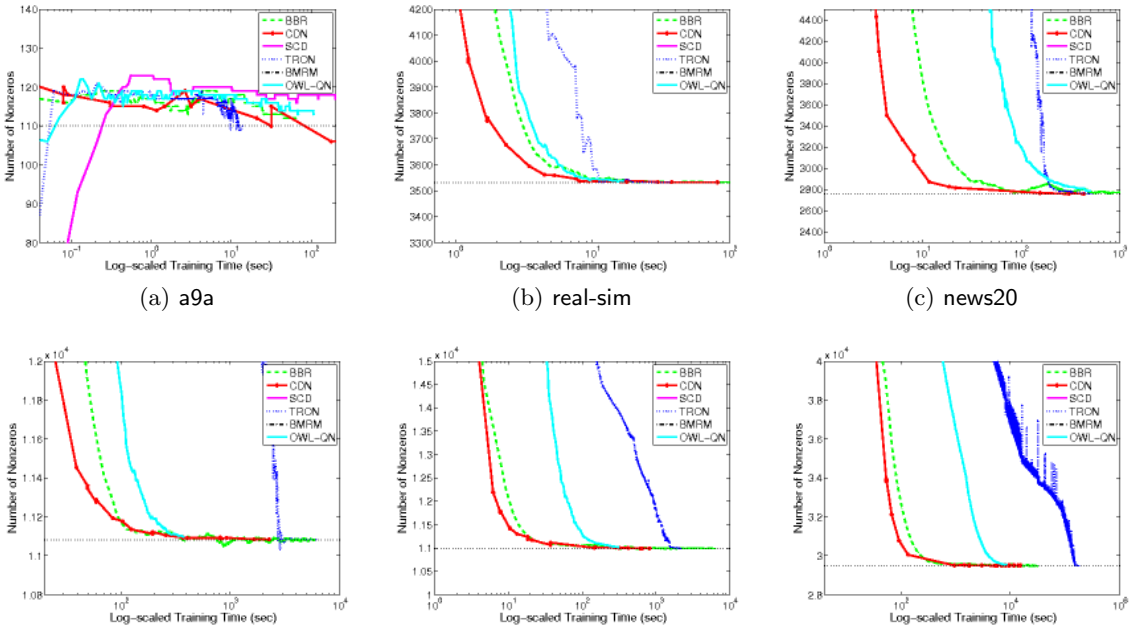
(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 3: The testing accuracy versus training time (log-scaled). Logistic regression without using the bias term is solved. Curves of BMRM may not be shown because values are out of the range.

that BBR and CDN achieve the final accuracy more quickly. In addition, we are interested in the progress of these methods on gaining the sparsity. Figure 4 shows the number of non-zero coefficients of $\boldsymbol{w}$ and the training time. For most data sets, BBR and CDN more efficiently obtain a sparse solution.

Next, we compare methods solving (5) with the bias term. These solvers include CDN, CGD-GS, IPM, Lassplore, and GLMNET. Although BBR can solve (5), we omit it in the comparison as its performance is similar to that of solving (1). Following the comparison for methods solving (1), we begin with checking the running time to reduce the relative error of the function value and the scaled norm of the minimum-norm sub-gradient; see (82) and (83), respectively. The results are given in Figures 5 and 6. The reference value f^* is obtained using IPM with a strict stopping condition. From Figure 5, CDN is the fastest, IPM comes the second, and CGD-GS is the third. However, in Figure 6 for reducing the gradient norm, IPM often surpasses CDN in the final stage.

GLMNET gives good performances in Figures 5(b)–5(d). In particular, it has fast local convergence; see curves that are close to vertical in Figures 5(c) and 5(d). This fast local convergence is due to using the quadratic approximation $q_k(\boldsymbol{d})$, which generates a Newton-like direction. We find that the approximation of replacing D in (69) with a constant diagonal matrix is effective for micro-array data used in Friedman et al. (2010). However, for large document sets, using the exact Hessian is better. Unfortunately, GLMNET fails to generates results for yahoo-japan and yahoo-korea after sufficient running time. We find that setting an appropriate stopping tolerance for minimizing the quadratic approximation



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

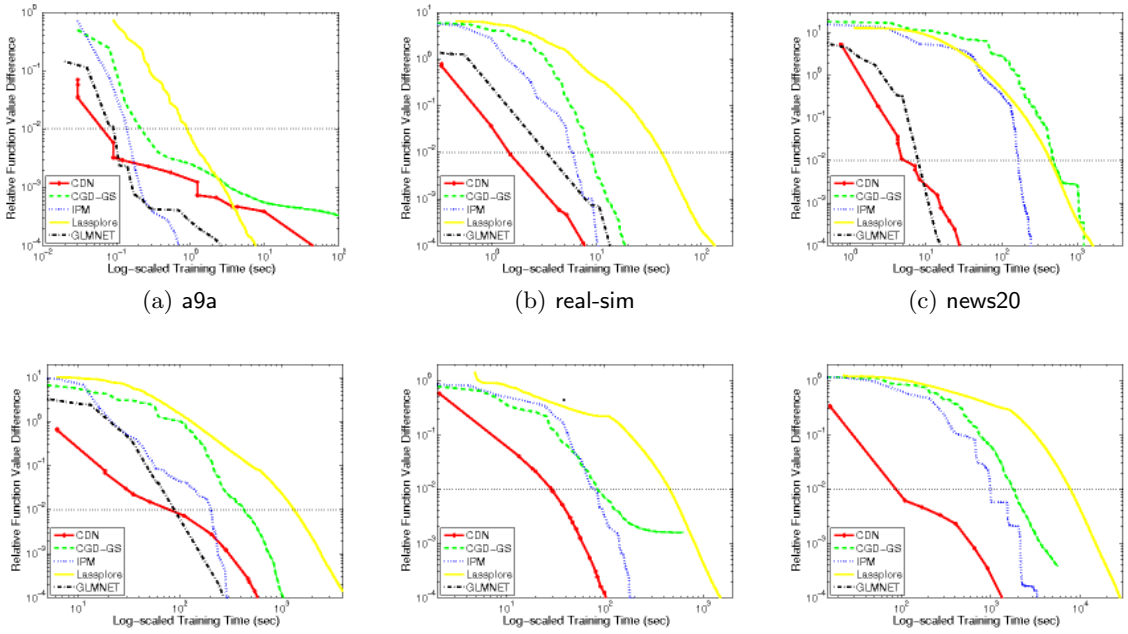
Figure 4: The number of non-zero coefficients versus training time (log-scaled). Logistic regression without using the bias term is solved. Curves of BMRM may not be shown because values are out of the range. The solid horizontal line indicates the final number of non-zero coefficients.

in (71) is sometimes difficult. This issue might cause the problem in training yahoo-japan and yahoo-korea. For IPM, the performance is in general competitive. Because IPM is a type of Newton method, it has fast local convergence.

The result that CDN is faster than CGD-GS is worth for further discussion. Tseng and Yun (2009) show that for some least-square regression problems, an implementation similar to CDN (i.e., a Gauss-Seidel rule for selecting working variables) is slower than CGD-GS, which selects variables using the gradient information. This result is opposite to ours. Two issues might cause the different results. First, problems are different. We aim at classifying large and sparse document data, but they solve regression problems. Second, we implement two techniques (permutation of sub-problems and shrinking) to improve the convergence.

Yun and Toh (2011) show that for some document data sets used here, their CGD-GS is faster than IPM. However, our results indicate that IPM is better. This difference is apparently due to that Yun and Toh (2011) run an earlier version (0.8.1) of IPM. We indicate in Section 8.2 that a minor problem in this version causes this version to be two or three times slower than a later version (0.8.2) used here.

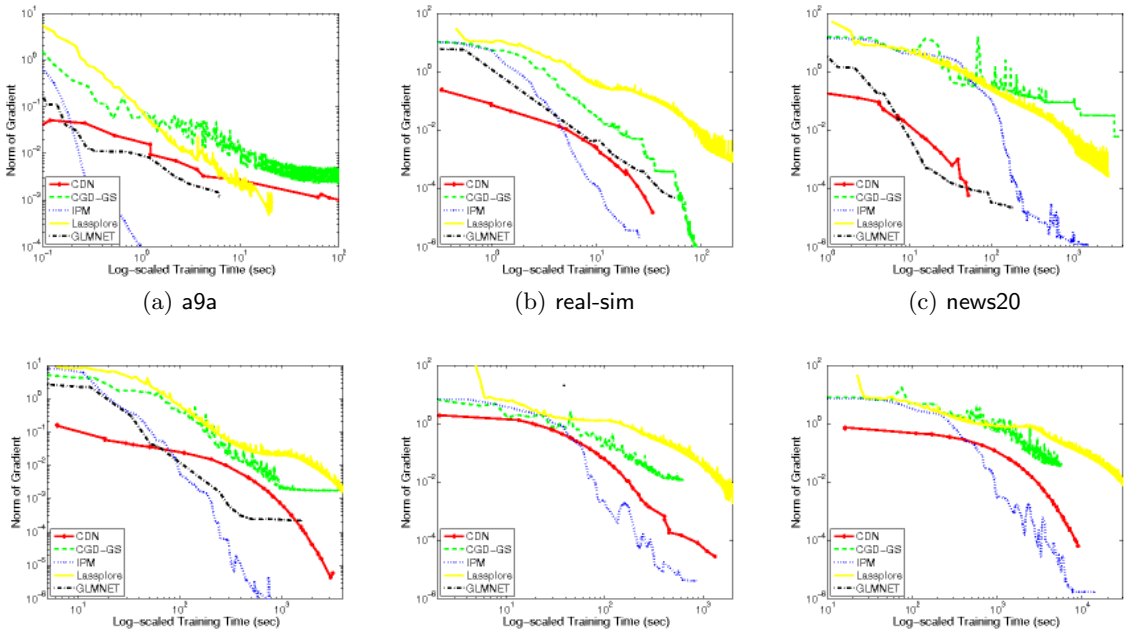
Figure 7 indicates the testing accuracy versus the training time. We can see in Figures 7(e) and 7(f) that IPM's accuracy may not improve as the training time increases. As we mentioned in Section 8.2, this result is because we modify certain $\boldsymbol{w}$ elements to zero. In the middle of the procedure, $\boldsymbol{w}$ is not close to an optimum yet, but many elements have satisfied $|\nabla_j L(\boldsymbol{w})| \leq 0.9999$ and are trimmed to zero. Hence, the resulting accuracy may be worse



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

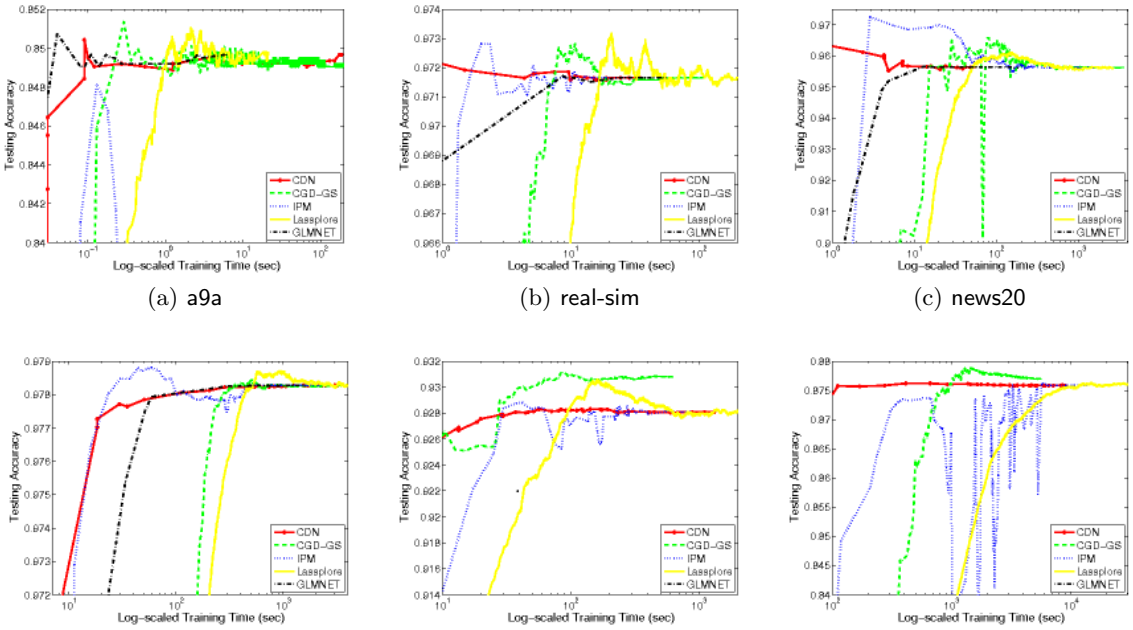
Figure 5: Relative difference between the objective function and the optimum value, versus training time. Logistic regression with the bias term is solved. Both x-axis and y-axis are log-scaled.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 6: The 2-norm of the minimum-norm sub-gradient (83) versus training time. Logistic regression with the bias term is solved. Both x-axis and y-axis are log-scaled.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 7: The testing accuracy versus training time (log-scaled). Logistic regression with the bias term is solved.

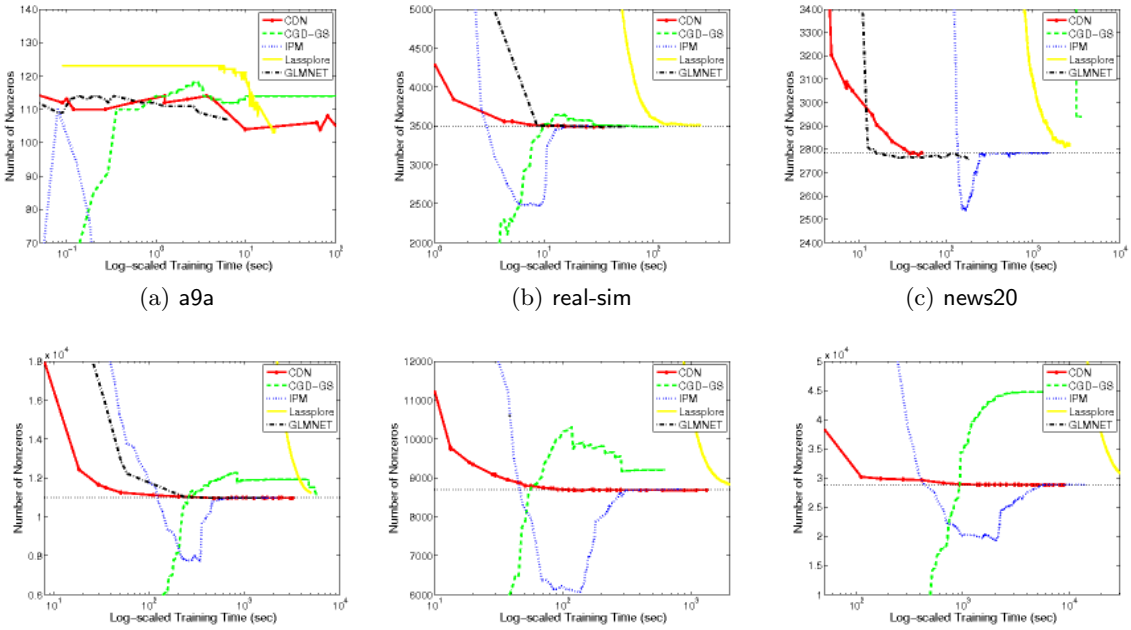
than that in the early stage of the procedure. In fact, due to IPM's dense $\boldsymbol{w}$ throughout iterations, to get the final sparsity and the testing accuracy, we need to accurately solve the optimization problem.

Figure 8 presents the number of $\boldsymbol{w}$'s non-zero coefficients. Similar to methods solving (1), all methods here, except CGD-GS, have solutions with many non-zero coefficients in the beginning and gradually gain the sparsity. In contrast, CGD-GS's numbers of non-zero coefficients in the whole optimization process are not much more than those of the final solutions. We find that this nice property is from using the gradient information for selecting working variables. In contrast, without the gradient information, CDN wrongly updates some elements of $\boldsymbol{w}$ to be non-zeros in the early stage of the procedure.

In summary, for large document data, coordinate descents methods such as CDN perform well in the early stage of the optimization procedure. As a result, they achieve the final accuracy more quickly. However, for some applications, a correct sparsity pattern of the model is important and a more accurate solution is sought. Then, GLMNET by combining both Newton-type and coordinate descent approaches is useful.

8.4 Comparing Methods for L1-regularized L2-loss SVMs and Investigating CDN's Shrinking Strategy

In Section 7, we have extended CDN and TRON to handle L2 loss, so methods included for comparison are CDN, TRON, and BMRM. Note that we solve (77) without considering the bias term.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

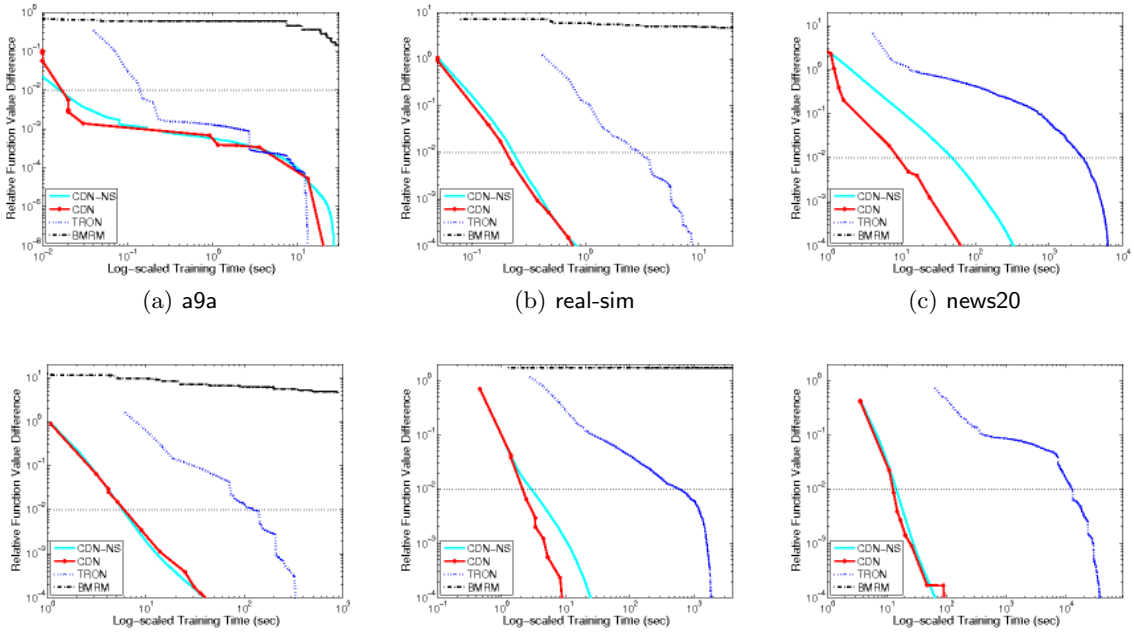
Figure 8: The number of non-zero coefficients versus training time (log-scaled). Logistic regression with the bias term is solved. The solid horizontal line indicates the final number of non-zero coefficients.

Following the experiment for logistic regression, we plot the relative difference to the optimal function value in Figure 9 and the scaled norm of the minimum-norm sub-gradient in Figure 10. The reference f^* is obtained by running TRON with a strict stopping condition. One can see that CDN's training time is the shortest among the three solvers and TRON is better than BMRM. BMRM does not properly decrease the function value and the norm of gradient on large data sets.

In Figures 9 and 10, we also present results of CDN without implementing the shrinking strategy (denoted as CDN-NS). In most cases, the implementation with shrinking is only slightly better. However, shrinking is effective if the sparsity is high (e.g., news20 and yahoo-japan). In such a situation, most $\boldsymbol{w}$ components are zero. We can safely remove some zero components and more efficiently solve smaller optimization problems.

9. Discussions and Conclusions

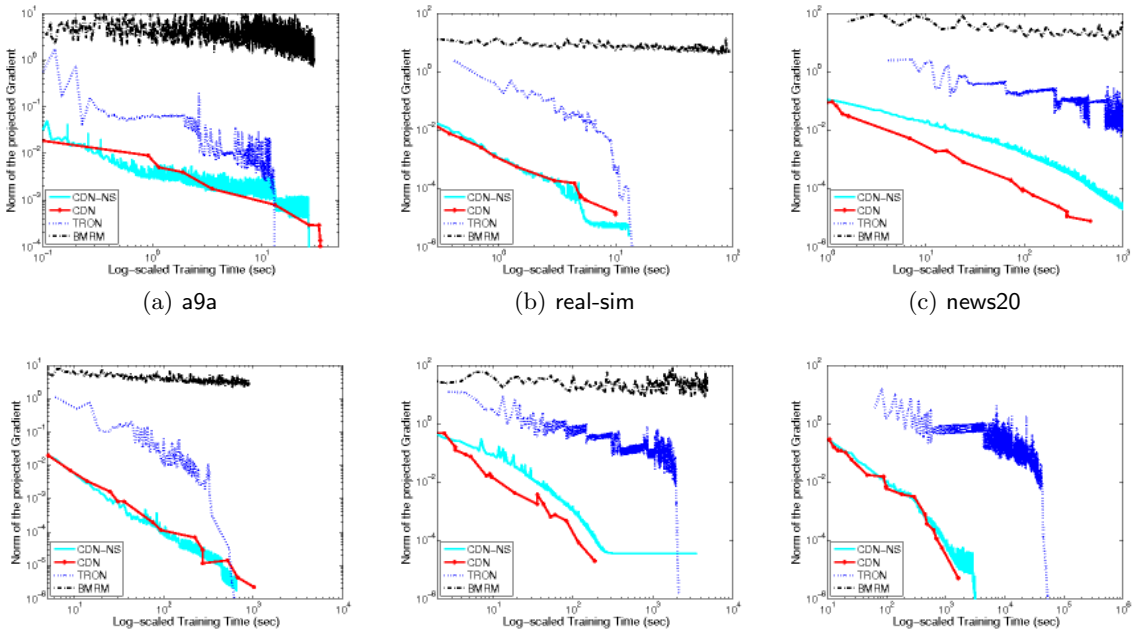
In Section 2.5, we briefly discuss optimization methods for L1-regularized least-square regression. Some comparisons can be seen in, for example, the experiment section of Wright et al. (2009) and Yun and Toh (2011). Note that an optimization method may perform differently on classification and regression problems. For instance, Yun and Toh (2011) show that CGD-GS is faster than CDN for regression problems, but here we have an opposite observation for document classification.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

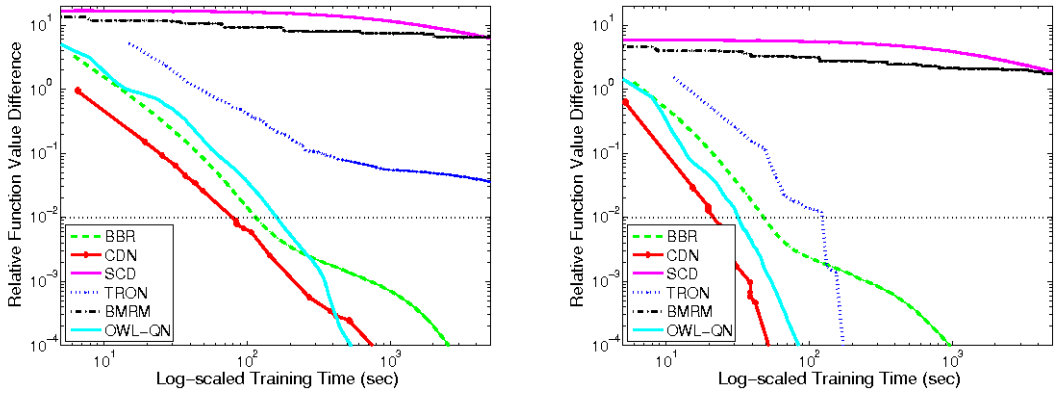
Figure 9: Relative difference between the objective function and the optimum value, versus training time. L2-loss SVM without using the bias term is solved. Both x-axis and y-axis are log-scaled.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 10: The 2-norm of the minimum-norm sub-gradient (83) versus training time. L2-loss SVM without using the bias term is solved. Both x-axis and y-axis are log-scaled.



(a) rcv1 with $10C^*$ (b) rcv1 with $0.1C^*$

Figure 11: A comparison of logistic regression solvers with regularization parameters $10C^*$ and $0.1C^*$. Both x-axis (relative difference between the objective function and the optimum value) and y-axis (training time) are log-scaled.

Figures 1–8 indicate that CDN is faster for solving (1) than (5). Our past work (Huang et al., 2010, Section 5) has indicated that the bias term may affect the running time of the same optimization method. As the accuracy does not differ much, it seems that in general the bias term should not be considered.

Among quasi Newton and Newton approaches, IPM and OWL-QN are faster than TRON. This result seems to indicate that TRON suffers from the difficulty for finding $\boldsymbol{w}$'s final non-zero coefficients. However, IPM's $\boldsymbol{w}$ elements are all non-zeros, so we must accurately solve the optimization problem before trimming some elements to zero.

Figures 9 and 10 indicate that CDN/TRON for L2-loss SVMs is faster than CDN/TRON for logistic regression. Each iteration of CDN/TRON for L2-loss SVMs is cheaper because no exp/log operations are involved. Because the accuracy is similar, in general we prefer L2-loss SVMs over logistic regression.

The choice of the regularization parameter C affects the performance of solvers. In Figure 11, we present results on the rcv1 data set with regularization parameters $10C^*$ and $0.1C^*$, respectively, where $C^* = 4$ is the best parameter obtained from cross validation; see Table 2. Results show that all solvers take longer running time when C is large. Therefore, one should avoid a value much larger than C^* by trying from a small C. Moreover, methods using low-order (e.g., gradient only) information seem to be more sensitive to the change of C. For example, with C^* and $0.1C^*$ we respectively observe in Figures 1(d) and 11(b) that CDN is faster than OWL-QN, but with $10C^*$, OWL-QN surpasses CDN in the final stage.

GLMNET iteratively considers quadratic approximations and applies coordinate descent methods at each iteration. The discussion in Section 8.3 indicates that GLMNET's speed may be further improved if an adaptive stopping condition is properly designed for minimizing each quadratic approximation.

It is challenging to efficiently solve L1-regularized linear classification problems. The 1-norm term causes the non-differentiability of the objective function. In this paper, we review

many existing methods for logistic regression and L2-loss SVMs. We discuss some state-of-the-art methods in detail. Our extensive comparison shows that carefully implemented coordinate descent methods are effective for L1-regularized classification with large-scale document data.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grant 98-2221-E-002-136-MY3. The authors thank associate editor, reviewers, Trevor Hastie, Robert Tibshirani, and Jun Liu for valuable comments.

Appendix A. Existence of Optimal Solutions of (1)

Consider the following level set:

$$S \equiv \{\boldsymbol{w} \mid f(\boldsymbol{w}) \leq f(\mathbf{0})\}.$$

We prove that S is compact (closed and bounded). Clearly, S is closed due to the continuity of $f(\boldsymbol{w})$. If S is not bounded, then there is a sequence $\{\boldsymbol{w}^k\} \subset S$ such that $\|\boldsymbol{w}^k\|_1 \to \infty$. Because we assume that $\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i) \geq 0$, $f(\boldsymbol{w}^k) \geq \|\boldsymbol{w}^k\|_1$. Then, $f(\boldsymbol{w}^k) \to \infty$ contradicts $f(\boldsymbol{w}^k) \leq f(\mathbf{0})$. Thus, S is a compact set. From Weierstrass' Theorem, $f(\boldsymbol{w})$ has at least one minimum in S.

Appendix B. Solution of (26)

We consider the following general form with $A > 0$:

$$\min_z \quad |w_j + z| + Bz + \frac{1}{2}Az^2. \tag{84}$$

Clearly,

$$|w_j + z| + Bz + \frac{1}{2}Az^2 = \begin{cases} g_1(z) & \text{if } z \geq -w_j, \\ g_2(z) & \text{if } z \leq -w_j, \end{cases}$$

where

$$g_1(z) \equiv w_j + z + Bz + \frac{1}{2}Az^2 \quad \text{and} \quad g_2(z) \equiv -w_j - z + Bz + \frac{1}{2}Az^2.$$

By checking if the minimum of the quadratic function occurs on the right or the left side of $-w_j$, we know

$$\arg\min_{z \geq -w_j} g_1(z) = \begin{cases} -\frac{B+1}{A} & \text{if } B + 1 \leq Aw_j, \\ -w_j & \text{otherwise,} \end{cases}$$

and

$$\arg\min_{z \leq -w_j} g_2(z) = \begin{cases} -\frac{B-1}{A} & \text{if } B - 1 \geq Aw_j, \\ -w_j & \text{otherwise.} \end{cases}$$

Because $B+1 \le Aw_j$ and $B-1 \ge Aw_j$ cannot hold at the same time, and $g_1(-w_j) = g_2(-w_j)$, we can easily conclude that the solution of (84) is

$$\begin{cases} -\frac{B+1}{A} & \text{if } B+1 \le Aw_j, \\ -\frac{B-1}{A} & \text{if } B-1 \ge Aw_j, \\ -w_j & \text{otherwise.} \end{cases}$$

Appendix C. Proof of Theorem 1

From the assumption that $\boldsymbol{w}^k \to \boldsymbol{w}^*$ and the way that $\boldsymbol{w}$ is cyclically updated, we have $\boldsymbol{w}^{k,j} \to \boldsymbol{w}^*$ as well. The first result $-1 < \nabla_j L(\boldsymbol{w}^{k,j}) < 1$ immediately follows from the assumption $-1 < \nabla_j L(\boldsymbol{w}^*) < 1$ and the continuity of $\nabla L(\boldsymbol{w})$.

We then focus on the second result to show that $w_j^{k,j} = 0$ after k is large enough. The optimality condition (6) and the assumption $-1 < \nabla_j L(\boldsymbol{w}^*) < 1$ imply that $w_j^* = 0$. From the continuity of $\nabla^2 L(\boldsymbol{w})$ and the compactness of the level set S proved in Appendix A, we can define

$$M \equiv \max\{\nabla^2_{jj} L(\boldsymbol{w}) \mid \boldsymbol{w} \in S\} \ge 0. \tag{85}$$

With the assumption $-1 < \nabla_j L(\boldsymbol{w}^*) < 1$ and the property $w_j^{k,j} \to w_j^* = 0$, for any $\sigma \in (0,1)$, there exists an iteration index K_j such that for all $k \ge K_j$,

$$-1 + \frac{M}{1-\sigma}|w_j^{k,j}| \le L_j'(0; \boldsymbol{w}^{k,j}) = \nabla_j L(\boldsymbol{w}^{k,j}) \le 1 - \frac{M}{1-\sigma}|w_j^{k,j}| \tag{86}$$

and

$$\left\{ \boldsymbol{w} \;\middle|\; \|\boldsymbol{w} - \boldsymbol{w}^{k,j}\| \le \|\boldsymbol{w}^{k,j} - \boldsymbol{w}^*\| \text{ for some } k \ge K_j \right\} \subset S.^8 \tag{87}$$

We prove that $w_j^{k,j} = 0,\ \forall k \ge K_j$. Recall that in the decomposition method we use (27) to obtain a direction d. From (86), we see that at $k = K_j$, the third case in (27) is taken. That is, $d = -w_j^{k,j}$. If σ in (86) is chosen to be the constant used in the sufficient decrease condition (28), we claim that $d = -w_j^{k,j}$ satisfies (28) with $\lambda = 1$: If $w_j^{k,j} \ge 0$,

$$\begin{aligned} & g_j(-w_j^{k,j}) - g_j(0) - \sigma\left(-w_j^{k,j} - L_j'(0; \boldsymbol{w}^{k,j}) w_j^{k,j}\right) \\ &= (1-\sigma)\left(-w_j^{k,j} - L_j'(0; \boldsymbol{w}^{k,j}) w_j^{k,j}\right) + \frac{1}{2} L_j''(\tilde{z}; \boldsymbol{w}^{k,j})(w_j^{k,j})^2 && (88) \\ &\le (1-\sigma)\left(-w_j^{k,j} - L_j'(0; \boldsymbol{w}^{k,j}) w_j^{k,j}\right) + \frac{1}{2} M (w_j^{k,j})^2 && (89) \\ &\le \frac{-1}{2} M (w_j^{k,j})^2 \le 0, && (90) \end{aligned}$$

where $\tilde{z}$ is between 0 and $-w_j^{k,j}$, Equation (88) follows from (20)–(21), Equation (89) is from (87) and (85),[9] and Equation (90) is from the first inequality of (86). The above result

8. We need $f(\boldsymbol{w}^*) < f(\boldsymbol{0})$. This property generally holds. If not, we can slightly enlarge S so that (87) and all subsequent derivations still follow.

9. More precisely,
$$\|\boldsymbol{w}^{k,j} + \tilde{z}\boldsymbol{e}_j - \boldsymbol{w}^{k,j}\| \le |w_j^{k,j}| = |w_j^{k,j} - w_j^*| \le \|\boldsymbol{w}^{k,j} - \boldsymbol{w}^*\|.$$

is precisely the sufficient decrease condition (28) with $\lambda = 1$. The situation for $w_j^{k,j} < 0$ is similar. By taking the step $d = -w_j^{k,j}$, we have $w_j^{k+1,j} = 0$. At the $(k+1)$st iteration, $w_j^{k+1,j} = 0$ and (86) imply that the optimality condition (18) for the sub-problem has been satisfied. Thus, the jth element of $\boldsymbol{w}$ remains zero in all subsequent iterations.

Appendix D. Convergence of CDN: Logistic Regression

If each time one variable is used, the sub-problem considered in Tseng and Yun (2009) is

$$\min_z \quad |w_j + z| - |w_j| + L'_j(0)z + \frac{1}{2}Hz^2. \tag{91}$$

Clearly, (26) is a special case of (91) by taking $H = L''_j(0)$. Therefore, we can apply results proved in Tseng and Yun (2009).

To have the finite termination of the line search procedure, Tseng and Yun (2009, Lemma 5) require that there exists $A > 0$ such that

$$\|\nabla L(\boldsymbol{w}_1) - \nabla L(\boldsymbol{w}_2)\| \le A\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|, \ \forall \boldsymbol{w}_1, \boldsymbol{w}_2 \in R^n \tag{92}$$

and

$$L''_j(0; \boldsymbol{w}) > 0, \tag{93}$$

where $\boldsymbol{w}$ is the current solution used to generate the direction d in (27). Equation (92) means that $\nabla L(\cdot)$ is globally Lipschitz continuous. For logistic regression, we have

$$\|\nabla L(\boldsymbol{w}_1) - \nabla L(\boldsymbol{w}_2)\| \le \|\nabla^2 L(\tilde{\boldsymbol{w}})\|\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|,$$

where $\tilde{\boldsymbol{w}}$ is between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. Moreover,

$$\|\nabla^2 L(\tilde{\boldsymbol{w}})\| = C\|X^T D(\tilde{\boldsymbol{w}})X\| \le C\|X^T\|\|X\|, \tag{94}$$

where $D(\tilde{\boldsymbol{w}}) \in R^{l\times l}$ is a diagonal matrix similar to (70). Because any diagonal element of $D(\tilde{\boldsymbol{w}})$ is less than one, we have the inequality in (94). Thus, we can use $C\|X^T\|\|X\|$ as the constant A in (92).

For (93), from (29), $L''_j(0; \boldsymbol{w})$ in the level set S is lower bounded by a positive value. The only exception is that $L''_j(0; \boldsymbol{w}) = 0$ when $x_{ij} = 0, \ \forall i = 1, \ldots, l$. In this situation, $L'_j(0; \boldsymbol{w}) = 0$ and the optimal $w_j^* = 0$. The one-variable function $g_j(z)$ is reduced to

$$g_j(z) = |w_j + z| - |w_j|,$$

so $d = -w_j$ from (27) and d satisfies the sufficient decrease condition (28). Therefore, w_j becomes zero after the first iteration and is not changed after. It is like that the jth feature has been removed in the beginning.

Next, we discuss the asymptotic convergence of function values. Tseng and Yun (2009) impose certain conditions on choosing the working set; see Equation (12) in their paper. If one variable is updated at a time, their condition is reduced to that between $\boldsymbol{w}^k$ and $\boldsymbol{w}^{k+1}$, one must go through n sub-problems covering all $w_1, \ldots, w_n$. This setting is exactly what we do, regardless of using a sequential order of $1, \ldots, n$ or a permutation $\pi(1), \ldots, \pi(n)$.

Tseng and Yun (2009) need an additional assumption for the asymptotic convergence (see Assumption 1 in their paper):

$$0 < \lambda_{\min} \le H^{k,j} \le \lambda_{\max}, \forall j = 1, \ldots, n, k = 1, \ldots, \tag{95}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are positive constants, and $H^{k,j} = \nabla^2_{jj} L(\boldsymbol{w}^{k,j})$ is the coefficient H in (91) at $\boldsymbol{w}^{k,j}$. From (29) and the boundedness of the level set S (see Appendix A), this assumption holds except that for some j, $x_{ij} = 0, \forall i$. For such j, $\nabla_j L(\boldsymbol{w}) = 0, \forall \boldsymbol{w}$. Hence, w_j becomes zero by (27) at the first iteration and is not changed subsequently. Because $w_j = 0$ is optimal in this situation, it is like that the jth variable has been removed for optimization. Therefore, we have (95) without problems.

Following Tseng and Yun (2009, Theorem 1(e)), any limit point of $\{\boldsymbol{w}^k\}$ is an optimum of (1) with logistic loss.

Appendix E. Convergence of TRON: Logistic Regression

Consider any limit point $\bar{\boldsymbol{w}}$ generated by $\{\boldsymbol{w}^k\}$. Lin and Moré (1999) proved that if

$$\nabla^2 \bar{f}(\bar{\boldsymbol{w}})_{J,J} \text{ is positive definite,} \tag{96}$$

where

$$J \equiv \{j \mid \nabla_j \bar{f}(\bar{\boldsymbol{w}}) = 0\},$$

then the following two results hold:

1. $\{\boldsymbol{w}^k\}$ globally converges to $\bar{\boldsymbol{w}}$.
2. $\{\boldsymbol{w}^k\}$ quadratically converges. That is, there exist $\mu \in (0, 1)$ and a positive integer K such that
$$\|\boldsymbol{w}^{k+1} - \bar{\boldsymbol{w}}\| \le (1 - \mu)\|\boldsymbol{w}^k - \bar{\boldsymbol{w}}\|^2, \ \forall k > K.$$

The remaining question is whether (96) holds. From (58), we have

$$\nabla^2 \bar{f}(\bar{\boldsymbol{w}}) = C \begin{bmatrix} X^T \\ -X^T \end{bmatrix} D \begin{bmatrix} X & -X \end{bmatrix}, \tag{97}$$

where D is defined in (57). $\nabla^2 \bar{f}(\bar{\boldsymbol{w}})$ cannot be positive definite if there exists $1 \le j \le n$ such that $\{j, j+n\} \subset J$. The reason is that one can easily find a vector $\boldsymbol{v} \in R^{2n}$ such that $\boldsymbol{v}^T \nabla^2 \bar{f}(\bar{\boldsymbol{w}}) \boldsymbol{v} = 0$. For example, let $v_j = v_{j+n} \ne 0$ and other elements be zero. However, we claim that $\{j, j+n\} \subset J$ does not happen. Otherwise,

$$\begin{aligned} \nabla_j \bar{f}(\bar{\boldsymbol{w}}) &= 1 + \nabla_j L(\bar{\boldsymbol{w}}_{1:n} - \bar{\boldsymbol{w}}_{n+1:2n}) \quad \text{and} \\ \nabla_{j+n} \bar{f}(\bar{\boldsymbol{w}}) &= 1 - \nabla_j L(\bar{\boldsymbol{w}}_{1:n} - \bar{\boldsymbol{w}}_{n+1:2n}) \end{aligned}$$

are both zero, but this situation is not possible.

From the optimality condition (6), J is the same as the following set:

$$\{j \mid \bar{w}_j > 0 \text{ or } \bar{w}_j = \nabla_j \bar{f}(\bar{\boldsymbol{w}}) = 0\}.$$

Therefore, if the solution is sparse and those $\bar{w}_j = 0$ or $\bar{w}_{j+n} = 0$, $1 \le j \le n$ satisfy

$$\begin{cases} \nabla_j \bar{f}(\bar{\boldsymbol{w}}) > 0 & \text{if } \bar{w}_j = 0, \\ \nabla_{j+n} \bar{f}(\bar{\boldsymbol{w}}) > 0 & \text{if } \bar{w}_{j+n} = 0, \end{cases}$$

(i.e., the optimization problem is non-degenerate), then $|J|$ is small. From (97), $\nabla^2 \bar{f}(\bar{\boldsymbol{w}})_{J,J}$ tends to be positive definite if $|J|$ is small. Then, we can have the quadratic convergence.

Appendix F. Convergence of CDN: L2-loss SVM

To have the finite termination of the line search procedure, we need conditions similar to (92) and (93). The condition (92) means that $\nabla L(\boldsymbol{w})$ is globally Lipschitz continuous. L2-loss SVM satisfies this property (Mangasarian, 2002, Section 3). For (93), the setting in (81) ensures that H in (91) is always positive. Therefore, the line search procedure stops in a finite number of steps.

For the asymptotic convergence, we need again the condition (95). Our setting in (81) meets this assumption, so any limit point of $\{\boldsymbol{w}^k\}$ is an optimal solution of (77).

References

Galen Andrew and Jianfeng Gao. Scalable training of L1-regularized log-linear models. In *Proceedings of the Twenty Fourth International Conference on Machine Learning (ICML)*, 2007.

Suhrid Balakrishnan and David Madigan. Algorithms for sparse linear classifiers in the massive data setting. 2005. URL `http://www.stat.rutgers.edu/~madigan/PAPERS/sm.pdf`.

Steven Benson and Jorge J. Moré. A limited memory variable metric method for bound constrained minimization. Preprint MCS-P909-0901, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 2001.

Paul S. Bradley and Olvi L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 1998.

Richard H. Byrd, Peihung Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190–1208, 1995.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale L2-loss linear SVM. *Journal of Machine Learning Research*, 9:1369–1398, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cdl2.pdf`.

David L. Donoho and Yaakov Tsaig. Fast solution of l1 minimization problems when the solution may be sparse. *IEEE Transactions on Information Theory*, 54:4789–4812, 2008.

John Duchi and Yoram Singer. Boosting with structural sparsity. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML)*, 2009.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

Mário A. T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1150–1159, 2003.

Mário A. T. Figueiredo, Robert Nowak, and Stephen Wright. Gradient projection for sparse reconstruction: Applications to compressed sensing and other inverse problems. *IEEE Journal on Selected Topics in Signal Processing*, 1:586–598, 2007.

Jerome H. Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

Wenjiang J. Fu. Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

Glenn M. Fung and Olvi L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational optimization and applications*, 28:185–202, 2004.

Eli M. Gafni and Dimitri P. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22:936–964, 1984.

Alexandar Genkin, David D. Lewis, and David Madigan. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

Joshua Goodman. Exponential priors for maximum entropy models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2004.

Elaine T. Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for l1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf`.

Fang-Lan Huang, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Iterative scaling and coordinate descent methods for maximum entropy. *Journal of Machine Learning Research*, 11:815–848, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/maxent_journal.pdf`.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–184, Cambridge, MA, 1998. MIT Press.

Jun'ichi Kazama and Jun'ichi Tsujii. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 137–144, 2003.

S. Sathiya Keerthi and Shrish Shevade. A fast tracking algorithm for generalized LARS/LASSO. *IEEE Transactions on Neural Networks*, 18(6):1826–1830, 2007.

Jinseog Kim, Yuwon Kim, and Yongdai Kim. A gradient-based optimization algorithm for LASSO. *Journal of Computational and Graphical Statistics*, 17(4):994–1009, 2008.

Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior point method for large-scale l1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1:606–617, 2007.

Yongdai Kim and Jinseog Kim. Gradient LASSO for feature selection. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–63, 1997.

Kwangmoo Koh, Seung-Jean Kim, and Stephen Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007. URL `http://www.stanford.edu/~boyd/l1_logistic_reg.html`.

Balaji Krishnapuram, Alexander J. Hartemink, Lawrence Carin, and Mário A. T. Figueiredo. A Bayesian approach to joint feature selection and classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1105–1111, 2004.

Balaji Krishnapuram, Lawrence Carin, Mário A. T. Figueiredo, and Alexander J. Hartemink. Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.

Jussi Kujala, Timo Aho, and Tapio Elomaa. A walk from 2-norm SVM to 1-norm SVM. Preprint available from `http://www.cs.tut.fi/~kujala2/papers/1norm2norm.pdf`, 2009.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:771–801, 2009.

Cheng-Yu Lee. A comparison of optimization methods for large-scale l1-regularized logistic regression. Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2008.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. Efficient l1 regularized logistic regression. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, pages 1–9, Boston, MA, USA, July 2006.

Chih-Jen Lin and Jorge J. Moré. Newton's method for large-scale bound constrained problems. *SIAM Journal on Optimization*, 9:1100–1127, 1999.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

Jun Liu and Jieping Ye. Efficient euclidean projections in linear time. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML)*, 2009.

Jun Liu, Jianhui Chen, and Jieping Ye. Large-scale sparse logistic regression. In *Proceedings of The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 547–556, 2009.

Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Olvi L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

Yurii E. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2003.

Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20:389–404, 2000a.

Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000b.

Mee Young Park and Trevor Hastie. L1 regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society Series B*, 69:659–677, 2007.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.

Saharon Rosset. Following curved regularized optimization solution paths. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1153–1160, Cambridge, MA, 2005. MIT Press.

Volker Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1): 16–28, 2004.

Sylvain Sardy, Andrew G. Bruce, and Paul Tseng. Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries. *Journal of Computational and Graphical Statistics*, 9:361–379, 2000.

Mark Schmidt, Glenn Fung, and Romer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *Proceedings of European Conference on Machine Learning*, pages 286–297, 2007.

Mark Schmidt, Glenn Fung, and Romer Rosales. Optimization methods for l1-regularization. Technical Report TR-2009-19, University of British Columbia, 2009.

Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1 regularized loss minimization. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML)*, 2009.

Shirish Krishnaj Shevade and S. Sathiya Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.

Jianing Shi, Wotao Yin, Stanley Osher, and Paul Sajda. A fast hybrid algorithm for large scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 11:713–741, 2010.

Choon Hui Teo, S.V.N. Vishwanathan, Alex Smola, and Quoc V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

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

Ryota. Tomioka and Masashi Sugiyama. Dual augmented Lagrangian method for efficient sparse reconstruction. *IEEE Signal Processing Letters*, 16(12):1067–1070, 2009.

Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2009.

Stephen J. Wright, Robert D. Nowak, and Mário A.T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57:2479–2493, 2009.

Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

Jin Yu, S.V.N. Vishwanathan, Simon Gunter, and Nicol N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization problems in machine learning. *Journal of Machine Learning Research*, 11:1–57, 2010.

Sangwoon Yun and Kim-Chuan Toh. A coordinate gradient descent method for l1-regularized convex minimization. *Computational Optimizations and Applications*, 48(2): 273–307, 2011.

Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.

Peng Zhao and Bin Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

Peng Zhao and Bin Yu. Stagewise lasso. *Journal of Machine Learning Research*, 8:2701–2726, 2007.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.