Journal of Machine Learning Research 13 (2012) 1999-2030 Submitted 5/11; Revised 1/12; Published 6/12

# An Improved GLMNET for L1-regularized Logistic Regression

**Guo-Xun Yuan** R96042@CSIE.NTU.EDU.TW
**Chia-Hua Ho** B95082@CSIE.NTU.EDU.TW
**Chih-Jen Lin** CJLIN@CSIE.NTU.EDU.TW
*Department of Computer Science*
*National Taiwan University*
*Taipei 106, Taiwan*

**Editor:** S. Sathiya Keerthi

## Abstract

Recently, Yuan et al. (2010) conducted a comprehensive comparison on software for L1-regularized classification. They concluded that a carefully designed coordinate descent implementation CDN is the fastest among state-of-the-art solvers. In this paper, we point out that CDN is less competitive on loss functions that are expensive to compute. In particular, CDN for logistic regression is much slower than CDN for SVM because the logistic loss involves expensive exp/log operations.

In optimization, Newton methods are known to have fewer iterations although each iteration costs more. Because solving the Newton sub-problem is independent of the loss calculation, this type of methods may surpass CDN under some circumstances. In L1-regularized classification, GLMNET by Friedman et al. is already a Newton-type method, but experiments in Yuan et al. (2010) indicated that the existing GLMNET implementation may face difficulties for some large-scale problems. In this paper, we propose an improved GLMNET to address some theoretical and implementation issues. In particular, as a Newton-type method, GLMNET achieves fast local convergence, but may fail to quickly obtain a useful solution. By a careful design to adjust the effort for each iteration, our method is efficient for both loosely or strictly solving the optimization problem. Experiments demonstrate that our improved GLMNET is more efficient than CDN for L1-regularized logistic regression.

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

## 1. Introduction

Logistic regression and support vector machines (SVM) are popular classification methods in machine learning. Recently, L1-regularized logistic regression and SVM are widely used because they can generate a sparse model. Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, l$, $\boldsymbol{x}_i \in \mathbf{R}^n$, $y_i \in \{-1, +1\}$, an L1-regularized classifier solves the following unconstrained optimization problem:

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}), \tag{1}$$

where

$$f(\boldsymbol{w}) \equiv \|\boldsymbol{w}\|_1 + L(\boldsymbol{w}),$$

$\|\cdot\|_1$ denotes the 1-norm, and $L(\boldsymbol{w})$ indicates training losses

$$L(\boldsymbol{w}) \equiv C \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \tag{2}$$

For logistic regression and L2-loss SVM, we have the following loss functions.

$$\begin{aligned} \xi_{\log}(\boldsymbol{w}; \boldsymbol{x}, y) &= \log(1 + e^{-y\boldsymbol{w}^T\boldsymbol{x}}) \quad \text{and} \\ \xi_{\text{svm}}(\boldsymbol{w}; \boldsymbol{x}, y) &= \max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x})^2. \end{aligned} \tag{3}$$

The regularization term $\|\boldsymbol{w}\|_1$ is used to avoid overfitting the training data. The user-defined parameter $C > 0$ is used to balance regularization and loss terms. Different from the 2-norm regularization, the 1-norm regularization gives a sparse solution of (1).

It is difficult to solve (1) because $\|\boldsymbol{w}\|_1$ is not differentiable. Many optimization approaches have been proposed and an earlier comparison is by Schmidt et al. (2009). Recently, Yuan et al. (2010) made a comprehensive comparison among state-of-the-art algorithms and software for L1-regularized logistic regression and SVM. For L1-regularized logistic regression, they compared CDN (Yuan et al., 2010), BBR (Genkin et al., 2007), SCD (Shalev-Shwartz and Tewari, 2009), CGD (Tseng and Yun, 2009), IPM (Koh et al., 2007), BMRM (Teo et al., 2010), OWL-QN (Andrew and Gao, 2007). Lassplore (Liu et al., 2009), TRON (Lin and Moré, 1999), and GLMNET (Friedman et al., 2010). Other existing approaches include, for example, Shevade and Keerthi (2003), Lee et al. (2006) and Shi et al. (2010). Yuan et al. (2010) conclude that carefully designed coordinate descent (CD) methods perform better than others for large sparse data (e.g., document data). As a result, their CD method (called CDN) was included in a popular package LIBLINEAR as the solver of L1-regularized logistic regression.

However, we point out in Section 2 that CDN becomes inefficient if the loss function is expensive to compute. An example is L1-regularized logistic regression, where exp/log operations are more expensive than other basic operations. We investigate this problem in detail to show that CDN suffers from frequent loss-function computation.

In Section 3, we show that for expensive loss functions, Newton-type methods are more suitable. A Newton method needs not compute the loss function when finding the Newton direction, which is the most time consuming part. Based on this point, we attempt to obtain an appropriate Newton-type method for L1-regularized logistic regression. We introduce an existing Newton-type algorithm GLMNET (Friedman et al., 2010). In Yuan et al.'s comparison, GLMNET, although inferior to CDN, performs reasonably well. However, GLMNET failed to train some large-scale data used in their experiments. In Sections 4 and 5, we improve GLMNET in theoretical and practical aspects, respectively. We call the improved method newGLMNET. Based on the modification in Section 4, we establish the asymptotic convergence of newGLMNET. By a careful design in Section 5 to adjust the effort for each iteration, newGLMNET is efficient for both loosely and strictly solving the optimization problem. Note that our discussion in Sections 2–4 is generic to all differentiable loss functions, although the focus is on logistic regression.

Experiments in Section 6 show that newGLMNET is more efficient than CDN, which was considered the state of the art for L1-regularized logistic regression. In particular, newGLMNET is much faster for dense problems. While logistic regression is an example of problems with expensive loss functions, to check the situation of cheap loss functions, in Section 7, we extend newGLMNET to solve L2-loss SVM. Experiments show that, contrary to logistic regression, CDN is slightly better. Therefore, our investigation in this work fully demonstrate that expensive loss functions need a different design of training algorithms from that of cheap loss functions. Section 8 concludes this work. A supplementary file including additional analysis and experiments is available at `http://www.csie.ntu.edu.tw/~cjlin/papers/l1_glmnet/supplement.pdf`.

Because the proposed newGLMNET is faster for logistic regression, we replace the CDN solver in the package LIBLINEAR with newGLMNET after version 1.8. This paper is an extension of an earlier conference paper (Yuan et al., 2011). In addition to a thorough reorganization of the main results, more analysis and theoretical results are included.

## 2. Coordinate Descent (CD) Method and Its Weakness

CD is a commonly-used optimization approach by iteratively solving one-variable sub-problems. For L1-regularized classification, past works (e.g., Genkin et al., 2007; Yuan et al., 2010) have shown that CD methods can quickly obtain a useful model. In this section, we first discuss a specific CD method called CDN (Yuan et al., 2010) and follow by showing its weakness.

### 2.1 CDN

At the $k$th iteration, a CD method cyclically selects a dimension $j \in \{1, 2, \ldots, n\}$ and solves the following one-variable sub-problem.

$$\min_d \quad f(\boldsymbol{w}^{k,j} + d\boldsymbol{e}_j) - f(\boldsymbol{w}^{k,j}), \tag{4}$$

展开

where

$$f(\boldsymbol{w}^{k,j} + d\boldsymbol{e}_j) - f(\boldsymbol{w}^{k,j}) = \|\boldsymbol{w}^{k,j} + d\boldsymbol{e}_j\|_1 - \|\boldsymbol{w}^{k,j}\|_1 + L(\boldsymbol{w}^{k,j} + d\boldsymbol{e}_j) - L(\boldsymbol{w}^{k,j}).$$

In (4), we define

$$\boldsymbol{w}^{k,j} \equiv [w_1^{k+1}, \ldots, w_{j-1}^{k+1}, w_j^k, \ldots, w_n^k]^T \tag{5}$$

and the indicator vector
指标向量

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

Let $\boldsymbol{w}^k = \boldsymbol{w}^{k,1} = \boldsymbol{w}^{k-1,n+1}$ at the beginning of each iteration. If $d$ is an optimal solution of (4), then $\boldsymbol{w}^{k,j}$ is updated to $\boldsymbol{w}^{k,j+1}$ by

$$w_t^{k,j+1} = \begin{cases} w_t^{k,j} + d & \text{if } t = j, \\ w_t^{k,j} & \text{otherwise.} \end{cases}$$

解析解

For logistic regression, the one-variable sub-problem (4) does not have a closed-form solution, so Yuan et al. (2010) approximately solve it using the second-order approximation of $L(\boldsymbol{w}^{k,j} - d\boldsymbol{e}_j) - L(\boldsymbol{w}^{k,j})$.

$$\min_d \quad \nabla_j L(\boldsymbol{w}^{k,j})d + \frac{1}{2}\nabla^2_{jj} L(\boldsymbol{w}^{k,j})d^2 + |w^k_j + d| - |w^k_j|. \tag{6}$$

For logistic regression,

$$\nabla L(\boldsymbol{w}) = C\sum_{i=1}^{l}(\tau(y_i\boldsymbol{w}^T\boldsymbol{x}_i)-1)y_i\boldsymbol{x}_i \quad \text{and} \quad \nabla^2 L(\boldsymbol{w}) = CX^TDX, \tag{7}$$

where $\tau(s)$ is the derivative of the logistic loss function $\log(1+e^s)$:

$$\tau(s) = \frac{1}{1+e^{-s}},$$

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

$$D_{ii} = \tau(y_i\boldsymbol{w}^T\boldsymbol{x}_i)\left(1-\tau(y_i\boldsymbol{w}^T\boldsymbol{x}_i)\right), \tag{8}$$

and

$$X \equiv \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_l^T \end{bmatrix} \in \mathbf{R}^{l\times n}.$$

It is well known that (6) has a simple closed-form solution

$$d = \begin{cases} -\frac{\nabla_j L(\boldsymbol{w}^{k,j})+1}{\nabla^2_{jj} L(\boldsymbol{w}^{k,j})} & \text{if } \nabla_j L(\boldsymbol{w}^{k,j}) + 1 \le \nabla^2_{jj} L(\boldsymbol{w}^{k,j}) w^{k,j}_j, \\ -\frac{\nabla_j L(\boldsymbol{w}^{k,j})-1}{\nabla^2_{jj} L(\boldsymbol{w}^{k,j})} & \text{if } \nabla_j L(\boldsymbol{w}^{k,j}) - 1 \ge \nabla^2_{jj} L(\boldsymbol{w}^{k,j}) w^{k,j}_j, \\ -w^{k,j}_j & \text{otherwise.} \end{cases} \tag{9}$$

坐标下降+牛顿法

Because (9) considers a Newton direction, Yuan et al. (2010) refer to this setting as CDN (CD method using one-dimensional Newton directions). For convergence, Yuan et al. follow Tseng and Yun (2009) to apply a line search procedure. The largest step size $\lambda \in \{\beta^i \mid i = 0, 1, \dots\}$ is found such that $\lambda d$ satisfies the following sufficient decrease condition.

$$f(\boldsymbol{w}^{k,j} + \lambda d\boldsymbol{e}_j) - f(\boldsymbol{w}^{k,j}) \le \sigma\lambda\left(\nabla_j L(\boldsymbol{w}^{k,j})d + |w^k_j + d| - |w^k_j|\right), \tag{10}$$

where $0 < \beta < 1$ and $0 < \sigma < 1$ are pre-specified parameters.

The basic structure of CDN is in Algorithm 1. To make CDN more efficient, Yuan et al. have considered some implementation tricks, but details are omitted here.

We discuss the computational complexity of CDN. While solving (6) by (9) takes a constant number of operations, calculating $\nabla_j L(\boldsymbol{w}^{k,j})$ and $\nabla^2_{jj} L(\boldsymbol{w}^{k,j})$ for constructing the sub-problem (6) is expensive. From (7), we need $O(nl)$ operations for obtaining $\boldsymbol{w}^T\boldsymbol{x}_i, \forall i$. A common trick to make CD methods viable for classification problems is to store and maintain $\boldsymbol{w}^T\boldsymbol{x}_i, \forall i$. Yuan et al. (2010) store $e^{\boldsymbol{w}^T\boldsymbol{x}_i}$ instead and update the values by

$$e^{\boldsymbol{w}^T\boldsymbol{x}_i} \leftarrow e^{\boldsymbol{w}^T\boldsymbol{x}_i} \cdot e^{\bar{\lambda}dx_{ij}}, \forall i, \tag{11}$$

**Algorithm 1** CDN framework in Yuan et al. (2010). Some implementation details are omitted.

1. Given $\boldsymbol{w}^1$.
2. For $k = 1, 2, 3, \ldots$ // iterations
   - For $j = 1, 2, 3, \ldots, n$ // $n$ CD steps
     - Compute the optimum $d$ of sub-problem (6) by (9).
     - Find the step size $\bar{\lambda}$ by (10).
     - $\boldsymbol{w}^{k,j+1} \leftarrow \boldsymbol{w}^{k,j} + \bar{\lambda} d \boldsymbol{e}_j$.

| Data set | exp/log | Total |
|---|---|---|
| epsilon | 64.25 (73.0%) | 88.18 |
| webspam | 72.89 (66.6%) | 109.39 |

Table 1: Timing analysis of the first CD cycle of CDN. Time is in seconds.

where $\bar{\lambda}$ is the step size decided by the line search procedure and $d$ is the optimal solution of (6). If $e^{\boldsymbol{w}^T \boldsymbol{x}_i}, \forall i$ are available, the evaluation of $\nabla_j L(\boldsymbol{w}^{k,j})$ and $\nabla^2_{jj} L(\boldsymbol{w}^{k,j})$ in (6) and $f(\boldsymbol{w}^{k,j} + \lambda d \boldsymbol{e}_j)$ in the sufficient decrease condition (10) takes $O(l)$ operations. Therefore, with $n$ CD steps in one iteration, the complexity of each iteration is:

$$n \cdot (1 + \#\text{ steps of line search}) \cdot O(l). \tag{12}$$

For sparse data, in (11), only $e^{\boldsymbol{w}^T \boldsymbol{x}_i}$ with $x_{ij} \neq 0$ needs to be updated. Then, $n \cdot O(l)$ in Equation (12) can be reduced to $O(\text{nnz})$, where nnz is the total number of non-zero elements in $X$ (i.e., training data). In Algorithm 1, one CD iteration contains $n$ CD steps to update $w_1, \ldots, w_n$ as a cycle. This concept of CD cycles will be frequently used in our subsequent analysis and experiments.

### 2.2 Weakness of CDN

Although CDN is reported as the best method in the comparison by Yuan et al. (2010), for the same data set, CDN's training time for logistic regression is more than L2-loss SVM. Motivated from this observation, we point out that CDN suffers from expensive exp/log operations of logistic regression.

In Table 1, we conduct an experiment on two data sets, epsilon and webspam.[1] We check the proportion of time for exp/log operations in the first CD cycle of CDN. The results clearly show that exp/log operations dominate the training time. We present results of more data sets in Section 6 and have similar observations.

Exp/log operations occur in two places (11) and (10), each of which costs $O(l)$. From (12), we can see that the complexity of exp/log operations is the same as that of all operations.[2] Because each exp/log is much more expensive than a basic operation like multiplication, a significant portion of running time is spent on exp/log operations.

1. Details of the data sets are in Section 6.1.
2. For binary-valued data, only $e^{\bar{\lambda} d x_{ij}}$ becomes $e^{\bar{\lambda} d}$ in (11), so $O(nl)$ exp/log operations can be reduced to $O(n)$. This has been pointed out in Huang et al. (2010).

## 3. GLMNET: A Method that Less Frequently Computes the Loss Function

Based on the observation in Section 2, to reduce the number of exp/log operations, we should consider methods which less frequently compute the loss function. In this section, we identify such methods and present one of them named GLMNET.

### 3.1 Algorithms that may Have Less Loss Computation

For logistic regression, exp/log operations occur in computing the function, gradient, and Hessian of the loss. To avoid frequent exp/log operations, we hope an optimization method could conduct enough basic (e.g., multiplication or division) operations between two function, gradient, or Hessian evaluations. However, these basic operations should also be useful for minimizing the optimization problem. We find that methods involving second-order approximation of $f(\boldsymbol{w})$ may fulfill the requirements. At the $k$th iteration, consider the following sub-problem to find a direction $\boldsymbol{d}$:

$$\min_{\boldsymbol{d}} \quad q_k(\boldsymbol{d}), \tag{13}$$

where

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

and $H^k$ is either $\nabla^2 L(\boldsymbol{w}^k)$ or its approximation. Then, $\boldsymbol{w}$ is updated by

$$\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \boldsymbol{d}. \tag{14}$$

Between two iterations, $H^k$ and $\nabla f(\boldsymbol{w}^k)$ are constants, so (13) is a quadratic program without involving exp/log operations. Further, (13) is not a trivial sub-problem to solve.

If $H^k = \nabla^2 L(\boldsymbol{w}^k)$, we have a Newton-type method that usually enjoys a small number of iterations. At each iteration, obtaining $\nabla f(\boldsymbol{w}^k)$ and $\nabla^2 f(\boldsymbol{w}^k)$ via (8) requires at least $O(nl)$ operations because of calculating $\sum_{i=1}^{l}(\tau(y_i \boldsymbol{w}^T \boldsymbol{x}) - 1) y_i \boldsymbol{x}_i$. However, the number of exp/log operations is only $O(l)$. With the cost of solving the sub-problem (13), the total cost of one iteration is at least $O(nl)$, but only a small portion, $O(l)$, is for exp/log computation. This situation is much better than CDN, which requires $O(nl)$ exp/log operations in $O(nl)$ overall operations. An existing Newton-type method for L1-regularized classification is GLMNET by Friedman et al. (2010). We will discuss its details in Section 3.2.[3]

Note that CDN also applies second-order approximation for solving the one-variable sub-problem (4). However, once a variable is changed in CDN, the gradient and Hessian become different. In contrast, for GLMNET, gradient and Hessian remain the same while (13) is being solved. The reason is that GLMNET applies second-order approximation on the whole objective function $f(\boldsymbol{w})$. Such differences explain why GLMNET needs less exp/log computation than CDN.

If we use an approximate Hessian as $H^k$, the analysis of $O(l)$ exp/log operations versus at least $O(nl)$ total operations per iteration still holds.[4] However, because minimizing

3. The GLMNET code by Friedman et al. (2010) supports using an approximate Hessian matrix, but here we consider only the case of using the exact Hessian matrix.
4. We assume that obtaining an approximation of $\nabla^2 f(\boldsymbol{w}^k)$ requires no more than $O(l)$ exp/log operations.

$q_k(\boldsymbol{d})$ in (13) becomes easier, exp/log operations may play a more important role in the whole procedure. In the extreme situation, $H^k$ is a constant diagonal matrix, so we have a gradient descent method. Existing approaches of using such $H^k$ include ISTA (Daubechies et al., 2004), FISTA (Beck and Teboulle, 2009), and others.

For L2-regularized logistic regression, Chang et al. (2008) have pointed out that Newton methods less frequently conduct exp/log operations than CD methods, but they did not conduct detailed analysis and comparisons.

### 3.2 GLMNET

We pointed out in Section 3.1 that GLMNET is an existing Newton-type method for L1-regularized classification. At each iteration, it solves the sub-problem (13) with $H^k = \nabla^2 L(\boldsymbol{w}^k)$ and updates $\boldsymbol{w}$ by (14). Although many optimization methods can be applied to solve the sub-problem (13), Friedman et al. (2010) consider a cyclic coordinate descent method similar to CDN in Section 2.1. Indeed, it is simpler than CDN because (13) is only a quadratic problem. We use $\boldsymbol{d}^p$ to denote the CD iterates (cycles) for solving (13). Each CD cycle is now considered as an inner iteration of GLMNET. Sequentially, $\boldsymbol{d}^p$'s values are updated by minimizing the following one-variable function.

$$\begin{aligned} & q_k(\boldsymbol{d}^{p,j} + z\boldsymbol{e}_j) - q_k(\boldsymbol{d}^{p,j}) \\ &= |w_j^k + d_j^p + z| - |w_j^k + d_j^p| + \nabla_j \bar{q}_k(\boldsymbol{d}^{p,j})z + \frac{1}{2}\nabla^2_{jj}\bar{q}_k(\boldsymbol{d}^{p,j})z^2, \end{aligned} \tag{15}$$

where the definition of $\boldsymbol{d}^{p,j}$ is similar to $\boldsymbol{w}^{k,j}$ of CDN in (5)

$$\boldsymbol{d}^{p,j} \equiv [d_1^{p-1}, d_2^{p-1}, \ldots, d_{j-1}^{p-1}, d_j^p, \ldots, d_n^p]^T,$$

and $\boldsymbol{d}^p = \boldsymbol{d}^{p,1} = \boldsymbol{d}^{p-1,n+1}$. Further,

$$\bar{q}_k(\boldsymbol{d}) \equiv \nabla L(\boldsymbol{w}^k)^T\boldsymbol{d} + \frac{1}{2}(\boldsymbol{d})^T\nabla^2 L(\boldsymbol{w}^k)\boldsymbol{d}$$

represents the smooth terms of $q_k(\boldsymbol{d})$ and plays a similar role to $L(\boldsymbol{w})$ for (1). We have

$$\begin{aligned} \nabla_j \bar{q}_k(\boldsymbol{d}^{p,j}) &= \nabla_j L(\boldsymbol{w}^k) + (\nabla^2 L(\boldsymbol{w}^k)\boldsymbol{d}^{p,j})_j \quad \text{and} \\ \nabla^2_{jj}\bar{q}_k(\boldsymbol{d}^{p,j}) &= \nabla^2_{jj} L(\boldsymbol{w}^k). \end{aligned} \tag{16}$$

Equation (15) is in the same form as (6), so it can be easily solved by (9). Because the one-variable function is exactly minimized, line search is not required in the CD procedure. The basic structure of GLMNET is in Algorithm 2.

Because an iterative procedure (CD method) is used to solve the sub-problem (13), GLMNET contains two levels of iterations. A suitable stopping condition for the inner level is very important. In Section 5.2, we will discuss GLMNET's stopping conditions and make some improvements.

We analyze GLMNET's complexity to confirm that it less frequently conducts exp/log operations. At each CD step, most operations are spent on calculating $\nabla_j \bar{q}_k(\boldsymbol{d}^{p,j})$ and $\nabla^2_{jj}\bar{q}_k(\boldsymbol{d}^{p,j})$ in (16). Note that $\nabla^2_{jj}\bar{q}_k(\boldsymbol{d}^{p,j}) = \nabla^2_{jj} L(\boldsymbol{w}^k), \forall j$ can be pre-calculated before the

**Algorithm 2** Basic framework of GLMNET (Friedman et al., 2010) for L1-regularized logistic regression.

1. Given $\boldsymbol{w}^1$.
2. For $k = 1, 2, 3, \ldots$
   - Compute $\nabla L(\boldsymbol{w}^k)$ and $\nabla^2 L(\boldsymbol{w}^k)$ by (7).
   - Obtain $\boldsymbol{d}^k$ by solving sub-problem (13) by CD with certain stopping condition.
   - $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \boldsymbol{d}^k$.

CD procedure. For $\nabla_j \bar{q}_k(\boldsymbol{d}^{p,j})$, the first term $\nabla_j L(\boldsymbol{w}^k)$ can also be pre-calculated. With (7), the second term is

$$(\nabla^2 L(\boldsymbol{w}^k)\boldsymbol{d}^{p,j})_j = C \sum_{t=1}^{n} \sum_{i=1}^{l} X_{ji}^T D_{ii} X_{it} d_t^{p,j} = C \sum_{i=1}^{l} X_{ji}^T D_{ii} (X\boldsymbol{d}^{p,j})_i.$$

If $X\boldsymbol{d}^{p,j}$ (i.e., $\boldsymbol{x}_i^T \boldsymbol{d}^{p,j}, \forall i$) is maintained and updated by

$$(X\boldsymbol{d}^{p,j+1})_i \leftarrow (X\boldsymbol{d}^{p,j})_i + X_{ij} z, \ \forall i, \tag{17}$$

then calculating $\nabla_j \bar{q}_k(\boldsymbol{d})$ costs $O(l)$ operations.[5] Therefore, the CD method for (13) requires

$$O(nl) \text{ operations for one inner iteration (cycle) of } n \text{ CD steps.} \tag{18}$$

The complexity of GLMNET is thus

$$\#\text{outer\_iters} \times (O(nl) + \#\text{inner\_iters} \times O(nl)),$$

where the first $O(nl)$ is for obtaining items such as $\nabla f(\boldsymbol{w}^k)$ before solving (13). We then compare the number of exp/log operations in CDN and GLMNET. Because they both use CD, we check in Table 2 that relative to the total number of operations of one CD cycle, how many exp/log operations are needed. Clearly, CDN's $O(nl)$ exp/log operations are much more than GLMNET's $O(l)$. The difference becomes smaller for sparse data because CDN's $O(nl)$ is reduced by $O(\text{nnz})$. From this analysis, we expect that CDN suffers from many slow exp/log operations if data are dense and $n$ is not small. This result will be clearly observed in Section 6.

Although our analysis indicates that GLMNET is superior to CDN in terms of the number of exp/log operations, experiments in Yuan et al. (2010) show that overall GLMNET is slower. The final local convergence of GLMNET is fast, but it often spends too much time in early iterations. Therefore, contrary to CDN, GLMNET does not quickly obtain a useful model. Further, GLMNET failed to solve two large problems in the experiment of Yuan et al. (2010) and its theoretical convergence is not guaranteed. In the next two sections, we will propose an improved GLMNET to perform faster than CDN for logistic regression.

5. This is like how $e^{\boldsymbol{w}^T \boldsymbol{x}_i}$, $\forall i$ are handled in Section 2.1.

| One cycle of $n$ CD steps | | CDN | GLMNET |
|---|---|---|---|
| Total # of operations | dense data | $O(nl)$ | $O(nl)$ |
| | sparse data | $O(\text{nnz})$ | $O(\text{nnz})$ |
| # exp/log operations | dense data | $O(nl)$ | $\leq O(l)$ |
| | sparse data | $O(\text{nnz})$ | $\leq O(l)$ |

Table 2: A comparison between CDN and GLMNET on the number of exp/log operations in one CD cycle. We assume that in (12), the number of line search steps of CDN is small (e.g., one or two). Note that in GLMNET, exp/log operations are needed in the beginning/end of an outer iteration. That is, they are conducted once every several CD cycles. We make each CD cycle to share the cost in this table even though a CD cycle in GLMNET involves no exp/log operations.

## 4. newGLMNET: An Improved GLMNET

As mentioned in Section 3.2, GLMNET lacks theoretical convergence properties. In this section, we modify GLMNET to be a special case of a class of methods by Tseng and Yun (2009), so the asymptotic convergence immediately follows. We refer to the improved algorithm as newGLMNET.

The framework by Tseng and Yun (2009) for L1-regularized problems is a generalized coordinate descent method. At each iteration, it selects some variables for update based on certain criteria. An extreme situation is to update one variable at a time, so CDN in Section 2 is a special case. The other extreme is to update all variables at an iteration. In this situation, Tseng and Yun's method considers a quadratic approximation the same as (13). However, for the convergence, they required $H^k$ in (13) to be positive definite. From (7), if $X$'s columns are independent, then $\nabla^2 L(\boldsymbol{w}^k)$ is positive definite. To handle the situation that $\nabla^2 L(\boldsymbol{w}^k)$ is only positive semi-definite, we slightly enlarge the diagonal elements by defining

$$H^k \equiv \nabla^2 L(\boldsymbol{w}^k) + \nu \mathcal{I}, \tag{19}$$

where $\nu > 0$ is a small value and $\mathcal{I} \in \mathbf{R}^{n \times n}$ is an identity matrix.

In addition, for convergence Tseng and Yun (2009) require that line search is conducted. After obtaining an optimal solution $\boldsymbol{d}$ of (13), the largest step size $\lambda \in \{\beta^i \mid i = 0, 1, \dots\}$ is found such that $\lambda \boldsymbol{d}$ satisfies the following sufficient decrease condition.

$$\begin{aligned} & f(\boldsymbol{w}^k + \lambda \boldsymbol{d}) - f(\boldsymbol{w}^k) \\ & \leq \sigma \lambda \big( \nabla L(\boldsymbol{w}^k)^T \boldsymbol{d} + \gamma \boldsymbol{d}^T H^k \boldsymbol{d} + \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1 \big), \end{aligned} \tag{20}$$

where $0 < \beta < 1$, $0 < \sigma < 1$, and $0 \leq \gamma < 1$. GLMNET does not conduct line search, so function values of its iterations may not be decreasing. In newGLMNET we use (20) with $\gamma = 0$.

If the sub-problem (13) is exactly solved, we prove that because all conditions needed in Tseng and Yun (2009, Theorems 1(e) and 3) are satisfied, line search is guaranteed to stop after a finite number of step sizes. Further, for asymptotic convergence, we have that any limit point of $\{\boldsymbol{w}^k\}$ generated by newGLMNET is an optimal solution of (1); see the proof

in Appendix A. For local convergence rate, we prove in Appendix B that if the loss function $L(\boldsymbol{w})$ is strictly convex,[6] then the objective function value converges at least linearly.

If we know that $L(\boldsymbol{w})$ is strictly convex beforehand, we can directly use $H^k = \nabla^2 L(\boldsymbol{w}^k)$ without adding $\nu\mathcal{I}$. The same explanation in Appendix A implies both the finite termination of line search and the asymptotic convergence. In this situation, we can obtain a better local quadratic convergence. See details in the supplementary document, in which we modify the proof in Hsieh et al. (2011) for L1-regularized Gaussian Markov random field.

### 4.1 Cost of Line Search

GLMNET does not conduct line search because of the concern on its cost. Interestingly, by the following analysis, we show that line search in newGLMNET introduces very little extra cost. For each step size $\lambda$ tried in line search, we must calculate $f(\boldsymbol{w}^k + \lambda\boldsymbol{d})$. A direct implementation requires $O(nl)$ operations to obtain $(\boldsymbol{w}^k + \lambda\boldsymbol{d})^T\boldsymbol{x}_i$, $\forall i = 1, \ldots, l$. If $e^{(\boldsymbol{w}^k)^T\boldsymbol{x}_i}, \forall i$ are available, we need only $O(l)$ by using $X\boldsymbol{d}$ maintained by (17).

$$e^{(\boldsymbol{w}^k+\lambda\boldsymbol{d})^T\boldsymbol{x}_i} = e^{(\boldsymbol{w}^k)^T\boldsymbol{x}_i} \cdot e^{\lambda(X\boldsymbol{d})_i}. \tag{21}$$

Thus, the cost for finding $f(\boldsymbol{w}^k + \lambda\boldsymbol{d})$ is reduced to $O(n + l)$, where $O(n)$ comes from calculating $\|\boldsymbol{w}^k + \lambda\boldsymbol{d}\|_1$. After the last $\lambda$ is obtained in line search, we have $e^{(\boldsymbol{w}^{k+1})^T\boldsymbol{x}_i} = e^{(\boldsymbol{w}^k+\lambda\boldsymbol{d})^T\boldsymbol{x}_i}, \forall i$ for the next iteration. If only a small number of $\lambda$'s are tried, then the $O(n + l)$ cost is negligible because the whole iteration costs at least $O(nl)$ from earlier discussion.

Discussion in Section 3 indicated that in every $O(nl)$ operations, GLMNET needs a much smaller amount of exp/log operations than CDN. We will show that a similar situation occurs for the line-search operations of newGLMNET and CDN. Note that line search is used in different places of the two methods. For CDN, at each CD step for updating one variable, line search is needed. In contrast, in newGLMNET, we conduct line search only in the end of an outer iteration. Following the discussion in Section 2.1 and this section, for each step size $\lambda$ tried in line search of CDN and newGLMNET, the cost is $O(l)$ and $O(n + l)$, respectively. If we distribute the line search cost of newGLMNET to its inner CD cycles, we have that, for the same $\lambda$ in one CD cycle,[7] CDN costs $O(nl)$ and newGLMNET costs no more than $O(n + l)$. This difference is similar to that of exp/log operations discussed in Section 3.

Because of CDN's high cost on line search, Yuan et al. (2010) develop the following trick. By deriving an upper-bound function $\delta(\lambda)$ such that

$$f(\boldsymbol{w}^{k,j} + \lambda d\boldsymbol{e}_j) - f(\boldsymbol{w}^{k,j}) \le \delta(\lambda), \quad \forall\lambda \ge 0, \tag{22}$$

they check first if

$$\delta(\lambda) \le \sigma\lambda(\nabla_j L(\boldsymbol{w}^{k,j})d + |w_j^k + d| - |w_j^k|). \tag{23}$$

This trick, used for each step size $\lambda$, is particularly useful if the above inequality holds at $\lambda = 1$. If $\delta(\lambda)$ can be calculated in $O(1)$, the $O(l)$ cost for line search at a CD step is significantly reduced to $O(1)$. More details about this upper-bound function can be found in Fan et al. (2008, Appendix G).

6. For situations such as $n > l$, $L(\boldsymbol{w}^k)$ is not strictly convex.
7. For simplicity, we assume that this $\lambda$ is tried in all $n$ CD steps of the cycle.

In Section 6.2, we will conduct experiments to investigate the line search cost of CDN and newGLMNET.

## 5. Implementation Issues of newGLMNET

Besides theoretical issues discussed in Section 4, in this section, we discuss some implementation techniques to make newGLMNET an efficient method in practice.

### 5.1 Random Permutation of One-variable Sub-problems

To solve the sub-problem (13), a conventional CD method sequentially updates variables $d_1, d_2, \ldots, d_n$. Many past works (e.g., Chang et al., 2008; Hsieh et al., 2008; Yuan et al., 2010) have experimentally indicated that using a random order leads to faster convergence. We adapt this strategy in the CD procedure of newGLMNET to solve sub-problem (13).

### 5.2 An Adaptive Inner Stopping Condition

GLMNET contains two levels of iterations. An "outer iteration" corresponds to the process from $\boldsymbol{w}^k$ to $\boldsymbol{w}^{k+1}$, while the "inner" level consists of CD iterations for solving (13). For an algorithm involving two levels of iterations, the stopping condition of the inner iterations must be carefully designed. A strict inner stopping condition may cause the algorithm to take a prohibitive amount of time at the first outer iteration. Alternatively, a loose inner condition leads to an inaccurate solution of (13) and possibly lengthy outer iterations. GLMNET terminates the CD procedure by checking if $\boldsymbol{d}$ is still significantly changed. That is, in the $p$th CD cycle to update $d_1^p, \ldots, d_n^p$, the corresponding changes $z_1, \ldots, z_n$ satisfy

$$\max_j (\nabla^2_{jj} L(\boldsymbol{w}^k) \cdot z_j^2) \le \epsilon_{\text{in}}, \tag{24}$$

where $\epsilon_{\text{in}}$ is the inner stopping tolerance. For the outer stopping condition, similarly, GLMNET checks if $\boldsymbol{w}$ is still significantly changed. Let $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \boldsymbol{d}^k$. GLMNET stops if the following condition holds:

$$\max_j (\nabla^2_{jj} L(\boldsymbol{w}^{k+1}) \cdot (d_j^k)^2) \le \epsilon_{\text{out}}, \tag{25}$$

where $\epsilon_{\text{out}}$ is the outer stopping tolerance. GLMNET uses the same value for inner and outer tolerances; that is, $\epsilon_{\text{in}} = \epsilon_{\text{out}}$. We find that if users specify a small $\epsilon_{\text{out}}$, a huge amount of time may be needed for the first outer iteration. This observation indicates that the inner tolerance must be carefully decided.

For newGLMNET, we propose an adaptive inner stopping condition. The design principle is that in the early stage, newGLMNET should behave like CDN to quickly obtain a reasonable model, while in the final stage, newGLMNET should achieve fast local convergence by using Newton-like directions. In the $p$th inner iteration $p$, we assume that $\boldsymbol{d}^{p,1}, \ldots, \boldsymbol{d}^{p,n}$ are sequentially generated and from $\boldsymbol{d}^{p,j}$ to $\boldsymbol{d}^{p,j+1}$, the $j$th element is updated. We propose the following inner stopping condition.

$$\sum_{j=1}^{n} |\nabla_j^S q_k(\boldsymbol{d}^{p,j})| \le \epsilon_{\text{in}}, \tag{26}$$

where $\nabla^S q(\boldsymbol{d})$ is the minimum-norm subgradient at $\boldsymbol{d}$.

$$\nabla_j^S q(\boldsymbol{d}) \equiv \begin{cases} \nabla_j \bar{q}(\boldsymbol{d}) + 1 & \text{if } w_j + d_j > 0, \\ \nabla_j \bar{q}(\boldsymbol{d}) - 1 & \text{if } w_j + d_j < 0, \\ \text{sgn}(\nabla_j \bar{q}(\boldsymbol{d})) \max(|\nabla_j \bar{q}(\boldsymbol{d})| - 1, 0) & \text{if } w_j + d_j = 0. \end{cases}$$

From standard convex analysis,

$$\nabla q^S(\boldsymbol{d}) = \mathbf{0} \quad \text{if and only if } \boldsymbol{d} \text{ is optimal for (13).} \tag{27}$$

Note that we do not need to calculate the whole $\nabla^S q_k(\boldsymbol{d}^{p,j})$. Instead, $\nabla_j^S q_k(\boldsymbol{d}^{p,j})$ is easily available via $\nabla_j \bar{q}_k(\boldsymbol{d}^{p,j})$ in (16).

If at one outer iteration, the condition (26) holds after only one cycle of $n$ CD steps, then we reduce $\epsilon_{\text{in}}$ by

$$\epsilon_{\text{in}} \leftarrow \epsilon_{\text{in}}/4. \tag{28}$$

That is, the program automatically adjusts $\epsilon_{\text{in}}$ if it finds that too few CD steps are conducted for minimizing $q_k(\boldsymbol{d})$. Therefore, we can choose a large $\epsilon_{\text{in}}$ in the beginning.

We use an outer stopping condition similar to (26).

$$\sum_{j=1}^{n} |\nabla_j^S f(\boldsymbol{w}^k)| \le \epsilon_{\text{out}}. \tag{29}$$

Like (27), $\nabla^S f(\boldsymbol{w}) = \mathbf{0}$ is an optimality condition for (1). In (29), we choose 1-norm instead of $\infty$-norm because 1-norm is not determined by extreme values in $\nabla_j^S f(\boldsymbol{w}^k), j = 1, \ldots, n$. In (7), $\nabla_j L(\boldsymbol{w})$ can be seen as a function of $x_{ij}, \forall i$. It is expected that $|\nabla_j^S f(\boldsymbol{w})|$ is relatively large if the $j$th column of $X$ has a larger norm than other columns. If using $\infty$-norm, the stopping condition may be dominated by a large $|\nabla_j^S f(\boldsymbol{w})|$; therefore, under a given $\epsilon_{\text{out}}$, the total number of iterations may be quite different for two feature-wisely norm-varying $X$'s. In contrast, the sum of violations should be less sensitive to the different numeric ranges of $X$'s columns.

### 5.3 A Two-level Shrinking Scheme

Shrinking is a common technique to heuristically remove some variables during the optimization procedure.[8] That is, some $\boldsymbol{w}$'s elements are conjectured to be already optimal, so a smaller optimization problem is solved. GLMNET applies this technique on the sub-problem (13) by selecting a working set $J \subset \{1, \ldots, n\}$. Sub-problem (13) becomes

$$\min_{\boldsymbol{d}} \quad q_k(\boldsymbol{d}) \quad \text{subject to} \quad d_j = 0, \quad \forall j \notin J. \tag{30}$$

More precisely, at the $k$th iteration, GLMNET conducts the following loop to sequentially solve some smaller sub-problems.

While (`TRUE`)

8. Shrinking is widely used in solving SVM optimization problems; see, for example, Joachims (1998) and Chang and Lin (2011).

- Conduct one cycle of $n$ CD steps. Let $J$ include indices of $\boldsymbol{d}$'s elements that still need to be changed.
- If (24) holds, then `break`.
- Use CD to solve a sub-problem with the working set $J$ until (24) holds.

The way to choose $J$ in the above procedure is by checking if $z = 0$ is optimal for $\min_z q_k(\boldsymbol{d} + z\boldsymbol{e}_j) - q_k(\boldsymbol{d})$.

For newGLMNET, we propose a heuristic shrinking scheme following its two-level structure: the outer level removes some $\boldsymbol{w}$'s elements so that a smaller sub-problem (13) similar to (30) is solved; the inner level is applied to remove elements in $\boldsymbol{d}$ so that (13) becomes an even smaller sub-problem. For each level, our setting is related to the shrinking implementation of CDN; see Yuan et al. (2010, Section 4.1.2).

In the beginning of each outer iteration, we remove $w_j$ if

$$w_j^k = 0 \quad \text{and} \quad -1 + \frac{M^{\text{out}}}{l} < \nabla_j L(\boldsymbol{w}^k) < 1 - \frac{M^{\text{out}}}{l}, \tag{31}$$

where

$$M^{\text{out}} \equiv \max\left(\left|\nabla_1^S f(\boldsymbol{w}^{k-1})\right|, \ldots, \left|\nabla_n^S f(\boldsymbol{w}^{k-1})\right|\right).$$

The conditions in (31) come from the optimality condition that an optimal solution $\boldsymbol{w}^*$ of (1) satisfies

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

Therefore, we conjecture that variables satisfying (31) are already optimal. $M^{\text{out}}$ in (31) is used to adjust the shrinking scheme from a conservative setting in the beginning to an aggressive setting in the end. Our shrinking implementation differs from GLMNET's in several aspects. First, by using $\nabla f(\boldsymbol{w}^k)$ that is available in the beginning of the $k$th iteration, we do not conduct a special cycle of $n$ CD steps in GLMNET for selecting variables. Note that $\nabla^S f(\boldsymbol{w}^k)$ can be easily obtained via $\nabla L(\boldsymbol{w}^k)$ and is used for obtaining $M^{\text{out}}$ of the next iteration.[9] Second, (31) shrinks only zero elements and uses an interval slightly smaller than $(-1, 1)$. Thus, newGLMNET is less aggressive than GLMNET in removing variables.

For the inner shrinking scheme of newGLMNET, assume the previous CD cycle contains points $\boldsymbol{d}^{p-1,1}, \ldots, \boldsymbol{d}^{p-1,|J^p|}$, where elements in the set $J^p = \{j_1, \ldots, j_{|J^p|}\}$ were updated. Because $J^p$ corresponds to the remained variables, at the current cycle, sequentially $j \in J^p$ is checked. An element $j$ is removed if

$$w_j^k + d_j^{p,t} = 0 \quad \text{and} \quad -1 + \frac{M^{\text{in}}}{l} < \nabla_j \bar{q}_k(\boldsymbol{d}^{p,t}) < 1 - \frac{M^{\text{in}}}{l}, \tag{32}$$

where $t$ is the iteration index of the current cycle and

$$M^{\text{in}} \equiv \max\left(\left|\nabla_{j_1}^S q_k(\boldsymbol{d}^{p-1,1})\right|, \ldots, \left|\nabla_{j_{|J^p|}}^S q_k(\boldsymbol{d}^{p-1,|J^p|})\right|\right).$$

If (32) does not hold, element $j$ remains.[10] After the set $J^p$ has been processed, a smaller subset $J^{p+1}$ is obtained and we move to the next CD cycle.[11]

9. If $k = 1$, $\nabla^S f(\boldsymbol{w}^{k-1})$ is not available. We set $M^{\text{out}} = \infty$, so no variables are shrunk at the first outer iteration.
10. Note that in (32), $t = 1, \ldots, |J^{p+1}|$ instead of $1, \ldots, |J^p|$.
11. Similar to the way to initialize $M^{\text{out}}$, for the first CD cycle, we set $M^{\text{in}} = \infty$.

**Algorithm 3** Overall procedure of newGLMNET

- Given $\boldsymbol{w}^1$, $\epsilon_{\text{in}}$, and $\epsilon_{\text{out}}$. Choose a small positive number $\nu$. Choose $\beta \in (0,1)$, $\gamma \in [0,1)$, and $\sigma \in (0,1)$.
- Let $M^{\text{out}} \leftarrow \infty$.
- For $k = 1, 2, 3, \ldots$ // outer iterations
  1. Let $J \leftarrow \{1, \ldots, n\}$, $M \leftarrow 0$, and $\bar{M} \leftarrow 0$.
  2. For $j = 1, \ldots, n$
     2.1. Calculate $H^k_{jj}$, $\nabla_j L(\boldsymbol{w}^k)$ and $\nabla^S_j f(\boldsymbol{w}^k)$.
     2.2. If $w^k_j = 0$ and $|\nabla_j L(\boldsymbol{w}^k)| < 1 - M^{\text{out}}/l$ // outer-level shrinking
          $J \leftarrow J \backslash \{j\}$.
          Else
          $M \leftarrow \max(M, |\nabla^S_j f(\boldsymbol{w}^k)|)$ and $\bar{M} \leftarrow \bar{M} + |\nabla^S_j f(\boldsymbol{w}^k)|$.
  3. If $\bar{M} \le \epsilon_{\text{out}}$
     `return` $\boldsymbol{w}^k$.
  4. Let $M^{\text{out}} \leftarrow M$.
  5. Get $\boldsymbol{d}$ and update $\epsilon_{\text{in}}$ by solving sub-problem (13) by Algorithm 4.
  6. Compute $\lambda = \max\{1, \beta, \beta^2, \ldots\}$ such that $\lambda \boldsymbol{d}$ satisfies (20).
  7. $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \lambda \boldsymbol{d}$.

The overall procedure of newGLMNET with two-level shrinking is shown in Algorithms 3 and 4. For theoretical properties, if the subproblem (13) is exactly solved, for any given outer stopping tolerance, newGLMNET terminates in finite iterations. Further, any limit point of $\{\boldsymbol{w}^k\}$ is an optimal solution. More details are in the supplementary document.

### 5.4 The Overall Procedure of newGLMNET

We use Algorithms 3 to illustrate the overall procedure of newGLMNET. Steps 1–4 are for outer-level shrinking and the stopping condition. In particular, in Step 2.2, $M$ is used to calculate $M^{\text{out}}$ in (31) for the outer-level shrinking, while $\bar{M}$ is for calculating $\sum_{j=1}^n |\nabla^S_j f(\boldsymbol{w})|$ in the outer stopping condition (29). Step 5 obtains the Newton direction by a CD method, where details are shown in Algorithm 4. Step 6 then conducts line search.

In Algorithm 4, besides the stopping condition (26), we set 1,000 as the maximal number of CD cycles. Some ill-conditioned sub-problem (13) may take lengthy inner CD iterations to satisfy (26), so a maximal number must be set. In the beginning of Algorithm 4, the working set $J$ is obtained from outer-level shrinking. Subsequently, in the inner CD iterations, $J$ is further shrunk; see the set $T$ in Algorithm 4. Because each CD cycle goes through only elements in $T$, the inner stopping condition (26) is also calculated using $T$. To ensure that sub-problem (13) with the working set $J$ has been accurately solved, if (26) holds on $T$, we reset $T$ to $J$; see Step 3 of Algorithm 4. That is, the inner iterations terminate only if the condition (26) holds on $J$ or the maximal number of iterations is reached. This way of resetting $T$ to $J$ has been used in LIBSVM (Chang and Lin, 2011, Section 5.1).

In (28), we reduce the inner stopping tolerance $\epsilon_{\text{in}}$ if the stopping condition (26) holds after one CD cycle. This is implemented in the last step of Algorithm 4.

**Algorithm 4** Inner iterations of newGLMNET with shrinking

- Given working set $J$, initial solution $\boldsymbol{d}$, inner stopping condition $\epsilon_{\text{in}}$, and a small positive number $\nu$ from the outer problem.
- Let $M^{\text{in}} \leftarrow \infty$, $T \leftarrow J$, and $\boldsymbol{d} \leftarrow \mathbf{0}$.
- For $p = 1, 2, 3, \ldots, 1000$ // inner iterations
  1. Let $m \leftarrow 0$ and $\bar{m} \leftarrow 0$.
  2. For $j \in T$
     - Let $\nabla^2_{jj}\bar{q}_k(\boldsymbol{d}) = H^k_{jj}$. Calculate $\nabla_j\bar{q}_k(\boldsymbol{d})$ and $\nabla^S_j q_k(\boldsymbol{d})$.
     - If $w^k_j + d_j = 0$ and $|\nabla_j\bar{q}_k(\boldsymbol{d})| < 1 - M^{\text{in}}/l$ // inner-level shrinking
       $T \leftarrow T\backslash\{j\}$.

       Else
       $m \leftarrow \max(m, |\nabla^S_j q_k(\boldsymbol{d})|)$ and $\bar{m} \leftarrow \bar{m} + |\nabla^S_j q_k(\boldsymbol{d})|$.
       $d_j \leftarrow d_j + \arg\min_z q_k(\boldsymbol{d}+z\boldsymbol{e}_j) - q_k(\boldsymbol{d})$.
  3. If $\bar{m} \le \epsilon_{\text{in}}$
     - If $T = J$ // inner stopping
       `break`.

       Else // active set reactivation
       $T \leftarrow J$ and $M^{\text{in}} \leftarrow \infty$.

     Else
     - $M^{\text{in}} \leftarrow m$.
- If $p = 1$, then $\epsilon_{\text{in}} \leftarrow \epsilon_{\text{in}}/4$.

## 6. Experiments on L1-regularized Logistic Regression

We investigate the performance of CDN, GLMNET, and newGLMNET on L1-regularized logistic regression. All these methods can be easily extended to solve logistic regression with a bias term $b$:

$$\min_{\boldsymbol{w},b} \quad \|\boldsymbol{w}\|_1 + C\sum_{i=1}^{l}\log(1 + e^{-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}). \tag{33}$$

Because the GLMNET implementation solves (33) instead of (1), in our comparison, (33) is used. We do not consider other methods because in Yuan et al. (2010), CDN is shown to be the best for sparse data.

Programs used in this paper are available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/exp.html`.

### 6.1 Data Sets and Experimental Settings

We use eight data sets in our experiments. Five of them (news20, rcv1, yahoo-japan, yahoo-korea, and webspam) are document data sets, where news20 is a collection of news documents, rcv1 is an archive of manually categorized news stories from Reuters, yahoo-japan and yahoo-korea are document data from Yahoo!, and webspam includes web pages represented in trigram. The other three data sets come from different learning problems: gisette is a handwriting digit recognition problem from NIPS 2003 feature selection challenge; epsilon is an artificial data set for Pascal large scale learning challenge in 2008; KDD2010-b includes student performance prediction data for a tutoring system and is used for the data

| Data set | #data train | #data test | #features | #nnz in training | #nnz per instance | Sparsity $C$ | Sparsity (%) |
|---|---|---|---|---|---|---|---|
| KDD2010-b | 19,264,097 | 748,401 | 29,890,095 | 566,345,888 | 29 | 0.5 | 97.4 |
| rcv1 | 541,920 | 135,479 | 47,236 | 39,625,144 | 73 | 4 | 76.9 |
| yahoo-japan | 140,963 | 35,240 | 832,026 | 18,738,315 | 133 | 4 | 99.0 |
| yahoo-korea | 368,444 | 92,110 | 3,052,939 | 125,190,807 | 340 | 4 | 99.1 |
| news20 | 15,997 | 3,999 | 1,355,191 | 7,281,110 | 455 | 64 | 99.1 |
| epsilon | 400,000 | 100,000 | 2,000 | 800,000,000 | 2,000 | 0.5 | 44.9 |
| webspam | 280,000 | 70,000 | 16,609,143 | 1,043,724,776 | 3,727 | 64 | 99.9 |
| gisette | 6,000 | 1,000 | 5,000 | 29,729,997 | 4,955 | 0.25 | 72.9 |

Table 3: Data statistics, the parameter $C$ selected after cross validation, and the model sparsity (%). Data sets are sorted by the number of nonzero elements per instance in the training data. We conduct five-fold cross validation on the training set to select $C$ in $\{2^k \mid k = -4, -3, \ldots, 6\}$. The model sparsity is the percentage of the number of zeros in the final model $\boldsymbol{w}$. #nnz denotes the number of nonzero elements.

mining competition KDD Cup 2010. Each instance in document data sets is normalized to a unit vector. For non-document data, features of gisette are linearly scaled to the $[-1, 1]$ interval. Features of epsilon are scaled to $N(0, 1)$ and each instance is normalized to a unit vector. Except yahoo-japan and yahoo-korea, all data sets and their detailed information are publicly available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

We prepare training and testing sets for each problem. For gisette and KDD2010-b, we use their original training and test sets. For others, we randomly split data to one fifth for testing and the remaining for training.

We choose the parameter $C$ in (33) by five-fold cross validation on the training set. All methods then solve (33) with the best $C$ to obtain the model for prediction. Table 3 shows the statistics and the best $C$ of all data sets. We can clearly see that two data sets (epsilon and gisette) are very dense, while others are sparse.

Next, we describe software information and parameter settings in our experiments.

- CDN: this coordinate descent method is described in Section 2.1. In the line search procedure, we use $\sigma = 0.01$ and $\beta = 0.5$. The C/C++ implementation is included in LIBLINEAR (version 1.7), which is available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/oldfiles`; see the implementation document (Fan et al., 2008) for more details.

- GLMNET: this method is described in Section 3.2. GLMNET is implemented in Fortran with an R interface. The source code (version 1.5.3) is available at `http://cran.r-project.org/web/packages/glmnet/`. GLMNET uses the regularization parameter $\lambda = 1/(Cl)$ instead of $C$. We ensure that the equivalent settings have been made in our experiments.

- newGLMNET: this improved GLMNET is described in Sections 4 and 5. For the positive definiteness of $H^k$, we set $\nu = 10^{-12}$ in (19). To check the sufficient decrease condition

(20), we use $\beta = 0.5$, $\gamma = 0$, and $\sigma = 0.01$. We choose the initial $\epsilon_{\text{in}} = \|\nabla^S f(\boldsymbol{w}^1)\|_1$. The C/C++ implementation is included in LIBLINEAR (version 1.8).

GLMNET offers an option to find a solution path $\{\boldsymbol{w}^{C_1}, \ldots, \boldsymbol{w}^{C_*}\}$ of an increasing parameter sequence $\{C_1, \ldots, C_*\}$. It applies a warm start technique so that the optimal solution of the previous $C_{i-1}$ is used as the initial point for the current $C_i$. The number of outer iterations should be small because of using a more accurate initial point. GLMNET authors suggest that finding a solution path may be faster than solving a single optimization problem under a fixed $C$ (Friedman et al., 2010, Section 2.5). We refer to this approach as GLMNETpath and include it for comparison. By their default setting, we consider a parameter sequence of length 100 starting from the smallest $C_1$ such that $\boldsymbol{w}^{C_1} = \mathbf{0}$. Given our desired parameter $C_*$, a geometric sequence is generated by a fixed ratio between successive $C$ values.

We set the initial $\boldsymbol{w}^1 = \mathbf{0}$ for all methods. All experiments are conducted 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 and the optimization flag is properly set.

### 6.2 Running Time Comparison

We begin with checking the change of function values along the running time in Figure 1. Given a stopping tolerance for running a solver, we can obtain a pair of (training time, function value). Using a decreasing sequence of the stopping tolerances, we obtain several pairs and then draw a curve.[12] The $x$-axis in Figure 1 is the log-scaled training time and the $y$-axis is the relative difference to the optimal function value:

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

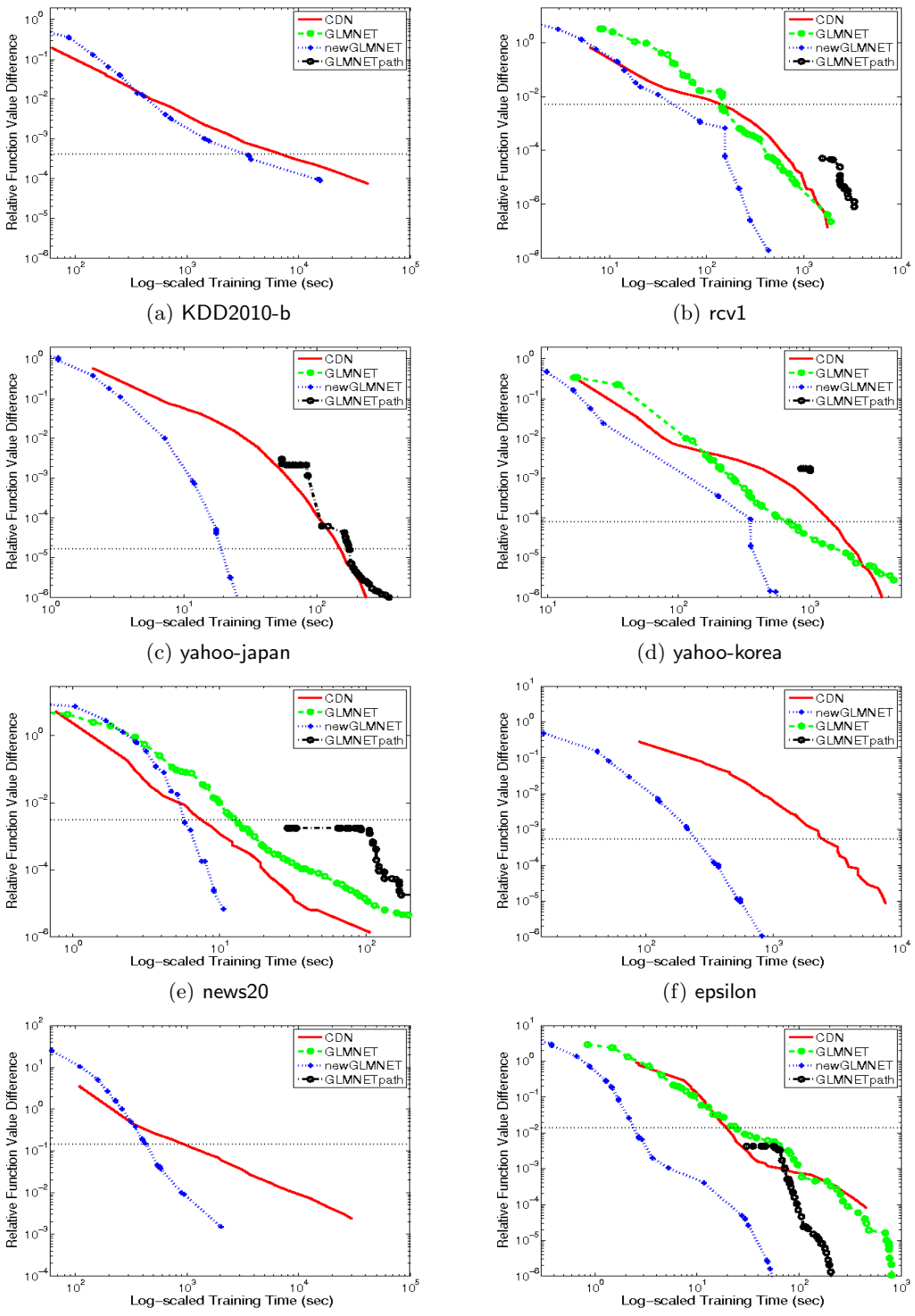
where $\boldsymbol{w}$ is the solution under the specified tolerance and $f^*$ is the optimal function value. Because $f^*$ is not available, we obtain an approximation by running newGLMNET with a small stopping tolerance

$$\epsilon_{\text{out}} = \epsilon \cdot \frac{\min(\#\text{pos}, \#\text{neg})}{l} \cdot \|\nabla^S f(\boldsymbol{w}^1)\|_1, \tag{34}$$

where $\epsilon = 10^{-8}$, and #pos and #neg indicate the numbers of positive and negative labels in the training set, respectively. The horizontal dotted line in Figure 1 indicates the relative function difference by running CDN using LIBLINEAR's default stopping tolerance with $\epsilon = 0.01$ in (34). The point where a method's curve passes this horizontal line roughly indicates the time needed to obtain an accurate enough solution.

From Figure 1, if the optimization problem is loosely solved using a large $\epsilon_{\text{out}}$, CDN is faster than newGLMNET and GLMNET. This result is reasonable because CDN uses a greedy setting to sequentially update variables. In contrast, in each outer iteration, newGLMNET

12. For GLMNET and newGLMNET, the tolerance means the outer tolerance $\epsilon_{\text{out}}$ in (25). Ranges of $\epsilon_{\text{out}}$ values used for GLMNET and newGLMNET differ because their stopping conditions are not the same. Note that for GLMNET, $\epsilon_{\text{out}}$ is not directly specified by users; instead, it is the product between a user-specified value and a constant. For GLMNETpath, under any given $\epsilon_{\text{out}}$, a sequence of problems (1) is solved.



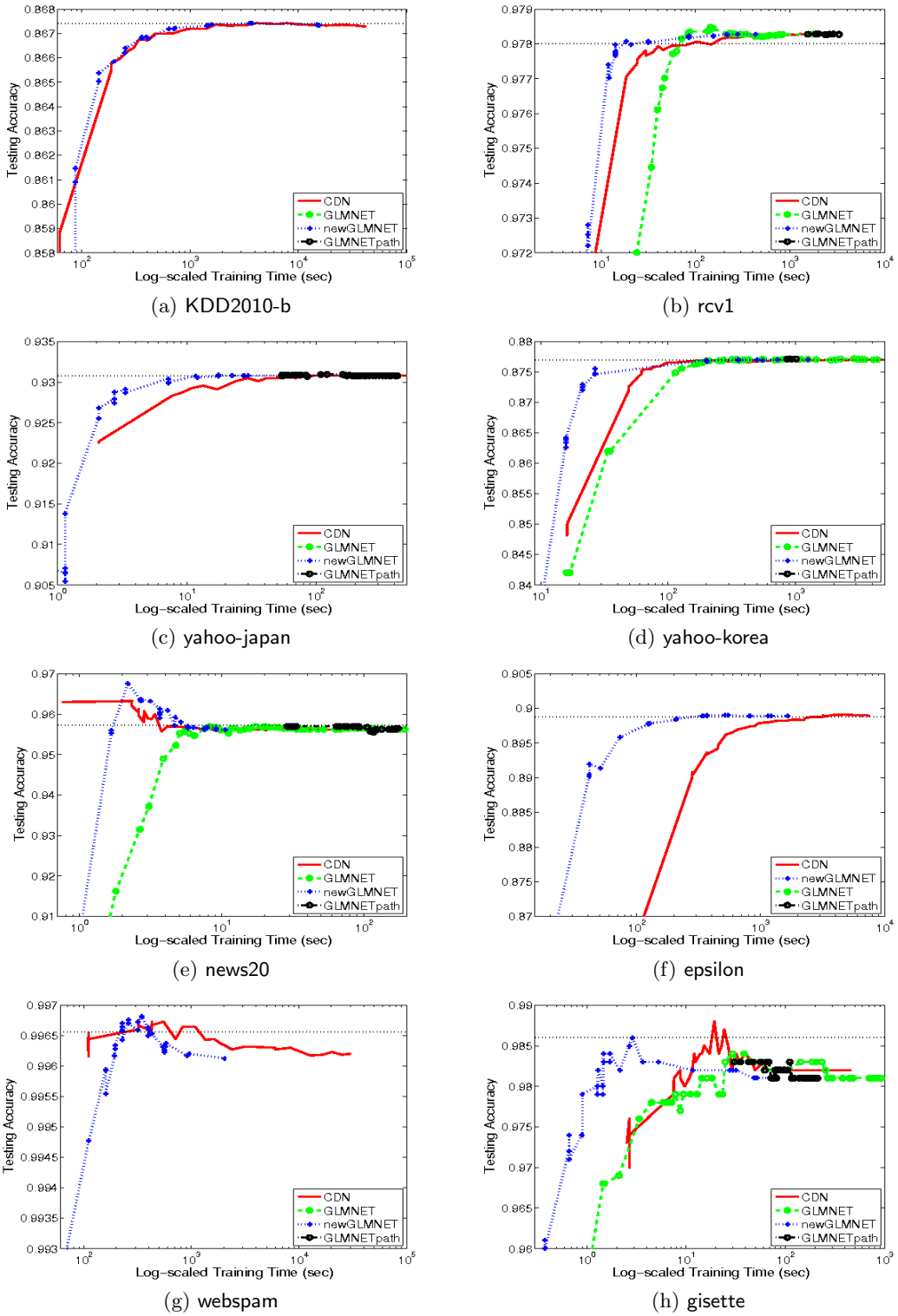

(a) KDD2010-b (b) rcv1

(c) yahoo-japan (d) yahoo-korea

(e) news20 (f) epsilon

(g) webspam (h) gisette

Figure 1: L1-regularized logistic regression: relative difference to the optimal function value versus training time. Both $x$-axis and $y$-axis are log-scaled. GLMNET and GLMNETpath failed to generate some results because of either memory problems or lengthy running time.




(a) KDD2010-b (b) rcv1

(c) yahoo-japan (d) yahoo-korea

(e) news20 (f) epsilon

(g) webspam (h) gisette

Figure 2: L1-regularized logistic regression: testing accuracy versus training time (log-scaled).

uses only a fixed $H^k$. If using a smaller $\epsilon_{\text{out}}$, newGLMNET surpasses CDN and achieves fast local convergence. For dense data (epsilon and gisette), newGLMNET is always better than CDN. Take epsilon as an example. In Figure 1(f), to reach the horizontal dotted line, newGLMNET is ten times faster than CDN. This huge difference is expected following the analysis on the number of exp/log operations in Sections 2 and 3.

From results above the horizontal lines in Figure 1, we see that newGLMNET is faster than GLMNET in the early stage. Recall that GLMNET sets $\epsilon_{\text{in}} = \epsilon_{\text{out}}$, while newGLMNET uses an adaptive setting to adjust $\epsilon_{\text{in}}$. Because a large $\epsilon_{\text{in}}$ is considered in the beginning, newGLMNET can compete with CDN in the early stage by loosely solving (13). We use an example to further illustrate the importance to properly choose $\epsilon_{\text{in}}$. By running GLMNET with the default $\epsilon_{\text{out}} = 10^{-6}$ on news20 and rcv1, the training time is 20.10 and 758.82 seconds, respectively. The first outer iteration already takes 6.99 seconds on news20 and 296.87 on rcv1. A quick fix is to enlarge the initial $\epsilon_{\text{in}}$, but the local convergence in the later stage may be slow. A better inner stopping condition should be adaptive like ours so that the sub-problem (13) can be solved properly at each outer iteration.

In Figure 1, GLMNET and GLMNETpath failed to generate some results because of either memory problems or lengthy running time. This indicates that a careful implementation is very important for large-scale problems. We also observe that GLMNETpath is not faster than GLMNET. Another drawback of GLMNETpath is that it is hard to quickly obtain an approximate solution. That is, regardless of $\epsilon_{\text{out}}$ specified, a sequence of problems (1) is always solved.

We further check the relationship between the testing accuracy and the training time. The comparison result, shown in Figure 2, is similar to that in Figure 1.

In summary, because of the proposed adaptive inner stopping condition, newGLMNET takes both advantages of fast approximation in the early stage like CDN and of fast local convergence in the final stage like GLMNET.

### 6.3 Analysis on Line Search

Recall in Section 4 we added a line search procedure in newGLMNET. To check if line search costs much in newGLMNET, we report the average number of line search steps per outer iteration in Table 4. Clearly, in all cases, $\lambda = 1$ satisfies the sufficient decrease condition (20), so conducting line search to ensure the convergence introduces very little cost. As a comparison, we also, respectively, show the average numbers of updated variables and line search steps per cycle of CDN in the first and second columns of the same table. Note that because a shrinking technique is applied to CDN, the number of updated variables in one cycle is much smaller than the number of features. We see that the number of line search steps is very close to the number of updated variables in a CD cycle. Therefore, in most cases, (10) holds when $\lambda = 1$. Although line search in both CDN and newGLMNET often succeeds at the first step size $\lambda = 1$, from Section 4.1, their numbers of operations are very different. The $O(nl)$ cost of CDN can be significantly reduced due to shrinking, but is still more expensive than $O(n + l)$ of newGLMNET.

In Section 4.1, we mentioned an $O(1)$-cost upper-bound function $\delta(\lambda)$ to efficiently check (10) in line search of CDN. In Table 4, we further report the average number of line search steps in a CD cycle where this check is successfully applied. We see that the

| Data set | CDN #variables in a CD cycle | CDN #$\lambda$ tried in line search | CDN # $\delta(\lambda)$ successfully applied in (23) | newGLMNET #$\lambda$ tried in an outer iteration |
|---|---|---|---|---|
| KDD2010-b | 630,455 | 630,588 | 622,267 | 1 |
| rcv1 | 11,396 | 11,398 | 449 | 1 |
| yahoo-japan | 8,269 | 8,270 | 922 | 1 |
| yahoo-korea | 27,103 | 27,103 | 1,353 | 1 |
| news20 | 6,395 | 6,396 | 2,413 | 1 |
| epsilon | 1,130 | 1,130 | 0 | 1 |
| webspam | 17,443 | 17,444 | 3,389 | 1 |
| gisette | 1,121 | 1,121 | 0 | 1 |

Table 4: Logistic regression: the average number of line search steps per CD cycle of CDN and per outer iteration of newGLMNET. The data are collected by running CDN and newGLMNET using the best $C$ and the default stopping condition of LIBLINEAR.

trick is particularly effective on KDD2010-b; however, it helps in a limited manner on other data sets. For KDD2010-b, in addition to a small nnz/$l$, the faster line search is another possible reason why CDN is comparable to newGLMNET; see Figure 1(a). For gisette and epsilon, the trick is not useful because the assumption $x_{ij} \geq 0,\ \forall i, j$ needed for deriving the upper-bound function does not hold.

### 6.4 Effect of Exp/log Operations

In Sections 2–3, we pointed out the difference between CDN's $O$(nnz) and newGLMNET's $O(l)$ exp/log operations per CD cycle. Figures 1–2 confirm this result because newGLMNET is much faster for the two dense data (epsilon and gisette). We further extend Table 1 to compare the running time of the first CD cycle in Table 5. For easy comparison, we deliberately sort data sets in all tables of this section by nnz/$l$, which is the average number of non-zero values per instance. We expect the ratio of time spent on exp/log operations gradually increases along with nnz/$l$, although this is not very clearly observed in Table 5. The reason might be either that the number of data sets used is small or other data characteristics affect the running time.

### 6.5 Approximate Exponential Operations for CDN

Because CDN suffers from slow exp/log operations, we tried to use the approximate exponentiation proposed by Schraudolph (1999). However, we failed to speed up CDN because of erroneous numerical results. One of the several possible reasons is that when $d$ in (11) is small, the approximate $\exp(dx_{ij})$ is inaccurate. The inaccurate $\exp(dx_{ij})$ makes $|\log(1+e^{-\boldsymbol{w}^T\boldsymbol{x}}\cdot e^{-dx_{ij}})-\log(1+e^{-\boldsymbol{w}^T\boldsymbol{x}})|$ have a large relative error because the correct value is near zero with small $d$. We may encounter this problem when calculating the function value difference needed by the sufficient decrease condition (10). In our experiment, the function-value difference using approximate exp/log operations tend to be larger than the

| Data set | CDN | | newGLMNET | |
|---|---|---|---|---|
| | exp/log | Total | exp/log | Total |
| KDD2010-b | 21.72 (30.7%) | 70.80 | 3.88 (6.9%) | 56.50 |
| rcv1 | 4.61 (73.8%) | 6.25 | 0.12 (5.3%) | 2.20 |
| yahoo-japan | 1.47 (70.9%) | 2.08 | 0.03 (4.2%) | 0.71 |
| yahoo-korea | 10.65 (66.3%) | 16.06 | 0.08 (1.2%) | 6.66 |
| news20 | 0.21 (27.3%) | 0.76 | 0.003 (0.5%) | 0.60 |
| epsilon* | 64.25 (73.0%) | 88.18 | 0.08 (0.7%) | 11.62 |
| webspam | 72.89 (66.6%) | 109.39 | 0.06 (0.1%) | 41.10 |
| gisette* | 1.66 (66.8%) | 2.49 | 0.002 (0.6%) | 0.27 |

Table 5: Timing analysis of the first cycle of $n$ CD steps. Time is in seconds. (*: dense data)

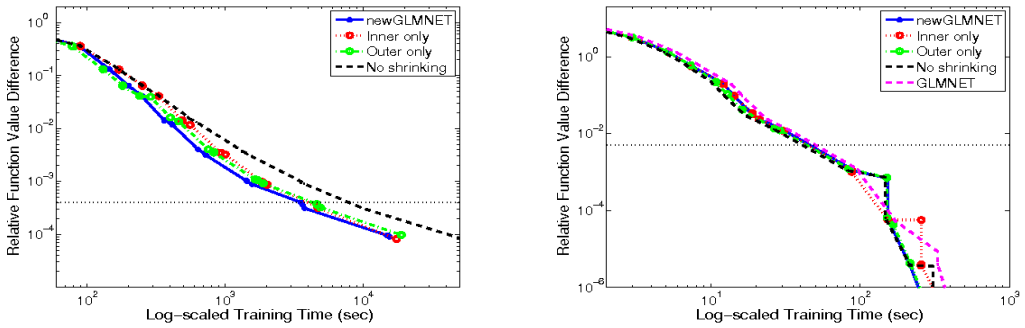
correct value; therefore, it is hard to find a step size $\bar{\lambda}$ satisfying (10). Consequently, if $d$ is small when the current $\boldsymbol{w}^k$ is near the optimal solution, line search terminates with a very small step size $\bar{\lambda}$ and results in bad convergence. Furthermore, because we update $e^{\boldsymbol{w}^T\boldsymbol{x}}$ by (11), the error is accumulated. Schraudolph (1999, Section 5) has mentioned that the approximation may not be suitable for some numerical methods due to error amplification. We also tried different reformulation of $\log(1 + e^{-\boldsymbol{w}^T\boldsymbol{x}} \cdot e^{-dx_{ij}}) - \log(1 + e^{-\boldsymbol{w}^T\boldsymbol{x}})$, but still failed to use an approximate exponential function.

### 6.6 Effect of Shrinking

Our investigation contains two parts. First, we investigate the effect of newGLMNET's two levels of shrinking by presenting results of only inner or outer level. Secondly, we compare the shrinking strategies of GLMNET and newGLMNET. Because these two implementations differ in many places, for a fair comparison, we modify newGLMNET to apply GLMNET's shrinking strategy. The comparison results are presented in Figure 3. We can clearly see that all shrinking implementations are better than the one without shrinking.

Results in Figure 3 show that the outer-level shrinking is more useful than the inner-level shrinking. We suspect that the difference is due to that in the CD procedure for sub-problem (13), the (inner-level) shrinking is done in a sequential manner. Thus, not only is $M^{\text{in}}$ not calculated based on the gradient at the same point, but also variables are not removed together. In contrast, for the outer-level shrinking, $M^{\text{out}}$ is calculated by the gradient at $\boldsymbol{w}^{k-1}$ and all variables are checked together. Therefore, the outer-level shrinking is a more integrated setting for checking and removing variables. The same explanation may also apply to the result that shrinking is slightly more effective for newGLMNET than CDN; see Yuan et al. (2010, Figure 9) and Figure 3 here.

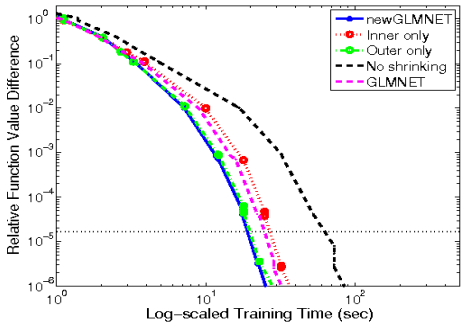
Regarding GLMNET's shrinking strategy, it performs slightly better than the inner-level shrinking of newGLMNET, but is worse than both the outer-level and the two-level settings.



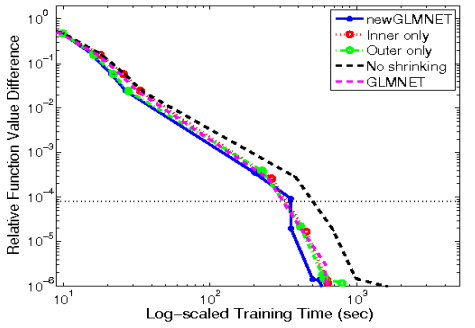

(a) KDD2010-b



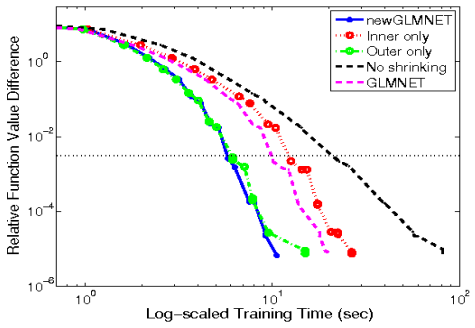

(b) rcv1



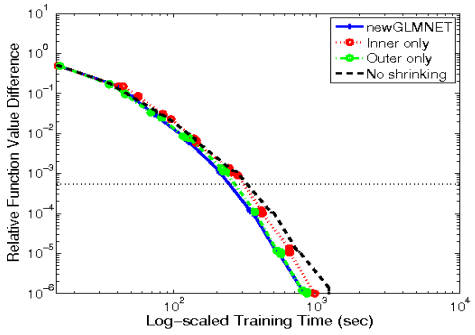

(c) yahoo-japan



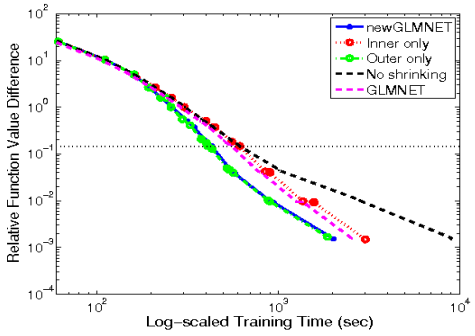

(d) yahoo-korea



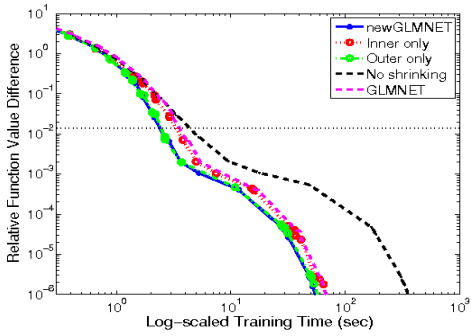

(e) news20




(f) epsilon




(g) webspam




(h) gisette

Figure 3: Effect of two-level shrinking. "Inner only" ("Outer only") indicates that only inner-level (outer-level) shrinking is conducted.

## 7. Using newGLMNET to Solve Problems with Cheap Loss Functions

The analysis and experiments in previous sections have shown that newGLMNET is more efficient than CDN for logistic regression. However, it is not clear if newGLMNET is superior when a loss function can be calculated cheaply. In this section, we consider the L2-loss function in Equation (3) and investigate the performance of newGLMNET in comparison with CDN. The CDN algorithm for L2-loss SVM has been developed in Fan et al. (2008, Appendix F) and Yuan et al. (2010, Section 7).

We briefly describe how to apply newGLMNET to solve L2-loss SVM. The objective function can be written as

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

where

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

Similar to (2), we define

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

The gradient of $L(\boldsymbol{w})$ is

$$\nabla L(\boldsymbol{w}) = -2C \sum_{i \in I(\boldsymbol{w})} b_i(\boldsymbol{w}) y_i \boldsymbol{x}_i. \tag{35}$$

Different from logistic loss, $L(\boldsymbol{w})$ is not twice differentiable. Following Mangasarian (2002) and Yuan et al. (2010), we consider the following generalized Hessian:

$$\nabla^2 L(\boldsymbol{w}) = 2CX^T DX, \tag{36}$$

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

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

At the $k$th outer iteration, newGLMNET solves a quadratic sub-problem

$$\min_{\boldsymbol{d}} \quad q_k(\boldsymbol{d}), \tag{37}$$

where

$$\begin{aligned} q_k(\boldsymbol{d}) &\equiv \|\boldsymbol{w}^k + \boldsymbol{d}\|_1 - \|\boldsymbol{w}^k\|_1 + \bar{q}_k(\boldsymbol{d}), \\ \bar{q}_k(\boldsymbol{d}) &\equiv \nabla L(\boldsymbol{w}^k)^T \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}^T H^k \boldsymbol{d} \quad \text{and} \quad H^k \equiv \nabla^2 L(\boldsymbol{w}^k) + \nu \mathcal{I}. \end{aligned}$$

To minimize (37), we also use a CD procedure to sequentially minimize one-variable functions at each inner iteration $p$.

$$\begin{aligned} & q_k(\boldsymbol{d}^{p,j} + z\boldsymbol{e}_j) - q_k(\boldsymbol{d}^{p,j}) \\ &= |w_j^k + d_j^p + z| - |w_j^k + d_j^p| + \nabla_j \bar{q}_k(\boldsymbol{d}^{p,j}) z + \frac{1}{2} \nabla_{jj}^2 \bar{q}_k(\boldsymbol{d}^{p,j}) z^2, \end{aligned} \tag{38}$$

where from (35) and (36),

$$\begin{aligned}\nabla_j \bar{q}_k(\boldsymbol{d}^{p,j}) &= \nabla_j L(\boldsymbol{w}^k) + (H^k \boldsymbol{d}^{p,j})_j \\ &= -2C \sum_{i \in I(\boldsymbol{w}^k)} b_i(\boldsymbol{w}^k) y_i x_{ij} + 2C \sum_{i \in I(\boldsymbol{w}^k)} (X^T)_{ji} D_{ii} (X \boldsymbol{d}^{p,j})_i + \nu d_j^p \quad \text{and} \\ \nabla^2_{jj} \bar{q}_k(\boldsymbol{d}^{p,j}) &= \nabla^2_{jj} L(\boldsymbol{w}^k) + \nu = 2C \sum_{i \in I(\boldsymbol{w}^k)} x_{ij}^2 + \nu.\end{aligned}$$

The CD procedure is almost the same as the one described in Section 3.2 for logistic regression. The function in (38) can be exactly minimized by (9) and line search is not needed. Moreover, $X\boldsymbol{d}^{p,j}$ is maintained by (17), so the cost per CD cycle is the same as that shown in (18).

### 7.1 Line Search and Asymptotic Convergence

At every outer iteration, after $\boldsymbol{d}$ is obtained by solving sub-problem (37), we need a line search procedure to find the maximal $\lambda \in \{\beta^i \mid i = 0, 1, \dots\}$ such that (20) is satisfied. Following the discussion in Section 4.1, the computational bottleneck is on calculating $(\boldsymbol{w}^k + \lambda \boldsymbol{d})^T \boldsymbol{x}_i, \forall i$. Similar to the trick in Equation (21), we maintain $b_i(\boldsymbol{w}^k),\ \forall i$ to save the cost. In line search, we use

$$\begin{aligned} b_i(\boldsymbol{w}^k + \beta^t \boldsymbol{d}) &= 1 - y_i(\boldsymbol{w}^k + \beta^{t-1}\boldsymbol{d})^T \boldsymbol{x}_i + (\beta^{t-1} - \beta^t) y_i (X\boldsymbol{d})_i \\ &= b_i(\boldsymbol{w}^k + \beta^{t-1}\boldsymbol{d}) + (\beta^{t-1} - \beta^t) y_i (X\boldsymbol{d})_i \end{aligned}$$

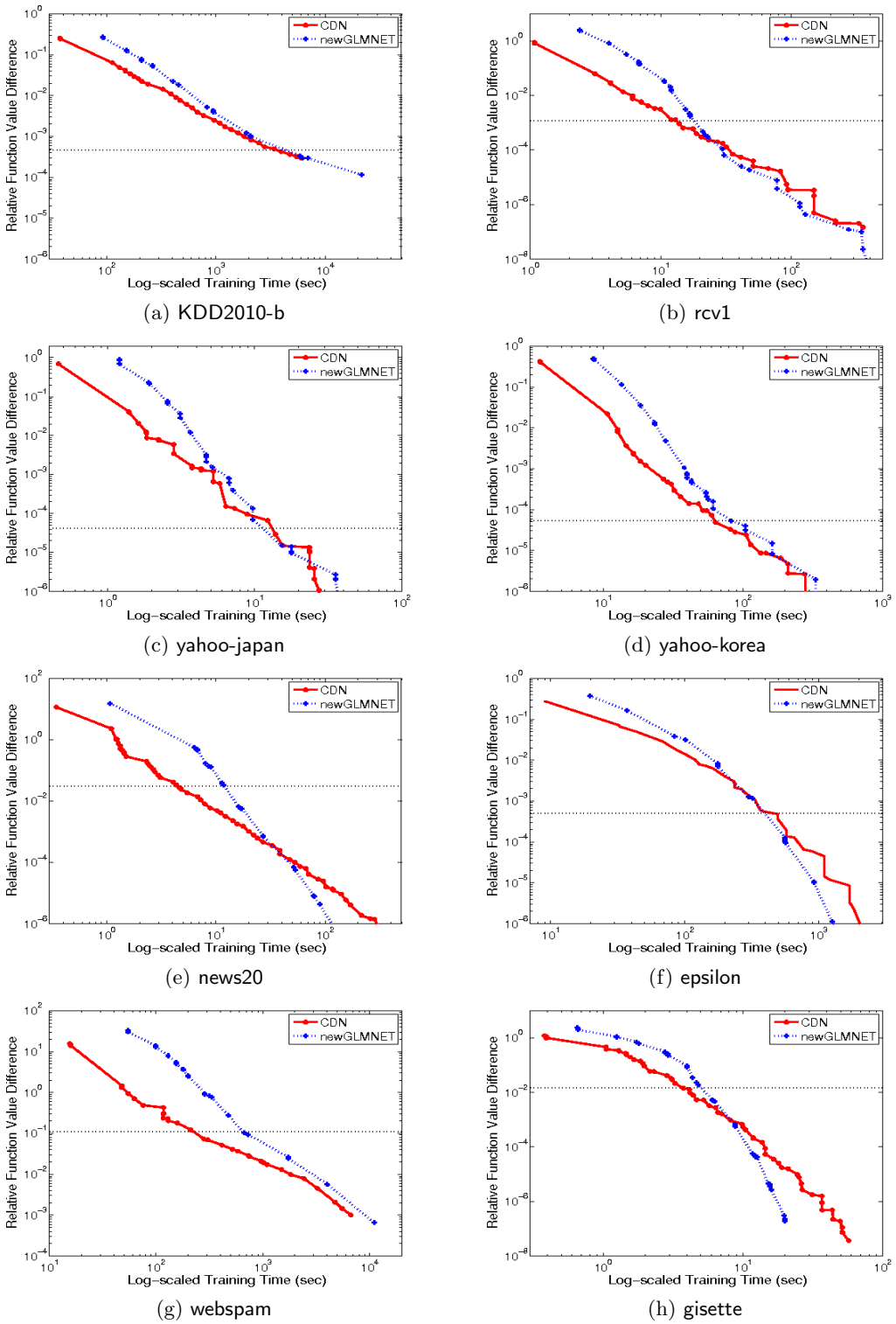
for calculating $f(\boldsymbol{w}^k + \beta^t \boldsymbol{d})$. The last $b_i(\boldsymbol{w}^k + \beta^t \boldsymbol{d})$ is passed to the next outer iteration as $b_i(\boldsymbol{w}^{k+1})$.

In Appendix C, we prove that newGLMNET for L2-loss SVM is an example of Tseng and Yun's framework, so the finite termination of line search holds and any limit point of $\{\boldsymbol{w}^k\}$ is an optimal solution.

### 7.2 Comparison with CDN

The analysis in Section 2 indicates that CDN needs more exp/log operations than newGLMNET. Experiments in Section 6.4 confirm this analysis by showing that CDN is much slower than newGLMNET on dense data. However, the situation for L2-loss SVM may be completely different because exp/log operations are not needed. Without this advantage, whether newGLMNET can still compete with CDN is an interesting question. We will answer this question by experiments in Section 7.3.

Following the analysis in Section 4.1, the cost of line search is still much different between CDN and newGLMNET for L2-loss SVM. For each cycle of $n$ CD steps, the $O(nl)$ cost is required in CDN, while less than $O(n + l)$ is required in newGLMNET. For the high cost of line search in CDN, Yuan et al. (2010) also find out an upper-bound function like (22), which can be obtained in $O(1)$; see Fan et al. (2008, Appendix F) for more details. If this trick succeeds at $\lambda = 1$ in every CD step of a cycle, then the $O(nl)$ cost is reduced to $O(l)$. In Section 7.3, we check if this trick is useful.



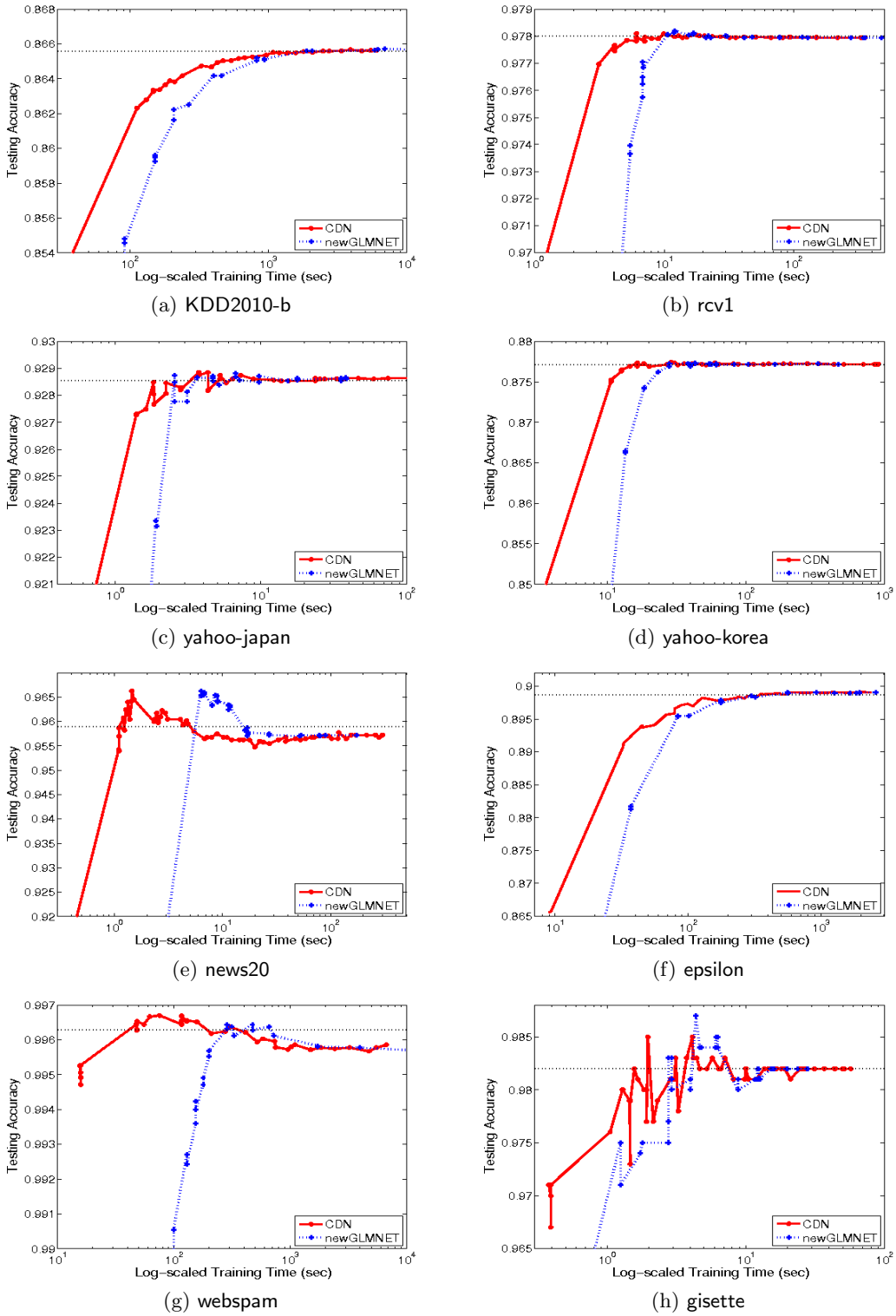

(a) KDD2010-b (b) rcv1

(c) yahoo-japan (d) yahoo-korea

(e) news20 (f) epsilon

(g) webspam (h) gisette

Figure 4: L1-regularized L2-loss SVM: relative difference to the optimal function value versus training time. Both $x$-axis and $y$-axis are log-scaled.




(a) KDD2010-b (b) rcv1

(c) yahoo-japan (d) yahoo-korea

(e) news20 (f) epsilon

(g) webspam (h) gisette

Figure 5: L1-regularized L2-loss SVM: testing accuracy versus training time (log-scaled).

| Data set | CDN #variables in a CD cycle | CDN #$\lambda$ tried in line search | CDN # $\delta(\lambda)$ successfully applied in (23) | newGLMNET #$\lambda$ tried in an outer iteration |
|---|---|---|---|---|
| KDD2010-b | 246,318 | 248,151 | 124,175 | 1 |
| rcv1 | 13,350 | 13,384 | 1,251 | 1 |
| yahoo-japan | 10,286 | 10,289 | 4,931 | 1 |
| yahoo-korea | 31,265 | 31,270 | 25,711 | 1 |
| news20 | 7,688 | 7,838 | 1,461 | 1 |
| epsilon | 1,136 | 1,137 | 501 | 1 |
| webspam | 8,165 | 8,312 | 361 | 1 |
| gisette | 1,145 | 1,145 | 76 | 1 |

Table 6: L2-loss SVM: the average number of line search steps per CD cycle of CDN and newGLMNET. The data are collected by running CDN and newGLMNET using the best $C$ and the default stopping tolerance.

### 7.3 Experiments

We compare CDN and newGLMNET under a similar experimental setting to that for logistic regression. Different from (33), we solve L2-loss SVM without a bias term $b$.[13]

We plot the relative difference to the optimal function value in Figure 4. The reference $f^*$ is obtained by running newGLMNET with a strict stopping tolerance $\epsilon_{\text{out}} = 10^{-8}$. Figure 5 presents the testing accuracy along training time. We can clearly see that CDN is much faster than newGLMNET in the early stage. While newGLMNET still enjoys fast local convergence, it catches up with CDN only in the very end of the training procedure. This result is consistent with our analysis in Section 7.2 showing that newGLMNET loses the advantages of taking fewer exp/log operations.

In Table 6, we analyze the line search procedure by a setting like Table 4. Similar results are observed: the sufficient decrease condition (20) always holds when $\lambda = 1$ for newGLMNET; moreover, for CDN, $\lambda = 1$ is successful almost all the time. One difference is that the trick of using an upper-bound function in CDN is slightly more effective for L2-loss SVM than logistic regression.

## 8. Discussions and Conclusions

In newGLMNET, a CD method is applied to solve the sub-problem (13). Using the property that CD involves simple and cheap updates, we carefully adjust the stopping condition for sub-problems. Then, newGLMNET is competitive with a CD method like CDN in the early stage, but becomes a Newton method in the end. This design is similar to "truncated Newton" methods in optimization. While CD seems to be a very good choice for solving the sub-problem, whether there are better alternatives is an interesting future issue.

13. Earlier we solved problem (33) in order to compare with the GLMNET implementation by Friedman et al..

In Section 5, we proposed several implementation techniques for newGLMNET. For shrinking, we consider thresholds $M^{\text{out}}/l$ and $M^{\text{in}}/l$ in (31) and (32), respectively. These values are heuristically chosen. While it is difficult to find an optimal setting for all data sets, we hope to investigate if the current thresholds are suitable.

Some recent works such as El Ghaoui et al. (2010) and Tibshirani et al. (2011) proposed rules to cheaply eliminate features prior to the L1 training. Preliminary results in the supplementary document show that training is more efficient if we can remove some zero variables beforehand. How to efficiently and correctly identify these variables before training is an interesting future topic.

In our newGLMNET implementation, the sub-problem (13) is approximately solved by CD. However, so far we only establish the convergence results of newGLMNET under the assumption that the sub-problem (13) is exactly solved. In the future, we will strive to address this issue.

In this work, we point out that a state-of-the-art algorithm CDN for L1-regularized logistic regression suffers from frequent exp/log operations. We then demonstrate that Newton-type methods can effectively address this issue. By improving a Newton-type method GLMNET in both theoretical and practical aspects, the proposed newGLMNET is more efficient than CDN for logistic regression. The difference is huge for dense data. However, if a loss function is cheap to compute (e.g., L2 loss), CDN is still competitive. Based on this research work, we have replaced CDN with newGLMNET as the solver of L1-regularized logistic regression in the software LIBLINEAR.

## 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 Mark Schmidt, associate editor, and anonymous reviewers for valuable comments. The authors also thank S. Sathiya Keerthi for pointing out a problem and providing a fix on proving Eq. (50).

## Appendix A. Convergence of newGLMNET for L1-regularized Logistic Regression

We have explained that newGLMNET is in the framework of Tseng and Yun (2009). Thus, it is sufficient to check conditions needed for their convergence result.

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

$$\|\nabla L(\boldsymbol{w}_1) - \nabla L(\boldsymbol{w}_2)\| \le \Lambda \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|, \ \forall \boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbf{R}^n \tag{39}$$

and

$$H^k \succ 0. \tag{40}$$

Note that "$A \succ B$" indicates that $A - B$ is positive definite.

Because $L(\boldsymbol{w})$ is twice differentiable,

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

where $\tilde{\boldsymbol{w}}$ is between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. Furthermore, $\|\nabla^2 L(\tilde{\boldsymbol{w}})\|$ is bounded:

$$\|\nabla^2 L(\tilde{\boldsymbol{w}})\| = C\|X^T D(\tilde{\boldsymbol{w}})X\| \le C\|X^T\|\|X\|. \tag{41}$$

Note that $D(\tilde{\boldsymbol{w}})$ is the diagonal matrix defined in (8) though here we denote it as a function of $\boldsymbol{w}$. The inequality in (41) follows from that all $D(\tilde{\boldsymbol{w}})$'s components are smaller than one. Thus, Equation (39) holds with $\Lambda = C\|X^T\|\|X\|$. For (40), $H^k \succeq \nu\mathcal{I} \succ 0$ because we add $\nu\mathcal{I}$ to $\nabla^2 L(\boldsymbol{w}^k)$ and $\nabla^2 L(\boldsymbol{w}^k)$ is positive semi-definite. With (39) and (40), the line search procedure terminates in finite steps.

For the asymptotic convergence, Tseng and Yun (2009) further assume that there exist positive constants $\lambda_{\min}$ and $\lambda_{\max}$ such that

$$\lambda_{\min}\mathcal{I} \preceq H^k \preceq \lambda_{\max}\mathcal{I}, \quad \forall k. \tag{42}$$

Since $H^k = \nabla^2 L(\boldsymbol{w}^k) + \nu\mathcal{I}$, clearly we can set $\lambda_{\min} = \nu$. For the upper bound, it is sufficient to prove that the level set is bounded. See the proof in, for example, Yuan et al. (2010, Appendix A).

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

## Appendix B. Linear Convergence of newGLMNET for L1-regularized Logistic Regression

To apply the linear convergence result in Tseng and Yun (2009), we show that L1-regularized logistic regression satisfies the conditions in their Theorem 3 if the loss term $L(\boldsymbol{w})$ is strictly convex (and therefore $\boldsymbol{w}^*$ is unique).

From Appendix A, we know L1-regularized logistic regression has the following properties.

1. $\nabla L(\boldsymbol{w})$ is Lipschitz continuous; see (39).
2. The level set is compact, and hence the optimal solution $\boldsymbol{w}^*$ exists.
3. $\lambda_{\min}\mathcal{I} \preceq H^k \preceq \lambda_{\max}\mathcal{I}$, $\forall k$; see (42).

In addition to the above three conditions, Tseng and Yun (2009, Theorem 3) require that for all $\zeta \ge \min_{\boldsymbol{w}} f(\boldsymbol{w})$, there exists $T > 0, \epsilon > 0$, such that

$$T\|\boldsymbol{d}_{\mathcal{I}}(\boldsymbol{w})\| \ge \|\boldsymbol{w} - \boldsymbol{w}^*\|, \quad \forall \boldsymbol{w} \in \{\boldsymbol{w} \mid f(\boldsymbol{w}) \le \zeta \text{ and } \|\boldsymbol{d}_{\mathcal{I}}(\boldsymbol{w})\| \le \epsilon\}, \tag{43}$$

where $\boldsymbol{d}_{\mathcal{I}}(\boldsymbol{w})$ is the solution of (13) at $\boldsymbol{w}$ with $H = \mathcal{I}$ (Tseng and Yun, 2009, Assumption 2). We prove (43) by following the approach in Tseng and Yun (2009, Theorem 4).

To simplify the notation, we denote $\boldsymbol{d}_{\mathcal{I}} \equiv \boldsymbol{d}_{\mathcal{I}}(\boldsymbol{w})$. For all $\zeta > 0$, we show that there exists $T > 0$ so that (43) is satisfied for all $\boldsymbol{w}$ with $f(\boldsymbol{w}) \le \zeta$. That is, a stronger result independent of $\epsilon$ is obtained. We assume $\boldsymbol{w}$ is in the level set $\{\boldsymbol{w} \mid f(\boldsymbol{w}) \le \zeta\}$ in the following proof. Because $\boldsymbol{d}_{\mathcal{I}}$ is the solution of (13) with $H = \mathcal{I}$, by checking the optimality condition,[14] $\boldsymbol{d}_{\mathcal{I}}$ is also an optimal solution of

$$\min_{\boldsymbol{d}} \quad (\nabla L(\boldsymbol{w}) + \boldsymbol{d}_{\mathcal{I}})^T \boldsymbol{d} + \|\boldsymbol{w} + \boldsymbol{d}\|_1.$$

14. For example, the minimal-norm subgradients of the two objective functions are the same at $\boldsymbol{d}_{\mathcal{I}}$.

Therefore,

$$(\nabla L(\boldsymbol{w}) + \boldsymbol{d}_{\mathcal{I}})^T \boldsymbol{d}_{\mathcal{I}} + \|\boldsymbol{w} + \boldsymbol{d}_{\mathcal{I}}\|_1 \leq (\nabla L(\boldsymbol{w}) + \boldsymbol{d}_{\mathcal{I}})^T (\boldsymbol{w}^* - \boldsymbol{w}) + \|\boldsymbol{w}^*\|_1. \tag{44}$$

Besides, because $\boldsymbol{w}^*$ minimizes $f(\boldsymbol{w})$, the following inequality holds for all $\boldsymbol{w}$ and $\delta \in (0, 1)$.

$$\begin{aligned}
& \frac{L(\boldsymbol{w}^* + \delta(\boldsymbol{w} - \boldsymbol{w}^*)) - L(\boldsymbol{w}^*)}{\delta} + \|\boldsymbol{w}\|_1 - \|\boldsymbol{w}^*\|_1 && (45)\\
\geq & \frac{L(\boldsymbol{w}^* + \delta(\boldsymbol{w} - \boldsymbol{w}^*)) - L(\boldsymbol{w}^*) + \|\boldsymbol{w}^* + \delta(\boldsymbol{w} - \boldsymbol{w}^*)\|_1 - \|\boldsymbol{w}^*\|_1}{\delta} && (46)\\
= & \frac{f(\boldsymbol{w}^* + \delta(\boldsymbol{w} - \boldsymbol{w}^*)) - f(\boldsymbol{w}^*)}{\delta} \\
\geq & 0,
\end{aligned}$$

where (46) is from the convexity of $\|\cdot\|_1$. Take $\delta \to 0$ and replace $\boldsymbol{w}$ with $\boldsymbol{w} + \boldsymbol{d}_{\mathcal{I}}$ in (45). We get

$$0 \leq \nabla L(\boldsymbol{w}^*)^T (\boldsymbol{w} + \boldsymbol{d}_{\mathcal{I}} - \boldsymbol{w}^*) + \|\boldsymbol{w} + \boldsymbol{d}_{\mathcal{I}}\|_1 - \|\boldsymbol{w}^*\|_1. \tag{47}$$

Adding (44) to (47) yields

$$(\nabla L(\boldsymbol{w}) - \nabla L(\boldsymbol{w}^*))^T (\boldsymbol{w} - \boldsymbol{w}^*) + \|\boldsymbol{d}_{\mathcal{I}}\|^2 \leq (\nabla L(\boldsymbol{w}^*) - \nabla L(\boldsymbol{w}))^T \boldsymbol{d}_{\mathcal{I}} + \boldsymbol{d}_{\mathcal{I}}^T (\boldsymbol{w}^* - \boldsymbol{w}). \tag{48}$$

Because the level set $\{\boldsymbol{w} \mid f(\boldsymbol{w}) \leq \zeta\}$ is compact, there exists $\bar{m} > 0$ such that

$$\nabla^2 L(\boldsymbol{w}) = X^T D(\boldsymbol{w}) X \succeq \bar{m} X^T X. \tag{49}$$

The reason is that $D_{ii}(\boldsymbol{w})$ is a continuous and positive function over a compact set on $\boldsymbol{w}$. We claim that $X^T X$ is positive definite because $L(\boldsymbol{w})$ is strictly convex. Otherwise, a vector $\boldsymbol{v} \neq \boldsymbol{0}$ satisfies $\boldsymbol{v}^T X^T X \boldsymbol{v} = 0$ and hence $X\boldsymbol{v} = \boldsymbol{0}$. Then from (3), we have

$$L(\boldsymbol{w} + \alpha \boldsymbol{v}) = L(\boldsymbol{w}), \forall \alpha, \boldsymbol{w},$$

a contradiction to the strict convexity of $L(\boldsymbol{w})$. With $\bar{m} X^T X$ being positive definite, (49) implies that there exists $m > 0$ such that

$$(\nabla L(\boldsymbol{w}) - \nabla L(\boldsymbol{w}^*))^T (\boldsymbol{w} - \boldsymbol{w}^*) \geq m \|\boldsymbol{w} - \boldsymbol{w}^*\|^2, \quad \forall \boldsymbol{w} \in \{\boldsymbol{w} \mid f(\boldsymbol{w}) \leq \zeta\}. \tag{50}$$

Then we can relax (48) to

$$m\|\boldsymbol{w} - \boldsymbol{w}^*\|^2 \leq m\|\boldsymbol{w} - \boldsymbol{w}^*\|^2 + \|\boldsymbol{d}_{\mathcal{I}}\|^2 \leq \Lambda \|\boldsymbol{w} - \boldsymbol{w}^*\| \|\boldsymbol{d}_{\mathcal{I}}\| + \|\boldsymbol{d}_{\mathcal{I}}\| \|\boldsymbol{w} - \boldsymbol{w}^*\|,$$

where $\Lambda$ is the Lipschitz constant in (39). Dividing both sides by $m\|\boldsymbol{w} - \boldsymbol{w}^*\|$ generates

$$\|\boldsymbol{w} - \boldsymbol{w}^*\| \leq \frac{\Lambda + 1}{m} \|\boldsymbol{d}_{\mathcal{I}}\|.$$

Then $T = (\Lambda + 1)/m$ satisfies condition (43). Therefore, all conditions in Tseng and Yun (2009, Theorem 3) are satisfied, so linear convergence is guaranteed.

## Appendix C. Convergence of newGLMNET for L1-regularized L2-loss SVM

Similar to Appendix A, we only check the conditions required by Tseng and Yun (2009). To have the finite termination of line search, we need Equations (39) and (40), while for asymptotic convergence, we need Equation (42). Following the same explanation in Appendix A, we easily have (39) and (42). For (40), which means that $\nabla L(\boldsymbol{w})$ is globally Lipschitz continuous, a proof is in, for example, Mangasarian (2002, Section 3). Therefore, any limit point of $\{\boldsymbol{w}^k\}$ is an optimum of (1) with L2 loss by Theorem 1(e) in Tseng and Yun (2009).

Further, if the L2-loss function $L(\boldsymbol{w})$ is strictly convex, (43) is satisfied with L2 loss following the proof in Appendix B. Hence, $\{\boldsymbol{w}^k\}$ converges to the unique optimum solution at least linearly.

## 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.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

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`.

Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57:1413–1457, 2004.

Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination in sparse supervised learning. Technical report, EECS Department, University of California, Berkeley, 2010.

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`.

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.

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

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`.

Cho-Jui Hsieh, Matyas A. Sustik, Pradeep Ravikumar, and Inderjit S. Dhillon. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems (NIPS) 24*, 2011.

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.

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`.

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.

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.

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

Nicol N. Schraudolph. A fast, compact approximation of the exponential function. *Neural Computation*, 11:853–862, 1999.

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, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonatha Taylor, and Ryan J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of Royal Statistical Society: Series B*, 2011.

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

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/l1.pdf`.

Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. An improved GLMNET for l1-regularized logistic regression. In *Proceedings of the Seventeenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 33–41, 2011.