Efficient L_1 Regularized Logistic Regression

Su-In Lee, Honglak Lee, Pieter Abbeel and Andrew Y. Ng

Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

L_1 regularized logistic regression is now a workhorse of machine learning: it is widely used for many classification problems, particularly ones with many features. L_1 regularized logistic regression requires solving a convex optimization problem. However, standard algorithms for solving convex optimization problems do not scale well enough to handle the large datasets encountered in many practical settings. In this paper, we propose an efficient algorithm for L_1 regularized logistic regression. Our algorithm iteratively approximates the objective function by a quadratic approximation at the current point, while maintaining the L_1 constraint. In each iteration, it uses the efficient LARS (Least Angle Regression) algorithm to solve the resulting L_1 constrained quadratic optimization problem. Our theoretical results show that our algorithm is guaranteed to converge to the global optimum. Our experiments show that our algorithm significantly outperforms standard algorithms for solving convex optimization problems. Moreover, our algorithm outperforms four previously published algorithms that were specifically designed to solve the L_1 regularized logistic regression problem.

Introduction

Logistic regression is widely used in machine learning for classification problems. It is well-known that regularization is required to avoid over-fitting, especially when there is a only small number of training examples, or when there are a large number of parameters to be learned. In particular, L_1 regularized logistic regression is often used for feature selection, and has been shown to have good generalization performance in the presence of many irrelevant features. (Ng 2004; Goodman 2004)

Unregularized logistic regression is an unconstrained convex optimization problem with a continuously differentiable objective function. As a consequence, it can be solved fairly efficiently with standard convex optimization methods, such as Newton's method or conjugate gradient. However, adding the L_1 regularization makes the optimization problem computationally more expensive to solve. If the L_1 regularization is enforced by an L_1 norm constraint on the parameter vector, then the optimization problem becomes a constraint optimization problem with the number of constraints equal to twice the number of parameters to be learned. The running time of standard algorithms to solve convex optimization problems increases with the number of constraints. If, alternatively, the L_1 regularization is implemented by adding (a constant times) the L_1 norm of the parameter vector to the objective, then the objective is not continuously differentiable anymore. This precludes the use of, for example, Newton's method to solve the optimization problem efficiently. Thus, both formulations of L_1 regularized logistic regression lead to a more complicated optimization problem than unregularized logistic regression. However, the L_1 constraint (or penalty term) has significant structure. For example, if one knew a priori the signs of each of the parameters, then the L_1 constraint would become a single linear constraint. Or, in case of regularization by adding the L_1 term to the objective, the L_1 term would become linear, resulting in a continuously differentiable objective. These (and similar) observations have led to various algorithms that try to exploit the structure in the optimization problem and solve it more efficiently.

Goodman (2004) proposed an algorithm for learning conditional maximum entropy models (of which logistic regression is a special case) with an exponential prior (also referred to as a one-sided Laplacian prior). His algorithm is based on generalized iterative scaling (GIS). The idea is to iteratively (efficiently) maximize a differentiable lower bound of the objective function. The proposed learning algorithm can easily be extended to the case of logistic regression with a Laplacian prior by duplicating all the features with the opposite sign. Logistic regression with a Laplacian prior is equivalent to L_1 regularized logistic regression.

Perkins et al. (2003) proposed a method called *grafting*. The key idea in grafting is to incrementally build a subset of the parameters, which are allowed to differ from zero. Grafting uses a local derivative test in each iteration of the conjugate gradient method, to choose an additional feature that is allowed to differ from zero. This way, the conjugate gradient method can be applied without explicitly dealing with the discontinuous first derivatives.

Roth (2004) proposed an algorithm called *generalized LASSO* that extends a LASSO algorithm proposed by Osborne et al. (2000). (The LASSO refers to an L_1 regularized

least squares problem. See, Tibshirani (1996) for details.)

In this paper, we propose a new, efficient algorithm for solving L_1 regularized logistic regression. Our algorithm is based on the iteratively reweighted least squares (IRLS) formulation of logistic regression. More specifically, in each iteration, our algorithm finds a step direction by optimizing the quadratic approximation of the objective function at the current point subject to the L_1 norm constraint. The IRLS formulation of logistic regression allows us to (iteratively) reformulate the quadratic approximation as a least squares objective. Thus our algorithm ends up solving an L_1 constrained least squares problem in every iteration. The L_1 constrained least squares problem can be solved very efficiently using least angle regression (LARS), proposed by Efron et al. (2004).

Our theoretical results show that our algorithm is guaranteed to converge to a global optimum. (In fact, our theoretical results could be generalized to a larger family of optimization problems, as discussed later.)

Our experiments show that our algorithm significantly outperforms the previously proposed algorithms for L_1 regularized logistic regression. Our algorithm also significantly outperforms standard gradient-based algorithms, such as conjugate gradient and Newton's method.[1]

We note that Lokhorst (1999) also proposed an algorithm that uses the IRLS formulation of logistic regression. However he used a different LASSO algorithm (Osborne, Presnell, & Turlach 2000) in the inner loop and did not provide any formal convergence guarantees.

L_1 Regularized Logistic Regression

We consider a supervised learning task where we are given M training instances $\{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \ldots, M\}$. Here, each $\mathbf{x}^{(i)} \in \mathbb{R}^N$ is an N-dimensional feature vector, and $y^{(i)} \in \{0, 1\}$ is a class label. Logistic regression models the probability distribution of the class label y given a feature vector $\mathbf{x}$ as follows:

$$p(y = 1|\mathbf{x}; \theta) = \sigma(\theta^\top \mathbf{x}) = \frac{1}{1 + \exp(-\theta^\top \mathbf{x})}. \quad (1)$$

Here $\theta \in \mathbb{R}^N$ are the parameters of the logistic regression model; and $\sigma(\cdot)$ is the sigmoid function, defined by the second equality.

Under the Laplacian prior $p(\theta) = (\beta/2)^N \exp(-\beta\|\theta\|_1)$ (with $\beta > 0$), the *maximum a posteriori* (MAP) estimate of the parameters θ is given by:

$$\min_\theta \sum_{i=1}^{M} -\log p(y^{(i)}|\mathbf{x}^{(i)}; \theta) + \beta\|\theta\|_1. \quad (2)$$

This optimization problem is referred to as L_1 regularized logistic regression. Often, it will be convenient to consider the following alternative parameterization of the L_1 regularized logistic regression problem:

$$\min_\theta \sum_{i=1}^{M} -\log p(y^{(i)}|\mathbf{x}^{(i)}; \theta), \quad (3)$$
$$\text{subject to } \|\theta\|_1 \leq C.$$

The optimization problems (2) and (3) are equivalent, in the following sense: for any choice of β, there is a choice of C such that both optimization problems have the same minimizing argument. This follows from the fact that, (up to a constant that does not depend on θ) the optimization problem (2) is the Lagrangian of the constrained optimization problem (3), where β is the Lagrange multiplier. (Rockafellar 1970) In practice, C (and/or β) may be either chosen manually or via cross-validation; for the sake of simplicity, in this paper, we will consider solving the problem (3) for a given, fixed, value of C.

Our Learning Algorithm

In this section, we describe our learning algorithm for L_1 regularized logistic regression. We also formally prove that our learning algorithm converges to the global optimum of the optimization problem (3).

Preliminaries

IRLS for unregularized logistic regression Our learning algorithm is based on iteratively reweighted least squares (IRLS). (Green 1984; Minka 2003) IRLS reformulates the problem of finding the step direction for Newton's method as a weighted ordinary least squares problem.

Consider the problem of finding the maximum likelihood estimate (MLE) of the parameters θ for the unregularized logistic regression model. The optimization problem can be written as:

$$\min_\theta \sum_{i=1}^{M} -\log p(y^{(i)}|\mathbf{x}^{(i)}; \theta). \quad (4)$$

In every iteration, Newton's method first finds a step direction by approximating the objective by the second order Taylor expansion at the current point, and optimizing the quadratic approximation in closed-form. More specifically, let $\theta^{(k)}$ denote the current point in the k'th iteration. Then Newton's method finds a step direction by computing the optimum $\gamma^{(k)}$ of the quadratic approximation at $\theta^{(k)}$ as follows:

$$\gamma^{(k)} = \theta^{(k)} - \mathbf{H}^{-1}(\theta^{(k)})\mathbf{g}(\theta^{(k)}). \quad (5)$$

Here $\mathbf{H}(\theta)$ and $\mathbf{g}(\theta)$ represent the Hessian and gradient of the objective function in (4) evaluated at θ. Once the step direction $\gamma^{(k)}$ is computed, Newton's method computes the next iterate

$$\theta^{(k+1)} = (1 - t)\theta^{(k)} + t\gamma^{(k)} \quad (6)$$

by a line search over the step size parameter t.

We now show how the Newton step direction can be computed by solving a weighted ordinary least squares problem

[1] The standard gradient-based algorithms are not directly applicable, because the objective function of the L_1 regularized logistic regression has discontinuous first derivatives. In our experiments, we used a smooth approximation of the L_1 loss function.

rather than using Equation (5). (See Green 1984, or Minka 2003 for details of this derivation.) Let $\mathbf{X}(\in \mathbb{R}^{N \times M})$ denote the design matrix:

$$\mathbf{X} = [\mathbf{x}^{(1)}\mathbf{x}^{(2)} \dots \mathbf{x}^{(M)}].$$

Let the diagonal matrix $\mathbf{\Lambda}$ and the vector $\mathbf{z}$ be defined as follows: for all $i = 1, 2, \dots, M$:

$$\Lambda_{ii} = \sigma(\theta^{(k)\top}\mathbf{x}^{(i)})[1 - \sigma(\theta^{(k)\top}\mathbf{x}^{(i)})], \quad (7)$$

$$z_i = \mathbf{x}^{(i)\top}\theta^{(k)} + \frac{[1 - \sigma(y^{(i)}\theta^{(k)\top}\mathbf{x}^{(i)})]y^{(i)}}{\mathbf{\Lambda}_{ii}}. \quad (8)$$

Then, we have that $\mathbf{H}(\theta^{(k)}) = -\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\top$ and $\mathbf{g}(\theta^{(k)}) = \mathbf{X}\mathbf{\Lambda}(\mathbf{z} - \mathbf{X}^\top\theta^{(k)})$, and thus Equation (5) can be rewritten as:

$$\gamma^{(k)} = (\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{\Lambda}\mathbf{z}. \quad (9)$$

Thus $\gamma^{(k)}$ is the solution to the following weighted least squares problem:

$$\gamma^{(k)} = \arg\min_\gamma \|(\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{X}^\top)\gamma - \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{z}\|_2^2. \quad (10)$$

Therefore, the Newton step direction can be computed by solving the least squares problem (10), and thus the logistic regression optimization problem can be solved using iteratively reweighted least squares (IRLS). This least squares formulation will play an important role in our algorithm. The term IRLS refers to the fact that in every iteration the least squares problem of Equation (10) has a different diagonal weighting matrix $\mathbf{\Lambda}$ and "right-hand side" $\mathbf{z}$, but the same design matrix $\mathbf{X}$. (Note that, although we did not make the dependence explicit, the diagonal matrix $\mathbf{\Lambda}$ and the vector $\mathbf{z}$ both depend on $\theta^{(k)}$ and therefore change in every iteration.)

IRLS for L_1 regularized logistic regression For the case of L_1 regularized logistic regression, as formulated in Equation (3), the objective is equal to the unregularized logistic regression objective. By augmenting the IRLS formulation of the unregularized logistic regression with the L_1 constraint, we get our IRLS formulation for L_1 regularized logistic regression (leaving out the dependencies on k for clarity):

$$\min_\gamma \|(\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{X}^\top)\gamma - \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{z}\|_2^2, \quad (11)$$
$$\text{subject to } \|\gamma\|_1 \leq C$$

Our algorithm (presented in more detail below) iteratively solves the L_1 regularized logistic regression problem by (in every iteration) first solving the L_1 constrained least squares problem (11) to find a step direction and then doing a line search to determine the step size. Thus our algorithm combines a local quadratic approximation with the global constraints.

LARS for solving L_1 constrained least squares. The L_1 constrained least squares problem of Equation (11) is also known as a LASSO problem. (Tibshirani 1996) Several algorithms have been developed to solve L_1 constrained least squares problems. In our algorithm, we use LARS with the LASSO modification. (Efron *et al.* 2004) LARS with the LASSO modification uses a stagewise procedure, based on a simple incremental update, to efficiently solve the L_1 regularized least squares problem.

Our algorithm

Our algorithm iteratively solves the L_1 constrained logistic regression problem of Equation (3). In the k'th iteration, it finds a step direction $\gamma^{(k)}$ by solving the constrained least squares problem of Equation (11). Then it performs a backtracking line search (see Boyd & Vandenberghe, 2004 for details) over the step size parameter $t \in [0, 1]$ to find the next iterate $\theta^{(k+1)} = (1-t)\theta^{(k)} + t\gamma^{(k)}$. To summarize, we propose the following algorithm, which we refer to as IRLS-LARS. (We discuss the stopping criterion used in more detail when discussing our experimental setup and results.)

Algorithm 1 IRLS-LARS

Set $\theta^{(0)} = \vec{0}$.
for ($k = 0$ **to** MaxIterations)
  Compute $\mathbf{\Lambda}$ and $\mathbf{z}$ using Equation (7) and Equation (8) (for $\theta = \theta^{(k)}$).
  Use LARS to solve the L_1 constrained least squares problem of Equation (11), and let $\gamma^{(k)}$ be the solution.
  Set $\theta^{(k+1)} = (1-t)\theta^{(k)} + t\gamma^{(k)}$, where t is found using a backtracking line-search that minimizes the objective function given in Equation (3).
  Evaluate the objective function, given in Equation (3) at $\theta^{(k+1)}$.
  if (the stopping criterion is satisfied)
    Break;
  end
end

We note that our algorithm can be applied to, not only L_1 constrained logistic regression, but rather to any L_1 constrained optimization problems, so long as the unconstrained problem can be solved using IRLS. In particular, our algorithm can be used for parameter learning for L_1 constrained generalized linear models. (Nelder & Wedderburn 1972; McCullagh & Nelder 1989)

Convergence and optimality guarantees

The following theorem shows that the IRLS-LARS algorithm presented in this paper converges to a global optimum.[2]

Theorem 1. *The IRLS-LARS algorithm presented in the paper converges to a global optimum of the L_1 constrained regularized logistic regression problem of Equation (3).*

[2]In fact—although the theorem statement is specific to our algorithm—the proof easily generalizes to any algorithm that iteratively minimizes the local quadratic approximation subject to the constraints (followed by a line search) so long as the objective function has a continuous 3rd order derivative, and the feasible set is convex, closed and bounded.

Proof. Our convergence proof uses the following fact:

Fact 1. *Let any point θ_0 that is not a global optimum be given. Then there is an open subset S_{θ_0} that includes θ_0, and there is a constant $K_{\theta_0} > 0$ such that the following holds: for all feasible $\phi_0 \in S_{\theta_0}$ we have that one iteration of our algorithm, starting from ϕ_0, will give us a (feasible) point ϕ_1 that satisfies $f(\phi_1) \leq f(\phi_0) - K_{\theta_0}$.*

Proof of Fact 1. We have a convex objective function f with convex constraints. Since θ_0 is not a global optimum, any open set that includes θ_0 includes a feasible point with strictly lower value of the objective function f. (In fact, all points on a line segment connecting θ_0 with a global optimum are feasible and have a strictly lower value of the objective function f.) The objective f is convex and has bounded second derivative over the feasible set, thus for any sufficiently small open set that includes θ_0, we have that this set includes a feasible point with strictly lower value of the local quadratic approximation at θ_0, which we denote by $\tilde{f}_{\theta_0}$. Thus when our algorithm globally optimizes the quadratic approximation over the feasible set, the resulting point θ_1 must satisfy $\tilde{f}_{\theta_0}(\theta_1) < \tilde{f}_{\theta_0}(\theta_0)$. Now define $\delta > 0$ as follows:

$$\delta = \tilde{f}_{\theta_0}(\theta_0) - \tilde{f}_{\theta_0}(\theta_1). \tag{12}$$

Since f is three times continuously differentiable and since the feasible set is bounded, there exists $\epsilon > 0$, such that for all $\phi_0 \in S_{\theta_0} = \{\theta : \|\theta - \theta_0\|_2 \leq \epsilon\}$ the following hold:

$$|f(\theta_0) - f(\phi_0)| \leq \frac{\delta}{8}, \tag{13}$$

and for all θ in the feasible set:

$$|\tilde{f}_{\theta_0}(\theta) - \tilde{f}_{\phi_0}(\theta)| \leq \frac{\delta}{8}. \tag{14}$$

From Equation (14) we have for all $\phi_0 \in S_{\theta_0}$ that

$$|\min_\theta \tilde{f}_{\theta_0}(\theta) - \min_\theta \tilde{f}_{\phi_0}(\theta)| \leq \tfrac{\delta}{4}. \tag{15}$$

Therefore, for all $\phi_0 \in S_{\theta_0}$ the following holds. Let ϕ_1 be the global optimum (within the feasible set) of the quadratic approximation at ϕ_0, then we have that:

$$\begin{aligned}
\tilde{f}_{\phi_0}(\phi_1) &\leq \tilde{f}_{\theta_0}(\theta_1) + \frac{\delta}{4} \\
&\leq \tilde{f}_{\theta_0}(\theta_0) - \delta + \frac{\delta}{4} \\
&= f(\theta_0) - \frac{3\delta}{4} \\
&= f(\phi_0) + f(\theta_0) - f(\phi_0) - \frac{3\delta}{4} \\
&\leq f(\phi_0) + \frac{\delta}{8} - \frac{3\delta}{4} \\
&\leq f(\phi_0) - \frac{\delta}{2}.
\end{aligned} \tag{16}$$

Here we used, in order: Equation (15); Equation (12); simplification ($\tilde{f}_{\theta_0}(\theta_0) = f(\theta_0)$); adding and subtracting the same term; Equation (14); simplification.

Let t be the step size resulting from the line search. Then we have that:

$$\begin{aligned}
\tilde{f}_{\phi_0}(\phi_0 + t(\phi_1 - \phi_0)) &\leq (1-t)\tilde{f}_{\phi_0}(\phi_0) + t\tilde{f}_{\phi_0}(\phi_1) \\
&\leq f(\phi_0) - t\frac{\delta}{2}.
\end{aligned} \tag{17}$$

Here we used, in order: $\tilde{f}_{\phi_0}$ is convex; $\tilde{f}_{\phi_0}(\phi_0) = f(\phi_0)$ and Equation (16).

Since f is three times continuously differentiable and the feasible set is bounded, there exists $C > 0$, such that for all $\phi_0 \in S_{\theta_0}$ the following holds for all θ in the feasible set:

$$f(\theta) \leq \tilde{f}_{\phi_0}(\theta) + C\|\theta - \phi_0\|_2^3. \tag{18}$$

Now, we choose $\theta = \phi_0 + t(\phi_1 - \phi_0)$ in Equation (18), and we use Equation (17) to obtain:

$$f(\phi_0 + t(\phi_1 - \phi_0)) \leq f(\phi_0) - t\tfrac{\delta}{2} + C\|t(\phi_1 - \phi_0)\|_2^3. \tag{19}$$

Let R be the radius of a sphere that includes the (bounded) feasible set, then we get that

$$f(\phi_0 + t(\phi_1 - \phi_0)) \leq f(\phi_0) - t\tfrac{\delta}{2} + Ct^3(2R)^3. \tag{20}$$

Choosing $t = \sqrt{\frac{\delta}{48CR^3}}$ gives us that for all $\phi_0 \in S_{\theta_0}$, we have that our algorithm finds a ϕ^* such that:

$$f(\phi^*) \leq f(\phi_0) - \sqrt{\frac{\delta^3}{192CR^3}}.$$

Choosing $K_{\theta_0} = \sqrt{\frac{\delta^3}{192CR^3}}$ proves Fact 1. □

Now we established Fact 1, we are ready to prove the theorem. Let any $\delta > 0$ be fixed. Let $P_\delta = \{\theta : \theta \text{is feasible}, \forall \text{ global optima } \theta^*, \|\theta - \theta^*\| \geq \delta\}$. We will show convergence to a global optimum by showing that the algorithm can only spend a finite number of iterations in the set P_δ.

From Fact 1 we have that for all $\theta \in P_\delta$, there exists an open set S_θ and a positive constant $K_\theta > 0$ such that the algorithm improves the objective by at least K_θ in one iteration, when started from a point in the set S_θ. Since such a set S_θ exists for all $\theta \in P_\delta$ we have that

$$P_\delta \subseteq \cup_{\theta \in P_\delta} S_\theta.$$

Since the set P_δ is a closed and bounded subset of $\mathbb{R}^N$, we have (from the Heine-Borel theorem) that P_δ is compact. By definition of compactness, for every open cover of a compact set there exists a finite sub-cover. (See, e.g., Royden, 1988, for more details.) Thus, there is a finite set Q_δ such that

$$P_\delta \subseteq \cup_{\theta \in Q_\delta} S_\theta. \tag{21}$$

Since Q_δ is finite, we have that there is some $C_\delta > 0$ such that

$$\inf_{\theta \in Q_\delta} K_\theta = C_\delta.$$

Since Q_δ covers P_δ, we have that for all points $\theta \in P_\delta$ our algorithm improves the objective by at least C_δ in every iteration. The objective is bounded, so this means the algorithm has to leave the set P_δ after a finite number of iterations. Since this holds true for arbitrary $\delta > 0$, we showed that the algorithm converges to a global optimum. □

Name	Description
IRLS-LARS	Our algorithm
GenLASSO	Generalized LASSO (Roth 2004)
SCGIS	Sequential Conditional GIS (Goodman 2004)
Gl1ce	Gl1ce (Lokhorst 1999)
Grafting	Grafting (Perkins & Theiler 2003)
CG-L1	Conjugate gradient with exact L_1
NM-epsL1	Newton's method with epsL1
CG-epsL1	Conjugate gradient with epsL1
CG-Huber	Conjugate gradient with Huber

Table 1: The nine algorithms that were used in our analysis. (See text for details.)

Experimental Results

We compared the computational efficiency of our algorithm with the eight other algorithms listed in Table 1. We tested each algorithm's performance on 12 different datasets, consisting of 9 UCI datasets (Newman *et al.* 1998), one artificial dataset called *Madelon* from the NIPS 2003 workshop on feature extraction,[3] and two gene expression datasets (Microarray 1 and 2).[4] Table 2 gives details on the number of training examples and the number of features for each dataset.

Experimental details on the other algorithms

We compared our algorithm (IRLS-LARS) to four previously published algorithms: Grafting (Perkins & Theiler 2003), Generalized LASSO (Roth 2004), SCGIS (Goodman 2004), and Gl1ce (Lokhorst 1999).

Three of these algorithms (Gl1ce, SCGIS and Grafting) and our algorithm (IRLS-LARS) were implemented in MATLAB. In our implementations, we went through great efforts to carefully implement all the optimization techniques suggested by the corresponding papers. We used Volker Roth's C-implementation for GenLASSO.[5]

We also compared IRLS-LARS to standard optimization algorithms, such as conjugate gradient method and Newton's method. These algorithms are not directly applicable to L_1 logistic regression because the objective function has discontinuous first derivatives. Thus, we used a smooth approximation of the L_1 penalty term. More specifically, we have that the L_1 term is given by

$$\|\theta\|_1 = \sum_{j=1}^{N} |\theta_j|.$$

We used the "epsL1" function defined as:

$$\text{epsL1}(\theta_j, \epsilon) = \sqrt{\theta_j^2 + \epsilon},$$

to obtain a smooth approximation for each of the terms $|\theta_j|$. We similarly used the Huber loss function (see, e.g., Boyd & Vandenberghe 2004):

$$\text{Huber}(\theta_j, \epsilon) = \begin{cases} \theta_j^2/(2\epsilon) & \text{if } |\theta_j| < \epsilon \\ |\theta_j| - \epsilon/2 & \text{otherwise.} \end{cases}$$

We also tested conjugate gradient with the exact (non-differentiable) L_1 norm penalty (CG-L1). In this case, we treated the first derivative of the L_1 terms as zero when evaluated at zero. More specifically we treated $\frac{\partial |\theta_j|}{\partial \theta_j}$ as being zero for $\theta_j = 0$.

We went through great efforts to ensure a fair comparison by optimizing the performance of all other algorithms. We carefully profiled all algorithms and made optimizations accordingly. We also spent significant amounts of time tuning each algorithm's parameters (when applicable). For example, for the algorithms that are based on conjugate gradient and Newton's method, we extensively tuned the parameters of the line-search algorithm; for the algorithms using conjugate gradient (Grafting, CG-epsL1, CG-Huber and CG-L1), we tested both our own conjugate gradient implementation as well as the MATLAB optimization toolbox's conjugate gradient; for the algorithms that use the approximate L_1 penalty term, we tried many choices for ϵ in the range ($10^{-15} < \epsilon < 0.01$) and chose the one with the shortest running time; etc. In all cases, we are reporting the running times for those parameter settings (and other choices) which resulted in the shortest running times.

Name	Number of features	Number of samples
Arrhythmia	279	420
Hepatitis	19	80
Ionosphere	33	351
Promoters	57	106
Spambase	57	4601
Spect	22	267
Spectf	44	349
Splice	60	1527
Wisconsin breast cancer	30	569
Madelon	500	2600
Microarray1	2000	62
Microarray2	3571	72

Table 2: The 12 datasets we used in our experiments. (See text for details.)

[3] Madelon is available at: `http://www.clopinet.com/isabelle/Projects/NIPS2003/`.

[4] Microarray 1 is a colon cancer microarray dataset available at: `http://microarray.princeton.edu/oncology/` Microarray 2 is a leukemia microarray dataset available at: `http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43`.

[5] The code for GenLASSO is available at: `http://www.informatik.uni-bonn.de/~roth/GenLASSO/index.html`. His implementation contained some unnecessary steps (for the purposes of our evaluation), such as automatically performing the cross-validation for determining the regularization parameter C. We went carefully through his code and removed these additional computations.

Choice of the regularization parameters

For a fixed dataset, the choice of the regularization parameter C (or β) greatly affects the running time of the algorithms. Thus, we ran 10-fold cross validation (for each dataset) and chose the regularization parameter that achieves the highest (hold-out) accuracy. Doing so ensures that we evaluate the algorithms in the regime we care about for classification.

In particular, we ran our algorithm (IRLS-LARS) to determine the regularization parameter C. This value of C was then used for all learning algorithms that use the constrained version of L_1 regularized logistic regression, namely, IRLS-LARS, GenLASSO, and Gl1ce.

The other algorithms solve the formulation with the L_1 penalty term (or a smooth version thereof), as given in Equation (2). As discussed in more detail in our description of L_1 regularized logistic regression, the optimization problems of Equation (2) and Equation (3) are equivalent, in the following sense: for any choice of C, there is a choice of β such that the solution to both problems is the same. We now show how to find β explicitly for a given value of C.

From the Karush-Kuhn-Tucker (KKT) conditions, we have that the optimal primal solution θ^* and the optimal dual solution β^* (of the constrained optimization problem) satisfy the following conditions:

$$\left.\frac{\partial \sum_{i=1}^{M} \log p(y^{(i)}|\mathbf{x}^{(i)};\theta)}{\partial \theta_j}\right|_{\theta^*} = \beta^* \text{sign}(\theta_j), \qquad (22)$$

for all $\theta_j^* \neq 0$. From Lagrange duality, we also have that the optimal primal solution θ^* is given by

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{M} -\log p(y^{(i)}|\mathbf{x}^{(i)};\theta) + \beta^* \|\theta\|_1. \qquad (23)$$

Thus, by choosing

$$\beta = \beta^* = \text{sign}(\theta_j) \left.\frac{\partial \sum_{i=1}^{M} \log p(y^{(i)}|\mathbf{x}^{(i)};\theta)}{\partial \theta_j}\right|_{\theta^*} \qquad (24)$$

(for any j such that ${\theta^*}_j \neq 0$). We ensure that the optimal solution of the formulation with L_1 penalty term gives the same optimal solution as the formulation with the L_1 constraint ($\|\theta\|_1 \leq C$).[6]

Running time

The efficiency of an algorithm was measured based on the running time, i.e., the CPU time measured in seconds until the algorithm meets the stopping criterion. Each algorithm was run on a Linux machine with AMD Opteron 2GHz with 4.6GB RAM.[7]

[6]Effectively the KKT condition as in (22) is not exactly satisfied for approximate L_1 functions. However, typical ϵ value turned out to be very small($\approx 10^{-10}$) such that the resulting approximate L_1 functions were very close to the true L_1 functions.

[7]For the runs that took longer than 500 seconds, we truncated the running time to 500 seconds.

Stopping criterion For each algorithm the stopping criterion is based on how close the current objective function value (denoted by obj) is to the optimal objective function value (denoted by obj^*). For each dataset, we first compute obj^* as follows. First, we run our algorithm (IRLS-LARS) for an extensive amount of time to obtain the solution θ^*. Then, we use θ^* to compute β^* from Equation (24). Finally, we verify whether the KKT optimality conditions (see Boyd & Vandenberghe, 2004) are satisfied for the values for θ^* and β^* we found. In all cases, we found that the KKT conditions were satisfied up to a very small error.[8]

Using the optimal objective obj^*, we defined the stopping criterion as:

$$\frac{|obj^* - obj|}{|obj^*|} < \tau. \qquad (25)$$

We used $\tau = 10^{-6}$ for our algorithm, Grafting, SCGIS, and GenLASSO. For the algorithms with the approximate L_1 functions, we used a less stringent bound ($\tau = 10^{-5}$) since the objective function with the approximate L_1 penalty term is slightly different from from the original one formulated in Equation (2).

Discussion of experimental results

Figure 1 shows the results for the five algorithms specifically designed for L_1 regularized logistic regression (IRLS-LARS, Grafting, SCGIS, GenLASSO and Gl1ce) on 12 datasets. Since the running times for the standard optimization methods (CG-L1, NM-epsL1, CG-epsL1 and CG-Huber) tend to be significantly higher, we do not report them in the same figure. (Doing so allows us to visualize more clearly the differences in performance for the first set of algorithms.) We report the performance of the standard optimization algorithms (and our algorithm) in Table 3. In all cases we report the average running time (in seconds) over 100 trials of the same experiment (dataset and algorithm combination).

The results reported in Figure 1 show that our algorithm (IRLS-LARS) was more efficient than all other algorithms in 11 out of 12 datasets. On one dataset (Microarray2) GenLasso and Gl1ce were slightly faster. All other algorithms were slower than our algorithm on every dataset.

IRLS-LARS significantly outperformed Grafting and SCGIS. More specifically, in 8 (out of 12) datasets our method was more than 8 times faster than Grafting. (In the other 4, it was also faster, although not by a factor of 8.) Our method was at least 10 times faster than SCGIS in 10 out of 12 datasets.

GenLASSO and Gl1ce are the strongest competitors, and even then our algorithm is more than twice as fast as Gl1ce

[8]In all cases, the primal constraint ($\|\theta\|_1 \leq C$) was satisfied with an error less than 10^{-13}, and Equation (24) was satisfied with an error less than 5×10^{-8}. All the other KKT conditions were satisfied exactly. We performed the same procedure using GenLASSO and obtained similar results. In all cases, the absolute difference between the obj^*'s obtained from running IRLS-LARS and the obj^*'s obtained from running GenLASSO was smaller than 10^{-8}; and the relative difference was always smaller than 10^{-10}.

Data	IRLS-LARS	NM-epsL1	CG-epsL1	CG-Huber	CG-L1
Arrhythmia	0.0420	2.8450	14.384	13.660	14.065
Hepatitis	0.0046	0.0258	0.0819	0.6580	0.6347
Ionosphere	0.0516	0.1534	0.1746	0.9717	0.9237
Promoters	0.0182	0.0632	0.2047	0.8418	0.7328
Spambase	4.0804	21.172	10.151	13.410	12.016
Spect	0.0345	0.0704	0.1206	0.7651	0.7293
Spectf	0.0646	0.2330	0.5493	1.1761	1.1235
Splice	0.8376	2.9828	0.9424	4.3407	4.8007
WBC[a]	0.1222	0.7225	4.9592	1.4854	1.3777
Madelon	0.8602	55.060	113.22	95.223	104.50
Microarray1	0.0330	242.88	500.00	500.00	500.00
Microarray2	0.4069	500.00	500.00	500.00	500.00

[a]Wisconsin breast cancer

Table 3: The running time (seconds) of IRLS-LARS, CG-L1 and three algorithms with the approximate L_1 functions (NM-epsL1, CG-epsL1 and CG-Huber). (See text for details.)

for 5 out of 12 datasets and more than twice as fast as GenLASSO for 8 out of 12 datasets. We note that (as opposed to ours and all the other methods) GenLASSO is highly-optimized C code (linked with highly optimized numerical libraries). It is widely accepted that C code is faster than MATLAB code, and thus we are giving GenLASSO a somewhat unfair advantage in our comparison.[9]

As reported in Table 3, conjugate gradient and Newton's method are an order of magnitude slower than our IRLS-LARS algorithm in most of the datasets (despite the less stringent stopping criterion). Among these methods, NM-epsL1 seems to be the second best algorithm on the datasets with a small number of features. However, this method quickly becomes inefficient as the number of features increases. Also other methods similarly suffered from the increasing number of features. We observe that CG-L1 is very competitive with CG-epsL1 and CG-Huber.

Conclusions

In this paper, we proposed a new algorithm for learning L_1 regularized logistic regression. Our theoretical results show that our algorithm is guaranteed to converge to a global optimum. Our experiments showed that our algorithm is computationally very efficient. In particular, it outperforms standard algorithms for solving convex optimization problems, as well as other algorithms previously developed specifically to solve the L_1 regularized logistic regression problem.

[9]We have implemented GenLASSO in MATLAB. However, the extensive optimization tricks used in the C-code and numerical libraries were hard to be duplicated in MATLAB, making it hard to evaluate GenLASSO correctly with our MATLAB code. Since C code is widely accepted to be at least as fast as MATLAB code, we decided to report the C code results (possibly favoring GenLASSO somewhat in our comparisons).

Acknowledgments

We give warm thanks to John Lafferty, Alexis Battle, Ben Packer, and Vivek Farias for helpful comments. H. Lee was supported by Stanford Graduate Fellowship (SGF). This work was supported in part by the Department of the Interior/DARPA under contract number NBCHD030010.

References

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Efron, B.; Hastie, T.; Johnstone, I.; and Tibshirani, R. 2004. Least angle regression. *Annals of Statistics* 32(2):407–499.

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

Green, P. J. 1984. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *Journal of the Royal Statistical Society, Series B, Methodological* 46:149–192.

Lokhorst, J. 1999. *The LASSO and Generalised Linear Models*. Honors Project, Department of Statistics, The University of Adelaide, South Australia, Australia.

McCullagh, P., and Nelder, J. A. 1989. *Generalized Linear Models*. Chapman & Hall.

Minka, T. P. 2003. A comparison of numerical optimizers for logistic regression. `http://research.microsoft.com/~minka/papers/logreg/`.

Nelder, J. A., and Wedderburn, R. W. M. 1972. Generalized linear models. *Journal of the Royal Statistical Society, Series A, General* 135(3):370–384.

Newman, D. J.; Hettich, S.; Blake, C. L.; and Merz, C. 1998. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Ng, A. Y. 2004. Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *International Conference on Machine Learning*.

Osborne, M.; Presnell, B.; and Turlach, B. 2000. On the lasso and its dual. *Journal of Computational and Graphical Statistics* 9:319–337.

Perkins, S., and Theiler, J. 2003. Online feature selection using grafting. In *International Conference on Machine Learning*.

Rockafellar, R. T. 1970. *Convex analysis*. Princeton Univ. Press.

Roth, V. 2004. The generalized lasso. *IEEE Trans. Neural Networks* 15(1):16–28.

Royden, H. 1988. *Real Analysis*. Prentice Hall.

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

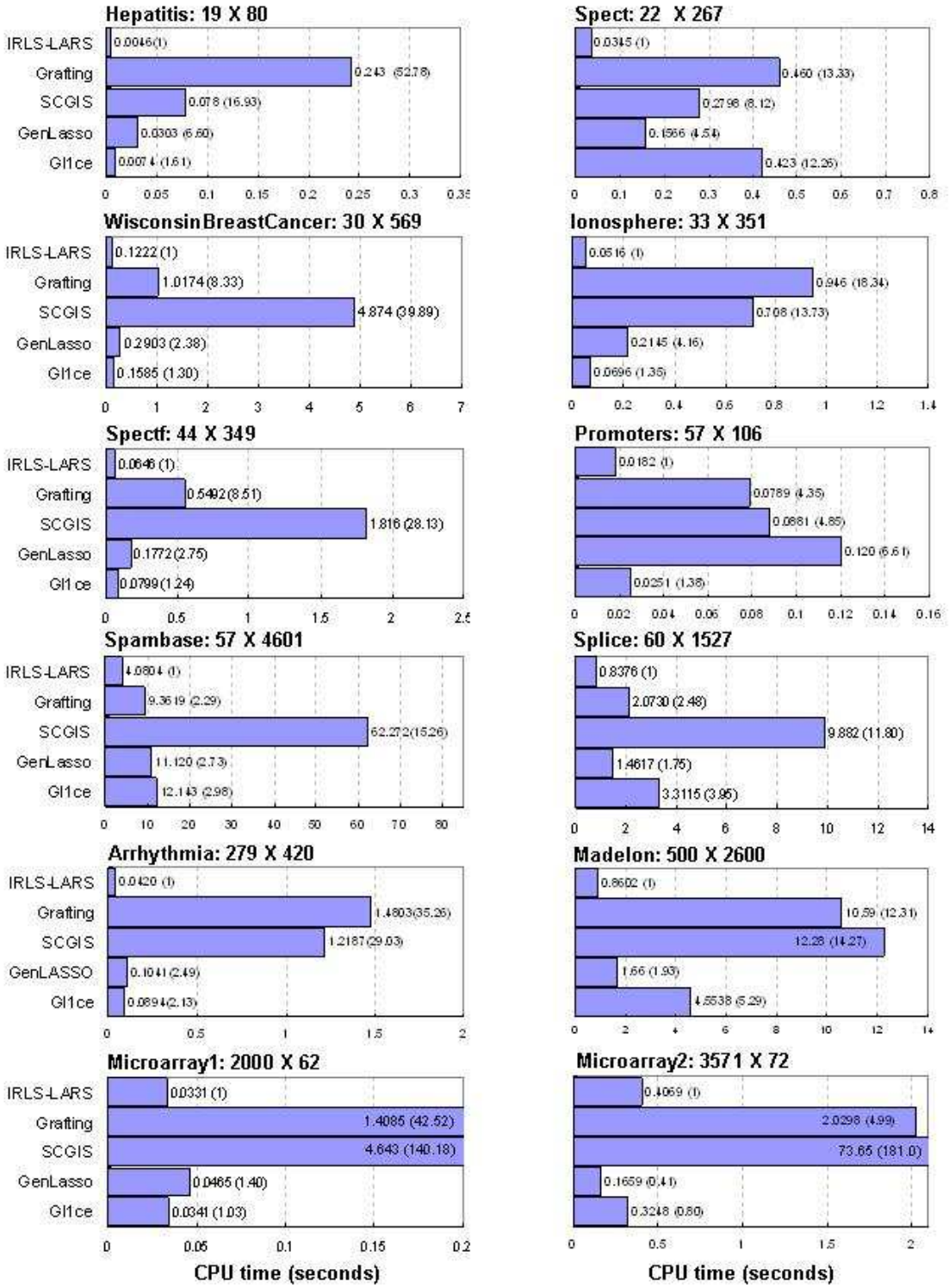

Figure 1: Comparison of the running time (seconds) among five algorithms (IRLS-LARS, Grafting, SCGIS, GenLASSO and Gl1ce) over 12 datasets. Each plot corresponds to one dataset. The name of the dataset is indicated at the top of the plot, and the number of features (N) and the number of training instances (M) are shown as "Name of the data: $N \times M$". For each experiment, we report the running time in seconds, and the running time relative to IRLS-LARS in parentheses. (See text for discussion.)