Journal of Machine Learning Research 8 (2007) 1519-1555 Submitted 7/06; Revised 1/07; Published 7/07

An Interior-Point Method for Large-Scale ℓ_1-Regularized Logistic Regression

Kwangmoo Koh DENEB1@STANFORD.EDU
Seung-Jean Kim SJKIM@STANFORD.EDU
Stephen Boyd BOYD@STANFORD.EDU
Information Systems Laboratory
Electrical Engineering Department
Stanford University
Stanford, CA 94305-9510, USA

Editor: Yi Lin

Abstract

Logistic regression with ℓ_1 regularization has been proposed as a promising method for feature selection in classification problems. In this paper we describe an efficient interior-point method for solving large-scale ℓ_1-regularized logistic regression problems. Small problems with up to a thousand or so features and examples can be solved in seconds on a PC; medium sized problems, with tens of thousands of features and examples, can be solved in tens of seconds (assuming some sparsity in the data). A variation on the basic method, that uses a preconditioned conjugate gradient method to compute the search step, can solve very large problems, with a million features and examples (e.g., the 20 Newsgroups data set), in a few minutes, on a PC. Using warm-start techniques, a good approximation of the entire regularization path can be computed much more efficiently than by solving a family of problems independently.

Keywords: logistic regression, feature selection, ℓ_1 regularization, regularization path, interior-point methods.

1. Introduction

In this section we describe the basic logistic regression problem, the ℓ_2- and ℓ_1-regularized versions, and the regularization path. We set out our notation, and review existing methods and literature. Finally, we give an outline of this paper.

1.1 Logistic Regression

Let $x \in \mathbf{R}^n$ denote a vector of explanatory or feature variables, and $b \in \{-1,+1\}$ denote the associated binary output or outcome. The logistic model has the form

$$\mathrm{Prob}(b|x) = \frac{1}{1+\exp\left(-b(w^Tx+v)\right)} = \frac{\exp\left(b(w^Tx+v)\right)}{1+\exp\left(b(w^Tx+v)\right)},$$

where $\mathrm{Prob}(b|x)$ is the conditional probability of b, given $x \in \mathbf{R}^n$. The logistic model has parameters $v \in \mathbf{R}$ (the intercept) and $w \in \mathbf{R}^n$ (the weight vector). When $w \neq 0$, $w^Tx+v=0$ defines the neutral hyperplane in feature space, on which the conditional probability of each outcome is $1/2$. On the shifted parallel hyperplane $w^Tx+v=1$, which is a distance $1/\|w\|_2$ from the neutral hyperplane, the conditional probability of outcome $b = 1$ is $1/(1+1/e) \approx 0.73$, and the conditional probability of $b = -1$ is $1/(1+e) \approx 0.27$. On the hyperplane $w^T x + v = -1$, these conditional probabilities are reversed. As $w^T x + v$ increases above one, the conditional probability of outcome $b = 1$ rapidly approaches one; as $w^T x + v$ decreases below -1, the conditional probability of outcome $b = -1$ rapidly approaches one. The slab in feature space defined by $|w^T x + v| \le 1$ defines the *ambiguity region*, in which there is substantial probability of each outcome; outside this slab, one outcome is much more likely than the other.

Suppose we are given a set of (observed or training) examples,

$$(x_i, b_i) \in \mathbf{R}^n \times \{-1, +1\}, \quad i = 1, \ldots, m,$$

assumed to be independent samples from a distribution. We use $p_{\text{log}}(v,w) \in \mathbf{R}^m$ to denote the vector of conditional probabilities, according to the logistic model,

$$p_{\text{log}}(v,w)_i = \text{Prob}(b_i|x_i) = \frac{\exp(w^T a_i + v b_i)}{1 + \exp(w^T a_i + v b_i)}, \quad i = 1, \ldots, m,$$

where $a_i = b_i x_i$. The likelihood function associated with the samples is $\prod_{i=1}^m p_{\text{log}}(v,w)_i$, and the log-likelihood function is given by

$$\sum_{i=1}^m \log p_{\text{log}}(v,w)_i = -\sum_{i=1}^m f(w^T a_i + v b_i),$$

where f is the *logistic loss function*

$$f(z) = \log(1 + \exp(-z)). \tag{1}$$

This loss function is convex, so the log-likelihood function is concave. The negative of the log-likelihood function is called the *(empirical) logistic loss*, and dividing by m we obtain the *average logistic loss*,

$$l_{\text{avg}}(v,w) = (1/m) \sum_{i=1}^m f(w^T a_i + v b_i).$$

We can determine the model parameters w and v by maximum likelihood estimation from the observed examples, by solving the convex optimization problem

$$\text{minimize} \quad l_{\text{avg}}(v,w), \tag{2}$$

with variables $v \in \mathbf{R}$ and $w \in \mathbf{R}^n$, and problem data $A = [a_1 \ \cdots \ a_m]^T \in \mathbf{R}^{m \times n}$ and the vector of binary outcomes $b = [b_1 \ \cdots \ b_m]^T \in \mathbf{R}^m$. The problem (2) is called the *logistic regression problem* (LRP).

The average logistic loss is always nonnegative, that is, $l_{\text{avg}}(v,w) \ge 0$, since $f(z) \ge 0$ for any z. For the choice $w = 0$, $v = 0$, we have $l_{\text{avg}}(0,0) = \log 2 \approx 0.693$, so the optimal value of the LRP lies between 0 and $\log 2$. In particular, the optimal value can range (roughly) between 0 and 1. The optimal value is 0 only when the original data are linearly separable, that is, there exist w and v such that $w^T x_i + v > 0$ when $b_i = 1$, and $w^T x_i + v < 0$ when $b_i = -1$. In this case the optimal value of the LRP (2) is not achieved (except in the limit with w and v growing arbitrarily large). The optimal value is $\log 2$, that is, $w = 0$, $v = 0$ are optimal, only if $\sum_{i=1}^m b_i = 0$ and $\sum_{i=1}^m a_i = 0$. (This follows from the expression for ∇l_{avg}, given in Section 2.1.) This occurs only when the number of positive examples (i.e., those for which $b_i = 1$) is equal to the number of negative examples, and the average of x_i over the positive examples is the negative of the average value of x_i over the negative examples.

The LRP (2) is a smooth convex optimization problem, and can be solved by a wide variety of methods, such as gradient descent, steepest descent, Newton, quasi-Newton, or conjugate-gradients (CG) methods (see, for example Hastie et al., 2001, § 4.4).

Once we find maximum likelihood values of v and w, that is, a solution of (2), we can predict the probability of the two possible outcomes, given a new feature vector $x \in \mathbf{R}^n$, using the associated logistic model. For example, when $w \neq 0$, we can form the logistic classifier

$$\phi(x) = \operatorname{sgn}(w^T x + v), \tag{3}$$

where

$$\operatorname{sgn}(z) = \begin{cases} +1 & z > 0 \\ -1 & z \leq 0, \end{cases}$$

which picks the more likely outcome, given x, according to the logistic model. This classifier is linear, meaning that the boundary between the two decision outcomes is a hyperplane (defined by $w^T x + v = 0$).

1.2 ℓ_2-Regularized Logistic Regression

When m, the number of observations or training examples, is not large enough compared to n, the number of feature variables, simple logistic regression leads to over-fit. That is, the classifier found by solving the LRP (2) performs perfectly (or very well) on the training examples, but may perform poorly on unseen examples. Over-fitting tends to occur when the fitted model has many feature variables with (relatively) large weights in magnitude, that is, w is large.

A standard technique to prevent over-fitting is *regularization*, in which an extra term that penalizes large weights is added to the average logistic loss function. The ℓ_2*-regularized logistic regression problem* is

$$\text{minimize} \quad l_{\text{avg}}(v,w) + \lambda\|w\|_2^2 = (1/m)\textstyle\sum_{i=1}^m f(w^T a_i + v b_i) + \lambda \sum_{i=1}^n w_i^2. \tag{4}$$

Here $\lambda > 0$ is the regularization parameter, used to control the trade-off between the average logistic loss and the size of the weight vector, as measured by the ℓ_2-norm. No penalty term is imposed on the intercept, since it is a parameter for thresholding the weighted sum $w^T x$ in the linear classifier (3). The solution of the ℓ_2-regularized regression problem (4) (which exists and is unique) can be interpreted in a Bayesian framework, as the maximum a posteriori probability (MAP) estimate of w and v, when w has a Gaussian prior distribution on $\mathbf{R}^n$ with zero mean and covariance λI and v has the (improper) uniform prior on $\mathbf{R}$; see, for example, Chaloner and Larntz (1989) or Jaakkola and Jordan (2000).

The objective function in the ℓ_2-regularized LRP is smooth and convex, and so (like the ordinary, unregularized LRP) can be minimized by standard methods such as gradient descent, steepest descent, Newton, quasi-Newton, truncated Newton, or CG methods; see, for example, Luenberger (1984), Lin et al. (2007), Minka (2003), Nocedal and Wright (1999), and Nash (2000). Other methods that have been used include optimization transfer (Krishnapuram and Hartemink, 2005; Zhang et al., 2004) and iteratively re-weighted least squares (Komarek, 2004). Newton's method and variants are very effective for small and medium sized problems, while conjugate-gradients and limited memory Newton (or truncated Newton) methods can handle very large problems. In Minka (2003) the author compares several methods for ℓ_2-regularized LRPs with large data sets. The fastest methods turn out to be conjugate-gradients and limited memory Newton methods, outperforming IRLS, gradient descent, and steepest descent methods. Truncated Newton methods have been applied to large-scale problems in several other fields, for example, image restoration (Fu et al., 2006) and support vector machines (Keerthi and DeCoste, 2005). For large-scale iterative methods such as truncated Newton or CG, the convergence typically improves as the regularization parameter λ is increased, since (roughly speaking) this makes the objective more quadratic, and improves the conditioning of the problem.

1.3 ℓ_1-Regularized Logistic Regression

More recently, *ℓ_1-regularized logistic regression* has received much attention. The ℓ_1-regularized logistic regression problem is

$$\text{minimize} \quad l_{\text{avg}}(v,w) + \lambda\|w\|_1 = (1/m)\textstyle\sum_{i=1}^m f(w^T a_i + v b_i) + \lambda \sum_{i=1}^n |w_i|, \tag{5}$$

where $\lambda > 0$ is the regularization parameter. The only difference with ℓ_2-regularized logistic regression is that we measure the size of w by its ℓ_1-norm, instead of its ℓ_2-norm. A solution of the ℓ_1-regularized logistic regression must exist, but it need not be unique. Any solution of the ℓ_1-regularized logistic regression problem (5) can be interpreted in a Bayesian framework as a MAP estimate of w and v, when w has a Laplacian prior distribution and v has the (improper) uniform prior. The objective function in the ℓ_1-regularized LRP (5) is convex, but not differentiable (specifically, when any of the weights is zero), so solving it is more of a computational challenge than solving the ℓ_2-regularized LRP (4).

Despite the additional computational challenge posed by ℓ_1-regularized logistic regression, compared to ℓ_2-regularized logistic regression, interest in its use has been growing. The main motivation is that ℓ_1-regularized LR typically yields a *sparse* vector w, that is, w typically has relatively few nonzero coefficients. (In contrast, ℓ_2-regularized LR typically yields w with all coefficients nonzero.) When $w_j = 0$, the associated logistic model does not use the jth component of the feature vector, so sparse w corresponds to a logistic model that uses only a few of the features, that is, components of the feature vector. Indeed, we can think of a sparse w as a *selection* of the relevant or important features (i.e., those associated with nonzero w_j), as well as the choice of the intercept value and weights (for the selected features). A logistic model with sparse w is, in a sense, simpler or more parsimonious than one with nonsparse w. It is not surprising that ℓ_1-regularized LR can outperform ℓ_2-regularized LR, especially when the number of observations is smaller than the number of features (Ng, 2004; Wainwright et al., 2007).

We refer to the number of nonzero components in w as its *cardinality*, denoted $\mathrm{card}(w)$. Thus, ℓ_1-regularized LR tends to yield w with $\mathrm{card}(w)$ small; the regularization parameter λ roughly controls $\mathrm{card}(w)$, with larger λ typically (but not always) yielding smaller $\mathrm{card}(w)$.

The general idea of ℓ_1 regularization for the purposes of model or feature selection (or just sparsity of solution) is quite old, and widely used in other contexts such as geophysics (Claerbout and Muir, 1973; Taylor et al., 1979; Levy and Fullagar, 1981; Oldenburg et al., 1983). In statistics, it is used in the well-known Lasso algorithm (Tibshirani, 1996) for ℓ_1-regularized linear regression, and its extensions such as the fused Lasso (Tibshirani et al., 2005), the grouped Lasso (Kim et al., 2006; Yuan and Lin, 2006; Zhao et al., 2007), and the monotone Lasso (Hastie et al., 2007). The idea also comes up in signal processing in basis pursuit (Chen and Donoho, 1994; Chen et al., 2001), signal recovery from incomplete measurements (Candès et al., 2006, 2005; Donoho, 2006), and wavelet thresholding (Donoho et al., 1995), decoding of linear codes (Candès and Tao, 2005), portfolio optimization (Lobo et al., 2005), controller design (Hassibi et al., 1999), computer-aided design of integrated circuits (Boyd et al., 2001), computer vision (Bhusnurmath and Taylor, 2007), sparse principal component analysis (d'Aspremont et al., 2005; Zou et al., 2006), graphical model selection (Wainwright et al., 2007), maximum likelihood estimation of graphical models (Banerjee et al., 2006; Dahl et al., 2005), boosting (Rosset et al., 2004), and ℓ_1-norm support vector machines (Zhu et al., 2004). A recent survey of the idea can be found in Tropp (2006). Donoho and Elad (2003) and Tropp (2006) give some theoretical analysis of why ℓ_1 regularization leads to a sparse model in linear regression. Recently, theoretical properties of ℓ_1-regularized linear regression have been studied by several researchers; see, for example, Zou (2006), Zhao and Yu (2006), and Zou et al. (2007).

To solve the ℓ_1-regularized LRP (5), generic methods for nondifferentiable convex problems can be used, such as the ellipsoid method or subgradient methods (Shor, 1985; Polyak, 1987). These methods are usually very slow in practice, however. (Because ℓ_1-regularized LR typically results in a weight vector with (many) zero components, we cannot simply ignore the nondifferentiability of the objective in the ℓ_1-regularized LRP (5), hoping to not encounter points of nondifferentiability.)

Another approach is to transform the problem to one with differentiable objective and constraint functions. We can solve the ℓ_1-regularized LRP (5), by solving an equivalent formulation, with linear inequality constraints,

$$\begin{array}{ll} \text{minimize} & l_{\text{avg}}(v,w) + \lambda \mathbf{1}^T u \\ \text{subject to} & -u_i \le w_i \le u_i, \quad i = 1,\ldots,n, \end{array} \tag{6}$$

where the variables are the original ones $v \in \mathbf{R}$, $w \in \mathbf{R}^n$, along with $u \in \mathbf{R}^n$. Here $\mathbf{1}$ denotes the vector with all components one, so $\mathbf{1}^T u$ is the sum of the components of u. (To see the equivalence with the ℓ_1-regularized LRP (5), we note that at the optimal point for (6), we must have $u_i = |w_i|$, in which case the objectives in (6) and (5) are the same.) The reformulated problem (6) is a convex optimization problem, with a smooth objective, and linear constraint functions, so it can be solved by standard convex optimization methods such as SQP, augmented Lagrangian, interior-point, and other methods. High quality solvers that can directly handle the problem (6) (and therefore, carry out ℓ_1-regularized LR) include for example LOQO (Vanderbei, 1997), LANCELOT (Conn et al., 1992), MOSEK (MOSEK ApS, 2002), and NPSOL (Gill et al., 1986). These general purpose solvers can solve small and medium scale ℓ_1-regularized LRPs quite effectively.

Other recently developed computational methods for ℓ_1-regularized logistic regression include the IRLS method (Lee et al., 2006; Lokhorst, 1999), a generalized LASSO method (Roth, 2004) that extends the LASSO method proposed in Osborne et al. (2000) to generalized linear models, generalized iterative scaling (Goodman, 2004), bound optimization algorithms (Krishnapuram et al., 2005), online algorithms (Balakrishnan and Madigan, 2006; Perkins and Theiler, 2003), coordinate descent methods (Friedman et al., 2007; Genkin et al., 2006), and the Gauss-Seidel method (Shevade and Keerthi, 2003). Some of these methods can handle very large problems (assuming some sparsity in the data) with modest accuracy. But the additional computational cost required for these methods to achieve higher accuracy can be very large.

The main goal of this paper is to describe a specialized interior-point method for solving the ℓ_1-regularized logistic regression problem that is very efficient, for all size problems. In particular our method handles very large problems, attains high accuracy, and is not much slower than the fastest large-scale methods (conjugate-gradients and limited memory Newton) applied to the ℓ_2-regularized LRP.

Numerical experiments show that our method is as fast as, or faster than, other methods, and reliably provides very accurate solutions. Compared with high-quality implementations of general purpose primal-dual interior-point methods, our method is far faster, especially for large problems. Compared with first-order methods such as coordinate descent methods, our method is comparable in solving large problems with modest accuracy, but is able to solve them with high accuracy with relatively small additional computational cost.

In this paper we focus on methods for *solving* the ℓ_1-regularized LRP; we do not discuss the benefits or advantages of ℓ_1-regularized LR, compared to ℓ_2-regularized LR or other methods for modeling or constructing classifiers for two-class data.

1.4 Regularization Path

Let $(v^\star_\lambda, w^\star_\lambda)$ be a solution for the ℓ_1-regularized LRP with regularization parameter λ. The family of solutions, as λ varies over $(0,\infty)$, is called the (ℓ_1-) *regularization path*. In many applications, the regularization path (or some portion) needs to be computed, in order to determine an appropriate value of λ. At the very least, the ℓ_1-regularized LRP must be solved for multiple, and often many, values of λ.

In ℓ_1-regularized linear regression, which is the problem of minimizing

$$\|Fz - g\|_2^2 + \lambda\|z\|_1$$

over the variable z, where $\lambda > 0$ is the regularization parameter, $F \in \mathbf{R}^{p\times n}$ is the covariate matrix, and $g \in \mathbf{R}^p$ is the vector of responses, it can be shown that the regularization path is piecewise linear, with kinks at each point where any component of the variable z transitions from zero to nonzero, or vice versa. Using this fact, the entire regularization path in a (small or medium size) ℓ_1-regularized linear regression problem can be computed efficiently (Hastie et al., 2004; Efron et al., 2004; Rosset, 2005; Rosset and Zhu, 2007; Osborne et al., 2000). These methods are related to numerical continuation techniques for following piecewise smooth curves, which have been well studied in the optimization literature (Allgower and Georg, 1993).

Path following methods have been applied to several problems (Hastie et al., 2004; Park and Hastie, 2006a,b; Rosset, 2005). Park and Hastie (2006a) describe an algorithm for (approximately) computing the entire regularization path for general linear models (GLMs) including logistic regression models. In Rosset (2005), a general path-following method based on a predictor-corrector method is described for general regularized convex loss minimization problems. Path-following methods can be slow for large-scale problems, where the number of kinks or events is very large (at least n). When the number of kinks on the portion of the regularization path of interest is modest, however, these path-following methods can be very fast, requiring just a small multiple of the effort needed to solve one regularized problem to compute the whole path (or a portion).

In this paper we describe a fast method for computing a large number of points on the regularization path, using a warm-start technique and our interior-point method. Unlike the methods mentioned above, our method does not attempt to track the path exactly (i.e., jumping from kink to kink on the path); it remains efficient even when successive values of λ jump over many kinks. This is essential when computing the regularization path in a large-scale problem. Our method allows us to compute a large number of points (but much fewer than n, when n is very large) on the regularization path, much more efficiently than by solving a family of the problems independently. Our method is far more efficient than path following methods in computing a good approximation of the regularization for a medium-sized or large data set.

1.5 Outline

In Section 2, we give (necessary and sufficient) optimality conditions, and a dual problem, for the ℓ_1-regularized LRP. Using the dual problem, we show how to compute a lower bound on the suboptimality of any pair (v,w). We describe our basic interior-point method in Section 3, and demonstrate its performance in Section 4 with small and medium scale synthetic and machine learning benchmark examples. We show that ℓ_1-regularized LR can be carried out within around 35 or so iterations, where each iteration has the same complexity as solving an ℓ_2-regularized linear regression problem.

In Section 5, we describe a variation on our basic method that uses a preconditioned conjugate gradient approach to compute the search direction. This variation on our method can solve very large problems, with a million features and examples (e.g., the 20 Newsgroups data set), in under an hour, on a PC, provided the data matrix is sufficiently sparse.

In Section 6, we consider the problem of computing the regularization path efficiently, at a variety of values of λ (but potentially far fewer than the number of kinks on the path). Using warm-start techniques, we show how this can done much more efficiently than by solving a family of problems independently. In Section 7, we compare the interior-point method with several existing methods for ℓ_1-regularized logistic regression. In Section 8, we describe generalizations of our method to other ℓ_1-regularized convex loss minimization problems.

2. Optimality Conditions and Dual

In this section we derive a necessary and sufficient optimality condition for the ℓ_1-regularized LRP, as well as a Lagrange dual problem, from which we obtain a lower bound on the objective that we will use in our algorithm.

2.1 Optimality Conditions

The objective function of the ℓ_1-regularized LRP, $l_{\text{avg}}(v,w)+\lambda\|w\|_1$, is convex but not differentiable, so we use a first-order optimality condition based on subdifferential calculus (see Bertsekas, 1999, Prop. B.24 or Borwein and Lewis, 2000, §2). The average logistic loss is differentiable, with

$$\begin{aligned}
\nabla_v l_{\text{avg}}(v,w) &= (1/m)\sum_{i=1}^{m} f'(w^T a_i + v b_i) b_i \\
&= -(1/m)\sum_{i=1}^{m}(1-p_{\text{log}}(v,w)_i) b_i \\
&= -(1/m) b^T(\mathbf{1}-p_{\text{log}}(v,w)),
\end{aligned}$$

and

$$\begin{aligned}\nabla_w l_{\text{avg}}(v,w) &= (1/m)\sum_{i=1}^m f'(w^T a_i + vb_i)a_i \\ &= -(1/m)\sum_{i=1}^m (1-p_{\text{log}}(v,w)_i)a_i \\ &= -(1/m)A^T(\mathbf{1}-p_{\text{log}}(v,w)).\end{aligned}$$

The subdifferential of $\|w\|_1$ is given by

$$(\partial\|w\|_1)_i = \begin{cases} \{1\} & w_i > 0, \\ \{-1\} & w_i < 0, \\ [-1,1] & w_i = 0. \end{cases}$$

The necessary and sufficient condition for (v,w) to be optimal for the ℓ_1-regularized LRP (5) is

$$\nabla_v l_{\text{avg}}(v,w) = 0, \qquad 0 \in \nabla_w l_{\text{avg}}(v,w) + \lambda\partial\|w\|_1,$$

which can be expressed as

$$b^T(\mathbf{1}-p_{\text{log}}(v,w)) = 0, \tag{7}$$

and

$$(1/m)\left(A^T(\mathbf{1}-p_{\text{log}}(v,w))\right)_i \in \begin{cases} \{+\lambda\} & w_i > 0, \\ \{-\lambda\} & w_i < 0, \\ [-\lambda,\lambda] & w_i = 0, \end{cases} \qquad i = 1,\ldots,n. \tag{8}$$

Let us analyze when a pair of the form $(v,0)$ is optimal. This occurs if and only if

$$b^T(\mathbf{1}-p_{\text{log}}(v,0)) = 0, \qquad \|(1/m)A^T(\mathbf{1}-p_{\text{log}}(v,0))\|_\infty \le \lambda.$$

The first condition is equivalent to $v = \log(m_+/m_-)$, where m_+ is the number of training examples with outcome 1 (called positive) and m_- is the number of training examples with outcome -1 (called negative). Using this value of v, the second condition becomes

$$\lambda \ge \lambda_{\max} = \|(1/m)A^T(\mathbf{1}-p_{\text{log}}(\log(m_+/m_-),0))\|_\infty.$$

The number $\lambda_{\max}$ gives us an upper bound on the useful range of the regularization parameter λ: For any larger value of λ, the logistic model obtained from ℓ_1-regularized LR has weight zero (and therefore has no ability to classify). Put another way, for $\lambda \ge \lambda_{\max}$, we get a maximally sparse weight vector, that is, one with $\mathbf{card}(w) = 0$.

We can give a more explicit formula for $\lambda_{\max}$:

$$\lambda_{\max} = (1/m)\left\|\frac{m_-}{m}\sum_{b_i=1} a_i + \frac{m_+}{m}\sum_{b_i=-1} a_i\right\|_\infty = (1/m)\left\|X^T\tilde{b}\right\|_\infty,$$

where

$$\tilde{b}_i = \begin{cases} m_-/m & b_i = 1 \\ -m_+/m & b_i = -1, \end{cases} \quad i = 1,\ldots,m.$$

Thus, $\lambda_{\max}$ is a maximum correlation between the individual features and the (weighted) output vector $\tilde{b}$. When the features have been standardized, we have $\sum_{i=1}^m x_i = 0$, so we get the simplified expression

$$\lambda_{\max} = (1/m)\left\|\sum_{b_i=1} x_i\right\|_\infty = (1/m)\left\|\sum_{b_i=-1} x_i\right\|_\infty.$$

2.2 Dual Problem

To derive a Lagrange dual of the ℓ_1-regularized LRP (5), we first introduce a new variable $z \in \mathbf{R}^m$, as well as new equality constraints $z_i = w^T a_i + vb_i$, $i = 1, \ldots, m$, to obtain the equivalent problem

$$\begin{array}{ll} \text{minimize} & (1/m)\sum_{i=1}^m f(z_i) + \lambda\|w\|_1 \\ \text{subject to} & z_i = w^T a_i + vb_i, \quad i = 1, \ldots, m. \end{array} \tag{9}$$

Associating dual variables $\theta_i \in \mathbf{R}$ with the equality constraints, the Lagrangian is

$$L(v, w, z, \theta) \quad = \quad (1/m)\sum_{i=1}^m f(z_i) + \lambda\|w\|_1 + \theta^T(-z + Aw + bv).$$

The dual function is

$$\begin{aligned} \inf_{v,w,z} L(v, w, z, \theta) &= (1/m)\inf_z \sum_{i=1}^m (f(z_i) - m\theta_i z_i) + \inf_w \left(\lambda\|w\|_1 + \theta^T Aw\right) + \inf_v \theta^T bv \\ &= \begin{cases} -(1/m)\sum_{i=1}^m f^*(-m\theta_i) & \|A^T\theta\|_\infty \le \lambda, \quad b^T\theta = 0, \\ -\infty & \text{otherwise,} \end{cases} \end{aligned}$$

where f^* is the *conjugate* of the logistic loss function f:

$$f^*(y) = \sup_{u \in \mathbf{R}} (yu - f(u)) = \begin{cases} (-y)\log(-y) + (1+y)\log(1+y), & -1 < y < 0 \\ 0 & y = -1 \text{ or } y = 0 \\ \infty, & \text{otherwise.} \end{cases}$$

For general background on convex duality and conjugates, see, for example, Boyd and Vandenberghe (2004, Chap. 5) or Borwein and Lewis (2000).

Thus, we have the following Lagrange dual of the ℓ_1-regularized LRP (5):

$$\begin{array}{ll} \text{maximize} & G(\theta) \\ \text{subject to} & \|A^T\theta\|_\infty \le \lambda, \quad b^T\theta = 0, \end{array} \tag{10}$$

where

$$G(\theta) = -(1/m)\sum_{i=1}^m f^*(-m\theta_i)$$

is the dual objective. The dual problem (10) is a convex optimization problem with variable $\theta \in \mathbf{R}^m$, and has the form of an ℓ_∞-norm constrained maximum generalized entropy problem. We say that $\theta \in \mathbf{R}^m$ is *dual feasible* if it satisfies $\|A^T\theta\|_\infty \le \lambda$, $b^T\theta = 0$.

From standard results in convex optimization we have the following.

- *Weak duality.* Any dual feasible point θ gives a lower bound on the optimal value $p^\star$ of the (primal) ℓ_1-regularized LRP (5):
$$G(\theta) \le p^\star. \tag{11}$$

- *Strong duality.* The ℓ_1-regularized LRP (5) satisfies a variation on Slater's constraint qualification, so there is an optimal solution of the dual (10) $\theta^\star$, which satisfies
$$G(\theta^\star) = p^\star.$$
In other words, the optimal values of the primal (5) and dual (10) are equal.

We can relate a primal optimal point $(v^\star, w^\star)$ and a dual optimal point $\theta^\star$ to the optimality conditions (7) and (8). They are related by

$$\theta^\star = (1/m)(\mathbf{1} - p_{\mathrm{log}}(v^\star, w^\star)).$$

We also note that the dual problem (10) can be derived starting from the equivalent problem (6), by introducing new variables z_i (as we did in (9)), and associating dual variables $\theta_+ \geq 0$ for the inequalities $w \leq u$, and $\theta_- \geq 0$ for the inequalities $-u \leq w$. By identifying $\theta = \theta_+ - \theta_-$ we obtain the dual problem (10).

2.3 Suboptimality Bound

We now derive an easily computed bound on the suboptimality of a pair (v, w), by constructing a dual feasible point $\bar{\theta}$ from an arbitrary w. Define $\bar{v}$ as

$$\bar{v} = \arg\min_v l_{\mathrm{avg}}(v, w), \tag{12}$$

that is, $\bar{v}$ is the optimal intercept for the weight vector w, characterized by $b^T(\mathbf{1} - p_{\mathrm{log}}(\bar{v}, w)) = 0$. Now, we define $\bar{\theta}$ as

$$\bar{\theta} = (s/m)(\mathbf{1} - p_{\mathrm{log}}(\bar{v}, w)), \tag{13}$$

where the scaling constant s is

$$s = \min\left\{m\lambda / \|A^T(\mathbf{1} - p_{\mathrm{log}}(\bar{v}, w))\|_\infty, 1\right\}.$$

Evidently $\bar{\theta}$ is dual feasible, so $G(\bar{\theta})$ is a lower bound on $p^\star$, the optimal value of the ℓ_1-regularized LRP (5).

To compute the lower bound $G(\bar{\theta})$, we first compute $\bar{v}$. This is a one-dimensional smooth convex optimization problem, which can be solved very efficiently, for example, by a bisection method on the optimality condition

$$b^T(\mathbf{1} - p_{\mathrm{log}}(v, w)) = 0,$$

since the lefthand side is a monotone function of v. Newton's method can be used to ensure extremely fast terminal convergence to $\bar{v}$. From $\bar{v}$, we compute $\bar{\theta}$ using (13), and then evaluate the lower bound $G(\bar{\theta})$.

The difference between the primal objective value of (v, w), and the associated lower bound $G(\bar{\theta})$, is called the *duality gap*, and denoted η:

$$\begin{aligned} \eta(v, w) &= l_{\mathrm{avg}}(v, w) + \lambda\|w\|_1 - G(\bar{\theta}) \\ &= (1/m)\textstyle\sum_{i=1}^m \left(f(w^T a_i + v b_i) + f^*(-m\theta_i)\right) + \lambda\|w\|_1. \end{aligned} \tag{14}$$

We always have $\eta \geq 0$; and (by weak duality (11)) the point (v, w) is no more than η-suboptimal. At the optimal point $(v^\star, w^\star)$, we have $\eta = 0$.

3. An Interior-Point Method

In this section we describe an interior-point method for solving the ℓ_1-regularized LRP (5), in the equivalent formulation

$$\begin{array}{ll} \text{minimize} & l_{\mathrm{avg}}(v, w) + \lambda \mathbf{1}^T u \\ \text{subject to} & -u_i \leq w_i \leq u_i, \quad i = 1, \ldots, n, \end{array}$$

with variables $w, u \in \mathbf{R}^n$ and $v \in \mathbf{R}$.

3.1 Logarithmic Barrier and Central Path

The *logarithmic barrier* for the bound constraints $-u_i \leq w_i \leq u_i$ is

$$\Phi(w,u) = -\sum_{i=1}^{n} \log(u_i + w_i) - \sum_{i=1}^{n} \log(u_i - w_i) = -\sum_{i=1}^{n} \log(u_i^2 - w_i^2),$$

with domain

$$\mathbf{dom}\,\Phi = \{(w,u) \in \mathbf{R}^n \times \mathbf{R}^n \mid |w_i| < u_i,\ i = 1,\ldots,n\}.$$

The logarithmic barrier function is smooth and convex. We augment the weighted objective function by the logarithmic barrier, to obtain

$$\phi_t(v,w,u) = t l_{\mathrm{avg}}(v,w) + t\lambda \mathbf{1}^T u + \Phi(w,u),$$

where $t > 0$ is a parameter. This function is smooth, strictly convex, and bounded below, and so has a unique minimizer which we denote $(v^\star(t), w^\star(t), u^\star(t))$. This defines a curve in $\mathbf{R} \times \mathbf{R}^n \times \mathbf{R}^n$, parametrized by t, called the *central path*. (See Boyd and Vandenberghe, 2004, Chap. 11 for more on the central path and its properties.)

With the point $(v^\star(t), w^\star(t), u^\star(t))$ we associate

$$\theta^\star(t) = (1/m)(\mathbf{1} - p_{\log}(v^\star(t), w^\star(t))),$$

which can be shown to be dual feasible. (Indeed, it coincides with the dual feasible point $\bar{\theta}$ constructed from $w^\star(t)$ using the method of Section 2.3.) The associated duality gap satisfies

$$l_{\mathrm{avg}}(v^\star(t), w^\star(t)) + \lambda \|w^\star(t)\|_1 - G(\theta^\star(t)) \leq l_{\mathrm{avg}}(v^\star(t), w^\star(t)) + \lambda \mathbf{1}^T u^\star(t) - G(\theta^\star(t)) = 2n/t.$$

In particular, $(v^\star(t), w^\star(t))$ is no more than $2n/t$-suboptimal, so the central path leads to an optimal solution.

In a primal interior-point method, we compute a sequence of points on the central path, for an increasing sequence of values of t, using Newton's method to minimize $\phi_t(v,w,u)$, starting from the previously computed central point. A typical method uses the sequence $t = t_0, \mu t_0, \mu^2 t_0, \ldots$, where μ is between 2 and 50 (see Boyd and Vandenberghe, 2004, §11.3). The method can be terminated when $2n/t \leq \varepsilon$, since then we can guarantee ε-suboptimality of $(v^\star(t), w^\star(t))$. The reader is referred to Nesterov and Nemirovsky (1994), Wright (1997), and Ye (1997) for more on (primal) interior-point methods.

3.2 A Custom Interior-Point Method

Using our method for cheaply computing a dual feasible point and associated duality gap for *any* (v,w) (and not just for (v,w) on the central path, as in the general case), we can construct a custom interior-point method that updates the parameter t at each iteration.

```
CUSTOM INTERIOR-POINT METHOD FOR $\ell_1$-REGULARIZED LR.
given tolerance $\varepsilon > 0$, line search parameters $\alpha \in (0,1/2)$, $\beta \in (0,1)$
Set initial values. $t := 1/\lambda$, $v := \log(m_+/m_-)$, $w := 0$, $u := \mathbf{1}$.
repeat
```

1. *Compute search direction.*

   Solve the Newton system $\nabla^2\phi_t(v,w,u)\begin{bmatrix}\Delta v\\ \Delta w\\ \Delta u\end{bmatrix} = -\nabla\phi_t(v,w,u).$
2. *Backtracking line search.* Find the smallest integer $k \geq 0$ that satisfies

   $\phi_t(v+\beta^k\Delta v, w+\beta^k\Delta w, u+\beta^k\Delta u) \leq \phi_t(v,w,u)+\alpha\beta^k\nabla\phi_t(v,w,u)^T\begin{bmatrix}\Delta v\\ \Delta w\\ \Delta u\end{bmatrix}.$
3. *Update.* $(v,w,u) := (v,w,u)+\beta^k(\Delta v,\Delta w,\Delta u)$.
4. Set $v := \overline{v}$, the optimal value of the intercept, as in (12).
5. Construct dual feasible point θ from (13).
6. Evaluate duality gap η from (14).
7. **quit** if $\eta \leq \varepsilon$.
8. *Update* t.

This description is complete, except for the rule for updating the parameter t, which will be described below. Our choice of initial values for v, w, u, and t can be explained as follows. The choice $w = 0$ and $u = \mathbf{1}$ seems to work very well, especially when the original data are standardized. The choice $v = \log(m_+/m_-)$ is the optimal value of v when $w = 0$ and $u = \mathbf{1}$, and the choice $t = 1/\lambda$ minimizes $\|(1/t)\nabla\phi_t(\log(m_+/m_-),0,\mathbf{1})\|_2$. (In any case, the choice of the initial values does not greatly affect performance.) The construction of a dual feasible point and duality gap, in steps 4–6, is explained in Section 2.3. Typical values for the line search parameters are $\alpha = 0.01$, $\beta = 0.5$, but here too, these parameter values do not have a large effect on performance. The computational effort per iteration is dominated by step 1, the search direction computation.

There are many possible update rules for the parameter t. In a classical primal barrier method, t is held constant until ϕ_t is (approximately) minimized, that is, $\|\nabla\phi_t\|_2$ is small; when this occurs, t is increased by a factor typically between 2 and 50. More sophisticated update rules can be found in, for example, Nesterov and Nemirovsky (1994), Wright (1997), and Ye (1997).

The update rule we propose is

$$t := \begin{cases} \max\{\mu\min\{\hat{t},t\},t\}, & s \geq s_{\min} \\ t, & s < s_{\min} \end{cases} \tag{15}$$

where $\hat{t} = 2n/\eta$, and $s = \beta^k$ is the step length chosen in the line search. Here $\mu > 1$ and $s_{\min} \in (0,1]$ are algorithm parameters; we have found good performance with $\mu = 2$ and $s_{\min} = 0.5$.

To explain the update rule (15), we first give an interpretation of $\hat{t}$. If (v,w,u) is on the central path, that is, ϕ_t is minimized, the duality gap is $\eta = 2n/t$. Thus $\hat{t}$ is the value of t for which the associated central point has the same duality gap as the current point. Another interpretation is that if t were held constant at $t = \hat{t}$, (v,w,u) would converge to $(v^\star(\hat{t}),w^\star(\hat{t}),u^\star(\hat{t}))$, at which point the duality gap would be exactly η.

We use the step length s as a crude measure of proximity to the central path. When the current point is near the central path, that is, ϕ_t is nearly minimized, we have $s = 1$; far from the central path, we typically have $s \ll 1$. Now we can explain the update rule (15). When the current point is near the central path, as judged by $s \geq s_{\min}$ and $\hat{t} \approx t$, we increase t by a factor μ; otherwise, we keep t at its current value.

We can give an informal justification of convergence of the custom interior-point algorithm. (A formal proof of convergence would be quite long.) Assume that the algorithm does not terminate. Since t never decreases, it either increases without bound, or converges to some value $\bar{t}$. In the first case, the duality gap η converges to zero, so the algorithm must exit. In the second case, the algorithm reduces (roughly) to Newton's method for minimizing $\phi_{\bar{t}}$. This must converge, which means that (v,w,u) converges to $(v^\star(\bar{t}), w^\star(\bar{t}), u^\star(\bar{t}))$. Therefore the duality gap converges to $\bar{\eta} = 2n/\bar{t}$. A basic property of Newton's method is that near the solution, the step length is one. At the limit, we therefore have

$$\bar{t} = \max\{\mu \min\{2n/\bar{\eta}, \bar{t}\}, \bar{t}\} = \mu \bar{t},$$

which is a contradiction since $\mu > 1$.

3.3 Gradient and Hessian

In this section we give explicit formulas for the gradient and Hessian of ϕ_t. The gradient $g = \nabla \phi_t(v,w,u)$ is given by

$$g = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} \in \mathbf{R}^{2n+1},$$

where

$$\begin{aligned} g_1 &= \nabla_v \phi_t(v,w,u) = -(t/m) b^T (\mathbf{1} - p_{\log}(v,w)) \in \mathbf{R}, \\ g_2 &= \nabla_w \phi_t(v,w,u) = -(t/m) A^T (\mathbf{1} - p_{\log}(v,w)) + \begin{bmatrix} 2w_1/(u_1^2 - w_1^2) \\ \vdots \\ 2w_n/(u_n^2 - w_n^2) \end{bmatrix} \in \mathbf{R}^n, \\ g_3 &= \nabla_u \phi_t(v,w,u) = t\lambda \mathbf{1} - \begin{bmatrix} 2u_1/(u_1^2 - w_1^2) \\ \vdots \\ 2u_n/(u_n^2 - w_n^2) \end{bmatrix} \in \mathbf{R}^n. \end{aligned}$$

The Hessian $H = \nabla^2 \phi_t(v,w,u)$ is given by

$$H = \begin{bmatrix} t b^T D_0 b & t b^T D_0 A & 0 \\ t A^T D_0 b & t A^T D_0 A + D_1 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix} \in \mathbf{R}^{(2n+1)\times(2n+1)},$$

where

$$\begin{aligned} D_0 &= (1/m)\,\mathrm{diag}(f''(w^T a_1 + v b_1), \ldots, f''(w^T a_m + v b_m)), \\ D_1 &= \mathrm{diag}\left(2(u_1^2 + w_1^2)/(u_1^2 - w_1^2)^2, \ldots, 2(u_n^2 + w_n^2)/(u_n^2 - w_n^2)^2\right), \\ D_2 &= \mathrm{diag}\left(-4u_1 w_1/(u_1^2 - w_1^2)^2, \ldots, -4u_n w_n/(u_n^2 - w_n^2)^2\right). \end{aligned}$$

Here, we use $\mathrm{diag}(z_1, \ldots, z_m)$ to denote the diagonal matrix with diagonal entries $z_1, \ldots, z_m$, where $z_i \in \mathbf{R}$, $i = 1, \ldots, m$. The Hessian H is symmetric and positive definite.

3.4 Computing the Search Direction

The search direction is defined by the linear equations (Newton system)

$$\begin{bmatrix} tb^T D_0 b & tb^T D_0 A & 0 \\ tA^T D_0 b & tA^T D_0 A + D_1 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta w \\ \Delta u \end{bmatrix} = - \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}.$$

We first eliminate Δu to obtain the reduced Newton system

$$H_{\text{red}} \begin{bmatrix} \Delta v \\ \Delta w \end{bmatrix} = -g_{\text{red}}, \tag{16}$$

where

$$H_{\text{red}} = \begin{bmatrix} tb^T D_0 b & tb^T D_0 A \\ tA^T D_0 b & tA^T D_0 A + D_3 \end{bmatrix}, \quad g_{\text{red}} = \begin{bmatrix} g_1 \\ g_2 - D_2 D_1^{-1} g_3 \end{bmatrix}, \quad D_3 = D_1 - D_2 D_1^{-1} D_2.$$

Once this reduced system is solved, Δu can be recovered as

$$\Delta u = -D_1^{-1}(g_3 + D_2 \Delta w).$$

Several methods can be used to solve the reduced Newton system (16), depending on the relative sizes of n and m and the sparsity of the data A.

3.4.1 More Examples than Features

We first consider the case when $m \geq n$, that is, there are more examples than features. We form H_{red}, at a cost of $O(mn^2)$ flops (floating-point operations), then solve the reduced system (16) by Cholesky factorization of H_{red}, followed by back and forward substitution steps, at a cost of $O(n^3)$ flops. The total cost using this method is $O(mn^2 + n^3)$ flops, which is the same as $O(mn^2)$ when there are more examples than features.

When A is sufficiently sparse, the matrix $tA^T D_0 A + D_3$ is sparse, so H_{red} is sparse, with a dense first row and column. By exploiting sparsity in forming $tA^T D_0 A + D_3$, and using a sparse Cholesky factorization to factor H_{red}, the complexity can be much smaller than $O(mn^2)$ flops (see Boyd and Vandenberghe, 2004, App. C or George and Liu, 1981).

3.4.2 Fewer Examples than Features

When $m \leq n$, that is, there are fewer examples than features, the matrix H_{red} is a diagonal matrix plus a rank $m+1$ matrix, so we can use the Sherman-Morrison-Woodbury formula to solve the reduced Newton system (16) at a cost of $O(m^2 n)$ flops (see Boyd and Vandenberghe, 2004, §4.3). We start by eliminating Δw from (16) to obtain

$$(tb^T D_0 b - t^2 b^T D_0 A S^{-1} A^T D_0 b)\Delta v = -g_1 + tb^T D_0 A S^{-1}(g_2 - D_2 D_1^{-1} g_3),$$

where $S = tA^T D_0 A + D_3$. By the Sherman-Morrison-Woodbury formula (Golub and Van Loan, 1996, p. 50), the inverse of S is given by

$$S^{-1} = D_3^{-1} - D_3^{-1} A^T \left((1/t) D_0^{-1} + A D_3^{-1} A^T \right)^{-1} A D_3^{-1}.$$

We can now calculate Δv via Cholesky factorization of the matrix $\left((1/t)D_0^{-1}+AD_3^{-1}A^T\right)$ and two backsubstitutions (Boyd and Vandenberghe, 2004, App. C). Once we compute Δv, we can compute the other components of the search direction as

$$\begin{aligned} \Delta w &= -S^{-1}(g_2 - D_2D_1^{-1}g_3 + tA^TD_0b\Delta v), \\ \Delta u &= -D_1^{-1}(g_3 + D_2\Delta w). \end{aligned}$$

The total cost of computing the search direction is $O(m^2n)$ flops. We can exploit sparsity in the Cholesky factorization, whenever $(1/t)D_0^{-1}+AD_3^{-1}A^T$ is sufficiently sparse, to reduce the complexity.

3.4.3 SUMMARY

In summary, the number of flops needed to compute the search direction is

$$O(\min(n,m)^2\max(n,m)),$$

using dense matrix methods. If $m \geq n$ and A^TA is sparse, or $m \leq n$ and AA^T is sparse, we can use (direct) sparse matrix methods to compute the search direction with less effort. In each of these cases, the computational effort per iteration of the interior-point method is the same as the effort of solving one ℓ_2-regularized linear regression problem.

4. Numerical Examples

In this section we give some numerical examples to illustrate the performance of the interior-point method described in Section 3, using algorithm parameters

$$\alpha = 0.01, \qquad \beta = 0.5, \qquad s_{\min} = 0.5, \qquad \mu = 2, \qquad \varepsilon = 10^{-8}.$$

(The algorithm performs well for much smaller values of ε, but this accuracy is more than adequate for any practical use.) The algorithm was implemented in both Matlab and C, and run on a 3.2GHz Pentium IV under Linux. The C implementation, which is more efficient than the Matlab implementation (especially for sparse problems), is available online (`www.stanford.edu/~boyd/l1_logreg`).

4.1 Benchmark Problems

The data are four small or medium standard data sets taken from the UCI machine learning benchmark repository (Newman et al., 1998) and other sources. The first data set is leukemia cancer gene expression data (Golub et al., 1999), the second is colon tumor gene expression data (Alon et al., 1999), the third is ionosphere data (Newman et al., 1998), and the fourth is spambase data (Newman et al., 1998).

For each data set, we considered four values of the regularization parameter: $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, $\lambda = 0.05\lambda_{\max}$, and $\lambda = 0.01\lambda_{\max}$. We discarded examples with missing data, and standardized each data set. The dimensions of each problem, along with the number of interior-point method iterations (IP iterations) needed, and the execution time, are given in Table 1. In reporting $\mathrm{card}(w)$, we consider a component w_i to be zero when

$$\left|(1/m)\left(A^T(\mathbf{1}-p_{\log}(v,w))\right)_i\right| \leq \tau\lambda,$$

Data	Features n	Examples m	$\lambda/\lambda_{\max}$	card(w)	IP iterations	Time (sec)
Leukemia (Golub et al., 1999)	7129	38	0.5	6	37	0.60
			0.1	14	38	0.62
			0.05	14	39	0.63
			0.01	18	37	0.60
Colon (Alon et al., 1999)	2000	62	0.5	7	35	0.26
			0.1	22	32	0.25
			0.05	25	33	0.26
			0.01	28	32	0.25
Ionosphere (Newman et al., 1998)	34	351	0.5	3	30	0.02
			0.1	11	29	0.02
			0.05	14	30	0.02
			0.01	24	33	0.03
Spambase (Newman et al., 1998)	57	4061	0.5	8	31	0.63
			0.1	28	32	0.66
			0.05	38	33	0.69
			0.01	52	36	0.75

Table 1: Performance of the interior-point method on 4 data sets, each for 4 values of λ.

where $\tau = 0.9999$. This rule is inspired by the optimality condition in (8).

In all sixteen examples, around 35 iterations were required. We have observed this behavior over a large number of other examples as well. The execution times are well predicted by the complexity order $\min(m,n)^2 \max(m,n)$.

Figure 1 shows the progress of the interior-point method on the four data sets, for the same four values of λ. The vertical axis shows duality gap, and the horizontal axis shows iteration number, which is the natural measure of computational effort when dense linear algebra methods are used. The figures show that the algorithm has linear convergence, with duality gap decreasing by a factor around 1.85 in each iteration.

4.2 Randomly Generated Problems

To examine the effect of problem size on the number of iterations required, we generate 100 random problem instances for each of 20 values of n, ranging from $n = 100$ to $n = 10000$, with $m = 0.1n$, that is, 10 times more features than examples. Each problem has an equal number of positive and negative examples, that is, $m_+ = m_- = m/2$. Features of positive (negative) examples are independent and identically distributed, drawn from a normal distribution $\mathcal{N}(v,1)$, where v is in turn drawn from a uniform distribution on $[0,1]$ ($[-1,0]$).

For each of the 2000 data sets, we solve the ℓ_1-regularized LRP for $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, and $\lambda = 0.05\lambda_{\max}$. The lefthand plot in Figure 2 shows the mean and standard deviation of the number of iterations required to solve the 100 problem instances associated with each value of n and λ. It can be seen that the number of iterations required is very near 35, for all 6000 problem instances.

In the same way, we generate a family of data sets with $m = 10n$, that is, 10 times more examples than features, with 100 problem instances for each of 20 values of n ranging from $n = 10$ to $n = 1000$, and for the same 3 values of λ. The righthand plot in Figure 2 shows the mean and standard deviation

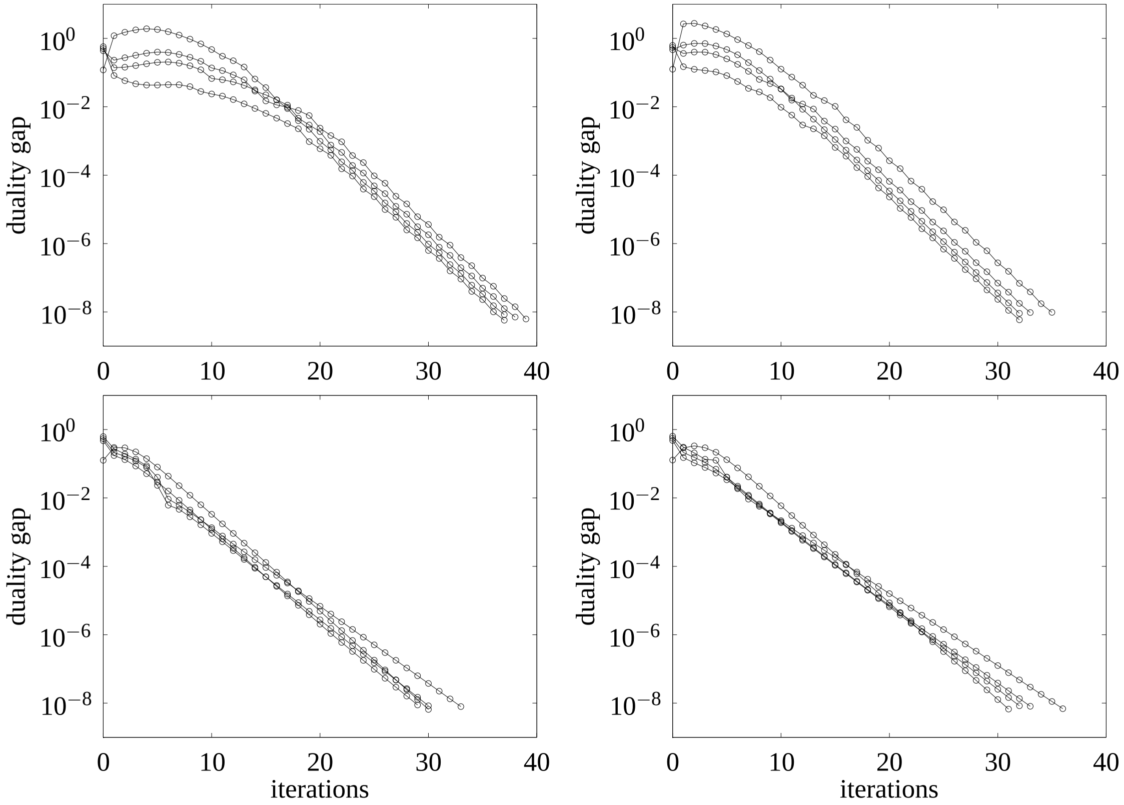

Figure 1: Progress of the interior-point method on 4 data sets, showing duality gap versus iteration number. *Top left*: Leukemia cancer gene data set. *Top right*: Colon tumor gene data set. *Bottom left*: Ionosphere data set. *Bottom right*: Spambase data set.

of the number of iterations required to solve the 100 problem instances associated with each value of n and λ. The results are quite similar to the case with $m = 0.1n$.

5. Truncated Newton Interior-Point Method

In this section we describe a variation on our interior-point method that can handle very large problems, provided the data matrix A is sparse, at the cost of having a run time that is less predictable. The basic idea is to compute the search direction approximately, using a preconditioned conjugate gradients (PCG) method. When the search direction in Newton's method is computed approximately, using an iterative method such as PCG, the overall algorithm is called a *conjugate gradient Newton method*, or a *truncated Newton method* (Ruszczynski, 2006; Dembo and Steihaug, 1983). Truncated Newton methods have been applied to interior-point methods (see, for example, Vandenberghe and Boyd, 1995 and Portugal et al., 2000).

5.1 Preconditioned Conjugate Gradients

The PCG algorithm (Demmel, 1997, §6.6) computes an approximate solution of the linear equations $Hx = -g$, where $H \in \mathbf{R}^{N\times N}$ is symmetric positive definite. It uses a preconditioner $P \in \mathbf{R}^{N\times N}$, also symmetric positive definite.

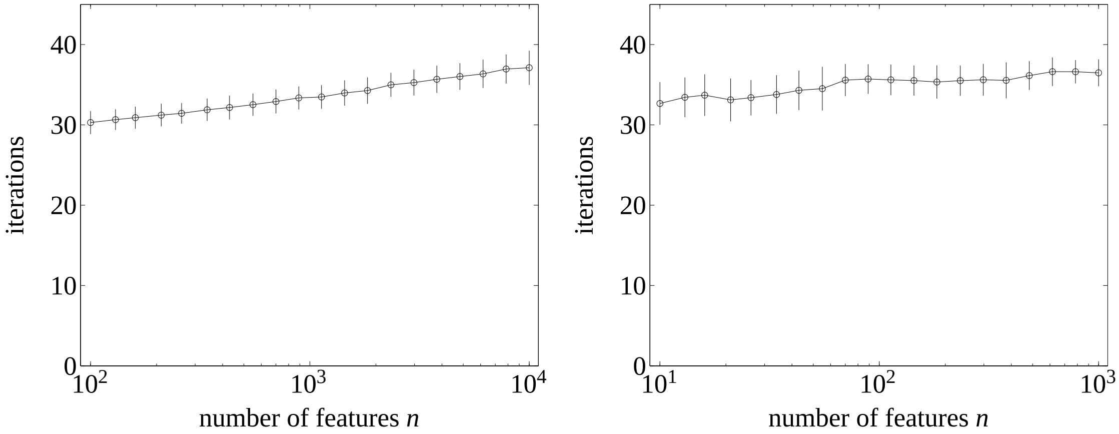

Figure 2: Average number of iterations required to solve 100 randomly generated ℓ_1-regularized LRPs with different problem size and regularization parameter. *Left:* $n = 10m$. *Right:* $n = 0.1m$. Error bars show standard deviation.

PRECONDITIONED CONJUGATE GRADIENTS ALGORITHM

```
given relative tolerance $\varepsilon_{\mathrm{pcg}} > 0$, iteration limit $N_{\mathrm{pcg}}$, and $x_0 \in \mathbf{R}^k$
$k := 0$, $r_0 := Hx_0 - g$, $p_1 := -P^{-1}g$, $y_0 := P^{-1}r_0$.
repeat
    $k := k+1$
    $z := Hp_k$
    $\theta_k := y_{k-1}^T r_{k-1} / p_k^T z$
    $x_k := x_{k-1} + \theta_k p_k$
    $r_k := r_{k-1} - \theta_k z$
    $y_k := P^{-1} r_k$
    $\mu_{k+1} := y_k^T r_k / y_{k-1}^T r_{k-1}$
    $p_{k+1} := y_k + \mu_{k+1} p_k$
until $\|r_k\|_2 / \|g\|_2 \le \varepsilon_{\mathrm{pcg}}$ or $k = N_{\mathrm{pcg}}$.
```

Each iteration of the PCG algorithm involves a handful of inner products, the matrix-vector product Hp_k and a solve step with P in computing $P^{-1}r_k$. With exact arithmetic, and ignoring the stopping condition, the PCG algorithm is guaranteed to compute the exact solution $x = -H^{-1}g$ in N steps. When $P^{-1/2}HP^{-1/2}$ is well conditioned, or has just a few extreme eigenvalues, the PCG algorithm can compute an approximate solution in a number of steps that can be far smaller than N. Since $P^{-1}r_k$ is computed in each step, we need this computation to be efficient.

5.2 Truncated Newton Interior-Point Method

The truncated Newton interior-point method is the same as the interior-point algorithm described in Section 3, with the search direction computed using the PCG algorithm.

We can compute Hp_k in the PCG algorithm using

$$\begin{aligned} Hp_k &= \begin{bmatrix} tb^T D_0 b & tb^T D_0 A & 0 \\ tA^T D_0 b & tA^T D_0 A + D_1 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix} \begin{bmatrix} p_{k1} \\ p_{k2} \\ p_{k3} \end{bmatrix} \\ &= \begin{bmatrix} b^T u \\ A^T u + D_1 p_{k2} \\ D_2 p_{k2} + D_1 p_{k3} \end{bmatrix}, \end{aligned}$$

where $u = tD_0(bp_{k1} + Ap_{k2}) \in \mathbf{R}^m$. The cost of computing Hp_k is $O(p)$ flops when A is sparse with p nonzero elements. (We assume $p \geq n$, which holds if each example has at least one nonzero feature.)

We now describe a simple choice for the preconditioner P. The Hessian can be written as

$$H = t\nabla^2 l_{\text{avg}}(v,w) + \nabla^2 \Phi(w,u).$$

To obtain the preconditioner, we replace the first term with its diagonal part, to get

$$P = \operatorname{diag}\left(t\nabla^2 l_{\text{avg}}(v,w)\right) + \nabla^2 \Phi(w,u) = \begin{bmatrix} d_0 & 0 & 0 \\ 0 & D_3 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix}, \tag{17}$$

where

$$d_0 = tb^T D_0 b, \qquad D_3 = \operatorname{diag}(tA^T D_0 A) + D_1.$$

(Here $\operatorname{diag}(S)$ is the diagonal matrix obtained by setting the off-diagonal entries of the matrix S to zero.) This preconditioner approximates the Hessian of tl_{avg} with its diagonal entries, while retaining the Hessian of the logarithmic barrier. For this preconditioner, $P^{-1}r_k$ can be computed cheaply as

$$\begin{aligned} P^{-1}r_k &= \begin{bmatrix} d_0 & 0 & 0 \\ 0 & D_3 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix}^{-1} \begin{bmatrix} r_{k1} \\ r_{k2} \\ r_{k3} \end{bmatrix} \\ &= \begin{bmatrix} r_{k1}/d_0 \\ (D_1 D_3 - D_2^2)^{-1}(D_1 r_{k2} - D_2 r_{k3}) \\ (D_1 D_3 - D_2^2)^{-1}(-D_2 r_{k2} + D_3 r_{k3}) \end{bmatrix}, \end{aligned}$$

which requires $O(n)$ flops.

We can now explain how *implicit standardization* can be carried out. When using standardized data, we work with the matrix A^{std} defined in (20), instead of A. As mentioned in Appendix A, A^{std} is in general dense, so we should not form the matrix. In the truncated Newton interior-point method we do not need to form the matrix A^{std}; we only need a method for multiplying a vector by A^{std} and a method for multiplying a vector by $A^{\text{std}\,T}$. But this is easily done efficiently, using the fact that A^{std} is a sparse matrix (i.e., A) times a diagonal matrix, plus a rank-one matrix; see (20) in Appendix A.

There are several good choices for the initial point in the PCG algorithm (labeled x_0 in Section 5.1), such as the negative gradient, or the previous search direction. We have found good performance with both, with a small advantage in using the previous search direction.

The PCG relative tolerance parameter ϵ_{pcg} has to be carefully chosen to obtain good efficiency in a truncated Newton method. If the tolerance is too small, too many PCG steps are needed to compute each search direction; if the tolerance is too high, then the computed search directions do not give adequate reduction in duality gap per iteration. We experimented with several methods of adjusting the PCG relative tolerance, and found good results with the adaptive rule

$$\epsilon_{\mathrm{pcg}} = \min\{0.1, \xi\eta/\|g\|_2\}, \tag{18}$$

where g is the gradient and η is the duality gap at the current iterate. Here, ξ is an algorithm parameter. We have found that $\xi = 0.3$ works well for a wide range of problems. In other words, we solve the Newton system with low accuracy (but never worse than 10%) at early iterations, and solve it more accurately as the duality gap decreases. This adaptive rule is similar in spirit to standard methods used in inexact and truncated Newton methods (see Nocedal and Wright, 1999).

The computational effort of the truncated Newton interior-point algorithm is the product of s, the total number of PCG steps required over all iterations, and the cost of a PCG step, which is $O(p)$, where p is the number of nonzero entries in A, that is, the total number of (nonzero) features appearing in all examples. In extensive testing, we found the truncated Newton interior-point method to be very efficient, requiring a total number of PCG steps ranging between a few hundred (for medium size problems) and several thousand (for large problems). For medium size (and sparse) problems it was faster than the basic interior-point method; moreover the truncated Newton interior-point method was able to solve very large problems, for which forming the Hessian H (let alone computing the search direction) would be prohibitively expensive.

While the total number of iterations in the basic interior-point method is around 35, and nearly independent of the problem size and problem data, the total number of PCG iterations required by the truncated Newton interior-point method can vary significantly with problem data and the value of the regularization parameter λ. In particular, for small values of λ (which lead to large values of $\mathrm{card}(w)$), the truncated Newton interior-point method requires a larger total number of PCG steps. Algorithm performance that depends substantially on problem data, as well as problem dimension, is typical of all iterative (i.e., non direct) methods, and is the price paid for the ability to solve very large problems.

5.3 Numerical Examples

In this section we give some numerical examples to illustrate the performance of the truncated Newton interior-point method. We use the same algorithm parameters for line search, update rule, and stopping criterion as those used in Section 4, and the PCG tolerance given in (18) with $\xi = 0.3$. We chose the parameter N_{pcg} to be large enough (5000) that the iteration limit was never reached in our experiments; the typical number of PCG iterations was far smaller. The algorithm is implemented in both Matlab and C, on a 3.2GHz Pentium IV running Linux, except for very large problems. For very large problems whose data could not be handled on this computer, the method was run on AMD Opteron 254 with 8GB main memory. The C implementation is available online at `www.stanford.edu/~boyd/l1_logreg`.

5.3.1 A Medium Sparse Problem

We consider the Internet advertisements data set (Newman et al., 1998) with 1430 features and 2359 examples (discarding examples with missing data). The total number of nonzero entries in the data

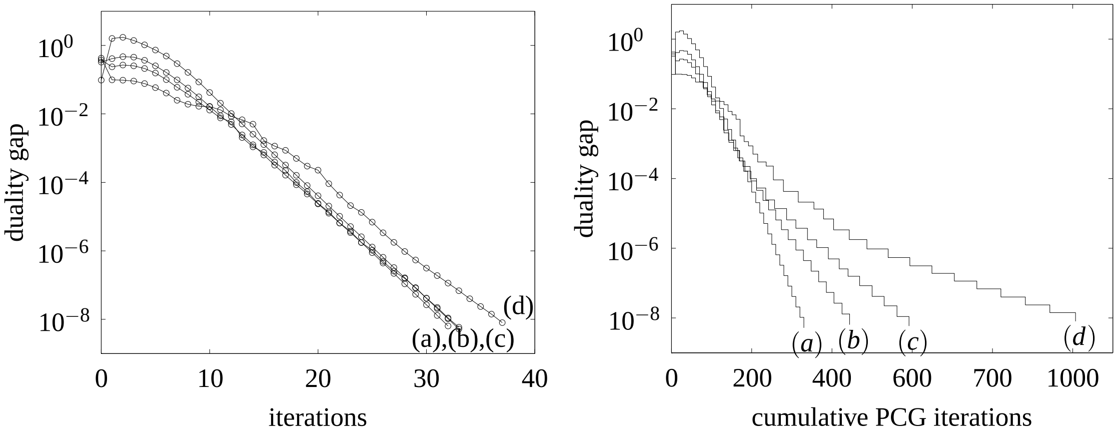

Figure 3: Progress of the truncated Newton interior-point method on the Internet advertisements data set with four regularization parameters: (a) $\lambda = 0.5\lambda_{\max}$, (b) $\lambda = 0.1\lambda_{\max}$, (c) $\lambda = 0.05\lambda_{\max}$, and (d) $\lambda = 0.01\lambda_{\max}$.

matrix A is $p = 39011$. We standardized the data set using implicit standardization, as explained in Section 5.2, solving four ℓ_1-regularized LRPs, with $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, $\lambda = 0.05\lambda_{\max}$, and $\lambda = 0.01\lambda_{\max}$. Figure 3 shows the convergence behavior. The lefthand plot shows the duality gap versus outer iterations; the righthand plot shows duality gap versus cumulative PCG iterations, which is the more accurate measure of computational effort.

The lefthand plot shows that the number of Newton iterations required to solve the problem is not much more than in the basic interior-point method described in Section 3. The righthand plot shows that the total number of PCG steps is several hundred, and depends substantially on the value of λ. Thus, the search directions are computed using on the order of ten PCG iterations.

To give a very rough comparison with the direct method applied to this sparse problem, the truncated Newton interior-point method is much more efficient than the basic interior-point method that does not exploit the sparsity of the data. It is comparable to or faster than the basic interior-point method that uses sparse linear algebra methods, when the regularization parameter is not too small.

5.3.2 A LARGE SPARSE PROBLEM

Our next example uses the 20 Newsgroups data set (Lang, 1995). We processed the data set in a way similar to Keerthi and DeCoste (2005). The positive class consists of the 10 groups with names of form sci.*, comp.*, and misc.forsale, and the negative class consists of the other 10 groups. We used McCallum's Rainbow program (McCallum, 1996) with the command

```
rainbow -g 3 -h -s -O 2 -i
```

to tokenize the (text) data set. These options specify trigrams, skip message headers, no stoplist, and drop terms occurring fewer than two times. The resulting data set has $n = 777811$ features (trigrams) and $m = 11314$ examples (articles). Each example contains an average of 425 nonzero features. The total number of nonzero entries in the data matrix A is $p = 4802169$. We standardized the data set using implicit standardization, as explained in Section 5.2, solving three ℓ_1-regularized LRPs, with $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, and $\lambda = 0.05\lambda_{\max}$. (For the value $\lambda = 0.01\lambda_{\max}$, the runtime is on the order of one hour. This case is not of practical interest, and so not reported here, since

$\lambda/\lambda_{\max}$	card(w)	Iterations	PCG iterations	Time (sec)
0.5	9	43	558	134
0.1	544	60	1036	256
0.05	2531	58	2090	501

Table 2: Performance of truncated Newton interior-point method on the 20 newsgroup data set ($n = 777811$ features, $m = 11314$ examples) for 3 values of λ.

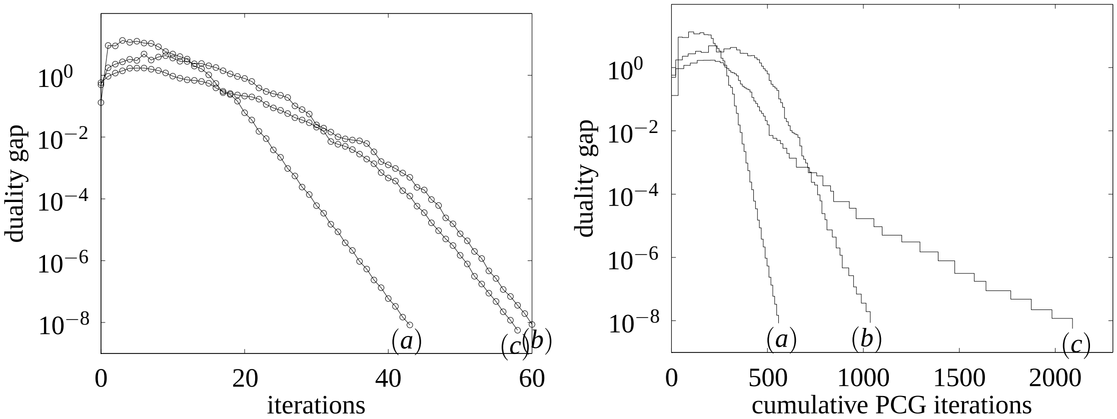

Figure 4: Progress of the truncated Newton interior-point method on the 20 Newsgroups data set for (a) $\lambda = 0.5\lambda_{\max}$, (b) $\lambda = 0.1\lambda_{\max}$, and (c) $\lambda = 0.05\lambda_{\max}$. *Left.* Duality gap versus iterations. *Right.* Duality gap versus cumulative PCG iterations.

the cardinality of the optimal solution is around 10000 and comparable to the number of examples.) The performance of the algorithm, and the cardinality of the weight vectors, is given in Table 2. Figure 4 shows the progress of the algorithm, with duality gap versus iteration (lefthand plot), and duality gap versus cumulative PCG iteration (righthand plot).

The number of iterations required to solve the problems ranges between 43 and 60, depending on λ. The more relevant measure of computational effort is the total number of PCG iterations, which ranges between around 500 and 2000, again, increasing with decreasing λ, which corresponds to increasing card(w). The average number of PCG iterations, per iteration of the truncated Newton interior-point method, is around 13 for $\lambda = 0.5\lambda_{\max}$, 17 for $\lambda = 0.1\lambda_{\max}$, and 36 for $\lambda = 0.05\lambda_{\max}$. (The variance in the number of PCG iterations required per iteration, however, is large.) The running time is consistent with a cost of around 0.24 seconds per PCG iteration. The increase in running time, for decreasing λ, is due primarily to an increase in the average number of PCG iterations required per iteration, but also from an increase in the overall number of iterations required.

5.3.3 RANDOMLY GENERATED PROBLEMS

We generated a family of 21 data sets, with the number of features n varying from one hundred to ten million, and $m = 0.1n$ examples. The data were generated using the same general method described in Section 4.2, but with A sparse, with an average number of nonzero features per example around 30. Thus, the total number of nonzero entries in A is $p \approx 30m$. We standardized the data

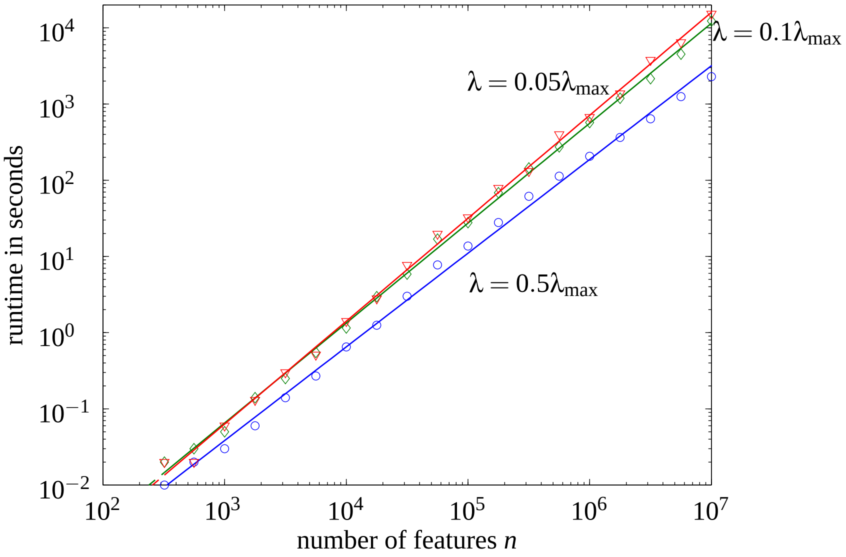

Figure 5: Runtime of the truncated Newton interior-point method, for randomly generated sparse problems, with three values of λ.

set using implicit standardization, as explained in Section 5.2, solving each problem instance for the three values $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, and $\lambda = 0.05\lambda_{\max}$. The total runtime, for the 63 ℓ_1-regularized LRPs, is shown in Figure 5. The plot shows that runtime increases as λ decreases, and grows approximately linearly with problem size.

We compare the runtimes of the truncated Newton interior-point and the basic interior-point method using dense linear algebra methods to compute the search direction. Figure 6 shows the results for $\lambda = 0.1\lambda_{\max}$. The truncated Newton interior-point method is far more efficient for medium problems. For large problems, the basic interior-point method fails due to memory limitations, or extremely long computation times.

By fitting an exponent to the data over the range from $n = 320$ to the largest problem successfully solved by each method, we find that the basic interior-point method scales as $O(n^{2.8})$ (which is consistent with the basic flop count analysis, which predicts $O(n^3)$). For the truncated Newton interior-point method, the empirical complexity is $O(n^{1.3})$.

When sparse matrix methods are used to compute the search direction in the basic interior-point method, we get an empirical complexity of $O(n^{2.2})$ for the Matlab implementation of the basic interior-point method that uses sparse matrix methods, showing a good efficiency gain over dense methods, for medium scale problems. The C implementation would have the same empirical complexity as the Matlab one with a smaller constant hidden in the $O(\cdot)$ notation.

5.3.4 PRECONDITIONER PERFORMANCE

To examine the effect of the preconditioner (17) on the efficiency of the approximate search direction computation, we compare the eigenvalue distributions of the Hessian H and the preconditioned Hessian $P^{-1/2}HP^{-1/2}$, for the colon gene tumor problem ($n = 2000$ features, $m = 62$ examples) at the 15th iterate, in Figure 7. The eigenvalues of the preconditioned Hessian are tightly clustered,

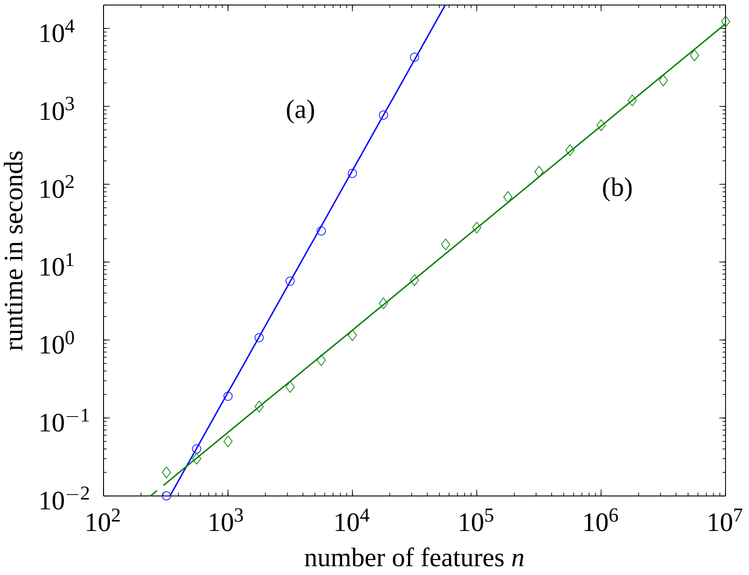

Figure 6: Runtime of (a) the basic interior-point method and (b) the truncated Newton interior-point method, for a family of randomly generated sparse problems.

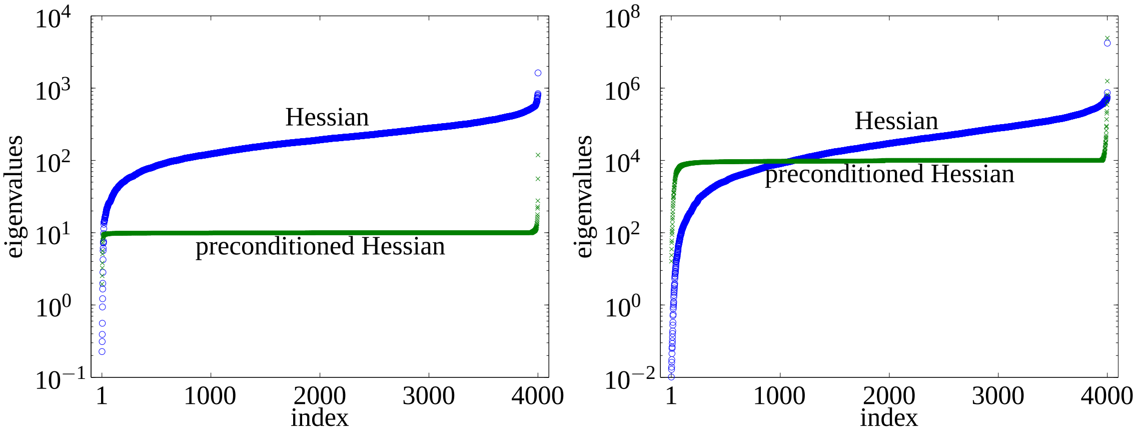

Figure 7: Eigenvalue distributions of Hessian and preconditionned Hessian, at the 15th iterate, for the colon gene tumor problem, for $\lambda = 0.5\lambda_{\max}$ (left) and $\lambda = 0.05\lambda_{\max}$ (right).

with just a few extreme eigenvalues, which explains the good performance with relatively few PCG iterations per iteration (Demmel, 1997, §6.6).

6. Computing the Regularization Path

In this section we consider the problem of solving the ℓ_1-regularized LRP for M values of the regularization parameter λ,

$$\lambda_{\max} = \lambda_1 > \lambda_2 > \cdots > \lambda_M > 0.$$

This can be done by applying the methods described above, for each of the M problems. This is called a *cold-start* approach, since each problem is solved independently of the others. This is efficient when multiple processors are used, since the LRPs can be solved simultaneously, on different processors. But when one processor is used, we can solve these M problems much more efficiently by solving them sequentially, using the previously computed solution as a starting point for the next computation. This is called a *warm-start* approach.

We first note that the solution for $\lambda = \lambda_1 = \lambda_{\max}$ is $(\log(m_+/m_-), 0, 0)$. Since this point does not satisfy $|w_i| < u_i$, it cannot be used to initialize the computation for $\lambda = \lambda_2$. We modify it by adding a small increment to u to get

$$(v^{(1)}, w^{(1)}, u^{(1)}) = (\log(m_+/m_-), 0, (\varepsilon_{\mathrm{abs}}/(n\lambda))\mathbf{1}),$$

which is strictly feasible. In fact, it is on the central path with parameter $t = 2n/\varepsilon_{\mathrm{abs}}$, and so is $\varepsilon_{\mathrm{abs}}$-suboptimal. Note that so far we have expended no computational effort.

Now for $k = 2, \ldots, M$ we compute the solution $(v^{(k)}, w^{(k)}, u^{(k)})$ of the problem with $\lambda = \lambda_k$, by applying the interior-point method, with starting point modified to be

$$(v_{\mathrm{init}}, w_{\mathrm{init}}, u_{\mathrm{init}}) = (v^{(k-1)}, w^{(k-1)}, u^{(k-1)}),$$

and initial value of t set to $t = 2n/\varepsilon_{\mathrm{abs}}$.

In the warm-start technique described above, the number of grid points, M, is fixed in advance. The grid points (and M) can be chosen adaptively on the fly, while taking into account the curvature of the regularization path trajectories, as described in Park and Hastie (2006a).

6.1 Numerical Results

Our first example is the leukemia cancer gene expression data, for $M = 100$ values of λ, uniformly distributed on a logarithmic scale over the interval $[0.001\lambda_{\max}, \lambda_{\max}]$. (For this example, $\lambda_{\max} = 0.37$.) The left plot in Figure 8 shows the regularization path, that is, $w^{(k)}$, versus regularization parameter λ. The right plot shows the number of iterations required to solve each problem from a warm-start, and from a cold-start.

The number of cold-start iterations required is always near 36, while the number of warm-start iterations varies, but is always smaller, and typically much smaller, with an average value of 3.1. Thus the computational savings for this example is over 11 : 1.

Our second example is the 20 Newsgroups data set, with $M = 100$ values of λ uniformly spaced on a logarithmic scale over $[0.05\lambda_{\max}, \lambda_{\max}]$. For this problem we have $\lambda_{\max} = 0.12$. The top plot in Figure 9 shows the regularization path. The bottom left plot shows the total number of PCG iterations required to solve each problem, with the warm-start and cold-start methods. The bottom right plot shows the cardinality of w as a function of λ.

Here too the warm-start method gives a substantial advantage over the cold-start method, at least for λ not too small, that is, as long as the optimal weight vector is relatively sparse. The total runtime using the warm-start method is around 2.8 hours, and the total runtime using the cold-start method is around 6.2 hours, so the warm-start methods gives a savings of around 2 : 1. If we consider only the range from $0.1\lambda_{\max}$ to $\lambda_{\max}$, the savings increases to 5 : 1.

We note that for this example, the number of events (i.e., a weight transitioning between zero and nonzero) along the regularization path is very large, so methods that attempt to track every event will be very slow.

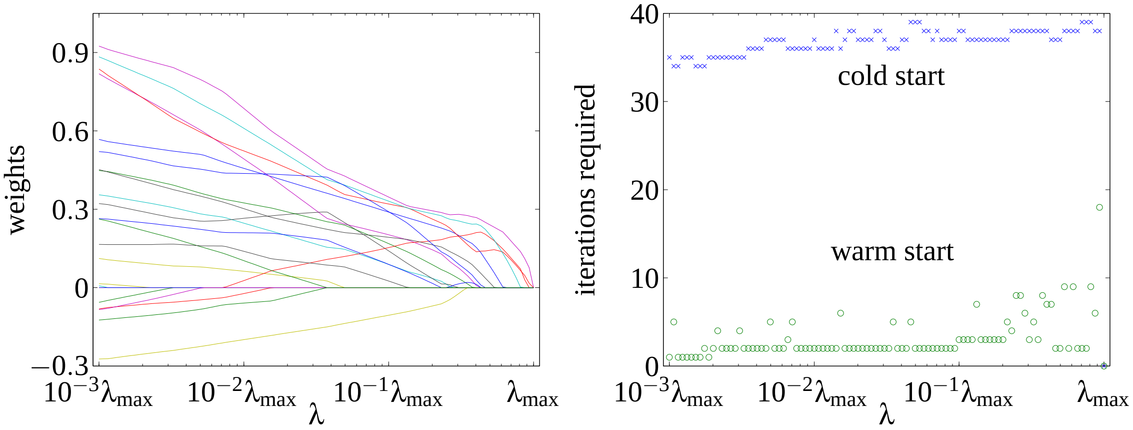

Figure 8: *Left.* Regularization path for leukemia cancer gene expression data. *Right.* Iterations required for cold-start and warm-start methods.

7. Comparison

In this section we compare the performance of our basic and truncated Newton interior-point methods, implemented in C (called `l1_logreg`), with several existing methods for ℓ_1-regularized logistic regression, We make comparisons with `MOSEK` (MOSEK ApS, 2002), `IRLS-LARS` (Lee et al., 2006), `BBR` (Genkin et al., 2006), and `glmpath` (Park and Hastie, 2006a).

`MOSEK` is a general purpose primal-dual interior-point solver, which is known to be quite efficient compared to other standard solvers. `MOSEK` can solve ℓ_1-regularized LRPs using the separable convex formulation (9), or by treating the problem as a geometric program (GP) (see Boyd et al., 2006). We used both formulations and report the better results here in each case. `MOSEK` uses a stopping criterion based on the duality gap, like our method.

`IRLS-LARS` alternates between approximating the average logistic loss by a quadratic approximation at the current iterate, and solving the resulting ℓ_1-regularized least squares problem using the LARS method (Efron et al., 2004) to update the iterate. `IRLS-LARS` outperforms many existing methods for ℓ_1-regularized logistic regression including `GenLASSO` (Roth, 2004), `SCGIS` (Goodman, 2004), `Gl1ce` (Lokhorst, 1999), and `Grafting` (Perkins and Theiler, 2003). `IRLS-LARS` used in our comparison is implemented in Matlab and C, with the LARS portion implemented in C. The hybrid implementation is called `IRLS-LARS-MC` and available from `http://ai.stanford.edu/~silee/softwares/irlslars.htm`. We ran it until the primal objective is within tolerance from the optimal objective value, which is computed using `l1_logreg`, with small (10^{-12}) duality gap.

`BBR`, implemented in C, uses the cyclic coordinate descent method for Bayesian logistic regression. The C implementation is available from `http://www.stat.rutgers.edu/~madigan/BBR/`. The stopping criterion is based on lack of progress, and not on a suboptimality bound or duality gap. Tightening the tolerance for `BBR` greatly increased its running time, and only had a minor effect on the final accuracy.

`glmpath` uses a path-following method for generalized linear models, including logistic models, and computes a portion of the regularization path. It is implemented in the R environment and available from `http://cran.r-project.org/src/contrib/Descriptions/glmpath.html`. We compared `glmpath` to our warm-start method, described in Section 6.

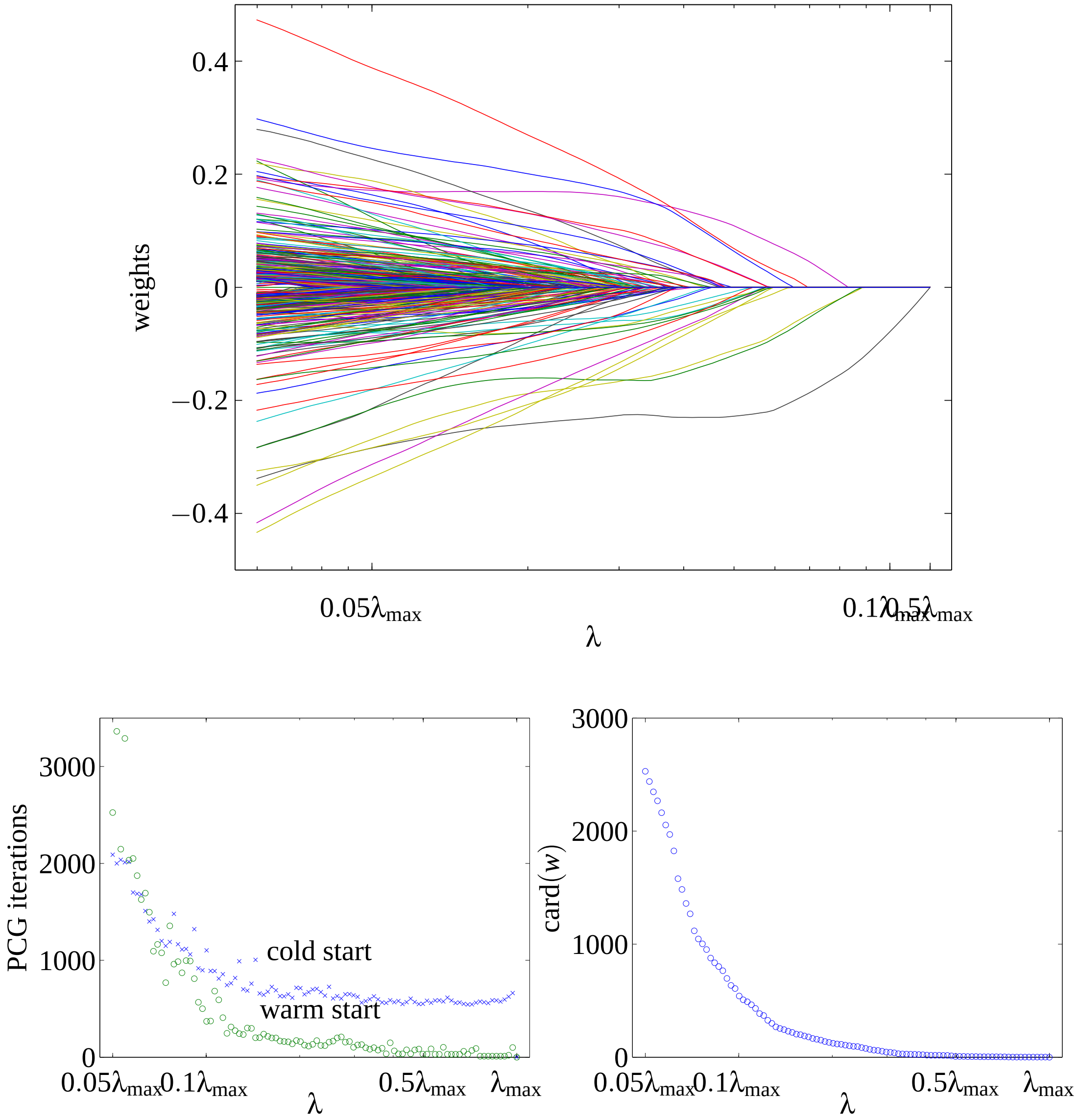

Figure 9: *Top*. Regularization path for 20 newsgroup data. *Bottom left*. Total PCG iterations required by cold-start and warm-start methods. *Bottom right*. card(w) versus λ.

We report the run times of the methods, using the different stopping criteria described above. We also report the actual accuracy achieved, that is, the difference between the achieved primal objective and the optimal objective value (as computed by l1_logreg with duality gap 10^{-12}). However, it is important to point out that it is very difficult, if not impossible, to carry out a fair comparison of solution methods, due to the issue of implementation (which can have a great influence on the algorithm performance), the choice of algorithm parameters, and the different stopping criteria. Therefore, the comparison results reported below should be interpreted with caution.

We report comparison results using four data sets: the leukemia cancer gene expression and spambase data sets, two dense benchmark data sets used in Section 4.1, the Internet advertisements data set, the medium sparse data set used in Section 5.3, and the 20 Newsgroups data set, the large sparse data set used in Section 5.3. When the large 20 Newsgroups data set was standardized, the three existing solvers could not handle a data set of this size, so it was not standardized. The solvers could handle the standardized Internet advertisements data set but do not exploit the sparse plus rank-one structure of the standardized data matrix. Therefore, the Internet advertisements data set was not standardized as well. For small problems (leukemia and spambase), the solvers were all run on a 3.2GHz Pentium IV under Linux; for medium and large problem (Internet advertisements and 20 Newsgroups), the solvers were run on an AMD Opteron 254 (with 8GB RAM) under Linux.

The regularization parameter λ can strongly affect the runtime of the methods, including ours. For each data set, we considered many values of the regularization parameter λ over the interval $[0.0001\lambda_{\max}, \lambda_{\max}]$ (which appears to cover the range of interest for many standardized problems). We report here the results with $\lambda = 0.001\lambda_{\max}$ for the two small benchmark data. The cardinality of the optimal solution is 21 for the leukemia data and 54 for the spambase data, larger than half of the minimum of the number of features and the number of examples. For the two unstandardized problems, ℓ_1-regularized LR with a regularization parameter in the interval $[0.0001\lambda_{\max}, \lambda_{\max}]$ yields a very sparse solution. We used $\lambda = 0.0001\lambda_{\max}$ for the Internet advertisements and $\lambda = 0.001\lambda_{\max}$ for the 20 Newsgroups data. The cardinalities of optimal solutions are relatively very small (19 for the Internet advertisements and 247 for the 20 Newsgroups) compared with the sizes of the problems.

Table 3 summarizes the comparison results for the four problems. Here, the tolerance has a different meaning depending on the method, as described above. As shown in this table, our method is as fast as, or faster than, existing methods for the small two data sets, but the discrepancy in performance is not significant, since the problems are small. For the unstandardized Internet advertisements data set, our method is most efficient. MOSEK could not handle the unstandardized large 20 Newsgroups data set, so we compared l1_logreg with BBR and IRLS-LARS-MC on the unstandardized 20 Newsgroups data set, for the regularization parameter $\lambda = 0.001\lambda_{\max}$. The truncated Newton interior-point method solves the problem to the accuracy 2×10^{-6} in around 100 seconds and to a higher accuracy with relatively small additional run time; BBR solves them to this ultimate accuracy in a comparable time, but slows down when attempting to compute a solution with higher accuracy.

Finally, we compare the runtimes of the warm-start method and glmpath. We consider the leukemia cancer gene expression data set and the Internet advertisements data set as benchmark examples of small dense and medium sparse problems, respectively. For each data set, the warm-start method finds M points ($w^{(k)},\ k = 1, \ldots, M$) on the regularization path, with λ uniformly spaced on a logarithmic scale over the interval $[0.001\lambda_{\max}, \lambda_{\max}]$. glmpath finds an approximation of the regularization path by choosing the kink points adaptively over the same interval. The results are

Program	Tol.	Leukemia		Spambase		Internet adv.		Newsgroups	
		time	accuracy	time	accuracy	time	accuracy	time	accuracy
l1_logreg	10^{-4}	0.37	2×10^{-6}	0.34	2×10^{-5}	0.14	4×10^{-5}	90	2×10^{-6}
	10^{-8}	0.57	1×10^{-10}	0.74	1×10^{-9}	0.27	5×10^{-9}	140	2×10^{-10}
IRLS-LARS-MC	10^{-4}	0.75	6×10^{-6}	0.29	9×10^{-6}	1.2	1×10^{-5}	450	8×10^{-5}
	10^{-8}	0.81	6×10^{-11}	0.37	3×10^{-11}	2.5	1×10^{-10}	1200	7×10^{-9}
MOSEK	10^{-4}	9	8×10^{-6}	10	8×10^{-8}	1.3	3×10^{-9}	-	-
	10^{-8}	10	3×10^{-9}	11	5×10^{-13}	1.4	4×10^{-13}	-	-
BBR	10^{-4}	15	1×10^{-7}	39	3×10^{-6}	0.44	1×10^{-7}	140	2×10^{-6}
	10^{-11}	73	4×10^{-8}	300	1×10^{-11}	1.1	1×10^{-12}	850	1×10^{-6}

Table 3: Comparison results with two standardized data sets (leukemia and spambase) and two unstandardized data sets (Internet advertisements and 20 Newsgroups). The regularization parameter is taken as $\lambda = 0.001\lambda_{\max}$ for leukemia and spambase, $\lambda = 0.0001\lambda_{\max}$ for Internet advertisements and $\lambda = 0.001\lambda_{\max}$ for 20 Newsgroups.

Data	warm-start ($M = 25$)	warm-start ($M = 100$)	glmpath
Leukemia	2.8	6.4	1.9
Internet	5.2	13	940

Table 4: Regularization path computation time (in seconds) of the warm-start method and glmpath for standardized leukemia cancer gene expression data and Internet advertisements data.

shown in Table 4. For the small leukemia data set, glmpath is faster than the warm-start method. The warm-start method is more efficient than glmpath for the Internet advertisements data set, a medium-sized sparse problem. The performance discrepancy is partially explained by the fact that our warm-start method exploits the sparse plus rank-one structure of the standardized data matrix, whereas glmpath does not.

8. Extensions and Variations

The basic interior-point method and the truncated Newton variation can be extended to general ℓ_1-regularized convex loss minimization problems, with twice differentiable loss functions, that have the form

$$\begin{array}{ll} \text{minimize} & (1/m)\sum_{i=1}^{m}\phi(z_i)+\lambda\|w\|_1 \\ \text{subject to} & z_i = w^T a_i + v b_i + c_i, \quad i=1,\ldots,m, \end{array} \tag{19}$$

with variables are $v \in \mathbf{R}$, $w \in \mathbf{R}^n$, and $z \in \mathbf{R}^m$, and problem data $c_i \in \mathbf{R}$, $a_i \in \mathbf{R}^n$, and $b_i \in \mathbf{R}$ determined by a set of given (observed or training) examples

$$(x_i, y_i) \in \mathbf{R}^n \times \mathbf{R}, \quad i = 1,\ldots,m.$$

Here $\phi : \mathbf{R} \to \mathbf{R}$ is a loss function which is convex and twice differentiable. Prior work related to the extension includes Park and Hastie (2006a), Rosset (2005), and Tibshirani (1997).

In ℓ_1-regularized (binary) classification, we have $y_i \in \{-1,+1\}$ (binary labels), and z_i has the form $z_i = y_i(w^T x_i + v)$, so we have the form (19) with $a_i = y_i x_i$, $b_i = y_i$, and $c_i = 0$. The associated classifier is given by $y = \operatorname{sgn}(w^T x + v)$. The loss function ϕ is small for positive arguments, and grows for negative arguments. When ϕ is the logistic loss function f in (1), this general ℓ_1-regularized classification problem reduces to the ℓ_1-regularized LRP. When ϕ is the convex loss function $\phi(u) = -\log \Phi(u)$, where Φ is the cumulative distribution function of the standard normal distribution, this ℓ_1-regularized classification problem reduces to the ℓ_1-regularized probit regression problem. More generally, ℓ_1-regularized estimation problems that arise in generalized linear models (McCullagh and Nelder, 1989; Hastie et al., 2001) for binary response variables (which include logistic and probit models) can be formulated problems of the form (19); see Park and Hastie (2006a) for the precise formulation.

In ℓ_1-regularized linear regression, we have $y_i \in \mathbf{R}$, and z_i has the form $z_i = w^T x_i + v - y_i$, which is the difference between y_i and its predicted value, $w^T x_i + v$. Thus ℓ_1-regularized regression problems have the form (19) with $a_i = x_i$, $b_i = 1$, and $c_i = -y_i$. Typically ϕ is symmetric, with $\phi(0) = 0$. When the loss function is quadratic, that is, $\phi(u) = u^2$, the convex loss minimization problem (19) is the ℓ_1-regularized least squares regression problem studied extensively in the literature.

The dual of the ℓ_1-regularized convex loss minimization problem (19) is

$$\begin{array}{ll} \text{maximize} & -(1/m)\sum_{i=1}^m \phi^*(-m\theta_i) + \theta^T c \\ \text{subject to} & \|A^T\theta\|_\infty \le \lambda, \quad b^T\theta = 0, \end{array}$$

where $A = [a_1 \ \cdots \ a_m]^T \in \mathbf{R}^{m\times n}$, the variable is $\theta \in \mathbf{R}^m$, and ϕ^* is the conjugate of the loss function ϕ,

$$\phi^*(y) = \sup_{u\in\mathbf{R}} (yu - \phi(u)).$$

As with ℓ_1-regularized logistic regression, we can derive a bound on the suboptimality of (v, w), by constructing a dual feasible point $\bar\theta$, from an arbitrary w,

$$\bar\theta = (s/m)p(\bar v, w), \qquad p(\bar v, w) = \begin{bmatrix} \phi'(w^T a_1 + \bar v b_1 + c_1) \\ \vdots \\ \phi'(w^T a_m + \bar v b_m + c_m) \end{bmatrix},$$

where $\bar v$ is the optimal intercept for the offset w,

$$\bar v = \arg\min_v (1/m)\sum_{i=1}^m \phi(w^T a_i + v b_i + c_i),$$

and the scaling constant s is given by $s = \min\left\{m\lambda/\|A^T p(\bar v, w))\|_\infty, 1\right\}$.

Using this method for cheaply computing a dual feasible point and associated duality gap for *any* (v, w), we can extend the custom interior-point method for ℓ_1-regularized LRPs to general ℓ_1-regularized convex (twice differentiable) loss problems.

Other possible extensions include ℓ_1-regularized Cox proportional hazards models (Cox, 1972). The associated ℓ_1-regularized problem does not have the form (19), but the idea behind the custom interior-point method for ℓ_1-regularized LRPs can be readily extended. The reader is referred to Park and Hastie (2006a) and Tibshirani (1997) for related work on computational methods for ℓ_1-regularized Cox proportional hazards models.

Acknowledgments

This work is supported in part by the National Science Foundation under grants #0423905 and (through October 2005) #0140700, by the Air Force Office of Scientific Research under grant #F49620-01-1-0365, by MARCO Focus center for Circuit & System Solutions contract #2003-CT-888, and by MIT DARPA contract #N00014-05-1-0700. The authors thank Michael Grant, Trevor Hastie, Honglak Lee, Suin Lee, Mee Young Park, Robert Tibshirani, Yinyu Ye, and Sungroh Yoon for helpful comments and suggestions. The authors thank anonymous reviewers for helpful comments and suggestions (especially on comparison with existing methods).

Appendix A. Standardization

Standardization is a widely used pre-processing step applied to the feature vector, so that each (transformed) feature has zero mean and unit variance (over the examples) (Ryan, 1997). The mean feature vector is $\mu = (1/m)\sum_{i=1}^m x_i$, and the vector of feature standard deviations σ is defined by

$$\sigma_j = \left((1/m)\sum_{i=1}^m (x_{ij} - \mu_j)^2\right)^{1/2}, \quad j = 1, \ldots, n,$$

where x_{ij} is the jth component of x_i. The *standardized feature vector* is defined as

$$x^{\text{std}} = \mathbf{diag}(\sigma)^{-1}(x - \mu).$$

When the examples are standardized, we obtain the standardized data matrix

$$A^{\text{std}} = \mathbf{diag}(b)(X - \mathbf{1}\mu^T)\,\mathbf{diag}(\sigma)^{-1} = A\,\mathbf{diag}(\sigma)^{-1} - b\mu^T\,\mathbf{diag}(\sigma)^{-1}, \tag{20}$$

where $X = [x_1 \cdots x_m]^T$. We carry out logistic regression (possibly regularized) using the data matrix A^{std} in place of A, to obtain (standardized) logistic model parameters w^{std}, v^{std}. In terms of the original feature vector, our logistic model is

$$\text{Prob}(b|x) = \frac{1}{1 + \exp\left(-b(w^{\text{std}\,T}x^{\text{std}} + v^{\text{std}})\right)} = \frac{1}{1 + \exp\left(-b(w^T x + v)\right)}$$

where

$$w = \mathbf{diag}(\sigma)^{-1}w^{\text{std}}, \qquad v = v^{\text{std}} - w^{\text{std}\,T}\,\mathbf{diag}(\sigma)^{-1}\mu.$$

We point out one subtlety here related to sparsity of the data matrix. For small or medium sized problems, or when the original data matrix A is dense, forming the standardized data matrix A^{std} does no harm. But when the original data matrix A is sparse, which is the key to efficient solution of large-scale ℓ_1-regularized LRPs, forming A^{std} is disastrous, since A^{std} is in general dense, even when A is sparse.

But we can get around this problem, when working with very large problems, by never actually forming the matrix A^{std}, as explained in Section 5.

References

E. Allgower and K. Georg. Continuation and path following. *Acta Numerica*, 2:1–64, 1993.

U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96:6745–6750, 1999.

S. Balakrishnan and D. Madigan. Algorithms for sparse linear classifiers in the massive data setting, 2006. Manuscript. Available from `www.stat.rutgers.edu/~madigan/papers/`.

O. Banerjee, L. El Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse Gaussian graphical models. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

A. Bhusnurmath and C. Taylor. Solving the graph cut problem via ℓ_1 norm minimization, 2007. University of Pennsylvania CIS Tech Report number MS-CIS-07-10.

J. Borwein and A. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer, 2000.

S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming, 2006. To appear in *Optimization and Engineering*.

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

S. Boyd, L. Vandenberghe, A. El Gamal, and S. Yun. Design of robust global power and ground networks. In *Proceedings of ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 60–65, 2001.

E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2005.

E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52 (2):489–509, 2006.

E. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

K. Chaloner and K. Larntz. Optimal Bayesian design applied to logistic regression experiments. *Journal of Statistical Planning and Inferenc*, 21:191–208, 1989.

S. Chen and D. Donoho. Basis pursuit. In *Proceedings of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44, 1994.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43 (1):129–159, 2001.

J. Claerbout and F. Muir. Robust modeling of erratic data. *Geophysics*, 38(5):826–844, 1973.

A. Conn, N. Gould, and Ph. Toint. *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*, volume 17 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1992.

D. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B*, 34(2): 187–220, 1972.

J. Dahl, V. Roychowdhury, and L. Vandenberghe. Maximum likelihood estimation of Gaussian graphical models: Numerical implementation and topology selection. Submitted. Available from `www.ee.ucla.edu/~vandenbe/covsel.html`, 2005.

A. d'Aspremont, L. El Ghaoui, M. Jordan, and G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming, 2005. In L. Saul, Y. Weiss and L. Bottou, editors, *Advances in Neural Information Processing Systems*, 17, pp. 41-48, MIT Press.

R. Dembo and T. Steihaug. Truncated-Newton algorithms for large-scale unconstrained optimization. *Math. Program.*, 26:190–212, 1983.

J. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

D. Donoho and M. Elad. Optimally sparse representation in general (non-orthogonal) dictionaries via ℓ^1 minimization. *Proc. Nat. Aca. Sci.*, 100(5):2197–2202, March 2003.

D. Donoho, I. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *J. R. Statist. Soc. B.*, 57(2):301–337, 1995.

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

J. Friedman, T. Hastie T, and R. Tibshirani. Pathwise coordinate optimization, 2007. Manuscript available from `www-stat.stanford.edu/~hastie/pub.htm`.

H. Fu, M. Ng, M. Nikolova, and J. Barlow. Efficient minimization methods of mixed ℓ_1-ℓ_1 and ℓ_2-ℓ_1 norms for image restoration. *SIAM Journal on Scientific computing*, 27(6):1881–1902, 2006.

A. Genkin, D. Lewis, and D. Madigan. Large-scale Bayesian logistic regression for text categorization, 2006. To appear in *Technometrics*. Available from `www.stat.rutgers.edu/~madigan/papers/`.

A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

P. Gill, W. Murray, M. Saunders, and M. Wright. User's guide for NPSOL (Version 4.0): A FORTRAN package for nonlinear programming. Technical Report SOL 86-2, Operations Research Dept., Stanford University, Stanford, California 94305, January 1986.

G. Golub and C. Van Loan. *Matrix Computations*, volume 13 of *Studies in Applied Mathematics*. John Hopkins University Press, third edition, 1996.

T. Golub, D. Slonim, P. Tamayo, C. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

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

A. Hassibi, J. How, and S. Boyd. Low-authority controller design via convex optimization. In *Proceedings of the IEEE Conference on Decision and Control*, pages 140–145, 1999.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29, 2007.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer-Verlag, New York, 2001. ISBN 0-387-95284-5.

T. Jaakkola and M. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10:25–37, 2000.

S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

Y. Kim, J. Kim, and Y. Kim. Blockwise sparse regression. *Statistica Sinica*, 16:375–390, 2006.

P. Komarek. *Logistic Regression for Data Mining and High-Dimensional Classification*. PhD thesis, Carnegie Mellon University, 2004.

B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.

B. Krishnapuram and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Mach. Intelligence*, 27(6): 957–968, 2005. ISSN 0162-8828.

K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 331–339, 1995.

S. Lee, H. Lee, P. Abeel, and A. Ng. Efficient l_1-regularized logistic regression. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, 2006.

S. Levy and P. Fullagar. Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution. *Geophysics*, 46(9):1235–1243, 1981.

C.-J. Lin, R. Weng, and S. Keerthi. Trust region Newton methods for large-scale logistic regression, 2007. To appear in Proceedings of the 24th International Conference on Machine Learning (ICML).

M. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 2005.

J. Lokhorst. The LASSO and generalised linear models, 1999. Honors Project, Department of Statistics, The University of Adelaide, South Australia, Australia.

D. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, second edition, 1984.

A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Available from `www.cs.cmu.edu/~mccallum/bow`, 1996.

P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapmand & Hall/CRC, second edition, 1989.

T. Minka. A comparison of numerical optimizers for logistic regression, 2003. Technical report. Available from `research.microsoft.com/~minka/papers/logreg/`.

MOSEK ApS. *The MOSEK Optimization Tools Version 2.5. User's Manual and Reference*, 2002. Available from `www.mosek.com`.

S. Nash. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*, 124:45–59, 2000.

Y. Nesterov and A. Nemirovsky. *Interior-Point Polynomial Methods in Convex Programming*, volume 13 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.

D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998. Available from `www.ics.uci.edu/~mlearn/MLRepository.html`.

A. Ng. Feature selection, ℓ_1 vs. ℓ_2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 78–85, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

D. Oldenburg, T. Scheuer, and S. Levy. Recovery of the acoustic impedance from reflection seismograms. *Geophysics*, 48(10):1318–1337, 1983.

M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.

M.-Y. Park and T. Hastie. An ℓ_1 regularization-path algorithm for generalized linear models, 2006a. To appear in *Journal of the Royal Statistical Society, Series B*. Available from `www-stat.stanford.edu/~hastie/pub.htm`.

M.-Y. Park and T. Hastie. Regularization path algorithms for detecting gene interactions, 2006b. Manuscript. Available from `www-stat.stanford.edu/~hastie/Papers/glasso.pdf`.

S. Perkins and J. Theiler. Online feature selection using grafting. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 592–599. ACM Press, 2003.

B. Polyak. *Introduction to Optimization*. Optimization Software, 1987. Translated from Russian.

L. Portugal, M. Resende, G. Veiga, and J. Júdice. A truncated primal-infeasible dual-feasible network interior point method. *Networks*, 35:91–108, 2000.

S. Rosset. Tracking curved regularized optimization solution paths. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths, 2007. To appear in *Annals of Statistics*.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

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

A. Ruszczynski. *Nonlinear Optimization*. Princeton university press, 2006.

T. Ryan. *Modern Regression Methods*. Wiley, 1997.

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

N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer Series in Computational Mathematics. Springer, 1985.

H. Taylor, S. Banks, and J. McCoy. Deconvolution with the l_1 norm. *Geophysics*, 44(1):39–52, 1979.

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

R. Tibshirani. The Lasso for variable selection in the Cox model. *Statistics in Medicine*, 16:385–395, 1997.

R. Tibshirani, M. Saunders, S. Rosset, and J. Zhu. Sparsity and smoothness via the fused Lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005.

J. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.

L. Vandenberghe and S. Boyd. A primal-dual potential reduction method for problems involving matrix inequalities. *Math. Program.*, 69:205–236, 1995.

R. Vanderbei. *LOQO User's Manual — Version 3.10*, 1997. Available from `www.orfe.princeton.edu/loqo`.

M. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression., 2007. To appear in *Advances in Neural Information Processing Systems (NIPS) 19*. Available from `http://www.eecs.berkeley.edu/~wainwrig/Pubs/publist.html#High-dimension%al`.

S. Wright. *Primal-dual Interior-point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. ISBN 0-89871-382-X.

Y. Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley & Sons, 1997.

M. Yuan and L. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68(1):49–67, 2006.

Z. Zhang, J. Kwok, and D. Yeung. Surrogate maximization/minimization algorithms for AdaBoost and the logistic regression model. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 927–934, New York, NY, USA, 2004. ACM Press.

P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties, 2007. Tech Report 703. Stat Dept. UCB. Available from `www.stat.berkeley.edu/~binyu/ps/703.pdf`.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In S. Thrun, L Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 49–56, Cambridge, MA, 2004. MIT Press.

H. Zou. The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):262–286, 2006.

H. Zou, T. Hastie, and R. Tibshirani. On the degrees of freedom of the Lasso, 2007. To appear in Annals of Statistics.