Optimization Methods for ℓ_1-Regularization

Mark Schmidt
Department of Computer Science
University of British Columbia

Glenn Fung Romer Rosaless
CAD and Knowledge Systems
Siemens Medical Solutions USA, Inc.

Original: March 11, 2008. Revised: August 4, 2009

Abstract

In this paper we review and compare state-of-the-art optimization techniques for solving the problem of minimizing a twice-differentiable loss function subject to ℓ_1-regularization. The first part of this work outlines a variety of the approaches that are available to solve this type of problem, highlighting some of their strengths and weaknesses. In the second part, we present numerical results comparing 14 optimization strategies under various scenarios.

1 Introduction

In this paper we will consider problems with the general form:

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq L(\mathbf{x}) + \lambda||\mathbf{x}||_1. \tag{1}$$

Here, $L(\mathbf{x})$ is a twice-differentiable loss function that is bounded from below, and the goal is to minimize this loss function with the ℓ_1-regularization. In addition to penalizing large values of the solution vector $\mathbf{x}$, for sufficiently large values of the scalar λ this yields solutions that are sparse in terms of $\mathbf{x}$ (having many values set to exactly 0). Because of these regularization and sparsity-inducing properties, there has been substantial recent interest in this type of ℓ_1-regularization in the statistics and signal processing communities, beginning with [Chen et al., 1999, Tibshirani, 1996].

Unfortunately, since the combined objective function $f(\mathbf{x})$ is non-differentiable when x contains values of 0, this precludes the use of standard unconstrained optimization methods. This has lead to a wide variety of approaches proposed in the literature to solve problems of this form. This includes a variety of algorithms for special cases where $L(\mathbf{x})$ has a specific functional form, such as the squared error in a linear regression model [Osborne et al., 2000, Efron et al., 2004] or the logistic regression negative log-likelihood [Roth, 2004, Lee et al., 2006, Koh et al., 2007]. In this paper, we focus on the more general case where $L(\mathbf{x})$ is simply a twice-differentiable function, and our contributions are (i) we discuss methods that are capable of solving the general problem (1), and (ii) we experimentally compare a variety of methods.

In the next three sections, we outline different approaches for solving (1), where the methods are roughly divided into their approach for handling the non-differentiability of the objective function. In particular, the next section discusses *sub-gradient* methods (§2), methods that directly optimize (1) and use sub-gradients at points of non-differentiability. We then move on to *unconstrained approximations* (§3), methods that circumvent the lack of smoothness by optimizing a smooth surrogate objective function. We finally discuss *constrained optimization* methods in §4, where the unconstrained non-differentiable problem is transformed into an equivalent differentiable problem with constraints. Our discussion in these three sections will largely focus on the properties of the methods in relation to Newton's method for unconstrained optimization, augmented with an Armijo backtracking line search (see [Fletcher, 1987]). Subsequently, these sections will assume that we are solving for a single value of the regularization parameter λ, and that the matrix of second partial derivatives of $L(\mathbf{x})$ (the Hessian) is available and is positive-definite (our experiments examine more general scenarios, such as cases where the Hessian is replaced by an approximation, or may not be positive-definite). §5 presents a set of numerical comparisons among fourteen optimization methods discussed in this

work. Our numerical results indicate that some strategies for addressing the non-differentiable loss are much more efficient than others. We conclude the paper with a discussion of the limitations of our experiments, and references to methods that are not discussed/evaluated in this work. To provide a more coherent exposition, we have omitted some implementation details, but implementations of all the methods discussed in this work are available on the author's homepage.

2 SubGradient Strategies

We first examine optimization strategies that treat the non-differentiable objective as a non-smooth optimization problem, by using sub-gradients of the objective function at non-differentiable points. In non-smooth optimization, a necessary condition for a parameter vector $\mathbf{x}$ to be a local minima is that the zero-vector $\mathbf{0}$ is an element of the sub-differential $\partial f(\mathbf{x})$, the set containing all sub-gradients at $\mathbf{x}$ [Fletcher, 1987]. The sub-differential of the absolute value function $|x_i|$ is given by the *signum* function $sgn(x_i)$. The signum function takes on the sign of x_i if x_i is non-zero, and if x_i is zero then the signum function can take any value in the range $[-1, 1]$. For our problem, the *optimality conditions* therefore translate to:

$$\begin{cases} \nabla_i L(\mathbf{x}) + \lambda \operatorname{sign}(x_i) = 0, & |x_i| > 0 \\ |\nabla_i L(\mathbf{x})| \leq \lambda, & x_i = 0 \end{cases}$$

We might also consider selecting (coordinate-wise) the following sub-gradient $\nabla_i^s f(x_i)$ for each x_i, whose negation represents the *steepest descent direction*:

$$\nabla_i^s f(\mathbf{x}) \triangleq \begin{cases} \nabla_i L(\mathbf{x}) + \lambda \operatorname{sign}(x_i), & |x_i| > 0 \\ \nabla_i L(\mathbf{x}) + \lambda, & x_i = 0, \nabla_i L(\mathbf{x}) < -\lambda \\ \nabla_i L(\mathbf{x}) - \lambda, & x_i = 0, \nabla_i L(\mathbf{x}) > \lambda \\ 0, & x_i = 0, -\lambda \leq \nabla_i L(\mathbf{x}) \leq \lambda \end{cases}$$

For a sub-optimal $\mathbf{x}$, this sub-gradient will yield a descent direction on the objective function, and several authors have proposed using this sub-gradient as a surrogate for the gradient within optimization procedures (we outline a few example below). In these methods, it is assumed that $\nabla^2 f(\mathbf{x}) \triangleq \nabla^2 L(\mathbf{x})$, even though this is not strictly true if any x_i is 0.

2.1 Coordinate Descent

Perhaps the simplest methods that we will discuss are sub-gradient methods that optimize over one variable at a time. These methods take the general form:

```
Do until termination criteria satisfied:
   1. Choose a coordinate i.
   2. Find the optimal value of x_i, given the remaining parameters.
```

The **shooting** algorithm of [Fu, 1998] is a specific instantiation of this method for the case of a least squares loss, $L(\mathbf{x}) \triangleq ||A\mathbf{x} - b||_2^2$. In the shooting algorithm, step 1 is implemented by repeatedly cycling through the variables in order, and the update in step 2 has a simple analytic solution, obtained by setting x_i to 0, and then setting $x_i \leftarrow x_i - \nabla_i^s f(\mathbf{x}) / \nabla_{ii}^2 L(\mathbf{x})$.

The **Gauss-Seidel**[1] algorithm of Shevade and Keerthi [2003] is another implementation of this general procedure, but for the case of the logistic regression negative log-likelihood. In this case, the update in step 2 is computed numerically with a line search[2]. Initializing this line search requires considering 7 possible scenarios in order to bracket the solution, and the line search proceeds by performing 1-dimensional (safeguarded) Newton steps of the form $x_i \leftarrow x_i - \nabla_i^s f(\mathbf{x}) / \nabla_{ii}^2 L(\mathbf{x})$, until a sufficiently accurate solution is found. In the special case of the least squares loss, this line search will converge in one iteration and is

[1] In the optimization literature, the algorithm of Shevade and Keerthi [2003] more closely resembles the 'Gauss-Southwell' procedure than the 'Gauss-Seidel' algorithm. We use 'Gauss-Seidel' to be consistent with Shevade and Keerthi [2003]

[2] In the case of logistic regression, an alternative that has also been explored is to minimize coordinate-wise an upper bound on the objective [Krishnapuram et al., 2005]

identical to the shooting update, but the algorithm of [Shevade and Keerthi, 2003] applies to general loss functions. The Gauss-Seidel method also differs from the shooting method in the selection of the coordinate to update in step 1. In the Gauss-Seidel method, we select the non-zero variable whose violation of the first-order optimality condition ($|\nabla_i L(\mathbf{x}) + \lambda \text{ sign}(x_i)|$) is largest. If all non-zero variables satisfy the optimality condition to within a threshold, then the most violating zero-valued variables is used. If no zero-valued variable violates the optimality condition ($|\nabla_i L(\mathbf{x})| \leq \lambda$), the algorithm terminates. When initialized with the zero-vector $\mathbf{0}$, the Gauss-Seidel method is advantageous over the shooting method for models with very sparse solutions, since it may only introduce a small number of non-zero variables.

Although coordinate descent methods are simple and can be effective if the variables have largely independent effects on the objective, their performance degrades as variables become more dependent. In particular, for difficult large-scale problems, coordinate descent methods may require so many coordinate updates that they become impractical. Thus, we are led to consider methods that update multiple variables on each iteration.

2.2 Active Set Methods

Since the the vector $-\nabla^s f(\mathbf{x})$ is analogous to the steepest descent (negative gradient) direction in smooth optimization, and since $\nabla^2 f(\mathbf{x}) = \nabla^2 L(\mathbf{x})$ it we pick any element of the sub-differential for the first derivative, we might be tempted to try and speed convergence of the method by using Newton-like iterations of the form

$$\mathbf{x} \leftarrow \mathbf{x} - \beta H^{-1} \nabla^s f(\mathbf{x}), \tag{2}$$

where $H \triangleq \nabla^2 L(\mathbf{x})$ and the step size $\beta > 0$ is chosen to satisfy a sufficient decrease rule (by a backtracking line search, for example). Unfortunately, the direction $(-H^{-1}\nabla^s f(\mathbf{x}))$ is not guaranteed to be a descent direction when $\mathbf{x}$ is at a non-differentiable point, so in general there will not exist a $\beta > 0$ that makes this update improve on the objective function. However, we can still consider applying this type of update to the non-zero variables within the context of an *active set* method. In particular, consider a method of the form:

```
Do until termination criteria satisfied:
   1.  Update the working set of non-zero variables.
   2.  Optimize the non-zero variables, given the active set of zero-valued variables.
```

Using $\mathcal{W}$ to denote the working set, step 2 of this algorithm can be implemented using Newton-like iterations applied only to the working set

$$\mathbf{x}_{\mathcal{W}} \leftarrow \mathbf{x}_{\mathcal{W}} - \beta H_{\mathcal{W}}^{-1} \nabla^s_{\mathcal{W}} f(\mathbf{x}),$$

which will be a descent direction provided that all elements of $\mathbf{x}_{\mathcal{W}}$ are non-zero (recall that we are assuming that $\nabla^2 L(\mathbf{x})$ is available, and is positive-definite). Below, we consider several methods for performing step 1.

When applied to ℓ_1-regularized problems, the **grafting** procedure [Perkins et al., 2003] implements an active set method where step 1 is implemented by adding the variable with the largest optimality violation ($|\nabla_i L(\mathbf{x}) + \lambda \text{ sign}(x_i)|$) to the working set (if $|\nabla_i L(\mathbf{x})| \leq \lambda$ for all zero-valued i, then the algorithm terminates). This will tend to generate a descent direction on the first iteration of step 2, and this descent can be guaranteed if the new working variable is perturbed away from zero. Grafting is reminiscent of the Gauss-Seidel coordinate descent method, but grafting tends to converge after far fewer line searches, because many variables are being updated at once. In the special case of a quadratic $L(\mathbf{x})$, the Newton iteration will converge in a single step, provided that all variables in $\mathbf{x}$ have the appropriate sign.

Another variant on the active set method has been proposed as a sub-routine in the context of path-following algorithms (for solving the problem for a sequence of values of λ) [Rosset, 2004, Park and Hastie, 2007]. We will refer to this method as **sub-gradient descent**, and in this method the working set is updated after each iteration of step 2. Here, the working set is defined as the non-zero variables, plus the zero-valued variables that do not satisfy the optimality condition for a zero-valued variable. Formally, we can write this as $\mathcal{W} \triangleq \{i|x_i \neq 0\} \cup \{i|x_i = 0, |\nabla_i L(\mathbf{x})| > \lambda\}$. Note that once again, we must consider perturbing the zero-valued variables away from zero if we want to ensure that the update will make progress in decreasing

the objective function. Unfortunately, this algorithm can converge very slowly, since it may introduce many more non-zero variables than needed, and subsequently needs to performs many iterations where several non-zero variables slowly move back to 0.

In our experiments, we also explored a method that combines ideas from the grafting and sub-gradient descent methods, that we refer to as the **max-k sub-gradient descent** method. In this method, we update the working set after each iteration of step 2 (as in the sub-gradient descent method). However, similar to grafting, we restrict the number of variables that can be added to the working set at each iteration. In particular, we define the working as the set of non-zero variables, combined with the k variables that have the largest violation in the optimality conditions. In the case of $k = 1$, this is similar to the grafting method, except that a new variable can potentially be introduced after each update, and new variables can be introduced before the current working set is fully optimized. In our experiments, this method required fewer iterations to converge than all the algorithms discussed above.

Active set methods have also been examined for specific loss functions, including least squares [Osborne et al., 2000], and logistic regression [Roth, 2004]. Homotopy methods and the LARS algorithm [Osborne et al., 2000, Efron et al., 2004] are also closely related. Active set methods are most efficient when we have a good guess of the initial active set. This could include starting with the zero vector $\mathbf{0}$ when the final solution is very sparse, or starting with a set of non-zero variables that is very close to the optimal set. However, typically active set methods only add/remove a small number of variables from the active set, and thus they may require a large number of iterations if we do not have a good initial guess of the active set (such as starting with the zero vector $\mathbf{0}$ when the final solution has many non-zero values).

2.3 Orthant-Wise Descent

Rather than using an active set method, [Andrew and Gao, 2007] consider a very different strategy for 'fixing' the Newton-like sub-gradient iteration (2). We will refer to this method as **orthant-wise descent**, and it consists of iterations taking the form

$$\mathbf{x} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{x} - \beta \mathcal{P}_S[H^{-1}\nabla^s f(\mathbf{x})]].$$

This is identical to (2), but it adds two projection operators, $\mathcal{P}_{\mathcal{O}}$ and $\mathcal{P}_S$. The operator $\mathcal{P}_S$ projects the Newton-like direction so that it is guaranteed to be a descent direction. This is accomplished by setting elements of the vector $(H^{-1}\nabla^s f(\mathbf{x}))$ to zero if they do not have the same sign as the corresponding element of $\nabla^s f(\mathbf{x})$ (in this context, 0 is assumed to have a different 'sign' than positive or negative values). Viewed from the point of view of an active set method, this is equivalent to setting the working set to those variables whose sign does not change after applying the inverse-Hessian scaling (and it can be shown that not all variables can change sign simultaneously). The second projection operator, $\mathcal{P}_{\mathcal{O}}$, projects the step onto the orthant containing $\mathbf{x}$ (variables with a value of 0 are assigned to the orthant that the steepest descent direction points into). This second projection operator is effective at sparsifying the current solution, since it can set many variables to 0.

The orthant-wise descent method is advantageous over active set methods when we do not have a good guess for the active set, since the orthant-wise descent method can quickly make a large number of changes to the set of non-zero variables. Despite this advantage, orthant-wise descent also has a disadvantage over active set methods; Once the correct set of non-zero variables (and their sign) is identified, orthant-wise descent does *not* reduce to applying Newton's method on this subset of variables (because of the $\mathcal{P}_{\mathcal{S}}$ projection), and thus does not achieve its local rate of convergence.

3 Differentiable Approximations

Rather than working directly with $f(\mathbf{x})$ and attempting to solve a non-smooth optimization problem, in this section we discuss methods that replace $f(\mathbf{x})$ with a twice-differentiable surrogate objective function $g(\mathbf{x})$, whose minimizer is sufficiently close to the minimizer of $f(\mathbf{x})$. The main advantage of this approach is that, since $g(\mathbf{x})$ is twice-differentiable, we can directly apply an unconstrained optimization method to minimize $g(\mathbf{x})$.

3.1 Smoothing

We first consider a very simple method for transforming the non-differentiable optimization into a differentiable optimization, by replacing the non-differentiable term with a differentiable approximation. In particular, consider the **epsL1** approximation defined in Lee et al. [2006], that replaces the ℓ_1 norm with the sum of multi-quadric functions,

$$g(\mathbf{x}) \triangleq L(\mathbf{x}) + \lambda \sum_i \sqrt{x_i^2 + \epsilon}.$$

This function is twice-differentiable and $\lim_{\epsilon \to 0^+} g(\mathbf{x}) = f(\mathbf{x})$, since $\lim_{\epsilon \to 0^+} \sqrt{x_i^2 + \epsilon} = |x_i|$. Subsequently, for a sufficiently small positive ϵ, a minimizer of $g(\mathbf{x})$ will correspond (numerically) to a minizer of $f(\mathbf{x})$. Given this approximation, we can now consider directly applying Newton iterations to minimize $g(\mathbf{x})$,

$$\mathbf{x} \leftarrow \mathbf{x} - \beta H^{-1} \nabla g(\mathbf{x}),$$

where $H \triangleq \nabla^2 g(\mathbf{x})$, and β is again chosen by a backtracking line search, or a more sophisticated line search that seeks to satisfy a stronger criteria like the strong Wolfe conditions [Fletcher, 1987]. Unfortunately, the strong local convergence properties of Newton's method (and other unconstrained optimization methods) depend on the condition number of the Hessian matrix H at the minimizer, and as we now show, the condition number of the regularizer's contribution to H is the downfall of the method.

Using $b(\mathbf{x}) \triangleq \sum_i \sqrt{x_i^2 + \epsilon}$, we see that $\nabla^2 b(\mathbf{x})$ has diagonal terms $\nabla^2_{ii} b(\mathbf{x}) = \epsilon / (\sqrt{x_i^2 + \epsilon})^3$, while off-diagonal terms are zero. For a sufficiently non-zero x_i (ie. $|x_i| >> \epsilon$), we can approximate the diagonal elements by $\nabla^2_{ii} b(\mathbf{x}) \approx \epsilon / |x|^3$, a positive number that becomes very small as $|x_i|$ increases. In contrast, if x_i is 0, then $\nabla^2_{ii} = \sqrt{\epsilon^{-1}}$, a relatively large value for small ϵ. Assuming that in the optimal solution we have some $|x_i| >> \epsilon$, and that some $x_i = 0$ (ie. exactly the type of minimizer we are looking for), the matrix condition number of the diagonal matrix $\nabla^2 b(\mathbf{x})$ is thus approximated by $\epsilon^{-3/2} \max_i (|x_i|)^3$, the product of a large number ($\epsilon^{-3/2}$) with the largest magnitude value of $\mathbf{x}$ raised to the third power. This ill-conditioning subsequently leads to slow convergence of the Newton iterations. Further, although we will only show it in this case, this poor conditioning is inevitable for any (sufficiently accurate) smooth approximation of the non-differentiable $f(\mathbf{x})$.

3.2 Bound Optimization

Rather than replacing the regularizer with a fixed smooth function, at each iteration bound optimization methods replace the regularizer with a convex upper bounding function (that is exact at the current value of $\mathbf{x}$). Subsequently, the method alternates between two simple steps:

```
Do until termination criteria satisfied:
   1. Compute convex upper bounding function g(x).
   2. Approximately minimize g(x).
```

Although many bounding functions are possible, following [Krishnapuram et al., 2005] we will consider a bound on the square root function,

$$\sqrt{u} \leq \frac{u}{2\sqrt{v}} + \frac{\sqrt{v}}{2}.$$

This holds with equality when $u = v$. Using y_i to denote the value of x_i from the previous iteration, and writing $|x_i| = \sqrt{x_i^2}$, we obtain the bounding function

$$|x_i| \leq \frac{1}{2} \frac{x_i^2}{|y_i|} + \frac{1}{2} |y_i|.$$

Note that the equality is exact when $y_i = x_i$, and that the right hand side is a convex function of x_i. Subsequently, to implement the bound optimization algorithm we alternate between setting $\mathbf{y} \leftarrow \mathbf{x}$ in step 1, and performing a Newton step on the twice-differentiable objective function

$$g(\mathbf{x}) \triangleq L(\mathbf{x}) + \lambda \sum_i \frac{1}{2} \frac{x_i^2}{|y_i|} + \frac{1}{2} |y_i|,$$

in step 2. Note that step 2 takes the form of ℓ_2-regularization, where the regularizer has a separate weight for each parameter. Thus, if $L(\mathbf{x})$ is quadratic, this update has a closed-form solution.

This algorithm has been proposed in different contexts by numerous authors [see Tibshirani, 1996, Grandvalet and Canu, 1998, Fan and Li, 2001, Figueiredo, 2003, Krishnapuram et al., 2005]. One of the reasons for this algorithm's popularity (in probabilistic contexts) is that it arises naturally as an Expectation Maximization (EM) algorithm under a scale mixture of Normals representation of the Laplace prior[3] [see Figueiredo, 2003]. In the context of this **EM** algorithm, updating $\mathbf{y}$ corresponds to the 'E-step' (computing the expectations of the variances in the mixture), while optimizing $g(\mathbf{x})$ given $\mathbf{y}$ corresponds to the 'M-step' (optimizing the parameter vector given the expectations over the variances). In this context, the method is typically implemented via an iteratively-reweighted least squares (IRLS) update[4], but the bound optimization perspective is more general since not all choices of $L(\mathbf{x})$ yield IRLS updates.

One problem with the bound optimization approach is that the surrogate objective function $g(\mathbf{x})$ is undefined whenever $\mathbf{y}$ contains a value of zero. To address this problem, the algorithm must be initialized with a non-zero $\mathbf{x}$, and implementations of the method typically remove variables from the problem if they become sufficiently close to 0, or use re-formulations that are equivalent to removing these variables from the model [Figueiredo, 2003, Krishnapuram et al., 2005]. Although removing the variables from the problem leads to computational gains, it can be problematic if a variable is removed that is non-zero in the optimal solution, since if this happens the algorithm will not converge to the optimal solution. Despite the popularity of the method, we have only seen this seemingly catastrophic flaw mentioned in Fan and Li [2001], where the authors state that this did not occur in their experiments. Indeed, it did not arise in our experiments either, although it is trivial to induce such behaviour (by initializing variables to zero, for example).

Ignoring the problematic aspects of the model when variables are exactly 0, the method is still somewhat problematic when variables become *close* to 0, since division by y_i for variables that are close to 0 causes the same sort of ill-conditioning that arises in smoothing methods. In particular, the condition number of the regularizer's contribution to the Hessian of $g(\mathbf{x})$ will be $(1/\min(|y_i|))/(1/\max_i(|y_i|))$, a value that can clearly grow quite large when the solution contains values that are non-zero and values that are (close to) zero. As in smoothing methods, this leads to slow convergence.

3.3 Smoothing with Continuation

In the EM method, we constructed a *sequence* of unconstrained approximations, where each unconstrained approximation represented an upper bound that was locally exact. In this section, we return to smoothing methods, and we consider constructing a *sequence* of smooth unconstrained approximations. In particular, we will design a sequence whose difference with the original function is bounded, and we will choose the sequence such that this error converges to zero. The intuition behind this type of approach is that if we start with an easily optimized smooth approximation, and if the smooth approximations converge to the objective function slowly enough, we may be able to keep the iterates close enough to a region where Newton's method achieves a fast rate of convergence. The general outline of this type of 'smoothing with continuation' method is:

```
Do until termination criteria satisfied:
   1.  Approximately minimize current smooth approximation g(x, α).
   2.  Update the smoothing parameter α.
```

The particular smooth approximation we consider results from writing the absolute value function as

$$|x| = (x)_+ + (-x)_+,$$

where the *plus* function is defined as $(x)_+ \triangleq \max\{x, 0\}$. Following [Lee and Mangasarian, 2001], the plus

[3]The scale mixture of Normals representation of the Laplace prior takes the form $x_i|\tau_i \sim N(0, \tau_i)$, where the variances of the individual Normals have an exponential prior, $p(\tau_i|\sqrt{\lambda}) = \frac{\sqrt{\lambda}}{2}\exp(\frac{-\tau_i\sqrt{\lambda}}{2})$. Under this representation, integrating over τ_i yields a Laplacian density for $p(x_i|\lambda)$ (and thus an ℓ_1-regularizer of the negative log-likelihood).

[4]IRLS is a technique widely used in statistics that allows the application of a pure Newton step (ie. $\beta = 1$) to be re-formulated as the solution of a weighted least squares problem for some objective functions

function can be smoothly approximated as

$$(x)_+ \approx x + \frac{1}{\alpha}\log(1+\exp(-\alpha x)). \tag{3}$$

By making use of this approximation twice, we obtain the smooth approximation to the absolute value

$$|x|_\alpha \triangleq \frac{1}{\alpha}\left[\log(1+\exp(-\alpha x)) + \log(1+\exp(\alpha x))\right],$$

and the twice-differentiable unconstrained approximation

$$g(\mathbf{x},\alpha) \triangleq L(\mathbf{x}) + \lambda \sum_i |x_i|_\alpha. \tag{4}$$

We refer to this loss function as the *smoothL1* approximation, and following arguments similar to [Lee and Mangasarian, 2001], we can prove the approximation accuracy bound

$$||x| - |x|_\alpha| \leq 2\frac{\log 2}{\alpha}. \tag{5}$$

By (approximately) minimizing $g(\mathbf{x},\alpha)$ using Newton iterations for an increasing sequence of α values, we partially avoid the slow convergence associated with the method, compared to using a large fixed value of α. In our experiments, we refer to using a fixed value of α as the **smoothL1-SC** (for 'short-cut') method, and using a slowly increasing sequence α values as the **smoothL1-CT** (for 'continuation') method. Note that we could also consider a continuation version of the epsL1 method, and in this case the bound on the approximation in terms of the smoothing parameter ϵ would be $\sqrt{\epsilon}$.

Although they seem appealing at first, we have seen in this section that differentiable approximations may converge slowly due to the effects of ill-conditioning. Although using a continuation strategy partially alleviates this issue, it introduces two new issues; continuation methods can not take fully advantage of a good initialization, and continuation methods initially make *all* variables non-zero (even if the final solution is sparse).

4 Constrained Optimization

The final class of methods that we will discuss are methods that solve the non-differentiable problem by re-formulating it as a differentiable problem with constraints, that shares the same minimizer. Subsequently, methods with fast convergence rates from the constrained optimization world can be applied.

We will consider a constrained formulation where each variable x_i is represented as the sum of two variables,

$$x_i = x_i^+ - x_i^-.$$

Further, we enforce that $x_i^+ \geq 0$ and $x_i^- \geq 0$. Since in this formulation $|x_i| = x_i^+ + x_i^-$, we can write the optimization problem as

$$\begin{array}{ll} \min\limits_{\mathbf{x}^+,\mathbf{x}^-} & L(\mathbf{x}^+ - \mathbf{x}^-) + \lambda\sum_i [x_i^+ + x_i^-], \\ s.t. & \forall_i\ x_i^+ \geq 0, x_i^- \geq 0. \end{array} \tag{6}$$

An obvious drawback of this approach is that we have doubled the number of variables in the optimization problem. Although other constrained formulations are possible [see Koh et al., 2007], these also have this drawback.

4.1 Primal Log-Barrier

Primal **log-barrier** methods applied to constrained formulations have been proposed for the special cases of the least squares loss [Kim et al., 2007], and the logistic regression loss [Koh et al., 2007]. Log-barrier methods are structurally similar to the continuation methods discussed in the previous section. They also form a sequence of unconstrained approximations that slowly converge to the true objective function. However,

rather than smoothing the objective, they append a 'barrier' function to the objective function that prevents the parameters from coming too close the edge of the feasible set (in this case, ensuring that x_i^+ and x_i^- remain sufficiently positive). Formally, the unconstrained approximations are parameterized in terms of a scalar μ, and take the form

$$g(\mathbf{x}^+, \mathbf{x}^-, \mu) \triangleq L(\mathbf{x}^+ - \mathbf{x}^-) + \lambda \sum_i [x_i^+ + x_i^-] - \mu \sum_i \log x_i^+ - \mu \sum_i \log x_i^-.$$

The log-barrier functions on $\mathbf{x}^+$ and $\mathbf{x}^-$ go to infinity as any element of these vectors approaches zero, and thus they explicitly force the iterations to stay within the interior of the constraint set (the line search in the Newton iterations must be truncated to ensure that this is true). However, as μ approaches zero, the minimizer of $g(\mathbf{x}^+, \mathbf{x}^-, \mu)$ converges to the minimizer of $f(\mathbf{x})$. As with other differentiable approximations, convergence of the log-barrier method will be slowed by the ill-conditioning caused by the log-barrier terms. For this reason, a continuation strategy is typically adopted, where μ is slowly decreased. Standard references detailing log-barrier methods include [Nocedal and Wright, 1999, Boyd and Vandenberghe, 2004].

4.2 Primal-Dual Log-Barrier

Log-barrier methods fall within the general class of interior point methods. In the basis pursuit denoising literature, primal-dual log-barrier methods have been proposed for the case of the least squares loss [Chen et al., 1999]. These methods simultaneously optimize both the primal variables $\{\mathbf{x}^+, \mathbf{x}^-\}$, and a set of dual variables $\{\mathbf{v}^+, \mathbf{v}^-\}$ that act like Lagrange multipliers on the constraints. Following [Boyd and Vandenberghe, 2004], in this section we outline a method of this type.

First, we note that the Lagrangian of the constrained formulation is

$$L(\mathbf{x}^+ - \mathbf{x}^-) + \lambda \sum_i [x_i^+ + x_i^-] - \sum_i v_i^+ x_i^+ - \sum_i v_i^- x_i^-.$$

For simplicity, we will use $\mathbf{x}^*$ to denote the concatenation of $\mathbf{x}^+$ and $\mathbf{x}^-$, and $\mathbf{v}^*$ to denote the concatenation of $\mathbf{v}^+$ and $\mathbf{v}^-$. Using this notation, the Karush-Kuhn-Tucker (KKT) optimality conditions for the constrained formulation state that the gradient of this Lagrangian (with respect to $\mathbf{x}^*$) must be the zero vector $\mathbf{0}$, that $\mathbf{x}^*$ and $\mathbf{v}^*$ are non-negative, and that the complementarity condition $\mathbf{x}^* \bullet \mathbf{v}^* = \mathbf{0}$ holds (where $\bullet$ denotes element-wise product). Primal-dual log-barrier methods typically work with a perturbed version of these conditions, where a barrier parameter μ is used in place of the zero vector $\mathbf{0}$ in the complementarity conditions. Assuming that all variables are non-negative, we can write the remaining KKT conditions as the non-linear system of equations

$$\mathbf{0} = r(\mathbf{x}^*, \mathbf{v}^*) \triangleq \begin{bmatrix} \nabla L(x^+ - x^-) + \lambda \mathbf{1} - \mathbf{v}^* \\ -\mathbf{v}^* \bullet \mathbf{x}^* - \mu \mathbf{1} \end{bmatrix}.$$

We seek to solve the equation $r(\mathbf{x}^*, \mathbf{v}^*) = \mathbf{0}$ by using Newton iterations of the form

$$(\mathbf{x}^*, \mathbf{v}^*) \leftarrow (\mathbf{x}^*, \mathbf{v}^*) - \beta \nabla r(\mathbf{x}^*, \mathbf{v}^*)^{-1} r(\mathbf{x}^*, \mathbf{v}^*),$$

where the step length β is truncated to ensure that $\mathbf{x}^*$ and $\mathbf{v}^*$ are positive. Between iterates, the barrier parameter μ is updated based on an update rate (we used 10), the number of constraints m, and the surrogate duality gap $\mathbf{v}^{*T}\mathbf{x}^*$,

$$\mu \leftarrow \frac{10m}{\mathbf{v}^{*T}\mathbf{x}^*}.$$

In this method the objective function $f(\mathbf{x})$ is not used directly as measure of progress, but rather we measure progress based on violation of the condition $r(\mathbf{x}^*, \mathbf{v}^*) = \mathbf{0}$. For more details, see [Boyd and Vandenberghe, 2004]. In our experiments, we refer to this method as the **interior-point** method. Our experiments indicate that the primal-dual log-barrier method takes fewer iterations to converge than the primal log-barrier method, although its implementation is more complex and requires four times the number of variables present in the original problem (instead of simply doubling the number of variables).

4.3 Projected Newton

A disadvantage of the log-barrier methods is that they are not able to make effective use of a good initialization. Further, the log-barrier methods require that all coefficients are non-zero (except in the limit), and therefore do not take advantage of the sparsity that may be present in the solution. In this section we discuss a projected Newton method, that does not have these disadvantages. An approach of this type for the special case of logistic regression was proposed in [Lee et al., 2006].

In the unconstrained Newton method, we define the search direction $\mathbf{d}$, in terms of the minimizer $\mathbf{y}^*$ of a truncated second-order Taylor expansion of the objective function $f(\mathbf{x}^*)$ around the current iterate,

$$\min_{\mathbf{y}^*} \ \nabla f(\mathbf{x}^*)^T(\mathbf{x}^* - \mathbf{y}^*) + \frac{1}{2}(\mathbf{x}^* - \mathbf{y}^*)^T\nabla^2 f(\mathbf{x}^*)(\mathbf{x}^* - \mathbf{y}^*),$$

as $\mathbf{d} \leftarrow \mathbf{x}^* - \mathbf{y}^*$. In the projected Newton method for constrained optimization, we define the search direction analogously, but enforce that $\mathbf{y}^*$ obeys the constraints,

$$\begin{aligned} \min_{\mathbf{y}^*} \quad & \nabla f(\mathbf{x}^*)^T(\mathbf{x}^* - \mathbf{y}^*) + \tfrac{1}{2}(\mathbf{x}^* - \mathbf{y}^*)^T\nabla^2 f(\mathbf{x}^*)(\mathbf{x}^* - \mathbf{y}^*), \\ & s.t. \ \forall_i \ y_i^+ \geq 0, y_i^- \geq 0. \end{aligned} \tag{7}$$

The (feasible descent) direction is then defined as $\mathbf{d}^* \leftarrow (\mathbf{x}^* - \mathbf{y}^*)$, and the iterations take the form

$$\mathbf{x}^* \leftarrow \mathbf{x}^* + \beta\mathbf{d},$$

but β is now restricted to lie in $(0, 1]$ to ensure that the constraints remain satisfied. This type of iteration has similar convergence properties to unconstrained Newton iterations [Bertsekas, 2004].

Although this seems to achieve our goal of developing a method with a Newton-like convergence rate (at the cost of doubling the number of variables), the main drawback of this method is that calculating $\mathbf{y}^*$ requires solving a quadratic program. Thus, depending on the form of $L(\mathbf{x})$, the iteration cost of the method may approach the cost of solving the full problem. Indeed, in the case where $L(\mathbf{x})$ is a quadratic function, this method will find the optimal solution in one iteration. Nevertheless, this sequential quadratic programming method (referred to as **SQP** in our experiments) may be useful in cases where the cost of evaluating the objective function is substantially larger than the cost of solving the quadratic program.

4.4 Two-Metric Projection

The constrained optimization methods discussed above represent general constrained optimization techniques, applied to this specific problem. In this final section, we consider two-metric projection methods [Gafni and Bertsekas, 1984], that are especially suitable for problems with simple bound constraints. In particular, they combine the advantages of the SQP method (fast convergence, able to take advantage of good initializations, can quickly make variables zero/non-zero, tend to maintain sparse solutions), but with a substantially reduced iteration cost. Our discussion in this section will largely follow [Bertsekas, 2004].

The standard gradient projection method applied to this problem would take steps of the form

$$\mathbf{x}^* \leftarrow [\mathbf{x}^* - \beta(\nabla f(\mathbf{x}^*)]^+.$$

The *plus* function projects the iterates onto the non-negative orthant, ensuring that the variables satisfy the constraints, while the use of the negative gradient direction ensures that the projection is a descent direction. We might consider accelerating this iteration by scaling the negative gradient by the inverse Hessian, as in Newton methods. Unfortunately, this will not generally yield a descent direction after projection under the Euclidean norm. Under a suitable choice of the projection norm, we can ensure that the projected Newton iteration yields a descent direction, and this leads to the projected Newton method of the previous section (and its associated costly iterations).

In two-metric projection methods, we keep the (inexpensive) Euclidean projection, but enforce that the scaling matrix has a special structure. This special structure will ensure that we obtain a direction of descent, even though the metric used in the projection is different than the metric used to scale the negative gradient. In particular, at each iteration we define the active set as the set of variables who are numerically close to

zero and have a positive gradient, $\mathcal{A} = \{i | x_i^* \leq \epsilon, \nabla_i f(\mathbf{x}^*) > 0\}$. For these variables, we do not scale the negative gradient direction (ie. they are scaled by the identity matrix). The remaining variables form the working set $\mathcal{W}$, and these variables are scaled by the inverse of the corresponding submatrix of the Hessian, leading to iterations for the form

$$\mathbf{x}_{\mathcal{A}}^* \leftarrow [\mathbf{x}_{\mathcal{A}}^* - \beta \nabla_{\mathcal{A}} f(\mathbf{x}^*)]^+, \tag{8}$$

$$\mathbf{x}_{\mathcal{W}}^* \leftarrow [\mathbf{x}_{\mathcal{W}}^* - \beta [\nabla_{\mathcal{W}}^2 f(\mathbf{x}^*)]^{-1} \nabla_{\mathcal{W}} f(\mathbf{x}^*)]^+. \tag{9}$$

We refer to this as the **projectionL1** method. It can be shown that this direction yields a feasible descent direction and that it will identify (and maintain) the correct set of non-zero variables after a finite number of iterations [Bertsekas, 1982]. At this point, the algorithm reduces to applying an unconstrained Newton method to the non-zero variables.

5 Experiments

To augment our discussion above, we performed a set of four experiments to test the algorithms in various scenarios. We concentrated on the case of logistic regression, since this is the non-quadratic loss that has received the most attention in the literature. The four experiments were:

1. **Logistic**: training a logistic regression classifier for a fixed value of λ, using second-order information.
2. **Logistic Warm-Start**: training a logistic regression classifier for a sequence of values of λ, using second-order information.
3. **Logistic First-Order**: training a logistic regression classifier for a fixed value of λ, using only first-order information.
4. **Softmax**: training a multinomial logistic regression classifier for a fixed value of λ, using second-order information.

We compared the following methods:

1. **Shooting**: A coordinate descent method where we cycle through the variables in order.
2. **Gauss-Seidel**: A coordinate descent method where we optimize the most violating non-zero variable, and if no non-zero variables are violating, we optimize the most violating zero-valued variable.
3. **Grafting**: An active set method we optimize all non-zero variables, then add the most violating non-zero variable to the working set.
4. **Sub-gradient descent**: An active set method where we optimize all non-zero variables and all zero-valued variables that do not satisfy the optimality condition.
5. **Max-k sub-gradient descent**: An active set method where we optimize the union of the non-zero variables and the k variables that are most violating.
6. **Orthant-wise descent**: The projected sub-gradient method where we project the Newton step onto the sign of the steepest descent step, and project the iterates onto the orthant of the previous iterate.
7. **EpsL1**: Applying a differentiable optimization method, where we replace the ℓ_1 norm with the epsL1 approximation.
8. **EM**: Applying the bound optimization algorithm, where at each iteration we replace the ℓ_1 norm with a differentiable convex upper bound.
9. **SmoothL1-SC**: Applying a differentiable optimization method, where we replace the ℓ_1 norm with the smoothL1 approximation.

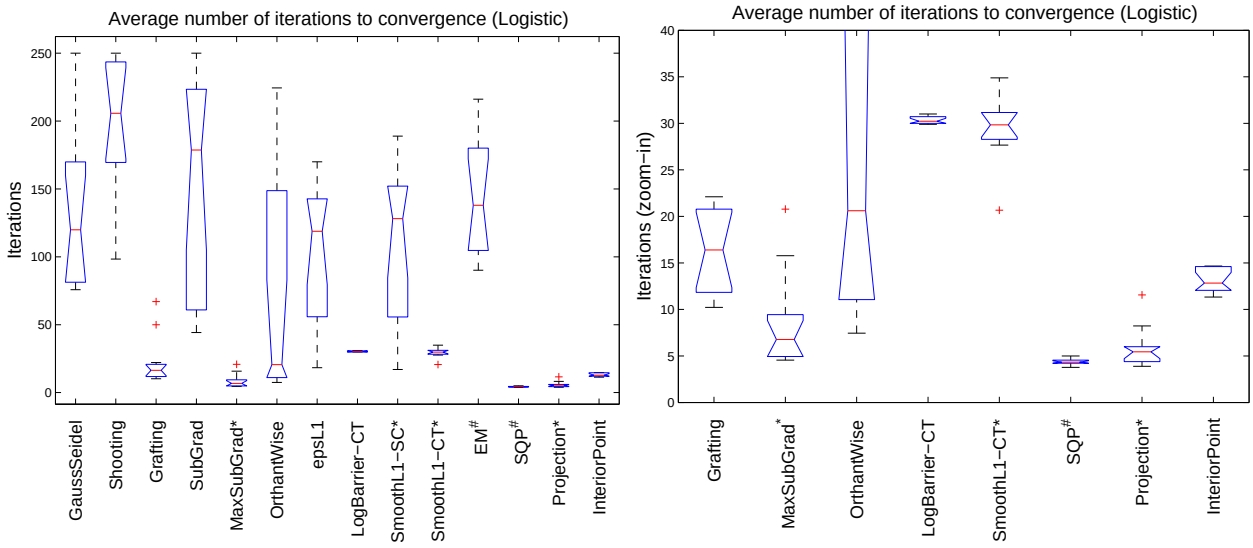

Figure 1: Distribution of function evaluations (averaged over λ) across 12 data sets to train a logistic regression classifier with ℓ_1-regularization. Left: All methods over the full range. Right: Selected methods over a restricted range.

10. **SmoothL1-CT**: Approximately solving a sequence of differentiable optimization problems, where we replace the ℓ_1 norm with the smoothL1 approximation.

11. **Log-barrier**: Approximately solving a sequence of differentiable optimization problems that converge to the constrained formulation.

12. **Interior-point**: Solving the constrained formulation by applying the primal-dual log-barrier method, solving a sequence of perturbed KKT systems and enforcing non-negativity of the primal and dual variables.

13. **SQP**: A projected newton method where we solve the constrained formulation by solving a sequence of quadratic programming problems.

14. **ProjectionL1**: A two-metric projection method applied to the constrained formulation.

For the coordinate descent methods, we used the exact line search of [Shevade and Keerthi, 2003]. All other algorithms are stabilized (to ensure global convergence) by using a back-tracking line search that finds a step length β along the algorithm-dependent step direction $\mathbf{d}$ satisfying the Armijo condition (or an appropriate close variation on this condition):

$$f(\mathbf{x} + \beta\mathbf{d}) \leq f(\mathbf{x}) + \sigma\beta\nabla f(\mathbf{x})^T\mathbf{d}$$

We initialize the line search using $\beta = 1$ (except for the interior point methods where the step is first truncated), and we generated successively smaller values of β using (safeguarded) cubic polynomial interpolation of function and directional derivative values. We set the sufficient decrease parameter σ to 10^{-4}.

We tried to standardize the termination criteria across the methods, stopping a method if any of (i) the norm of the step length between iterations, (ii) the change in function value between iteration, (iii) the negative directional derivative, or (iv) the norm of the optimality conditions, was less than 10^{-6}. We assessed the ability of the methods to optimize the loss function when it was known only through a 'black box' functions that returns the objective value and its derivatives for a given parameter setting. Performance was measured based on the number of times the algorithm invoked this 'black box', and we made a reasonable effort to tune all the methods to optimize this performance measure. We set the maximum number of

iterations allowed by the methods to 250. Since some methods occasionally terminated early but with sub-optimal solutions, we set the iteration count to 250 if the final loss achieved by a method was greater than 10^{-3} times the minimum found across the methods. With the exception of the sub-gradient descent method, this was rarely invoked, since the other methods typically either reached the iteration limit, or terminated with a reasonably accurate solution.

5.1 Logistic Regression

Our first set of experiments focused on the logistic regression negative log-likelihood loss function,

$$L(\mathbf{x}) \triangleq \sum_i \log[1 + \exp(b_i \mathbf{x}^T \mathbf{a}_i)],$$

where $\mathbf{a}_i$ is the set of features for instance i, and b_i is the class label among $\{-1, 1\}$. We applied all 14 optimization strategies to 12 publicly available data sets[5] from the UCI repository[6], Statlog project[7], and the Delve respository[8]. For each data set, we standardized the features so that each feature would have a mean of 0 and a standard deviation of 1. Since the Hessian matrix was not strictly positive-definite for all of these data sets, we solve the Newton systems using a modified LDL^T factorization, modifying the diagonals during the factorization as discussed in [Nocedal and Wright, 1999].

All methods were initialized with $\mathbf{x} = \mathbf{0}$, except the interior point and EM methods that don't allow this (for these methods we used 10^{-2}). Using $\lambda_{max} \triangleq \max_i |\nabla_i L(\mathbf{0})|$ as the maximum value of λ for each data set[9], we evaluated each data set at λ_{max} multiplied by each of $[.1, .2, .3, .4, .5, .6, .7, .8, .9]$ and recorded the average number of function evaluations across these regularization parameter values for each data set before the convergence criteria was met.

In Fig. 1, we plot a summary of the distribution of the average number of function evaluations across the 12 data sets. In the box plot, notches represent a 95% confidence interval of the median value, whiskers are placed at 1.5 times the interquartile range, and the symbol + denotes outliers (data beyond the end of the whiskers). In terms of their performance under this criteria, we can roughly place the methods into four categories:

1. The constrained optimization (log-barrier, interior-point, SQP, and projectionL1), incremental active set methods (grafting and max-k sub-gradient), and the unconstrained continuation method (smoothL1-CT); these methods all converge to a high accuracy solution in a very small number of iterations.

2. The remaining differentiable approximations (epsL1, EM, and smoothL1-SC); these methods also converged to a sufficiently high accuracy solution within the iteration limit, but had a much higher mean and variance.

3. The coordinate descent (shooting, Gauss-Seidel) and sub-gradient descent methods; these methods typically did not achieve a high accuracy solution within the iteration limit.

4. The orthant-wise descent algorithm; this method typically required few iterations, but its distribution was heavily skewed by cases where the algorithm required a large number of iterations before finally terminating.

As an example of the behavior of the various algorithms, in Figure 2, we visualize the objective function value and the number of non-zero variables against the number of function evaluations for the various optimization methods at the mid-point along the regularization path for the UCI 'Sonar' data set, a data set that highlights the problematic aspects of several of the approaches. For increased visibility, the methods have been divided into several groups.

[5] 1: Wisconsin Breast Cancer, 2: Australian Heart, 3: Pima Diabetes, 4: Australian Credit, 5: Sonar, 6: Ionosphere, 7: German, 8: Bright, 9: Dim, 10: Adult, 11: Census, 12: 2Norm

[6] http://www.ics.uci.edu/∼mlearn/MLRepository.html

[7] http://www.liacc.up.pt/ML/old/statlog/

[8] http://www.cs.toronto.edu/ delve/

[9] From the optimality conditions, this and higher values produce $\mathbf{x} = \mathbf{0}$ as their solution.

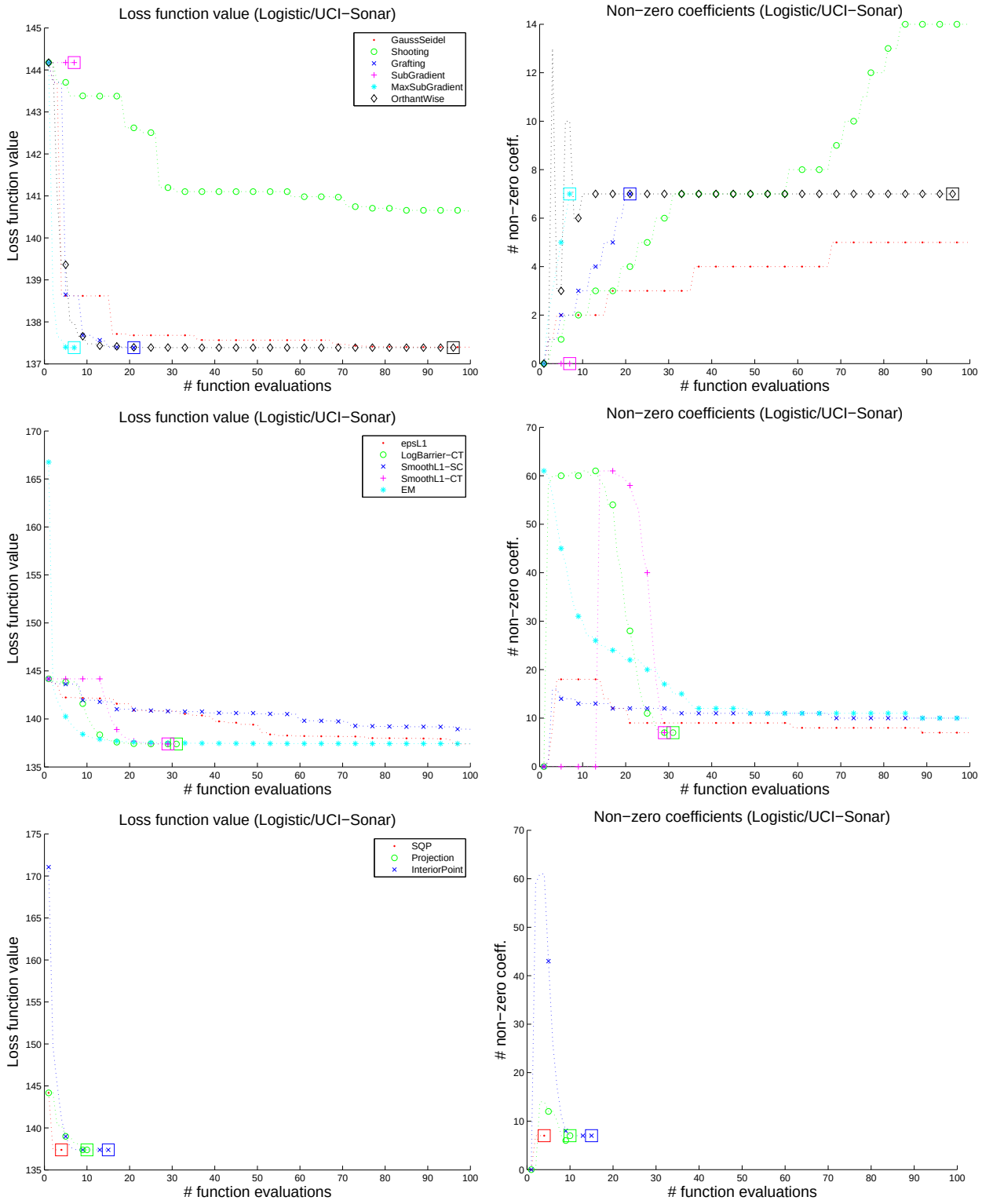

Figure 2: Loss value (left) and number of non-zero coefficients (right) versus function evaluation for training an ℓ_1-regularized logistic regression classifier on the UCI Sonar data set. A box indicates the termination point of the algorithm.

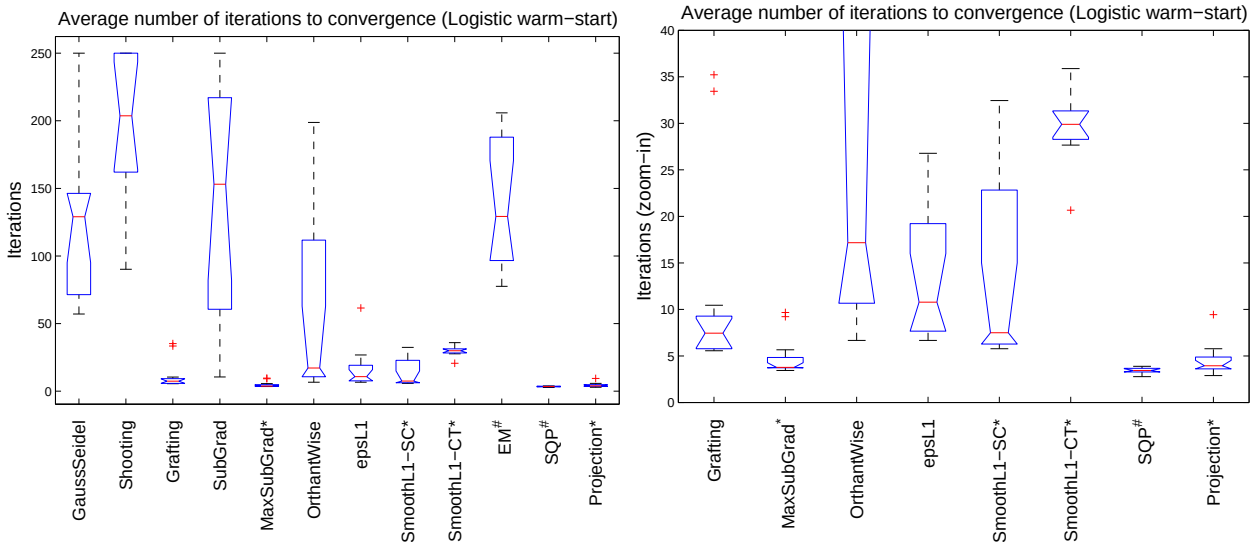

Figure 3: Distribution of function evaluations (averaged over λ) across 12 data sets to train a logistic regression classifier with ℓ_1-regularization, using a 'warm-start' initilization. Left: All methods over the full range. Right: Selected methods over a restricted range.

The first row of Figure 2 shows the sub-gradient methods. In this plot, four of the methods (Gauss-Seidel, shooting grafting, max-k sub-gradient) exhibit a 'staircase' effect, where the number of non-zero coefficients stays constant or increases by at most one variable on each iteration. The Gauss-Seidel method requires the most evaluations to make variables non-zero while the max-k sub-gradient method requires the least. In contrast, the orthant-wise method briefly oscillates between various levels of sparsity, but quickly settles on the correct sparsity pattern. However, slow local convergence meant that the orthant-wise method required a substantial number of iterations before the convergence criteria was met. In the sub-gradient descent method, the initial search direction was not a descent direction, and the algorithm terminated without making any progress.

The second row of Figure 2 shows the differentiable approximations, and the primal log-barrier method. In these plots, we see that the continuation methods initially explore very dense models, but quickly converge to the correct sparse solution. The EM method also initially explores dense models, but does not converge as quickly. Finally, the smoothing methods that don't employ a continuation strategy explore sparse solutions, but also do not converge quickly. The final row of Figure 2 shows the remaining constrained optimization methods. These methods all quickly find a solution, but the interior-point method starts with a dense solution, while the SQP and projectionL1 methods only explore sparse solutions.

5.2 Warm-Starting

Many practical scenarios require finding the optimal parameters for a sequence of λ values. In this scenario, we may be able to improve performance using a 'warm-start' strategy, where we use solution of one optimization problem to initialize a related problem. To assess the performance of the methods at this task, we repeated the previous experiment, but instead of initializing at $\mathbf{x} = \mathbf{0}$, we initialized the methods at the optimal solution for the next largest value of λ. We did not include the log-barrier and interior-point methods in this experiment, as this initialization is typically not within the interior of the constraints. We plot the results of this experiment in Figure 3.

In most cases, the warm-start decreased the number of iterations required. One exception was the smoothL1-CT method, whose continuation strategy makes the warm-start largely irrelevant. The biggest difference was observed in the smoothing methods (epsL1 and smoothL1-SC), that had a drastic reduction in the number of iterations. We believe that this is because varying λ has a similar effect to changing the barrier/smoothing parameter in a continuation method. Suprisingly, the sub-gradient descent algorithm still

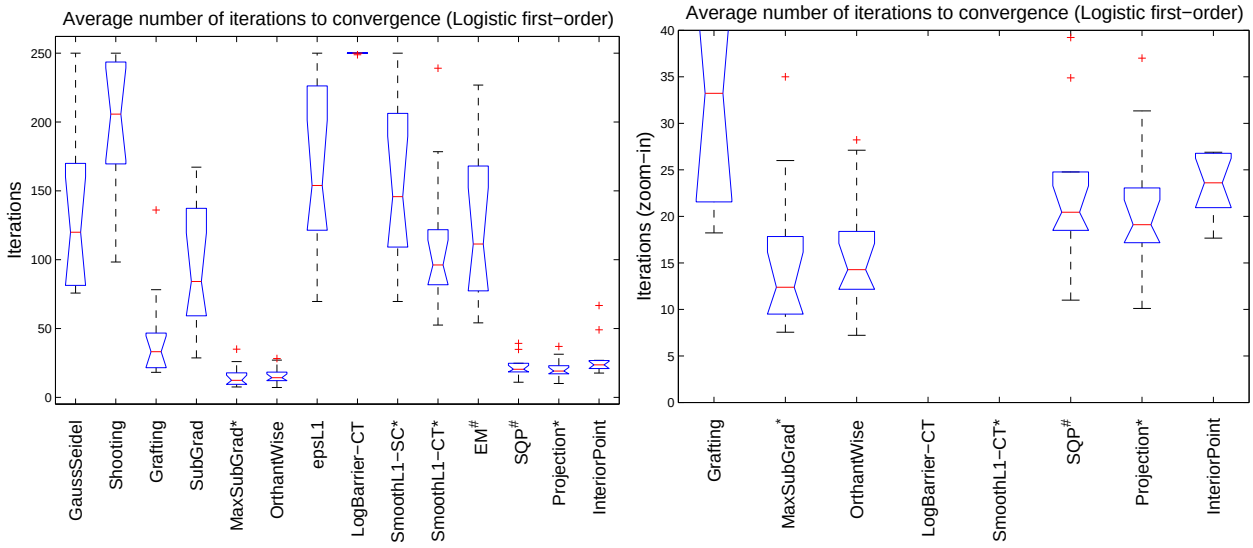

Figure 4: Distribution of function evaluations (averaged over λ) across 12 data sets to train a logistic regression classifier with ℓ_1-regularization, using a BFGS Hessian approximation. Left: All methods over the full range. Right: Selected methods over a restricted range.

performed poorly (despite being proposed for use in the warm-start scenario). To confirm that this was not due to an error in our implementation, we confirmed that for smaller changes in λ, the sub-gradient descent algorithm only requires a small number of iterations.

5.3 First-Order Approximation

Our assumption that we can analytically compute and store the Hessian is often not reasonable. Our third experiment thus examined the performance of the methods when the exact Hessian is replaced by a BFGS approximation [see Nocedal and Wright, 1999]. Although this may slow the convergence rate, this type of approximation typically leads to cheaper iterations. While methods that (implicitly) invert the Hessian require $O(p^3)$ in terms of p (the number of variables) to solve the Newton system, the BFGS approximation can be implemented to require $O(p^2)$, while the limited-memory BFGS approximation only requires $O(mp)$ (where m is user-specified) [Nocedal and Wright, 1999]. For the coordinate descent methods in this experiment, we used numerical differentiation with complex differential (rather than a BFGS approximation) to calculate the needed second-order terms to a high degree of accuracy (this process adds no function evaluations) [Vishwanathan et al., 2006].

The results of this experiment are plotted in Figure 4. The active set methods (grafting, max-k sub-gradient) and several of the constrained optimization methods (SQP, interior-point, projectionL1) also had strong performance when using only first-order information. However, the continuation methods (smoothL1-CT and log-barrier) were far less effective under the BFGS Hessian approximation. The other differentiable approximation strategies also saw a degradation in performance, while the coordinate descent methods were unaffected. One of the most interesting aspects of this experiment was that two methods had *better* performance under the BFGS approximation; the sub-gradient descent method typically converged to a high accuracy solution under this approximation, while the orthant-wise descent algorithm proved to be among the strongest methods.

5.4 Multinomial Logistic Regression

Our final experiment examined the performance of the methods for optimizing a multinomial logistic regression negative log-likelihood, subject to ℓ_1-regularization. In particular, we use the softmax function to define

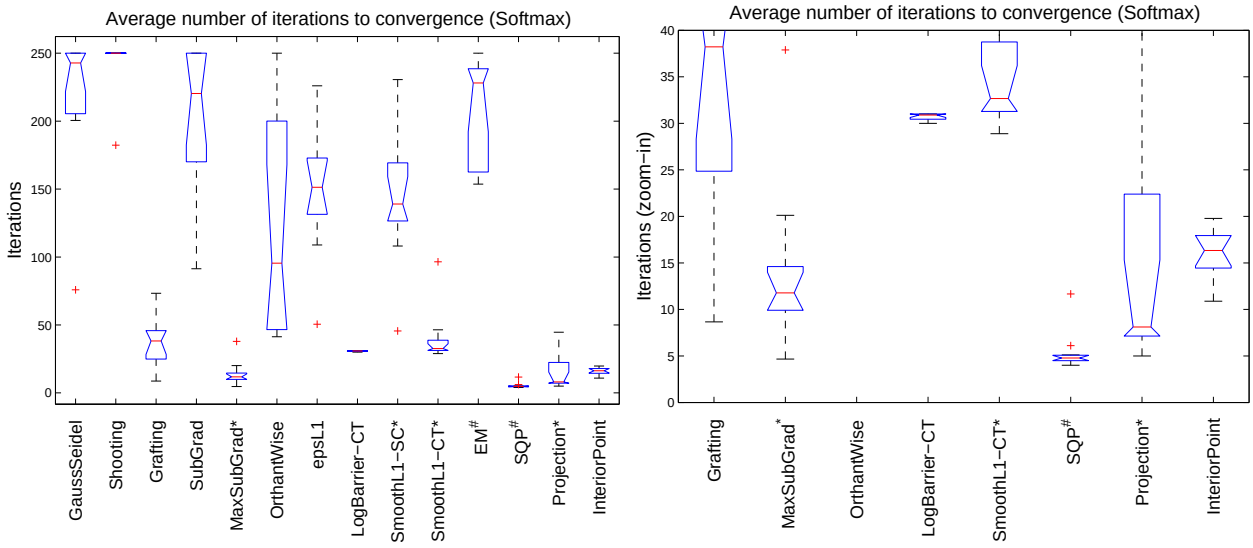

Figure 5: Distribution of function evaluations (averaged over λ) across 12 data sets to train a multinomial logistic regression classifier with ℓ_1-regularization. Left: All methods over the full range. Right: Selected methods over a restricted range.

a K-class logistic regression model,

$$p(b_i = k|\mathbf{a}_i, \mathbf{x}) = \frac{\exp(\mathbf{x}_k^T \mathbf{a}_i)}{\sum_{k'=1}^{K} \exp(\mathbf{x}_{k'}^T \mathbf{a}_i)}$$

Here, we have a vector $\mathbf{x}_k$ for each class k, and this typically represents a more challenging scenario than the binary case, since the objective function is typically more ill-conditioned. We set the weight vector of one class to $\mathbf{0}$ to avoid over-parameterization. Using the negative log of this likelihood as the loss function, we trained an ℓ_1-regularized multinomial logistic regression on 11 data sets[10] from the UCI repository and Statlog project. The results of this experiment are summarized in Figure 5. Although the mean and variance of the number iterations was typically higher for this loss function than for the binary loss function, we see that the same trends are still evident.

6 Discusssion

A limitation of our empirical evaluation is that we have measured the performance of the methods in terms of 'function evaluations, which we used as a surrogate for runtime comparisons in order to make our comparisons largely independent of low-level implementation details and of the actual hardware architecture used in performing the experiments. A drawback of this is that our experiments do not take into account the iteration cost, which varies substantially across the methods (ie. the SQP method has a very expensive iteration, while the shooting method has a trivial iteration cost). In Table 1, we give a summary of the empirical performance of the methods across our four experiments, alongside a measure of the iteration cost (referred to as *Iter Cost* in the Table). We divide the methods by iteration cost into the following six categories, from fastest to slowest.

1. Methods that modify a single variable at a time.

2. Methods that greedily add variables to the working set, and must solve a linear system involving the working set variables.

[10] Iris, Glass, Wine, Vowel, Vehicle, LED, Satellite, Waveform21, DNA, Waveform40 and Shuttle

Method	Iter Cost	Dense	Logistic	Warm	BFGS	Softmax
GaussSeidel	1		*	*	*	*
Shooting	1		*	*	*	*
Grafting	2		2	2	3	4
SubGradient	3		*	*	4	*
MaxSubGradient	2		1	1	1	2
OrthantWise	3		4	4	1	*
epsL1	3		5	2	*	5
EM	3	Yes	6	5	6	*
SmoothL1-SC	3		5	2	*	5
SmoothL1-CT	3	Yes	3	3	5	4
LogBarrier-CT	4	Yes	3	**	*	3
InteriorPoint	5	Yes	1	**	2	2
SQP	6		1	1	1	1
ProjectionL1	4		1	1	1	1

Table 1: Summary of properties and empirical performance of methods. See the text for descriptions of the 'Iter Cost' and 'Dense' fields. The rankings on the four experiments are determined by comparing the mean number of iterations to convergence across the data sets, with methods whose means are within 5 iterations receiving the same ranking (the fastest methods are ranked 1). *: method has an average iteration count of 250 on at least one data set. **: method was not applied in this experiment.

3. Methods that must solve a linear system involving all variables.
4. Methods that must solve a linear system involving double the number of variables present in the original problem.
5. Methods that must solve a linear system involving four times the number of variables present in the original problem.
6. Methods that must solve a quadratic program.

Since methods with higher iteration costs tend to have faster convergence rates, for a given loss function we must decide on the correct balance between iteration cost and convergence rate. Another factor that our experiments do not consider is that some methods must evaluate the function with a dense vector, that is, a vector where all variables are non-zero. Our evaluation thus does not reflect that some methods may be able to take advantage of the fact that they maintain a sparse solution on all iterations. For this reason, in Table 1 we also included a column called *Dense*, where the four methods that must evaluate dense parameter vectors are noted. Another limitation of our experiments was that the solutions in most cases were extremely sparse. Because of this sparsity, our experiments do not reflect that the incremental active set set methods (grafting and max-k sub-gradient descent) degrade in performance when the number of non-zero variables increases.

Beyond replacing the exact Hessian with a BFGS approximation, we could further reduce the memory requirements and iteration cost by using a limited-memory BFGS approximation [see Nocedal and Wright, 1999]. Another simple variation would be to use Hessian-free Newton methods. [Lin et al., 2007] have recently shown that these can be effective for logistic regression with ℓ_2-regularization. These methods require Hessian-vector products, and approximations of these products can readily be obtained even when the Hessian is infeasible to compute [see Nocedal and Wright, 1999, Vishwanathan et al., 2006].

We have made the code used to produce our results available on the web[11]. In addition to allowing reproduction of our results, this code can be applied to different data sets and loss functions (we have included an L-BFGS version of the projectionL1 method that we have applied to problems with millions of variables), and will allow new algorithms or improvements to existing algorithms to be easily assessed.

[11] `http://pages.cs.wisc.edu/~gfung/GeneralL1/`

Other Methods

A large variety of additional methods for solving the problem discussed here have also appeared in the literature. Although most of these represent variations on the methods discussed above, several of these methods take notably different approaches, including some that were proposed after the original version of this report was written. In this section, we briefly comment on several of the methods that do not fit as neatly among the methods discussed above, and we comment on some preliminary experiments we have performed with the methods.

- **Path Following with Coordinate Descent**: Several works have explored computing solutions for a sequence of λ values by taking a small fixed-length step in the steepest descent direction for one (or two) of the most violating variables. An example of this type of method is the 'boosted Lasso' [Zhao and Yu, 2004]. We have found that in some cases this procedure can be more efficient than the coordinate optimization methods, but suffers from the same drawbacks as these methods when applied to ill-conditioned problems. Further, selecting the step size can be problematic; if the step size is too large then the algorithm will do a poor job of following the path, while if the step size is too small it may require too many steps.

- **Block Coordinate Descent**: [Sardy et al., 2000] consider a generalization of the coordinate optimization method where, instead of optimizing a single coordinate at a time, you optimize a block of coordinates. They apply this to the least squares loss, where the design matrix is a union of orthogonal basis matrices, leading to efficient closed-form solutions for the updates of entire blocks. In the general case, we could consider an algorithm where we select the k most violating variables and optimize these variable as a block.

- **Spectral Projected Gradient**: The use of the Barzilai-Borwein step length and a non-monotonic Armijo line search within a projected gradient method is known as the spectral projected gradient (SPG) method [Birgin et al., 2000]. The Barzilai-Borwein step length can be viewed as a quasi-Newton method that uses a much simpler Hessian approximation than the BFGS approximation. Applying SPG to the constrained formulation was explored in [Figueiredo et al., 2007]. We have found that SPG requires more iterations to converge than the related approach of using L-BFGS within the projectionL1 algorithm, but the iterations of SPG are slightly more efficient. SPG has also been explored for the related problem of minimizing a loss function subject to a constraint on the ℓ_1 norm of the solution [van den Berg and Friedlander, 2008], taking advantage of algorithms that efficiently project onto the ℓ_1 'ball'.

- **Iterative Soft-Thresholding**: An approach that has been explored by several authors consists of the following simple iteration: take a step in the negative gradient direction of $L(\mathbf{x})$, and then apply the soft-thresolding operator to the result. One of the most efficient methods of this type is [Wright et al., 2008], which incorporates the Barzilai-Borwein step length into this framework. We have found that this gives similar performance to applying SPG to the constrained formulation, but the iterative soft-thresholding method has the advantage that it does not need to double the number of variables.

- **Optimal Gradient**: In this work, we have focused our discussion on convergence rates under the usual assumptions associated with Newton-like methods, namely (local) strong convexity. In [Nesterov, 2007], Nesterov considers the case of worst case complexity (ie. how does the algorithm behave for the *worst* function in its class). Nesterov proposes a method that achieves the optimal worst-case behaviour for a first-order method. Although theoretically appealing, our preliminary experiments with the method indicate that it requires more iterations than the fastest methods examined in this work.

- **Stochastic Approximation**: Many loss functions can be interpreted as a set of independent samples from a distribution (ie. least squares, logistic regression, etc.). When the number of such samples (ie. rows of A in the least squares problem $\min_{\mathbf{x}} ||A\mathbf{x}-\mathbf{b}||_2^2$) is enormous, it may be economical to consider a method that updates the parameter vector after calculating the loss function on each row. Algorithms of this type generally have slow (ie. sub-linear) convergence rates even for well-behaved functions. However, the iterations under such a scheme are inexpensive, and when the number of samples is

enormous and the problem is sufficiently well-conditioned, such an approach can outperform methods with better convergence properties but that require evaluating the function over all samples at each iteration (ie. all the methods discussed in this paper). Among the many recent examples of methods of this type is [Langford et al., 2008], who apply an iterative soft-thresholding algorithm in the stochastic approximation scenario.

- **Predictor-Corrector Path Following**: Some authors have considered more efficient ways to 'warm-start' when solving for a sequence of values of λ. Rather than simply using the parameter vector from the previous value in the sequence, [Park and Hastie, 2007] consider calculating the related rate $\partial \mathbf{x}/\partial \lambda$, and using a linear approximation to 'predict' the solution for the next value of λ (applying the optimizer to find a precise solution is then referred to as the 'correction' step). In our preliminary experiments, we found that the additional cost of performing the predictor step was similar to the gains achieved by the optimizer given the better warm-start. Nevertheless, developing improved methods of following the regularization path is an interesting direction of future research.

- **Projected Scaled Sub-Gradient**: Over the past several months, we have been using an algorithm that combines ideas from sub-gradient methods and the two-metric projection algorithm. The problem with the sub-gradient methods discussed in this report is that we either need to use an active set strategy that can only add/remove a small number of variables on each iteration (ie. grafting and the max-k sub-gradient descent method), or we must abandon the property that our algorithm eventually reduces to Newton iterations on the non-zero variables (ie. the orthant-wise descent method). In contrast, the two-metric projection method can quickly change the active set and reduces to Newton's method when we identify the correct set of non-zero variables, but it has the disadvantage that we must solve a problem with double the number of variables, and the Hessian of the transformed problem is singular. The new method is a sub-gradient method that seeks to avoid all of these disadvantages. It consists of three simple components: (i) we use the Newton direction on the non-zero variables, (ii) we use the steepest descent direction on the zero-valued variables, and (iii) we perform the $\mathcal{P}_{\mathcal{O}}$ orthant projection of the iterates. The combination of (i) and (ii) allow us to quickly add variables to the working set and guarantees that the method generates descent directions, while (iii) allows us to quickly remove variables from the working set. In our preliminary experiments, this method is among the most efficient methods in all the scenarios examined in this paper.

References

G. Andrew and J. Gao. Scalable training of l1-regularized log-linear models. In *ICML*, pages 33–40, New York, NY, USA, 2007. ACM Press.

D. Bertsekas. Projected newton methods for optimization problems with simple constraints. *SIAM J. Control Optim.*, 20:221–246, 1982.

D. Bertsekas. *Nonlinear programming*. Athena Scientific, 2004.

E. Birgin, J. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. on Optimization*, 10(4):1196–1211, 2000.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1999.

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

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

M. Figueiredo. Adapative sparseness for supervised learning. *IEEE. Trans. Pattern. Anal. Mach. Intell.*, 25(9): 1150–1159, 2003.

M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586, 2007.

R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

W. Fu. Penalized regressions: The bridge versus the LASSO. *J. Comput. Graph. Stat.*, 7(3):397–416, 1998.

E. Gafni and D. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM J. Contr. Optim.*, 22(6):936–964, 1984.

Y. Grandvalet and S. Canu. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. In *NIPS*, pages 445–451, 1998.

S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. Selected Topics in Signal Processing. *IEEE Journal of*, 1(4):606–617, 2007.

K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 8:1519–1555, 2007.

B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Learning sparse bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE. Trans. Pattern. Anal. Mach. Intell.*, 2005.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Advances in Neural Information Processing Systems*, 21, 2008.

S.-I. Lee, H. Lee, P. Abbeel, and A. Ng. Efficient L1 regularized logistic regression. In *AAAI*, 2006.

Y.-J. Lee and O. L. Mangasarian. SSVM: A smooth support vector machine. *Comput. Optim. Appl.*, 20:5–22, 2001.

C. Lin, R. Weng, and S. Keerthi. Trust region Newton methods for large-scale logistic regression. *Proceedings of the 24th international conference on Machine learning*, pages 561–568, 2007.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, 2007.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

M. Osborne, B. Presnell, and B. Turlach. On the LASSO and its dual. *J. Comput. Graph. Stat.*, 9:319–337, 2000.

M. Park and T. Hastie. L1-regularization path algorithm for generalized linear models. *J. Roy. Stat. Soc. B*, 69(4): 659–677, September 2007.

S. Perkins, K. Lacker, and J. Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *J. Mach. Learn. Res.*, 3:1333–1356, 2003.

S. Rosset. Following curved regularized optimization solution paths. In *NIPS*, 2004.

V. Roth. The generalized LASSO. *IEEE Trans. Neural Networks*, 15(1), January 2004.

S. Sardy, A. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries. *J. Comput. Graph. Stat.*, 9:361–379, 2000.

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

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc. B*, 58(1):267–288, 1996.

E. van den Berg and M. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

S. Vishwanathan, N. Schraudolph, M. Schmidt, and K. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. *Proceedings of the 23rd international conference on Machine learning*, pages 969–976, 2006.

S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008.*, pages 3373–3376, 2008.

P. Zhao and B. Yu. Boosted lasso. Technical report, University of California, Berkeley, 2004.