Least Squares Optimization with L1-Norm Regularization

Mark Schmidt
CS542B Project Report
December 2005

Abstract

This project surveys and examines optimization approaches proposed for parameter estimation in Least Squares linear regression models with an L1 penalty on the regression coefficients. We first review linear regression and regularization, and both motivate and formalize this problem. We then give a detailed analysis of 8 of the varied approaches that have been proposed for optimizing this objective, 4 focusing on constrained formulations and 4 focusing on the unconstrained formulation. We then briefly survey closely related work on the orthogonal design case, approximate optimization, regularization parameter estimation, other loss functions, active application areas, and properties of L1 regularization. Illustrative implementations of each of these 8 methods are included with this document as a web resource.

1 Introduction

This report focuses on optimizing on the Least Squares objective function with an L1 penalty on the parameters. There is currently significant interest in this and related problems in a wide variety of fields, due to the appealing idea of creating accurate predictive models that also have interpretable or parsimonious representations. Rather than focus on applications or properties of this model, the main contribution of this project is an examination of a variety of the approaches that have been proposed for parameter estimation in this model.

The bulk of this document focuses on surveying 8 different optimization strategies proposed in the literature for this problem. Each will be examined in detail, focusing on their comparative strengths and weaknesses, and how each deals with a function that has non-differentiable points. After examining these methods in detail, we will briefly discuss several closely related issues. These include optimization in the special case of an orthogonal design matrix, approximate optimization for large problems, selecting appropriate values for the regularization parameter, optimizing other loss functions subject to an L1 penalty, several active application areas, and finally we discuss several interesting properties of L1 regularization.

However, we will first briefly review the linear regression problem, the Least Squares approach, L2 and L1 penalties, and finally formalize the problem being examined. We do this review in order to motivate the interest in this model, and to introduce the notation that will be used consistently throughout the report. We emphasize the latter point because much of the related work was produced in different areas, and the notation used is not consistent across works.

1.1 Linear Regression

In the typical univariate linear regression scenario, the input is n instances of a process described by $p+1$ variables, and we seek to find a linear combination of a selected p variables (the features) that best predicts the remaining variable (the target). We typically define the design matrix X as a matrix having n rows and p columns, representing the p variables for n instances. Similarly, we define the target vector y as a column vector of length n containing the corresponding values of the target variable. The problem can then be formulated as finding a 'good' value for the length p coefficient vector w and the scalar bias term w_0 in the following model (taken over i from 1 to n):

$$y_i = w_0 + \sum_{j=1}^{p} x_{ij} w_j + \epsilon_i \tag{1}$$

ϵ_i denotes the noise (or error) term for instance i, and it is typically assumed that the values of ϵ_i have zero mean, and are both independently and identically distributed. Thus, we typically do not model ϵ directly. However, we cannot solve for w and w_0 directly given only X and y, because ϵ is unknown. In addition, the noise term may cause the same or similar values of x_i to yield different values. We thus define a 'loss' function that assesses how well a given set of parameters w predicts y from X.

1.2 Least Squares

By far the most popular loss function used for regression problems the Least Squares estimate, alternately referred to as minimizer of the residual sum of squared errors (RSS) [1]:

$$RSS = \sum_{i=1}^{n}(y_i - w_0 - \sum_{j=1}^{p} x_{ij}w_j)^2 \tag{2}$$

We can remove the need to write w_0 by appending a column vector of 1 values to X and increasing the length w by one. This lets us write (2) more elegantly in matrix form:

$$RSS = ||Xw - y||_2^2 \tag{3}$$

The Least Squares estimate is defined as the w that minimizes this expression. This minimization forms a continuously differentiable unconstrained convex optimization problem. Differentiating (3) with respect to w (and dividing by 2) we obtain:

$$X^T(Xw - y) \tag{4}$$

Since we will need it later, we see that the Hessian of (3) is simply:

$$X^T X \tag{5}$$

We can obtain the *normal equations* by equating (4) with 0 and solving for w [1]:

$$w = (X^T X)^{-1} X^T y \tag{6}$$

However, the matrix $X^T X$ may be singular (or nearly singular), making it difficult to invert. Iterative solvers or the pseudoinverse can be used to overcome this problem, however these methods do not overcome several problems associated with using the RSS formulation itself.

1.3 L2 Regularization

The instability of minimizing the RSS can be illustrated with an example. Consider the case where two of the variables are highly correlated. This allows the w_i value associated with one variable to grow very large in the positive direction, while the other grows appropriately large in the negative direction cancelling the first variable's effect [1]. This yields a high variance model that becomes an increasingly unrealistic model as the correlation increases. These high variance models can produce wildly differing w values (and subsequent predictions) given different instances, or even minor perturbations of the same instances. Thus, the final result of producing high variance models is directly related to the potential numerical instability of the Least Squares procedure.

Tikhonov regularization addresses the numerical instability of the matrix inversion and subsequently produces lower variance models. This method adds a positive constant to the diagonals of $X^T X$, to make the matrix non-singular [2]. The analytic solution then becomes:

$$w = (X^T X + \lambda I)^{-1} X^T y \tag{7}$$

If we replace one diagonal value of the I matrix with zero (corresponding to w_0), it is easy to show that this minimizes the following penalized RSS function with respect to w and w_0:

$$\sum_{i=1}^{n}(y_i - w_0 - \sum_{j=1}^{p} x_{ij}w_j)^2 + \lambda \sum_{j=1}^{p} w_j^2 \tag{8}$$

Examining the form of (8), we see that this is an unconstrained continuously differentiable convex optimization problem, and that it is the RSS with a penalty term proportional to the squared values of w (except w_0). This is referred to as L2 regularization. We do not penalize w_0 since we do not want the model to depend on the mean of the y vector. However, in order to simplify both the notation used in this work and in the associated implementations, we *center* the data by setting w_0 to $\bar{y} = \sum_1^n y_i/n$, and work with a *centered* target vector (as in [1]), reducing to the following problem (in matrix notation):

$$||Xw - y||_2^2 + \lambda||w||_2^2 \tag{9}$$

[2] provides several intuitive motivations for this form of regularization: (i) In terms of statistics, it can be interpreted as prior knowledge that w should not be too large, (ii) in terms of optimal design, it can be interpreted as balancing large values of the variables compared to meeting the target, (iii) in terms of prediction, large w values cause large variations in Xw (when applying the model to new instances), and may not achieve high predictive performance (iv) in terms of optimization, it gives a compromise between solving the system and having a small w.

1.4 L1 Regularization

While L2 regularization is an effective means of achieving numerical stability and increasing predictive performance, it does not address another problem with Least Squares estimates, parsimony of the model and interpretability of the coefficient values. While the size of the coefficient values is bounded, minimizing the RSS with a penalty on the L2-norm does not encourage sparsity, and the resulting models typically have non-zero values associated with all coefficients. It has been proposed that, rather than simply achieving the goal of 'shrinking' the coefficients, higher λ values for the L2 penalty force the coefficients to

be more similar to each other in order to minimize their joint 2-norm [3]. A recent trend has been to replace the L2-norm with an L1-norm. This L1 regularization has many of the beneficial properties of L2 regularization, but yields sparse models that are more easily interpreted [1].

An additional advantage of L1 penalties is that the models produced under an L1 penalty often outperform those produced with an L2 penalty, when irrelevant features are present in X. This property provides an alternate motivation for the use of an L1 penalty. It provides a regularized feature selection method, and thus can give low variance feature selection, compared to the high variance performance of typical subset selection techniques [1]. Furthermore, this does not come with a large disadvantage over subset selection methods, since it has been shown that Least Squares with an L1 penalty comes as close as subset selection techniques do to an ideal subset selector [3].

1.5 Unconstrained Formulation

Replacing the squared values in (9) with the L1 norm yields the following expression:

$$||Xw - y||_2^2 + \lambda||w||_1 \tag{10}$$

It is clear that this remains an unconstrained convex optimization problem in terms of w. However, this problem is now non-differentiable when $w_i = 0$ for any w_i. Thus, we cannot obtain a closed form solution for the global minimum in the same way that is done with the L2 penalty. This drawback has led to the recent introduction of a multitude of techniques for determining the optimal parameters. Several of these algorithms directly use the above unconstrained optimization problem, while other techniques use equivalent constrained formulations.

1.6 Constrained Formulation

The most straightforward method to represent (10) as a constrained optimization problem is as follows (note that t is inversely related to λ):

$$\begin{aligned} &||Xw - y||_2^2 \\ &s.t. ||w||_1 \leq t \end{aligned} \tag{11}$$

The objective function in this minimization is convex, and the constraints define a convex set. Thus, this forms a convex optimization problem. From this, we know that any local minimizer of the objective subject to the constraints will also be global minimizer. As we will review shortly, the non-differentiable constraints can be converted into a set of linear constraints, and thus the feasible region forms a polyhedron. Combining this with the property that the objective is quadratic in w, we see that the problem can be interpreted as a Quadratic Programming problem.

2 Least Squares Optimization with L1 Regularization

Although it is likely that it had been explored earlier, estimating Least Squares parameters subject to an L1 penalty was presented and popularized independently under the names *Least Absolute Selection and Shrinkage Operator* (LASSO) in [3] and *Basis Pursuit Denoising* [4]. The acronym for the former has become the dominant expression describing this problem, and for the remainder of the paper we will use the term LASSO to denote the RSS problem with L1 regularization. [3] presented several different methods for optimizing the LASSO, each of which differed significantly from the method used in [4]. Since these original works, there have been a wide variety of approaches proposed for the LASSO minimization problem. Many have simply been minor variations on methods already proposed, but many offer unique ways of viewing the problem. This section will examine in detail 8 different ways to compute the LASSO estimate. These 8 methods were selected to represent very different approaches to computing the LASSO estimate, and includes both the most influential works that are not minor variations on previous works, and several less influential but unique works (in our opinion). We think that these works represent a substantial sample of the different approaches that have been proposed for circumventing the non-differentiability of the problem. We will first overview 4 methods for optimizing constrained formulations. We focus on these approaches first primarily for historical reasons, since 3 of these types of approaches were used by the authors of the original LASSO and Basis Pursuit Denoising works. We then turn our focus to 4 methods that directly minimize the unconstrained formulation, 2 of which are much more recent contributions.

2.1 Quadratic Programming, Sequentially Added Sign Constraints

The influential work in [3] that proposed the acronym LASSO presented two methods to convert the constraints in the constrained formulation (11) into a a set of linear constraints. We will first examine the method recommended by the authors.

A naive method to represent the constraints in (11) as a set of linear equalities is to make one linear constraint for each of the possible combinations of the signs of the elements of w. This is best illustrated by example, below we show the constraints required for 3 variables:

$$+w_1 + w_2 + w_3 \leq t$$

$$
\begin{aligned}
+w_1 + w_2 - w_3 &\leq t \\
+w_1 - w_2 + w_3 &\leq t \\
+w_1 - w_2 - w_3 &\leq t \\
-w_1 + w_2 + w_3 &\leq t \\
-w_1 + w_2 - w_3 &\leq t \\
-w_1 - w_2 + w_3 &\leq t \\
-w_1 - w_2 - w_3 &\leq t
\end{aligned}
$$

It is clear that any minimizer of the RSS subject to these constraints will also be the minimizer of (11). However, it is also clear that this forms a Quadratic Programming problem with 2^p constraints. Thus, we cannot even tractably iterate over all the constraints generated by this expansion for non-trivial values of p (for example, $p = 50$). To avoid this major drawback, Tibshirani proposed the following algorithm (where LS(X,y) is the solution to the unpenalized Least Squares problem):

Algorithm 1 Tibshirani's Method

1: $w = LS(X, y)$
2: $constraints = \{null\}$
3: **while** $||w||_1 \leq t$ **do**
4: add $sign(w)$ to $constraints$
5: $w = min_w ||Xw - y||_2^2$ subject to $constraints$
6: **end while**

At termination, this algorithm must reach a solution that satisfies the constraints (in order to exit the $while$ loop). Furthermore, we see that all possible constraints in the expansion must be satisfied, even those that were not introduced. This algorithm therefore achieves the optimal solution subject to the constraints. It is natural to ponder the worst case performance of this approach. If we consider the case where $t = 0$, then this approach will need to introduce $O(2^p)$ constraints, and subsequently needs to solve $O(2^p)$ Quadratic Programming problems. Although this case seems pathological and the optimal solution when $t = 0$ is $w = 0$, in practice we may still be interested in very small values of t.

Fortunately, we can make a slight modification to this algorithm to dramatically improve its performance. The modification that we can make is that rather than testing whether $||w||_1 \leq t$, we can test whether $||w||_1 \leq t + \epsilon$ for some small positive ϵ. Since the problems becomes more restrictive at each iteration, the value of $||w||_1$ shrinks and this minor modification requires us to solve a significantly smaller number of Quadratic Programs. Although this modification is not discussed in [3], it is obvious that the authors must use this or a similar modification in order to achieve their stated result that the algorithm converges in approximately $0.5p$ to $0.75p$ iterations. Note that our empirical results found that a slightly larger number of iterations (although still linear in terms of p) was required to achieve an acceptable level of precision.

Despite immensely reducing the iteration count and number of constraints required by making this simple modification, this method still has one obvious weakness. We still need to solve a potentially large number of Quadratic Programming problems to obtain the solution to the original Quadratic Programming problem. Furthermore, the solution from the previous iteration is typically not well-suited for use in initializing the current iteration. This first reason that the previous solution is not well-suited is that the solution to the previous iteration is by definition infeasible for the current iteration's constraints, so we must expend effort finding a feasible point. Although finding a 'close' feasible point is trivial, the second reason that the previous iteration's solution may not be a good starting point is that the values (initially) change wildly between iterations. Once a sign constraint is introduced, it is common to observe that variables change wildly from highly to positive to highly negative or vice versa (especially in the case of correlated variables).

2.2 Quadratic Programming, Non-Negative Variables

The second method proposed in [3] to represent the constraint in (11) as a set of linear equalities avoids the need to solve more than 1 Quadratic Program. This approach uses a modified design matrix $\bar{X} = [X, -X]$ and represents each variable w_i as the difference between two non-negative variables (corresponding to the positive and negative components of w_i):

$$w_i = w_i^+ - w_i^-$$

If w_i is positive, then $w_i^+ = w_i$ and $w_i^- = 0$, if w_i is negative then $w_i^- = w_i$ and $w_i^+ = 0$, and if w_i is zero then $w_i^+ = w_i^- = 0$. These identities can be enforced by forcing both variables to be non-negative introducing the equality constraint $|w_i| = w_i^+ + w_i^0$. However, this latter constraint proves unnecessary since we only require the positive and negative components to satisfy an upper bound on $||w||_1$.

We can use this representation to create an equivalent form of (11) that has $2p$ variables, $2p$ constraints enforcing non-negativity and 1 constraint enforcing that the positive and negative components satisfy the original constraint. For the case of 3 variables, this would involve the following constraints:

$$
\begin{aligned}
w_1+ &\geq 0 \\
w_1- &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
w_2+ &\geq 0 \\
w_2- &\geq 0 \\
w_3+ &\geq 0 \\
w_3- &\geq 0 \\
\sum_{i=1}^{3}[w_i^+ + w_i^-] &\leq t
\end{aligned}
$$

Although this method doubles the number of variables in the problem, it requires only a (known and bounded) linear number of constraints, and only requires the solution to 1 Quadratic Programming problem. However, note that these constraints are degenerate, and thus the Quadratic Programming solution used to solve this problem must be handle degenerate constraints.

Tibshirani states that this method is generally faster than the method presented earlier, but not always. Our empirical experiments support this conclusion. The non-negative variable approach is clearly faster when t is close to 0, and is generally faster over a large range of the values that t can take. However, when t becomes sufficiently loose that the optimal solution closely resembles the Least Squares solution (and degeneracy of the constraints has a larger influence), Tibshirani's method might offer an advantage.

2.3 Interior Point Method, Non-Negative Variables with Log Barrier

Basis Pursuit Denoising was proposed in [4], and the equivalence between the LASSO and Basis Pursuit Denoising was originally recognized in [5]. We follow the derivation of the algorithm proposed in [4], based on the more elegant presentation of the approach given in [5], stating this method in terms of our own notation. To (hopefully) avoid confusion, note that in this section we will use λ to represent the dual variables, and t to represent the L1 penalty parameter, even though this method actually is supposed to be parametrized in terms of λ.

The method of [4] also operates on non-negative variables w^+ and w^- and thus operates on $2p$ non-negative variables. Note also that we must again double the size of our design matrix and represent it as $\bar{X} = [X, -X]$. However, in this case we reformulate our optimization problem into the following equivalent form (with $\bar{1}$ being a vector of ones):

$$
\begin{aligned}
min_{w,\lambda} \frac{1}{2}||\lambda||_2^2 + t\bar{1}^T w \\
s.t.\ \lambda = y - \bar{X}w
\end{aligned}
\tag{12}
$$

Now, to deal with the non-negativity constraint on the variables, we use a log-barrier penalty with a positive parameter μ:

$$
\begin{aligned}
min_{w,\lambda} \frac{1}{2}||\lambda||_2^2 + t\bar{1}^T w - \mu\sum_{p=1}^{2p}\log w_p \\
s.t.\ \lambda = y - \bar{X}w
\end{aligned}
\tag{13}
$$

Introducing slack variables z, and using $W = diag(w)$ and $Z = diag(z)$, [5] writes the KKT conditions of this system as (note, that we are defining r_a, r_b, and r_c here as the left-hand sides of the equations):

$$
\begin{aligned}
r_a :\ -\bar{X}^T\lambda - z + t\bar{1}^T &= 0 \\
r_b :\ y - \bar{X}w - \lambda &= 0 \\
r_c :\ \mu 1^T - WZ\bar{1} &= 0
\end{aligned}
\tag{14}
$$

The main idea behind the method of [4] is to, at each iteration, perform a single Newton step on these KKT conditions, followed by a decrease in the vale of μ. [5] provides the following analytic solutions for computing the Newton directions (Δw, $\Delta\lambda$, Δz) for this system (using $D = Z^{-1}W$, and solving these equations in order):

$$
\begin{aligned}
\Delta\lambda = (I + XDX)^{-1}(r_b - X(Z^{-1}r_c - Dr_a) \\
\Delta z = r_a - X^T\Delta\lambda \\
\Delta w = Z^{-1}(r_c - W\Delta z)
\end{aligned}
\tag{15}
$$

Conjugate Gradient methods are used to solve for $\Delta\lambda$. Due to the non-negativity of the variables, the Newton step is truncated as follows (note that this slightly modified from [5] on include the possibility of a full Newton step):

$$
\begin{aligned}
\beta_w = min_{p:\Delta w_p<0}\{-w_p/\Delta w_p, 1\} \\
\beta_{\lambda,z} = min_{p:\Delta z_p<0}\{-z_p/\Delta z_p, 1\}
\end{aligned}
\tag{16}
$$

This gives the following update equations for the the variables, and the method suggested for updating the log barrier parameter:

$$
\begin{aligned}
w^{new} &= w + .99\beta_w\Delta w \\
\lambda^{new} &= \lambda + .99\beta_{\lambda,z}\Delta\lambda \\
z^{new} &= z + .99\beta_{\lambda,z}\Delta z \\
\mu^{new} &= (1 - min(.99, \beta_w, \beta_{\lambda,z}))
\end{aligned}
\tag{17}
$$

[5] provides expressions for initializing the variables w, λ, z, and for diagnosing convergence using a tolerance on the primal and dual feasibility, and the duality gap. To illustrate the main idea behind this approach, we present high-level pseudocode in Algorithm 2.

Algorithm 2 Chen's Method

1: initialize w, λ, z, μ
2: **while** {PRIM_FEAS,DUAL_FEAS,DUAL_GAP} $> \epsilon$ **do**
3: Compute $\Delta\lambda, \Delta w, \Delta z$,
4: Compute $\beta_x, \beta_{\lambda,z}$
5: Update w, λ, z
6: Update μ
7: **end while**

This Interior Point algorithm provides a very different approach to solving the problem than those proposed by Tibshirani. The main advantage of this method is its extremely fast convergence in terms of the number of iterations required to reach a solution of suitable precision. However, this method has two major disadvantages. Firstly, convergence of the method is sensitive to the initial value of μ, and no method is provided for computing an appropriate initial value. Secondly, the computationally complexity of computing the Newton direction for the dual variables is high, and this computation dominates the runtime of this algorithm. Furthermore, [5] reports that the complexity of solving this system increases as we approach the optimal solution.

2.4 Active Set Method, Local Linearization

[6, 7] has been the most influential work since the original LASSO and Basis Pursuit Denoising works. This work presents a method that does not require doubling the number of variables in the problem, does not need an exponential number of constraints, does not give degenerate constraints, has fast convergence properties, and finally the iteration cost can be kept relatively low through efficient implementation.

The two key components of the implementation of [6, 7] are the use of a local linearization about w, and the use of an Active Set method that operates only on the non-zero variables and a single zero-valued variable. This leads to the following optimization problem (for now we will assume that $\theta = sign(w)$ and note that σ simply indicates members of the Active Set):

$$\begin{aligned} &min_{h_\sigma} f(w_\sigma + h_\sigma) \\ &s.t.\ \theta_\sigma^T(w_\sigma + h_\sigma) \leq t \end{aligned} \tag{18}$$

Here we have (and will continue to) abuse the original notation slightly to avoid introducing permutation matrices (which are not really necessary to understand the algorithm). The solution to the KKT conditions of this problem is:

$$\mu = max(0, \frac{\theta_\sigma^T(X_\sigma^T X_\sigma)^{-1} X_\sigma^T y - t}{\theta_\sigma^T(X_\sigma^T X_\sigma)^{-1}\theta_\sigma}) \tag{19}$$

$$h_\sigma = (X_\sigma^T X_\sigma)^{-1}(X_\sigma^T(y - X_\sigma w_\sigma) - \mu\theta_\sigma)$$

The Active Set is initially empty, and at the end of each iteration we add one zero-valued element to this set. Specifically, we add the element (not already in σ) that has the largest (absolute value of its) violation. Using $w^\dagger = w + h$ (remember that elements outside the active set will be 0 as will corresponding elements of h) and $r^\dagger = y - Xw^\dagger$, we define the violation as follows:

$$v^\dagger = \frac{X^T r^\dagger}{||X_\sigma^T r^\dagger||_\infty} \tag{20}$$

Since $sign(0)$ is not well defined, the θ_i value for the element to be introduced into the Active Set is set to the sign of its violation. If the magnitude of the violation for all variables outside the active set is less than 1, then optimality is achieved. Using this, we see that in each iteration the algorithm adds the most violating variable to the active set, then solves for μ and subsequently h_σ. However, the algorithm becomes slightly more complicated when we consider that a variable in the active set may change sign during an iteration. In this case, the estimate $w_\sigma^\dagger$ is termed *sign infeasible*. If an estimate is sign infeasible, then the algorithm finds the first new zero component in the direction h_σ, and determines whether a descent direction can be achieved by flipping its sign (in θ_σ), or whether this variable should become 0 and thus be removed from the active set.

It is clear that the cost of early iterations of this method will be small, since the active set will be small. However, note the active set grows as large as the number of variable on later iterations. The key to maintaining efficient iterations with larger active sets is the maintenance and updating of a QR factorization of $X_\sigma^T X_\sigma$.

We will now consider convergence properties of this method. It is clear that the potential for removing elements from the active set could lead to a large number of iterations. However, in our experiments variables rarely needed to be removed from the active set. Furthermore, [8] later proved that this method shares the same asymptotic complexity as solving an ordinary Least Squares problem. Another appealing property of this method is the potential to use a 'warm-start' to compute the LASSO parameters for several (increasing) values of t (under the same asymptotic complexity). However, note that in this case it is more common to need to remove variables from the active set ([8] presents empirical results on this).

The method of [7, 6] (and the subsequent work of [8] under the Least Angle Regression framework) is currently the most widely used methods for estimating the LASSO parameters for non-trivial problems. It has several appealing properties that make it a natural choice among those methods discussed thus far. However, one of its disadvantages is that is must perform at least one iteration for every variable present in the final model. If this number is large, this

approach may not be as effective as approaches such as the Interior Point method that optimizes over all variables. Another disadvantage of this method is that it is the most difficult to implement (followed by the Interior Point method). This may seem to be a minor disadvantage, but many existing implementations of this method do not scale up to very large problems, and the non-trivial nature of the implementation could make several of the upcoming methods (which yield trivial implementations even when considering huge problems) more appealing for some researchers.

2.5 Iterated Ridge Regression

We now turn our focus to methods that directly address the unconstrained formulation. As with the constrained methods, we begin with a method that is primarily of historical interest. In addition to the two Quadratic Programming approaches discussed already, in [3] a third method was mentioned (lacking significant details) for computing LASSO parameters. The reason cited for the lack of details was that the method proved to be innefficient. The method was later re-invented (with details filled) by [9], describing it as an efficient alternative (due to its relationship to Newton's method). The method is based on the following approximation:

$$|w_i|_1 \approx \frac{w_i^2}{|w_i|_1} \tag{21}$$

Substituting this approximation into the unconstrained formulation (10), we can obtain an expression similar to Least Square with an L2 penalty (Ridge Regression):

$$w_{new} = (X^T X + \lambda diag(|w|)^{-1})^{-1} X^T y \tag{22}$$

We can now use w_{new} and resolve the above to obtain a better estimate. This can be iterated beginning from the L2 penalized solution until convergence. However, note that this approximation becomes numerically unstable as any w_i approaches 0, and that this is exactly what we expect at the optimal solution. To avoid this problem, we can use a generalized inverse of $diag(|w|)$. This removes values that are too close to 0 from the estimation, and avoids this problem. However, this inverse introduces a new problem, variables that are set to 0 can never move away from 0, and thus it could potentially lead to sub-optimal results if the initialization is inadequate [9].

In all of our experiments, we found that this algorithm did achieve the optimal solution, and we were not able to find a problem where the optimal is not found, unless a variable is initialized close enough to 0 that it is removed immediately. This agrees with the experiments in [9], where the optimal solution was always found despite their conjecture that it may not be. In our experiments, we found that this method seemed to have no difficulty dealing with variables that have the wrong sign (even if they are close to 0). Nevertheless, it is possible that our tests (and those of [9]) were not thorough enough to produce a case where the results were sub-optimal.

Although it is not mentioned in [3] or [9], it is clear that we can remove columns of X that correspond to variables that have been set to 0. This significantly speeds up the runtime of the algorithm, since later iterations solve smaller problems for sparse models. Another point to be made with respect to our implementation is that in general this method proved highly efficient, but occasionally the method took a large number of iterations to converge. It is possible that this might be resolved with a better stopping condition (neither [3] or [9] proposed one, so a very naive approach was temporarily used, and there was not sufficient time to update this before the deadline), or if a line minimization approach was used to potentially truncate the step sizes.

In summary, this method may represent a highly effective method, its iterations have a relatively low computational cost and the method has very good global convergence properties. However, we can't fully recommend this approach at this time due to the above implementation issues, and the possibility that this method could produce sub-optimal results.

2.6 Grafting

[10] present an optimization approach for a variety of loss functions in a general regularization framework. In a special case of this framework, we can compute LASSO estimates. This method also addresses the unconstrained problem, and the key idea behind the method is to use a definition of the function gradient that gives variables currently at 0 the appropriate sign.

Using the Least Squares derivative, we see that the derivative of the unconstrained formulation (10) (assuming variables are non-zero) is:

$$X^T(y - Xw) + \lambda sign(w) \tag{23}$$

For variables that have a value of zero, we use the following convention:

$$\begin{aligned} sign(w_i) = 1 \text{ if } X_i^T(y - Xw) > \lambda \\ sign(w_i) = -1 \text{ if } X_i^T(y - Xw < \lambda \end{aligned} \tag{24}$$

And by convention the gradient is set to 0 if $X_i^T(y - Xw) = \lambda$. This definition of the derivative gives the zero-valued variables the appropriate sign when they are introduced. We sketch out the Grafting procedure as follows. We begin with all variables set to a value of 0. At each iteration, we first test convergence by testing whether the following is true for all i:

$$|X_i^T(y - Xw)| \leq \lambda \tag{25}$$

If this equation is false, the variable whose derivative has the largest magnitude is added to the free set, and then an ('off the shelf') Quasi-Newton method with BFGS updating is used to optimize all the variables in the free set. This is quite similar at a high level to the active set method discussed earlier (although now for the unconstrained formulation), and the methods produce similar results after each iteration. One of the advantages that Grafting has over the active set method is the much simpler bookkeeping of the set of variables being optimized. The 'free set' is recomputed after each Quasi-Newton optimization terminates as the set of non-zero variables, and the zero-valued variable that has the largest absolute value of its derivative. Thus, the method is simpler to implement since it does not need to treat 'sign infeasibility' as a special case.

[10] argues that Grafting is much more efficient than directly applying a Quasi-Newton to the full variable set (assuming that the final model is sparse). However, with our implementations Grafting was significantly slower than the active set method discussed earlier. In its defense, this may have been primarily due to the use of Matlab's 'fminunc' function, and the methods may be closer in performance given a more optimized implementation of a (limited-memory) Quasi-Newton method or other optimization strategy.

2.7 Gauss-Seidel

Recently presented in [11] is an approach that promises "no external matrix storage", no "matrix operations", and no "optimization software" required, in order to provide an attractive avenue between the biological sciences community and the LASSO. They propose a Gauss-Seidel method presented for the case of Logistic Regression, but we can derive a Least Squares formulation of the same method that yields an interesting property.

This method is fairly similar to Grafting, but has a notable difference. Before we examine this difference, we'll examine the similarities. As in Grafting, we begin with all variables set to 0, and at each iteration we introduce 1 zero-valued variable with maximum 'violation', optimize this free set, then recompute the free set and continue optimizing until we satisfy an optimality criteria. This method uses the same definition of the derivative as in Grafting, and the maximum 'violating' variable among zero-valued variables is exactly the variable with highest magnitude of derivative as defined in Grafting.

The difference between Grafting and the Gauss-Seidel method is thus the method used to optimize the free variables. While Grafting uses a Quasi-Newton routine to find the optimal solution for the variables in the free set, the Gauss-Seidel method optimizes the most violating variable in the free set one at a time (beginning with the single zero-valued variable). In the case of Logistic Regression, this 1-dimensional minimization involves a bracketed line minimization. However, for the Least Squares problem, the objective function is quadratic, and thus we can exactly compute the minimizer of this quadratic objective through a Newton update (using g_i to denote the gradient with respect to variable i as defined above):

$$w_i^{new} = w_i - g_i/H_{ii} \tag{26}$$

This requires the diagonal elements of the Hessian, in the case of Least Squares with an L1 penalty the diagonals of the Hessian have the following form:

$$H_{ii} = \sum_{j=1}^{n} X(j,i)^2 \tag{27}$$

However, since the Hessian does not depend on w, we only need to compute these values once (although this is not mentioned in the original paper). To summarize, although we would make much more progress on each iteration using a Quasi-Newton method, the individual iterations of the Gauss-Seidel method have a closed form solution and we can compute the necessary Hessian elements offline (although we must still compute the gradient at every iteration). Because of these extremely cheap line searches, the Gauss-Seidel algorithm may outperform Grafting (and in fact its runtime was competitive with most methods discussed in this document for variables with smaller values of p). The obvious advantages of the Gauss-Seidel approach are its simplicity and its low iteration cost. Consequently, the obvious disadvantage of this approach is its convergence properties. Although convergence is proven, the number of iterations required for this algorithm is substantially larger than for any others. In addition, as more variables are added to the free set, a larger number of iterations is required to optimize them. Clearly, after a large enough number of variables have been introduced, this approach becomes infeasible.

2.8 Shooting

Moving from Gauss-Seidel (one of the most recent methods proposed) we can naturally return to one of the first methods proposed for this problem (beyond the methods of the original authors). One of the first alternate optimization strategies is outlined in [12], where an optimization strategy referred to as 'Shooting' is presented. Despite its colorful name (to accompany the LASSO), this algorithm is rarely referred to in the related works on this problem. This is possibly due to the fact that it is overshadowed by a Newton method presented for a larger set of regularization penalties

in the same work, and that it occupies less than $1/4$ of a page to describe in its entirety. We will present this algorithm in its full form in a way that avoids introducing the unnecessary notation used in [12]. Define S_j as follows:

$$S_j = -X_j^T y + \sum_{i \neq j} X_j^T X_i w_i \tag{28}$$

Now, define an update to variable w_j as:

$$\begin{aligned} \text{if } S_j > \lambda \text{ , then: } w_j &= \frac{\lambda - S_j}{X_j^T X_j} \\ \text{if } S_j < -\lambda \text{ , then: } w_j &= \frac{-\lambda - S_j}{X_j^T X_j} \\ \text{if } |S_j| \leq \lambda \text{ , then: } w_j &= 0 \end{aligned} \tag{29}$$

The algorithm proceeds as follows: Beginning from the Ridge Regression solution, (1) for each j from 1 to p, compute S_j and update w_j, repeat (1) until convergence.

Although not mentioned in the original work, clearly we can pre-compute all needed values of $X_j^T y$ and $X_j^T X_j$ since they do not depend on w. And so for each iteration through the for loop we only need to compute $\sum_{i \neq j} X_j^T X_i w_i$. We can see that this closely resembles the Gauss-Seidel approach, but does away with keeping track of free variables, and computing the values for the full gradient or finding the most violating variable. It simply needs to compute 1 element of the gradient at each iteration, then update the variable. In addition, we found that it converges in a much smaller number of iterations than the Gauss-Seidel approach, although it still requires substantially more than all other methods discussed in this document. However, in terms of iteration cost and implementation simplicity, 'Shooting' is the clear winner.

3 Related Work

The remaining sections of this document briefly survey related work on several topics. We first review methods that have been proposed in the case where the design matrix X is orthogonal. We then turn our attention to methods that have been proposed for optimizing the LASSO that do not achieve the optimal solution. We then present a survey of other loss functions that an L1 penalty has been applied to, and some of the notable optimization methods used in these works. Our focus then shifts to methods that have been proposed for finding an appropriate value of the regularization parameter, and we find that this ties back to our discussion of optimization methods. Finally, we overview some interesting properties of L1 penalties, and examine several important application areas where L1 penalties are currently or may soon have an impact.

3.1 Orthogonal Design

There have several approaches proposed for computing the LASSO estimate in the special case where $X^T X = I$. In the Basis Pursuit Denoising literature (where Orthogonal Design matrices can be constructed), [5] introduced a method based on the Block Coordinate Relaxation (BCR) strategy. Specifically, it minimizes the objective with respect to a block of variables, keeping the others fixed. 2 methods are proposed for selecting these blocks (systematic cycling and optimal descent). Convergence of the algorithm is proved, and an empirical test (based on CPU cycles) with the Interior Point method of [4] is performed. It is shown that the BCR strategy is at least as fast as this Interior Point method, and that the BCR strategy can yield approximate solutions much more efficiently.

Tibshirani also briefly discussed the orthogoncal design case, defining the optimal solution and showing that the LASSO gives the same estimate as the Garotte function in this case [3]. Later, an efficient and trivial algorithm that takes advantage of this definition was presented in [6].

3.2 Approximation Methods

Computing the optimal LASSO parameters is a convex optimization problem, and thus any local minimum found is guaranteed to be a global minimum. In addition, we have surveyed in this work several highly efficient algorithms for computing the optimal LASSO parameters. Nevertheless, several works (in prominent Machine Learning venues) have presented highly efficient but sub-optimal algorithms.

[13] made the observation that the LASSO is equivalent to a technique called the 'adaptive ridge', that places a separate non-negative penalty on the absolute value of each coefficient (similar to a Relevance Vector Machine). This equivalence simply requires an obvious constraint on the sum of the values of these penalties. Using this equivalence, they propose to use a Fixed Point algorithm for computing the adaptive ridge, in order to compute LASSO parameters. Beginning from an L2 penalized solution, the Fixed Point algorithm iterates between estimating these values, and estimating the coefficient vector (similar to the Expectation Maximization algorithm). However, the authors say that the method is likely not to be globally convergent. This counter-intuitive result appears to be due to the slightly different constraints that the adaptive ridge uses, and that the constraint enforcing the equivalence with the LASSO is not properly taken advantage of during the Fixed Point iterations. The author's implementation of this approach was included in several experiments. Based on these experiments, (i) we confirmed that this method indeed does not find the optimal solution, (ii) the models generated are too sparse

for small value of λ (as in Relevance Vector Machines) and not sparse enough for large values of λ, and (iii) the method finds its sub-optimal solution highly efficiently, but there is no significant saving over some of the optimal methods.

More recently, [14] presented a highly efficient but sub-optimal approach for the LASSO problem. This approach used a gradient descent-based approach related to L1 boosting. The motivation in this work was to avoid Quadratic Programming (we hope it clear from this work that this can be done while maintaining optimality) and approximately estimate the parameters for very large problem sizes. An implementation of this technique is also available (in R), but we did not experiment with it.

3.3 Other Loss Functions

Although the LASSO formulation has become very popular, L1 regularization has become an even more popular topic recently in the context of classification. Here, the target variable takes one of several classes, and Least Squares generally gives poor predictions compared to techniques such as Support Vector Machines, Boosting, and of particular interest to the topic of this work, Logistic Regression.

Presented in [3] was a strategy for estimating L1 penalized parameters of loss functions that can yield a quadratic IRLS approximation. This simply involves using an IRLS loop that uses a (weighted) LASSO solver. [15] extended the Active Set method of [6, 7] to IRLS models (under the name 'Generalized LASSO'), focusing specifically on Logistic Regression. The 'Grafting' method of [10] and the Gauss-Seidel method of [11] were also presented for the case of Logistic Regression. Finally, note that it is trivial to convert *any* LASSO solver into a weighted LASSO solver by running the algorithm on re-weighted inputs. Thus, notice that the use of a potentially generic LASSO solver in the IRLS iterations proposed by [3] means that any algorithm proposed for optimizing the LASSO can be used for Logistic Regression (and other IRLS approximated functions such as the L1 or Huber loss) with an L1 penalty.

Several techniques have also been presented exclusively for the case of Logistic Regression with an L1 penalty, although it is clear that many of these techniques would also apply in the LASSO scenario. [16] used a strategy for Logistic Regression with an L1 penalty called stochastic sub-gradient descent. The key idea behind this method is to 'jitter' away from the point of non-differentiability by taking a small step along a sub-gradient. The weights do not become exactly zero in this model, and are thresholded when the algorithm terminates. [17] presented an approach based on cyclic coordinate descent (with a minor modification to allow re-introduction of variables currently at 0). Three different approaches based on iterative scaling for Logistic Regression with an L1 loss were suggested in [18].

Although Least Squares and Logistic Regression are applicable in a wide variety of scenarios, L1 penalties have been extended to even wider array of problems. This includes Multi-layer Perceptrons (Neural Networks trained by backpropagation) [10], Support Vector Machines [13], Generalized Additive Models [13], Probit Regression [19], the L1 loss [20], and the Huber loss [20]. Finally, as discussed in [21], the Boosting algorithm optimizes a criteria that is approximated by an L1 penalty on the appropriate loss function.

3.4 Regularization Parameter Estimation

A parallel issue to optimizing the LASSO parameters given a fixed value of λ is selecting a good value of the regularization parameter λ. [3] proposed three methods for computing an appropriate value of λ. The first was the simple but computationally expensive cross-validation procedure (since it involves solving a large number of similar problems). The second was a computationally simple but less accurate unbiased estimate of risk (here only one problem is solved). Finally, the third (and recommended by the author) method is a generalized cross-validation scheme that is less computationally expensive than cross-validation but provides a better estimate than the unbiased estimate of risk. [16] later proposed a randomized variant of Generalized Approximate Cross Validation for regularization parameter estimation. This work also proposed to use a strategy called 'slice modeling' to solve the similar optimization problems more effectively.

Another contribution of the influential work of [6, 7] was an extremely useful tool for hyperparameter estimation, that also vastly increases the interpretability of the models. They observed that the coefficient values follow a piecewise-linear path as t is changed. Combined with their active set method that allows 'warm-starts' from lower t values, they presented a homotopy-based algorithm that computes the LASSO coefficients for all values of t. [8] later termed the phrase 'regularization path' for this idea, and showed that this method allows computation of all possible values of t using the same asymptotic complexity of solving an ordinary Least Squares problem. Although it was not explored in detail in this work, it likely that several of the other LASSO optimization methods discussed could also be used for efficient computation of the regularization path.

3.5 Properties of L1 Regularization

From a statistical point of view, it well-known that optimizing parameters under an L2 penalty is equivalent to finding the mean (and mode) of the posterior distribution of the parameters (ie. a MAP estimate) subject to Gaussian

prior probabilities on the parameters [3]. It was shown in [3] that an L1 penalty is equivalent to finding the mode (but not necessarily mean) of the posterior distribution of the parameters under a double exponential prior (whose heavy-tailed nature yields further insights into the sparse nature of the solutions). Finally, [15] presented a method to compute posterior predictive probabilities over the predictions made using a model that was estimated subject to an L1 penalty, by using exponential hyper-priors and analytically integrating them out. This leads to an efficient method for estimating the variance of predictions made by the model.

The concept of Bridge regularization is discussed in [1]. Bridge regularization involves a penalty term on the L^q norm of the parameters, with $q \geq 0$ (and not necessarily integral). L2 penalties correspond to the case where q is 2, while L1 penalties correspond to the case where q is 1 (subset selection is defined as the case where q is 0). An interesting note is that L1 penalties are the smallest Bridge penalties that yield a convex set for the constraint region [3]. This view also leads to a geometric interpretation of the sparseness properties of the Lasso (see [1]). [12] gives a more general form of the corresponding statistical priors associated with Bridge penalties.

An interesting observation about the sparsity of the LASSO coefficients was made in [8]. They state that the number of non-zero coefficients is bounded by $n - 1$. Interestingly, they state that the $n - 1$ coefficients for a given value of λ will not necessarily include those selected for the maximum value of λ. A related observation made by [6] is that the search for t can be restricted to the range $[0, |X^T y|_\infty]$ (although we found that in practice this bound can be very generous).

A final interesting and important property of L1 regularization is the recent work in [22] on the effective sample complexity of using L1 regularization compared to L2 regularization. This work shows that the sample complexity (ie. which the authors define as the number of instances needed to learn 'well') grows at least linearly in the number of irrelevant features for many loss functions when using L2 regularization (this includes if L2-based preprocessing such PCA was used), while the sample complexity grows only logarithmically when L1 regularization is used.

3.6 Applications

Many of the works related to the LASSO have focused exclusively on publicly available (small) benchmark data sets. Among the more ambitious and diverse applications, [5] applied the method to detection of incoming radar signatures, [16] applied Basis Pursuit to epidemiological studies, and [23] applied Logistic Regression with an L1 penalty for identifying features associated with program crashes. Among the most recent works, two of the areas where the LASSO is showing significant potential are the analysis of microarray and other forms of genetic data [15, 11], and in Natural Language Processing applications [18]. This data usually has an extremely large number of features and relatively few instances. Thus, sparse interpretable models are highly desirable. Another area where the author of this work sees a need for sparse regularization is Computer Vision, where computationally expensive and redundant filter banks are currently used. Increasing the sparsity of these filter banks would be highly desirable, and could vastly increase the speed at which data can be analyzed.

3.7 Conclusion

This project has surveyed and examined a variety of approaches proposed for parameter estimation in Least Squares linear regression models with an L1 penalty on the regression coefficients. It was shown that there are several varied methods to circumvent the non-differentiability of the objective function, and that there currently exist highly effective methods for solving this problem.

We showed that despite the stigma surrounding Tibshirani's proposed method, that an exponential number of constraints are never required. We discussed Tibshirani's other proposed (and more popular) method, and observed that doubling the number of variables to avoid an exponential number of constraints may not be beneficial, and that when using this formulation we must be careful to address degeneracy. We then reviewed an Interior Point Log-Barrier method, which in general had the best convergence rate for non-trivial problems, but had the highest iteration cost and proved to be sensitive to the initial value of the log-barrier parameter. We outlined the currently very popular Active Set method. This method introduces variables one at a time and maintains a QR factorization of part of the design matrix to allow fast iterations while maintaining a fast convergence rate.

For methods that directly address the unconstrained problem, we first discussed an Iterated Ridge Regression approach that may be an excellent option, but we were unable to say for certain due to issues of convergence and optimality of the solution. We also proposed a simple extension to this method that discards unneeded columns of X to significantly speed up the algorithm. We discussed Grafting, a method similar to the Active Set approach that uses existing Quasi-Newton software to enable a simpler, but slower, implementation. After Grafting, we discussed the very similar Gauss-Seidel method. This method took advantage of the simple nature of 1-dimensional line searches over this quadratic objective to yield an extremely low iteration cost, although the number of iterations required is substantially higher. Finally, we presented the extremely simple but surprisingly effective Shooting method. Through appropriate

pre-computation, we showed each iteration of this algorithm is dominated by simply computing a single element of the gradient, and that this is by far the simplest method to implement.

Finally, accompanying this document is a webpage located at http://www.cs.ubc.ca/ schmidtm/c542B.html. This page contains contains simple implementations of the 8 techniques that were the focus of this paper (in addition to several others). Since these implementations were created primarily to help understanding of the algorithms themselves, the webpage also contains links to existing implementations available online, that may be more efficient and address more general cases.

References

[1] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[3] R. Tibshirani. Regression shrinkage and selection via the lasso. Technical report, University of Toronto, 1994.

[4] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[5] S. Sardy, A. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries. Technical report, Seattle, WA, 1998.

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

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

[8] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. Technical report, Stanford University, 2002.

[9] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. 96(456):1348–??, December 2001.

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

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

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

[13] Yves Grandvalet and Stéphane Canu. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. In *NIPS*, pages 445–451, 1998.

[14] Yongdai Kim and Jinseog Kim. Gradient lasso for feature selection. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 60, New York, NY, USA, 2004. ACM Press.

[15] Volker Roth. The generalized lasso. *IEEE Transactions on Neural Networks*, 15(1), January 2004.

[16] Hao Zhang, Grace Wahba, Yi Lin, Meta Voelker, Michael Ferris, Ronald Klein, and Barbara Klein. Variable selection and model building via likelihood basis pursuit. Technical Report 1059, University of Wisconsin, Department of Statistics, Jul 2002.

[17] A. Genkin, D. Lewis, and D. Madigan. Large-scale bayesian logistic regression for text categorization. Submitted.

[18] Joshua Goodman. Exponential priors for maximum entropy models. In *HLT-NAACL*, pages 305–312, 2004.

[19] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Learning sparse bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2005.

[20] Sylvain Sardy, Paul Tseng, and A.G. Bruce. Robust wavelet denoising. *IEEE Transactions on Signal Processing*, 49(6), 2001.

[21] Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

[22] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 78, New York, NY, USA, 2004. ACM Press.

[23] Alice X. Zheng, Michael I. Jordan, Ben Liblit, and Alex Aiken. Statistical debugging of sampled programs. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.