Boosted Lasso

Peng Zhao
University of California, Berkeley, USA.
Bin Yu
University of California, Berkeley, USA.

Summary.
In this paper, we propose the Boosted Lasso (BLasso) algorithm that is able to produce an approximation to the complete regularization path for general Lasso problems. BLasso is derived as a coordinate descent method with a fixed small step size applied to the general Lasso loss function (L_1 penalized convex loss). It consists of both a forward step and a backward step and uses differences of functions instead of gradient. The forward step is similar to Boosting and Forward Stagewise Fitting, but the backward step is new and crucial for BLasso to approximate the Lasso path in all situations. For cases with finite number of base learners, when the step size goes to zero, the BLasso path is shown to converge to the Lasso path. For nonparametric learning problems with a large or an infinite number of base learners, BLasso remains valid since its forward steps are Boosting steps and its backward steps only involve the base learners that are included in the model from previous iterations. Experimental results are also provided to demonstrate the difference between BLasso and Boosting or Forward Stagewise Fitting. In addition, we extend BLasso to the case of a general convex loss penalized by a general convex function and illustrate this extended BLasso with examples.

Keywords: Coordinate Descent; Backward; Boosting; Lasso; Regularization Path.

1. Introduction

Lasso (Tibshirani, 1996) is an important idea recently originated in statistics. It regularizes or shrinks a fitted model through an L_1 penalty or constraint. Its popularity can be explained in several ways. Since nonparametric models that fit training data well often have low biases but large variances, prediction accuracies can sometimes be improved by shrinking a model or making it more sparse. The regularization resulting from the L_1 penalty leads to sparse solutions, that is, there are few basis functions with nonzero weights (among all possible choices). This statement is proved asymptotically by Knight and Fu (2000) and for the finite case by Donoho (2004) in the specialized setting of over-complete libraries and large under-determined systems of linear equations. Furthermore, the sparse models induced by Lasso are more interpretable and often preferred in sciences and social sciences.

Another vastly popular and recent idea is Boosting. Since its inception in 1990 (Schapire, 1990; Freund, 1995; Freund and Schapire, 1996), it has become one of the most successful machine learning methods and is now understood as an iterative method leading to an additive model (Breiman, 1998, Mason et al., 1999 and Friedman et al., 2001).

While it is a natural idea to combine boosting and Lasso to have a regularized Boosting procedure, it is also intriguing that Boosting, without any additional regularization, has its own resistance to overfitting. For specific cases such as L_2Boost (Friedman, 2001), this resistance is understood to some extent (Buhlmann and Yu, 2001). However, it was not until later when Forward Stagewise Fitting (FSF) was introduced as a boosting based procedure with much more cautious steps that a similarity between FSF and Lasso was observed (Hastie et al., 2001, Efron et al., 2004).

This link between Lasso and FSF is formally described for the linear regression case through LARS (Least Angle Regression, Efron et al. 2004). It is also known that for special cases (such as orthogonal designs) FSF can approximate Lasso path infinitely close, but in general, it is unclear what regularization criteria FSF optimizes. As can be seen in our experiments (Figure 1), FSF solutions can be significantly different from the Lasso solutions in the case of strongly correlated predictors which are common in high-dimensional data problems. However, it is still used as an approximation to Lasso because it is often computationally prohibitive to solve Lasso with general loss functions for many regularization parameters through Quadratic Programming.

In this paper, we propose a new algorithm **Boosted Lasso** (**BLasso**). It approximates the Lasso path in all situations. The motivation comes from a critical observation that both FSF and Boosting only work in a forward fashion (so is FSF named). They always take steps that reduce empirical loss the most regardless of the impact on model complexity (or the L_1 penalty in the Lasso case). This often proves to be too greedy – the algorithms are not able to correct mistakes made in early stages. Taking a coordinate descent view point of the Lasso minimization with a fixed step size, we introduce an innovative "backward" step. This step utilizes the same minimization rule as the forward step to define each fitting stage with an additional rule to force the model complexity to decrease. By combining backward and forward steps, Boosted Lasso is able to go back and forth and approximates the Lasso path correctly.

BLasso can be seen as a marriage between two families of successful methods. Computationally, BLasso works similarly to Boosting and FSF. It isolates the sub-optimization problem at each step from the whole process, i.e. in the language of the Boosting literature, each base learner is learned separately. This way BLasso can deal with different loss functions and large classes of base learners like trees, wavelets and splines by fitting a base learner at each step and aggregating the base learners as the algorithm progresses. But unlike FSF which has implicit local regularization in the sense that at any iteration, FSF with a fixed small step size only searches over those models which are one small step away from the current one in all possible directions corresponding to the base learners, BLasso can be proven to converge to the Lasso solutions which have explicit global L_1 regularization for cases with a finite number of base learners. And the algorithm uses only the differences of the loss function and basic operations without taking any first or second order derivative or matrix inversion. The fact that BLasso can be generalized to give regularized path for other convex penalties also comes as a pleasant surprise which further shows a connection between statistical estimation methods and the Barrier Method (Fiacco and McCormick, 1968, and cf. Boyd and Vandenberghe 2004) in the constrained convex optimization literature. For the original Lasso problem, i.e. the least squares problem with L_1 penalty, algorithms that give the entire Lasso path have been established (Shooting algorithm by Fu 1998 and LARS by Efron et al. 2004). These algorithms are not suitable for nonparametric learning where the number of base learners can be very large or infinite. BLasso remains valid since it treats each fitting step as a sub-optimization problem. Even for cases where the number of base learners is small, BLasso is still an attractive alternative to existing algorithms because of its simple and intuitive nature and the fact that it can be stopped early to find an solution from the regularization path.

The rest of the paper is organized as follows. A brief review of Boosting and FSF is provided in Section 2.1 and Lasso in Section 2.2. Section 3 introduces BLasso. Section 4 discusses the backward step and gives the intuition behind BLasso and explains why FSF is unable to give the Lasso path. Section 5 contains a more detailed discussion on solving L_1 penalized least squares problem using BLasso. Section 6 discusses BLasso for nonparametric learning problems. Section 7 introduces a Generalized BLasso algorithm which deals with general convex penalties. In Section 8, results of experiments with both simulated and real data are reported to demonstrate the attractiveness of BLasso and differences between BLasso and FSF. Finally, Section 9 contains a discussion on the choice of step sizes, a summary of the paper, and future research directions.

2. Boosting, Forward Stagewise Fitting and the Lasso

Boosting is an iterative fitting procedure that builds up a model stage by stage. FSF can be viewed as Boosting with a fixed small step size at each stage and it produces solutions that are often close to the Lasso solutions (path). This section gives a brief review of the FSF and Boosting algorithms followed by a review of Lasso.

2.1. Boosting and Forward Stagewise Fitting

The boosting algorithms can be seen as functional gradient descent techniques (Friedman et al. 2000, Mason et al. 1999). The task is to estimate the function $F : R^d \to R$ that minimizes an expected loss

$$E[C(Y, F(X))], \quad C(\cdot,\cdot) : R \times R \to R^+ \tag{1}$$

based on data $Z_i = (Y_i, X_i)(i = 1, ..., n)$. The univariate Y can be continuous (regression problem) or discrete (classification problem). The most prominent examples for the loss function $C(\cdot,\cdot)$ include Classification Margin, Logit Loss and L_2 Loss functions.

The family of $F(\cdot)$ being considered is the set of ensembles of "base learners"

$$D = \{F : F(x) = \sum_j \beta_j h_j(x), x \in R^d, \beta_j \in R\}, \tag{2}$$

where the family of base learners can be very large or contain infinite members, e.g. trees, wavelets and splines.

Let $\beta = (\beta_1, ...\beta_j, ...)^T$, we can reparametrize the problem using

$$L(Z, \beta) := C(Y, F(X)), \tag{3}$$

where the specification of F is hidden by L to make our notation simpler.

To find an estimate for β, we set up an empirical minimization problem:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} L(Z_i; \beta). \tag{4}$$

Despite the fact that the empirical loss function is often convex in β, exact minimization is usually a formidable task for a moderately rich function family of base learners and with such function families the exact minimization leads to overfitted models. Boosting is a progressive procedure that iteratively builds up the solution (and it is often stopped early to avoid overfitting):

$$\begin{aligned}(\hat{j}, \hat{g}) &= \arg\min_{j,g} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + g1_j) &(5)\\ \hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{g}1_{\hat{j}} &(6)\end{aligned}$$

where 1_j is the jth standard basis, i.e. the vector with all 0s except for a 1 in the jth coordinate and $g \in R$ is a step size parameter. However, because the family of base learners is usually large, finding j is still difficult, for which approximate solutions are found. A typical strategy that Boosting implements is by applying functional gradient descent. This gradient descent view has been recognized and refined by various authors including Breiman (1998), Mason et al. (1999), Friedman et al. (2000), Friedman (2001), and Buhlmann and Yu (2001). The well-known AdaBoost, LogitBoost and L_2Boosting can all be viewed as implementations of this strategy for different loss functions.

Forward Stagewise Fitting (FSF) is a similar method for approximating the minimization problem described by (5) with some additional regularization. Instead of optimizing the step size as in (6), FSF updates $\hat{\beta}^t$ by a small fixed step size ϵ as in Friedman (2001):

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \epsilon \cdot \text{sign}(g) 1_{\hat{j}}$$

When FSF was introduced (Hastie et al. 2001, Efron et al. 2002), it was only described for the L_2 regression setting. For general loss functions, it can be defined by removing the minimization over g in (5):

$$\begin{aligned}(\hat{j}, \hat{s}) &= \arg\min_{j,s=\pm\epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), &(7)\\ \hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s}1_{\hat{j}}, &(8)\end{aligned}$$

This description looks different from the FSF described in Efron et al.(2002), but the underlying mechanic of the algorithm remains unchanged (see Section 5). Initially all coefficients are zero. At each successive step, a predictor or coordinate is selected that reduces most the empirical loss. Its corresponding coefficient $\beta_{\hat{j}}$ is then incremented or decremented by a small amount, while all other coefficients β_j, $j \neq \hat{j}$ are left unchanged.

By taking small steps, FSF imposes some implicit regularization. After $T < \infty$ iterations, many of the coefficients will be zero, namely those that have yet to be incremented. The others will tend to have absolute values smaller than the unregularized solutions. This shrinkage/sparsity property is reflected in the similarity between the solutions given by FSF and Lasso which is reviewed next.

2.2. Lasso

Let $T(\beta)$ denote the L_1 penalty of $\beta = (\beta_1, ..., \beta_j, ...)^T$, that is, $T(\beta) = \|\beta\|_1 = \sum_j |\beta_j|$, and let $\Gamma(\beta; \lambda)$ denote the Lasso (least absolute shrinkage and selection operator) loss function

$$\Gamma(\beta; \lambda) = \sum_{i=1}^{n} L(Z_i; \beta) + \lambda T(\beta). \tag{9}$$

The Lasso estimate $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_j, ...)^T$ is defined by

$$\hat{\beta}_\lambda = \min_\beta \Gamma(\beta; \lambda).$$

The parameter $\lambda \geq 0$ controls the amount of regularization applied to the estimate. Setting $\lambda = 0$ reverses the Lasso problem to minimizing the unregularized empirical loss. On the other hand, a very large λ will completely shrink $\hat{\beta}$ to 0 thus leading to the empty or null model. In general, moderate values of λ will cause shrinkage of the solutions towards 0, and some coefficients may end up being exactly 0. This sparsity in Lasso solutions has been researched extensively, e.g. Efron et al. (2004) and Donoho (2004). (Sparsity can also result from other penalizations as in Gao and Bruce, 1997 and Fan and Li, 2001.)

Computation of the solution to the Lasso problem for a fixed λ has been studied for special cases. Specifically, for least squares regression, it is a quadratic programming problem with linear inequality constraints; for 1-norm SVM, it can be transformed into a linear programming problem. But to get a model that performs well on future data, we need to select an appropriate value for the tuning parameter λ. Practical algorithms have been proposed to give the entire regularization path for the squared loss function (Shooting algorithm by Fu, 1998 and LARS by Efron et al., 2004) and SVM (1-norm SVM by Zhu et al, 2003).

But how to give the entire regularization path of the Lasso problem for general convex loss function remained open. More importantly, existing methods are only efficient when the number of base learners is small, that is, they can not deal with large numbers of base learners that often arise in nonparametric learning, e.g. trees, wavelets and splines. FSF exists as a compromise since, like Boosting, it is a nonparametric learning algorithm that works with different loss functions and large numbers of base learners but it only has implicit local regularization and is often too greedy comparing to Lasso as can be seen from our analysis (Section 4) and experiment (Figure 1).

Next we propose the Boosted Lasso (BLasso) algorithm which works in a computationally efficient fashion as FSF. But unlike FSF, BLasso is able to approximate the Lasso path for general convex loss functions infinitely close.

3. Boosted Lasso

We start with the description of the BLasso algorithm. A small step size constant $\epsilon > 0$ and a small tolerance parameter $\xi \geq 0$ are needed to run the algorithm. The step size ϵ controls how well BLasso approximates the Lasso path. The tolerance ξ controls how large a descend needs to be made for a backward step and is set to be smaller than ϵ to have a good approximation. When the number of base learners is finite, the tolerance can be set to 0 or a small number to accommodate the numerical precision of the computer.

Boosted Lasso (BLasso)

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, ..., n$, take an initial forward step

$$\begin{aligned} (\hat{j}, \hat{s}_{\hat{j}}) &= \arg\min_{j, s=\pm\epsilon} \sum_{i=1}^n L(Z_i; s1_j), \\ \hat{\beta}^0 &= \hat{s}_{\hat{j}} 1_{\hat{j}}, \end{aligned}$$

Then calculate the initial regularization parameter

$$\lambda^0 = \frac{1}{\epsilon}(\sum_{i=1}^{n} L(Z_i; 0) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^0))$$

.

Set the active index set $I_A^0 = \{\hat{j}\}$. Set $t = 0$.

Step 2 (Backward and Forward steps). Find the "backward" step that leads to the minimal empirical loss

$$\hat{j} = \arg\min_{j \in I_A^t} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s_j 1_j) \quad \text{where } s_j = -\text{sign}(\hat{\beta}_j^t)\epsilon. \tag{10}$$

Take the step if it leads to a decrease of moderate size ξ in the Lasso loss, otherwise force a forward step (as (7), (8) in FSF) and relax λ if necessary. That is if $\Gamma(\hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}; \lambda^t) - \Gamma(\hat{\beta}^t, \lambda^t) < -\xi$, then

$$\begin{aligned} \hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}, \\ \lambda^{t+1} &= \lambda^t, \\ I_A^{t+1} &= I_A^t / \{\hat{j}\}, \text{ if } \hat{\beta}_{\hat{j}}^{t+1} = 0. \end{aligned}$$

Otherwise,

$$\begin{aligned} (\hat{j}, \hat{s}) &= \arg\min_{j, s=\pm\epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s 1_j), \qquad (11) \\ \hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s} 1_{\hat{j}}, \qquad (12) \\ \lambda^{t+1} &= \min[\lambda^t, \frac{1}{\epsilon}(\sum_{i=1}^{n} L(Z_i; \hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^{t+1}))], \\ I_A^{t+1} &= I_A^t \cup \{\hat{j}\}. \end{aligned}$$

Step 3 (iteration). Increase t by one and repeat Step 2 and 3. Stop when $\lambda^t \leq 0$.

We will discuss forward and backward steps in depth in the next section. Immediately, the following properties can be proved for BLasso (see Appendix for the proof).

Lemma 1.

(a) For any $\lambda \geq 0$, if there exist j and s with $|s| = \epsilon$ such that $\Gamma(s1_j; \lambda) \leq \Gamma(0; \lambda)$, we have $\lambda^0 \leq \lambda$.
(b) For any t such that $\lambda^{t+1} = \lambda^t$, we have $\Gamma(\hat{\beta}^{t+1}; \lambda^t) \leq \Gamma(\hat{\beta}^t; \lambda^t)$.
(c) For $\xi = 0$ and any t such that $\lambda^{t+1} < \lambda^t$, we have $\Gamma(\hat{\beta}^t; \lambda^t) < \Gamma(\hat{\beta}^t \pm \epsilon 1_j; \lambda^t)$ for every j and $\|\hat{\beta}^{t+1}\|_1 = \|\hat{\beta}^t\|_1 + \epsilon$.

Lemma 1 (a) guarantees that it is safe for BLasso to start with an initial λ_0 which is the largest λ that would allow an ϵ step away from 0 (i.e., larger λ's correspond to $\hat{\beta}_\lambda = 0$). Lemma 1 (b) says that for each value of λ, BLasso performs coordinate descent until there is no descent step. Then, by Lemma 1 (c), the value of λ is reduced and a forward step is forced. Since the minimizers corresponding to adjacent λ's are usually close, this procedure moves from one solution to the next within a few steps and effectively approximates the Lasso path. Actually, we have a convergence result for BLasso:

Theorem 1. For finite number of base learners and $\xi = 0$, if $L(Z;\beta)$ is strictly convex and continuously differentiable in β, then as $\epsilon \to 0$, the BLasso path converges to the Lasso path.

Many popular loss functions, e.g. squared loss, logistic loss and negative log-likelihood functions of exponential families, are convex and continuously differentiable. Other functions like the hinge loss (SVM) is continuous and convex but not differentiable. In fact, BLasso does not use any gradient or higher order derivatives but only the differences of the loss function. When the function is differentiable, because of the small fixed step size, the differences are good approximations to the gradients, but they are simpler to compute. When the loss function is not differentiable, BLasso still runs because it does not rely on derivatives. It is theoretically possible that BLasso's coordinate descent strategy gets stuck at nonstationary points for functions like the hinge loss. However, as illustrated in our third experiment, BLasso may still work for 1-norm SVM empirically.

Note that Theorem 1 does not cover nonparametric learning problems with an infinite number of base learners. In fact, for problems with a large or an infinite number of base learners, the minimization in (11) can be carried out approximately by functional gradient descent, but a tolerance $\xi > 0$ needs to be chosen to avoid oscillation between forward and backward steps caused by slow descending. We will return to this topic in Section 6.

4. The Backward Boosting Step

We now explain the motivation and working mechanic of BLasso. Observe that FSF only uses "forward" steps, that is, it only takes steps that lead to a direct reduction of the empirical loss. Comparing to classical model selection methods like Forward Selection and Backward Elimination, Growing and Pruning of a classification tree, a "backward" counterpart is missing. Without the backward step, FSF can be too greedy and deviate from the Lasso path in some cases (cf. Figure 1). This backward step naturally arises in BLasso because of our coordinate descent view of the minimization of the Lasso loss.

For a given $\beta \neq 0$ and $\lambda > 0$, consider the impact of a small $\epsilon > 0$ change of β_j to the Lasso loss $\Gamma(\beta;\lambda)$. For an $|s| = \epsilon$,

$$\begin{aligned}\Delta_j\Gamma(Z;\beta) &= (\sum_{i=1}^n L(Z_i;\beta + s1_j) - \sum_{i=1}^n L(Z_i;\beta)) + \lambda(T(\beta + s1_j) - T(\beta)) \\ &:= \Delta_j(\sum_{i=1}^n L(Z_i;\beta)) + \lambda\Delta_j T(\beta). \qquad (13)\end{aligned}$$

Since $T(\beta)$ is simply the L_1 norm of β, $\Delta T(\beta)$ reduces to a simple form:

$$\begin{aligned}\Delta_j T(\beta) &= \|\beta + s1_j\|_1 - \|\beta\|_1 = |\beta_j + s| - |\beta_j| \\ &= \text{sign}^+(\beta_j, s) \cdot \epsilon \qquad (14)\end{aligned}$$

where $\text{sign}^+(\beta_j, s) = 1$ if $s\beta_j > 0$ or $\beta_j = 0$, $\text{sign}^+(\beta_j, s) = -1$ if $s\beta_j < 0$ and $\text{sign}^+(\beta_j, s) = 0$ if $s = 0$.

Equation (14) shows that an ϵ step's impact on the penalty is a fixed ϵ for any j. Only the sign of the impact may vary. Suppose given a β, the "forward" steps for different j have impacts on the penalty of the same sign, then $\Delta_j T$ is a constant in (13) for all j. Thus, minimizing the Lasso loss using fixed-size steps is equivalent to minimizing the empirical loss directly. At the "early" stages of FSF, all forward steps are parting from zero, therefore all the signs of the "forward" steps' impact on penalty are positive. As the algorithm proceeds into "later" stages, some of the signs may change into negative and minimizing the empirical loss is no longer equivalent to minimizing the Lasso loss. Hence in the beginning, FSF carries out a steepest descent algorithm that minimizes the Lasso loss and follows Lasso's regularization path, but as it goes into later stages, the equivalence is broken and they part ways. In fact, except for special cases like orthogonal designed covariates, FSF usually go into "later" stages, then the signs of impacts on penalty on some directions can change from positive to negative. These directions then reduce the empirical loss and penalty simultaneously therefore they should be preferred over other directions. Moreover, there can also be occasions where a step goes "backward" to reduce the penalty with a small sacrifice in empirical loss. In general, to minimize the Lasso loss, one needs to go "back and forth" to trade off the penalty with empirical loss for different regularization parameters. We call a direction that leads to reduction of the penalty a "backward" direction and define a backward step as the following:

For a given $\hat\beta$, a **backward step** is such that:

$$\Delta\hat\beta = s_j 1_j,\ \text{subject to } \hat\beta_j \neq 0,\ \text{sign}(s) = -\text{sign}(\hat\beta_j) \text{ and } |s| = \epsilon.$$

Making such a step will reduce the penalty by a fixed amount $\lambda \cdot \epsilon$, but its impact on the empirical loss can be different, therefore as in (10) we want:

$$\hat j = \arg\min_j \sum_{i=1}^n L(Z_i; \hat\beta + s_j 1_j) \text{ subject to } \hat\beta_j \neq 0 \text{ and } s_j = -\text{sign}(\hat\beta_j)\epsilon,$$

i.e. $\hat j$ is picked such that the empirical loss after making the step is as small as possible.

While forward steps try to reduce the Lasso loss through minimizing the empirical loss, the backward steps try to reduce the Lasso loss through minimizing the Lasso penalty. Although rare, it is possible to have a step reduce both the empirical loss and the Lasso penalty. It thus works both as a forward step and a backward step. We do not distinguish such steps as they do not create any confusions.

In summary, by allowing the backward steps, we are able to work with the Lasso loss directly and take backward steps to correct earlier forward steps that are too greedy.

5. Least Squares Problem

For the most common particular case – least squares regression, the forward and backward steps in BLasso become very simple and more intuitive. To see this, we write out the empirical loss function $L(Z_i;\beta)$ in its L_2 form,

$$\sum_{i=1}^n L(Z_i;\beta) = \sum_{i=1}^n (Y_i - X_i\beta)^2 = \sum_{i=1}^n (Y_i - \hat Y_i)^2 = \sum_{i=1}^n \eta_i^2. \tag{15}$$

where $\hat Y = (\hat Y_1, ..., \hat Y_n)^T$ are the "fitted values" and $\eta = (\eta_1, ..., \eta_n)^T$ are the "residuals".

Recall that in a penalized regression setup $X_i = (X_{i1}, ..., X_{im})$, every covariates $X^j = (X_{1j}, ..., X_{nj})^T$ is normalized, i.e. $\|X^j\|^2 = \sum_{i=1}^n X_{ij}^2 = 1$ and $\sum_{i=1}^n X_{ij} = 0$. For a given $\beta = (\beta_1, ...\beta_m)^T$, the impact of a step s of size $|s| = \epsilon$ along β_j on the empirical loss function can be written as:

$$\begin{aligned}\Delta(\sum_{i=1}^n L(Z_i;\beta)) &= \sum_{i=1}^n [(Y_i - X_i(\beta + s1_j))^2 - (Y_i - X_i\beta)^2] \\ &= \sum_{i=1}^n [(\eta_i - sX_i 1_j)^2 - \eta_i^2] = \sum_{i=1}^n (-2s\eta_i X_{ij} + s^2 X_{ij}^2) \\ &= -2s(\eta \cdot X^j) + s^2. \end{aligned} \tag{16}$$

The last line of these equations delivers a strong message – in least squares regression, given the step size, the impact on the empirical loss function is solely determined by the inner-product (correlation) between the fitted residuals and each coordinate. Specifically, it is proportional to the negative inner-product between the fitted residuals and the covariate plus the step size squared. Therefore coordinate steepest descent with a fixed step size on the empirical loss function is equivalent to finding the covariate that is best correlated with the fitted residuals at each step, then proceed along the same direction. This is in principle the same as FSF.

Translate this for the forward step where originally

$$(\hat{j}, \hat{s}_{\hat{j}}) = \arg\min_{j, s=\pm\epsilon} \sum_{i=1}^n L(Z_i; \beta + s1_j),$$

we get

$$\hat{j} = \arg\max_j |\eta \cdot X^j| \quad \text{and} \quad \hat{s} = \text{sign}(\eta \cdot X^{\hat{j}})\epsilon, \tag{17}$$

which coincides exactly with the stagewise procedure described in Efron (2002) and is in general the same principle as L_2 Boosting, i.e. recursively refitting the regression residuals along the most correlated direction except the difference in step size choice (Friedman 2001, Buhlmann and Yu 2003). Also, under this simplification, a backward step becomes

$$\hat{j} = \arg\min_j(-s(\eta \cdot X^j)) \text{ subject to } \hat{\beta}_j \neq 0 \text{ and } s_j = -\text{sign}(\hat{\beta}_j)\epsilon. \tag{18}$$

Clearly that both forward and backward steps are based only on the correlations between fitted residuals and the covariates. It follows that BLasso in the L_2 case reduces to finding the best direction in both forward and backward directions by examining the inner-products, and then deciding whether to go forward or backward based on the regularization parameter. This not only simplifies the minimization procedure but also significantly reduces the computation complexity for large datasets since the inner-product between η_t and X_j can be updated by

$$(\eta^{t+1})'X_j = (\eta^t - sX_{\hat{j}^t})'X_j = (\eta^t)'X_j - sX_{\hat{j}^t}'X_j. \tag{19}$$

which takes only one operation if $X_{\hat{j}^t}'X_j$ is precalculated.

Therefore, when the number of base learners is small, based on precalculated $X'X$ and $Y'X$, BLasso without the original data by using (19) which makes its computation complexity independent from the number of observations. When the number of base learners is large (or infinite), this strategy because inefficient (or unsuitable). The forward step is then done by a sub-optimization procedure e.g. fitting smoothing splines. For the backward step, only inner-products between base learners with nonzero coefficients need to be calculated once. Then the inner products between these base learners and residuals can be updated by (19). This makes the backward steps' computation complexity proportional to the number of base learners that are already chosen instead of the number of all possible base learners. Therefore BLasso works efficiently not only for cases with large sample size but also for cases where a class of a large or an infinite number of possible base learners is given but we only look for a sparse model.

As mentioned earlier, there are already established algorithms for solving the least square Lasso problem e.g. Shooting algorithm (Fu 1998), and LARS (Efron et al. 2004). These algorithms are efficient for giving the exact Lasso path when the number of base learners is small but neither algorithm is adequate for nonparametric learning problems with a large or an infinite number of base learners. Also, given the entire Lasso path, one solution needs to be selected. Previous algorithms suggest using (generalized) cross validation procedures that minimized a (generalized) cross validation loss over a grid of λ. BLasso, as these strategies still apply, can also take advantage of early stopping as in the original boosting literature and potentially saves computation from further unnecessary iterations. Particularly, in cases where the Ordinary Least Square (OLS) model performs well, BLasso can be modified to start from the OLS model, go backward and stop in a few iterations. Overall, for least squares problem, BLasso is an algorithm with Boosting's computational efficiency and flexibility but also produces Lasso solutions as LARS and Shooting algorithms.

6. Boosted Lasso for Nonparametric Learning

As we discussed for the least squares problem, Boosted Lasso can be used to deal with nonparametric learning problems with a large or an infinite number of base learners. This is because Boosted Lasso has two types of steps: forward and backward, for both of which a large or an infinite number of base learners does not create a problem.

The forward steps are similar to the steps in Boosting which can handle a large or an infinite number of base learners. That is, the forward step, as in Boosting, is a sub-optimization problem by itself and Boosting's functional gradient descend strategy applies and this functional gradient strategy works even with a large or an infinite number of base learners. For example, in the case of classification with trees, one can use the classification margin or the logistic loss function as the loss function and use a reweighting procedure to find the appropriate tree at each step (for details see e.g. Breiman 1998 and Friedman et al. 2000). In case of regression with the L^2 loss function, the minimization as in (11) is equivalent to refitting the residuals as we described in the last section.

The backward steps consider only those base learners already in the model and the number of these learners is always finite. This is because for an iterative procedure like Boosted Lasso, we usually stop early to avoid overfitting and to get a sparse model. This results in a finite number of base learners altogether. And even if the algorithm is kept running, it usually reaches a close-to-perfect fit without too many iterations (i.e. the "strong learnability" in Schapire 1990). Therefore, the backward step's computation complexity is limited because it only involves base learners that are already included from previous steps.

There is, however, a difference in the BLasso algorithm between the case with a small number of base learners and the one with a large or an infinite number of base learners. For the finite case, since BLasso requires a backward step to be strictly descending and relax λ whenever no descending step is available, therefore BLasso never reach the same solution more than once and the tolerance constant ξ can be set to 0 or a very small number to accommodate the program's numerical accuracy. In the nonparametric learning case, oscillation could occur when BLasso keeps going back and forth in different directions but only improving the penalized loss function by a diminishing amount, therefore a positive tolerance ξ is mandatory. Since a step's impact on the penalized loss function is on the order of ϵ, we suggest using $\xi = c \cdot \epsilon$ where $c \in (0, 1)$ is a small constant.

7. Generalized Boosted Lasso

As stated earlier, BLasso not only works for general convex loss functions, but also extends to convex penalties other than the L_1 penalty. For the Lasso problem, BLasso does a fixed step size coordinate descent to minimize the penalized loss. Since the penalty has the special L_1 norm and (14) holds, a step's impact on the penalty has a fixed size ϵ with either a positive or a negative sign, and the coordinate descent takes form of "backward" and "forward" steps. This reduces the minimization of the penalized loss function to unregularized minimizations of the loss function as in (11) and (10). For general convex penalties, since a step on different coordinates does not necessarily have the same impact on the penalty, one is forced to work with the penalized function directly. Assume $T(\beta)$: $R^m \to R$ is a convex penalty function. We now describe the Generalized Boosted Lasso algorithm:

Generalized Boosted Lasso

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, ..., n$ and a fixed small step size $\epsilon > 0$ and a small tolerance parameter $\xi > 0$, take an initial forward step

$$(\hat{j}, \hat{s}_{\hat{j}}) = \arg \min_{j, s=\pm\epsilon} \sum_{i=1}^{n} L(Z_i; s1_j),\ \hat{\beta}^0 = \hat{s}_{\hat{j}} 1_{\hat{j}}.$$

Then calculate the corresponding regularization parameter

$$\lambda^0 = \frac{\sum_{i=1}^n L(Z_i; 0) - \sum_{i=1}^n L(Z_i; \hat{\beta}^0)}{T(\hat{\beta}^0) - T(0)}.$$

Set $t = 0$.
Step 2 (steepest descent on Lasso loss). Find the steepest coordinate descent direction on the penalized loss

$$(\hat{j}, \hat{s}_{\hat{j}}) = \arg \min_{j, s=\pm\epsilon} \Gamma(\hat{\beta}^t + s1_j; \lambda^t).$$

Update $\hat{\beta}$ if it reduces Lasso loss; otherwise force $\hat{\beta}$ to minimize L and recalculate the regularization parameter. That is if $\Gamma(\hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}; \lambda^t) - \Gamma(\hat{\beta}^t, \lambda^t) < -\xi$, then

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}},\ \lambda^{t+1} = \lambda^t.$$

Otherwise,

$$(\hat{j}, \hat{s}_{\hat{j}}) \quad = \quad \arg \min_{j, |s|=\epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j),$$

$$\begin{aligned}\hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}},\\ \lambda^{t+1} &= \min[\lambda^t, \frac{\sum_{i=1}^n L(Z_i;\hat{\beta}^t) - \sum_{i=1}^n L(Z_i;\hat{\beta}^{t+1})}{T(\hat{\beta}^{t+1}) - T(\hat{\beta}^t)}].\end{aligned}$$

Step 3 (iteration). Increase t by one and repeat Step 2 and 3. Stop when $\lambda^t \leq 0$.

In the Generalized Boosted Lasso algorithm, explicit "forward" or "backward" steps are no longer seen. However, the mechanic remains the same – minimize the penalized loss function for each λ, relax the regularization by reducing λ through a "forward" step when the minimum of the loss function for the current λ is reached.

Another method that works in a similar fashion is the Barrier Method proposed by Fiacco and McCormick (1968) (cf. Boyd and Vandenberghe, 2004). It is an advanced optimization method for constrained convex minimization problems which solves a sequence of unconstrained minimization problems that approach the constrained problem. We note that BLasso also solves a sequence of optimization problems indexed by λ. Both methods use the last solution found as the starting point for the next unconstrained minimization problem. However, there exists a fundamental difference: the Barrier Method targets to get to the end of the path ($\lambda = 0$) so it jumps quickly from solution to solution according to a geometric schedule. In comparison, BLasso is designed to produce a well sampled solution path. It therefore places solution points evenly on the path (more discussions will be offered when we compare BLasso with the algorithm in Rosset (2004) in the discussion section). This subtle connection between BLasso and the extensively studied Barrier Method provides opportunities to explore further convergence properties of BLasso and to study the interplay between optimization and statistical estimation theories.

8. Experiments

In this section, three experiments are carried out to implement BLasso. The first experiment runs BLasso on the diabetes dataset (cf. Efron et al. 2004) with an added artificial covariate variable under the classical Lasso setting, i.e. L_2 regression with an L_1 penalty. This added covariate is strongly correlated with a couple of the original covariates. In this case, BLasso is seen to produce a path almost exactly the same as the Lasso path, while FSF's path parts drastically from Lasso's due to the added strongly correlated covariate.

In the second experiment, we use the diabetes dataset for L_2 regression without the added covariate but with several different penalties, therefore Generalized BLasso is used. The penalties are bridge penalties (cf. Fu 1998) with different bridge parameters $\gamma = 1$, $\gamma = 1.1$, $\gamma = 2$, $\gamma = 4$ and $\gamma = \infty$. Generalized BLasso produced the solution paths successfully for all cases.

The last experiment is on classification where we use simulated data to illustrate BLasso's solving regularized classification problem under the 1-norm SVM setting. Here, since the loss function is not continuously differentiable, it is theoretically possible that BLasso does not converge to the true regularization path. However, as we illustrate, BLasso runs without a problem and produces reasonable solutions.

8.1. L_2 Regression with L_1 Penalty (Classical Lasso)

The dataset used in this experiment is from a Diabetes study where 442 diabetes patients were measured on 10 baseline variables. A prediction model was desired for the response variable, a quantitative measure of disease progression one year after baseline. One additional variable, $X^{11} = -X^7 + X^8 + 5X^9 + e$ where e is i.i.d. Gaussian noise (mean zero and $\mathrm{Var}(e) = 1/442$), is added to make the difference between FSF and Lasso solutions more visible. This added covariate is strongly correlated with X^9, with correlations as follows (in the order of $X^1, X^2, ..., X^{10}$) :

$$(0.25\ ,\ 0.24\ ,\ 0.47\ ,\ 0.39\ ,\ 0.48\ ,\ 0.38\ ,\ -0.58\ ,\ 0.76\ ,\ 0.94\ ,\ 0.47).$$

The classical Lasso – L_2 regression with L_1 penalty is used for this purpose. Let $X^1, X^2, ..., X^m$ be $n-$vectors representing the covariates and Y the vector of responses for the n cases, $m = 11$ and $n = 442$ in this study. Location and scale transformations are made so that all covariates are standardized to have mean 0 and unit length, and the response has mean zero.

The penalized loss function has the form:

$$\Gamma(\beta;\lambda) = \sum_{i=1}^{n}(Y_i - X_i\beta)^2 + \lambda\|\beta\|_1 \tag{20}$$

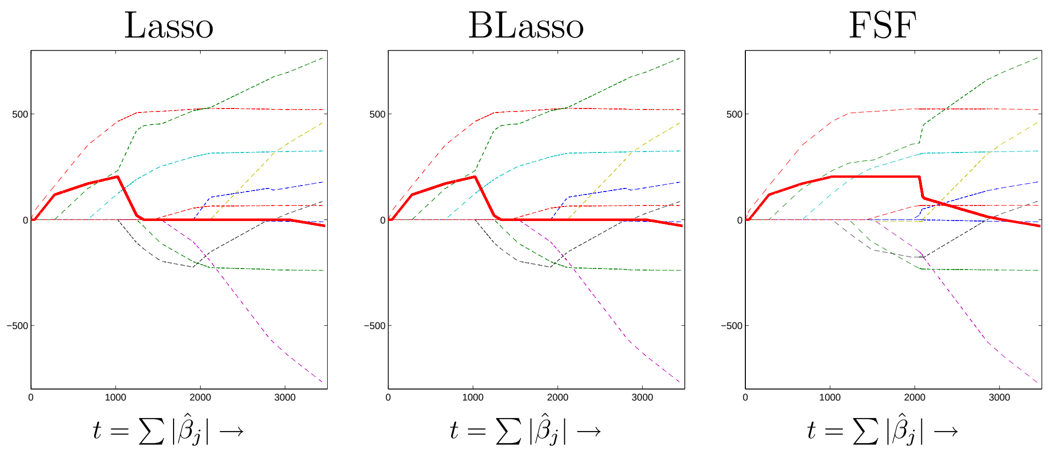

Figure 1. Estimates of regression coefficients $\hat{\beta}_j$, j=1,2,...10,11 for the diabetes data. *Left Panel:* Lasso solutions (produced using simplex search method on the penalized empirical loss function for each λ) as a function of $t = \|\beta\|_1$. *Middle Panel:* BLasso solutions, which can be seen indistinguishable to the Lasso solutions. *Right Panel:* FSF solutions, which are different from Lasso and BLasso.

The middle panel of Figure 1 shows the coefficient plot for BLasso applied to the modified diabetes data. Left (Lasso) and Middle (Middle) panels are indistinguishable from each other. Both FSF and BLasso pick up the added artificial and strongly correlated X^{11} (the solid line) in the earlier stages, but due to the greedy nature of FSF, it is not able to remove X^{11} in the later stages thus every parameter estimate is affected leading to significantly different solutions from Lasso.

The BLasso solutions were built up in 8700 steps (making the step size $\epsilon = 0.5$ small so that the coefficient paths are smooth), 840 of which were backward steps. In comparison, FSF took 7300 pure forward steps. BLasso's backward steps mainly concentrate around the steps where FSF and BLasso tend to differ.

8.2. L_2 Regression with L_γ Penalties (the Bridge Regression)

To demonstrate the Generalized BLasso algorithm or different penalties. We use the Bridge Regression setting with the diabetes dataset (without the added covariate in the first experiment). The Bridge Regression method was first proposed by Frank and Friedman (1993) and later more carefully discussed and implemented by Fu (1998). It is a generalization of the ridge regression (L_2 penalty) and Lasso (L_1 penalty). It considers a linear (L_2) regression problem with L_γ penalty for $\gamma \geq 1$ (to maintain the convexity of the penalty function).

The penalized loss function has the form:

$$\Gamma(\beta;\lambda) = \sum_{i=1}^{n}(Y_i - X_i\beta)^2 + \lambda\|\beta\|_\gamma \tag{21}$$

where γ is the bridge parameter. The data used in this experiment are centered and rescaled as in the first experiment.

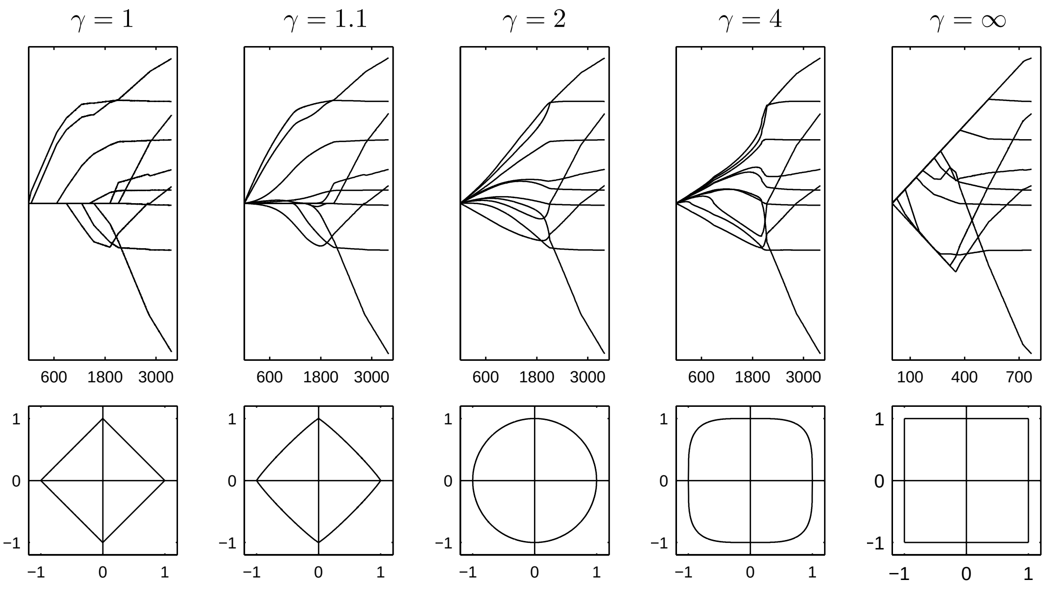

Figure 2. *Upper Panel*: Solution paths produced by BLasso for different bridge parameters. , from left to right: Lasso ($\gamma = 1$), near-Lasso ($\gamma = 1.1$), Ridge ($\gamma = 2$), over-Ridge ($\gamma = 4$), max ($\gamma = \infty$). The Y-axis is the parameter estimate and has the range $[-800, 800]$. The X-axis for each of the left 4 plots is $\sum_i |\beta_i|$, the one for the 5th plot is $\max(|\beta_i|)$ because $\sum_i |\beta_i|$ is unsuitable. *Lower Panel*: The corresponding penalty equal contours for $\beta_1^\gamma + \beta_2^\gamma = 1$.

Generalized BLasso successfully produced the paths for all 5 cases which are verified by pointwise minimization using simplex method ($\gamma = 1$, $\gamma = 1.1$, $\gamma = 4$ and $\gamma = max$) or close form solutions ($\gamma = 2$). It is interesting to notice the phase transition from the near-Lasso to the Lasso as the solution paths are similar but only Lasso has sparsity. Also, as γ grows larger, estimates for different β_j tend to have more similar sizes and in the extreme $\gamma = \infty$ there is a "branching" phenomenon – the estimates stay together in the beginning and branch out into different directions as the path progresses. Since no differentiation is carried out or assumed, Generalized BLasso produced all of these significantly different solution paths by simply plugging in different penalty functions in the algorithm.

8.3. Classification with 1-norm SVM (Hinge Loss)

To demonstrate the Generalized BLasso algorithm for a nondifferentiable loss function, we now look at binary classification. As in Zhu et al. (2003), we generate 50 training data points in each of two classes. The first class has two standard normal independent inputs X^1 and X^2 and class label $Y = -1$. The second class also has two standard normal independent inputs, but conditioned on $4.5 \leq (X^1)^2 + (X^2)^2 \leq 8$ and has class label $Y = 1$. We wish to find a classification rule from the training data. so that when given a new input, we can assign a class Y from $\{1, -1\}$ to it.

1-norm SVM (Zhu et al. 2003) is used to estimate β:

$$(\hat{\beta}_0, \beta) = \arg\min_{\beta_0,\beta} \sum_{i=1}^{n} (1 - Y_i(\beta_0 + \sum_{j=1}^{m} \beta_j h_j(X_i)))^+ + \lambda \sum_{j=1}^{5} |\beta_j| \tag{22}$$

where $h_i \in D$ are basis functions and λ is the regularization parameter. The dictionary of basis functions is $D = \{\sqrt{2}X^1, \sqrt{2}X^2, \sqrt{2}X^1X^2, (X^1)^2, (X^2)^2\}$. Notice that β_0 is left unregularized so the penalty function is not the L_1 penalty.

The fitted model is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{j=1}^{m} \hat{\beta}_j h_j(x),$$

and the classification rule is given by $\text{sign}(\hat{f}(x))$.

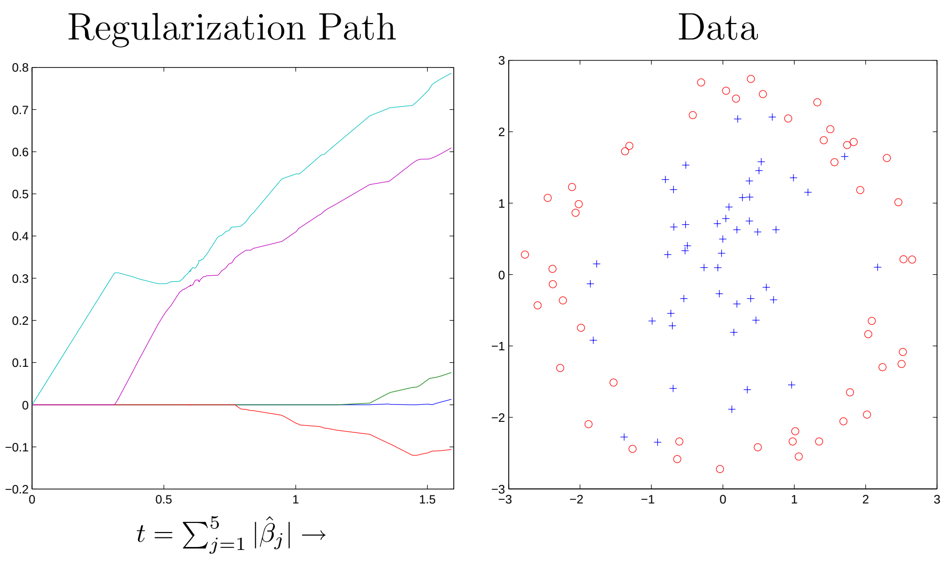

Figure 3. Estimates of 1-norm SVM coefficients $\hat{\beta}_j$, j=1,2,...,5, for the simulated two-class classification data. *Left Panel* BLasso solutions as a function of $t = \sum_{j=1}^{5} |\hat{\beta}_j|$. *Right Panel* Scatter plot of the data points with labels: '+' for $y = -1$; 'o' for $y = 1$.

Since the loss function is not differentiable and the penalty function is not the original Lasso penalty, we do not have a theoretical guarantee that BLasso works. Nonetheless the solution path produced by Generalized BLasso has the same sparsity and piecewise linearality as the 1-norm SVM solutions shown in Zhu et al. (2003). It takes Generalized BLasso 490 iterations to generate the solutions. The covariates enter the regression equation sequentially as t increase, in the following order: the two quadratic terms first, followed by the interaction term then the two linear terms. As 1-norm SVM in Zhu et al. (2003), BLasso correctly picked up the quadratic terms early. The interaction term and linear terms that are not in the true model comes up much later. In other words, BLasso results are in good agreement with Zhu et al.'s 1-norm SVM results and we regard this as a confirmation for BLasso's effectiveness in this nondifferentiable example.

9. Discussion and Concluding Remarks

As seen from the experiments, BLasso is effective for solving the Lasso problem and general convex penalized loss minimization problems. One practical issue left undiscussed is the choice of the step size ϵ. In general, BLasso takes $O(1/\epsilon)$ steps to produce the whole path. Depending on the actual loss function, base learners and minimization method used in each step, the actual computation complexity varies. Although choosing a smaller step size gives smoother solution path and more accurate estimates, we observe that the the actual coefficient estimates are pretty accurate even for relatively large step sizes.

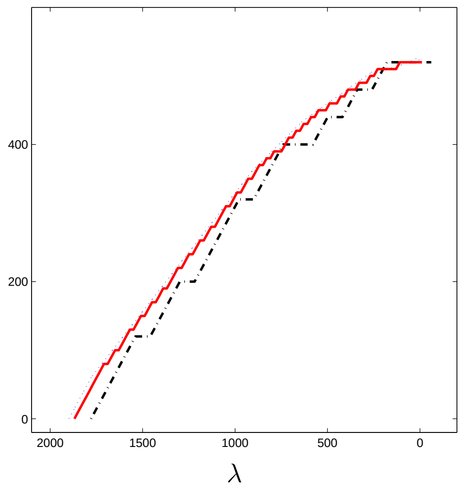

Figure 4. Estimates of regression coefficients $\hat{\beta}_3$ for the diabetes data. Solutions are plotted as functions of λ. *Dotted Line:* Estimates using step size $\epsilon = 0.05$. *Solid Line:* Estimates using step size $\epsilon = 10$. *Dash-dot Line:* Estimates using step size $\epsilon = 50$.

For the diabetes data, using a small step size $\epsilon = 0.05$, the solution path can not be distinguished from the exact regularization path. However, even when the step size is as large as $\epsilon = 10$ and $\epsilon = 50$, the solutions are still good approximations.

BLasso has only one step size parameter. This parameter controls both how close BLasso approximates the minimization coefficients for each λ and how close two adjacent λ on the regularization path are placed. As can be seen from Figure 4, a smaller step size leads to a closer approximation to the solutions and also finer grids for λ. We argue that, if λ is sampled on a coarse grid we should not spend computational power on finding a much more accurate approximation of the coefficients for each λ. Instead, the available computational power spent on these two coupled tasks should be balanced. BLasso's 1-parameter setup automatically balances these two aspects of the approximation which is graphically expressed by the staircase shape of the solution paths.

Another algorithm similar to Generalized BLasso is developed independently by Rosset (2004). There, starting from $\lambda = 0$, a solution is generated by taking a small Newton-Raphson step for each λ, then λ is increased by a fixed amount. The algorithm assumes twice-differentiability of both loss function and penalty function and involves calculation of the Hessian matrix. It works in a very similar fashion as the Barrier Method with a linear update schedule for λ (Barrier Method uses a geometric schedule). The Barrier Method usually uses a second order optimization method for each λ as well. In comparison, BLasso uses only the differences of the loss function and involves only basic operations. BLasso's step size is defined in the original parameter space which makes the solutions evenly spread in β's space rather than in λ. In general, since λ is approximately the reciprocal of size of the penalty, as fitted model grows larger and λ becomes smaller, changing λ by a fixed amount makes the algorithm in Rosset (2004) move too fast in the β space. On the other hand, when the model is close to empty and the penalty function is very small, λ is very large, but the algorithm still uses same small steps thus computation is spent to generate solutions that are too close from each other.

One of our current research topics is to apply BLasso in an online or time series setting. Since BLasso has both forward and backward steps, we believe that an adaptive online learning algorithm can be devised based on BLasso so that it goes back and forth to track the best regularization parameter and the corresponding model.

We end with a summary of our main contributions:

(a) By combining both forward and backward steps, a Boosted Lasso (BLasso) algorithm is constructed to minimize an L_1 penalized convex loss function. While it maintains the simplicity and flexibility of Boosting (or Forward Stagewise Fitting) algorithms, BLasso efficiently produces the Lasso solutions for general loss functions and large classes of base learners. This can be proven rigorously under the assumption that the loss function is convex and continuously differentiable.

(b) The backward steps introduced in this paper are critical for producing the Lasso path. Without them, the FSF algorithm can be too greedy and in general does not produce Lasso solutions, especially when the base learners are strongly correlated as in cases where the number of base learners is larger than the number of observations.

(c) When the loss function is squared loss, BLasso takes a simpler and more intuitive form and depends only on the inner products of variables. For problems with large number of observations and fewer base learners, after initialization, BLasso does not depend on the number of observations. For nonparametric learning problems where previous methods do not cover due to a large or an infinite number of base learners, BLasso remains computationally efficient since the forward steps are boosting steps and the backward steps deal only with base learners that have already been included in the model.

(d) We generalize BLasso to deal with other convex penalties (generalized BLasso) and show a connection with the Barrier Method for constrained convex optimization.

A. Appendix: Proofs

First, we offer a proof for Lemma 1.

PROOF. (Lemma 1)

(a) Suppose $\exists \lambda, j, |s| = \epsilon$ s.t. $\Gamma(\epsilon 1_j; \lambda) \le \Gamma(0; \lambda)$. We have

$$\sum_{i=1}^{n} L(Z_i; 0) - \sum_{i=1}^{n} L(Z_i; s1_j) \ge \lambda T(s1_j) - \lambda T(0),$$

therefore

$$\begin{aligned} \lambda &\le \frac{1}{\epsilon}\{\sum_{i=1}^{n} L(Z_i; 0) - \sum_{i=1}^{n} L(Z_i; s1_j)\} \\ &\le \frac{1}{\epsilon}\{\sum_{i=1}^{n} L(Z_i; 0) - \min_{j,|s|=\epsilon} \sum_{i=1}^{n} L(Z_i; s1_j)\} \\ &= \frac{1}{\epsilon}\{\sum_{i=1}^{n} L(Z_i; 0) - \sum_{i=1}^{n} L(Z_i; \hat\beta^0)\} \\ &= \lambda^0. \end{aligned}$$

(b) Since a backward step is only taken when $\Gamma(\hat\beta^{t+1}; \lambda^t) < \Gamma(\hat\beta^t; \lambda^t)$, so we only need to consider forward steps. When a forward step is forced, if $\Gamma(\hat\beta^{t+1}; \lambda^t) > \Gamma(\hat\beta^t; \lambda^t)$, then

$$\sum_{i=1}^{n} L(Z_i; \hat\beta^t) - \sum_{i=1}^{n} L(Z_i; \hat\beta^{t+1}) < \lambda^t T(\hat\beta^{t+1}) - \lambda^t T(\hat\beta^t),$$

therefore

$$\lambda^{t+1} = \frac{1}{\epsilon}\{\sum_{i=1}^{n} L(Z_i; \hat\beta^t) - \sum_{i=1}^{n} L(Z_i; \hat\beta^{t+1})\} < \lambda^t$$

which contradicts the assumption.

(c) Since $\lambda^{t+1} < \lambda^t$ and λ can not be relaxed by a backward step, we immediately have $\|\hat\beta^{t+1}\|_1 = \|\hat\beta^t\|_1 + \epsilon$. Then from

$$\lambda^{t+1} = \frac{1}{\epsilon}\{\sum_{i=1}^{n} L(Z_i; \hat\beta^t) - \sum_{i=1}^{n} L(Z_i; \hat\beta^{t+1})\}$$

we get

$$\Gamma(\hat\beta^t; \lambda^{t+1}) = \Gamma(\hat\beta^{t+1}; \lambda^{t+1}).$$

Plus both sides by $\lambda^t - \lambda^{t+1}$ times the penalty term, and recall $T(\hat\beta^{t+1}) = \|\hat\beta^{t+1}\|_1 > |\hat\beta^t\|_1 = T(\hat\beta^t)$, we get

$$\begin{aligned} \Gamma(\hat\beta^t; \lambda^t) &< \Gamma(\hat\beta^{t+1}; \lambda^t) \\ &= \min_{j,|s|=\epsilon} \Gamma(\hat\beta^t + s1_j; \lambda^t) \\ &\le \Gamma(\hat\beta^t \pm \epsilon 1_j; \lambda^t) \end{aligned}$$

for $\forall j$. This completes the proof.

Theorem 1 claims "the BLasso path converges to the Lasso path", by which we mean:

(a) As $\epsilon \to 0$, for $\forall t$ s.t. $\lambda^{t+1} < \lambda^t$, $\Gamma(\hat{\beta}^t; \lambda^t) - \min_\beta \Gamma(\beta; \lambda^t) = o(1)$.
(b) For each $\epsilon > 0$, it takes finite steps to run BLasso.

Proof. (Theorem 1)

(a) Since $L(Z; \beta)$ is strictly convex in β, so $\Gamma(\beta; \lambda)$ is strictly convex in β for $\forall\lambda$. So $\Gamma(\beta; \lambda)$ has unique minimum $\beta^*(\lambda)$. For j, that $\beta_j^*(\lambda) \neq 0$, Γ is differentiable in that direction and $d\Gamma = 0$, for the j that $\beta_j^*(\lambda) = 0$, the right and left partial derivative of Γ w.r.t. β_j are both bigger than or equal to 0 and at least one of them is strictly bigger than 0. Now, recall Lemma 1 says BLasso does steepest coordinate descent with fixed step size ϵ. We want to show it does not get stuck near a nonstationary point as $\epsilon \to 0$.
Consider at $\hat{\beta}^t$, we look for the steepest descent on the surface of a polytope defined by $\beta = \hat{\beta}^t + \Delta\beta$ where $\|\Delta\beta\|_1 = \epsilon$, i.e.

$$\min_{\Delta\beta} \Delta\Gamma(\hat{\beta}^t + \Delta\beta; \lambda) \text{ subject to } \|\Delta\beta\|_1 = \epsilon. \tag{23}$$

Here
$$\Delta\Gamma = \Delta L + \lambda\Delta T.$$

Since L is continuously differentiable w.r.t. β, we have

$$\Delta L = \sum_j \frac{\partial L}{\partial \beta_j} \Delta\beta_j + o(\epsilon). \tag{24}$$

And since $T(\beta) = \sum_j |\beta_j|$, we have

$$\Delta T = \sum_j \text{sign}^+(\hat{\beta}^t, \Delta\beta_j) \|\Delta\beta_j\|. \tag{25}$$

Therefore as $\epsilon \to 0$, (23) becomes a linear programming problem. This indicates, near a nonstationary point, if there is a descending point on the polytope then there is a descending direction on the vertices. More precisely, if $\exists\Delta\beta$ such that $\Delta\Gamma(\hat{\beta}^t + \Delta\beta; \lambda) < 0$ and is not $o(\epsilon)$, then there exists a descending direction j. This contradicts with Lemma 1(c).
Therefore, for any j, either $\Delta\Gamma(\hat{\beta}^t + \Delta\beta_j; \lambda) = o(\epsilon)$ or it is not $o(\epsilon)$ but less than zero in both positive and negative directions where the later corresponds to zero entries in $\hat{\beta}_t$. Thus, by the continuous differentiability assumption, $\hat{\beta}_t$ is in the $o(1)$ neighborhood of $\beta^*(\lambda_t)$ which implies $\Gamma(\hat{\beta}^t; \lambda^t) - \Gamma(\beta^*(\lambda^t); \lambda^t) = o(1)$. This concludes the proof.
(b) First, suppose we have $\lambda^{t+1} < \lambda^t$, $\lambda^{t'+1} < \lambda^{t'}$ and $t < t'$. Immediately, we have $\lambda^t > \lambda^{t'}$, then
$$\Gamma(\beta^{t'}; \lambda^{t'}) < \Gamma(\beta^t; \lambda^{t'}) < \Gamma(\beta^t; \lambda^t) < \Gamma(\beta^{t'}; \lambda^t).$$
Therefore
$$\Gamma(\beta^{t'}; \lambda^t) - \Gamma(\beta^{t'}; \lambda^{t'}) > \Gamma(\beta^t; \lambda^t) - \Gamma(\beta^t; \lambda^{t'}),$$
from which we get
$$T(\beta^{t'}) > T(\beta^t).$$

So the BLasso solution before each time λ gets relaxed strictly increases in L_1 norm. Then since the L_1 norm can only change on an ϵ-grid, so λ can only be relaxed finite times till BLasso reaches the unregularized solution.

Now for each value of λ, since BLasso is always strictly descending, the BLasso solutions never repeat. By the same ϵ-grid argument, BLasso can only take finite steps before λ has to be relaxed.

Combining the two arguments, we conclude that for each $\epsilon > 0$ it can only take finite steps to run BLasso.

Acknowledgments. B. Yu would like to gratefully acknowledge the partial supports from NSF grants FD01-12731 and CCR-0106656 and ARO grant DAAD19-01-1-0643, and the Miller Research Professorship in Spring 2004 from the Miller Institute at University of California at Berkeley. Both authors thank Dr. Chris Holmes and Mr. Guilherme V. Rocha for their very helpful comments and discussions on the paper.

References

[1] Boyd, S. and Vandenberghe L. (2004). *Convex Optimization*, Cambridge University Press.

[2] Breiman, L. (1998). “Arcing Classifiers”, *Ann. Statist.* **26**, 801-824.

[3] Buhlmann, P. and Yu, B. (2001). “Boosting with the L2 Loss: Regression and Classification”, *J. Am. Statist. Ass.* **98**, 324-340.

[4] Donoho, D. (2004). “For Most Large Underdetermined Systems of Linear Equations, the minimal l1-norm near-solution approximates the sparsest near-solution”, *Technical reports, Statistics Department, Stanford University.*

[5] Efron, B., Hastie,T., Johnstone, I. and Tibshirani, R. (2004). “Least Angle Regression”, *Ann. Statist.* **32**, 407-499.

[6] Fan, J. and Li, R.Z. (2001). “Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties”, *J. Am. Statist. Ass.* **96**, 1348-1360.

[7] Fiacco, A. V. and McCormick, G. P. (1968). “Nonlinear Programming. Sequential Unconstrained Minimization Techniques”, *Society for Industrial and Applied Mathematics*, 1990. First published in 1968 by Research Analysis Corporation.

[8] Frank, I.E. and Friedman, J.H. (1993), “A Statistical View of Some Chemometrics Regression Tools”, *Technometrics*, **35**, 109-148.

[9] Freund, Y. (1995). “Boosting a weak learning algorithm by majority”, *Information and Computation* **121**, 256-285.

[10] Freund, Y. and Schapire, R.E. (1996). “Experiments with a new boosting algorithm”, *Machine Learning: Proc. Thirteenth International Conference*, 148-156. Morgan Kauffman, San Francisco.

[11] Friedman, J.H., Hastie, T. and Tibshirani, R. (2000). “Additive Logistic Regression: a Statistical View of Boosting”, *Ann. Statist.* **28**, 337-407.

[12] Friedman, J.H. (2001). “Greedy Function Approximation: a Gradient Boosting Machine”, *Ann. Statist.* **29**, 1189-1232.

[13] Fu, W. J. (1998). “Penalized regressions: the Bridge versus the Lasso” *J. Comput. Graph. Statist.* **7**, 397-416.

[14] Fu, W. and Knight, K. (2000). “Asymptotics for lasso-type estimators” *Ann. Statist.* **28**, 1356-1378.

[15] Gao, H. Y. and Bruce, A. G. (1997) “WaveShrink With Firm Shrinkage,” *Statistica Sinica,* **7**, 855-874.

[16] Hansen, M. and Yu, B. (2001). “Model Selection and the Principle of Minimum Description Length”, *J. Am. Statist. Ass.* **96**, 746-774.

[17] Hastie, T., Tibshirani, R. and Friedman, J.H. (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer Verlag, New York.

[18] Mason, L., Baxter, J., Bartlett, P. and Frean, M. (1999). “Functional Gradient Techniques for Combining Hypotheses”, In *Advance in Large Margin Classifiers.* MIT Press.

[19] Rosset, S. (2004). “Tracking Curved Regularized Optimization Solution Paths”, *NIPS 2004.*

[20] Schapire, R.E. (1990). “The Strength of Weak Learnability”. *Machine Learning* **5(2)**, 1997-227.

[21] Tibshirani, R. (1996). “Regression shrinkage and selection via the lasso”, *J. R. Statist. Soc. B*, **58(1)** 267-288.

[22] Zhang, T. and Yu, B.(2003). “Boosting with early stopping: convergence and consistency”, *Ann. Statist.*, to appear.

[23] Zhu, J. Rosset, S., Hastie, T. and Tibshirani, R.(2003) “1-norm Support Vector Machines”, *Advances in Neural Information Processing Systems* **16**. MIT Press.