Boosted Lasso

Peng Zhao PENGZHAO@STAT.BERKELEY.EDU
Department of Statistics
University of Berkeley
367 Evans Hall Berkeley, CA 94720-3860, USA

Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics
University of Berkeley
367 Evans Hall Berkeley, CA 94720-3860, USA

Editor: ???

Abstract

In this paper, we propose the Boosted Lasso (BLasso) algorithm which ties the Boosting algorithm with the Lasso method. BLasso is derived as a coordinate descent method with a fixed 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. The forward step is similar to Boosting and Forward Stagewise Fitting, but the backward step is new and makes the Boosting path to approximate the Lasso path. In the cases of a finite number of base learners and a bounded Hessian of the loss function, when the step size goes to zero, the BLasso path is shown to converge to the Lasso path. For cases with a large number of base learners, our simulations show that since BLasso approximate the Lasso paths, the model estimates are sparser than Forward Stagewise Fitting with equivalent or better prediction performance when the true model is sparse and there are more predictors than the sample size. In addition, we extend BLasso to minimizing a general convex loss penalized by a general convex function. Since BLasso relies only on differeneces not derivatives, we demonstrate this extension as a simple off-the-shelf algorithm for tracing the solution paths of regularization problems.

Keywords: Backward step, Boosting, Convexity, Lasso, Regularization Path

1. Introduction

Lasso (Tibshirani, 1996) is an important idea which has received much attention recently in statistics, signal processing (under the name Basis Pursuit Chen and Donoho, 1994) and machine learning. 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 under various conditions by different authors (see e.g. Osborne et al., 2000a,b; Donoho et al., 2004; Donoho, 2004; Tropp, 2004;

Rosset et al., 2004). Furthermore, the sparse models induced by Lasso are usually more interpretable and therefore 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 gradient descent method leading to an additive model (Breiman, 1998; Mason et al., 1999; Friedman et al., 2000).

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, 2003). 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 (Rosset et al., 2004; Hastie et al., 2001; Efron et al., 2004).

This link between Lasso and FSF is more formally described for the linear regression case through the LARS algorithm (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 general 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 (difference) 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 but with an extra rule to force the model complexity to decrease. By combining backward and forward steps, Boosted Lasso is able to go back and forth to approximate 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. Moreover, BLasso can be proven to converge to the Lasso solutions which have explicit global L_1 regularization for cases with finite number of base learners. In contrast, FSF can be seen as 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 (cf. Hastie et al. 2006 for a recent interpretation of the $\epsilon \to 0$ case).

Three contributions are made in this paper through the BLasso algorithm. First, by introducing the backward step we modify Boosting and FSF to follow the Lasso path and consequently generate models that are sparser with equivalent or better prediction performance in our simulations (with true sparse models). Secondly, by showing convergence of BLasso to the Lasso path, we further tighten the conceptual ties between Boosting, Forward Stagewise Fitting and Lasso that have been considered in previous works. Finally, since BLasso can be generalized to deal with other convex penalties and does not use any derivatives of the Loss function or penalty, we provide the Generalized BLasso algorithm as a simple off-the-shelf method for approximating the regularization path for a general loss function and a general convex penalty. Moreover, we point out the connection between BLasso the Barrier Method (cf. Boyd and Vandenberghe, 2004) in the constrained convex optimization literature.

We would like to note that, for the original Lasso problem, i.e. the least squares problem with an L_1 penalty, algorithms that give the entire Lasso path have been established, namely, the homotopy method by Osborne et al. (2000b) and the LARS algorithm by Efron et al. (2004). For parametric least squares problems, these methods are very efficient as their computational complexity is on the same order as a single Ordinary Least Squares regress. For other problems such as classification and for nonparametric setups like model fitting with trees, FSF has been used as a tool for approximating the Lasso path (Rosset et al., 2004). For such problems, BLasso operates in a similar fashion as FSF but, unlike FSF, BLasso can be shown to converge to the Lasso path quite generally when the step size goes to 0.

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 elaborates on the tie between Boosting, FSF, BLasso and Lasso through a more detailed discussion on solving L_1 penalized least squares problem using BLasso. Section 6 introduces a Generalized BLasso algorithm which deals with general convex penalties. In Section 7, results of experiments with both simulated and real data are reported to demonstrate the attractiveness of BLasso both as a learning algorithm that give sparse models and good prediction and as a simple plug-in method for approximating the regularization path for different convex loss functions and penalties. Finally, Section 8 contains discussions on the choice of step sizes and conjectures the regularization effect of using moderate stepsizes, BLasso for nonparametric learning problems, a summary of the paper, and future research directions.

2. Boosting, Forward Stagewise Fitting and the Lasso

Originally Boosting was proposed as an iterative fitting procedure that builds up a model stage by stage and then does a weighted averaging. 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). We now give a brief gradient descent view of FSF and Boosting algorithms followed by a review of the Lasso.

2.1 Boosting and Forward Stagewise Fitting

The boosting algorithms can be seen as functional gradient descent techniques (Breiman, 1998; Mason et al., 1999; Friedman et al., 2000). 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 re-parametrize 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):

$$(\hat{j}, \hat{g}) = \arg\min_{j,g} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + g1_j) \tag{5}$$

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{g}1_{\hat{j}} \tag{6}$$

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 (2003). 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):

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

For general loss functions, FSF can be defined by removing the minimization over g in (5):

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

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}1_{\hat{j}}, \tag{8}$$

This description looks different from the FSF described in Efron et al. (2004), 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 local 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 $\Gamma(\beta; \lambda)$ 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. Osborne et al., 2000a,b; Efron et al., 2004; Donoho et al., 2004; Donoho, 2004; Tropp, 2004; Rosset et al., 2004). (Sparsity can also result from other penalties as in e.g. 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. Very efficient algorithms have been proposed to give the entire regularization path for the squared loss function (the homotopy method by Osborne et al. 2000b and similarly LARS by Efron et al. 2004) and SVM (1-norm SVM by Zhu et al. 2003).

However, it remains open how to give the entire regularization path of the Lasso problem for general convex loss function. 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 is local regularization and does not converge to the Lasso path in general. As can be seen from our simulations (Section 7.2), this results in less sparse solutions comparing to the Lasso ones.

Next we propose the Boosted Lasso (BLasso) algorithm which works in a computationally efficient fashion as FSF. In contrast to FSF, BLasso converges to the Lasso path for general convex loss functions when the step size goes to 0. This relationship between Lasso and BLasso leads to sparser solutions for BLasso comparing to FSF with equivalent or better prediction performance in our simulations using different choices of step sizes.

3. Boosted Lasso

We first describe the BLasso algorithm.

Boosted Lasso (BLasso)

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

If $\Gamma(\hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}; \lambda^t) - \Gamma(\hat{\beta}^t, \lambda^t) \leq -\xi$, then

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

Otherwise,

$$\begin{aligned}
(\hat{j}, \hat{s}) &= \arg\min_{j, s=\pm\epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), && (11)\\
\hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s}1_{\hat{j}}, && (12)\\
\lambda^{t+1} &= \min[\lambda^t, \frac{1}{\epsilon}(\sum_{i=1}^{n} L(Z_i; \hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^{t+1}) - \xi)],\\
I_A^{t+1} &= I_A^t \cup \{\hat{j}\}.
\end{aligned}$$

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.

1. 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 $\lambda^0 \leq \lambda$.
2. For any t, we have $\Gamma(\hat{\beta}^{t+1}; \lambda^t) \leq \Gamma(\hat{\beta}^t; \lambda^t) - \xi$.
3. For $\xi = 0$ and any t such that $\lambda^{t+1} < \lambda^t$, we have $\Gamma(\hat{\beta}^t \pm \epsilon 1_j; \lambda^t) > \Gamma(\hat{\beta}^t; \lambda^t) - \xi$ for every j and $\|\hat{\beta}^{t+1}\|_1 = \|\hat{\beta}^t\|_1 + \epsilon$.

Lemma (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 (b) says that for each value of λ, BLasso performs coordinate descent until there is no descent step. Then, by Lemma (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. The step size ϵ controls fineness of the grid BLasso runs on. The tolerance ξ controls how big a descend need to be made for a backward step to be taken. It is set to be much smaller than ϵ to have a good approximation and can be set to 0 when the number of base learners is finite. Actually, we have a convergence result for BLasso (detailed proof is included in the Appendix):

Theorem 1. For finite number of base learners and $\xi = o(\epsilon)$, if $\sum L(Z_i; \beta)$ is strongly convex with bounded second derivatives in β then as $\epsilon \to 0$, the BLasso path converges to the Lasso path uniformly.

Many popular loss functions, e.g. squared loss, logistic loss and negative log-likelihood functions of exponential families, are convex and twice differentiable. Other functions like the hinge loss (SVM) is continuous and convex but not differentiable. The differentiability, however, is only necessary for the proof of Theorem 1. BLasso does not utilize any gradient or higher order derivatives but only the differences of the loss function therefore remain applicable to loss functions that are not differentiable or of which differentiation is too complex or computationally expensive. It is theoretically possible that BLasso's coordinate descent strategy gets stuck at nondifferentiable points for functions like the hinge loss. However, as illustrated in our third experiment, BLasso may still work for cases like 1-norm SVM empirically.

Theorem 1 does not cover nonparametric learning problems with infinite number of base learners either. In fact, for problems with large or infinite number of base learners, the minimization in (11) is usually done approximately by functional gradient descent and a tolerance $\xi > 0$ needs to be chosen to avoid oscillation between forward and backward steps caused by slow descending. We cover more on this topic in the discussion.

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 pick up more irrelevant variables comparing to the Lasso path in some cases (cf. Figure 1 and Section 7.2). 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 $|s| = \epsilon$,

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

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

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

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. Therefore, 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. 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 seem too greedy and might have picked up variables that irrelevant.

5. Least Squares Problem

Since a large portion of the discussions on the similarity and difference between FSF and Lasso are focused on Least Squares problems (e.g. Efron et al. 2004; Hastie et al. 2001), we describe the BLasso algorithm on this case to echo on some of these literatures.

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:

$$\begin{aligned}\Delta(\sum_{i=1}^{n} L(Z_i;\beta)) &= \sum_{i=1}^{n}[(Y_i - X_i(\beta + s1_j))^2 - (Y_i - X_i\beta)^2] \\ &= \sum_{i=1}^{n}[(\eta_i - sX_i1_j)^2 - \eta_i^2] = \sum_{i=1}^{n}(-2s\eta_i X_{ij} + s^2 X_{ij}^2) \\ &= -2s(\eta \cdot X^j) + s^2. \end{aligned} \tag{16}$$

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.

Translating 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) makes its computation complexity independent from the number of

observations. This nice property is not surprising as it is also observed in established algorithms like LARS and Osborne's homotopy method which are specialized for Least Squares Regression.

For nonparametric learning type of situations, the number of base learners is large therefore the aforementioned strategy becomes inefficient. The forward step is then carried out by a sub-optimization procedure such as fitting trees, smoothing splines or stumps. For the backward step, only inner-products between base learners that have entered the model need to be calculated. 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 large or infinite number of possible base learners is given.

As mentioned earlier, there are already established efficient algorithms for solving the least square Lasso problem, e.g. the homotopy method by Osborne et al. (2000b) and LARS (Efron et al. 2004). These algorithms are very efficient for giving the exact Lasso paths for parametric settings. For nonparametric learning problems with a large or an infinite number of base learners, we believe BLasso is a attractive strategy for approximating the path, as it shares the same computational strategy as Boosting which has proven itself successful in applications. Also, 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.

6. 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 $\xi \geq 0$, 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 :

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,

$$\begin{aligned}
(\hat{j}, \hat{s}_{\hat{j}}) &= \arg\min_{j, |s|=\epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), \\
\hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}, \\
\lambda^{t+1} &= \min[\lambda^t, \frac{\sum_{i=1}^{n} L(Z_i; \hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^{t+1})}{T(\hat{\beta}^{t+1}) - T(\hat{\beta}^t)}].
\end{aligned}$$

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 in the 1960s (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.

7. Experiments

In this section, three experiments are carried out to illustrate the attractivenss of BLasso. The first experiment runs BLasso under the classical Lasso setting on the diabetes dataset

(cf. Efron et al. 2004) often used in studies of Lasso with an added artificial covariate variable to highlight the difference between BLasso and FSF. 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 which shrinks the added irrelevant variable back to zero, while FSF's path parts drastically from Lasso's due to the added strongly correlated covariate and does not move it back to zero.

In the second experiment, we compare the prediction and variable selection performance of FSF and BLasso in a least squares regression simulation using a large number ($p = 500 >> n = 50$) of randomly correlated base learners to mimic the nonparametric learning scenario when the true model is sparse. The result shows, overall, comparing with FSF, BLasso gives sparser solutions with equivalent or better predictions.

The last experiment is to illustrate BLasso as an off-the-shelf method for computing the regularization path for general loss convex functions and general convex penalties. Two cases are presented. The first case is bridge regression (Frank and Friedman, 1993) on diabetes data using different L_γ ($\gamma \geq 1$) norms as penalties. The other is a simulated classification problem using 1-norm SVM (Zhu et al., 2003) with the hinge loss.

7.1 L_2 Regression with L_1 Penalty (Classical Lasso)

The data set 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 the covariates or predictors are standardized to have mean 0 and unit length, and the response has mean zero.

The penalized loss function has the form:

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

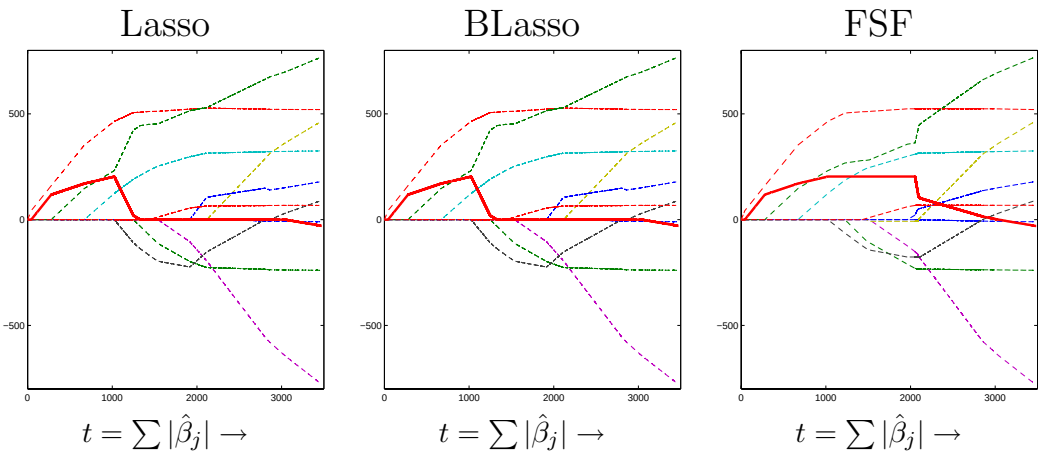

Figure 1. Estimates of regression coefficients $\hat{\beta}_j$, j=1,2,...10,11 for the diabetes data. *Left Panel:* Lasso solutions (produced using simplex search method on the penalized empirical loss function for each λ) as a function of $t = \|\beta\|_1$. *Middle Panel:* BLasso solutions, which can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

7.2 Comparison of Boosted Lasso of Forward Stagewise Fitting by Simulation

In this experiment, we compare the model estimates generated by FSF and BLasso in a large p(=500) and small n(=50) setting to mimic a nonparametric learning scenario where FSF and BLasso are computationally attractive. In this least squares regression simulation, the design is randomly generated as described below to guarantee a fair amount of correlation between the covariates. Otherwise, if the design is close to orthogonal, the FSF and BLasso paths will be too similar for this simulation to yield interesting results.

We first draw 5 covariance matrices C_i, $i = 1, .., 5$ from $.95 \times Wishart(20, 500) + .05 I_{p \times p}$. The Wishart distribution creates a fair amount of correlations (average absolute value is about 0.2) between the covariates and the identity matrix guanrantees C_i to be full rank. For each of the covariance matrix C_i, the design X is then drawn independently from $N(0, C_i)$ with $n = 50$.

The target variable Y is then computed as

$$Y = X\beta + e$$

where β_1 to β_q with $q = 7$ are drawn independently from $N(0, 1)$ and β_8 to β_{500} are set to zero to create a sparse model. e is the gaussian noise vector with mean zero and variance 1. For each of the 5 cases with different C_i, both BLasso and FSF are run using stepsizes $\epsilon = \frac{1}{5},\ \frac{1}{10},\ \frac{1}{20},\ \frac{1}{40}$ and $\frac{1}{80}$.

To compare the performances, we examine the solutions on the solutions paths that give the smallest mean squared error $\|X\beta - X\hat{\beta}\|^2$. The mean squared error of these solutions are recorded on the log scale together with the number of nonzero estimates in each solution. All cases are run 50 times and the average results are reported in Table 1.

Design			$\epsilon = \frac{1}{5}$	$\epsilon = \frac{1}{10}$	$\epsilon = \frac{1}{20}$	$\epsilon = \frac{1}{40}$	$\epsilon = \frac{1}{80}$
C_1	MSE	BLasso	16.78	16.82	16.97	17.35	17.49
		FSF	16.88	17.12	17.09	17.24	17.42
	$\hat{q}$	BLasso	13.82	17.84	19.60	20.50	20.28
		FSF	14.86	20.06	23.80	27.08	28.72
C_2	MSE	BLasso	15.39	15.35	15.56	15.81	15.98
		FSF	15.52	15.77	15.84	15.98	16.09
	$\hat{q}$	BLasso	14.56	17.54	18.50	19.34	20.02
		FSF	16.02	19.42	23.92	26.08	28.70
C_3	MSE	BLasso	14.85	14.71	15.01	15.23	15.39
		FSF	15.13	14.83	15.05	15.06	15.17
	$\hat{q}$	BLasso	13.62	15.72	17.88	18.16	18.10
		FSF	14.54	18.00	20.78	23.16	24.68
C_4	MSE	BLasso	16.86	16.45	16.60	16.81	17.02
		FSF	16.76	16.62	16.83	17.05	17.22
	$\hat{q}$	BLasso	15.04	17.58	19.46	19.98	20.40
		FSF	15.88	20.28	24.28	27.16	28.23
C_5	MSE	BLasso	15.50	15.12	15.14	15.31	15.47
		FSF	15.24	15.23	15.39	15.56	15.72
	$\hat{q}$	BLasso	14.00	16.68	19.20	19.18	19.64
		FSF	14.62	19.20	22.76	25.76	27.40

Table 1. Comparison of FSF and BLasso in a simulated nonparametric regression setting. The log of MSE and $\hat{q}$ =# of nonzeros are reported for the oracle solutions on the regularization paths. All results are averaged over 50 runs.

As can be seen from Table 1, since our true model is sparse, in almost all cases the BLasso solutions are sparser and have equivalent or better prediction performances comparing to the FSF solutions with the same stepsize. It is also interesting to note that, smaller stepsizes require more computation but often give worse predictions and much less sparsity. We believe that this is also a regularization effect caused by the discretization of the solution paths (more discussion in Section 8).

7.3 Generalized Boosted Lasso for Other Penalties and Nondifferentiable Loss Functions

First, to demonstrate Generalized BLasso for different penalties, we use the Bridge Regression setting with the diabetes dataset (without the added covariate in the first experiment). The Bridge Regression (first proposed by Frank and Friedman (1993) and later more carefully discussed and implemented by Fu (1998)) 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.

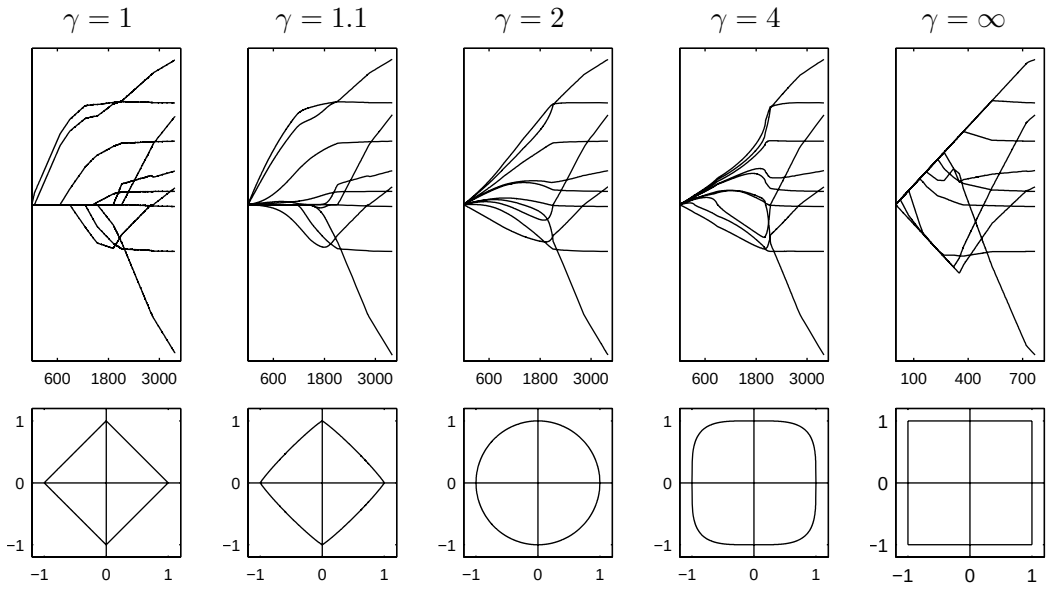

Figure 2. *Upper Panel*: Solution paths produced by BLasso for different bridge parameters. , from left to right: Lasso ($\gamma = 1$), near-Lasso ($\gamma = 1.1$), Ridge ($\gamma = 2$), over-Ridge ($\gamma = 4$), max ($\gamma = \infty$). The Y-axis is the parameter estimate and has the range $[-800, 800]$. 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.

Now, to demonstrate the Generalized BLasso algorithm for classification using an non-differentiable loss function with a L_1 penalty function, we now look at binary classification with the hinge loss. 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 $\text{sign}(\hat{f}(x))$.

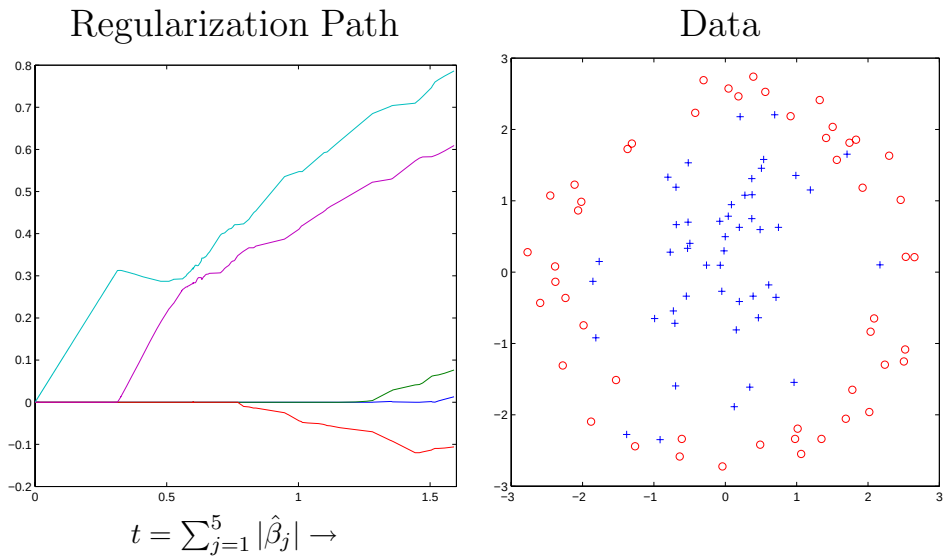

Figure 3. Estimates of 1-norm SVM coefficients $\hat{\beta}_j$, j=1,2,...,5, for the simulated two-class classification data. *Left Panel* BLasso solutions as a function of $t = \sum_{j=1}^{5} |\hat{\beta}_j|$. *Right Panel* Scatter plot of the data points with labels: '+' for $y = -1$; 'o' for $y =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

Since the loss function is not differentiable, we do not have a theoretical guarantee that BLasso works. Nonetheless the solution path produced by Generalized BLasso has the same sparsity and piecewise linearity 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.

8. Discussion and Concluding Remarks

As seen from our simulations under sparse true models, BLasso generates sparser solutions with equivalent or better predictions comparing to FSF which is partially explained by its convergence to the Lasso path as the step size goes to 0. The generalized version is also effective as an off-the-shelf algorithm for the general convex penalized loss minimization problems. One practical issue left undiscussed is the choice of the step size ϵ. Computationally, BLasso takes roughly $O(1/\epsilon)$ 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 but it does not guarantee better predictions as we observed in the simulations. In addition, we observe that BLasso's actual coefficient estimates are pretty close to the Lasso solutions even for relatively large step sizes. Without any solid proof yet, this suggests that using a moderate step size provide more regularization at the cost of a small bias.

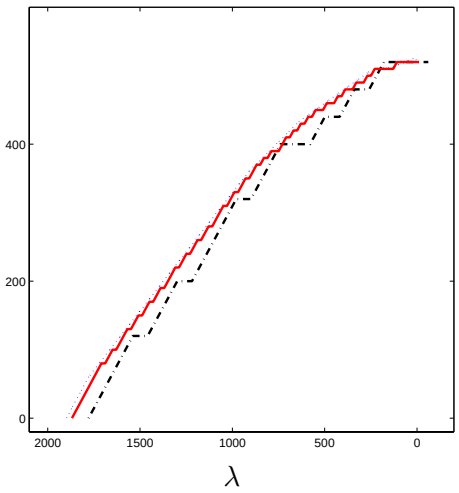

Figure 4. Estimates of regression coefficients $\hat{\beta}_3$ for the diabetes data. Solutions are plotted as functions of λ. *Dotted Line:* Estimates using step size $\epsilon = 0.05$. *Solid Line:* Estimates using step size $\epsilon = 10$. *Dash-dot Line:* Estimates using step size $\epsilon = 50$.

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 and does not require advanced mathematical knowledge of the loss function or penalty. Therefore, it can be used a simple plug-in method for dealing with other convex penalties.

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.

As we discussed for the least squares problem, Boosted Lasso may be also computationally attractive for dealing with nonparametric learning problems with large or infinite number of base learners. This is mainly due to two facts. First, the forward step, as in Boosting, is a sub-optimization problem by itself and Boosting's functional gradient descend strategy applies. 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. Breimann 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 second fact is that, when using an iterative procedure like Boosted Lasso, we usually stop early to avoid overfitting and to get a sparse model. And even if the algorithm is kept running, it usually reaches a close-to-perfect fit without too many iterations. 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 infinite number of base learners. For the finite case, since BLasso avoids oscillation by requiring 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, a different type of oscillation can occur when BLasso keeps going back and force in different directions but only improving the penalized loss function by a diminishing amount, therefore a positive tolerance ξ is mandatory. As suggested by the proof of Theorem 1, we suggest choosing $\xi = o(\epsilon)$ to warrant good approximation to the Lasso path.

Another 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 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:

1. 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 approximate the Lasso solutions for general loss functions and large classes of base learners. This can be proven rigorously for finite number of base learners under the assumption that the loss function is strongly convex with a bounded Hessian.

2. The backward steps introduced in this paper are critical for producing the Lasso path. Without them, the FSF algorithm 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. As a result, FSF lose some of the sparsity provided by Lasso and might also suffer in prediction performance as suggested by our simulations.

3. We generalized BLasso as a simple plug-in method for approximating the regularization path for other convex penalties. This also shows a connection with the Barrier Method for constrained convex optimization.

4. Discussions based on intuition and simulation results are made on the regularization effect of using stepsizes that are not very small.

Appendix A. Appendix: Proofs

First, we offer a proof for Lemma 1.

Proof (Lemma 1)

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

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

therefore

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

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

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

therefore

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

which contradicts the algorithm.

3. 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

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

we get

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

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

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

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

■

Proof of Theorem 1

Theorem 3.1 claims "the BLasso path converges to the Lasso path uniformly" for $\sum L(Z;\beta)$ that is strongly convex with bounded second derivatives in β. The strong convexity and bounded second derivatives imply the hessian w.r.t. β satisfies

$$mI \preceq \nabla^2 \sum L \preceq MI \tag{23}$$

for positive constants $M \geq m > 0$. Using these notations, we will show that for any t s.t. $\lambda^{t+1} > \lambda^t$, we have

$$\|\hat{\beta}^t - \beta^*(\lambda^t)\|_2 \leq (\frac{M}{m}\epsilon + \frac{\xi}{\epsilon}\frac{2}{m})\sqrt{p}. \tag{24}$$

where $\beta^*(\lambda^t)$ is the Lasso estimates with regularization parameter λ^t.

The proof of (24) relies on the following inequalities for strongly convex functions, some of which can be found in Boyd and Vandenberghe (2004). First, because of the strong convexity, we have

$$\sum L(Z;\beta^*(\lambda^t)) \geq \sum L(Z;\hat{\beta}^t) + \nabla \sum L(Z;\hat{\beta}^t)^T(\beta^*(\lambda^t) - \hat{\beta}^t) + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2 \tag{25}$$

The L_1 penalty function is also convex although not strictly convex nor is it differentiable at 0, but we have

$$\|\beta^*(\lambda^t)\|_1 \geq \|\hat{\beta}^t\| + \delta^T(\beta^*(\lambda^t) - \hat{\beta}^t) \tag{26}$$

hold for any p-dimensional vector δ with δ_i the i'th entry of $\text{sign}(\hat{\beta}^t)^T$ for the nonzero entries and $|\delta_i| \leq 1$ otherwise.

Putting both inequalities together, we have

$$\Gamma(\beta^*(\lambda^t);\lambda^t) \geq \Gamma(\hat{\beta}^t;\lambda^t) + (\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t \delta)^T (\beta^*(\lambda^t) - \hat{\beta}^t) \tag{27}$$

Using equation (27), we can bound the L^2 distance between $\beta^*(\lambda_t)$ and $\hat{\beta}^t$ by applying Cauchy-Schwartz to get

$$\Gamma(\beta^*(\lambda^t);\lambda^t) \geq \Gamma(\hat{\beta}^t;\lambda^t) - \|\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t \delta\|_2 \|\beta^*(\lambda^t) - \hat{\beta}^t\|_2 + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2 \tag{28}$$

then since $\Gamma(\beta^*(\lambda^t);\lambda^t) \leq \Gamma(\hat{\beta}^t;\lambda^t)$, we have

$$\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2 \leq \frac{2}{m}\|\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t \delta\|_2. \tag{29}$$

By (c) of Lemma 3.1, for $\hat{\beta}_j^t \neq 0$ we have

$$\sum L(Z;\hat{\beta}^t + \epsilon \text{sign}(\hat{\beta}_j^t) 1_j) \geq \sum L(Z;\hat{\beta}^t) - \lambda_t \epsilon - \xi \tag{30}$$

as the same time, by the bounded hessian assumption we have

$$\sum L(Z;\hat{\beta}^t + \epsilon \text{sign}(\hat{\beta}_j^t) 1_j) \leq \sum L(Z;\hat{\beta}^t) + \epsilon \nabla \sum L(Z;\hat{\beta}^t)^T \text{sign}(\hat{\beta}_j^t) 1_j + \frac{M}{2}\epsilon^2 \tag{31}$$

therefore

$$|\nabla \sum L(Z;\hat{\beta}^t)^T 1_j - \lambda_t \text{sign}(\hat{\beta}_j^t)| \leq \frac{M}{2}\epsilon + \frac{\xi}{\epsilon} \tag{32}$$

Similarly, for $\hat{\beta}_j^t = 0$ following the same steps it is easy to derive

$$|\nabla \sum L(Z;\hat{\beta}^t)^T 1_j| \leq \lambda_t + \frac{M}{2}\epsilon + \frac{\xi}{\epsilon} \tag{33}$$

Therefore for j such that $\hat{\beta}_j^t = 0$ choose δ_j appropriately, then combine with (32), we know that the right hand side of (29) is controlled by $\sqrt{p} \times \frac{2}{m} \times (\frac{M}{2}\epsilon + \frac{\xi}{\epsilon})$. This way we obtain (24). This completes the proof.

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

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

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26:801–824, 1998.

P. Buhlmann and B. Yu. Boosting with the l2 loss: Regression and classification. *Journal of American Statistical Association*, 98, 2003.

S. Chen and D. Donoho. Basis pursuit. Technical report, Department of Statistics, Stanford University, 1994.

D. Donoho. For most large undetermined system of linear equatnions the minimal l_1-norm near-solution approximates the sparsest solution. *Technical Reports, Statistics Department, Stanford University*, 2004.

D. Donoho, M. Elad, and V. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *Preprint*, 2004.

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

J. Fan and R.Z. Li. Variable selection via nonconcave penalized likelihood and its oracle propertie. *Journal of American Statistical Association*, (32):407–499, 2001.

I. Frank and J. Friedman. A statistical view od some chemometrics regression tools. *Technometrics*, 35:109–148, 1993.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285, 1995.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. pages 148–156. Morgan Kauffman, San Francisco, 1996.

J.H. Friedman. Greedy function approximation: a gradient boosting machine. *Annal of Statistics*, 29:1189–1232, 2001.

J.H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annal of Statistics*, 28:337–407, 2000.

T. Hastie, Tibshirani, R., and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag, 2001.

T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. Technical report, Department of Statistics, Stanford University, 2006.

K. Knight and W. J. Fu. Asymptotics for lasso-type estimators. *Annals of Statistics*, 28: 1356–1378, 2000.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. *Advance in Large Margin Classifiers*, 1999.

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

M.R. Osborne, B. Presnell, and B.A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000b.

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

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

R.E. Schapire. The strength of weak learnability. *machine Learning*, 5(2):1997–2027, 1990.

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

J.A. Tropp. Just relax: Convex programming methods for subset selection and sparse approximation. *ICES Report 04-04, UT-Austin, February.*, 2004.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. *Advances in Neural Information Processing Systems*, 16, 2003.