Stagewise 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: Saharon Rosset

Abstract

Many statistical machine learning algorithms (in regression or classification) minimize either an empirical loss function as in AdaBoost, or a penalized empirical loss as in SVM. A single regularization tuning parameter controls the trade-off between fidelity to the data and generalibility, or equivalently between bias and variance. When this tuning parameter changes, a regularization "path" of solutions to the minimization problem is generated, and the whole path is needed to select a tuning parameter to optimize the prediction or interpretation performance. Algorithms such as Lasso and Forward Stagewise Fitting (FSF) (aka e-Boosting) are of great interest because of their resulted sparse models for interpretation in addition to prediction.

In this paper, we propose the BLasso algorithm that ties the FSF (e-Boosting) algorithm with the Lasso method that minimizes the L_1 penalized L_2 loss. BLasso is derived as a coordinate descent method with a fixed stepsize applied to the general Lasso loss function (L_1 penalized L_2 loss). It consists of both a forward step and a backward step. The forward step is similar to e-Boosting or FSF, but the backward step is new and revises the FSF (or e-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, the BLasso path is shown to converge to the Lasso path when the stepsize goes to zero. For cases with a larger number of base learners than the sample size and when the true mdoel is sparse, our simulations indicate that the BLasso model estimates are sparser than those from FSF with comparable or slightly better prediction performance, and that the the discrete stepsize of BLasso and FSF has an additional regularization effect in terms of prediction and sparsity. Moreover, we introduce the Generalized BLasso algorithm to minimizing a general convex loss penalized by a general convex function. Since the (Generalized) BLasso relies only on differences not derivatives, we conclude that it provides a class of simple and easy-to-implement algorithms for tracing the regularization or solution paths of penalized minimization problems.

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

1. Introduction

Many statistical machine learning algorithms minimize either an empirical loss function or a penalized empirical loss, in a regression or a classification setting where covariate or

predictor variables are used to predict a response variable and i.i.d. samples of training data are available. For example, in classification, both AdaBoost (Schapire, 1990; Freund, 1995; Freund and Schapire, 1996) and SVM (Vapnik, 1995; Cristianini and Shawe-Taylor, 2002; Schölkopf and Smola, 2002) build linear classification functions of basis functions of the covariates (predictors). Adaboost minimizes an exponential loss function of the margin, while SVM a penalized hinge loss function of the margin. A single regularization tuning parameter, the number of iterations in AdaBoost and the smoothing parameter in SVM, controls the bias-and-variance trade-off or whether the algorithm overfits or underfits the data. When this tuning parameter changes, a regularization "path" of solutions to the minimization problem is generated. These algorithms have been shown to achieve start-of-the-art prediction performance. A tuning parameter, or equivalently a point on the path of solutions, is chosen to minimize the estimated prediction error over a proper test set or through cross-validation. Hence it is necessary to have algorithms that can generate the path of solutions in an efficient manner. Path following algorithms have also been devised in other statistical penalized minimization problems such as the problems of information bottleneck Tishby et al. (1999) and information distortion Gedeon et al. (2002) where the loss and penalty functions are different from these discussed in this paper. In fact, following the solution paths of numerical problems is the focus of homotopy, a sub-area in numerical analysis. Interested readers are referred to the book Allgower and Georg (1980) (http://www.math.colostate.edu/emeriti/georg/AllgGeorgHNA.pdf) and references at http://www.math.colostate.edu/emeriti/georg/georg.publications.html.

Among all the machine learning methods, those that produce sparse models are of great interest because sparse models lend themselves more easily to interpretation and therefore preferred in sciences and social sciences. Lasso (Tibshirani, 1996) is such a method. It minimizes the L_2 loss function with an L_1 penalty on the parameters in a linear regression model. In signal processing it is called Basis Pursuit (Chen and Donoho, 1994). The L_1 penalty leads to sparse solutions, that is, there are few predictors or basis functions with nonzero weights (among all possible choices). This statement is proved asymptotically under various conditions by different authors (see e.g. Knight and Fu, 2000; Osborne et al., 2000a,b; Donoho et al., 2006; Donoho, 2006; Tropp, 2004; Rosset et al., 2004; Zhao and Yu, 2006; Zou, 2006; Meinshausen and Yu, 2006; Zhang and Huang, 2006).

Sparsity has also been observed in the models generated by Forward Stagewise Fitting or e-Boosting that is a gradient descent procedure with much more cautious steps than L_2Boosting (Rosset et al., 2004; Hastie et al., 2001; Efron et al., 2004). Moreover, these papers also study the similarities between FSF (e-Boosting) with Lasso. This link between Lasso and e-Boosting or FSF is more formally described for the linear regression case through the LARS algorithm (Least Angle Regression Efron et al., 2004). It is also shown in Efron et al. (2004) that for special cases (such as orthogonal designs) e-Boosting or FSF can approximate Lasso path infinitely close, but in general, it is unclear what regularization criterion e-Boosting or FSF optimizes. As can be seen in our experiments (Figure 1 in Section 6.1), e-Boosting or 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, FSF 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 **BLasso** that connects Lasso with FSF or e-Boosting (and the B in the name stands for this connection to boosting). BLasso generates approximately the Lasso path in general situations for both regression and classification for L_1 penalized convex loss function. The motivation for BLasso is a critical observation that FSF or e-Boosting only works in a forward fashion. It takes steps that reduce empirical loss the most regardless of the impact on model complexity (or the L_1 penalty). Hence it is not able to adjust earlier steps. Taking a coordinate (difference) descent view point of the Lasso minimization with a fixed stepsize, 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 or L_1 penalty to decrease. By combining backward and forward steps, BLasso 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 e-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, e-Boosting or FSF can be seen as local regularization in the sense that at any iteration, FSF with a fixed small stepsize only searches over those models which are one small step away from the current one in all possible directions corresponding to the base learners or predictors (cf. Hastie et al. 2006 for a recent interpretation of the $\epsilon \to 0$ case).

In particular, we make three contributions in this paper via BLasso. First, by introducing the backward step we modify e-Boosting or FSF to follow the Lasso path and consequently generate models that are sparser with equivalent or slightly better prediction performance in our simulations (with true sparse models and more predictors than the sample size). Secondly, by showing convergence of BLasso to the Lasso path, we further tighten the conceptual ties between e-Boosting or FSF 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 and easy-to-implement off-the-shelf method for approximating the regularization path for a general loss function and a general convex penalty.

We would like to note that, for the original Lasso problem, i.e. the least squares problem (L_2 loss) 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 where the number of predictors is not large, these methods are very efficient as their computational complexity is on the same order as a single Ordinary Least Squares regression. For other problems such as classification and for nonparametric setups like model fitting with trees, FSF or e-Boosting 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 or e-Boosting but, unlike FSF, BLasso can be shown to converge to the Lasso path quite generally when the stepsize goes to zero.

The rest of the paper is organized as follows. In Section 2.1, the gradient view of Boosting is provided and FSF (e-Boosting) is reviewed as a coordinatewise descent method with a fixed stepsize on the L_2 loss. In Section 2.2, the Lasso empirical minimization problem is reviewed. Section 3 introduces BLasso that is a coordinate descent algorithm with a fixed stepsize applied to the Lasso minimization problem. 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 introduces a Generalized BLasso algorithm which deals with general convex penalties. In Section 6, results of experiments with both simulated and real data are reported to demonstrate the attractiveness of BLasso. BLasso is shown 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. Moreover, we compare different choices of the stepsize and give evidence for the regularization effect of using moderate stepsizes. Finally, Section 7 is a discussion and a summary. In particular, it comments on the computational complexity of BLasso, compares with the algorithm in Rosset (2004), explores the possibility of BLasso for nonparametric learning problems, summaries the paper, and points to future directions.

2. Boosting, Forward Stagewise Fitting and the Lasso

Originally boosting was proposed as an iterative fitting procedure that builds up a model sequentially using a weak or base learner and then carries out a weighted averaging (Schapire, 1990; Freund, 1995; Freund and Schapire, 1996). More recently, boosting has been interpreted as a gradient descent algorithm on an empirical loss function. FSF or e-Boosting can be viewed as a gradient descent with a fixed small stepsize 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 Boosting and of FSF (e-Boosting), followed by a review of the Lasso minimization problem.

2.1 Boosting and Forward Stagewise Fitting

Given data $Z_i = (Y_i, X_i)(i = 1, ..., n)$, where the univariate Y can be continuous (regression problem) or discrete (classification problem), our task is to estimate the function $F : R^d \to R$ that minimizes an expected loss

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

The most prominent examples of the loss function $C(\cdot, \cdot)$ include exponential loss (AdaBoost), logit loss and L_2 loss.

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. Because the family of base learners is usually large, Boosting can be viewed as finding approximate solutions by applying functional gradient descent. This gradient descent view has been recognized and studied by various authors including Breiman (1998), Mason et al. (1999), Friedman et al. (2000), Friedman (2001) and Buhlmann and Yu (2003). Precisely, 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 vector with all 0's except for a 1 in the jth coordinate, and $g \in R$ is stepsize. In other words, Boosting favors the direction $\hat{j}$ that reduces most the empirical loss and $\hat{g}$ is found through a line search. 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) (Efron et al. (2004)) is a similar method for approximating the minimization problem described by (5) with some additional regularization. FSF has also been called e-Boosting for ϵ-Boosting as in Rosset et al. (2004). Instead of optimizing the stepsize as in (6), FSF updates $\hat{\beta}^t$ by a fixed stepsize ϵ as in Friedman (2001).

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 basis function or predictor or coordinate is selected that reduces most the empirical loss. Its corresponding coefficient $\beta_{\hat{j}}$ is then incremented or decremented by a fixed amount ϵ, while all other coefficients β_j, $j \neq \hat{j}$ are left unchanged.

By taking small steps, FSF imposes some local regularization or shrinkage. A related approach can be found in Zhang (2003) where a relaxed gradient descent method is used.

After $T < \infty$ iterations, many of the estimated coefficients by FSF 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 in recent years (e.g. Osborne et al., 2000a,b; Efron et al., 2004; Donoho et al., 2006; Donoho, 2006; Tropp, 2004; Rosset et al., 2004; Meinshausen and Bühlmann, 2005; Candes and Tao, 2007; Zhao and Yu, 2006; Zou, 2006; Wainwright, 2006; Meinshausen and Yu, 2006; Zhang and Huang, 2006). Sparsity can also result from other penalties as in, for example, 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 (predictors) but it is local regularization and does not converge to the Lasso path in general. As can be seen in Sec. 6.2, FSF has also less sparse solutions comparing to Lasso in our simulations.

Next we propose the 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 stepsize goes to 0. This relationship between Lasso and BLasso leads to sparser solutions for BLasso comparing to FSF with similar or slightly better prediction performance in our simulation set-up with different choices of stepsizes.

3. The BLasso Algorithm

We first describe the BLasso algorithm (**Algorithm 1**).This algorithm has two related input parameters, a stepsize ϵ and a tolerance level ξ.

Algorithm 1 BLasso

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, ..., n$ and a small stepsize 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 $\lambda^t \leq 0$.

The tolerance level is only needed to avoid numerical instability when assessing changes of the empirical loss function and should be set as small as possible while accommodating

the numerical accuracy of the implementation. (ξ is set to 10^{-6} in the implementation of the algorithm that used in this paper.)

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 $\lambda^0 \geq \lambda$.
2. For any t, we have $\Gamma(\hat{\beta}^{t+1}; \lambda^t) \leq \Gamma(\hat{\beta}^t; \lambda^t) - \xi$.
3. For $\xi \geq 0$ 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 1 (1) 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 (2) says that for each value of λ, BLasso performs coordinate descent until there is no descent step. Then, by Lemma 1 (3), the value of λ is reduced and a forward step is forced. The stepsize 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 needed to accommodate for numerical error and should be set to be much smaller than ϵ to have a good approximation (see Proof of Theorem 1). In fact, we have a convergence result for BLasso (detailed proof is included in the Appendix):

Theorem 1. For a 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.

Note that Conjecture 2 of Rosset et al. (2004) follows from Theorem 1. This is because if all the optimal coefficient paths are monotone, then BLasso will never take a backward step, so it will be equivalent to e-Boosting.

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 remains 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 discuss more on this topic in the discussion (Sec. 7).

4. The Backward 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, when FSF picks up more irrelevant variables as compared to the Lasso path in some cases (cf. Figure 1 in Section 6.2), it does not have a mechanism to remove them. As seen below, this backward step naturally arises in BLasso because of our coordinate descent view of the minimization of the Lasso loss. (Since ξ exists for numerical purpose only, it is assumed to be 0 thus excluded in the following theoretical discussion.)

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:

$$\begin{aligned} \Delta_j T(\beta) &= \|\beta + s1_j\|_1 - \|\beta\|_1 = |\beta_j + s| - |\beta_j| \\ &= \epsilon \cdot \text{sign}^+(\beta_j, s) \\ &= \epsilon \cdot \begin{cases} 1 & \text{if} \quad s\beta_j > 0 \text{ or } \beta_j = 0 \\ \text{-1} & \text{if} \quad s\beta_j < 0 \\ 0 & \text{if} \quad s = 0 \end{cases} \end{aligned} \tag{14}$$

Equation (14) shows that an ϵ step changes the penalty by a fixed ϵ in absolute value for any j. That is, only the sign of the penalty change may vary. In the beginning of BLasso, all j directions are leaving zero and hence changing the L_1 penalty by the same positive amount $\lambda \cdot \epsilon$. Therefore the first step of BLasso is a forward step because minimizing Lasso loss is now equivalent to minimizing the L_2 loss due to the same positive change of the L_1 penalty. As the algorithm proceeds, some of the penalty changes might become negative and minimizing the empirical loss is no longer equivalent to minimizing the Lasso loss. In fact, except for special cases like orthogonal covariates (predictors), the FSF steps now might result in negative changes of the L_1 penalty. In some of these situations, a step that goes "backward" reduces the penalty with a small sacrifice in the empirical loss. In general, to minimize the Lasso loss, one needs to go "back and forth" to trade off the penalty with the empirical loss for different regularization parameters.

To be precise, 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 selected 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 might have picked up irrelevant variables.

Since much of the discussion on the similarity and difference between FSF and Lasso is focused on Least Squares problems (e.g. Efron et al. 2004; Hastie et al. 2001), we now examine the BLasso algorithm in this case. It is straightforward to see that in LS problems both forward and backward steps in BLasso are based only on the correlations between fitted residuals and the covariates (predictors). It follows that BLasso in this 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 a large number of observations since the inner-product between η_t and X_j can be updated by

$$(\eta^{t+1})' X_j = (\eta^t - s X_{\hat{j}^t})' X_j = (\eta^t)' X_j - s X'_{\hat{j}^t} X_j. \tag{15}$$

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 (15) 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 LS problems.

In nonparametric situations, the number of base learners is large therefore the aforementioned strategy becomes inefficient. BLasso has a natural extention to this case as follows: similar to boosting, the forward step is 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 (15). 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 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 (L_2) 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.

5. Generalized BLasso

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 stepsize 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 BLasso algorithm (**Algorithm 2**).

In the Generalized BLasso 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.

6. Experiments

In this section, three experiments are carried out to illustrate the attractiveness 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 (predictors). 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 emulate the nonparametric learning scenario and when the true model is sparse. The result shows, overall, comparing with FSF, BLasso gives sparser solutions with similar or slightly better predictions, for various stepsizes. Moreover, we find that when the stepsize increases, there is a regularization effect in terms of both prediction and sparsity, for both BLasso and FSF.

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.

Algorithm 2 Generalized BLasso

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, ..., n$ and a fixed small stepsize $\epsilon > 0$ 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 by at least a ξ amount, 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 $\lambda^t \leq 0$.

6.1 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 $\text{Var}(e) = 1/442$), 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 (predictors) 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:

$$\Gamma(\beta; \lambda) = \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda \|\beta\|_1 \tag{16}$$

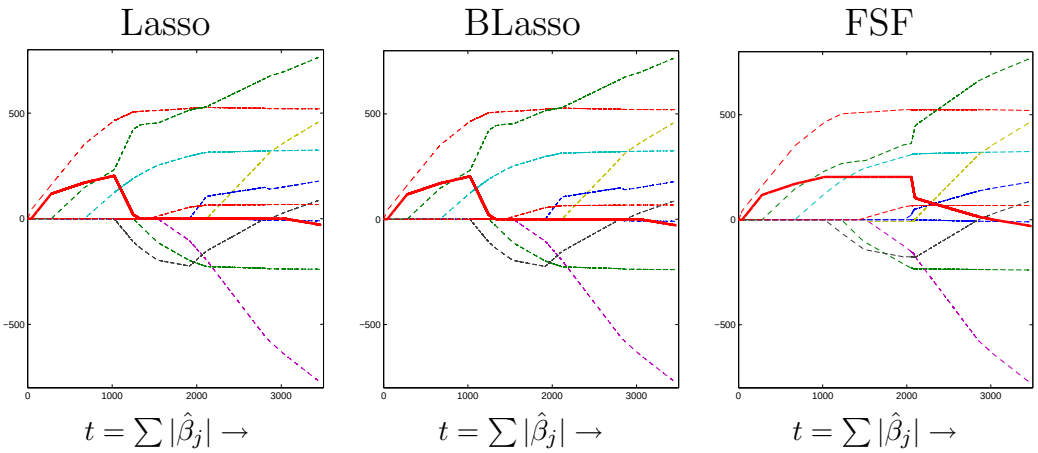

Figure 1. Regularization path plots, for the diabetes data set, of Lasso, BLasso and FSF: the curves or paths to estimates of coefficient $\hat{\beta}_j$ for 10 original and 1 added covariates (predictors), as the regularization is relaxed or t tends to infinity. The red sold curves correspond to the 11th added covariate. *Left Panel:* Lasso solution paths (produced using simplex search method on the penalized empirical loss function for each λ) as a function of $t = \|\beta\|_1$. *Middle Panel:* BLasso solution paths, which can be seen indistinguishable to the Lasso solutions. *Right Panel:* FSF solution paths, 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.

6.2 Comparison of BLasso and 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 among the covariates (predictors). 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 $.95 \times D_i + .05 I_{p \times p}$ where D_i is sampled from $Wishart(20, p)$ then normalized to have 1's on diagonal. The Wishart distribution creates a fair amount of correlation in C_i (average absolute value is about 0.18) between the covariates and the added identity matrix guarantees 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 $\frac{1}{80}$. We also run Lasso which is listed as BLasso when $\epsilon = 0$.

To compare the performances, we examine the solutions on the regularization 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			$\epsilon = \frac{1}{5}$	$\epsilon = \frac{1}{10}$	$\epsilon = \frac{1}{20}$	$\epsilon = \frac{1}{40}$	$\epsilon = \frac{1}{80}$	Lasoo ($\epsilon = 0$)
C_1	MSE	BLasso	18.60	18.27	18.33	18.60	19.42	19.98
		FSF	19.77	19.40	19.60	19.82	19.96	
	$\hat{q}$	BLasso	15.38	20.08	21.76	21.44	20.50	21.86
		FSF	18.32	24.00	27.28	30.48	32.14	
C_2	MSE	BLasso	19.58	19.28	19.65	19.94	20.76	21.12
		FSF	20.67	20.29	20.63	20.94	21.11	
	$\hat{q}$	BLasso	14.80	18.92	20.18	21.22	20.52	21.82
		FSF	18.34	21.90	25.70	28.80	29.38	
C_3	MSE	BLasso	18.83	18.14	18.55	18.90	19.32	20.15
		FSF	19.35	19.11	19.52	19.78	19.93	
	$\hat{q}$	BLasso	15.22	19.10	19.92	20.02	19.52	21.08
		FSF	15.38	19.72	23.30	25.88	27.30	
C_4	MSE	BLasso	20.09	19.88	19.85	20.20	21.84	21.70
		FSF	21.53	21.09	21.13	21.35	21.57	
	$\hat{q}$	BLasso	15.76	20.82	22.20	22.42	21.12	22.24
		FSF	18.90	24.64	30.38	32.02	34.16	
C_5	MSE	BLasso	18.79	18.62	18.70	19.09	19.47	20.12
		FSF	19.99	19.92	19.84	20.19	20.36	
	$\hat{q}$	BLasso	15.58	19.16	21.26	21.92	22.18	22.76
		FSF	17.10	23.24	28.24	30.94	32.84	

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.

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−Lasoo	-1.38 (0.37)	-1.71 (0.23)	-1.65 (0.21)	-1.38 (0.21)	-0.56 (0.35)
		BLasso−FSF	-1.17 (0.27)	-1.13 (0.28)	-1.27 (0.26)	-1.22 (0.26)	-0.54 (0.24)
	$\hat{q}$	BLasso−Lasoo	-6.48 (0.64)	-1.78 (0.70)	-0.10 (0.67)	-0.42 (0.63)	-1.36 (0.65)
		BLasso−FSF	-2.94 (0.89)	-3.92 (1.22)	-5.52 (1.26)	-9.04 (1.43)	-11.64 (1.64)
C_2	MSE	BLasso−Lasoo	-1.54 (0.37)	-1.84 (0.29)	-1.47 (0.26)	-1.18 (0.25)	-0.36 (0.45)
		BLasso−FSF	-1.09 (0.32)	-1.01 (0.27)	-0.98 (0.23)	-1.00 (0.23)	-0.35 (0.38)
	$\hat{q}$	BLasso−Lasoo	-7.02 (0.58)	-2.90 (0.65)	-1.64 (0.52)	-0.60 (0.50)	-1.30 (0.48)
		BLasso−FSF	-3.54 (0.99)	-2.98 (0.88)	-5.52 (1.09)	-7.58 (1.31)	-8.86 (1.41)
C_3	MSE	BLasso−Lasoo	-1.32 (0.35)	-2.01 (0.36)	-1.60 (0.33)	-1.25 (0.32)	-0.83 (0.32)
		BLasso−FSF	-0.53 (0.28)	-0.97 (0.22)	-0.97 (0.23)	-0.88 (0.23)	-0.62 (0.24)
	$\hat{q}$	BLasso−Lasoo	-5.86 (0.81)	-1.98 (0.72)	-1.16 (0.54)	-1.06 (0.55)	-1.56 (0.56)
		BLasso−FSF	-0.16 (0.78)	-0.62 (0.87)	-3.38 (1.05)	-5.86 (0.97)	-7.78 (1.08)
C_4	MSE	BLasso−Lasoo	-1.61 (0.45)	-1.82 (0.33)	-1.85 (0.33)	-1.50 (0.33)	0.14 (0.66)
		BLasso−FSF	-1.44 (0.30)	-1.20 (0.28)	-1.28 (0.24)	-1.15 (0.29)	0.27 (0.67)
	$\hat{q}$	BLasso−Lasoo	-6.48 (0.71)	-1.42 (0.85)	-0.04 (0.73)	0.18 (0.52)	-1.12 (0.67)
		BLasso−FSF	-3.14 (0.92)	-3.82 (1.16)	-8.18 (1.12)	-9.60 (1.35)	-13.04 (1.68)
C_5	MSE	BLasso−Lasoo	-1.33 (0.38)	-1.50 (0.26)	-1.41 (0.26)	-1.03 (0.22)	-0.65 (0.22)
		BLasso−FSF	-1.20 (0.25)	-1.30 (0.23)	-1.14 (0.28)	-1.10 (0.29)	-0.89 (0.28)
	$\hat{q}$	BLasso−Lasoo	-7.18 (0.84)	-3.60 (0.64)	-1.50 (0.58)	-0.84 (0.52)	-0.58 (0.55)
		BLasso−FSF	-1.52 (0.88)	-4.08 (1.10)	-6.98 (1.08)	-9.02 (1.21)	-10.66 (1.50)

Table 2. Means and Standard Errors of the differences of MSE and $\hat{q}$ between BLasso and Lasso, and between Blasso and FSF in Table 1.

As can be seen from Table 1, since our true model is sparse, in almost all cases the BLasso solutions are sparser and have similar 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 conjecture that this is also a regularization effect caused by the discretization of the solution paths (more discussion in Section 8) and this effect has also been observed by Gao et al. (2006) in a language ranking problem.

Table 2 gives a further analysis of the results in Table 1. It contains means and standard errors of the differences of MSE and $\hat{q}$, between BLasso and Lasso and between BLasso and FSF for the stepsizes given in Table 1. First of all, all the mean differences are negative and when compared with their SE's, the differences are also significant except for few cells for small stepsizes 1/40 and 1/80 (in the last two columns). This overwhelming pattern of significant negative difference suggests that BLasso is better than Lasso and FSF from this simulation in terms of both prediction and sparsity unless the stepsize is very small as in the last two columns. Moreover, for MSE the stepsize $\epsilon = 1/10$ seems to bring the best improvement of BLasso over Lasso, while the improvement is pretty robust against the choice of stepsize. On the other hand, the improvements of BLasso over FSF on MSE are less then those of BLasso over Lasso because FSF has the same discrete stepsizes. Hence these improvements reflect the gains only by the backward steps since FSF takes also forward steps. In terms of $\hat{q}$, the number of covariates selected, as expected, the larger the stepsize, the sparser the BLasso model is relative to the Lasso model or the FSF model. The sparsity improvements over Lasso are significant for all cells except for the last column with $\epsilon = 1/80$. When compared with FSF, the sparsity improvements are less and smaller (still significant) . In terms of gains on both MSE and sparsity and relative to both Lasso and FSF, stepsizes 1/10 and 1/20, i.e. 0.1 or 0.05, seem good overall choices for this simulation study.

As suggesetd by one referee, we now compare the Lasso empirical loss functions induced by BLasso, FSF and Lasso (through LARS). Figure 2 shows plots of in-sample Mean Squared Error versus L_1 norms of the coefficients taken from one typical run of the simulation conducted in this section. As shown by the plots, BLasso approximates the Lasso path better than FSF under both big and small step sizes. In particular, when the step size is small, the BLasso path is almost indiscernible from the Lasso path.

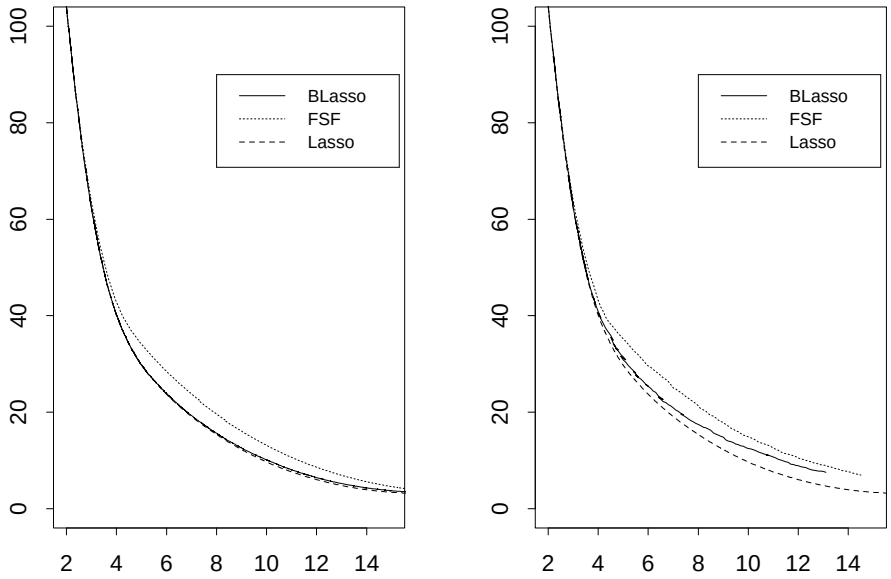

Figure 2. Plots of Mean Squared Error (y-axis) versus $\|\beta\|_1$ (x-axis) for a typical realization of the experiment (on run under C_2 from Table 1). The step size is set to $\epsilon = \frac{1}{80}$ in the left plot and $\epsilon = \frac{1}{5}$ in the right.

6.3 Generalized BLasso 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:

$$\Gamma(\beta; \lambda) = \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda \|\beta\|_\gamma \tag{17}$$

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

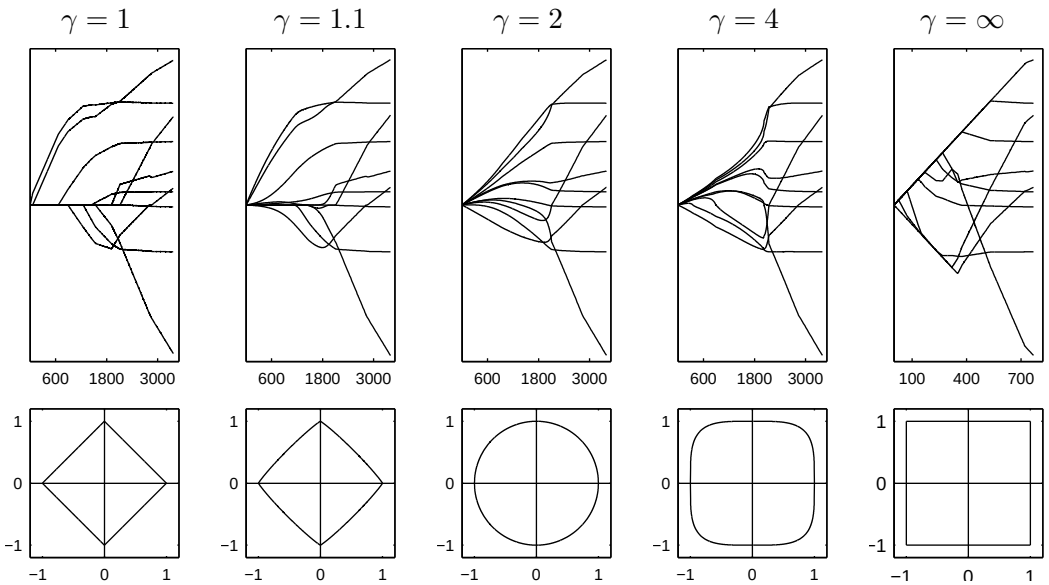

Figure 3. *Upper Panel*: Solution paths produced by BLasso for different bridge parameters, on the diabetes data set. From left to right: Lasso ($\gamma = 1$), near-Lasso ($\gamma = 1.1$), Ridge ($\gamma = 2$), over-Ridge ($\gamma = 4$), max ($\gamma = \infty$). The Y-axis is the parameter estimate and 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 n=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:

$$(\hat{\beta}_0, \beta) = \arg\min_{\beta_0, \beta} \sum_{i=1}^{n} (1 - Y_i(\beta_0 + \sum_{j=1}^{m} \beta_j h_j(X_i)))^+ + \lambda \sum_{j=1}^{5} |\beta_j| \tag{18}$$

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 $\text{sign}(\hat{f}(x))$.

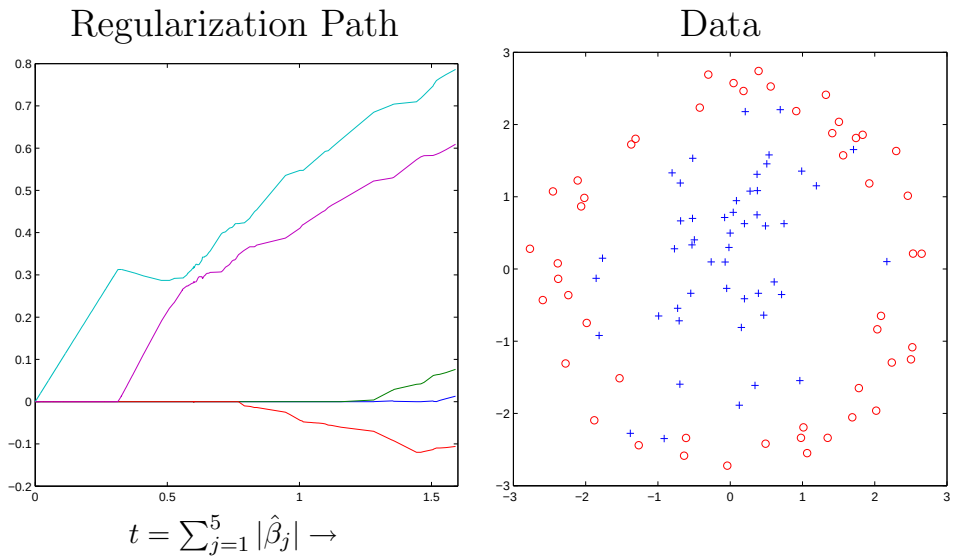

Figure 4. Estimates of 1-norm SVM coefficients $\hat{\beta}_j$, j=1,2,...,5, for the simulated two-class classification data. *Left Panel* BLasso solutions as a function of $t = \sum_{j=1}^{5} |\hat{\beta}_j|$. *Right Panel* Scatter plot of the data points with labels: '+' for $y = -1$; 'o' for $y = 1$.

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 come 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.

7. Discussion and Concluding Remarks

As seen from our simulations under sparse true models, BLasso generates sparser solutions with similar or slightly better predictions relative to Lasso and FSF. The behavior relative to Lasso is due to the discrete stepsize of BLasso, while the behavior relative to FSF is

partially explained by its convergence to the Lasso path as the stepsize goes to 0. We believe that the generalized version is also effective as an off-the-shelf algorithm for the general convex penalized loss minimization problems.

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. As shown in the simulations, choosing a smaller stepsize gives smoother solution path but it does not guarantee better predictions. Actually, for the particular simulation set-up in Sec. 6.2, moderate stepsizes gave better results both in terms of MSE and sparisty. Now we observe that BLasso's actual coefficient estimates are pretty close to the Lasso solutions even for relatively large stepsizes.

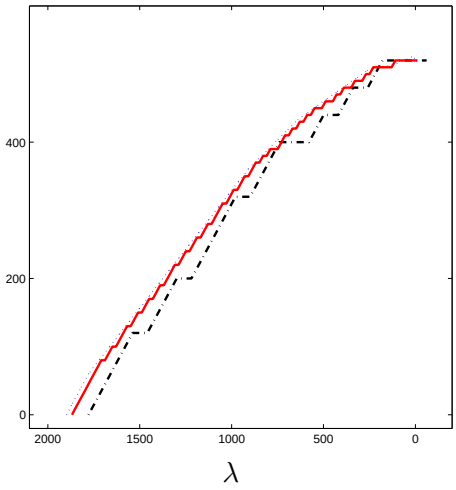

Figure 5. Estimates of regression coefficients $\hat{\beta}_3$ for the diabetes data set. Solutions are plotted as functions of λ. *Dotted Line:* Estimates using stepsize $\epsilon = 0.05$. *Solid Line:* Estimates using stepsize $\epsilon = 10$. *Dash-dot Line:* Estimates using stepsize $\epsilon = 50$.

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

BLasso has only one stepsize parameter (with the exception of the numerical tolerance ξ which is implementation specific but not necessarily a user 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 5, a smaller stepsize 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 which could be heavy-duty computationally when the number p of covariates is not small. 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. It can also be used a simple plug-in method for dealing with other convex penalties. Hence BLasso is easy to program and allows testing of different loss and peanlty functions. Admittedly, this ease of implementation can cost computation time in large p situations.

BLasso's stepsize 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 to each other.

As we discussed for the least squares problem, BLasso 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 BLasso, 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 kind 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 a good approximation to the Lasso path.

One direction for future research 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, the BLasso algorithm is constructed to minimize an L_1 penalized convex loss function. While it maintains the simplicity and flexibility of e-Boosting (or Forward Stagewise Fitting), 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 loses 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, easy-to-implement, plug-in method for approximating the regularization path for other convex penalties.

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

Last but not least, matlab codes for BLasso in the case of L_2 loss and L_1 penalty can be downloaded at http://www.stat.berkeley.edu/twiki/Research/YuGroup/Software. (Thanks to Guilherme V. Rocha)

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 $(\lambda^t - \lambda^{t+1})\|\hat{\beta}^t\|_1$, and recall $T(\hat{\beta}^{t+1}) = \|\hat{\beta}^{t+1}\|_1 > |\hat{\beta}^t\|_1 = T(\hat{\beta}^t)$, we get

$$\begin{aligned}\Gamma(\hat{\beta}^t;\lambda^t) - \xi &< \Gamma(\hat{\beta}^{t+1};\lambda^t)\\ &= \min_{j',|s|=\epsilon}\Gamma(\hat{\beta}^t + s1_{j'};\lambda^t)\\ &\le \Gamma(\hat{\beta}^t \pm \epsilon 1_j;\lambda^t)\end{aligned}$$

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

$$mI \preceq \nabla^2 \sum L \preceq MI \tag{19}$$

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

$$\|\hat{\beta}^t - \beta^*(\lambda^t)\|_2 \le (\frac{M}{m}\epsilon + \frac{\xi}{\epsilon}\frac{2}{m})\sqrt{p}. \tag{20}$$

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

The proof of (20) 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

$$\sum L(Z;\beta^*(\lambda^t)) \geq \sum L(Z;\hat{\beta}^t) + \nabla \sum L(Z;\hat{\beta}^t)^T(\beta^*(\lambda^t) - \hat{\beta}^t) + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2 \quad (21)$$

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

$$\|\beta^*(\lambda^t)\|_1 \geq \|\hat{\beta}^t\|_1 + \delta^T(\beta^*(\lambda^t) - \hat{\beta}^t) \quad (22)$$

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

$$\Gamma(\beta^*(\lambda^t);\lambda^t) \geq \Gamma(\hat{\beta}^t;\lambda^t) + (\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t\delta)^T(\beta^*(\lambda^t) - \hat{\beta}^t) + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2 \quad (23)$$

Using equation (23), we can bound the L^2 distance between $\beta^*(\lambda_t)$ and $\hat{\beta}^t$ by applying Cauchy-Schwartz to get

$$\Gamma(\beta^*(\lambda^t);\lambda^t) \geq \Gamma(\hat{\beta}^t;\lambda^t) - \|\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t\delta\|_2\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2 + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2 \quad (24)$$

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

$$\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2 \leq \frac{2}{m}\|\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t\delta\|_2. \quad (25)$$

By statement (3) of Lemma 1, for $\hat{\beta}_j^t \neq 0$ we have

$$\sum L(Z;\hat{\beta}^t \pm \epsilon\text{sign}(\hat{\beta}_j^t)1_j) \pm \lambda_t\epsilon \geq \sum L(Z;\hat{\beta}^t) - \xi \quad (26)$$

at the same time, by the bounded Hessian assumption we have

$$\sum L(Z;\hat{\beta}^t \pm \epsilon\text{sign}(\hat{\beta}_j^t)1_j) \leq \sum L(Z;\hat{\beta}^t) \pm \epsilon\nabla \sum L(Z;\hat{\beta}^t)^T\text{sign}(\hat{\beta}_j^t)1_j + \frac{M}{2}\epsilon^2. \quad (27)$$

Connect these two inequalities, we have

$$\mp\epsilon \times (\nabla \sum L(Z;\hat{\beta}^t)^T 1_j\text{sign}(\hat{\beta}_j^t) + \lambda_t) \leq \frac{M}{2}\epsilon^2 + \xi \quad (28)$$

therefore

$$|(\nabla \sum L(Z;\hat{\beta}^t)^T 1_j\text{sign}(\hat{\beta}_j^t) + \lambda_t)| \leq \frac{M}{2}\epsilon + \frac{\xi}{\epsilon} \quad (29)$$

Similarly, for $\hat{\beta}_j^t = 0$, instead of (26), we have

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

Combine with (27), we have

$$|\nabla \sum L(Z;\hat{\beta}^t)^T 1_j| - \lambda_t \leq \frac{M}{2}\epsilon + \frac{\xi}{\epsilon} \quad (31)$$

Therefore for j such that $\hat{\beta}_j^t = 0$ choose δ_j appropriately, then combine with (29), we know that the right hand side of (25) is controlled by $\sqrt{p} \times \frac{2}{m} \times (\frac{M}{2}\epsilon + \frac{\xi}{\epsilon})$. This way we obtain (20). 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. We thank Dr. Chris Holmes and Mr. Guilherme V. Rocha for their very helpful comments and discussions on the paper. Finally, we would like to thank two referees and the action editor for their thoughtful and detailed comments on an earlier version of the paper.

References

E.L. Allgower and K. Georg. Homotopy methods for approximating several solutions to nonlinear systems of equations. In W. Forster, editor, *Numerical solution of highly nonlinear problems*, pages 253–270. North-Holland, 1980.

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.

E. Candes and T. Tao. The danzig selector: statistical estimation when p is much larger than n. *Annals of Statistics (to appear)*, 2007.

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

N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2002.

D. Donoho. For most large undetermined system of linear equatnions the minimal l_1-norm near-solution approximates the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829), 2006.

D. Donoho, M. Elad, and V. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Trans. Information Theory*, 52(1):6–18, 2006.

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.

J. Gao, H. Suzuki, and B. Yu. Approximate lasso methods for language modeling. pages 225–232, 2006.

T. Gedeon, A. E. Parker, and A. G. Dimitrov. Information distortion and neural coding. *Canadian applied mathematics quarterly*, 2002.

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.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2005.

N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Technical Report 720, Statistics Department, UC Berkeley*, 2006.

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. *Journal of machine Learning*, 5(2):1997–2027, 1990.

B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization and beyond*. MIT Press, 2002.

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

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. 1999.

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

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

M. J. Wainwright. Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1-constrained quadratic programming. *Technical Report 709, Statistics Department, UC Berkeley*, 2006.

C.-H. Zhang and J. Huang. Model-selection consistency of the lasso in high dimensional linear regression. *Technical Report 003, Statistics Department, Rutgers University*, 2006.

T. Zhang. Sequentiall greedy approximation for certain convex optimization problems. *IEEE Trans. on Information Theory*, 49(3):682–691, 2003.

P. Zhao and B. Yu. On model selection consistency of lasso. Technical report, 2006.

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

H. Zou. The adaptive lasso and its oracle properties. *Journal of American Statistical Association*, 101:1418–1429, 2006.