Submitted to the Annals of Statistics

PIECEWISE LINEAR REGULARIZED SOLUTION PATHS

By Saharon Rosset and Ji Zhu

IBM T.J. Watson Research Center and University of Michigan

We consider the generic regularized optimization problem $\hat{\beta}(\lambda) = \arg\min_\beta L(\mathsf{y}, X\beta) + \lambda J(\beta)$. Recently, Efron et al. (2004) have shown that for the Lasso — that is, if L is squared error loss and $J(\beta) = \|\beta\|_1$ is the ℓ_1 norm of β — the optimal coefficient path is piecewise linear, i.e., $\partial\hat{\beta}(\lambda)/\partial\lambda$ is piecewise constant. We derive a general characterization of the properties of (loss L, penalty J) pairs which give piecewise linear coefficient paths. Such pairs allow for efficient generation of the full regularized coefficient paths. We investigate the nature of efficient path following algorithms which arise. We use our results to suggest robust versions of the Lasso for regression and classification, and to develop new, efficient algorithms for existing problems in the literature, including Mammen & van de Geer's locally adaptive regression splines.

1. Introduction. Regularization is an essential component in modern data analysis, in particular when the number of predictors is large, possibly larger than the number of observations, and non-regularized fitting is likely to give badly over-fitted and useless models.

In this paper we consider the generic regularized optimization problem. The inputs we have are:

- A training data sample $X = (\mathbf{x}_1, ..., \mathbf{x}_n)^\mathsf{T}$, $\mathsf{y} = (y_1, ..., y_n)^\mathsf{T}$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ for regression, $y_i \in \{\pm 1\}$ for 2-class classification.
- A convex non-negative loss functional $L : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$
- A convex non-negative penalty functional $J : \mathbb{R}^p \to \mathbb{R}$, with $J(0) = 0$. We will almost exclusively use $J(\beta) = \|\beta\|_q$ in this paper, i.e., penalizing the ℓ_q norm of the coefficient vector.

We want to find:

$$\hat{\beta}(\lambda) = \arg\min_{\beta \in \mathbb{R}^p} L(\mathsf{y}, X\beta) + \lambda J(\beta), \tag{1}$$

where $\lambda \geq 0$ is the regularization parameter: $\lambda = 0$ corresponds to no regularization, while $\lim_{\lambda \to \infty} \hat{\beta}(\lambda) = \mathsf{0}$. In choosing a value for the regularization parameter, a general approach is to solve (1) for a representative set of λ values and choose among these models, using model selection techniques like cross validation or generalized cross validation.

Many of the commonly used methods for data mining, machine learning and statistical modeling can be described as exact or approximate regularized optimization approaches. The obvious examples from the statistics literature are explicit regularized linear regression approaches, such as ridge regression [11] and the Lasso [20]. Both of these use squared error loss, but they differ in the penalty they impose on the coefficient vector β:

$$\text{Ridge:} \qquad \hat{\beta}(\lambda) = \min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\intercal \beta)^2 + \lambda \|\beta\|_2^2, \tag{2}$$

$$\text{Lasso:} \qquad \hat{\beta}(\lambda) = \min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\intercal \beta)^2 + \lambda \|\beta\|_1. \tag{3}$$

Another example from the statistics literature is the penalized logistic regression model [25] for classification, which is widely used in medical decision and credit scoring models:

$$\hat{\beta}(\lambda) = \min_{\beta} \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{x}_i^\intercal \beta}) + \lambda \|\beta\|_2^2,$$

Many "modern" methods for machine learning and signal processing can also be cast in the framework of regularized optimization. For example, the regularized support vector machine [24] uses the hinge loss function and the ℓ_2-norm penalty[1]:

$$\hat{\beta}(\lambda) = \min_{\beta} \sum_{i=1}^{n} (1 - y_i \mathbf{x}_i^\intercal \beta)_+ + \lambda \|\beta\|_2^2, \tag{4}$$

where $(\cdot)_+$ is the positive part of the argument. Boosting [7] is a popular and highly successful method for iteratively building an additive model from a dictionary of "weak learners". In [18] we show that the AdaBoost algorithm *approximately* follows the path of the ℓ_1-regularized solutions to the exponential loss function e^{-yf} as the regularizing parameter λ decreases.

In this paper, we concentrate our attention on (loss L, penalty J) pairings where the optimal path $\hat{\beta}(\lambda)$ is *piecewise linear* as a function of λ, i.e., $\exists \lambda_0 = 0 < \lambda_1 < \ldots < \lambda_m = \infty$ and $\gamma_0, \gamma_1, \ldots, \gamma_{m-1} \in \mathbb{R}^p$ such that $\hat{\beta}(\lambda) = \hat{\beta}(\lambda_k) + (\lambda - \lambda_k)\gamma_k$ for $\lambda_k \leq \lambda \leq \lambda_{k+1}$. Such models are attractive because they allow us to generate the whole regularized path $\hat{\beta}(\lambda),\ 0 \leq \lambda \leq \infty$ simply by sequentially calculating the "step sizes" between each two

[1]This representation differs from the "standard" optimization representation of the regularized SVM, however it is mathematically equivalent to it.

consecutive λ values and the "directions" $\gamma_1, \ldots, \gamma_{m-1}$. Our discussion will concentrate on $(L,\ J)$ pairs which allow efficient generation of the whole path and give statistically useful modeling tools.

A canonical example is the Lasso (3). Recently [4] have shown that the piecewise linear coefficient paths property holds for the Lasso, and suggested the LAR-Lasso algorithm which takes advantage of it. Similar algorithms were suggested for the Lasso in [16, 17] and for total-variation penalized squared error loss in [14]. We have extended some path-following ideas to versions of the regularized support vector machine [27, 10].

In this paper, we systematically investigate the usefulness of piecewise linear solution paths. We aim to combine efficient computational methods based on piecewise linear paths and statistical considerations in suggesting new algorithms for existing regularized problems and in defining new regularized problems. We tackle three main questions:

1. What are the "families" of regularized problems that have the piecewise linear property? The general answer to this question is that the loss L has to be a *piecewise quadratic* function and the penalty J has to be a *piecewise linear* function. We give some details and survey the resulting "piecewise linear toolbox" in Section 2.
2. For what members of these families can we design efficient algorithms, either in the spirit of the LAR-Lasso algorithm, or using different approaches? Our main focus in this paper is on direct extensions of LAR-Lasso to "almost-quadratic" loss functions (Section 3) and to non-parametric regression (Section 4). We briefly discuss a non-LAR type algorithm for ℓ_1 loss in Section 5.
3. Out of the regularized problems we can thus solve efficiently, which ones are of statistical interest? This can be due to two distinct reasons:
   (a) Regularized problems that are widely studied and used are obviously of interest, if we can offer new, efficient algorithms for solving them. In this paper we discuss in this context locally adaptive regression splines [14] (Section 4.1), quantile regression [13], and support vector machines (Section 5).
   (b) Our efficient algorithms allow us to pose — and solve efficiently — statistically motivated regularized problems that have not been considered in the literature. In this context, we propose robust versions of the Lasso for regression and classification (Section 3).

2. The Piecewise Linear Toolbox. For the coefficient paths to be piecewise linear, we require that $\frac{\partial\hat{\beta}(\lambda)}{\partial\lambda}/\|\frac{\partial\hat{\beta}(\lambda)}{\partial\lambda}\|$ is a piecewise constant vector

as a function of λ. Using Taylor expansions of the normal equations for the minimizing problem (1), we can show that if L, J are both twice differentiable in the neighborhood of a solution $\hat{\beta}(\lambda)$, then:

$$\frac{\partial \hat{\beta}(\lambda)}{\partial \lambda} = -\left[\nabla^2 L\left(\hat{\beta}(\lambda)\right) + \lambda \nabla^2 J\left(\hat{\beta}(\lambda)\right)\right]^{-1} \nabla J\left(\hat{\beta}(\lambda)\right), \tag{5}$$

where we are using the notation $L(\hat{\beta}(\lambda))$ in the obvious way, i.e., we make the dependence on the data X, y (here assumed constant) implicit.

From this we get that:

Proposition 1 *A sufficient and necessary condition for the solution path to be linear at λ_0 when L, J are twice differentiable in a neighborhood of $\hat{\beta}(\lambda_0)$ is that:*

$$-\left[\nabla^2 L\left(\hat{\beta}(\lambda)\right) + \lambda \nabla^2 J\left(\hat{\beta}(\lambda)\right)\right]^{-1} \nabla J\left(\hat{\beta}(\lambda)\right) \tag{6}$$

is a proportional (i.e., constant up to multiplication by scalar) vector in $\mathbb{R}^p$ as a function of λ in a neighborhood of λ_0.

The sufficient conditions for piecewise linearity implied by Proposition 1 are:

- L is piecewise quadratic as a function of β along the optimal path $\hat{\beta}(\lambda)$, when X, y are assumed constant at their sample values; and
- J is piecewise linear as a function of β along this path.

We devote the rest of this paper to examining some families of regularized problems which comply with these conditions, asserting the piecewise linearity of their regularized paths and designing efficient algorithms for tracking these paths.

On the loss side, this leads us to consider functions L which are:

- Pure quadratic loss functions, like that of the linear regression.
- A mixture of quadratic and linear pieces, like Huber's loss [12]. These loss functions are of interes t because they generate robust modeling tools. They will be the focus of section 3.
- Loss functions which are piecewise linear. These include several widely used loss functions, like the hinge loss of support vector machines (4) and the loss used in quantile regression [13]:

$$L(\mathsf{y}, X\beta) = \sum_i l(y_i, \beta^\mathsf{T} \mathbf{x}_i),$$

where

$$l(y_i, \beta^\mathsf{T}\mathbf{x}_i) = \begin{cases} \tau \cdot (y_i - \beta^\mathsf{T}\mathbf{x}_i) & \text{if } y_i - \beta^\mathsf{T}\mathbf{x}_i \geq 0 \\ (1-\tau) \cdot (\beta^\mathsf{T}\mathbf{x}_i - y_i) & \text{otherwise} \end{cases}, \qquad (7)$$

and $\tau \in (0, 1)$ indicates the quantile of interest.

On the penalty side, our results lead us to consider the ℓ_1 and ℓ_∞ penalties as building blocks for piecewise linear solution paths. We are not aware of any use of the ℓ_∞ penalty in statistical data modeling. Thus we limit the discussion in this paper to the ℓ_1 penalty and its variants (like total variation penalties discussed in Section 4).

ℓ_1 regularization has several favorable statistical properties. Using ℓ_1 regularization results in "sparse" solutions with a relatively small fraction of non-zero coefficients, as opposed to ℓ_2 regularization which forces all non-zero coefficients [20]. In particular, if the number of predictors is larger than the number of observations ($p > n$), then for any λ, there exists an ℓ_1-regularized solution with at most n non-zero coefficients [18]. Thus, in situations where the number of relevant variables is small and there are a lot of irrelevant "noise" variables, ℓ_1 regularization may prove far superior to ℓ_2 regularization from a prediction error perspective. Indeed, a sense can be defined in which ℓ_1 regularized problems have lower complexity than ℓ_2 regularized ones in high dimension [1, 8, 15]. From an inference/interpretation perspective, ℓ_1 regularization gives "smooth" variable selection and more compact models than ℓ_2 regularization [4]. In the case of orthogonal wavelet bases, the soft thresholding method proposed by [3], which is equivalent to ℓ_1 regularization, is asymptotically nearly optimal (in a minimax sense) over a wide variety of loss functions and estimated functions.

It is not surprising, therefore, that ℓ_1 regularization and its variants have been widely and successfully used in different fields, including engineering and signal processing (as basis pursuit and wavelet thresholding), machine learning (as boosting and ℓ_1 SVM) and, obviously, statistics, where ℓ_1 and total variation penalties are prevalent.

3. Almost Quadratic Loss Functions with ℓ_1 Penalty. In this section, we first define a family of "almost quadratic" loss functions whose ℓ_1-penalized versions generate piecewise linear solution paths. We formulate and prove an algorithm, which is an extension of the LAR-Lasso algorithm, that generates the ℓ_1-regularized solution paths for all members of this family. We then concentrate on two members of this family — Huberized Lasso for regression and ℓ_1-penalized Huberized squared hinge loss for classification — which define new, robust, efficient and adaptable modeling tools.

An R implementation of these tools is available from the second author's homepage[2].

3.1. *Main results.* We fix the penalty to be the ℓ_1 penalty:

$$J(\beta) = \|\beta\|_1 = \sum_j |\beta_j|, \tag{8}$$

and the loss is required to be differentiable and piecewise quadratic in a fixed function of the sample response and the "prediction" $\beta^T x$:

$$L(\mathbf{y}, X\beta) = \textstyle\sum_i l(y_i, \beta^T \mathbf{x}_i) \tag{9}$$

$$l(y, \beta^T \mathbf{x}) = a(r)r^2 + b(r)r + c(r) \tag{10}$$

where $r = (y - \beta^T \mathbf{x})$ is the residual for regression and $r = (y\beta^T \mathbf{x})$ is the margin for classification; and $a(\cdot), b(\cdot), c(\cdot)$ are piecewise constant functions, with a finite (usually small) number of pieces, defined so as to make the function l differentiable *everywhere*.

Some examples from this family are:

- The squared error: $l(y, \beta^T \mathbf{x}) = (y - \beta^T \mathbf{x})^2$, i.e., $a \equiv 1, b \equiv 0, c \equiv 0$.
- Huber's loss function with *fixed* knot t:

$$l(y, \beta^T \mathbf{x}) = \begin{cases} (y - \beta^T \mathbf{x})^2 & \text{if } |y - \beta^T \mathbf{x}| \le t \\ 2t|y - \beta^T \mathbf{x}| - t^2 & \text{otherwise.} \end{cases} \tag{11}$$

- Squared hinge loss for classification:

$$l(y, \beta^T \mathbf{x}) = \begin{cases} (1 - y\beta^T \mathbf{x})^2 & \text{if } y\beta^T \mathbf{x} \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

  Note that the hinge loss of the support vector machine (4) does not belong to this family as it is not differentiable at $y\beta^T \mathbf{x} = 1$.

Theorem 2 *All regularized problems of the form (1) using (9)–(11) (with r being either the residual or the margin) generate piecewise linear optimal coefficient paths $\hat{\beta}(\lambda)$ as the regularization parameter λ varies.*

Proof Since we do not assume twice-differentiability of either the loss or the penalty, (but we do assume differentiability of the loss), we use Proposition 1 as an intuitive argument only, and prove the theorem formally using the Karush-Kuhn-Tucker (KKT) formulation of the optimization problem implied.

[2] www.stat.lsa.umich.edu/∼jizhu/piecewise

We code $\beta_j = \beta_j^+ - \beta_j^-$, with $\beta_j^+ \geq 0, \beta_j^- \geq 0$, and write the regularized optimization problem:

$$\begin{aligned} \min_{\beta^+,\beta^-} \quad & \sum_i l(y_i, (\beta^+ - \beta^-)^\intercal \mathbf{x}_i) + \lambda \sum_j (\beta_j^+ + \beta_j^-) \\ \text{subject to} \quad & \beta_j^+ \geq 0,\ \beta_j^- \geq 0,\ \ \forall j. \end{aligned}$$

Then the Lagrange dual function is:

$$\sum_i l(y_i, (\beta^+ - \beta^-)^\intercal \mathbf{x}_i) + \lambda \sum_j (\beta_j^+ + \beta_j^-) - \sum_j \lambda_j^+ \beta_j^+ - \sum_j \lambda_j^- \beta_j^-.$$

The derivatives of the dual and the corresponding KKT conditions imply:

$$\begin{aligned} (\nabla L(\beta))_j + \lambda - \lambda_j^+ &= 0 \\ -(\nabla L(\beta))_j + \lambda - \lambda_j^- &= 0 \\ \lambda_j^+ \beta_j^+ &= 0 \\ \lambda_j^- \beta_j^- &= 0 \end{aligned}$$

Using these we can figure that at the optimal solution for fixed λ the following scenarios should hold:

$$\begin{aligned} \lambda = 0 \quad & \Rightarrow\ (\nabla L(\beta))_j = 0\ \forall j \text{ (unconstrained solution)} \\ \beta_j^+ > 0, \lambda > 0 \quad & \Rightarrow\ \lambda_j^+ = 0\ \Rightarrow\ (\nabla L(\beta))_j = -\lambda < 0\ \Rightarrow \\ & \Rightarrow\ \lambda_j^- > 0\ \Rightarrow\ \beta_j^- = 0 \\ \beta_j^- > 0, \lambda > 0 \quad & \Rightarrow\ \lambda_j^- = 0\ \Rightarrow\ (\nabla L(\beta))_j = \lambda > 0\ \Rightarrow \\ & \Rightarrow\ \lambda_j^+ > 0\ \Rightarrow\ \beta_j^+ = 0 \\ |(\nabla L(\beta))_j| > \lambda \quad & \Rightarrow \text{ contradiction} \end{aligned}$$

Based on these possible scenarios we can see that:

- Variables can have non-zero coefficients at $\hat{\beta}(\lambda)$ only if their "generalized absolute correlation" $|\nabla L(\hat{\beta}(\lambda))_j|$ is equal to λ. Thus, for every value of λ we have a set of "active" variables $\mathcal{A} = \{j : \hat{\beta}_j(\lambda) \neq 0, j = 1, \ldots, p\}$ such that:

$$\begin{aligned} j \in \mathcal{A} \quad & \Rightarrow\ |\nabla L(\hat{\beta}(\lambda))_j| = \lambda,\ \operatorname{sgn}(\nabla L(\hat{\beta}(\lambda))_j) = -\operatorname{sgn}(\hat{\beta}(\lambda)_j) \\ j \notin \mathcal{A} \quad & \Rightarrow\ |\nabla L(\hat{\beta}(\lambda))_j| \leq \lambda \end{aligned}$$

- When λ changes, the direction in which $\hat{\beta}(\lambda)$ is moving, i.e. $\frac{\partial \hat{\beta}(\lambda)}{\partial \lambda}$, should be such that it maintains the conditions $|\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}}| = \lambda$ and $|\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}^C}| \leq \lambda$.

So, if we know what the "active" set $\mathcal{A}$ is, it is a simple task to check that as long as we are in a region where the loss is twice differentiable and the penalty is right differentiable, we will have

$$\frac{\partial \hat{\beta}(\lambda)_{\mathcal{A}}}{\partial \lambda} = -(\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}})^{-1} \mathrm{sgn}(\hat{\beta}(\lambda)_{\mathcal{A}}), \tag{13}$$

which is just a version of (5), limited to only the active variables and substituting the ℓ_1 penalty for J.

For the family of almost quadratic loss functions, we can derive $\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}}$ explicitly, because the second derivative of the constant and linear parts in (11) is 0. Thus we obtain:

$$\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}} = \sum_i 2a(r(y_i, \hat{\beta}(\lambda)_{\mathcal{A}}^{\mathsf{T}} \mathbf{x}_{\mathcal{A}i})) \mathbf{x}_{\mathcal{A}i} \mathbf{x}_{\mathcal{A}i}^{\mathsf{T}}.$$

Since $a(\cdot)$ is a piecewise constant function, then $\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}}$ and $\partial \hat{\beta}(\lambda)_{\mathcal{A}}/\partial \lambda$ are also piecewise constant; therefore, the solution path $\hat{\beta}(\lambda)$ is piecewise linear.

When one of the following "events" occurs, twice differentiability is violated and hence the direction in (14) will change:

- Add a variable: A new variable should join $\mathcal{A}$, i.e., we reach a point where $|\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}^C}| \leq \lambda$ would cease to hold if $\hat{\beta}(\lambda)$ keeps moving in the same direction.
- Drop a variable: A coefficient in $\mathcal{A}$ hits 0. In that case, we reach a non-differentiability point in the penalty and we can see that $\mathrm{sgn}(\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}}) = -\mathrm{sgn}(\beta_{\mathcal{A}})$ would cease to hold if we continue in the same direction. Thus we need to drop the coefficient hitting 0 from $\mathcal{A}$.
- Cross a knot: A "generalized residual" $r(y_i, \hat{\beta}(\lambda)^{\mathsf{T}} \mathbf{x}_i)$ hits a non-twice differentiability point (a "knot") in L, for example the "Huberizing" point t in (12), or the hinge point 1 in (13).

So we conclude that the path $\hat{\beta}(\lambda)$ will be piecewise linear, with the direction given by (14) and direction changes occurring whenever one of the three events above happens. When it happens, we need to update $\mathcal{A}$ or $a(r)$ to get a feasible scenario and re-calculate the direction using (14). ■

Based on the arguments in the proof we can derive a generic algorithm to generate coefficient paths for all members of the "almost quadratic" family of loss functions with ℓ_1 penalty. The LAR-Lasso algorithm [4] is a simplified version of this algorithm since "knot crossing" events do not occur in the

Lasso (as the loss is twice differentiable). Our algorithm starts at $\lambda = \infty$ and follows the linear pieces, while identifying the "events" and re-calculating the direction when they occur.

Algorithm 1 *An algorithm for "almost quadratic" loss with ℓ_1 penalty*

1. *Initialize:*

$$\beta = 0,\ \mathcal{A} = \arg\max_j |\nabla L(\beta)|_j,\ \gamma_{\mathcal{A}} = -sgn(\nabla L(\beta))_{\mathcal{A}},\ \gamma_{\mathcal{A}^C} = 0.$$

2. *While (*$\max|\nabla L(\beta)| > 0$*)*
   (a) $d_1 = \min\{d > 0 : |\nabla L(\beta + d\gamma)_j| = |\nabla L(\beta + d\gamma)_{\mathcal{A}}|,\ j \notin \mathcal{A}\}$
   (b) $d_2 = \min\{d > 0 : (\beta + d\gamma)_j = 0,\ j \in \mathcal{A}\}$ *(hit* 0*)*
   (c) $d_3 = \min\{d > 0 : r(y_i, (\beta + d\gamma)^{\mathsf{T}}\mathbf{x}_i)$ *hits a "knot",* $i = 1, \ldots, n\}$
   (d) *set* $d = \min(d_1, d_2, d_3)$
   (e) $\beta \leftarrow \beta + d\gamma$
   (f) *If* $d = d_1$ *then add variable attaining equality at* d *to* $\mathcal{A}$.
   (g) *If* $d = d_2$ *then remove variable attaining* 0 *at* d *from* $\mathcal{A}$.
   (h) *If* $d = d_3$ *for observation* i^**, then decide an appropriate value for* $a(r(y_{i^*}, \beta^{\mathsf{T}}\mathbf{x}_{i^*}))$ *from (11).*
   (i) $C = \sum_i a(r(y_i, \beta^{\mathsf{T}}\mathbf{x}_i))\mathbf{x}_{\mathcal{A},i}\mathbf{x}_{\mathcal{A},i}^{\mathsf{T}}$
   (j) $\gamma_{\mathcal{A}} = C^{-1} \cdot sgn(\beta_{\mathcal{A}})$
   (k) $\gamma_{\mathcal{A}^C} = 0$

It should be noted that our formulation here of the "almost quadratic" family with ℓ_1 penalty has ignored the existence of a non-penalized intercept. This has been done for simplicity of exposition, however incorporating a non-penalized intercept into the algorithm is straight forward.

3.2. *Computational considerations.* What is the computational complexity of running Algorithm 1 on a dataset with n observations and p variables? The major computational cost for each step involves figuring out the step length in (2a–2d), and updating the new direction in (2j). The former takes $O(np)$ calculations, and the latter requires $O(|\mathcal{A}|^2)$ computations by using inverse updating and downdating.

It is difficult to predict the number of steps on the solution path for any arbitrary data. According to our experience, the total number of steps taken by the algorithm is on average $O(n)$. This can be heuristically understood

as follows: if $n > p$, it takes $O(p)$ steps to add all variables and $O(n)$ steps for knot crossing; if $n < p$, since at most n variables are allowed in the fitted model, it takes $O(n)$ steps for both adding variables and crossing knots; the "drop events" are usually rare $O(1)$. Since the maximum value of $|\mathcal{A}|$ is $\min(n, p)$, it suggests the overall computational cost is $O(n^2 p)$.

Notice that if there is no knot crossing event, such as in the Lasso, the total number of steps becomes $O(\min(n, p))$, and the overall computational cost can be reduced to $O(np \min(n, p))$.

3.3. *The Huberized Lasso for regression.* We now concentrate on a detailed statistical and computational analysis of two members of the "almost quadratic" family of loss functions — one for regression and one for classification. We use Algorithm 1 to generate the solution paths $\hat{\beta}(\lambda)$ for the ℓ_1 regularized versions of these fitting problems.

We first consider the Huberized Lasso for regression. The loss is given by (12). It is "almost quadratic" as defined in Section 3.1, and so Theorem 2 and Algorithm 1 apply to its ℓ_1 regularized solution paths.

In [12], Huber has shown that "Huberizing" has asymptotic optimality properties in protecting against "contamination" of the assumed normal errors. Huber also suggests the knot $t = 1.345\sigma$, where σ^2 is the variance of the non-contaminated normal, and describes an iterative algorithm for fitting the data and selecting t, when the variance is unknown. For clarity and brevity we use a fixed-knot approach in our example.

Prostate cancer dataset. The "prostate cancer" dataset, used in the original Lasso paper [20], is available from:

http://www-stat.stanford.edu/ElemStatLearn/.

We use this dataset to compare the prediction performance of the "Huberized" Lasso to that of the Lasso on the original data and after we artificially "contaminate" the data by adding large constants to a small number of responses.

We use the training-test configuration as in [9], page 48. The training set consists of 67 observations and the test set of 30 observations. We ran the Lasso and the Huberized Lasso with knot at $t = 1$ on the original dataset, and on the "contaminated" dataset where 5 has been added to the response of six observations, and 5 was subtracted from the response of six other observations.

Figure 1 shows the mean squared error on the 30 test set observations for the four resulting regularized solution paths from solving the Lasso and Huberized Lasso for all possible values of λ on the two datasets. We observe

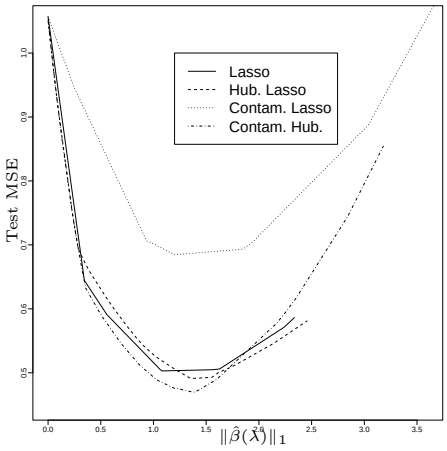

FIG 1. *Mean test squared error of the models along the regularized path for the Lasso (solid), Huberized Lasso (dashed), Lasso on contaminated training data (dotted) and Huberized Lasso on contaminated data (dash-dotted). We observe that the Huberized Lasso is not affected by contamination, while the Lasso performance deteriorates significantly.*

that on the non-contaminated data, the Lasso (solid) and Huberized Lasso (dashed) perform quite similarly. When we add contamination, the Huberized Lasso (dash-dotted) does not seem to suffer from it at all, in that its best test set performance is comparable to that of both regularized models on the non-contaminated data. The prediction performance of the standard Lasso (dotted), on the other hand, deteriorates significantly (t-test p-value 0.045) when contamination is added, illustrating the lack of robustness of squared error loss.

The two Lasso solutions contain 9 linear pieces each, while the Huber-Lasso path for the non-contaminated data contains 41 pieces, and the one for the contaminated data contains 39 pieces; both agree with our conjecture in Section 3.2 that the number of steps is $O(n)$. Figure 2 shows the solution paths for the contaminated Lasso model and the contaminated Huber-Lasso model. We observe that the two paths are quite different and the two best models (corresponding to the solid vertical lines) are also different.

3.4. *The Huberized squared hinge loss for classification.* For classification we would like to have a loss which is a function of the margin: $r(y, \beta^{\mathsf{T}}\mathbf{x}) = (y\beta^{\mathsf{T}}\mathbf{x})$. This is true of all loss functions typically used for classification, like the negative binomial log-likelihood for logistic regression, the hinge loss for the support vector machine, and the exponential loss for boosting. The

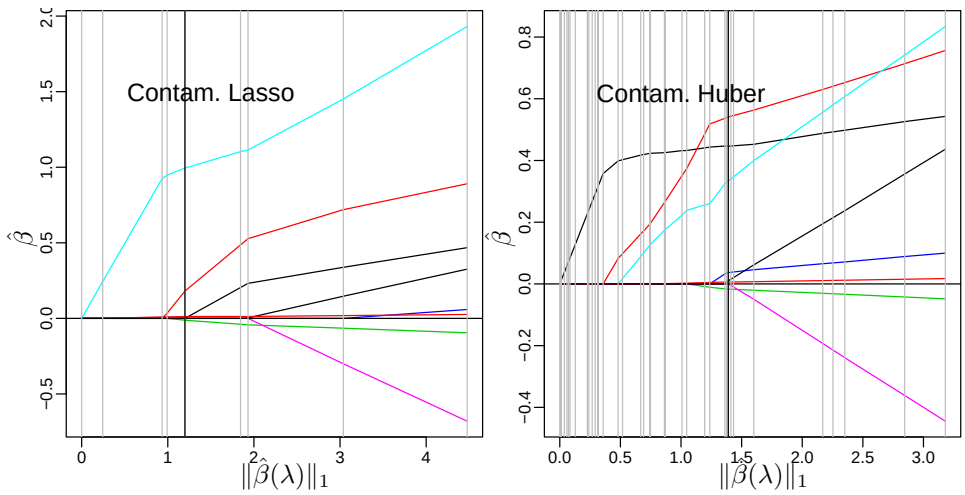

FIG 2. *Solution paths of the Lasso (left) and the Huberized Lasso (right) on the contaminated prostate cancer training data. The vertical grey lines correspond to the steps along the solution paths. The vertical solid lines correspond to the models that give the best performances on the test data.*

properties we would like from our classification loss are:

- We would like it to be "almost quadratic", so we can apply the efficient Algorithm 1.
- We would like it to be robust, i.e., linear for large absolute value negative margins (like the logistic or hinge), so that outliers would have a small effect on the fit.

This leads us to suggesting for classification the "Huberized squared hinge loss", i.e., (13) "Huberized" at $t < 1$:

$$l(y, \beta^{\mathsf{T}}\mathbf{x}) = \begin{cases} (1-t)^2 + 2(1-t)(t - y\beta^{\mathsf{T}}\mathbf{x}) & \text{if } y\beta^{\mathsf{T}}\mathbf{x} \le t \\ (1 - y\beta^{\mathsf{T}}\mathbf{x})^2 & \text{if } t < y\beta^{\mathsf{T}}\mathbf{x} \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

It is a simple task to show that

$$\text{argmin}_f \mathbf{E}_y l(y, f) = 2\Pr(y = 1) - 1.$$

Hence the population minimizer of the Huberized squared hinge loss gives the correct sign for classification.

To illustrate the robustness of our Huberized squared hinge loss (15), and its computational superiority over the logistic loss, we considered the

following simple simulated example: $\mathbf{x} \in \mathbb{R}^2$ with class centers at $(-1,-1)$ (class "-1") and $(1,1)$ (class "1") with one big outlier at $(30,100)$, belonging to the class "-1". The Bayes model, ignoring the outlier, is to classify to class "1" if and only if $x_1 + x_2 > 0$.

Figure 3 shows the regularized model paths and misclassification rate for this example using the logistic loss (left), the Huberized squared hinge loss (middle) and the squared hinge loss (right), all with ℓ_1 penalty. We observe that the logistic and Huberized regularized model paths are both less affected by the outlier than the non-Huberized squared loss. However, logistic loss does not allow for efficient calculation of the ℓ_1 regularized path.

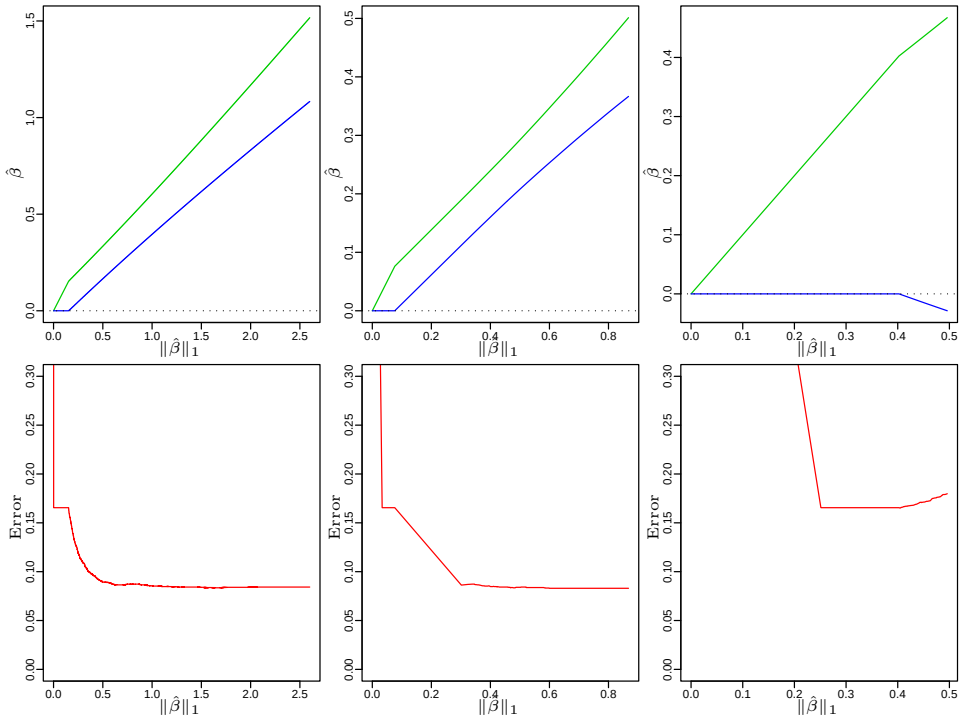

FIG 3. *Regularized paths and prediction errors for the logistic loss (left), Huberized squared hinge loss (middle) and squared hinge loss (right). The logistic loss and the Huberized squared hinge loss are both less affected by the outlier. However, the Huberized regularized path can be computed via the efficient Algorithm 1.*

4. Non-parametric Regression, Total Variation Penalties and Piecewise Linearity. Total variation penalties and closely associated spline methods for non-parametric regression have experienced a surge of interest

in the statistics literature in recent years. The total variation of a univariate differentiable function $f(x)$ is:

$$TV_{dif}(f) = \int_{-\infty}^{\infty} |f'(x)|dx.$$

If f is non-differentiable on a countable set $x_1, x_2, ...$, then $TV(f)$ is the sum of $TV_{dif}(f)$, calculated over the differentiable set only and the absolute "jumps" in f where it is non-continuous. In what follows we assume the range of f is limited to $[0, 1]$.

Total variation penalties tend to lead to regularized solutions which are polynomial splines. [14] investigate the solutions to total-variation penalized least squares problems. They use total variation of $(k-1)$th order derivatives:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \cdot TV(f^{(k-1)}). \tag{15}$$

They show (proposition 1) that there always exists a solution $\hat{f}_{k,\lambda}$ such that $\hat{f}_{k,\lambda}^{(k-1)}$ is piecewise constant. That is, $\hat{f}_{k,\lambda}$ is a polynomial spline of order k (or degree $k-1$). For $k \in \{1, 2\}$ the knots of the spline solutions are guaranteed to be at the data points $x_i, i = 1, \ldots, n$.

[2] consider a similar setup, and propose the taut-string method. The resulting solution is a piecewise constant function in x with possible jumps at the data points. [2] also consider the "local squeezing" method, which leads to minimizing the following criterion:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \sum_{i=2}^{n}\lambda_i|f(x_i) - f(x_{i-1})|, \tag{16}$$

with data driven λ_i. [2] propose an algorithm to adaptively find λ_i and the corresponding $\hat{f}$ via grid search.

In these examples the solutions are polynomial splines of degree 0 or 1, with knots at the data points. More generally, consider a polynomial spline f of order k, with h knots located at $0 < t_1 < ... < t_h < 1$, that is:

$$f(x) = \sum_{j=1}^{h}\beta_j(x - t_j)_+^{k-1} + q(x), \tag{17}$$

where $q(x)$ is a polynomial of degree $(k-1)$. The total variation of the $(k-1)$th derivative of f clearly corresponds to an ℓ_1 norm of the set of

coefficients of the appropriate spline basis functions, anchored at the knots:

$$TV(f^{(k-1)}) = (k-1)! \cdot \sum_{j=1}^{h} |\beta_j|. \tag{18}$$

If the knots $t_1, ..., t_h$ are fixed in advance (e.g., at the data points), then a total variation penalized problem is equivalent to an ℓ_1-penalized regression problem, with $p = h$ derived predictors. If we also employ squared error loss, we get a Lasso problem, and we can use the LAR-Lasso algorithm to compute the complete regularized solution path [3]. This leads to essentially the same algorithm as Algorithm 2 of [14] for finding the regularized path for any k with a *fixed, pre-determined* set of candidate knots. [14] also show that this algorithm can in fact generate an exact path of solutions for $k \in \{1, 2\}$, because the knots are guaranteed to be at the data points.

4.1. *Locally adaptive regression splines.* We now concentrate on the family of penalized problems defined by Mammen and van de Geer (16). As we mentioned, [14] develop an exact method for finding $\hat{f}_{k,\lambda}$ when $k \in \{1, 2\}$ and approximate methods for $k > 2$ (where the knots of the optimal solutions are not guaranteed to be at the data points). We now show how we can use our approach to find the spline solution $\hat{f}_{k,\lambda}$ exactly for any natural k. The resulting algorithms get practically more complicated as k increases, but their theoretical computational complexity remains fixed.

When the knots are not guaranteed to be at the data points, we can still write the total variation of polynomial splines as the sum of ℓ_1 norms of coefficients of basis functions, as in (19). However, we do not have a finite pre-defined set of candidate basis functions. Rather, we are dealing with an infinite set of candidate basis functions of the form:

$$\mathcal{X} = \{(x-t)_+^{k-1} \ : \ 0 \le t \le 1\}.$$

Our algorithm for tracking the regularized solution path $\hat{f}_{k,\lambda}$ in this case proceeds as follows. We start at the solution for $\lambda = \infty$, which is the least squares $(k-1)$th degree polynomial fit to the data. Given a solution $\hat{f}_{k,\lambda_0}$ for some value of λ_0, which includes n_{λ_0} knots at $t_1, ..., t_{n_{\lambda_0}}$, denote:

$$\mathbf{z}(x) = \left(1, x, x^2, ..., x^{k-1}, (x-t_1)_+^{k-1}, ..., (x-t_{n_{\lambda_0}})_+^{k-1}\right)^{\mathsf{T}}$$

[3] The only difference from the standard Lasso is the existence of k non-penalized coefficients for the polynomial $q(x)$, instead of the intercept only for the Lasso. This requires only a slight modification to the LAR-Lasso algorithm.

is the current predictor vector, so that following (18) we can write:

$$\hat{f}_{k,\lambda_0}(x) = \hat{\beta}(\lambda_0)^\intercal \mathbf{z}(x).$$

Following the logic of the LAR-Lasso algorithm, we see that the solution will change as:

$$\hat{f}_{k,\lambda_0-d} = (\hat{\beta}(\lambda_0) + d\gamma)^\intercal \mathbf{z}(x),$$

where $\gamma = -(k-1)! \cdot (Z^\intercal Z)^{-1} \cdot \mathbf{s}$, $Z = (\mathbf{z}(x_1), ..., \mathbf{z}(x_n))^\intercal$ and $\mathbf{s} \in \mathbb{R}^{k+n_{\lambda_0}}$ is a vector with 0 components corresponding to $1, x, ..., x^{k-1}$ and ± 1 components corresponding to each $(x - t_j)_+^{k-1}$ (with the sign being the opposite of the sign of $(x - t_j)_+^{k-1\,\intercal}(\mathbf{y} - \hat{f}_{k,\lambda_0})$). What we now need to identify is the value of λ at which an additional knot needs to be added, and the location of that knot. Consider first a fixed knot candidate t. Then we can see that the LAR-Lasso criterion for adding this knot to the set of "active" knots is:

$$|\mathbf{x}_t^\intercal(\mathbf{y} - Z\hat{\beta}(\lambda_0) - (\lambda_0 - \lambda)Z\gamma)| = \lambda.$$

where $\mathbf{x}_t = (\mathbf{x} - t)_+^{k-1}$ (column vector of length n). More explicitly, define:

$$\lambda_+(t) = \frac{\mathbf{x}_t^\intercal(\mathbf{y} - Z\hat{\beta}(\lambda_0) - \lambda_0 Z\gamma)}{1 - \mathbf{x}_t^\intercal Z\gamma}, \tag{19}$$

$$\lambda_-(t) = \frac{\mathbf{x}_t^\intercal(\mathbf{y} - Z\hat{\beta}(\lambda_0) - \lambda_0 Z\gamma)}{-1 - \mathbf{x}_t^\intercal Z\gamma}. \tag{20}$$

Then we can write:

$$\lambda(t) = \begin{cases} \max(\lambda_+(t), \lambda_-(t)) & \text{if } \max(\lambda_+(t), \lambda_-(t)) \leq \lambda_0 \\ \min(\lambda_+(t), \lambda_-(t)) & \text{if } \max(\lambda_+(t), \lambda_-(t)) > \lambda_0 \end{cases}. \tag{21}$$

Now we see that we can in fact let t be a parameter and find the next knot to be added to the optimal solution path by maximizing $\lambda(t)$, that is:

$$\lambda_{add} = \max_{t \in (0,1) \setminus \{t_1, ..., t_{n_{\lambda_0}}\}} \lambda(t) \tag{22}$$

is the value of λ where we stop moving in direction γ, add a knot at the argument of the maximum, and re-calculate the direction γ.

Solving (23) requires finding the local extrema of the functions in (22), which are rational functions within each interval between two points which are either data points or knots (with numerator and denominator both of degree $k-1$). Thus, a reasonable tactic is to find the extrema within each such interval, then compare them between the intervals to find the overall solution to (23). For smaller values of k it can be solved manually and exactly:

- For $k \in \{1, 2\}$, we get a ratio of constant or linear functions in (22), and therefore the extrema — and the knots — are guaranteed to be at the data points. [14] have used this fact in formulating an exact algorithm for finding the path of piecewise-constant or piecewise-linear solutions $\hat{f}_{k,\lambda}$.
- For $k = 3$ we get a ratio of quadratics in (22), and we can find the extrema within each segment analytically. These extrema may not correspond to the segment's end points, and so we may have knots that are not at data points.

Assuming we have the code to solve the maximization problem in (23), Algorithm 2 gives a general schema for following the solution path $\hat{f}_{k,\lambda}$ for any value of k.

Algorithm 2 *Tracking the path of TV-penalized solutions*

1. *Initialize:*

$$\begin{aligned}
f(x) &= (1, x, ..., x^{k-1})^\intercal \beta_{ls} \text{ is the LS polynomial fit of degree } k-1 \\
u &= \arg\max_{t \in (0,1)} |(\mathbf{x} - t)_+^{k-1\,\intercal} (\mathbf{y} - f(\mathbf{x}))| \text{ (assumed unique)} \\
\mathcal{T} &= \{u\} \\
\lambda_0 &= (k-1)! \cdot |(\mathbf{x} - u)_+^{k-1\,\intercal} (\mathbf{y} - f(\mathbf{x}))| \\
Z &= (\mathbf{1}, \mathbf{x}, ..., \mathbf{x}^{k-1}, (\mathbf{x} - u)_+^{k-1}) \\
\hat{\beta}(\lambda_0) &= (\beta_{ls}^\intercal, 0)^\intercal \\
\mathbf{s} &= \left(\mathbf{0}_k^\intercal, -sgn\{(\mathbf{x} - u)_+^{k-1\,\intercal} (\mathbf{y} - f(\mathbf{x}))\}\right)^\intercal
\end{aligned}$$

2. *While* $\sum_i (y_i - f(x_i))^2 > 0$
   (a) *Set* $\gamma = -(k-1)!(Z^\intercal Z)^{-1}\mathbf{s}$
   (b) $\forall t \in (0,1) \setminus \mathcal{T}$ *define* $\lambda_+(t), \lambda_-(t), \lambda(t)$ *as in (20, 21, 22)*
   (c) *Solve the maximum problem in (23) to get* λ_{add}
   (d) *Let* $\lambda_{rem} = \lambda_0 - \min\{d > 0 : \exists j > k \text{ s.t. } \hat{\beta}_j(\lambda_0) + d\gamma_j = 0\}$
   (e) *If* $\lambda_{add} > \lambda_{rem}$ *add a knot at the point attaining the maximum in (23), update:*
   add the new knot to $\mathcal{T}$*;*
   add a column to Z*; and*
   add the appropriate sign to $\mathbf{s}$*.*
   (f) *Similarly, if* $\lambda_{add} < \lambda_{rem}$ *remove the knot attaining* 0 *at* λ_{rem}*.*

(g) In both cases, update:
$\hat{\beta}(\lambda_0) \leftarrow \hat{\beta}(\lambda_0) + (\lambda_0 - \max(\lambda_{add}, \lambda_{rem}))\gamma$;
$\lambda_0 = \max(\lambda_{add}, \lambda_{rem})$.

Since we can never have more than $(n-k)$ knots in a solution $\hat{f}_{k,\lambda}$ [18], the computational complexity of each iteration of the algorithm is bounded at $O(n^2)$ calculations for finding the next knot and $O(n^2)$ for calculating the next direction (using updating formulae). The number of steps of the algorithm is difficult to bound, but from our limited experience seems to behave like $O(n)$ (which is the conjecture of [14]).

4.2. *Simple data example: $k = 3$.* We illustrate our algorithm on a simple data example. We select 100 x samples uniformly on $(0, 1)$. We draw the corresponding y values as $N(g(x), 0.03^2)$, where $g(x)$ is a polynomial spline with knots at $0.25, 0.5, 0.75$:

$$g(x) = 0.125 + 0.125x - x^2 + 2(x - 0.25)_+^2 - 2(x - 0.5)_+^2 + 2(x - 0.75)_+^2.$$

$g(x)$ is plotted as the solid line in Figure 4, and the noisy y values as circles. We can see that the magnitude of the noise is large enough, that the human eye cannot reasonably recover $g(x)$ visually (signal-to-noise ratio of about 1.4).

We apply our Algorithm 2 with $k = 3$. Figure 4 shows the resulting models after $5, 15$ and 50 iterations of the algorithm. After 5 iterations, the regularized spline contains 3 knots like the true g, but these are all around 0.5. The fitted model, drawn as a dashed curve, is clearly underfitted. The corresponding reducible squared prediction error is 9.5×10^{-4}.

After 15 iterations, the spline contains 4 knots, at $0.225, 0.255, 0.485, 0.755$. The first one has a small coefficient, and the other three closely correspond to the true knots in g. The resulting fit, drawn as a dotted curve, is a reasonable approximation of g, and the reducible squared error is about 3.1×10^{-4}.

After 50 iterations the model contains 10 knots and the data is clearly overfitted (dash-dotted curve, reducible squared error 8.2×10^{-4}). Although the algorithm should in principle continue until it interpolates the data, it terminates after 54 iterations and is numerically unable to further improve the fit. This is analogous to the situation described in [10] for kernel SVM, where the effective rank of the kernel matrix is significantly smaller than n, since many eigenvalues are effectively zero.

5. Using ℓ_1 Loss and Its Variants. Piecewise-linear non-differentiable loss functions appear in practice in both regression and classification problems. For regression, absolute value loss variants like the quantile regression

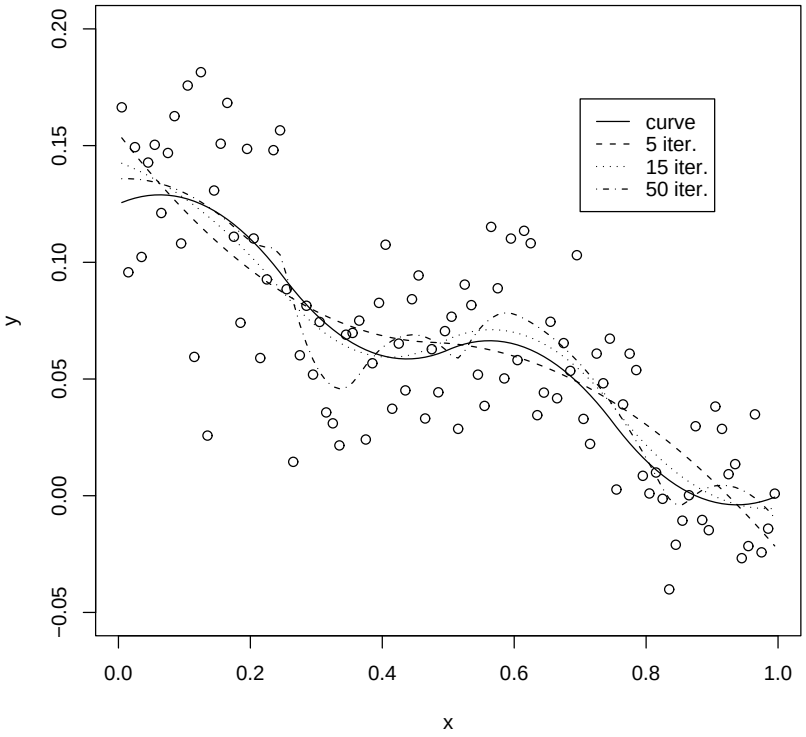

FIG 4. *Applying Algorithm 2 (with $k = 3$) to a simple data example where the underlying true function is a polynomial spline with knots at $0.25, 0.5, 0.75$. The solid curve indicates the underlying true function. The dashed curve, the dotted curve and the dash-dotted curve correspond to fitted models after* 5, 15 *and* 50 *steps respectively.*

loss (7) are quite popular [13]. For classification, the hinge loss (4) is of great importance, as it is the loss underlying support vector machines [24]. Here we consider a generalized formulation, which covers both of (7) and (4). The loss function has the form:

$$l(r) = \begin{cases} b_1 \cdot |a + r| & \text{if } a + r \geq 0 \\ b_2 \cdot |a + r| & \text{if } a + r < 0 \end{cases}, \tag{23}$$

with the generalized "residual" being $r = (y - \beta^{\mathsf{T}}\mathbf{x})$ for regression and $r = (y \cdot \beta^{\mathsf{T}}\mathbf{x})$ for classification.

When these loss functions are combined with ℓ_1 penalty (or total variation penalty, in appropriate function classes [13]), the resulting regularized prob-

lems can be formulated as linear programming problems. When the path of regularized solutions $\hat{\beta}(\lambda)$ is considered, it turns out to have interesting structure with regard to λ:

Proposition 3 *For loss functions of the form (24), there exists a set of values of the regularization parameter $\lambda_0 = 0 < \lambda_1 < \ldots < \lambda_m = \infty$ such that:*

- *The solution $\hat{\beta}(\lambda_k)$ is not uniquely defined, and the set of optimal solutions for each λ_k is a straight line in $\mathbb{R}^p$.*
- *For any $\lambda \in (\lambda_k, \lambda_{k+1})$, the solution $\hat{\beta}(\lambda)$ is fixed and equal to the minimum ℓ_1 norm solution for λ_k and the maximum ℓ_1 norm solution for λ_{k+1}.*

Proposition 3 generalizes observations on the path of solutions made in the context of quantile regression in [13] and in the context of 1-norm support vector machines in [27]. The arguments given in these references naturally generalize to prove it. Note that this leads to describing a regularized path which is *piecewise constant* as a function of the regularization parameter λ, with jumps at the values $\lambda_1, ..., \lambda_m$. However, it is still *piecewise linear* in the ℓ_1 norm of the solution path, $\|\hat{\beta}(\lambda)\|_1$.

The algorithm for computing the solution path follows the spirit of our earlier work [27]. For lack of space, we skip the details. We make a note that it is fundamentally *different* from the LARS-Lasso algorithm and Algorithm 1, because we are now dealing with a non-differentiable loss function.

An interesting variant of piecewise linear loss is to replace the ℓ_1 loss with an ℓ_∞ loss, which is also piecewise linear and non-differentiable. It leads to interesting "mini-max" estimation procedures, popular in many areas, including engineering and control. For example, [23] propose the use of ℓ_1-penalized ℓ_∞-loss solutions in an image reconstruction problem (but do not consider the solution path). The solution paths are piecewise-linear and path-following algorithms can be designed in the same spirit as the ℓ_1 loss case.

6. Discussion. In this paper we combine computational and statistical considerations in designing regularized modeling tools. We emphasize the importance of both appropriate regularization and robust loss functions for successful practical modeling of data. From a statistical perspective, we can consider robustness and regularization as almost independent desirable properties dealing with different issues in predictive modeling:

- Robustness mainly protects us against wrong assumptions about our error model. It does little or nothing to protect us against the uncer-

tainty about our model structure which is inherent in the finiteness of our data. For example, if our errors really are normal, then squared error loss minimizes the asymptotic variance of the coefficients, no matter how little data we have or how inappropriate our model is [12]. Using a robust loss in such a situation is *always* counter productive.

- Regularization deals mainly with the uncertainty about our predictive model structure by limiting the model space. Note, in this context, the equivalence between the "penalized" formulation (1) and a "constrained" formulation $\min_\beta L(\mathsf{y}, X\beta)$ subject to $J(\beta) \leq s$. The two formulations share the same solution path. The constrained formulation exposes the goal of regularization as "simplifying" the model estimation problem, by limiting the set of considered models.

There are many interesting directions in which our work can be extended. We now mention a few.

How can our geometric understanding of the regularized solution paths help us to analyze the statistical properties of the models along the path? For example, [4, 28] have offered analysis of the degrees of freedom of models along the Lasso path. This becomes much more challenging once we stray away from squared error loss.

Beyond the simple one-penalty formulation we examine here, we can consider having multiple penalties, either motivated by domain knowledge about the models we expect to find (e.g., [21], where a problem in mass spectroscopy inspires a two-penalty formulation) or because of statistical considerations. An example of the latter is the use of local penalties (i.e., ℓ_1 penalty with a penalty parameter per variable). For example, [2] suggest a solution which essentially fits the local penalty parameters to data. An interesting extension to our results in this paper would be to apply different local penalties to variables that participate in the current solution and ones that do not. Briefly, the suggestion is to adaptively set the "slope" of the local penalty vector as:

$$\frac{\partial \lambda_k}{\lambda} = \begin{cases} c & \text{if } \beta_k = 0 \\ 1 & \text{if } \beta_k \neq 0 \end{cases}.$$

If $c < 1$, then penalty parameters corresponding to 0 coefficients will decrease more slowly, and the corresponding coefficients will become non-0 later in the regularized path. The paths traced by this solutions can be followed by a path algorithm like the ones we designed here.

It is also worth noting that limiting our discussion to convex problems, for which efficient algorithms can be designed, leaves out some statistically well-motivated fitting approaches which lead to problems that are similar to

the ones we formulate, except they are non-convex optimization problems. The use of non-convex penalty was advocated by Fan and collaborators in several papers [1, 5, 6]. They expose the favorable variable selection and properties of the penalty functions they offer, which can be viewed as an improvement over the use of ℓ_1 penalty in that respect. [19] advocate the use of non-convex ψ-loss in the classification setting, minimizing the effect of outliers and misclassified points. Again, their approach can be viewed as a more robust version of our Huberized loss functions, with strong statistical motivation in terms of asymptotic behavior.

Acknowledgments. We thank the referee and the Associate Editor for their thoughtful and useful comments, and in particular for introducing us to the relevant literature on total variation penalties. We thank Brad Efron, Jerry Friedman, Trevor Hastie, Rob Tibshirani, Bin Yu and Tong Zhang for their helpful comments and suggestions.

REFERENCES

[1] Antoniadis, A. & Fan, J. (2001). Regularized wavelet approximations (with discussion). *Journal of American Statistical Association* **96**:939-967.

[2] Davies, P. & Kovac, A. (2001). Local extremes, runs, strings and multiresolution. *The Annals of Statistics* **29**:1-65.

[3] Donoho, D., Johnstone, I., Kerkyachairan, G. & Picard, D. (1995). Wavelet shrinkage: asymptopia? (with discussion). *Journal of the Royal Statistical Society B* **57**:201-337.

[4] Efron, B., Hastie, T., Johnstone, I.M. and Tibshirani, R. (2004). Least Angle Regression (with discussion). *The Annals of Statistics* **32**(2).

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

[6] Fan, J. & Peng, H. (2003). Nonconcave penalized likelihood with a diverging number of parameters. *The Annals of Statistics* **32**(3).

[7] Freund, Y. & Schapire, R.E. (1996). Experiments with a new boosting algorithm. *Proc. 13th International Conference on Machine Learning*.

[8] Friedman, H., Hastie, T., Rosset, S., Tibshirani, R. & Zhu, J. (2004). Discussion of 3 boosting papers. *The Annals of Statistics. The Annals of Statistics* **32**(1).

[9] Hastie, T., Tibshirani, R. & Friedman, J. (2001). *Elements of Statistical Learning*. Springer-Verlag, New York.

[10] Hastie, T., Rosset, S., Tibshirani, R. & Zhu, J. (2004). The entire regularization path of the support vector machine. *Journal of Machine Learning Research* **5**(Oct)

[11] Hoerl, A & Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* **12**(3):55-67.

[12] Huber, P. (1964). Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, **35**(1):73-101.

[13] Koenker, R., Ng, P. & Portnoy, S. (1994) Quantile smoothing splines. *Biometrika* **81**(4): 673–680.

[14] Mammen, E. and van de Geer, S. (1997). Locally Adaptive Regression Splines. *The Annals of Statistics* **25**(1):387-413.

[15] Ng, A. (2004) Feature selection, L_1 vs. L_2 regularization, and rotational invariance. *Proceedings of the Twenty-First International Conference on Machine Learning.* Banff, Canada.

[16] Osborne, M., Presnell, B. & Turlach, B. (2000a). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis* **20**:389-404.

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

[18] Rosset, S., Zhu, J. & Hastie, T. (2004). Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, **5**(Aug).

[19] Shen, X., Tseng, G., Zhang, X. & Wong, W. (2003). On ψ-learning. *Journal of the American Statistical Association* **98**.

[20] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society, Series B* **58**(1).

[21] Tibshrani, R., Saunders, M., Rosset, S., Zhu, J. & Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society, Series B* **67**(1).

[22] Tondel, P., Johansen, T. A. & Bemporad, A. (2003). An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica* **39**(3):489-497.

[23] Tsuda, K. & Raetsch, G. (2005). Image reconstruction by linear programming. *IEEE Trans. on Image Processing.* **14**(6):737-744.

[24] Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer.

[25] Wahba, G. (1990) Spline models for observational data. SIAM. CBMS-NSF Regional Conference Series in Applied Mathematics, V. 59.

[26] Zhang, T. (2004). Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics.* **32**(1).

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

[28] Zou, H., Hastie, T. and Tibshirani, R. (2004). On the degrees of freedom of the Lasso. *Techincal report.*

Predictive Modeling Group
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA
E-mail: srosset@us.ibm.com

Department of Statistics
University of Michigan
439 West Hall
1085 South University Avenue
Ann Arbor, MI 48109
USA
E-mail: jizhu@umich.edu