Piecewise Linear Regularized Solution Paths

Saharon Rosset
Data Analytics Research Group
IBM T.J. Watson Research Center
Yortown Heights, NY 10598

Ji Zhu
Department of Statistics
University of Michigan
Ann Arbor, MI 48109

Abstract

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 guaranteed 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 $X = (\mathbf{x}_1, ..., \mathbf{x}_n)^\intercal$, $\mathsf{y} = (y_1, ..., y_n)^\intercal$, $\mathbf{x}_i \in \mathbb{R}^p$ 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 (hence $\hat{\beta}(0) = \arg\min_\beta L(\mathsf{y}, X\beta)$), while $lim_{\lambda\to\infty}\hat{\beta}(\lambda) = \mathsf{0}$. In choosing a value for the regularization parameter, we can take a Bayesian approach, and consider the penalty as a log-prior on model space, with λ corresponding to its strength parameter (e.g., variance in the case of the l_2, or Ridge, penalty). A more 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 and the Lasso. Both of these use squared error loss, but they differ in the penalty they impose on the coefficient vector β describing the fitted model:

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

$$\text{Lasso [16]:} \qquad \hat{\beta}(\lambda) = \min_\beta \sum_{i=1}^{n} (y_i - x_i^\intercal \beta)^2 + \lambda\|\beta\|_1. \tag{3}$$

Another example from the statistics literature is the penalized logistic regression model [20] for classification, which is widely used in medical decisions and credit scoring:

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

where the loss is the negative binomial log-likelihood. 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 [19] uses the hinge loss function and the l_2-norm penalty[1]:

$$\hat{\beta}(\lambda) = \min_\beta \sum_{i=1} (1 - y_i x_i^\intercal \beta)_+ + \lambda\|\beta\|_2^2,$$

where $(\cdot)_+$ is the positive part of the argument. Boosting [4] is another popular and highly successful method for iteratively building an additive model from a dictionary of "weak learners". In [14] 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

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

to generate the whole regularized path $\hat{\beta}(\lambda)$, $0 \le \lambda \le \infty$ simply by sequentially calculating the "step sizes" between each two consecutive λ values and the "directions" $\gamma_1, \dots, \gamma_{m-1}$. 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 [3] 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 [11, 13] and for total-variation penalized squared error loss in [10]. We have extended some path-following ideas to versions of the regularized support vector machine [22, 7]. The results in [3] show that the number of linear pieces in the Lasso path is approximately the number of the variables in X, and the complexity of generating the whole solution path, for all values of λ, using the LAR-Lasso algorithm, is approximately equal to one least square calculation on the full sample.

A simple example to illustrate the piecewise linear property can be seen in Figure 1, where we show the Lasso optimal solution paths for a 4-variable synthetic dataset. The plot shows the optimal Lasso solutions $\hat{\beta}(\lambda)$. To follow the tradition of the original Lasso paper [16], we plot $\hat{\beta}(\lambda)$ as a function of $\|\hat{\beta}(\lambda)\|_1$, rather than the regularization parameter λ [2]. Each line represents one coefficient and gives its values at the optimal solution for the range of λ values. We observe that between every two "+" signs the lines are straight, i.e., the coefficient paths are piecewise linear, as a function of λ, and the 1-dimensional curve $\hat{\beta}(\lambda)$ is piecewise linear in $\mathbb{R}^4$.

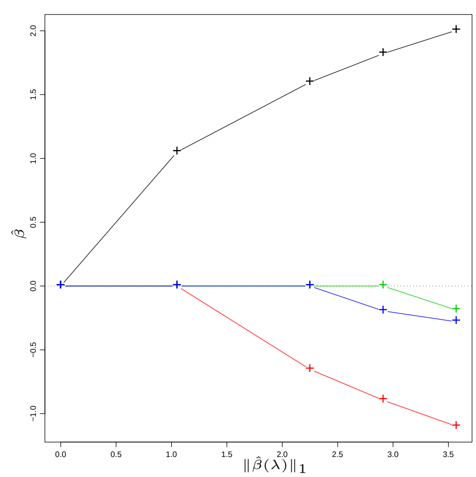

Figure 1: Piecewise linear solution paths for the Lasso on a simple 4-variable example

[2] It is easy to show that there is a one to one correspondence between λ and $\|\hat{\beta}(\lambda)\|_1$ and if $\hat{\beta}(\lambda)$ is piecewise linear in λ, then it is also piecewise linear in $\|\hat{\beta}(\lambda)\|_1$; the opposite is not necessarily true.

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 also discuss some non-LAR type algorithms 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 [10] (Section 4.1), support vector machines (Section 5.1) and others.

   (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 concentrate on robust versions of the lasso for regression and classification (Section 3).

1.1 Illustrative example

Before we delve into the technical details, let us consider an artificial, but realistic, example, to illustrate the importance of all three components of the regularized optimization problem for building good prediction models:

- An appropriate loss function that either matches data error distribution or is robust enough to account for lack of knowledge of this distribution.
- An appropriate regularization scheme.
- Selection of an appropriate value for the regularization parameter.

Our example has $n = 100$ observations and $p = 80$ predictors. All x_{ij} are i.i.d $N(0, 1)$ and the true model is:

$$\begin{aligned} y_i &= 10 \cdot x_{i1} + \epsilon_i \\ \epsilon_i &\overset{iid}{\sim} 0.9 \cdot N(0,1) + 0.1 \cdot N(0,100) \end{aligned}$$

So the true model only depends on the first predictor, and the normality of residuals, implicitly assumed by using squared error loss, is violated. On the loss side, this encourages us to use a robust loss function. As we explain in Section 2.3, sparsity implies the l_1 penalty may be appropriate.

We thus consider two regularized coefficient paths:

- For the l_1-penalized "Huber's loss" with knot at 1:

$$\begin{aligned}\hat{\beta}(\lambda) = \quad & \arg\min_{\beta} \sum_{|y_i-\beta^\mathsf{T}\mathbf{x}_i|\le 1} (y_i - \beta^\mathsf{T}\mathbf{x}_i)^2 + \\ & + \sum_{|y_i-\beta^\mathsf{T}\mathbf{x}_i|>1} 2(|y_i - \beta^\mathsf{T}\mathbf{x}_i| - 0.5) + \lambda \sum_j |\beta_j| \end{aligned} \tag{4}$$

- For the Lasso (3).

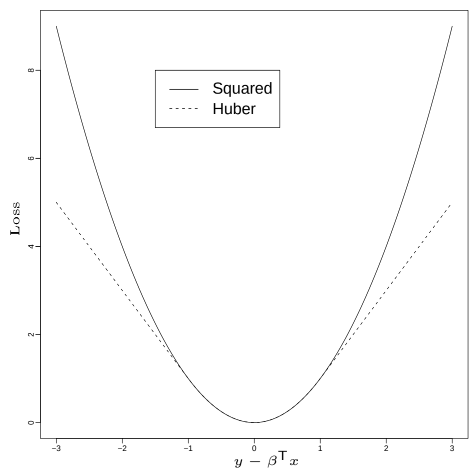

Figure 2: Squared error loss and Huber's loss with knot at 1.

Figure 2 shows the squared error loss and Huber's loss and Figure 3 shows the two regularized paths $\hat{\beta}(\lambda)$ as a function of $\|\hat{\beta}\|_1$. The "true" coefficient $\hat{\beta}_1$ is the solid line, all other coefficient paths are dashed lines. For the "Huberized" loss (on the left) we observe that $\hat{\beta}_1$ is the only non-zero coefficient until it approaches its true value of 10. At this point the regularized model is the correct model. Only then do any of the other coefficients become non-zero, and as $\|\hat{\beta}\|_1$ increases (or equivalently as the regularization parameter λ decreases) the model becomes less and less adequate. The Lasso (on the right) does not do as well — the lack of robustness of the loss function (i.e., the implicit normality assumption of squared error loss) causes the regularized path to miss the true model badly at all λ values.

Figure 4 shows the reducible "future" squared loss as a function of $\|\hat{\beta}\|_1$ for the Huberized-Lasso and Lasso coefficient paths we got. The reducible "future"

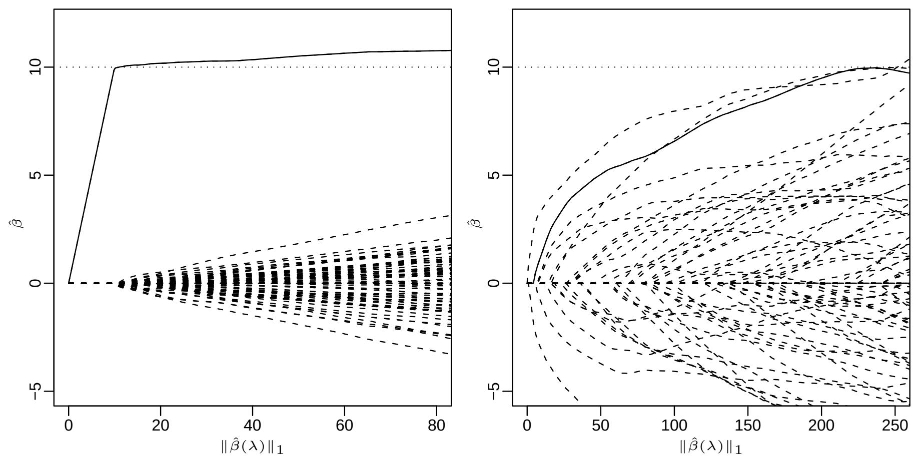

Figure 3: Coefficient paths for Huberized Lasso (left) and Lasso (right) for simulated data example. $\hat{\beta}_1(\lambda)$ is the unbroken line, and the true model is $Ey = 10x_1$

squared loss is defined as $\mathbf{E}_X\|X\hat{\beta} - 10X_1\|_2^2$. Note that the expectation is taken only with respect to "future" X and $\hat{\beta}$ is fixed. The plot re-iterates our observation that the Huberized version practically hits 0 reducible loss at $\|\hat{\beta}\|_1 = 10$, and the model then degrades significantly as $\|\hat{\beta}\|_1$ increases, while the Lasso does not do particularly well at any point in the regularized path in terms of reducible loss.

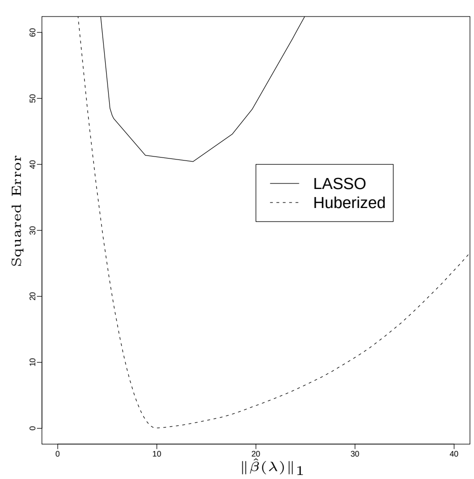

Figure 4: Reducible error for the models along the regularized paths

2 The piecewise linear toolbox

In this section, we first develop a general criterion for piecewise linear solution paths in the case that the loss and penalty are both twice differentiable. We then use it as an intuitive guide to identify families of regularized problems where we can expect piecewise linearity.

2.1 Sufficient condition under twice differentiability

For the coefficient paths to be piecewise linear, we require that $\frac{\partial\hat{\beta}(\lambda)}{\partial\lambda}$ is piecewise constant as a function of λ, or in other words that over ranges of λ values:

$$\lim_{\epsilon\to 0}\frac{\hat{\beta}(\lambda+\epsilon)-\hat{\beta}(\lambda)}{\epsilon} = \text{constant vector} \tag{5}$$

To get a condition, let us start by considering only $\hat{\beta}$ values where L, J are both twice differentiable, with bounded second derivatives in the relevant region. Throughout this section we are going to use the notation $L(\hat{\beta})$ in the obvious way, i.e., we make the dependence on the data X, y implicit, since we are dealing with optimization problems in the coefficients β only, and assuming the data is fixed.

We can now write the normal equations for (1) at λ and $\lambda+\epsilon$:

$$\nabla L(\hat{\beta}(\lambda)) + \lambda\nabla J(\hat{\beta}(\lambda)) = 0 \tag{6}$$

$$\nabla L(\hat{\beta}(\lambda+\epsilon)) + (\lambda+\epsilon)\nabla J(\hat{\beta}(\lambda+\epsilon)) = 0 \tag{7}$$

Developing (7) in Taylor series around $\hat{\beta}(\lambda)$ we get:

$$\begin{aligned}&\nabla L(\hat{\beta}(\lambda)) + \nabla^2 L(\hat{\beta}(\lambda))[\hat{\beta}(\lambda+\epsilon)-\hat{\beta}(\lambda)] + \\ &+\lambda\nabla J(\hat{\beta}(\lambda)) + \lambda\nabla^2 J(\hat{\beta}(\lambda))[\hat{\beta}(\lambda+\epsilon)-\hat{\beta}(\lambda)] + \\ &+\epsilon\nabla J(\hat{\beta}(\lambda)) + O(\epsilon^2) = 0\end{aligned} \tag{8}$$

From (6) and (8) we easily get:

$$\frac{\hat{\beta}(\lambda+\epsilon)-\hat{\beta}(\lambda)}{\epsilon} = -\left[\nabla^2 L\left(\hat{\beta}(\lambda)\right) + \lambda\nabla^2 J\left(\hat{\beta}(\lambda)\right)\right]^{-1}\nabla J\left(\hat{\beta}(\lambda)\right) + O(\epsilon) \tag{9}$$

And from (5) and (9) we get that:

Lemma 1 *For twice differentiable loss and penalty, a sufficient and necessary condition for piecewise linear coefficient paths is that the direction*

$$-\left[\nabla^2 L\left(\hat{\beta}(\lambda)\right) + \lambda\nabla^2 J\left(\hat{\beta}(\lambda)\right)\right]^{-1}\nabla J\left(\hat{\beta}(\lambda)\right) \tag{10}$$

is a piecewise constant vector in $\mathbb{R}^p$ as a function of λ.

When we move to the more realistic domain of non-twice-differentiable loss and penalty functions, the sufficient conditions which Lemma 1 implies (but does not prove) are required to get piecewise linear coefficient paths 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.
- 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.

2.2 Relevant loss functions

Thus, we can consider loss 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 (4). These loss functions are of interest 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:

  $$L(\mathsf{y}, X\beta) = \sum_i (1 - y_i \beta^{\mathsf{T}} \mathbf{x}_i)_+$$

  where $(u)_+ = \max(u, 0)$, and the loss used in quantile regression [9]:

  $$L(\mathsf{y}, X\beta) = \sum_i c_i \ , \ \text{where } c_i = \begin{cases} \tau \cdot (y_i - \beta^{\mathsf{T}} \mathbf{x}_i) & \text{if } y_i - \beta^{\mathsf{T}} \mathbf{x}_i \geq 0 \\ (1 - \tau) \cdot (\beta^{\mathsf{T}} \mathbf{x}_i - y_i) & \text{if } y_i - \beta^{\mathsf{T}} \mathbf{x}_i < 0 \end{cases}$$

As we will see later, both Huber's-type loss functions and l_1-type loss functions do indeed lead to piecewise linear regularized solution paths when l_1 regularization is used. Both also offer similar robustness properties, in that they are linear for large residuals. However the algorithms they yield for following the solution paths differ significantly. The algorithm for differentiable, Huber's-type loss functions is a generalization of the LAR-Lasso algorithm of [3]. The algorithms for non-differentiable loss functions are fundamentally different and we discuss them in Section 5.1.

2.3 l_1 regularization: efficient and effective

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).

There are several motivations for using l_1 regularization. One is computational, since the resulting problems have piecewise linear solution paths. However there are also important statistical reasons why l_1 regularization should be considered, and why it is often preferable to commonly used l_2 regularization.

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 [16]. 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 [14].

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 much 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 [5, 12]. From an inference/interpretation perspective, l_1 regularization gives "smooth" variable selection and more compact models than l_2 regularization [3]. In the case of orthogonal wavelet bases, the soft thresholding method proposed by [2], 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 Robust Lasso for regression and classification

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 home-page:
`www.stat.lsa.umich.edu/~jizhu/huberized`

3.1 "Almost quadratic" loss functions with l_1 penalty

We fix the penalty to be the l_1(Lasso) penalty:

$$J(\beta) = \|\beta\|_1 = \sum_j |\beta_j|, \tag{11}$$

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

$$L(\mathsf{y}, X\beta) = \sum_i l(y_i, \beta^\intercal \mathbf{x}_i) \tag{12}$$

$$l(y, \beta^\intercal x) = a(r)r^2 + b(r)r + c(r) \tag{13}$$

where $r = r(y, \beta^\intercal x) = (y - \beta^\intercal x)$ is the residual for regression and $r = r(y, \beta^\intercal x) = (y\beta^\intercal x)$ 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 Lasso: $l(y, \beta^\intercal x) = (y - \beta^\intercal x)^2$, i.e., $a \equiv 1, b \equiv 0, c \equiv 0$.
- Huber's loss function with *fixed* knot t (define $r = y - \beta^\intercal x$ the residual):

$$l(r) = \begin{cases} r^2 & \text{if } |r| \le t \\ 2t|r| - t^2 & \text{otherwise} \end{cases} \tag{14}$$

- Squared hinge loss for classification (define $r = y\beta^\intercal x$ the margin):

$$l(r) = \begin{cases} (r-1)^2 & \text{if } r \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

  as well as its Huberized counterpart described below. Note that the hinge loss of the support vector machine [19] does not belong to this family as it is not differentiable at $r = 1$.

Theorem 2 *All regularized problems of the form (1) using (11)—(13) (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 lemma 1 as an intuitive argument only, and prove the theorem formally using the Karush-Kuhn-Tucker (KKT) formulation of the optimization problem implied.

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

$$\begin{array}{ll} \min_{\beta^+, \beta^-} & \sum_i l(y_i, (\beta^+ - \beta^-)^\intercal \mathbf{x}_i) + \lambda \sum_j (\beta_j^+ + \beta_j^-) \\ \text{s.t.} & \beta_j^+ \ge 0,\ \beta_j^- \ge 0,\ \ \forall j \end{array}$$

Then the Lagrange dual function of our minimization problem is:

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

And the corresponding KKT optimality conditions:

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

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

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

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 $\mathcal{A} = \{j \in \{1, ..., m\} : \hat{\beta}_j(\lambda) \neq 0\}$ such that:

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

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

So, if we know what the "active" set $\mathcal{A}$ is we can use a similar argument to the one we used for proving lemma 1 in (6)—(10) to argue that, as long as we are in a region where the loss and penalty are both right differentiable, we will have

$$\frac{\partial \hat{\beta}(\lambda)_{\mathcal{A}}}{\partial \lambda} = -(\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}})^{-1} \mathrm{sgn}(\hat{\beta}(\lambda)_{\mathcal{A}}) \tag{16}$$

which is just a version of (10), 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, since the second derivative of the constant and linear parts in (13) is 0. Thus we easily see:

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

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 occur, twice differentiability is violated and hence the direction in (16) will change:

- 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.
- A coefficient in $\mathcal{A}$ hits 0. This is possible, just as in the Lasso [3]. In that case, we reach a non-differentiability point in the penalty and we can see that $\text{sgn}(\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}}) = -\text{sgn}(\beta_{\mathcal{A}})$ would cease to hold if we continue in the same direction. Thus we need to drop the coefficient hitting 0 from $\mathcal{A}$.
- A "generalized residual" $r(y_i, \hat{\beta}(\lambda)^\intercal \mathbf{x}_i)$ hits a non-twice differentiability point (a "knot") in L, for example the "Huberizing" point t in (14), or the hinge point 1 in (15).

And so we conclude that the path $\hat{\beta}(\lambda)$ will be piecewise linear, with the direction given by (16) and direction changes occurring whenever one of the three events above happens. When it happens, we need to update $\mathcal{A}$ to get a legal scenario and re-calculate the direction using (16). ■

Based on this theorem and the arguments in its 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 [3] 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. Notice that the algorithm does not calculate λ explicitly, since the direction calculations and step-length calculations do not require to track the λ values.

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) $d_3 = \min\{d > 0 : r(y_i, (\beta + d\gamma)^\intercal \mathbf{x}_i)$ *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* $a(r(y_{i^*}, \beta^\intercal \mathbf{x}_{i^*}))$ *from (13).*

(i) $C = \sum_i a(r(y_i, \beta^\intercal \mathbf{x}_i))\mathbf{x}_{\mathcal{A},i}\mathbf{x}_{\mathcal{A},i}^\intercal$

(j) $\gamma_{\mathcal{A}} = C^{-1}(-sgn(\beta_{\mathcal{A}}))$

(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 (alas not as straight forward as it is for squared error loss, where centering the predictors allows us to fit the intercept separately). The functions we use for numerical examples in the next sections allow for the inclusion of a non-penalized intercept.

3.2 Computational considerations

What is the computational complexity of running Algorithm 1 on a dataset with n observations and p variables? Let us start by considering the simpler question of the complexity of the LAR-Lasso algorithm, i.e., Algorithm 1 when the loss is squared error loss. As [3] discuss, the complexity of LAR-Lasso is "approximately" that of one least squares calculation, i.e., $O(p^3+np^2) = O(np^2)$ when $n > p$. The qualification refers to the fact that "drop" events in Algorithm 1 — that is, if $d = d_2$ in step $2(b)$ — are not accounted for, and there is no simple theoretical bound on how many of these events can occur. It stands to reason that "on average" these events are rare ($O(1)$, see [3] for discussion) but that cannot be guaranteed.

In the same sense we can say that in the "average" case, the complexity of Algorithm 1 is $O(n^2p)$ when $n > p$. The additional complexity incurred by this algorithm is twofold:

- Figuring out the step size requires considering all possible $O(n)$ "knot crossing" events (step $2(c)$), which does not occur in the Lasso.

- Knot crossing events increase the number of steps of the algorithm. The rarity assumption on drop events made the number of steps of the Lasso $O(p)$. Here we add the assumption that "knot crossing" events occur only $O(n)$ times. This second assumption is very reasonable in the "average" case, since the residuals tend to move monotonically towards 0 as the regularization parameter λ decreases.

Thus overall we assume we have $O(n + p)$ steps, each requiring:

- $O(np + p) = O(np)$ calculations to figure out the step length (steps 2(a)-2(d) of Algorithm 1)

- $O(p^2)$ calculations to calculate the new direction (step 2(j)) using the Sherman-Morrison-Woodbury lemma and the fact that we are either adding or dropping a single variable or a single observation.

which gives us an overall complexity of $O((n+p)np+(n+p)p^2) = O(n^2p)$ when $n > p$, given our (reasonable, but not guaranteed) assumptions on the number of steps.

An interesting case, especially in microarray data analysis, is when $p \gg n$. In that case, we can find *a least squares solution* with a computational cost of only $O(n^3)$ by choosing any n linearly independent predictor variables. Using similar assumptions to the ones above (i.e., the number of "drop" events is $O(1)$, and the number of steps is $O(n)$), we get a computational complexity of $O(pn^2)$ for both the Lasso and Algorithm 1.

In both cases, the complexity of Algorithm 1 is dominated by the step-length calculations, and the direct "least squares" calculations are less costly.

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 choose loss functions which are robust, i.e., linear for large residuals (or margins). We will build on the previous section, in particular 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 (14). It is obviously "almost quadratic" as defined in Section 3.1, and so Theorem 2 and Algorithm 1 apply to its l_1 regularized solution paths.

In Section 1.1 we have illustrated on an artificial example the robustness of the Huberized Lasso against incorrect distributional assumption. In [8], Huber has shown that "Huberizing" has asymptotic optimality properties in protecting against "contamination" of the assumed normal errors. For our (very practical) purpose the main motivating property of the Huberized loss is that it protects us against both extreme "outliers" and long tailed error distributions.

The one open issue is how to select the "knot" t in (14). [8] suggests $t = 1.345\sigma$, 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. Since our Algorithm 1 does not naturally admit changing the knot as it advances, we need to iterate the whole algorithm to facilitate adaptive selection of t. This is certainly possible and practical, however for clarity and brevity we use a fixed-knot approach in our examples.

Prostate cancer dataset

The "prostate cancer" dataset, used in the original Lasso paper [16], 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 [6], 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. This contamination is extreme in the sense that the original responses vary between -0.4 and 5.6, so their range is more than doubled by the contamination, and practically all contaminated observations become outliers.

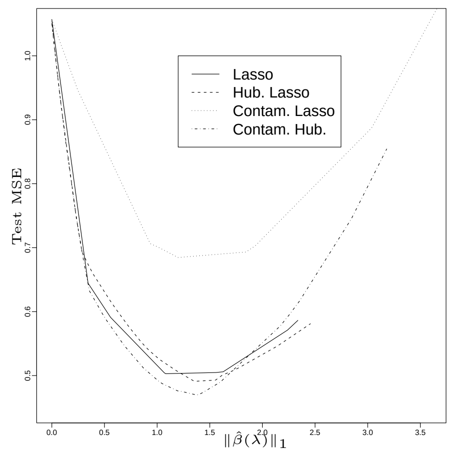

Figure 5: Mean test squared error of the models along the regularized path for the Lasso (full), Huberized Lasso (dashed), Lasso on contaminated training data (dotted) and Huberized Lasso on contaminated data (dash-dotted). We observe that the Huberized Lasso is not affected at all by contamination, while the Lasso performance deteriorates significantly.

Figure 5 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 that on the non-contaminated data, the Lasso (full) 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 non-Huberized Lasso (dotted), on the other hand, deteriorates significantly 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 $O(n + p)$. Figure 6 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.

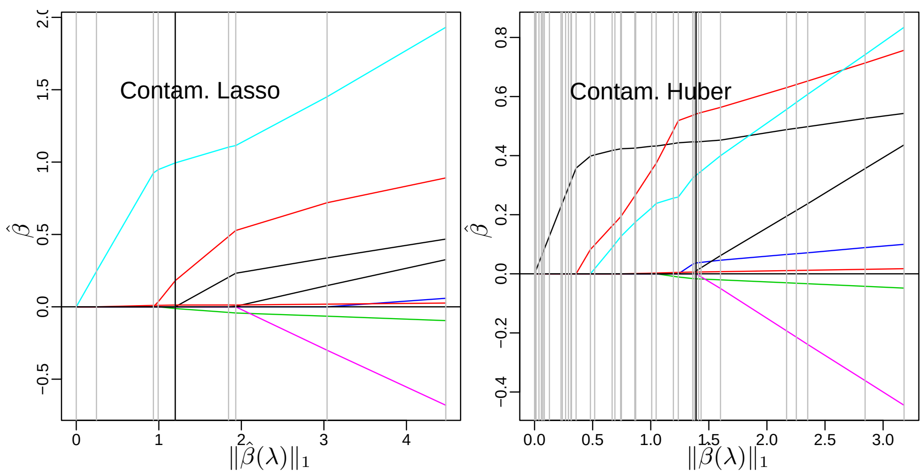

Figure 6: Solution paths of the Lasso (left) and the Huberized Lasso (right) on the contaminated prostate cancer training data. The vertical grey lines correspond to the steps along the solution paths. The vertical solid lines correspond to the models that give the best performances on the test data.

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: $r(y, \beta^\intercal x) = (y\beta^\intercal x)$. This is true of all loss functions typically used for classification:

$$\begin{array}{rl} \text{Logistic regression:} & l(y, \beta^\intercal x) = log(1 + \exp(-y\beta^\intercal x)) \\ \text{Support vector machines:} & l(y, \beta^\intercal x) = (1 - y\beta^\intercal x)_+ \\ \text{Exponential loss (boosting):} & l(y, \beta^\intercal x) = \exp(-y\beta^\intercal x) \end{array}$$

The properties we would like from our classification loss are:

- We would like it to be "almost quadratic", so we can apply Algorithm 1.
- We would like it to be robust, i.e., linear for large absolute value negative margins, so that outliers would have a small effect on the fit. This property is shared by the loss functions used for logistic regression and support vector machines. The squared hinge loss (15) and more so the exponential loss are non-robust in this sense.

This leads us to suggesting for classification the "Huberized squared hinge loss", i.e., (15) "Huberized" at $t < 1$:

$$l(r) = \begin{cases} (1-t)^2 + 2(1-t)(t-r) & \text{if } r \le t \\ (1-r)^2 & \text{if } t < r \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

It is trivial to show that $\text{argmin}_f \mathbf{E}_y l(r(y,f)) = 2\Pr(y=1) - 1$, hence the population minimizer of the Huberized squared hinge loss gives the correct sign for classification. [21] has also considered this loss function for the $t = -1$ case, but more so from a theoretical perspective. It is also worth to note that [15] has proposed a ψ loss function that is as robust as the $0-1$ loss function; but since the ψ loss function is non-convex, we do not consider it in this paper.

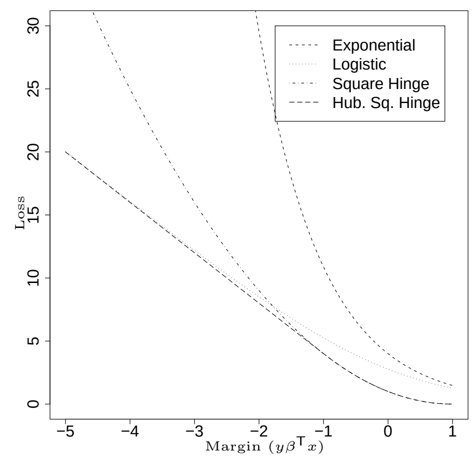

Figure 7: Classification loss functions

Figure 7 compares some of these classification loss functions: the logistic, exponential, squared hinge loss and our suggested loss function (17), with $t = -1$. The exponential and logistic are scaled up by 4 to make comparison easier. We can see the non-robustness of the squared and more so the exponential in the way they "diverge" as the margins become negative.

To illustrate the similarity between our loss (17) and the logistic loss, and their difference from the squared hinge loss (15), consider the following simple simulated example: $x \in \mathbb{R}^2$ 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" iff $x_1 + x_2 > 0$. The data and optimal separator can be seen in Figure 8.

Figure 9 shows the regularized model paths and misclassification rate for this data using the logistic loss (left), the Huberized squared hinge loss (17) (middle) and the squared hinge loss (15) (right), all with l_1 penalty. We observe that the logistic and Huberized regularized model paths are similar and they are both less affected by the outlier than the non-Huberized squared loss. Of course, logistic loss does not allow for efficient calculation of the l_1 regularized path, and so the logistic regularized path in Figure 9 was calculated by solving the problem separately for a large number of values of λ and interpolating between them.

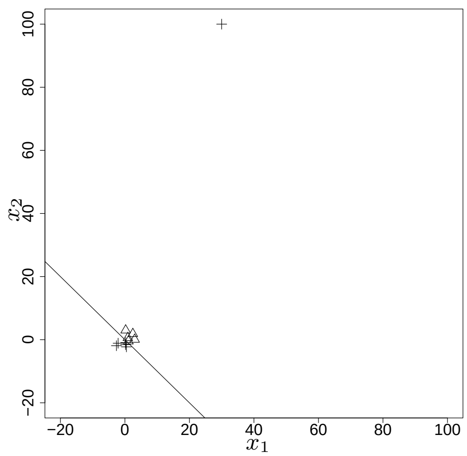

Figure 8: Simulated classification data (for clarity, only a subset of the data is shown)

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. These give a theoretically tractable approach for constructing spline approximations and analyzing their asymptotic properties. 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 $x_1, x_2, \ldots$, then $TV(f)$ is the sum of $TV_{dif}(f)$, calculated over the differentiable set only and the absolute "jumps" at the non-differentiability points. 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. [10] investigate the solutions to total-variation penalized least squares problems. They use total variation of $(k-1)$-th order derivatives:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda TV(f^{(k-1)}) \quad (18)$$

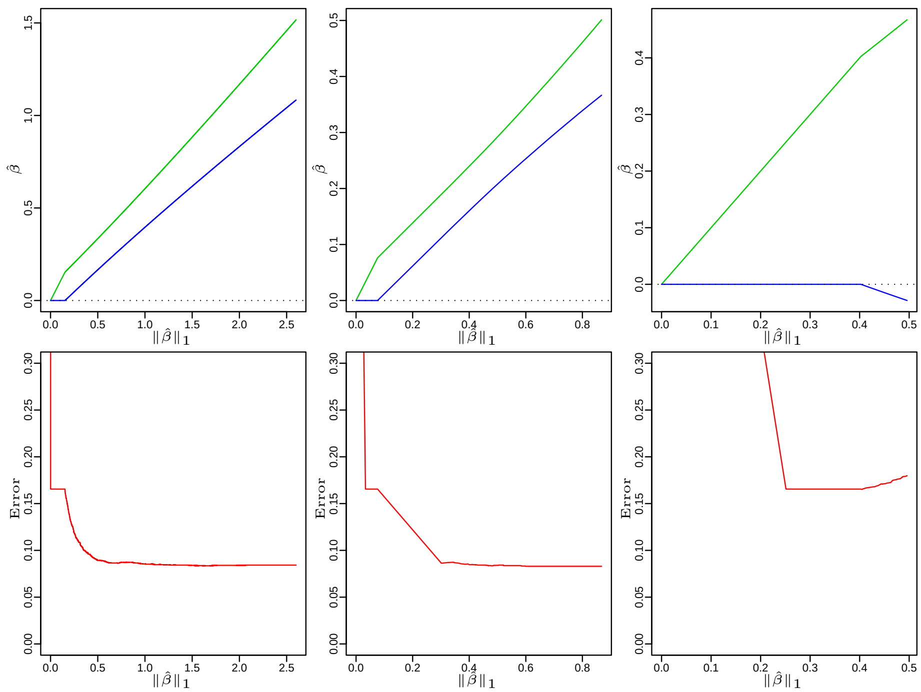

Figure 9: Regularized paths and prediction error for the logistic loss (left), Huberized squared hinge loss (middle) and squared hinge loss (right)

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 $x_i, i = 1, \ldots, n$.

A similar setup is considered by [1], whose taut-string method eventually leads to an optimization problem of the sort considered by [10], with $k = 1$. The resulting solution is a piecewise constant function in x with possible jumps at the data points. [1] also consider the "local squeezing" method, which leads to minimizing the following criterion:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \sum_{i=2}^{n} \lambda_i |f(x_i) - f(x_{i-1})| \tag{19}$$

with data driven λ_i. [1] proposed an algorithm to adaptively find λ_i and the corresponding $\hat{f}$ via some grid search.

Although nonparametric regression has traditionally focused on the estimation of conditional mean functions, nonparametric estimation of conditional quantile functions is often of substantial practical interest. [9] considered the regularized problem:

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{n} \rho_\tau(y_i - f(x_i)) + \lambda \int_0^1 |f''(x)|dx \tag{20}$$

where $x_i \in [0,1]$, $\rho_\tau(r) = r(\tau - \mathbb{I}(r < 0))$ is the check function, and $\tau \in (0,1)$ indicates the quantile of interest. With appropriately chosen $\mathcal{F}$, [9] showed that the solution is a linear spline with knots at the points $x_i, i = 1, \ldots, n$.

In these examples the solutions are polynomial splines of degree 0 or 1 (that is, piecewise constant functions or piecewise linear continuous functions), 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:

$$f(x) = \sum_{j=1}^{h} c_j (x - t_i)_+^{k-1} + q(x) \tag{21}$$

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:

$$TV(f^{(k-1)}) = (k-1)! \cdot \sum_{j=1}^{r} |c_j| \tag{22}$$

If the knots $t_1, ..., t_h$ are fixed in advance (in the examples above, at the data points), then a total variation penalized problem thus reduces to an equivalent 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 [10] for finding the regularized path for any k with a fixed, pre-determined set of candidate knots. [10] 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. If instead of squared loss we employ a quantile regression loss in as in (20), this leads to an equivalent l_1-penalized, l_1-loss problem, of the kind we discuss in Section 5.1.

4.1 Locally Adaptive Regression Splines

We now concentrate on the family of penalized problems defined by Mammen and van de Geer (18). As we mentioned, [10] 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 results above to find the spline solution $\hat{f}_{k,\lambda}$

[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.

exactly for any natural k. The resulting algorithms get more complicated as k increases, but their 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 (22). 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:

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

is the current predictor vector, so that following (21) 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:

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

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 $(x-t_j)_+^{k-1\,\intercal}(\mathbf{y} - \mathbf{x}\hat{\beta}(\lambda_0)))$. 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:

$$\lambda_+(t) = \frac{\mathbf{x}_t^\intercal(\mathbf{y} - Z\hat{\beta}(\lambda_0) - \lambda_0 Z\gamma)}{1 - \mathbf{x}_t^\intercal Z\gamma} \tag{23}$$

$$\lambda_-(t) = \frac{\mathbf{x}_t^\intercal(\mathbf{y} - Z\hat{\beta}(\lambda_0) - \lambda_0 Z\gamma)}{-1 - \mathbf{x}_t^\intercal Z\gamma} \tag{24}$$

then we can write:

$$\lambda(t) = \begin{cases} \max(\lambda_+(t), \lambda_-(t)) & \text{if } \max(\lambda_+(t), \lambda_-(t)) \le \lambda_0 \\ \min(\lambda_+(t), \lambda_-(t)) & \text{if } \max(\lambda_+(t), \lambda_-(t)) > \lambda_0 \end{cases} \tag{25}$$

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:

$$\lambda_{add} = \max_{t \in (0,1) \setminus \{t_1, \ldots, t_{n_{\lambda_0}}\}} \lambda(t) \tag{26}$$

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 (26) requires finding the local extrema of the functions in (25), 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 (26). This is practical for any natural k with numerical programs like Mathematica. 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 (25), and therefore the extrema — and the knot points — are guaranteed to be at the data points. [10] 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 (25), and we can find the extrema within each segment analytically by solving for the roots of the quadratic polynomial we get by differentiating and intersecting them with the segment. These extrema may not correspond to the segment's end points, and so we may have knots that are not at data points.

Note that we do not need to worry about zeros of the denominator in (23,24), because they can never correspond to a value of t attaining the maximum in (26).

Assuming we have the code to solve the maximization problem in (26), 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:

$f(x) = (1, x, \ldots, x^{k-1})^{\mathsf{T}} \beta_{ls}$ is the LS polynomial fit of degree $k-1$

$u = \arg\max_{t \in (0,1)} |(\mathbf{x} - t)_+^{k-1\,\mathsf{T}} (\mathbf{y} - f(\mathbf{x}))|$

$\mathcal{T} = u$

$\lambda_0 = (k-1)! \cdot |(\mathbf{x} - u)_+^{k-1\,\mathsf{T}} (\mathbf{y} - f(\mathbf{x}))|$

$Z = (\mathbf{1}, \mathbf{x}, \ldots, \mathbf{x}^{k-1}, (\mathbf{x} - u)_+^{k-1})$

$\hat{\beta}(\lambda_0) = (\beta_{ls}^{\mathsf{T}}, 0)^{\mathsf{T}}$

$\mathbf{s} = \left(\mathbf{0}_k^{\mathsf{T}}, -\mathrm{sgn}\{(\mathbf{x} - u)_+^{k-1\,\mathsf{T}} (\mathbf{y} - f(\mathbf{x}))\}\right)^{\mathsf{T}}$

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 (23, 24, 25)

   (c) Solve the maximum problem in (26) to get λ_{add}.

   (d) Find λ_{rem}, that is:

$$\lambda_{rem} = \lambda_0 - \arg\min\{l > 0 : \exists j > k \text{ s.t. } \hat{\beta}_j(\lambda_0) + l\gamma_j = 0\}$$

   (e) If $\lambda_{add} > \lambda_{rem}$ add a knot at the point attaining the maximum in (26), update:
   add the new knot to $\mathcal{T}$;
   add a column to Z; and
   add the appropriate sign to $\mathbf{s}$.

   (f) If $\lambda_{add} < \lambda_{rem}$ remove the knot attaining 0 at λ_{rem}:
   remove the knot from $\mathcal{T}$;
   remove the corresponding column from Z; and
   remove the corresponding entry from $\mathbf{s}$.

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

   (h) Return to 2.

Since we can never have more than $(n-k)$ knots in a solution $\hat{f}_{k,\lambda}$ [14], 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). We conjecture that B-splines can be leveraged to reduce complexity as in [10] but have no concrete results on that. 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 [10]).

4.2 Simple data example: $k = 3$

We illustrate our algorithm on a simple data example, with no noise, but where the true knots are not at data points. We generate the data according to the model:

$$y = \begin{cases} 0.125 - x^2 & \text{if } x \leq 0.25 \\ (x-0.5)^2 & \text{if } 0.25 \leq x \leq 0.5 \\ -(x-0.5)^2 & \text{if } 0.5 \leq x \leq 0.75 \\ -0.125 + x^2 & \text{if } 0.75 \leq x \end{cases} \tag{27}$$

that is, a quadratic spline with knots at $0.25, 0.5, 0.75$. We generate uniformly spaced data in $x \in (0, 0.4] \cup [0.6, 1)$. Thus, the knot at 0.5 is far from the closest data points. The resulting sample is shown as the small circles in Figure 10.

We then run our Algorithm 2 with $k = 3$. The first knot selected is indeed at 0.5 exactly, and the algorithm takes six more steps to converge to the correct

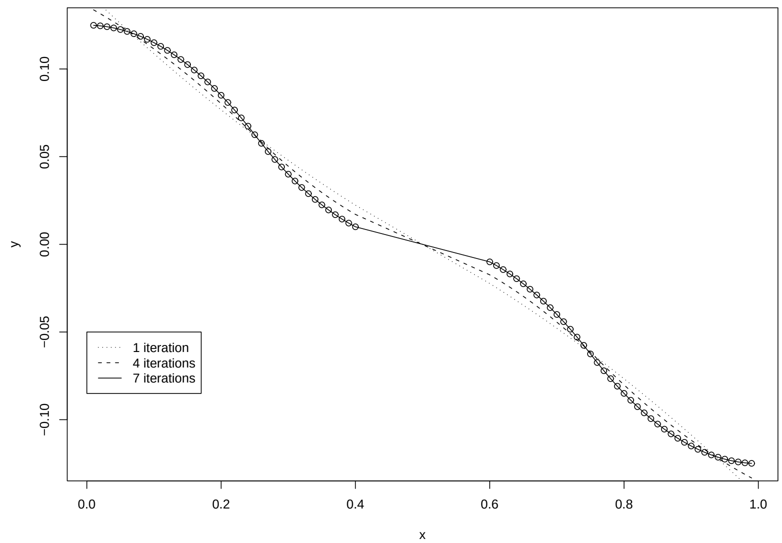

Figure 10: Applying Algorithm 2 to our data example with $k = 3$

set of knots at $0.25, 0.5, 0.75$ and consequently the correct model from which the data was generated. Figure 10 shows the models generated by Algorithm 2 after one, four and finally seven steps.

5 Other piecewise linear models of interest

Our discussion so far has concentrated on generalizations of the LAR-Lasso algorithm. We now survey some problems with piecewise linear solution paths, but whose general solution requires us to build fundamentally different, and more complex, algorithms. We first discuss the use of l_1-type loss functions with l_1 regularization, as in the 1-norm support vector machine and l_1 regularized quantile regression [9]. We then consider the case where multiple penalty functions are required to represent the appropriate regularization.

5.1 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 are quite popular, in particular quantile regression [9]:

$$l(y, \beta^\intercal x) = \begin{cases} \tau \cdot |y - \beta^\intercal x| & \text{if } y - \beta^\intercal x \geq 0 \\ (1-\tau) \cdot |y - \beta^\intercal x| & \text{if } y - \beta^\intercal x < 0 \end{cases} \tag{28}$$

For classification, the hinge loss is of great importance, as it is the loss underlying support vector machines [19]:

$$l(y, \beta^\intercal x) = (1 - y\beta^\intercal x)_+ \tag{29}$$

Here we consider a generalized formulation, which covers both of (28,29), with a loss function of the form:

$$l(r) = \begin{cases} b_1 \cdot |a + r| & \text{if } a + r \geq 0 \\ b_2 \cdot |a + r| & \text{if } a + r < 0 \end{cases} \tag{30}$$

with the generalized "residual" being:

$$r = \begin{cases} y - \beta^\intercal \mathbf{x} & \text{for regression} \\ y \cdot \beta^\intercal \mathbf{x} & \text{for classification} \end{cases} \tag{31}$$

When these loss functions are combined with l_1 penalties (or total variation, in appropriate function classes), the resulting regularized problems 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 (30), 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 $\mathbb{R}^p$*
- *$\forall \lambda \in (\lambda_k, \lambda_{k+1})$ 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 [9] and in the context of 1-norm support vector machines in [22]. The arguments given in these references easily 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 $\lambda_1, ..., \lambda_m$. It is still piecewise linear in the l_1 norm of the solution path, $|\hat{\beta}(\lambda)|$. Figure 11, reproduced from [22], illustrates this for the 1-norm SVM on a simple 5-variable simulated example.

Algorithm 3 gives a general path-following solution to all problems of the form (30) with l_1 penalty. It is interesting to note the fundamental differences between this algorithm and Algorithm 1 for differentiable almost-quadratic loss functions. In Algorithm 1 the direction of the regularized path is solely determined by the points in the "quadratic" pieces of the loss function. Here the direction of the path is solely determined by the points directly at the "elbow" $\mathcal{E}$ (the non-differentiability point of the loss function).

To formulate our algorithm for finding the solution path, we first define:

$$\begin{aligned}
\mathcal{A} &= \{j : \beta_j \neq 0\} \;\; \text{Active set of variables} \\
\mathcal{E} &= \{i : a + r_i = 0\} \;\; \text{Elbow set of observations} \\
\mathcal{L} &= \{i : a + r_i < 0\} \;\; \text{Left of the elbow} \\
\mathcal{R} &= \{i : a + r_i > 0\} \;\; \text{Right of the elbow} \\
\Delta r(y, \Delta f) &= \begin{cases} -\Delta f & \text{for regression} \\ y \cdot \Delta f & \text{for classification} \end{cases} \\
\Delta L(\gamma) &= b_1 \sum_{\mathcal{R}} \Delta r(y_i, \gamma^\intercal \mathbf{x}_i) - b_2 \sum_{\mathcal{L}} \Delta r(y_i, \gamma^\intercal \mathbf{x}_i)
\end{aligned}$$

The idea of the algorithm is to start with a variable that can decrease the loss the fastest per unit of change in this variable's coefficient. The solution moves along that direction and pauses whenever one of the following two events happens:

- A data point hits the non-differentiable point of the loss, i.e. $a + r \neq 0$ becomes $a + r = 0$ for some i. Hence this data point is added to $\mathcal{E}$.
- A fitted coefficient hits the non-differentiable point of the penalty, i.e. β_j changes from non-zero to zero for some j. Hence this variable is removed from $\mathcal{A}$.

Then there are two possible types of actions one can take

- Add a variable into $\mathcal{A}$
- Remove a data point from $\mathcal{E}$

Again, the chosen action is determined by the one that can decrease the loss the fastest, per "unit" of l_1 norm increase in the coefficient vector, and the solution would change along a new direction such that the data points in $\mathcal{E}$ linger on $\mathcal{E}$. The detailed algorithm proceeds as the following:

Algorithm 3 *An algorithm for l_1 loss + l_1 penalty*

1. *Initialize*

$$\begin{aligned}
\beta &= 0 \\
\mathcal{A} &= \arg\max_j |\Delta L(\mathbf{e}_j)| \\
\gamma_j &= -sign(\Delta L(\mathbf{e}_j)), \quad j \in \mathcal{A} \\
\gamma_j &= 0, \quad j \in \mathcal{A}^c \\
\Delta L^* &= \max_j |\Delta L(\mathbf{e}_j)|
\end{aligned}$$

where $\mathbf{e}_j$ is a vector with the jth element equal to 1 and other elements equal to 0.

2. *While* $\Delta L^* \neq 0$

(a) $d_1 = \min\{d > 0 : (\beta + d\gamma)_j = 0, j \in \mathcal{A}\}$

(b) $d_2 = \min\{d > 0 : a + r(y_i, (\beta + d\gamma)^\intercal \mathbf{x}_i) = 0, i = 1, \ldots, n\}$

(c) set $d = \min(d_1, d_2)$

(d) $\beta \leftarrow \beta + d\gamma$

(e) If $d = d_1$*, then remove the variable attaining* 0 *at* d *from* $\mathcal{A}$*; if* $d = d_2$*, add the observation attaining the elbow to* $\mathcal{E}$

(f) For each $j^* \in \mathcal{A}^c$*, solve for* γ

$$\begin{cases} \sum_{j \in \mathcal{A}} \gamma_j x_{ij} + \gamma_{j^*} x_{ij^*} & = & 0 & \text{for } i \in \mathcal{E} \\ \sum_{j \in \mathcal{A}} sign(\beta_j)\gamma_j + |\gamma_{j^*}| & = & 1 & \\ \gamma_j & = & 0 & \text{for } j \notin \mathcal{A} \cup \{j^*\} \end{cases}$$

Compute

$$\Delta L(\gamma) = b_1 \sum_{\mathcal{R}} \Delta r(y_i, \gamma^\intercal \mathbf{x}_i) - b_2 \sum_{\mathcal{L}} \Delta r(y_i, \gamma^\intercal \mathbf{x}_i)$$

(g) For each $i^* \in \mathcal{E}$*, solve for* γ

$$\begin{cases} \sum_{j \in \mathcal{A}} \gamma_j x_{ij} & = & 0 & \text{for } i \in \mathcal{E} \backslash \{i^*\} \\ \sum_{j \in \mathcal{A}} sign(\beta_j)\gamma_j & = & 1 & \\ \gamma_j & = & 0 & \text{for } j \in \mathcal{A}^c \end{cases}$$

Compute

$$\Delta L(\gamma) = b_1 \sum_{\mathcal{R}} \Delta r(y_i, \gamma^\intercal \mathbf{x}_i) - b_2 \sum_{\mathcal{L}} \Delta r(y_i, \gamma^\intercal \mathbf{x}_i)$$

(h) Choose the smallest value of $\Delta L(\gamma)$ *from step 2f and step 2g. Let* $\Delta L^* = \min \Delta L(\gamma)$.

- *If* ΔL^* *corresponds to a* j^* *in step 2f, update* γ *and*

$$\mathcal{A} \leftarrow \mathcal{A} \cup \{j^*\}$$

- *If* ΔL^* *corresponds to a* i^* *in step 2g, update* γ *and*

$$\mathcal{E} \leftarrow \mathcal{E} \backslash \{i^*\}, \quad \mathcal{L} \leftarrow \mathcal{L} \cup \{i^*\} \text{ or } \mathcal{R} \leftarrow \mathcal{R} \cup \{i^*\}$$

- *If* ΔL^* *is non-negative, set* $\Delta L^* = 0$.

(i) Return to 2

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. Similar properties hold for that case, i.e., the solution paths are piecewise-linear and path-following algorithms can be used to generate them. However working out the details and implementing algorithms for this case remains as a future task.

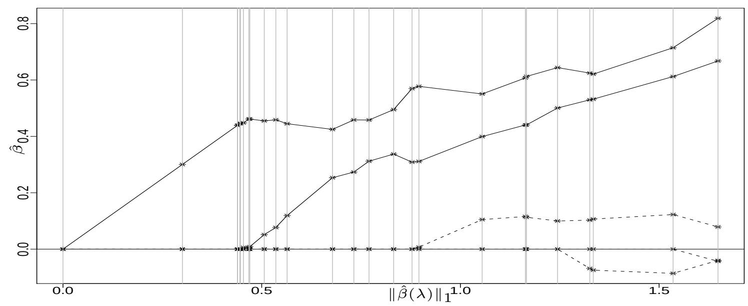

Figure 11: The solution path $\hat{\beta}(\lambda)$ as a function of $\|\hat{\beta}(\lambda)\|_1$ for the 1-norm SVM.

5.2 Multiple penalty problems

Suppose now that we would like to penalize more than one function of the coefficients, i.e., solve:

$$\hat{\beta}(\lambda_1, \lambda_2, ..., \lambda_q) = \min_{\beta} L(\mathsf{y}, X\beta) + \lambda_1 J_1(\beta) + \lambda_2 J_2(\beta) + ... + \lambda_q J_q(\beta)$$

then the theory we have developed tells us that as long as *all* penalties and the loss follow the conditions we have defined for piecewise linearity, the solution $\hat{\beta}(\lambda_1, \lambda_2, ..., \lambda_q)$ will be a *piecewise affine* surface in $\mathbb{R}^p$. In particular, if we limit our interest to a 1-dimensional line in $(\lambda_1, \lambda_2, ..., \lambda_q)$ space, i.e., $\lambda_k = b_k\lambda_1$, we get a piecewise linear solution path. We consider here two examples of multiple penalty problems which are of statistical interest.

A natural application of this idea is for local penalties, where the penalty functions are $J_k(\beta) = |\beta_k|$ and the different λ_k's correspond to different scaling factors for the predictor variables. [1] suggested a problem formulation in this spirit for non-parametric regression, and solved it by an ad-hoc search method over the space spanned by $\lambda_1, ..., \lambda_p$. This is attractive because the l_1 penalty is not scale or rotation invariant, and so the "correct" scaling for the predictor variables may not be known in advance. It can also be viewed as assigning varying "importance" to different variables by penalizing them more or less. General exploration of the p-dimensional surface of solution implied by setting the values of $\lambda_1, ..., \lambda_p$ is a difficult problem, which requires significant generalization of the theory presented here (the area of multi-parametric quadratic programming offers some relevant tools, although no general efficient tools. See for example [18]). However, if we limit our interest to a line in this p-dimensional space, then our algorithms for the relevant loss can be applied almost as-is. For example, assume we wanted to find the solution path for a local-penalty version of the Lasso, with the λ_k's limited to a line:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \|\mathsf{y} - X\beta\|_2^2 + \lambda|\beta_1| + b_2\lambda|\beta_2| + ... + b_p\lambda|\beta_p|$$

with $b_1 = 1, b_2, ..., b_p$ being non-negative constants (independent of λ). Then the LAR-Lasso algorithm, or alternatively our Algorithm 1, can be applied almost as-is, with minimal changes required to account for the "scaling factors" $b_k,\ k = 1, ..., p$. For example, step 2(a) should be changed to:

$$\mathit{2(a)^*}\ d_1 = \min\{d > 0 : \frac{|\nabla L(\beta+d\gamma)_j|}{b_j} = \frac{|\nabla L(\beta+d\gamma)_k|}{b_k},\ j \notin \mathcal{A},\ k \in \mathcal{A}\}$$

We eliminate the rest of the changes for brevity.

An example where a specific two-penalty formulation is natural and of great practical interest in a specific application can be found in our recent paper [17]. The problem considered is that of protein mass spectroscopy, and the predictors correspond to a continuum of "time of flight" sites. Thus the predictors have an order and a distance between them, and there are scientific reasons why we would expect neighboring coefficients to be much more correlated than distant ones. We would like a method which strongly prefers to make neighboring coefficients similar, while still limiting the amount of non-zero coefficients. A natural way to achieve that is to include a separate penalty on magnitude of coefficients and on differences of neighboring coefficients. This gives us the Fused Lasso estimate:

$$\min_\beta \quad \|\mathsf{y} - X\beta\|^2 + \lambda_1 \sum_j |\beta_j| + \lambda_2 \sum_{j>1} |\beta_j - \beta_{j-1}|$$

In [17] we discuss the issues involved in using the piecewise linearity of the solution path to design an efficient algorithm for this problem. Our future project is to formulate a path-following algorithm which would be computationally preferable to solving the problem separately at a number of (λ_1, λ_2) values.

6 Discussion

In this paper we have combined 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 (or noise) model. It does little or nothing to protect us against the uncertainty 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 [8]. Using a robust loss in such a situation is *always* counter productive.

- Regularization deals mainly with the uncertainty about our model structure by limiting the model space. Note, in this context, the equivalence between the "penalized" formulation (1) and a "constrained" formulation:

$$\min_{\beta} L(\mathsf{y}, X\beta) \;\text{ s.t. } J(\beta) \leq s \tag{32}$$

The two formulations are equivalent in the sense that every solution $\hat{\beta}(s)$ to (32) is equal to a solution $\hat{\beta}(\lambda)$ to (1) for some value of λ and vice versa ((1) is in fact just a Lagrange multiplier version of (32)). Thus the main goal of regularization is to make the model estimation problem easier, and match it to the amount of information we have for estimating it, namely our training data.

We can also take a Bayesian approach, in which case the penalized optimization problem (1) corresponds to maximizing a posterior likelihood, if $J(\beta)$ is a log-prior on the model space and L is a log-likelihood (for example, the Lasso maximizes posterior with i.i.d Gaussian error model and i.i.d double-exponential priors on the model coefficients). The more vague Bayesian interpretation for using robust loss would be that it represents a reluctance to commit to a specific error model, electing instead to use a "non-informative" one which can accommodate a range of reasonable error models, in particular ones contaminated by outliers, without suffering significant degradation in performance.

There are many interesting directions in which our work can be extended:

- 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, [3] have offered a limited analysis of the degrees of freedom of models along the Lasso path. This becomes much more challenging, of course, once we stray away from squared error loss.

- Our algorithms typically require considering all possible variables in X in every step, and so are not generally applicable for very high dimensional problems, or ones where the data is embedded into high (even infinite) dimensional spaces. However, we have seen in this paper one example (Section 4.1) where the algorithm we suggest actually finds the path of optimal solutions in an infinite dimensional predictor space, defined by all possible k-th order splines on $[0, 1]$. The interesting question is, what characterizes these high dimensional spaces, where we can design and implement efficient approaches for searching for the variables in our regularized models, instead of enumerating over all possible candidates?

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

[2] Donoho, D., Johnstone, I., Kerkyachairan, G. & Picard, D. (1995). Wavelet shrinkage: asymptopia? (with discussion). *JRSSB* Vol. 57: 201 - 337.

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

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

[5] 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).

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

[7] 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) *Elements of Statistical Learning.* Springer-Verlag, New York.

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

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

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

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

[12] 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.

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

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

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

[16] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, Vol. 58, No. 1.

[17] Tibshrani, R., Saunders, M., Rosset, S., Zhu, J. & Knight, K. (2004). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society, Series B*, to appear.

[18] Tndel, P., Johansen, T. A., and Bemporad, A. (2003). An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, vol. 39, no. 3, pp. 489-497.

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

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

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

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