Gradient Directed Regularization for Linear Regression and Classification

Jerome H. Friedman* Bogdan E. Popescu†

March 29, 2004

Abstract

Regularization in linear modeling is viewed as a two–stage process. First a set of candidate models is defined by a path through the space of joint parameter values, and then a point on this path is chosen to be the final model. Various pathfinding strategies for the first stage of this process are examined, based on the notion of generalized gradient descent. Several of these strategies are seen to produce paths that closely correspond to those induced by commonly used penalization methods. Others give rise to new regularization techniques that are shown to be advantageous in some situations. In all cases, the gradient descent pathfinding paradigm can be readily generalized to include the use of a wide variety of loss criteria, leading to robust methods for regression and classification, as well as to apply user defined constraints on the parameter values, all with highly efficient computational implementations.

Key words and phrases: linear models, regularization, regression, classification, gradient descent, robustness, constrained estimation, lasso, ridge–regression, least–angle regression LARS, partial least squares PLS, linear support vector machines SVM.

1 Introduction

Linear structural models are among the most popular for data fitting. One is given N observations of the form

$$\{y_i, \mathbf{x}_i\}_1^N = \{y_i, x_{i1}, \cdots, x_{in}\}_1^N \tag{1}$$

considered to be a random sample from some joint (population) distribution with probability density $p(\mathbf{x}, y)$. The random variable y is the "outcome" or "response" and $\mathbf{x} = \{x_1, \cdots, x_n\}$ are the predictor variables. These predictors may be the original measured variables and/or selected functions constructed from them as with learning ensembles (Friedman and Popescu 2003). The goal is to estimate the parameters $\mathbf{a} = \{a_0, a_1, \cdots, a_n\}$ of the linear model

$$F(\mathbf{x}; \mathbf{a}) = a_0 + \sum_{j=1}^{n} a_j x_j \tag{2}$$

for predicting y given $\mathbf{x}$, that minimize the expected loss ("risk")

$$R(\mathbf{a}) = E_{\mathbf{x},y} L(y, F(\mathbf{x}; \mathbf{a})) \tag{3}$$

over future predictions $\mathbf{x}, y \backsim p(\mathbf{x}, y)$. Here $L(y, F(\mathbf{x}; \mathbf{a}))$ is a loss criterion that specifies the cost of predicting a response value y by the value of $F(\mathbf{x}; \mathbf{a})$ (2). The optimal parameter values are

*Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94305 (jhf@stanford.edu)

†Department of Statistics, Stanford University, Stanford, CA 94305 (bogdan@stat.stanford.edu)

thereby defined to be

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} E_{\mathbf{x},y} L\left(y,\, a_0 + \sum_{j=1}^{n} a_j x_j\right). \tag{4}$$

Since the population probability density is unknown, a common practice is to substitute an empirical estimate of the expected value in (4) based on the available data (1) yielding

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a}} \frac{1}{N} \sum_{i=1}^{N} L\left(y_i,\, a_0 + \sum_{j=1}^{n} a_j x_{ij}\right) \tag{5}$$

as an estimate for $\mathbf{a}^*$.

1.1 Penalization

It is well known that $\hat{\mathbf{a}}$ (5) often provides poor estimates of $\mathbf{a}^*$; that is $R(\hat{\mathbf{a}}) >> R(\mathbf{a}^*)$. This is especially the case when the sample size N is not large compared to the number of parameters $(n+1)$. This is caused by the high variability of the estimates (5) when based on different random samples drawn from the population distribution. A common remedy is to modify ("regularize") (5) in order to stabilize the estimates by adding a penalty $\lambda \cdot P(\mathbf{a})$ to the empirical risk

$$\hat{\mathbf{a}}(\lambda) = \arg\min_{\mathbf{a}} \frac{1}{N} \sum_{i=1}^{N} L\left(y_i,\, a_0 + \sum_{j=1}^{n} a_j x_{ij}\right) + \lambda \cdot P(\mathbf{a}). \tag{6}$$

For a given set of parameter values $\mathbf{a}$, the penalty function $P(\mathbf{a})$ returns a (deterministic) value that is independent of the particular random sample drawn from the population distribution. It thereby provides a stabilizing influence on the criterion being minimized (6) which in turn stabilizes the corresponding estimates $\hat{\mathbf{a}}(\lambda)$. The penalty multiplier λ (≥ 0) is a "meta"–parameter of the procedure that controls the degree of stabilization; larger values provide increased regularization producing more stable estimates. For $\lambda = 0$ one obtains the least stable estimates (5), whereas for $\lambda = \infty$ the estimates are completely deterministic (provided the minimum of $P(\mathbf{a})$ is unique), being independent of the particular sample (1) drawn from the population.

Commonly employed penalty functions include

$$P_2(\mathbf{a}) = \sum_{j=1}^{n} |\, a_j \,|^2 \tag{7}$$

("ridge regression" Horel and Kannard 1970, "support vector machines" Vapnik 1996), and more recently

$$P_1(\mathbf{a}) = \sum_{j=1}^{n} |\, a_j \,| \tag{8}$$

("lasso" Tibshirani 1996 and "Sure Shrink" Donoho and Johnstone 1993). Both of these functions increasingly penalize (in a different manner) larger absolute values for the coefficients multiplying the corresponding predictor variables $\{x_j\}_1^n$ in the linear model (2). Note that these penalties do not involve the intercept parameter a_0 which is not generally subjected to regularization. Use of the penalty $P_2(\mathbf{a})$ produces solutions that are equivariant under rotations but not to transformations involving scale change. Solutions using the penalty $P_1(\mathbf{a})$ are not equivariant under either rotations or scale change. Often the input variables are standardized $\{var(x_j) = 1\}_1^n$ so that they have equal a priori influence as predictors.

For a given penalty function $P(\mathbf{a})$, the procedure represented by (6) produces a family of estimates, in which each member of the family is indexed by a particular value for the strength

parameter λ. This family thus lies on a one–dimensional path of finite length in the $(n+1)$–dimensional space of all joint parameter values. At one end of this path is the point $\hat{\mathbf{a}}$ (5) corresponding to $\lambda = 0$. For the penalties (7) (8), the other end ($\lambda = \infty$) is given by

$$a_0 = \arg\min_a \sum_{i=1}^{N} L(y_i, a),\ \{a_j = 0\}_1^n. \tag{9}$$

1.2 Model selection

The optimal parameter values $\mathbf{a}^*$ (4) also represent a point in the parameter space. For a given path, the goal is to find a point on that path $\hat{\mathbf{a}}(\lambda^*)$ that is closest to $\mathbf{a}^*$, where distance is characterized by the prediction risk (3)

$$D(\mathbf{a}, \mathbf{a}^*) = R(\mathbf{a}) - R(\mathbf{a}^*). \tag{10}$$

This is a classic model selection problem where one attempts to obtain an estimate $\hat{\lambda}$, for the optimal value of the strength parameter

$$\lambda^* = \arg\min_{0 \le \lambda \le \infty} D(\hat{\mathbf{a}}(\lambda), \mathbf{a}^*). \tag{11}$$

There are a wide variety of model selection procedures depending on the choice of loss criterion (3) and penalty $P(\mathbf{a})$. Among the most general, applicable to any loss and/or penalty, is cross–validation. The data are randomly partitioned into two subsets (learning and test). The path is constructed using only the learning sample. The test sample is then used as an empirical surrogate for the population density $p(\mathbf{x}, y)$ to compute the corresponding (estimated) risks in (3) (10). These estimates are then used in (11) to obtain the estimate $\hat{\lambda}$. Sometimes the risk used in (11) is estimated by averaging over several (K) such partitions ("K–fold" cross–validation).

1.3 Path Selection

Given a model selection procedure, the goal is to construct a path $\hat{\mathbf{a}}(\lambda)$ in parameter space such that some of the points on that path are close to the point $\mathbf{a}^*$ (4) representing the optimal solution. If no points on the path come close to $\mathbf{a}^*$ then no model selection procedure can produce accurate estimates $\hat{\mathbf{a}}(\hat{\lambda})$. Since the path produced by (6) depends on the data, different randomly drawn training data sets T (1) will produce different paths for the same penalty. Thus, the paths are themselves random. Each value of the strength parameter λ produces a joint distribution $p_\lambda(\hat{\mathbf{a}})$ for its parameter values $\hat{\mathbf{a}}(\lambda)$ induced by the data distribution $T \backsim p(\mathbf{x}, y)$. Therefore, the goal becomes one of selecting a procedure (penalty) that produces a distribution of paths whose average closest distance (10) from $\mathbf{a}^*$ (4) is small.

These concepts are most easily illustrated for squared–error loss

$$L(y, F(\mathbf{x}; \mathbf{a})) = (y - F(\mathbf{x}; \mathbf{a}))^2/2. \tag{12}$$

In this case, the average distance (10) between $\hat{\mathbf{a}}(\lambda)$ and $\mathbf{a}^*$ can be expressed as

$$E_T D(\hat{\mathbf{a}}(\lambda), \mathbf{a}^*) = D(\bar{\mathbf{a}}(\lambda), \mathbf{a}^*) + E_{\mathbf{x}} var_T[F(\mathbf{x}; \hat{\mathbf{a}}(\lambda))]. \tag{13}$$

Here $\bar{\mathbf{a}}(\lambda)$ is the mean value of $\hat{\mathbf{a}}$ from $p_\lambda(\hat{\mathbf{a}})$ for the given value of λ. In the second term, the expected value is over the marginal distribution $p(\mathbf{x})$ of the predictor variables, the variance is over the distribution $p_\lambda(\hat{\mathbf{a}})$ induced by the random nature of the training data T, and $F(\mathbf{x}; \mathbf{a})$ is given by (2).

The first term on the right side of (13) ("bias–squared") involves the (deterministic) average path $\bar{\mathbf{a}}(\lambda)$ through the parameter space induced by the chosen penalty $P(\mathbf{a})$. For penalties (7) (8), one end point $\bar{\mathbf{a}}(\infty)$ is the expected value of (9). The other end point $\bar{\mathbf{a}}(0)$ is given by $\mathbf{a}^*$ (4). The interior points ($\infty > \lambda > 0$) are determined by the particular penalty employed.

The second term on the right side of (13) ("variance") reflects the stochastic nature of the path $\hat{\mathbf{a}}(\lambda)$ due to its dependence on the random training data. This variance is smallest for $\lambda = \infty$ and increases as the value of λ decreases owing to the increasing relative influence of the stochastic component in (6). In order to produce paths that on average come close to $\mathbf{a}^*$ one must choose a penalty that causes its average path $\bar{\mathbf{a}}(\lambda)$ to come close to $\mathbf{a}^*$ (first term), at locations for which the variability (second term) is small. This means that $\bar{\mathbf{a}}(\lambda)$ should come close to $\mathbf{a}^*$ as early as possible (larger values of λ) in its trajectory from $\bar{\mathbf{a}}(\infty)$ to $\bar{\mathbf{a}}(0)$.

Since the true coefficient values $\mathbf{a}^*$ are unknown, path (penalty) selection should reflect whatever is known about the nature of $\mathbf{a}^*$. This is illustrated in Fig. 1 for $\hat{a}_1 = 2$, $\hat{a}_2 = 1$ (5) and squared–error loss (12), with $E(\mathbf{x}) = E(y) = 0$, $var(x_1) = var(x_2) = 1$, and $cov(x_1, x_2) = 0.9$. The lower (blue) curve shows the estimated path $(\hat{a}_1(\lambda), \hat{a}_2(\lambda))$, $\infty \geq \lambda \geq 0$, for the lasso penalty (8). The middle (green) curve is the corresponding ridge regression (7) path. (The other paths in Fig. 1 are discussed in Section 2.) For $\lambda > 0$ both paths represent a sequence of shrunken estimates $||\,\hat{\mathbf{a}}(\lambda)\,||_2 < ||\,\hat{\mathbf{a}}(0)\,||_2$, with the degree of shrinkage increasing with the value of λ (more regularization). However the actual paths are quite different. For the lasso, the dispersion (coefficient of variation) of the absolute values of the individual coefficients $\{\hat{a}_1(\lambda), \hat{a}_2(\lambda)\}$ is *larger* than that of $\hat{\mathbf{a}}(0)$ and *increases* as λ increases until $\hat{a}_2(\lambda) = 0$. For the ridge path this dispersion is always *smaller* than that of $\hat{\mathbf{a}}(0)$ and *decreases* with increasing λ.

As the value of λ increases the variance of the data induced paths $\hat{\mathbf{a}}(\lambda)$ about the expected path $\bar{\mathbf{a}}(\lambda)$ decreases. Thus ridge regression tends to produce paths $\hat{\mathbf{a}}(\lambda)$ for which the absolute coefficients of highly correlated variables are shrunk to a common value whereas the lasso produces an opposite effect. Although illustrated here in only two dimensions, these respective characteristics are well known to extend to higher dimensional settings as well (Frank and Friedman 1993, Donoho, *et. al* 1995, Tibshirani 1996). Thus, if one suspected that the components of $\mathbf{a}^*$ had highly disparate (absolute) values the lasso would likely produce paths in the parameter space that come closer to $\mathbf{a}^*$, whereas if the components of $\mathbf{a}^*$ had roughly equal (absolute) values ridge regression might produce closer paths. If one had no prior suspicions concerning the nature of $\mathbf{a}^*$, a model selection technique (such as cross–validation) might be used in an attempt to determine which of the estimated closest points on the respective paths is actually closer to $\mathbf{a}^*$.

2 Gradient directed paths

The upper two paths in Fig. 1 are not constructed from a penalty. They are paths $\hat{\mathbf{a}}(\nu)$ directly constructed in the parameter space in a sequential manner, starting at the point given by (9) and ending at $\hat{\mathbf{a}}$ (5). The lower of these two (red) is the gradient descent (GD) path. Each successive point on this path $\hat{\mathbf{a}}(\nu + \Delta\nu)$ is derived from the previous point $\hat{\mathbf{a}}(\nu)$ by

$$\hat{\mathbf{a}}(\nu + \Delta\nu) = \hat{\mathbf{a}}(\nu) + \Delta\nu \cdot \mathbf{g}(\nu), \tag{14}$$

where $\Delta\nu > 0$ is an infinitesimal increment and $\mathbf{g}(\nu)$ is the negative gradient of the empirical risk evaluated at $\hat{\mathbf{a}}(\nu)$

$$\mathbf{g}(\nu) = -\frac{d}{d\mathbf{a}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i; \mathbf{a})) \Big/_{\mathbf{a}=\hat{\mathbf{a}}(\nu)}, \tag{15}$$

here (Fig. 1) for squared–error loss (12).

The upper (black) curve is constructed from the ordered sequence $(k = 0, 1, \cdots, n)$ of conjugate gradient steps with exact line search (Gill, Murray, and Wright 1981):

$$\hat{\mathbf{a}}_{k+1} = \hat{\mathbf{a}}_k + \rho_k \cdot \mathbf{s}_k \tag{16}$$

with $\mathbf{s}_k$ given by

$$\mathbf{s}_k = \mathbf{g}_k - \frac{\mathbf{g}_k^t \mathbf{g}_k}{\mathbf{g}_{k-1}^t \mathbf{g}_{k-1}} \mathbf{s}_{k-1},$$

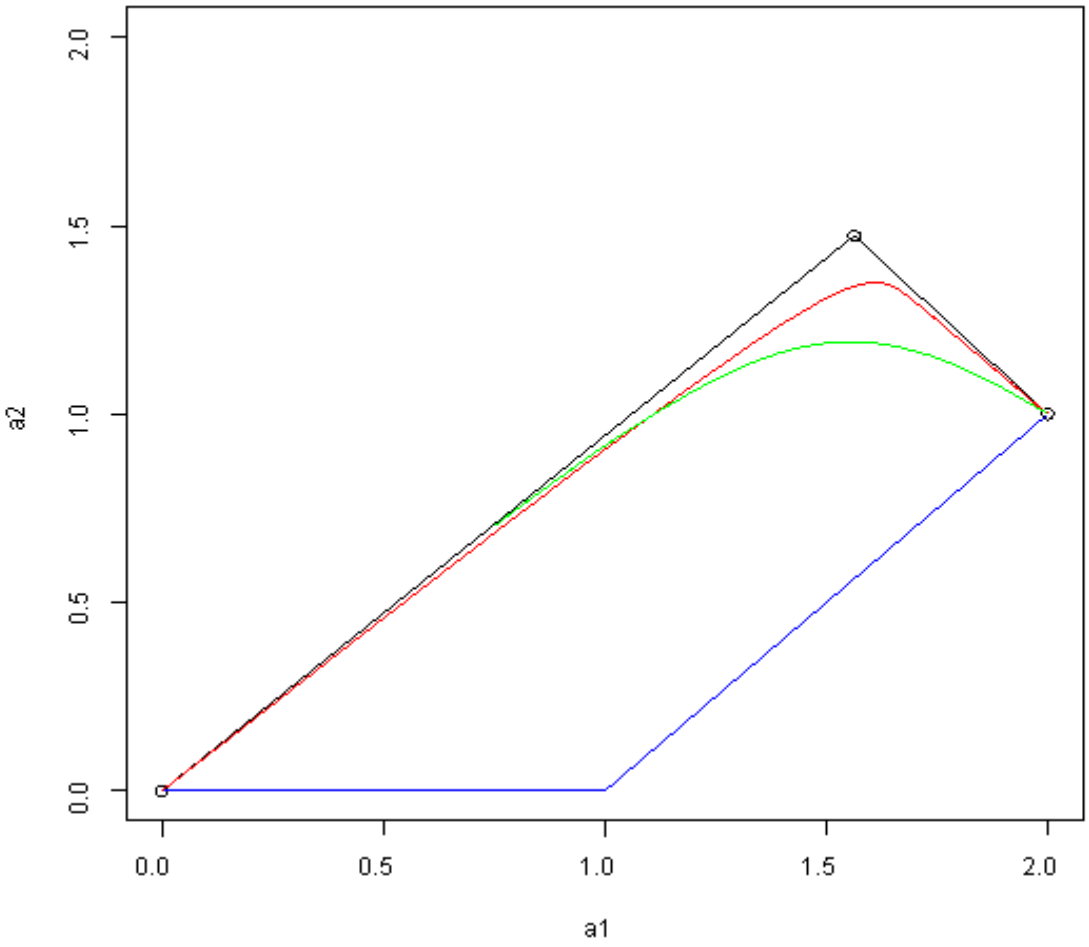

Figure 1: Paths in parameter space generated by the lasso (blue), ridge–regression (green), gradient descent (red), and PLS (black), bottom to top respectively. The top three paths are quite similar and discourage diversity of the coefficient values, whereas the lasso path is quite different, encouraging such diversity.

with $\mathbf{s}_0$ defined to be the zero vector, and $\mathbf{g}_k$ is the negative gradient (15) evaluated at $\mathbf{a} = \hat{\mathbf{a}}_k$. The step size ρ_k is given by a line search along the direction $\mathbf{s}_k$:

$$\rho_k = \arg\min_{\rho} \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i;\, \hat{\mathbf{a}}_k + \rho \cdot \mathbf{s}_k)).$$

For squared error loss (12), this conjugate gradient procedure is called *partial least squares* (PLS) regression (Wold, Ruhe, Wold, and Dunn 1984), and k is referred to as the "number of components" associated with the point $\hat{\mathbf{a}}_k$. Usually with PLS only the points $\{\hat{\mathbf{a}}_k\}_0^n$ are examined for model selection, but it can be easily generalized to the continuous path generated by connecting successive points by straight lines, as in Fig. 1.

From Fig. 1 one sees that the paths produced by ridge regression (RR green), GD (red), and PLS (black) are very similar. Over most of their range they yield nearly identical coefficient values. Only near the upper end point (less regularization) do they diverge a little. The GD path is seen to exaggerate the ridge tendency to shrink the coefficients of correlated variables toward a common absolute value; PLS does this to an even greater degree.

These tendencies persist in higher dimensional settings as well. RR, GD, and PLS are all equivariant under rotations of the predictor variables. It is therefore sufficient to examine their properties in the coordinate system in which the corresponding rotated (and centered) variables $\{z_j\}_1^n$ are orthogonal and uncorrelated ("principal axes"). In this coordinate system (for squared–error loss (12)) the RR, GD, and PLS solutions for the coefficient $\hat{\alpha}_j$ of z_j can all be expressed as

$$\hat{\alpha}_j(\text{RR:GD:PLS}) = f_j(\text{RR:GD:PLS}) \cdot \hat{\alpha}_j(0). \tag{17}$$

Here $\hat{\alpha}_j(0)$ is the unregularized least–squares solution

$$\hat{\alpha}_j(0) = \frac{1}{N} \sum_{i=1}^{N} y_i z_{ij} / v_j^2, \tag{18}$$

where $\{v_j^2\}_1^n$ are the respective variances of the rotated variables $\{z_j\}_1^n$

$$v_j^2 = \frac{1}{N} \sum_{i=1}^{N} z_{ij}^2. \tag{19}$$

The factors f_j(RR:GD:PLS) each scale the respective $\hat{\alpha}_j(0)$ by different amounts depending on the method (RR, GD, PLS).

For RR these scale factors are given by

$$f_j(\text{RR}) = \frac{v_j^2}{v_j^2 + \lambda}, \tag{20}$$

where λ is the regularization strength parameter (6) (7). For GD the corresponding scale factors are given by

$$f_j(\text{GD}) = 1 - (1 - \Delta\nu \cdot v_j^2)^t \tag{21}$$

(Bishop 1995) where $\Delta\nu$ is the step size in (14) and t is the number of steps (of size $\Delta\nu$) along the GD path starting at the point given by (9). Note that both RR and GD are linear estimation methods; their scale factors (20) (21) do not involve the data response values $\{y_i\}_1^N$ so that from (17) (18) their corresponding estimates are linear functions of the responses.

The scale factors $\{f_j(\text{PLS})\}_1^n$ for PLS are not simple functions but they can be computed (see Frank and Friedman 1993). They involve $\{v_j^2\}_1^n$ and the relative values of the least–squares coefficients $\{\hat{\alpha}_j(0)\}_1^n$ (18). Thus, PLS is not strictly a linear method. However, the dependence on the least–squares coefficients is not strong, especially in collinear settings.

Figure 2 illustrates the similarity of the shrinking patterns of RR, GD, and PLS. It shows a comparison of the scale factors f_j(RR) (blue), f_j(GD) (black) and f_j(PLS) (red), in three

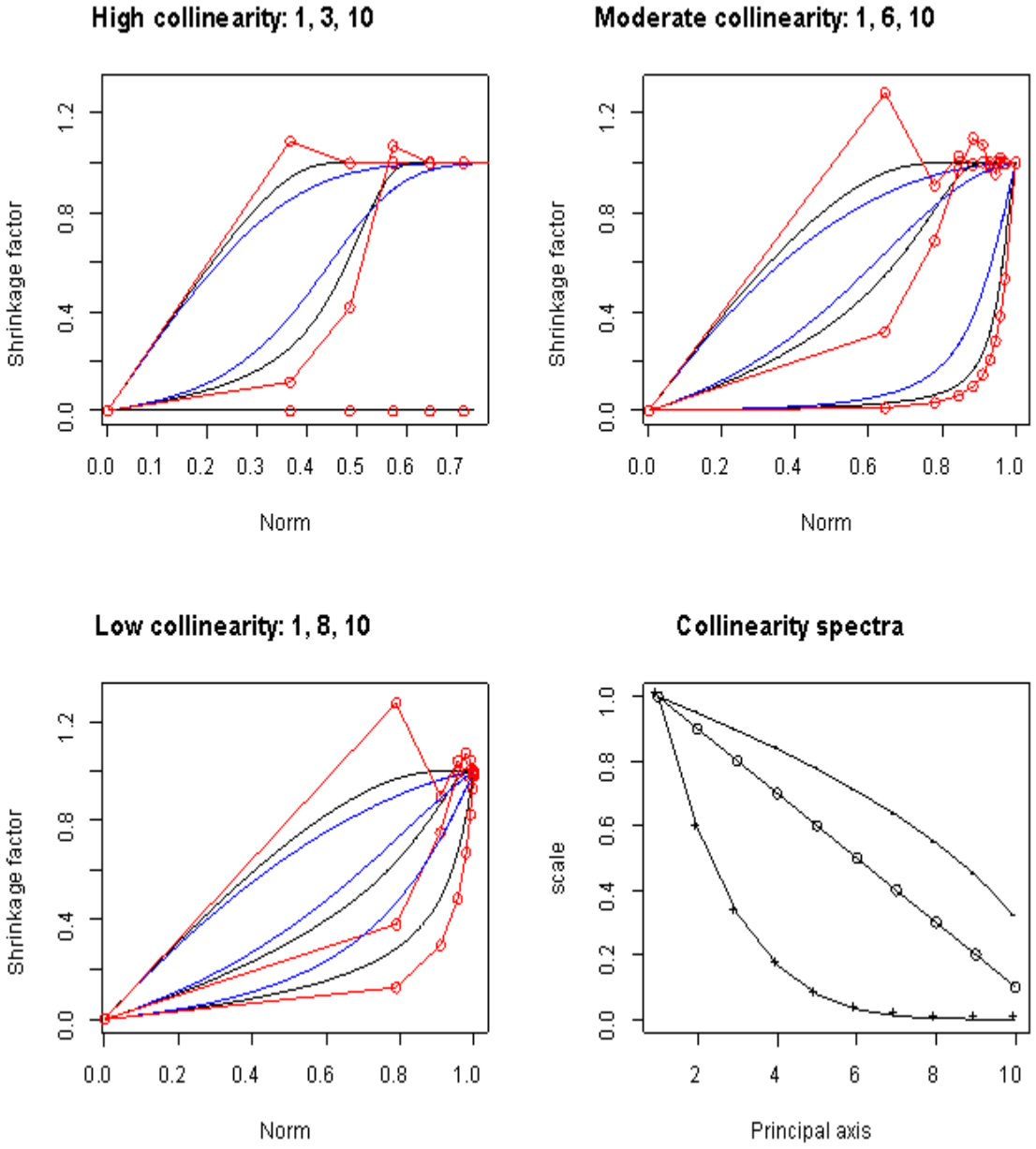

Figure 2: Shrinkage patterns of selected principal component regression coefficients as indicated in the plot titles, for ridge–regression (blue), gradient descent (black), and PLS (red), for high (upper left), moderate (upper right), and low (lower left) collinearity of the original predictor variables. The square–roots of the eigenvalues of the corresponding predictor variable covariance matricies are shown in the lower right panel, bottom to top respectively. The shrinkage patterns for these three methods are all quite similar, especially for higher collinearity.

collinearity settings, for selected variables z_j (numbered in decreasing order of their values of v_j (19)), as indicated in the respective frame titles. The three settings are characterized by high collinearity, $\{v_j = (10 - j)^2\}_1^{10}$, moderate collinearity $\{v_j = (10 - j)\}_1^{10}$, and low collinearity $\{v_j = (10 - j)^{1/2}\}_1^{10}$ of the original variables $\mathbf{x}$. The lower right frame in Fig. 2 displays $\{v_j\}_1^{10}$ for each of these settings. The scale factors are plotted along a common abscissa as a function of their overall norm $|| \hat{\mathbf{a}} ||_2$ of the corresponding coefficient vectors

$$|| \hat{\mathbf{a}} ||_2 = \left[\sum_{j=1}^{n} (f_j \cdot \hat{\alpha}_j(0))^2 \right]^{1/2} .$$

The unregularized solutions were taken to be $\{\hat{\alpha}_j(0) = 1\}_1^n$.

The similarity of RR, GD, and PLS indicated in Fig. 1 is reflected in the results presented in the first three frames of Fig. 2. For increasing levels of regularization (decreasing $|| \hat{\mathbf{a}} ||_2$) they all shrink the unregularized coefficients $\hat{\alpha}_j(0)$ corresponding to smaller values of v_j to a greater extent than those with larger values of v_j. Especially for higher regularization levels, their respective shrinkage patterns are strikingly similar. For almost all levels, PLS provides the most extreme effect with RR being the least extreme. As in Fig. 1, GD lies in between the two extremes. Thus, GD can be considered as a compromise between PLS and RR.

The strong similarity of PLS and RR was reported in Frank and Friedman 1993 where the two methods were shown to have nearly identical performance characteristics. The results presented in Fig. 1 and Fig. 2 suggest that GD, being a compromise between these two methods, should also produce quite similar characteristics. Least angle regression (LARS) (Efron, Hastie, Johnstone, and Tibshirani 2003) is a non gradient (least–squares) method for sequentially inducing paths in parameter space that closely correspond to those produced by the lasso (8). All of these methods therefore divide into two groups at opposite extremes in terms of the characteristics of their corresponding generated paths in parameter space; RR, GD, and PLS produce paths that in a very similar manner, discourage dispersion among the (absolute) coefficient values, whereas LARS and the lasso produce paths that encourage such dispersion, again in a very similar way.

With gradient descent based procedures, the updating step (14) (16) can be restricted to apply only to the coefficients $\{\hat{a}_j(\nu)\}_1^n$; the intercept $\hat{a}_0(\nu)$ is set to the value that minimizes (5), given the current coefficient values $\{\hat{a}_j(\nu)\}_1^n$. This has the effect of removing the intercept from the regularization process in analogy with the penalties (7) (8). For squared–error loss (12) centering the predictor variables $\{mean(x_j) = 0\}_1^n$ automatically produces this effect since $g_0(\nu) = 0$ everywhere along the path. For other loss criteria considered below this centering approximately produces the same effect, so that including the intercept in the updating (14) (16) subjects it to a correspondingly small degree of regularization that tends to decreasingly penalize its absolute difference from the initial value (9) as the path is traversed.

3 Pathfinding by generalized gradient descent

If the optimal parameter values $\mathbf{a}^*$ (4) have highly diverse absolute values then LARS or the lasso (8) would likely provide paths in the parameter space that come close to $\mathbf{a}^*$. At the other extreme, if the components of $\mathbf{a}^*$ all have quite similar absolute values, then paths produced by RR, GD, or PLS would come close to the point $\mathbf{a}^*$. However, for situations in between these extremes in which the optimal coefficient vector $\mathbf{a}^*$ is characterized by moderately diverse/similar values, none of these methods may produce appropriate paths in the parameter space. It may therefore be profitable to consider alternative techniques that provide paths appropriate for such situations.

One way to define a path is to specify a starting and an ending point for the path, and given any point on the path $\hat{\mathbf{a}}(\nu)$ a prescription defining the next point $\hat{\mathbf{a}}(\nu + \Delta\nu)$. For example,

$$\hat{\mathbf{a}}(\nu + \Delta\nu) = \hat{\mathbf{a}}(\nu) + \Delta\nu \cdot \mathbf{h}(\nu). \tag{22}$$

Here $\mathbf{h}(\nu)$ represents a direction in the parameter space tangent to the path at $\hat{\mathbf{a}}(\nu)$. This (22) is the prescription used by GD (14) where $\mathbf{h}(\nu)$ is the negative gradient $\mathbf{g}(\nu)$ (15) and $\Delta\nu$ represents infinitesimal steps. This is also the strategy used by PLS (16) and LARS (Efron *et al* 2003), but where each $\Delta\nu \cdot \mathbf{h}(\nu)$ represents a non infinitesimal increment in a finite sequence of steps.

A very large number of potential paths with highly varying properties can be defined by different starting and ending points, as well as different prescriptions for computing $\Delta\nu \cdot \mathbf{h}(\nu)$ along the path. The paths induced by RR, GD, PLS, LARS and lasso all start at the point defined by (9). This represents the most heavily regularized solution, using the data only to estimate the intercept a_0. The path end points for all of these methods are also the same, namely the completely unregularized solution $\hat{\mathbf{a}}$ (5). Although by no means necessary, it seems reasonable to use these same points (9) (5) to respectively start and end alternatively defined paths as well.

Although they use the same starting and ending points, the paths produced by RR, GD, PLS, LARS and lasso differ in how they define the interior points along their respective paths. However, they all share a *monotonicity* property: starting at (9) each successive point on the path produces lower empirical risk on the training data. That is

$$\hat{R}(\hat{\mathbf{a}}(\nu + \Delta\nu)) < \hat{R}(\hat{\mathbf{a}}(\nu)), \tag{23}$$

where

$$\hat{R}(\mathbf{a}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i; \mathbf{a})) \tag{24}$$

and $F(\mathbf{x}; \mathbf{a})$ is given by (2). This means that for all of these methods the corresponding tangent vector $\mathbf{h}(\nu)$ (22) at each step represents a *descent* direction; that is it projects positively on the negative gradient (15), $\mathbf{h}^t(\nu)\mathbf{g}(\nu) > 0$. Again, although not necessary, it seems reasonable to impose this restriction on the alternatively defined paths to be considered below. Namely we will consider "generalized" gradient descent paths generated by (22) using a small ($\backsim$ infinitesimal) constant value for $\Delta\nu$, and with tangent direction

$$\mathbf{h}(\nu) = \{h_j(\nu)\}_0^n = \{f_j(\nu) \cdot g_j(\nu)\}_0^n, \tag{25}$$

where $\{g_j(\nu)\}_0^n$ are the components of the negative gradient (15), and $\{f_j(\nu) \geq 0\}_0^n$ are non negative factors scaling their respective gradient components. Although not necessary, the form (25) with the non negative constraint on the factors, is sufficient to insure a descent direction at every step.

3.1 Threshold gradient descent

Setting all of the factors $\{f_j(\nu)\}_0^n$ in (25) to the same positive value at each step yields the GD strategy (14) (15). As shown in Section 2 this produces paths in parameter space on which the coefficients $\{\hat{a}_j(\nu)\}_1^n$ tend to have similar absolute values. One way to direct the path towards parameter points with more diverse component values is to increase the diversity of the factor values used in (25). In particular, we take

$$f_j(\nu) = I\left[|\, g_j(\nu)\,| \geq \tau \cdot \max_{0 \leq k \leq n} |\, g_k(\nu)\,|\right] \tag{26}$$

where $I[\cdot]$ is an indicator function of the truth of its argument, and $0 \leq \tau \leq 1$ is a threshold parameter that regulates the diversity of the values of $\{f_j(\nu)\}_0^n$; larger values of τ lead to more diversity.

Setting $\tau = 0$ produces the standard GD procedure that encourages equal coefficient values for points $\hat{\mathbf{a}}(\nu)$ on its path. At the opposite extreme, setting $\tau = 1$ encourages the most diversity among the parameter values by causing only the single component $\hat{a}_{j^*}(\nu)$ whose absolute derivative is largest,

$$j^*(\nu) = \arg\max_{0 \leq j \leq n} |\, g_j(\nu)\,| \tag{27}$$

to be incremented (22) (25) (26) at each step. For squared–error loss (12) this is called the "incremental" forward stagewise strategy in Hastie, Tibshirani, and Friedman 2001 (see also Efron *et al* 2003) where it is shown that the resulting path very closely corresponds to that produced by the lasso (8) (and LARS). As discussed in these references and in Section 2, such paths emphasize highly diverse parameter values. Values of τ in between these two extremes ($0 < \tau < 1$) create paths that involve more diverse coefficient absolute values than PLS, GD, or RR but less than LARS or the lasso. Smaller values of τ create paths closer to the former whereas larger values produce paths closer to LARS and the lasso.

4 Illustration: latent variable model

We illustrate these concepts in the context of a specific problem. Data are simulated from a model (Frank and Friedman 1993) in which a small number L of independent unknown intrinsic processes are responsible for the systematic variation of both the response y and the predictors $\mathbf{x}$. These processes are represented by unobserved latent variables $\{l_k\}_1^L$, and the model is

$$y = \sum_{k=1}^{L} a_k \, l_k + \varepsilon, \tag{28}$$

where $\{a_k\}_1^L$ are coefficients relating the respective latent variables to y, and ε is a random error representing the variation in y not related to the latent variables. The predictor variables $\mathbf{x} = \{x_j\}_1^n$ are similarly related to the intrinsic latent variables by

$$x_j = \sum_{k=1}^{L} b_{jk} \, l_k + \delta_j, \quad 1 \leq j \leq n. \tag{29}$$

In this example each random latent variable has a standard normal distribution. The noise terms ε, $\{\delta_j\}_1^n$ are also normally distributed with the variance of each set to produce a two to one signal to noise ratio: $var(\varepsilon) = var(y)/5$ in (28), and $\{var(\delta_j) = var(x_j)/5\}_1^n$ in (29). The coefficient values $\{a_k\}_1^L$ (28) are taken to be

$$a_k = L - k + 1, \quad 1 \leq k \leq L \tag{30}$$

to produce some dispersion in the relative influences of the respective latent variables on the response y.

A subset of size n_s of the n predictor variables were partitioned into L groups of the same size (n_s/L), and the respective coefficients in (29) for those variables ($1 \leq j \leq n_s$) were taken to be

$$b_{jk} = \begin{cases} 1 & \text{if } j \in k\text{th group} \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

That is, each predictor variable in the same group is an independent noisy measurement of the (same) latent variable associated with that group. Thus, the predictor variable covariance matrix has a block structure with variables in the same group being highly correlated with each other, while between group correlations are small. The remaining $n - n_s$ predictor variables have zero valued coefficients for all latent variables, $\{b_{jk} = 0\}_{j=n_s+1}^n$, ($1 \leq k \leq L$). Thus these are pure noise variables unrelated to the intrinsic processes, and thereby unrelated to the response.

Samples of $N = 150$ observations were generated according to the above prescription with $L = 5$ latent variables and $n_s = 100$ active predictor variables (20 in each latent group). Three situations are considered in terms of the total number of predictor variables: $n = 100$, $n = 1000$, $n = 10000$. The first case involves only the $n_s = 100$ active variables; there are no pure noise variables. The second and third cases involve respectively $n - n_s = 900$ and 9900 pure noise variables unrelated to the response y.

Performance is evaluated in terms of the scaled average absolute error

$$sae = \frac{E_{\mathbf{x}} \, |F(\mathbf{x}; \mathbf{a}^*) - F(\mathbf{x}; \hat{\mathbf{a}}) \,|}{E_{\mathbf{x}} \, |F(\mathbf{x}; \mathbf{a}^*) - median_{\mathbf{x}} F(\mathbf{x}; \mathbf{a}^*) \,|} \tag{32}$$

where $F(\mathbf{x}; \mathbf{a}^*)$ is the optimal model (2) based on the true parameter values derived from (30) (31), and $F(\mathbf{x}; \hat{\mathbf{a}})$ represents the corresponding function using the estimated parameter values $\hat{\mathbf{a}}$. All estimates were obtained from (15) (22) (25) (26) using squared error loss (12). Three–fold cross validation was used for model selection (Section 1.2) to estimate the optimal point $\hat{\mathbf{a}}(\nu^*)$ (10) (11) along each estimated path. The criterion (32) was evaluated using an independently generated data sample of size 10000. A total of 100 data sets of size $N = 150$ were generated for each situation ($n = 100, 1000, 10000$) and the resulting average of (32) over these 100 trials is reported.

4.1 Threshold parameter value

Figure 3 shows the resulting average scaled absolute error (32) for the three situations ($n = 100, 1000, 10000$), each for eleven values of the gradient threshold parameter τ (26) in the range $\tau \in [0, 1]$. The lower (green) points (connected by straight lines) correspond to $n = 100$ predictor variables, the middle (blue) points correspond to $n = 1000$, and the upper (red) to $n = 10000$. The points at the left represented by asterisks (nearly hidden by the $\tau = 0$ points) are the corresponding results for PLS.

From Fig. 3 one sees that the best threshold parameter value is different for these three situations. For $n = 100$ (no pure noise variables) $\tau = 0$ appears optimal. For $n = 1000$ (900 pure noise variables) $\tau \simeq 0.5$ produces the smallest error, whereas for $n = 10000$ (9900 pure noise variables) the average error is minimized between $0.6 < \tau < 0.7$. Also, PLS and GD ($\tau = 0$) are seen to produce nearly identical results in accordance with the discussion in Section 2. Increasing the number of pure noise variables severely degrades performance for smaller values of τ (and PLS), whereas using larger values of τ produces more tolerance to many such irrelevant variables.

Figure 4 shows plots similar to Fig. 3 , but where the error values for each situation are divided by their corresponding minimizing error over $\tau \in [0, 1]$. For $n = 100$, one sees that values $\tau \lesssim 0.4$ all provide comparable results to the optimal value $\tau = 0$, but for larger values performance degrades rapidly approaching 60% increased error for $\tau = 1$. For $n = 1000$, resulting error is more sensitive to the value of τ, increasing in both directions from $\tau = 0.5$ to about a 50% increase at both extremes ($\tau = 0, 1$). For $n = 10000$, using $\tau = 1$ results in an increase of 30% over the error at $\tau = 0.6$ and a 260% increase at $\tau = 0$.

Figure 5 shows the first 200 coefficient estimates for the first data set (out of 100) generated with $n = 10000$ predictor variables. The first 100 (blue) are the coefficients of the $n_s = 100$ active variables, the next 100 (red) are those for the first 100 pure noise variables. (The distribution of the coefficient values for the other 9800 pure noise variables is similar the the first 100 shown here.) Three sets of estimates are displayed corresponding to gradient threshold values $\tau \in \{0.0, 0.6, 1.0\}$. Note that the vertical scales of these plots are very different, each increasing roughly by a factor of five from the previous one (top to bottom).

As seen in Fig. 5 the dispersion in absolute values of the coefficient estimates increases (fairly dramatically) with increasing values of the gradient threshold. For $\tau = 0$, the coefficients of the active variables are heavily shrunk and most of the pure noise variables have absolute coefficient values comparable in size to the active ones. The ratio of the absolute average coefficient value for the 9900 pure noise variables to that of the active ones is 0.254. The optimal threshold value for this situation, $\tau = 0.6$, produces larger coefficients for the active variables with somewhat larger dispersion, and estimates most of the pure noise variable coefficients to be zero. The corresponding ratio of average pure noise to active variable absolute coefficient values is 0.015. For $\tau = 1$, the coefficient estimates of the 100 active variables have highly diverse values with only 29 of them being non zero. As with $\tau = 0.6$, the vast majority of pure noise variables have estimated coefficient values of zero, the ratio of average pure noise to active absolute coefficient values being 0.014.

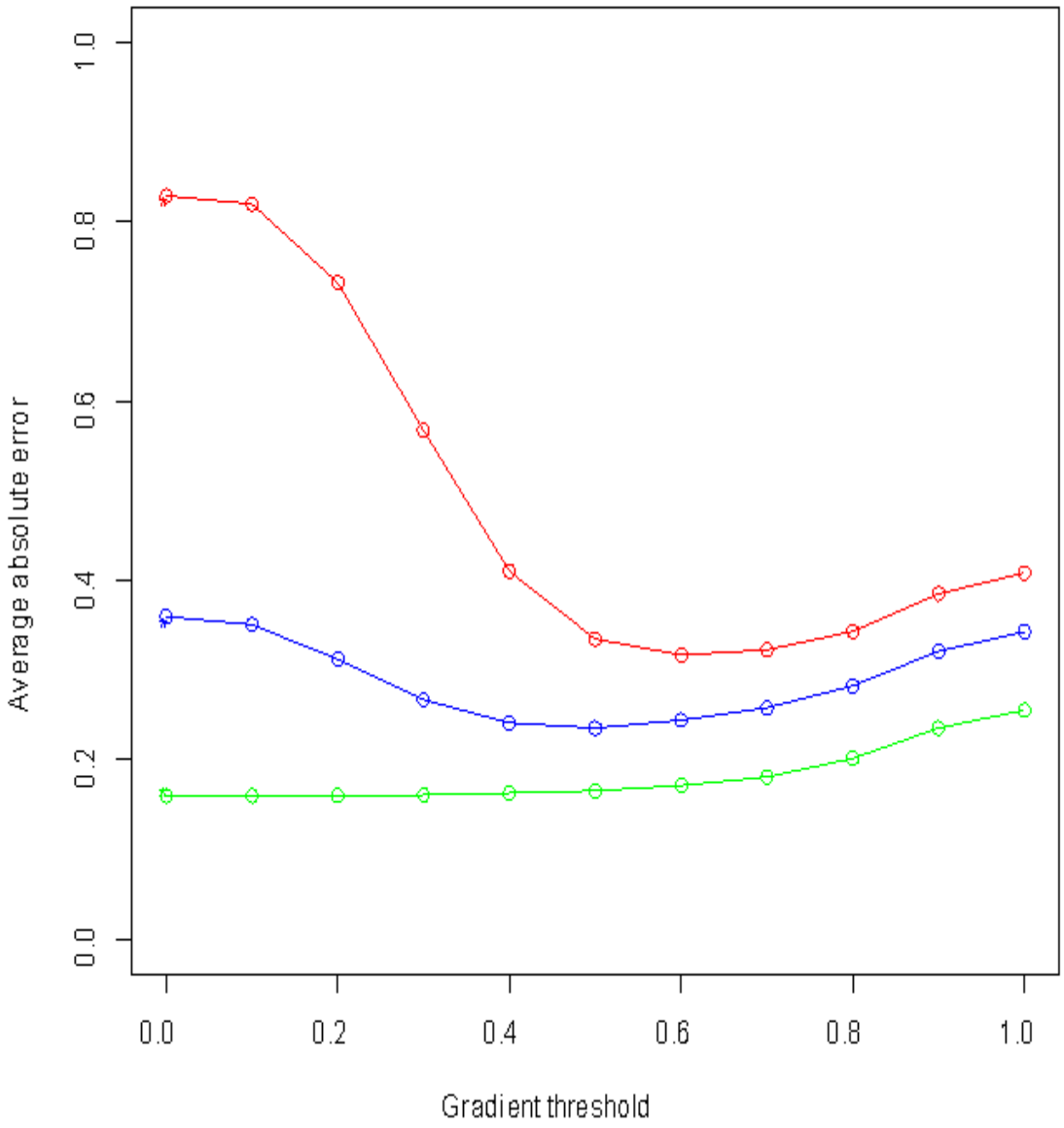

Figure 3: Expected scaled absolute error as a function of gradient threshold τ for the latent variable regression problem, for $n = 100$ (lower green), 1000 (middle blue), and 10000 (upper red) total predictor variables. The asterisks at the left nearly hidden by the $\tau = 0$ points, represent the corresponding results for PLS. The optimal threshold value is seen here to increase with the number of (irrelevant) predictor variables. Larger values of τ provide increased resistance to more irrelevant variables.

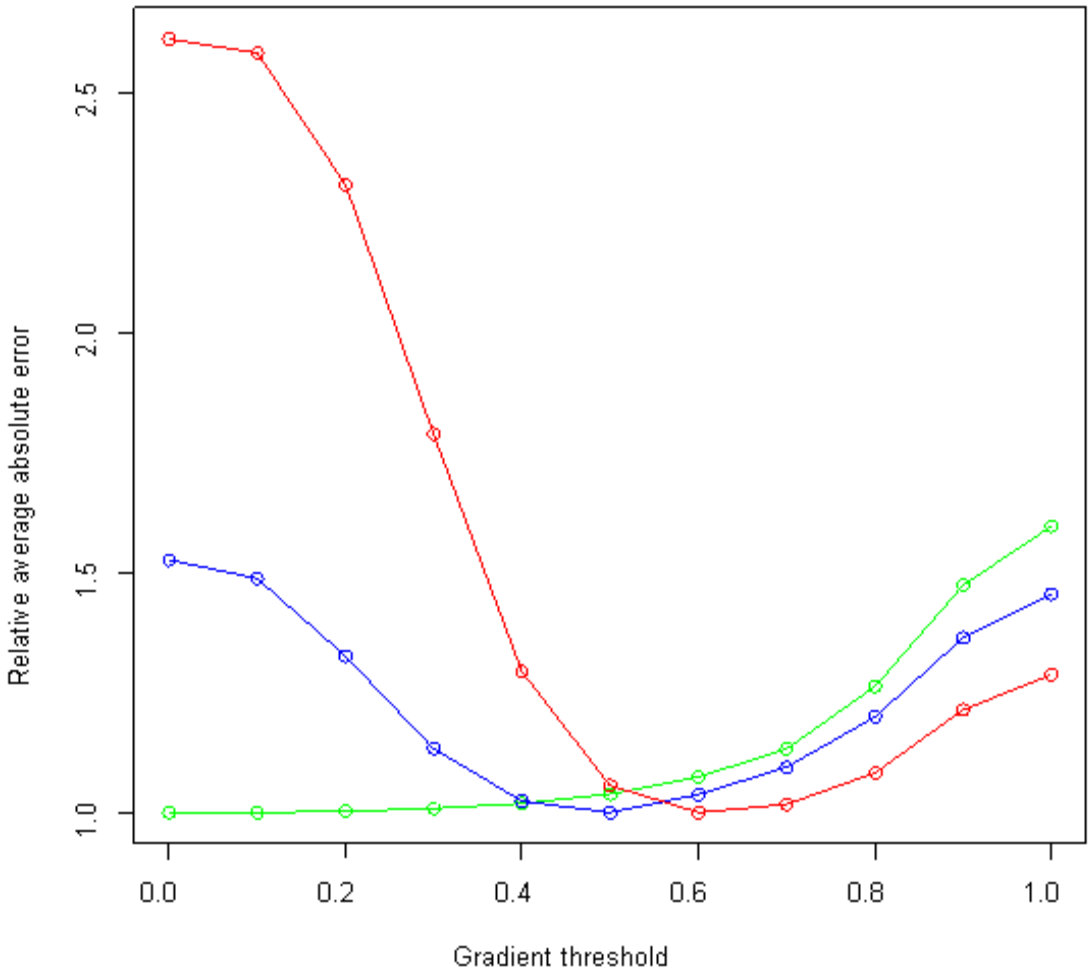

Figure 4: Results presented in Fig. 3, with each expected scaled absolute error divided by the corresponding minimizing value over $\tau \in [0, 1]$. For $n = 100$ (green) using $\tau = 1$ produces a 60% error increase over the optimal $\tau = 0$ value. For $n = 1000$ (blue) there is a 50% increase at both extremes $\tau = 0, 1$ from its optimal value at $\tau = 0.5$, while for $n = 10000$ (red) there is a 30% increase at $\tau = 1$ and 260% increase at $\tau = 0$ over the optimal value at $\tau = 0.6$.

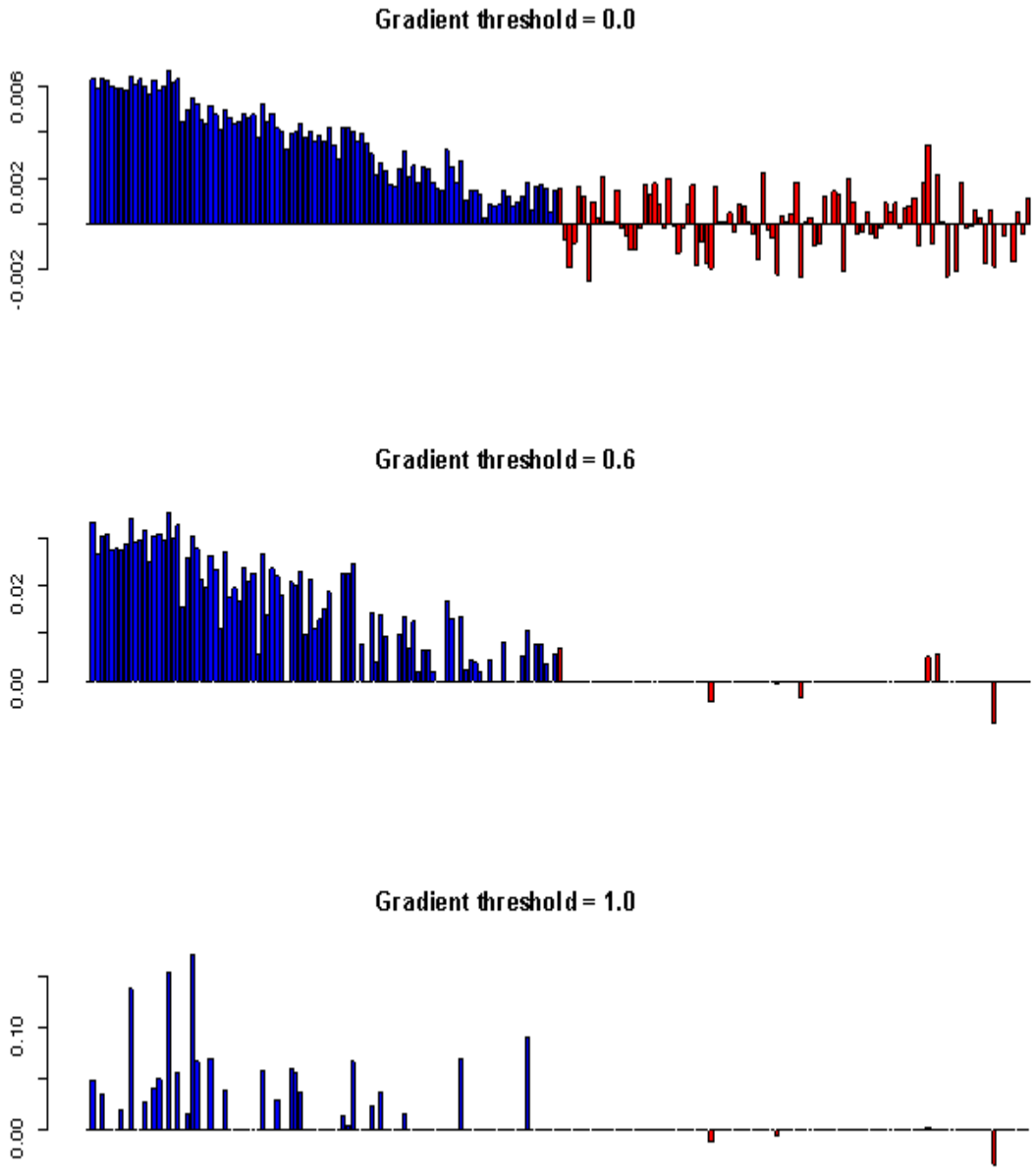

Figure 5: First 200 estimated coefficient values for the latent variable regression problem with $n = 10000$ predictor variables, for gradient threshold values $\tau = 0$ (upper), $\tau = 0.6$ (middle), and $\tau = 1$ (lower) panels. The first 100 variables (blue) have non zero population optimal values, decreasing from left to right, whereas the other 100 (red) along with the remaining 9800 (not shown) have optimal values of zero. Note the different vertical scales. Higher threshold values are seen to produce more diversity in the estimates of the absolute coefficient values.

The coefficient estimates for the three values of gradient threshold τ shown in Fig. 5 reflect the properties discussed in Section 3. Small values ($\tau \preceq 0$) tend to produce coefficient estimates that have similar absolute values, whereas large values ($\tau \succeq 1$) tend to produce sparse highly diverse estimates. Moderate values of τ produce estimates in between these extremes. The results in Figs. 3 and 4 show that there are situations for which such moderate gradient threshold values can produce results superior to either extreme. This suggests (Sections 2 and 3) that for some problems, threshold gradient descent with intermediate threshold parameter values ($0 < \tau < 1$) might give rise to more accurate predictive models than RR, GD, PLS at the one extreme, or LARS and the lasso at the other.

4.2 Total model selection

In the simulated examples above the true underlying parameter values $\mathbf{a}^*$ (4) were known so that performance for various gradient threshold values could be computed. Model selection (here three fold cross–validation) was used only to estimate the optimal predicting point along each of the respective paths. In actual practice, the true underlying parameter values not known, and it is the purpose of the exercise to attempt to estimate them. Thus, model selection must be used to jointly estimate a good value for the gradient threshold τ as well as a corresponding predicting point along its path. This introduces additional variability into the problem that degrades the quality of the parameter estimates from that obtained using the corresponding (unknown) optimal value, thereby reducing (or possibly eliminating) the advantage of gradient descent with adjustable threshold.

Figure 6 shows the distributions (boxplots) of the estimated optimal threshold values $\{\hat{\tau}_i\}_1^{100}$, using three fold cross–validation to jointly estimate them, along with their corresponding path optimal predicting points, over the 100 data sets for each of the three situations ($n = 100, 1000, 10000$) displayed in Figs. 3 and 4. For $n = 100$, this distribution is rather broad with a median of $\hat{\tau} = 0.3$. For $n = 1000, 10000$ the distributions are narrower with medians of $\hat{\tau} = 0.6, 0.7$ respectively. Comparing these distributions with the results shown in Figs. 3 and 4, one sees that although these estimates are variable, they do tend to predominately lie preferentially close to their (here known) optimal values.

Figure 7 repeats Fig. 3, but with the addition of horizontal lines indicating the average (over the corresponding 100 data sets) of the scaled absolute error (32), using the estimated threshold ($0 \leq \tau \leq 1$) for each data set, in each of the three situations: $n = 100$ (green), $n = 1000$ (blue), and $n = 10000$ (red). In all situations the uncertainty associated with estimating the threshold parameter is seen to increase average error over that of using the optimal threshold value by around 5%. Thus, at least for this illustration, the increased error associated with total model selection is rather modest and the advantage of gradient descent with adjustable threshold is maintained.

5 Fast algorithms

Implementation of threshold gradient descent from (15) (22) (25) (26) is straight forward for any differentiable loss criterion $L(y, F(\mathbf{x}; \mathbf{a}))$. However such direct implementation can require excessive computation rendering it impractical for many large problems. The computation involved scales as $n \cdot N \cdot M$ where n is the number of predictor variables, N is the data sample size, and M is the number of steps (22) used to construct the path; typically $M \backsim 10000, 20000$. For many problems encountered in data mining ($N \backsim 10^4 - 10^6, n \backsim 10 - 10^3$) direct implementation can be too computationally demanding. In this section we describe threshold gradient descent algorithms for several useful loss criteria that make use of fast updating strategies to dramatically reduce computation in these situations.

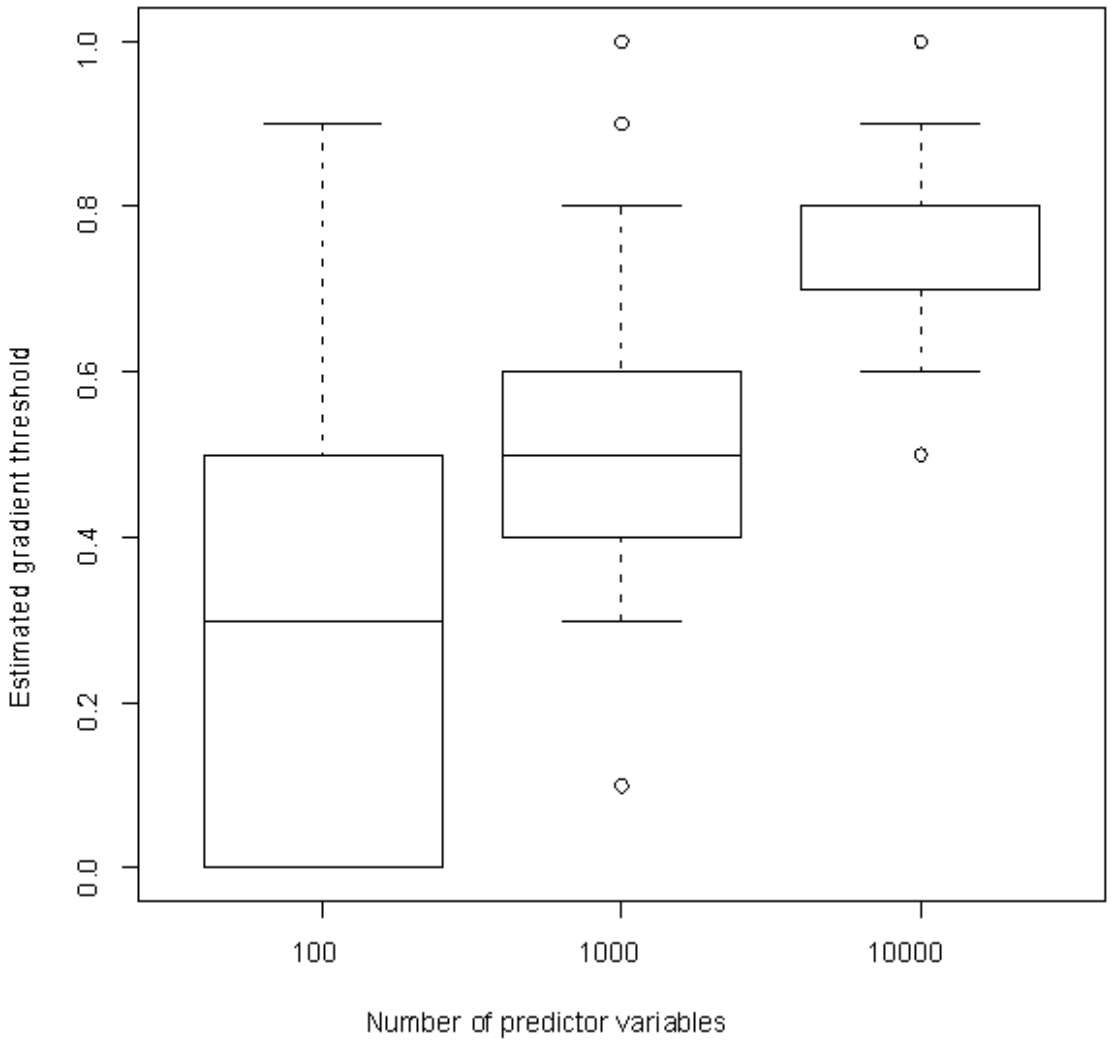

Figure 6: Distributions of estimated optimal gradient threshold values over the 100 trials of the latent variable regression problem for $n = 100$ (left), 1000 (middle), and 10000 (right) predictor variables, using total model selection (three–fold cross–validation) within each trial. The estimated values are seen to preferentially lie in regions of (here known) small error.

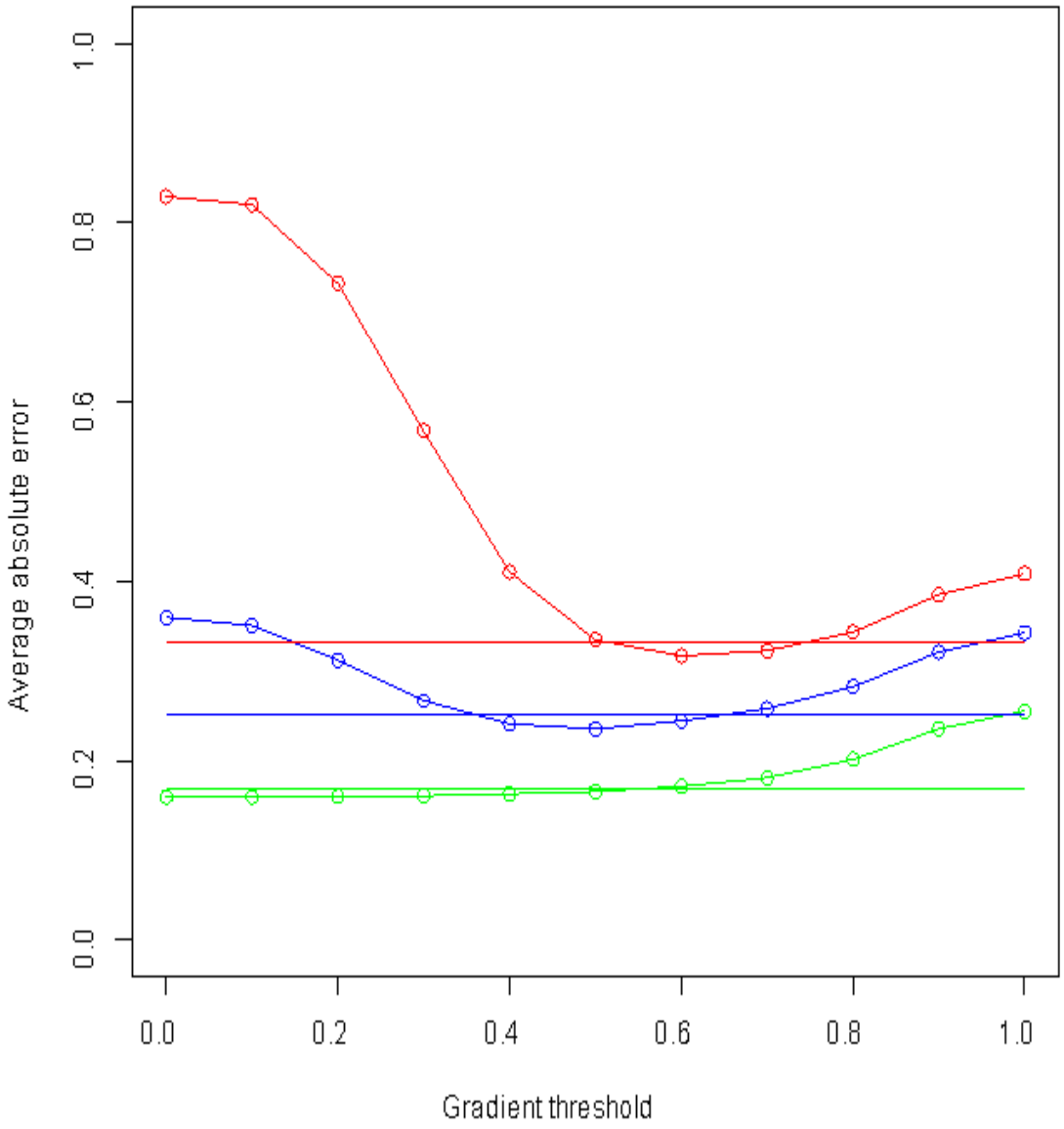

Figure 7: Repeat of Fig. 3 with the addition of horizontal lines representing the scaled absolute error averaged over the 100 trials, using the separately estimated optimal threshold value for each trial. The estimation is seen to increase error from the (here known) optimal threshold value by about 5% in all three $n = 100$ (green), 1000 (blue), and 10000 (red) situations.

5.1 Least–squares threshold gradient descent

For squared–error loss (12) the components of the negative gradient vector $\mathbf{g}(\nu)$ (15) are given by

$$g_k(\nu) = c(y, x_k) - \sum_{j=1}^{n} \hat{a}_j(\nu)\, c(x_j, x_k) \tag{33}$$

where $c(u, v)$ is the covariance of (centered) variables u and v as estimated from the training data (1)

$$c(u, v) = \frac{1}{N} \sum_{i=1}^{N} u_i v_i. \tag{34}$$

Thus, starting with $\{a_j(0) = 0\}_1^n$, the component updates along the path (22) are

$$\hat{a}_j(\nu + \Delta\nu) = \hat{a}_j(\nu) + \Delta\nu \cdot h_j(\nu) \tag{35}$$

with $h_j(\nu)$ given by (25) (26) (33).

Similarly, starting with $\{g_j(0) = c(y, x_j)\}_1^n$, each component of the negative gradient can be updated by

$$g_j(\nu + \Delta\nu) = g_j(\nu) - \Delta\nu \cdot \sum_{k=1}^{n} h_k(\nu)\, c(x_j, x_k). \tag{36}$$

Note that in (35) only components $\hat{a}_j(\nu)$ corresponding to variables for which $|\, g_j(\nu)\,| \geq \tau \cdot |\, g_{j^*}(\nu)\,|$ (27) change value at each step. Let $G(\nu)$ be the corresponding subset of variables

$$G(\nu) = \{j \; : |\, g_j(\nu)\,| \geq \tau \cdot |\, g_{j^*}(\nu)\,|\}. \tag{37}$$

Then the gradient update (36) becomes

$$g_j(\nu + \Delta\nu) = g_j(\nu) - \Delta\nu \cdot \sum_{k \in G(v)} h_k(\nu)\, c(x_j, x_k). \tag{38}$$

Thus, the computation required to compute the new negative gradient vector $\mathbf{g}(\nu + \Delta\nu)$ at each step ν is proportional to $n \cdot |\, G(\nu)\,|$, where $|\, G(\nu)\,|$ is the cardinality of $G(\nu)$. This cardinality depends on the value of the gradient threshold τ; larger values of τ give rise to smaller cardinalities. For $\tau = 0$, $|\, G(\nu)\,| = n$, whereas for $\tau = 1$, $|\, G(\nu)\,| = 1$.

The dominant computation is associated with calculating the predictor variable covariance matrix $\{c(x_j, x_k)\}$, being proportional to $Nn^2/2$. However, at any point ν on the path, only those columns $c(\cdot, x_k) = \{c(x_j, x_k)\}_{j=1}^n$ for which $\hat{a}_k(\nu) \neq 0$ are required to update the gradient (38). Therefore each column $c(\cdot, x_k)$ needs to be computed only at that point when its corresponding variable x_k first enters the model ($\hat{a}_k(\nu) \neq 0$). Furthermore, only those column elements $c(x_j, x_k)$ for which $\hat{a}_j(\nu) = 0$ need to be computed, since by symmetry $c(x_j, x_k) = c(x_k, x_j)$.

If the entire path is calculated then all n columns of the covariance matrix are ultimately required and computation is proportional to $Nn^2/2$. However, if one applies an "early stopping" strategy considerable computation can often be avoided. As the path is sequentially constructed, prediction risk is monitored on an independent "test" data sample T (not used for path construction)

$$\hat{R}_T(\nu) = \frac{1}{|\, T\,|} \sum_{i \in T} L(y_i, F(\mathbf{x}_i; \hat{\mathbf{a}}(\nu))), \tag{39}$$

with $F(\mathbf{x}; \mathbf{a})$ given by (2). At the point $\nu = \tilde{\nu}$ for which

$$\hat{R}_T(\tilde{\nu}) > \eta \cdot \min_{\nu < \tilde{\nu}} \hat{R}_T(\nu) \tag{40}$$

path construction is terminated. Here $\eta > 1$ is a stopping threshold. In all examples presented in this paper a default value of $\eta = 1.1$ was used. In practice, test set risk (39) need not be calculated at every step $\nu \leftarrow \nu + \Delta\nu$, typically every 100th step is sufficient.

The computational gain associated with early stopping depends on the value of the gradient threshold τ. For $\tau = 0$, all variables enter the model at the first step $\{\hat{\mathbf{a}}(\Delta\nu) \neq 0\}_1^n$ and there is little gain (See Section 5.4). However, for larger values of τ the estimated optimal solution $\hat{\mathbf{a}}(\hat{\nu})$ may contain many zero valued components. This will especially be the case if there is wide dispersion among the absolute values of the underlying true coefficients $\mathbf{a}^*$ (4). As an example, for the problem illustrated in Fig. 5 ($n = 10000$) only 333 coefficients were non zero for $\tau = 0.6$, and 150 for $\tau = 1$. For this problem ($N = 150$, $n = 10000$) actual running times on a Athlon 64 2.2 Ghz. computer were 3 and 2.5 seconds respectively for $\tau = 0.6$ and 1.0. For $N = 150$ and $n = 1000$, the corresponding times are roughly ten times faster.

5.2 Robust threshold gradient descent

It is well known that using squared–error loss (12) produces parameter estimates $\hat{\mathbf{a}}$ that can be heavily influenced by a small number of unusually extreme observations ("outliers") for which the response values y_i do not follow the dominant linear trend of most of the data. The presence of such outliers can often severely degrade the predictive accuracy of the resulting linear model $F(\mathbf{x}; \hat{\mathbf{a}})$ (2). The use of appropriate alternative loss criteria can mitigate this outlier effect, while maintaining accuracy comparable to that of least–squares in outlier free situations. One such popularly used loss is the Huber 1964 criterion

$$L(y, F(\mathbf{x}; \mathbf{a})) = \begin{cases} (y - F(\mathbf{x}; \mathbf{a}))^2/2 & | \, y - F(\mathbf{x}; \mathbf{a}) \, | < \delta \\ \delta(| \, y - F(\mathbf{x}; \mathbf{a}) \, | - \delta/2) & | \, y - F(\mathbf{x}; \mathbf{a}) \, | \geq \delta. \end{cases} \tag{41}$$

This is a compromise between squared–error loss (12) and least absolute deviation loss $L(y, F(\mathbf{x}; \mathbf{a})) = | \, y - F(\mathbf{x}; \mathbf{a}) \, |$. The value of the "transition" point δ differentiates the residuals that are treated as outliers being subject to absolute loss, from the other residuals subject to squared–error loss.

From (15) (41) the components of the negative gradient vector $\mathbf{g}(\nu)$ evaluated on the training data set (1) are

$$g_k(\nu) = \frac{1}{N} \sum_{i=1}^{N} \max(-\delta, \min(\delta, r_i(\nu))) \cdot x_{ik} \tag{42}$$

with

$$\{r_i(\nu) = y_i - F(\mathbf{x}_i; \hat{\mathbf{a}}(\nu)\}_1^N \tag{43}$$

being the current residuals at point ν along the path. This can be used directly with (15) (22) (25) (26) in a straight forward implementation of threshold gradient descent based on the Huber loss criterion (41). As with squared–loss (Section 5.1) such an implementation would be too slow for many large problems ($N \gtrsim n$). Unfortunately simple fast updating formulae, such as (38) for squared–error loss, do not exist for this criterion (41). However, it is possible to develop an approximate threshold gradient descent algorithm that maintains the accuracy and robustness properties of the direct implementation while achieving computational speed close to that of least–squares.

5.2.1 Fast algorithm

Defining

$$\{s_i(\nu) = sign(r_i(\nu))\}_1^N \tag{44}$$

and in analogy with (34)

$$\tilde{c}(u, v) = \tilde{c}(u, v; \nu) = \frac{1}{N} \sum_{i=1}^{N} u_i v_i I(| \, r_i(\nu) \, | < \delta), \tag{45}$$

the components of the negative gradient (42) can be expressed as

$$g_k(\nu) = \tilde{c}(y, x_k) + \delta \cdot (c(s(\nu), x_k) - \tilde{c}(s(\nu), x_k)) - \sum_{j=0}^{n} \hat{a}_j(\nu) \, \tilde{c}(x_j, x_k) \tag{46}$$

with $\{x_{i0} = 1\}_1^N$, and $c(s(\nu), x_k)$ defined by (34). Note that only the last term in (46) directly involves the current parameter values $\hat{\mathbf{a}}(\nu)$. The other terms involve the signs of the current residuals (44) and an indicator of their magnitudes being less than the transition point δ (45). If as a consequence of an update (22) none of these signs or indicators change value, then the negative gradient can be updated by (38) with $\tilde{c}(x_j, x_k)$ replacing $c(x_j, x_k)$.

Furthermore, under the assumption that the fraction of outliers

$$1 - \alpha(\nu) = \frac{1}{N} \sum_{i=1}^{N} I(|\, r_i(\nu)\,| \geq \delta) \tag{47}$$

is small, and/or the corresponding outlier indicators are not strongly correlated with the products $\{x_{ij}x_{ik}\}$, (45) can be approximated by

$$\tilde{c}(x_j, x_k) \backsimeq \alpha(\nu) \cdot c(x_j, x_k), \tag{48}$$

producing the negative gradient updates

$$g_j(\nu + \Delta\nu) = g_j(\nu) - \Delta\nu \cdot \alpha(\nu) \cdot \sum_{k \in G(v)} h_k(\nu)\, c(x_j, x_k). \tag{49}$$

Here $h_k(\nu)$ is given by (25) (26) and $G(\nu)$ by (37).

The updating formula (49) accounts only for the change in the last term in (46) (assuming (48)) as the parameter vector is incremented (22). Thus, it cannot be expected to remain even approximately valid over intervals of large length along the path. As the parameter values $\hat{\mathbf{a}}(\nu)$ change so do the signs of at least some of the residuals (44) and the contributions of the first two terms in (46) to the changes in the derivatives cannot be ignored. However, within relatively short intervals of small path length, over which the parameter values do not experience large changes, (49) can serve as a useful approximation. This suggests a hybrid strategy in which at various points ("milestones") along the path (say every 100 iterations of (22)) the exact derivatives are recomputed using (42). In between these milestones the approximate formula (49) is used to update the respective components of the negative gradient at each step (22).

For a given gradient threshold value τ (26), this hybrid between direct implementation of threshold gradient descent (42) and approximate updating (49), does not produce a path in the parameter space that exactly coincides with that produced by using the direct implementation alone. The assumption is is that the respective paths will not be too different (usually they are almost identical). In any case, using total model selection (Section 4.2), all that is required is that the paths generated for a sequence of threshold values $\tau \in [0, 1]$ cover the parameter space in a similar manner. There is no strong a priori reason to expect that a path generated by either strategy will come closest to the optimal predicting point $\mathbf{a}^*$ (4).

The hybrid strategy does maintain the robustness properties of the direct implementation. All estimates involving the response values $\{y_i\}_1^N$ are robust in that they are used only in the exact derivative calculations (42), yielding correspondingly robust gradient estimates. These are then used to define $\{h_k(\nu)\}_0^n$ (25) (26) used in the approximate updates (49).

The advantage of the hybrid strategy is computing speed. The dominant part of the computation is in the updating (49) and calculating the covariances $c(x_j, x_k)$. These have the same computing requirement as the least–squares algorithm (Section 5.1). Recalculating the exact gradient (42) at each milestone involves additional computation. However, being performed relatively infrequently only at the milestones, this computation is a small fraction of that required by a direct implementation that recalculates the gradient (42) at every step. As an example, for the problem shown in Fig. 5 ($N = 150$, $n = 10000$) running the robust algorithm took only 20% longer than the least–squares algorithm.

In order to use the Huber loss criterion (41) one must specify the value of the transition point δ. In the present implementation, its value was taken to be the αth quantile of the current absolute residuals (43) at each point (milestone) along the path

$$\delta(\nu) = \text{quantile}_\alpha \{|\, r_i(\nu)\,|\}_1^N. \tag{50}$$

Here $1 - \alpha$ is a specified fraction of the observations that are treated as outliers, subject to absolute loss, for each solution $\hat{\mathbf{a}}(\nu)$. Thus in (47) $\alpha(\nu) = \alpha$ is a constant over the entire generated path.

5.2.2 Illustration

Figure 8 illustrates the robustness properties of threshold gradient descent using this hybrid algorithm based on Huber loss (41). The data set is generated from the latent variable model (Section 4) with $N = 150$ observations and $n = 10000$ predictor variables. Added to these data are N_0 observations generated in the same way except that the standard deviation of the additive noise term ε in (28) was increased by a factor of ten. Thus, each of these added observations has a high probability of being an outlier. The respective number of added outliers shown in Fig. 8 are $N_0 \in \{0, 7, 15, 30, 50\}$ plotted along the abscissa as a fraction of the 150 non outliers. The three sets of points connected by straight lines are the corresponding values of scaled average absolute error (32) (averaged over 100 replications) for three values of $\alpha \in \{1.0$ (red), 0.8 (blue), 0.6 (green)$\}$ (50) identified in ascending order of their respective values of (32) at $N_0 = 0$.

From Fig. 8 one sees that decreasing the value of α produces a small loss of accuracy with respect to the least–squares algorithm ($\alpha = 1.0$) for outlier free data ($N_0 = 0$) with homoskedastic Gaussian distributed errors. The fractional increase in average error for $\alpha = 0.8, 0.6$ is 7% and 12% respectively. However, in the presence of increasing numbers of outliers the performance of the least–squares algorithm degrades very rapidly with its average error more than doubling for 10% added outliers ($N_0 = 15$). The performance of the robust competitors degrade much more gracefully, with increased average error at $N_0 = 15$ of 15% ($\alpha = 0.8$) and 9% ($\alpha = 0.6$) from their outlier free values ($N_0 = 0$). As would be expected, smaller values of α produce more robustness to increasing number of outliers among the training data, at the price of slightly lower efficiency (higher error) for less corrupted data.

Therefore, one's choice of a value for α reflects (prior) suspicions concerning the fraction of outliers that might be present in the training data (1). Smaller values of α provide more insurance against very bad situations (many outliers), at a modest increased cost of slightly larger expected error if there are only a few or no outliers. For $\alpha < 1.0$ there is also a modest increase in computational cost for training.

5.3 Binary classification: squared–error ramp

In the binary classification problem, the response variable realizes two values, e.g. $y \in \{-1, 1\}$. The goal is to produce a linear model $F(\mathbf{x}; \mathbf{a})$ (2) that represents a score reflecting confidence that $y = 1$, given a set of joint values for the predictor variables $\mathbf{x}$. This score can then be used in a decision rule to obtain a corresponding prediction

$$\hat{y}(\mathbf{x}, \mathbf{a}) = \begin{cases} 1 & \text{if } F(\mathbf{x}; \mathbf{a}) > t^* \\ -1 & \text{otherwise.} \end{cases} \tag{51}$$

Here t^* is a threshold whose value is chosen to minimize misclassification risk

$$t^* = \arg \min_{-\infty \leq t \leq \infty} E_{y\mathbf{x}}[(1 + y)\, l_+\, I(F(\mathbf{x}; \mathbf{a}) \leq t)] + (1 - y)\, l_- I(F(\mathbf{x}; \mathbf{a}) > t)]/2 \tag{52}$$

where $l_\pm$ is is the cost of misclassifying an observation with actual value $y = \pm 1$ respectively. In many applications it is the scoring function $F(\mathbf{x}; \mathbf{a})$ itself that is of primary interest.

Ideally, one would like to use misclassification risk (52) to directly construct a corresponding loss criterion $L(y, F(\mathbf{x}; \mathbf{a}))$ for obtaining path estimates $\hat{\mathbf{a}}(\nu)$ from the training data (1). As is well known, this strategy is generally unsuccessful because misclassification risk is not a continuous function of the parameter values; for example the corresponding gradient $\mathbf{g}(\nu)$ (15) does not exist. One must therefore use a smooth criterion as a surrogate for misclassification risk.

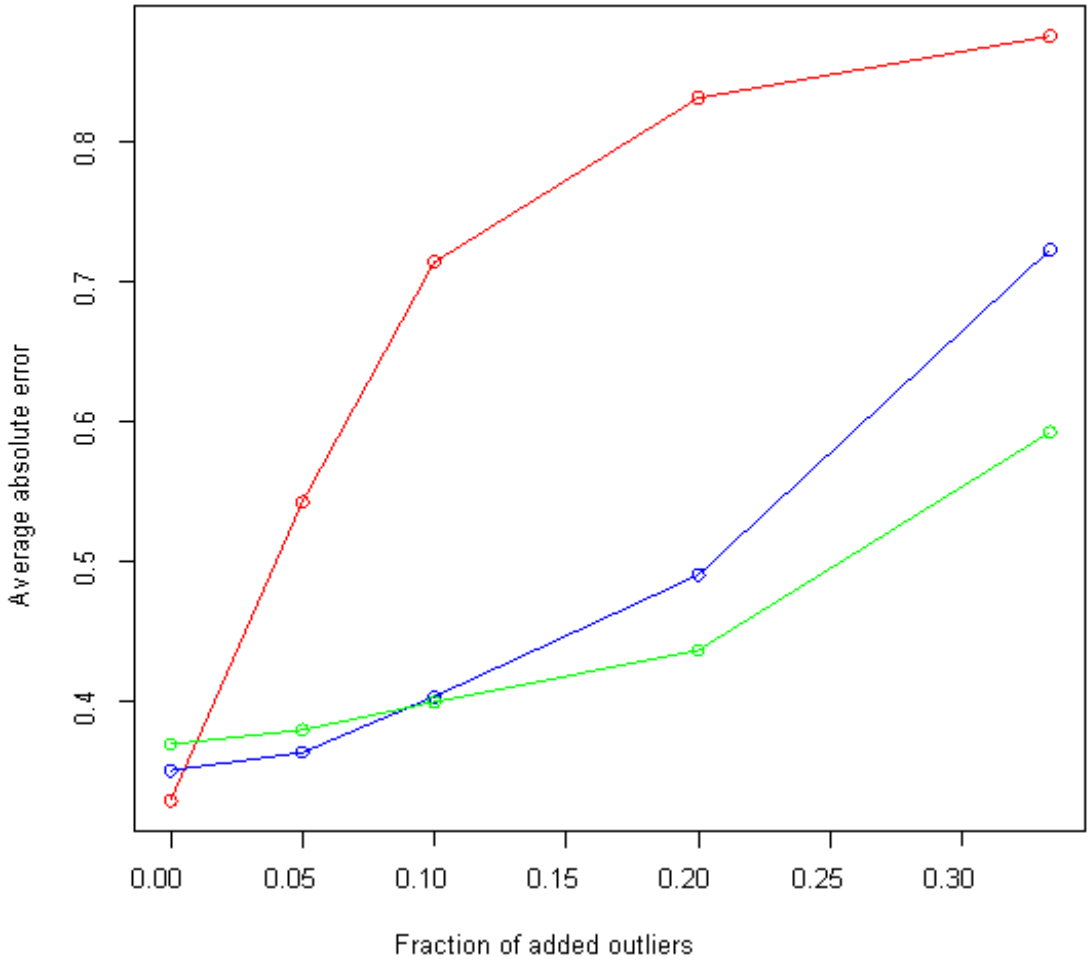

Figure 8: Expected scaled absolute error as a function of the fraction of added outliers, for three values of the Huber loss parameter $\alpha = 1$ (red), $\alpha = 0.8$ (blue), and and $\alpha = 0.6$ (green), in the latent variable regression problem with $n = 10000$ predictor variables. With no outliers, $\alpha = 1$ (squared–error loss) provides the best accuracy with smaller values of α showing slight degradation. Smaller α values provide more robustness with their errors increasing less rapidly with increasing numbers of outliers in the data.

A wide variety of such surrogates have been proposed (see for example Hastie *et al* 2001). Here we propose the squared–error "ramp" loss criterion

$$L(y, F(\mathbf{x};\mathbf{a})) = (y - H(F(\mathbf{x};\mathbf{a}))^2 \tag{53}$$

where

$$H(F) = \max(-1, \min(1, F)) \tag{54}$$

is a linear "ramp" function truncated at -1 and 1. This loss can alternatively be expressed as

$$L(y, F(\mathbf{x};\mathbf{a})) = \min(4, [1 - y\,F(\mathbf{x};\mathbf{a})]_+^2) \tag{55}$$

so that the least–squares ramp (53) (54) can be viewed as a "capped" version of the square of the hinge loss

$$L(y, F(\mathbf{x};\mathbf{a})) = [1 - y\,F(\mathbf{x};\mathbf{a})]_+ \tag{56}$$

used with support vector machines (Vapnik 1996). The population minimizer of the corresponding risk using (53) (54) is

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y\mathbf{x}} L(y, F(\mathbf{x})) = 2 \cdot \Pr(y = 1 \,|\, \mathbf{x}) - 1 \tag{57}$$

which being monotone in $\Pr(y = 1 \,|\, \mathbf{x})$ represents an optimal scoring function. Capping this loss (55) at the value four when $y\,F(\mathbf{x};\mathbf{a}) < -1$ provides robustness against "outliers" by diminishing the influence of misclassified observations far from the decision boundary. This is especially important for linear models (2), since the presence of these outliers can often badly distort the resulting parameter estimates $\hat{\mathbf{a}}(\nu)$, thereby degrading classification accuracy. Such outliers, for example, could be introduced by mislabeling among some of the training observations.

The robust loss criterion (53) (54) is not a strictly convex function of the parameters $\mathbf{a}$. Thus, the corresponding training data risk based on this loss is similarly non convex and may contain multiple local minima. This has the potential to increase instability (variance) in the gradient descent paths, especially near the end $\hat{\mathbf{a}}(\nu) = \hat{\mathbf{a}}$ (5) corresponding to less regularization. This lack of convexity is a result of capping the loss $L(y_i, F(\mathbf{x}_i;\hat{\mathbf{a}}(\nu))) = 4$ for outliers $y_i F(\mathbf{x}_i;\hat{\mathbf{a}}(\nu)) < -1$. At the beginning of the path $\hat{\mathbf{a}}(\nu_0)$ (9) there are no such outliers and the risk appears convex for subsequent steps (22) until outliers may begin to appear. The degree of apparent non convexity and the corresponding increase in instability tends to increase gradually as the path proceeds towards less regularization, to the extent that additional outliers are identified. Thus, as with convex loss criteria, more regularization implies more stability. The advantage of the robust loss criterion is that the influence of outliers on the path solutions $\hat{\mathbf{a}}(\nu)$ does not increase with decreasing values of $y_i F(\mathbf{x}_i;\hat{\mathbf{a}}(\nu))$. Convex loss criteria are increasingly influenced by such observations which can thereby have a large impact on the resulting paths, decreasing accuracy if their high influence is a result of mislabeling.

5.3.1 Fast algorithm

Defining $\{x_{i0} = 1\}_1^N$, the components of the negative gradient are from (2) (15) (53) (54)

$$g_k(\nu) = \tilde{c}(y, x_k;\nu) - \sum_{j=0}^{n} \hat{a}_j(\nu)\tilde{c}(x_j, x_k;\nu) \tag{58}$$

where here

$$\tilde{c}(u, v;\nu) = \frac{1}{N}\sum_{i=1}^{N} u_i v_i I(|\, F(\mathbf{x}_i;\hat{\mathbf{a}}(\nu)\,| < 1). \tag{59}$$

If at a point ν on the path, none of the indicators in (59) change value for any observation as a result of a step (22), then updating formulae analogous to (38) can be used to compute the components of the new negative gradient

$$g_k(\nu + \Delta\nu) = g_k(\nu) \; - \Delta\nu \cdot \sum_{j \in G(\nu)} h_j(\nu)\, \tilde{c}(x_j, x_k;\nu). \tag{60}$$

If over intervals of short path length representing small changes in $\hat{\mathbf{a}}(\nu)$, relatively few observations change their corresponding indicator values, then using (60) to update the gradient can serve as a reasonable approximation, using the covariance values (59) computed at the beginning of the interval. These gradient updates (60) require the same computation as those for squared–error loss (38). Furthermore, as with squared–error loss, covariances $\{\tilde{c}(x_j, x_k; \nu)\}_{j=0}^{n}$ need only be calculated at that point where the corresponding variable x_k first enters the model ($\hat{a}_k(\nu) \neq 0$). Thus, using early stopping (39) (40), considerable computation can often be saved, especially for larger values of of the gradient threshold parameter τ (26).

Over longer path intervals representing substantial changes in the parameter values $\hat{\mathbf{a}}(\nu)$, the values of the covariances (59) can be expected to substantially change as well, since many of the indicators will change value. This is especially the case at the early stages of path construction. Thus, at various milestones during path construction (say every 100 steps (22)) the covariances (59) must be recomputed. However there are updating formulae that can reduce this computation as well.

Let ν be a point on the path at a particular milestone and ν_0 be that at the previous milestone. Define the indicators

$$\{\eta_i(\nu) = I(|\, F(\mathbf{x}_i; \hat{\mathbf{a}}(\nu))\,| < 1)\}_1^N$$

and

$$s_i(\nu_0, \nu) = \begin{cases} -1 & \eta_i(\nu_0) = 1 \;\&\; \eta_i(\nu) = 0 \\ 0 & \eta_i(\nu_0) = \eta_i(\nu) \\ 1 & \eta_i(\nu_0) = 0 \;\&\; \eta_i(\nu) = 1. \end{cases}$$

Then

$$\tilde{c}(u, v; \nu) = \tilde{c}(u, v; \nu_0) + \frac{1}{N} \sum_{s_i(\nu_0,\nu)\neq 0} u_i v_i \, s_i(\nu_0, \nu) \tag{61}$$

can be used at each successive milestone to update the covariances used in (60). This can save considerable computation since the cardinality of $\{s_i(\nu_0, \nu) \neq 0\}$ is usually small compared to the total number of observations N, especially during the later stages of path construction when the coefficient values $\hat{\mathbf{a}}(\nu)$ tend to stabilize.

Updating formulae analogous to (60) (61), and a corresponding fast algorithm, can as well be derived for the *uncapped* version of squared SVM loss

$$L(y, F(\mathbf{x}; \mathbf{a})) = [1 - y\, F(\mathbf{x}; \mathbf{a})]_+^2. \tag{62}$$

This convex criterion has been proposed for binary classification (Zhang 2003). Although less robust than the squared–error ramp (53) (55) (or standard SVM loss (56)) this criterion may prove useful in settings where classification outliers are known not to exist.

5.3.2 Illustration

We illustrate application of this threshold gradient descent algorithm based on squared–error ramp loss (53) (54) (55) in a binary classification problem. The data are generated using the latent variable model (Section 4), but with response values taken to be

$$\{\tilde{y}_i = sign(y_i - \bar{y})\}_1^N. \tag{63}$$

Here $\bar{y}$ is the response mean. Performance is measured in terms of misclassification risk using $l_\pm = 1$ (52) (error rate). The corresponding Bayes optimal error rate for this problem is 0.15.

Figure 9 shows results for this classification problem analogous to those shown in Fig. 7 for regression. Error rate (averaged over 100 replications) is plotted as a function of threshold parameter value τ (26) for three situations corresponding to $n = 100$ (green), 1000 (blue), 10000 (red) total predictor variables (0, 900, 9900 pure noise variables respectively). The horizontal lines represent the the corresponding error rates using total model selection (Section 4.2). The points on the left represented by asterisks represent the corresponding error rates for a linear

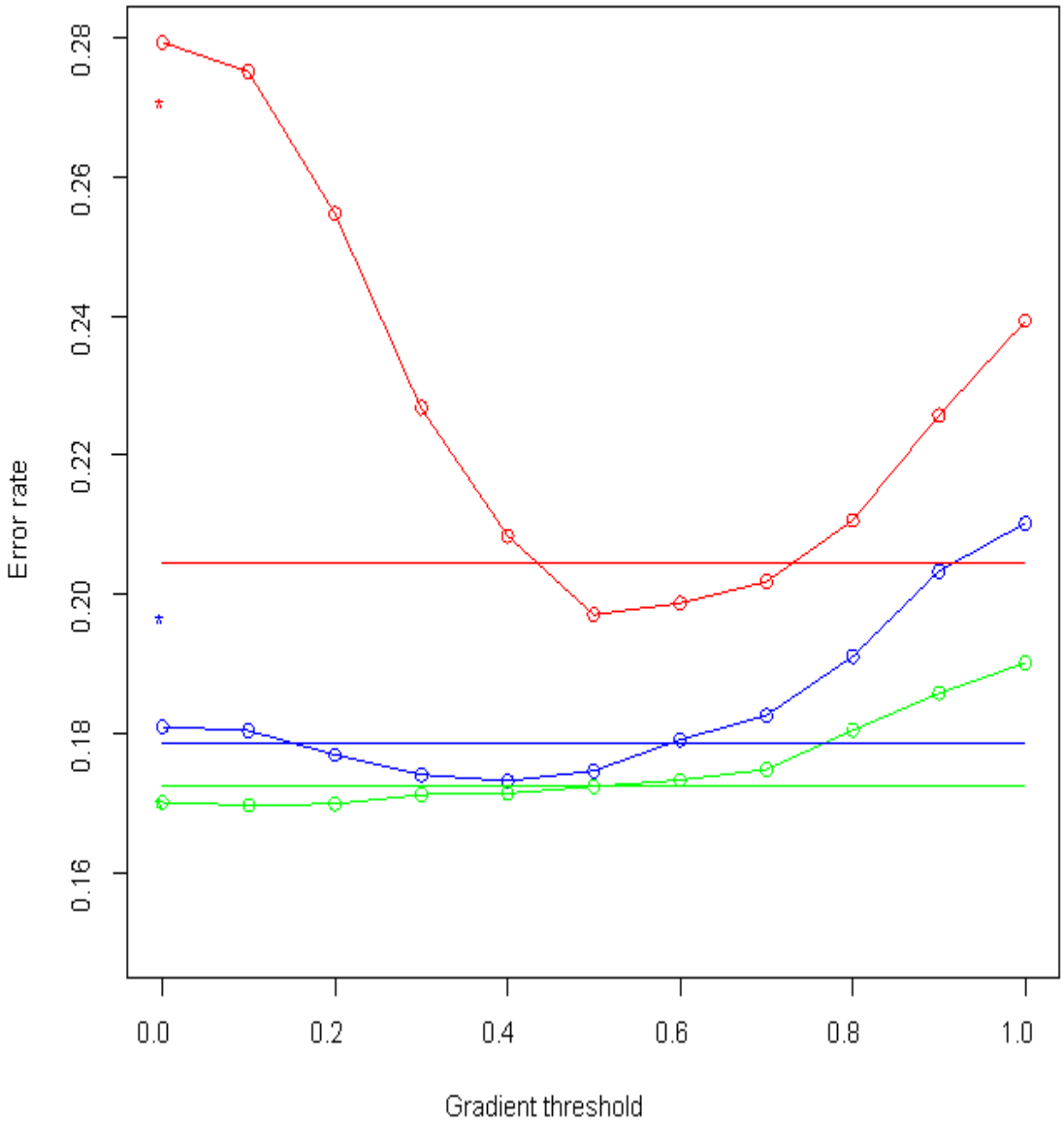

Figure 9: Results analogous to Fig. 7, here for the latent variable *classification* problem. The error rate of threshold gradient descent based on the ramp loss criterion is plotted as a function of gradient threshold value for $n = 100$ (green), 1000 (blue), and 10000 (red) predictor variables. The asterisks at the left represent the corresponding results for a linear kernel support vector machine. The horizontal lines are the corresponding results from estimating the optimal threshold value. The overall results seen here are qualitatively similar to those of the regression problem. Quantitatively, there is less dependence of error rate on the specifics of the estimated model, or on the number of irrelevant variables.

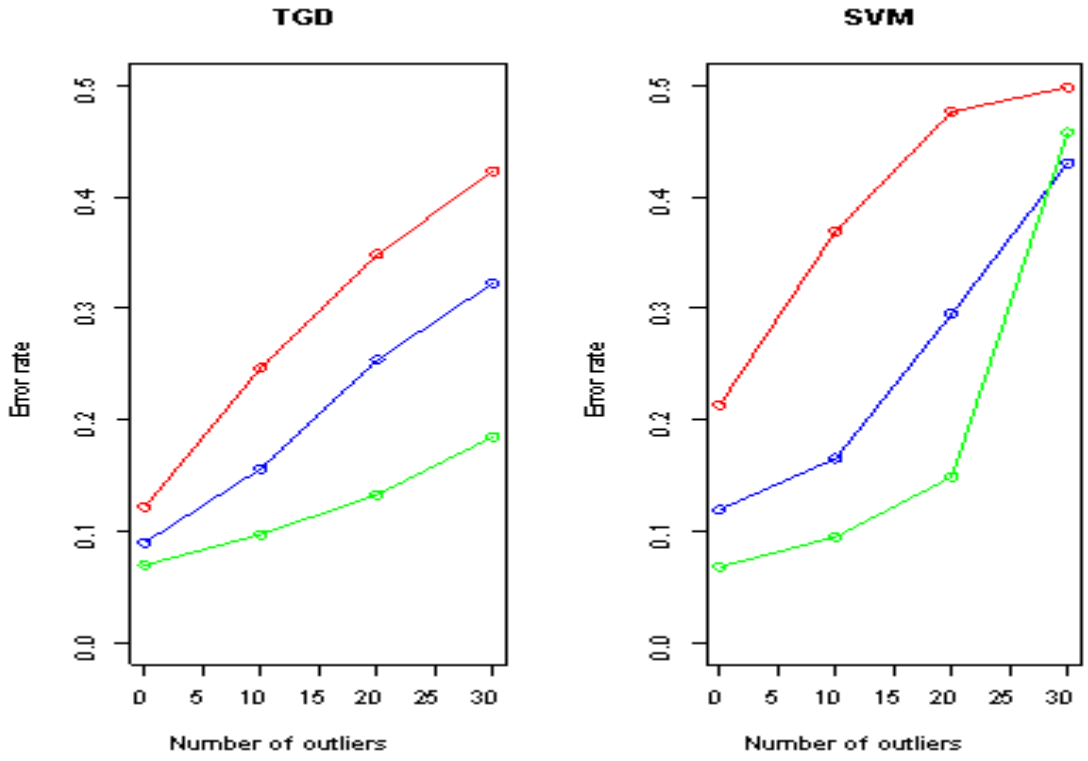

Figure 10: Error rate as a function of number of added classification outliers for threshold gradient descent based on ramp loss (left panel) and a linear kernel support vector machine (right panel), for the latent variable classification problem with $n = 100$ (green), 1000 (blue), and 10000 (red) predictor variables. Both methods are fairly resistant, with ramp loss providing more robustness in the presence of large numbers of outliers.

kernel support vector machine (SVM^{light} Joachims 1999 using the supplied default procedure parameter values). Note that the origin of the vertical scale is 0.15 representing the Bayes optimal error rate. Thus, the results shown in Fig. 9 reflect reducible error in analogy with Figs. 3 and 7.

The results presented in Fig. 9 for classification qualitatively reflect those for the regression problem shown in Fig. 7. The distinguishing characteristic of the classification results is considerably less sensitivity of error rates to the values used for the gradient threshold τ (26). They are also less sensitive to the presence of increasing numbers of pure noise variables. This is to be expected since misclassification risk depends only on the $sign(F(\mathbf{x}, \hat{\mathbf{a}}) - t^*)$ (51), whereas absolute loss (32) depends on the actual value of $F(\mathbf{x}, \hat{\mathbf{a}})$. Thus, less accurate parameter estimates $\hat{\mathbf{a}}$ tend to produce less degradation in misclassification risk.

Support vector machines use the ridge penalty (7) as a basis for regularization. As discussed in Section 2 this should produce results similar to that of full gradient descent regularization (14) (58). This is reflected in Fig. 9 where the results for SVM^{light} are seen to be fairly similar to those for zero gradient threshold value ($\tau = 0$ (26)).

5.3.3 Classification robustness

Robustness to the presence of mislabeled training data (outliers) is studied in Fig. 10. Here the data are generated as in the classification example above, except that a specified number

N_0 of the observations $(y_i, \mathbf{x}_i)$ with the *smallest* $\Pr(y_i = 1 \,|\, \mathbf{x}_i)$ are labeled as $y_i = 1$. These N_0 observations thus represent extreme outliers for the classification problem. Additionally, the standard deviation of the additive noise in (28) was reduced so as to produce a ten–to–one signal–to–noise ratio. After thresholding (63), the Bayes optimal error rate is thereby reduced to 0.033 so that the effect of the outliers present in the training data is more apparent.

Figure 10 shows the error rate (averaged over 100 replications) as a function of the number of added extreme outliers $N_0 \in \{0, 10, 20, 30\}$ for the three situations corresponding to $n = 100$ (green), 1000 (blue), and 10000 (red) total predictor variables. The left panel shows results for threshold gradient descent (TGD) with total model selection, and the right panel for SVM^{light}.

Support vector machines are based on the hinge loss (56). Among convex loss criteria it is the least affected by the presence of classification outliers $y_i F(\mathbf{x}_i; \mathbf{a}) < -1$, especially when compared with squared–error loss (12), squared–hinge loss (62), or exponential loss $L(y, F) = \exp(-yF)$ used with AdaBoost (Freund and Schapire 1996). This is reflected in the results presented in Fig.10 (right panel). Depending on the total number of predictor variables n, SVM^{light} is fairly resistant to the presence of extreme outliers, seriously breaking down only when there are a large number of them. Threshold gradient descent based on squared–error ramp loss (53-55) is seen (left panel) to have similar robustness properties to support vector machines for smaller numbers of added extreme outliers, and somewhat more resistance when larger numbers are added.

5.4 Straight forward implementation

The algorithms described in Sections 5.1–5.3 can dramatically reduce computation compared to the straight forward implementation (15) (22) (25) (26) for their respective loss criteria (12) (41) (53), especially when the early stopping strategy (40) is employed. There are however special situations for which the straight forward implementation can be considerably faster.

Let $\tilde{\nu}$ (40) be the point on the path where iterations terminate through early stopping, $n(\tilde{\nu})$ the number of non zero components of the corresponding solution coefficient vector $\hat{\mathbf{a}}(\tilde{\nu})$, and $M(\tilde{\nu})$ the number of steps (22) up to that point. If

$$n(\tilde{\nu}) >> M(\tilde{\nu}) \tag{64}$$

then the straightforward implementation is likely to require less computation than those described in Sections 5.1–5.3.

This condition (64) occurs with small gradient threshold values ($\tau \preceq 0$) (26) in settings for which heavy regularization, corresponding to small $M(\tilde{\nu})$ values, is appropriate. Such heavy regularization is required in situations where the number of predictor variables n is much larger than the number of observations N. As discussed in Section 2, small gradient threshold values generate paths quite similar to those derived from using a ridge penalty (7), especially in regions of heavy regularization. Therefore in those situations for which $n >> N$ *and* ridge–like paths are appropriate, gradient descent ($\tau \preceq 0$) based on the straight forward implementation represents an attractive approach with computation proportional to $n \cdot N \cdot M(\tilde{\nu})$. As an example, for the problem illustrated in Fig. 5 ($n = 10000, N = 150$) the straightforward implementation required less that two seconds for $\tau = 0$, rather than 42 seconds needed for the "fast" algorithm with updating (Section 5.1). Also this implementation involves negligible random access memory beyond that required to store the data. In nearly all other situations, the algorithms described in Sections 5.1–5.3 are much faster with computation being proportional to $n \cdot N \cdot n(\tilde{\nu})$ and memory requirement $n \cdot n(\tilde{\nu})$.

6 Constrained threshold gradient descent

Sometimes domain knowledge is available concerning the nature of the optimal parameter values $\mathbf{a}^*(4)$. This information often takes the form of constraints on the allowed values of the corresponding components $\{a_j^*\}_1^n$. To the extent that this information is correct, forcing the corresponding solution coefficients $\{\hat{a}_j\}_1^n$ to respect these constraints has the potential to produce

more accurate estimates and thereby better prediction. In the context of threshold gradient descent, imposing many types of user specified constraints is especially straight forward. In this section we illustrate how the pathfinding algorithms can be modified to respect two such types of constraints: smoothness of the coefficient values $\{\hat{a}_j\}_1^n$ on the variable index j, and non negativity of the solution values $\hat{a}_j \geq 0$ on selected coefficients.

6.1 Smooth threshold gradient descent

For the problems considered above, the predictor variables $\{x_j\}_1^n$ are taken to be measurements of attributes with no special a priori relationship. In particular the index j that labels the respective attributes is presumed to realize unorderable categorical values, and solutions to the prediction problem are required to be equivariant under all possible labelings. That is, modifications to the variable labels produce equivalent solutions. In some problems this is not the case. Variables with particular joint label values have special relationships.

A common application of this is when some or all the variables represent functions $f(t)$, such as analog signals, to be used to predict the value of an outcome or response variable y using the linear model

$$y = a_0 + \int a(t)\, f(t)\, dt. \tag{65}$$

The predictor variables $\{x_j\}_1^n$ represent measurements of the function value at various discretization points $\{t_j\}_1^n$ (time, frequency, location, etc.). If all signals are measured at the same set of discretization points then the corresponding function values $\{x_j = f(t_j)\}_1^n$ can be regarded as predictor variables in the standard learning (regression/classification) framework, treating the respective function measurements $\{x_j\}_1^n$ as being categorically labeled in the linear model (2). This approach assumes nothing concerning the relative values of the coefficient ("contrast") function $a(t)$ as a function of t.

In many applications it is known (or presumed) that the true (population) contrast function $a^*(t)$ is a smooth function of its argument. That is, $a^*(t) \backsimeq a^*(t')$ for $t \backsimeq t'$. If successive discretization points are not to widely separated one might expect a similar relation to hold for them

$$a^*(t_j) \backsimeq a^*(t_{j'}),\ |\, j - j' \,| < j_0, \tag{66}$$

or at least that there be a smooth relationship among $\{a^*(t_j)\}$ with close index values j. Here $\mathbf{a}^*$ is given by (4) and $j_0 << n$ is a relatively small integer. To the extent this presumption holds, constraining the solution coefficients $\{\hat{a}_j\}_1^n$ to respect this smoothness property may improve performance.

The central ingredient in all threshold gradient descent algorithms is the negative gradient $\mathbf{g}(\nu)$ (15) of the empirical risk, evaluated at the current parameter estimates $\hat{\mathbf{a}}(\nu)$ at each successive point ν along the path. Its individual components $\{g_j(\nu)\}_1^n$ determine the respective updates to the corresponding parameter values $\{\hat{a}_j(\nu + \Delta\nu)\}_1^n$ through (22) (25) (26). The only aspect of these algorithms that explicitly involves the predictor variable index j is in labeling the individual components of the gradient $g_j(\nu)$ connecting them to their corresponding coefficient $\hat{a}_j(\nu)$.

As motivated in Section 1.3, the paths created by threshold gradient descent in this setting should be encouraged to visit points $\hat{\mathbf{a}}(\nu)$ for which there is a smooth relationship among its components $\{\hat{a}_j(\nu)\}_1^n$ as a function of the index j. This is directly accomplished by imposing such a relationship on the components of the negative gradient $\{g_j(\nu)\}_1^n$ at each point ν along the path. Specifically, the empirical negative gradient (15) is replaced by a corresponding quantity $\tilde{\mathbf{g}}(\nu)$ that is as close as possible to it under a smoothness constraint. That is

$$\{\tilde{g}_j(\nu)\}_1^n = S_\gamma(\{g_j(\nu)\}_1^n) \tag{67}$$

where $S_\gamma(\{\eta_j\}_1^n)$ ("smoother") represents an operator that produces a smoothed version of the values $\{\eta_j\}_1^n$ as a function of j, and γ is a parameter whose value regulates the degree of smoothness imposed. If the predictor variable index j is univariate $j \in R^1$ ("signal") then a standard

univariate smoother can be used. For a bivariate index $j \in R^2$ ("image") a corresponding two–dimensional smoother is appropriate. All other aspects of the threshold gradient descent procedure remain unchanged including the fast algorithms described in Section 5.

For a specified degree of smoothness the details of the particular smoother employed are not likely to be very important, especially since in most applications the successive discretization points $\{t_j\}_1^n$ are equidistant realizing values on a regular grid. Also, as noted in Section 1.3, all that is required is that the corresponding paths induced by gradient threshold values (26) $\tau \in [0, 1]$ cover the space in a similar manner.

As in any smoothing problem the degree of smoothness, regulated by the parameter γ, is likely to be important. A good value will depend on the smoothness properties of the optimal coefficients $a^*(t_j)$, namely the value of j_0 in (66). To the extent such smoothness properties are known a reasonable value for γ can be chosen. If there is no such knowledge, γ can be regarded as an additional regularization parameter of the procedure whose value is estimated through a model selection procedure such as cross–validation. The threshold gradient descent algorithms described in Section 5 are generally fast enough to permit this with feasible computation. In any case, imposing even a small degree of smoothness in these situations is likely to produce improvement over none at all.

In the context of smoothed threshold gradient descent the smoothness properties of the path solutions $\hat{\mathbf{a}}(\nu)$ will also be somewhat dependent on the value of the gradient threshold parameter τ (26), especially for large values $\tau \backsimeq 1$. A smoothness constraint on the solution coefficients $\{\hat{a}_j\}_1^n$ inherently implies a reduction in the diversity of their values. As discussed in Section 3.1, larger values of τ encourage increased diversity thereby favoring rougher solutions for a given gradient smoother (67) with a specified degree of smoothness γ. Thus, if smooth solutions are to be encouraged, larger values of τ are less appropriate.

6.1.1 Illustrations

These concepts are first illustrated in the context of a simple problem. Data were simulated from the model

$$y = \sum_{j=1}^{n} a_j^* x_j + \varepsilon \tag{68}$$

with a_j^* being a smooth function of j

$$a_j^* = \exp[-\frac{1}{2}((j - 50)/20)^2]. \tag{69}$$

There are $n = 1000$ predictor variables $\{x_j\}_1^{1000}$, generated from a standard normal distribution. The additive error ε was normally distributed with variance chosen to produce a 2/1 signal to noise ratio, $var(\varepsilon) = var(y)/5$. $N = 150$ observations were generated according to this prescription for each of 100 replications.

Smoothed threshold gradient descent (67) based on squared–error loss (Section 5.1) was applied to each of the 100 data sets using a simple local linear smoother. The smoothed version $\tilde{g}_j(\nu)$ of each gradient component $g_j(\nu)$ is obtained by its predicted value from a linear least–squares fit of the $\{g_k(\nu)\}$ on k, using the K nearest discretization points to j. Thus, the degree of smoothness imposed is regulated by the size K of the corresponding neighborhoods ("smoother span").

Figure 11 shows scaled absolute error (32), averaged over the 100 replications, jointly as a function of the smoother span K and gradient threshold τ (26). Here gradient smoothing is seen to dramatically increase accuracy for many joint (K, τ) values. Coefficient estimates produced without smoothing ($K = 1$) have little predictive power beyond that of the overall response median. For all values of K prediction error is minimized at gradient threshold values $0.3 \lesssim \tau \lesssim 0.4$. Note that these minimizing values are similar to those for the latent variable model ($n = 1000$) shown in Fig. 4. The largest gradient threshold values $\tau \gtrsim 0.8$ are seen to provide increasingly poor performance, with $\tau = 1$ being slightly worse than no smoothing at

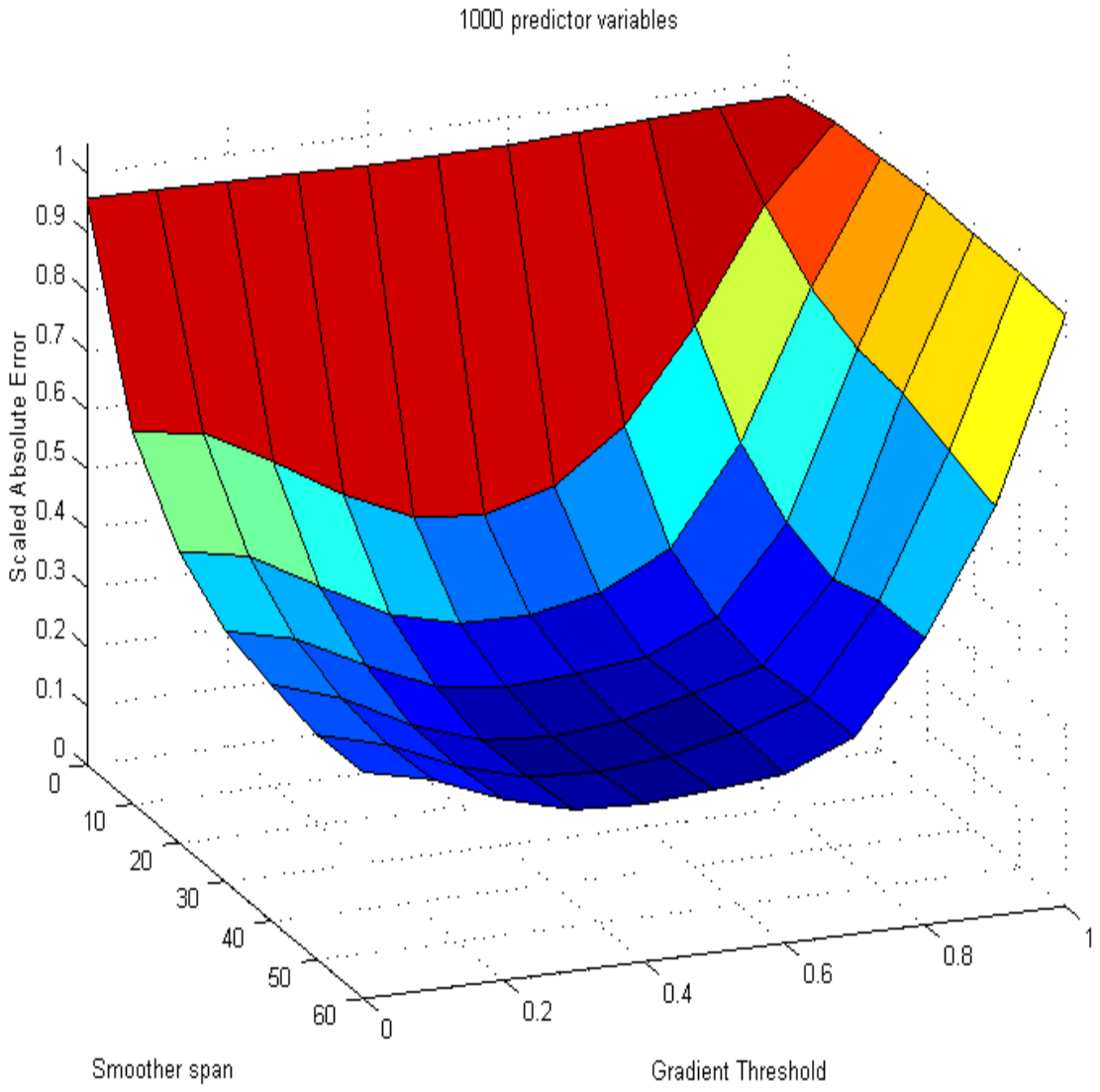

Figure 11: Expected scaled absoute error as a joint function of gradient threshold τ and smoother span K, used with smooth threshold gradient descent, for the smooth coefficient regression problem with *non smooth* predictor variables. Here a smoothness constraint on the parameter estimates is seen to dramatically improve accuracy over no smoothing ($K = 1$), for joint values $0.2 \leq \tau \leq 0.8$ and $5 \leq K \leq 60$, with optimal values $\tau = 0.5$ and $K = 40$.

all. For all gradient threshold values prediction error is minimized at $K \backsimeq 40$ nearest neighbors, and is slowly increasing for larger smoother span values. This is consistent with the properties of the optimal coefficient values $\{a_j^*\}_1^n$ defined by (69).

Corresponding results for $n = 100$ and $n = 10000$ predictor variables (not shown) have quite similar characteristics to that for $n = 1000$ shown here. Gradient smoothing dramatically reduces prediction error in these settings as well. For all values of the gradient threshold parameter τ prediction error is minimized at $K \backsimeq 40$, increasing slowly for larger smoother spans and somewhat more rapidly for smaller spans. For fixed value of the smoother span K, the optimal gradient threshold values are $\tau = 0$ for $n = 100$, and $\tau \backsimeq 0.6$ for $n = 10000$, again being similar the the corresponding values for the latent variable model as seen in Fig. 4.

This example illustrates that there are situations for which gradient smoothing (67) can substantially reduce prediction error in the context of threshold gradient descent when the optimal coefficients $\{a_j^*\}_1^n$ are smooth functions of the index j. However, this example is not representative of many such applications. Often, the predictor variables $\{x_j\}_1^n$ are also themselves smooth functions of the index j. This is especially the case when they are measurements of analog signals. As discussed in Frank and Friedman 1993 (rejoinder) this type of smoothness in the predictor variables automatically induces a corresponding degree of smoothness on the coefficient estimates $\{\hat{a}_j\}_1^n$ in the presence of regularization that discourages diversity of their absolute values.

In the context of threshold gradient descent regularization this effect can be seen directly from the expression for the components of the gradient

$$g_j(\nu) = \frac{1}{N} \sum_{i=1}^{N} L'(y_i, F(\mathbf{x}_i; \mathbf{a}(\nu))) \cdot x_{ij} \tag{70}$$

with $L'(y, F)$ being the derivative of $L(y, F)$ with respect to its second argument. To the extent that the measurements $\{x_{ij}\}_{j=1}^n$ are smooth functions of j for each observation i, the gradient components $\{g_j(\nu)\}_1^n$ will be correspondingly smooth, and imposing an additional external smoothing constraint can have little effect. Only when the characteristic smoothness of the optimal coefficients $\{a_j^*\}_1^n$ is much greater than that of the predictor variables $\{x_{ij}\}_{j=1}^n$ would one expect substantial improvement from gradient smoothing.

This effect is illustrated in the context of a somewhat more "realistic" example. Data ($N = 150$ observations, 100 replications) are generated from the model

$$y = \sum_{j=1}^{n} a_j^* x_j^* + \varepsilon$$

with a_j^* given by (69) and $var(\varepsilon) = var(y)/5$ (2/1 signal to noise ratio). Each of the $n = 1000$ predictor variables x_j is taken to be a noisy measurement of a corresponding "true" signal x_j^*

$$x_j = x_j^* + \delta_j \tag{71}$$

with the variance of the added (Gaussian) noise δ_j chosen to produce 2/1 signal to noise. The true underlying signals are generated as

$$x_j^* = \sum_{k=1}^{L} b_{jk} l_k, \tag{72}$$

with each (latent variable) l_k generated from a standard normal distribution. Here the loading coefficients b_{jk} for each variable x_j are taken to be smooth functions of the index j

$$b_{jk} = \exp[-\frac{1}{2}((j - m_k)/\sigma)^2] \tag{73}$$

with $m_k = 20 \cdot k - 10$ and $\sigma = 10$. There are $L = n/20 = 50$ latent variables. By construction (72) (73) the true underlying signals $\{x_j^*\}_1^n$ are smooth functions of the index j. As an example,

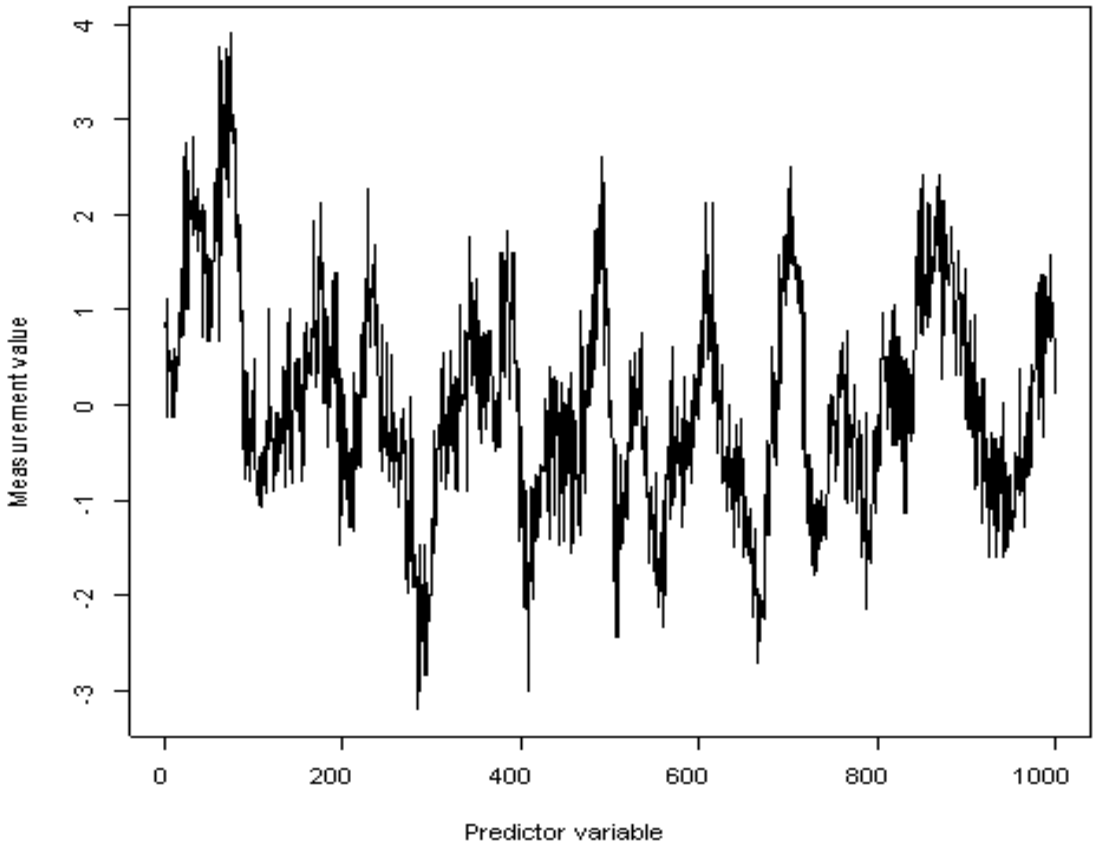

Figure 12: Predictor variable values for a typical observation in the smooth predictor variable regression problem.

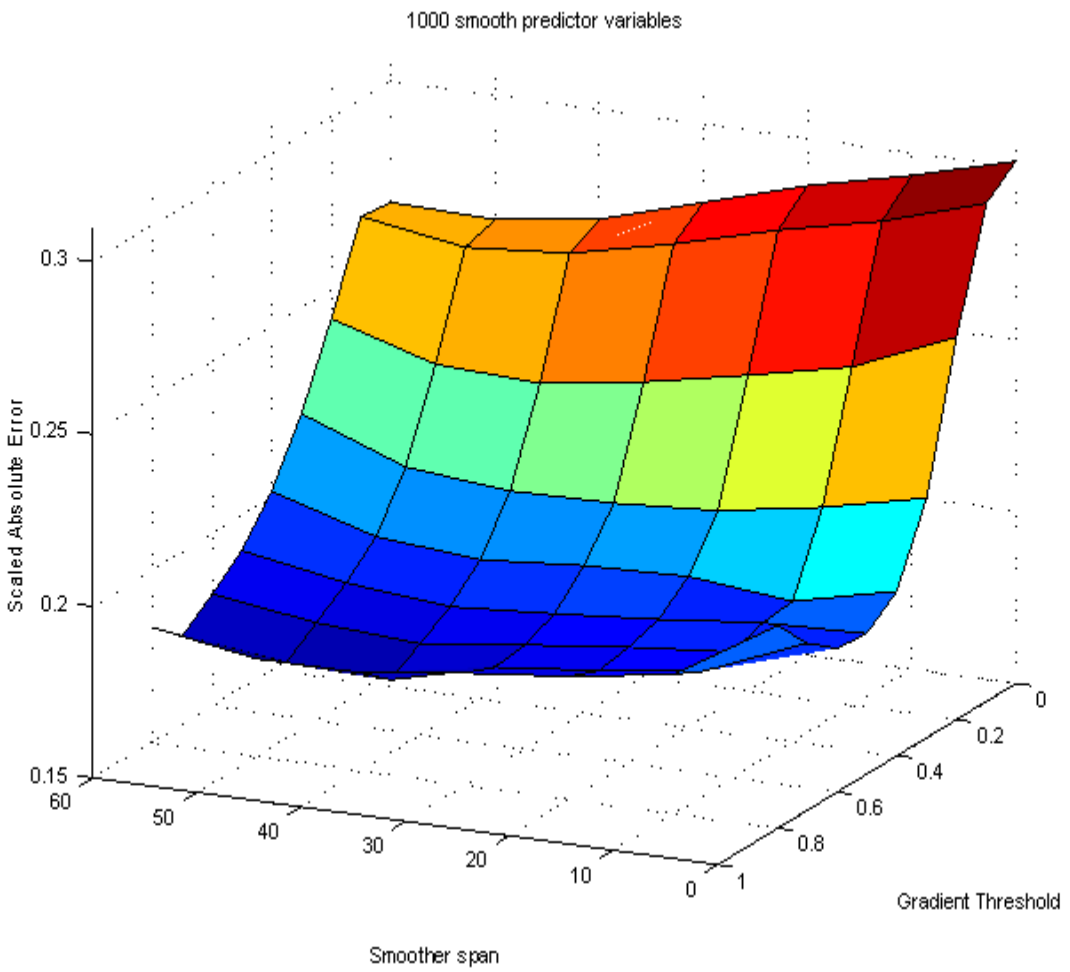

Figure 13: Expected scaled absolute error as a joint function of (τ, K), for the smooth coefficient regression problem with *smooth* predictor variables. Note the different orientation from Fig. 11, and that the region $0.9 \leq \tau \leq 1.0$ is not shown, to reveal the minimizing region. The qualitative aspects are similar to those seen in Fig. 11, but quantitatively the beneficial effects of smoothing $(K > 1)$ are less dramatic here.

Fig. 12 displays the corresponding predictor variables $\{x_{ij}\}_{j=1}^{n}$ for the first observation of the first replication.

Figure 13 shows results analogous to those of Fig. 11 for this smooth predictor variable problem. Note that plot orientation is different, and that results for $0.9 \leq \tau \leq 1.0$ have been omitted, so that the minimizing region is visible. At all smoother span values K, prediction error for $0.9 \leq \tau \leq 1.0$ is considerably higher than that for smaller τ values nearer the minimum. Also note the difference in vertical location and scale.

The qualitative aspects of the dependence of scaled absolute error (32) on smoother span K and gradient threshold τ are seen to be similar to that for the case of no predictor variable smoothness (Fig. 11). For all values of τ error is minimized at smoother span $K \backsimeq 40$. For all smoother span values the minimizing gradient threshold value is near $\tau \backsimeq 0.7$, which is slightly higher than that seen in Fig. 11. Combinations of high joint values for both parameters ($\tau \geq 0.9$, $K \geq 40$, not shown) result in especially poor performance, as suggested above.

The major difference between these two examples is quantitative; for each value of gradient

threshold τ the improvement resulting from gradient smoothing is considerably less in the presence of smoother predictor variables. Instead of reductions of over a factor of four as in the non smooth case, reductions of around 10% to 20% are typical here. Although such improvements are not negligible, they are far from dramatic with respect to the performance of threshold gradient descent without smoothing. The results from these examples, as well as the discussion above (70), suggest that imposing a smoothness constraint on the coefficient estimates $\{\hat{a}_j\}_1^n$ is likely to be highly beneficial only in settings where the degree of smoothness of the predictor variables is considerably less than that of the optimal coefficients $\{a_j^*\}_1^n$ (4). This was the case for the example shown in Fig. 11. For the present example (Fig. 13) the smoothness of the predictor variables (71) (72) (73) is comparable to that of the optimal coefficients (69) and gradient smoothing (67) provides less improvement.

6.2 Non negative threshold gradient descent

In some applications one may know (or suspect) that the signs of certain coefficients are non negative $a_j^* \geq 0$. To the extent that this presumption holds, constraining the corresponding solution coefficients accordingly $\hat{a}_j \geq 0$ has the potential to improve prediction. Note that this mechanism can also be used to constrain selected solution coefficients to be non positive $\hat{a}_j \leq 0$ by simply changing the signs of the corresponding predictor variables $x_j \leftarrow -x_j$.

As with the smoothness constraint (Section 6.1) imposing a non negative constraint on some (or all) of the coefficients is especially straight forward in the context of threshold gradient descent. In this setting the produced paths should be encouraged to visit points $\hat{\mathbf{a}}(\nu)$ that observe the non negative constraints on the selected coefficients. That is each step (22) should be as close as possible to that produced by the unconstrained prescription (15) (25) (26) while observing the specified constraints.

Let $C_j(a_j)$ be a constraint indicator function for each coefficient a_j,

$$C_j(a_j) = \begin{cases} 1 & \text{if } a_j \geq 0 \text{ or } a_j \text{ is to be unconstrained} \\ 0 & \text{otherwise.} \end{cases} \tag{74}$$

Define an "eligible" set of indices $H(\nu)$ at each point ν along the path as

$$H(\nu) = \{j : C_j(\hat{a}_j(\nu) + \Delta\nu \cdot g_j(\nu)) = 1\}. \tag{75}$$

That is, $H(\nu)$ labels the set of coefficients for which the constraints are not violated by the next gradient descent step. Allowing only these coefficients $\{\hat{a}_j(\nu) : j \in H(\nu)\}$ to be updated, while setting the values of those in the complement set to zero $\{\hat{a}_j(\nu) = 0 : j \notin H(\nu)\}$, produces a steepest descent step at each point ν along the path under the constraint restrictions. A threshold gradient descent step is produced by applying the thresholding strategy (25) (26) to this restricted set of variables. That is,

$$f_j(\nu) = I\left[\, j \in H(\nu) \;\&\; |\, g_j(\nu)\,| \geq \tau \cdot \max_{k \in H(\nu)} |\, g_k(\nu)\,|\right] \tag{76}$$

replaces (26) in the threshold gradient descent strategy. All other aspects of the procedure remain unaltered including the fast algorithms described in Section 5, and even the smoothing strategy of Section 6.1.

This approach can be generalized to apply to other types of inequality constraints on the coefficient values as well. One defines the appropriate constraint indicators analogous to (74), producing a corresponding set of eligible indices $H(\nu)$ (75) to be used for thresholding and updating (76) at each step. The coefficients $\hat{a}_j(\nu)$, $j \notin H(\nu)$, are set to their closest corresponding constraint boundary values.

6.2.1 Illustration

For the latent variable model described in Section 4 the optimal coefficients (4) are all non negative $\{a_j^* \geq 0\}_1^n$. The first $n_s = 100$ (active) predictor variables have positive coefficients

while the remaining $n - n_s$ coefficients have zero value. Figure 14 compares the results of applying non negative threshold gradient descent, to those of the unconstrained algorithm (Fig. 3), for various gradient threshold values $\tau \in [0, 1]$ in the same three situations corresponding to $n = 100$ (lower green), $n = 1000$ (middle blue), and $n = 10000$ (upper red) total predictor variables. In Fig. 14 each situation is represented by two sets of points connected by straight lines. The upper set for each one is the same as that of the unconstrained algorithm shown in Fig. 3. The lower set of points represent the corresponding scaled average absolute error (32) using non negative threshold gradient descent.

For $n = 100$ predictors (no pure noise variables) the results are seen to be nearly identical at all gradient threshold values. The non negative constraint produces no improvement in this setting where all optimal coefficient values are not close to zero. In the presence of large numbers of irrelevant noise variables with small (here zero) valued optimal coefficients ($n = 1000, 10000$) substantial improvement is seen. This improvement is greatest for smaller gradient threshold values where the unconstrained procedure tends to produce non negligible negative coefficient estimates for more of the irrelevant variables, as seen for example in Fig. 5.

7 Proteomics data

In this section we present the results of applying the threshold gradient descent pathfinding strategy to a data set from proteomics. These data consist of surfaced–enhanced laser desorption ionization time–of–flight (SELDI–TOF) mass spectra of microdissected prostate cells, obtained from the NCI/CCR and FDA/CBER data bank, http://ncifdaproteomics.com/download-prost.php (Petricoin *et al* 2002). Each observation (spectrum) is composed of peak amplitude measurements at 15,154 points defined by a corresponding m/z value. The spectra are measured on 259 samples taken from men with serum prostate–specific antigen (PSA) levels $\geq$ 4ng/mL. PSA level is commonly used to determine the need for prostate biopsy in asymptomatic men.

In this data set there are 69 samples that subsequent biopsies revealed to have prostate cancer and 190 that revealed benign conditions. The goal is to use the recorded mass spectra to differentiate these conditions in the absence of biopsy information. From a statistical perspective this can be viewed as a binary classification problem in which the outcome (response) variable y is the presence ($y = 1$) or absence ($y = -1$) of cancer, and the predictor variables $\mathbf{x}$ are the amplitudes of the 15,154 spectral lines. Here threshold gradient descent is applied to these data to produce a linear model (2) representing a score reflecting confidence that $y = 1$, given the measured amplitudes. This score is then thresholded (52) to produce a prediction ($\hat{y} = \pm 1$), with $l_+/l_- = 3$.

In order to guard against possible unusually extreme amplitude measurements (outliers) each of the predictor variable values x_{ij} was "Winsorized"

$$x_{ij} \leftarrow \max(\eta_j^{(-)}, \min(\eta_j^{(+)}, x_{ij}))$$

where $\eta_j^{(\pm)}$ are the $1 - \delta$ and δ quantiles respectively of the original measurements $\{x_{ij}\}_{i=1}^{259}$; here $\delta = 0.05$. The observations were weighted so that each of the two classes had equal total mass. Three–fold cross–validation was used for model selection (Sections 1.2 and 4.2). In order to obtain unbiased estimates of the error rates an outer (double) ten–fold cross–validation was applied to the whole process.

Table 1 shows for several gradient threshold τ values (column 1), the estimated number of errors of each type: benign classified as cancer (column 2) and cancer classified as benign (column 3). Column 4 shows the number of coefficients estimated to be non zero at each threshold value. The last row ("est") and column ("times") of Table 1 are discussed below.

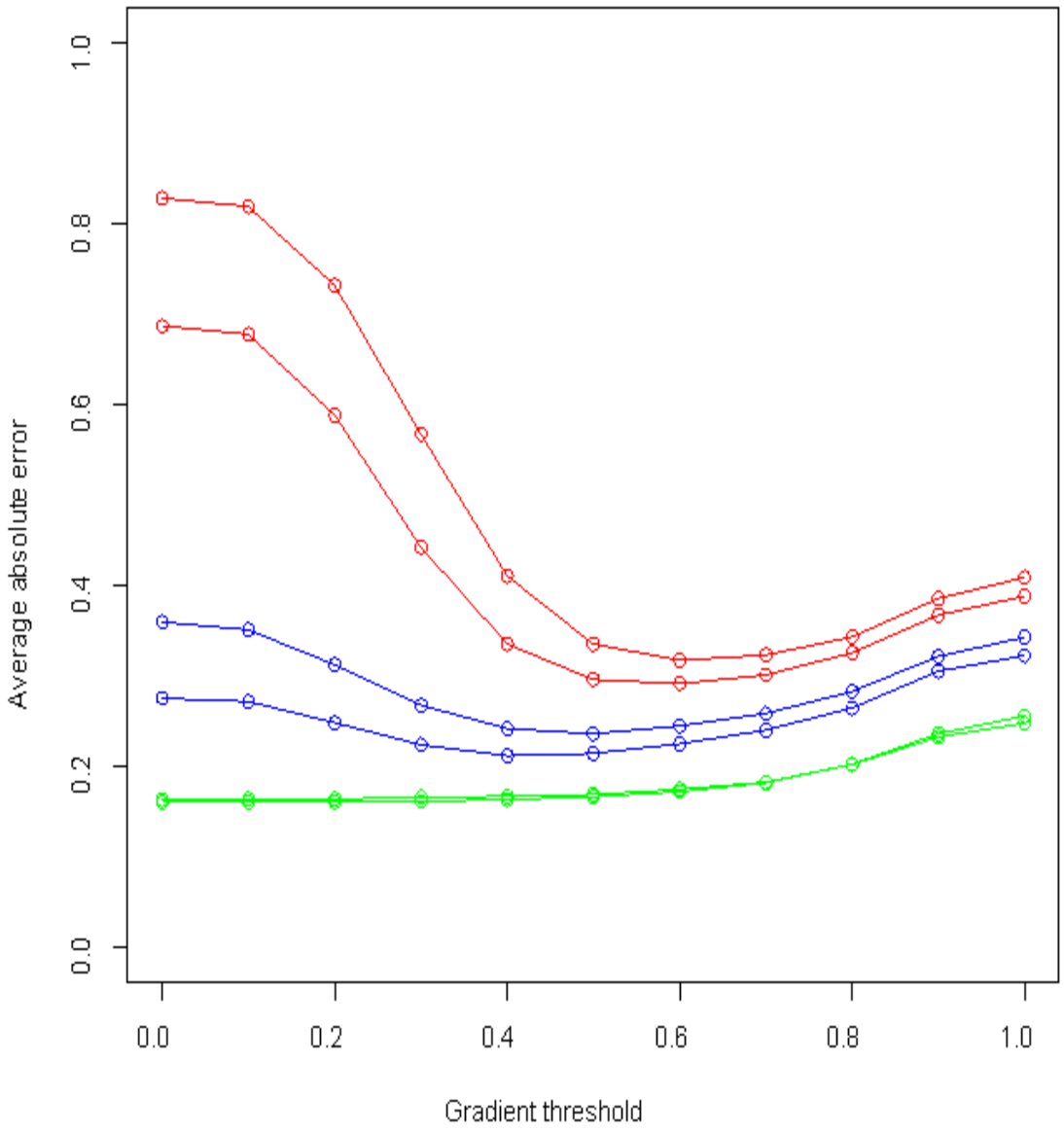

Figure 14: Comparison of non negative to ordinary threshold gradient descent for the latent variable regression problem with $n = 100$ (green), 1000 (blue), and 10000 (red) predictor variables. The upper curve for each setting is the expected scaled absolute error of the unconstrained solutions from Fig. 3. The respective lower curves are the corresponding results for the non negative constrained solutions. Applying the non negative constraints is most beneficial in the presence of many irrelevant predictor variables with small population optimal coefficient values.

Table 1
Errors as a function of gradient threshold value τ for the proteomics prostate cancer data.

τ	benign	cancer	coeffs	times
1.0	10	3	39	0
0.8	11	3	151	0
0.6	8	3	546	3
0.4	7	3	4347	5
0.2	13	4	9372	2
0.0	13	6	15154	0
est	10	3		

Although with such small sample sizes (and error rates) statistically significant differences are difficult to obtain, the pattern reflected in Table 1 is similar to that shown in Fig. 9. The lowest numbers of errors are obtained for threshold values $0.4 \leq \tau \leq 0.6$ with benign errors increasing for larger values and both types of errors increasing for smaller values. As expected, larger values of τ give rise to fewer non zero coefficient values, with only 39 out of the 15,154 spectral lines having any influence for $\tau = 1.0$.

Choosing the minimum error result ($\tau = 0.4$) as reflecting that of the procedure is somewhat over optimistic since this implicitly optimizes over the (outer) ten–fold cross–validation results. A more honest estimate is obtained by performing total model selection (Section 4.2) to estimate a threshold value within each of the ten outer cross–validation trials, thereby reflecting the actual application of the procedure. The corresponding results are shown on the last row in Table 1. One sees a slight increase in the number of benign errors over that of the $\tau = 0.4$ result, again in similarity with the horizontal lines shown in Fig. 9. The last column of Table 1 shows the number of times each respective threshold value was chosen over the ten outer cross–validation trials. These are see to concentrate in the interval $0.4 \leq \tau \leq 0.6$.

Figure 15 shows plots of the solution coefficients for several gradient threshold values as a function of variable (spectral line) number. All solutions put substantial absolute coefficient values in similar regions, with higher values of τ selecting fewer variables within each such region. Note the different vertical scales on each plot. As fewer variables have influence, the respective contributions of those that do correspondingly increase. Although the respective solutions involve quite different numbers of influential spectral lines, the resulting error rates are surprisingly similar. As discussed in Section 5.3.2, this may reflect the insensitivity of error rate to the specifics of the estimated model.

The results obtained here suggest that these mass spectra contain considerable information concerning the cancerous state of the corresponding samples that can be extracted using statistical classification procedures. The simple linear model (2) estimated by threshold gradient descent has both estimated sensitivity and specificity of around 95%, although these estimates are fairly uncertain owing to the small numbers of counts involved. It is possible (but not certain) that more complex procedures involving nonlinear structural models might produce even better results. In any case, such statistical classification methods may prove valuable as potential secondary screens for men with elevated PSA levels.

8 Discussion

The first example of regularization by direct path generation appears to be PLS (Wold 1975). Although it was originally motivated and derived from a very different perspective, its equivalence to conjugate gradient descent with squared–error loss was established by Wold *et al* 1984. Its similarity to ridge–regression was discussed in Frank and Friedman 1993. Regularization by direct gradient descent with early stopping is a commonly used technique in neural network training (see Bishop 1995). The generalizations discussed in this paper can be applied in that setting as well. The similarity of the incremental stagewise approach (threshold gradient descent

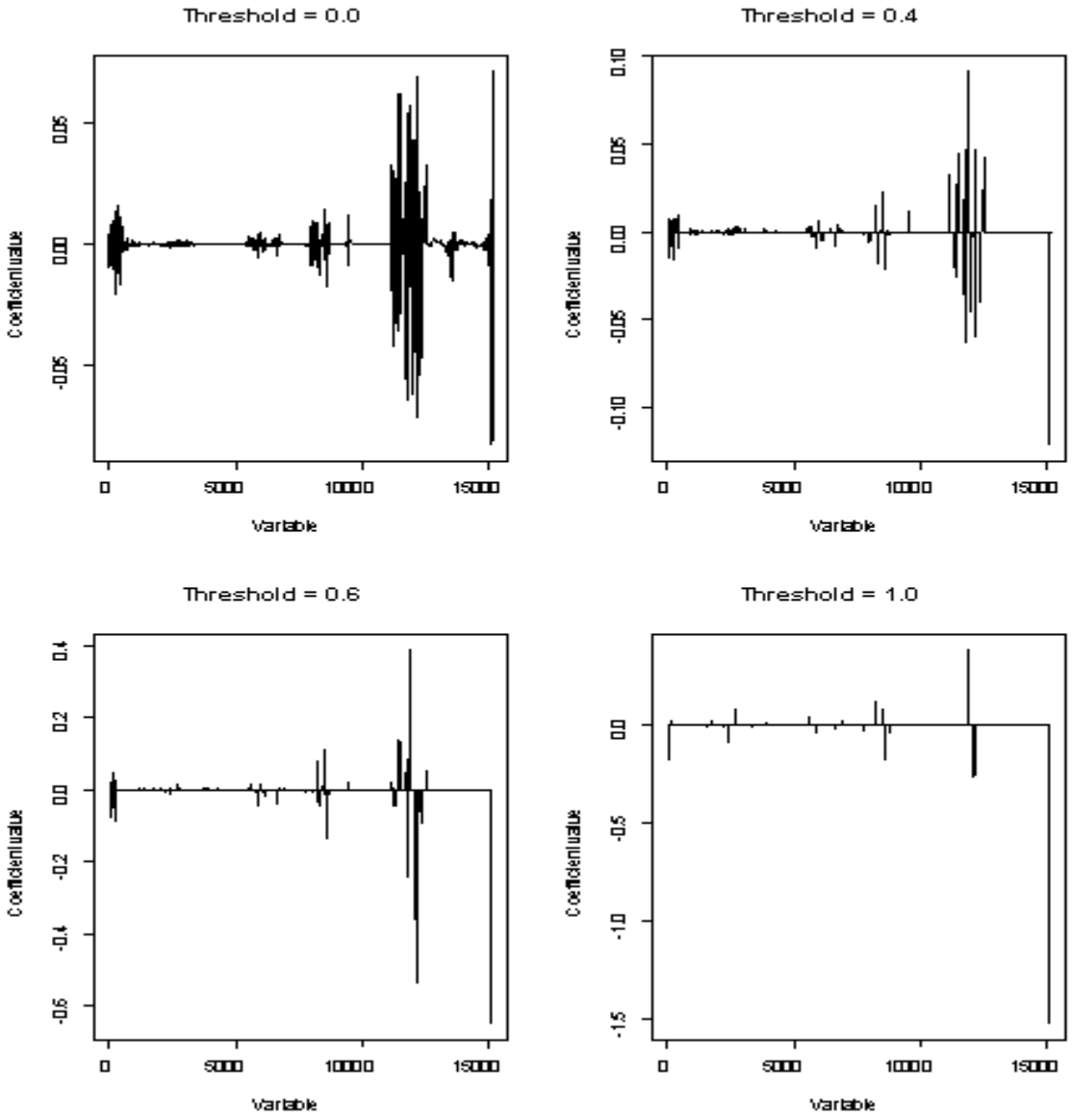

Figure 15: Coefficient values obtained for the proteomics data using threshold gradient descent with threshold values $\tau = 0$ (upper left), $\tau = 0.4$ (upper right), $\tau = 0.6$ (lower left), and $\tau = 1$ (lower right). All solutions produce relatively large absolute coefficient values in similar regions of spectral line location, with larger values of τ selecting fewer non zero values within each one. The estimated optimal solution is for $\tau = 0.4$.

with $\tau = 1$) to the lasso was first suggested by Hastie, Tibshirani and Friedman 2001. This was more rigorously established in Efron *et al* 2003.

Penalization methods with squared–error loss for applying smoothness constraints to the solution coefficient values have been studied by many authors (see for example Hastie and Mallows 1993). The one with the goal closest to that of the threshold gradient descent technique is the "fused lasso" (Tibshirani, Saunders, Rosset and Zhu 2003) based on squared–error loss, where the penalty is taken to be a convex combination of the lasso penalty and one that penalizes successive differences in the coefficient values. Zou and Hastie 2003 describe a squared–error loss procedure using a convex combination of the ridge and lasso penalties.

The distinguishing characteristics of the threshold gradient descent approach are generality, robustness and speed. The straight forward implementation (Section 5.4) can be used with any differentiable loss criterion and represents a fast algorithm for $n >> N$ and/or smaller values of τ. Sections 5.1–5.3 develop fast algorithms for several useful loss criteria that are appropriate for $N \gtrsim n$ and/or larger values of τ. Together these algorithms permit applications to large problems for which there are hundreds of variables and tens of thousands of observations on the one hand (see Friedman and Popescu 2003), and those that involve tens of thousands of predictor variables and hundreds of observations at the other extreme. For example, the proteomics problem in Section 7 ($n = 15154$, $N = 259$), using 19 cross–validation runs of the procedure for total model selection, took a total of 2.5 minutes to execute on an Athlon 64 2.2 Ghz computer. The algorithms are numerically stable as well requiring no manipulations of matrix inverses, or in fact no linear algebra at all beyond the evaluation of simple inner products.

Further generality is achieved by the ease of applying user specified constraints on the solution coefficient values, as described in Section 6. Such constraints can improve estimation and prediction accuracy in the presence of appropriate domain knowledge. Application of these constraints involves little additional computation beyond that required by the unconstrained threshold gradient descent algorithms.

9 Acknowledgements

The work of Jerome Friedman was partially supported by the Department of Energy under contract DE-AC03-76SF00515 and the National Science Foundation under grant DMS-97-64431. The work of Bogdan Popescu was partially supported by the National Science Foundation under grant DMS-97-64431.

References

[1] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

[2] Donoho, D., Johnstone, I. (1993). Ideal spatial adaptation by wavelet shrinkage. *Biometrika* **81**, 425-455.

[3] Donoho, D., Johnstone, I., Kerkuacharian, G. and Picard, D. (1995). Wavelet shrinkage; asymptotya? (with discussion). *J. Royal. Statist. Soc.* **57**, 201-337.

[4] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2003). Least angle regression (with discussion). *Annals of Statistics*. To appear.

[5] Frank, I. and Friedman, J.H. (1993). A statistical view of some chemometrics regression tools (with discussion), *Technometrics* **35**(2), 109-148.

[6] Freund, Y. and Schapire, R.E. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kauffmann, San Francisco, 148-156.

[7] Friedman, J. H., and Popescu, B. E. (2003). Importance sampled learning ensembles. *Stanford University, Department of Statistics.* Technical Report.

[8] Gill, P. E., Murray, W., and Wright, M. (1981). *Practical Optimization.* Academic Press, Inc.

[9] Hastie, T. and Mallows, C. (1993). Discussion of Frank and Friedman. *Technometrics* **35**(2), 140-143.

[10] Hastie, T., Tibshirani, R. and Friedman, J.H. (2001). *Elements of Statistical Learning.* Springer-Verlag.

[11] Huber, P. (1964). Robust estimation of a location parameter. *Annals of Math. Stat.* **53**, 73-101.

[12] Hoerl, A. E. and Kennard, R. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* **12**, 55-67.

[13] Joachims, T. (1999) Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.). MIT-Press, 1999.

[14] Petricoin, E. F. III, Ornstein, D. K, Paweletz, C. P., Ardekani, A., Hackett, P. S., Hitt, B. A., Velassco, A., Trucco, C., Wiegand, L., Wood, K., Simone, C. B., Levine, P. J., Linehan, W. M., Emmert–Buck, M. R., Steinberg, S. M., Kohn, E. C., and Liotta, L. A. (2002). Serum proteomic patterns for detection of prostate cancer. *J. Nat. Cancer Inst.* **94**, 1576-1578.

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

[16] Tibshirani, R., Saunders, M., Rosset, R., and Zhu, J. (2003). Sparsity and smoothness via the fused lasso. *Stanford University, Department of Statistics.* Technical Report.

[17] Vapnik, V. (1996). *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.

[18] Wold, H. (1975). Soft modeling by latent variables: the non-linear iterative partial least squares approach. *Perspectives in Probability and Statistics, Papers in Honour of M.S. Bartlett,* ed. J. Gani. Academic Press, London. 117-144.

[19] Wold, S., Ruhe, A., Wold H. and W. J. Dunn III (1984). The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. *SIAM Journal of Scientific and Statistical Computing,* **5**(3) 58, 735-742.

[20] Zhang, T. (2003). Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics.* To appear.

[21] Zou, J. and Hastie, T. (2003). Regression shrinkage and selection via the elastic net, with application to microarrays. *Stanford University, Department of Statistics.* Technical Report.