Lecture 17
Intro to Lasso Regression

11 November 2015

Taylor B. Arnold
Yale Statistics
STAT 312/612

Notes

- problem set 5 posted; due today

Goals for today

- introduction to lasso regression
- the subdifferential
- basics of the lars algorithm

Lasso Regression

Two lectures ago, I introduced the ridge estimator:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2 + \lambda ||b||_2^2 \right\}$$

Two lectures ago, I introduced the ridge estimator:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2 + \lambda ||b||_2^2 \right\}$$

Notice that for any $\lambda > 0$ there exists an s_λ equal to $||\widehat{\beta}_\lambda||_2^2$ where:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2, \quad \text{s.t.} \quad ||b||_2^2 \leq s_\lambda \right\}$$

Two lectures ago, I introduced the ridge estimator:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2 + \lambda ||b||_2^2 \right\}$$

Notice that for any $\lambda > 0$ there exists an s_λ equal to $||\widehat{\beta}_\lambda||_2^2$ where:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2, \quad \text{s.t.} \quad ||b||_2^2 \leq s_\lambda \right\}$$

This is the dual form of the optimization problem (essentially, the opposite of using Lagrangian multipliers).

Lasso regression replaces the ℓ_2 penalty with an ℓ_1 penalty, and looks deceptively similar to the ridge regression:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2 + \lambda ||b||_1 \right\}$$

Where the ℓ_1-norm is defined as the sum of the absolute values of the vector's components:

$$||\beta||_1 = \sum_i |\beta_i|$$

Lasso regression replaces the ℓ_2 penalty with an ℓ_1 penalty, and looks deceptively similar to the ridge regression:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2 + \lambda ||b||_1 \right\}$$

Where the ℓ_1-norm is defined as the sum of the absolute values of the vector's components:

$$||\beta||_1 = \sum_i |\beta_i|$$

Similarly, for any $\lambda > 0$ there exists an s_λ equal to $||\widehat{\beta}_\lambda||_1$ where:

$$\widehat{\beta}_\lambda = \arg\min_b \left\{ ||y - Xb||_2^2, \quad \text{s.t.} \quad ||b||_1 \leq s_\lambda \right\}$$

The change in the norm of the penalty may seem like only a minor difference, however the behavior of the ℓ_1-norm is significantly different than that of the ℓ_2-norm.

The change in the norm of the penalty may seem like only a minor difference, however the behavior of the ℓ_1-norm is significantly different than that of the ℓ_2-norm.

A classic illustration, with $p = 2$ and using the dual form of the estimators, shows exactly how this comes about.

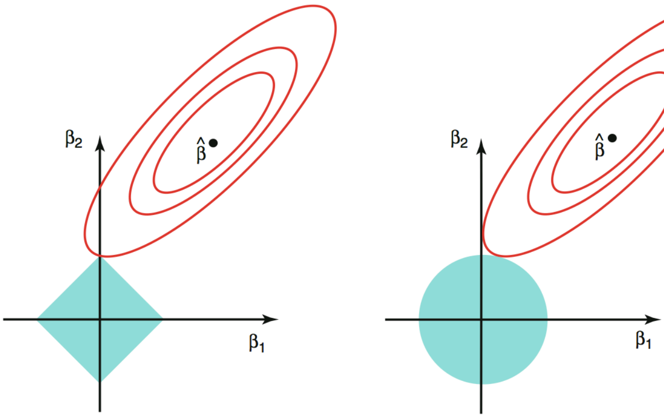


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.*

I think of there being three major difference between ridge and lasso:

1. the sharp, non-differentiable corners of the ℓ_1-ball produce parsimonious models for sufficiently large values of λ

I think of there being three major difference between ridge and lasso:

1. the sharp, non-differentiable corners of the ℓ_1-ball produce parsimonious models for sufficiently large values of λ
2. the lack of rotational invariance limits the use of the singular value theory that is so useful in unpenalized and ridge regression

I think of there being three major difference between ridge and lasso:

1. the sharp, non-differentiable corners of the ℓ_1-ball produce parsimonious models for sufficiently large values of λ
2. the lack of rotational invariance limits the use of the singular value theory that is so useful in unpenalized and ridge regression
3. the lasso lacks an analytic solution, making both computation and theoretical results more difficult

At the start of this semester, most of you had already worked with multivariate regression, and many had even seen ridge, PCR, and lasso regression.

At the start of this semester, most of you had already worked with multivariate regression, and many had even seen ridge, PCR, and lasso regression.

By looking at these techniques at a deeper level with the help of matrix analysis you have (hopefully) seen that there is a substantial amount of nuance that is not immediately obvious.

At the start of this semester, most of you had already worked with multivariate regression, and many had even seen ridge, PCR, and lasso regression.

By looking at these techniques at a deeper level with the help of matrix analysis you have (hopefully) seen that there is a substantial amount of nuance that is not immediately obvious.

The lasso is very much the same (in that there is significant depth beyond what you would see in a first pass), but the lack of differentiability and rotational invariance make most matrix methods insufficient. Instead we will be looking to *convex analysis* for tools of study to use in the remainder of the semester.

In slight reverse of the other techniques, we are going to study the lasso in the following order:

1. computation
2. application
3. theory

With additional computational considerations to follow, time permitting.

Ordinary least squares and ridge regression have what are called *analytic solutions*, we can write down an explicit formula for what the estimators are.

Ordinary least squares and ridge regression have what are called *analytic solutions*, we can write down an explicit formula for what the estimators are.

Generalized linear models, on the other hand, have only *numerical solutions* with iterative methods. We have algorithm that, when run sufficiently long enough should yield a solution with a reasonable accuracy.

Ordinary least squares and ridge regression have what are called *analytic solutions*, we can write down an explicit formula for what the estimators are.

Generalized linear models, on the other hand, have only *numerical solutions* with iterative methods. We have algorithm that, when run sufficiently long enough should yield a solution with a reasonable accuracy.

The lasso falls somewhere in-between these two cases, as it has a *direct numerical solution.* We cannot write out an explicit analytic form, but the algorithm is not iterative and would yield the exact solution given a machine with infinite precision.

The subdifferential

The lasso solution does not have a derivative for any point where β_j is equal to zero. We instead use the concept of a *subdifferential*.

The lasso solution does not have a derivative for any point where β_j is equal to zero. We instead use the concept of a *subdifferential.*

Let h be a convex function. The subdifferential at point x_0 in the domain of h is equal to the set:

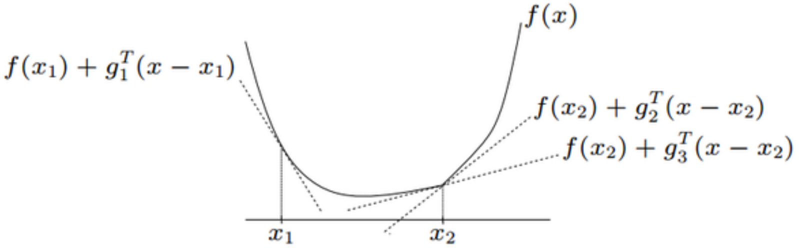
$$\partial(h)(x_0) = \left\{ c \in \mathbb{R} \quad \text{s.t.} \quad c \leq \frac{h(x) - h(x_0)}{x - x_0} \quad \forall \quad x \in \text{Dom}(h) \right\}$$

Less formally, it is the set of all slopes which are tangent to the function at the point x_0.

For example, the subdifferential of the absolute value function is

$$\partial(|\cdot|)(x) = \begin{cases} -1, & x < 0 \\ [-1, 1], & x = 0 \\ 1, & x > 0 \end{cases}$$

An illustration of elements of the subdifferential for a more complex convex function:



$f(x)$
$f(x_1) + g_1^T(x - x_1)$
$f(x_2) + g_2^T(x - x_2)$
$f(x_2) + g_3^T(x - x_2)$
x_1
x_2

As shown in the figure, the subdifferential can be generalized to higher dimensions as follows:

$$\partial(h)(x_0) = \left\{ g \in \mathbb{R}^p \quad \text{s.t.} \quad h(x_0) \geq h(x) + g^t(x - x_0) \quad \forall \quad x \in \text{Dom}(h) \right\}$$

We will use three properties of the subdifferential of a convex function h:

1. the gradient $\nabla(h)$ exists at the point x_0 if and only if $\partial(h)(x_0)$ is equal to a single value, which is equal to $\nabla(h)(x_0)$

We will use three properties of the subdifferential of a convex function h:

1. the gradient $\nabla(h)$ exists at the point x_0 if and only if $\partial(h)(x_0)$ is equal to a single value, which is equal to $\nabla(h)(x_0)$
2. for every point x_0, the set $\nabla(h)(x_0)$ is a nonempty closed interval $[a, b]$

We will use three properties of the subdifferential of a convex function h:

1. the gradient $\nabla(h)$ exists at the point x_0 if and only if $\partial(h)(x_0)$ is equal to a single value, which is equal to $\nabla(h)(x_0)$
2. for every point x_0, the set $\nabla(h)(x_0)$ is a nonempty closed interval $[a, b]$
3. the point x_0 is a global minimum of h if and only if the subdifferential contains zero; in other words, $0 \in \partial(h)(x_0)$

Least angle regression

The algorithm for solving the exact form of the lasso comes from

> *Efron, Bradley, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. "Least angle regression." The Annals of Statistics 32, no. 2 (2004): 407-499.*

The algorithm for solving the exact form of the lasso comes from

> *Efron, Bradley, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. "Least angle regression." The Annals of Statistics 32, no. 2 (2004): 407-499.*

Today, I will sketch out how the algorithm works in the simple case where X^tX is the identity matrix. Problem set 6 illustrates how this works in the more general case.

The lasso equation is a convex function, so let us try to calculate the subdifferential and determine what solution β give us $\partial(h)(\beta)$ that contain 0.

To simplify matters, assume that $X^t X$ is equal to the identity matrix. We will also add a factor of 2 to the penalty, so that it because $2\lambda||\beta||_1$.

The lasso equation, in our simpler case, is given by:

$$\begin{aligned} f(b) &= ||y - Xb||_2^2 + 2\lambda||b||_1 \\ &= y^t y + b^t b - 2y^t Xb + 2\lambda||b||_1 \end{aligned}$$

The lasso equation, in our simpler case, is given by:

$$\begin{aligned} f(b) &= ||y - Xb||_2^2 + 2\lambda||b||_1 \\ &= y^t y + b^t b - 2y^t Xb + 2\lambda||b||_1 \end{aligned}$$

We know that the gradient of the front terms is just $2b - 2X^t y$, and the subdifferential of the ℓ_1 norm is equal to λ times that of the absolute value function.

The *j*-th component of the subdifferential is then given by:

$$\partial(h)(b_j) = \begin{cases} 2b_j - 2x_j^t y + 2\lambda, & b_j > 0 \\ [-2\lambda, 2\lambda] - 2x_j^t y, & b_j = 0 \\ 2b_j - 2x_j^t y - 2\lambda, & b_j < 0 \end{cases}$$

So what would make these equal to zero for all *j*?

If b_j is greater than zero, we need the following to hold:

$$\begin{aligned} 2b_j - 2x_j^t y + 2\lambda &= 0 \\ b_j &= x_j^t y - \lambda \end{aligned}$$

If b_j is greater than zero, we need the following to hold:

$$\begin{aligned} 2b_j - 2x_j^t y + 2\lambda &= 0 \\ b_j &= x_j^t y - \lambda \end{aligned}$$

So b_j is a linear function of λ, with an intercept of $x_j^t y$ and a slope of $-\lambda$.

If b_j is greater than zero, we need the following to hold:

$$\begin{aligned} 2b_j - 2x_j^t y + 2\lambda &= 0 \\ b_j &= x_j^t y - \lambda \end{aligned}$$

So b_j is a linear function of λ, with an intercept of $x_j^t y$ and a slope of $-\lambda$. However this only holds when:

$$\begin{aligned} x_j^t y - \lambda &> 0 \\ x_j^t y &> \lambda \end{aligned}$$

So, for all λ between $x_j^t y$ and 0. So clearly b_j can only be positive if $x_j^t y$ is positive.

What if b_j is less than zero? We instead need the following:

$$\begin{aligned} 2b_j - 2x_j^t y - 2\lambda &= 0 \\ b_j &= x_j^t y + \lambda \end{aligned}$$

Whenever:

$$\begin{aligned} x_j^t y + \lambda &< 0 \\ x_j^t y &< -\lambda \\ -x_j^t y &> \lambda \end{aligned}$$

As $\lambda > 0$, therefore, b_j is only negative if $x_j^t y$ is negative.

We can combine these two conditions together, by saying that whenever $\lambda < |x_j^t y|$ we have:

$$b_j = x_j^t y - \text{sign}(x_j^t y) \cdot \lambda$$

What about the third case in the subdifferential? If b_j is equal to zero, we need:

$$0 \in [-2\lambda, 2\lambda] - 2x_j^t y$$

Which implies that both

$$\begin{aligned} -2\lambda - 2x_j^t y &< 0 \\ \lambda &> -x_j^t y \end{aligned}$$

What about the third case in the subdifferential? If b_j is equal to zero, we need:

$$0 \in [-2\lambda, 2\lambda] - 2x_j^t y$$

Which implies that both

$$\begin{aligned} -2\lambda - 2x_j^t y &< 0 \\ \lambda &> -x_j^t y \end{aligned}$$

and

$$\begin{aligned} 2\lambda - 2x_j^t y &> 0 \\ \lambda &> x_j^t y \end{aligned}$$

hold.

What about the third case in the subdifferential? If b_j is equal to zero, we need:

$$0 \in [-2\lambda, 2\lambda] - 2x_j^t y$$

Which implies that both

$$\begin{aligned} -2\lambda - 2x_j^t y < 0 \\ \lambda > -x_j^t y \end{aligned}$$

and

$$\begin{aligned} 2\lambda - 2x_j^t y > 0 \\ \lambda > x_j^t y \end{aligned}$$

hold.

This can again be simplified into one equation, which here yields $\lambda > |x_j^t y|$ whenever b_j is equal to zero.

Now, amazingly, we have a full solution to the lasso equation in the case where $X^t X$ is the identity matrix:

$$\widehat{\beta}_j^{\lambda} = \begin{cases} 0, & \lambda > |x_j^t y| \\ x_j^t y - \operatorname{sign}(x_j^t y) \cdot \lambda, & \lambda \leq |x_j^t y| \end{cases}$$

Let's construct a test dataset, with $X^t X$ equal to the identity matrix. One easy way is with the `poly` function, which produces an orthogonal polynomial basis of an arbitrarily high order.

```
> n <- 1000
> p <- 5
> X <- poly(seq(0,1,length.out=n),degree=p)
> round(t(X) %*% X,6)
  1 2 3 4 5
1 1 0 0 0 0
2 0 1 0 0 0
3 0 0 1 0 0
4 0 0 0 1 0
5 0 0 0 0 1
> beta <- c(1,0,1,0,0)
> y <- X %*% beta + rnorm(n,sd=0.3)
```

Notice that the regression vector has only 2 non-zero components.

Now we start by constructing a sequence of λ values to which we want to fit the model. We know the non-zero solutions lie between 0 and $||X^t y||_\infty$, so we'll pick twice that range:

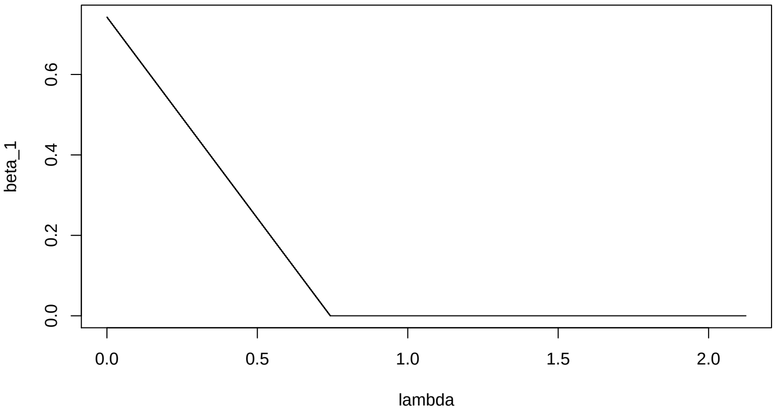
```
> Xty <- t(X) %*% y
> lambda <- seq(0,max(abs(Xty))*2,length.out=1e5)
```

The first element of $\widehat{\beta}$ can then be simply calculated according to our formula.

```
> j <- 1
> beta <- Xty[j] - lambda * sign(Xty[j])
> beta[lambda > abs(Xty[j])] <- 0
```

We then plot this as a *path* of solutions with respect to λ:

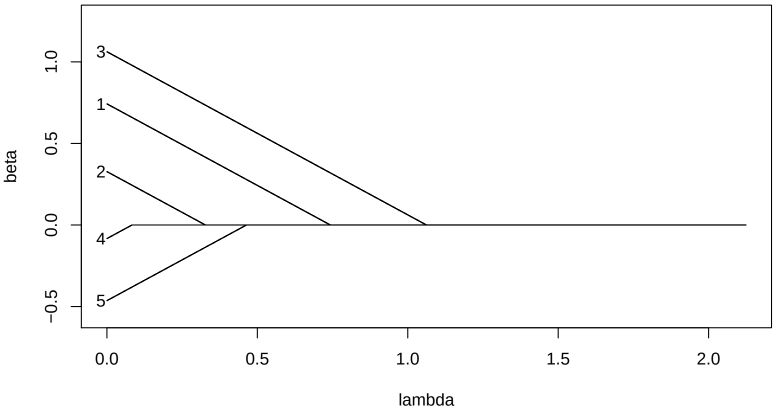
```
> plot(lambda, beta, type='l')
```


beta_1
0.0
0.2
0.4
0.6
0.0
0.5
1.0
1.5
2.0
lambda

Putting this logic into a loop and saving the results as a matrix, yields the full set of coordinates.

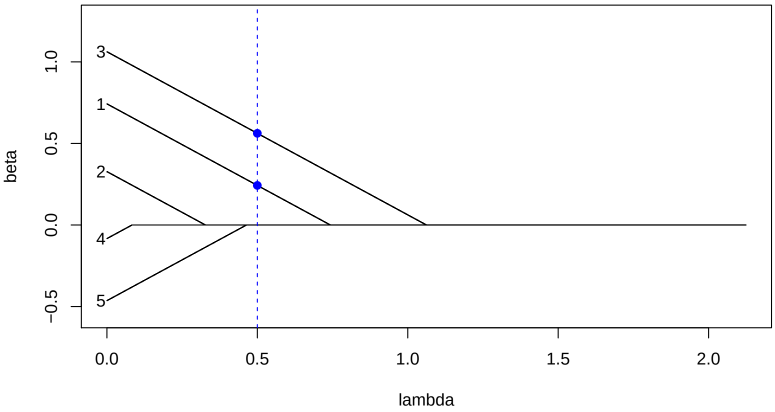
```
> beta <- matrix(0,nrow=length(lambda),ncol=p)
> for (j in 1:p) {
+   beta[,j] <- Xty[j] - lambda * sign(Xty[j])
+   beta[lambda > abs(Xty[j]),j] <- 0
> }
```

These can, similarly, be plotted as a function of λ.



3
1
2
4
5
beta
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5
2.0
lambda

Recall that only elements 1 and 3 of the original problem were actually non-zero. Look what happens if we set λ to 0.5.



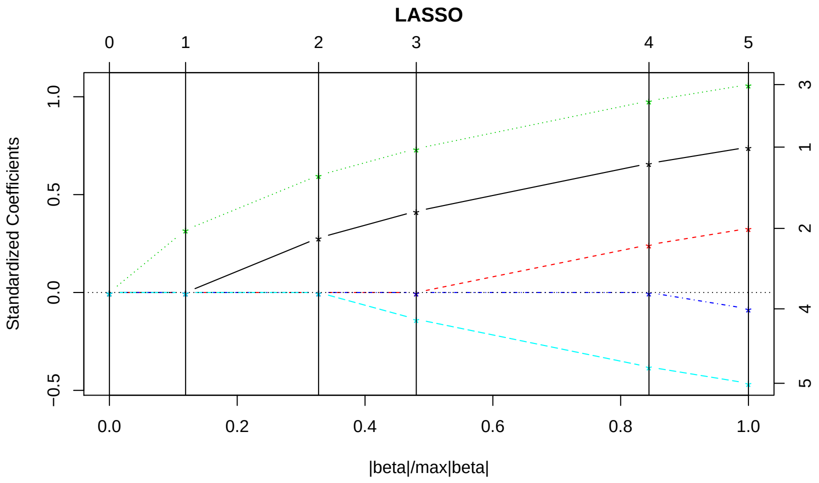
3
1
2
4
5
1.0
0.5
0.0
−0.5
beta
0.0
0.5
1.0
1.5
2.0
lambda

The `lars` package, written by the authors of the aforementioned paper, provides a method for calculating this without having to write the code manually (and obviously also handles cases with arbitrary X^tX).

```
> library(lars)
> out <- lars(X,y,normalize=FALSE,intercept=FALSE)
> out

Call:
lars(x = X, y = y, normalize = FALSE, intercept = FALSE)
R-squared: 0.021
Sequence of LASSO moves:
     3 1 5 2 4
Var  3 1 5 2 4
Step 1 2 3 4 5

> plot(out)
```


LASSO
0
1
2
3
4
5
Standardized Coefficients
1.0
0.5
0.0
−0.5
3
1
2
4
5
0.0
0.2
0.4
0.6
0.8
1.0
|beta|/max|beta|

Turning on the trace feature of lars shows the path like logic of the algorithm:

```
> out <- lars(X,y,normalize=FALSE,intercept=FALSE,
+              trace=TRUE)
LASSO sequence
Computing X'X .....
LARS Step 1 :  Variable 3   added
LARS Step 2 :  Variable 1   added
LARS Step 3 :  Variable 5   added
LARS Step 4 :  Variable 2   added
LARS Step 5 :  Variable 4   added
Computing residuals, RSS etc .....
```

What is so much more difficult about the general case? Consider the function to be minimized:

$$\begin{aligned} f(b) &= ||y - Xb||_2^2 + 2\lambda ||b||_1 \\ &= y^t y + b^t X^t X b - 2 y^t X b + 2\lambda ||b||_1 \end{aligned}$$

The subdifferential is still fairly easy to calculate, with the *j*-th component equal to:

$$\partial(h)(b) = \begin{cases} (2X^t X b - 2X^t y)_j + 2 \cdot \text{sign}(b_j) \cdot \lambda, & b_j \neq 0 \\ (2X^t X b - 2X^t y)_j + 2 \cdot [-\lambda, \lambda], & b_j = 0 \end{cases}$$

The subdifferential is still fairly easy to calculate, with the *j*-th component equal to:

$$\partial(h)(b) = \begin{cases} (2X^tXb - 2X^ty)_j + 2 \cdot \text{sign}(b_j) \cdot \lambda, & b_j \neq 0 \\ (2X^tXb - 2X^ty)_j + 2 \cdot [-\lambda, \lambda], & b_j = 0 \end{cases}$$

But consider how hard this is to solve directly now that the p equations are no longer decoupled given that (1) we need to consider all permutations of the whether b_j is zero and (2) we have a system of, possibly, multivalued equations.

The subdifferential is still fairly easy to calculate, with the *j*-th component equal to:

$$\partial(h)(b) = \begin{cases} (2X^tXb - 2X^ty)_j + 2 \cdot \text{sign}(b_j) \cdot \lambda, & b_j \neq 0 \\ (2X^tXb - 2X^ty)_j + 2 \cdot [-\lambda, \lambda], & b_j = 0 \end{cases}$$

But consider how hard this is to solve directly now that the p equations are no longer decoupled given that (1) we need to consider all permutations of the whether b_j is zero and (2) we have a system of, possibly, multivalued equations.

The good news is that once we find some $\widehat{\beta}_\lambda$ such that $\partial(h)(\widehat{\beta}_\lambda)$ contains the zero vector we are done. No need to calculate or prove anything; a global minimum is assured.

For example, consider any λ greater than $|X^t y|_\infty$ (the maximum absolute value). I know that $\widehat{\beta}_\lambda$ will be equal to all zeros, and can show this very quickly as the subdifferential becomes:

$$\partial(h)(b) = 2x_j^t y + 2 \cdot [-\lambda, \lambda], \qquad\qquad b_j = 0$$

For example, consider any λ greater than $|X^t y|_\infty$ (the maximum absolute value). I know that $\widehat{\beta}_\lambda$ will be equal to all zeros, and can show this very quickly as the subdifferential becomes:

$$\partial(h)(b) = 2x_j^t y + 2 \cdot [-\lambda, \lambda], \qquad\qquad b_j = 0$$

Which contains zero because:

$$-\lambda < x_j^t y$$

and

$$\lambda > -x_j^t y.$$

Now, assume that the values of $X^t y$ are unique, positive, and ordered from highest to lower values. These conditions simply keep the notation easier, with all but the uniqueness being easy to account for with a few more conditional statements.

It turns out that for λ between $x_1^t y$ and $x_2^t y$, we get the same solution as in the uncorrelated case (with the addition of scaling factor as $x_1^t x_1$ may not be 1):

$$\widehat{\beta}_\lambda = \frac{1}{x_1^t x_1} \cdot \begin{pmatrix} x_1^t y - \lambda \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

This can be shown fairly easily by plugging into the subdifferential equation.

For j not equal to 1, we will have:

$$0 \in 2x_j^t y + 2 \cdot [-\lambda, \lambda]$$

Because $\lambda > x_j^t y$, as the inner products are sorted from largest to smallest.

For the $j = 1$ case, we have:

$$\begin{aligned} 0 &= \left(2X^t X b - 2X^t y\right)_j + 2 \cdot \operatorname{sign}(b_j) \cdot \lambda \\ 0 &= 2x_1^t x_1 b_1 - 2x_1^t y + \lambda \\ b_1 &= \frac{1}{x_1^t x_1} \left(x_1^t y - \lambda\right) \end{aligned}$$

Which finishes the result.

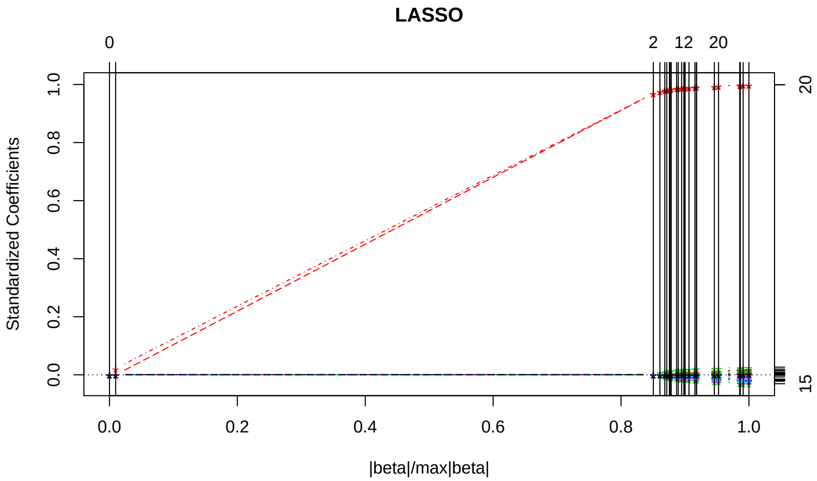
For values with $x_3^t y \leq \lambda \leq x_2^t y$, the solution for $\widehat{\beta}$ linear with non-zero elements in just the first 2 components.

For values with $x_3^t y \leq \lambda \leq x_2^t y$, the solution for $\widehat{\beta}$ linear with non-zero elements in just the first 2 components.

The details from there on out are what is covered on problem set 6.

As a final example, we'll construct some correlated data with a sparse model and show what the lars path solution looks like:

```
> n <- 1000
> p <- 25
> X <- matrix(rnorm(p*n),ncol=p)
> X <- X*0.8 + X[,1]*0.2
> beta <- sample(0:1,p,replace=TRUE,prob=c(9,1))
> which(beta != 0)
[1] 14 20
> y <- X %*% beta + rnorm(n,sd=0.3)
```


LASSO
0
2
12
20
Standardized Coefficients
0.0
0.2
0.4
0.6
0.8
1.0
20
15
|beta|/max|beta|