

Linear Models and Regularization

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

Rob McCulloch

1. Linear Regression
2. Linear Regression Review
3. Shrinkage-L2, Ridge Regression
4. Shrinkage-L1: The Lasso
5. Understanding the Lasso Solution
6. The Elastic Net
7. Subset Selection
8. Maximum Likelihood Estimation
9. AIC and BIC in Linear Regression
10. The Diabetes Data

1. Linear Regression

Part of the exciting part of modern ‘machine learning’ is a set of relatively new methods with the ability to fit complex relationships (*nonlinearity, interaction*) in high dimensions.

However,

the time honored linear model is still a major player:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \epsilon$$

Why?

In high dimensions (big p), it may be very hard to look for complex relationships.

The linear model may have *acceptable bias* and *low variance*.

We can also make the linear model more flexible by throwing in a lot of transformations of the original x 's.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

“Throw in” squares and cross product: \Rightarrow

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 + \epsilon$$

Example:

Start with $p = 10$. Throw in all squares and cross products.
Now have $10 + 10 + (10*9/2) = 65$ variables.

Also, a linear model, without too many transformations thrown in, may be more interpretable.

It may also be useful to have a simple mathematical representation of $f(x)$, because you may want to use it as one input to a more complex decision, in which case being able to manipulate it or easily optimize it may be important.

We can make regression interesting by having lots of x 's !!

However, if we just throw in a lot of x 's we could have a high variance overfit situation.

We will explore ways to *constrain the fit* so that we do not overfit.

Complex Model: Lot's of x 's, not very constrained.

Simpler Model: Lot's of x 's, constrained.

What does constrained mean??

Set some β_j to 0 and/or *shrink* some β_j towards 0.

This kind of shrinkage is known as **regularization**.

2. Linear Regression Review

In this section we present the linear Regression model in matrix form and review some its basic properties.

The linear model has a lot of nice simple properties. These properties will also figure in some more complex models (e.g. iteratively reweighted least squares).

Note that sometimes we will include an intercept in which case I may use p to mean the number of x 's+1 or sometimes I may use p for just the number of x 's.

Matrix notation for the linear model

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \varepsilon_i$$

$i = 1, 2, \dots, n$

$$x'_i = [1, x_{i1}, x_{i2}, \dots, x_{ip}] \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$
$$X = \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{bmatrix} \quad (n \times (p+1))$$

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$Y = X\beta + \varepsilon$$

Cut from R Hello world:

$$y = X\beta + \epsilon.$$

Let's check this.

```
X = cbind(rep(1,nrow(cd)),as.matrix(cd[,c(2,3)]))
head(X)
##           mileage year
## [1,] 1  36.858 2008
## [2,] 1  46.883 2012
## [3,] 1 108.759 2007
## [4,] 1  35.187 2007
## [5,] 1  48.153 2007
## [6,] 1 121.748 2002
```

We also write $X = [X_1, X_2, \dots, X_p]$ so that X_j is a column of n values for the j^{th} x .

X_1 may or may not be a column of 1's.

p can mean *either* the number of x variables or the number of columns.

So if an intercept is included, p could be number of x 's or number of x 's +1.

You can tell from the context and I don't want to write $(p + 1)$ half the time.

2 x's + intercept

$$x_i = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

$$x_i' \beta = \begin{bmatrix} 1 & x_{i1} & x_{i2} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$$

$$X = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix}$$

$$1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad X_1 = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{bmatrix} \quad X_2 = \begin{bmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{n2} \end{bmatrix}$$

$$X\beta = \begin{bmatrix} x_1' \beta \\ x_2' \beta \\ \vdots \\ x_i' \beta \\ \vdots \\ x_n' \beta \end{bmatrix}$$

$$X\beta = \begin{bmatrix} 1 & X_1 & X_2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \beta_0 1 + \beta_1 X_1 + \beta_2 X_2$$

Given training data, how do we estimate β ?

Minimize the in-sample mean-square-error (MSE) which we will denote by L for "loss".

$$\min_{\beta} \sum_{i=1}^n (y_i - x_i^T \beta)^2 = L(\beta)$$

To minimize we solve

$$\nabla L(\beta) = 0$$

$$\nabla L(\beta) = \left[\frac{\partial L}{\partial \beta_1}(\beta), \frac{\partial L}{\partial \beta_2}(\beta), \dots, \frac{\partial L}{\partial \beta_p}(\beta) \right]$$

To minimize we will set the gradient to 0.

Note that the gradient is a row vector.

Let's compute the gradient:

For a single observation (x, y) ,
the loss is

$$L(\beta) = (y - x^T \beta)^2$$

$$\frac{\partial L}{\partial \beta_j} = 2(y - x^T \beta) [-x_j]$$

(could be a^T)

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix}$$

Summing over observations:

$$L(\beta) = \sum_{i=1}^n (y_i - x_i^T \beta)^2 = \sum_{i=1}^n (y_i - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_p x_{ip})^2$$

$$\frac{\partial L}{\partial \beta_j} = -2 \sum_{i=1}^n (y_i - x_i^T \beta) x_{ij} = -2 \langle y - x\beta, x_j \rangle$$

$$\nabla L(\beta) = -2 (y - x\beta)^T x$$

where the inner product is

$$\langle x, y \rangle = \sum x_i y_i$$

Let's unpack that last step from the previous slide.

$$\begin{aligned}(Y - X\beta)^T X &= (Y - X\beta)^T \{x_1, x_2, \dots, x_p\} \\ &= [(Y - X\beta)^T x_1, (Y - X\beta)^T x_2, \dots, (Y - X\beta)^T x_p] \\ (Y - X\beta)^T x_j &= \sum_{i=1}^n (y_i - x_i^T \beta) x_{ij}\end{aligned}$$

Recall that two vectors are “orthogonal” (perpendicular) if their inner product is 0.

We call $y - X\beta$ the “residuals”.

$\nabla L = 0$ means the residuals are orthogonal to each column of X !!!!

Now we set the gradient equal 0 and solve.

$$\nabla L(\beta) = 0$$

$$X^T (y - X\beta) = 0$$

$$X^T y = X^T X \beta$$

$$\Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y$$

Note

Geometrically we talk about x and y being orthogonal:

$$\langle x, y \rangle = x'y = y'x = \sum x_i y_i = 0.$$

If we demean the variables so that we use $x_i - \bar{x}$ and $y_i - \bar{y}$, then we have

$$\sum (x_i - \bar{x})(y_i - \bar{y})$$

and the sample covariance is

$$\frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n - 1}$$

So, after you demean, saying they are orthogonal is the same as saying they are uncorrelated.

3. Regularized Linear Regression - L2, Ridge Regression

How can we make a linear regression model with a lot of x 's (and corresponding β 's) simple?

One way way would be to set some of the β_j to 0.

*Another way is to **shrink** the β s towards 0 !!!!!!!*

Ridge Regression:

Recall that least squares works by picking the coefficients to minimize

$$RSS = \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2.$$

Ridge regression works by minimizing:

$$\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p \beta_j^2.$$

For large λ you pay a price to make a coefficient large !!

Minimize:

$$\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p \beta_j^2.$$

λ will be our “walk the bias-variance trade-off” parameter.

small λ : can have big coefficient \Rightarrow *complex model*.

big λ : can't have many big coefficients \Rightarrow *simple model*.

So, for every λ , you will get a different optimizing β :

$$\lambda \Rightarrow \hat{\beta}_{\lambda}^R.$$

For example $\hat{\beta}_0^R$ is just the least squares estimator.

How do you choose λ ?

cross-validation, or another out-of-sample criterion!!.

Note:

We are minimizing

$$\text{fit: } \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2$$

+

$$\text{penalty: } \lambda \sum_{j=1}^p \beta_j^2.$$

Since the penalty treats all the β_j the same you have to be thinking about all the x 's the same. What are the units of β_j ?

Usually people *standardize* the x 's before they do this kind of shrinkage.

Hitters Data

Let's look at the "Hitters" example used in the Lab in the ISLR book.

Each observation corresponds to a baseball player.

Y: Salary:

1987 annual salary on opening day in thousands of dollars.

x1: AtBat:

Number of times at bat in 1986

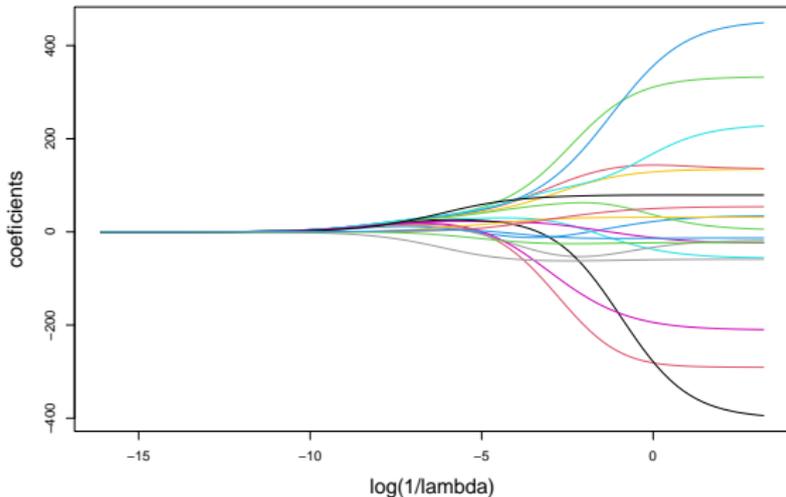
...

x19: NewLeague:

A factor with levels 'A' and 'N' indicating player's league at the beginning of 1987.

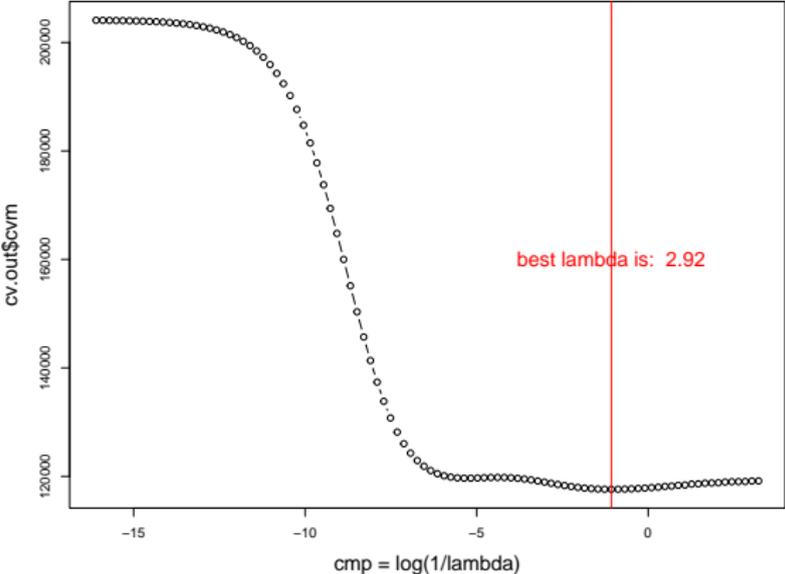
Let's try Ridge regression with the Hitters data.
I standardized all the x 's.

Here we plot $\log(1/\lambda)$ vs. $\hat{\beta}_\lambda^R$.

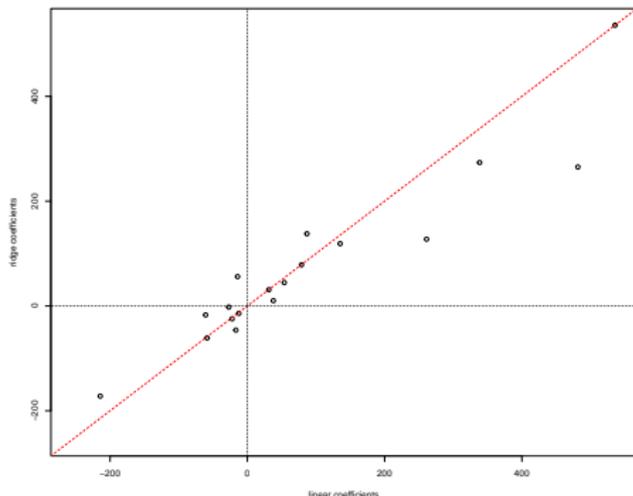


A *complex* model is one where the coefficients are allowed to be big.

Here is the cross-validation estimate of the out of sample loss.



Here we plot the coefficients from linear regression against those we get using ridge regression with the optimal λ .



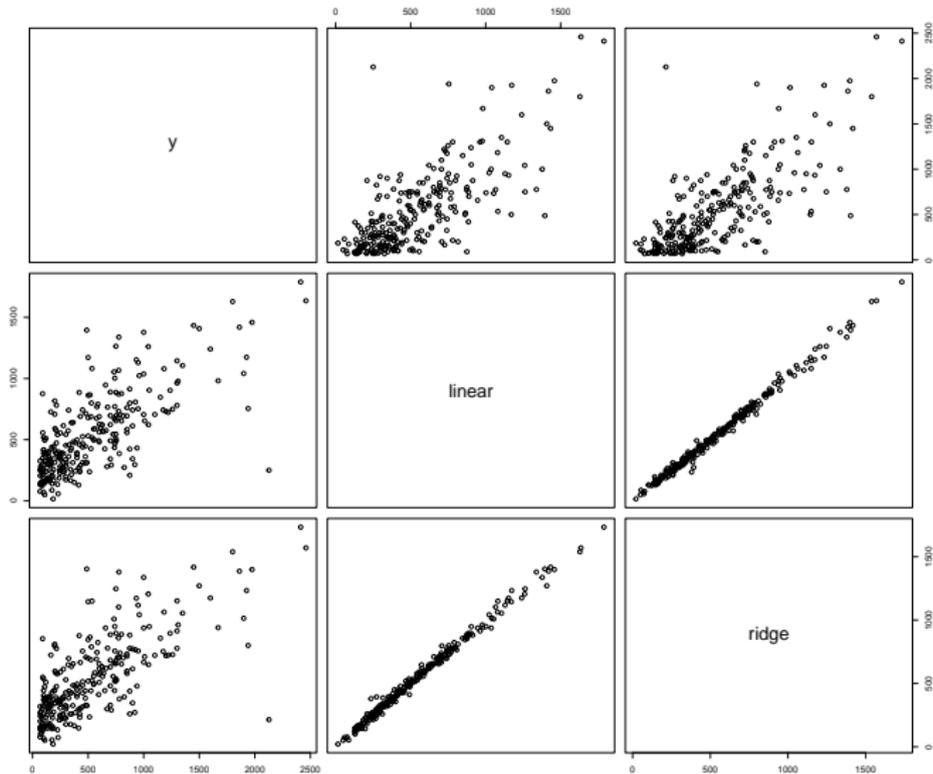
They are not too different in this case.

You can see some of the bigger coefficients are shrunk a bit.

A lot of the coefficients are close to 0, (we standardized the x's).

The x's with absolute values bigger than 100 are "AtBat" "Hits" "Walks" "CAtBat" "CHits" "CRuns" "CRBI" "CWalks"

Here we compare the in-sample fits from regression and ridge.



What is the ridge regression $\hat{\beta}_\lambda^R$?

Let's assume that we have subtracted the mean from y and each x .

We don't shrink the intercept so we can go ahead and just use \bar{y} to estimate it.

So, now our problem is just:
minimize:

$$\sum_{i=1}^n (y_i - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p \beta_j^2.$$

or

$$\sum_{i=1}^n (y_i - x_i' \beta)^2 + \lambda \sum \beta_i^2 = \|y - X\beta\|^2 + \lambda \|\beta\|_2^2$$

What is the ridge regression $\hat{\beta}_\lambda^R$?

$$\min_{\beta} \sum (y_i - x_i^T \beta)^2 + \lambda \sum \beta_i^2$$

$$\frac{\partial L}{\partial \beta_i} = -2 \sum (y_i - x_i^T \beta) [x_i] + 2\lambda \beta_i$$

$$\nabla L = 0 : \quad \lambda \beta = X^T (Y - X\beta)$$

$$\lambda \beta = X^T Y - X^T X \beta$$

$$(X^T X + \lambda I) \beta = X^T Y$$

$$\beta = (X^T X + \lambda I)^{-1} X^T Y.$$

What happens if the x 's are orthogonal (uncorrelated) so that $X'X =$ is diagonal?

If $X'X = \text{diag}(x_j'x_j)$
then

$$\hat{\beta}_j = \frac{\langle y, x_j \rangle}{\langle x_j, x_j \rangle + \lambda}$$

which is an extremely simple and intuitive version of *shrinkage*.

If $\lambda = 0$ we have the usual OLS solution, but as λ increases, our solution is pushed towards 0.

Regularization and Constrained Optimization

If we let $f(\beta) = \|y - X\beta\|^2$ and $p(\beta) = \|\beta\|^2$ then we are minimizing

$$f(\beta) + \lambda p(\beta)$$

More generally if f is our “fit” and p is our “penalty” we have a very general approach to walking the bias-variance trade-off as we vary λ .

As long as p does not like big β , then large λ will give us “simple” models.

This approach is often called *regularization*.

It is also useful to view the problem as a constrained fit.

Minimizing the unconstrained

$$f(\beta) + \lambda p(\beta)$$

is related to solving the *constrained optimization*

$$\min f(\beta) \text{ subject to } p(\beta) \leq k$$

min fit + penalty, *or* min fit subject to penalty not big.

Clearly if β^* minimizes $f(\beta) + \lambda p(\beta)$ then it must also solve

$$\min f(\beta) \text{ subject to } p(\beta) = p(\beta^*)$$

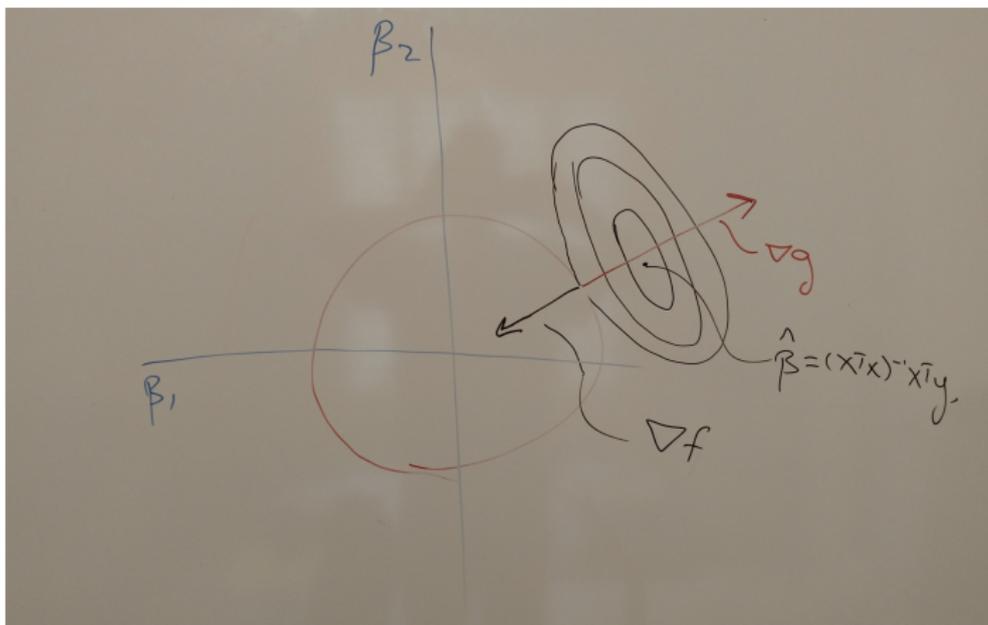
For our Ridge regression problem we have $f(\beta) = \|y - X\beta\|^2$ and $g(\beta) = \|\beta\|^2 - k$ where k is a positive constant.

In this case the contours $f(\beta) = c$ are ellipses and the contours $g(\beta) = c$ are circles.

We have a very nice picture which makes the lagrangian FOC intuitive.

Here is the key picture for the case where the constraint is binding.

Remember, ∇f is the direction in which f goes up the fastest!!
 ∇f points perpendicularly to the contour of f .



It is intuitive that $\nabla f + \alpha \nabla g = 0$ with $\alpha > 0$.

To solve our Ridge regularization as a constrained problem we have:

$$-\nabla f' = 2X'(Y - X\beta).$$

$$\nabla g' = 2\beta.$$

$$2\alpha\beta = 2X'(Y - X\beta).$$

$$\beta_{\alpha}^* = (X'X + \alpha I)^{-1}X'Y.$$

We would then solve (the easy problem) of finding the α such that $\|\beta_{\alpha}^*\|^2 = k$.

Regularization in Machine Learning

We can think of ridge regression as a basic example of a general setup we see in Machine Learning.

We have a “model” that defines our *action* given the information in the observed vector of features x .

Our action is characterized by a vector of parameters θ .

Let's denote our action by $f(x, \theta)$.

Given our action and the then observed y we incur a *loss* $L(y, f(x, \theta))$.

In our linear regression example, θ is β and

$$f(x, \beta) = x' \beta, \quad L(y, f(x, \beta)) = (y - x' \beta)^2.$$

We have “complexity measure” $C(\theta)$.

In ridge regression, $C(\beta) = \|\beta\|_2^2$.

Given training observation (x_i, y_i) and parameter θ we “learn” θ by

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^n L(y_i, f(x_i, \theta)) + \lambda C(\theta)$$

The parameter λ walks you from simple to complex models.

4. Shrinkage: The Lasso

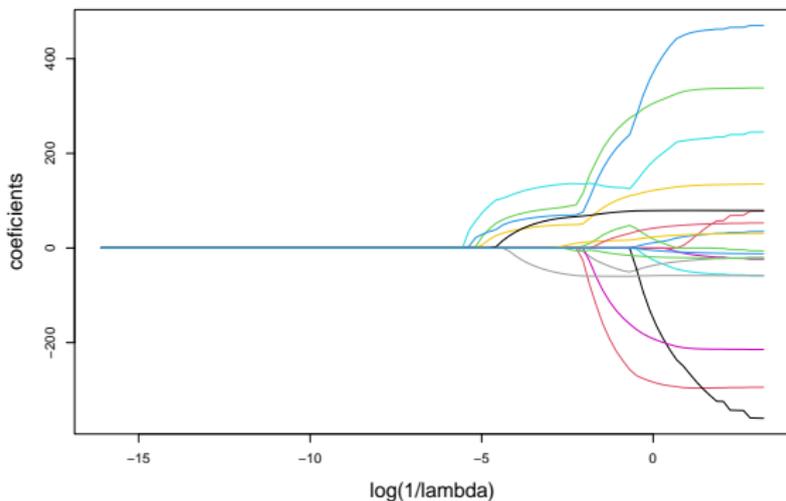
The Lasso (least absolute shrinkage and selection operator) changes the form of the penalty.

Now, we minimize:

$$\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p |\beta_j|.$$

This may not seem like a big deal, but it turns out the solution to this problem can set a β_j exactly to 0, so that you get variable selection.

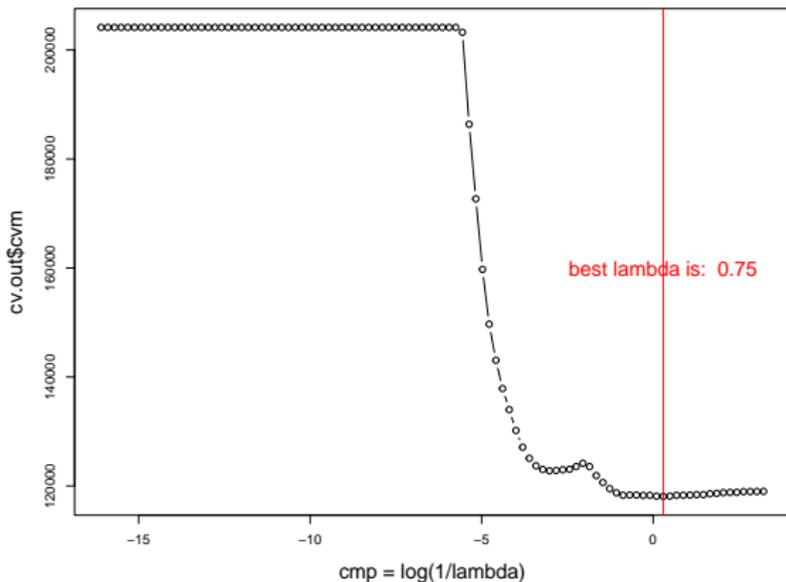
Here are the lasso solutions as λ varies.



The point is, that for big λ , coefficients get set to 0.

In the Lasso, there is shrinkage as well as selection and the shrinkage takes on a different form than in L2 regularization.

As usual, we can choose λ using cross-validation.



For big enough λ , all the coefficients are set to 0.

Here are the $\hat{\beta}_\lambda^L$ for the best CV lambda.

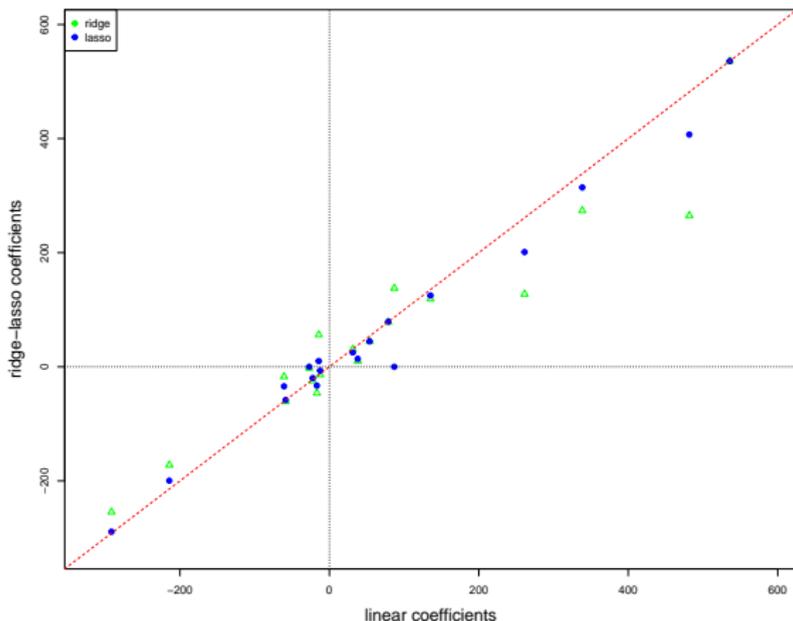
| | | | | | |
|-------------|-------------|-------------|------------|------------|------------|
| (Intercept) | AtBat | Hits | HmRun | Runs | RBI |
| 535.925882 | -289.109678 | 314.374161 | 13.916703 | -34.239618 | 0.000000 |
| Walks | Years | CAtBat | CHits | CHmRun | CRuns |
| 124.971379 | -33.041086 | -191.538897 | 0.000000 | 10.072782 | 407.092162 |
| CRBI | CWalks | LeagueN | DivisionW | PutOuts | Assists |
| 201.114077 | -199.619610 | 25.169164 | -58.149563 | 79.100366 | 44.502170 |
| Errors | NewLeagueN | | | | |
| -19.688952 | -6.704629 | | | | |

We see that a couple are 0.

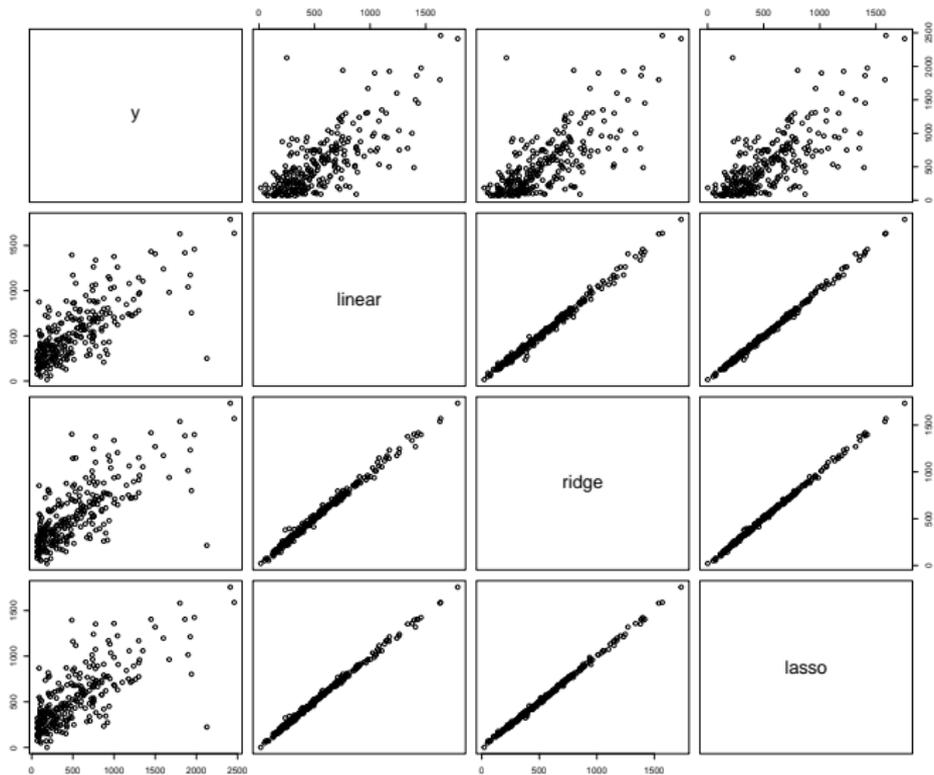
Plot Ridge and Lasso coefficients vs the least squares coefficients.

A couple of the Lasso coefficients are 0.

A big coefficient is shrunk less under Lasso than Ridge.



Fits are not actually too different.



Why do people like the Lasso?

- ▶ Simple way to walk the bias variance trade-off.
- ▶ Zero coefficients give variable selection, can get more interpretable models.
- ▶ Computationally fast.

5. Understanding the Lasso Solution

Why does the Lasso give solutions with coefficients at 0?

How is Ridge different from Lasso?

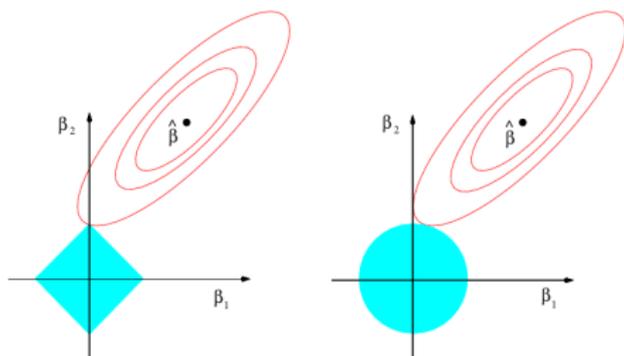
To get a good simple intuition, it is helpful to consider the constrained optimization view of Lasso and Ridge.

Ridge:

$$\begin{aligned} & \underset{\beta_0, \beta}{\text{minimize}} && \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 \\ & \text{subject to} && \sum_{j=1}^p \beta_j^2 \leq t^2 \end{aligned}$$

Lasso:

$$\begin{aligned} & \underset{\beta_0, \beta}{\text{minimize}} && \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 \\ & \text{subject to} && \sum_{j=1}^p |\beta_j| \leq t \end{aligned}$$



At left we have the Lasso problem, where the constraint set looks like a diamond.

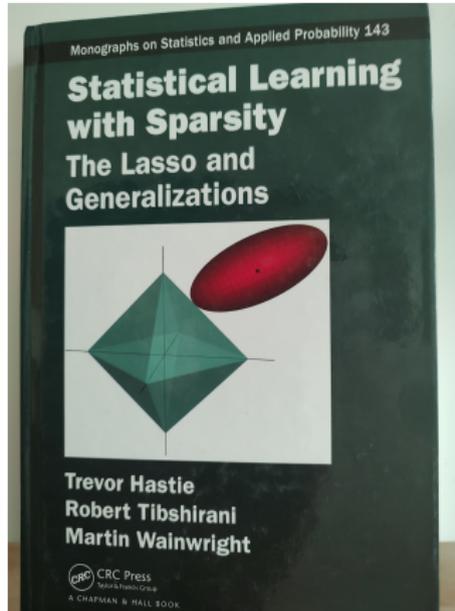
At right we have the Ridge problem, where the constraint set looks like a circle.

The diamond constraint can give solutions at an axis.

This is a very famous picture !!!!!

“Sparsity” is another term you often hear.

The Lasso give sparsity because some the coefficients are set to 0.



sparsity, regularization, shrinkage

With L2 and L1 you have a convex optimization problem.

You are minimizing a convex function on a convex constraint set.

If you go L_p , $p < 1$, you get even more variable selection, but you lose the convexity.



$$\|x\|_p = \left(\sum |x_i|^p \right)^{1/p}$$

The Simplest Version

Let's consider the simplest possible version of our problems back in the "Lagrangian" formulation:

Ridge:

$$\underset{\beta}{\text{minimize}} \quad (y - \beta)^2 + \lambda \beta^2$$

Lasso:

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2} (y - \beta)^2 + \lambda |\beta|$$

Adding the 1/2 for the Lasso changes nothing and makes the expressions look nicer.

$$\underset{\beta}{\text{minimize}} (y - \beta)^2 + \lambda \beta^2$$

For the Ridge version we are minimizing a quadratic so we can easily find the global minimum by setting the derivative equal to 0:

$$2(y - \beta)(-1) + 2\lambda\beta = 0 \Rightarrow \hat{\beta}^R = \frac{y}{1 + \lambda}.$$

Of course the unconstrained solution is $\hat{\beta} = y$ so we can very nicely see how a choice of λ shrinks the estimate towards 0.

For the Lasso problem, we suddenly have a basic technical problem.

The function

$$g(\beta) = |\beta|$$

is not differentiable at 0!!

Our function is convex, so there is a global minimum, but can we find it in a simple way?

We can, and the solution will shed light on the Lasso and on how to solve the general regression problem.

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2} (y - \beta)^2 + \lambda |\beta|$$

To derive the Lasso solution, suppose the optimal β is greater than 0.

Then, locally, our differential first order conditions apply *and* our criterion is differentiable since we know $|\beta| = \beta$.

$$(y - \beta)(-1) + \lambda = 0 \Rightarrow \hat{\beta}^L = y - \lambda.$$

Similarly, if the optimal is less than 0, then $|\beta| = -\beta$ so,

$$(y - \beta)(-1) - \lambda = 0 \Rightarrow \hat{\beta}^L = y + \lambda.$$

Shrink towards 0 by an amount λ !!

Now we only have three possibilities for the optimal β and you can just check that the minimum is obtained with

$$\hat{\beta}^L = \begin{cases} y - \lambda & y > \lambda \\ 0 & |y| \leq \lambda \\ y + \lambda & y < -\lambda \end{cases}$$

For example, suppose $0 < y < \lambda$.

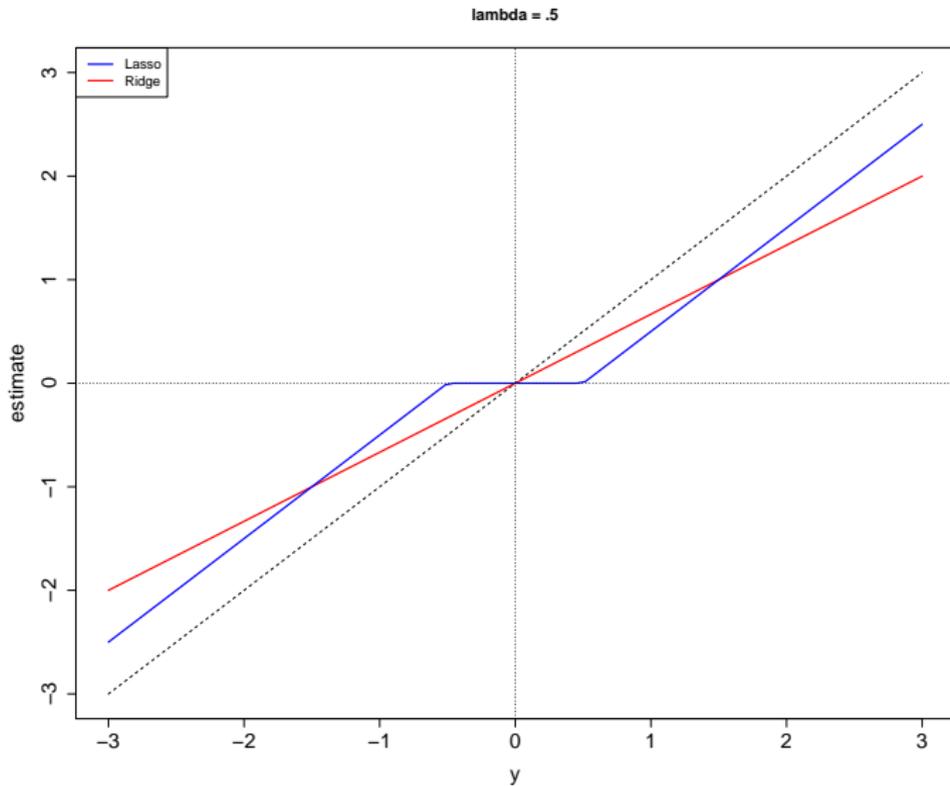
Which is better, $\beta = 0$ or $\beta = y - \lambda$?

At $y - \lambda$ we have

$$\begin{aligned} (y - (y - \lambda))^2 + \lambda|y - \lambda| &= \lambda^2 + \lambda|y - \lambda| \\ &\geq (y - 0)^2 + \lambda|0|. \end{aligned}$$

Intuitively, if $0 < y < \lambda$, there is no way I want negative estimate $y - \lambda$.

Here is a plot of the Lasso and Ridge shrinkage.



We can express these solutions succinctly using the *soft thresholding function* S_λ .

$$\hat{\beta}^R = \frac{y}{1 + \lambda}.$$

$$\hat{\beta}^L = S_\lambda(y)$$

where

$$S_\lambda(y) = \text{sign}(y)(|y| - \lambda)_+$$

with $x_+ = x$ if x is positive and 0 otherwise.

Let's see how it works out in practice.

Let's say $y = 1$.

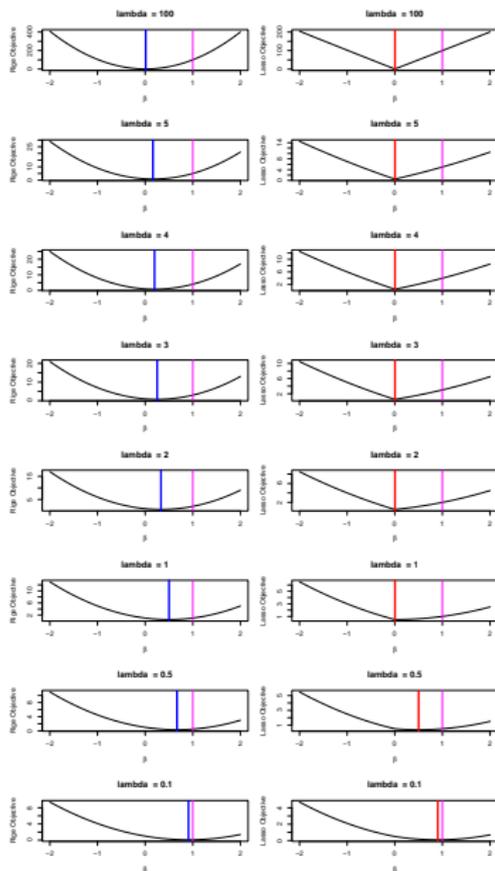
We plot y with the solid magenta line.

λ decreases as we go down the plots.

At left we have the Ridge criterion plotted with the minimizing β indicated by the solid blue line.

At right we have the Lasso criterion plotted with the minimizing β indicated by the solid red line.

Each estimate moves from 0 to 1, but the Lasso estimate sticks at 0 for a while and then moves faster to 1.



Standardization:

Ok, now let's try Lasso with some x 's !!

But first, we emphasize again that for this to make sense you have to put the x 's on the same scale by standardizing them.

The Lasso literature strongly favors standardization using the sample mean and variance.

Since we are not trying to regularize (shrink) the intercept, it is usual to start by demeaning y and x :

$$y_i \rightarrow y_i - \bar{y}; \quad x_{ij} \rightarrow x_{ij} - \bar{x}_j.$$

Recall that if you run a regression using the demeaned variables, you get the same slope estimates.

We then scale the x 's:

$$x_{ij} \rightarrow \frac{x_{ij}}{s_j}$$

where

$$s_j^2 = \frac{\sum x_{ij}^2}{n}$$

Note that after you do this standardization $\sum_i x_{ij}^2 = n$ for each $j = 1, 2, \dots, p$.

Lasso with one x

Let's now see what happens when we just have one x variable.

After standardizing we minimize:

$$\frac{1}{2n} \sum_{i=1}^n (y_i - \beta x_i)^2 + \lambda |\beta|.$$

Dividing by $2n$ does not change the problem, but makes the formulas turn out nicer.

Again if the solution were positive, we must have

$$\frac{1}{n} \sum (y_i - \beta x_i)(-x_i) + \lambda = 0 \rightarrow \hat{\beta}^L = \frac{1}{n} \langle x, y \rangle - \lambda.$$

And if negative,

$$\frac{1}{n} \sum (y_i - \beta x_i)(-x_i) - \lambda = 0 \rightarrow \hat{\beta}^L = \frac{1}{n} \langle x, y \rangle + \lambda.$$

So that,

$$\hat{\beta}^L = S_\lambda\left(\frac{1}{n} \langle x, y \rangle\right).$$

Note that with our standardization, we are basically soft-thresholding the least-squares $\hat{\beta}$.

$$\hat{\beta} = \frac{\langle x, y \rangle}{\langle x, x \rangle} = \frac{\langle x, y \rangle}{n}$$

so

$$\hat{\beta}^L = S_\lambda(\hat{\beta}).$$

How about this way??

$$\begin{aligned} & \|y - x\beta\|^2 \\ &= \|y - x\hat{\beta}\|^2 + \|x\beta - x\hat{\beta}\|^2 \\ &= \|y - x\hat{\beta}\|^2 + (\beta - \hat{\beta})^2 x^T x \\ &= S^2 + n(\beta - \hat{\beta})^2 \\ &\frac{1}{2n} \|y - x\beta\|^2 = \frac{1}{2n} S^2 + \frac{1}{2} (\beta - \hat{\beta})^2 \end{aligned}$$

Then by the result we got in our simplest problem
 $\hat{\beta}^L = S_\lambda(\hat{\beta})$.

The General Problem, p Variables:

$$\underset{\beta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_j \beta_j x_{ij})^2 + \lambda \sum_j |\beta_j|.$$

or,

$$\underset{\beta}{\text{minimize}} \frac{1}{2n} \|y - X\beta\|^2 + \lambda \|\beta\|_1$$

Cyclic Coordinate Descent:

Given a choice of λ , suppose we knew all of the coefficients except β_j .

We can write our objective as:

$$\text{minimize}_{\beta_j} \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_{k \neq j} \beta_k x_{ik} - \beta_j x_{ij})^2 + \lambda |\beta_j| + \lambda \sum_{k \neq j} |\beta_k|.$$

Which is the same problem as

$$\text{minimize}_{\beta_j} \frac{1}{2n} \sum_{i=1}^n (r_i^{(j)} - \beta_j x_{ij})^2 + \lambda |\beta_j|$$

with

$$r_i^{(j)} = y_i - \sum_{k \neq j} \beta_k x_{ik}$$

The $r_i^{(j)}$ are the *partial residuals*.

$$\underset{\beta_j}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^n (r_i^{(j)} - \beta_j x_{ij})^2 + \lambda |\beta_j|$$

But we know how to solve this problem:

$$\hat{\beta}_j = S_\lambda\left(\frac{1}{n} \langle x_j, r^{(j)} \rangle\right).$$

This gives us a very simple *cyclic coordinate descent algorithm*

- ▶ Pick a fixed order for the coefficients (variables), e.g $1, 2, \dots, p$.
- ▶ Cycle through the coefficient updating each with the soft thresholding formula: $\hat{\beta}_j = S_\lambda(\frac{1}{n} \sum x_j, r^{(j)})$.
- ▶ Repeat until convergence.

Simple !!!

Note:

We often want to do this for a set of λ values.

If we start with all the β_j at 0, then our initial $r^{(j)} = y$.

Thus we know that if we set

$$\lambda_{max} = \max_j \left| \frac{1}{n} \langle x_j, y \rangle \right|$$

then for that λ , and all larger, no matter what coefficient we attempted to update, we would get 0. So, there is no need to consider $\lambda > \lambda_{max}$.

So, we can,

- ▶ Start at $\lambda = \lambda_{max}$.
- ▶ Slowly decrease, λ .
- ▶ At each λ , find a solution using cyclic coordinate descent.
- ▶ *warm start*, each cyclic descent by starting at the solution from the previous λ .

Note:

Suppose our x 's are orthogonal:

$$\langle x_i, x_j \rangle = x_j' x_i = 0, \quad i \neq j.$$

Since we have demeaned, this is equivalent to the x 's being uncorrelated.

Then,

$$\langle x_j, r^{(j)} \rangle = \langle x_j, y \rangle$$

So our cyclic algorithm converges immediately to

$$\hat{\beta}_j = S_\lambda\left(\frac{1}{n} \langle x_j, y \rangle\right).$$

Just as in least squares regression, we can fit the model one x at a time if the x 's are uncorrelated.

6. The Elastic Net

Once you have the idea of penalized regression, you can imagine cooking up lots of different penalty specifications.

There are lots of variations in the literature, let's look at the *Elastic Net* which simply combines L1 and L2 penalties.

The motivation for the Elastic Net comes from the observation that if x 's are highly correlated then the Lasso may behave erratically.

To see this, consider a regression where

$$y = \beta_1 x_1 + \epsilon$$

Suppose $\hat{\beta}_1 \approx 1$ works pretty well.

Suppose $x_2 \approx x_1$ and we run the regression

$$y = \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

what will happen?

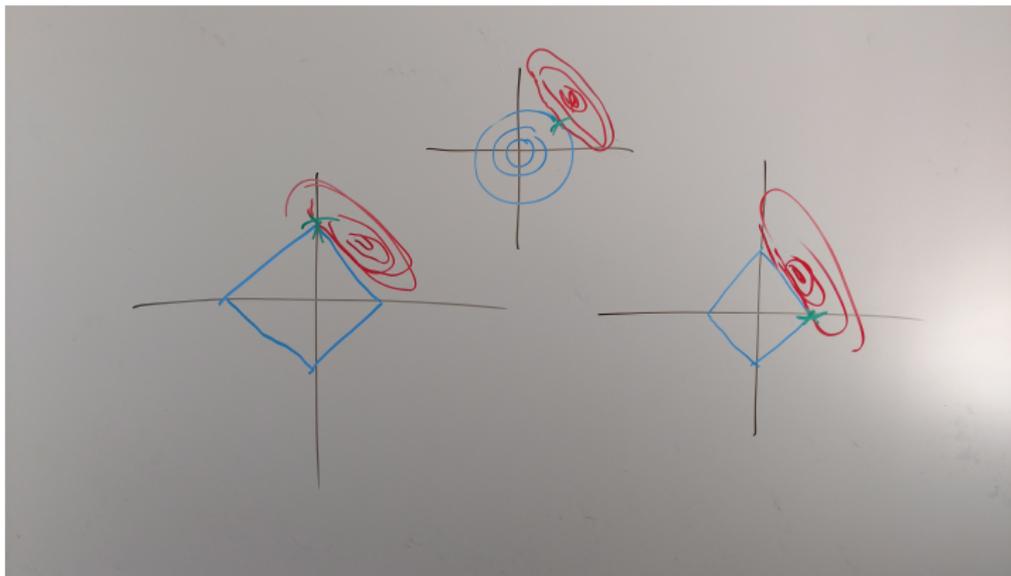
In this case any regression where $\beta_1 + \beta_2 = 1$ will fit pretty well.

Thus, we *almost* have a lack of identification.

The contours of the of the likelihood will be ellipses along the line $\beta_2 = 1 - \beta_1$ which will align with the Lasso constraint.

Thus, small changes in the data, or introduction of other x variables could swing the solution from $\beta = (1, 0)$ to $\beta = (0, 1)$ erratically.

What will the L2 penalty do?



The L2 penalty would divy up the fit 50/50 preferring a solution with $\beta = (.5, .5)$.

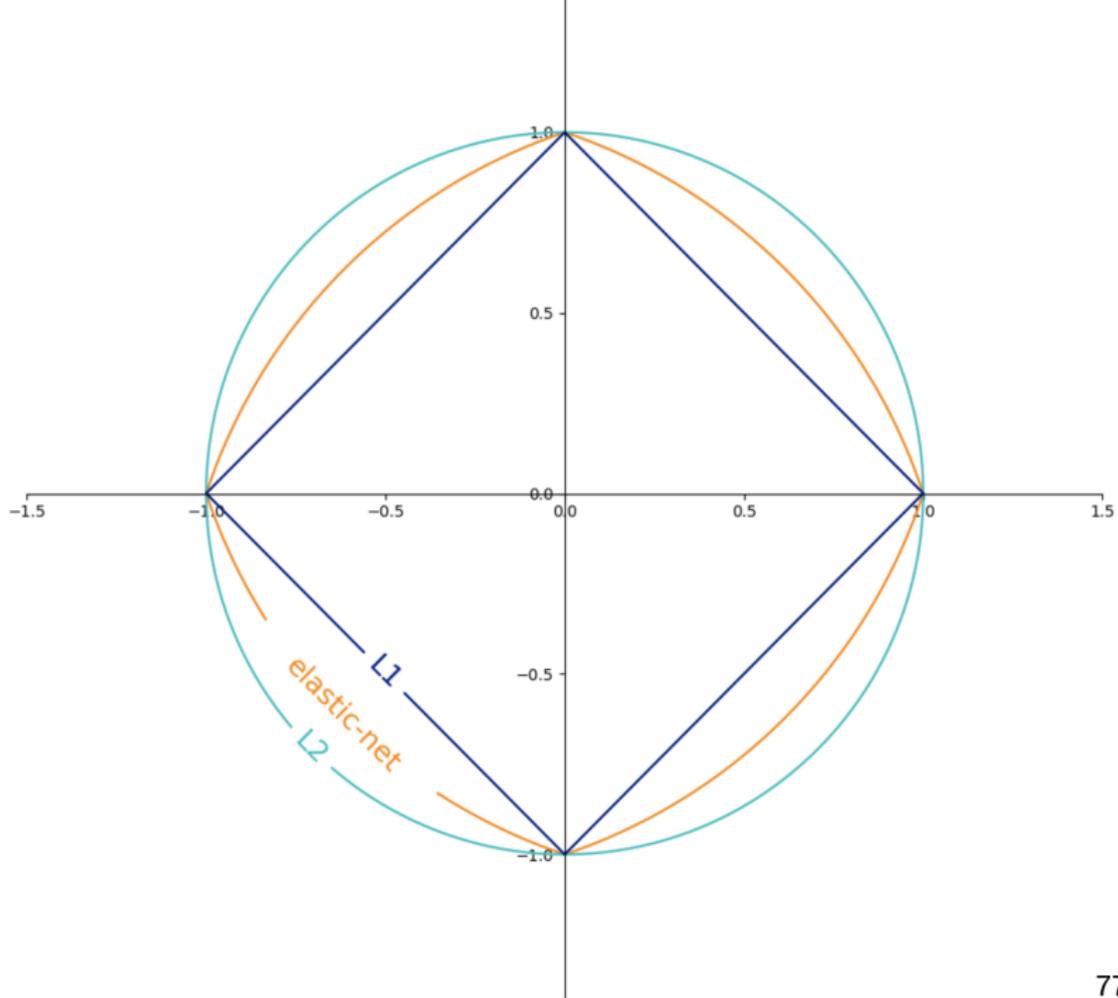
This motivates the Elastic net which just mixes in the L1 with the L2:

$$\text{minimize}_{\beta_0, \beta} \left\{ \frac{1}{2} \sum_{i=1}^n (y_i - \beta_0 - \sum_j \beta_j x_{ij})^2 + \lambda \left[\frac{1}{2} (1 - \alpha) \|\beta\|_2^2 + \alpha \|\beta\|_1 \right] \right\}$$

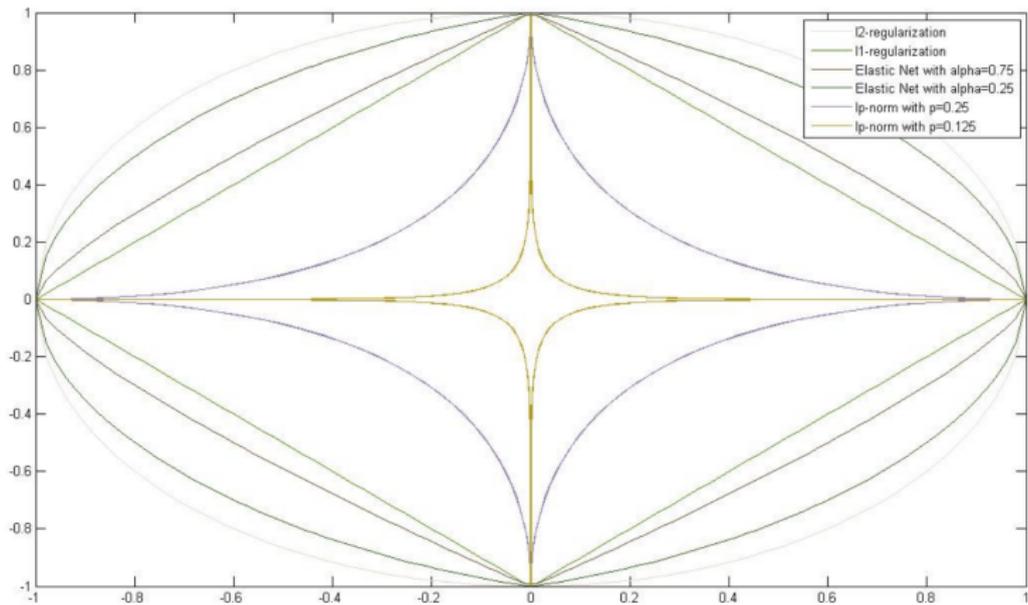
For an individual coefficient, the penalty is then

$$\lambda \left[\frac{1}{2} (1 - \alpha) \beta_j^2 + \alpha |\beta_j| \right].$$

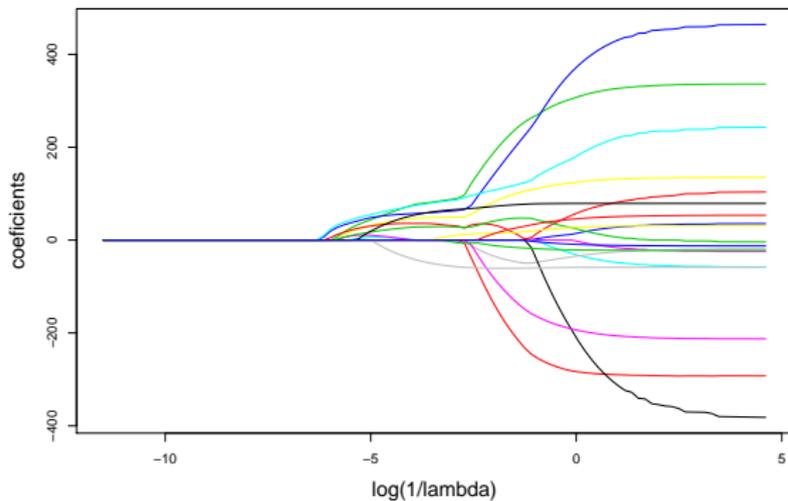
With the elastic net, we can still get solutions with zero coefficients, but the solution for highly correlated x variables is stabilized.



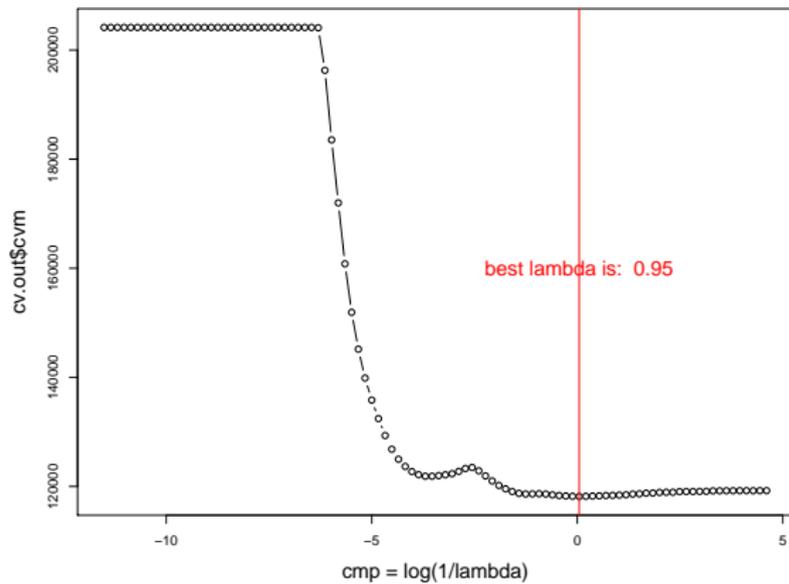
Cornell computer science:



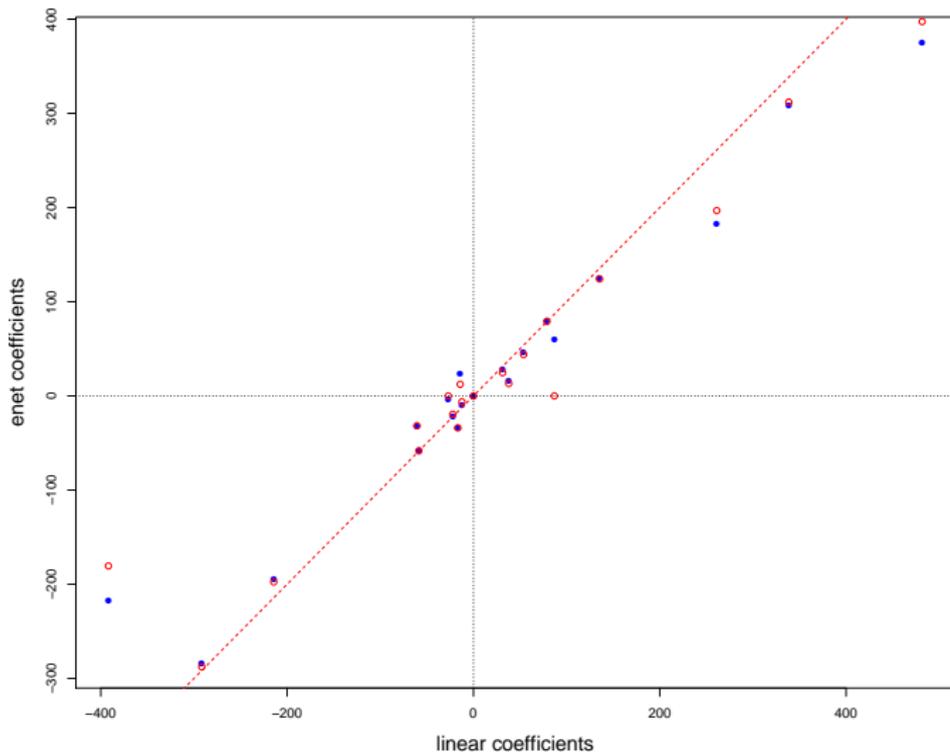
Hitters Data, Elastic net ($\alpha = .5$) solution path.



CV for elastic net.



Elastic net coefficients vs linear (blue).
Lasso net coefficients vs linear (red).



7. Subset Selection

If we just “throw in a ton of x 's” our model may be too complex, we may overfit.

Often, we try to start with a “ton of x 's” and then see how many we can throw out and still have good fit.

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \epsilon$$

Throwing out an x is equivalent to setting its coefficient to 0.

Which coefficients do we set to 0?

The key idea is *the bias variance trade-off !!!*

If we set too many coefficients to 0, we may be throwing out some variables that do important work in explaining Y , \Rightarrow *bias*.

If we keep too many variables, it may be difficult to get good estimates of all the corresponding coefficients \Rightarrow *variability*.

Our basic problem is that there are a lot of possible ways to pick a subset of variables to keep!!

Let k denote the number of variables kept.

How many ways can you choose k from p : $\binom{p}{k} = \frac{p!}{k!(p-k)!}$.

And, summing over possible $k = 0, 1, 2, \dots, p$, there are 2^p possible regression models.

Example, $p=20, k=10$:

$$2^{20} = 1,048,576$$

$$\text{choose}(20,10) = 184,756$$

What we need is a simple way to move from simpler models to more complex models (recall k in kNN).

In subset selection, we will **let k denote the number of variables used**, so that k goes from 0 to p .

Big k : complex model, Small k : simple model !!

For each k we will choose a single regression model from the $\binom{p}{k}$ possible models.

Two possible ways of choosing a subset (a model) given k are:

small p :

All subsets:

For p less than about 40, it is possible to run all the possible regressions.

Given the number of variables k , we will pick the subset of variables of size k with the highest R^2 .

big p :

Forward Stepwise Selection:

- ▶ Start with $k=0$, no variables selected.
- ▶ Given a current k and corresponding subset, add in the new variable which gives you the biggest increase in R^2 .
- ▶ Stop at $k = p$.

This is a greedy forward search!!

We can now choose k , the number of variables, the same way we chose k in kNN.

A simple validation set approach simply splits the data into train and validate, and sees which value of k gives the best prediction.

Or, we could use cross validation.

Hitters Example

Let's look at the "Hitters" example used in the Lab in the ISLR book.

Each observation corresponds to a baseball player.

Y: Salary:

1987 annual salary on opening day in thousands of dollars.

x1: AtBat:

Number of times at bat in 1986

...

x19: NewLeague:

A factor with levels 'A' and 'N' indicating player's league at the beginning of 1987.

For $k = 1, 2, 3, 4, 5$,
 here are the variables that give you the highest R^2 :

| | (Intercept) | AtBat | Hits | HmRun | Runs | RBI | Walks | Years | CAtBat | CHits | CHmRun |
|---|-------------|-------|--------|---------|-----------|---------|---------|--------|------------|-------|--------|
| 1 | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 2 | TRUE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 3 | TRUE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 4 | TRUE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 5 | TRUE | TRUE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| | CRuns | CRBI | CWalks | LeagueN | DivisionW | PutOuts | Assists | Errors | NewLeagueN | | |
| 1 | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 2 | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 3 | FALSE | TRUE | FALSE | FALSE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 4 | FALSE | TRUE | FALSE | FALSE | TRUE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE |
| 5 | FALSE | TRUE | FALSE | FALSE | TRUE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE |

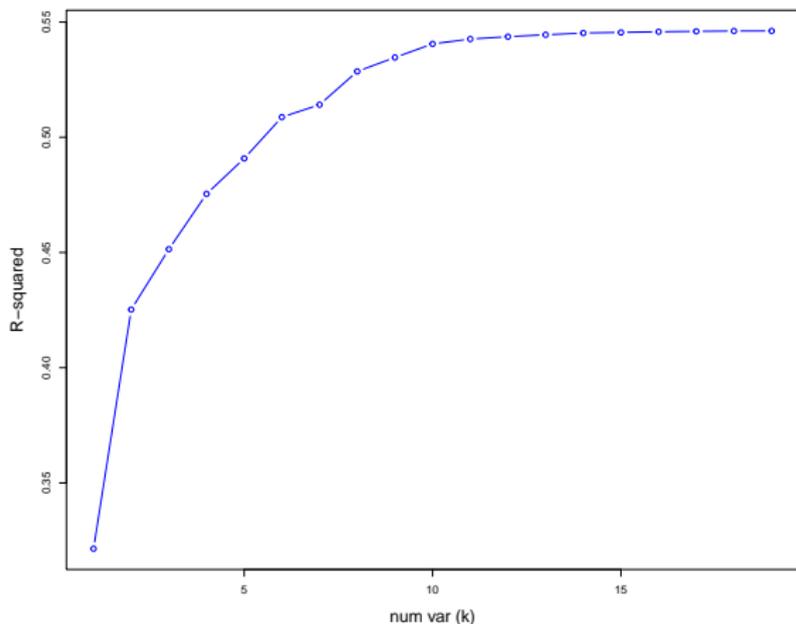
So, if $k = 3$, you use Hits, CRBI, and PutOuts.

'Hits': Number of hits in 1986

'CRBI': Number of runs batted in during his career

'PutOuts': Number of put outs in 1986

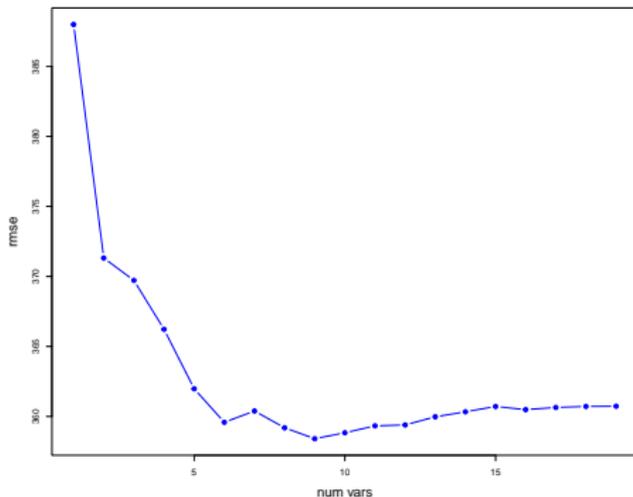
Here is a plot of k vs. the R^2 for the model having the highest R^2 out of all models of size k .



Of course, we may not want the model with the highest in-sample R^2 !!!

Split the data 50/50 into train/validate. Get the best subset for each k using the train, and then predict on the validate.

I repeated the train/validate split 100 times and then averaged the results. Maybe better to do 10-fold cross validation.



I'll choose $k = 6$.

Given the choice $k = 6$, we then get the best subset of size 6, using all the data. Here is the regression.

Call:

```
lm(formula = Salary ~ ., data = ddfsub)
```

Residuals:

| | Min | 1Q | Median | 3Q | Max |
|--|---------|---------|--------|--------|---------|
| | -873.11 | -181.72 | -25.91 | 141.77 | 2040.47 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) | |
|-------------|------------|------------|---------|----------|-----|
| (Intercept) | 91.51180 | 65.00006 | 1.408 | 0.160382 | |
| AtBat | -1.86859 | 0.52742 | -3.543 | 0.000470 | *** |
| Hits | 7.60440 | 1.66254 | 4.574 | 7.46e-06 | *** |
| Walks | 3.69765 | 1.21036 | 3.055 | 0.002488 | ** |
| CRBI | 0.64302 | 0.06443 | 9.979 | < 2e-16 | *** |
| DivisionW | -122.95153 | 39.82029 | -3.088 | 0.002239 | ** |
| PutOuts | 0.26431 | 0.07477 | 3.535 | 0.000484 | *** |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 319.9 on 256 degrees of freedom

Multiple R-squared: 0.5087, Adjusted R-squared: 0.4972

F-statistic: 44.18 on 6 and 256 DF, p-value: < 2.2e-16

8. Maximum Likelihood Estimation

For some things that we do it will be helpful to view the least squares estimator as a maximum likelihood estimate.

Recall that if we have parametric model for observable y with parameter θ

$$p(y | \theta)$$

then, we obtain the maximum likelihood estimate (MLE) of θ by solving:

$$\max_{\theta} p(y | \theta).$$

This is very intuitive, find the parameter value which makes what you have seen (the y) most likely.

To compute an MLE for regression we again need an assumption about the errors.

Again, let's use normal errors:

$$Y_i = x_i' \beta + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2), \quad \epsilon_i \text{ iid.}$$

or,

$$Y_i \sim N(x_i' \beta, \sigma^2), \quad \textit{independent}$$

Our parameter is (β, σ) and the likelihood has the form

$$p(y | \beta, \sigma^2) = \prod_{i=1}^n p(y_i | \beta, \sigma^2)$$

We will maximize this over (β, σ^2) .

$$\begin{aligned} p(y_1, y_2, \dots, y_n | \beta, \sigma) &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}} \frac{1}{\sigma} \exp\left(-\frac{1}{2\sigma^2} (y_i - x_i' \beta)^2\right) \\ &= (2\pi)^{-n/2} \sigma^{-n} \exp\left(-\frac{1}{2\sigma^2} \|Y - X\beta\|^2\right). \end{aligned}$$

So for any σ , the likelihood is minimized at the least squares $\hat{\beta} = (X'X)^{-1}X'y$.

Let $v = \sigma^2$. Let $S = \|y - X\hat{\beta}\|^2$.

$$\log(L(v, \hat{\beta})) = -\frac{n}{2} \log(v) - \frac{1}{2v} S + C.$$

$$-2\log(L(v, \hat{\beta})) = n \log(v) + \frac{1}{v} S + C.$$

Taking the derivative wrt v and setting it equal to 0, we have:

$$\frac{n}{v} - \frac{S}{v^2} = 0, \Rightarrow \hat{v} = \frac{S}{n}.$$

$$\text{And, } -2\log(L(\hat{v}, \hat{\beta})) = n \log(\hat{v}) + n + C.$$

9. AIC and BIC in Linear Regression

Suppose we have a parametric model $f(y | \theta)$.

For example, in our regression model (suppressing x) we have $\theta = (\beta_0, \beta_1, \dots, \beta_p, \sigma)$.

Now suppose we have two parametric models for the same y with two corresponding parameters:

$$p(y | \theta_1) \text{ and } p(y | \theta_2).$$

For example, in our regression model we could have $\theta_1 = (\beta_0, \beta_1, \dots, \beta_p, \sigma)$ and $\theta_2 = (\beta_0, \beta_1, \sigma)$

Given observed data y how can we decide which model is better?

Well, we could compare the maximized likelihoods:

$$L_1 = L(\hat{\theta}_1) \text{ and } L_2 = L(\hat{\theta}_2)$$

where the parameter estimates are the MLEs.

But, if we just choose the model with the largest maximized likelihood, we could overfit.

Rather than use a train/test strategy to choose the models, AIC and BIC compare the maximized log Likelihoods but subtract off a “penalty term” which depends on the number of parameters in the model.

Let m be the number of parameters in the model.

$$\text{AIC: } -2 \log(\hat{L}) + 2m$$

$$\text{BIC: } -2 \log(\hat{L}) + \log(n)m.$$

The idea is that you choose the model which has the *smallest* AIC or BIC.

So, AIC charges you 2 per parameter and BIC charges you $\log(n)$.

BIC selects a simpler model since it charges more per parameter.

AIC: $-2 \log(\hat{L}) + 2m$

BIC: $-2 \log(\hat{L}) + \log(n)m$.

For example the model $\theta_1 = (\beta_0, \beta_1, \dots, \beta_p, \sigma)$ has $p + 2$ parameters and the model $\theta_2 = (\beta_0, \beta_1, \sigma)$ has 3 parameters.

This is not guaranteed to work as a lot of approximations and assumptions go into their derivations.

What does “work” mean?

Choose a model that gives good out-of-sample predictions!!!

AIC and BIC for Regression:

Let \hat{L} denote the maximized likelihood
(the likelihood evaluated at the MLE's).

Suppose we include the intercept and use k x variables:

AIC:

$$n \log(\hat{\sigma}_{MLE,k}^2) + 2k + C(n)$$

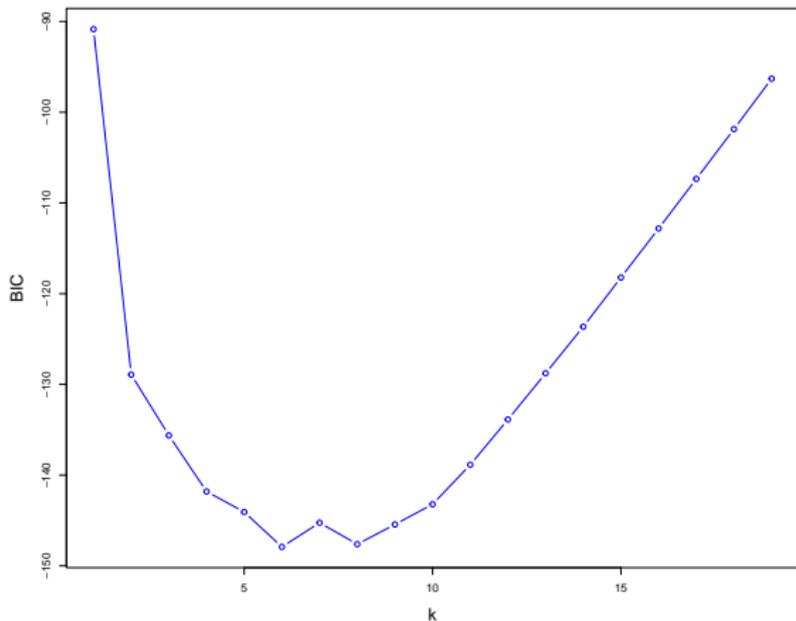
BIC:

$$n \log(\hat{\sigma}_{MLE,k}^2) + \log(n) k + C(n)$$

You are supposed to prefer the model which has the smallest AIC or BIC.

$C(n)$ is a constant that only depends on n , since this is fixed, we can ignore it.

Here is a plot of k vs. BIC .



This suggests k of about 6.

The Deviance

The basic idea behind AIC and BIC is the $L(\hat{\theta})$ can be used as measure of fit on the training data.

The is often expressed in terms of the deviance:

$$D = -2 \log(L(\hat{\theta}))$$

So, a *small deviance means a good fit*.

$$\text{AIC: } D + 2k, \quad \text{BIC: } D + \log(n) k$$

We choose the model with the smallest AIC or BIC.

Warning: D or $D + c$ where c is an constant that does not depend on the model, so when you see a deviance computed it may be $\pm c$.

10. The Diabetes Data

Let's look at all this stuff with the Diabetes data.

<http://web.stanford.edu/~hastie/StatLearnSparsity/data.html>

Diabetes data

These data consist of observations on 442 patients, with the response of interest being a quantitative measure of disease progression one year after baseline.

There are ten baseline variables---

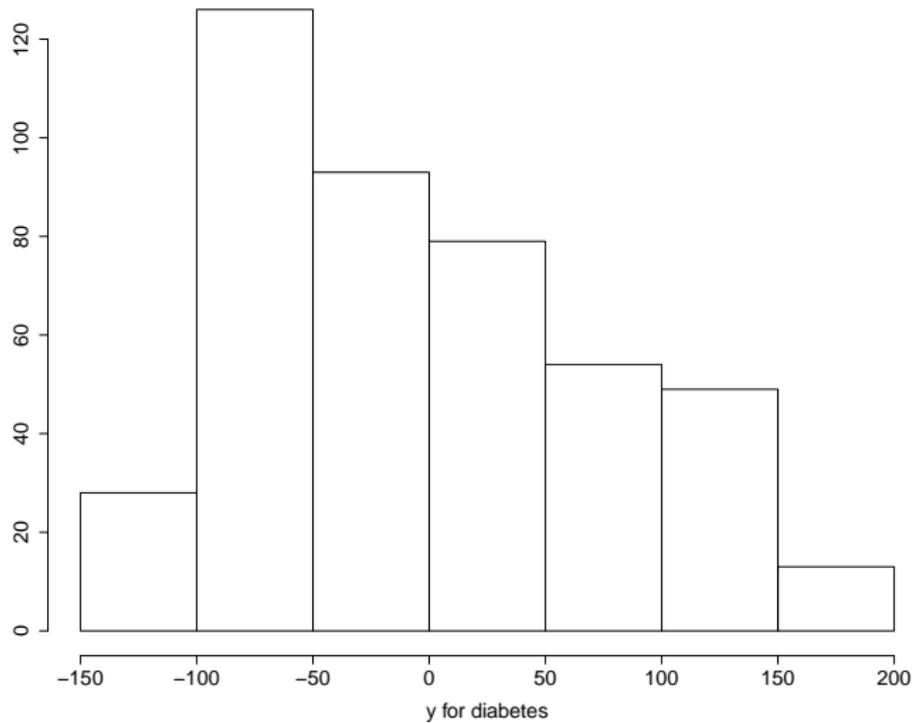
age, sex, body-mass index, average blood pressure, and six blood serum measurements

---plus quadratic terms, giving a total of 64 features.

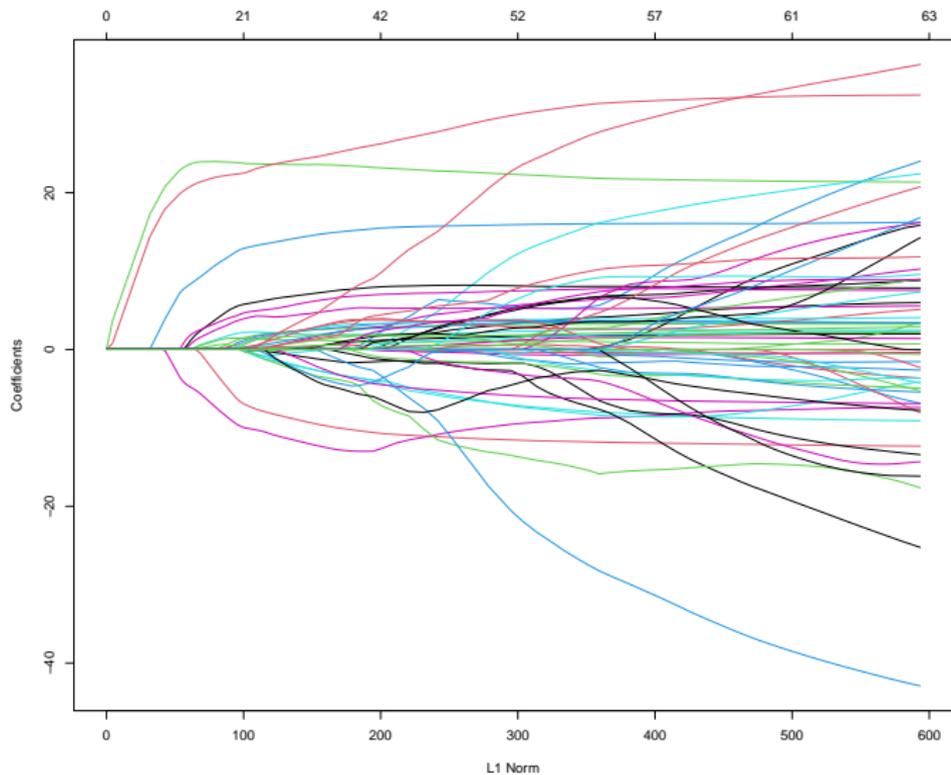
```
> 10+10+10*9/2 #linear + quadratic + interactions  
[1] 65
```

But you don't square sex because it is a binary dummy so you get 64 variables.

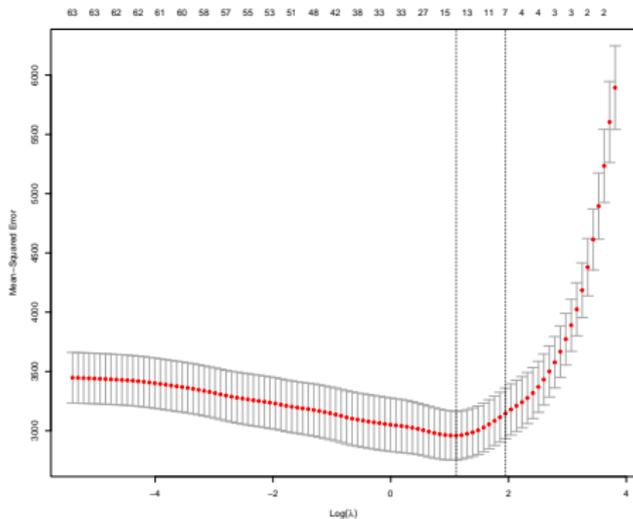
Here is the response.



Here is the Lasso coefficient plot.



Here is the Lasso cv plot.



```
[1] "minlam and minlam (1se) are:  3.0377 7.0175"
```

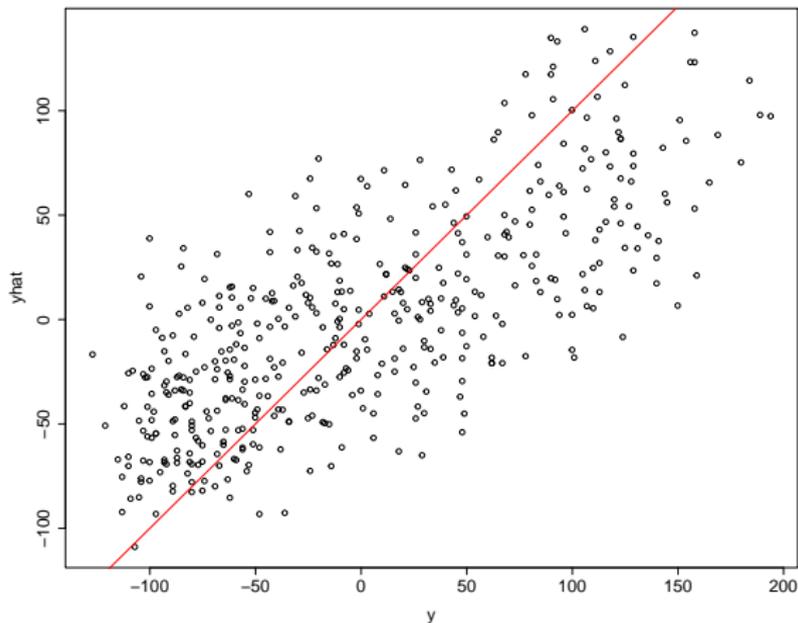
Here are the non-zero coefficients:

| | | | | | | |
|------------|------------|------------|------------|------------|-----------|-----------|
| sex | bmi | map | hdl | ltg | glu | age.2 |
| -5.3240588 | 23.8840329 | 11.9768009 | -8.9267013 | 22.2766341 | 0.8536991 | 0.3510477 |
| bmi.2 | glu.2 | age.sex | age.map | age.ltg | age.glu | bmi.map |
| 1.8401302 | 3.3142418 | 5.1180918 | 1.4271455 | 0.4050495 | 0.5559682 | 4.0729018 |

Wow.

This corresponds to a very simple nonlinear function using the 7 variables sex, bmi, map, hdl, ltg, glu, age.

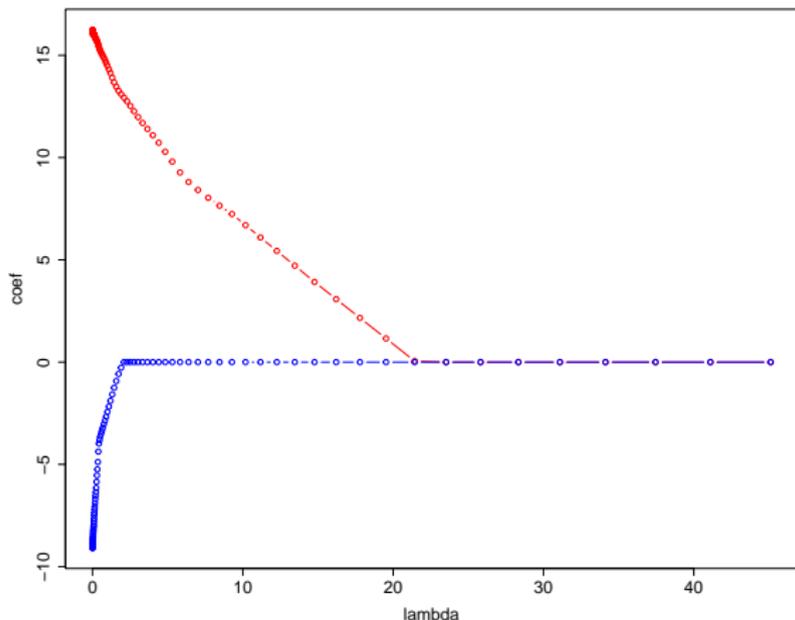
Here are the in-sample fits at the λ chosen by cv.



Can we see the effect of the shrinkage??!

Seeing the Lasso Fit:

Here are the map and tc coefficients as functions of λ .



Notice how they look linear between knots.

Compare to Least-squares:

Suppose you do it the old multiple regression output way.

Note that *we all know you can't do variable selection by seeing which coefficients are significant.*

But that is what most people do.

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) | |
|-------------|------------|------------|---------|----------|-----|
| (Intercept) | -3.710e-08 | 2.532e+00 | 0.000 | 1.0000 | |
| age | 2.415e+00 | 3.120e+00 | 0.774 | 0.4393 | |
| sex | -1.273e+01 | 3.108e+00 | -4.096 | 5.15e-05 | *** |
| bmi | 2.194e+01 | 4.029e+00 | 5.446 | 9.32e-08 | *** |
| map | 1.633e+01 | 3.450e+00 | 4.734 | 3.13e-06 | *** |
| tc | -1.717e+02 | 2.885e+03 | -0.060 | 0.9526 | |
| ldl | 1.444e+02 | 2.535e+03 | 0.057 | 0.9546 | |
| hdl | 5.263e+01 | 1.078e+03 | 0.049 | 0.9611 | |
| tch | 3.568e+00 | 1.313e+01 | 0.272 | 0.7860 | |
| ltg | 8.715e+01 | 9.483e+02 | 0.092 | 0.9268 | |
| glu | 2.988e+00 | 3.352e+00 | 0.891 | 0.3733 | |
| age.2 | 3.223e+00 | 3.308e+00 | 0.974 | 0.3305 | |
| bmi.2 | 2.183e+00 | 3.966e+00 | 0.550 | 0.5823 | |
| map.2 | -4.028e-01 | 3.412e+00 | -0.118 | 0.9061 | |
| tc.2 | 3.175e+02 | 3.361e+02 | 0.945 | 0.3455 | |
| ldl.2 | 1.706e+02 | 2.536e+02 | 0.673 | 0.5016 | |
| hdl.2 | 8.246e+01 | 7.574e+01 | 1.089 | 0.2770 | |
| tch.2 | 3.683e+01 | 2.890e+01 | 1.274 | 0.2034 | |
| ltg.2 | 6.913e+01 | 8.239e+01 | 0.839 | 0.4019 | |
| glu.2 | 5.436e+00 | 4.482e+00 | 1.213 | 0.2260 | |
| age.sex | 7.080e+00 | 3.496e+00 | 2.025 | 0.0435 | * |
| age.bmi | -8.596e-01 | 3.791e+00 | -0.227 | 0.8208 | |
| age.map | 8.825e-01 | 3.633e+00 | 0.243 | 0.8082 | |
| age.tc | -7.566e+00 | 2.939e+01 | -0.257 | 0.7969 | |
| age.ldl | -3.204e+00 | 2.355e+01 | -0.136 | 0.8919 | |
| age.hdl | 9.964e+00 | 1.336e+01 | 0.746 | 0.4563 | |
| age.tch | 8.808e+00 | 1.002e+01 | 0.879 | 0.3798 | |
| age.ltg | 5.937e+00 | 1.066e+01 | 0.557 | 0.5778 | |
| age.glu | 2.980e+00 | 3.827e+00 | 0.779 | 0.4367 | |

| | | | | |
|---------|------------|-----------|--------|--------|
| sex.bmi | 3.077e+00 | 3.710e+00 | 0.829 | 0.4074 |
| sex.map | 4.213e+00 | 3.559e+00 | 1.184 | 0.2373 |
| sex.tc | 2.065e+01 | 2.813e+01 | 0.734 | 0.4634 |
| sex.ldl | -1.680e+01 | 2.233e+01 | -0.752 | 0.4523 |
| sex.hdl | -5.940e+00 | 1.304e+01 | -0.455 | 0.6491 |
| sex.tch | -6.249e+00 | 9.510e+00 | -0.657 | 0.5115 |
| sex.ltg | -5.666e+00 | 1.079e+01 | -0.525 | 0.5996 |
| sex.glu | 2.179e+00 | 3.507e+00 | 0.621 | 0.5348 |
| bmi.map | 7.368e+00 | 4.111e+00 | 1.792 | 0.0739 |
| bmi.tc | -1.438e+01 | 3.181e+01 | -0.452 | 0.6514 |
| bmi.ldl | 1.150e+01 | 2.672e+01 | 0.431 | 0.6670 |
| bmi.hdl | 5.807e+00 | 1.571e+01 | 0.370 | 0.7118 |
| bmi.tch | -1.593e+00 | 1.099e+01 | -0.145 | 0.8849 |
| bmi.ltg | 5.461e+00 | 1.219e+01 | 0.448 | 0.6544 |
| bmi.glu | 1.113e+00 | 4.335e+00 | 0.257 | 0.7975 |
| map.tc | 2.278e+01 | 3.249e+01 | 0.701 | 0.4837 |
| map.ldl | -1.556e+01 | 2.735e+01 | -0.569 | 0.5698 |
| map.hdl | -8.919e+00 | 1.474e+01 | -0.605 | 0.5455 |
| map.tch | -2.776e+00 | 9.457e+00 | -0.294 | 0.7693 |
| map.ltg | -7.371e+00 | 1.295e+01 | -0.569 | 0.5696 |
| map.glu | -6.356e+00 | 4.348e+00 | -1.462 | 0.1447 |
| tc.ldl | -4.435e+02 | 5.605e+02 | -0.791 | 0.4294 |
| tc.hdl | -1.872e+02 | 1.817e+02 | -1.030 | 0.3036 |
| tc.tch | -1.050e+02 | 8.390e+01 | -1.252 | 0.2113 |
| tc.ltg | -1.810e+02 | 6.270e+02 | -0.289 | 0.7730 |
| tc.glu | -8.395e+00 | 2.836e+01 | -0.296 | 0.7673 |
| ldl.hdl | 1.258e+02 | 1.508e+02 | 0.835 | 0.4045 |
| ldl.tch | 5.747e+01 | 7.002e+01 | 0.821 | 0.4124 |
| ldl.ltg | 1.320e+02 | 5.219e+02 | 0.253 | 0.8004 |
| ldl.glu | 4.077e+00 | 2.405e+01 | 0.170 | 0.8655 |
| hdl.tch | 5.659e+01 | 4.773e+01 | 1.186 | 0.2365 |
| hdl.ltg | 6.988e+01 | 2.195e+02 | 0.318 | 0.7504 |
| hdl.glu | 1.036e+01 | 1.413e+01 | 0.733 | 0.4640 |
| tch.ltg | 1.856e+01 | 2.975e+01 | 0.624 | 0.5330 |
| tch.glu | 1.122e+01 | 1.119e+01 | 1.003 | 0.3167 |
| ltg.glu | 3.977e+00 | 1.261e+01 | 0.316 | 0.7525 |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

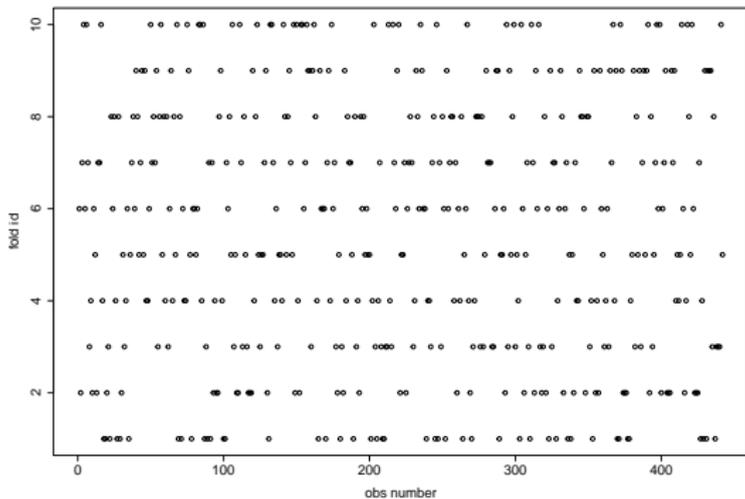
Only 4 variables are “significant”

Comparing Lasso, Ridge, and Elastic-Net:

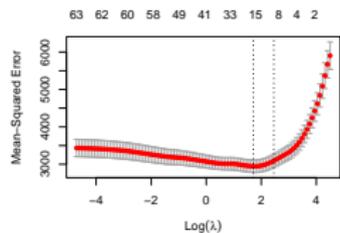
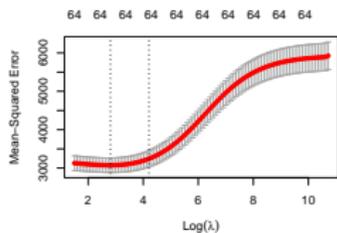
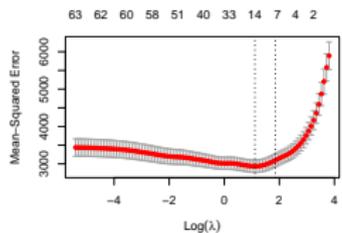
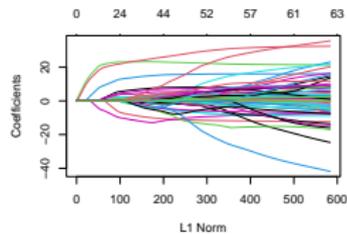
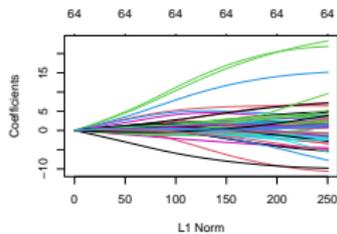
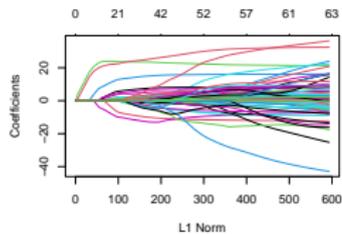
Let's try Lasso, Ridge, and elastic-net with $\alpha = .5$ and see which seems to work best.

To do this we will have to give `cv.glmnet` a prechosen set of `cv` folds.

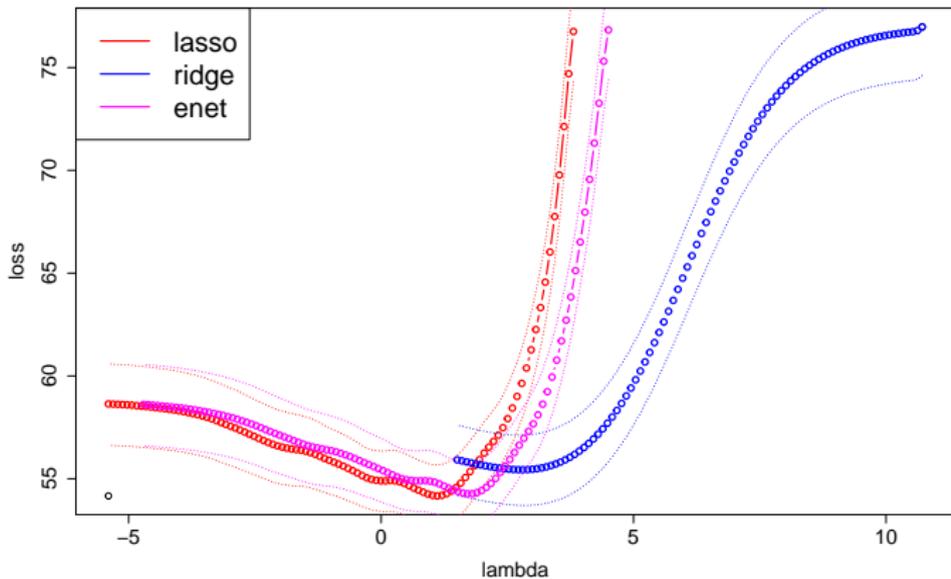
Here is a set of folds for 10-fold cv.



Left to right, Lasso, Ridge, Elastic(.5).



I took the square root of the loss measures so that we are looking at RMSE.



It looks like Lasso and Enet are similar and better than Ridge, but, as a *practical matter* they are all about the same.

Seeing the Ridge Fit:

To check our understanding of the Ridge fit, we let $x_1 = bmi$ and $x_{2.1}$ be the standardized residuals from regressing map on bmi .

```
x1 = x[,3]; x2 = x[,4]
x2.1 = x2 - (sum(x1*x2)/sum(x1^2))*x1
x2.1 = scale(x2.1)
```

Then we should have

$$\hat{\beta}_\lambda = \frac{\hat{\beta}}{(1 + c \lambda)}$$

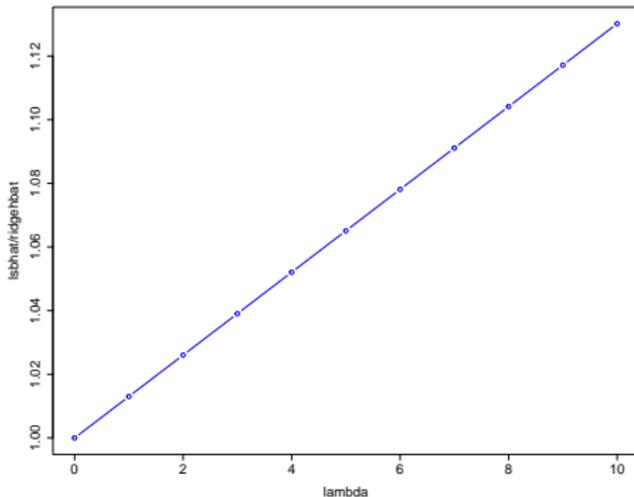
where

$$\hat{\beta} = \frac{\langle x, y \rangle}{\langle x, x \rangle}$$

and c depends on exactly how the penalty was scaled.

Below is a plot of

$$\frac{\hat{\beta}}{\hat{\beta}_\lambda} \text{ versus } \lambda.$$



Does indeed look like

$$\frac{\hat{\beta}}{\hat{\beta}_\lambda} = 1 + c \lambda.$$

BIC and Forward Step-Wise:

Let's try BIC and forward stepwise.

First note that if you run the regression without the transformations, that is with just the 10 x variables, then

$BIC = 3584.648$ (with 11 parameters).

With all the transformations

$BIC = 3839.201$ (with 65 parameters, counting the intercept).

If we run forwards step-wise using BIC as our greedy loss and stopping criterion we get the model:

Call:

```
lm(formula = y ~ bmi + ltg + map + age.sex + bmi.map + hdl +  
sex, data = ddf)
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|----------|---------|--------|--------|---------|
| -150.077 | -39.269 | -1.481 | 32.423 | 139.891 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|------------|------------|---------|--------------|
| (Intercept) | -3.710e-08 | 2.523e+00 | 0.000 | 1.000000 |
| bmi | 2.481e+01 | 3.039e+00 | 8.165 | 3.53e-15 *** |
| ltg | 2.406e+01 | 3.069e+00 | 7.840 | 3.53e-14 *** |
| map | 1.477e+01 | 2.952e+00 | 5.004 | 8.16e-07 *** |
| age.sex | 8.892e+00 | 2.552e+00 | 3.484 | 0.000545 *** |
| bmi.map | 8.385e+00 | 2.552e+00 | 3.286 | 0.001100 ** |
| hdl | -1.324e+01 | 3.063e+00 | -4.323 | 1.91e-05 *** |
| sex | -1.133e+01 | 2.811e+00 | -4.029 | 6.61e-05 *** |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 53.05 on 434 degrees of freedom

Multiple R-squared: 0.534, Adjusted R-squared: 0.5265

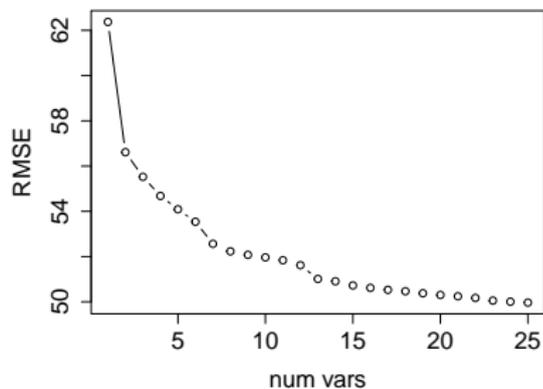
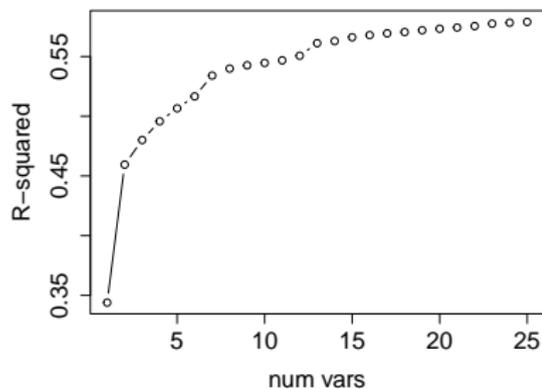
F-statistic: 71.05 on 7 and 434 DF, p-value: < 2.2e-16

which has 7 terms and the variables bmi, ltg, map, age, sex, hdl.

Almost the same as Lasso!

BIC for forward model is 3551.2, with 8 parameters.

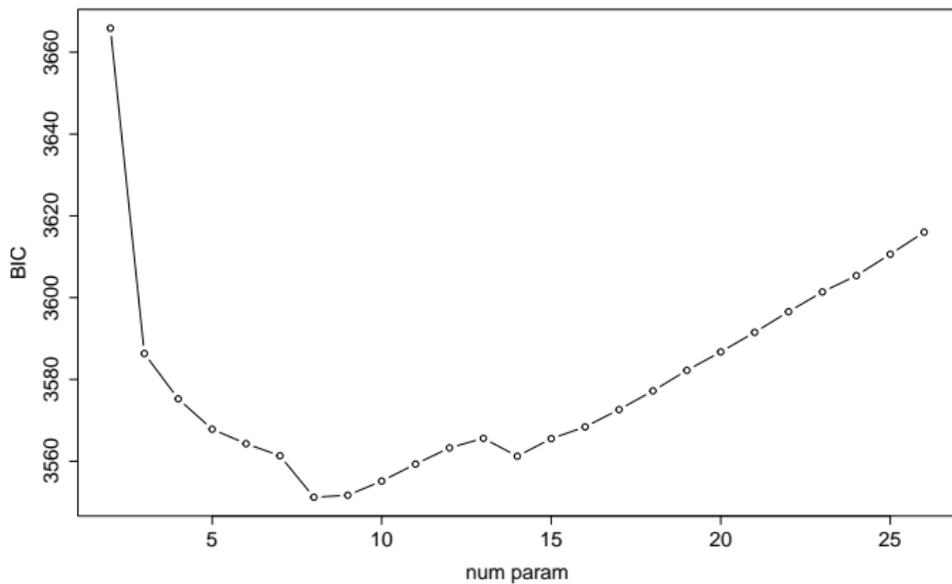
Here are the in-sample R-squared and RMSE from the forward-stepwise:



And the names of the variables as they come in in:

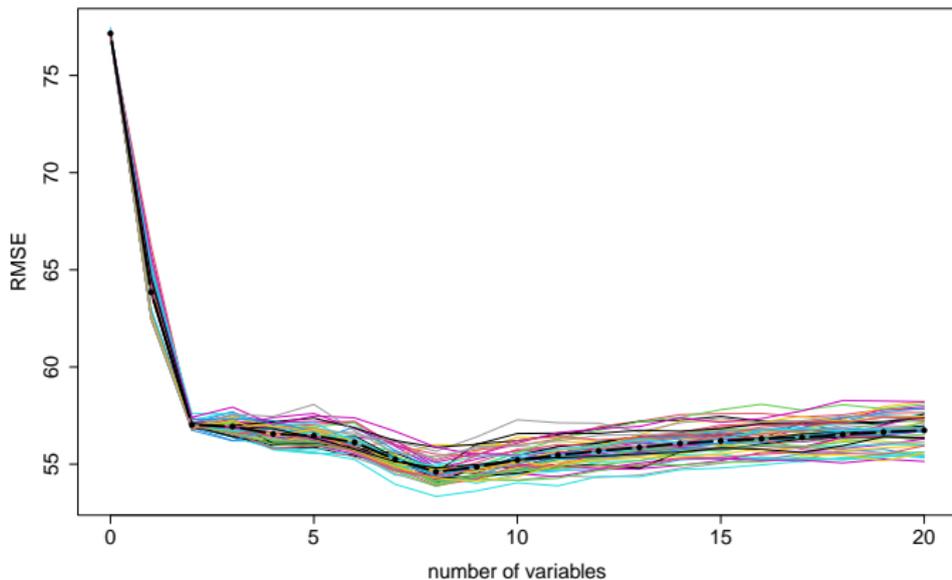
```
[1] "bmi"      "ltg"      "map"      "age.sex"  "bmi.map"  "hdl"      "sex"
[8] "glu.2"    "age.2"    "map.glu"  "tc"       "ldl"       "ltg.2"    "age.ldl"
[15] "age.tc"   "sex.map"  "glu"      "tch"      "sex.tch"   "sex.bmi"  "tc.tch"
[22] "tch.glu"  "hdl.glu"  "map.tc"   "bmi.ltg"
```

Here are the BIC's of the models found by forward stepwise.



Here are the RMSE's from 50 runs of 10-fold cv using forwards step.

Remember, our knob is how many steps to take = number of variables



55 again !!!

So, the methods are giving very similar results,
except for the p-values stuff.