Chapter 6 in ISL: Regularization

DJM

28 April 2020

Regularization

- Another way to control bias and variance is through ~~regularization~~ or ~~shrinkage~~.
- Rather than selecting a few predictors that seem reasonable, maybe trying a few combinations, use them all.
- I mean ~~ALL~~.
- But, make your estimates of β "smaller"

Some optimization terms

- An optimization problem has 2 components:
  1. The "Objective function": e.g. $\frac{1}{n}\sum_i (y_i - x_i'\beta)^2$.
  2. The "constraint": e.g. "fewer than 5 non-zero entries in β".
- A constrained minimization problem is written

$$\min_\beta f(\beta) \text{ subject to } C(\beta)$$

- $f(\beta)$ is the objective function
- $C(\beta)$ is the constraint

Regularization

One way to do this for regression is to solve (say):

$$\min_\beta \frac{1}{n}\sum_i (y_i - x_i'\beta)^2$$
$$\text{s.t.} \sum_j \beta_j^2 < t$$

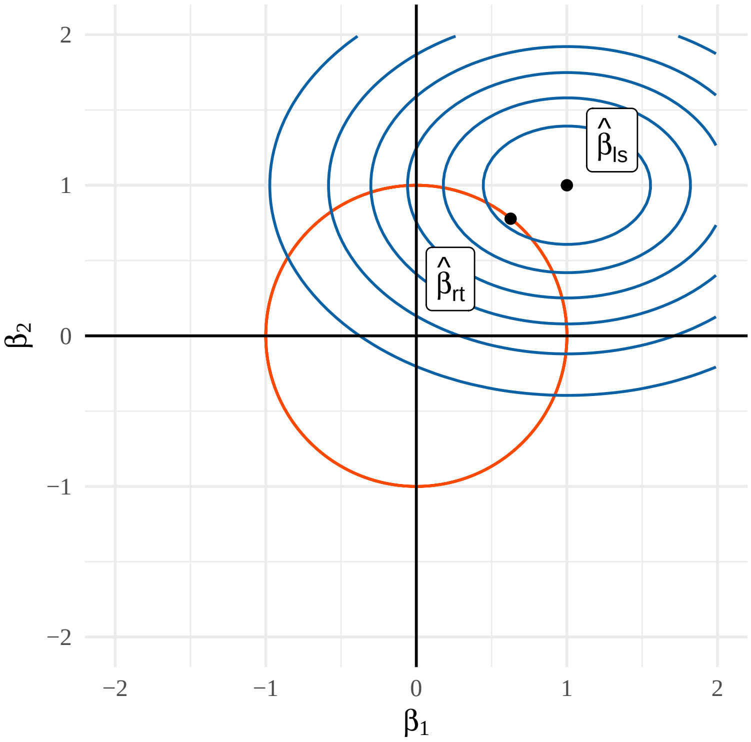
for some $t > 0$.

- This is called "ridge regression".
- The ~~minimizer~~ of this problem is called $\widehat{\beta}_{r,t}$

Compare this to least squares:

$$\min_\beta \frac{1}{n}\sum_i (y_i - x_i'\beta)^2$$
$$\text{s.t.} \beta \in \mathbb{R}^p$$

Geometry of ridge regression (2 dimensions)



Ridge regression

An equivalent way to write

$$\widehat{\beta}_{r,t} = \arg \min_{||\beta||_2^2 \le t} \frac{1}{n} \sum_i (y_i - x_i'\beta)^2$$

is in the ~~Lagrangian~~ form

$$\widehat{\beta}_{r,\lambda} = \arg \min_{\beta} \frac{1}{n} \sum_i (y_i - x_i'\beta)^2 + \lambda ||\beta||_2^2.$$

For every λ there is a unique t (and vice versa) that makes

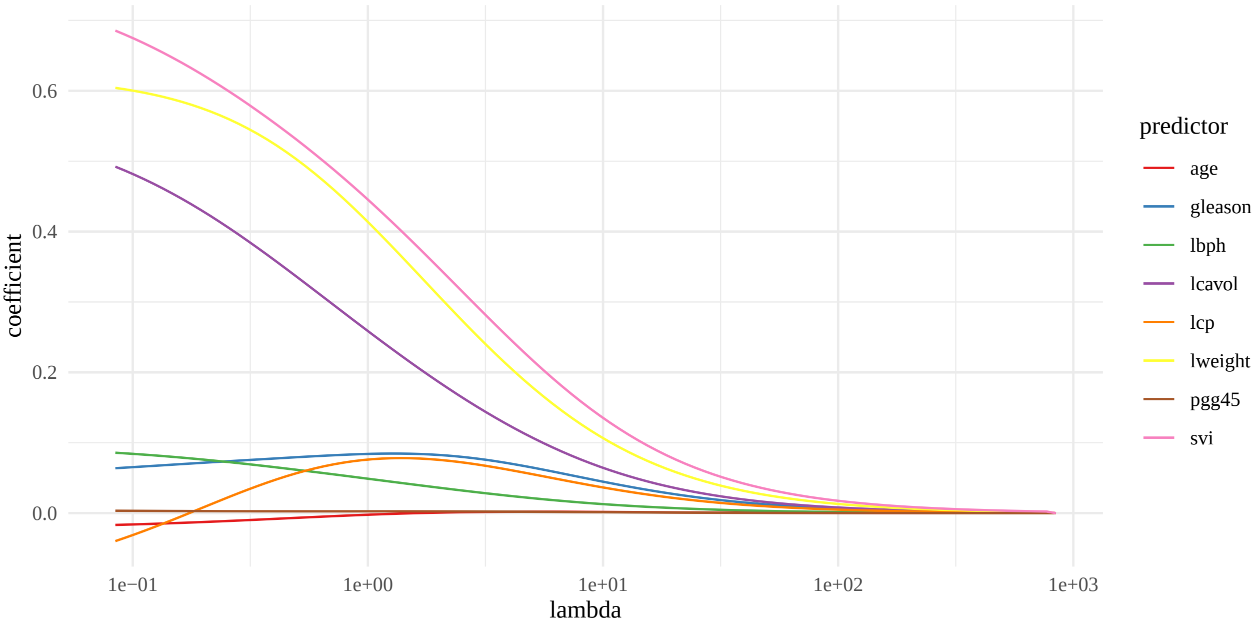
$$\widehat{\beta}_{r,t} = \widehat{\beta}_{r,\lambda}$$

Observe:

- $\lambda = 0$ (or $t = \infty$) makes $\widehat{\beta}_{r,\lambda} = \widehat{\beta}_{ls}$
- Any $\lambda > 0$ (or $t < \infty$) penalizes larger values of β, effectively shrinking them.

Note: λ and t are known as ~~tuning parameters~~

Ridge regression path



Regularization and rescaling

Least squares is invariant to rescaling

Let's multiply our design matrix by a factor of 10 to get $\widetilde{\mathbb{X}} = 10\mathbb{X}$.

Then:

$$\widetilde{\beta}_{\text{ls}} = (\widetilde{\mathbb{X}}^{\top}\widetilde{\mathbb{X}})^{-1}\widetilde{\mathbb{X}}^{\top}Y = \frac{1}{10}(\widetilde{\mathbb{X}}^{\top}\widetilde{\mathbb{X}})^{-1}\widetilde{\mathbb{X}}^{\top}Y = \frac{\widehat{\beta}_{\text{ls}}}{10}$$

So, multiplying our data by ten just results in our estimates being reduced by one tenth.
Hence, any prediction is left unchanged:

$$\widetilde{\mathbb{X}}\widetilde{\beta}_{\text{ls}} = \mathbb{X}\widehat{\beta}_{ls}$$

This means, for instance, if we have a covariate measured in miles, then we will get the "same" answer if we change it to kilometers

Least squares is invariant to rescaling: example

```r
n = 20
set.seed(2018-04-10)
X = matrix(runif(2*n,0,1), ncol=2)
Y = X %*% c(.5,1.5) + rnorm(n,0,.25)
Xtilde   = 2*X
Ytilde   = Y - mean(Y)
summary(lm(Y~X))$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 0.08267477  0.1479918 0.5586444 5.836887e-01
## X1          0.33116021  0.2329056 1.4218645 1.731531e-01
## X2          1.43583489  0.2033167 7.0620598 1.908643e-06
```

```r
summary(lm(Y~Xtilde))$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 0.08267477  0.1479918 0.5586444 5.836887e-01
## Xtilde1     0.16558011  0.1164528 1.4218645 1.731531e-01
## Xtilde2     0.71791745  0.1016584 7.0620598 1.908643e-06
```

```r
summary(lm(Ytilde~Xtilde))$coefficients # note what happened here!!
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -0.8103178  0.1479918 -5.475425 4.104055e-05
## Xtilde1      0.1655801  0.1164528  1.421864 1.731531e-01
## Xtilde2      0.7179174  0.1016584  7.062060 1.908643e-06
```

Ridge regression (and other regularized methods) is not

```r
library(MASS)
lm.ridge(Y~X, lambda=1)$coef
```

```
##         X1         X2
## 0.08864934 0.41708721
```

```r
lm.ridge(Y~Xtilde, lambda=1)$coef
```

```
##    Xtilde1    Xtilde2
## 0.08864934 0.41708721
```

```r
lm.ridge(Ytilde~Xtilde, lambda=1)$coef
```

```
##    Xtilde1    Xtilde2
## 0.08864934 0.41708721
```

- This actually works ok because `lm.ridge` automatically scales every column of $\mathbb{X}$ to have mean zero and Euclidean norm 1.
- It also centers Y.
- Together, this means there is no intercept. (We don't penalize the intercept)
- In `R`: `scale(X)` defaults to mean 0, SD 1. But you can change either.
- Another version is in the package `glmnet`. More on this in a bit.

Solving the minimization

- One nice thing about ridge regression is that it has a closed-form solution (like OLS)

$$\widehat{\beta}_{r,\lambda} = (\mathbb{X}'\mathbb{X} + \lambda I)^{-1}\mathbb{X}'Y$$

- This is easy to calculate in `R` for any λ.
- However, computations and interpretation are simplified if we examine the Singular Value Decomposition of $\mathbb{X} = UDV'$.
- Then,

$$\widehat{\beta}_{r,\lambda} = (\mathbb{X}'\mathbb{X} + \lambda I)^{-1}\mathbb{X}'Y = (VD^2V' + \lambda I)^{-1}VDU'Y = V(D^2 + \lambda I)^{-1}DU'Y.$$

- For computations, now we only need to invert a diagonal matrix.
- For interpretations, we can compare this to OLS:

$$\widehat{\beta}_{ls} = (\mathbb{X}'\mathbb{X})^{-1}\mathbb{X}'Y = (VD^2V')^{-1}VDU'Y = VD^{-2}DU'Y = VD^{-1}U'Y$$

- Notice that $\widehat{\beta}_{ls}$ depends on d_j/d_j^2 while $\widehat{\beta}_{r,\lambda}$ depends on $d_j/(d_j^2 + \lambda)$.
- Ridge regression makes the coefficients smaller relative to OLS.
- But if $\mathbb{X}$ has small singular values, ridge regression compensates with λ in the denominator.

Ridge regression and multicollinearity

Multicollinearity is a phenomenon in which a combination of predictor variables is extremely similar to another predictor variable. Some comments:

- A better phrase that is sometimes used is "$\mathbb{X}$ is ill-conditioned"
- It means that one of its columns is nearly (or exactly) a linear combination of other columns. This is sometimes known as "(numerically) rank-deficient".
- If $\mathbb{X} = UDV'$ is ill-conditioned, then some elements of D are nearly zero
- If we form $\widehat{\beta}_{ls} = VD^{-1}U'Y$, then we see that the small entries of D are now huge (due to the inverse). This in turn creates a huge variance.
- Recall: $\mathbb{V}\widehat{\beta}_{ls} = \sigma^2(\mathbb{X}'\mathbb{X})^{-1} = \sigma^2 VD^{-2}V'$

Ridge Regression fixes this problem by preventing the division by a near zero number

Conclusion: $(\mathbb{X}^\top\mathbb{X})^{-1}$ can be really unstable, while $(\mathbb{X}^\top\mathbb{X} + \lambda I)^{-1}$ is not.

Can we get the best of both worlds?

To recap:

- Deciding which predictors to include, adding quadratic terms, or interactions is ~~model selection~~.
- Ridge regression provides regularization, which trades off bias and variance and also stabilizes multicollinearity.

Ridge regression: $\min ||\mathbb{Y} - \mathbb{X}\beta||_2^2$ subject to $||\beta||_2^2 \leq t$

Best linear regression model: $\min ||\mathbb{Y} - \mathbb{X}\beta||_2^2$ subject to $||\beta||_0 \leq t$

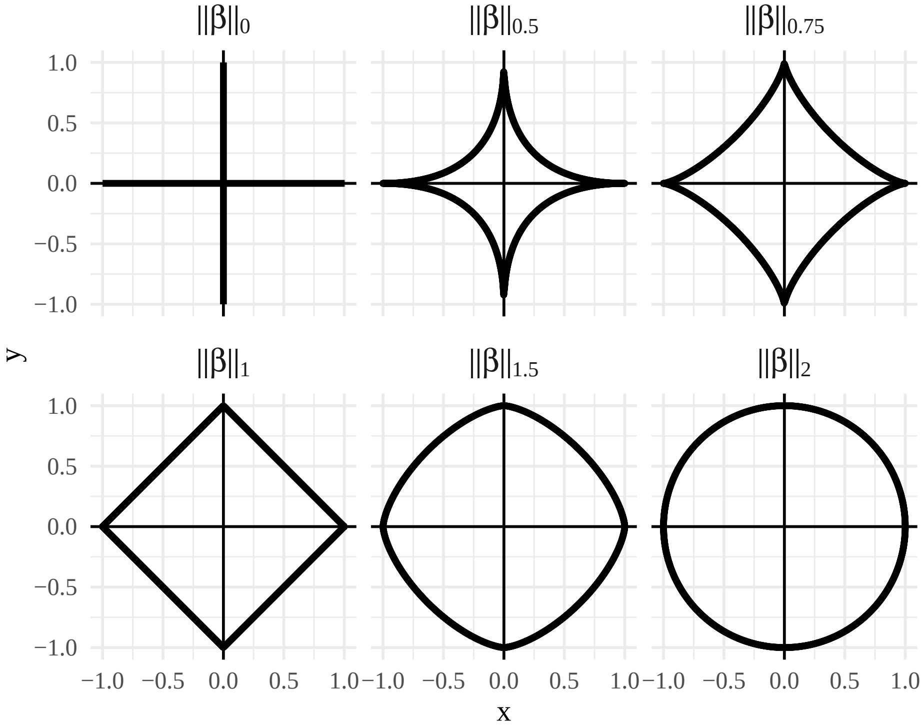
$||\beta||_0$ is the number of nonzero elements in β

Finding the best linear model is a nonconvex optimization problem (In fact, it is NP-hard)

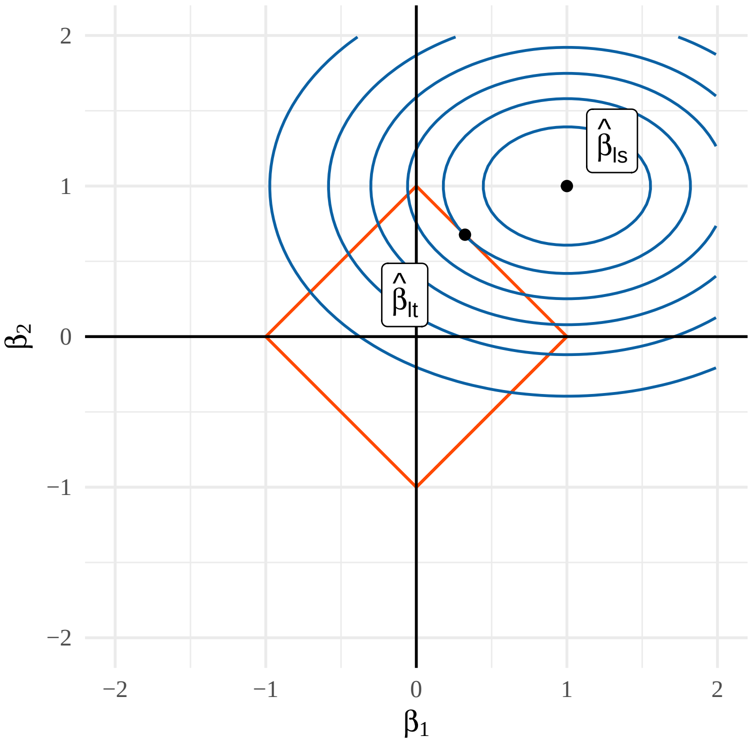
Ridge regression is convex (easy to solve), but doesn't do model selection

Can we somehow "interpolate" to get both?

Geometry of convexity



The best of both worlds



This regularization set. . .

- … is convex (computationally efficient)
- … has corners (performs model selection)

The lasso

ℓ_1-regularized regression

Known as

- "lasso"
- "basis pursuit"

The estimator satisfies

$$\widehat{\beta}_{l,t} = \arg\min_{||\beta||_1 \leq t} ||\mathbb{Y} - \mathbb{X}\beta||_2^2$$

In its corresponding Lagrangian dual form:

$$\widehat{\beta}_{l,\lambda} = \arg\min_{\beta} ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1$$

Lasso

While the ridge solution can be easily computed

$$\widehat{\beta}_{r,\lambda} = \arg\min_{\beta} ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_2^2 = (\mathbb{X}^\top\mathbb{X} + \lambda I)^{-1}\mathbb{X}^\top Y$$

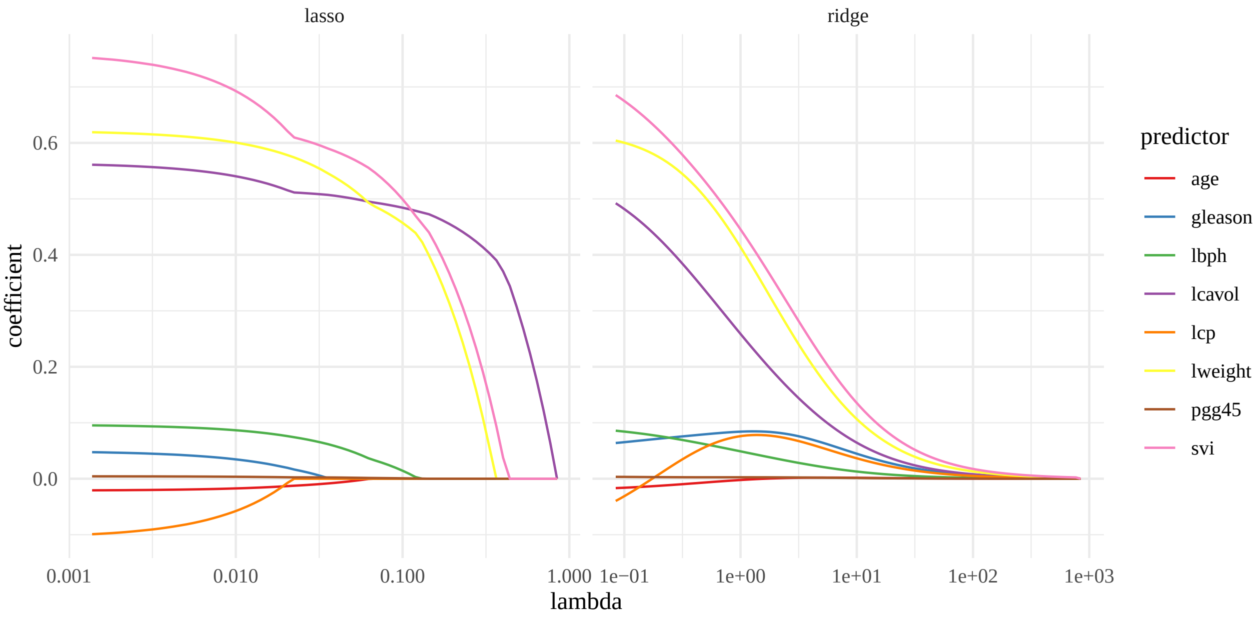
the lasso solution

$$\widehat{\beta}_{l,\lambda} = \arg\min_{\beta} ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1 = ??$$

doesn't have a closed form solution.

However, because the optimization problem is convex, there exist efficient algorithms for computing it

Coefficient path: ridge vs lasso



Packages

There are two main `R` implementations for finding lasso

- Using `glmnet`: `lasso.out = glmnet(X, Y, alpha=1)`.

- Setting `alpha=0` gives ridge regression (as does `lm.ridge` in the `MASS` package)
- Setting `alpha` $\in (0, 1)$ gives a method called the "elastic net" which combines ridge regression and lasso.
- Alternatively, there is `lars`: `lars.out = lars(X, Y)`
- `lars` also does other things called "Least angle", "forward stepwise", and "forward stagewise" regression

Two packages

1. `lars` (this is the first one)
2. `glmnet` (this one is faster)

These use different algorithms, but both compute the ~~path~~ for a range of λ.

`lars` starts with an empty model and adds coefficients until saturated. The sequence of λ's comes from the nature of the optimization problem.

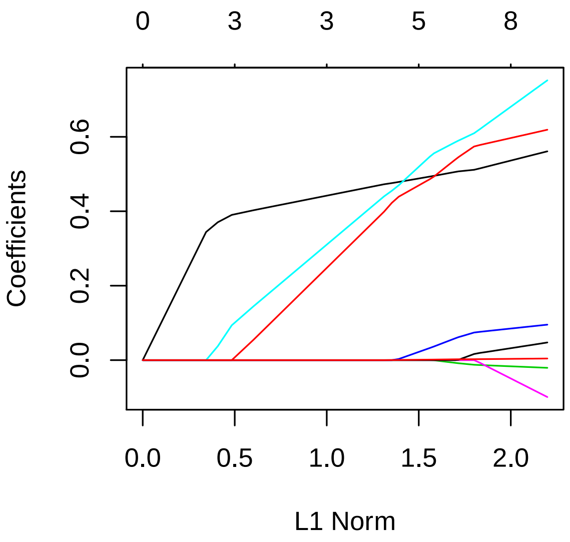
`glmnet` starts with an empty model and examines each value of λ using previous values as "warm starts". It is generally much faster than `lars` and uses lots of other tricks (as well as compiled code) for extra speed.

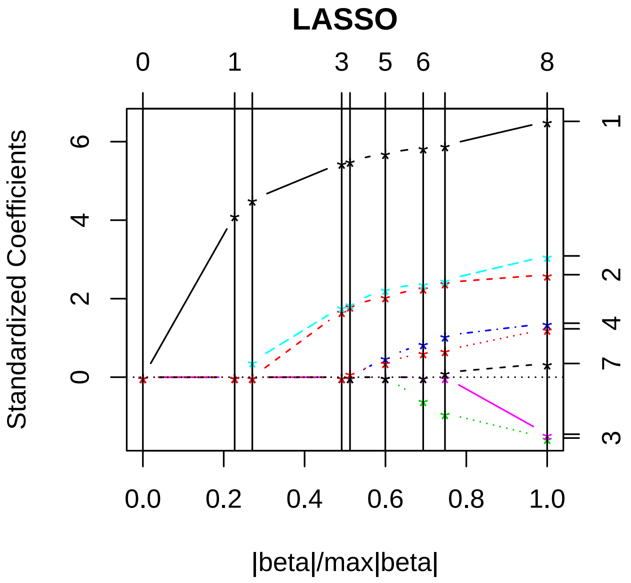
The path returned by `lars` as more useful than that returned by `glmnet`.

Lasso paths

```r
lasso = glmnet(X,Y)
lars.out = lars(X,Y)
par(mfrow=c(1,2))
```

```r
plot(lasso)
plot(lars.out,main='')
```





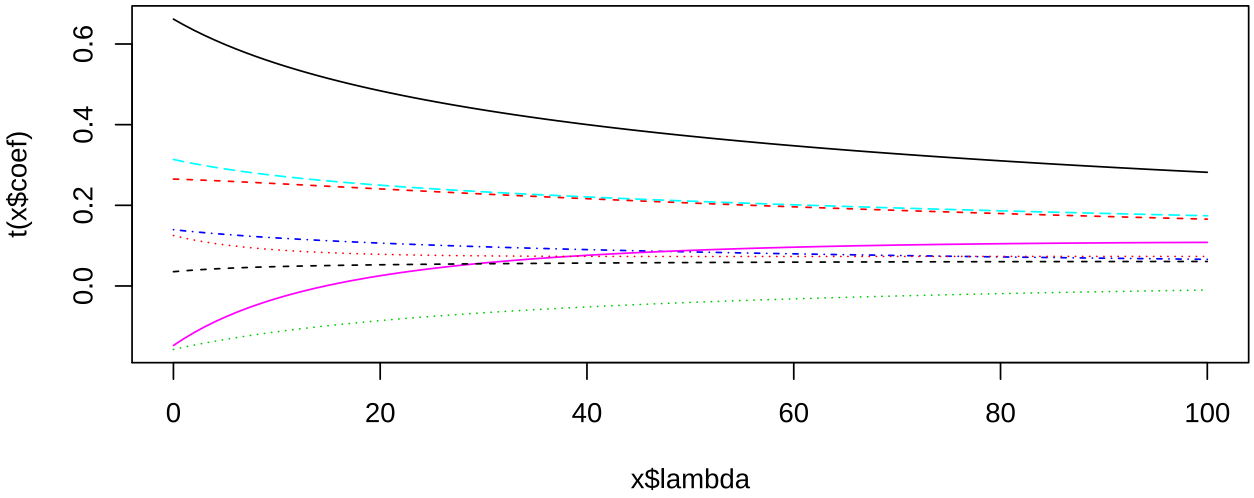
Model selection

Choosing the lambda

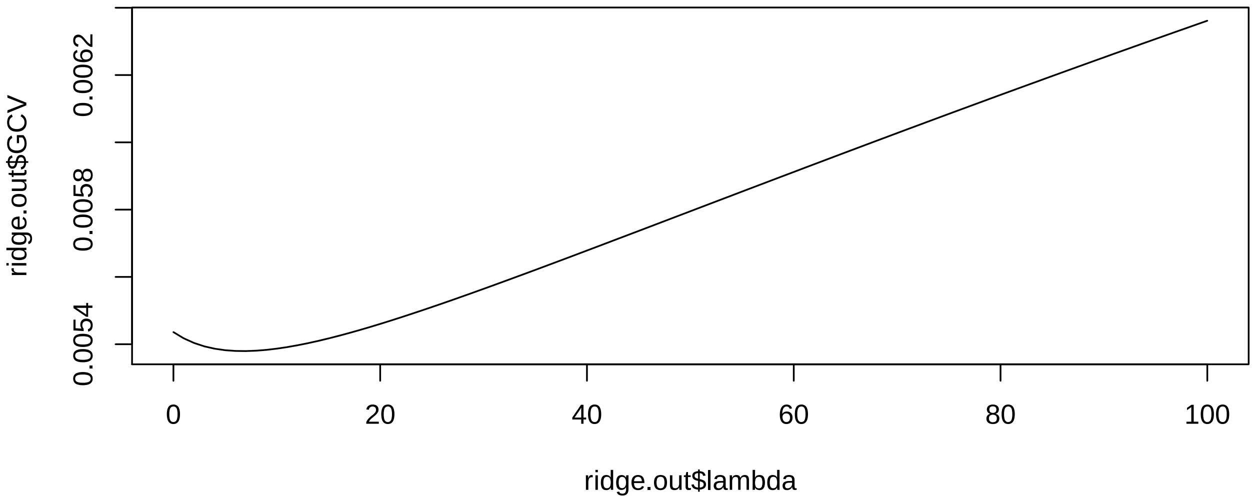
- You have to choose λ in lasso or in ridge regression
- lasso selects a model (by setting coefficients to zero), but the value of λ determines how many/which.
- All of these packages come with CV built in.
- However, the way to do it differs from package to package (whomp whomp)

Ridge regression, `lm.ridge` version

```r
par(mfrow=c(1,1))
# 1. Estimate the model (note, this uses a formula, and you must supply lambda)
ridge.out = lm.ridge(lpsa~.-train, data=prostate, lambda = 0:100)
# 2. Plot it
plot(ridge.out)
```



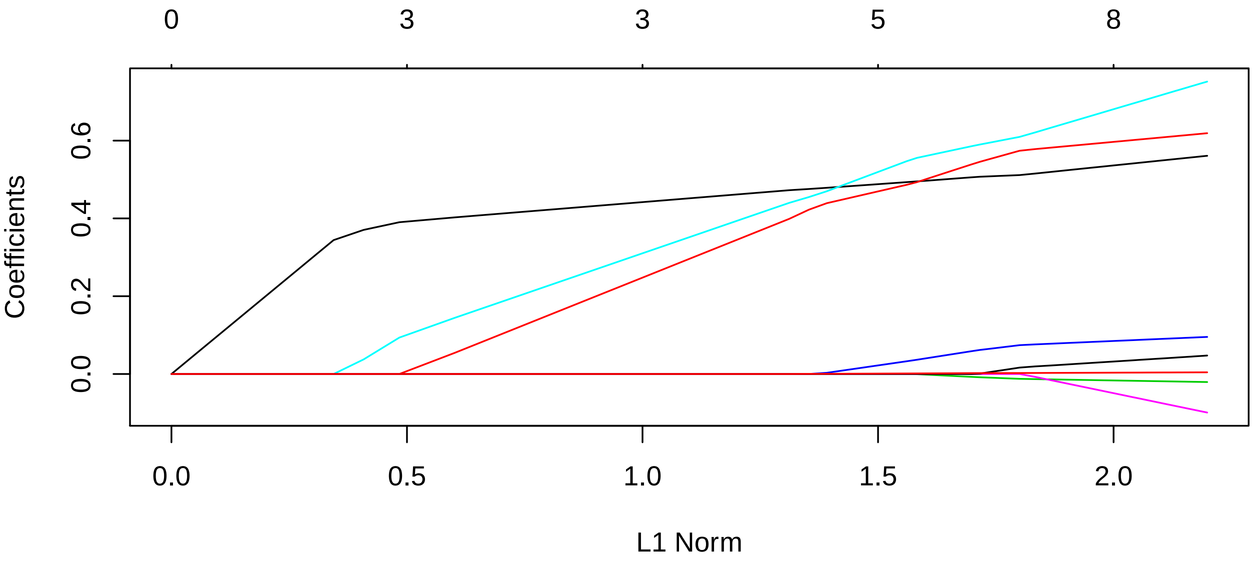
```
# (2a). If you chose lambda poorly, this will look bad, try again
# 3. Choose lambda using GCV
plot(ridge.out$lambda,ridge.out$GCV,ty='l')
```



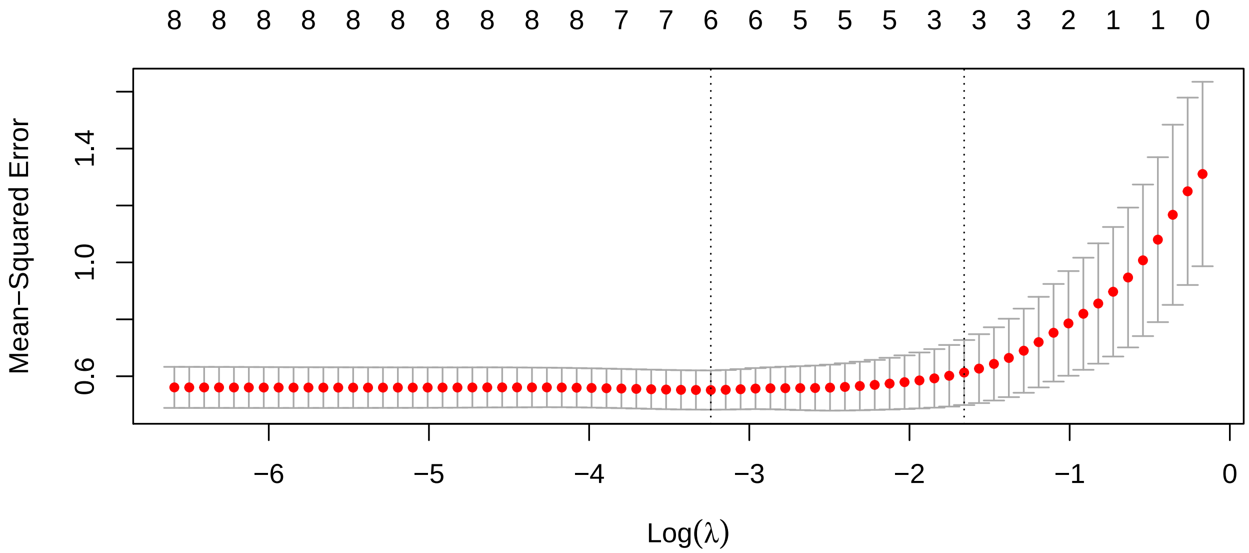
```
# 4. If there's a minimum, FIND IT, else try again
best.lam = which.min(ridge.out$GCV)
# 5. Return the coefs/predictions for the best model
coefs = coefficients(ridge.out)[best.lam,]
preds = as.matrix(dplyr::select(prostate,-lpsa,-train)) %*% coefs[-1] + coefs[1]
```

glmnet version (lasso or ridge)

```
# 1. Estimate cv and model at once, no formula version
lasso.glmnet = cv.glmnet(X,Y)
# 2. Plot the coefficient path
plot(lasso.glmnet$glmnet.fit) # the glmnet.fit == glmnet(X,Y)
```



```
# 3. Choose lambda using CV
plot(lasso.glmnet) #a different plot method for the cv fit
```



```
# 4. If the dashed lines are at the boundaries, redo it with a better set of lambda
best.lam = lasso.glmnet$lambda.min # the value, not the location (or lasso$lambda.1se)
# 5. Return the coefs/predictions for the best model
coefs.glmnet = coefficients(lasso.glmnet, s = best.lam)
preds.glmnet = predict(lasso.glmnet, newx = X, s = best.lam) # must supply `newx`
```

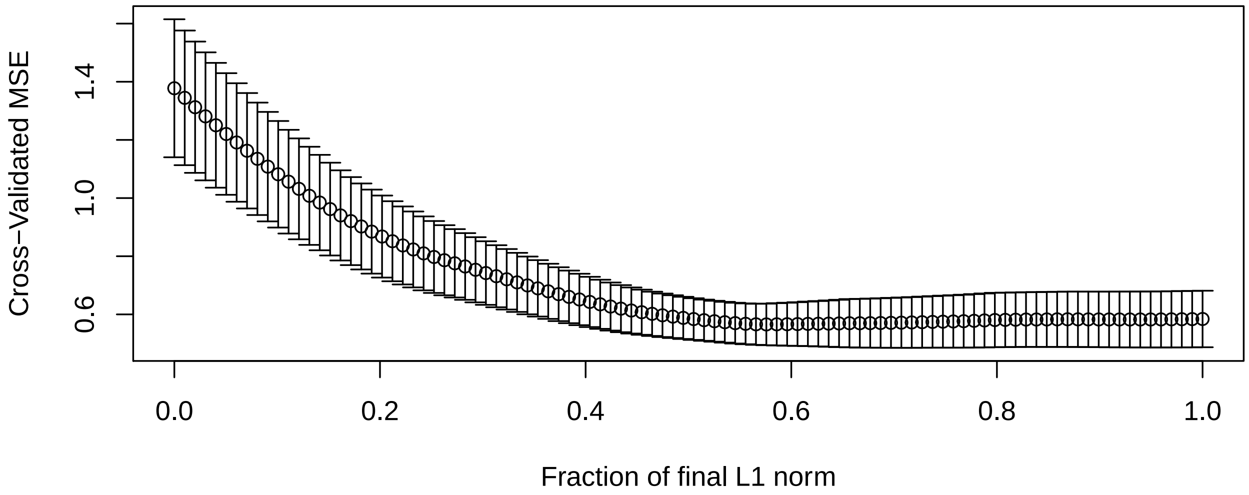
lars version

This is incredibly difficult to cross-validate.

The path changes from fold to fold, so things can get hairy.

In principle, the following should work.

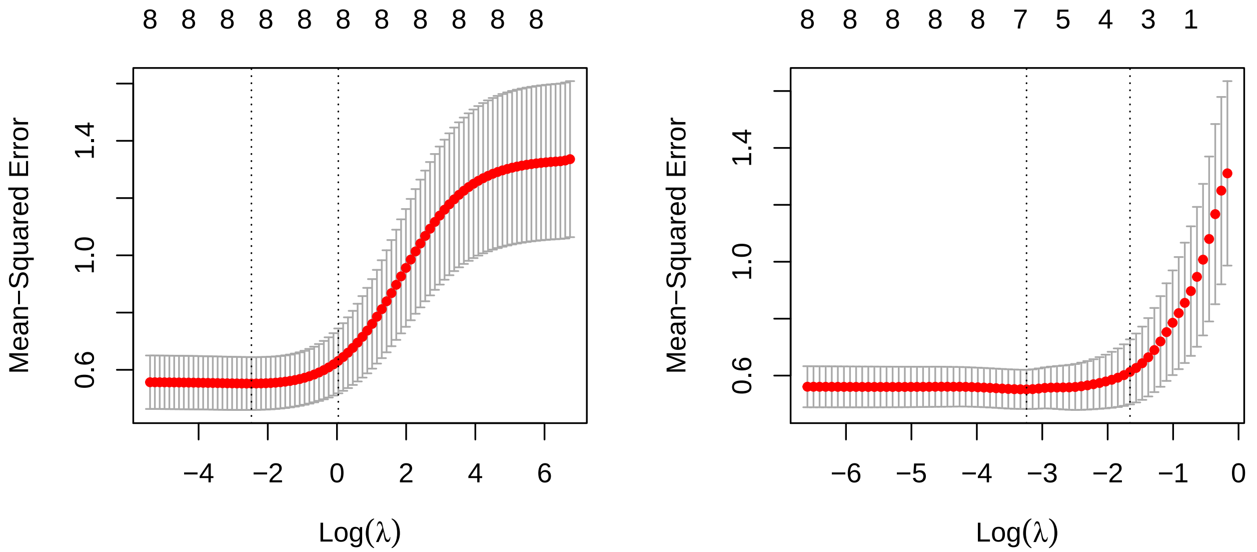
```
# 1. Estimate cv, no formula version
lasso.lars.cv = cv.lars(X,Y) # also plots it
```



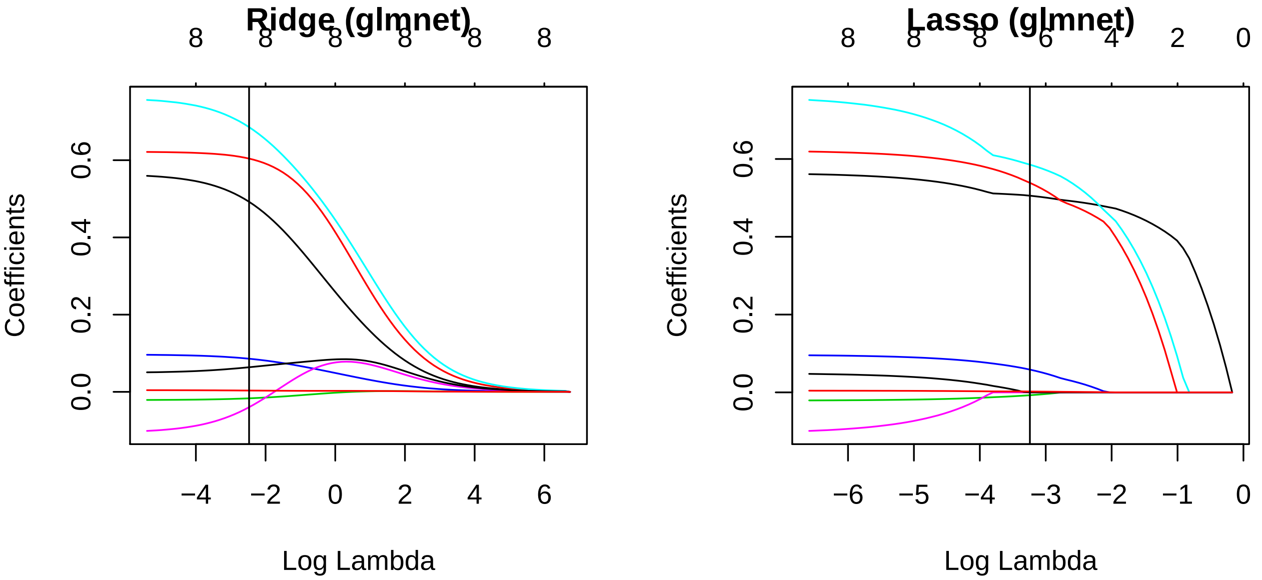
```
# 2. Choose lambda using CV
best.lam.lars = lasso.lars.cv$index[which.min(lasso.lars.cv$cv)] # the location, not the value
# 3. Estimate the lasso and plot
lasso.lars = lars(X,Y) # still the whole path
# 5. Return the coefs/predictions for the best model
coefs.lars = coefficients(lasso.lars, s = best.lam.lars, mode='fraction') # annoying mode argument is r
preds.lars = predict(lasso.lars, newx=X, s = best.lam.lars, mode='fraction') # must supply `newx`
```

Paths with chosen lambda (lasso and ridge)

```
ridge.glmnet = cv.glmnet(X,Y,alpha=0,lambda.min.ratio=1e-10) # added to get a minimum
par(mfrow=c(1,2))
plot(ridge.glmnet)
plot(lasso.glmnet)
```



```
best.lam.ridge = ridge.glmnet$lambda.min
plot(ridge.glmnet$glmnet.fit,xvar='lambda', main='Ridge (glmnet)')
abline(v=log(best.lam.ridge))
plot(lasso.glmnet$glmnet.fit,xvar='lambda', main='Lasso (glmnet)')
abline(v=log(best.lam))
```


Ridge (glmnet)
Coefficients
Log Lambda
Lasso (glmnet)
Coefficients
Log Lambda