Benefits of shrinkage/bias

In "high-dimensions" (p > 2)

Shrinking estimates/models toward a pre-specified point

Stein "paradox" and bias

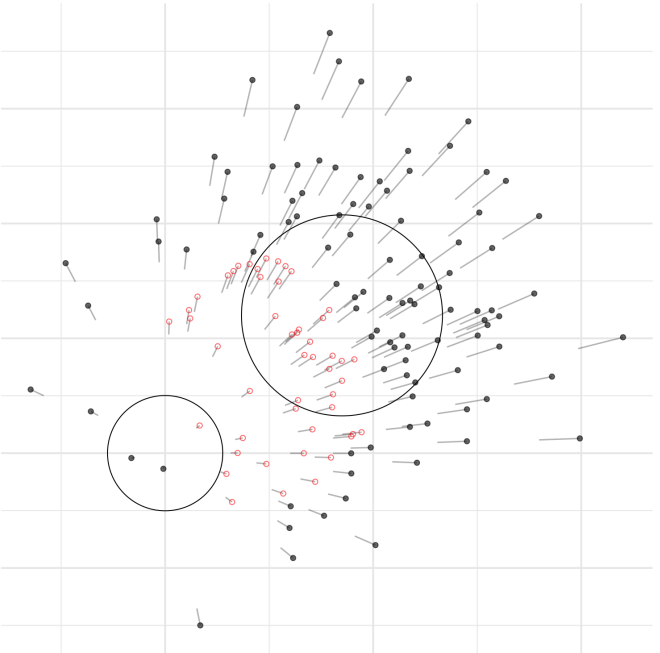
Estimating $\mu \in \mathbb{R}^p$ from an i.i.d. sample $\mathbf{Y}_1, \ldots, \mathbf{Y}_n \sim N(\mu, \sigma^2 I)$

- The MLE is $\bar{\mathbf{Y}}$ (obvious and best, right?)
- Charles Stein discovered in the 1950's that the MLE is *inadmissible* if $p > 2$ 🤯
- The James-Stein estimator **shrinks** $\bar{\mathbf{Y}}$ toward some other point, any other point, chosen *a priori*, e.g. 0

$$\mathrm{MSE}(\hat{\mu}_{\mathrm{JS}}) < \mathrm{MSE}(\bar{\mathbf{Y}}) \text{ for all } \mu, \text{ if } p > 2$$

$$\hat{\mu}_{\mathrm{JS}} = \left(1 - \frac{(p-2)\sigma^2/n}{\|\bar{\mathbf{Y}}\|^2}\right) \bar{\mathbf{Y}}$$

Shrinkage: less variance, more bias



Solid points are improved by shrinking, hollow red points do worse

If $\bar{Y}$ is between μ and 0 then shrinking does worse

In higher dimensions, a greater portion of space is *not* between μ and 0

e.g. 2^p orthants in p-dimensional space, and only 1 contains $\mu - 0$

(*Not meant to be a proof*)

Historical significance

Statisticians (particularly frequentists) emphasized unbiasedness

But after Stein's example, we must admit bias is not always bad

Opens the doors to many interesting methods

Most (almost all?) ML methods use bias this way

(Even if some famous CS profs say otherwise on twitter 😒)

Regularized (i.e. penalized) regression

Motivation: If the JS estimator can do better than the MLE at estimating a sample mean, does a similar thing happen when estimating regression coefficients?

For some penalty function $\mathcal{P}_\lambda$, which depends on a tuning parameter λ, the estimator

$$\hat{\beta}_\lambda = \arg\min_\beta \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \mathcal{P}_\lambda(\beta)$$

is "regularized" or shrunk (shranken?) toward values that decrease the penalty. Often $\mathcal{P}_\lambda = \lambda\|\cdot\|$ for some norm

Many ML methods are optimizing "loss + penalty"

Ridge (i.e. L2 penalized) regression

- Originally motivated by problems where $\mathbf{X}^T\mathbf{X}$ is uninvertible (or badly conditioned, i.e. almost uninvertible)
- If $p > n$ then this always happens
- Least squares estimator is undefined or numerically unstable

For some constant $\lambda > 0$,

$$\text{minimize } \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|^2$$

Shrinks coefficients $\hat{\beta}$ toward 0

Larger coefficients are penalized more (squared penalty)

One common ML story: more flexible/complex models increase prediction accuracy by decreasing bias, but...

Bias *can* be good, actually

Especially in higher dimensions

Lasso regression

Comparison to ridge regression

Sparse models are more interpretable

Optimality and degrees of freedom for lasso

Inference

ML/optimization finds the "best" model

But is the best model actually good?

Interpretable

high-dimensional regression

with the lasso

Lasso vs ridge

- Generate some fake data from a linear model
- Introduce lasso by comparison to ridge

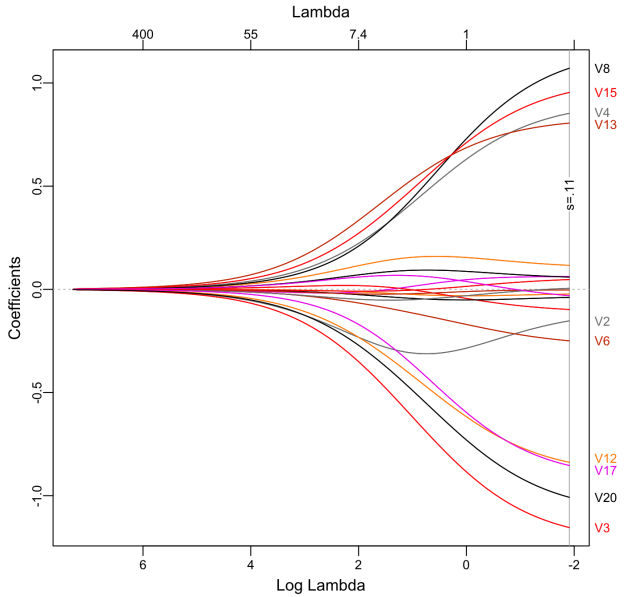
```
library(glmnet)
library(plotmo) # for plot_glmnet
n <- 100
p <- 20
X <- matrix(rnorm(n*p), nrow = n)
beta = sample(c(-1,0,0,0,1), p, replace = TRUE)
Y <- X %*% beta + rnorm(n)
lasso_fit <- glmnet(X, Y)
ridge_fit <- glmnet(X, Y, alpha = 0)
which(beta != 0)
```

```
## [1]  3  4  8 12 13 15 17 20
```

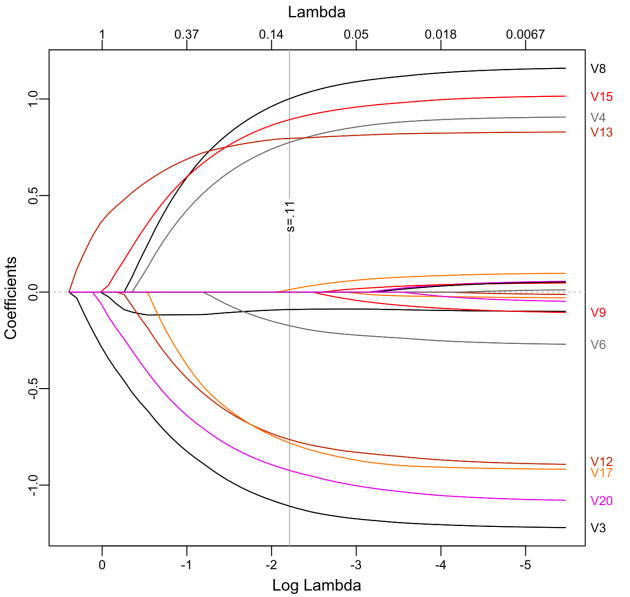
Only 8 of the 20 variables have nonzero coefficients

Lasso vs ridge: solution paths of $\hat{\beta}$

```
plot_glmnet(ridge_fit,
  s = cv_ridge$lambda.1se)
```




```
plot_glmnet(lasso_fit,
  s = cv_lasso$lambda.1se)
```



Lasso vs ridge: L1 vs L2 norm penalties

A simple `diff` to remember lasso/ridge is via the penalty/constraint (1-norm instead of 2-norm). Lasso is

$$\text{minimize } \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 \text{ s. t. } \|\beta\|_1 \leq t$$

where

$$\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$$

Lagrangian form

$$\text{minimize } \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_1$$

Lasso vs ridge: sparsity of solutions

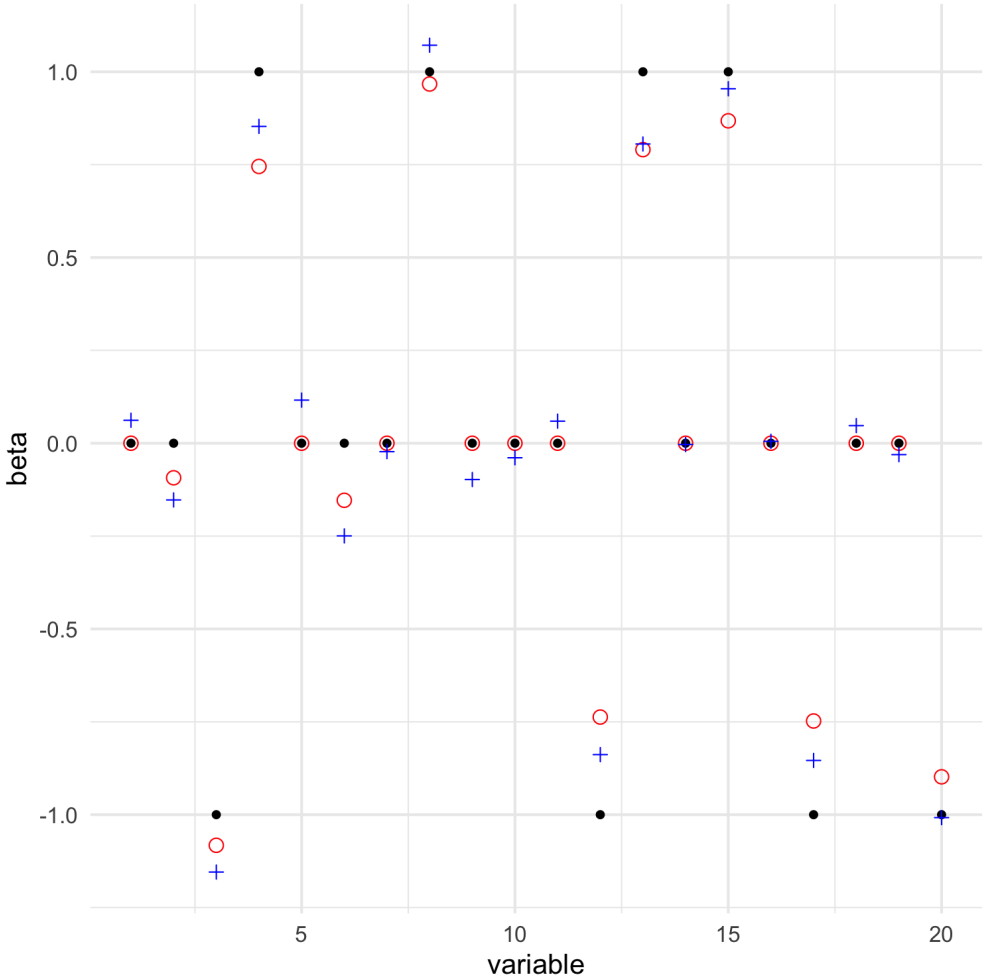
- For both ridge and lasso
- Choose λ with cross-validation
- Fit model on full data at the chosen $\hat{\lambda}$
- Look at the estimate $\hat{\beta}$ values...

```
cv_lasso <- cv.glmnet(X, Y)
coef_lasso <- coef(lasso_fit, s = cv_lasso$lambda.1se)
cv_ridge <- cv.glmnet(X, Y, alpha = 1)
coef_ridge <- coef(ridge_fit, s = cv_ridge$lambda.1se)
```

Note: `lambda.1se` larger `lambda.min` $\rightarrow$ heavier penalty

Lasso vs ridge: sparsity of solutions

```
##    variable true_beta beta_hat_lasso beta_hat_ridge
## 1         1         0          0.000         0.0618
## 2         2         0         -0.093        -0.1527
## 3         3        -1         -1.082        -1.1544
## 4         4         1          0.745         0.8530
## 5         5         0          0.000         0.1161
## 6         6         0         -0.154        -0.2495
## 7         7         0          0.000        -0.0226
## 8         8         1          0.967         1.0715
## 9         9         0          0.000        -0.0977
## 10       10         0          0.000        -0.0391
## 11       11         0          0.000         0.0594
## 12       12        -1         -0.737        -0.8383
## 13       13         1          0.790         0.8056
## 14       14         0          0.000        -0.0035
## 15       15         1          0.868         0.9543
## 16       16         0          0.000         0.0052
## 17       17        -1         -0.748        -0.8540
## 18       18         0          0.000         0.0474
## 19       19         0          0.000        -0.0308
## 20       20        -1         -0.898        -1.0079
```



True values are solid black dots, lasso estimates are hollow red circles, ridge estimates are blue crosses

High dimensional example

Lasso: cool or extremely cool?

- High-dimensional: $p > n$ means we can't fit OLS

But all is not lost! Penalized regression to the rescue

- True model is sparse

Only 21 of 200 variables have nonzero coefficients

- Ridge estimates are dense

All coefficients nonzero 😰

- Lasso estimates are sparse

Nonzero estimates largely coincide with true model 😎

Lessons about sparsity

Solving otherwise impossible problems

Curse of dimensionality / NP-hard optimization (best subsets) / unidentifiable statistical estimation / overfitting vs generalization

Need special mathematical structure like sparsity to make things tractable

Sparsity helps with interpretation

Easier to interpret 38 nonzero coefficients than all 200

Sparse models are more interpretable

Usual linear model interpretation of coefficients

If the conditional expectation function (CEF) is linear

$$f(\mathbf{x}) = \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}] = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

Then

$$\hat{\beta}_j \approx \frac{\partial}{\partial x_j} \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}]$$

"Change in CEF *holding other variables constant*"

Small set of **other variables** $\rightarrow$ easy (human) understanding

Why are lasso estimates sparse?

- **Lagrangian form**

$$\hat{\beta}_{\text{lasso}}(\lambda) = \arg\min_{\beta} \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\beta\|_1$$

- **Constrained form**

$$\hat{\beta}_{\text{lasso}}(t) = \arg\min_{\beta} \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 \text{ s. t. } \|\beta\|_1 \leq t$$

Level sets $\{\beta : \|\mathbf{y} - \mathbf{X}\beta\|_2^2 = \text{const.}\}$ are **ellipsoids**

Level sets: $\{\beta : \|\beta\|_1 = t\}$ are **diamond thingies**

(i.e. "cross polytope" or L^1 ball)

KKT optimality conditions

Constrained optimization in multivariate calculus:

- Switch to lagrangian form
- Check stationary points (vanishing gradient)
- Check boundary/singularity points
- Verify feasibility (constraints satisfied)

(Exam note: problems may use multivariate calculus like this)

The Karush-Kuhn-Tucker (KKT) conditions generalize these, useful for analysis of constrained optimization problems (i.e. almost all ML methods) -- more advanced, **not examinable**

Recall: optimizer occurs at point of intersection between level sets of constraints and level sets of objective

Why are lasso solutions sparse? ISLR Figure 6.7

Why are lasso solutions sparse?

The L^1 ball in $\mathbb{R}^p$ $\{x : \|x\|_1 = \text{const}\}$ contains

- $2p$ points that are 1-sparse $x_j = \pm 1$, $x_{-j} = 0$
- $\binom{p}{k} 2^k$ points k-sparse with elements $\pm k^{-1}$
- Higher dimensional edges/faces spanning (some) sets of these points, etc
- Each of these is sparse, i.e. many coordinates are 0

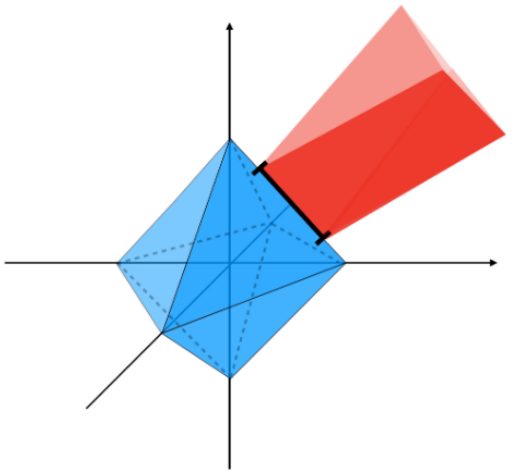
The ellipsoid $\|\mathbf{y} - \mathbf{X}\beta\|_2^2 = \text{const}$ *probably* intersects one of these *sharp parts of the diamond thingies*

Solution to optimization problem is an intersection point

Why are lasso solutions sparse?

At the point of intersection of the ellipse and the L^1 ball, the normal vector of the ellipse has to be in the *normal cone* of the L^1 ball (at the same point)

Another intuition: consider projecting to the nearest point on the surface of the L^1 ball (see figure)



(a) All the points β in the red area are projected to θ in the line segment $\{(x, 0, r - x) : 0.2 < x < 0.8\}$, which contains exact zero $\theta_2 = 0$.

From Figure 1 in Xu and Duan (2020)

Optimality: stationary points

- Of the **OLS** objective (uncorrelated residuals)

$$\frac{1}{n}\mathbf{X}^T(\mathbf{X}\hat{\beta} - \mathbf{y}) = 0$$

- For **ridge** (larger $|\hat{\beta}_j|$ have larger resid. covariance)

$$\frac{1}{n}\mathbf{X}^T(\mathbf{X}\hat{\beta} - \mathbf{y}) = -2\lambda\hat{\beta}$$

- For **lasso** (resid. |covar| = λ if $\hat{\beta}_j \neq 0$ and $\leq \lambda$ otherwise)

$$\frac{1}{n}\mathbf{X}^T(\mathbf{X}\hat{\beta} - \mathbf{y}) = -\lambda\,\mathrm{sign}(\hat{\beta})$$

Lasso treats predictors more "democratically"

Optimism / generalization gap

Recall: for some $\mathrm{df}(\hat{f}) > 0$ (depends on problem/fun. class)

$$\Delta = \mathbb{E}_{Y|\mathbf{x}_1,\ldots,\mathbf{x}_n}[R(\hat{f}) - \hat{R}(\hat{f})] = \frac{2\sigma_\varepsilon^2}{n}\mathrm{df}(\hat{f}) > 0$$

Fairly general case

For many ML tasks and fitting procedures

$$\mathrm{df}(\hat{f}) = \frac{1}{\sigma_\varepsilon^2}\mathrm{Tr}[\mathrm{Cov}(\hat{f}(\mathbf{X}), \mathbf{y})] = \frac{1}{\sigma_\varepsilon^2}\sum_{i=1}^{n}\mathrm{Cov}(\hat{f}(\mathbf{x}_i), y_i)$$

Degrees of freedom: classic regression case

If $\hat{f}$ is linear with deterministic set of p predictors (or p basis function transformations of original predictors) then

$$\mathrm{df}(\hat{f}) = p, \text{ so } \Delta = 2\sigma_\varepsilon^2 \frac{p}{n}$$

But if we do model/variable selection and use the data to choose $\hat{p}$ predictors then *usually*

$$\mathrm{df}(\hat{f}) > \hat{p}$$

And the more optimization / larger search this involves, it becomes more likely that

$$\mathrm{df}(\hat{f}) \gg \hat{p}$$

Degrees of freedom: lasso case

The "0-norm" (not really a norm) counts sparsity

$$\|\beta\|_0 = \sum_{j=1}^{p} \mathbf{1}_{\beta_j \neq 0} = |\{j : \beta_j \neq 0\}|$$

e.g. for OLS with deterministic choice of variables

$$\mathrm{df}(\hat{\beta}_{\mathrm{OLS}}) = \|\hat{\beta}_{\mathrm{OLS}}\|_0$$

Surprisingly, under fairly weak conditions on $\mathbf{X}$ (columns in "general position"), for the lasso solution $\hat{\beta}(\lambda)$

$$\mathbb{E}[\|\hat{\beta}_{\mathrm{lasso}}(\lambda)\|_0] = \mathrm{df}(\hat{\beta}_{\mathrm{lasso}}(\lambda))$$

Solution sparsity is unbiased estimate of df - like OLS case

Choosing λ for lasso

- Could use degrees of freedom combined with AIC, BIC, etc
- Most commonly people use **cross-validation**

Important note

$\hat{\beta}_{\text{lasso}}(\lambda)$ at fixed λ vs $\hat{\beta}_{\text{lasso}}(\hat{\lambda})$ at data-chosen $\hat{\lambda}$

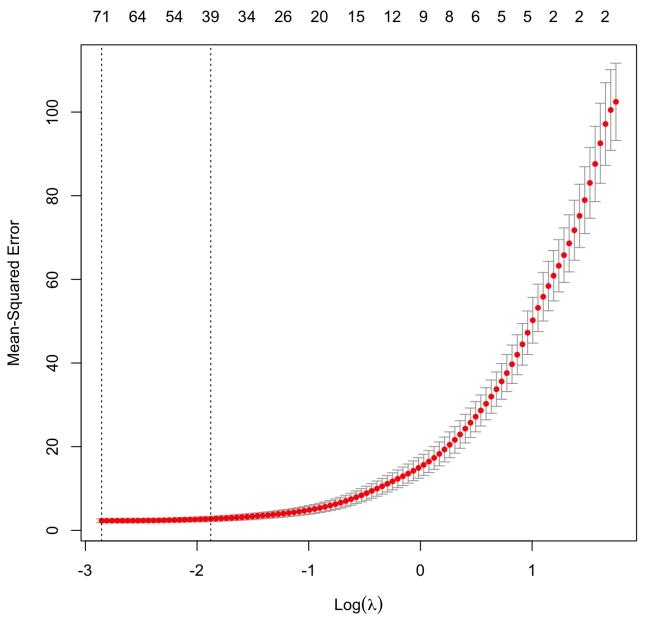
Different in general!

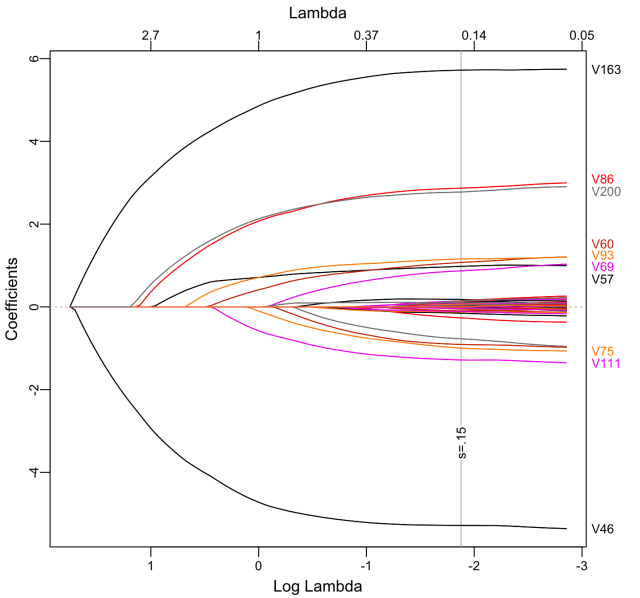
e.g. Theoretical results about first may not apply to the second

plot(cv.glmnet) and plot(glmnet)

```
cv_fit <- cv.glmnet(X, Y)
plot(cv_fit)
```

```
plot_glmnet(lasso_fit,
    s = cv_fit$lambda.1se)
```





Inference

for high-dimensional regression

We have used regularization to avoid overfitting

But this results in bias, e.g. $||\hat{\beta}||$ smaller than true $||\beta||$

Inference must correct for this somehow

Approaches to inference

- Debiased inference
- Selective inference
- Post-selection inference
- Stability selection

R packages for some of these

Topic for future study? 😁

One example

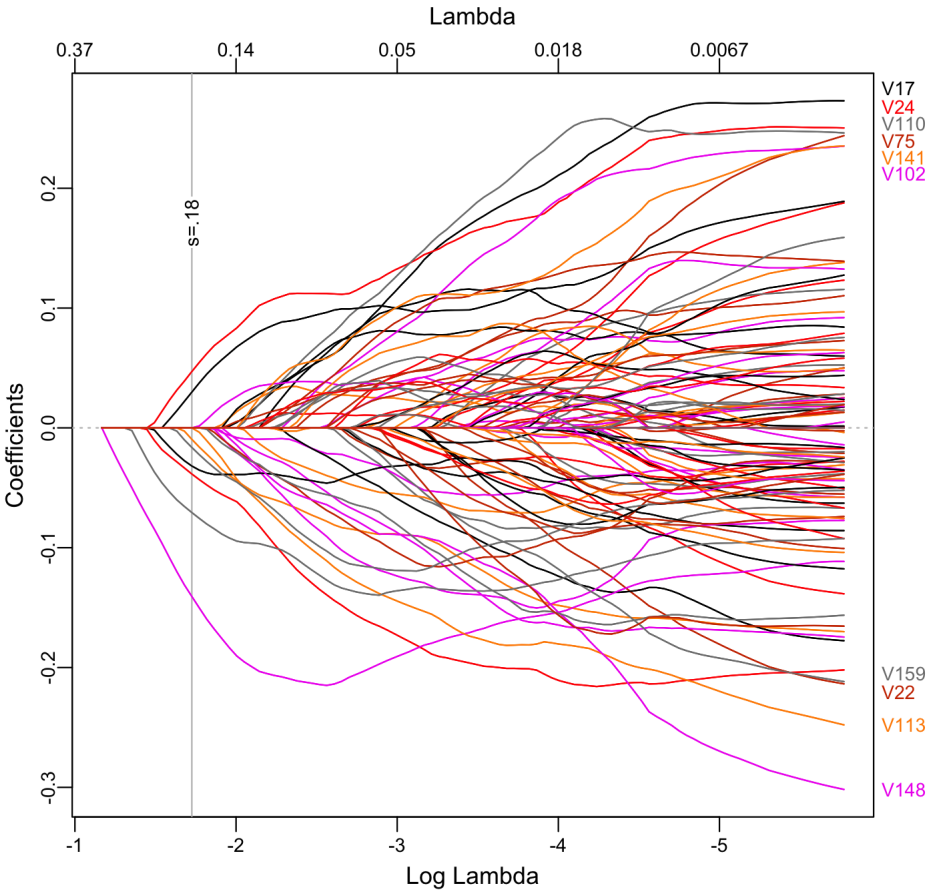
```
set.seed(1)
n <- 100; p <- 200
X <- matrix(rnorm(n*p), nrow = n)
beta = sample(c(-1, rep(0, 20), 1), p, replace = TRUE)
Y <- X %*% beta + rnorm(n)
```

Cross-validation plot (next slide)

```
beta_hat <- coef(lasso_fit, s = cv_lasso$lambda.min)[-1]
vars <- which(beta_hat != 0)
vars
```

```
## [1]  24  34  43  90 111 125 156 168
```

Idea: since $\hat{\beta}$ is biased by penalization, how about refitting OLS (unpenalized) using only these variables?


Lambda
0.37
0.14
0.05
0.018
0.0067
s=.18
Coefficients
0.2
0.1
0.0
-0.1
-0.2
-0.3
-1
-2
-3
-4
-5
Log Lambda
V17
V24
V110
V75
V141
V102
V159
V22
V113
V148

```r
summary(lm(Y ~ X[,vars]-1))
```

```
##
## Call:
## lm(formula = Y ~ X[, vars] - 1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.598 -0.238  0.212  0.772  2.609
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## X[, vars]1   0.2373     0.0871    2.72  0.00771 **
## X[, vars]2  -0.1335     0.0806   -1.66  0.10079
## X[, vars]3  -0.1677     0.0768   -2.18  0.03160 *
## X[, vars]4  -0.3063     0.0817   -3.75  0.00031 ***
## X[, vars]5  -0.1351     0.0797   -1.70  0.09330 .
## X[, vars]6   0.1881     0.0808    2.33  0.02209 *
## X[, vars]7  -0.1481     0.0816   -1.81  0.07279 .
## X[, vars]8  -0.1530     0.0807   -1.90  0.06121 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.81 on 92 degrees of freedom
## Multiple R-squared:  0.375,    Adjusted R-squared:  0.32
## F-statistic: 6.89 on 8 and 92 DF,  p-value: 4.5e-07
```

Looks good, time to publish!

- Sparse, interpretable model
- Some significant predictors
- Decent R^2 value showing predictive accuracy

Pretty good... hey wait, what was this line in the code...

```
Y <- rnorm(n)
lasso_fit <- glmnet(X, Y)
cv_lasso <- cv.glmnet(X, Y)
```

It was a trick 😈 🙈 model was *actually* fit on pure noise

Classic inference methods don't work for *selected* models 😱

Idea: compute inferences (`summary`) on new validation data

Lessons about high-dimensional regression

- Can fit to noise, even with cross-validation
- Theoretical results (advanced, not examinable)

Lasso "performs well" (prediction error, estimation error, sparse support recovery) under various sets of sufficient conditions, derived/proven using KKT conditions and probability bounds (see SLS Chapter 11)

Roughly:

- $\mathbf{X}$ has to be well-conditioned in some sense (null and non-null predictors not too correlated)
- True β has to be sparse enough
- Solution still generally includes some false positives

Sparsity

Useful simplifying assumption, especially in higher dimensions

Lasso

Penalized/regularized regression with the L^1 norm

Interpretation

Sparse models are easier to interpret

Model selection bias

If we use the same data to (1) select a model and (2) compute tests / intervals / model diagnostics / etc, then those inferences will be biased (similar to overfitting)