Lecture 2

DJM

2 October 2018

Statistics vs. ML

- Lots of overlap, both try to "extract information from data"

Venn diagram

Probability

1. X_n converges *in probability* to X, $X_n \xrightarrow{P} X$, if for every $\epsilon > 0$, $\mathbb{P}\left(|X_n - X| < \epsilon\right) \to 1$.
2. X_n converges *in distribution* to X, $X_n \rightsquigarrow X$, if $F_n(t) \to F(t)$ at all continuity points t.
3. (Weak law) If $X_1, X_2, \ldots$ are iid random variables with common mean m,then $\overline{X}_n \xrightarrow{P} m$.
4. (CLT) If $X_1, X_2, \ldots$ are iid random variables with common mean m and variance $s^2 < \infty$, then $\sqrt{n}(\overline{X}_n - m)/s \rightsquigarrow \mathrm{N}(0,1)$.

Big-Oh and Little-Oh

Deterministic:

1. $a_n = o(1)$ means $a_n \to 0$ as $n \to \infty$
2. $a_n = o(b_n)$ means $\frac{a_n}{b_n} = o(1)$.
   Examples:
   - If $a_n = \frac{1}{n}$, then $a_n = o(1)$
   - If $b_n = \frac{1}{\sqrt{n}}$, then $a_n = o(b_n)$
3. $a_n = O(1)$ means a_n is eventually bounded for all n large enough, $|a_n| < c$ for some $c > 0$. Note that $a_n = o(1)$ implies $a_n = O(1)$
4. $a_n = O(b_n)$ means $\frac{a_n}{b_n} = O(1)$. Likewise, $a_n = o(b_n)$ implies $a_n = O(b_n)$. Examples:
   - If $a_n = \frac{n}{2}$, then $a_n = O(n)$

Stochastic analogues:

1. $Y_n = o_p(1)$ if for all $\epsilon > 0$, then $P(|Y_n| > \epsilon) \to 0$
2. We say $Y_n = o_p(a_n)$ if $\frac{Y_n}{a_n} = o_p(1)$
3. $Y_n = O_p(1)$ if for all $\epsilon > 0$, there exists a $c > 0$ such that $P(|Y_n| > c) < \epsilon$
4. We say $Y_n = O_p(a_n)$ if $\frac{Y_n}{a_n} = O_p(1)$
   Examples:
   - $\overline{X}_n - \mu = o_p(1)$ and $S_n - \sigma^2 = o_p(1)$. By the the Law of Large Numbers.
   - $\sqrt{n}(\overline{X}_n - \mu) = O_p(1)$ and $\overline{X}_n - \mu = O_p(\frac{1}{\sqrt{n}})$. By the Central Limit Theorem.

Statistical models

A statistical model $\mathcal{P}$ is a collection of probability distributions or densities. A parametric model has the form

$$\mathcal{P} = \{p(x;\theta) : \theta \in \Theta\}$$

where $\Theta \subset \mathbb{R}^d$ in the parametric case.

Examples of nonparametric statistical models:

- $\mathcal{P} = \{$ all continuous CDF's $\}$
- $\mathcal{P} = \{f : \int (f''(x))^2 dx < \infty\}$

Evaluating estimators

An *estimator* is a function of data that does not depend on θ.

Suppose $X \sim N(\mu, 1)$.
-μ is not an estimator.
-Things that are estimators: X, any functions of X, 3, $\sqrt{X}$, etc.

1. Bias and Variance
2. Mean Squared Error
3. Minimaxity and Decision Theory
4. Large Sample Evaluations

MSE

Mean Squared Error (MSE). Suppose $\theta, \widehat{\theta}$, define

$$\mathbb{E}\left[\left(\theta - \widehat{\theta}\right)^2\right] = \int \cdots \int \left[\left(\widehat{\theta}(x_1, \ldots, x_n) - \theta\right) f(x_1;\theta)^2 \cdots f(x_n;\theta)\right] dx_1 \cdots dx_n.$$

Bias and Variance The bias is

$$B = \mathbb{E}\left[\widehat{\theta}\right] - \theta,$$

and variance is

$$V = \mathbb{V}\left[\widehat{\theta}\right].$$

Bias-Variance Decomposition

$$MSE = B^2 + V$$

$$\begin{aligned}
MSE &= \mathbb{E}\left[(\widehat{\theta} - \theta)^2\right] \\
&= \mathbb{E}\left[\left(\widehat{\theta} - \mathbb{E}\left[\widehat{\theta}\right] + \mathbb{E}\left[\widehat{\theta}\right] - \theta\right)^2\right] \\
&= \mathbb{E}\left[\widehat{\theta} - \mathbb{E}\left[\widehat{\theta}\right]\right] + \left(\mathbb{E}\left[\widehat{\theta}\right] - \theta\right)^2 + \underbrace{2\mathbb{E}\left[\widehat{\theta} - \mathbb{E}\left[\widehat{\theta}\right]\right]}_{=0} \left(\mathbb{E}\left[\widehat{\theta}\right] - \theta\right) \\
&= V + B^2
\end{aligned}$$

An estimator is unbiased if $B = 0$. Then $MSE = Variance$.
Let $x_1, \ldots, x_n \overset{iid}{\sim} N(\mu, \sigma^2)$.

$$\begin{aligned}
&\mathbb{E}[\overline{x}] = \mu, & &\mathbb{E}\left[s^2\right] = \sigma^2 \\
&\mathbb{E}\left[(\overline{x} - \mu)^2\right] = \frac{\sigma^2}{n} = O\left(\frac{1}{n}\right) & &\mathbb{E}\left[(s^2 - \sigma^2)^2\right] = \frac{2\sigma^4}{n-1} = O\left(\frac{1}{n}\right).
\end{aligned}$$

Minimaxity

Let $\mathcal{P}$ be a set of distributions. Let θ be a parameter and let $L(\theta, \theta')$ be a loss function.

The **minimax risk** is

$$R_n(\mathcal{P}) = \inf_{\widehat{\theta}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[L(\theta, \widehat{\theta})]$$

If $\sup_{P \in \mathcal{P}} \mathbb{E}_P[L(\theta, \widehat{\theta})] = R_n$ then $\widehat{\theta}$ is a minimax estimator.

- $X_1, X_2, \ldots, X_n \overset{iid}{\sim} N(\theta, 1)$ Then $\overline{X}$ is minimax for many loss functions. It's risk is $R_n = \frac{1}{n}$ which is the "Parametric Rate".
- $X_1, X_2, \ldots, X_n \sim f$, where $f \in \mathcal{F}$ is some density. Let $\mathcal{F}$ be the class of smooth densities: $\mathcal{F} = \left\{f; \int (f'')^2 < c_0\right\}$ Then $R_n \leq Cn^{-4/5}$ for $L(\widehat{f}, f) = \int (f - \widehat{f})^2 dx$.

Linear model, introduction

The Setup

Suppose we have data

$$\mathcal{D} = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\},$$

where

- $X_i \in \mathbb{R}^p$ are the *features*

  (or explanatory variables or predictors or covariates. NOT INDEPENDENT VARIABLES!)
- $Y_i \in \mathbb{R}$ are the response variables.

  (NOT DEPENDENT VARIABLE!)

Our goal for this class is to find a way to explain (at least approximately) the relationship between X and Y.

Prediction risk for regression

Given the *training data* $\mathcal{D}$, we want to predict some independent *test data* $Z = (X, Y)$

This means forming a $\widehat{f}$, which is a function of both the range of X and the training data $\mathcal{D}$, which provides predictions $\widehat{Y} = \widehat{f}(X)$.

The quality of this prediction is measured via the prediction risk

$$R(\widehat{f}) = \mathbb{E}\left[(Y - \widehat{f}(X))^2\right].$$

We know that the *regression function*, $f_*(X) = \mathbb{E}[Y|X]$, is the best possible predictor.

Note that f_* is *unknown*.

A linear model: Multiple regression

If we assume: $f_*(X) = X^\top \beta = \sum_{j=1}^p x_j \beta_j$

$$\Rightarrow Y_i = X_i^\top \beta + \epsilon_i$$

.

There's generally no reason to make this assumption.

We'll examine a few cases:

1. f_* is linear, low dimensions.

2. f_* is ~~not~~ linear, but we use a linear model anyway

3. f_* is linear, high dimensions.

4. f_* isn't linear, high dimensions.

Important: Calling f_* "linear", means that $f_*(x) = x'\beta$

Kernelization

We'll come back to this more rigorously later in the course.

Suppose $x \in [0, 2\pi]$ and $f_*(x) = \sin(x)$.

f_* isn't linear in x.

But

$$\sin(x) = \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots = \sum_{j=1}^{\infty} \beta_j x^j$$

by Taylor's theorem (of course this works for any function).

If I have some map $\Phi(x) \to [\phi_1(x), \ldots, \phi_K(x)]$, then I can estimate a linear model using the new features $\phi_1, \ldots, \phi_K$.

I can even take $K = \infty$.

This is still a "linear" model in the sense we're using today, though it isn't "linear" in the original x.

Low-dimension, high-assumptions

Let $x_i \in \mathbb{R}^p$, $p < n$.

If f_* is linear, and $\epsilon_i \sim N(0, \sigma^2)$ (independent)

Then all the good things happen:

1. $R(\widehat{f}) = \sigma^2 \left[1 + \frac{p}{n}\right]$
2. $\left\|\beta_* - \widehat{\beta}\right\|_2^2 = O_p(p/n)$
3. Coefficient estimates are normally distributed.
4. etc.

Low-dimension, low-assumptions

Let $\beta_* =$ be the best linear predictor for the feature X.

Note that this is well defined: $\beta_* = \mathbb{E}[XX']^{-1}\mathbb{E}[XY] =: \Sigma_{XX}^{-1}\sigma_{XY}$.

Call $R(\beta_*) = \mathbb{E}\left[(Y - X\beta_*)^2\right]$.

We call $R(\beta) - R(\beta_*)$ the *excess risk* of using β relative to the best linear predictor β_*.

Note that

$$R(\beta) - R(\beta_*) = (\beta - \beta_*)'\Sigma(\beta - \beta_*).$$

Then, (simplified result), See Theorem 11.3 of Györfi et al. (2002),

$$R(\beta) \leq C\left[R(\beta_*) + \frac{p\log n}{n}\right]$$

Note that if the model were linear, $R(\beta_*) = \sigma^2$

We also have a CLT (see Berk et al. 2014):

$$\sqrt{n}(\widehat{\beta} - \beta_*) \rightsquigarrow N(0, \Gamma)$$
$$\Gamma = \Sigma^{-1}\mathbb{E}\left[(Y - X\beta)^2 XX'\right]\Sigma^{-1}$$

Bias and variance

Prediction risk for regression

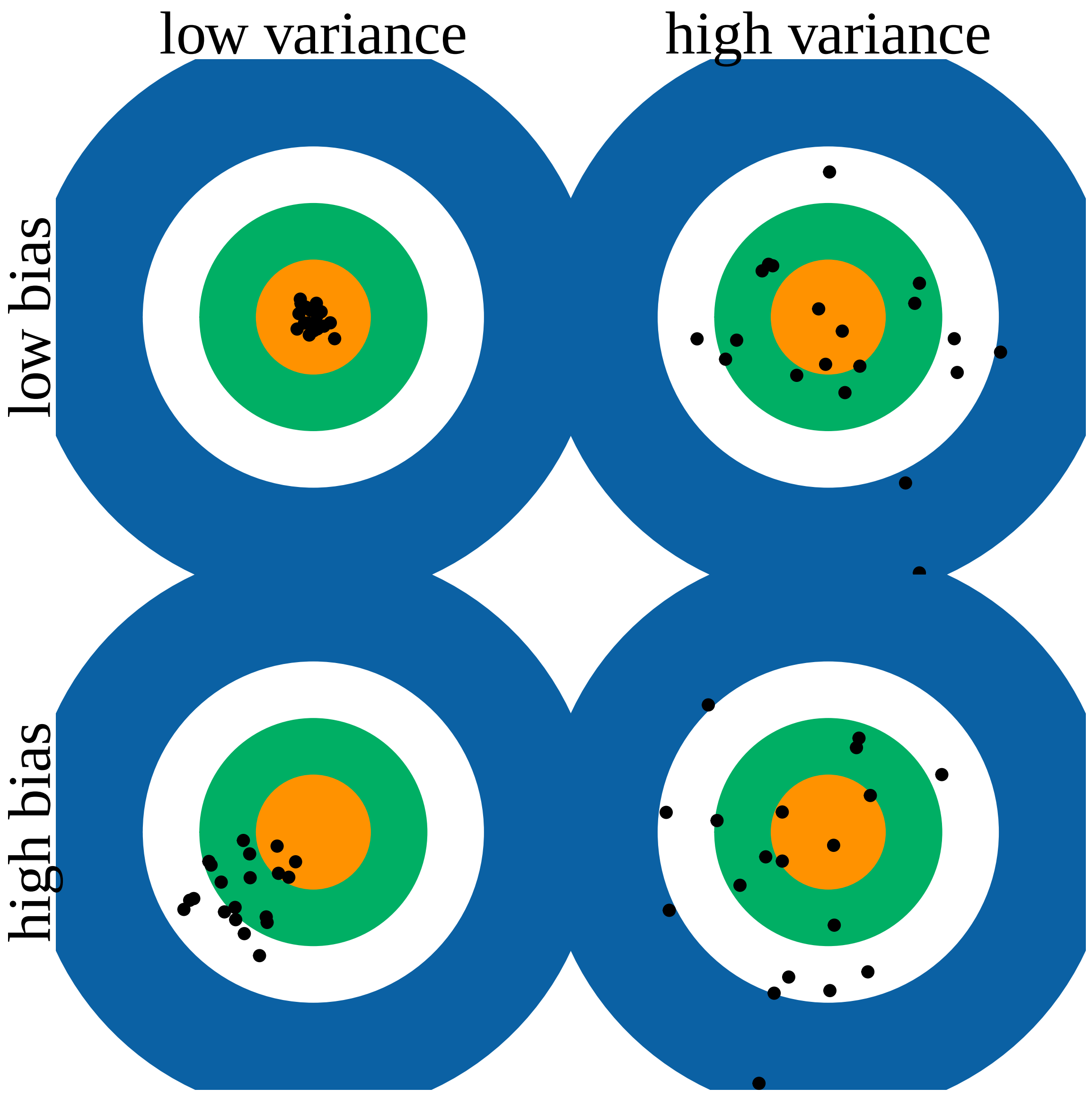
Note that $R(\widehat{f})$ can be written as

$$R(\widehat{f}) = \int \text{bias}^2(x)d\mathbb{P}_X + \int \text{var}(x)d\mathbb{P}_X + \sigma^2$$

where

$$\text{bias}(x) = \mathbb{E}\left[\widehat{f}(x)\right] - f_*(x)$$
$$\text{var}(x) = \mathbb{V}\left[\widehat{f}(x)\right]$$
$$\sigma^2 = \mathbb{E}\left[(Y - f_*(X))^2\right]$$

This decomposition applies to much more general loss functions (James 2003)



Bias-variance tradeoff

This can be heuristically thought of as

$$\text{Prediction risk} = \text{Bias}^2 + \text{Variance}.$$

There is a natural conservation between these quantities

Low bias $\rightarrow$ complex model $\rightarrow$ many parameters $\rightarrow$ high variance

The opposite also holds

(Think: $\widehat{f} \equiv 0$.)

We'd like to 'balance' these quantities to get the best possible predictions

Classical regime

The Gauss-Markov theorem assures us that OLS is the best linear *unbiased* estimator of β

Also, it is the maximum likelihood estimator under a homoskedastic, independent Gaussian model, has the other nice properties listed above.

Does that necessarily mean it is any good?

Write $X = UDV'$ for the SVD of the design matrix X.

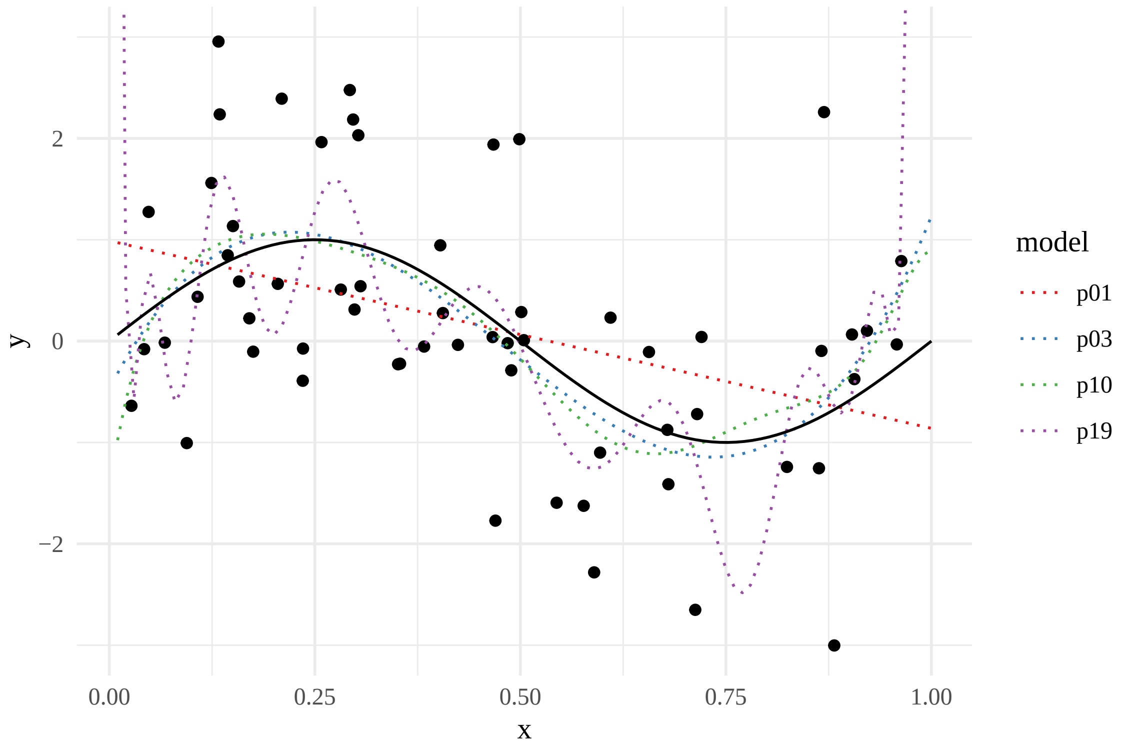
Then $\mathbb{V}\left[\widehat{\beta}_{LS}\right] \propto (X^\top X)^{-1} = VD^{-1}\underbrace{U^\top U}_{=I}D^{-1}V^\top = VD^{-2}V^\top$

Thus,

$$\mathbb{E}||\widehat{\beta}_{LS} - \beta||_2^2 = \text{trace}(\mathbb{V}\widehat{\beta}) \propto \sum_{j=1}^{p} \frac{1}{d_j^2}$$

Important: Even in the classical regime, we can do arbitrarily badly if $d_p \approx 0$!

An example



Using a Taylor's series, for all X

$$\sin(X) = \sum_{q=0}^{\infty} \frac{(-1)^q X^{2q+1}}{(2q+1)!} = \Phi(X)^\top \beta$$

Additional polynomial terms will **reduce** the bias but the variance can get nasty.

Returning to polynomial example: Variance

The least squares solution is given by solving $\min \|X\beta - Y\|_2^2$

$$X = \begin{bmatrix} 1 & X_1 & \ldots & X_1^{p-1} \\ & \vdots & & \\ 1 & X_n & \ldots & X_n^{p-1} \end{bmatrix},$$

is the associated Vandermonde matrix.

This matrix is well known for being numerically unstable

(Letting $\mathbb{X} = UDV^\top$, this means that $d_1/d_p \to \infty$)

Hence

$$\left\|(\mathbb{X}^\top\mathbb{X})^{-1}\right\|_2 = \frac{1}{d_p^2}$$

grows larger, where here $\|\cdot\|_2$ is the spectral (operator) norm.

In the example, I used the *orthogonal* polynomials, so $d_j = 1$.

So, $\mathbb{V}\left[\widehat{\beta}\right] = \sigma^2 p$.

Conclusion: Fitting the full least squares model, even in low dimensions, can lead to poor prediction/estimation performance.

Big data regime

Big data: The computational complexity scales extremely quickly. This means that procedures that are feasible classically are not for large data sets

Example: Fit $\widehat{\beta}_{LS}$ with $\mathbb{X} \in \mathbb{R}^{n \times p}$. Next fit $\widehat{\beta}_{LS}$ with $\mathbb{X} \in \mathbb{R}^{3n \times 4p}$

The second case will take $\approx (3 * 4^2) = 48$ times longer to compute, as well as ≈ 12 times as much memory!

(Actually, for software such as `R` it might take 36 times as much memory, though there are data structures specifically engineered for this purpose that update objects 'in place')

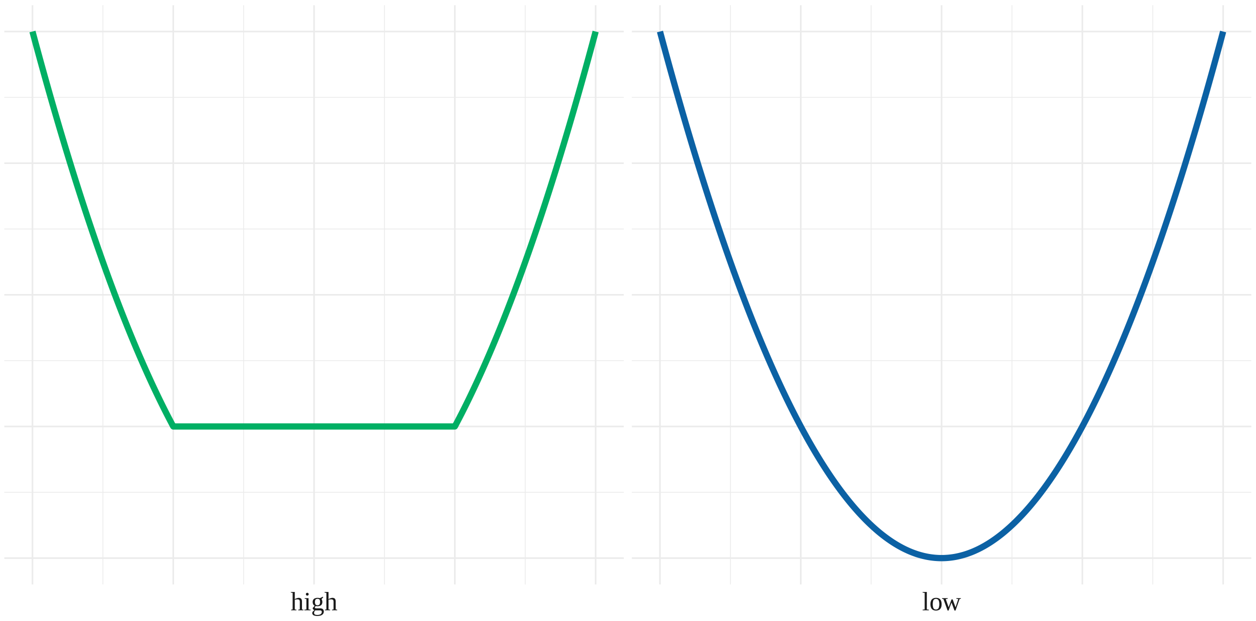
High dimensional regime

High dimensional: These problems tend to have many of the computational problems as "Big data", as well as a "rank problem"

Suppose $\mathbb{X} \in \mathbb{R}^{n \times p}$ and $p > n$

Then $\text{rank}(\mathbb{X}) = n$ and the equation $\mathbb{X}\widehat{\beta} = Y$:

- can be solved *exactly* (that is; the training error is 0)
- has an infinite number of solutions



High dimensional linear methods

Theoretical meanings

1. Low dimensional
    - finite sample $p < n$
    - asymptotics $p/n \to 0$
2. High dimensional
    - finite sample $n > p$
    - asymptotics $p/n \to \infty$
3. "Big data"
    - finite sample n or p or both are huge
    - no real asymptotic interpretation

Approaches for big data

1. Reduce the dimension. Try PCA on the features, cluster features, screen the features.
2. Use all the covariates, but shrink the coefficients.
3. Select some useful covariates, throw away the rest.

Shrinkage

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

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

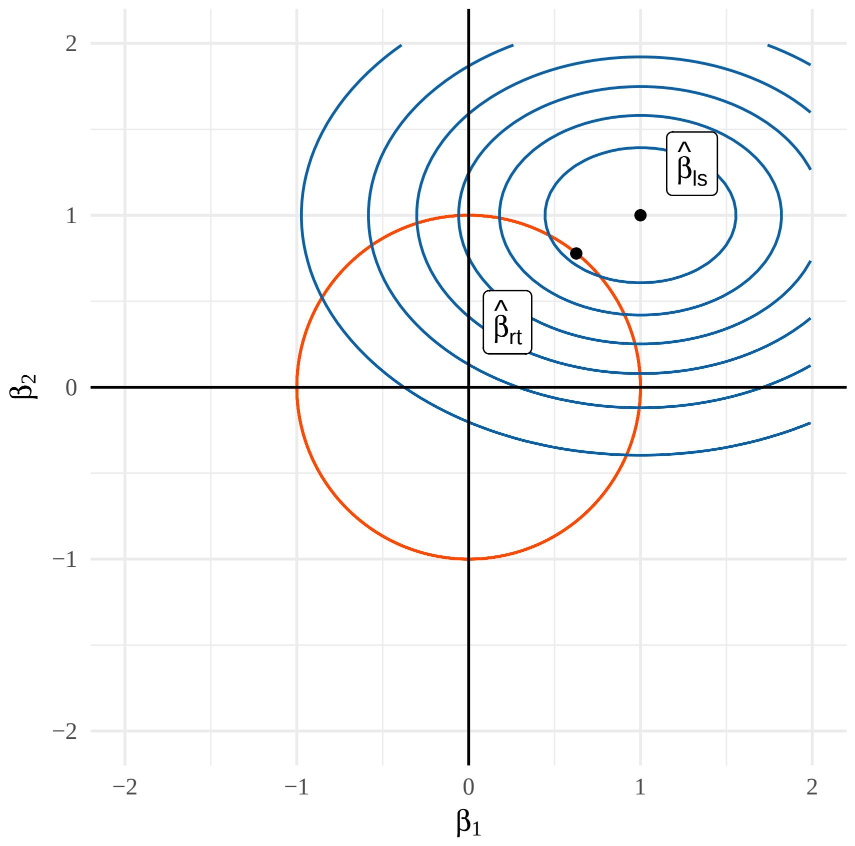
for some $t > 0$.

- 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

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

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

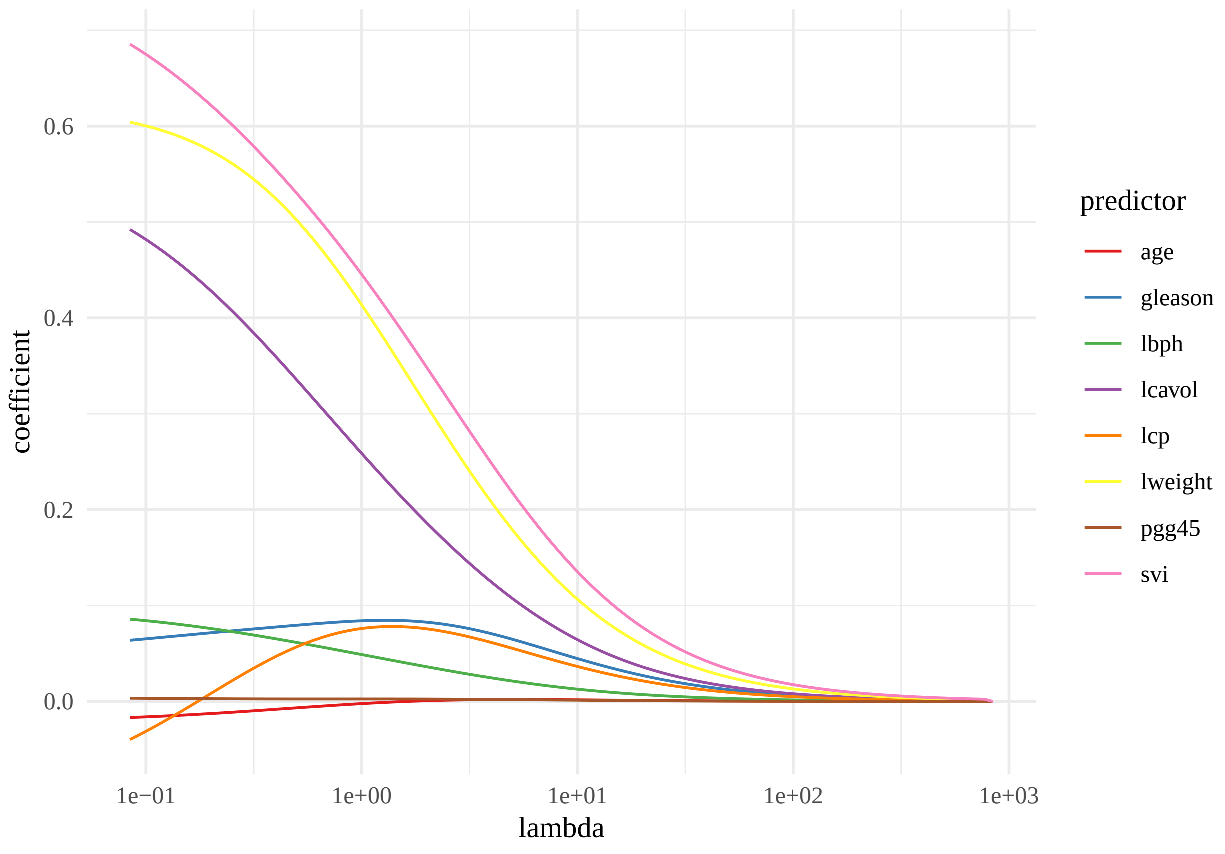
$$\widehat{\beta}_{r,t} = \widehat{\beta}_{r,\lambda}$$

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



Least squares is invariant to rescaling, regularized methods aren't

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

- `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. But you need to recalculate for each λ.
- 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.

Finally,

- $p > n$, $(\mathbb{X}'\mathbb{X} + \lambda I_p)^{-1}$ requires $O(p^3)$ computations and $O(p^2)$ storage
- But $X'(\mathbb{X}\mathbb{X}' + \lambda I_n)^{-1}$ requires $O(n^3)$ computations and $O(n^2)$ storage

Searle's matrix identity shows that these are equal.

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.

Ridge theory

Recalling that $\beta'_* x$ is the best linear approximation to $f_*(x)$

If $\|x\|_\infty < r$, (Hsu, Kakade, and Zhang 2014),

$$R(\widehat{\beta}_\lambda) - R(\beta_*) \leq \left(1 + O\left(\frac{1 + r^2/\lambda}{n}\right)\right) \frac{\lambda \|\beta_*\|_2^2}{2} + \frac{\sigma^2 \mathrm{tr}(\Sigma)}{2n\lambda}$$

Optimizing over λ, and setting $B = \|\beta_*\|$ gives

$$R(\widehat{\beta}) - R(\beta_*) \leq \sqrt{\frac{\sigma^2 r^2 B^2}{n} \left(1 + O(1/n)\right)} + O\left(\frac{r^2 B^2}{n}\right)$$

Lower bound

$$\inf_{\widehat{\beta}} \sup_{\beta_*} R(\widehat{\beta}) - R(\beta_*) \geq C\sqrt{\frac{\sigma^2 r^2 B^2}{n}}$$

We call this behavior *rate minimax*: essential meaning,

$$R(\widehat{\beta}) - R(\beta_*) = O\left(\inf_{\widehat{\beta}} \sup_{\beta_*} R(\widehat{\beta}) - R(\beta_*)\right)$$

In this setting, Ridge regression does as well as we could hope, up to constants.

Bayes interpretation

If

1. $Y = X'\beta + \epsilon$,
2. $\epsilon \sim N(0, \sigma^2)$
3. $\beta \sim N(0, \tau^2 I_p)$,

Then, the posterior mean (median, mode) is the ridge estimator with $\lambda = \sigma^2/\tau^2$.

In-class exercise

Subset selection methods

Lasso and family

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 $||\beta||_2^2 \le t$

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

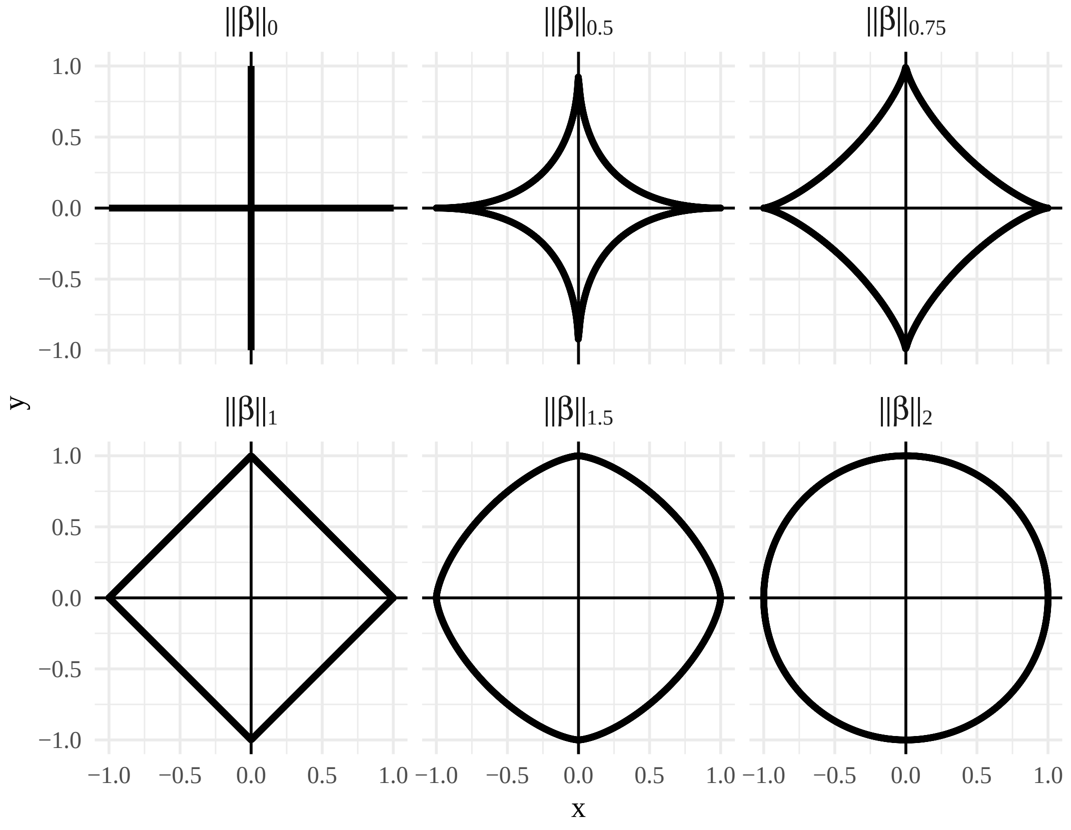
$||\beta||_0$ is the number of nonzero elements in β

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

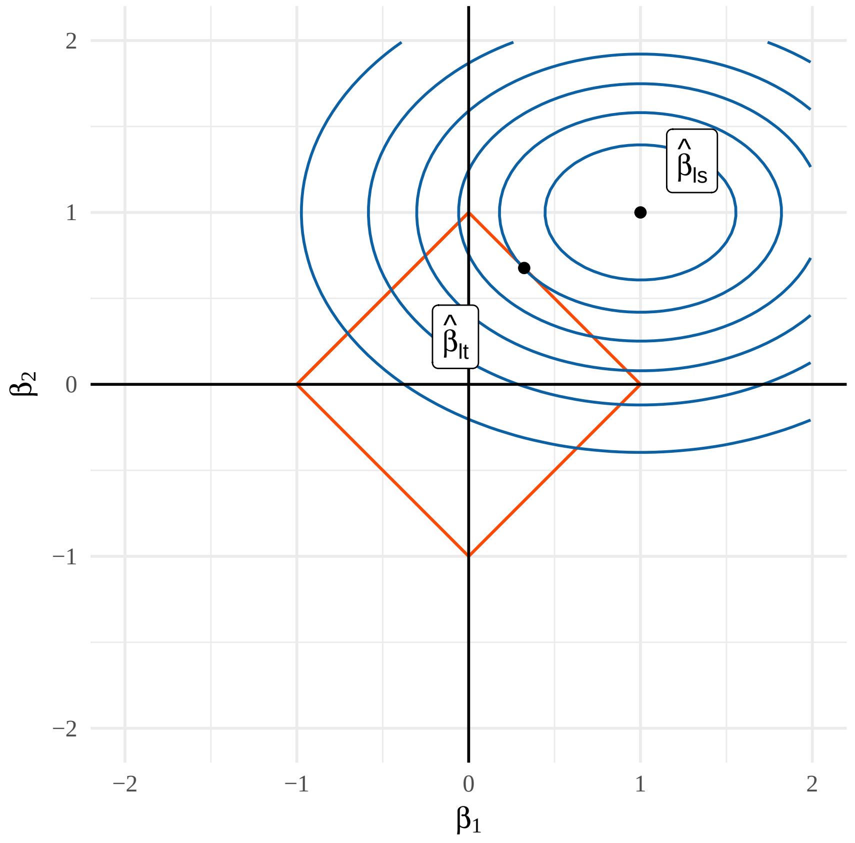
Ridge regression is convex (easy to solve), but doesn't do model selection

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)

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

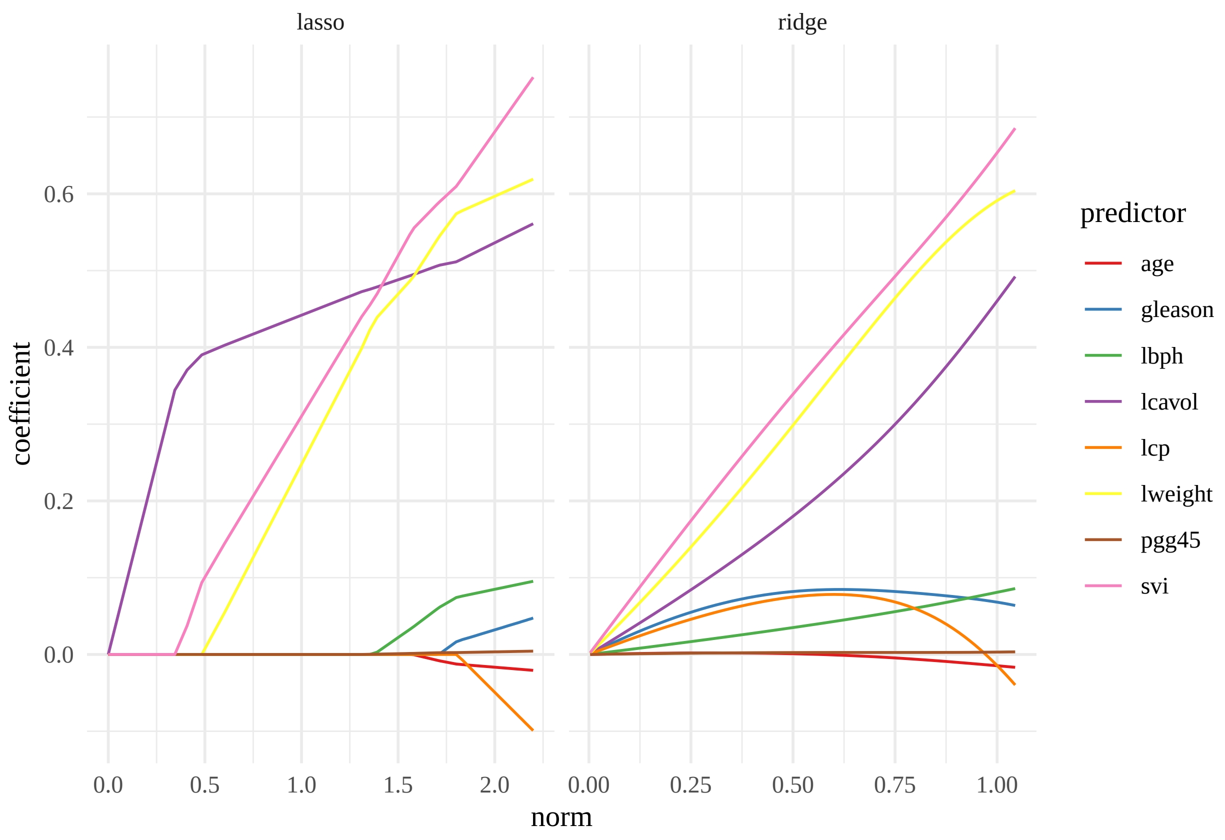
$$\widehat{\beta}_{l,\lambda} = \arg\min_{\beta} ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1 = ??$$

doesn't have a closed form solution.

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

We'll talk algorithms next week.

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 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)

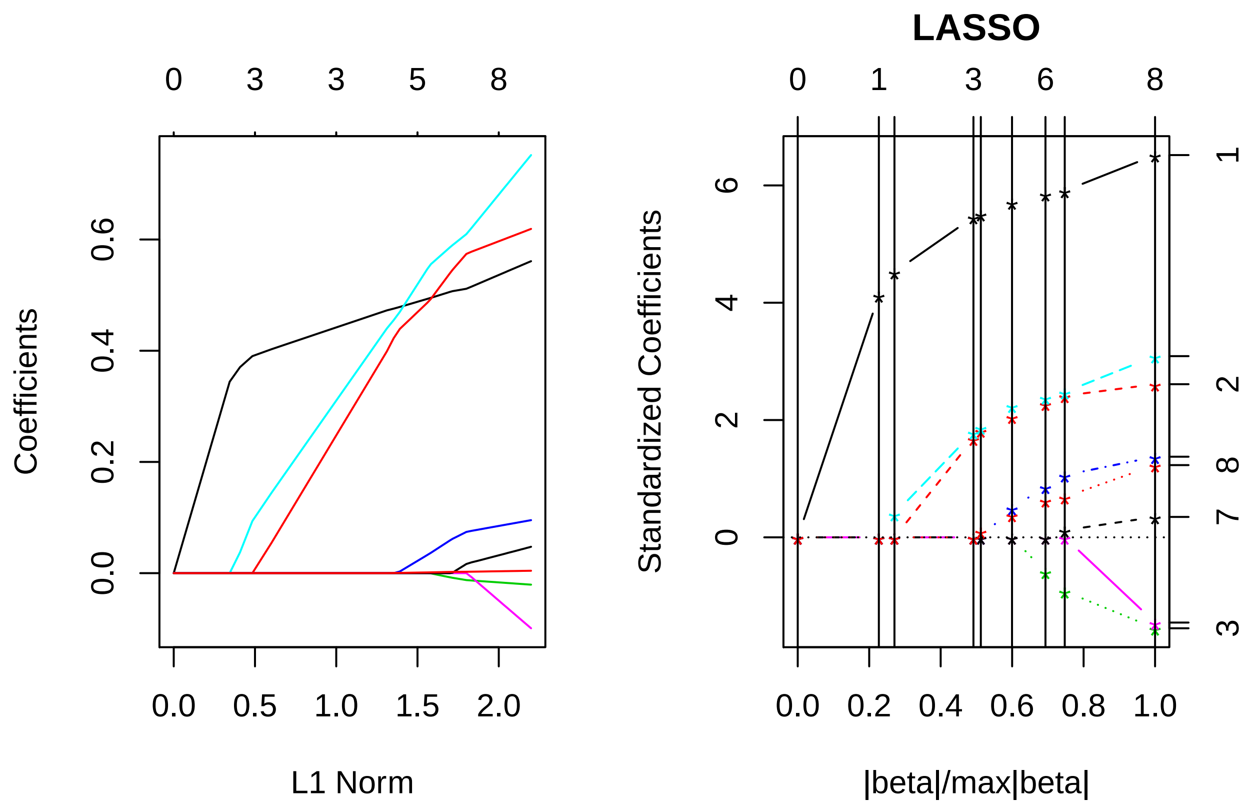
These use different algorithms, but both compute the ~~path~~ for a range of λ.

`lars`
- starts with an empty model and adds coefficients until saturated
- Uses the entire sequence of λ based on the nature of the optimization problem.
- Doesn't support other likelihoods
- Slower - The path returned by `lars` as more useful than that returned by `glmnet`.

`glmnet`
- starts with an empty model and examines each value of λ using previous values as "warm starts".
- $\lambda \in [\epsilon \|\mathbb{X}'Y\|_\infty, \|\mathbb{X}'Y\|_\infty)$ for some small ϵ
- It is generally much faster than `lars` and uses lots of other tricks (as well as compiled code) for extra speed.
- Easier to cross validate sensibly
- Can ocassionaly give boundary solutions, or too many non-zero coefs

Lasso paths

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



Lasso theory under strong conditions

Support recovery (Wainwright 2009), see also (Meinshausen and Bühlmann 2006; Zhao and Yu 2006)

1. The truth is linear.
2. $\left\|\mathbb{X}'_{S^c}\mathbb{X}_S(\mathbb{X}'_S\mathbb{X}_S)^{-1}\right\|_\infty < 1-\epsilon$.
3. $\lambda_{\min}(\mathbb{X}'_S\mathbb{X}_S) \geq C_{\min} > 0$.
4. The columns of $\mathbb{X}$ have 2-norm n.
5. The noise is iid Normal.
6. λ_n satisfies $\frac{n\lambda^2}{\log(p-s)} \to \infty$.
7. $\min_j\{|\beta_j| : j \in S\} \geq \rho_n > 0$ and

$$\rho_n^{-1}\left(\sqrt{\frac{\log s}{n}} + \lambda_n \left\|(\mathbb{X}'_S\mathbb{X}_S)^{-1}\right\|_\infty\right) \to 0$$

Then, $P(\operatorname{supp}(\widehat{\beta}_{l,\lambda}) = \operatorname{supp}(\beta_*)) \to 1$.

Estimation consistency (Negahban et al. 2012) also (Meinshausen and Yu 2009)

1. The truth is linear.
2. $\exists\kappa$ such that for all vectors $\theta \in \mathbb{R}^p$ that satisfy $\|\theta_{S^C}\|_1 \leq 3\|\theta_S\|_1$, we have $\|X\theta\|_2^2/n \geq \kappa\|\theta\|_2^2$ (Compatibility)
3. The columns of $\mathbb{X}$ have 2-norm n.
4. The noise is iid sub-Gaussian.
5. $\lambda_n > 4\sigma\sqrt{\log(p)/n}$.

Then, with probability at least $1 - c\exp(-c'n\lambda_n^2)$,

$$\left\|\widehat{\beta}_{l,\lambda} - \beta_*\right\|_2^2 \leq \frac{64\sigma^2}{\kappa^2}\frac{s\log p}{n}.$$

Warning: These conditions are very strong, uncheckable in practice, unlikely to be true for real datasets. But theory of this type is the standard for these procedures.

Lasso under weak/no conditions

If Y and X are bounded by B, then with probability at least $1-\delta^2$,

$$R_n(\widehat{\beta}_{l,t}) - R_n(\beta_*) \leq \sqrt{\frac{16(t+1)^4B^2}{n}\log\left(\frac{\sqrt{2}p}{\delta}\right)}.$$

This is a simple version of a result in (Greenshtein and Ritov 2004).

Note that it applies to the L_1 version rather than the Lagrangian formulation.

The boundedness can be replaced by sub-Gaussian or other conditions.

Bartlett, Mendelson, and Neeman (2012) derives the same rate for the Lagrangian form up to log factors.

Again, this rate is (nearly) optimal:

$$c\sqrt{\frac{s}{n}} < C\sqrt{\frac{s\log p}{n}}.$$

$\log p$ is the penalty you pay for selection.

Model selection

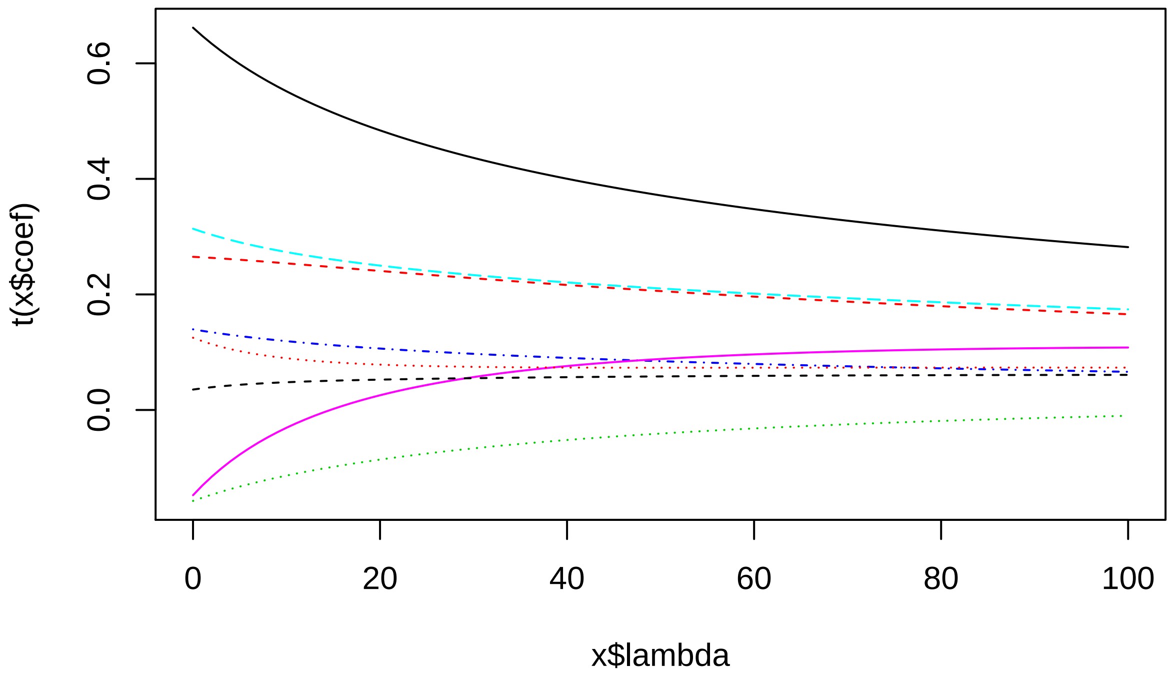
Choosing the lambda

- You have to choose λ in lasso or in ridge regression

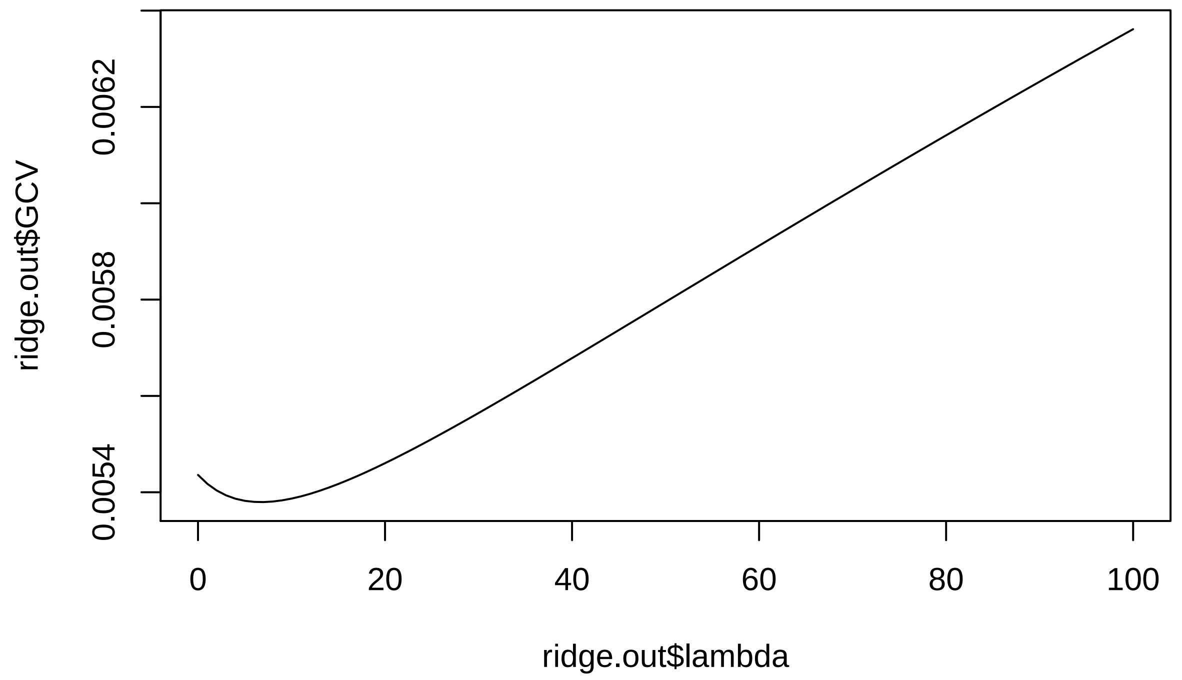
- lasso selects a model (by setting coefficients to zero), but the value of λ determines how many/which.
- All of these packages come with CV built in.
- However, the way to do it differs from package to package (whomp whomp)

Ridge regression, `lm.ridge` version

```r
library(MASS)
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)
```



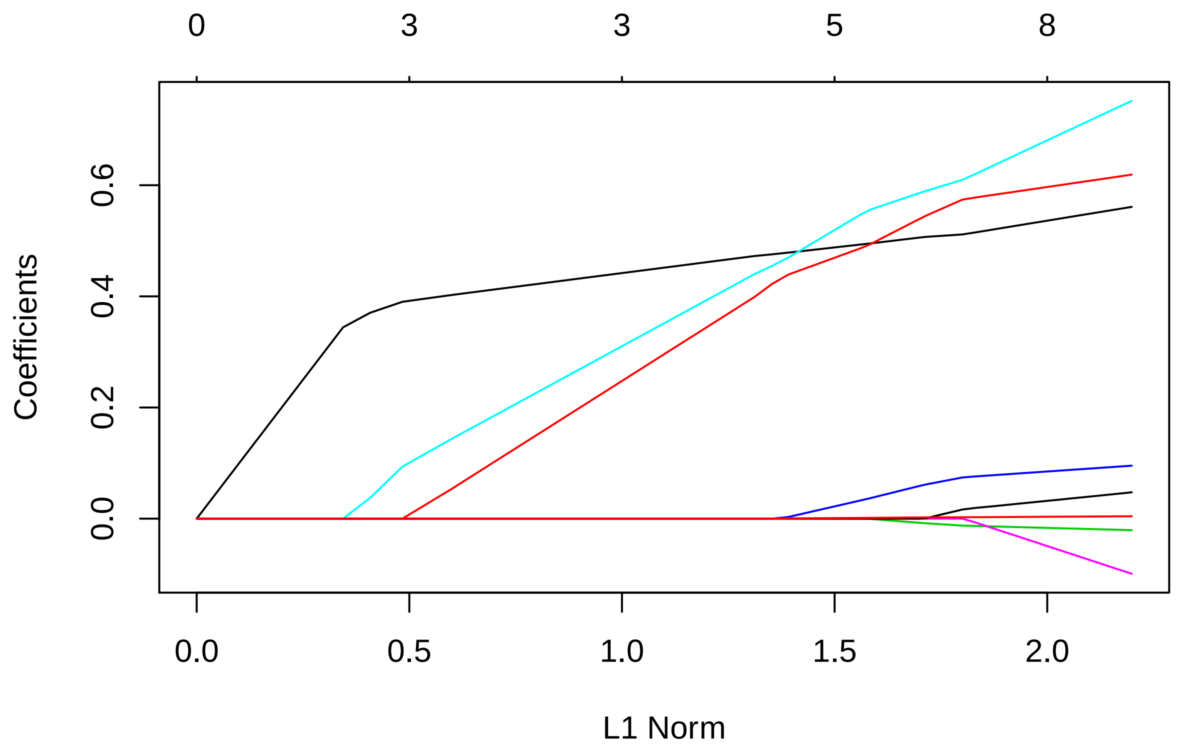
```r
# (2a). If you chose lambda poorly, this will look bad, try again
# 3. Choose lambda using GCV
plot(ridge.out$lambda,ridge.out$GCV,ty='l')
```



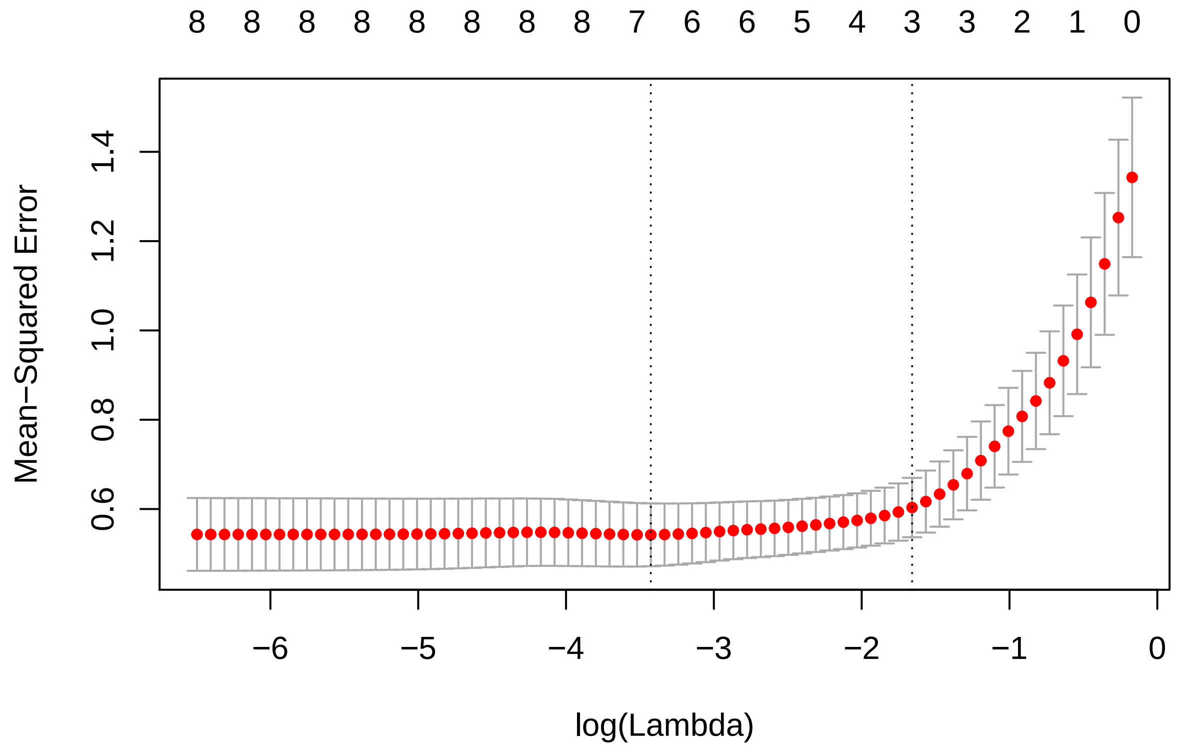
```
# 4. If there's a minimum, FIND IT, else try again
best.lam = which.min(ridge.out$GCV)
# 5. Return the coefs/predictions for the best model
coefs = coefficients(ridge.out)[best.lam,]
preds = as.matrix(dplyr::select(prostate,-lpsa,-train)) %*% coefs[-1] + coefs[1]
```

`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`
```

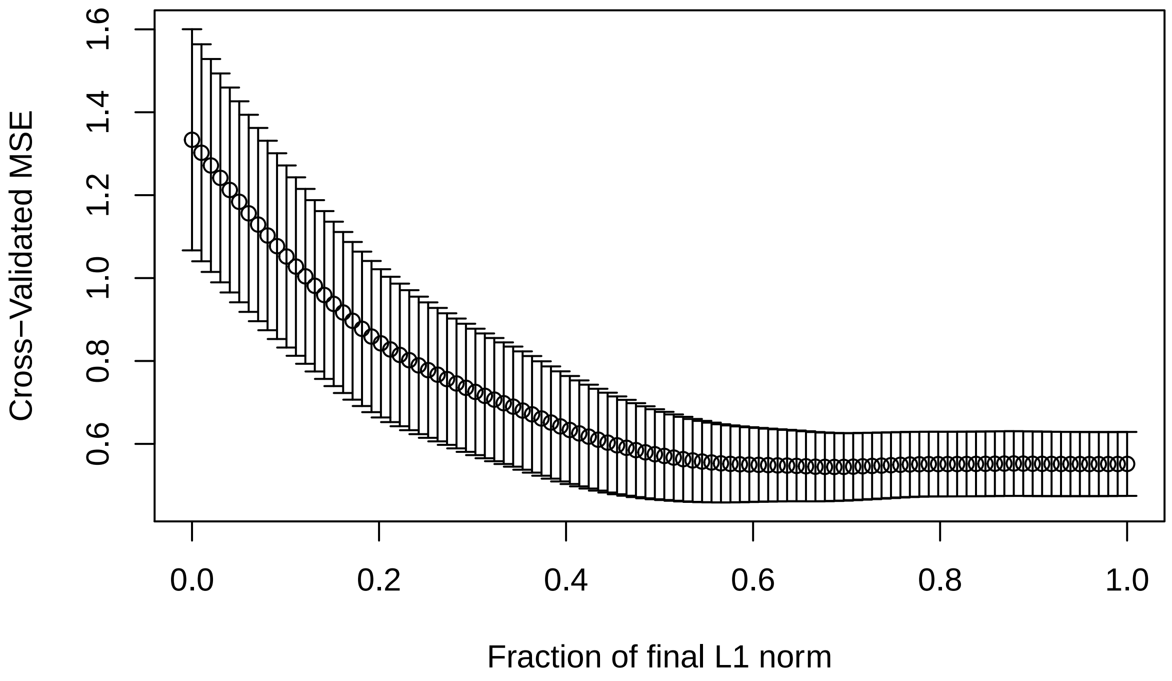
lars version

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.

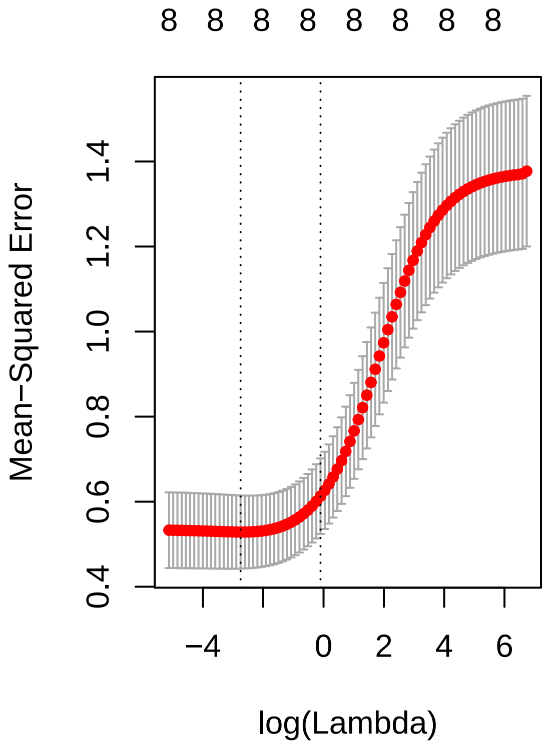
```
# 1. Estimate cv, no formula version
lasso.lars.cv = cv.lars(X,Y) # also plots it
```

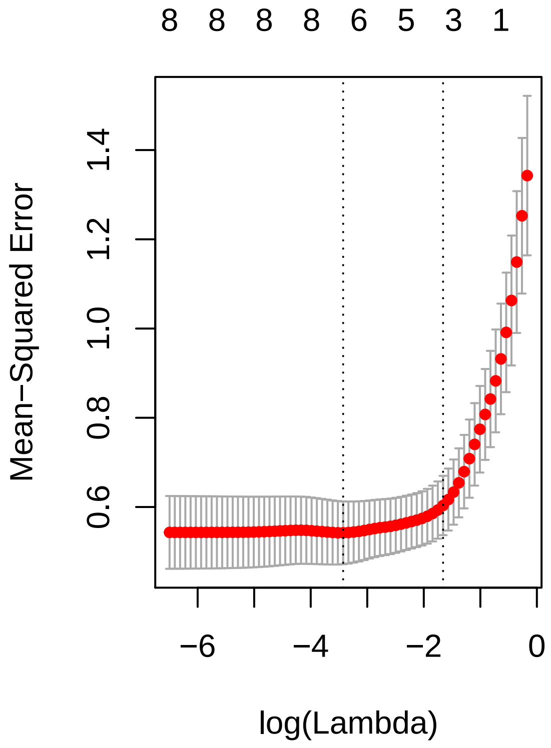


```
# 2. Choose lambda using CV
best.lam.lars = lasso.lars.cv$index[which.min(lasso.lars.cv$cv)] # the location, not the value
# 3. Estimate the lasso and plot
lasso.lars = lars(X,Y) # still the whole path
# 5. Return the coefs/predictions for the best model
coefs.lars = coefficients(lasso.lars, s = 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`
```

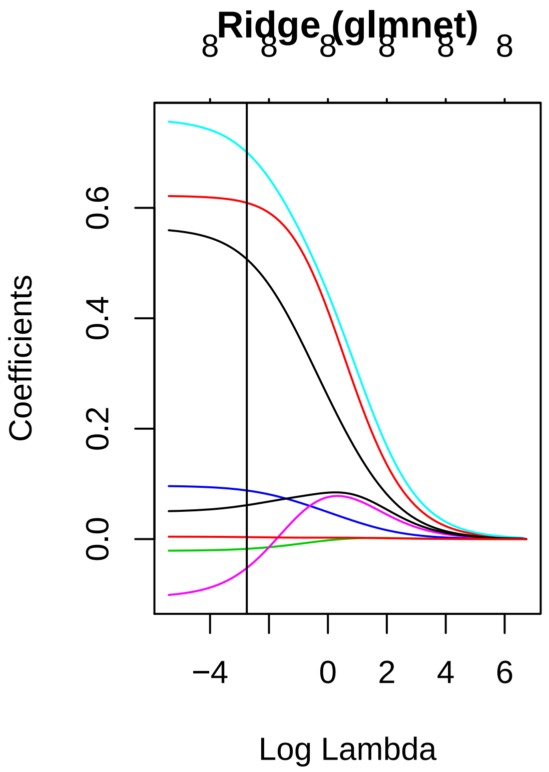
Paths with chosen lambda (lasso and ridge)

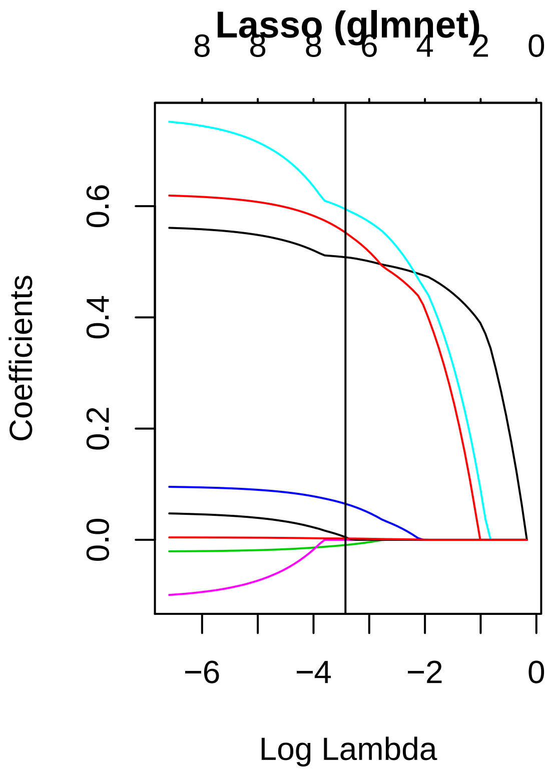
```
ridge.glmnet = cv.glmnet(X,Y,alpha=0,lambda.min.ratio=1e-10) # added to get a minimum
par(mfrow=c(1,2))
plot(ridge.glmnet)
plot(lasso.glmnet)
```





```r
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))
```





References

Bartlett, Peter L., Shahar Mendelson, and Joseph Neeman. 2012. “ℓ_1-Regularized Linear Regression: Persistence and Oracle Inequalities.” *Probability Theory and Related Fields* 154 (1-2): 193–224.

Berk, Richard, Lawrence Brown, Andreas Buja, Edward George, Emil Pitkin, Kai Zhang, and Linda Zhao. 2014. “Misspecified Mean Function Regression: Making Good Use of Regression Models That Are Wrong.” *Sociological Methods & Research* 43 (3): 422–51.

Greenshtein, Eitan, and Ya’acov Ritov. 2004. “Persistence in High-Dimensional Linear Predictor Selection and the Virtue of Overparametrization.” *Bernoulli* 10 (6): 971–88.

Györfi, László, Michael Kohler, Adam Krzyżak, and Harro Walk. 2002. *A Distribution-Free Theory of Nonparametric Regression.* Springer-Verlag, New York.

Hsu, Daniel, Sham M Kakade, and Tong Zhang. 2014. “Random Design Analysis of Ridge Regression.” *Foundations of Computational Mathematics* 14 (3): 569–600.

James, Gareth M. 2003. “Variance and Bias for General Loss Functions.” *Machine Learning* 51 (2): 115–35.

Meinshausen, Nicolai, and Peter Bühlmann. 2006. “High-Dimensional Graphs and Variable Selection with the Lasso.” *The Annals of Statistics* 34 (3): 1436–62.

Meinshausen, Nicolai, and Bin Yu. 2009. “Lasso-Type Recovery of Sparse Representations for High-Dimensional Data.” *The Annals of Statistics* 37 (1): 246–70.

Negahban, Sahand, Pradeep Ravikumar, Martin J Wainwright, and Bin Yu. 2012. “A Unified Framework for High-Dimensional Analysis of M-Estimators with Decomposable Regularizers.” *Statistical Science* 27: 538–337.

Wainwright, Martin J. 2009. “Sharp Thresholds for High-Dimensional and Noisy Sparsity Recovery Using ℓ_1-Constrained Quadratic Programming (Lasso).” *IEEE Transactions on Information Theory* 55 (5): 2183–2202.

Zhao, Peng, and Bin Yu. 2006. “On Model Selection Consistency of Lasso.” *The Journal of Machine Learning Research* 7: 2541–63.