Linear Methods for Regression

-Statistical Machine Learning-

Lecturer: Darren Homrighausen, PhD

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

The quality of this prediction is measured via the prediction risk[1]

$$R(\hat{f}) = \mathbb{P}_{\mathcal{D},Z}(Y - \hat{f}(X))^2.$$

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

However, it is *unknown*.

[1] Note: sometimes we integrate with respect to $\mathcal{D}$ only, Z only, neither (loss), or both.

Prediction risk for regression

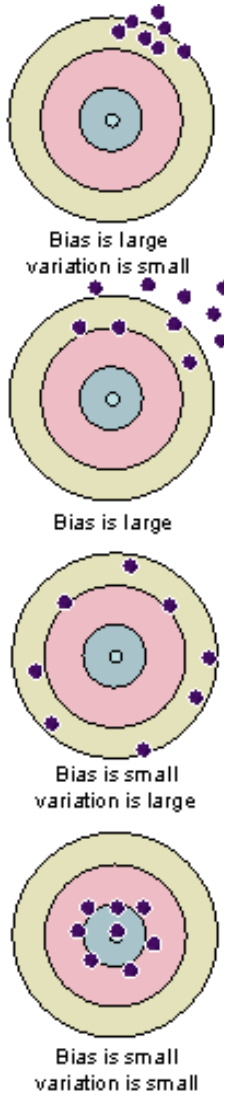
Note that $R(\hat{f})$ can be written as

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

where

$$\text{bias}(x) = \mathbb{P}\hat{f}(x) - f_*(x)$$
$$\text{var}(x) = \mathbb{V}\hat{f}(x)$$
$$\sigma^2 = \mathbb{P}(Y - f_*(X))^2$$

(As an aside, this decomposition applies to much more general loss functions[a]) [b]



[a] *Variance and Bias for General Loss Functions*; **James**, Machine Learning 2003

[b] (http://home.ubalt.edu/ntsbarsh/Business-stat/)

Bias-variance tradeoff

This can be heuristically thought of as

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

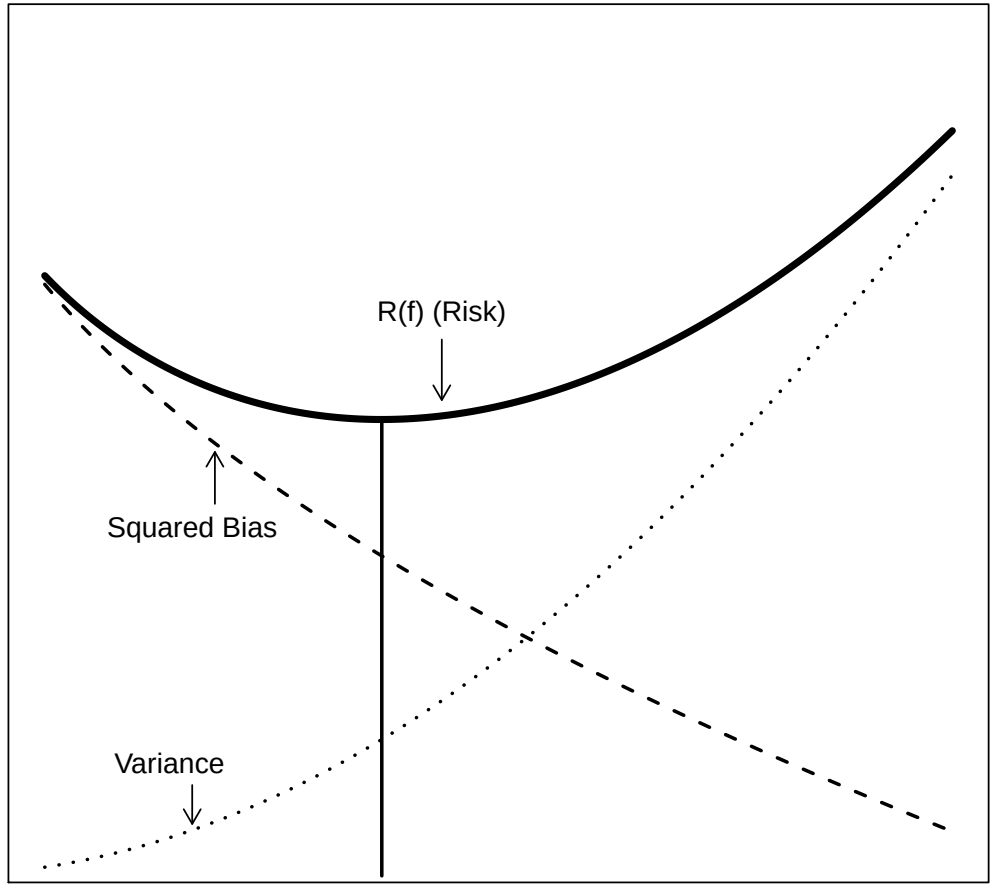
There is a natural conservation between these quantities

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

The opposite also holds
(Think: $\hat{f} \equiv 0$.)

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

Bias-variance tradeoff



Training error and risk estimation

Using $\hat{\mathbb{P}}$, and writing $\ell_f(Z) = (f(X) - Y)^2$, we can form the training error

$$\hat{R}(f) = \hat{\mathbb{P}}\ell_f = \frac{1}{n}\sum_{i=1}^{n} \ell_f(Z_i) = \frac{1}{n}\sum_{i=1}^{n}(f(X_i) - Y_i)^2.$$

In many cases in applied statistical applications, this plug-in estimator is used

(Think of how many techniques rely on an unconstrained minimization of squared error, or maximum likelihood, or estimating equations, or ...)

This sometimes has disastrous results

Example

Let's suppose $\mathcal{D}$ is drawn from

```
n = 30
X = (0:n)/n*2*pi
Y = sin(X) + rnorm(n,0,.25)
```

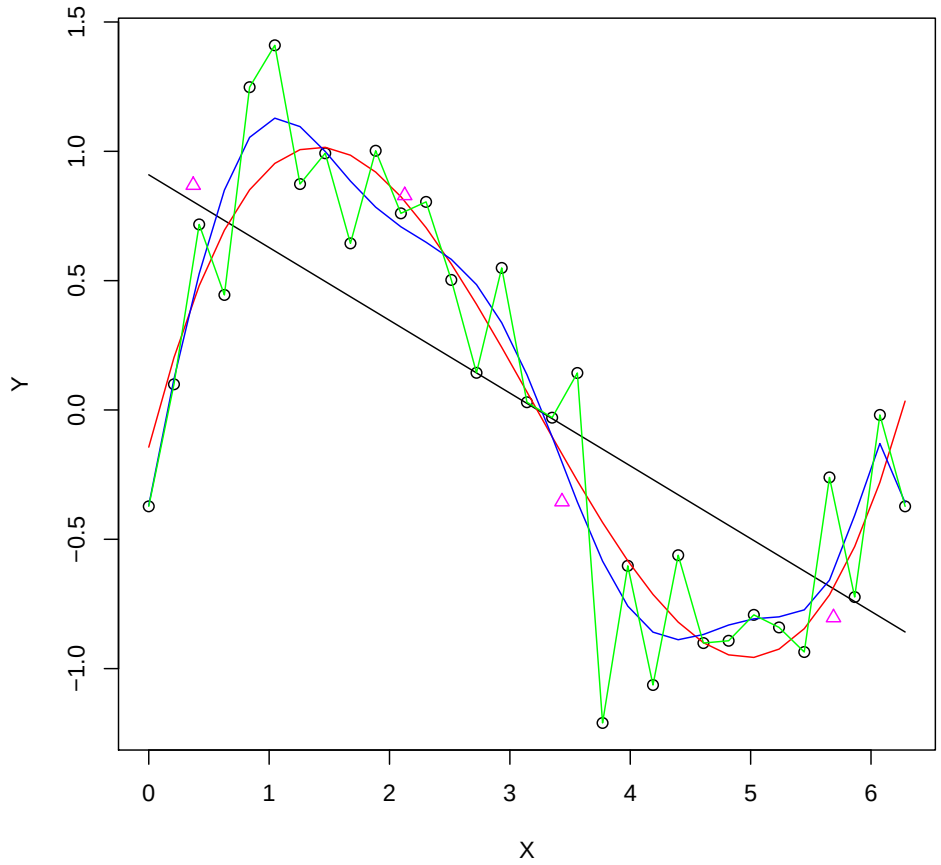
Now, let's fit some polynomials to this data.

We consider the following models:

- Model 1: $f(X_i) = \beta_0 + \beta_1 X_i$
- Model 2: $f(X_i) = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3$
- Model 3: $f(X_i) = \sum_{k=0}^{10} \beta_k X_i^k$
- Model 4: $f(X_i) = \sum_{k=0}^{n-1} \beta_k X_i^k$

Let's look at what happens...

Example



The $\hat{R}$'s are:

$\hat{R}(\text{Model 1}) = 10.98$

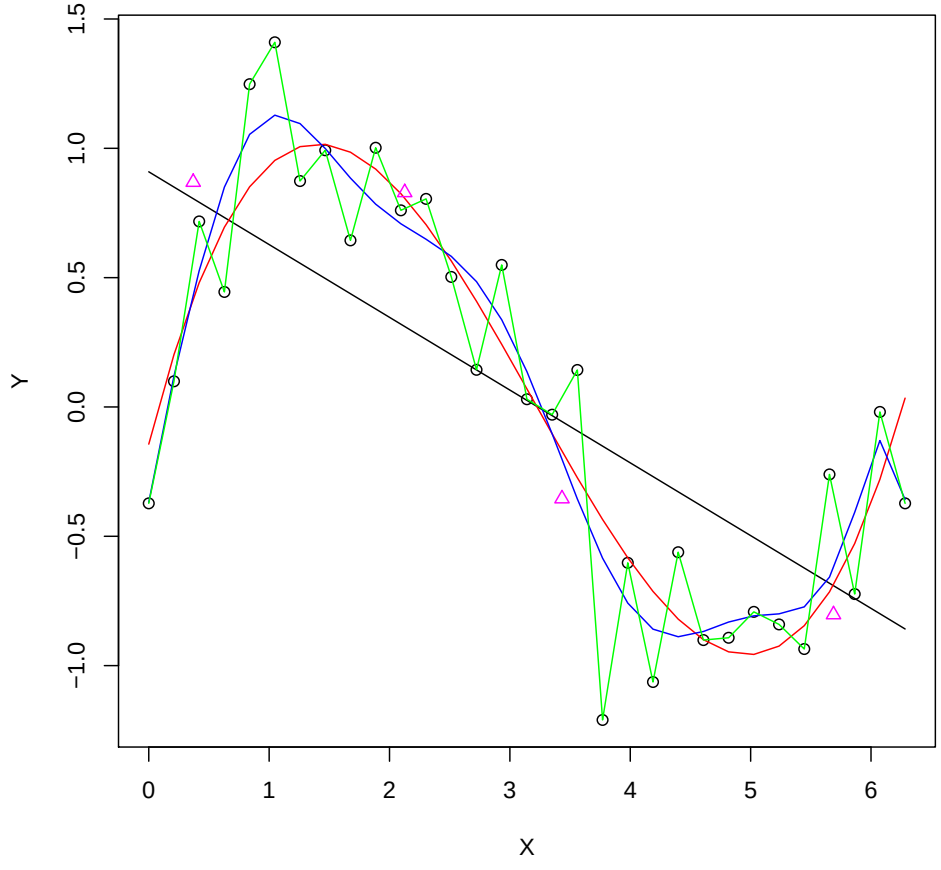
$\hat{R}(\text{Model 2}) = 2.86$

$\hat{R}(\text{Model 3}) = 2.28$

$\hat{R}(\text{Model 4}) = 0$

What about predicting new observations (Δ)?

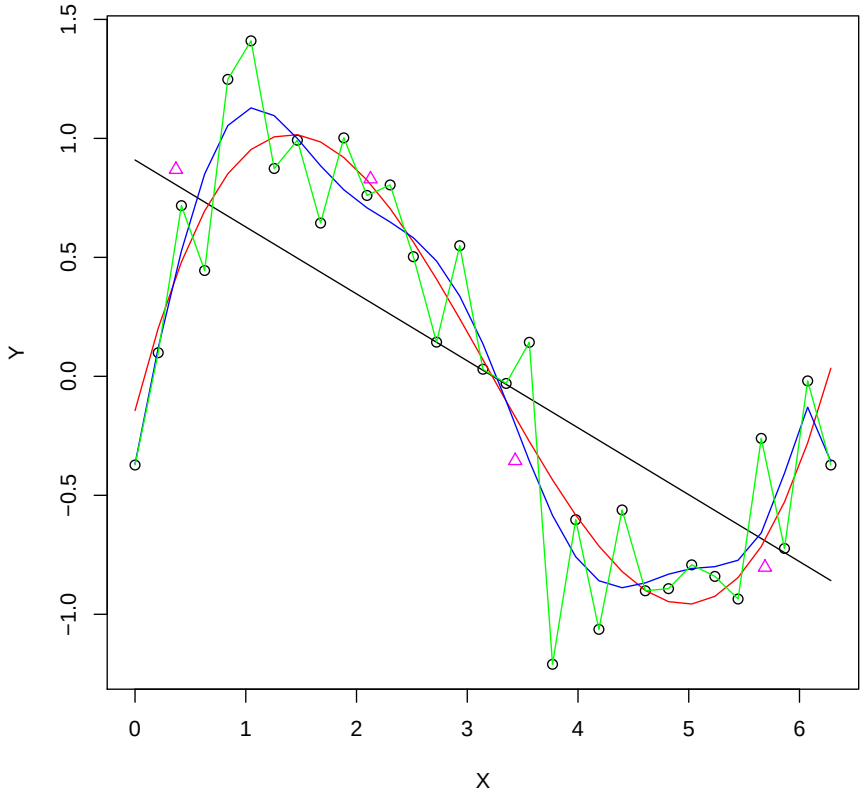
Example

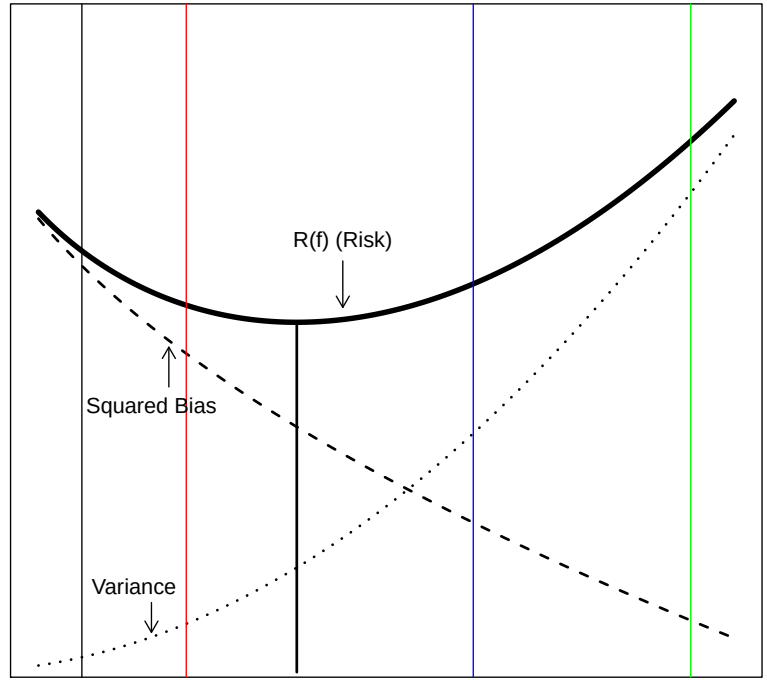


- Black model has low variance, high bias
- Green model has low bias, but high variance
- Red model and Blue model have intermediate bias and variance.

We want to balance these two quantities.

Bias vs. Variance





Model Complexity ↗

A linear model

A linear model

Specify functions $\phi_k : \mathbb{R}^p \to \mathbb{R}$, $k = 1, \ldots, K$

$$\mathbb{X} = [\phi_k(X_i)] = \begin{bmatrix} \Phi(X_1)^\top \\ \Phi(X_2)^\top \\ \vdots \\ \Phi(X_n)^\top \end{bmatrix} \in R^{n \times K},$$

where $\Phi(\cdot)^\top = (\phi_1(\cdot), \ldots, \phi_K(\cdot))$.

Here, $f_*(X) = f_{*,\Phi_K}(X) + f_{*,\Phi_K^c}(X)$ and

$$\Phi_K = \{f : \exists(\beta_k)_{k=1}^K \text{ such that } f = \sum_{k=1}^K \beta_k \phi_k = \beta^\top \Phi\}$$

and

$$f_{*,\Phi_K} = \operatorname*{argmin}_{f \in \Phi_K} \mathbb{P}\ell_f.$$

The function f_{*,Φ_K} is known as the linear oracle

A linear model: Ordinary Least Squares

Let $K = p$ and define ϕ_k to be the coordinate projection map

That is, for $x = (x_1, \ldots, x_p)^\top$

$$\phi_k(x) \equiv x_k$$

Then we recover the usual linear regression formulation

$$\mathbb{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & & & \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix} = \begin{bmatrix} X_1^\top \\ X_2^\top \\ \vdots \\ X_n^\top \end{bmatrix}.$$

Commonly, $X_{i1} = 1$, which encodes an intercept term in the model.

A linear model: Orthogonal basis expansion

Suppose $f_* \in \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space with norm induced by the inner product $\langle \cdot, \cdot \rangle$.

Let $(\phi_k)_{k=1}^{\infty}$ be an orthonormal basis for $\mathcal{F}$

Write

$$f_* = \sum_{k=1}^{\infty} \langle f_*, \phi_k \rangle \phi_k = \sum_{k=1}^{\infty} \beta_k \phi_k$$

Then we can estimate f_{*,Φ_K} by finding the coefficients of the projection on Φ_K.

By Parseval's theorem$^\sharp$for Hilbert spaces this induces an approximation error of $\sum_{k=K+1}^{\infty} \beta_k^2$.

This is small if f_* is smooth

(for instance, if f_* has m derivatives, then $\beta_k \asymp k^{-m}$)

A linear model: Neural Nets

Let

$$\phi_k(x) = \sigma(\alpha_k^\top x + b_k),$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function.

Then f_{*,Φ_K} is called an[1]:

"single-layer feed-forward neural network model with linear output"

[1] More on this later

A linear model: Radial basis functions

Let

$$\phi_k(x) = e^{-||\mu_k - x||_2^2/\lambda_k}.$$

Then f_{*,Φ_K} is called an[2]:

"Gaussian radial-basis function estimator".

This turns out to be a parametric form of a more general technique known as Gaussian process regression.

[2] More on this later

The limits of linearity

Each of these methods have parameters to choose:

- Ordinary least squares: how large is p?
- Orthogonal basis expansion: which basis and how large is K?
- Neural Nets: The activation function σ, the directions α_k, bias terms b_k, as well as K.
- Radial basis functions: The choice of the kernel (here we have used Gaussian), the centroids μ_k, the scales λ_k, and K again.

We would like the data to inform these parameters

However, this requirement turns the problem from a straightforward optimization problem (with closed form solution) into a combinatorially hard nonlinear problem (In fact, NP hard)

In practice, we use greedy algorithms, iterative schemes, and convex relaxation to solve the computational problem.

The statistical problem is still fundamentally projection onto a function space, with the function space estimated from the data.

Low-dimensional linear regression ($K \leq n$)

Supposing that the functions ϕ_k are *known*, we can form the least-squares estimator by "plugging in" the empirical measure for the unknown true measure

$$\hat{f} = \operatorname*{argmin}_{f \in \Phi_K} \hat{\mathbb{P}} \ell_f,$$

where, for each $f \in \Phi_K$, there is a coefficient vector β such that $\ell_f(Z) = (X^\top \beta - Y)^2$.

In this case,

$$\hat{f}(X) = X^\top \hat{\beta},$$

where

$$\hat{\beta} = \mathbb{X}^\dagger Y = (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top Y$$

and $\mathbb{X}^\dagger$ is the Moore-Penrose pseudo inverse

In particular, the fitted values are $\mathbb{X}\hat{\beta} = HY$, where H is the orthogonal projection onto the column space of $\mathbb{X}$

Low-dimensional linear regression ($K \leq n$)

We can examine the first and second moment properties of $\hat{\beta}$
Focusing on the $f_{*,\Phi_K} = \beta^\top \Phi$ component[3]

$$\mathbb{E}\hat{\beta} = \beta \qquad \text{(unbiased)} \tag{1}$$

$$\mathbb{V}\hat{\beta} = \mathbb{X}^\dagger(\mathbb{V}Y)(\mathbb{X}^\dagger)^\top = \mathbb{V}[Y_i](\mathbb{X}^\top\mathbb{X})^{-1} \tag{2}$$

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

(That is, equation (2) is minimized subject to equation (1))

Also, it is the maximum likelihood estimator under a homoskedastic, independent Gaussian model

(Hence, it is asymptotically efficient)

Does that necessarily mean it is any good?

[3] This is important! In claiming $\hat{\beta}$ is 'unbiased' we are asserting either that: (1) $f_{*,\Phi_K^c} \equiv 0$ or (2) it is unbiased for the coefficients of the linear oracle f_{*,Φ_K}.

Low-dimensional linear regression ($K \leq n$)

Write $\mathbb{X} = UDV^\top$ for the SVD of $\mathbb{X}$

Then $\mathbb{V}\hat{\beta} \propto (\mathbb{X}^\top \mathbb{X})^{-1} = VD^{-2}V^\top$.

We know that the diagonal elements of D, d_j, are then lengths of the axes of the ellipse induced by $\mathbb{X}$

Also, suppose we are interested in estimating β,

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

Even in the low-dimensional case, we can do arbitrarily badly if $d_p \approx 0$.

Returning to polynomial example: Bias

Using a Taylor's series, for all x

$$\sin(x) = \sum_{q=0}^{\infty} \frac{(-1)^q x^{2q+1}}{(2q+1)!}$$

Therefore, higher order polynomial models will reduce the approximation (bias) part

Returning to polynomial example: Variance

However, the least squares solution is given by solving $Ax = b$, where

$$A = \begin{bmatrix} 1 & X_1 & \dots & X_1^{K-1} \\ & \vdots & & \\ 1 & X_n & \dots & X_n^{K-1} \end{bmatrix},$$

is the associated Vandermonde$^\sharp$matrix.

This matrix is well known for being numerically unstable as K increases (Letting $A = UDV^\top$, this means that $d_1/d_K \to \infty$)

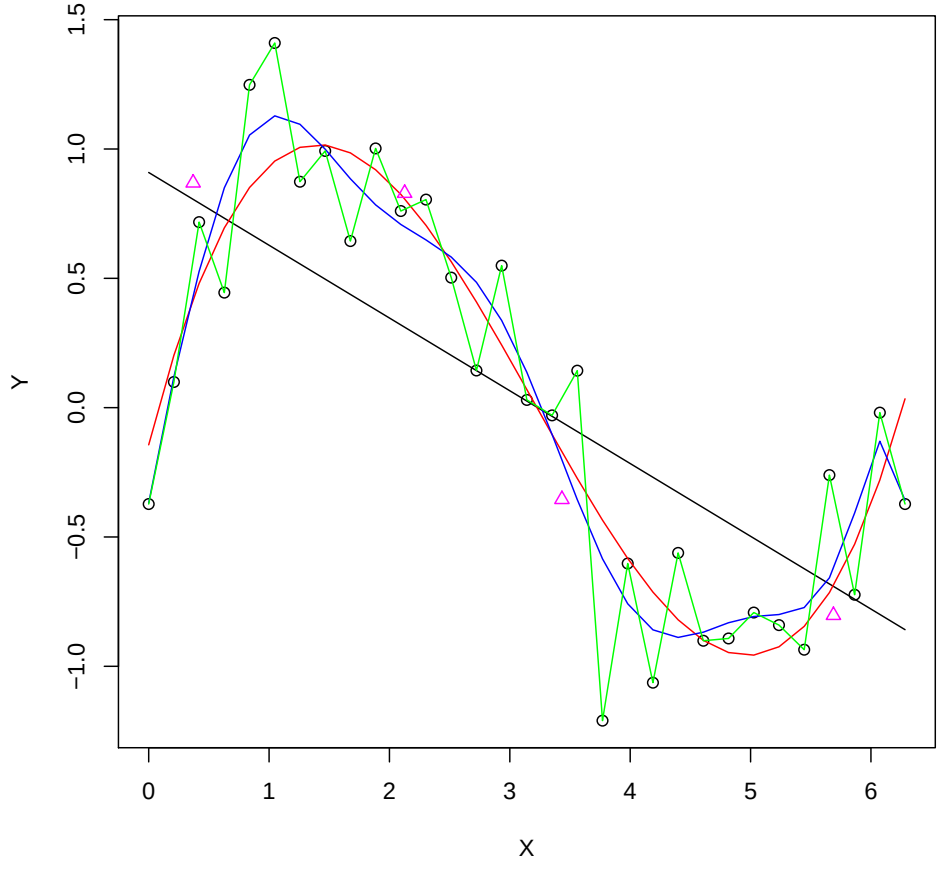
Hence[4]

$$||(A^\top A)^{-1}||_2 = \frac{1}{d_K^2}$$

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

[4] This should be compared with the variance computation in equation (2)

Returning to the polynomial example



Many questions left to answer

- Is a linear estimator appropriate (what is the size of $f_{*,\Phi_K^c}(X)$)?
- What is a good choice of ϕ_k?
- How well does $\operatorname{argmin}_{f \in \Phi_K} \hat{\mathbb{P}} \ell_f$ mimic $\operatorname{argmin}_{f \in \Phi_K} \mathbb{P} \ell_f$?
- What is a good choice of K? Are all the ϕ_k needed?
- What happens if we want to choose $K > n$?

Subset selection, regularization, and risk estimation

Subset selection and regularization

For now, let's assume we are doing ordinary least squares, and hence the design (feature) matrix is $\mathbb{X} \in \mathbb{R}^{n \times p}$.

We want to do model selection for at least two reasons:

- Prediction accuracy: Can essentially *always* be improved by introducing some bias
- Interpretation: A large number of features can sometimes be distilled into a smaller number that comprise the "big (little?) picture"

We will address three related ideas

- Model selection: Selection of only some of the original p features
- Dimension reduction/expansion: Creation of new features to help with prediction
- Regularization: Add constraints to optimization problems to provide stabilization

Risk estimation

Reminder: Prediction risk is

$$R(f) = \mathbb{P}_{Z,\mathcal{D}}\ell_f \leftrightarrow \text{Bias} + \text{Variance}$$

The overridding theme is that we would like to add a judicious amount of bias to get lower risk

As R isn't known, we need to estimate it

As discussed, $\hat{R}_{\text{train}} = \hat{\mathbb{P}}\ell_f$ isn't very good

(In fact, one tends to not add bias when estimating R with $\hat{\mathbb{P}}\ell_f$)

Risk estimation: a general form

Assume that we get a new draw of the training data, $\mathcal{D}^0$, such that $\mathcal{D} \sim \mathcal{D}^0$ and

$$\mathcal{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \quad \text{and} \quad \mathcal{D}^0 = \{(X_1, Y_1^0), \ldots, (X_n, Y_n^0)\}$$

If we make a small compromise to risk, we can form a sensible suite of risk estimators

To wit, letting $Y^0 = (Y_1^0, \ldots, Y_n^0)^\top$, define

$$R_{in} = \mathbb{E}_{Y^0|\mathcal{D}} \hat{\mathbb{P}}_{\mathcal{D}^0} \ell_{\hat{f}} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Y^0|\mathcal{D}} \ell(\hat{f}(X_i), Y_i^0)$$

Then the average optimism is

$$\text{opt} = \mathbb{E}_Y[R_{in} - \hat{R}_{\text{train}}]$$

Typically, opt is positive as $\hat{R}_{\text{train}}$ will underestimate the risk

Risk estimation: a general form

It turns out for a variety of ℓ (such as squared error and 0-1)

$$\text{opt} = \frac{2}{n} \sum_{i=1}^{n} \text{Cov}(\hat{f}(X_i), Y_i)$$

Therefore, we get the following expression of risk

$$\mathbb{E}_Y R_{in} = \mathbb{E}_Y \hat{R}_{\text{train}} + \frac{2}{n} \sum_{i=1}^{n} \text{Cov}(\hat{f}(X_i), Y_i),$$

which has unbiased estimator (i.e. $\mathbb{E}_Y R_{\text{gic}} = \mathbb{E}_Y R_{in}$)

$$R_{\text{gic}} = \hat{R}_{\text{train}} + \frac{2}{n} \sum_{i=1}^{n} \text{Cov}(\hat{f}(X_i), Y_i)$$

Degrees of freedom

We call the term (where $\sigma^2 = \mathbb{V} Y_i$)

$$\mathrm{df} = \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}(\hat{f}(X_i), Y_i)$$

the degrees of freedom

(This is really the effective number of parameters, with some caveats)

Our task now is to either estimate or compute opt to produce $\widehat{\mathrm{opt}}$ and form:

$$\hat{R}_{\mathrm{gic}} = \hat{R}_{\mathrm{train}} + \widehat{\mathrm{opt}}$$

This leads to AIC, BIC, Cp, and others

(See the homework for an exploration of this topic)

Comparing probability measures

Kullback-Leibler

We've produced a suite of (generalized) information criteria

Suppose we have data Y that comes from the probability density function f.

What happens if we use the probability density function g instead?

One central idea is Kullback-Leibler$^\sharp$discrepancy[5]

$$\begin{aligned} KL(f,g) &= \int \log\left(\frac{f(y)}{g(y)}\right) f(y)dy \\ &\propto -\int \log(g(y))f(y)dy \qquad \text{(ignore term without } g\text{)} \\ &= -\mathbb{P}_f[\log(g(Y))] \end{aligned}$$

This gives us a sense of the loss incurred by using g instead of f.

[5] This has many features of a distance, but is not a true distance as $KL(f,g) \neq KL(g,f)$.

Kullback-Leibler discrepancy

Usually, g will depend on some parameters, call them θ

Example: In regression, we can specify $f = N(X^\top \beta, \sigma^2)$ for a fixed (true)[6] β, and let $g_\theta = N(X^\top \beta, \sigma^2)$ over all $\theta \in \mathbb{R}^p \times \mathbb{R}^+$

As $KL(f, g_\theta) = -\mathbb{P}_f[\log(g(Y;\theta))]$, we minimize this over θ.

Again, f is unknown, so we minimize $-\log(g(Y;\theta))$ instead. This is the maximum likelihood value

$$\hat{\theta}_{ML} = \arg\max_\theta g(y;\theta)$$

[6] We actually don't need to assume things about a true model nor have it be nested in the alternative models.

Kullback-Leibler discrepancy

Now, to get an operational characterization of the KL divergence at the ML solution

$$-\mathbb{P}[\log(g(Y;\hat{\theta}_{ML}))]$$

we need an approximation (don't know f, still)

This approximation[7] is exactly AIC$^\sharp$:

$$\mathrm{AIC} = -2\log(g(Y;\hat{\theta}_{ML})) + 2|\hat{\beta}_{ML}|$$

Example: Let $\log(g(y;\theta)) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}||Y - \mathbb{X}\beta||_2^2$

σ^2 KNOWN: $\hat{\beta} = \mathbb{X}^\dagger Y$

$$\mathrm{AIC} \propto n\hat{R}_{\mathrm{train}}/\sigma^2 + 2p.$$

σ^2 UNKNOWN: $\hat{\beta} = \mathbb{X}^\dagger Y$, $n\hat{\sigma}^2 = (I - \mathbb{X}\mathbb{X}^\dagger)Y = n\hat{R}_{\mathrm{train}}$

$$\mathrm{AIC} \propto n\log(\hat{R}_{\mathrm{train}}) + 2p$$

[7] See "Multimodel Inference" Burnham, Anderson (2004)

Cross-validation

A different approach to risk estimation

Let (X_0, Y_0) be a test observation, identically distributed as an element in $\mathcal{D}$, but also independent of $\mathcal{D}$.

Prediction risk: $R(f) = \mathbb{E}(Y_0 - f(X_0))^2$

Of course, the quantity $(Y_0 - f(X_0))^2$ is an unbiased estimator of $R(f)$ and hence we could estimate $R(f)$

However, we don't have any such new observation

Or do we?

An intuitive idea

Let's set aside one observation and predict it

For example: Set aside (X_1, Y_1) and fit $\hat{f}^{(1)}$ on $(X_2, Y_2), \ldots, (X_n, Y_n)$

(The notation $\hat{f}^{(1)}$ just symbolizes leaving out the first observation before fitting $\hat{f}$)

$$R_1(\hat{f}^{(1)}) = (Y_1 - \hat{f}^{(1)}(X_1))^2$$

As the left off data point is independent of the data points used for estimation,

$$\mathbb{E}_{(X_1,Y_1)|\mathcal{D}_{(1)}} R_1(\hat{f}^{(1)}) \stackrel{D}{=} R(\hat{f}(\mathcal{D}_{n-1})) \approx R(\hat{f}(\mathcal{D}))$$

Leave-one-out cross-validation

Cycling over all observations and taking the average produces leave-one-out cross-validation

$$\mathrm{CV}_n(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} R_i(\hat{f}^{(i)}) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}^{(i)}(X_i))^2.$$

More general cross-validation schemes

Let $\mathcal{N} = \{1, \ldots, n\}$ be the index set for $\mathcal{D}$

Define a distribution $\mathcal{V}$ over $\mathcal{N}$ with (random) variable v

Then, we can form a general cross-validation estimator as

$$\mathrm{CV}_{\mathcal{V}}(\hat{f}) = \mathbb{E}_{\mathcal{V}} \hat{\mathbb{P}}_{v} \ell_{\hat{f}^{(v)}}$$

More general cross-validation schemes: Examples

$$\mathrm{CV}_{\mathcal{V}}(\hat{f}) = \mathbb{E}_{\mathcal{V}} \hat{\mathbb{P}}_v \ell_{\hat{f}^{(v)}}$$

- K-fold: Fix $V = \{v_1, \ldots, v_K\}$ such that $v_j \cap v_k = \emptyset$ and $\bigcup_j v_j = \mathcal{N}$

$$\mathrm{CV}_K(\hat{f}) = \frac{1}{K} \sum_{v \in V} \frac{1}{|v|} \sum_{i \in v} (Y_i - \hat{f}^{(v)}(X_i))^2$$

- Bootstrap: Let $\mathcal{V}$ be given by the bootstrap distribution over $\mathcal{N}$ (that is, sampling with replacement many times)
- Factorial: Let $\mathcal{V}$ be given by all subsets (or a subset of all subsets) of $\mathcal{N}$ (that is, putting mass $1/(2^n - 2)$ on each subset)

More general cross-validation schemes: A comparison

- CV_K gets more computationally demanding as $K \to n$
- The bias of CV_K goes down, but the variance increases as $K \to n$
- The factorial version isn't commonly used except when doing a 'real' data example for a methods paper
- There are many other flavors of CV. One of them, called "consistent cross validation" [Homework] is a recent addition that is designed to work with sparsifying algorithms

Summary time

Risk estimation methods

CV	Prediction risk consistent (Dudoit, van der Laan (2005)). Generally selects a model larger than necessary (unproven)
AIC	Minimax optimal risk estimator (Yang, Barron (1998)). Model selection inconsistent*
BIC	Model selection consistent (Shao (1997) [low dimensional]. Wang, Li, Leng (2009) [high dimensional]). Slow rate for risk estimation*

(Stone (1977) shows that CV_n and AIC are asymptotically equivalent.)

(*Yang (2005) gives an impossibility theorem: for a linear regression problem it is impossible for a model selection criterion to be both consistent and achieve minimax optimal risk estimation)

Summary

The overall scheme is a three(four?)-fold process

1. Select a method suited to your task
2. Choose a risk estimation method that has the properties that you desire (see next slide)
3. Perform the necessary computations to minimize 2. constrained to be in the family of procedures in 1.
4. Show theoretically that your procedure has desirable properties

Summary

The overall scheme is a three(four?)-fold process

1. Select a method suited to your task
2. Choose a risk estimation method that has the properties that you desire (see next slide)
3. Perform the necessary computations to minimize 2. constrained to be in the family of procedures in 1.
4. Show theoretically that your procedure has desirable properties

Brief optimization and convexity detour

Optimization

An optimization problem (program) can be generally formulated as

$$\text{minimize } F(x) \tag{3}$$
$$\text{subject to } f_j(x) \leq 0 \text{ for } j = 1, \ldots, m \tag{4}$$
$$h_k(x) = 0 \text{ for } k = 1, \ldots, q \tag{5}$$

Here

- $x = (x_1, \ldots, x_n)^\top$ are the parameters
- $F : \mathbb{R}^n \to \mathbb{R}$ is the objective function
- $f_j, h_k : \mathbb{R}^n \to \mathbb{R}$ are constraint functions

The optimal solution x^* is such that $F(x^*) \leq F(x)$ for any x^*, x that satisfies equations (4) and (5).

Convexity

The main dichotomy of optimization programs is convex vs. nonconvex

Generally speaking, a convex program is one in which the objective and contraint functions are all convex, that is
$\forall t \in [0,1], \forall x \in D = \left(\bigcap_{j=1}^{m} \text{dom } f_i\right) \cap \left(\bigcap_{k=1}^{q} \text{dom } h_k\right) \cap (\text{dom F})$,
and $\forall f \in \{f_1, \ldots, f_m, h_1, \ldots, h_q, F\}$

$$f(tx + (1-t)x') \leq tf(x) + (1-t)f(x')$$

This can be thought of (for smooth enough f)

$$f(x') \geq f(x) + (\nabla f|_x)^\top (x' - x)$$

Intuition: This means that the function values at a point x' are above the supporting hyperplane given by the tangent space at any point x

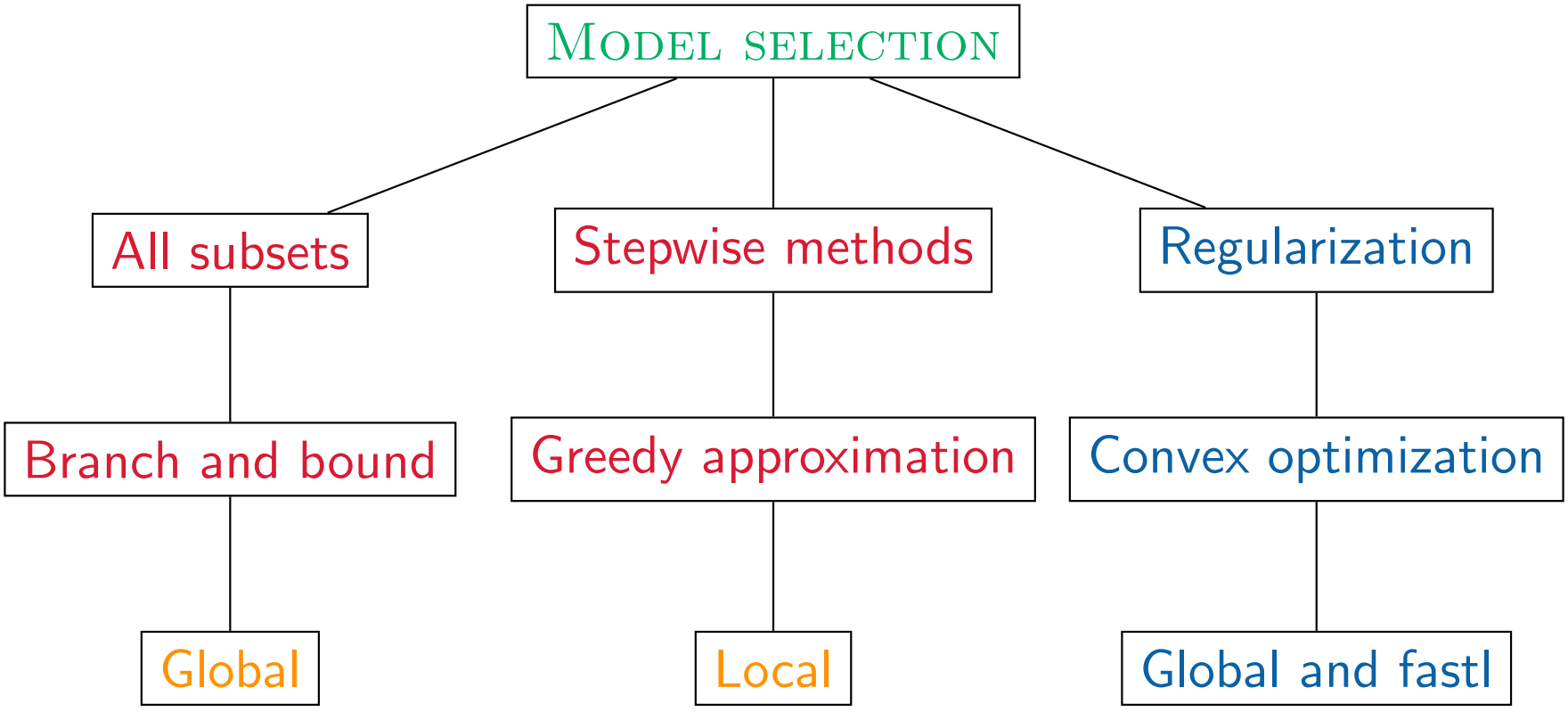
Convexity

Methods for convex optimization programs are (roughly) always global and fast

For general nonconvex problems, we have to give up one of these:

- Local optimization methods that are fast, but need not find global solution
- Global optimization methods that find global solutions, but are not always fast (indeed, are often slow)

Model selection



Some comments:

- Non convex programs
- Can be seen as a convex relaxation of the nonconvex program giving all subsets[8]

[8] We'll return to this shortly

All Subsets Regression: A Big Problem (Literally)

If there are p predictors then there are $2^p - 1$ possible models

(Without considering interactions or transformations)

In general, this is a nonconvex problem

If $p = 40$ (which is considered a small problem these days), then the number of possible models is

$$2^{40} - 1 \approx 1,099,512,000,000 \Rightarrow \text{More than 1 trillion!}$$

If $p = 265$, then the number of possible models is more than the number of atoms in the universe[9]

We must sift through the models in a computationally feasible way

[9] It is estimated there are 10^{80} atoms in the universe.

All Subsets Regression

This can efficiently be computed via the leaps package in R, using either the leaps or regsubsets functions.

This is a specific case of branch and bound

(The statistical implementation is based on the paper Furnival and Wilson (1974))

It is by far the most widely used tool for solving large scale NP-hard combinatorial optimization problems.

Note, however, that though it can speed up the optimization immensely, it cannot reduce the complexity of the problem (still exponential)

Branch and bound

Let $M = \{M_1, \ldots, M_K\}$ be the set of all possible solutions and a partition comprised of branches, respectively.

(Statistically, we think of M as the set of all possible models.)

Let F be the objective function and $m_* = \max_{m \in M} F(m)$.

For each M_k, define

$$m_k = \max_{m \in M_k} F(m)$$

and let $\underline{m}_k, \overline{m}_k$ be a bracket such that

$$\underline{m}_k \leq m_k \leq \overline{m}_k$$

(Note that m_k is in general not explicitly constructed)

Then

$$\max_k \underline{m}_k = \underline{m} \leq m_* \leq \overline{m} = \max_k \overline{m}_k$$

Branch and bound

The main realization is that the branch M_k does not need to be explored if either of the following occur

i. Bound

$$\overline{m}_k \leq \underline{m}$$

ii. Optimality

$$\max_{m \in M_k} F(m) \text{ has been found}$$

The two main questions remain:

1. How to choose the partition(s)?
2. How to form the upper/lower bounds?

These are very case specific. Let's return to model selection

Branch and bound for model selection

Let's suppose we set[10]

$$F(m) = n \log(\hat{R}_{\text{train}}(\hat{\beta}_m)) + 2|m|$$

For the M_k, let

$m_{k,inf}$ be the largest model contained[11] in every model in M_k

$m_{k,sup}$ be a smallest model that contains every model in M_k

[10] Note: we are trying to minimize F, not maximize

[11] This does not have to be in M_k

Branch and bound for model selection

Example: Let $x_1, \ldots, x_5$ be covariates

$$M = \cup_{k=1}^{3} M_k,$$

where

$$\begin{aligned} M_1 &= \{\{x_1, x_3\}, \{x_2\}\}, \\ M_2 &= \{\{x_2, x_3, x_4\}, \{x_3, x_4\}\}, \\ M_3 &= \{\{x_3, x_5\}, \{x_3\}\}, \end{aligned}$$

Branch and bound for model selection

Example: Let $x_1, \ldots, x_5$ be covariates

$$M = \cup_{k=1}^{3} M_k,$$

where

$$\begin{aligned} M_1 &= \{\{x_1, x_3\}, \{x_2\}\}, \\ M_2 &= \{\{x_2, x_3, x_4\}, \{x_3, x_4\}\}, \\ M_3 &= \{\{x_3, x_5\}, \{x_3\}\}, \end{aligned}$$

$$\begin{aligned} m_{2,inf} &= \{x_3, x_4\} \\ m_{2,sup} &= \{x_2, x_3, x_4\} \end{aligned}$$

Branch and bound for model selection

Reminder:

For the M_k, let

$m_{k,inf}$ be the largest model contained in every model in M_k

$m_{k,sup}$ be a smallest model that contains every model in M_k

Then, $\forall m \in M_k$

$$F(m) \geq n \log(\hat{R}_{\text{train}}(\hat{\beta}_{m_{k,\text{sup}}})) + 2|m_{k,\text{inf}}| = L_k$$

$$F(m) \leq n \log(\hat{R}_{\text{train}}(\hat{\beta}_{m_{k,\text{inf}}})) + 2|m_{k,\text{sup}}| = U_k$$

Branch and bound for model selection: An algorithm

1. Define a global variable $b = F(m)$ for any $m \in M$
   (As an aside, every time $F(m)$ is computed, update b if $F(m) < b$)
2. Partition $M = \{M_1, \ldots, M_K\}^\sharp$
3. For each k, if $L_k > b$, eliminate the branch M_k
4. Else, recurse and return to 2., substituting M_k for M

(Forward) stepwise selection

In the likely event that $|M|$ is too large to be searched over exhaustively, a common greedy approximation is the following

1. Find $b = F(\emptyset)$, where $\emptyset$ is the empty set
2. Search over all p singleton sets and record $m_{1,\text{min}} = \text{argmin}\, F(m)$. If $F(m_{1,\text{min}}) < b$ set $b \leftarrow F(m_{1,\text{min}})$, else return $\emptyset$
3. Now search over all $p-1$ models that contain $m_{1,\text{min}}$ and form $m_{2,\text{min}} = \text{argmin}\, F(m_{1,\text{min}} \cup \{x_j\})$. If $F(m_{2,\text{min}}) < b$ set $b \leftarrow F(m_{2,\text{min}})$, else return $m_{1,\text{min}}$
4. $\ldots$

General stepwise selection

This algorithm can can adapted to..

- start with the full model and stepwise remove covariates
  (useful if the full model isn't too large and a superset of the important covariates is desired)
- consider both adding and removing covariates at each step

This can efficiently be computed via either regsubsets or step in R
(See website for example code for doing model selection in R)

Regularization

Regularization

Another way to control bias and variance is through regularization or shrinkage.

The idea is to make your estimates of β 'smaller', rather than set them to zero

(which is what all subsets does)

One way to do this is called ridge regression[12]:

$$\hat{\beta}_{\text{ridge},t} = \underset{||\tilde{\beta}||_2^2 \leq t}{\text{argmin}} \, ||Y - \mathbb{X}\tilde{\beta}||_2^2$$

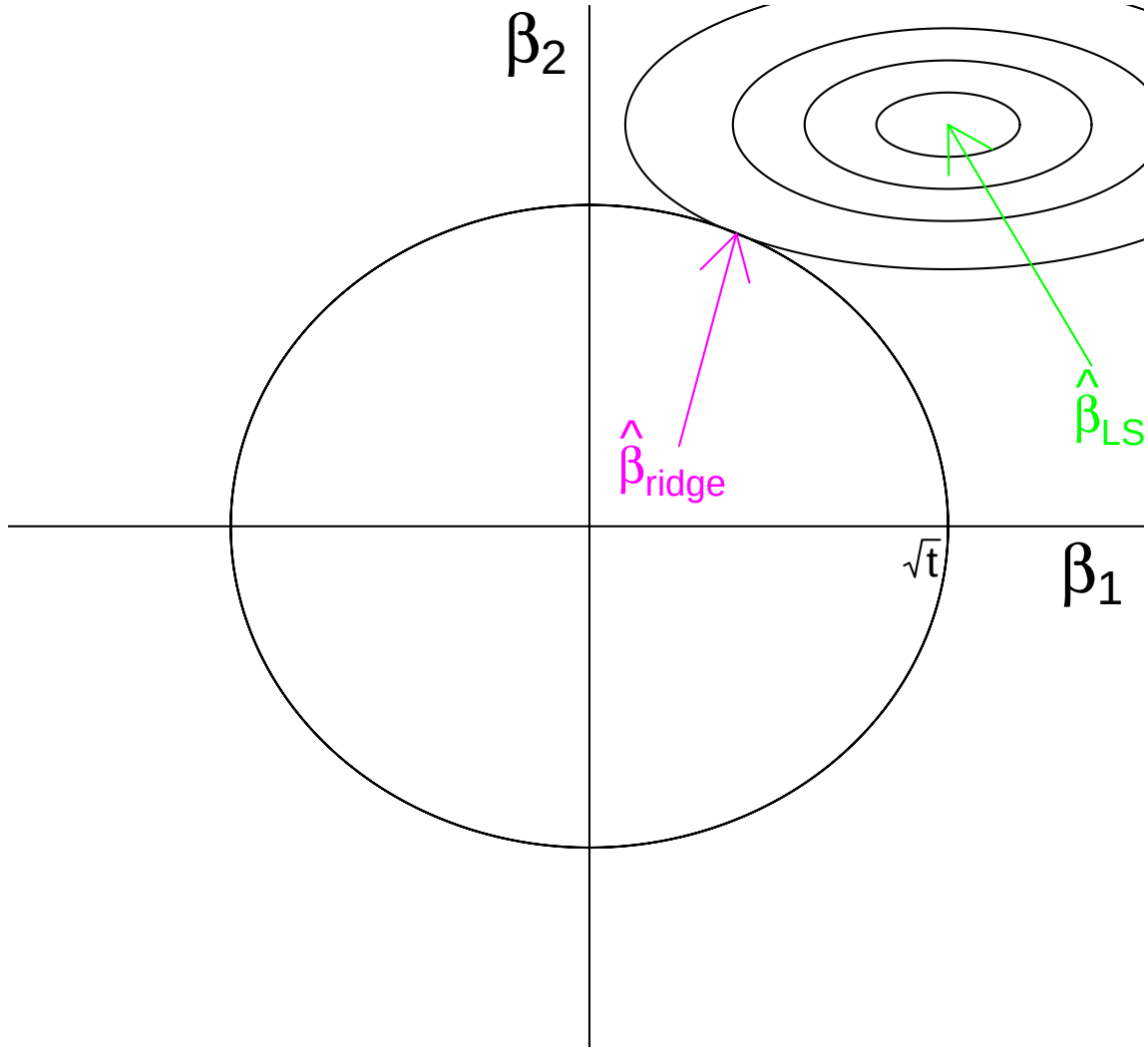
for any $t \geq 0$.

Compare this to least squares

$$\hat{\beta}_{LS} = \underset{\tilde{\beta} \in \mathbb{R}^p}{\text{argmin}} \, ||Y - \mathbb{X}\tilde{\beta}||_2^2$$

[12] Hoerl, Kennard (1970)

Geometry of ridge regression in $\mathbb{R}^2$



Ridge regression

An equivalent way to write

$$\hat{\beta}_{\text{ridge},t} = \underset{||\beta||_2^2 \leq t}{\text{argmin}} \, ||Y - \mathbb{X}\beta||_2^2 \tag{6}$$

is in the Lagrangian form[♯]

$$\hat{\beta}_{\text{ridge},\lambda} = \underset{\beta}{\text{argmin}} \, ||Y - \mathbb{X}\beta||_2^2 + \lambda||\beta||_2^2. \tag{7}$$

For every λ' there is a unique t' (and vice versa) that makes

$$\hat{\beta}_{\text{ridge},\lambda'} = \hat{\beta}_{\text{ridge},t'}$$

Ridge regression

An equivalent way to write

$$\hat{\beta}_{\text{ridge},t} = \underset{||\beta||_2^2 \leq t}{\text{argmin}} \, ||Y - \mathbb{X}\beta||_2^2 \tag{6}$$

is in the Lagrangian form$^\sharp$

$$\hat{\beta}_{\text{ridge},\lambda} = \underset{\beta}{\text{argmin}} \, ||Y - \mathbb{X}\beta||_2^2 + \lambda||\beta||_2^2. \tag{7}$$

For every λ' there is a unique t' (and vice versa) that makes

$$\hat{\beta}_{\text{ridge},\lambda'} = \hat{\beta}_{\text{ridge},t'}$$

Important:

- As the constraint set is a sphere, each direction is treated equally. You should standardize your coefficient before fitting.
- Likewise, don't penalize the intercept. If the sample means of the covariates are zero, then make the response have mean zero as well (and don't include intercept)

Ridge regression

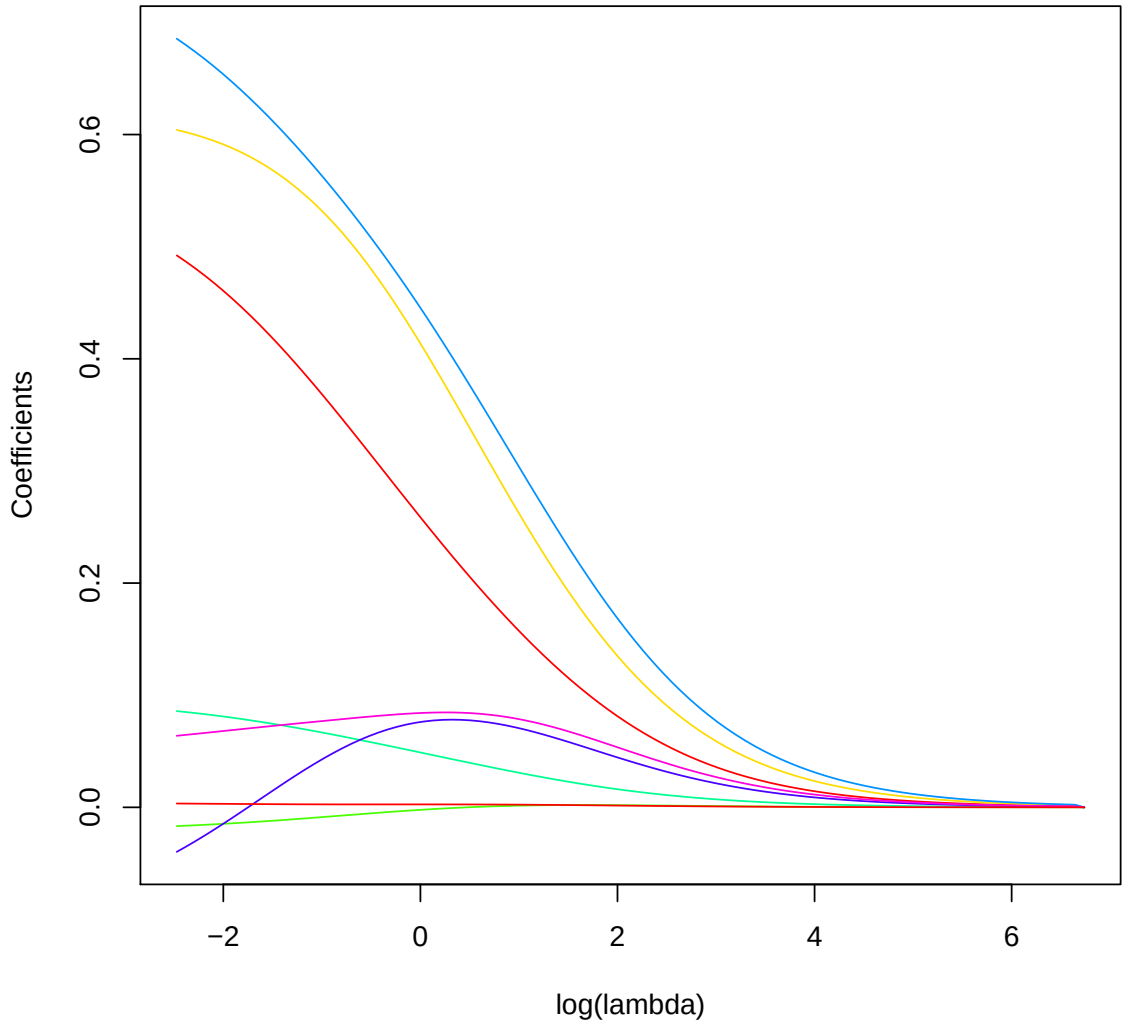
Observe:

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

Note: λ and t are known as tuning parameters

(Alternatively, hyper-parameters)

Ridge regression path



Ridge regression

Reminder: The least squares solution can be written:

$$\hat{\beta}_{\text{LS}} = (\mathbb{X}^\top \mathbb{X})^\dagger \mathbb{X}^\top Y.$$

However, if $\text{rank}(\mathbb{X}) < p$, then $\hat{\beta}_{\text{LS}}$ is not unique. In fact,

$$\forall b \in \{b : \mathbb{X}b = 0\}$$

$\hat{\beta}_{\text{LS}} + b$ is a valid least squares solution.

It turns out through differential calculus, we can write out the ridge regression solution as well:

$$\hat{\beta}_{\text{ridge},\lambda} = (\mathbb{X}^\top \mathbb{X} + \lambda I)^{-1} \mathbb{X}^\top Y$$

Quite similar. However, the λ can make all the difference..

Regularization - Ridge Regression

Using the SVD ($\mathbb{X} = UDV^\top$), we can look even deeper.

$$\hat{\beta}_{\text{LS}} = VD^{-1}U^\top Y \qquad = \sum_{j=1}^{p} \mathbf{v}_j \left(\frac{1}{d_j}\right) \mathbf{u}_j^\top Y$$

$$\hat{\beta}_{\text{ridge},\lambda} = V(D^2 + \lambda I)^{-1} DU^\top Y \qquad = \sum_{j=1}^{p} \mathbf{v}_j \left(\frac{d_j}{d_j^2 + \lambda}\right) \mathbf{u}_j^\top Y.$$

Similarly

$$\mathbb{X}\hat{\beta}_{\text{LS}} = UU^\top Y \qquad = \sum_{j=1}^{p} \mathbf{u}_j \left(\frac{1}{d_j}\right) \mathbf{u}_j^\top Y$$

$$\mathbb{X}\hat{\beta}_{\text{ridge},\lambda} = UD(D^2 + \lambda I)^{-1} DU^\top Y \qquad = \sum_{j=1}^{p} \mathbf{u}_j \left(\frac{d_j^2}{d_j^2 + \lambda}\right) \mathbf{u}_j^\top Y.$$

$\Rightarrow$ **Ridge shrinks the data by an additional factor of λ.**

Ridge Regression: A Bayesian approach

Suppose we specify the likelihood as

$$Y_i \sim N(X_i^\top \beta, \sigma^2)$$

and put a prior distribution of $\beta \sim N(0, \tau^2 I)$.

Then we have the following posterior (making some conditional independence assumptions)

$$p(\beta|Y, X, \sigma^2, \tau^2) \propto p(Y|X, \beta, \sigma^2)p(\beta|\tau^2).$$

After kernel matching, we find that the posterior mode/mean is

$$\hat{\beta}_{\text{ridge},\lambda=\sigma^2/\tau^2}$$

Ridge regression in a new space

Note the matrix identity

$$(A - BC^{-1}E)^{-1}BC^{-1} = A^{-1}B(C - EA^{-1}B)^{-1}$$

(Henderson, Searle (1980), equation (13))
Then,

$$\hat{\beta}_{\text{ridge},\lambda} = (\mathbb{X}^\top \mathbb{X} + \lambda I)^{-1}\mathbb{X}^\top Y = \mathbb{X}^\top(\mathbb{X}\mathbb{X}^\top + \lambda I)^{-1}Y$$

Now, the inversion is in n-space instead of p, which could be a substantial savings

However, a much deeper realization is possible..

(Kernel) ridge regression

Suppose we want to predict at x, then

$$\hat{f}(x) = x^\top \hat{\beta}_{\text{ridge},\lambda} = x^\top \mathbb{X}^\top (\mathbb{X}\mathbb{X}^\top + \lambda I)^{-1} Y$$

Also,

$$\mathbb{X}\mathbb{X}^\top = \begin{bmatrix} \langle X_1, X_1 \rangle & \langle X_1, X_2 \rangle & \cdots & \langle X_1, X_n \rangle \\ & \vdots & & \\ \langle X_n, X_1 \rangle & \langle X_n, X_2 \rangle & \cdots & \langle X_n, X_n \rangle \end{bmatrix}$$

and

$$x^\top \mathbb{X}^\top = [\langle x, X_1 \rangle, \langle x, X_2 \rangle, \cdots, \langle x, X_n \rangle]$$

where $\langle x, x' \rangle$ is the Euclidean inner product.

If we transform $X_i \mapsto \Phi(X_i)$, and the range of Φ is equipped with an inner product, we can use $\langle \Phi(X_i), \Phi(X_{i'}) \rangle$ instead[13].

[13] We'll return to this shortly

Ridge in practice

Ridge Regression: picking the tuning parameter

We can use a degrees of freedom based risk estimator to choose λ

The degrees of freedom of $\hat{\beta}_{\text{ridge},\lambda}$ can be seen to be[♯]

$$\text{df}(\hat{\beta}_{\text{ridge},\lambda}) = \text{trace}\mathbb{X}(\mathbb{X}^\top\mathbb{X} + \lambda I)^{-1}\mathbb{X}^\top = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$$

(As $\lambda \to 0$, we get the number of parameters)

A common, classic choice is generalized cross-validation (GCV), which has the form:

$$\text{GCV}(\hat{\beta}) = \frac{\hat{\mathbb{P}}\ell_{\hat{\beta}}}{(1 - \text{df}(\hat{\beta})/n)^2}$$

(Golub, Heath, Wahba (1979))

Note that this looks a lot like AIC with unknown variance, but with $\log(1 - \text{df}/n)$ as penalty

Ridge Regression: picking the tuning parameter

Nowadays, using K-fold cross-validation is common

Think of CV_K as a function of λ, and pick its minimum:

$$\hat{\lambda} = \underset{\lambda \geq 0}{\text{argmin}}\, CV_K(\lambda)$$

Now, we report $\hat{\beta}_{\text{ridge},\hat{\lambda}}$ as our estimator

Ridge Regression: Computation

There are several ways to compute ridge regression

We can follow any conventional least squares solving technique (i.e.: QR factorization, Cholesky Decomposition, SVD,...):

$$(\mathbb{X}^\top \mathbb{X} + \lambda I)\beta = \mathbb{X}^\top Y$$

Alternatively, we can actually solve it using lm in R if we make the following augmentation

$$\tilde{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{n+p} \text{ and } \tilde{\mathbb{X}} = \begin{bmatrix} \mathbb{X} \\ \sqrt{\lambda} I \end{bmatrix}$$

Ridge Regression in R

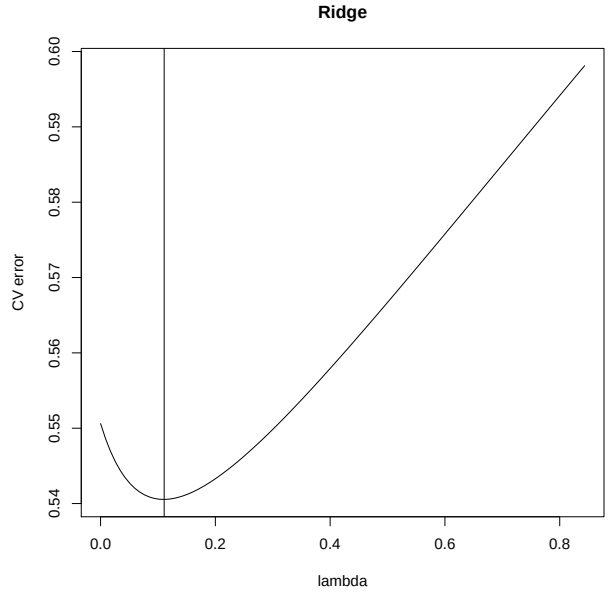
We will concentrate on a slightly more complicated way, as it will make things easier later.

```
install.packages('glmnet')
library(glmnet)
ridge.out = cv.glmnet(x=X,y=Y,alpha=0)
```

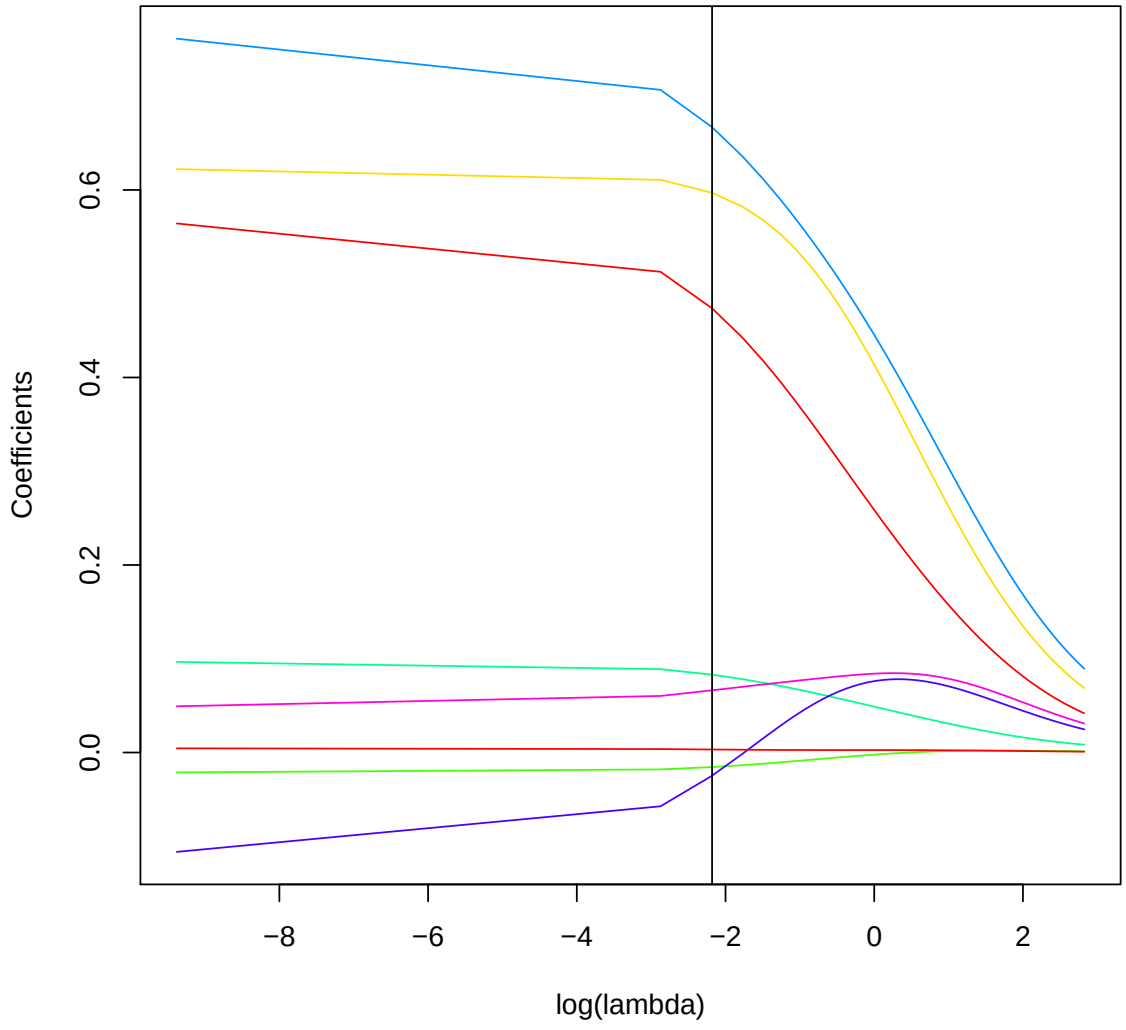
Ridge Regression: CV plot

```
X = as.matrix(X)
ridge.out = cv.glmnet(x=X,y=Y,alpha=0)

plot(ridge.out$lambda,ridge.out$cvm,
       xlab='lambda',ylab='CV error',main='Ridge',type='l')
abline(v=ridge.out$lambda[which.min(ridge.out$cvm)])
```



Ridge regression path



Can we get the best of both worlds?

To recap:

- Forward, backward, and all subsets regression offer good tools for model selection.

  (but the optimization problem is nonconvex)

- Ridge regression provides regularization, which trades off bias and variance and also stabilizes multicollinearity.

  (problem is convex, but doesn't do model selection)

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 $\|\|\beta\|\|_0 \leq t$
	($\|\|\beta\|\|_0 =$ the number of nonzero elements in β)

An intuitive idea

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

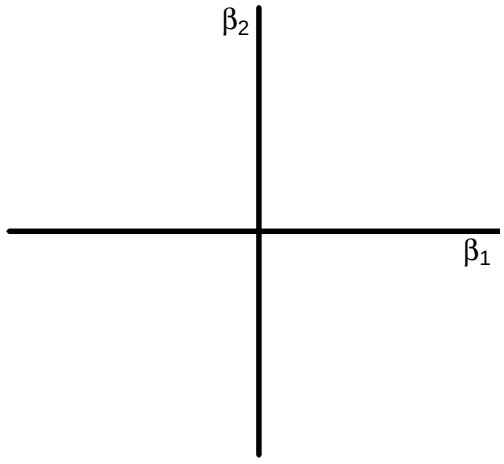
Best linear regression model $\min ||\mathbb{Y} - \mathbb{X}\beta||_2^2$ subject to $||\beta||_0 \leq t$

($||\beta||_0 =$ the number of nonzero elements in β)

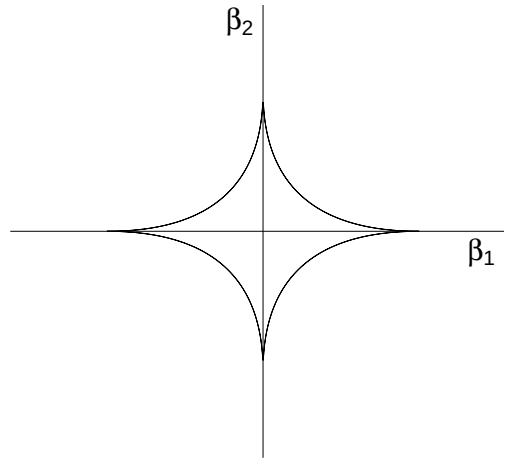
	Best linear regression model	Ridge regression
Computationally Feasible?	No	Yes
Does Model Selection?	Yes	No

Can we 'interpolate' $||\beta||_2$ and $||\beta||_0$ to find a method that does both?

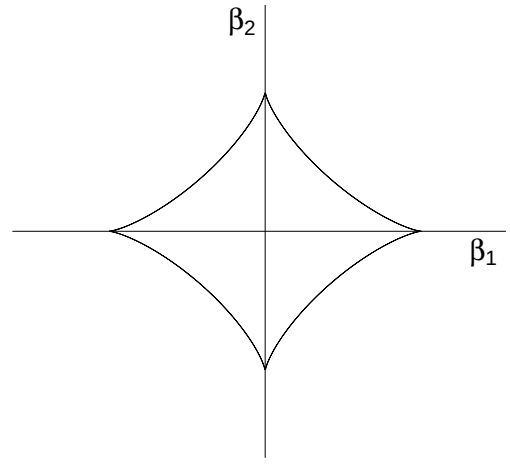
Geometry of regularization in $\mathbb{R}^2$: Convexity



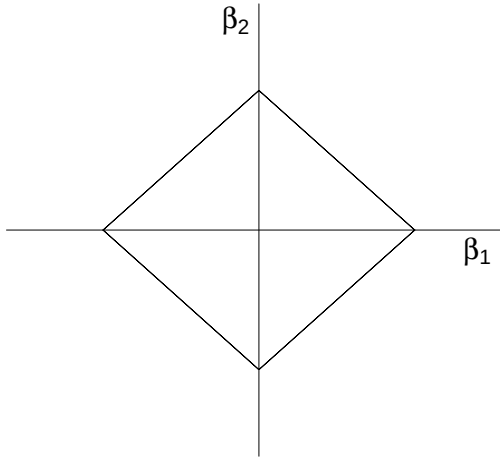
$||\beta||_0 \leq t$



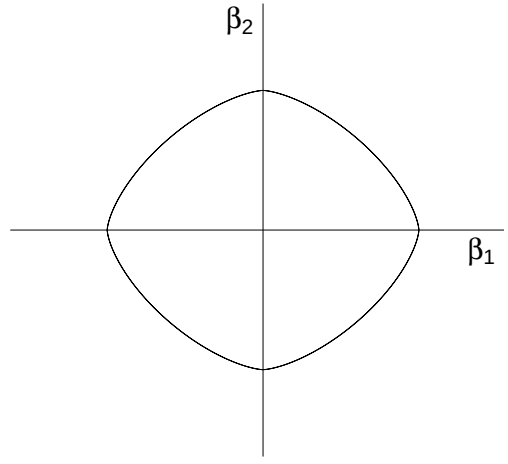
$||\beta||_{\frac{1}{2}} \leq t$



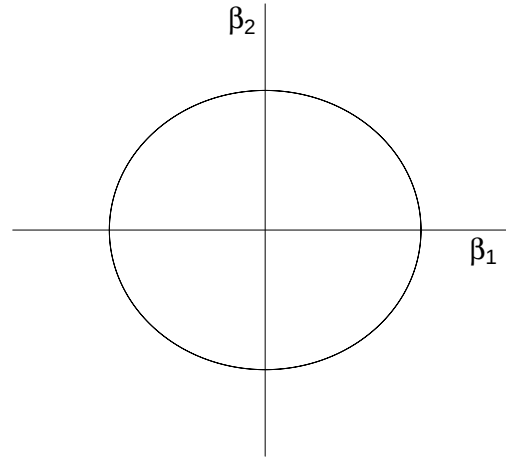
$||\beta||_{\frac{3}{4}} \leq t$



$||\beta||_1 \leq t$

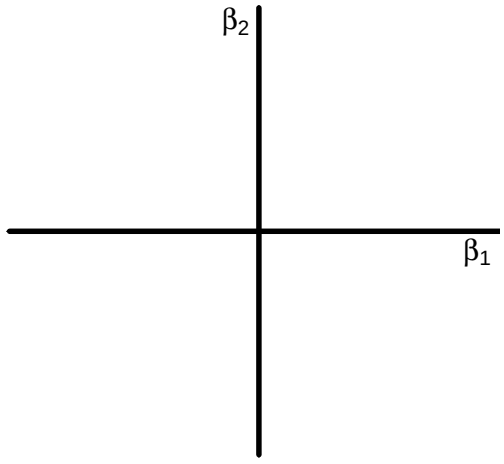


$||\beta||_{\frac{3}{2}} \leq t$

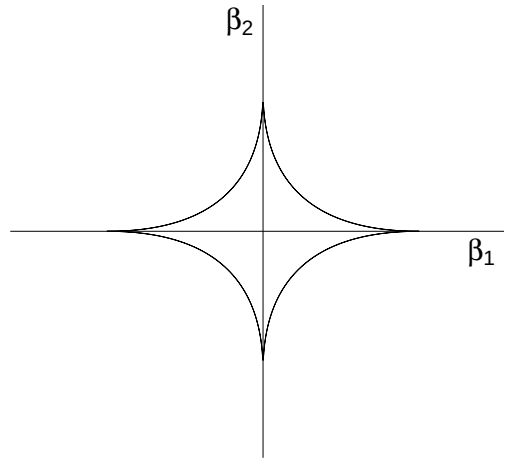


$||\beta||_2 \leq t$

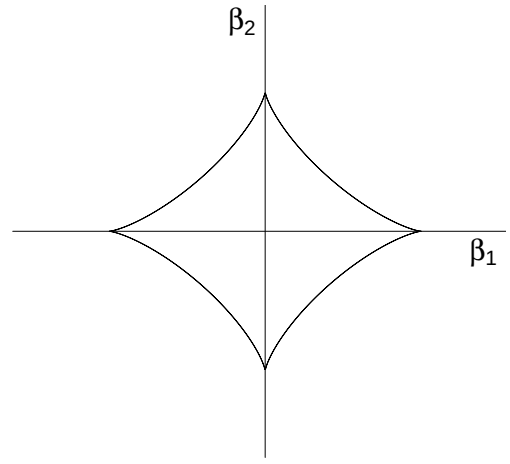
Geometry of regularization in $\mathbb{R}^2$: Convexity



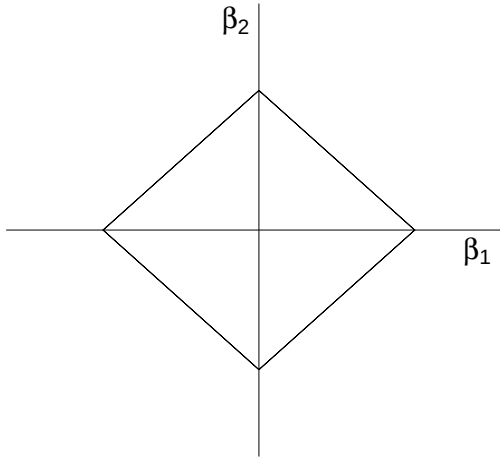
$||\beta||_0 \leq t$



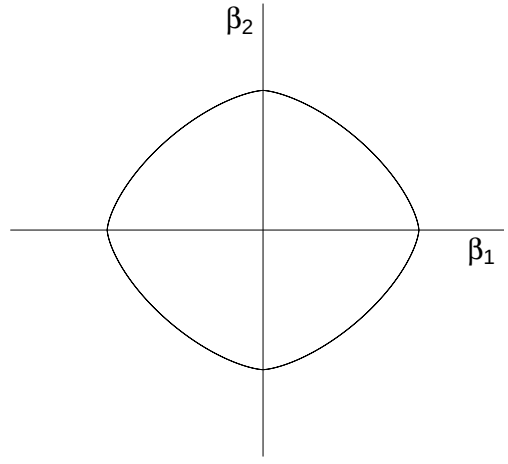
$||\beta||_{\frac{1}{2}} \leq t$



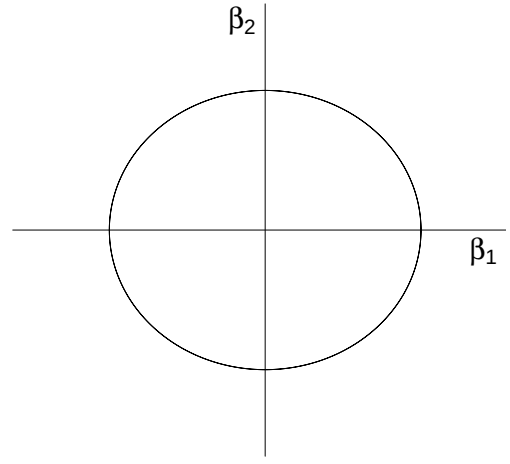
$||\beta||_{\frac{3}{4}} \leq t$



$||\beta||_1 \leq t$

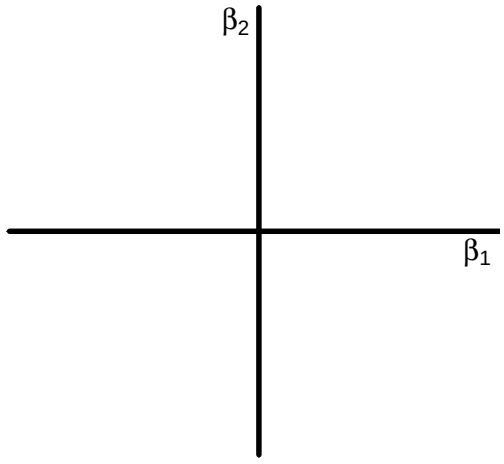


$||\beta||_{\frac{3}{2}} \leq t$

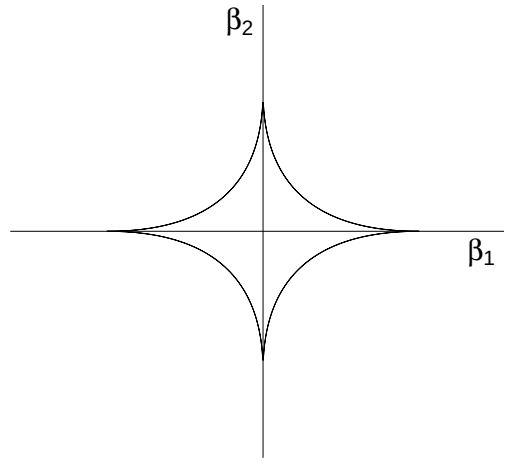


$||\beta||_2 \leq t$

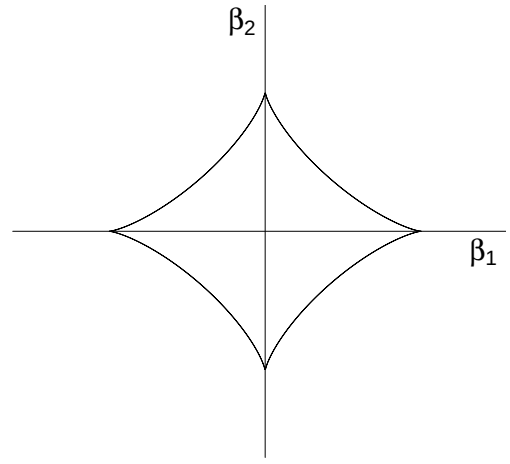
Geometry of regularization in $\mathbb{R}^2$: Convexity



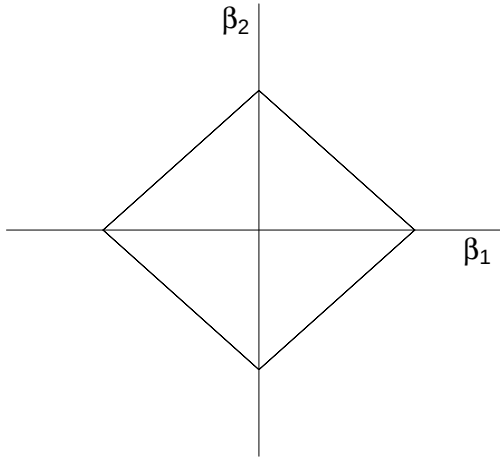
$||\beta||_0 \leq t$



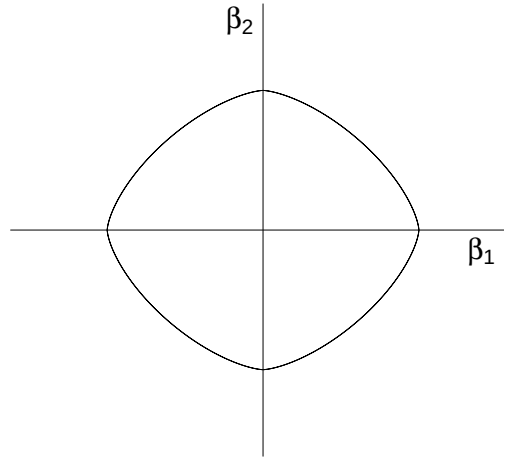
$||\beta||_{\frac{1}{2}} \leq t$



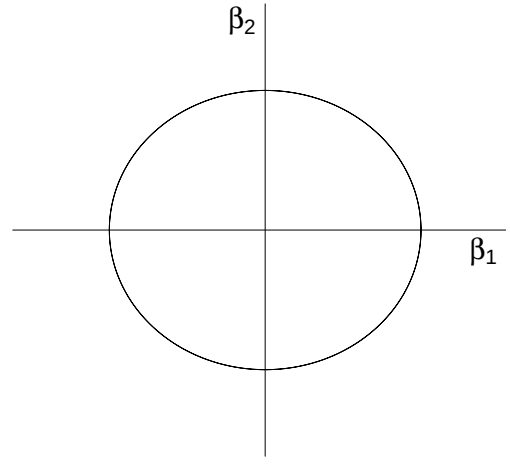
$||\beta||_{\frac{3}{4}} \leq t$



$||\beta||_1 \leq t$

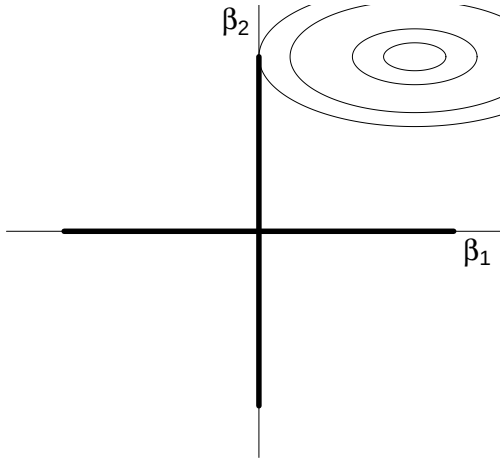


$||\beta||_{\frac{3}{2}} \leq t$

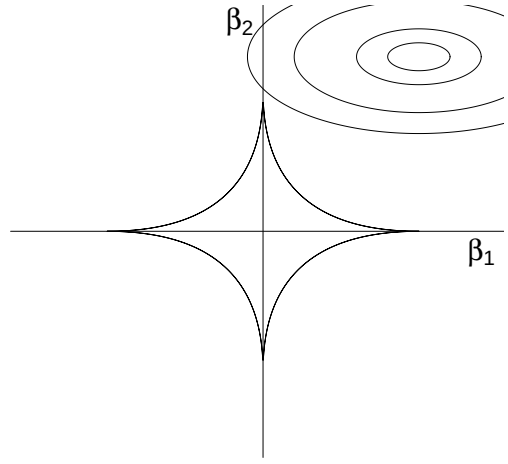


$||\beta||_2 \leq t$

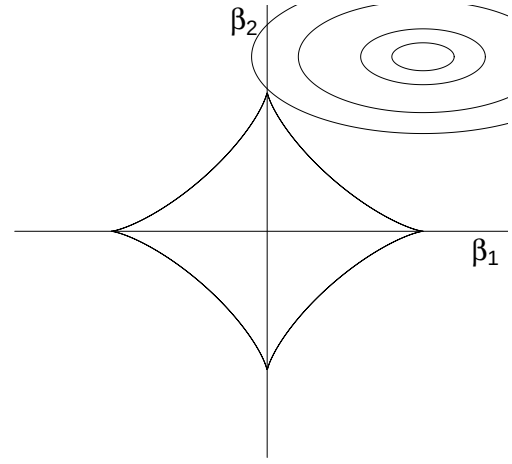
Geometry of regularization in $\mathbb{R}^2$: Model selection



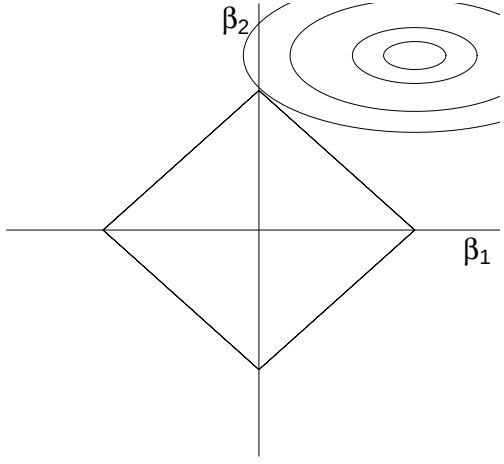
$||\beta||_0 \leq t$



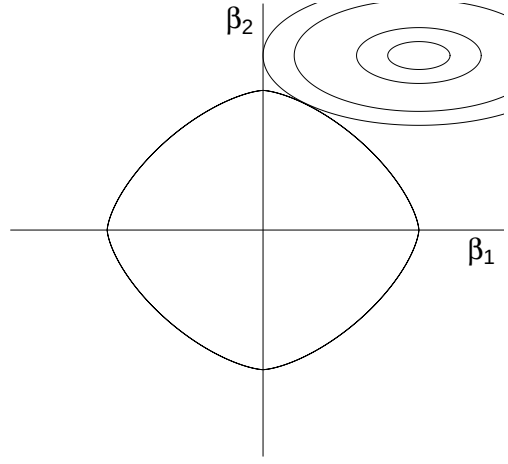
$||\beta||_{\frac{1}{2}} \leq t$



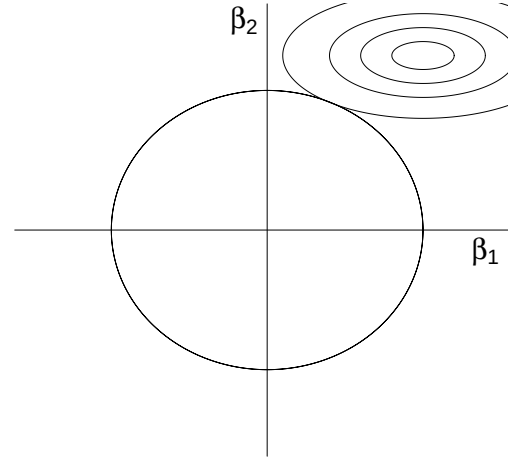
$||\beta||_{\frac{3}{4}} \leq t$



$||\beta||_1 \leq t$

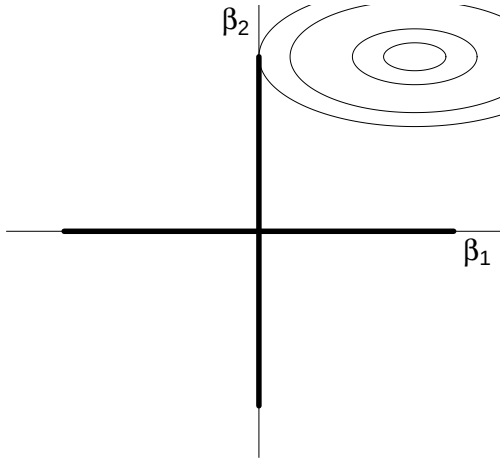


$||\beta||_{\frac{3}{2}} \leq t$

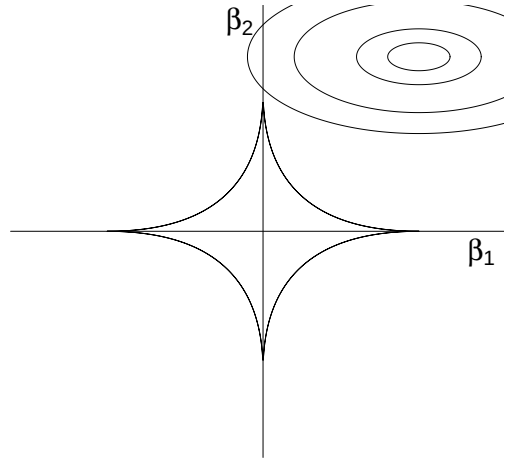


$||\beta||_2 \leq t$

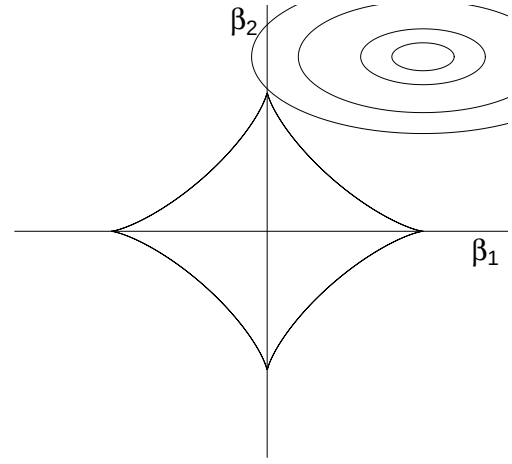
Geometry of regularization in $\mathbb{R}^2$: Model selection



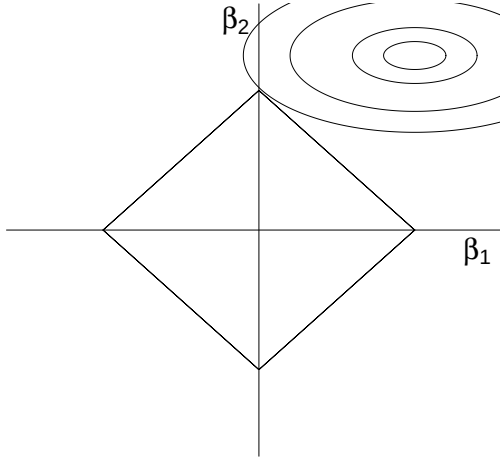
$||\beta||_0 \leq t$



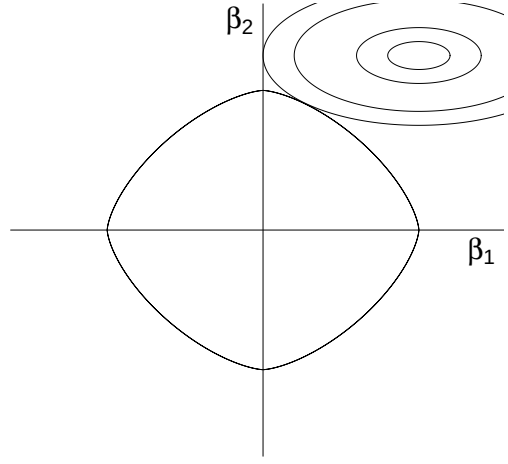
$||\beta||_{\frac{1}{2}} \leq t$



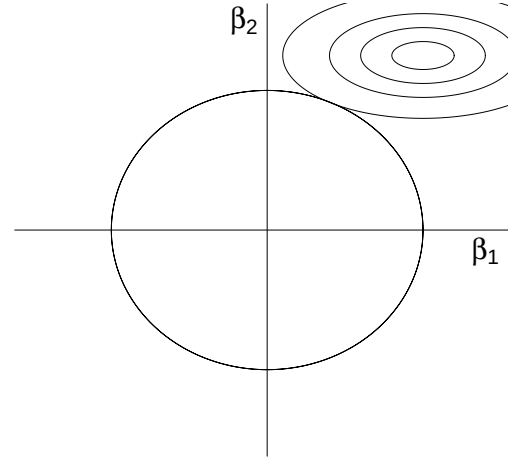
$||\beta||_{\frac{3}{4}} \leq t$



$||\beta||_1 \leq t$

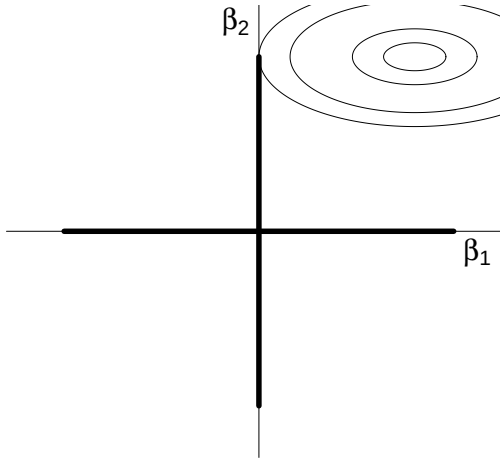


$||\beta||_{\frac{3}{2}} \leq t$

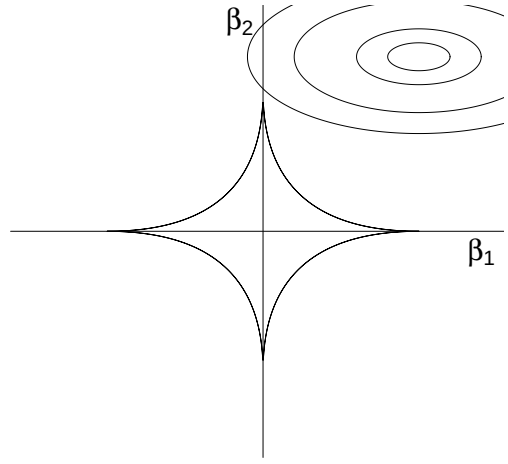


$||\beta||_2 \leq t$

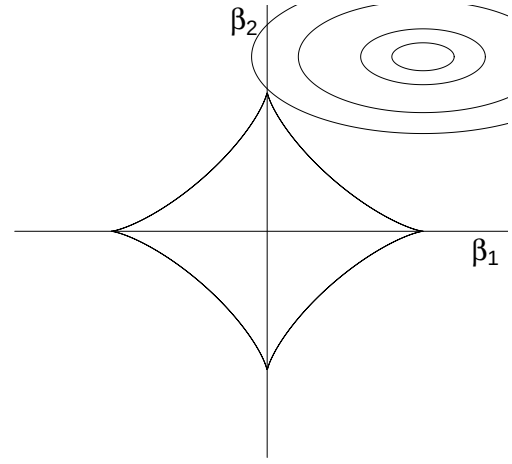
Geometry of regularization in $\mathbb{R}^2$: Model selection



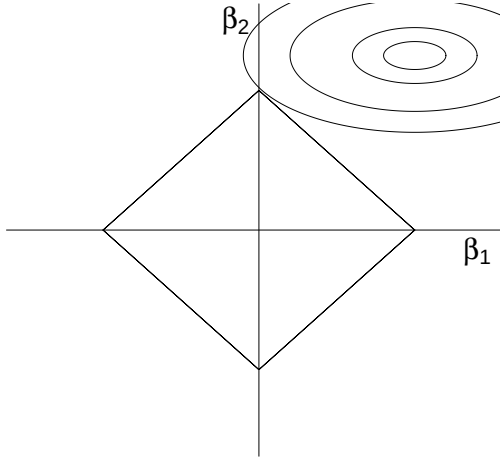
$||\beta||_0 \leq t$



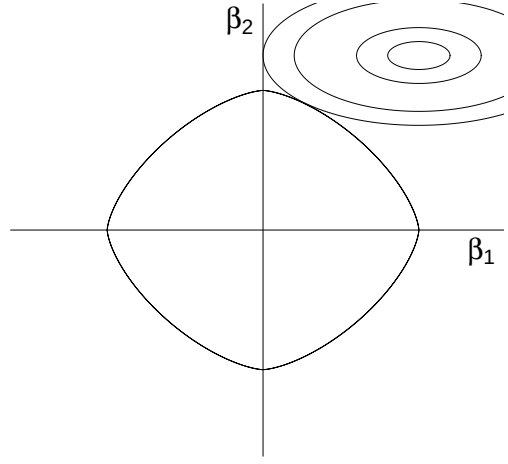
$||\beta||_{\frac{1}{2}} \leq t$



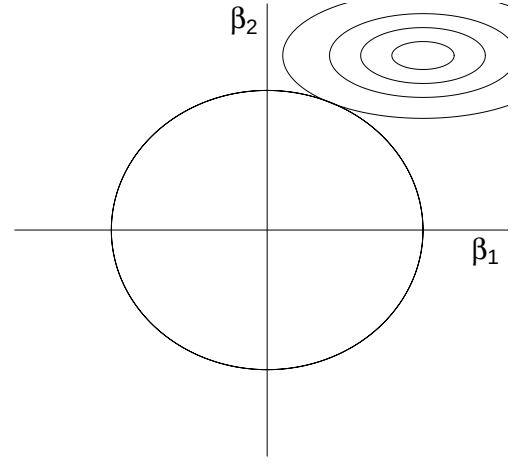
$||\beta||_{\frac{3}{4}} \leq t$



$||\beta||_1 \leq t$

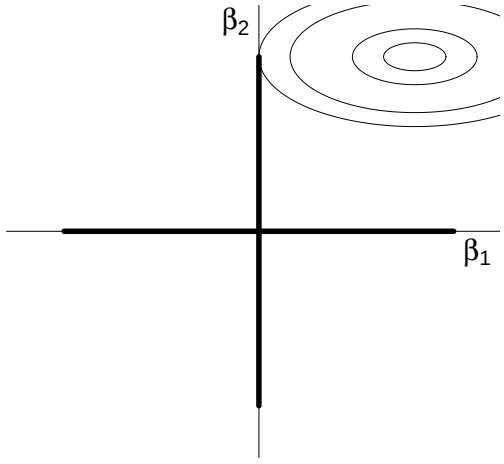


$||\beta||_{\frac{3}{2}} \leq t$

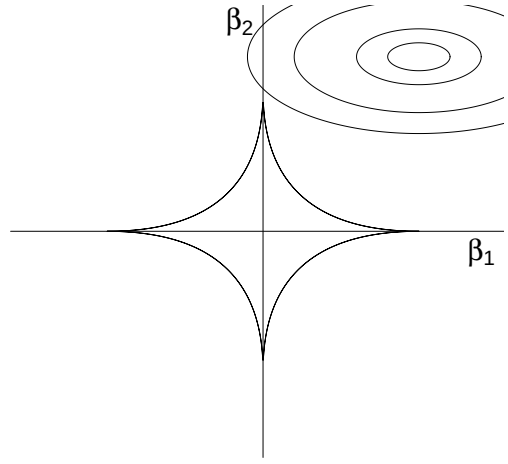


$||\beta||_2 \leq t$

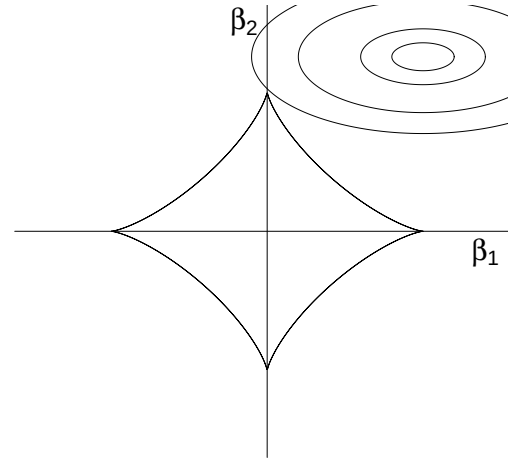
Geometry of regularization in $\mathbb{R}^2$: Both



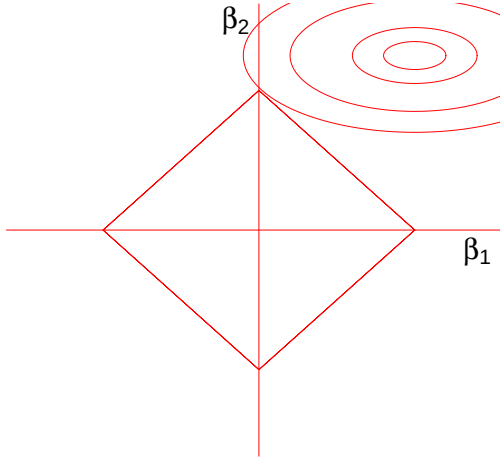
$||\beta||_0 \leq t$



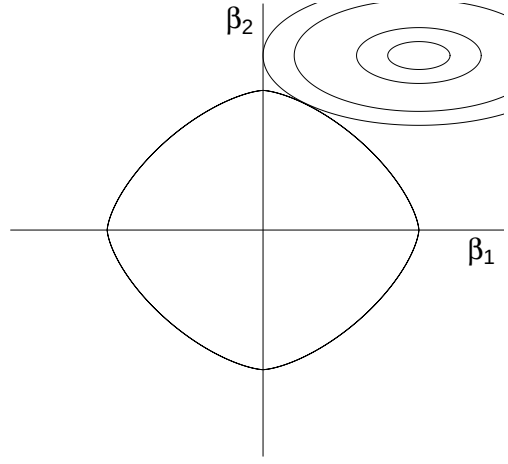
$||\beta||_{\frac{1}{2}} \leq t$



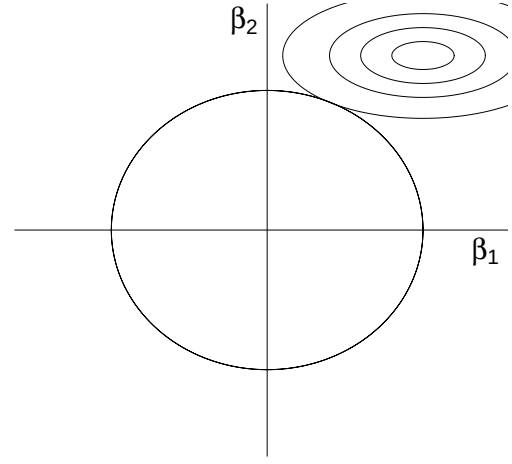
$||\beta||_{\frac{3}{4}} \leq t$



$||\beta||_1 \leq t$



$||\beta||_{\frac{3}{2}} \leq t$

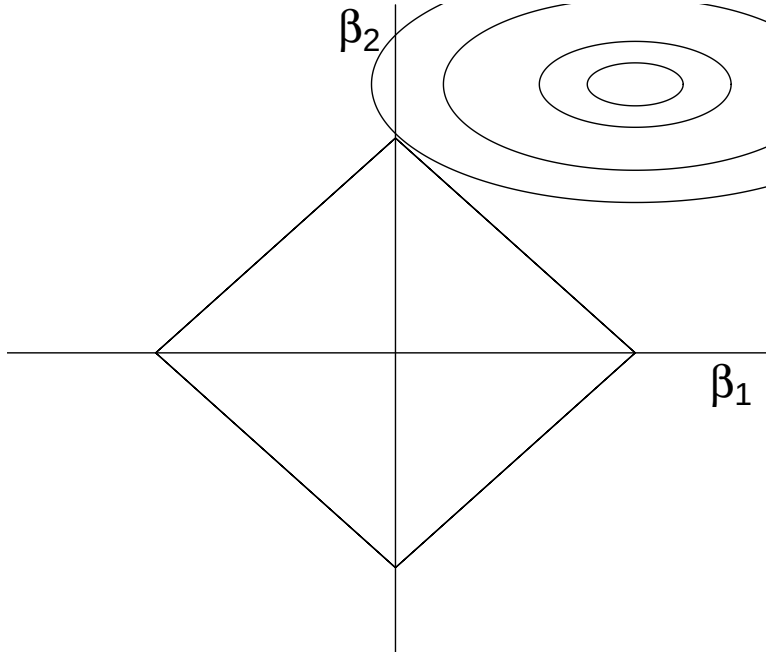


$||\beta||_2 \leq t$

Summary

	Convex?	Corners?	
$\|\|\beta\|\|_0$	No	Yes	
$\|\|\beta\|\|_{\frac{1}{2}}$	No	Yes	
$\|\|\beta\|\|_{\frac{3}{4}}$	No	Yes	
$\|\|\beta\|\|_1$	Yes	Yes	✓
$\|\|\beta\|\|_{\frac{3}{2}}$	Yes	No	
$\|\|\beta\|\|_2$	Yes	No	

The best of both worlds: $||\beta||_1$



This regularization set...

... is convex (computationally efficient)

... has corners (performs model selection)

Lasso in practice

ℓ_1-REGULARIZED REGRESSION

Related methods are known as

- LASSO: The covariates are recorded
- BASIS PURSUIT: The covariates are frames comprised of various bases
- COMPRESSED SENSING: The covariates are random draws from some distribution

The estimator satisfies

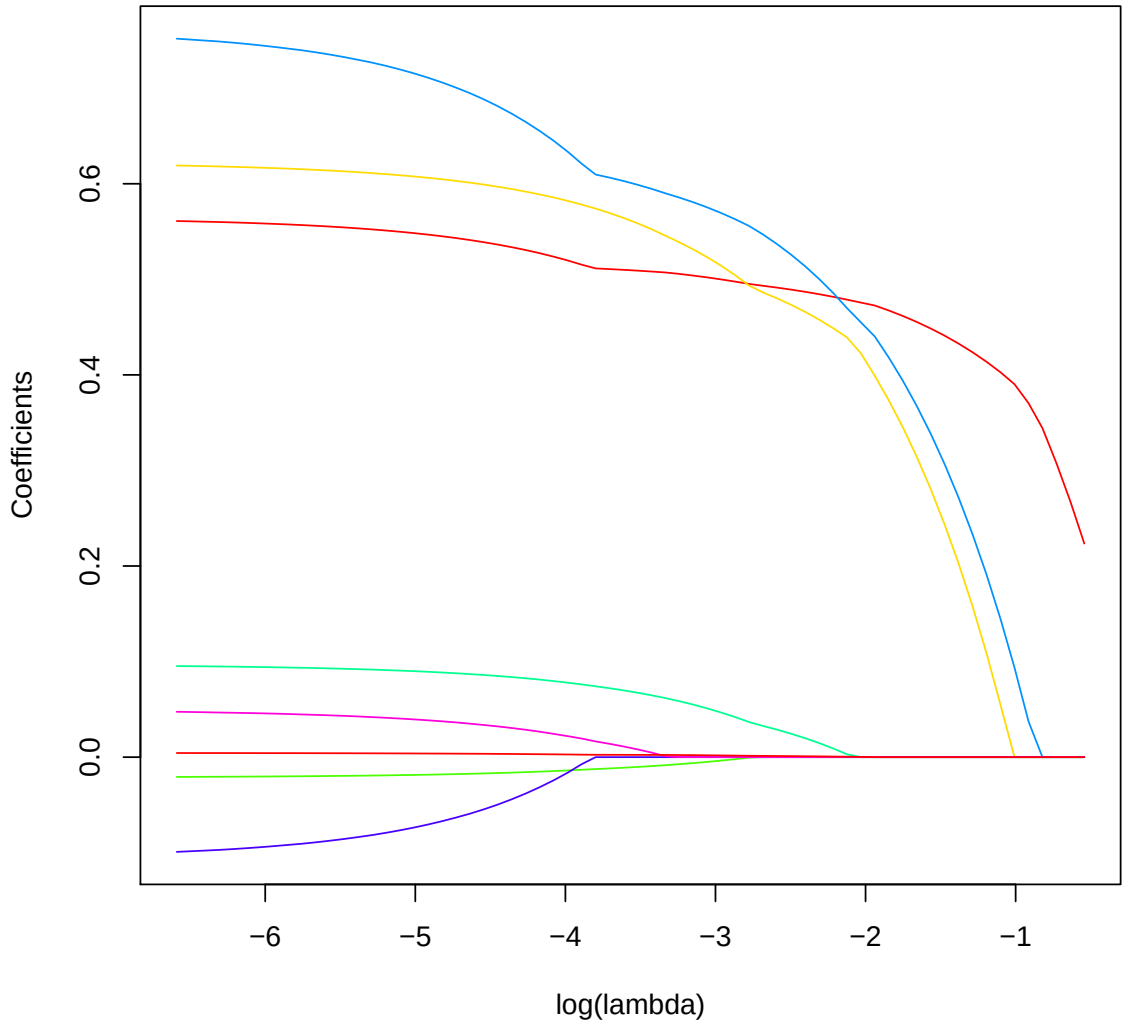
$$\hat{\beta}_{lasso}(t) = \underset{||\beta||_1 \leq t}{\text{argmin}} \, ||\mathbb{Y} - \mathbb{X}\beta||_2^2$$

In its corresponding Lagrangian dual form:

$$\hat{\beta}_{lasso}(\lambda) = \underset{\beta}{\text{argmin}} \, ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1$$

(Note that if $\text{rank}(X) < p$, then the objective function is not strictly convex. There are now an infinite number of possible lasso solutions. (all must have the same fitted value and $||\cdot||_1$))

Lasso regression path



The lasso in R: glmnet

Luckily, we already know how to lasso.

Just change the 'alpha =0' to 'alpha =1', and you're lassoing.

```
lasso.out = glmnet(x=as.matrix(X),y=Y,alpha=1)
#Note: glmnet automatically scales X
```

glmnet uses gradient descent to quickly fit the lasso solution

It can...

- handle other likelihoods than Gaussian
- supports/exploits sparse matrices (e.g. for text processing)
- use warm restarts for the grid of λ to produce more stable fits/faster computations

(See (Friedman et al. (2007) for details))

Optimality conditions: Review

$$\text{minimize } F(x) \tag{8}$$
$$\text{subject to } x \in \mathbb{R}^p \tag{9}$$

$$\text{Search for } x_* \text{ such that } \nabla F|_{x_*} = 0$$

- Turns a geometric problem into an algebraic problem: solve for the point where the gradient vanishes
- Is necessary for optimality of x_*. Is sufficient if F is convex and smooth.

Gradient descent: Intuition

A summary:

1. Start with some initial x^0
2. Propose x to reduce $F(x)$
3. Alternate between 1. and 2. until the objective function doesn't change (much).

Algorithmically, the implementations tend to look like

$$x[k+1] \leftarrow x[k] + \alpha_k v[k],$$

where

- $x[k]$ is the current value of the minimizing parameter
- $v[k]$ is a direction that (hopefully) reduces F
- α_k is a relaxation term.

Gradient descent

Assume $\exists x_* \in D$ such that $\nabla F|_{x_*} = 0$

Define the map

$$\psi(x) = x - \alpha \nabla F|_x$$

(Recall the general form $x[k+1] \leftarrow x[k] + \alpha_k v[k]$)

If ψ is contractive, ie

$$||\psi(x) - \psi(x')|| \leq c||x - x'||$$

where $c \in [0, 1)$, then...

Gradient descent is guaranteed to converge

Gradient descent: Convergence proof

$$
\begin{aligned}
||x[k+1] - x_*|| &= ||x[k] - \alpha \nabla F|_x - x_*|| && (10)\\
&= ||\psi(x[k]) - \psi(x_*)|| && (11)\\
&\leq c||x[k] - x_*|| && (12)\\
&\vdots && (13)\\
&\leq c^{k+1}||x[0] - x_*|| && (14)\\
& && (15)
\end{aligned}
$$

(This means we get exponential convergence[14])

Important fact: If F is $2\times$ differentiable, contractivity means F is convex on D

[14] Optimization people call this linear convergence due to equation (12)

The lasso in R: LARS

Alternatively, the lars package exploits the fact that the coefficient profiles are piecewise linear, which leads to an algorithm with the same computational cost as the full least-squares fit on the data

(See Osborne et al. (2000) for details on the convex optimization, Efron et al. (2004) for the LARS algorithm)

Choosing the tuning parameter for lasso

Of course, just like in Ridge, we need a way of choosing this tuning parameter.

We can just use cross-validation again, though this is still an area of active research:

Homrighausen, D. and McDonald, D.J. *Leave-one-out cross-validation is risk consistent for lasso*, Machine Learning

Homrighausen, D. and McDonald, D.J. *Risk consistency of cross-validation for lasso-type procedures*, Journal of Machine Learning Research

Homrighausen, D. and McDonald, D.J. *The lasso, persistence, and cross-validation*, (2013) International Conference on Machine Learning, JMLR 28(3), 1031–1039.

Choosing the tuning parameter for lasso

For cross-validation, the heavy lifting has been done for us

```
cv.glmnet(x=as.matrix(X),y=Y,alpha=1)
```

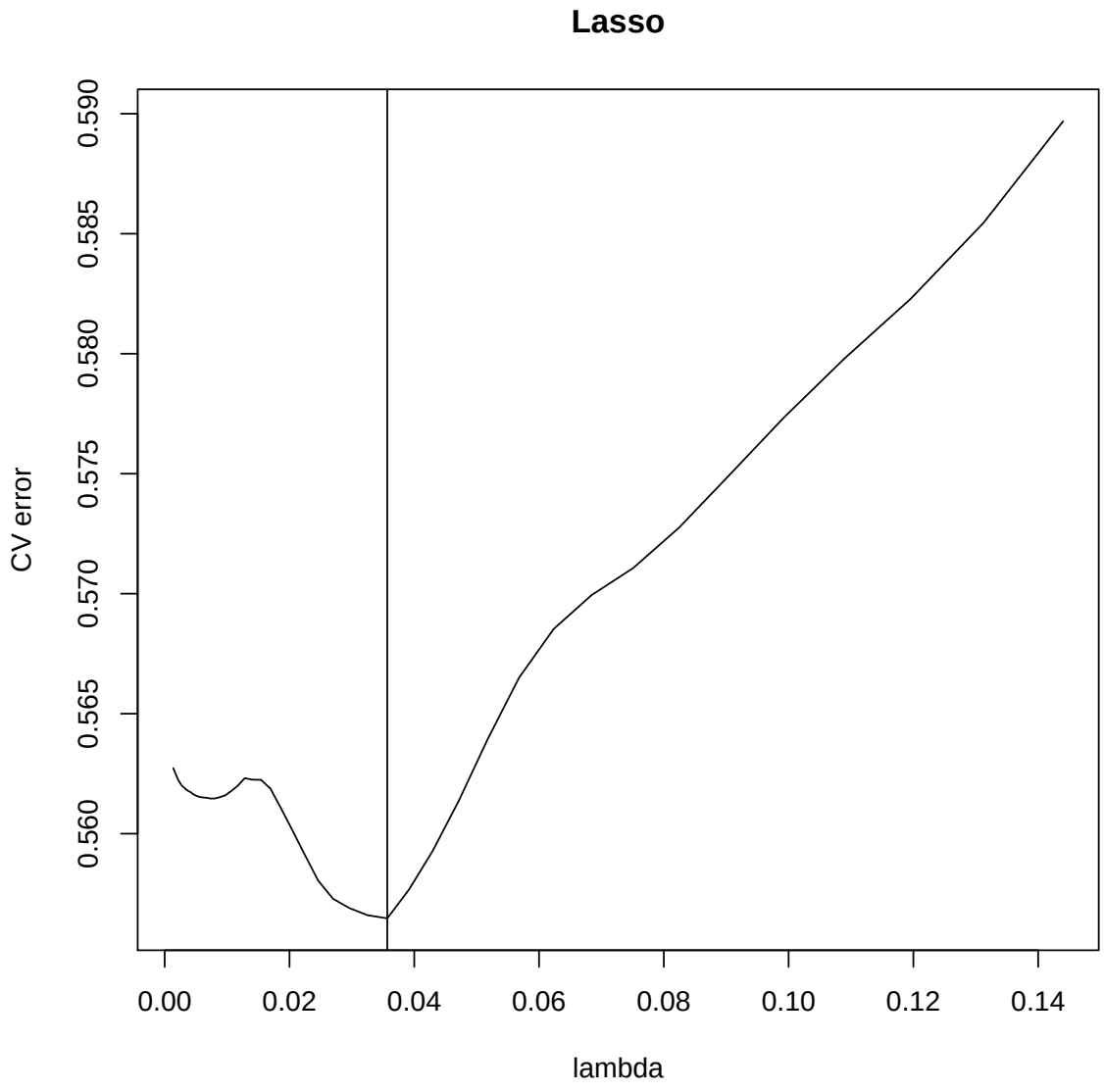
```
cv.lars(x=as.matrix(X),y=Y,type='lasso')
```

Note that for the grid λ, we need only look over the interval $\left[0, ||\mathbb{X}^\top Y||_\infty\right)$

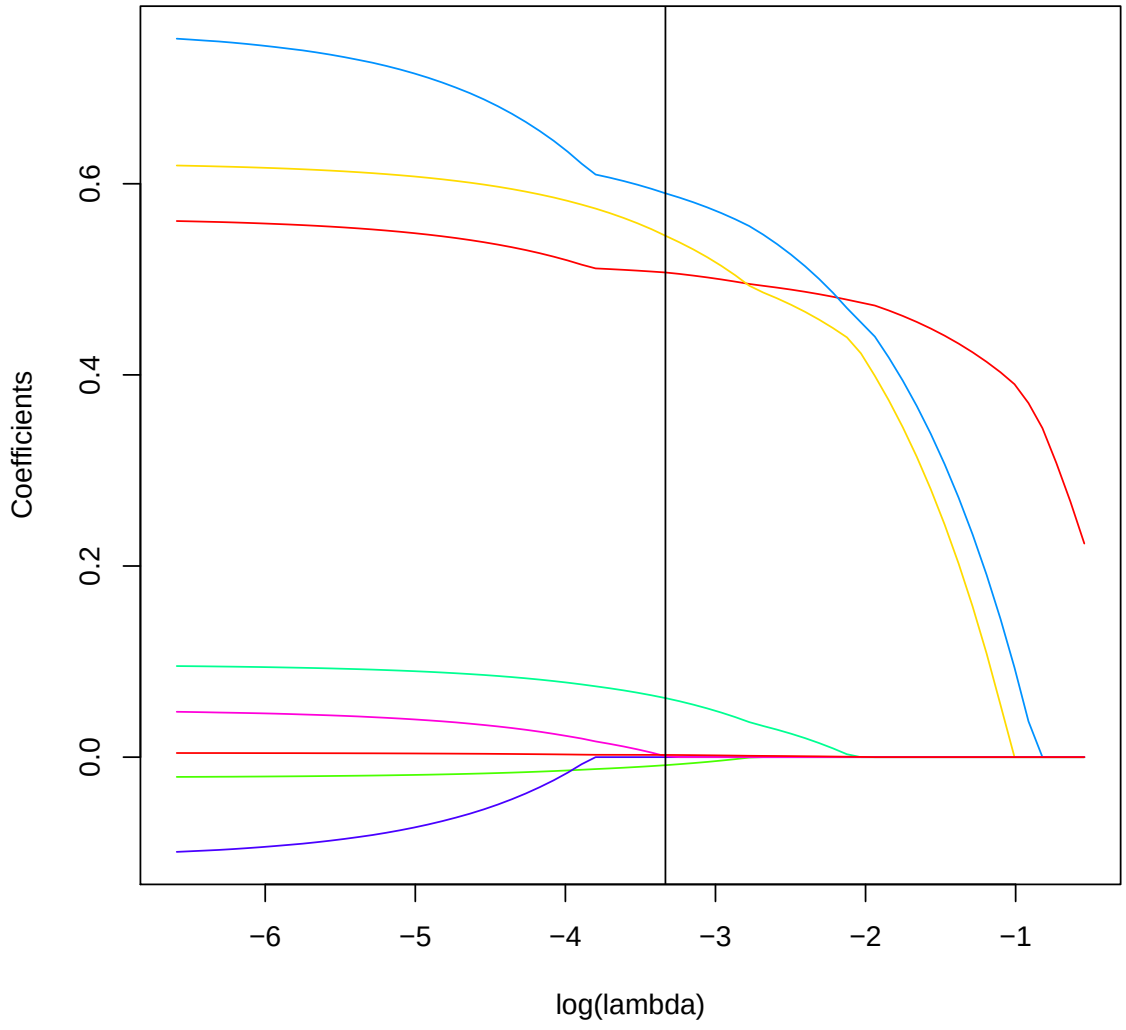
A grid of t has a similar restriction $[0, t_0)$, where

$$t_0 = \min_{\{b:\mathbb{X}b=0\}} ||\hat{\beta}_{\text{LS}} + b||_1$$

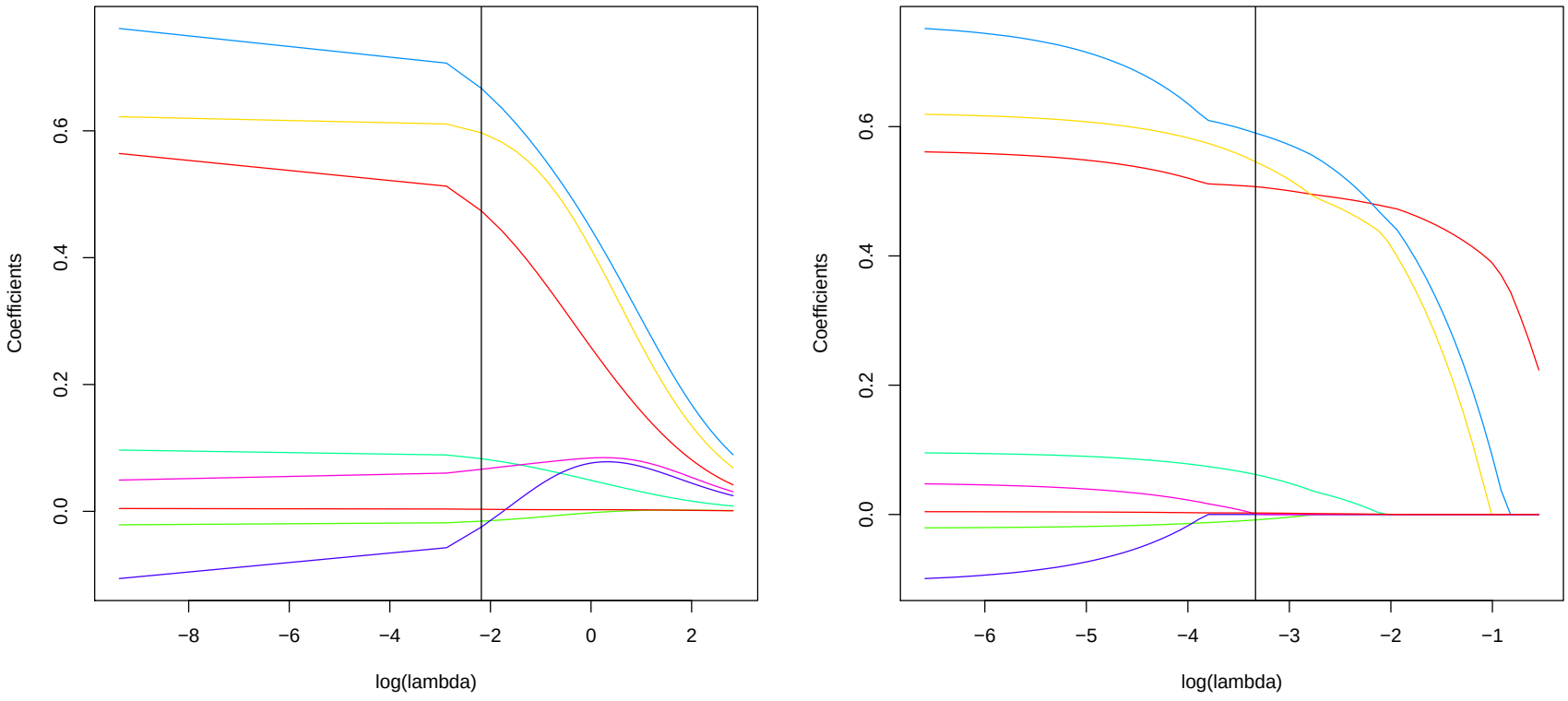
The lasso in R



Lasso regression path



Comparison: Regression path



Vertical line at minimum CV tuning parameter

Comparison of lars and glmnet

There are two main problems with glmnet

- In practice, the λ interval looks like $\left[\epsilon||\mathbb{X}^\top Y||_\infty, ||\mathbb{X}^\top Y||_\infty\right)$ for a small ϵ. Sometimes, this results in finding a boundary solution.
- The iterative nature sometimes results in bad coefficient vectors (such as having more than $\min\{n, p\}$ nonzero coefficients, which is impossible[15])

There are two main problems with lars

- It is slow(er)
- It doesn't support other likelihoods

[15] This is not quite true (Tibshirani (2013), Lemma 13). However, see Lemma 15 in same paper: For any $\mathbb{X}, \lambda$ and *almost* all Y, the column space of $\mathbb{X}_\mathcal{S}$ is the same for every $\mathcal{S}$, where $\mathcal{S} = \{j : |\hat{\beta}_{\lambda,j}| > 0\}$

Alternative method for choosing λ: Scaled sparse regression

Scaled sparse regression (Sun, Zhang (2012)). Uses the idea that we know that the optimal λ scales like σ^2. Form the penalized joint likelihood:

$$L_{\lambda_0}(\beta, \sigma) = \frac{\hat{\mathbb{P}}\ell_\beta}{2n\sigma} + (1-a)\sigma/2 + \lambda_0||\beta||_1$$

Alternatively minimize L in β and σ via:

1. $\hat{\sigma} \leftarrow ||Y - \mathbb{X}\hat{\beta}_{old}||_2^2/\sqrt{n(1-a)}$
2. $\lambda \leftarrow \hat{\sigma}\lambda_0$
3. $\hat{\beta} \leftarrow \hat{\beta}_\lambda$
4. If $||\hat{\beta} - \hat{\beta}_{old}||$ is not small, set $\hat{\beta}_{old} \leftarrow \hat{\beta}$ and return to 1.

Flavors of lasso

- Grouped lasso (Yuan and Lin (2007), Meier et al. (2008)), where variables are included or excluded in groups.
- Refitted lasso (e.g. Lederer 2013). Takes the estimated model from lasso and fits the full least squares solution on selected covariates (less bias, more variance).
- Dantzig selector (Candes, Tao (2007)), a slightly modified version of the lasso
- The elastic net (Zou, Hastie (2005)), generally used for correlated variables that combines a ridge/lasso penalty. Included in glmnet. Fixes non-uniqueness problem of lasso (although, see Tibshirani (2013)).
- SCAD (Fan and Li (2005)), a non-convex version of lasso that adds a more severe variable selection penalty
- $\sqrt{\text{lasso}}$ (Belloni et al. (2011)), claims to be tuning parameter free (but isn't). Uses $||\cdot||_2$ instead of $||\cdot||_2^2$ for the loss.
- Generalized lasso (Tibshirani, Taylor (2011)). Adds various additional penalty matrices to the penalty term (ie: $||D\beta||_1$).