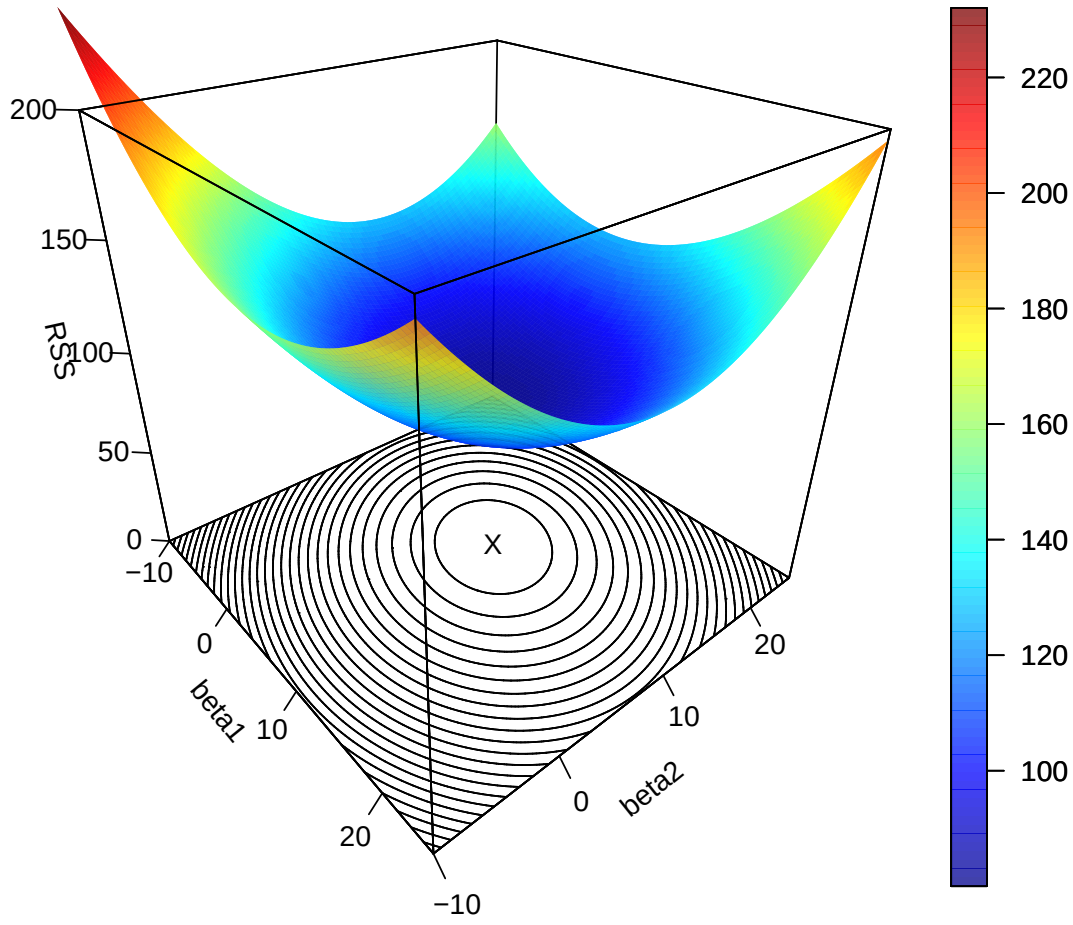
Linear Methods for Regression: Regularization

-Statistical Machine Learning-

Lecturer: Darren Homrighausen, PhD

Least squares



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

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[1]:

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

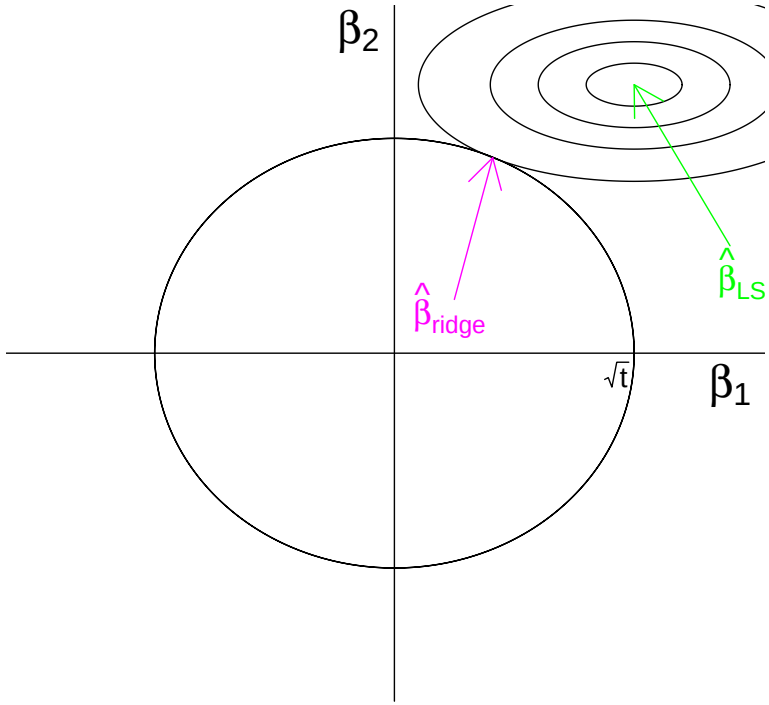
for any $t \geq 0$.

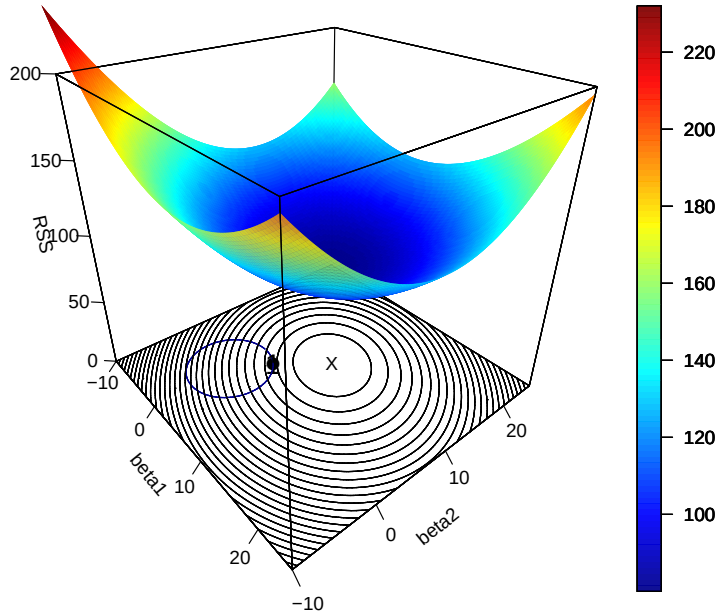
Compare this to least squares

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

[1] Hoerl, Kennard (1970)

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





Ridge regression

An equivalent way to write

$$\hat{\beta}_{\text{ridge}}(t) = \underset{||\beta||_2^2 \leq t}{\text{argmin}} \, ||Y - \mathbb{X}\beta||_2^2 \tag{1}$$

is in the Lagrangian form

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

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

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

Regularization and standardization

The coefficient vector isn't invariant to rescaling

If an intercept is included, do not penalize it:

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} (Y_i - \beta_0 + \beta^\top X_i)^2 + \lambda ||\beta||_2^2$$

The usual way of addressing this in regression is:

- Standardize all features for which scale is meaningful:

$$x_j \leftarrow \frac{(x_j - \text{mean}(x_j))}{\text{sd}(x_j)}$$

  (So, don't standardize indictor variables, for instance)

- Standardize the response $Y \leftarrow Y - \text{mean}(Y)$
- Don't include an 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

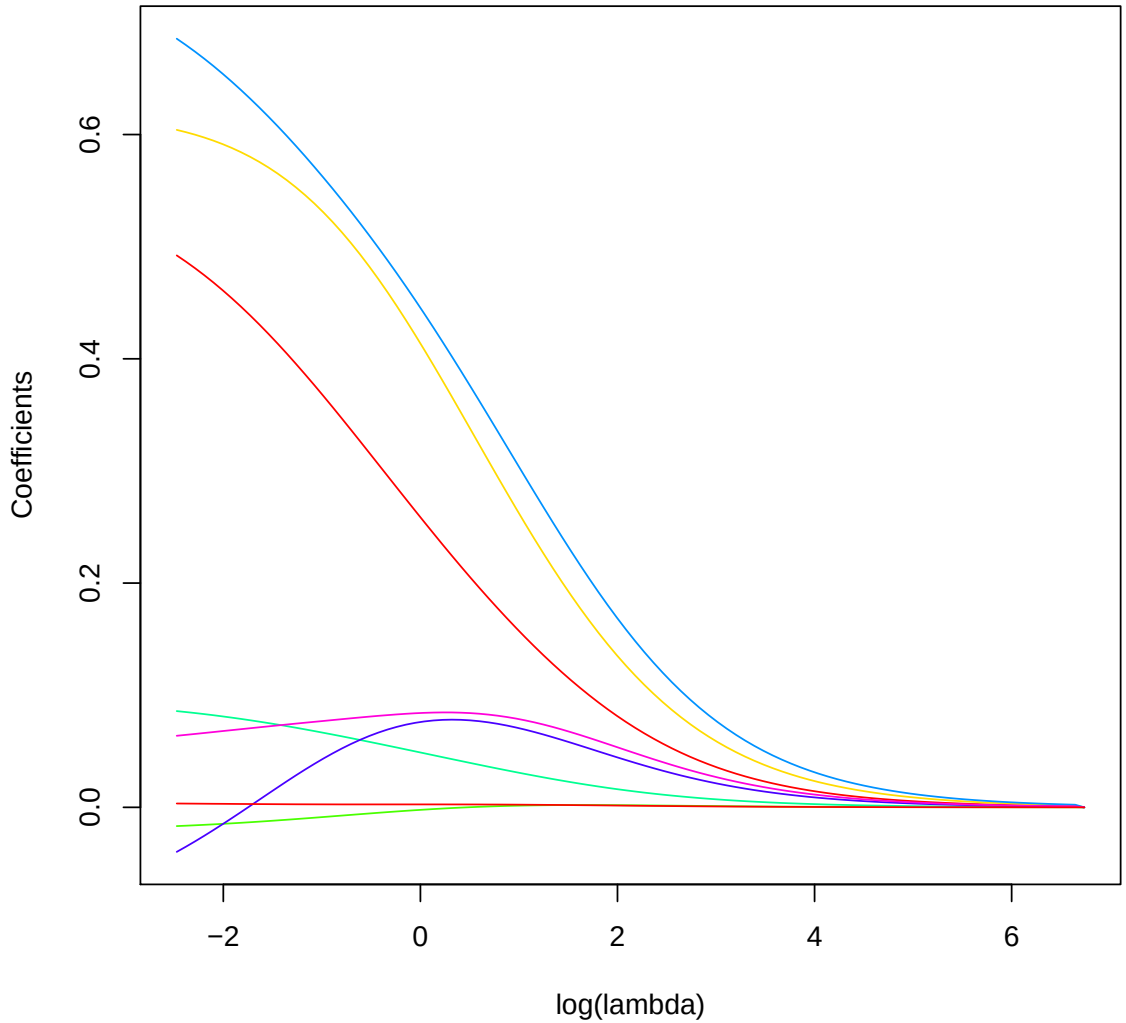
(Alternatively, hyper-parameters)

However we think about it, we have produced a suite of solutions

$$\{\hat{\beta}_{\text{ridge}}(\lambda) : \lambda \in [0, \infty)\}$$

What do these solutions look like?

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.

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

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

Similarly

$$\mathbb{X}\hat{\beta}_{\text{LS}} = UU^\top Y \qquad = \sum_{j=1}^{p} u_j u_j^\top Y$$

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

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

Regularization - Ridge Regression

Computationally, we need the ridge solution at many values of λ

Naively, that would mean solving the normal equations many times

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

However, with the SVD form, we can compute $\mathbb{X} = UDV^\top$ once and "solve" with only matrix multiplications

$$\hat{\beta}_{\text{ridge}}(\lambda) = \sum_{j=1}^{p} v_j \left(\frac{d_j}{d_j^2 + \lambda} \right) u_j^\top Y$$

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,

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

Ridge in a new space: Computations

The ridge solution solves either the normal equations

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

or the adjoint problem

$$\mathbb{X}^\top (\mathbb{X}\mathbb{X}^\top + \lambda I)^{-1} Y$$

The 'heavy lifting' in each case is done with the inversion

- $\mathbb{X}^\top \mathbb{X} \in \mathbb{R}^{p \times p} \Longrightarrow$ takes p^3 computations, p^2 space
- $\mathbb{X}\mathbb{X}^\top \in \mathbb{R}^{n \times n} \Longrightarrow$ takes n^3 computations, n^2 space

Conclusion: Depending on the relative size of n and p, this could be 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 $\langle X, X' \rangle = X^\top X'$ 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 $\langle \Phi(X_i), \Phi(X_{i'}) \rangle$

Inserting Φ is known as kernelization or a kernel trick

(Kernel) ridge regression

Example: Suppose $X = (\text{income}, \text{height})^\top$

Then we could specify the map

$$\Phi(X)^\top = (\text{income}, \text{height}, \text{income} * \text{height}, \text{income}^2, \text{height}^2)$$

The induced feature matrix is then

$$\mathbb{X} = \begin{bmatrix} \Phi(X_1) \\ \vdots \\ \Phi(X_n) \end{bmatrix} \in \mathbb{R}^{n \times 5}$$

(Kernel) ridge regression

Ordinarily, this would mean we need to solve the normal equation inversion for $p = 5$

- $\mathbb{X}^\top \mathbb{X} \in \mathbb{R}^{p \times p} \implies$ takes p^3 computations, p^2 space

However, using the kernel trick we can solve instead

- $\mathbb{X}\mathbb{X}^\top \in \mathbb{R}^{n \times n} \implies$ takes n^3 computations, n^2 space

which is fixed in p

Implication: We can add essentially arbitrary nonlinearity without paying higher computational, storage cost!

We will return to this again with support vector machines (SVM)

Ridge in practice

Ridge Regression: 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

$$\text{df} = \text{trace}\left[\mathbb{X}(\mathbb{X}^\top\mathbb{X} + \lambda I)^{-1}\mathbb{X}^\top\right] = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$$

(As $\lambda \to 0$, we get the rank of $\mathbb{X}$)

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: The tuning parameter

To see this last claim, observe

$$\log\left(\mathrm{GCV}(\hat{\beta})\right) \propto \log(\hat{R}_{\mathrm{train}}) - 2\log(1 - \mathrm{df}(\hat{\beta})/n)$$

VERSUS

$$\mathrm{AIC}(\hat{\beta}) \propto \log(\hat{R}_{\mathrm{train}}) + 2n^{-1}\mathrm{df}(\hat{\beta})$$

Ridge Regression: 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 obtuse way, as it will make things easier later.

```
if(!require(glmnet)){install.packages('glmnet');require(glmnet)}
ridge.out = cv.glmnet(x=X,y=Y,alpha=0)
```

glmnet automatically scales X, unless you set

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

Note:

- The function cv.glmnet computes CV_K for a grid of λ values as well as the ridge solutions
- The function glmnet just computes the ridge solutions over a grid of λ values
- The feature matrix X must be a "matrix" data structure

```
> class(X)
[1] "matrix"
```

Ridge computations

Ridge Regression: CV plot

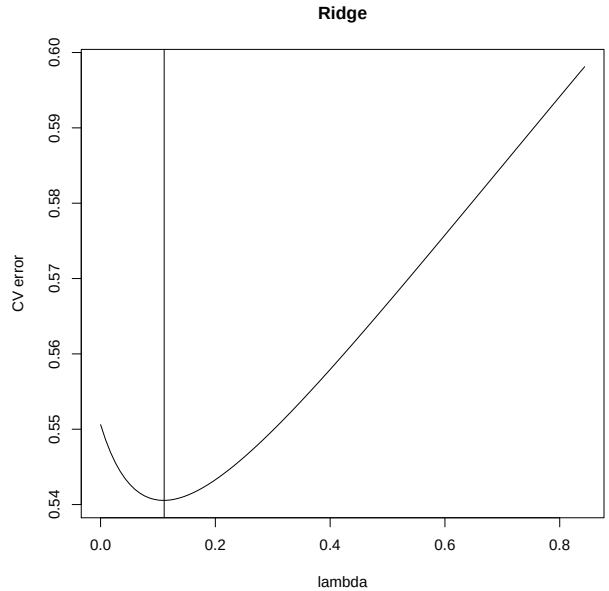
```
lam = 1
svd.out   = svd(X)
ridge.svd = svd.out$v %*%
                  diag(svd.out$d/(svd.out$d**2 + lam)) %*%
                  t(svd.out$u) %*% Y

Ytilde   = c(Y,rep(0,p))
Xtilde   = rbind(X,diag(sqrt(lam),p))
ridge.lm = lm(Ytilde~Xtilde-1)

library(glmnet)
ridge.glmnet = glmnet(x=X,y=Y,alpha=0,
                      standardize=FALSE,intercept=FALSE,
                      lambda = seq(1.1/n,.9/n,length.out = 1000))
```

Ridge Regression: Compare

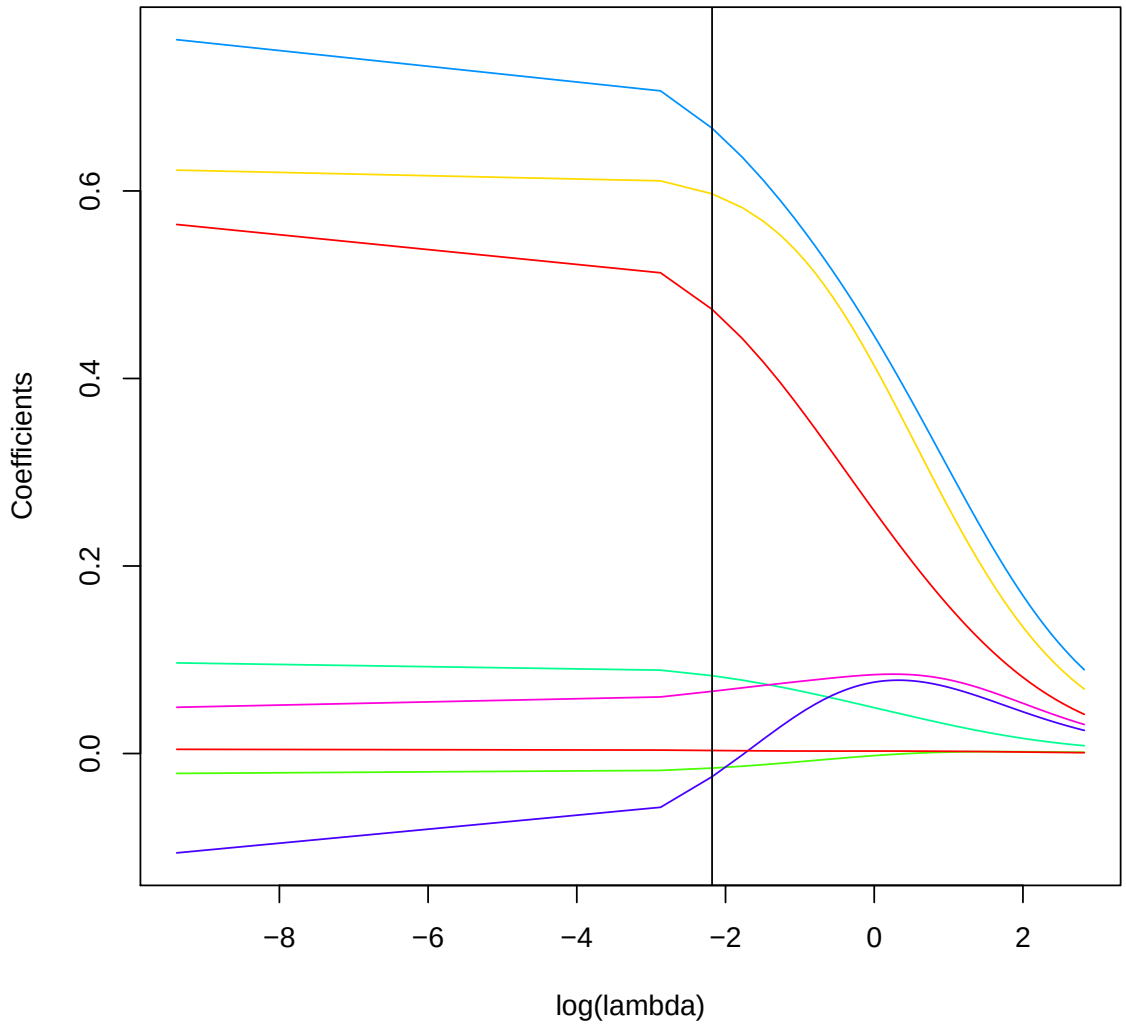
```
> print(ridge.svd[1:3])
[1]  0.55869858  0.60999902 -0.01933688

> print(coef(ridge.lm)[1:3])
[1]  0.55869858    0.60999902   -0.01933688

> print(coef(ridge.glmnet,s=lam/n)[1:4])
[1]  0.00000000  0.55911894  0.61313693 -0.01950889
```

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)

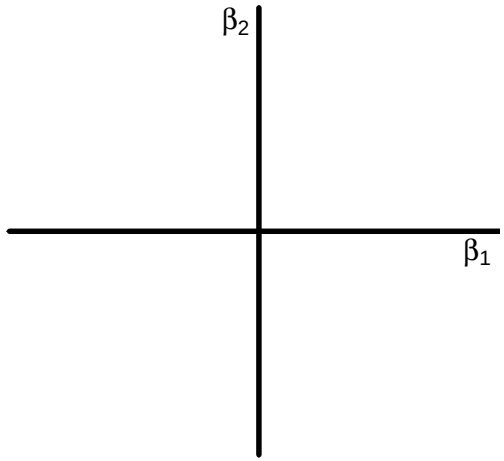
An intuitive idea

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)

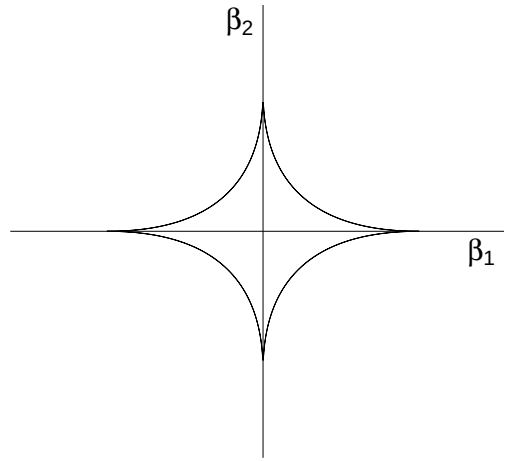
	Best linear regression model	Ridge regression
Computationally Feasible?	No	Yes
Does Model Selection?	Yes	No

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

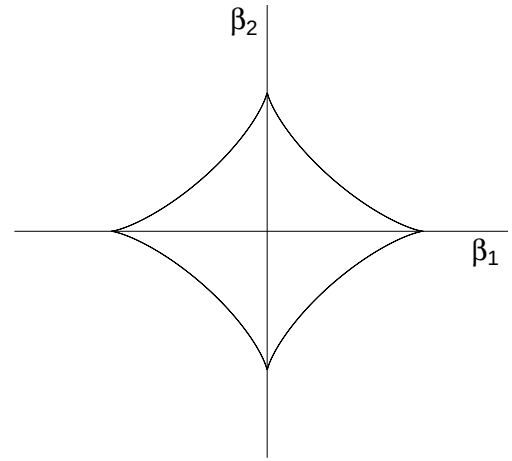
Geometry of regularization in $\mathbb{R}^2$: Convexity



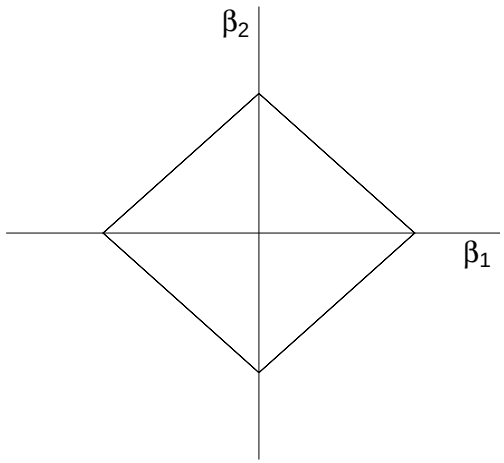
$||\beta||_0 \leq t$



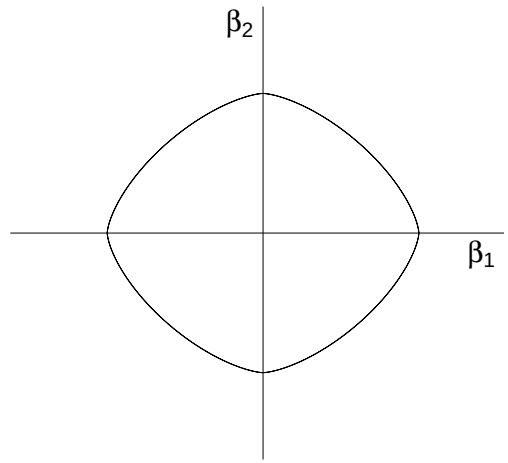
$||\beta||_{\frac{1}{2}} \leq t$



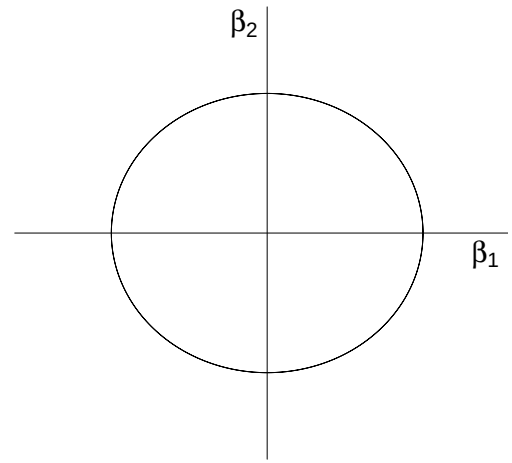
$||\beta||_{\frac{3}{4}} \leq t$



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

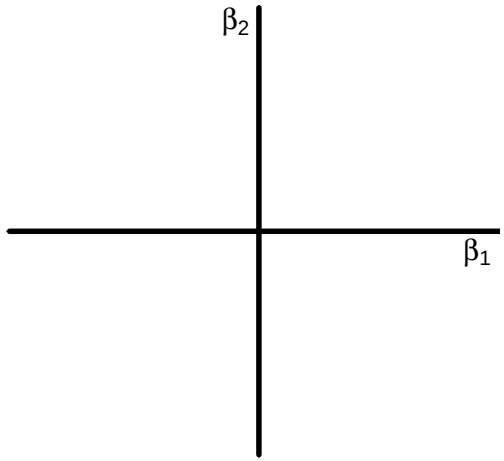


$||\beta||_{\frac{3}{2}} \leq t$

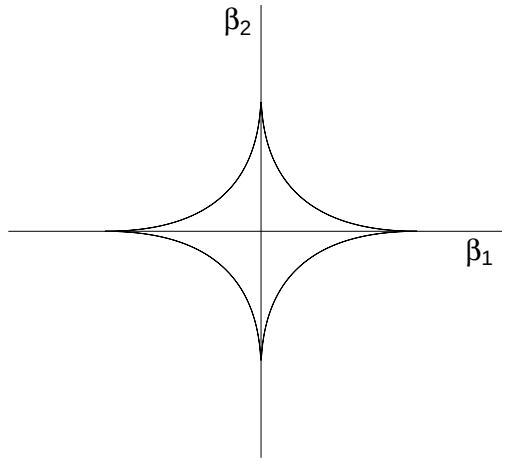


$||\beta||_2 \leq t$

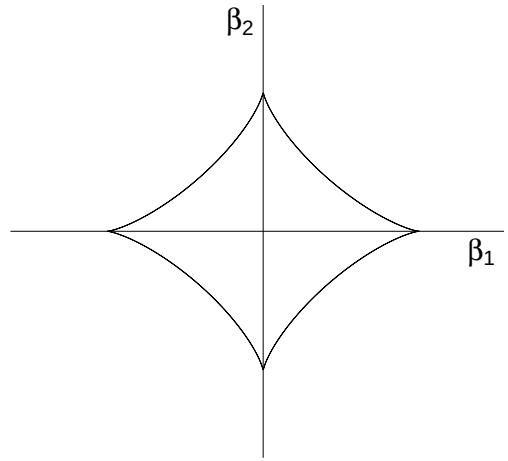
Geometry of regularization in $\mathbb{R}^2$: Convexity



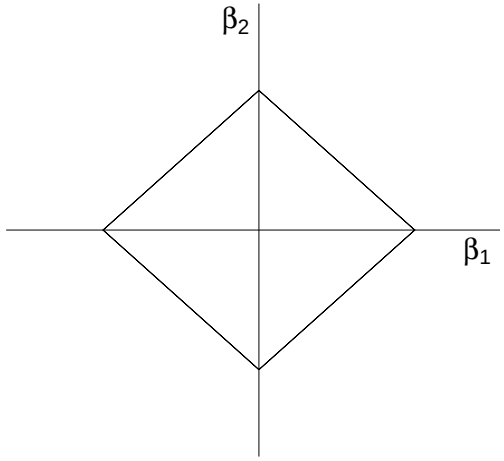
$||\beta||_0 \leq t$



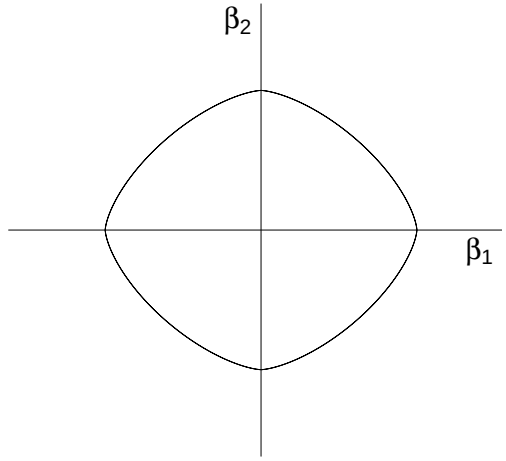
$||\beta||_{\frac{1}{2}} \leq t$



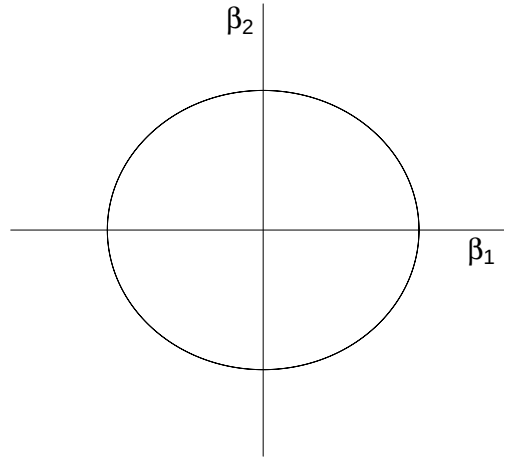
$||\beta||_{\frac{3}{4}} \leq t$



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

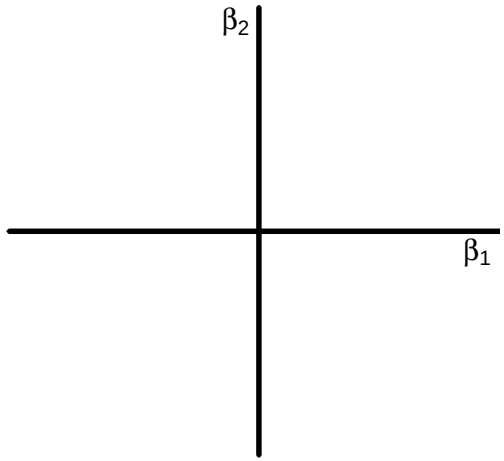


$||\beta||_{\frac{3}{2}} \leq t$

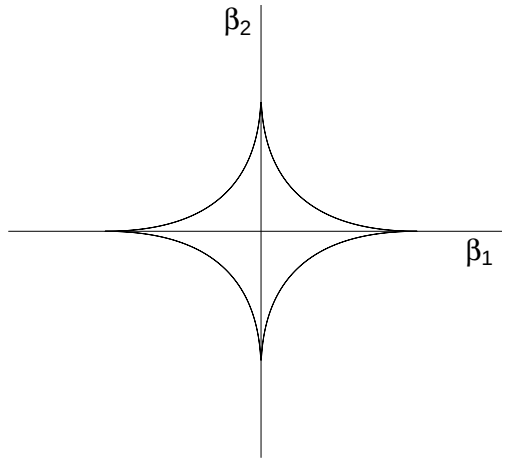


$||\beta||_2 \leq t$

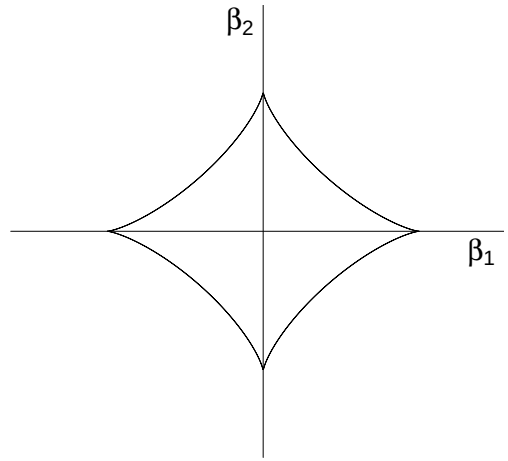
Geometry of regularization in $\mathbb{R}^2$: Convexity



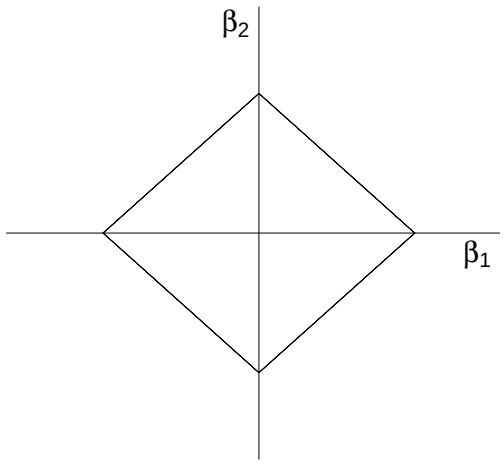
$||\beta||_0 \leq t$



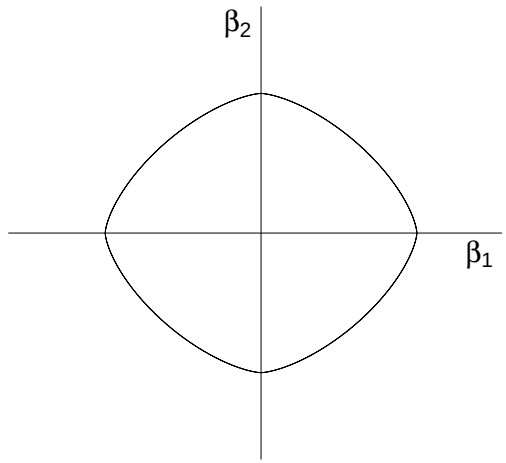
$||\beta||_{\frac{1}{2}} \leq t$



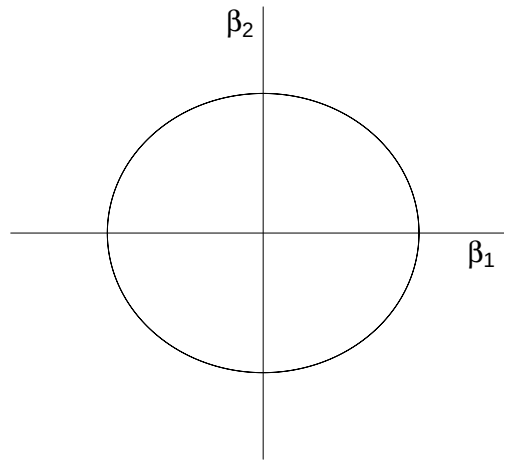
$||\beta||_{\frac{3}{4}} \leq t$



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

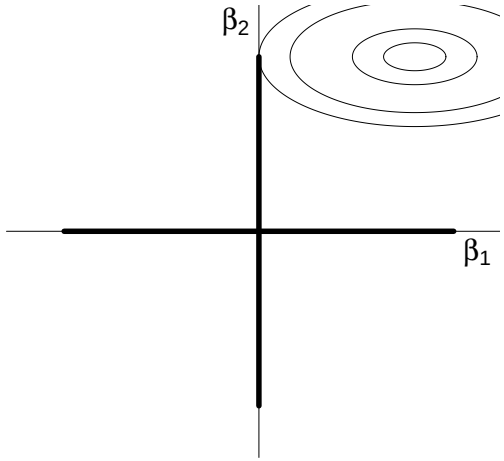


$||\beta||_{\frac{3}{2}} \leq t$

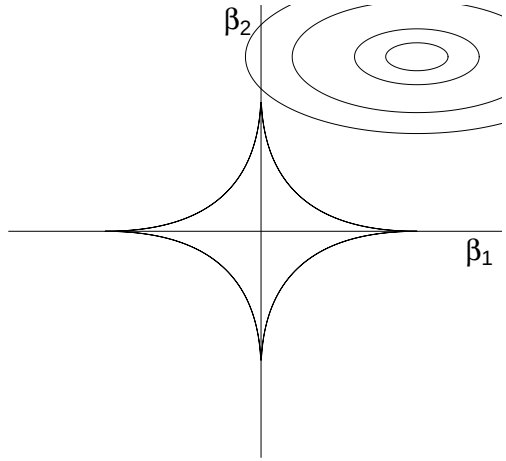


$||\beta||_2 \leq t$

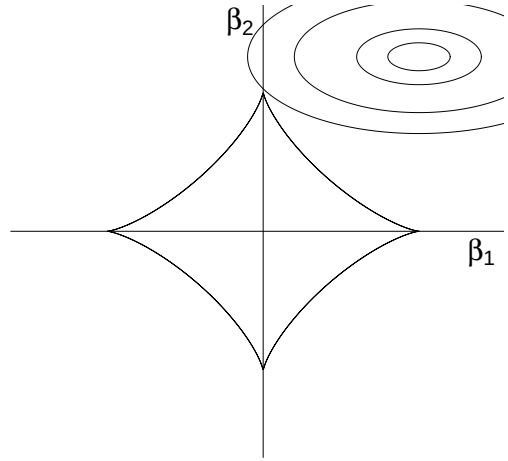
Geometry of regularization in $\mathbb{R}^2$: Model selection



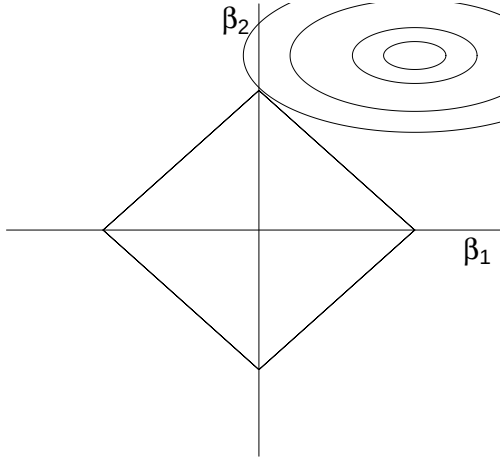
$||\beta||_0 \leq t$



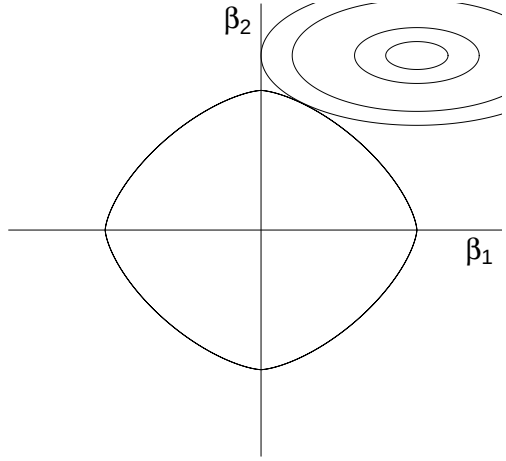
$||\beta||_{\frac{1}{2}} \leq t$



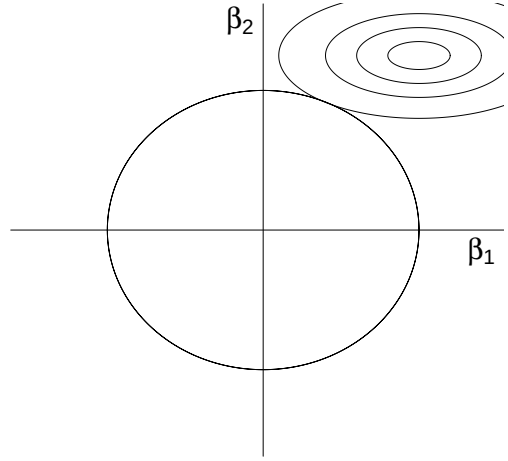
$||\beta||_{\frac{3}{4}} \leq t$



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

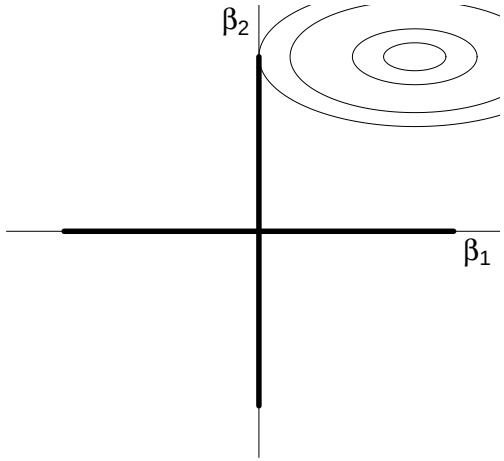


$||\beta||_{\frac{3}{2}} \leq t$

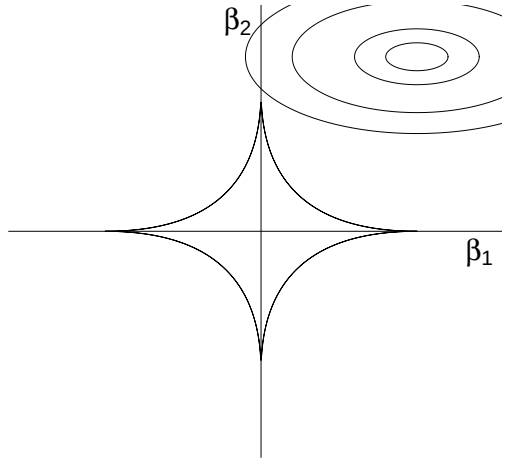


$||\beta||_2 \leq t$

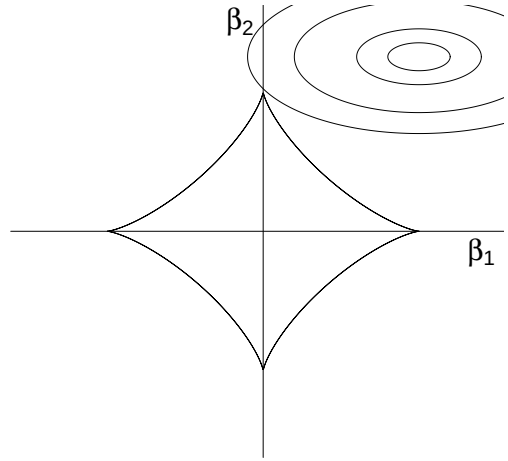
Geometry of regularization in $\mathbb{R}^2$: Model selection



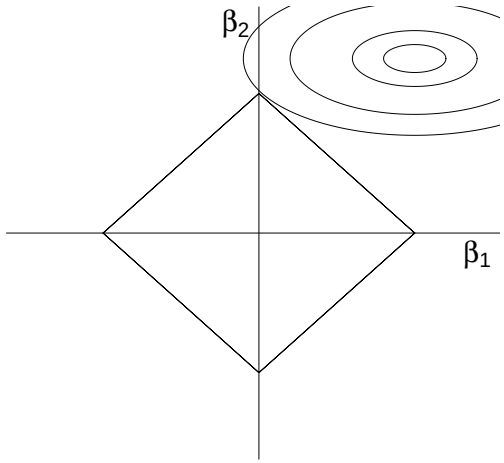
$||\beta||_0 \leq t$



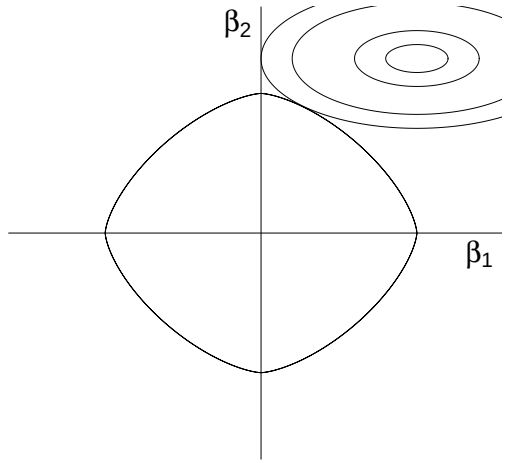
$||\beta||_{\frac{1}{2}} \leq t$



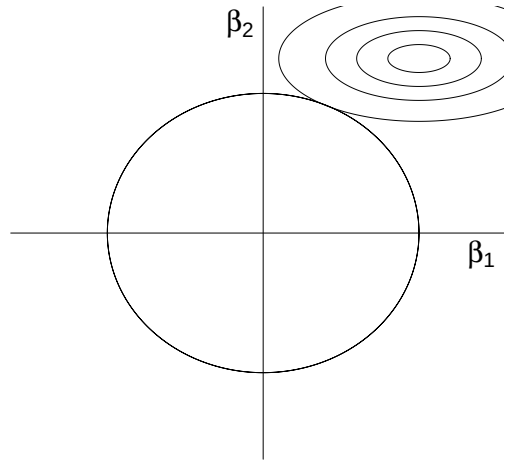
$||\beta||_{\frac{3}{4}} \leq t$



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

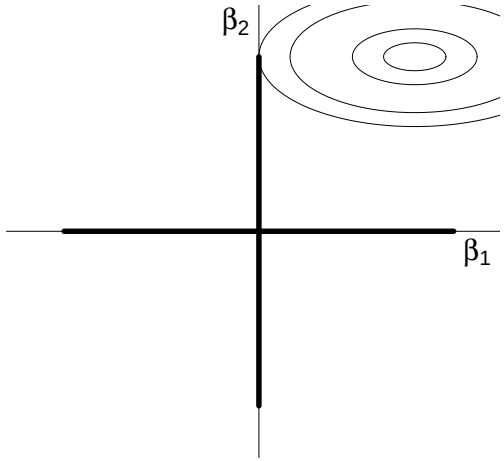


$||\beta||_{\frac{3}{2}} \leq t$

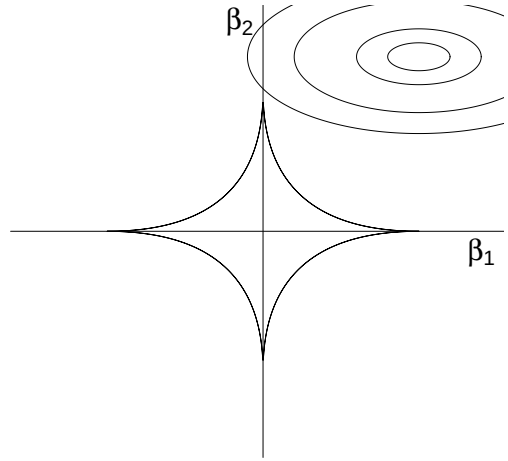


$||\beta||_2 \leq t$

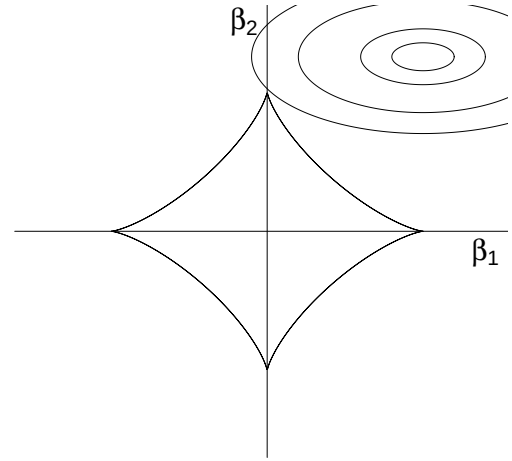
Geometry of regularization in $\mathbb{R}^2$: Model selection



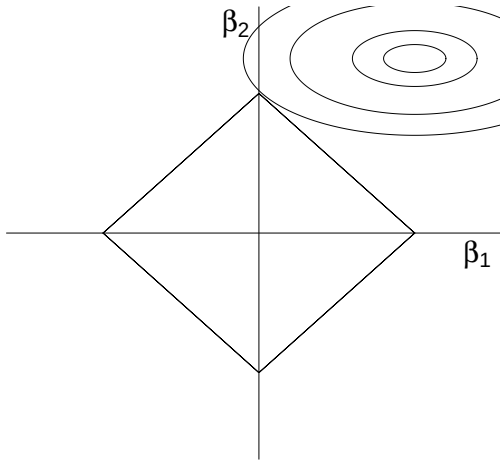
$||\beta||_0 \leq t$



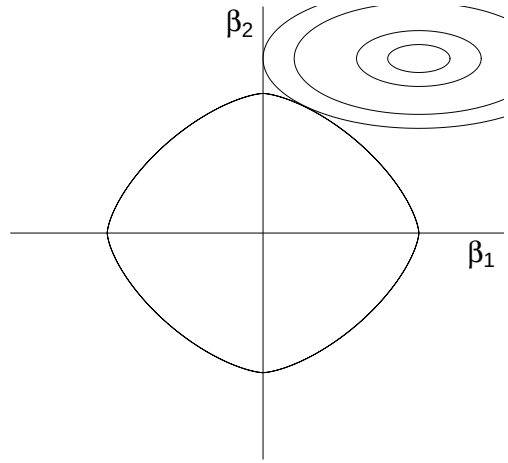
$||\beta||_{\frac{1}{2}} \leq t$



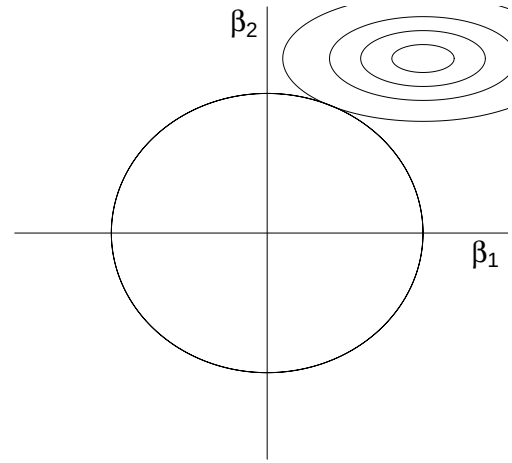
$||\beta||_{\frac{3}{4}} \leq t$



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

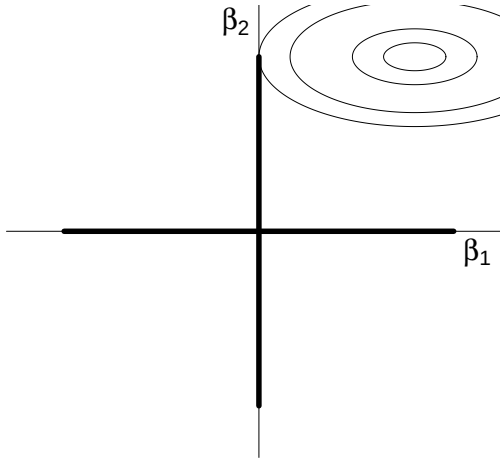


$||\beta||_{\frac{3}{2}} \leq t$

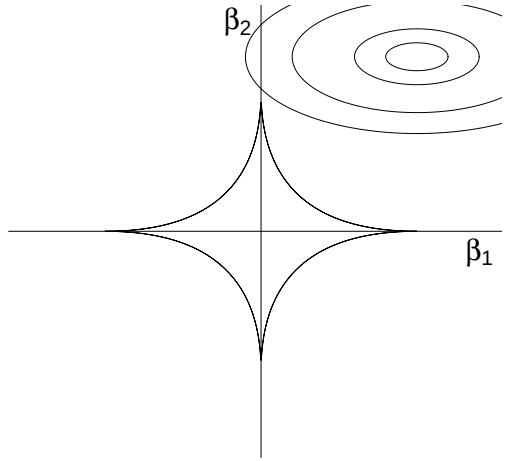


$||\beta||_2 \leq t$

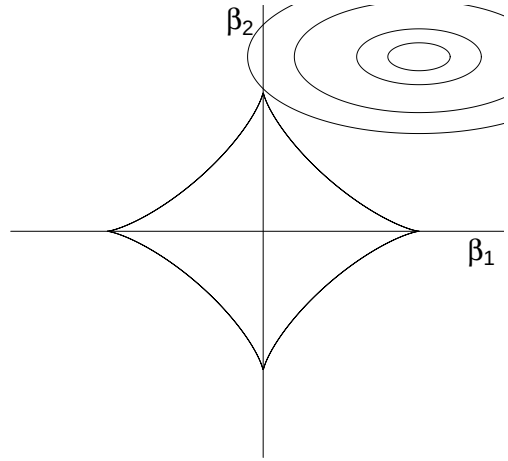
Geometry of regularization in $\mathbb{R}^2$: Both



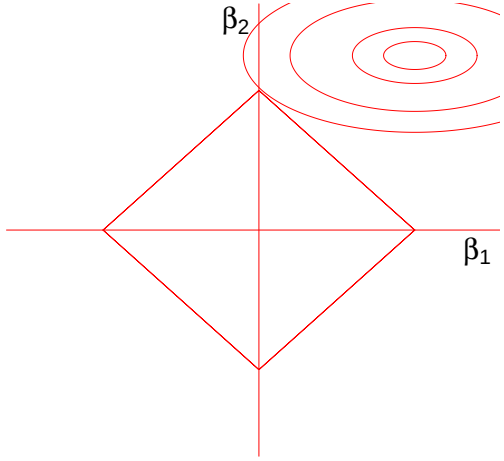
$||\beta||_0 \leq t$



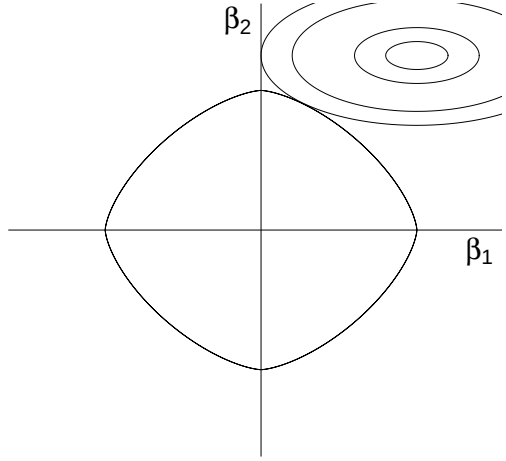
$||\beta||_{\frac{1}{2}} \leq t$



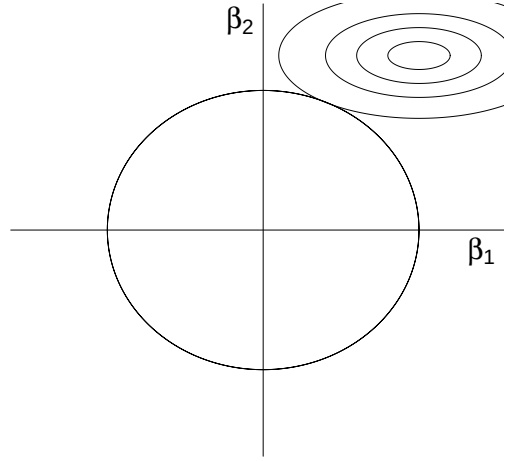
$||\beta||_{\frac{3}{4}} \leq t$



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



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

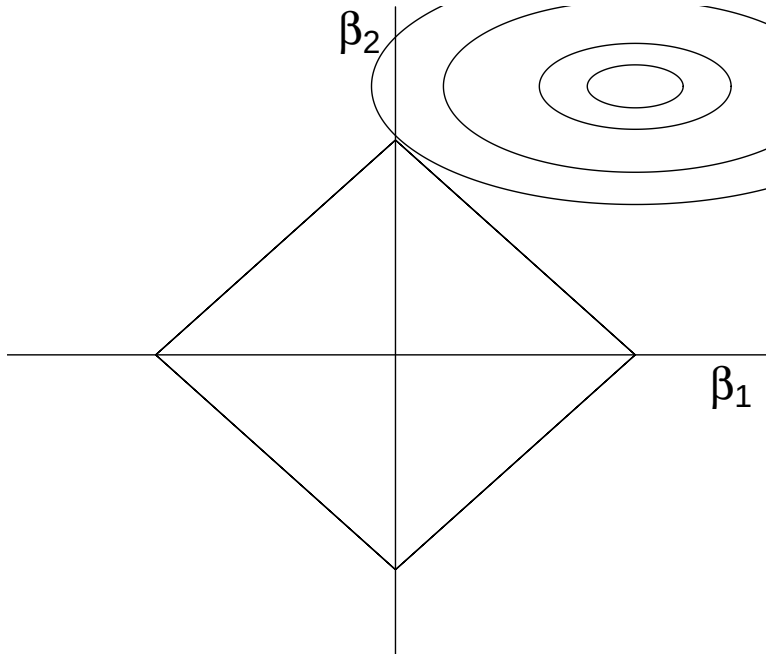


$||\beta||_2 \leq t$

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)

ℓ_1-regularized regression

ℓ_1-REGULARIZED REGRESSION

Related methods are known as

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

The estimator satisfies

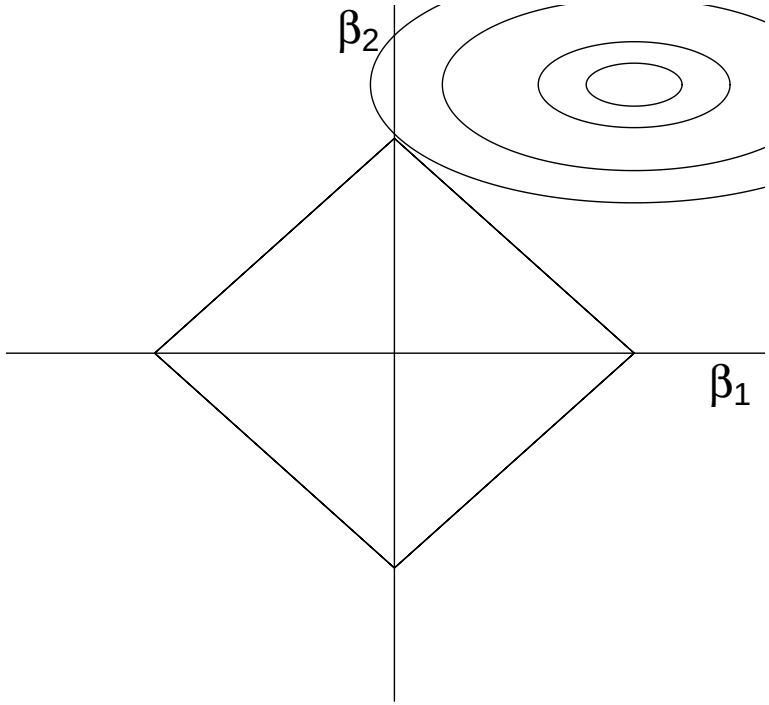
$$\hat{\beta}_{\text{lasso}}(t) = \underset{||\beta||_1 \leq t}{\text{argmin}} \, ||\mathbb{Y} - \mathbb{X}\beta||_2^2$$

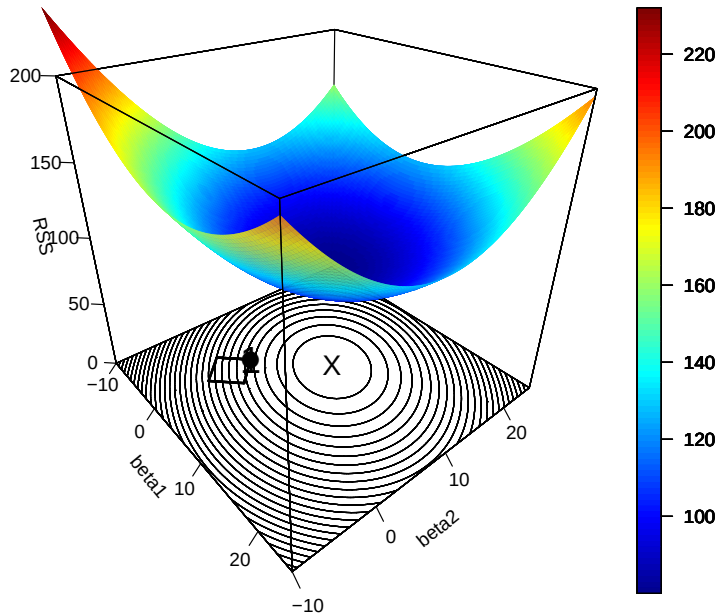
In its corresponding Lagrangian dual form:

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

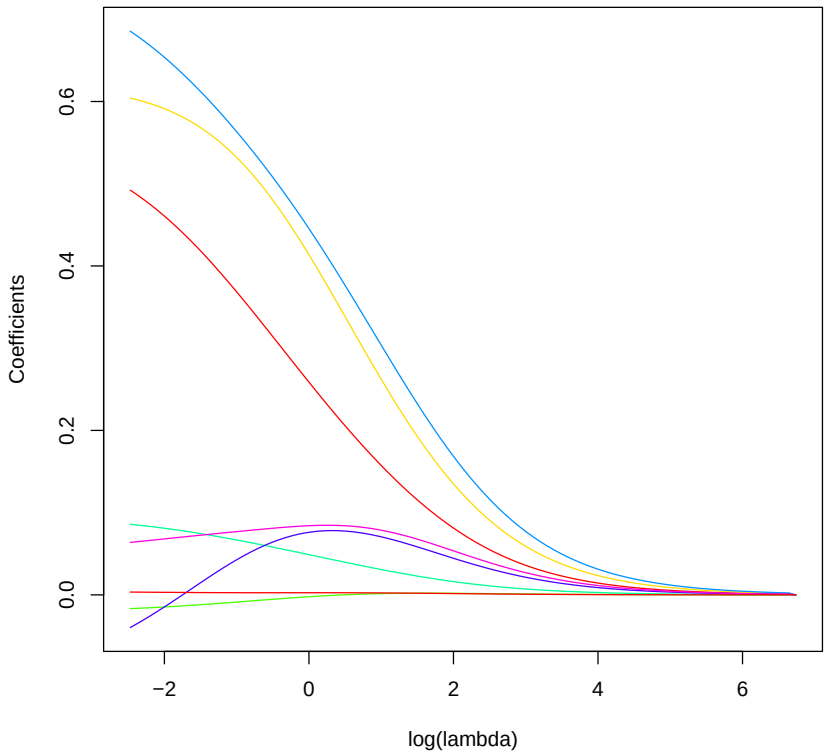
(Note that if $\text{rank}(X) < p$, then the objective function is not strictly convex. There are now an infinite number of possible lasso solutions. (all must have the same fitted value and $||\cdot||_1$))

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

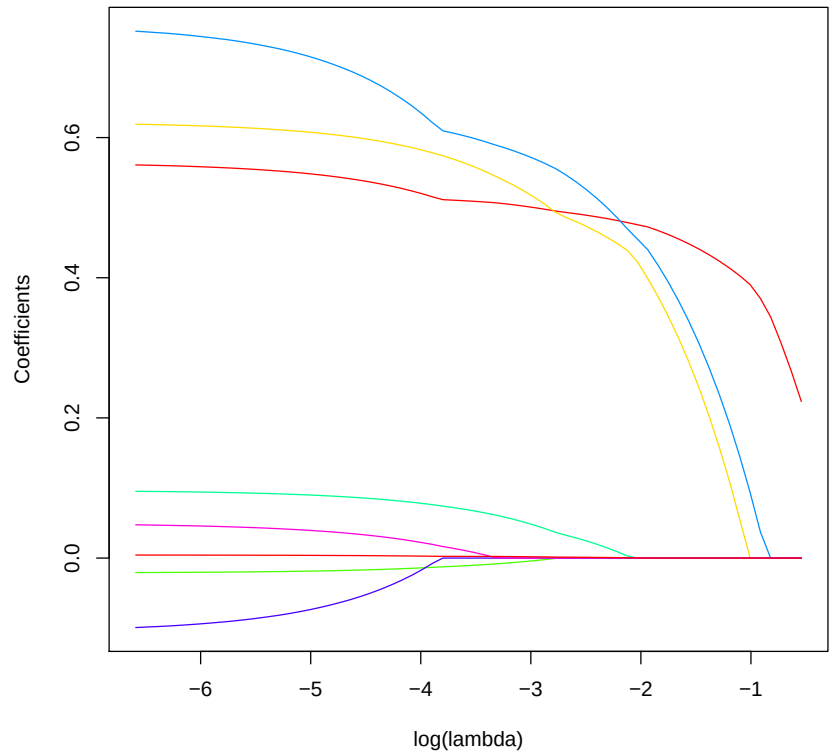




Lasso regression path



Ridge



Lasso

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 a version of 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

$$\text{minimize } F(x) \tag{3}$$
$$\text{subject to } x \in \mathbb{R}^p \tag{4}$$

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

- 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

$$
\begin{aligned}
||x[k+1] - x_*|| &= ||x[k] - \alpha \nabla F|_x - x_*|| && (5)\\
&= ||\psi(x[k]) - \psi(x_*)|| && (6)\\
&\leq c||x[k] - x_*|| && (7)\\
&\vdots && (8)\\
&\leq c^{k+1}||x[0] - x_*|| && (9)\\
& && (10)
\end{aligned}
$$

(This means we get exponential convergence[2])

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

[2] Optimization people call this linear convergence due to equation (7)

Gradient descent algorithm

A natural choice for the direction $v[k]$ is the (negative) gradient: $-\nabla F$

Initialize $x \in D$

1. Compute $v = -\nabla F|_x$
2. Determine α, such as $\alpha = \text{argmin}_{a \geq 0} F(x + av)$
   (This is called a line search)
3. Update $x = x + \alpha v$

Terminate if $||\nabla F|_x||_2 \leq \eta$ (Usually this is checked after step 1. instead of 3.)

If F is strongly convex$^\sharp$ with convexity parameter m, then

- $||\nabla F|_x||_2 \leq (2m\epsilon)^{1/2} \Rightarrow F(x) - F(x_*) \leq \epsilon$
- $||x - x_*||_2 \leq \frac{2}{m} ||\nabla F|_x||_2$

(See Chapter 9.1 in "Convex Optimization")

Gradient descent example

If we look at multiple regression via least squares we get:

$$\begin{aligned}\min_{\beta} ||Y - \mathbb{X}\beta||_2^2 \Rightarrow \frac{\partial}{\partial \beta_j} ||Y - \mathbb{X}\beta||_2^2 \\ &= \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} (Y_i - X_i^\top \beta)^2 \\ &= 2 \sum_{i=1}^{n} (Y_i - X_i^\top \beta) X_{ij}\end{aligned}$$

Hence, we will cycle over j and make the update $k = 1, \ldots, K$ iterations:

$$\hat{\beta}_j^{k+1} = \hat{\beta}_j^k - \alpha \sum_{i=1}^{n} (Y_i - X_i^\top \hat{\beta}^k) X_{ij}$$

Stochastic gradient descent

The gradient descent update again:

$$\hat{\beta}_j^{k+1} = \hat{\beta}_j^k - \alpha[k] \sum_{i=1}^{n} (Y_i - X_i^\top \hat{\beta}^k) X_{ij}$$

(We call this batch gradient descent)

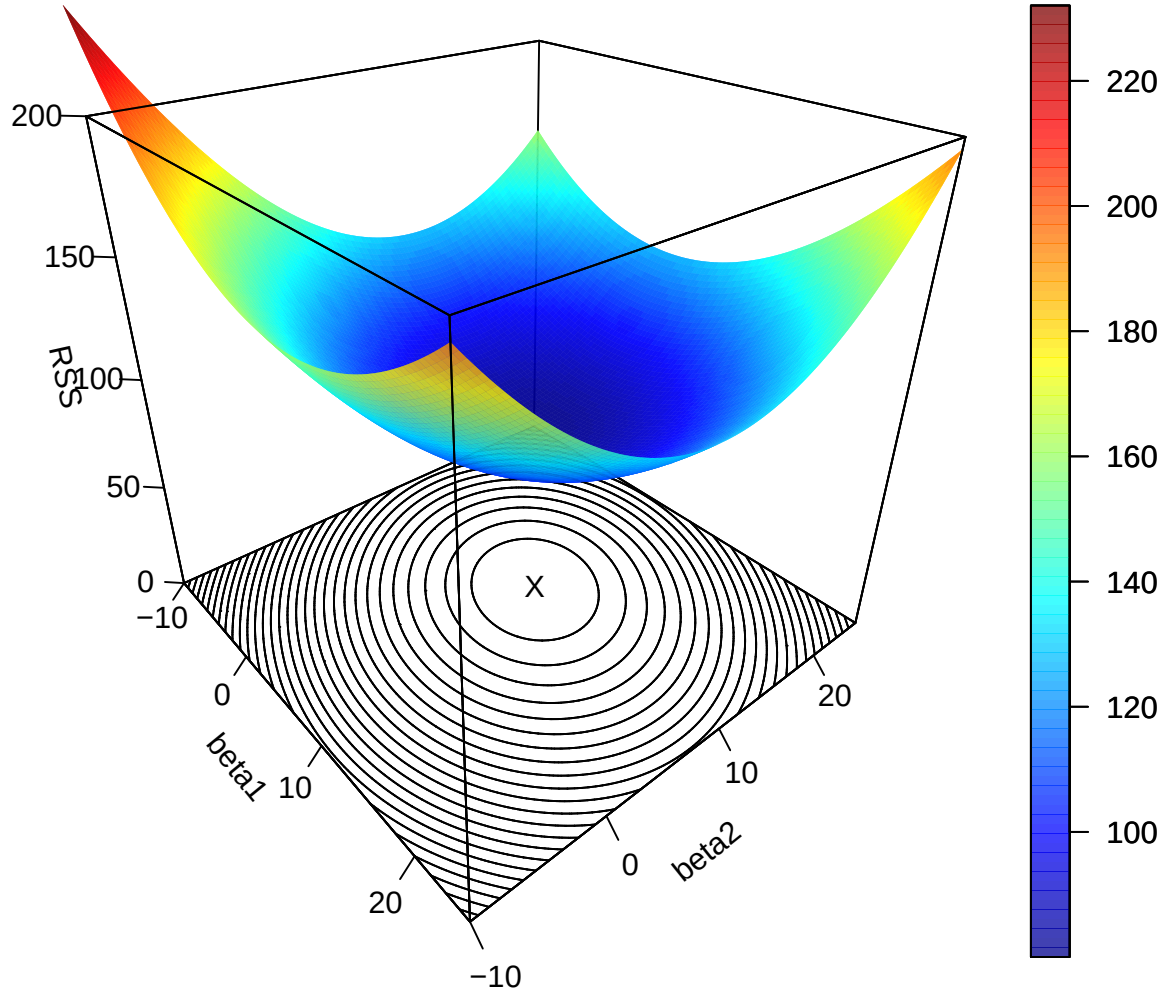
Notice that it depends on all of the data $\rightarrow$ expensive to compute

Instead, we can randomly sample i from $\{1, 2, \ldots, n\}$ and use:

$$\hat{\beta}_j^{k+1} = \hat{\beta}_j^k - \alpha[k](Y_i - X_i^\top \hat{\beta}^k) X_{ij}$$

(We call this stochastic gradient descent)

Gradient descent example



With $RSS = ||Y - \mathbb{X}\beta||_2^2$ for $p = 2$

Gradient descent: Relaxation term

For either batch of stochastic gradient descent, we need to choose $\alpha[k]$

Computing the line search to find α can be annoying/infeasible

(Note that there are other, cheaper alternatives like backtracking line search)

It turns out that the line search α will look like

$$\frac{1}{(\text{max singular value})}$$

of $\nabla^2 F|x$

(That is, the Hessian of F at x)

If this is available, we can use it instead of doing the search

Alternately ...

Gradient descent: Relaxation term

The gradient descent update again:

$$\hat{\beta}_j^{k+1} = \hat{\beta}_j^k - \alpha[k] \sum_{i=1}^{n} (Y_i - X_i^\top \hat{\beta}^k) X_{ij}$$

Two common choices:

- Set $\alpha[k] = \alpha$ to some constant
  (Choose this constant very small)
- Set $\alpha[k] = \alpha/k$ for a large(ish) α

In both cases, check how fast $||\nabla F|_x||_2$ is decreasing. If too slow, increase α

Coordinate descent

As mentioned, glmnet uses a variant of gradient descent: coordinate descent

Result: Suppose $x \in \mathbb{R}^Q$ and $F(x) = g(x) + \sum_{q=1}^{Q} h_q(x_q)$

(With g and h_q convex and g differentiable)

Then finding

$$F(x + be_q) \geq F(x) \text{ for all } q, b \Rightarrow F(x) = \min_{x'} F(x')$$

(Here, e_q is the q^{th} canonical basis vector)

This suggests we can minimize in each coordinate separately

Coordinate descent

$$
\begin{aligned}
x_1^k &= \underset{x_1}{\operatorname{argmin}}\, F(x_1, x_2^{k-1}, \ldots, x_Q^{k-1}) \\
x_2^k &= \underset{x_2}{\operatorname{argmin}}\, F(x_1^k, x_2, \ldots, x_Q^{k-1}) \\
&\vdots \\
x_Q^k &= \underset{x_Q}{\operatorname{argmin}}\, F(x_1^k, x_2^k, \ldots, x_Q)
\end{aligned}
$$

Notes:

- The order of cycling over $\{1, 2, \ldots, Q\}$ is arbitrary
- Can use blocks of coordinates instead

Coordinate descent: Least squares

Let $F(\beta) = \frac{1}{2} ||\mathbb{X}\beta - Y||_2^2$

Minimize over β_j

$$0 = \frac{\partial F}{\partial \beta_j}\bigg|_{\beta} = x_j^\top (\mathbb{X}\beta - Y) = x_j^\top (x_j \beta_j + \mathbb{X}_{(j)}\beta_{(j)} - Y)$$

If we solve this for β_j, we find the coordinate descent update

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*, Statistica Sinica

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

Homrighausen, D. and McDonald, D.J. *Risk estimation for high-dimensional lasso regression*

Choosing the tuning parameter for lasso

Note that for the grid λ, we need only look over the interval

$$\left[0, ||\mathbb{X}^\top Y||_\infty\right) = \left[0, \max_{1 \le j \le p} |x_j^\top Y|\right)$$

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

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

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

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

Alternatively, we can use GIC with:

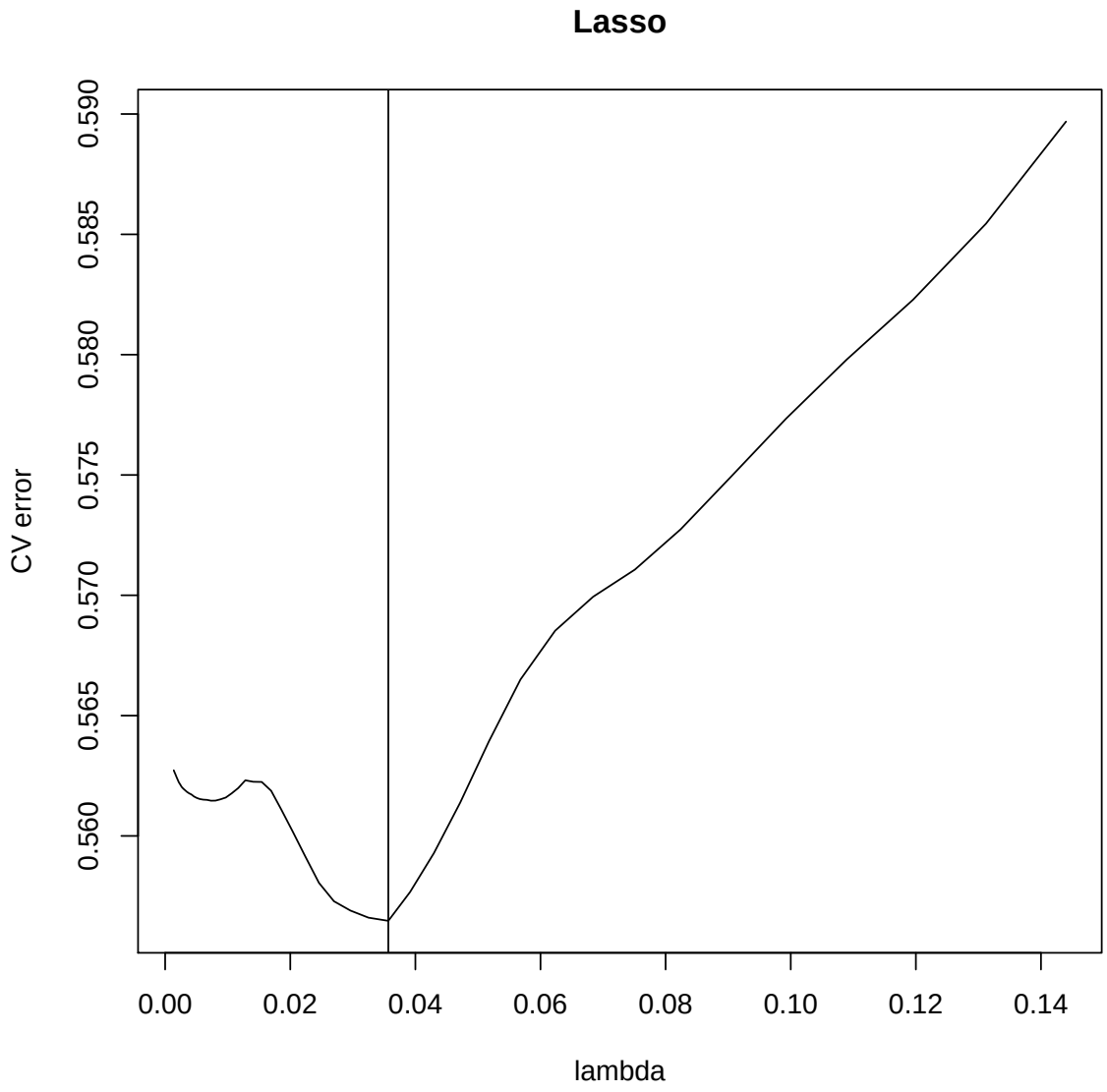
$$\text{df}(\lambda) = \mathbb{E}[\text{rank}(\mathbb{X}_{\mathcal{S}(\lambda)})]$$

where

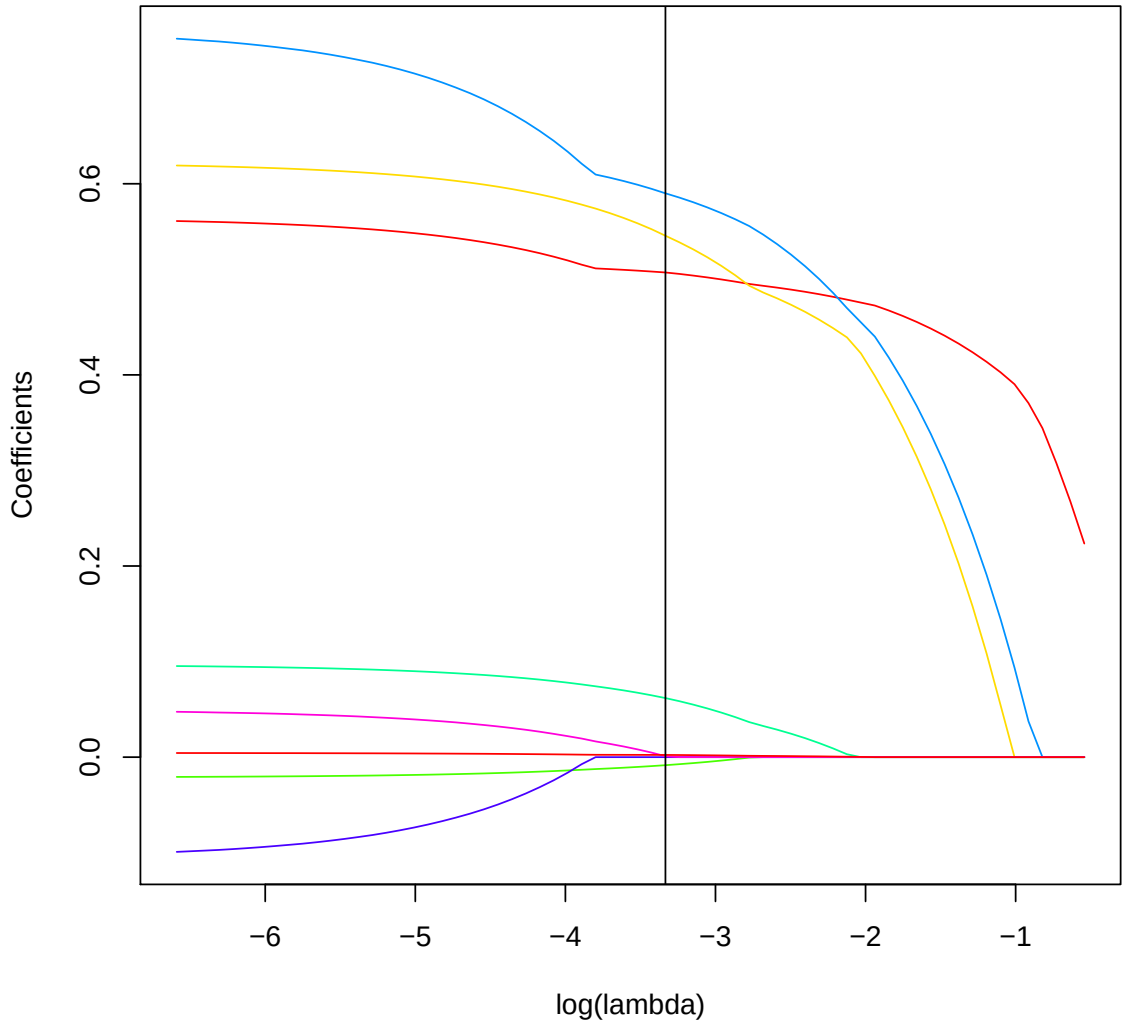
$$\mathcal{S}(\lambda) = \{j : |\hat{\beta}_{j,\text{lasso}}(\lambda)| > 0\}$$

(Tibshirani R.J and Taylor (2012)$^\sharp$)

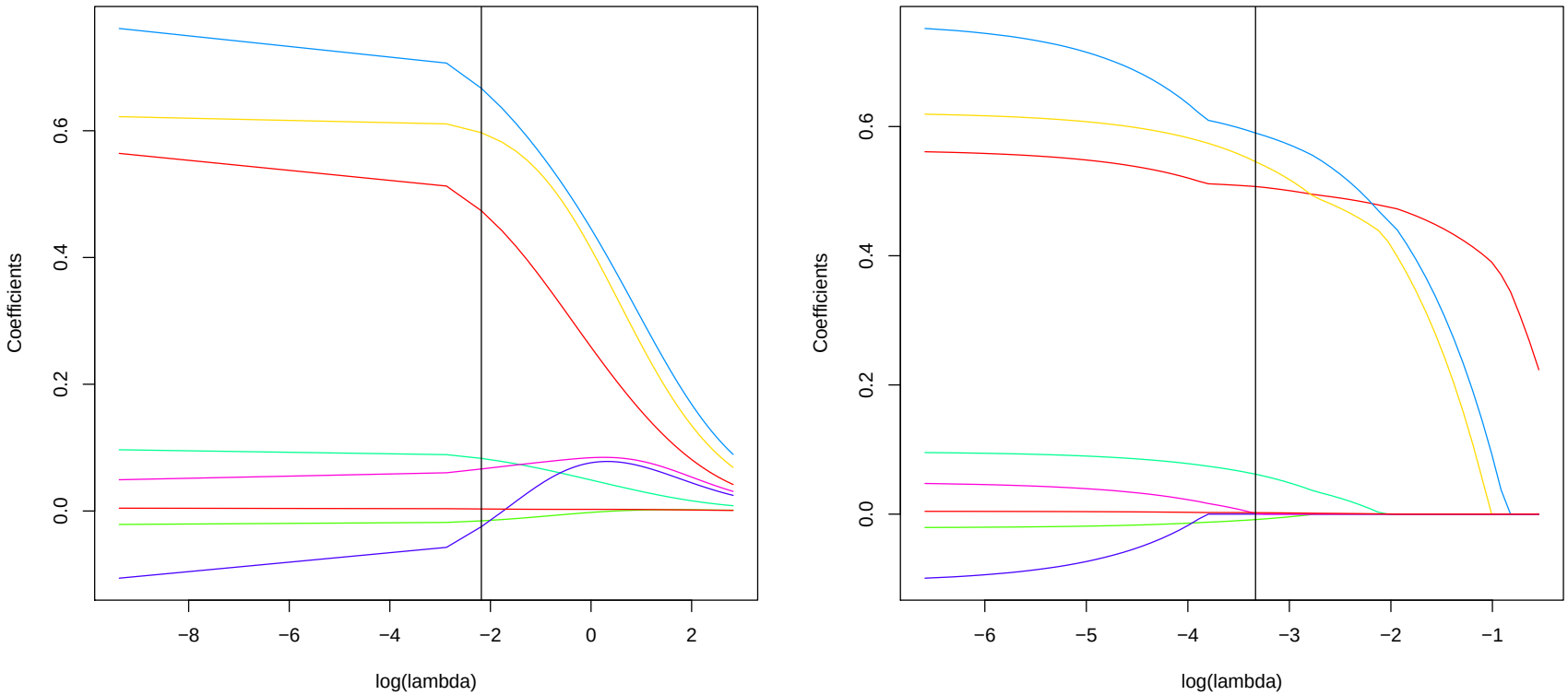
The lasso in R



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[3])

There are two main problems with lars

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

[3] This is not quite true (Tibshirani, R.J. (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 $\mathcal{S} = \{j : |\hat{\beta}_{\lambda,j}| > 0\}$

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 features (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: $||D\beta||_1$).