Model Selection and Tree-Based Techniques

Thiemo Fetzer

University of Warwick & Bonn & CEPR & LSE & NIESR & AEAI

October 23, 2025

Plan

Ridge Regression

Lasso

Elastic Net

Regression Trees

Bagging and Random Forests

Ridge regression

- The reason for high variance in the test error may be due to large variance of some coefficient estimates.
- We can "shrink" them to zero, which can dramatically improve the performance for prediction.
- We first discuss **ridge regression**. The objective function here is to find a vector β_λ, such that the following expression is minimized:

$$argmin_\beta \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge regression

- The reason for high variance in the test error may be due to large variance of some coefficient estimates.
- We can "shrink" them to zero, which can dramatically improve the performance for prediction.
- We first discuss **ridge regression**. The objective function here is to find a vector β_λ, such that the following expression is minimized:

$$argmin_\beta \underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{k=1}^{p} x_{ij}\beta_j)^2}_{\text{Residual Sum of Squares}} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge regression

- The reason for high variance in the test error may be due to large variance of some coefficient estimates.
- We can "shrink" them to zero, which can dramatically improve the performance for prediction.
- We first discuss **ridge regression**. The objective function here is to find a vector β_λ, such that the following expression is minimized:

$$argmin_\beta \underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_k\beta_k)^2}_{\text{Residual Sum of Squares}} + \underbrace{\lambda\sum_{j=1}^{p}\beta_j^2}_{PenalityTerm}$$

Ridge regression [a bit mathy]

$$argmin_\beta \underbrace{\sum_{i=1}^{n}(y_i - \sum_j x_{ij}\beta_j)^2}_{\text{Residual Sum of Squares}} + \underbrace{\lambda \sum_{j=1}^{p}\beta_j^2}_{PenalityTerm}$$

We can rewrite the objective function in matrix notation as:

$$argmin_\beta (y - \mathbf{X}\beta)'(y - \mathbf{X}\beta) + \lambda\beta'\beta$$

- This is very close to the objective function for plain OLS.
- The First Order Conditions are:

$$-2\mathbf{X}'(y - X\beta) + 2\lambda\beta$$

- Setting equal to zero

$$\mathbf{X}'y = (\lambda I + \mathbf{X}'X)\beta$$

- From this follows directly:

$$\beta^R = (\lambda I + \mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'y$$

Ridge regression

- You can see directly that if $\lambda = 0$, then the solution is just OLS.
- As $\lambda \to \infty$, $\beta \to 0$.
- For every value of λ you get a different estiamted coefficient vector.
- Note that we dont have a penality for the intercept β_0, which provides the mean value of the dependent variable y, when all the x_i are zero.
- But how to choose λ, we will explore this further below.

Another way to write down the Ridge regression problem

This problem

$$argmin_{\beta} \underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2}_{\text{Residual Sum of Squares}} + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{PenalityTerm} \quad (1)$$

is the same as

$$min_{\beta} \underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{k=1}^{p} x_{ij}\beta_j)^2}_{\text{Residual Sum of Squares}} \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq s \quad (2)$$

- This is a constrained optimization problem, which you should know from Microeconomics (maximize utility subject to a budget constraint)

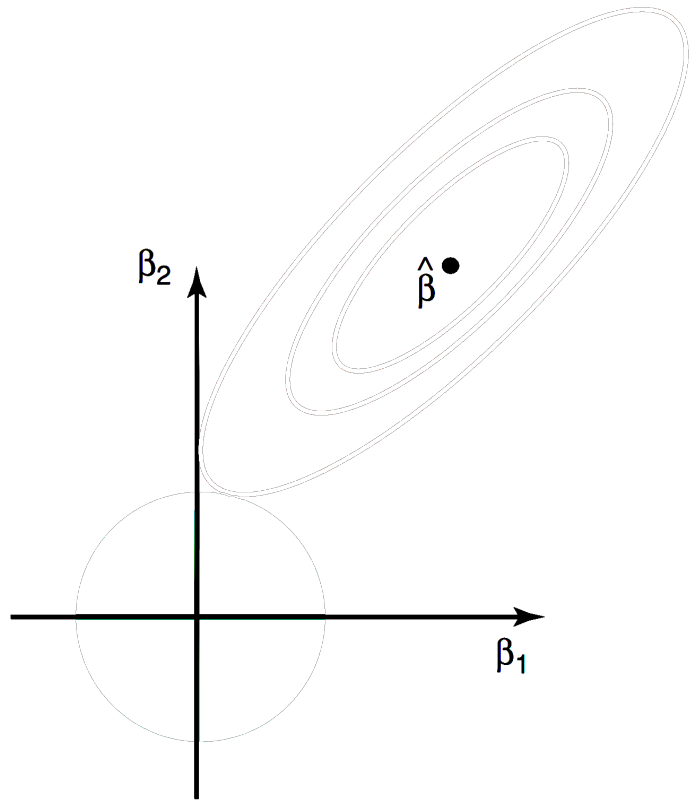
Some graphical intuition for $p = 2$

Suppose you only have $p = 2$, i.e. there are two features

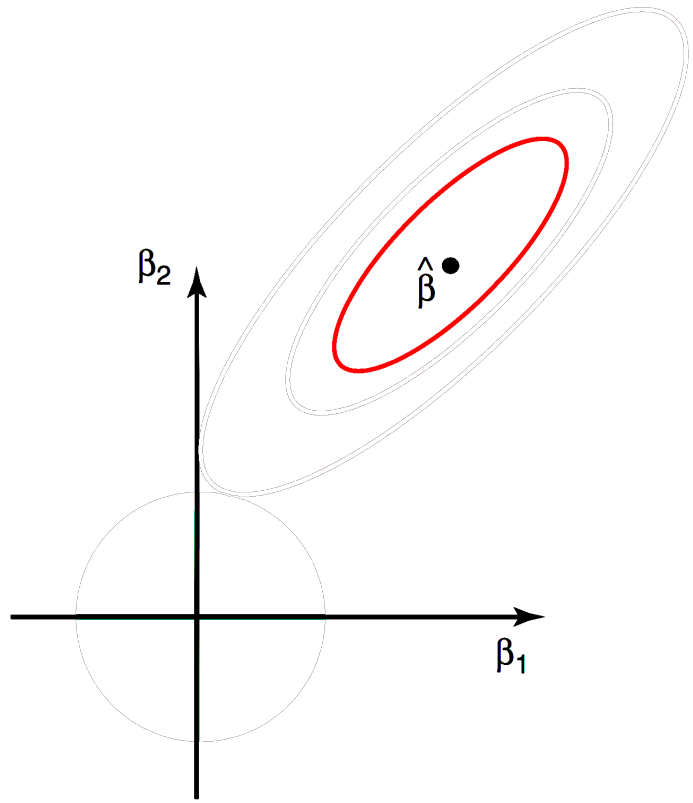
$$min_{\beta} \underbrace{\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2})^2}_{\text{Residual Sum of Squares}} \text{ subject to } \beta_1^2 + \beta_2^2 \leq s \quad (3)$$

What shape does the objective function and the constraint have?

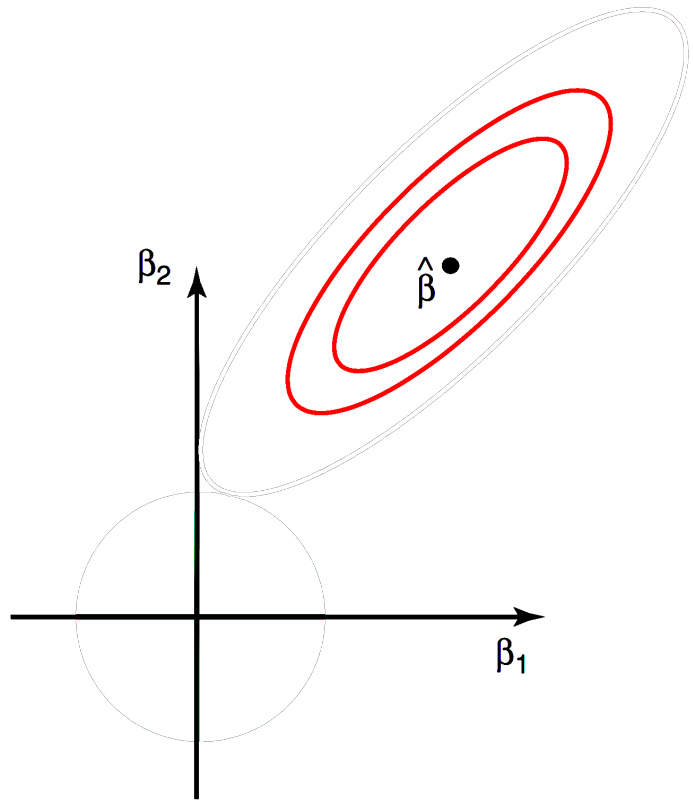
Some graphical intuition for $p = 2$



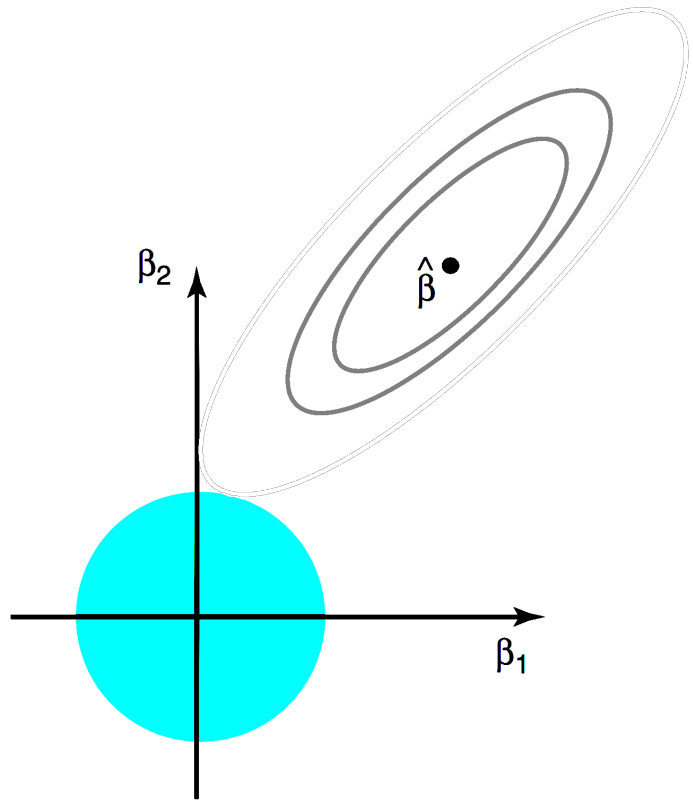
Some graphical intuition for $p = 2$



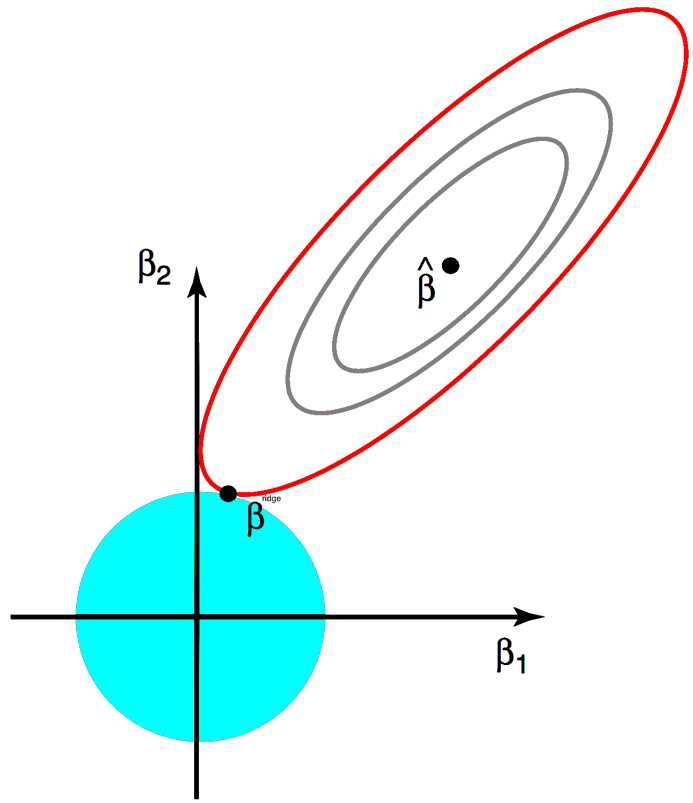
Some graphical intuition for $p = 2$



Some graphical intuition for $p = 2$



Some graphical intuition for $p = 2$



Ridge regression is not scale invariant

- *Remember*: The standard OLS coefficient estimates are scale equivariant: multiplying x_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$. In other words, regardless of how the jth predictor is scaled, $X_j\hat{\beta}_j$ will remain the same.
- For ridge regression, this is not the case since the β's are directly in the penality term.
- You dont want to penalize coefficients that are large due to the scaling of the underlying x variable.
- We need to make the different β_j's comparable, which we can do easily by standardizing them.

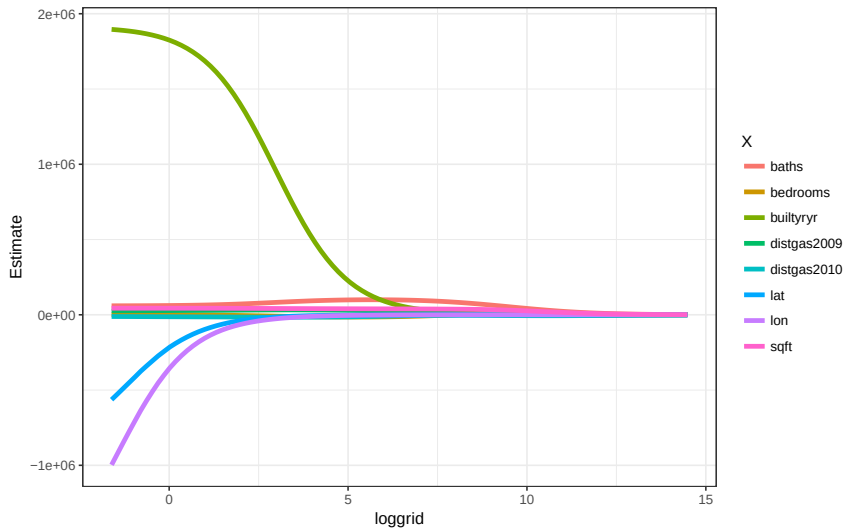
Ridge regression is not scale invariant

- We can scale the x_j by its standard deviation, then all the x_ij values are expressed in terms of standard deviations.
- i.e. the common unit is a standard deviation.
- Divide the value by an estimate of the standard deviation, which is:

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}}$$

Shrinking hedonic pricing coefficients?

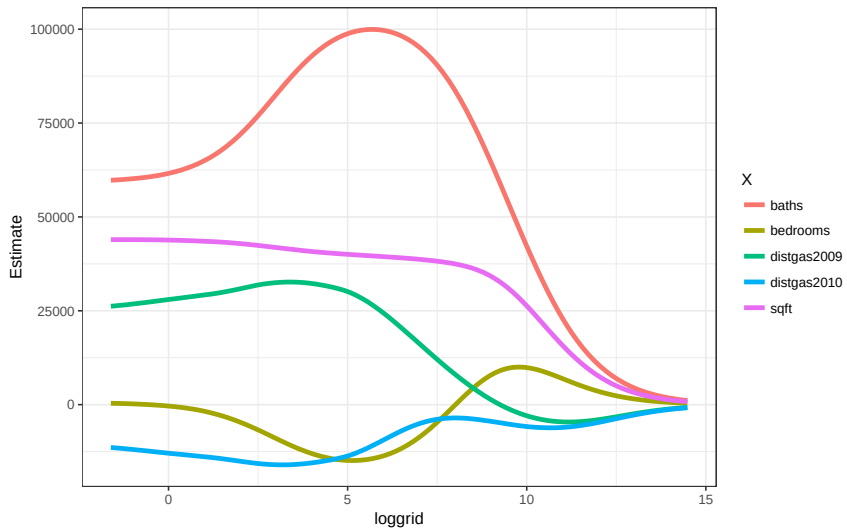
Lets plot some of the coefficients as they evolve with λ increasing.



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $log(\lambda)$. A large value means a high value of λ, which imply significant shrinking of coefficients. Due to the very large values that λ takes in this example, I used a logaritmic transformation to condense the picture.

Shrinking hedonic pricing coefficients?

Lets plot some of the coefficients as they evolve with λ increasing.



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $log(\lambda)$. A large value means a high value of λ, which imply significant shrinking of coefficients. Due to the very large values that λ takes in this example, I used a logaritmic transformation to condense the picture.

Shrinking hedonic pricing coefficients?

An alternative way to plot these is to express the penality term relative to the benchmark OLS, i.e. to normalize the horizontal axis to range from 0 - 1, where 1 corresponds to OLS.

$$\frac{\|\hat{\beta}_\lambda\|_2}{\|\hat{\beta}_{OLS}\|_2} \tag{4}$$

where

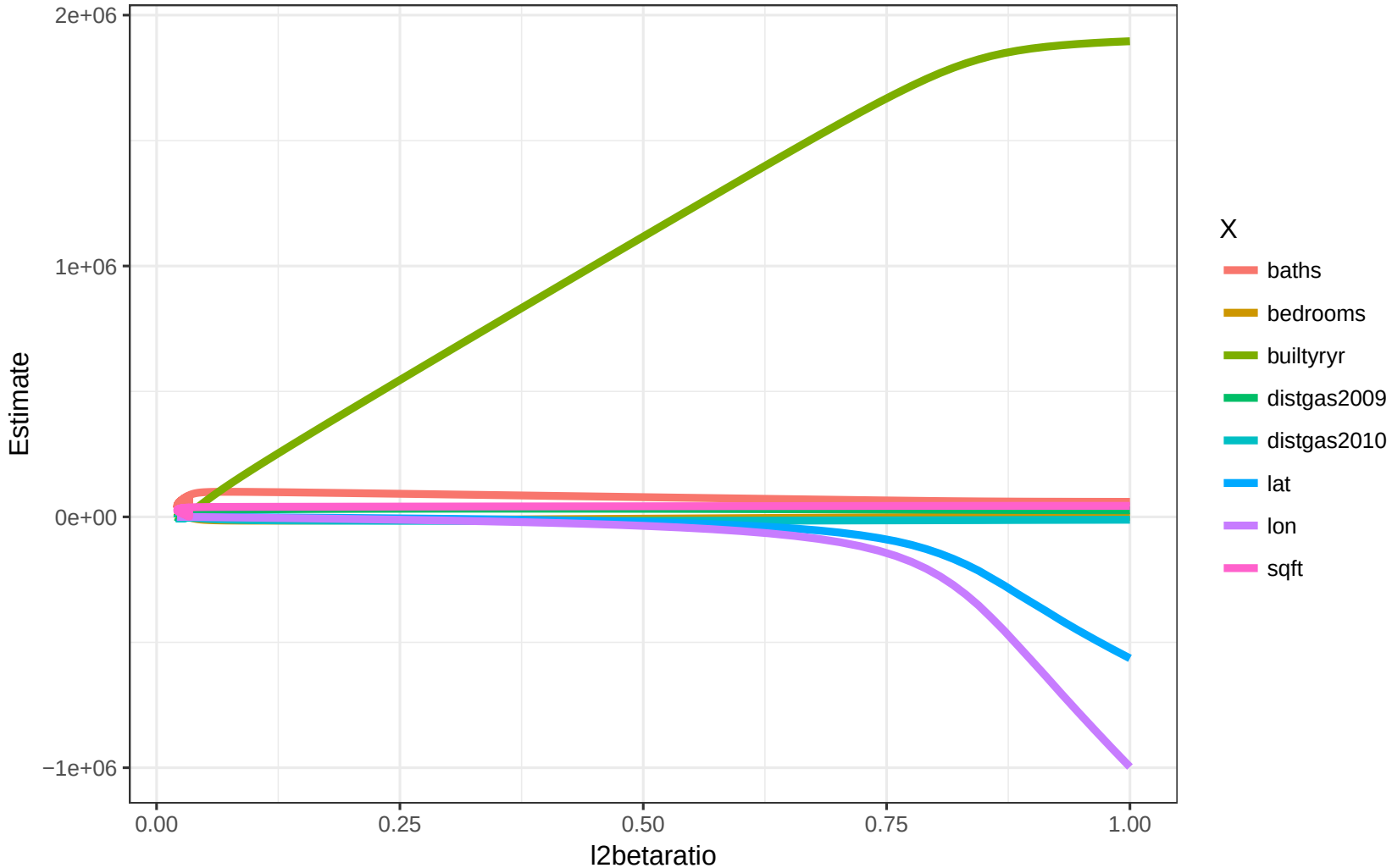
$$\|\beta\|_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$$

This is called the $\mathcal{L}_2$ norm, which measures the Euclidian distance from origin. For $p = 2$, this is: $\|\beta\|_2 = \sqrt{\beta_1^2 + \beta_2^2}$.

So when $\lambda \to 0$, $\frac{\|\hat{\beta}_\lambda\|_2}{\|\hat{\beta}_{OLS}\|_2} \to 1$.

Shrinking hedonic pricing coefficients?

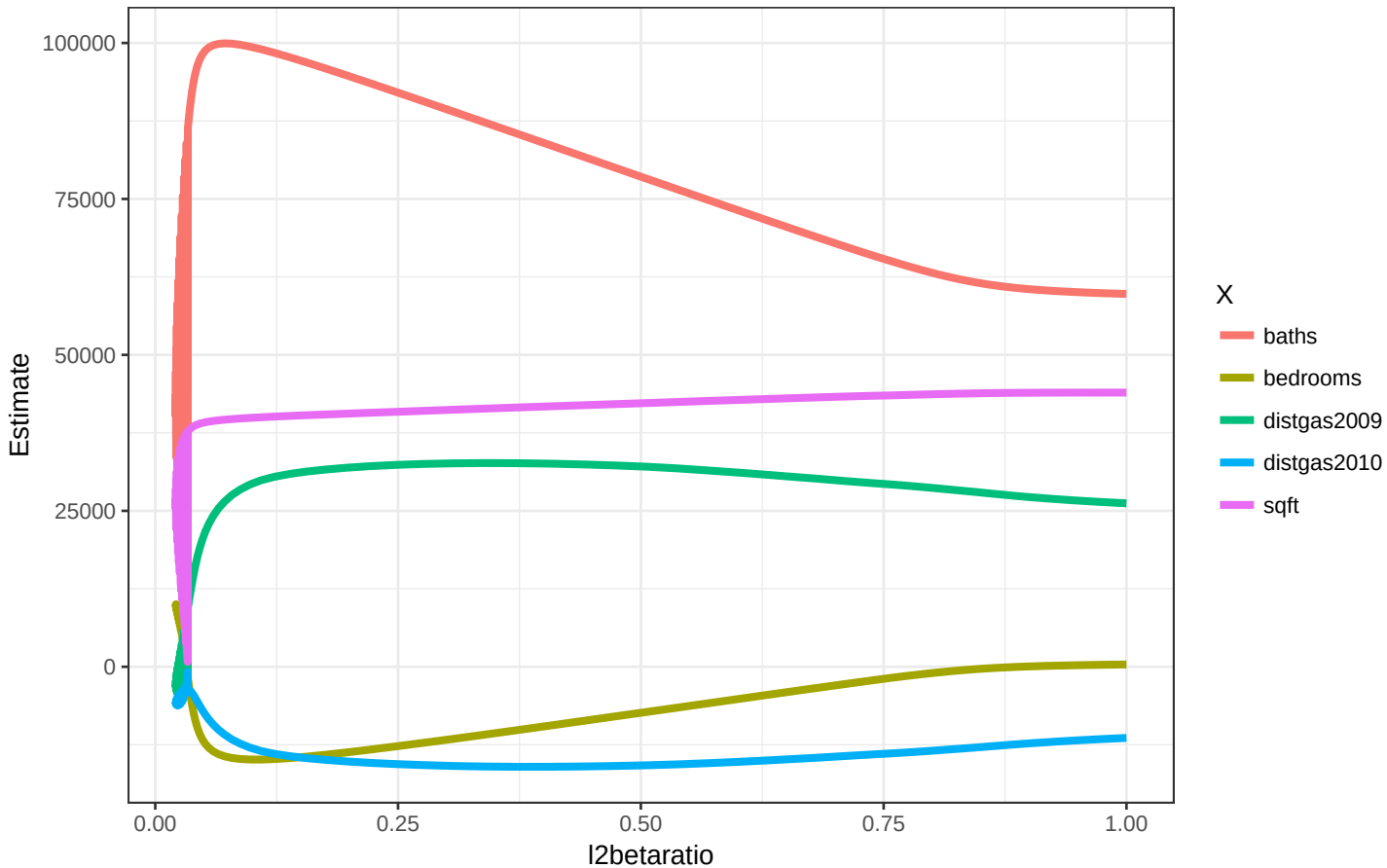
An alternative way to plot these is to express the penality term relative to the benchmark OLS, i.e. to compute:



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $\frac{\|\hat{\beta}_\lambda\|_2}{\|\hat{\beta}_{OLS}\|_2}$. The closer the value is to 1 on the horizontal axis, the closer we get to the OLS solution.

Shrinking hedonic pricing coefficients?

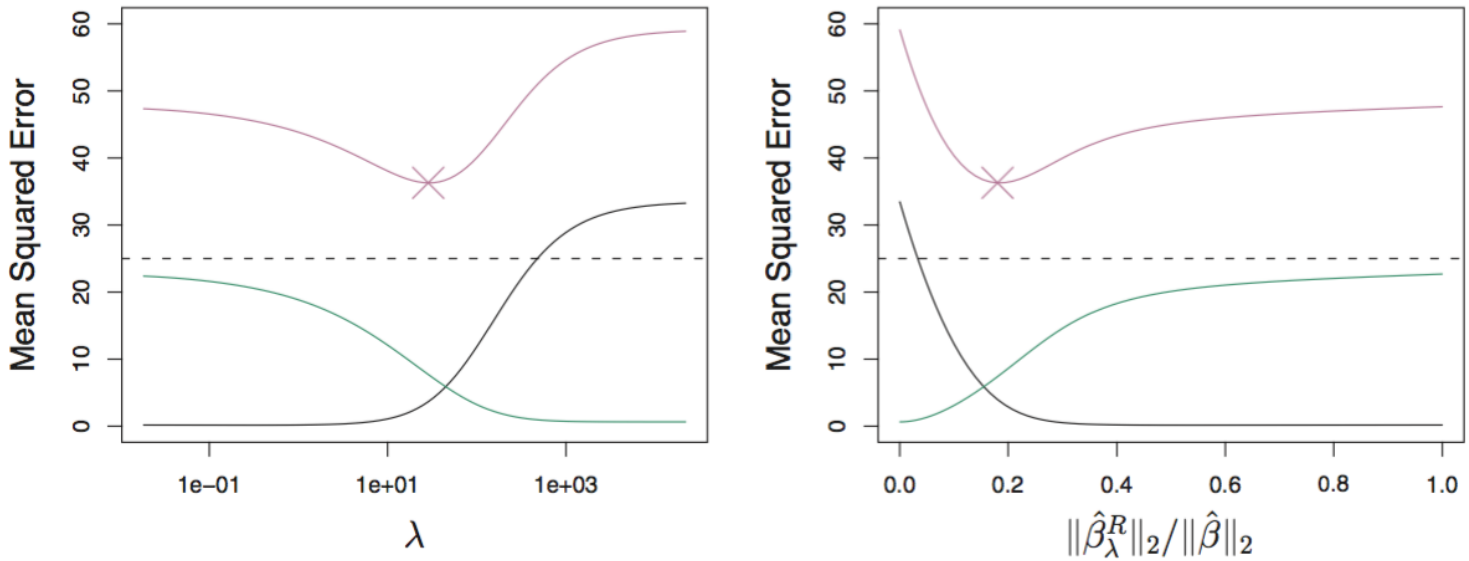
An alternative way to plot these is to express the penality term relative to the benchmark OLS, i.e. to compute:



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $\frac{\|\hat{\beta}_\lambda\|_2}{\|\hat{\beta}_{OLS}\|_2}$. The closer the value is to 1 on the horizontal axis, the closer we get to the OLS solution.

What do we gain using Ridge regression?

- This course revolves around the bias variance tradeoff.
- Ridge regression, by shrinking coefficients, helps reduce the variance of the prediction and thus, reduces prediction error.



What do we dislike about Ridge regression?

- As opposed to subset selection, ridge regression does not actually result in simpler models.
- Some estimated coefficients are simply reduced in their absolute value, but we dont get completely rid of irrelevant predictors as we did wit subset selection.
- Ideally, we want to allow for coefficients to be exactly equal to zero.
- This can be achieved, if we make the optimization problem slightly more complicated.

Plan

Ridge Regression

Lasso

Elastic Net

Regression Trees

Bagging and Random Forests

Lasso

- As opposed to subset selection, ridge regression does not actually result in simpler models.
- Some estimated coefficients are simply reduced in their absolute value, but we dont get completely rid of predictors as we did wit subset selection.
- Ideally, we want to allow for coefficients to be exactly equal to zero.
- This can be achieved, if we make the optimization problem slightly more complicated.

Lasso optimization problem

$$argmin_\beta \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \mid \beta_j \mid$$

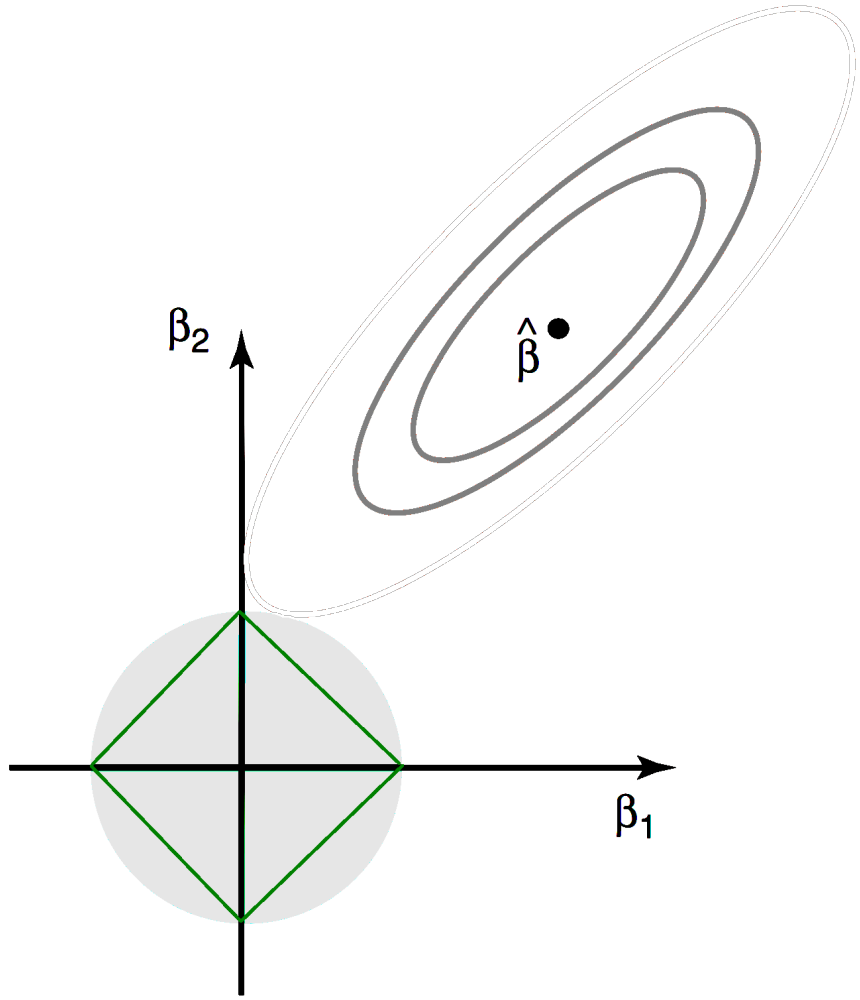
- Again, this can be rewritten as:

$$min_\beta \underbrace{\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2}_{\text{Residual Sum of Squares}} \text{ subject to } \sum_{j=1}^{p} \mid \beta_j \mid \leq s \quad (5)$$

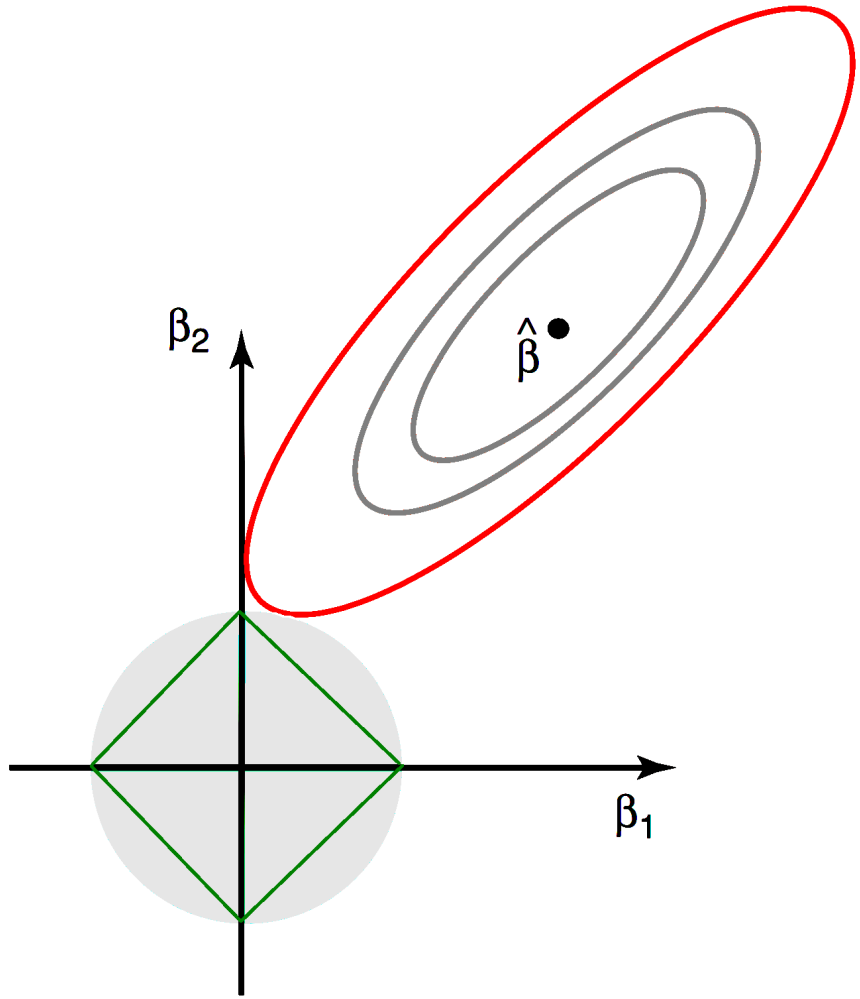
- The only thing that changes is the shape of the "circle".
- The constraint in absolute values is also called the $\mathcal{L}_1$ norm of the vector β, i.e.

$$\|\beta\|_1 = \sum_{k=1}^{p} \mid \beta_k \mid$$

Some graphical intuition for $p = 2$



Some graphical intuition for $p = 2$

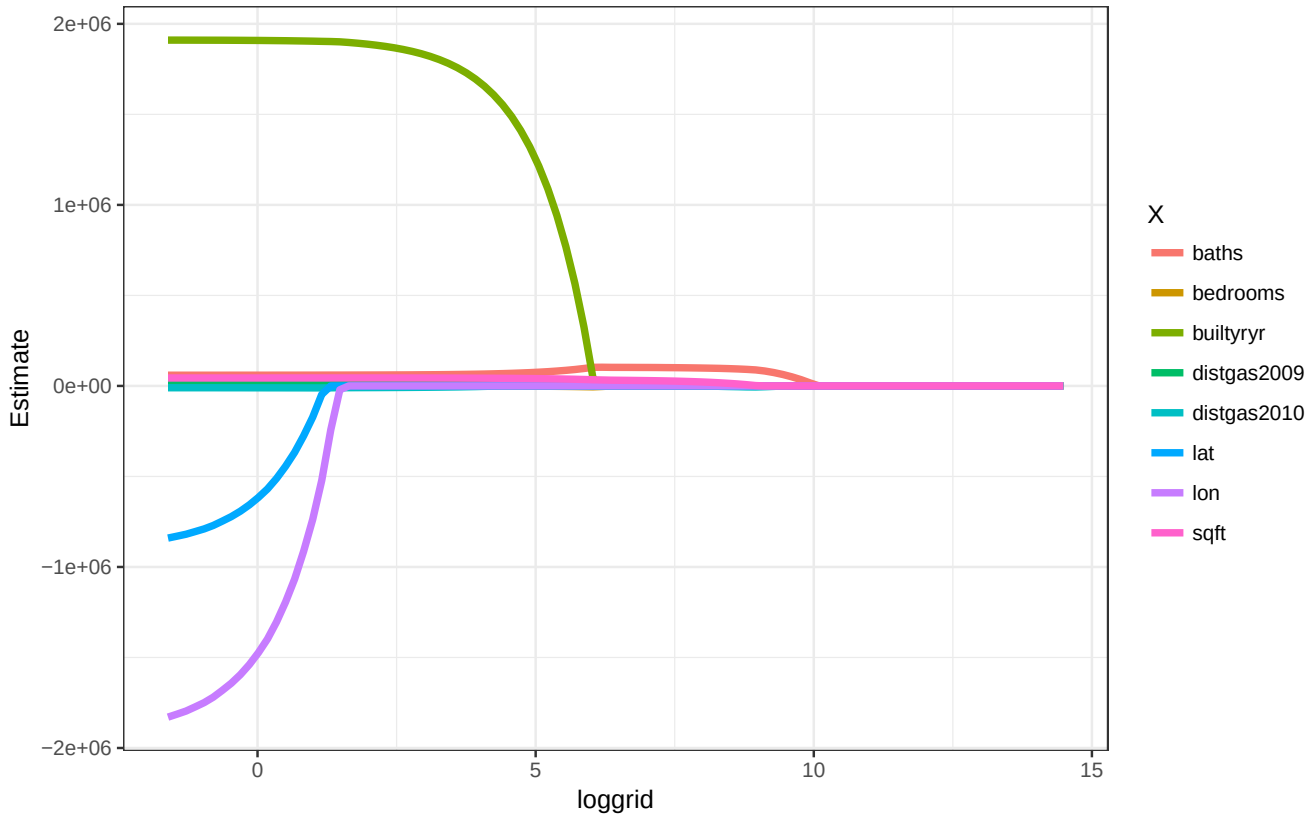


Lasso selects variables...

- Lasso shrinks some variables to be exactly equal to zero, rather than just close to zero.
- This variable selection method makes the Lasso particularly attractive.
- In particular with a large number of features p, the Lasso can help to significantly reduce the dimensionality of the matrix X.
- Lets now apply the Lasso to our example on Housing prices.

Shrinking hedonic pricing coefficients?

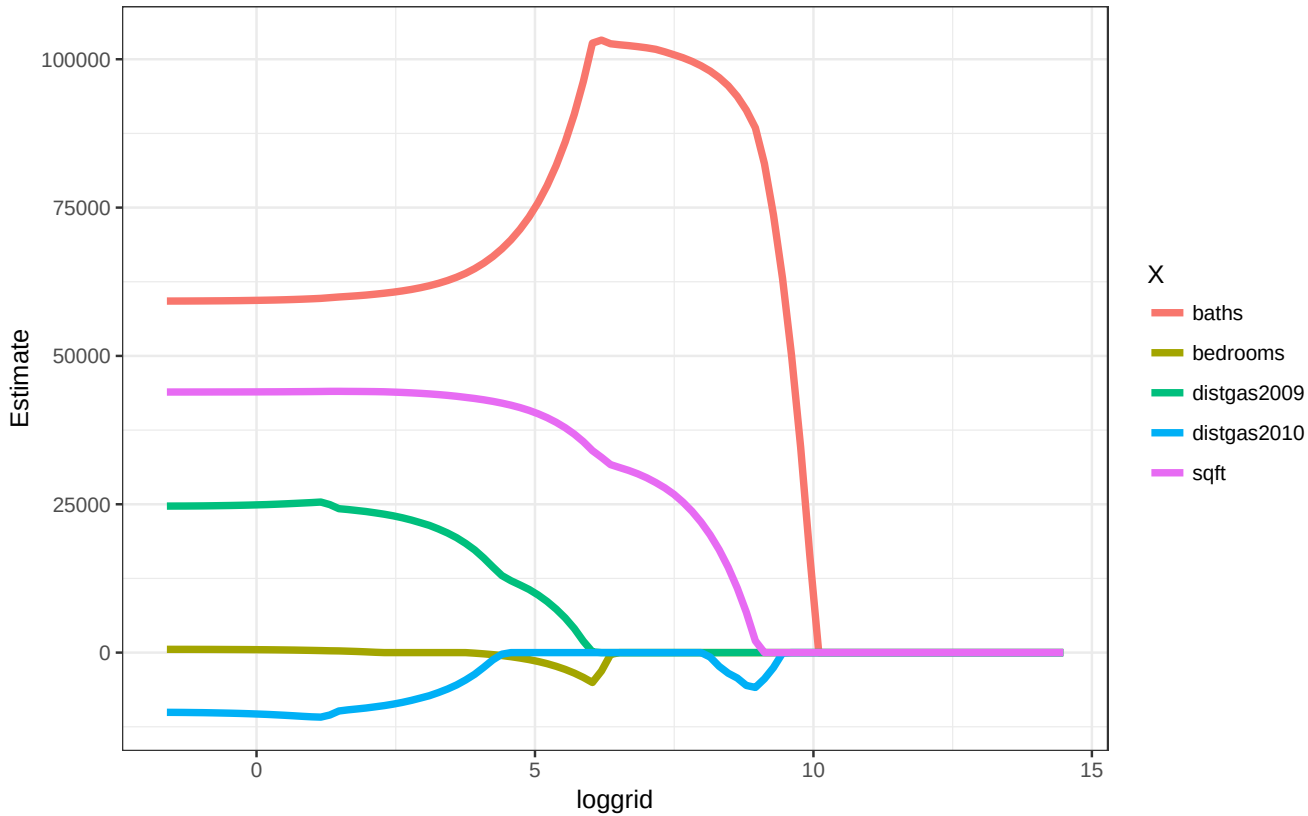
Lets plot some of the coefficients as they evolve with λ increasing.



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $log(\lambda)$. A large value means a high value of λ, which imply significant shrinking of coefficients. Due to the very large values that λ takes in this example, I used a logaritmic transformation to condense the picture.

Shrinking hedonic pricing coefficients?

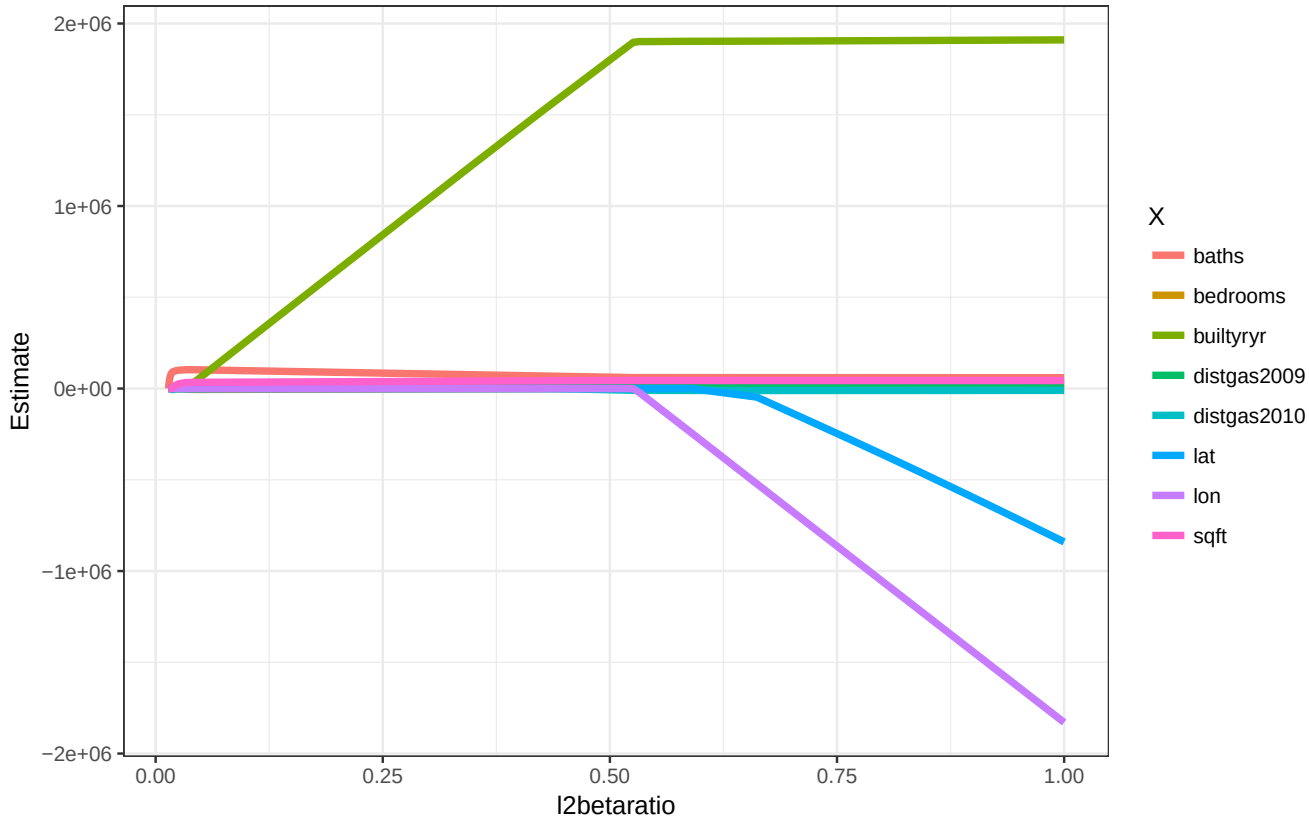
Lets plot some of the coefficients as they evolve with λ increasing.



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $log(\lambda)$. A large value means a high value of λ, which imply significant shrinking of coefficients. Due to the very large values that λ takes in this example, I used a logaritmic transformation to condense the picture.

Shrinking hedonic pricing coefficients?

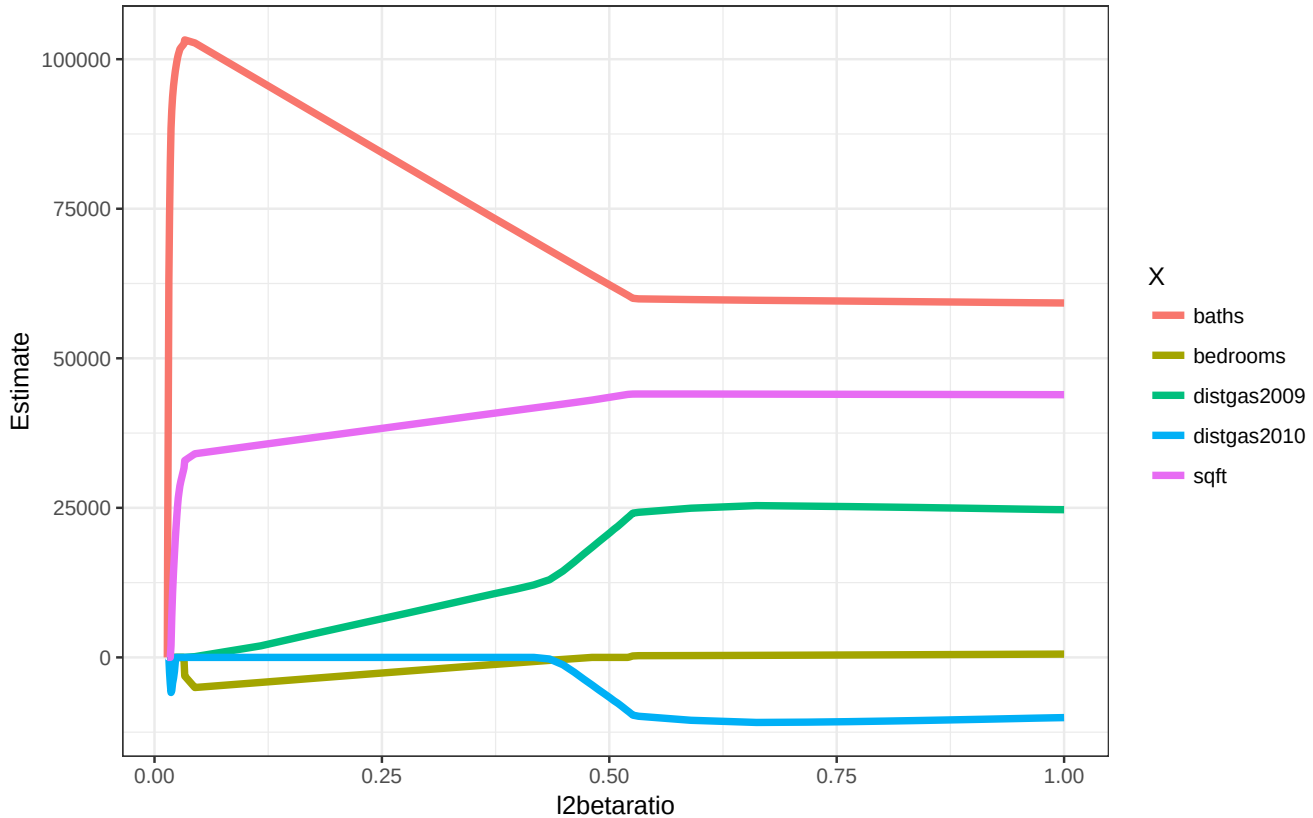
Lets plot some of the coefficients as they evolve with λ increasing.



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $\frac{\|\hat{\beta}_\lambda\|_1}{\|\hat{\beta}_{OLS}\|_1}$.

Shrinking hedonic pricing coefficients?

Lets plot some of the coefficients as they evolve with λ increasing.



The vertical axis presents the estimated coefficients β_{i_λ}, while the horizontal axis plots out $\frac{\|\hat{\beta}_\lambda\|_1}{\|\hat{\beta}_{OLS}\|_1}$.

Lasso or Ridge regression?

- Neither ridge regression nor the lasso will universally dominate the other.
- Lasso performs generally better in a situation, where a relatively small number of predictors have substantial coefficients, and the remaining predictors have coefficients that are very small or that equal zero.
- Ridge regression will perform better when the response is a function of many predictors, the coefficients of which are not very dissimilar from one another.
- Lasso and Ridge regression can yield a reduction in variance at the expense of a small increase in bias
- There are very efficient algorithms for fitting both ridge and lasso models; in both cases the entire coefficient paths can be computed with about the same amount of work as a single least squares fit

Lasso for variable selection illustrated

I will present two use cases that may be illustrative.

- In a previous lecture, I was showing the UK EU Referendum in the context of building a robust predictive exercise - we used BSS, but LASSO is an approximation of the BSS algorithm.
- As indicated, with $p > 40$, BSS becomes infeasible very fast.
- In the paper we used a blocked BSS and combined regressors carrying similar variation to navigate the computational constraint. How does LASSO perform?
- In total there are 76 continuous regressors...

Get the data `https://www.dropbox.com/s/3fxh0a2n2lca26b/Becker_Fetzer_Novy_2017_EP.zip?dl=1`

Lasso for Brexit

```
#to load data from other formats, such as stata data files
library(haven)
library(glmnet)
DAT<-data.table(read_dta("R/brexitevote-ep2017.dta"))
#remove irrelevant variable
DAT<-DAT[, setdiff(names(DAT),"BREXITINDEXNARROW"),with=F]

#we need to normalize the variables to have unit standard deviation
#inspect the data frame and see the variables are in columns
DAT<-DAT[, c(13,15:91), with=F]

dim(DAT)

## [1] 382  78

#missing values for two areas
DAT<-DAT[!is.na(Pct_Leave)]

#remove columns with any missing values
DAT<-DAT[, which(colSums(is.na(DAT))==0), with=F]
dim(DAT)

## [1] 380  65

class(DAT)

## [1] "data.table" "data.frame"

#standardize, scale and center
DAT<-data.table(scale(DAT,center=TRUE))

#create a model matrix for regression
x=model.matrix(Pct_Leave ~ ., data=DAT)[,-1]
x[1:5, 1:4]

##   EU75Leaveshare pensionergrowth20012011 ResidentAge30to44share ResidentAge45to59share
## 1          0.249                -0.55928              -0.077721                -0.42167
## 2          0.249                -1.13089              -0.386752                -1.00202
## 3          0.249                -0.38467              -0.548088                 0.32559
## 4          0.249                 0.00712               0.407054                -0.15890
## 5          0.847                -0.28806               0.000242                 0.00298

class(x)

## [1] "matrix" "array"
```

Lasso for Brexit

```
#to load data from other formats, such as stata data files

library(glmnet)

#run lasso
lasso.mod=glmnet(x,DAT$Pct_Leave,alpha=1, standardize=TRUE, family="gaussian")

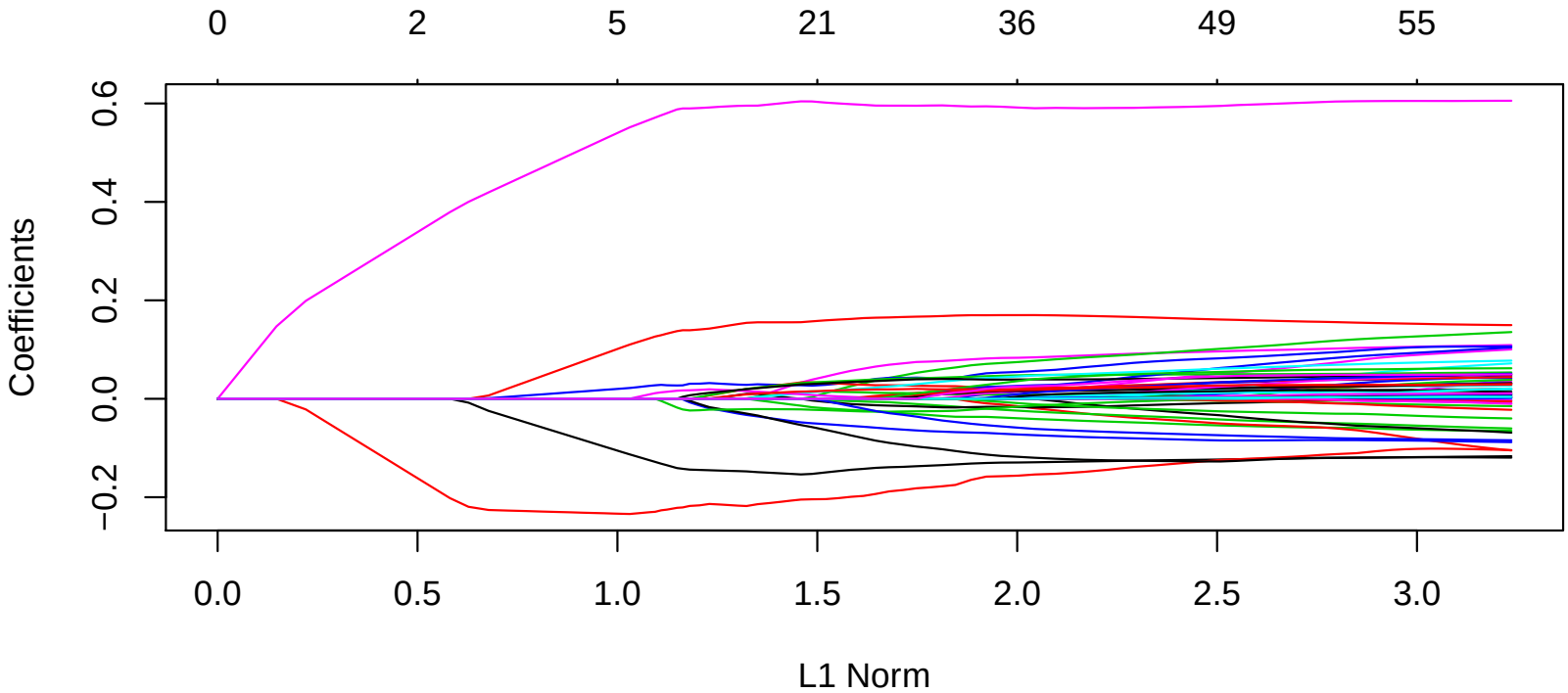

summary(lasso.mod)

##           Length Class     Mode
## a0          91   -none-    numeric
## beta      5824   dgCMatrix S4
## df          91   -none-    numeric
## dim          2   -none-    numeric
## lambda      91   -none-    numeric
## dev.ratio   91   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         6   -none-    call
## nobs         1   -none-    numeric
```

Lasso for Brexit

```
#to load data from other formats, such as stata data files

plot(lasso.mod)
```



Plan

Ridge Regression

Lasso

Elastic Net

Regression Trees

Bagging and Random Forests

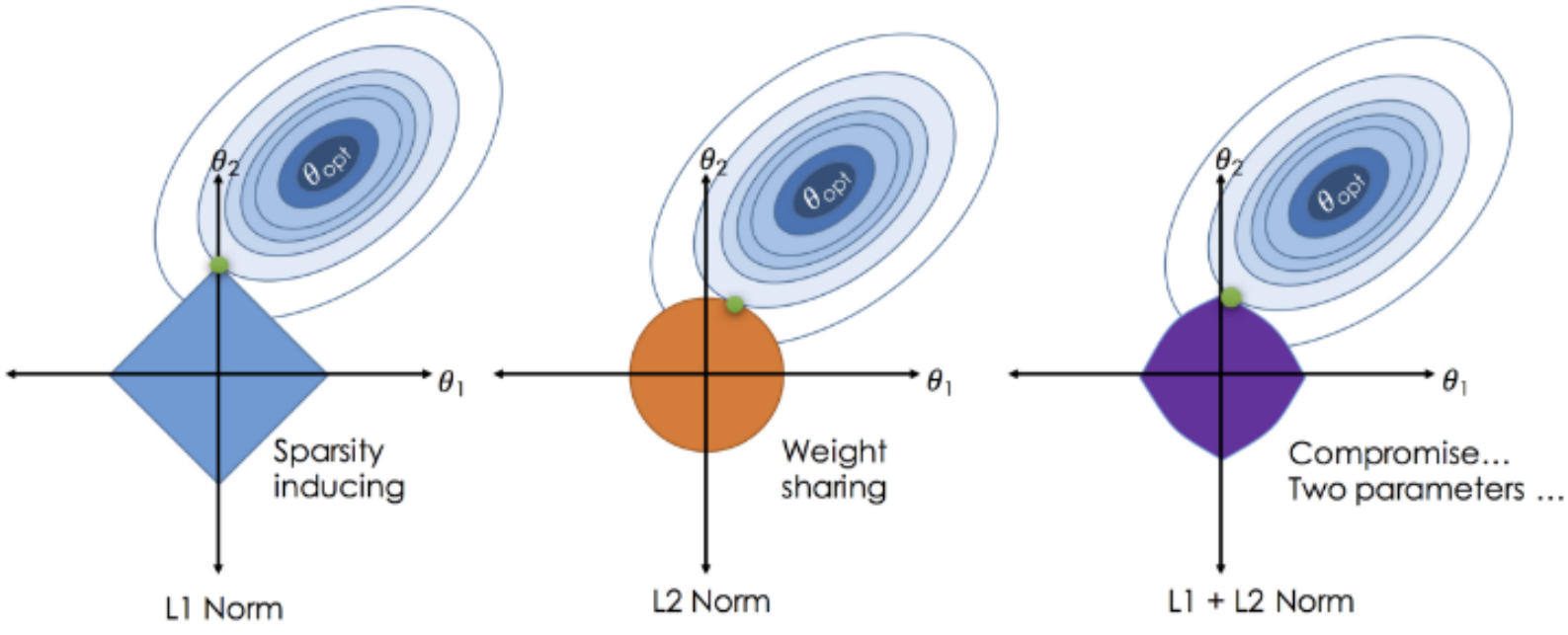
Elastic Net

- An elastic net combines L1 and L2 penalization
- The objective function becoms

$$argmin_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda[\alpha \sum_{j=1}^{p} \mid \beta_j \mid -(1-\alpha) \sum_{j=1}^{p} \beta_j^2]$$

- This now has two tuning parameters to select – the weight on the penalization form α and the λ to navigate the bias-variance trade-off.

Elastic Net


θ2
θ opt
θ1
Sparsity inducing
L1 Norm
θ2
θ opt
θ1
Weight sharing
L2 Norm
θ2
θ opt
θ1
Compromise... Two parameters ...
L1 + L2 Norm

Plan

Ridge Regression

Lasso

Elastic Net

Regression Trees

Bagging and Random Forests

Regression Trees

- Linear regression is a global model, where there is a single predictive formula holding over the entire data-space. Very difficult to capture or adequately model non-linearities, e.g. interactions.
- The idea of Regression Trees is to sub-divide, or partition, the feature space into smaller regions, where the interactions are explicit.
- Divide the feature space, that is the $\mathbf{X}$ into J distinct regions that do not overlap, call them $R_1, ..., R_J$.
- For every test observation x_j that falls into a region R_i, the prediction $\hat{f}(x_j)$ is simply the mean of the training observations y_i that fall into the region R_i

$$\hat{y}_{R_j} = 1/n_j \sum_{i \in R_j} y_i$$

where n_j is number of training observations that fall into R_j.

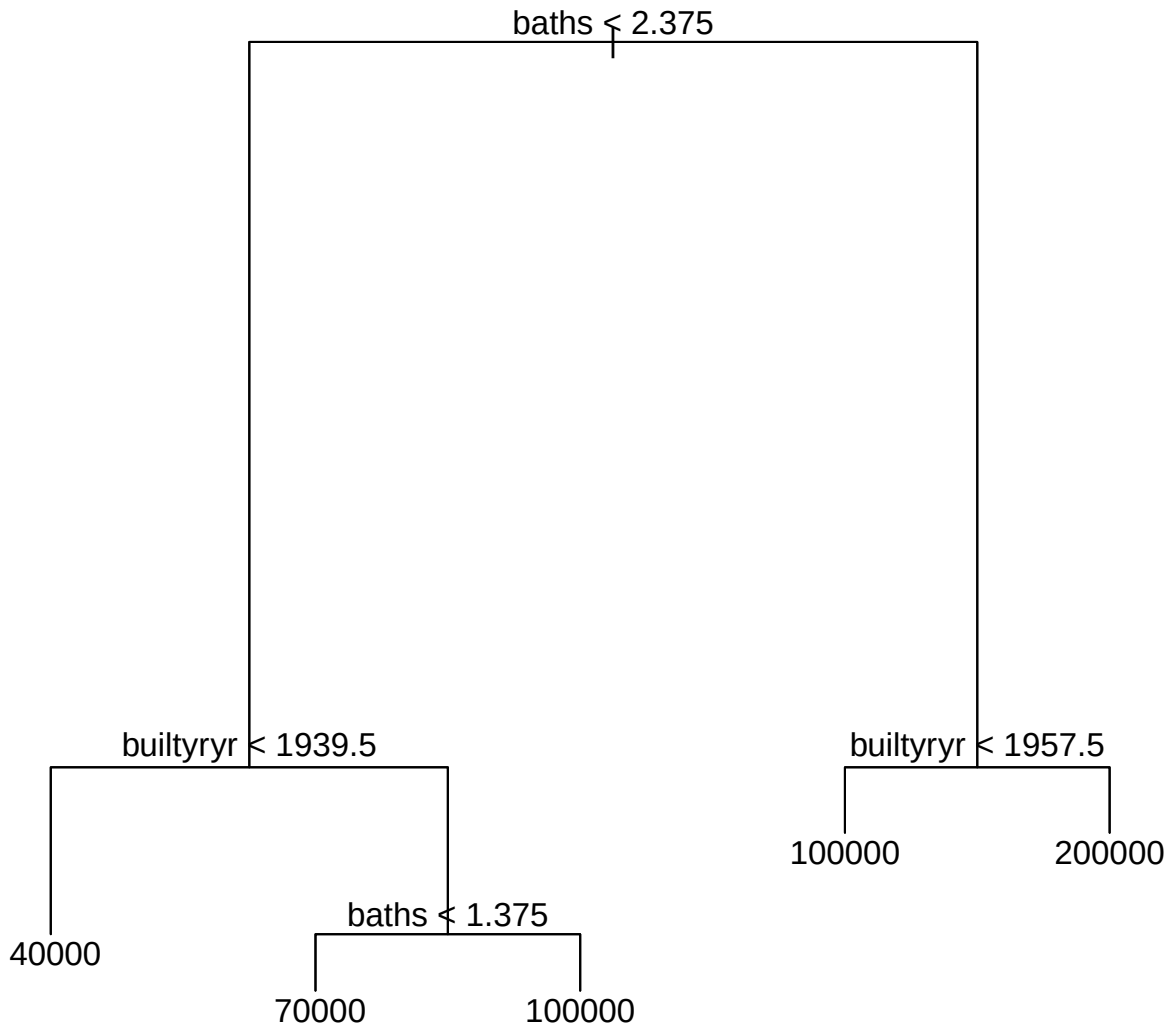
Regression Trees: Hedonic Pricing Example

```
library(tree)
set.seed(1312)
SAMPLE<-sample(1:nrow(HOUSES),18000)
TRAINING<-HOUSES[SAMPLE]
res<-tree(soldprice~builtyryr+baths, data=TRAINING)
plot(res)
text(res)

## Warning in text.tree(res): 'digits' must be positive, so increased to 1
```



Regression Trees: Hedonic Pricing Example



Regression Trees can generate non-linear models

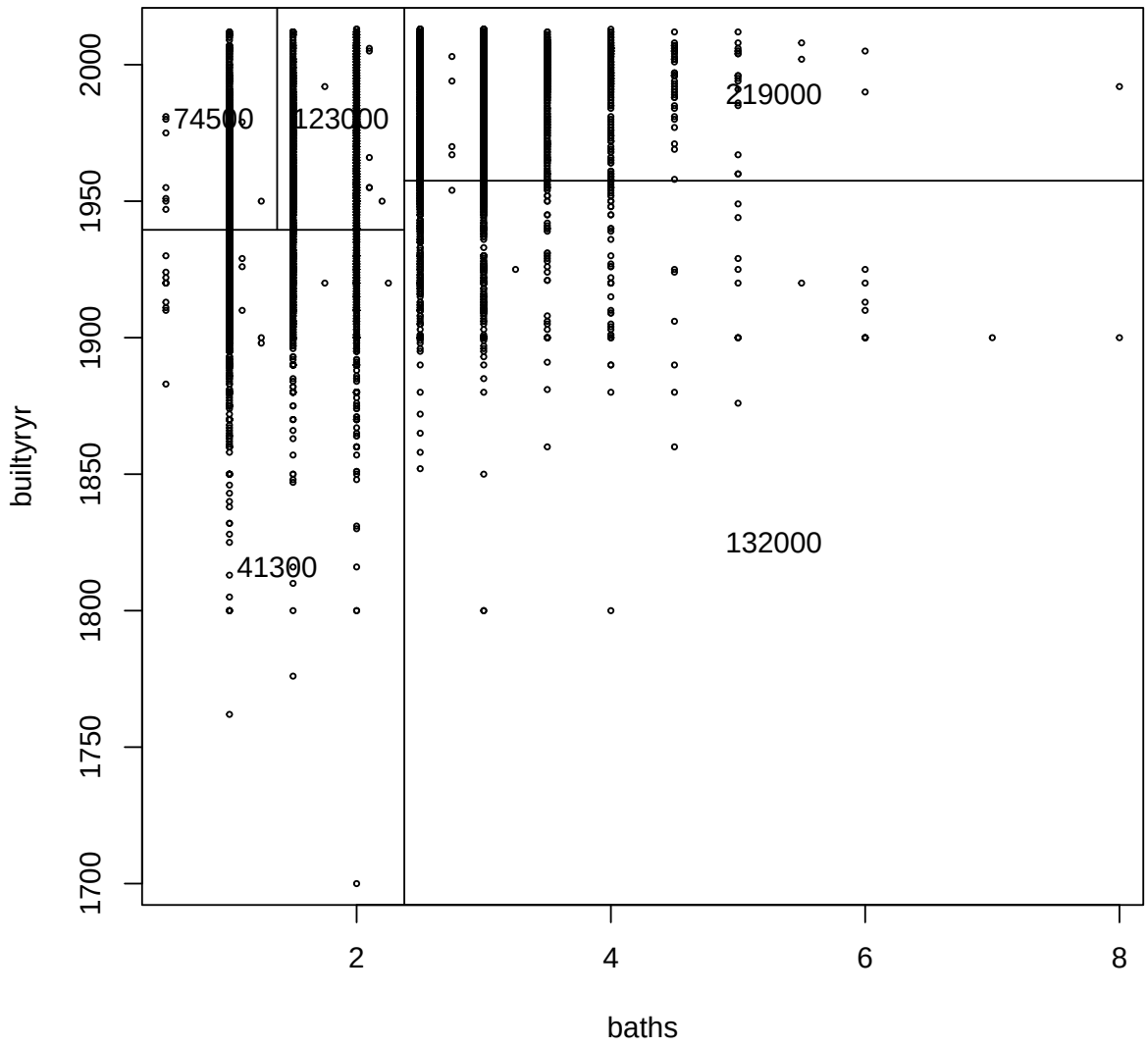
- In the above example, its easy to show that we have interactive effects.
- We can, in fact, write the tree as a sequence of and interaction between different indicator functions. We have X_1 = Bathrooms, X_2 = Built Year

$$I_1 = \mathbf{I}(X_1 \leq 2.375)$$
$$I_2 = \mathbf{I}(X_1 \leq 1.375)$$
$$I_3 = \mathbf{I}(X_2 \leq 1939.5)$$
$$I_4 = \mathbf{I}(X_2 \leq 1957.5)$$

- Our regression tree is a model of the form:

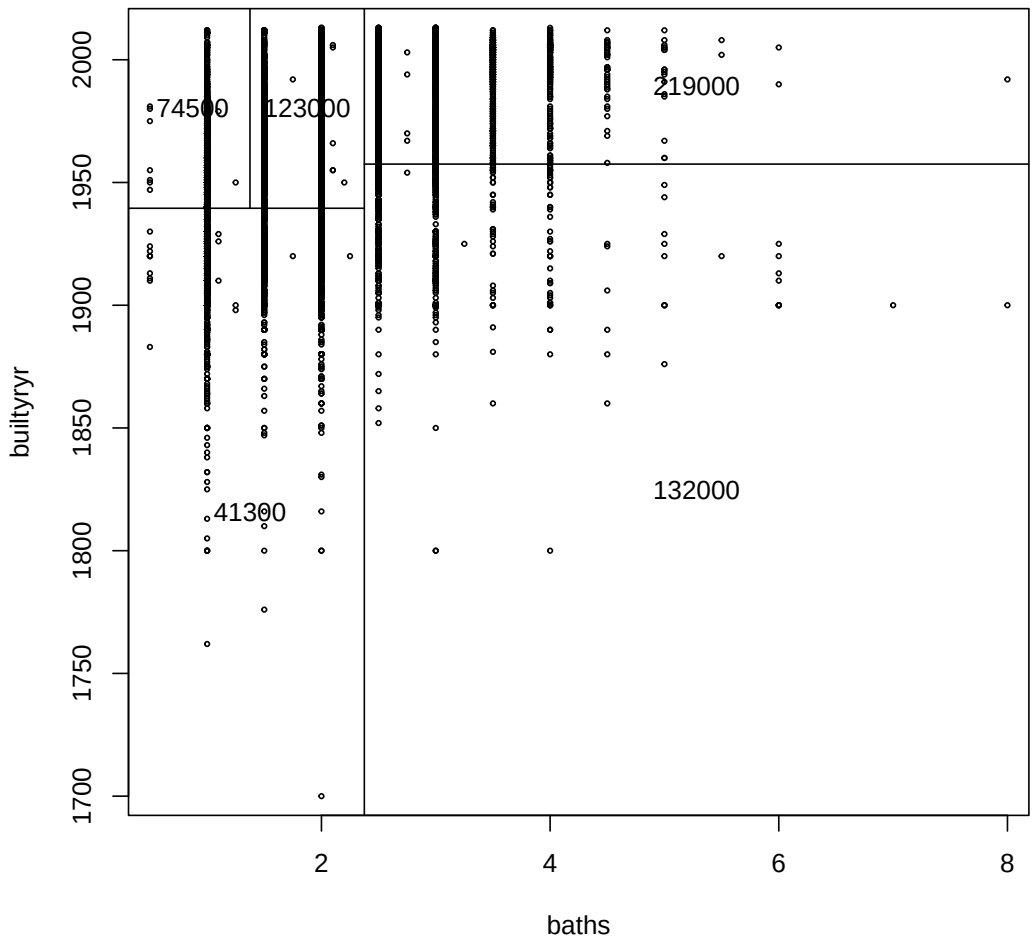
$$\begin{aligned} Y &= \beta_0 \times \mathbf{I}(X_1 \leq 2.375) \times \mathbf{I}(X_2 \leq 1939.5) \\ &+ \beta_1 \times \mathbf{I}(X_1 \leq 2.375) \times \mathbf{I}(X_2 > 1939.5) \times \mathbf{I}(X_1 \leq 1.375) \\ &+ \beta_2 \times \mathbf{I}(X_1 \leq 2.375) \times \mathbf{I}(X_2 > 1939.5) \times \mathbf{I}(X_1 > 1.375) \\ &+ \beta_3 \times \mathbf{I}(X_1 > 2.375) \times \mathbf{I}(X_2 < 1957.5) \\ &+ \beta_4 \times \mathbf{I}(X_1 > 2.375) \times \mathbf{I}(X_2 < 1957.5) \end{aligned}$$

Implied Partition of feature space



Regression Trees: Hedonic Pricing Example

```
library(tree)
set.seed(1312)
SAMPLE<-sample(1:nrow(HOUSES),18000)
TRAINING<-HOUSES[SAMPLE]
res<-tree(soldprice~builtyryr+baths, data=TRAINING)
partition.tree(res)
points(TRAINING[, c("baths","builtyryr"),with=F], cex=.4)
```



How to build a tree? Binary Recursive Splitting

- As before, we want to minimize prediction error RSS, computed as

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- There is an uncountably large ways that you could split the predictor space into $R_1, ..., R_J$ boxes.
- We follow an algorithmic approach known as *greedy recursive binary splitting*.
- At each step, identify the variable j and the split point s such that the resulting reduction in training RSS is the largest.
- Perform this step iteratively on the sub trees until you are left with a mininum number of observations in each subtree (there are alternative stopping criteria)

Top Down Greedy Binary Recursive Splitting

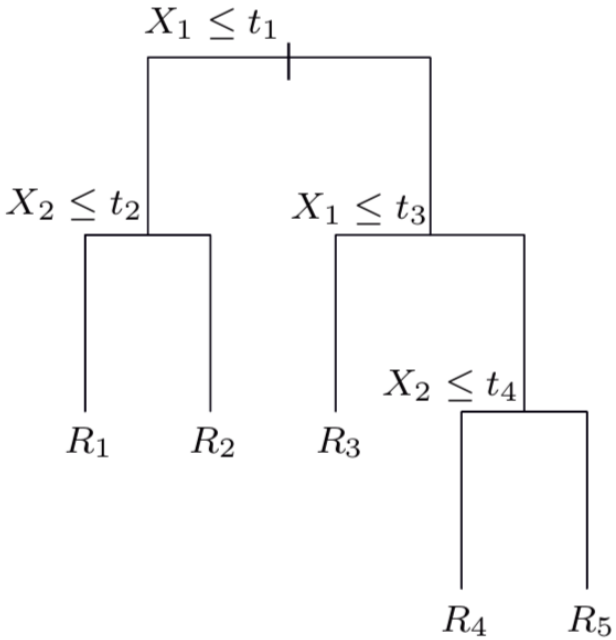
Define two half planes as $R_1(j, s) = \{X | X_j < s\}$ and $R_2(j, s) = \{X | X_j \geq s\}$. [Binary Recursive Splitting]

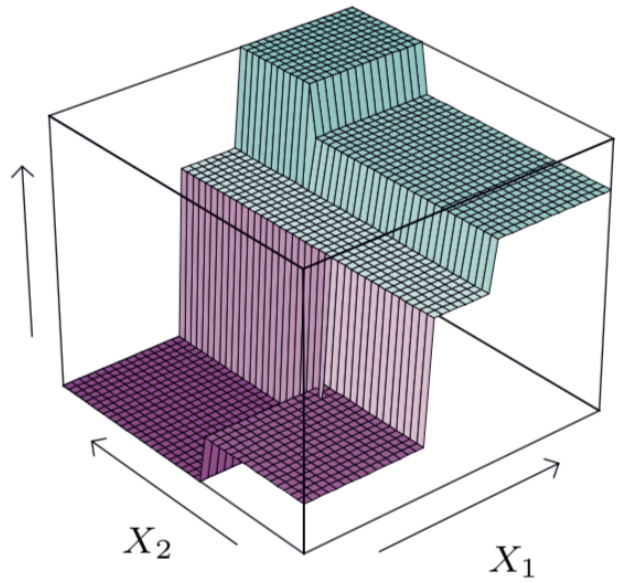
1. Initialize with empty tree $\mathcal{T}$
2. For each $j = 1, ..., k$, find the s that minimizes

$$\min_s RSS_j = \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

3. Choose the j and corresponding optimal $\tilde{s}$ at which the value of RSS_j is lowest and split the data at this point into two half planes R_1, R_2.
4. For each new half plane, repeat (2)-(4) so long as cutting criterion is not reached.

(0,0) – (3,0) – (3,3) – (0,3) – (0,0) ; (1.2,0) – (1.2,3); (2.0,0) – (2.0,3); (0,1) – (1.2,1); (





Graph from Chapter 9 of Hastie, Tibshirani and Friedman, "The Elements of Statistical Learning", Springer 2009

Bias vs Variance for Large Trees

- In the extreme case, each terminal node will contain only a single observation.
- This obviously results in bad prediction performance on the test set due to high variance, as we are overfitting the data.
- So we could do two things:
  - Require each cut to result in a significant reduction in RSS. Why is that not a good idea?
  - Build a very large tree, and then remove (or combine) regions - **tree pruning**

Pruning of Trees

- Starting out from a large tree $\mathcal{T}$, we could "prune" the tree, i.e. remove and combine regions from the bottom and creating subtrees.
- We want to select a subtree that has lowest test error. For any subtree, we could estimate test error using cross-validation, but this is (often) not feasible due to large number of trees (like best subset selection)
- *Cost complexity pruning* considers a sequence of trees that are a function of a tuning parameter α.

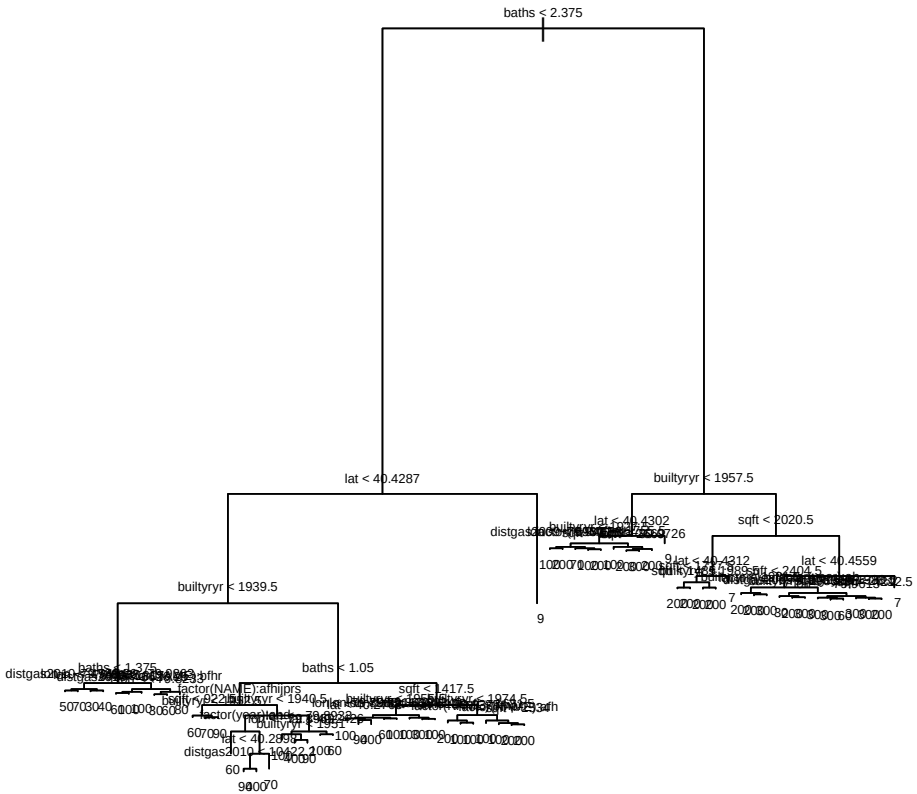
$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

$|T|$ is the number of terminal nodes in a tree, i.e. the number of boxes, R_m is rectangle pertaining to terminal node m and $\hat{y}_{R_m}$ is mean of training observations in R_m.

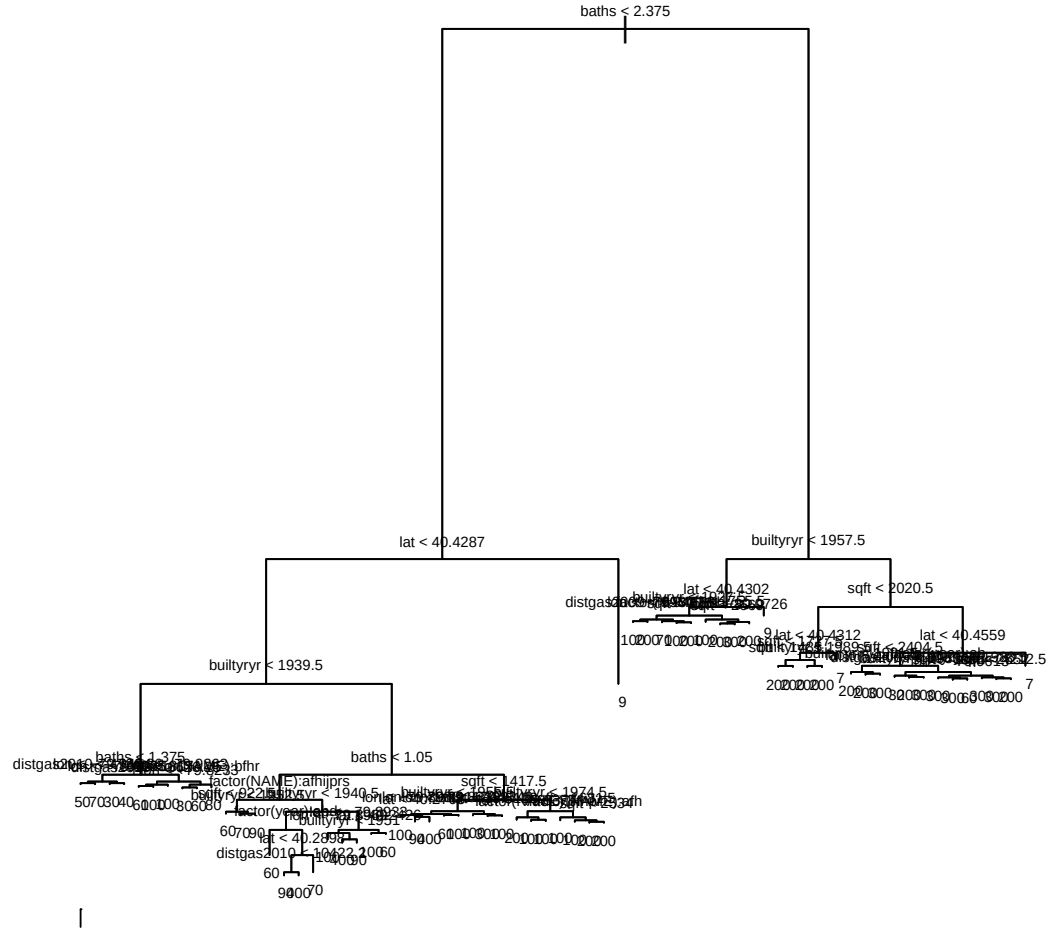
Complex "Bushy" Tree

```
res<-tree(soldprice/1000~ sqft+baths+builtyryr+distgas2009+
            distgas2010+lat+lon+factor(NAME)+factor(year), data=TRAINING,
          control=tree.control(nrow(TRAINING),mincut=2,minsize=5,mindev=0.0005))
plot(res)
text(res, cex=.5)

## Warning in text.tree(res, cex = 0.5):  'digits' must be positive, so increased to 1
```

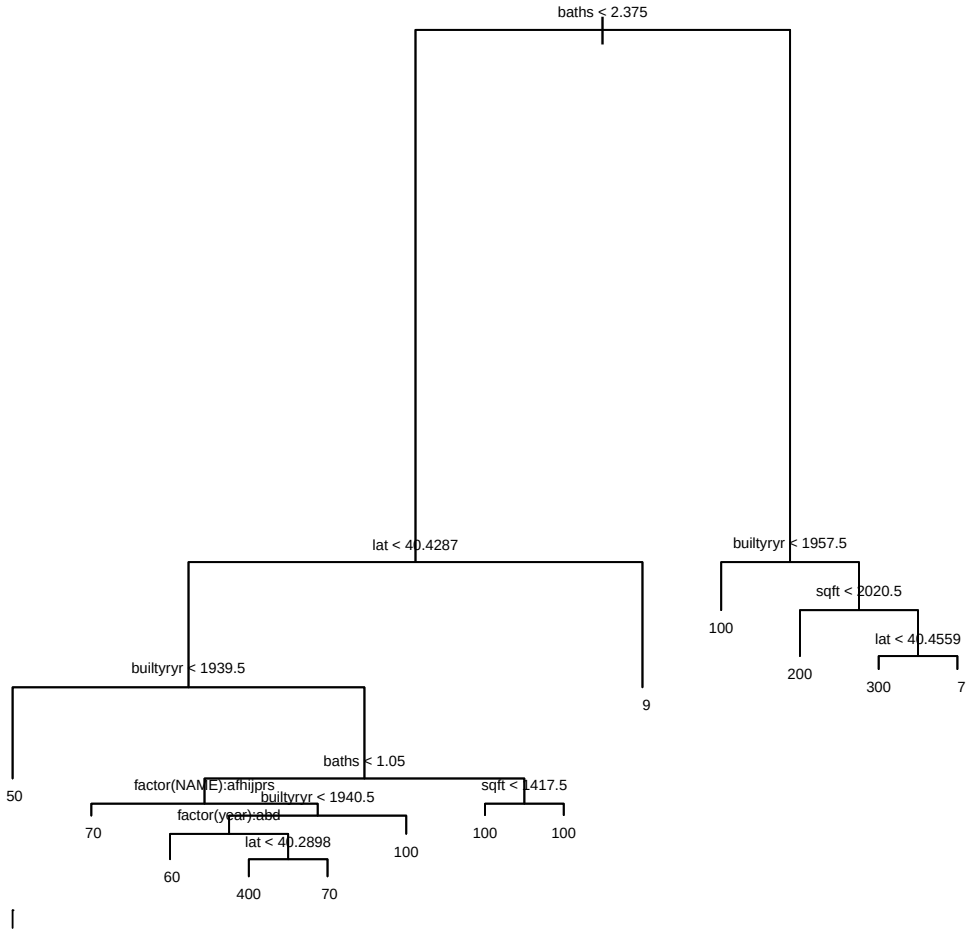


Pruning a "Bushy" Tree



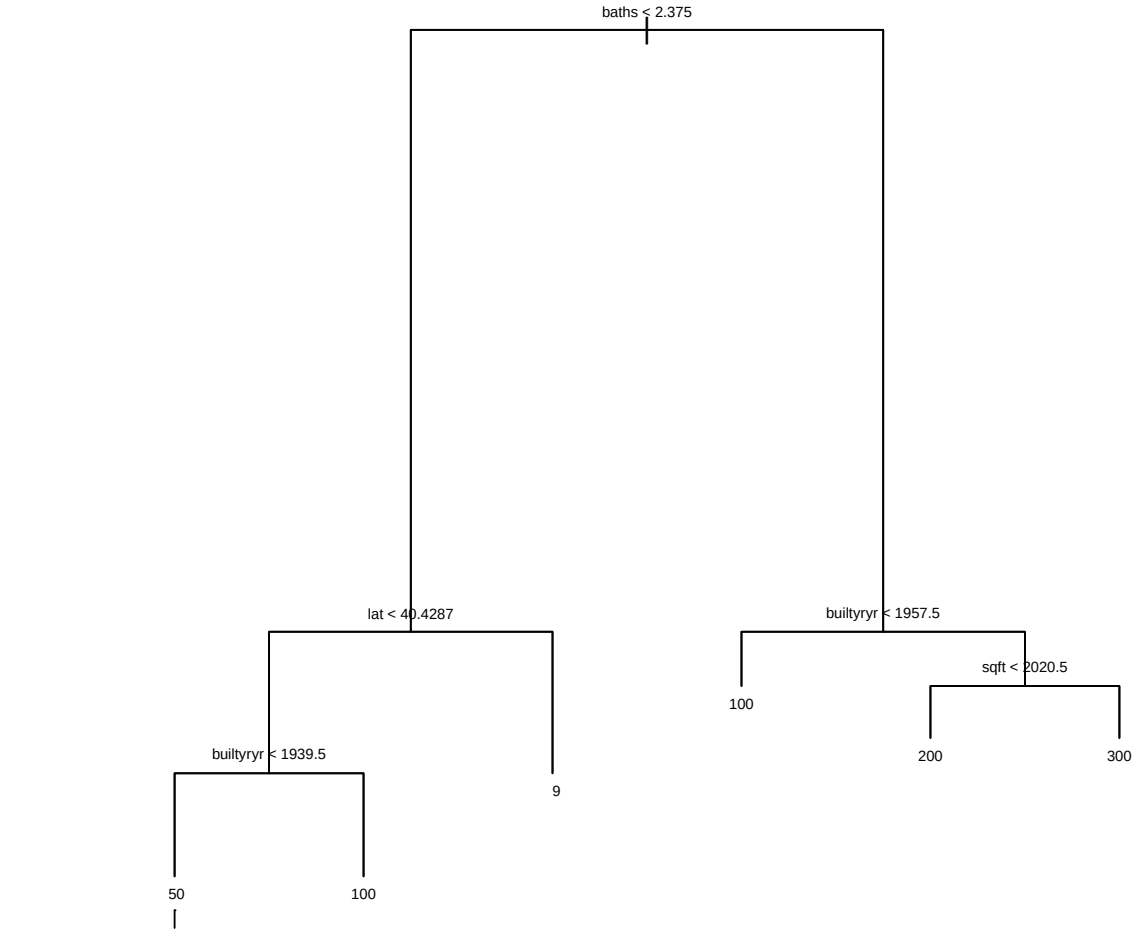
Cost Complexity 0

Pruning a "Bushy" Tree



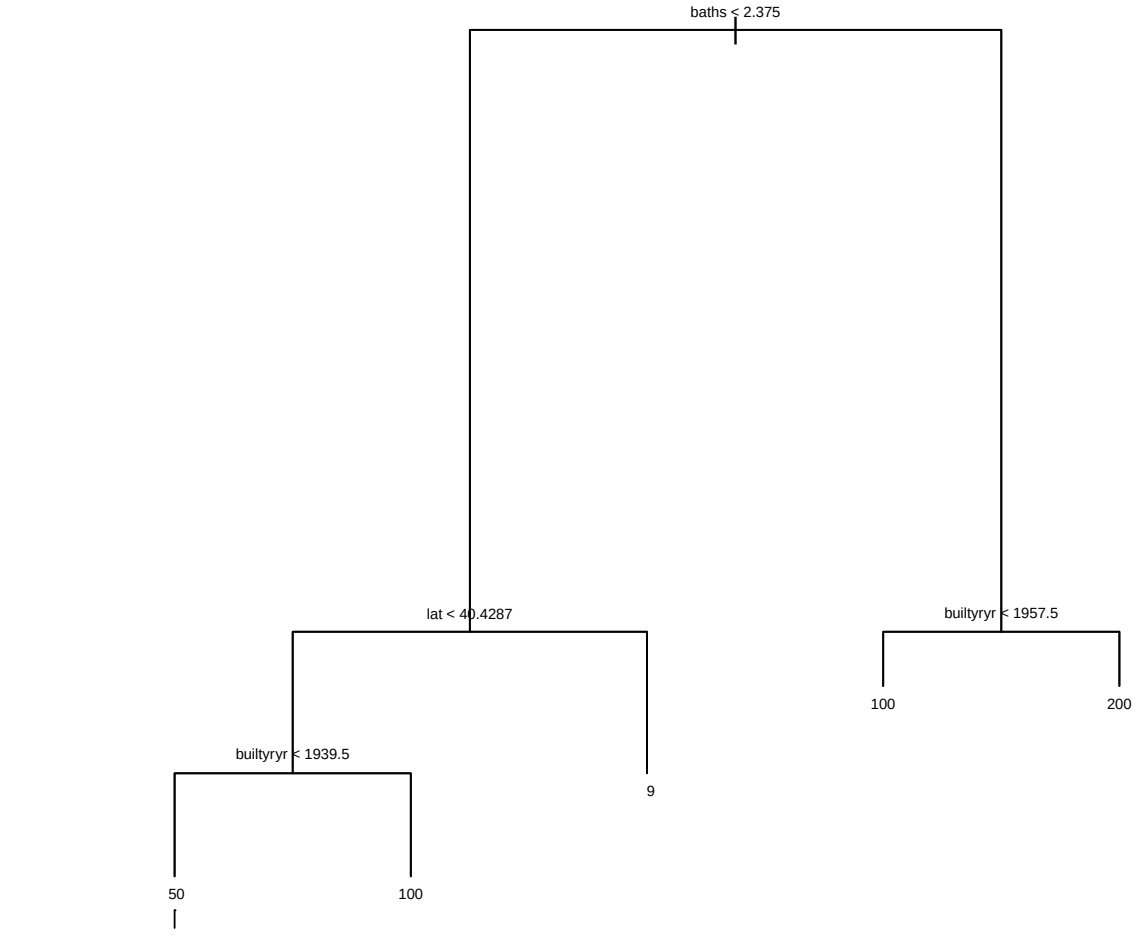
Cost Complexity 975000

Pruning a "Bushy" Tree



Cost Complexity 3475000

Pruning a "Bushy" Tree

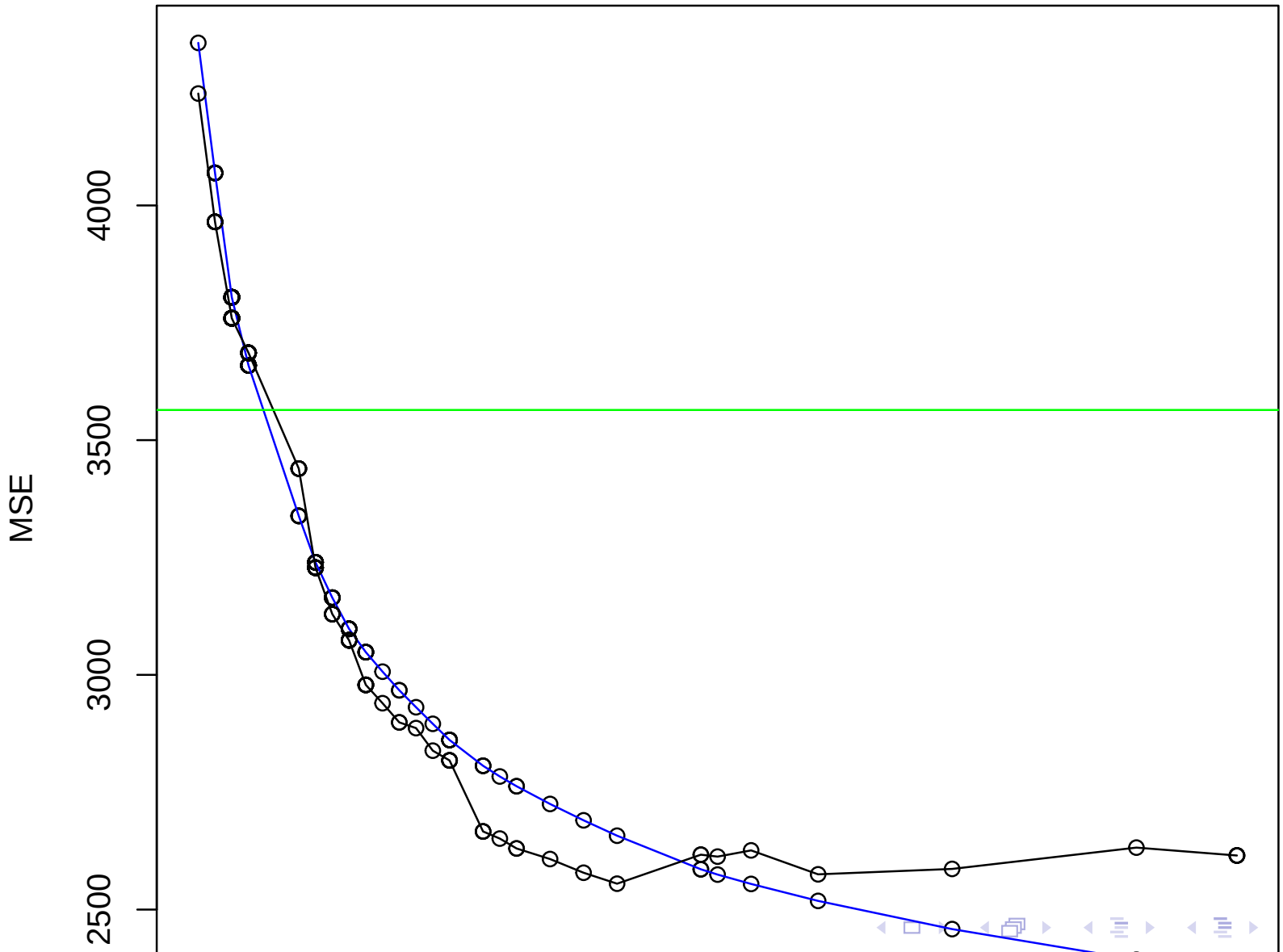


Cost Complexity 4975000

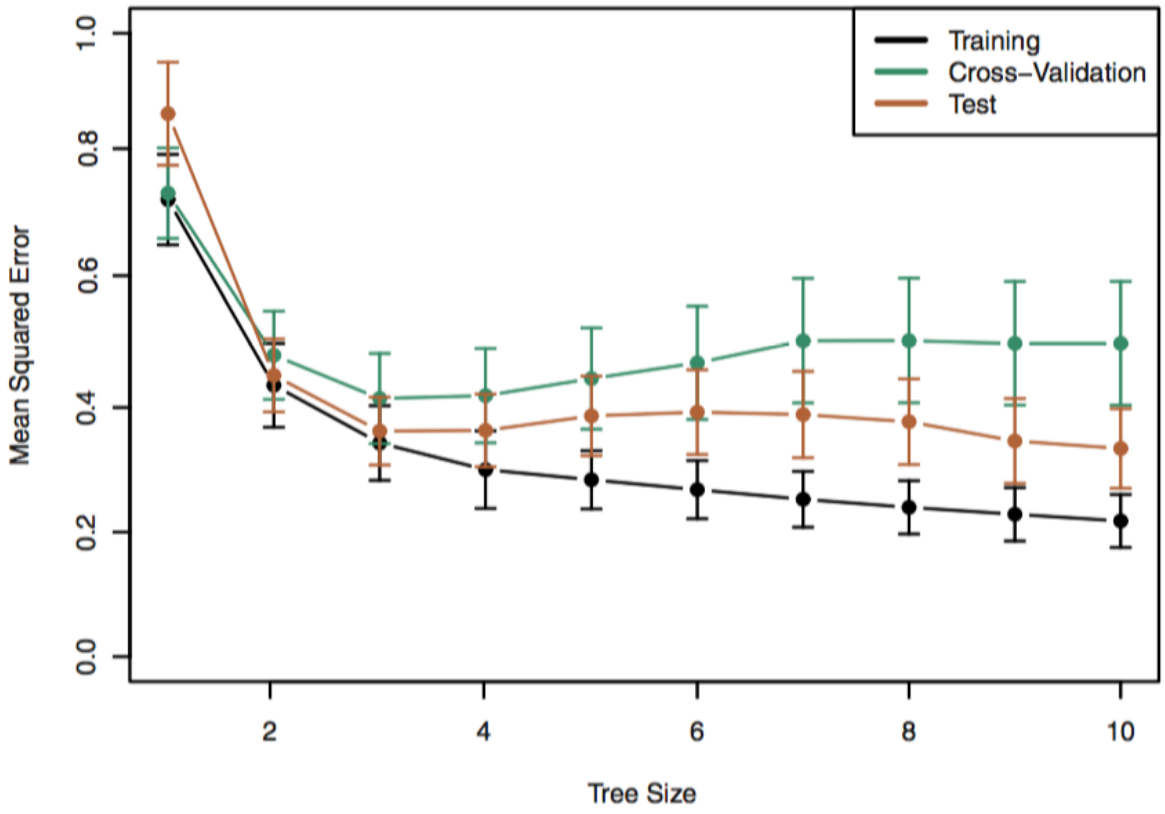
Cross Validation Error, Test Error and Training Error

- Hence the idea is to vary α and have the computer compute different subtrees, one for each value of α.
- As $\alpha \to 0$, we obtain the unpruned tree.
- As $\alpha \to \infty$, we obtain a tree with a single terminal node.
- How to decide on optimal α? Two approaches...
  - We can perform a *validation set* approach to compute test and training error and based on the minimum point of test error, choose
  - Alternatively, we can obtain a sequence of subtrees and then compute the K-fold cross validation error.
- Present each in turn for our example

Validation Set Approach: Training and Test MSE



Cross validation



Building Regression Trees

[Building Regression Trees]

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each k $= 1, \ldots,$ K:
   - 3.1 Repeat Steps 1 and 2 on all but the kth fold of the training data.
   - 3.2 Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α
4. Average the results for each value of α, and pick α to minimize the average error.
5. Return the subtree from Step 2 that corresponds to the chosen value of α.

Regression Trees versus Linear Models

- We can write a Regression Tree predictive model as

$$f(X) = \sum_{i=1}^{M} c_m \mathbf{I}(X \in R_m)$$

- The basis functions are indicator functions and c_m is the estimated average of the response variable.
- It becomes clear from the previous examples, that regression trees are highly non-linear as they allow interactive effects between variables. How?
- Rectangular partitioning of predictor space may be adequate, but in some cases, the partition may be non-linear.

Regression Trees Pros and Cons

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do the regression approaches seen so far.
- Trees can easily handle qualitative predictors without the need to create dummy variables.
- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression methods.
- Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

Next, we explore Bagging and Boosting as an approach to aggregate many regression trees into Forests.

Plan

Ridge Regression

Lasso

Elastic Net

Regression Trees

Bagging and Random Forests

Bagged Trees

- Suppose you calculate $\hat{f}^1(x), ..., \hat{f}^B(x)$ from B different randomly selected training sets, your prediction is given as

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

- We sample from our set of observations n with replacement, so some observations may randomly appear multiple times.
- We compute B regression trees using B bootstrapped training sets, and average the resulting predictions.
- *No need to prune* these trees. An individual tree has high variance, but low bias.
- Averaging across these B trees *reduces the variance*.
- Intuition: Variance in prediction due to individual trees fitting (non existing) patterns in the error term ϵ. Each individual bootstrapped tree fits a different pattern, which washes out as we average.

Variance reduction explained

- Why add the *bagging* step?
  - The various predictions from the tree are likely to be very correlated among them, since they were generated from the same original set of data – so they are hardly *independent*
  - If there is a positive pairwise correlation ρ between the predictions $\hat{y}_i$, then the variance of the averaged prediction can be worked out to be

$$\text{Var}\left[\frac{1}{B}\sum_{b=1}^{B}\hat{y}_i\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

    Now, the second term disappears with $B \to \infty$, but the first will not go away
  - *Bootstrapping* is also a way to reduce the dependence among the different samples, and therefore reduce the dependence in the predictions $\Rightarrow$ *reduction in the variance*

Random Trees

- **Random Trees**: small modification of the **regression trees**
- **Step-wise algorithm**, random-tree version:

1. Start with the full data and a single R_1, so $\hat{y} = \bar{y}$

2a. Randomly a subset $m \leq p$ of the covariates as candidates for splitting (usual value is $m = \sqrt{p}$)

2b. Find the variable j **within those selected in step 2a** and the split point s such that the prediction error is minimized

$$\min_{j,s} \left[\sum_{x_i \in R_1(j,s)} (y_i - \hat{c}_1) + \sum_{x_i \in R_2(j,s)} (y_i - \hat{c}_2) \right]$$

3. Keep splitting until stopping criterion is reached

Random Forests

- The problem: our different bootstrapped $\hat{f}^b$'s are correlated mechanically, since they are obtained using the same set of observations. This increases the variance of our prediction.
- Correlation arises from using the same set of X's in the construction of each bootstrapped tree.
- One way to avoid this is to try to *decorrelate* individual $\hat{f}^b$'s
- To avoid this, when building these trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Usually $m \approx \sqrt{p}$ is chosen.
- This mechanically decorrelates the fitted values.