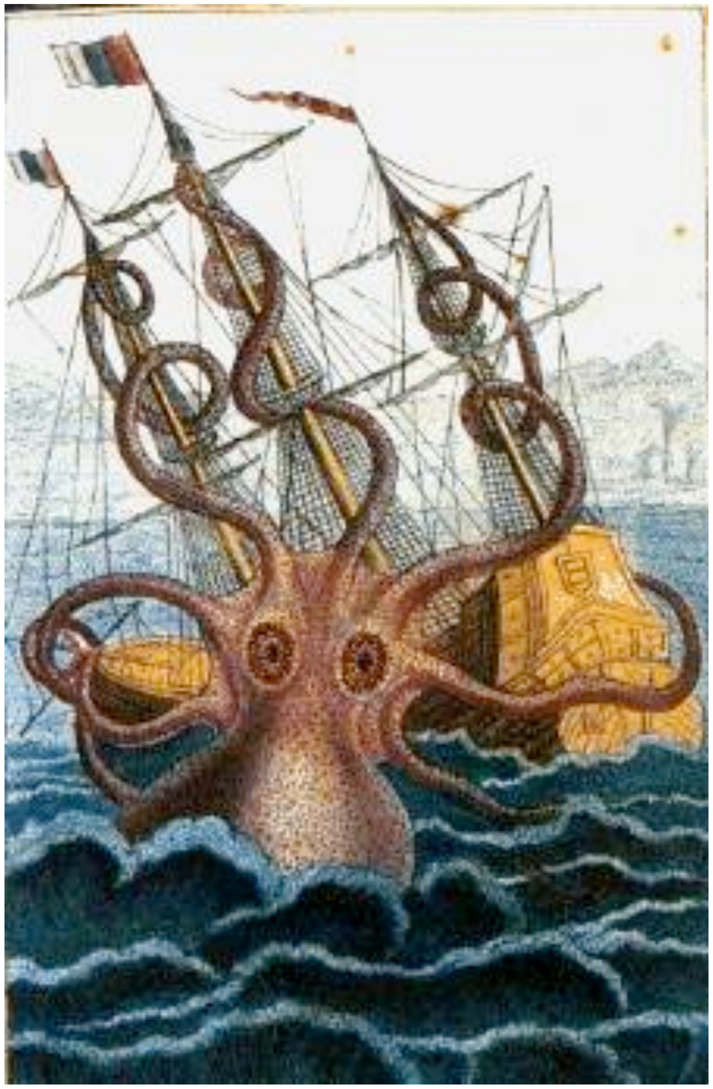


Lecture 7: Modeling Krak(en)

Variable selection

Last time we saw two selection problems -- How do we pick either the subset of important variables (in the sense of in-sample error) or the shrinkage parameter?

In the 1970s Colin Mallows, Gideon Schwarz and Hirotsugu Akaike collectively released the regression modeling Kraken -- Cp, BIC (“Bayesian information criterion”) and AIC (“an information criterion”) are a family of automatic rules that attempt to guide the selection process toward “favorable” models

We can motivate Cp from the standpoint of in-sample error...

*Much of this material is taken from Hastie, Tibshirani and Friedman

Variable selection

Throughout this section, we now assume that our data are generated via the model

$$\mathsf{y_i = f(m_i) + \epsilon_i, \ \ for\ i = 1, \ldots, n}$$

We no longer assume that we know very much about f -- We might apply a linear form with some of the variables or (eventually) we might try “smooth” components that allow for more flexible descriptions of the dependence of the response on the inputs

Variable selection

First off, RSS is entirely unsatisfactory as a figure of merit for selecting the complexity of a model -- By enlarging the search space, it has nowhere to go but down as you add terms or reduce the constraints on a penalized problem

But let's see how bad things can be -- Let's let err denote an in-sample error estimate

$$\overline{\mathrm{err}} = \frac{\mathrm{RSS}}{n} = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - \widehat{f}(m_i)\right]^2$$

In our previous in-sample error expression, we considered $E_y[f(m_i) - \widehat{f}(m_i)]^2$ where the expectation is over the training data y, holding our inputs fixed -- Today we will consider instead an expectation taken with respect to a new vector of observations $Y^0 = (Y^0_1, \ldots, Y^0_n)^t$ where each element Y^0_i comes from the conditional distribution of $Y|X_1 = x_{i1}, \ldots, X_p = x_{ip}$

$$\mathrm{Err}_{in} = \frac{1}{n}\sum_{i=1}^{n} E_{Y^0}[Y^0_i - \widehat{f}(m_i)]^2$$

Variable selection

We will define the optimism in $\overline{\mathsf{err}}$ as an estimate of $\mathsf{Err}_{\mathsf{in}}$ to be

$$\mathsf{op} = \mathsf{Err}_{\mathsf{in}} - \overline{\mathsf{err}}$$

We expect the optimism to be positive because the same data are used for training and validation -- How bad can it be?

Variable selection

The average optimism is computed by (once again) averaging across the training data, or in symbols $E_y \, \text{op}$ -- That is, we consider (as we did last time) averaging across all possible realizations of our experiment, holding the input data fixed

Then for OLS (and other loss functions!) you can show that

$$E_y \, \text{op} = \frac{2}{n} \sum_{i=1}^{n} \text{cov}\left[y_i, \widehat{f}(m_i)\right]$$

In words, this means that the amount by which err underestimates the in-sample error is, on average, a function of how strongly a point y_i affects its own prediction -- Put another way, the “harder” we fit the data, the greater the covariance between y_i and $\widehat{f}(m_i)$

Variable selection

To find the expression for optimism (like “the glass is half full!”), we can simply expand err by adding and subtracting some terms

$$[y_i - \widehat{f}(m_i)]^2 = [y_i - f(m_i) + f(m_i) - E_y\widehat{f}(m_i) + E_y\widehat{f}(m_i) - \widehat{f}(m_i)]^2$$

Squaring this expression and passing through an expectation, you find that the only terms that survive are

$$E_y[y_i - f(m_i)]^2 + [f(m_i) - E_y\widehat{f}(m_i)]^2 + E_y[\widehat{f}(m_i) - E_y\widehat{f}(m_i)]^2$$
$$+2E_y\left[\left(y_i - f(m_i)\right)\left(\widehat{f}(m_i) - E_y\widehat{f}(m_i)\right)\right]$$

The first term is just the error variance (also known as the “irreducible error” because we can’t model our way out of it), the second is squared bias and the third is the variance of $\widehat{f}(m_i)$ -- The last term is the covariance we wanted

Variable selection

Playing the addition and subtraction game with the expression for Err_{in} we find the same squared terms but none of the cross terms survive -- Therefore, the difference between Err_{in} and the expectation of err is the sum of covariances

$$E_y \, \text{Err}_{\text{in}} = E_y \overline{\text{err}} + \frac{2}{n} \sum_{i=1}^{n} \text{cov} \left[\, y_i, \widehat{f}(m_i) \, \right]$$

Suppose we want to model f with a linear form estimated via OLS (keeping in mind that we are not assuming the true f has this form)

$$\widehat{f}(m_i) = \widehat{\beta}_1 x_{i1} + \cdots + \widehat{\beta}_p x_{ip}$$

In this case, the sum of covariances simplifies to $\sigma^2 \, \text{trace} \left[M(M^t M)^{-1} M^t \right]$ or just $p\sigma^2$

Variable selection

The krak unleashed on the world in the 1970s was the simple expression

$$\widehat{\mathrm{Err}_{\mathrm{in}}} = \overline{\mathrm{err}} + \widehat{\mathrm{op}}$$

When identifying a good subset of predictors for linear regression, Mallows' Cp statistic is

$$\mathrm{Cp} = \frac{\mathrm{RSS}}{\mathrm{n}} + \frac{2\mathrm{p}}{\mathrm{n}}\widehat{\sigma}^2$$

where $\widehat{\sigma}^2$ is an estimate of the noise variance, usually from a low-bias model (like using all the predictors, assuming n > p)

Variable selection

AIC (the “A” stands for “an” not “Akaike”) is derived in an entirely different fashion -- One form reduces to Mallows’ Cp when the error variance is known, but another can be derived when it is unknown

$$\mathsf{AIC} = \frac{\mathsf{RSS}}{\mathsf{n}} + \frac{\mathsf{2p}}{\mathsf{n}}\sigma^2 \quad \text{and} \quad \mathsf{AIC} = \log\,\mathsf{RSS} + \frac{\mathsf{2p}}{\mathsf{n}}$$

BIC (the Bayesian information criterion) takes, well, a Bayesian approach involving a so-called Laplace expansion of a posterior distribution, leading to a criterion of the form

$$\mathsf{BIC} = \log\,\mathsf{RSS} + \frac{\mathsf{p}\log\mathsf{n}}{\mathsf{n}}$$

Variable selection

These criteria are all some form of penalized least RSS -- While RSS skids downward with increased model complexity, these provide an offset that increases with complexity

BIC is said to be consistent in that if the true model is among our candidates, it will choose it with probability tending to one as our sample size increases -- In infinite dimensional settings (all finite models are wrong), AIC, on the other hand, is said to be prediction optimal

Notice also that the offset is additive -- These criteria all order models of the same size according to their RSS, a fact that will prove important when they are paired with mid-70s computational procedures for “all subsets” regression

Aside: BIC

As an estimate of the posterior probability of a model (a collection of variables in a regression equation, say), we might expect that it could be used for something other than simply finding a minimum

Indeed, we could “average” the predictions over models using these posterior probabilities to come up with a new estimate -- This is often called Bayesian model averaging and was all the rage about 5-10 years ago

If we have K models, we would weight the kth as

$$w_k = \frac{e^{-\mathrm{BIC}_k/2}}{\sum_{l=1}^{K} e^{-\mathrm{BIC}_l/2}}$$

and form an estimate

$$\widetilde{f}(m) = \sum_{k=1}^{K} w_k \widehat{f}_k(m)$$

Regression by leaps and bounds

As we noted, Cp and the other criteria order models with the same number of variables **according to their RSS** -- This means we can identify the “good” models by looking for the small RSS fits of each size

In their “Regresison by Leaps and Bounds” technique, Furnival and Wilson (1974) do this by **organizing models in a tree structure** -- On the next slide each of the 32-1 = 31 models that can be constructed from 5 variables (leaving out the “null” model involving none of the 5)

The numbers on each node represent the indices of the variables included in that node’s model -- Those that are underlined will be dropped from the node’s children (and ignore the “.” notation for the moment in the node labels)

When computing the best (in terms of RSS) models of a given size, we can avoid examining all 32-1 models with a simple bounding argument -- Recall that **the residual sum of squares can only increase if we drop terms from a model**

Therefore, if we are at node 2345 and we’ve already seen models of size 1, 2 and 3 that all have lower RSS values, then we can safely ignore the 14 descendants of this model (!)

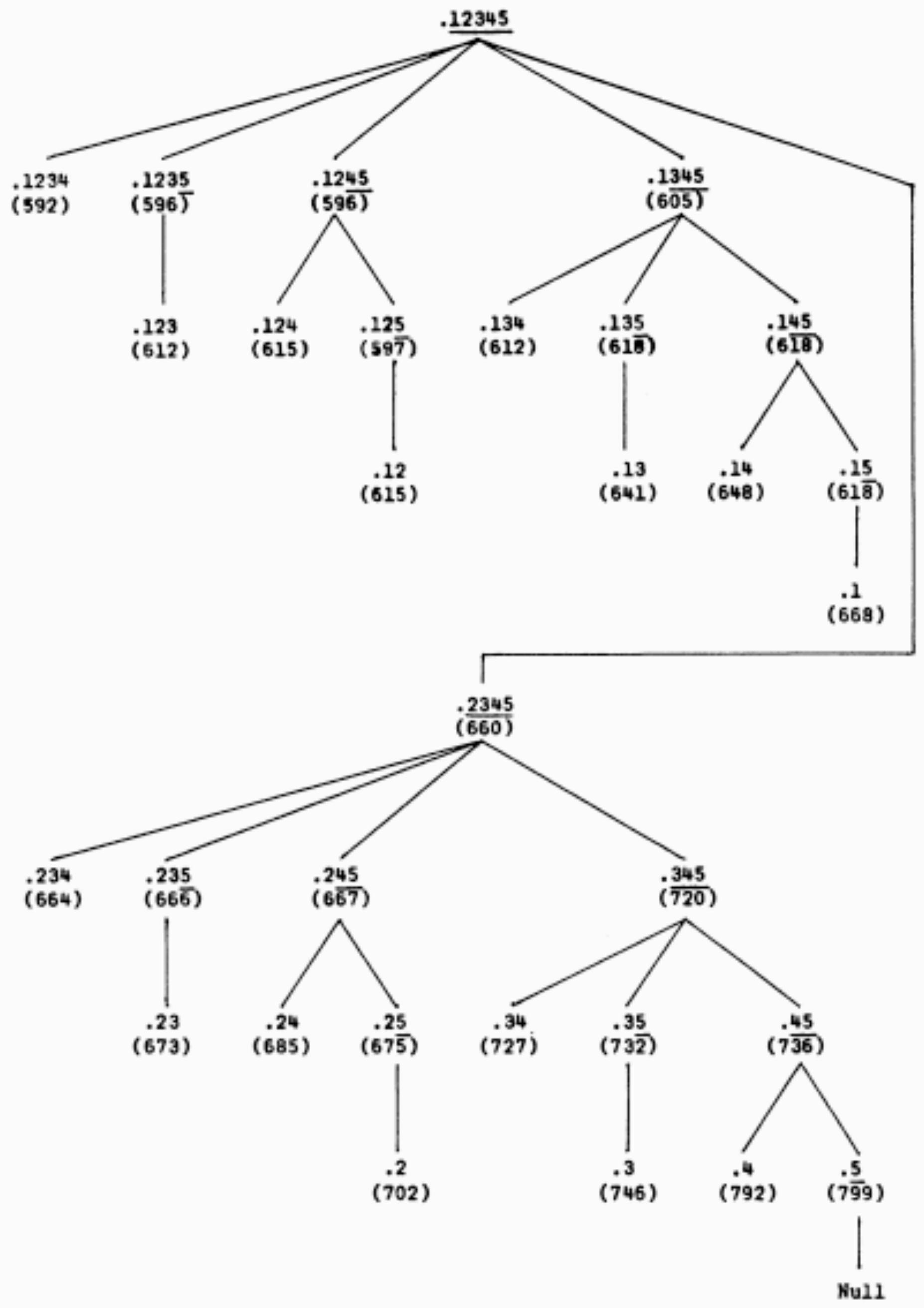


FIGURE 5—The inverse tree

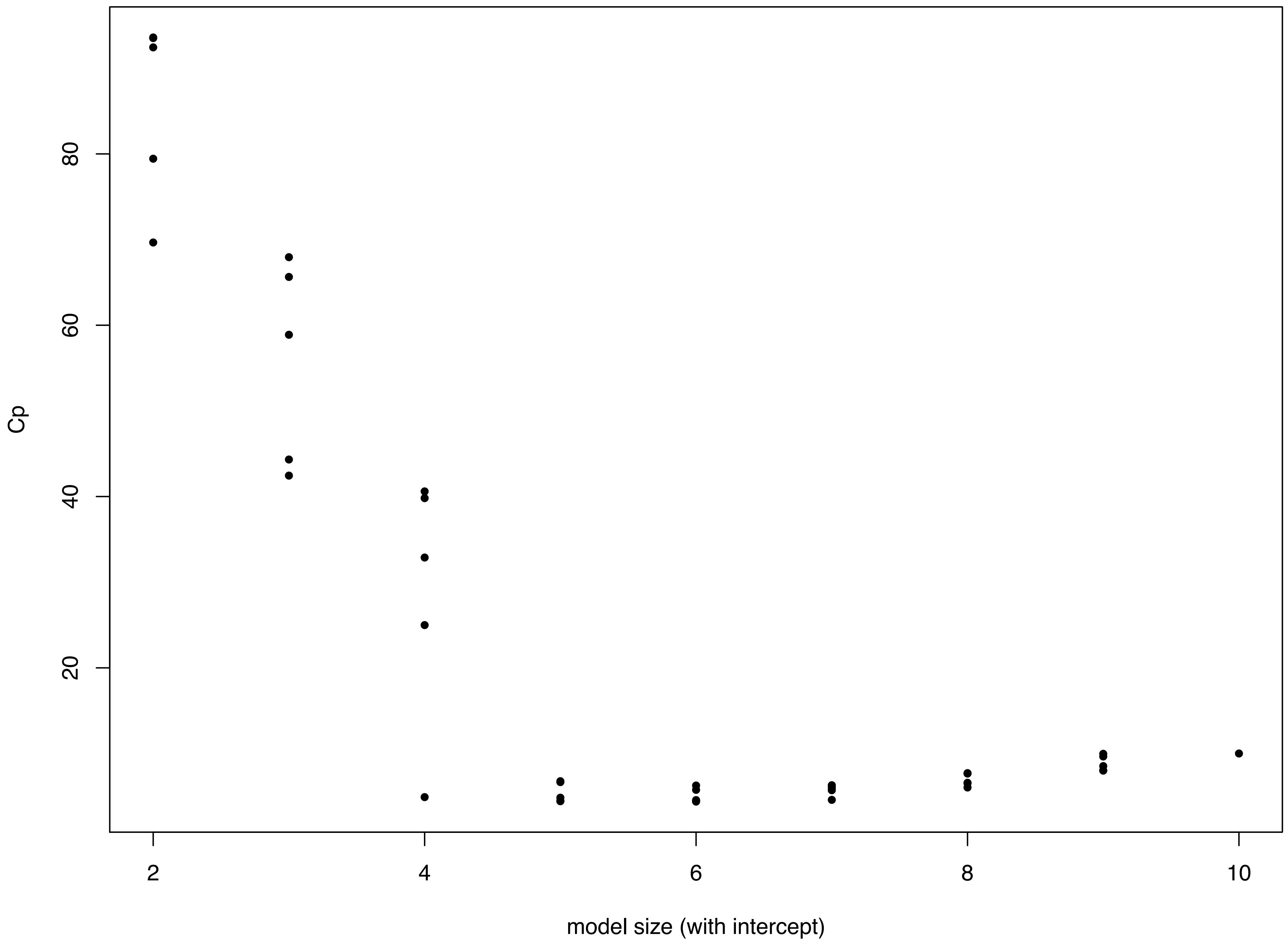
Variable selection

There are other approaches to “all subsets” selection and the procedure only words for models that aren’t too big -- Other “greedier” strategies can also be applied that traverse a portion of the model tree in various ways

As we’ve mentioned, prior to all subsets techniques, researchers applied techniques like **forward selection** (start with an intercept-only model and successively add terms that produce the largest drop in RSS) or **backward deletion** (start with the full model and successively add terms that produce the smallest rise in RSS) or **various combinations** of the two

Ultimately, these greedy schemes walk a fairly narrow path through along the tree and can, as a result, be somewhat unstable -- There has been interesting (relatively) recent work that tries to expand these paths using McMC and other random devices (Smith and Kohn 1996, Stone et al 1994)

```
library(leaps)

M <- model.matrix(ln_death_risk~ln_pop+ln_fert+ln_events+hdi-1,data=vul)

# add five “noise” variables

M <- cbind(M,matrix(rnorm(nrow(M)*5),ncol=5))

# perform leaps and look at the results

fits <- leaps(M,vul$ln_death_risk,nbest=5)
plot(fits$size,fits$Cp,pch=20)

cbind(fits$which,fits$Cp)[fits$size %in% c(4,5,6),]

#   1 2 3 4 5 6 7 8 9
# 3 1 1 1 0 0 0 0 0 0  4.909156
# 3 1 0 1 1 0 0 0 0 0 24.991749
# 3 0 1 1 1 0 0 0 0 0 32.880029
# 3 0 1 1 0 1 0 0 0 0 39.814804
# 3 0 1 1 0 0 0 0 0 1 40.598689
# 4 1 1 1 1 0 0 0 0 0  4.430491
# 4 1 1 1 0 1 0 0 0 0  4.463918
# 4 1 1 1 0 0 0 0 0 1  4.843043
# 4 1 1 1 0 0 1 0 0 0  6.664199
# 4 1 1 1 0 0 0 0 1 0  6.768688
# 5 1 1 1 1 0 0 0 0 1  4.387238
# 5 1 1 1 0 1 0 0 0 1  4.480889
# 5 1 1 1 1 1 0 0 0 0  4.560858
# 5 1 1 1 1 0 1 0 0 0  5.759277
# 5 1 1 1 0 1 1 0 0 0  6.261972
```


Cp
20
40
60
80
2
4
6
8
10
model size (with intercept)

A graphical device

In his original paper on Cp, Mallows suggests graphics to help examine promising models -- He intended for a researcher to examine the structures common to these models (Do they share variables?)

His paper is beautiful and worth spending some time with -- He had a follow up in the mid-90s that discusses how Cp was meant to be used and how it functions in modern applications..

Variable selection

And with that turn, we see how a well-meaning procedure becomes a mid-century Kraken -- Mallows in his 1973 paper explicitly warns agains slavishly minimizing Cp as a modeling strategy

But very few people listen -- In 2002 Miller even notes that “their usage has long been a quiet scandal in the statistical community”

What can go wrong?

Penalization

In general, if we have a linear model where “linear” means we can express the conditional means in the form Sy for a matrix S that does not depend on the responses y, then our selection rules become

$$\mathsf{Cp} = \frac{\mathsf{RSS}(\lambda)}{\mathsf{n}} + \frac{2\,\mathsf{trace}\,\mathsf{S}(\lambda)}{\mathsf{n}}\widehat{\sigma}^2$$

Note that the trace of S is playing the role of p in the OLS case, giving us one of our three interpretations of the degrees of freedom

Variable selection

If we return for a moment to orthonormal predictors and assume that the error variance is known -- The RSS for a model using any collection $\mathcal{I} \subset \{1, \ldots, p\}$ of variables is simply

$$\mathrm{RSS}(\mathcal{I}) = \sum_{i=1}^{n} y_i^2 - \sum_{j \in \mathcal{I}} \widehat{\beta}_j^2$$

Therefore, our best subset of size 1 consists of the model with the largest coefficient in absolute value -- The best subset of size 2 consists of the two variables with the largest coefficients in absolute value and so on

What about Cp?

Variable selection

Using our expression for RSS, we find that the Cp statistic associated with a model using the variables with indices in $\mathcal{I}$ is

$$\mathsf{Cp}(\mathcal{I}) = \frac{\mathsf{RSS}(\mathcal{I})}{\mathsf{n}} + \frac{2\#\mathcal{I}}{\mathsf{n}}\,\sigma^2$$

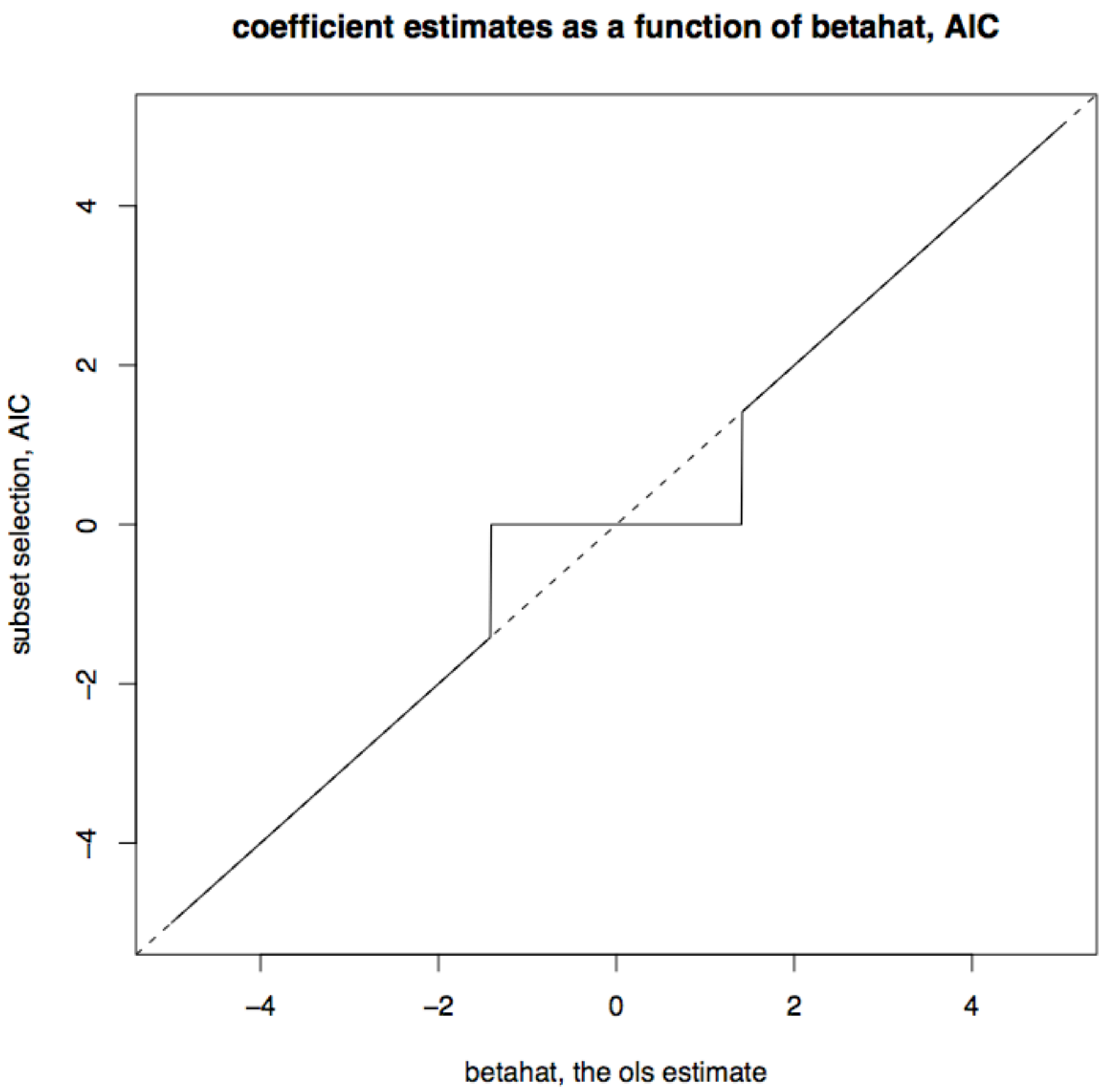
Therefore

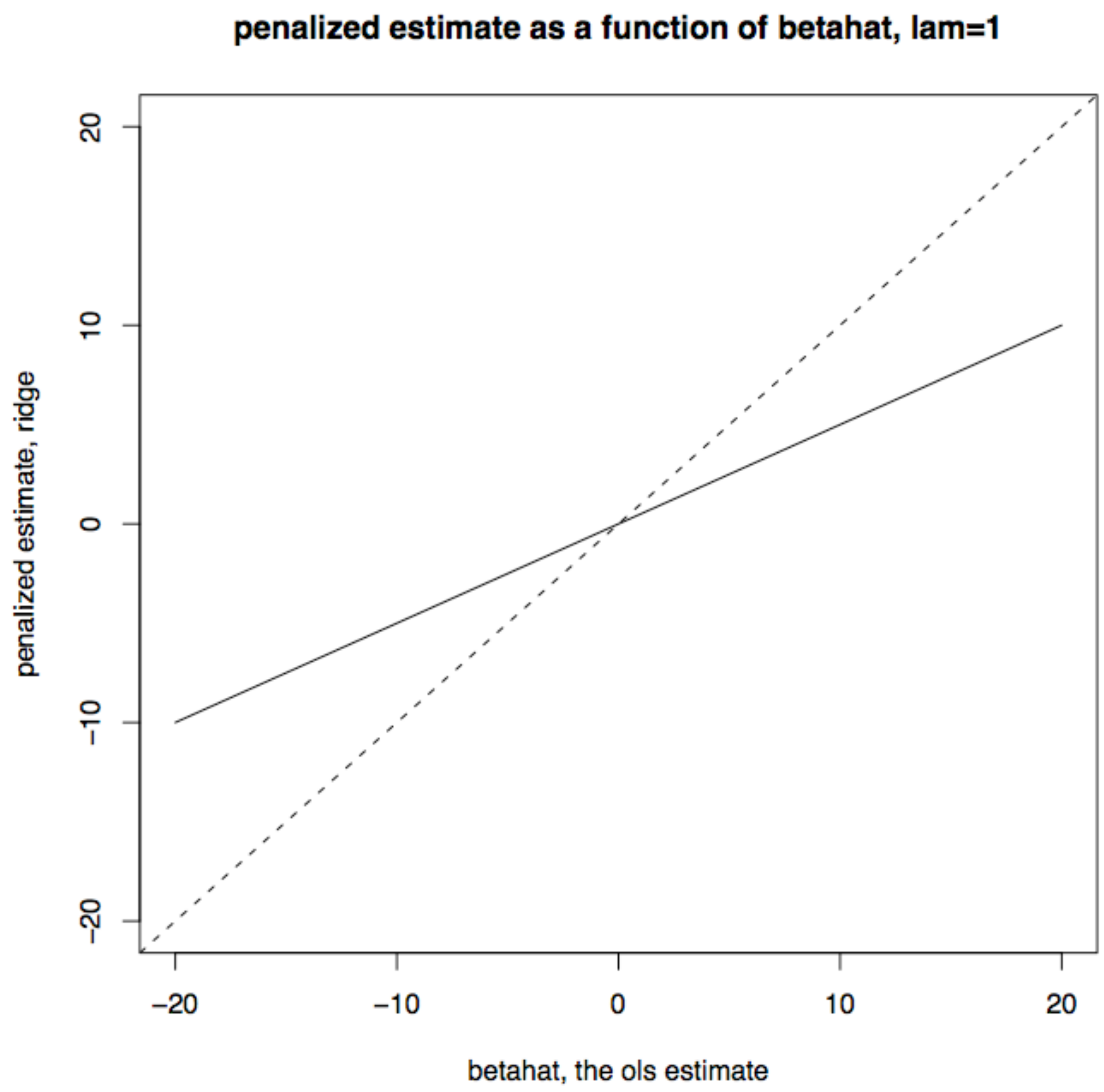
$$\begin{aligned}
\mathsf{Cp}(\mathcal{I}) &= \frac{\mathsf{RSS}(\mathcal{I})}{\mathsf{n}} + \frac{2\#\mathcal{I}}{\mathsf{n}}\sigma^2 \\
&\propto \sum_{i=1}^{n} y_i^2 - \sum_{j\in\mathcal{I}} \widehat{\beta}_j^2 + 2\#\mathcal{I}\sigma^2 \\
&= \sum_{i=1}^{n} y_i^2 + \sum_{j\in\mathcal{I}} (2\sigma^2 - \widehat{\beta}_j^2)
\end{aligned}$$

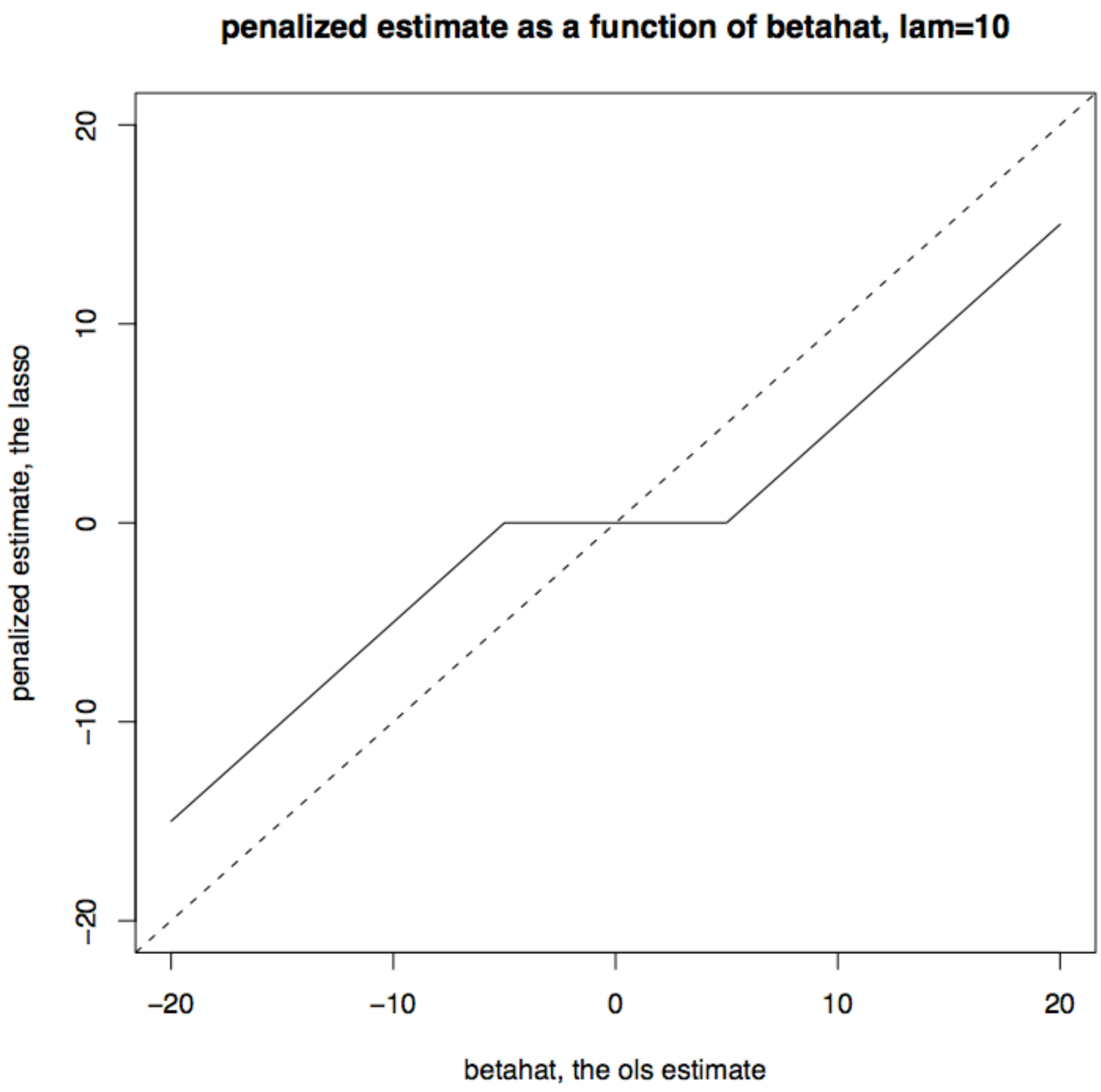
Variable selection

This means that the minimizer for Cp can be written explicitly as the set of variables for which the associated coefficients $\widehat{\beta}_j$ have standardized statistics $|\widehat{\beta}_j|/\sigma$ larger than $\sqrt{2}$ (Standardized? In what sense?)

Let's see what this looks like...


coefficient estimates as a function of betahat, AIC
subset selection, AIC
betahat, the ols estimate
−4
−2
0
2
4
−4
−2
0
2
4


penalized estimate as a function of betahat, lam=1
penalized estimate, ridge
betahat, the ols estimate
-20
-10
0
10
20
-20
-10
0
10
20


penalized estimate as a function of betahat, lam=10
penalized estimate, the lasso
betahat, the ols estimate
−20
−10
0
10
20
−20
−10
0
10
20

Alternate approaches

If we had enough data, we could set aside a portion to validate models, computing the error on data not used for fitting (in fact, we might create separate sets for training, validation, and testing) -- When data are somewhat less plentiful, we can appeal to so-called K-fold cross validation instead

Here, we divide the data randomly into K equally-sized pieces -- Then, we successively leave out each piece, fitting a model on the remaining K-1 sets and then evaluate the fit using the held out data

Cross-validation

Let $\kappa(\mathsf{i})$ be a mapping that indicates which of the K sets each data point i belongs to and let $\widehat{\mathsf{f}}_{-\mathsf{k}}$ denote a fitted model leaving out the kth set

With this notation, we can now write

$$\mathsf{CV} = \frac{1}{\mathsf{n}} \sum_{\mathsf{i}=1}^{\mathsf{n}} [\mathsf{y}_\mathsf{i} - \widehat{\mathsf{f}}_{-\kappa(\mathsf{i})}(\mathsf{m}_\mathsf{i})]^2$$

Cross-validation

Whereas we were previously interested in in-sample error

$$\mathrm{Err}_{\mathrm{in}} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{Y^0}[Y_i^0 - \widehat{f}(m_i)]^2$$

(holding training data fixed, drawing new responses) K-fold cross-validation can best be thought of as estimating the expected error

$$\mathrm{Err} = \mathrm{E}_{\mathcal{T}}\mathrm{E}_{m^0,Y^0}[Y^0 - \widehat{f}(m^0)]^2$$

where now we average over new training sets $\mathcal{T}$ (inputs and responses)

Cross-validation

For the ridge regression and subset selection, situations in which we need to pick a “complexity parameter” α (amount of shrinkage, the number of variables to include), we let $\widehat{f}_{-k}(m, \alpha)$ denote the fit with complexity α leaving out the kth portion of the data

$$CV(\alpha) = \frac{1}{n} \sum_{i=1}^{n} [y_i - \widehat{f}_{-\kappa(i)}(m_i, \alpha)]^2$$

Our final model is then fit to the complete data set using the value of $\widehat{\alpha}$ that produces a minimum value for $CV(\alpha)$

Cross-validation

If we let K=n, meaning we divide our data into n pieces, we know from our work with the SMW formula that we don't literally have to fit n separate models but can instead write

$$\frac{1}{n}\sum_{i=1}^{n}[y_i - \widehat{f}_{-i}(m_i)]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \widehat{f}(m_i)}{1 - h_{ii}}\right]^2$$

where h_{ii} is the ith diagonal element of the hat matrix

Cross-validation

It turns out that for many linear smoothers, those for which our predictions are given by $\mathsf{S}y$, we have a generalization

$$\frac{1}{n}\sum_{i=1}^{n}[y_i - \widehat{f}_{-i}(m_i)]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \widehat{f}(m_i)}{1 - S_{ii}}\right]^2$$

The generalized cross-validation (GCV) approximation replaces S_{ii} with the average diagonal elements of S -- That is,

$$\frac{1}{n}\sum_{i=1}^{n}[y_i - \widehat{f}_{-i}(m_i)]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \widehat{f}(m_i)}{1 - \mathrm{trace}(S)/n}\right]^2$$

GCV

The rationale behind this approximation is that in some settings (we'll see smoothing splines as one), it is easier to compute the trace of S than the individual diagonal elements

We can relate this final expression to AIC using the fact that

$$\frac{1}{(1-z)^2} \approx 1 + 2z$$

Connections

We’re now going to make a deeper connection between these methods -- Specifically we’ll consider the relationship between subset selection and the lasso

This will lead us to something called least angle regression, a relatively recent (say, 2002) development in modeling technology...

~~Forward stagewise modeling~~

~~Suppose we have a model g that involves a so-called basis expansion~~

$$g(x;\beta,\gamma)=\sum_{k=1}^{K}\beta_k B(x;\gamma_k)$$

~~where the B's are basis elements (think of them as your original variables from an OLS fit or maybe polynomial terms or -- later in the quarter -- spline terms or trees) and they in turn depend (possibly) on some auxiliary parameters γ_k~~

~~Then, given data $y_i,(x_{i1},\ldots,x_{ip})^t$, i=1,..,n, we would like to select the parameters β_k,γ_k to minimize some loss function~~

$$\sum_{i=1}^{n} L\big(y_i, g(x_i;\beta,\gamma)\big)$$

~~Forward stagewise modeling approximates the solution to this problem by sequentially adding new basis functions to the expansion without adjusting the parameters and coefficients of those already in the fit~~

Forward stagewise modeling

That was a bit opaque -- While a stagewise approach has connections to optimization, it might be better to consider its action first and then talk about what it’s trying to do

Stagewise modeling constructs a model in small steps...

Forward stagewise modeling

Initialize by standardizing the predictors $\widetilde{x}_{ij} = (x_{ij} - \overline{x}_j)/\mathrm{sd}(x_j)$ (center and scale to norm 1) and centering the responses $\widetilde{y}_i = y_i - \overline{y}$ -- Then set

$$\widehat{\mu} = (0, \ldots, 0)^t \quad r = \widetilde{y}, \text{ and } \widehat{\beta}_1 = \cdots, \widehat{\beta}_p = 0$$

Repeat a large number of times

Find the predictor $\widetilde{x}_j$ most correlated with r

Set $\delta = \epsilon \, \mathrm{sign}(\widetilde{x}_j^t r)$ and form the updates

$$\widehat{\beta}_j \leftarrow \widehat{\beta}_j + \delta$$

$$\widehat{\mu} \leftarrow \widehat{\mu} + \delta \widetilde{x}_j$$

$$r \leftarrow r - \delta \, \widetilde{x}_j$$

Forward stagewise modeling

Because of the standardization at the first step, each of the correlations in question are just inner products $\widetilde{x}_j^t r$ for j=1,...,p -- Also the correlation criterion is equivalent to choosing the variable that would produce the largest drop in the sum of squares of r

Now, at each step selecting $\epsilon = |\widetilde{x}_j^t r|$ would advance us to the next regression involving $\widetilde{x}_j$ in one step -- In short, this “big” value of ϵ **reduces forward stagewise modeling to the classic forward stepwise procedure**

The idea behind stagewise procedures is that the big-step greedy nature of stepwise approaches can make them notoriously unstable -- Small changes in the data, for example, can create different paths on the model tree

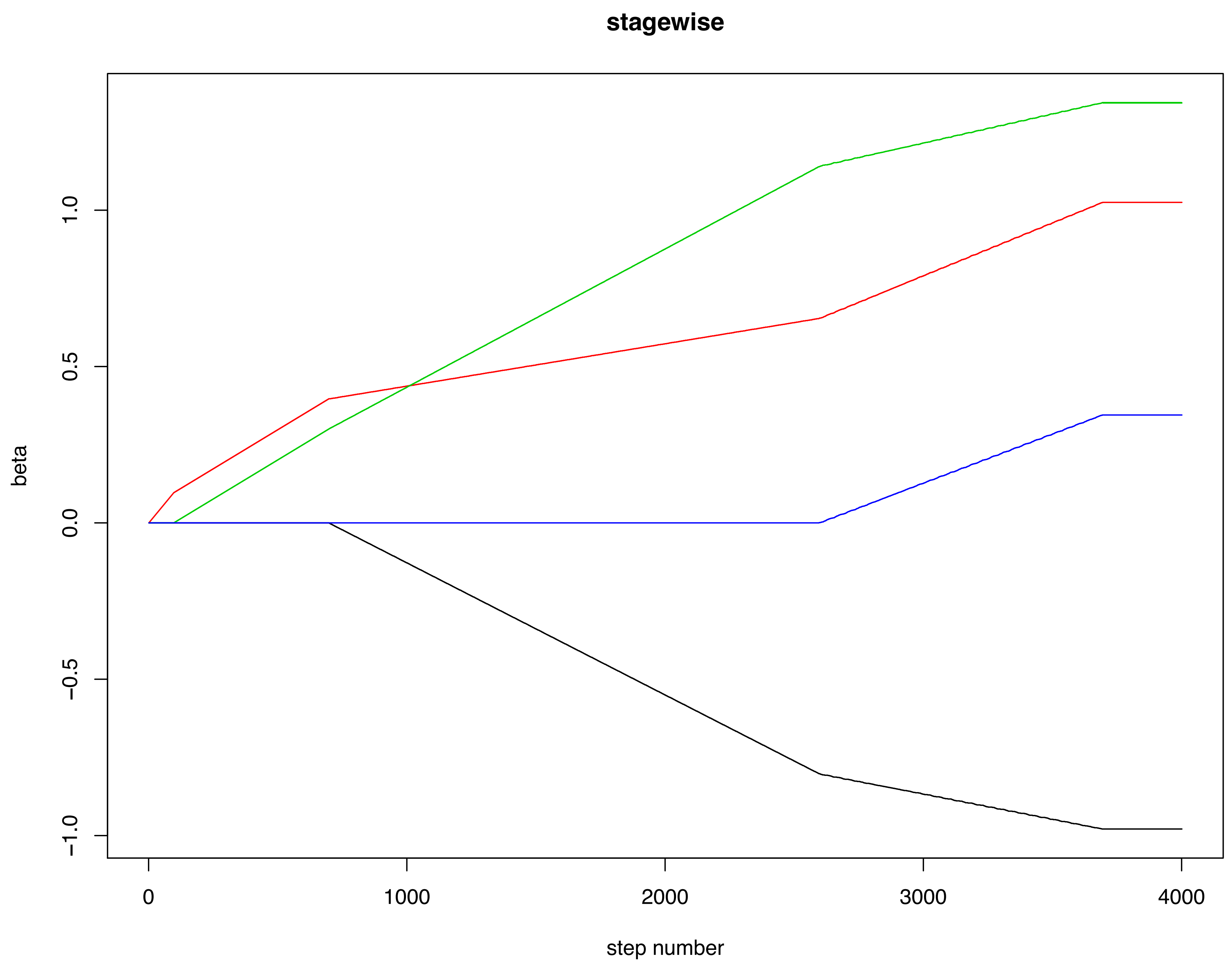
Forward stagewise modeling

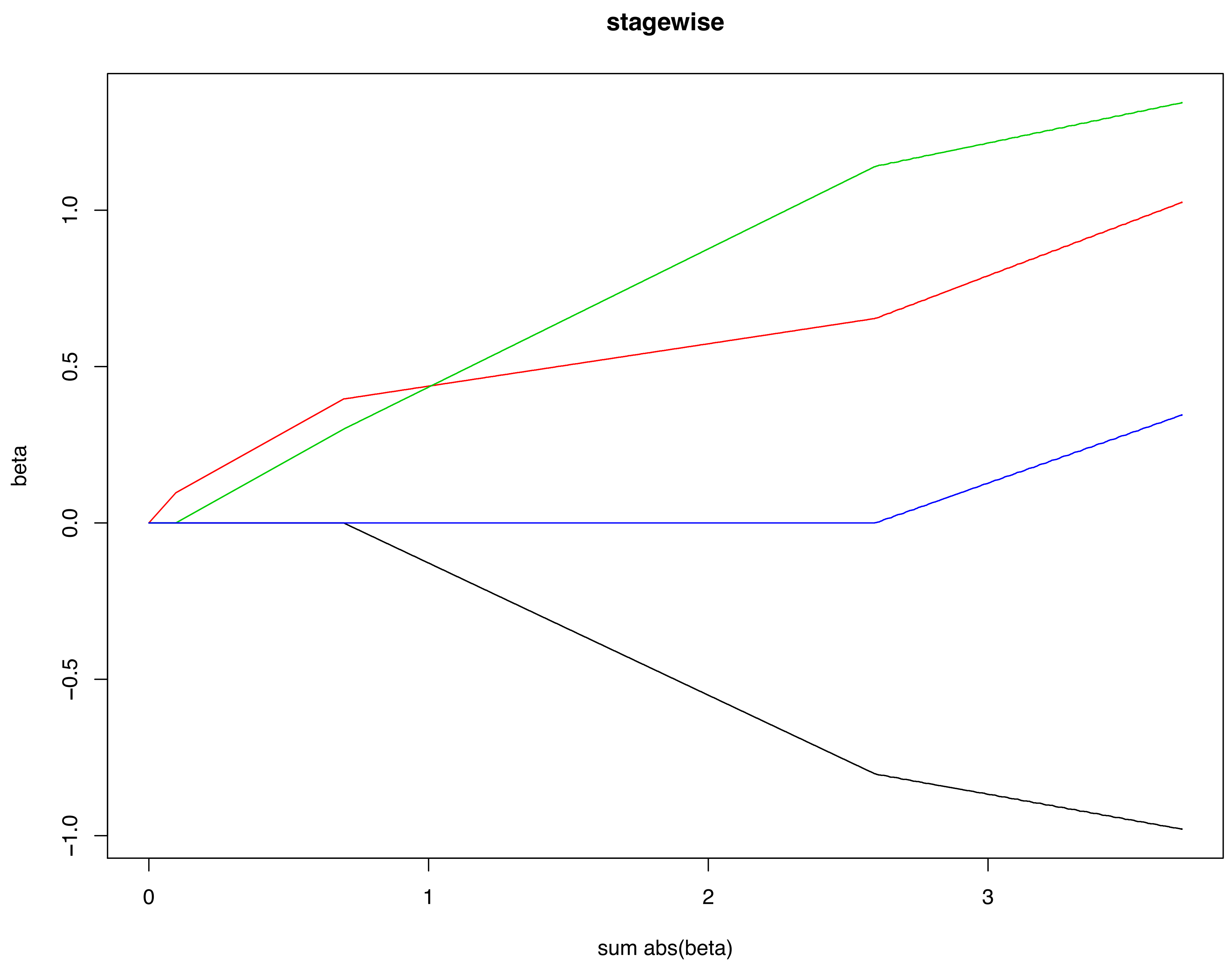
This approach to modeling was inspired by another relatively recent advance in machine learning -- It’s basic idea is to combine many “weak” models to build a powerful “committee”

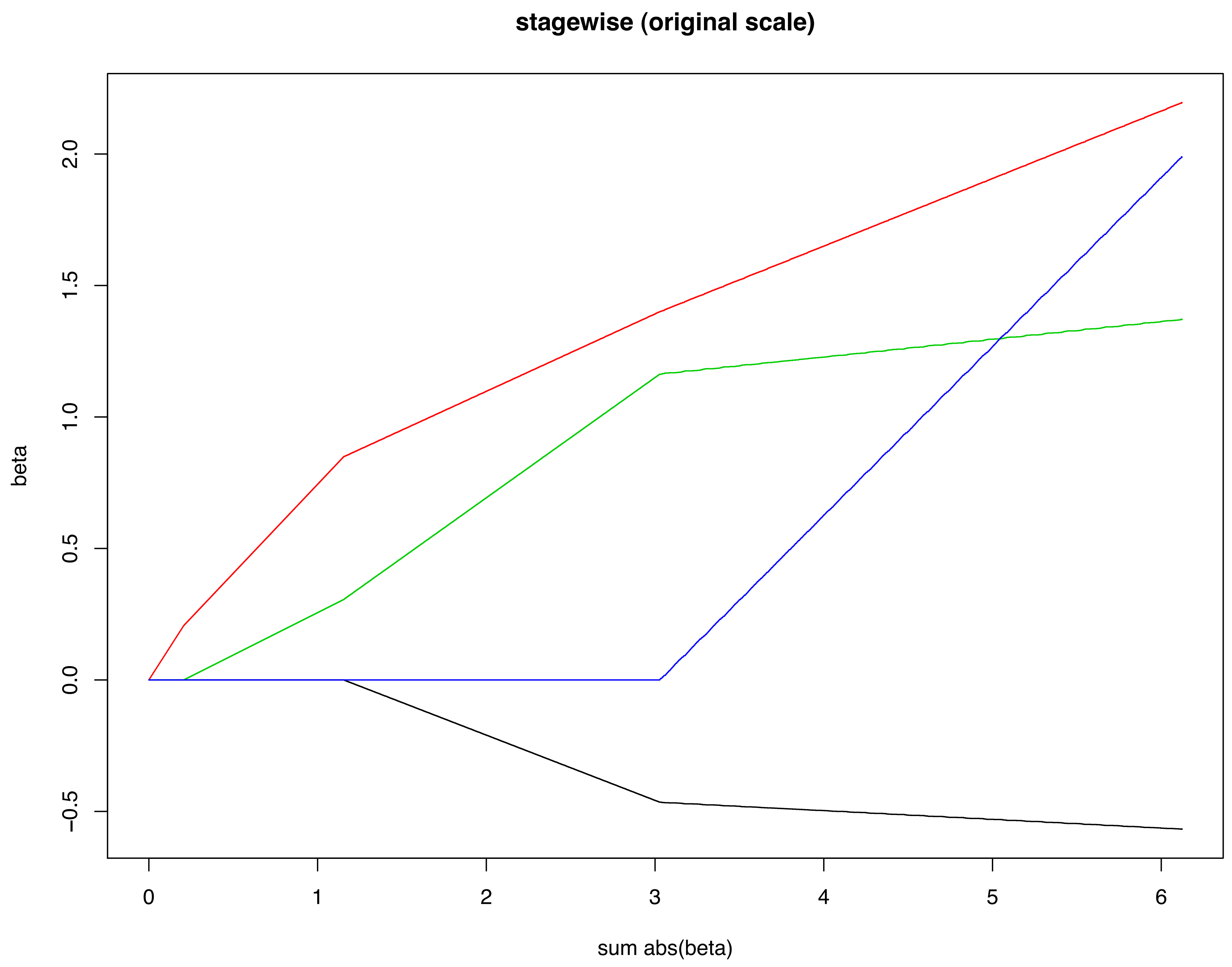
In short, we apply our “learning scheme” (here finding the single variable that best describes the data) successively to the residuals from our current model, our current set of errors

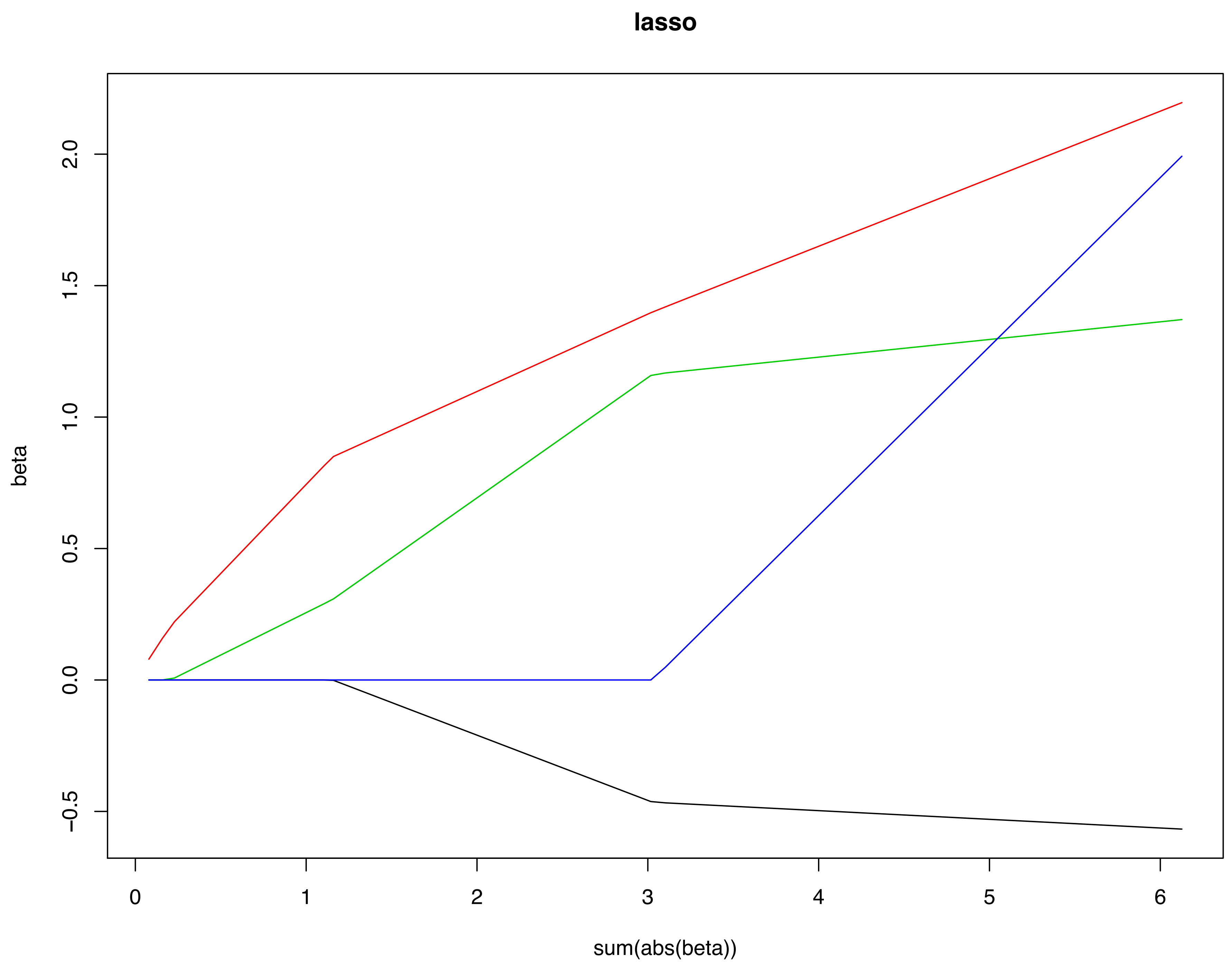
You’ll also see boosting referred to as “slow learning” and from this stagewise example you can see why -- Now, let’s have a look at what this does in practice... Any ideas?

```
y <- y-mean(y)
M <- M-matrix(apply(M,2,mean),ncol=ncol(M),nrow=nrow(M),byrow=T)
M <- M/matrix(apply(M,2,sd),ncol=ncol(M),nrow=nrow(M),byrow=T)

beta <- matrix(0,ncol=ncol(M),nrow=1)
r <- y
eps <- 0.001

lots <- 4000

for(i in 1:lots){

   co <- t(M)%*%r
   j <- (1:ncol(M))[abs(co)==max(abs(co))][1]
   delta <- eps*sign(co[j])

   b <- beta[nrow(beta),]

   b[j] <- b[j] + delta
   beta <- rbind(beta,b)

   r <- r - delta*M[,j]
}

matplot(beta,type="l",lty=1,xlab="step number",ylab="beta",main="stagewise")

matplot(apply(beta,1,function(x) sum(abs(x))),
      beta,type="l",lty=1,xlab="sum abs(beta)",ylab="beta",main="stagewise")

matplot(apply(
        beta/matrix(s,ncol=ncol(beta),nrow=nrow(beta),byrow=T),1,function(x) sum(abs(x))),
        beta/matrix(s,ncol=ncol(beta),nrow=nrow(beta),byrow=T),
        type="l",lty=1,xlab="sum abs(beta)",ylab="beta",main="stagewise (original scale)")
```


stagewise
beta
step number
1.0
0.5
0.0
−0.5
−1.0
0
1000
2000
3000
4000


stagewise
beta
sum abs(beta)
−1.0
−0.5
0.0
0.5
1.0
0
1
2
3


stagewise (original scale)
beta
sum abs(beta)
−0.5
0.0
0.5
1.0
1.5
2.0
0
1
2
3
4
5
6



lasso
beta
sum(abs(beta))
−0.5
0.0
0.5
1.0
1.5
2.0
0
1
2
3
4
5
6

A connection

It would appear that the two paths are nearly (if not exactly) the same, if only because the stagewise approach is jagged -- This is a remarkable fact that allows us to gain insight into a variety of properties of the lasso

For orthogonal problems, the two paths are guaranteed to be the same -- To be precise, if you could imagine taking a very large number of infinitesimally small steps, you would end up walking the same path as the lasso

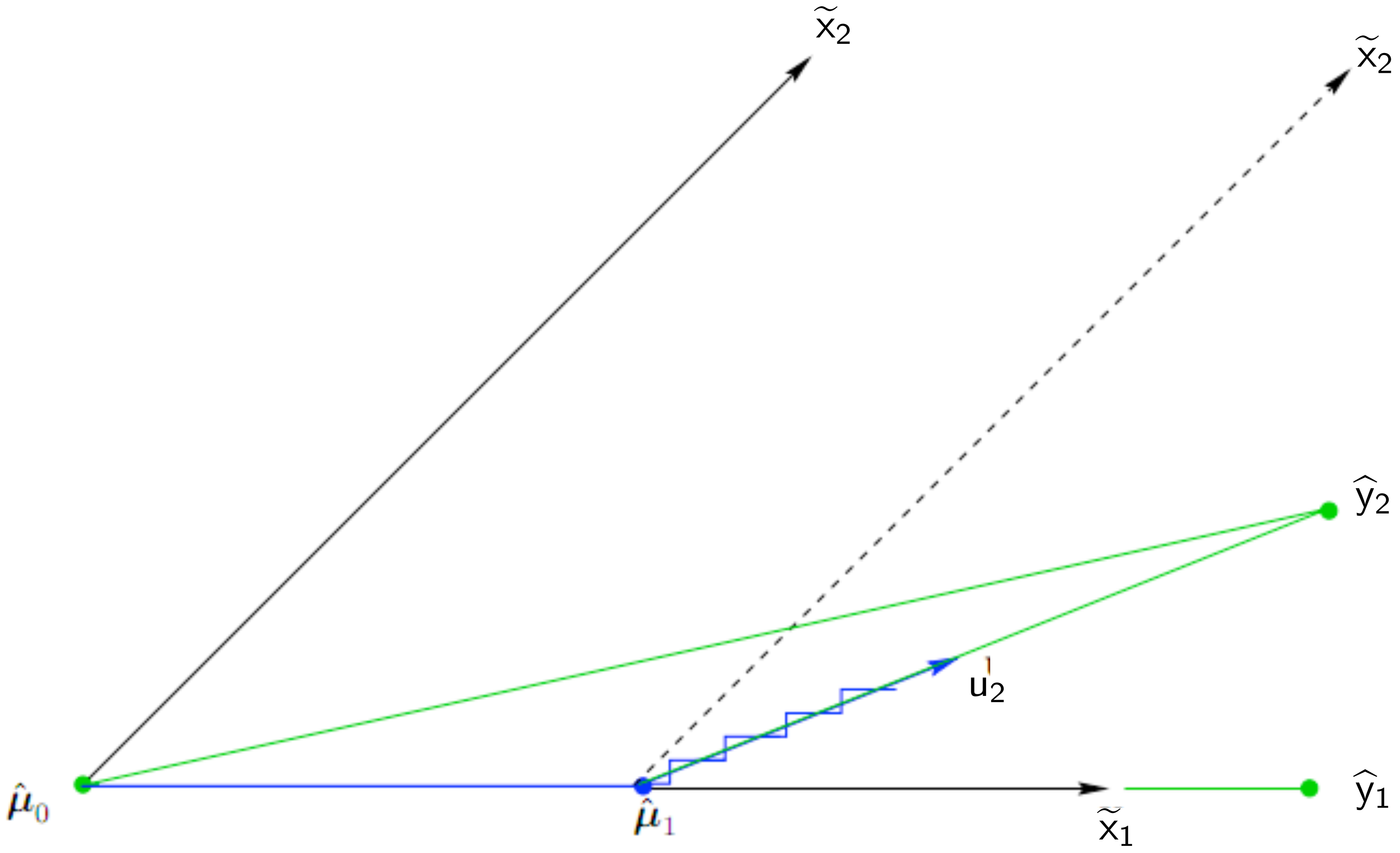
Stagewise, a closer look

On the next page we have a simple diagram of how the algorithm works (at least in 2d) -- We let $\widehat{y}_2$ be the projection of our data y into the space spanned by $\widetilde{x}_1$ and $\widetilde{x}_2$

We start with $\widehat{\mu}_0 = 0$ and $r_0 = \widehat{y}_2 - \widehat{\mu}_0$ and consider the variable ($\widetilde{x}_1$ or $\widetilde{x}_2$) that has the greatest correlation with r_0 -- In this case, it's $\widetilde{x}_1$ and we take steps in that direction

Classical stepwise regression would take a big step and move from $\widehat{\mu}_0$ to $\widehat{y}_1$, the projection of y onto the space spanned by the vector $\widetilde{x}_1$ -- The stagewise approach is different in that it will only go as far as $\widehat{\mu}_1$

At that point, the correlation between $\widehat{y}_2 - \widehat{\mu}_1$ and each of $\widetilde{x}_1$ and $\widetilde{x}_2$ is the same and we let u_2 denote the unit vector lying along the bisector -- The procedure marches stepwise along this path until it reaches $\widehat{y}_2$



$\widetilde{x}_2$
$\widetilde{x}_2$
$\widehat{y}_2$
u_2
$\hat{\boldsymbol{\mu}}_0$
$\hat{\boldsymbol{\mu}}_1$
$\widetilde{x}_1$
$\widehat{y}_1$

Least angle regression

LARS is motivated by the fact that it's relatively easy to identify the points in the figure where the "infinitesimal" version of the stagewise procedure would turn a corner and start following a new vector

The LARS procedure works (essentially) as follows -- We start with all the coefficients in our model set to zero and we look for the variable $\tilde{x}_{j_1}$ that is most correlated (makes the smallest angle) with our response y

We then take the largest step possible in the direction $\tilde{x}_{j_1}$ until some other predictor, $\tilde{x}_{j_2}$, has as much correlation with the accompanying residual -- At that point LARS switches to a direction that is "equiangular" between the two predictors $\tilde{x}_{j_1}$ and $\tilde{x}_{j_2}$

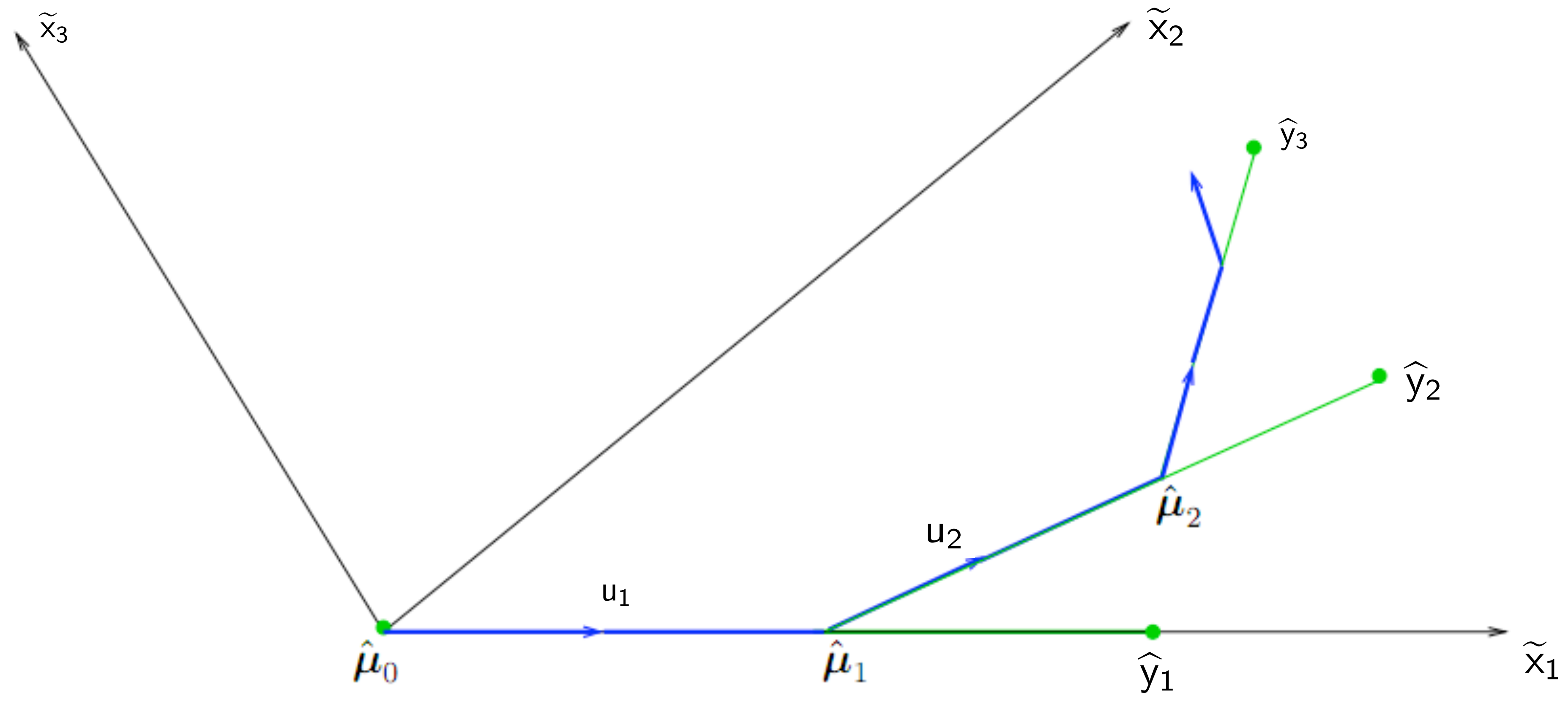
We continue in this way until a third variable $\tilde{x}_{j_3}$ joins the "most correlated" set and we move equiangularly between $\tilde{x}_{j_1}$, $\tilde{x}_{j_2}$ and $\tilde{x}_{j_3}$ -- If we had p original predictors we continue in this way for p-1 steps and take as our last move a jump to the OLS fit using all p predictors

Least angle regression

Without getting into too many details, suppose we have completed k-1 steps of the LARS algorithm -- The kth step will see us introduce a new variable to the set and we can label them $x_{j_1}, \ldots, x_{j_k}$

One can show that the LARS estimate for the kth step $\widehat{\mu}_k$ lies along the line between $\widehat{\mu}_{k-1}$ and $\widehat{y}_k$, the OLS fit to the response using $x_{j_1}, \ldots, x_{j_k}$

In this sense, the LARS estimate is always approaching, but never reaching the OLS estimates at each stage


$\widetilde{x}_3$
$\widetilde{x}_2$
$\widehat{y}_3$
$\widehat{y}_2$
$\hat{\boldsymbol{\mu}}_2$
u_2
u_1
$\hat{\boldsymbol{\mu}}_0$
$\hat{\boldsymbol{\mu}}_1$
$\widehat{y}_1$
$\widetilde{x}_1$

Some facts about least angle regression

First, it is computationally “thrifty” in the words of Efron and company -- It is said to be no harder than fitting a full least squares solution to all p input variables

Next, LARS can be modified so that it can produce the complete paths for both stagewise regression and the lasso -- This has the effect of both providing each with a computationally efficient algorithm as well as offering insight into their operations (LARS moves along a “compromise” direction, equiangular, while the lasso and stagewise restrict strategy in some way)

Finally, the “tuning” parameter for a LARS fit is the number of steps k -- To choose k, Efron et al appeal to a Cp statistic where the “degrees of freedom” for a LARS model after k steps is, well, just k

$$C_p(k) = \frac{RSS}{n} + \frac{2k}{n}\widehat{\sigma}^2$$

where k is the number of steps and $\widehat{\sigma}^2$ is the estimated error variance (from the full model, say if n > p)

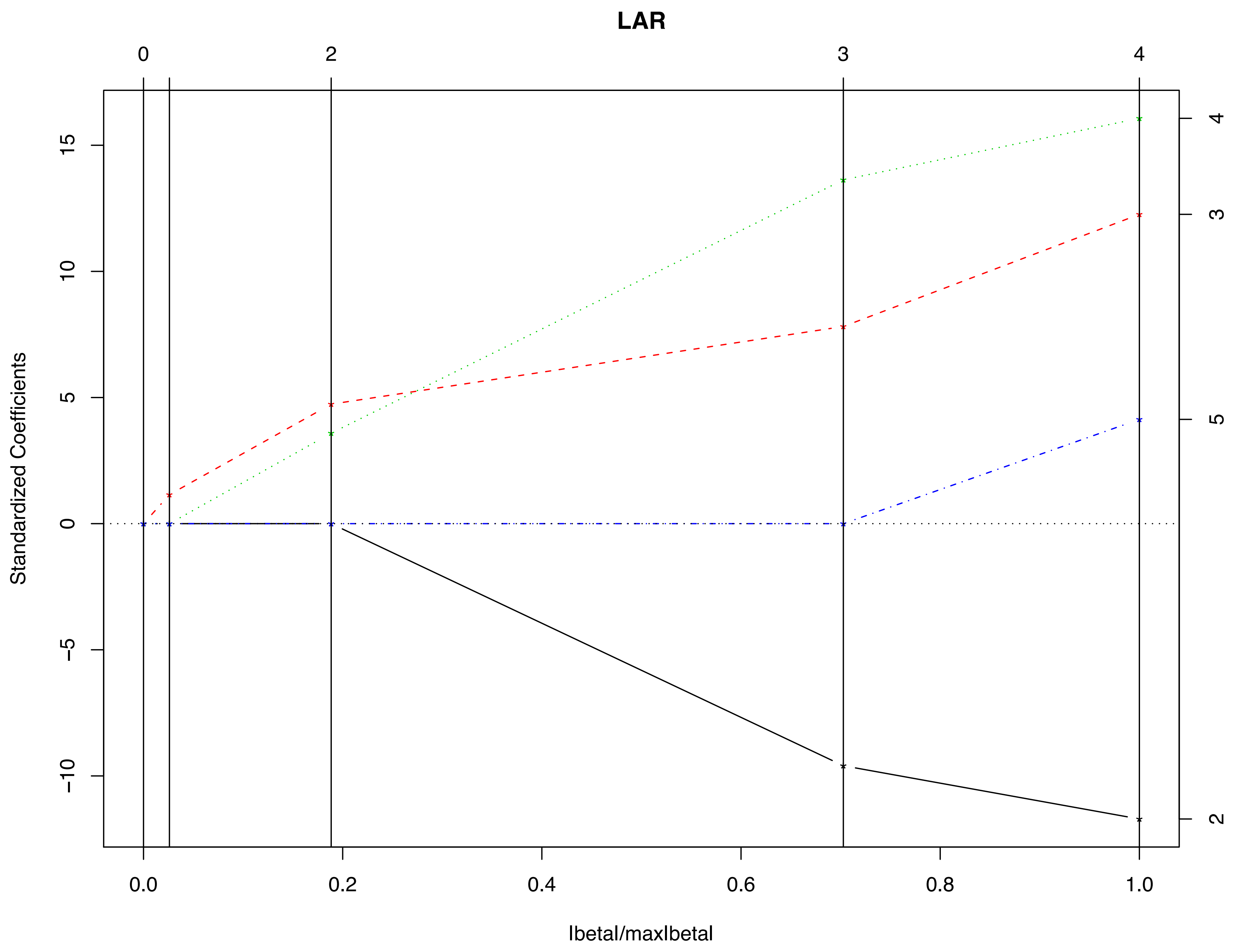
Degrees of freedom

The simple expression for the degrees of freedom in the fit comes from a simulation experiment in which the optimism

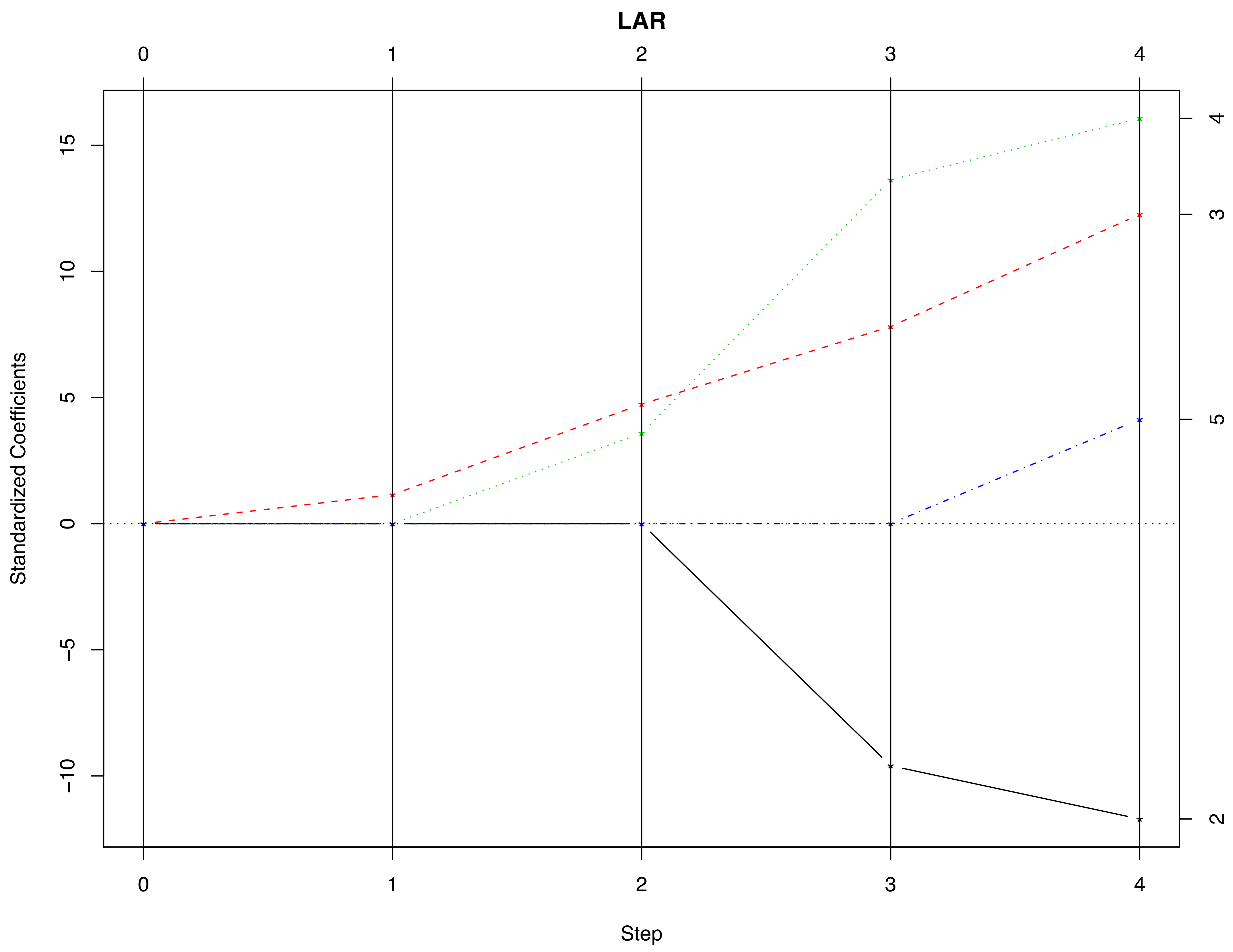
$$\frac{2}{n}\sum_{i=1}^{n}\mathrm{cov}\left(y_i, \widehat{\mu}_{ik}\right)$$

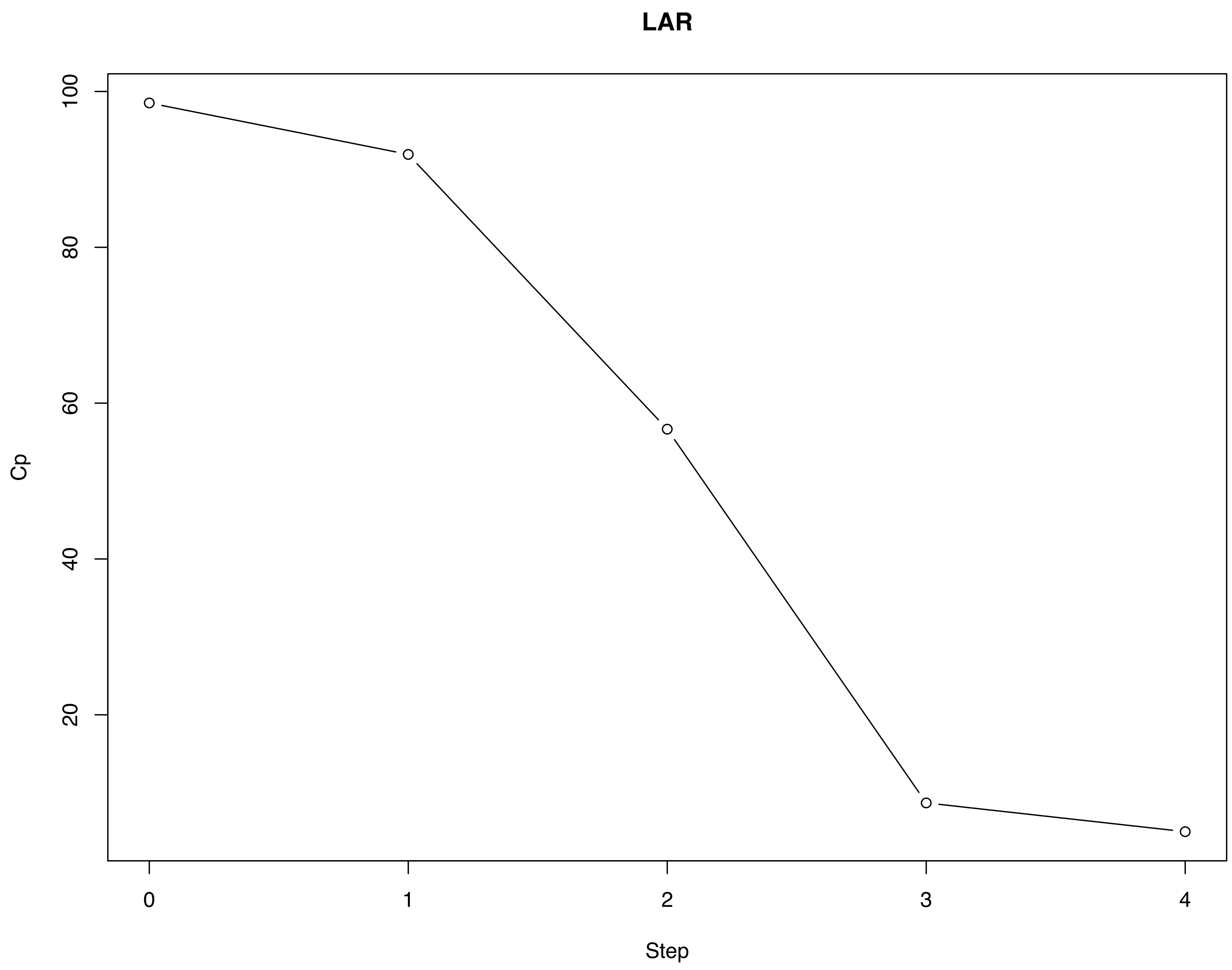
was estimated via the bootstrap (later) and then plots were made against k -- The result can be established theoretically for orthogonal models and for designs satisfying certain (easily checked) properties

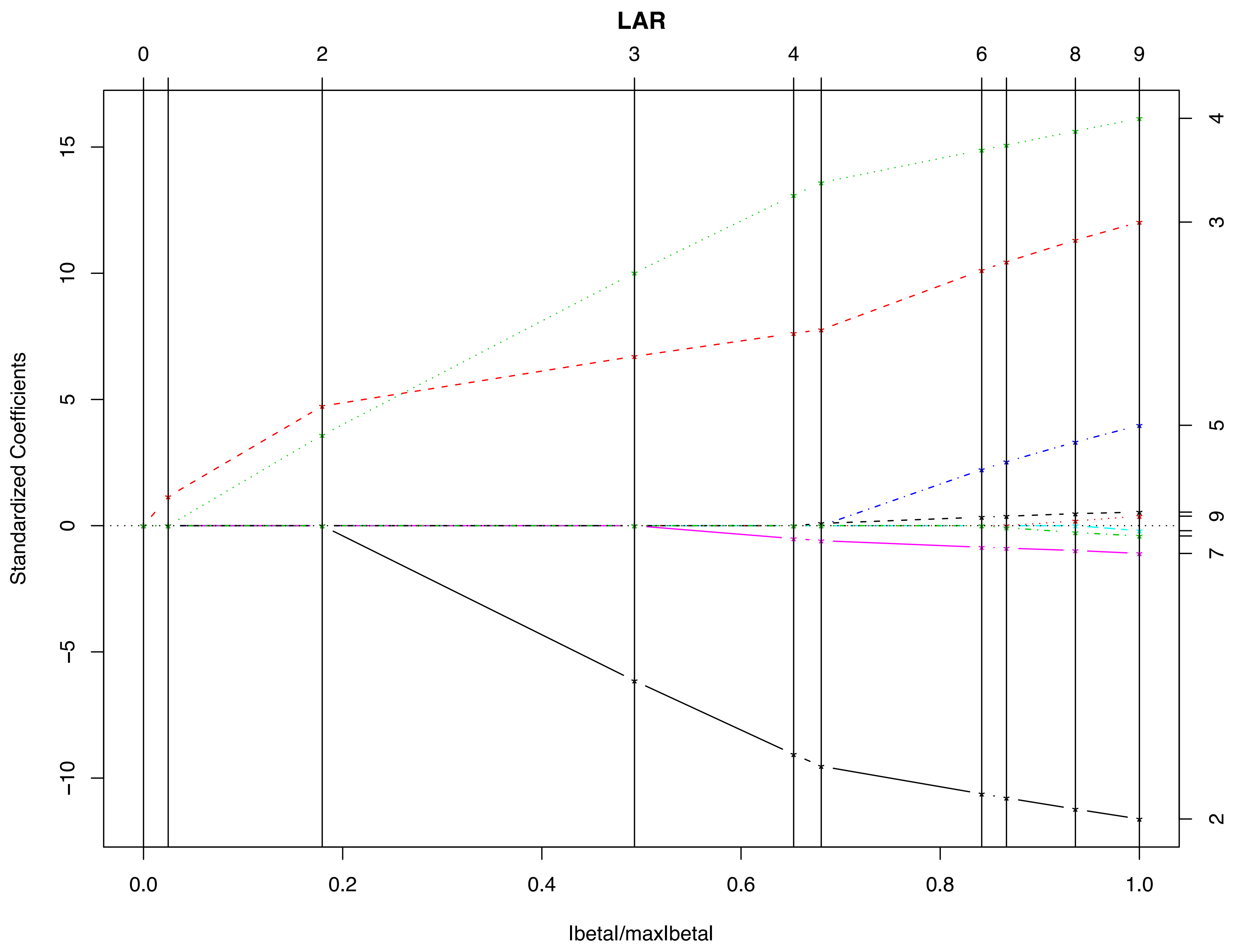
```
library(lars)

M <- model.matrix(ln_death_risk~ln_pop+ln_fert+ln_events+hdi,data=vul)
y <- vul$ln_death_risk

fit <- lars(M,y,"lar")

plot(fit)
plot(fit,xvar="step")

plot(fit,plottype="Cp",xvar="step")

# add noise variables

M <- cbind(M,matrix(rnorm(nrow(M)*5),ncol=5))
fit <- lars(M,y,"lar")

plot(fit)
plot(fit,plottype="Cp",xvar="step")
```

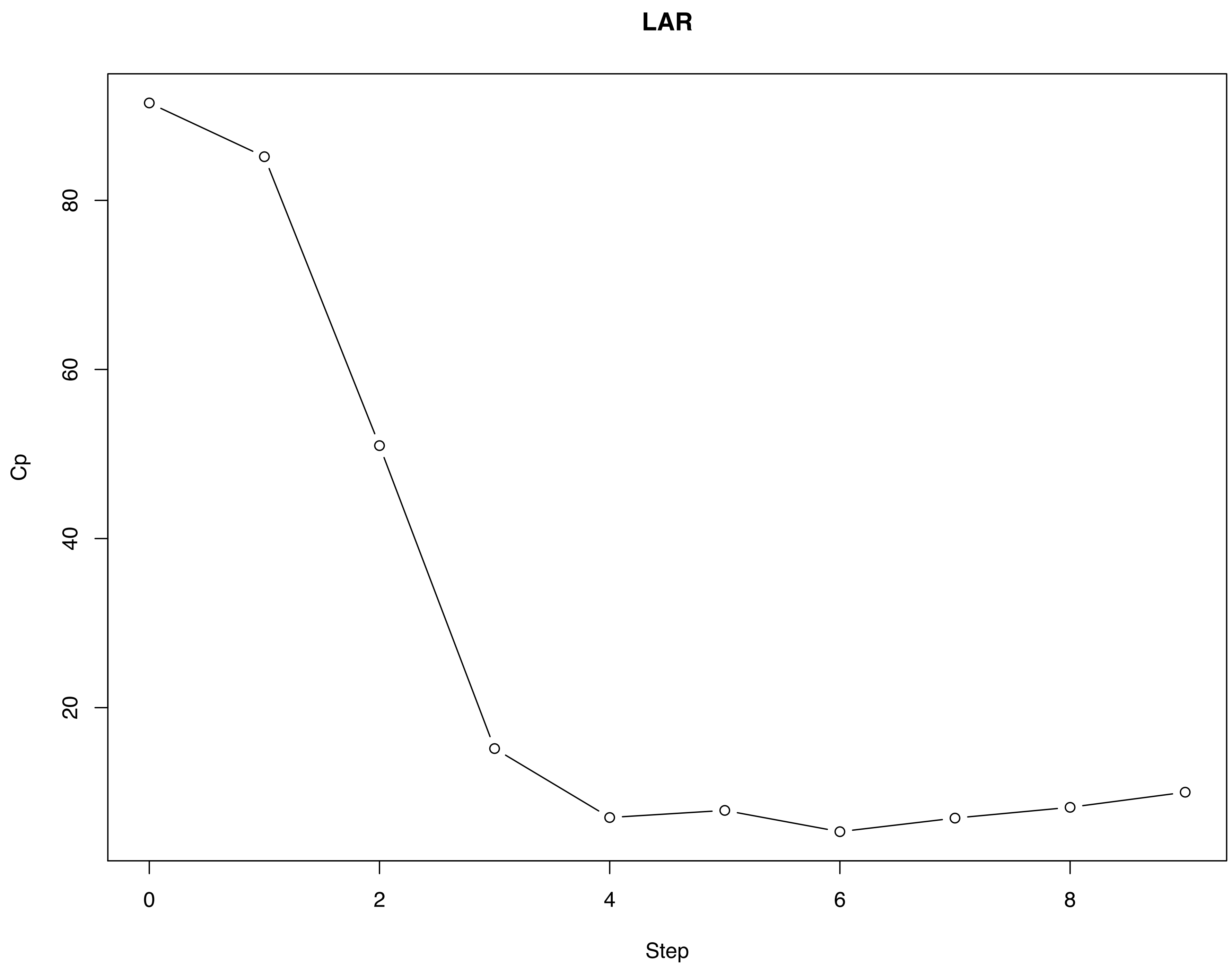


LAR
0
2
3
4
Standardized Coefficients
|beta|/max|beta|
15
10
5
0
−5
−10
0.0
0.2
0.4
0.6
0.8
1.0
4
3
5
2


LAR
0
1
2
3
4
Standardized Coefficients
15
10
5
0
−5
−10
4
3
5
2
0
1
2
3
4
Step


LAR
Cp
Step
100
80
60
40
20
0
1
2
3
4


LAR
0
2
3
4
6
8
9
Standardized Coefficients
15
10
5
0
−5
−10
0.0
0.2
0.4
0.6
0.8
1.0
|beta|/max|beta|
4
3
5
9
7
2


LAR
Cp
Step
0
2
4
6
8
20
40
60
80

To sum

Stepwise regression: Pick the predictor most closely correlated with y and add it completely to the model

Forward stagewise: Pick the predictor most correlated with y and increment the coefficient for that predictor

Least angle regression: Pick the predictor most correlated with y but add it to the model only to the extent that it has greater correlation than the others -- Move in the least-squares direction until another variable is as correlated

Quo vadis?

We started with ordinary least squares as a methodology -- We studied some of its basic properties, developing an intuition for the various pieces that went into the fitting procedure

We discussed some diagnostics that helped us assess when things might be going wrong -- Perhaps we need to add something to a model or consider dropping something

We then looked at penalization and how that affects a fit, first from the standpoint of stability and then from pure predictive power -- We saw two different kinds of penalization that resulted in very different kinds of fits

We finally made a connection between these penalties and a relatively new procedure for model building called least angle regression -- Ultimately our geometric perspective came in handy here!