Linear Regression Model

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_{p-1} X_{i,p-1} + \varepsilon_i, \quad i = 1,\ldots n$$

- Y_i is the response for the i^{th} subject
- $X_{i1}, X_{i2}, \ldots X_{i,p-1}$ are the values of the predictor variables for the i^{th} subject. Some can be transformed predictors: $X_{i2}=\text{Log}(X_{i1})$ or interactions $X_{i3}= X_{i1}X_{i2}$ or polynomial expansion.
- $\beta_1, \beta_2, \ldots \beta_{p-1}$ are *unknown parameters* to be estimated from the data (they are also called *partial regression coefficients*)
- Regression (response) surface:

$$E(Y_i) = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_{p-1} X_{i,p-1}$$

- $E(\varepsilon_i)=0$, $\text{Cov}(\varepsilon_i, \varepsilon_j)=0$ for $i \neq j$, $\text{Var}(\varepsilon_i)=\sigma^2>0$

Linear Regression Model(Matrix form)

$$\underset{n\times 1}{Y} = \underset{n\times p}{X}\ \underset{p\times 1}{\beta} + \underset{n\times 1}{\varepsilon}$$

- Y – vector of responses
- β- vector of parameters
- X – matrix of constants (design matrix)
- $\varepsilon \sim N(0,\sigma^2 I)$ and hence $Y \sim N(X\beta,\sigma^2 I)$

Estimation of regression coefficients

- Least square estimates are obtained by minimizing the sum of distances from the points to the regression plane:

$$Q = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_{i1} - \beta_2 X_{i2} - \ldots - \beta_{p-1} X_{i,p-1})^2$$

- Denote the vector of the least squares estimated regression coefficients as b:

$$\underset{p\times 1}{b} = \begin{pmatrix} b_0 \\ b_1 \\ \ldots \\ b_{p-1} \end{pmatrix}$$

- Least squares normal equations:

$$X'Xb = X'Y$$

- Least squares estimates

$$\underset{p\times 1}{b} = \underset{p\times p}{(X'X)^{-1}} \underset{p\times 1}{(X'Y)}$$

Maximum-likelihood estimates are the same

Fitted values and residuals

$$\underset{n\times 1}{\hat{Y}} = \begin{pmatrix} \hat{Y}_1 \\ \hat{Y}_2 \\ \dots \\ \hat{Y}_n \end{pmatrix} = \underset{n\times p}{X}\ \underset{p\times 1}{b} = X(X'X)^{-1}X'Y = \underset{n\times n}{H}\ \underset{n\times 1}{Y}$$

$$H = X(X'X)^{-1}X' \qquad \text{(Hat matrix)}$$

$$\underset{n\times 1}{e} = \begin{pmatrix} e_1 \\ e_2 \\ \dots \\ e_n \end{pmatrix} = \underset{n\times 1}{Y} - \underset{n\times 1}{\hat{Y}} = \underset{n\times 1}{Y} - \underset{n\times p}{X}\ \underset{p\times 1}{b} = (\underset{n\times n}{I} - \underset{n\times n}{H})\ \underset{n\times 1}{Y}$$

$$\sigma^2\{e\} = \sigma^2(I-H), \qquad s^2\{e\} = MSE(I-H)$$

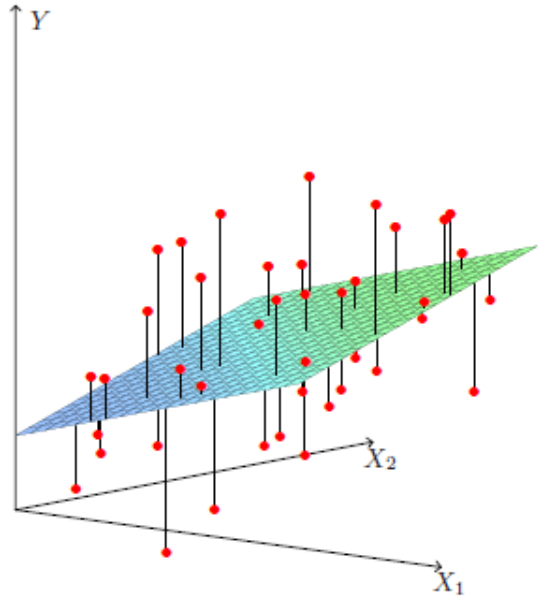


FIGURE 3.1. *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.*

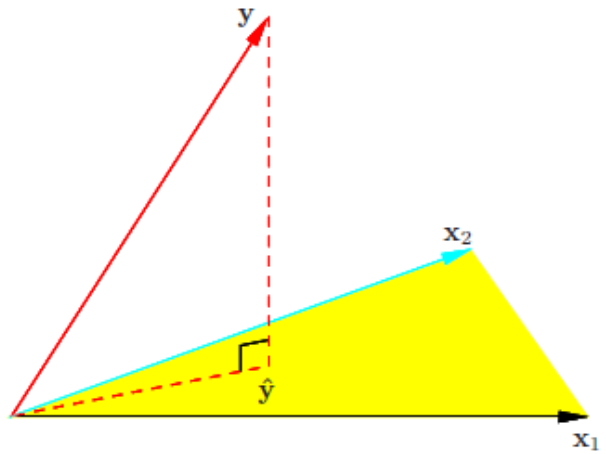


FIGURE 3.2. *The N-dimensional geometry of least squares regression with two predictors. The outcome vector $\mathbf{y}$ is orthogonally projected onto the hyperplane spanned by the input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. The projection $\hat{\mathbf{y}}$ represents the vector of the least squares predictions*

Sums of squares and mean squares

$$SSR = b' X' Y - \frac{1}{n} Y' JY = Y'[H - \frac{1}{n} J]Y$$

$$MSR = \frac{SSR}{p-1}$$

$$SSE = Y'Y - b' X' Y = Y'(I - H)Y$$

$$MSE = \frac{SSE}{n-p}$$

$$SSTO = Y'Y - \frac{1}{n} Y' JY = Y'[I - \frac{1}{n} J]Y$$

ANOVA table

Source of variation	df	SS	MS	F
Regression	p-1	SSR	MSR	MSR/MSE
Error	n-p	SSE	MSE	
Total	n-1	SSTO		

Gauss-Markov Theorem

Consider any linear combination of the β's: $\theta = a^T \beta$

The least squares estimate of θ is:

$$\hat{\theta} = a^T \hat{\beta} = a^T (X^T X)^{-1} X^T y$$

If the linear model is correct, this estimate is unbiased (*X* fixed):

$$E(\theta) = E(a^T (X^T X)^{-1} X^T y) = a^T (X^T X)^{-1} X^T X\beta = a^T \beta$$

Gauss-Markov states that for any other linear unbiased estimator $\tilde{\theta} = c^T y$

i.e., $E(c^T y) = E(a^T \beta)$,

$$\mathrm{Var}(a^T \hat{\beta}) \leq \mathrm{Var}(c^T y)$$

Of course, there might be a *biased* estimator with lower MSE…

bias-variance

For any estimator $\tilde{\theta}$:

$$\begin{aligned}\mathrm{MSE}(\tilde{\theta}) &= E(\tilde{\theta}-\theta)^2 \\ &= E(\tilde{\theta}-E(\tilde{\theta})+E(\tilde{\theta})-\theta)^2 \\ &= E(\tilde{\theta}-E(\tilde{\theta}))^2 + E(E(\tilde{\theta})-\theta)^2 \\ &= Var(\tilde{\theta}) + (E(\tilde{\theta})-\theta)^2\end{aligned}$$

bias

Note MSE closely related to prediction error:

$$E(Y_0 - x_0^T\tilde{\beta})^2 = E(Y_0 - x_0^T\beta)^2 + E(x_0^T\tilde{\beta} - x_0^T\beta)^2 = \sigma^2 + MSE(x_0^T\tilde{\beta})$$

Modern procedures for model selection to avoid overfitting

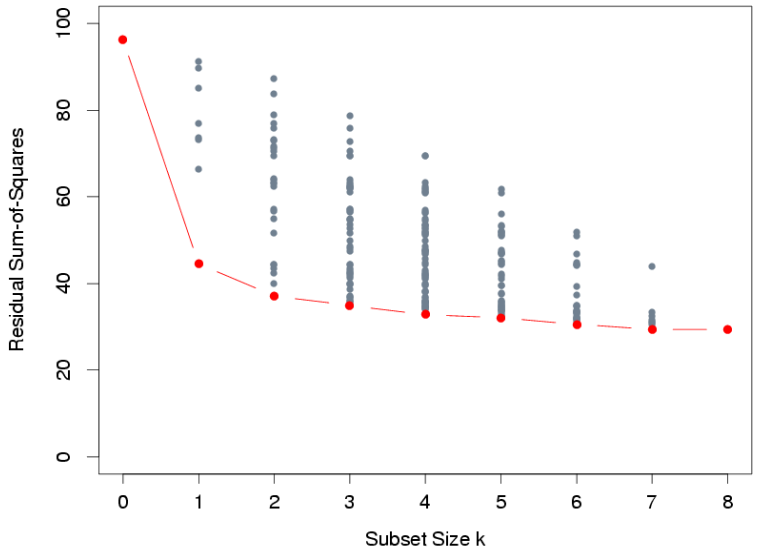
When there are too many *X*'s, noise X's can improve the fit just by chance. Overfitting causes bad prediction. Two avoid it we do:

1. Classical variable selection
   - Backward, Foreword, Stepwise methods,
   - All subsets
   - Best criteria: AIC, MAIC, BIC

2. Shrinkage Methods:
   Ridge Regression
   LASSO
   ELASTIC NET/GLM NET

Subset Selection

•Standard "all-subsets" finds the subset of size k, k=1,…,p, that minimizes RSS:



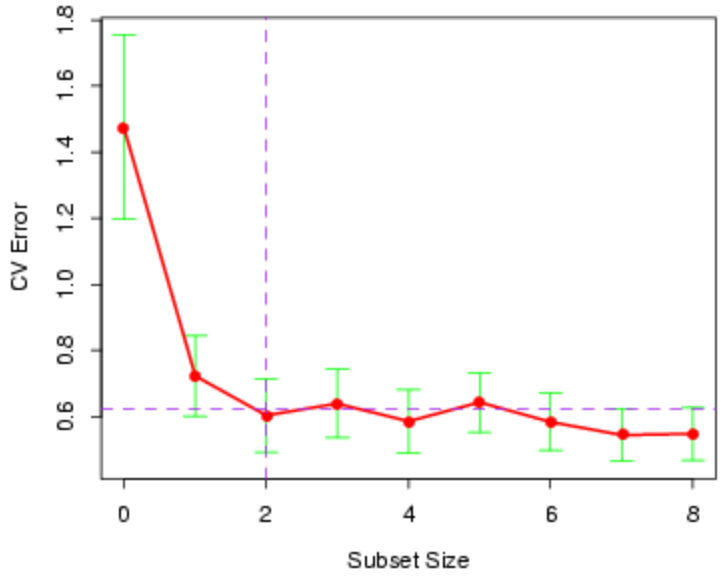
•In R function "leaps" will do it

•Choice of subset size requires tradeoff – AIC, BIC, marginal likelihood, cross-validation, etc.

•"Leaps and bounds" is an efficient algorithm to do all-subsets

Cross-Validation

- e.g. 10-fold cross-validation:
  - Randomly divide the data into ten parts
  - Train model using 9 tenths and compute prediction error on the remaining 1 tenth
  - Do these for each 1 tenth of the data
  - Average the 10 prediction error estimates



"One standard error rule"

pick the simplest model within one standard error of the minimum

Shrinkage Methods

- Subset selection is a discrete process – individual variables are either in or out
- This method can have high variance – a different dataset from the same source can result in a totally different model
- Shrinkage methods allow a variable to be partly included in the model. That is, the variable is included but with a shrunken co-efficient.

Ridge Regression

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

subject to: $$\sum_{j=1}^{p} \beta_j^2 \le s$$

Equivalently:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left(\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right)$$

This leads to:

$$\hat{\beta}^{\text{ridge}} = (X^T X + \gamma I)^{-1} X^T y$$

works even when $X^T X$ is singular

Choose λ by cross-validation. Predictors should be centered.

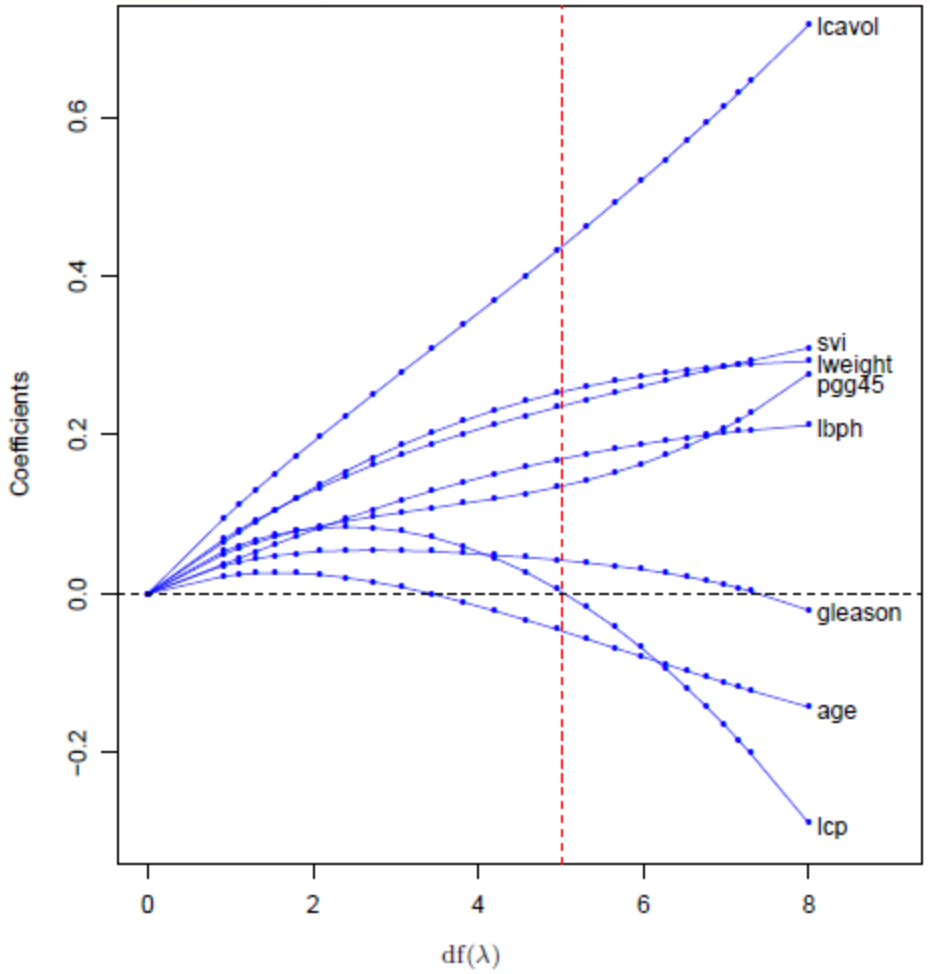


FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter* λ *is varied. Coefficients are plotted versus* $\mathrm{df}(\lambda)$*, the effective degrees of freedom. A vertical line is drawn at* $\mathrm{df} = 5.0$*, the value chosen by cross-validation.*

Ridge Regression = Bayesian Regression

$$y_i \sim N(\beta_0 + x_i^T \beta, \sigma^2)$$

$$\beta_j \sim N(0, \tau^2)$$

same as ridge with $\lambda = \sigma^2 / \tau^2$

The Lasso

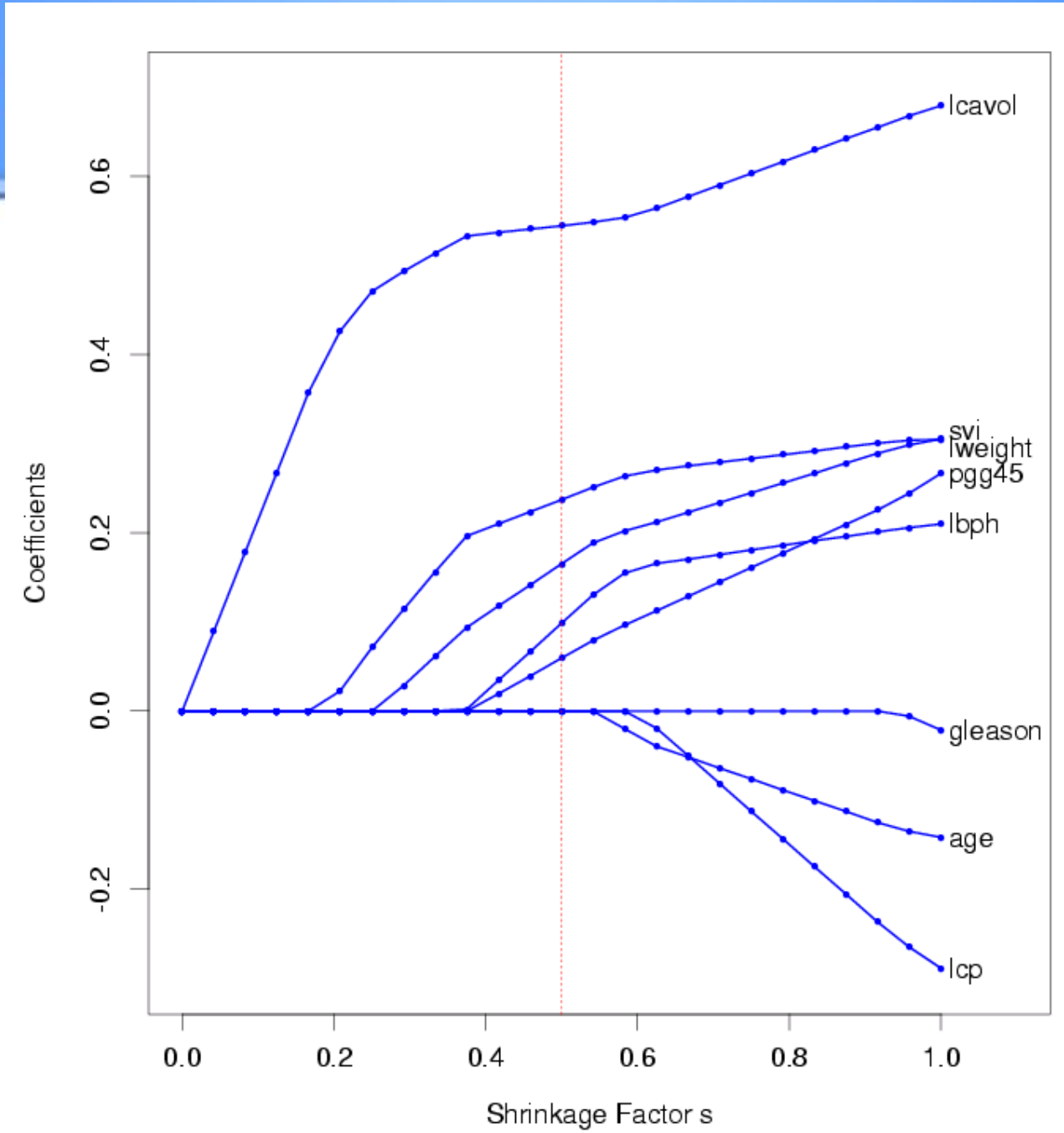
$$\hat{\beta}^{Lasso} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

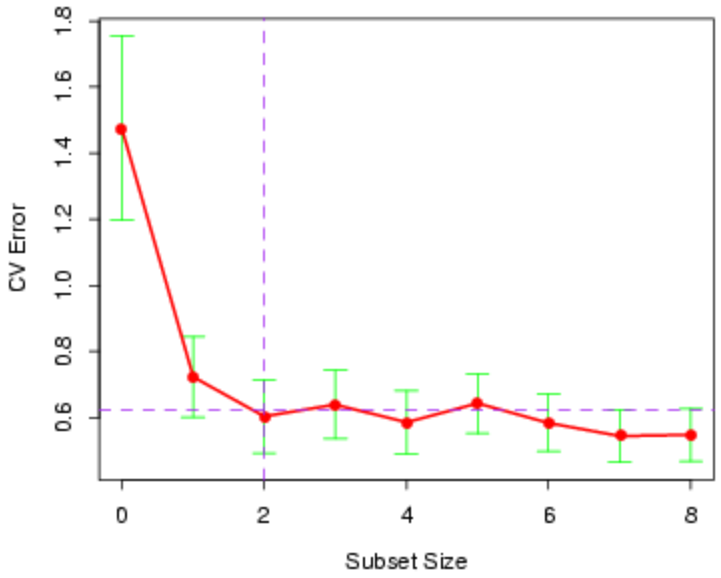
subject to: $$\sum_{j=1}^{p} |\beta_j| \leq s$$

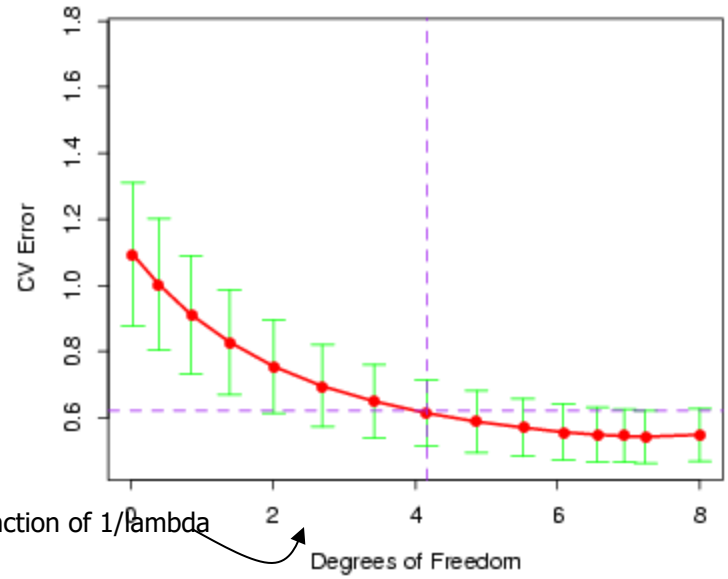
Quadratic programming algorithm needed to solve for the parameter estimates. Choose *s* via cross-validation.

$$\tilde{\beta} = \arg\min_{\beta} \left(\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right)$$

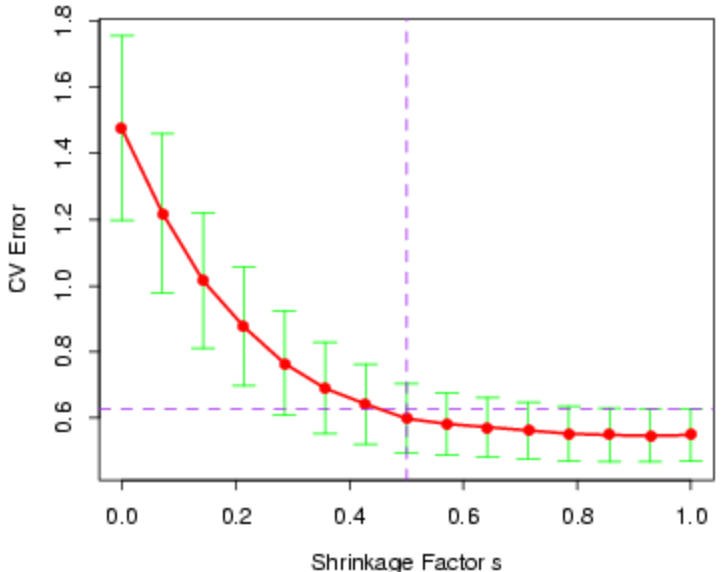
q=0: var. sel.
q=1: lasso
q=2: ridge
Learn *q*?

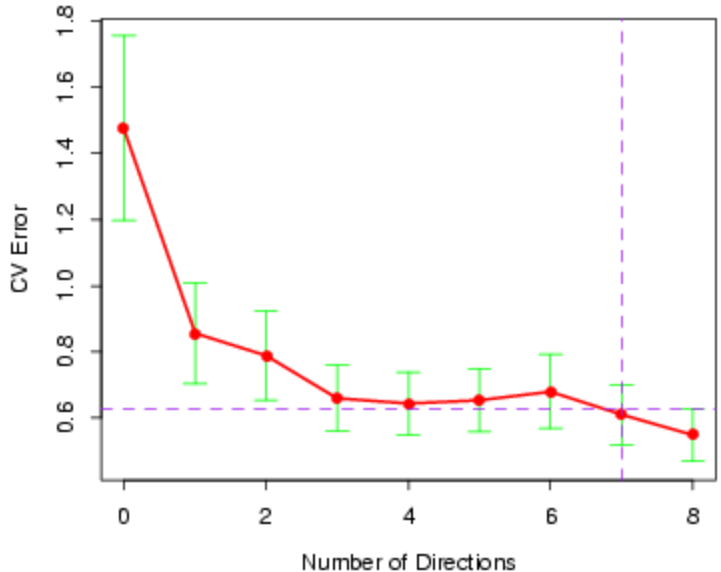

lcavol
svi
lweight
pgg45
lbph
gleason
age
lcp
Coefficients
0.6
0.4
0.2
0.0
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
Shrinkage Factor s





function of 1/lambda





Elastic Net

$$\hat{\beta}^{Enet} = \arg\min_{\beta} \left(\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda(\alpha \sum_{j=1}^{p} \beta_j^2 + (1-\alpha) \sum_{j=1}^{p} |\beta_j|) \right)$$

Quadratic programming algorithm needed to solve for the parameter estimates.

Choose λ via cross-validation.

The parameter α is given in advance and therefore must be chosen by user before applying the elastic net.

Simple R program for final

```
## y_train, y_test are your responses for training and testing.
## To make x_train and x_test use the following code were
## x_hbeat are you features constructed from hbeat fitbit data
## x_sleep are you features constructed from sleep fitbit data
## x_stp are you features constructed from stp fitbit data
## x_med are you features constructed from med fitbit data
## make sure that these datasets are all numeric matrices not data.frames

## weights are inverse to number of columns so the sum is the same for each data set
w1 = 1/ncol(x_hbeat)
w2 = 1/ncol(x_sleep)
w3 = 1/ncol(x_stp)
w4 = 1/ncol(x_med)

## Standardize the datasets by the weights
x_hbeat_std = apply(x_hbeat,2,function(x)x/sd(x)) * w1
x_sleep_std = apply(x_sleep,2,function(x)x/sd(x)) * w2
x_stp_std = apply(x_stp,2,function(x)x/sd(x))  * w3
x_med_std = apply(x_med,2,function(x)x/sd(x))  *w4

## combine datasets
x_train = cbind(x_hbeat_std,x_sleep_std ,x_stp_std,x_med_std )

## Chose the last row as the test set
x_test = x_train[nrow(x_train),,drop=F]
x_train = x_train[-nrow(x_train),]

## Run glm net
mx0 <- cv.glmnet(x_train,y_train, alpha=1,family = "gaussian",standardize=F)
ll = c( mx0$lambda.min,mx0$lambda.1se)
mx0 <- glmnet(x_train,y_train , alpha=1,family = "gaussian",standardize=F, lambda= ll[2])
y_test_pred = predict(mx0,newx=x_test)
```