

## Lecture 3

# Penalized Methods

# Review of Linear Regression Model

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

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

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

- $E(\varepsilon_i) = 0$ ,  $\text{Cov}(\varepsilon_i, \varepsilon_j) = 0$  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
- $\beta$ - vector of parameters
- $X$  – matrix of constants (design matrix)
- $\varepsilon \sim N(0, \sigma^2 I_p)$  and hence  $Y \sim N(X\beta, \sigma^2 I_p)$  , where  $I_p$  is p-dimensional identity matrix.

# Estimation of regression coefficients

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

$$Q = \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 X_{i1} - \beta_2 X_{i2} - \dots - \beta_{p-1} X_{i,p-1})^2 \implies \hat{\beta} = \underset{\beta}{\operatorname{argmin}} \|y - X\beta\|^2$$

- Denote the vector of the least squares estimated regression coefficients as  $\hat{\beta}$ :

$$\hat{\beta}_{p \times 1} = \begin{pmatrix} b_0 \\ b_1 \\ \dots \\ b_{p-1} \end{pmatrix}$$

- Least squares normal equations:

$$X' X b = X' Y$$

- Least squares estimates

$$\hat{\beta}_{p \times 1} = (X' X)_{p \times p}^{-1} (X' Y)_{p \times 1}$$

Maximum-likelihood estimates are the same

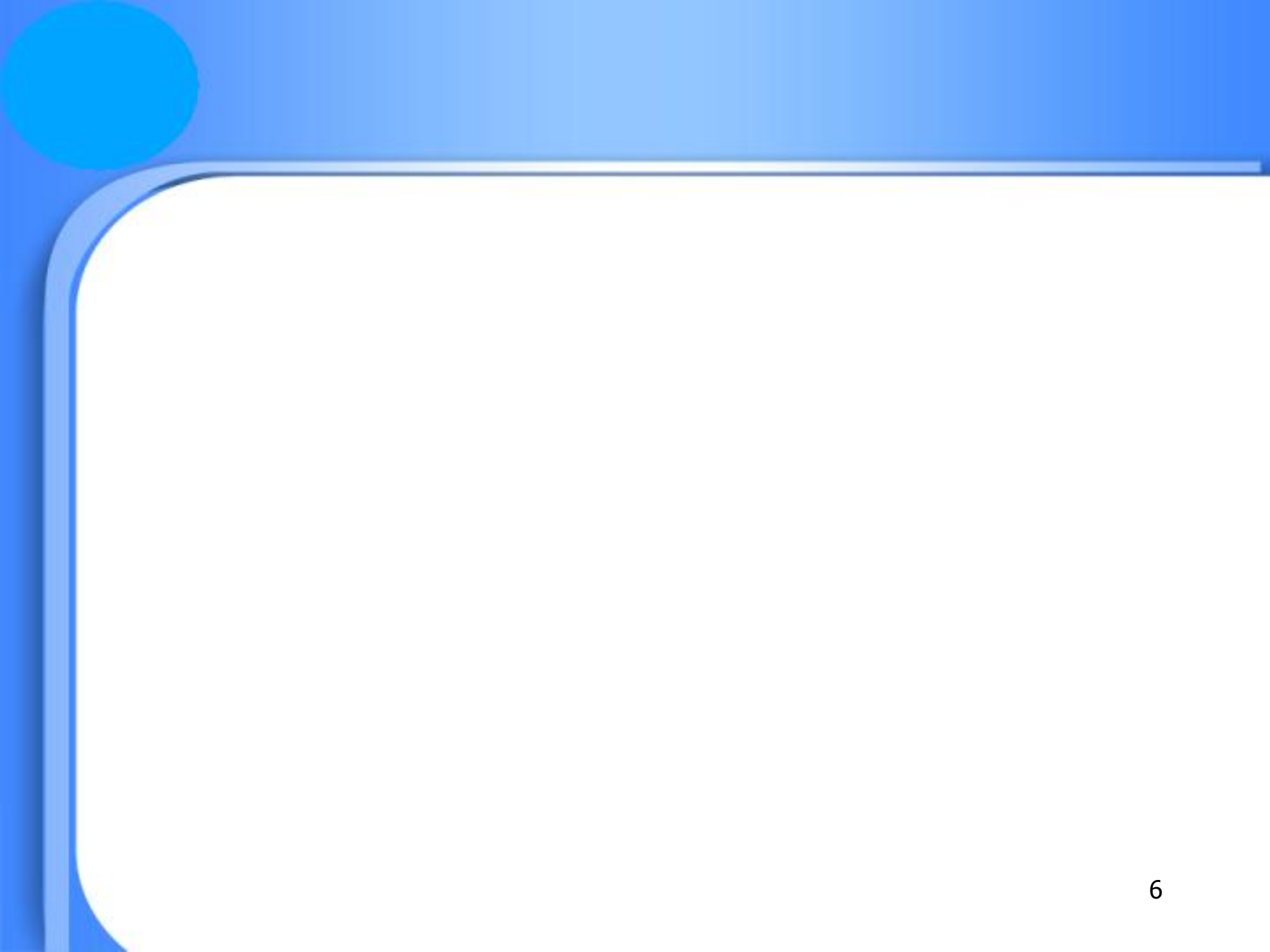
# Fitted values and residuals

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

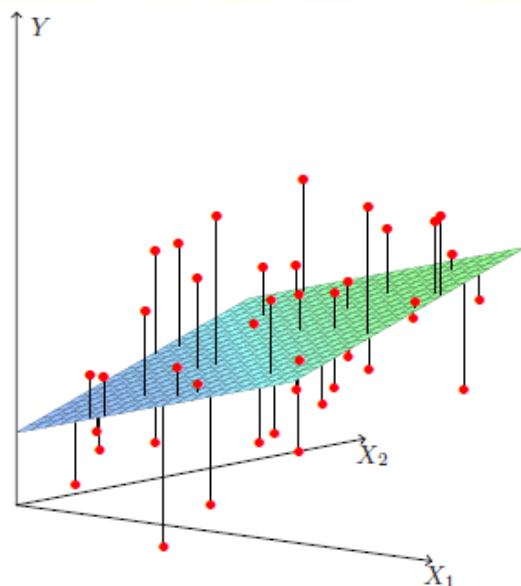
$$H = X(X'X)^{-1}X' \quad (\text{Hat matrix})$$

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

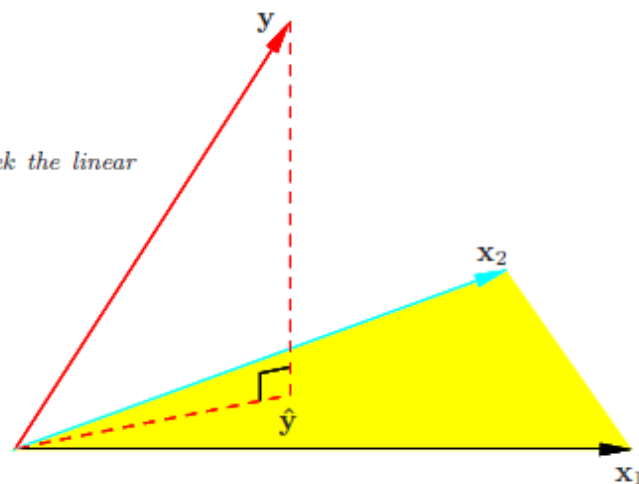
$$\sigma^2\{e\} = \sigma^2(I - H), \quad s^2\{e\} = \text{MSE}(I - H)$$



# Geometric Interpretation



**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  $y$  is orthogonally projected onto the hyperplane spanned by the input vectors  $x_1$  and  $x_2$ . The projection  $\hat{y}$  represents the vector of the least squares predictions

# Sums of squares and mean squares

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

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

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

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

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



# 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  $\beta$ 's:  $\theta = a^T \beta$

The least squares estimate of  $\theta$  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.,  $E(c^T y) = E(a^T \beta)$ ,

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

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

# bias-variance

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

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

bias



Note MSE closely related to prediction error:

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

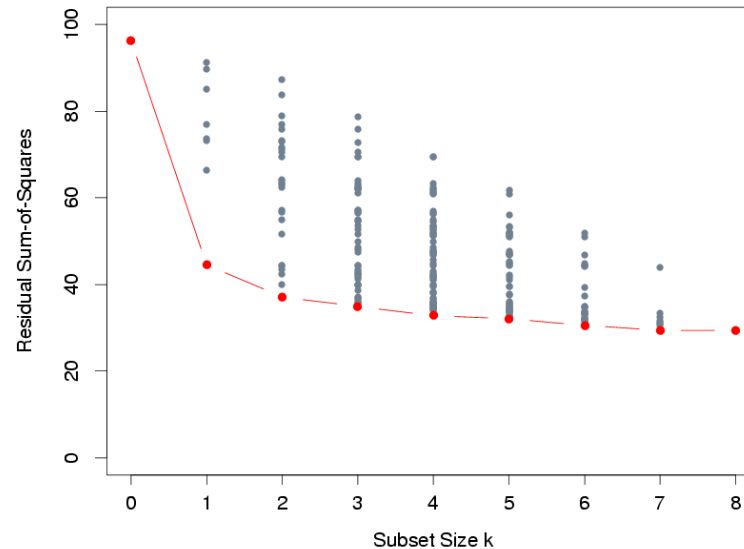
# Modern procedures for model selection to avoid overfitting

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

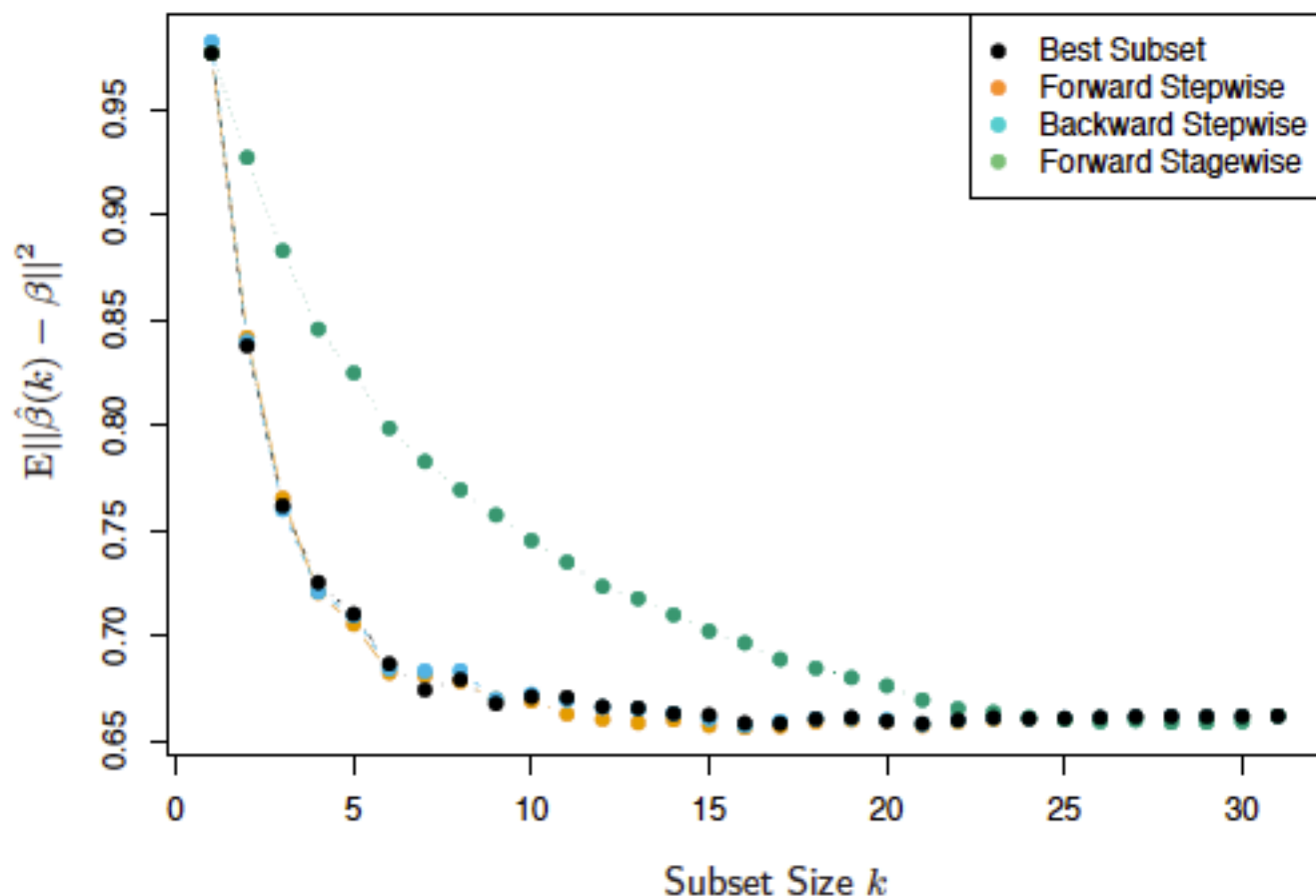
1. Classical variable selection
  - Backward, Forward, Stepwise methods,
  - All subsets
  - Best criteria: AIC, MAIC, BIC
2. Shrinkage Methods:
  - Ridge Regression
  - LASSO
  - ELASTIC NET/GLM NET

# Subset Selection

- Standard “all-subsets” finds the subset of size  $k$ ,  $k=1,\dots,p$ , that minimizes RSS:



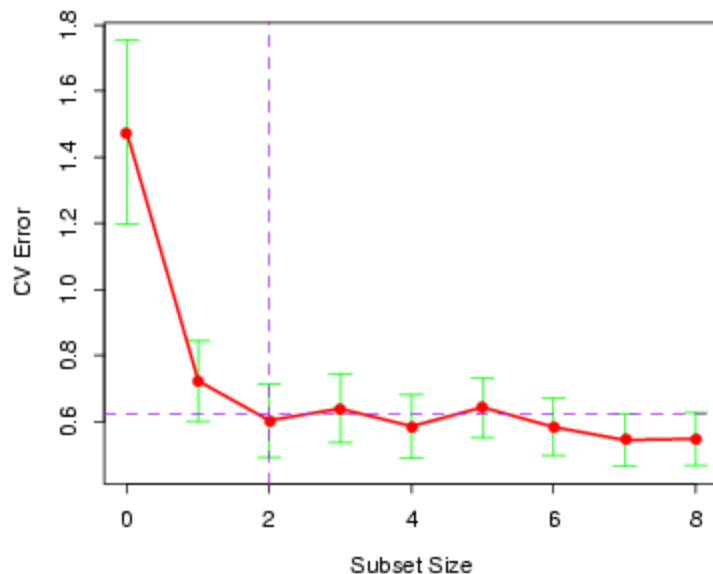
- In R function “leaps” will do it
- Choice of subset size requires tradeoff – AIC, BIC, marginal likelihood, cross-validation, etc.
- “Leaps and bounds” is an efficient algorithm to do all-subsets



**FIGURE 3.6.** Comparison of four subset-selection techniques on a simulated linear regression problem  $Y = X^T \beta + \varepsilon$ . There are  $N = 300$  observations on  $p = 31$  standard Gaussian variables, with pairwise correlations all equal to 0.85. For 10 of the variables, the coefficients are drawn at random from a  $N(0, 0.4)$  distribution; the rest are zero. The noise  $\varepsilon \sim N(0, 6.25)$ , resulting in a signal-to-noise ratio of 0.64. Results are averaged over 50 simulations. Shown is the mean-squared error of the estimated coefficient  $\hat{\beta}(k)$  at each step from the true  $\beta$ .

# 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

# Penalized methods (Shrinkage Methods)

- Stepwise or more generally Subset selection is a discrete process – individual variables are either in or out
- It can have high variability – a different dataset from the same source can result in a totally different model
- Overfitting is still a big problem with stepwise or subset selection
- 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

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

Equivalently:

$$\left( \sum_{i=1}^N (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right) = |Y - X\beta|^2 + \lambda |\beta|^2 =$$
$$Y'Y - 2Y'X\beta + \beta'(X'X + \lambda I)\beta$$

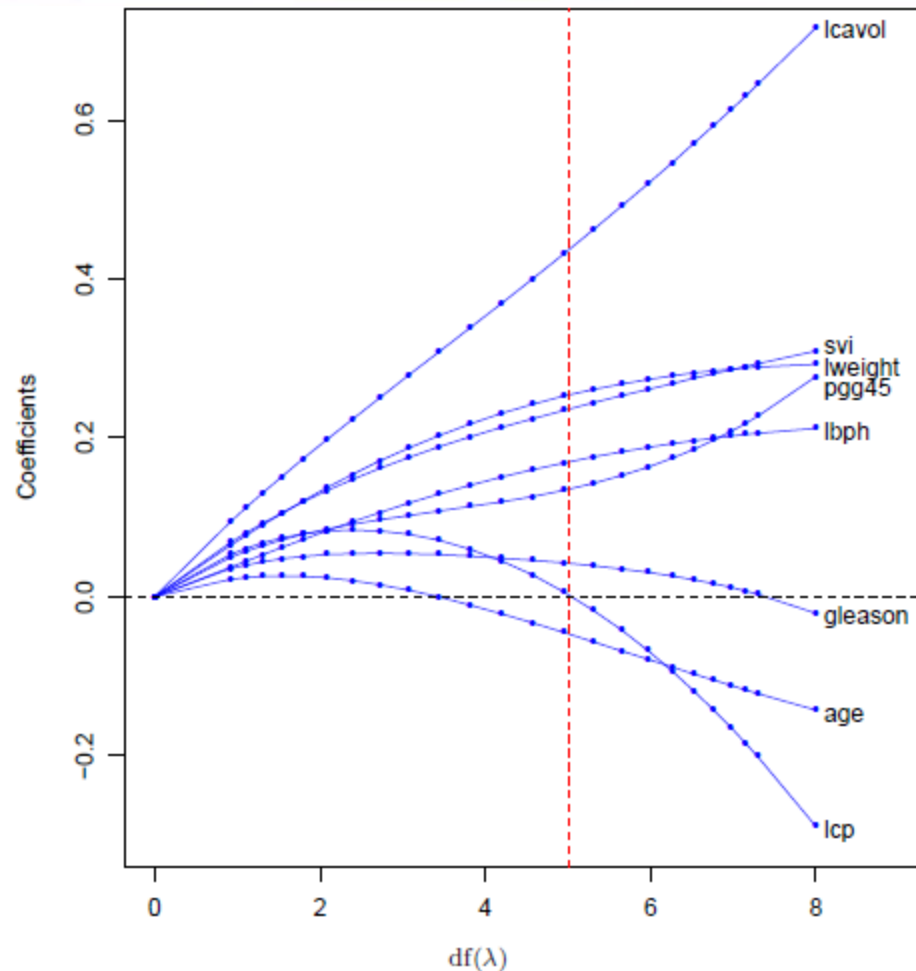
Taking deriv w.r.t.  $\beta$ :  $-2Y'X + 2(X'X + \lambda I)\beta = 0$

This leads to:  $\hat{\beta}^{\text{ridge}} = (X'X + \lambda I)^{-1} X'Y$

works even when  
 $X'X$  is singular

Choose  $\lambda$  by cross-validation. Predictors should be centered.

# Solution path for Ridge Regression



**FIGURE 3.8.** Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter  $\lambda$  is varied. Coefficients are plotted versus  $df(\lambda)$ , the effective degrees of freedom. A vertical line is drawn at  $df = 5.0$ , 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

Replace  $L_2$  penalty by  $L_1$  penalty  $\Rightarrow$  solution path eliminates variables

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

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

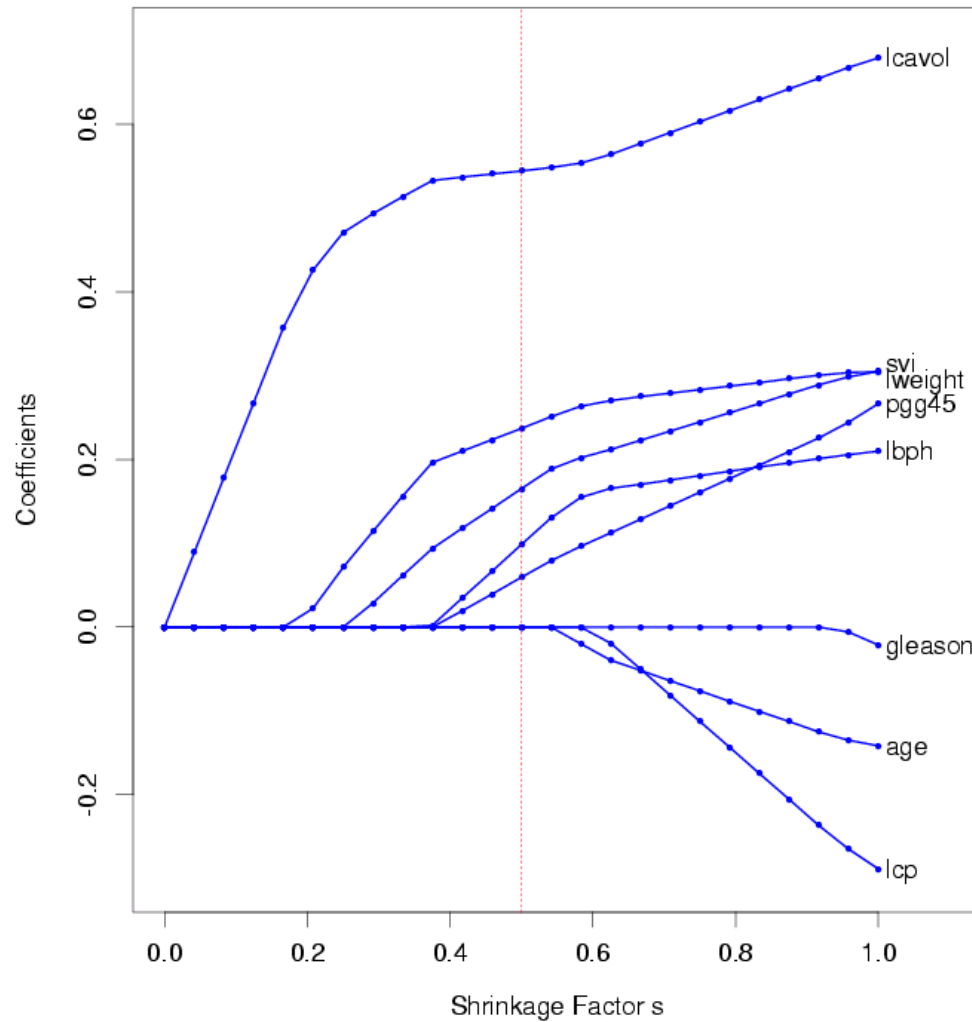
Quadratic programming algorithm needed to solve for the parameter estimates. Choose  $s$  via cross-validation.

---

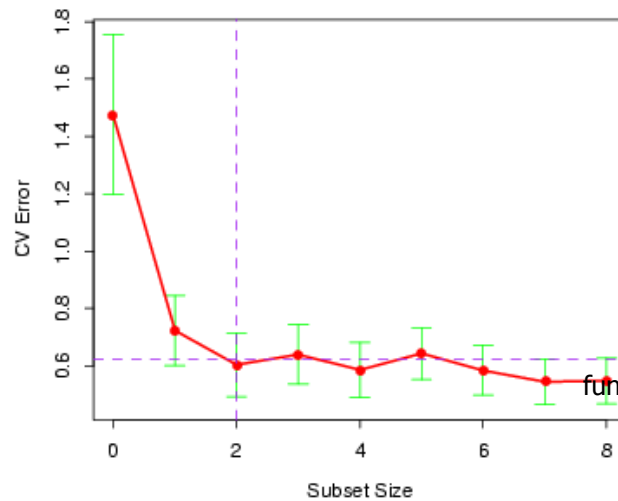
$$\tilde{\beta} = \arg \min_{\beta} \left( \sum_{i=1}^N (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j|^q \right)$$

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

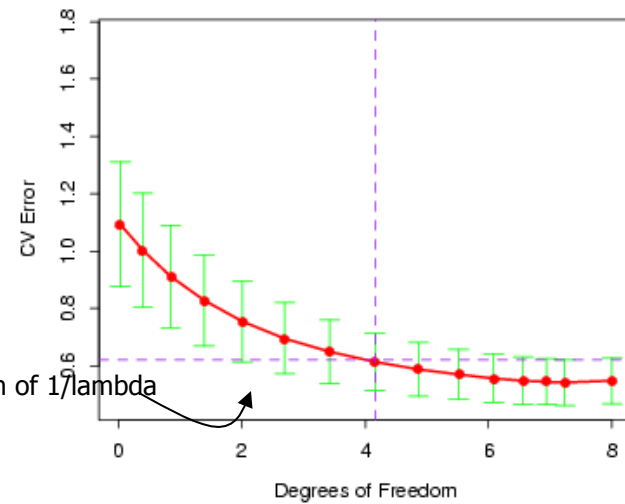
# Solution path for Lasso



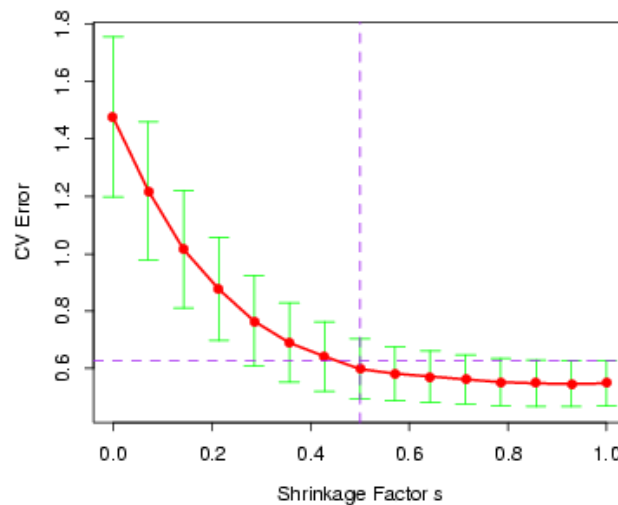
### All Subsets



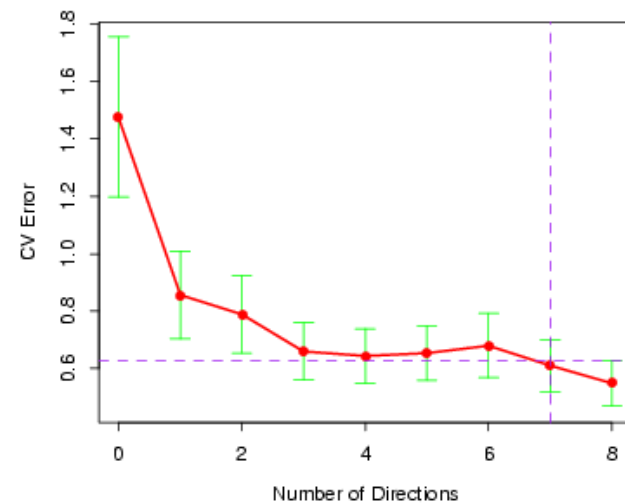
### Ridge Regression



### Lasso



### Principal Components Regression



# Elastic Net

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

FOR NORMAL MODEL:  $\frac{1}{2} \text{RSS}/N + \lambda \text{ PENALTY}$

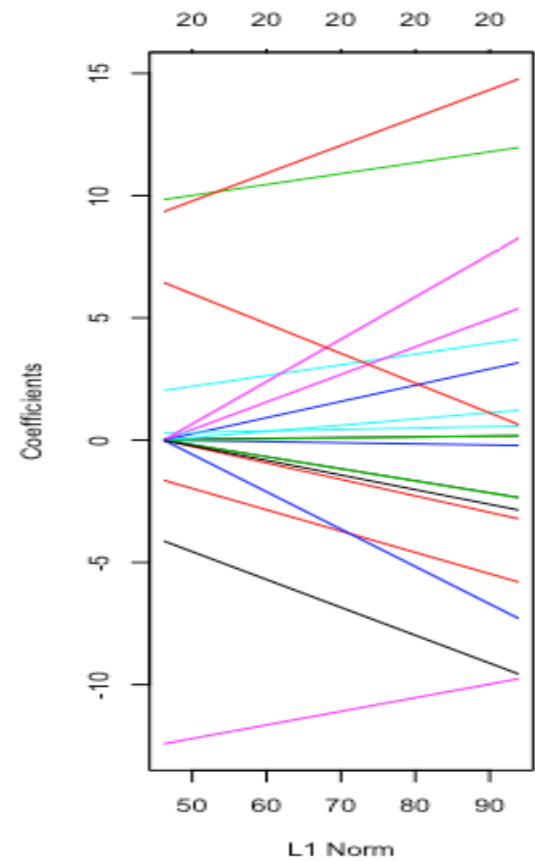
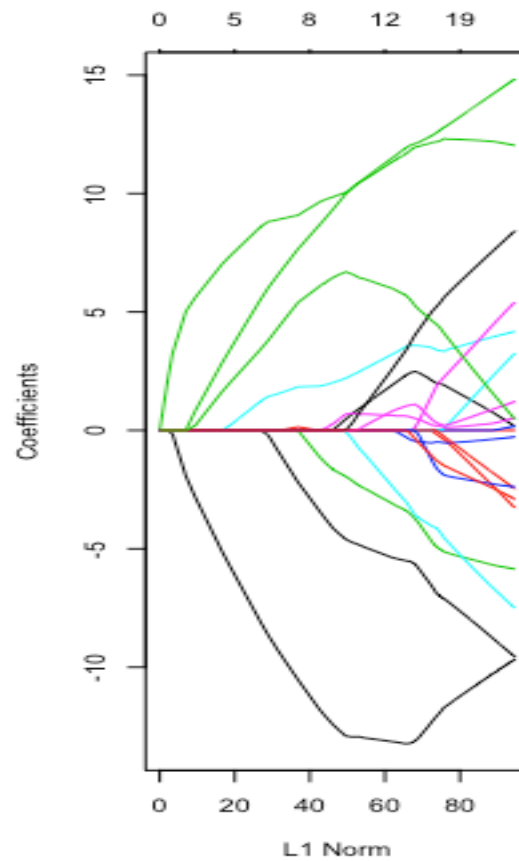
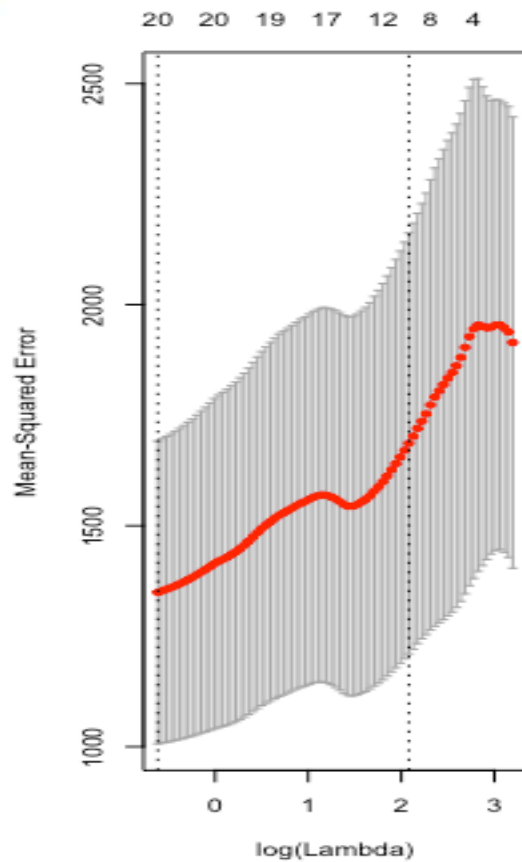
FOR OTHER MODELS:  $-\text{Loglik}/N + \lambda \text{ PENALTY}$

# R Code for Ridge( $\alpha=0$ ), Lasso( $\alpha=1$ ) and Enet ( $\alpha=\text{any}$ ) using GLM net

```
myglmnet = function(z, y, a=0.5) {  
  # Center and scale variables  
  z = as.matrix(t((t(z)-apply(z,2,mean))/apply(z,2,sd)))  
  # Find lambda by CV  
  plot(u<-cv.glmnet(z,y,alpha=a))  
  # Plot full solution path  
  plot(glmnet(z,y,alpha=a ))  
  lam=c(u$lambda.1se,u$lambda.min)  
  v <- glmnet(z,y,alpha=a,lambda=lam)  
  # Plot Lambda path  
  plot(v)  
  # Output lambda and estimates  
  list(lambda=lam,beta=v$beta)  
}  
library(glmnet) ; set.seed(2016)  
n=20; p=60; k=4  
x=matrix(rnorm(n*p),n,p)  
b= rt(k,20)*5 ; y=x[,1:k] %*% b+rnorm(n)*2  
par(mfrow=c(1,3))  
myglmnet(x,y,a=1)
```



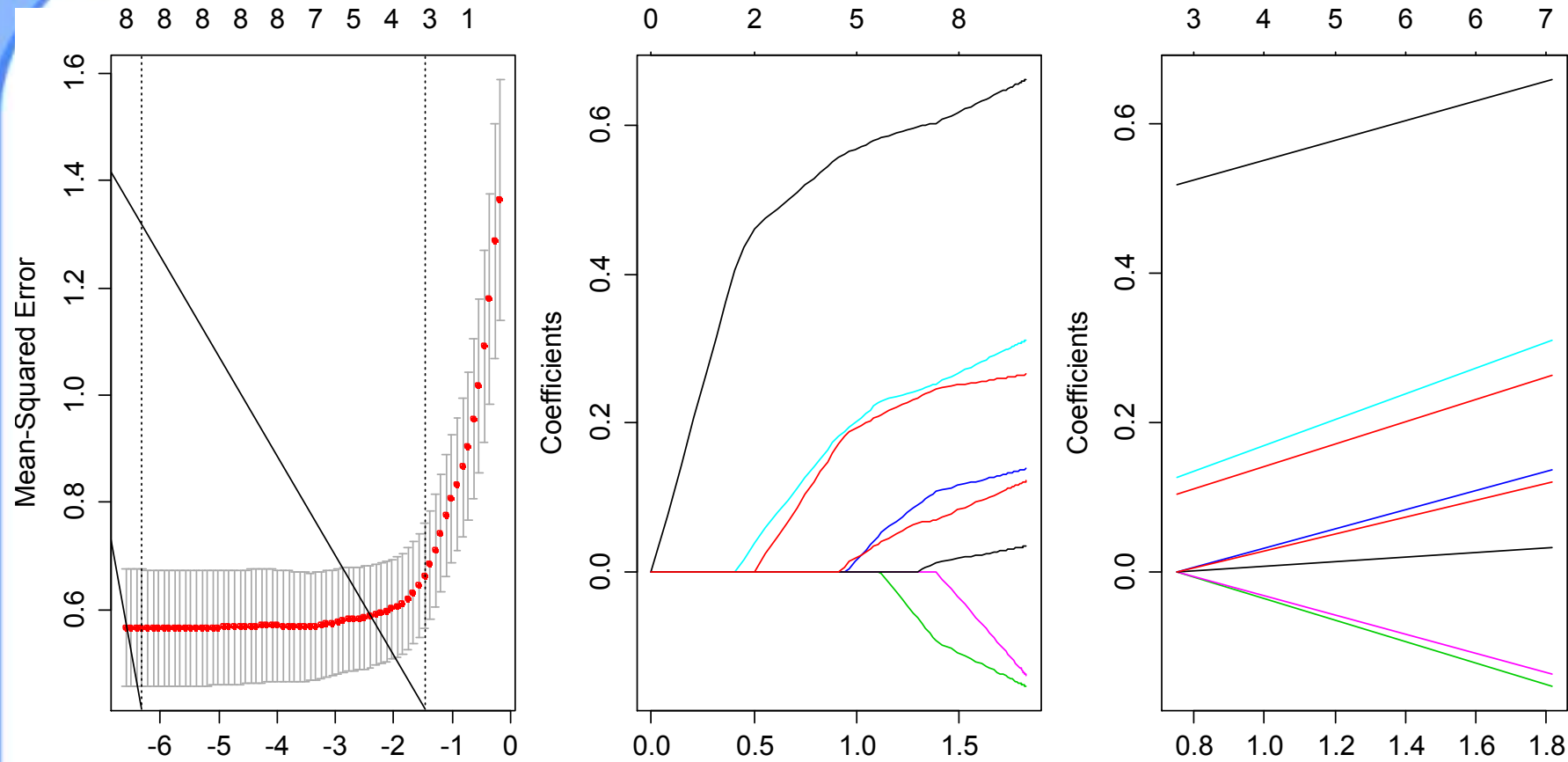
# Ridge Output from GLM net



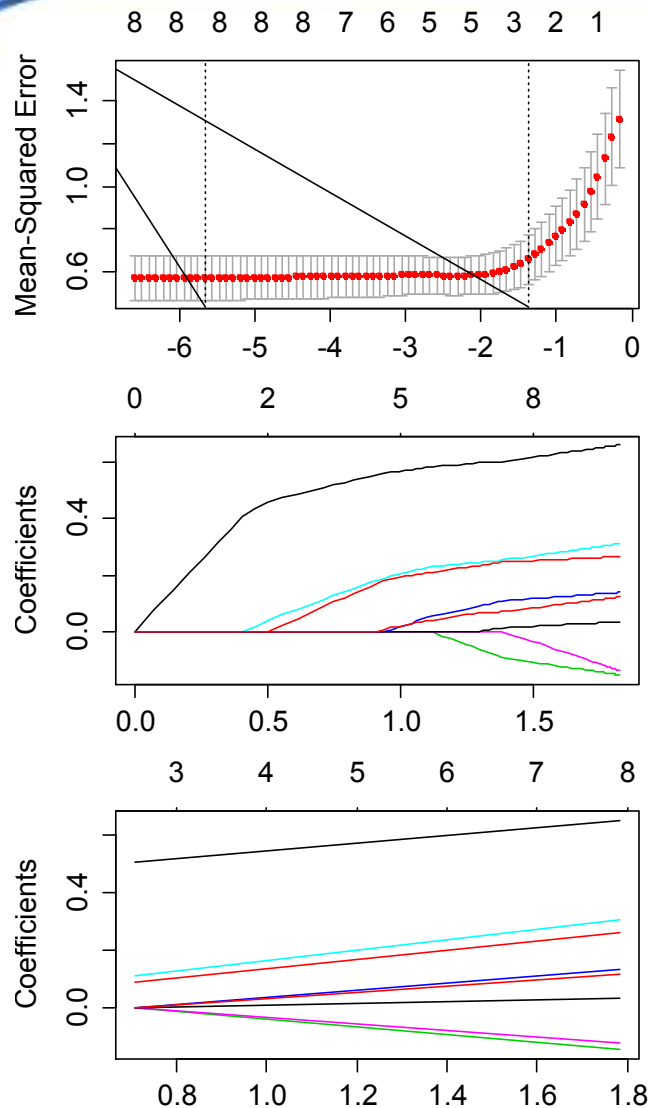
# Lasso Output from GLM net

```
#prost=url("http://www-stat.stanford.edu/~tibs/
ElemStatLearn/datasets/prostate.data")
> prost=read.table("prost.txt",head=T)
> myglmnet(prost[,1:8],prost[,9],a=0)
$lambda
[1] 0.86327415 0.08434274
$beta
8 x 2 sparse Matrix of class "dgCMatrix"
               s0               s1
lcavol    0.32398301    0.58020370
lweight    0.18612911    0.25875483
age        -0.02308884   -0.12450751
lbph        0.07491161    0.12456784
svi         0.19230986    0.28396396
lcp         0.10308122   -0.05559908
gleason     0.06022965    0.04594414
pgg45       0.07370693    0.09625296
```

# Lasso Output from GLM net



# Ridge Output from GLM net

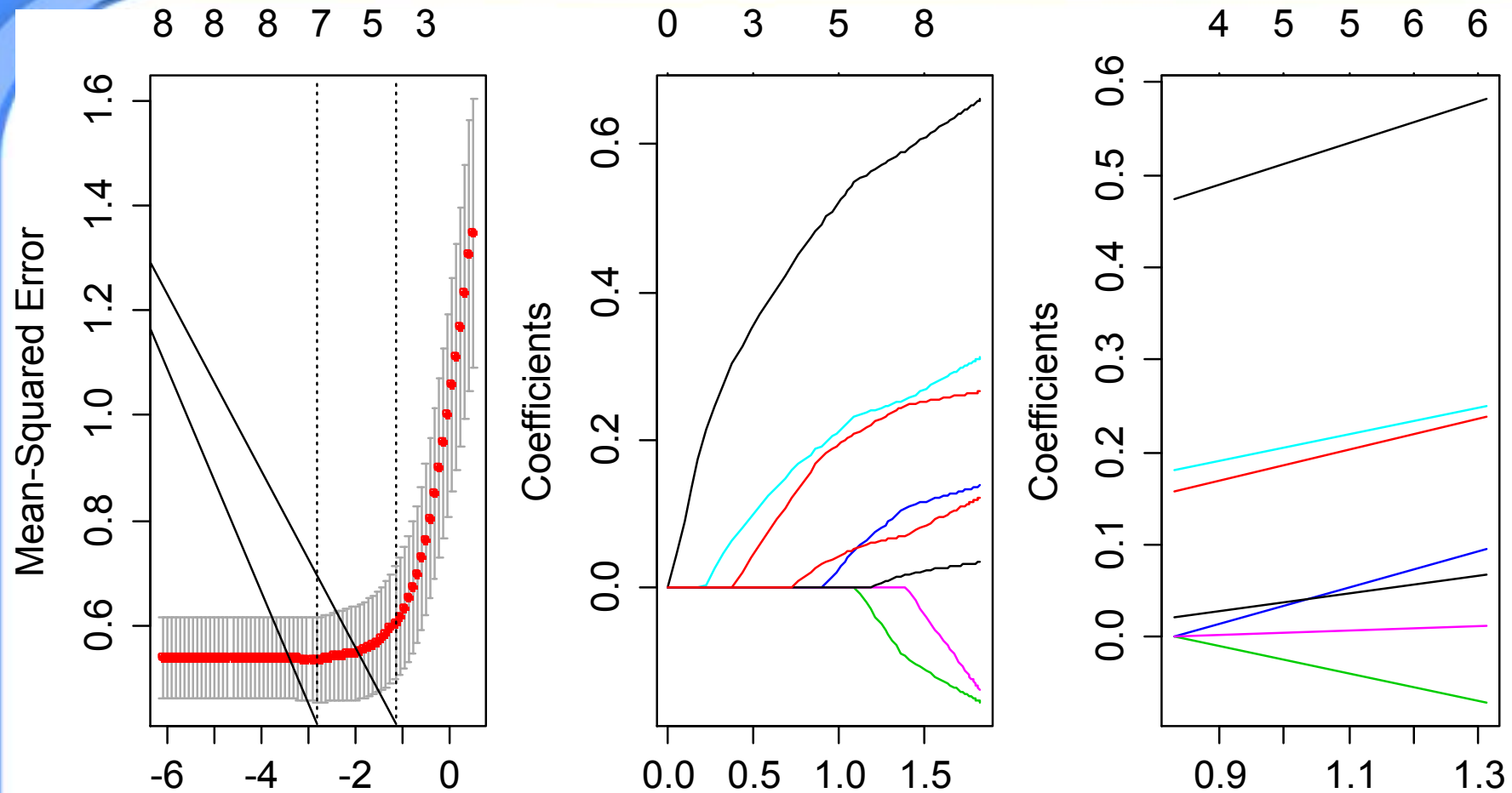


```
> myglmnet(prost[,1:8],prost[,9],a=1)
$lambda
[1] 0.22929319 0.02961435
```

```
$beta
8 x 2 sparse Matrix of class
"dgCMatrix"
```

	s0	s1
lcavol	0.5197469	0.599935919
lweight	0.1041996	0.239353146
age	.	-0.076377704
lbph	.	0.097817827
svi	0.1268638	0.248051507
lcp	.	.
gleason	.	0.006135243
pgg45	.	0.067144999

# ENET $\alpha=0.5$ Output from GLM net



# ENET $\alpha=0.5$ Output from GLM net

```
> myglmnet(prost[,1:8],prost[,9],a=0.5)
$lambda
[1] 0.31608581 0.05922871
$beta
8 x 2 sparse Matrix of class "dgCMatrix"

           s0           s1
lcavol  0.4733168  0.58093039
lweight 0.1566415  0.23748890
age      .         -0.07015625
lbph     .         0.09417850
svi      0.1808864  0.24915717
lcp      .         .
gleason  .         0.01241265
pgg45    0.0208983  0.06719576
```

# ENET Output from GLM net

## Iterate twice

```
myglmnet(prost[,c(1:5,7:8)],prost[,9],a=0.5)
$lambda
[1] 0.26241999 0.01939473
$beta
s0      s1
lcavol  0.49281146  0.60105347
lweight 0.17661952  0.25678221
age      .          -0.11896632
lbph     .           0.12304168
svi      0.19240422  0.26050966
gleason  .           0.02448453
pgg45    0.03130061  0.07446277
```

```
myglmnet(prost[,c(1:5,7:8)],prost[,9],a=0.5)
$lambda
[1] 0.38072647 0.02813844
$beta
s0      s1
lcavol  0.450558382  0.59661837
lweight 0.133438442  0.25247009
age      .          -0.10798982
lbph     .           0.11658586
svi      0.167118111  0.25803568
gleason  .           0.02175753
pgg45    0.008532863  0.07284914
```

# ENET $\alpha=0.5$ Output from GLM net

CV leave one out

```
myglmnet(prost[,c(1:5,7:8)],prost[,9],a=0.5)
```

```
$lambda
```

```
[1] 0.34690378 0.01610185
```

```
$beta
```

```
7 x 2 sparse Matrix of class "dgCMatrix"
```

	s0	s1
lcavol	0.46238264	0.60281372
lweight	0.14546563	0.25837993
age	.	-0.12309295
lbph	.	0.12552973
svi	0.17430576	0.26152330
gleason	.	0.02553766
pgg45	0.01497629	0.07497455



# Principal Component Regression

Consider an eigen-decomposition of  $X^T X$  (and hence the covariance matrix of  $X$ ):

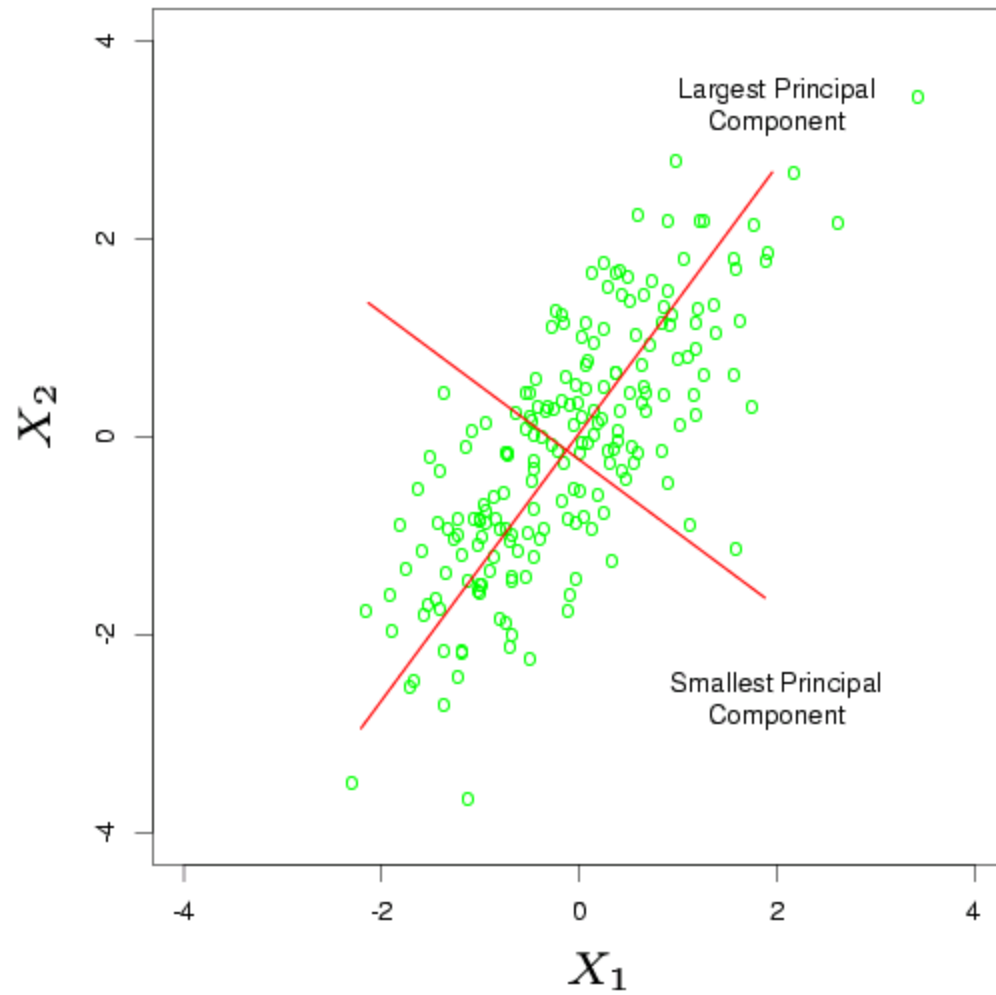
$$X^T X = V D^2 V^T$$

The eigenvectors  $v_j$  are called the *principal components* of  $X$   
 $D$  is diagonal with entries  $d_1 \geq d_2 \geq \dots \geq d_p$

$Xv_1$  has largest sample variance amongst all normalized linear combinations of the columns of  $X$

$$(\text{var}(Xv_1) = \frac{d_1^2}{N})$$

$Xv_k$  has largest sample variance amongst all normalized linear combinations of the columns of  $X$  subject to being orthogonal to all the earlier ones



# Principal Component Regression

PC Regression regresses on the first  $M$  principal components where  $M < p$

Similar to ridge regression in some respects –

# Partial Least Squares Regression

$Y$  Centered,  $X_i$  has  $\text{mean}(X_i)=0$ ,  $\text{Var}(X_i)=1$  for all  $i$ .

1.  $\hat{\varphi}_{1j} = \langle \mathbf{x}_j, \mathbf{y} \rangle$  : regressing  $\mathbf{y}$  on each  $\mathbf{x}_j$
2.  $\mathbf{z}_1 = \sum \hat{\varphi}_{1j} \mathbf{x}_j$
3.  $\hat{\theta}_1 = \langle \mathbf{z}_1, \mathbf{y} \rangle / \langle \mathbf{z}_1, \mathbf{z}_1 \rangle$  coefficient of regressing  $\mathbf{y}$  on  $\mathbf{z}_1$ ,
4. Update the  $\mathbf{x}_i$  's by orthogonalizing them w/r  $\mathbf{z}_1$ .
5. Update  $\mathbf{y}$  by the residuals of the previous linear fit.

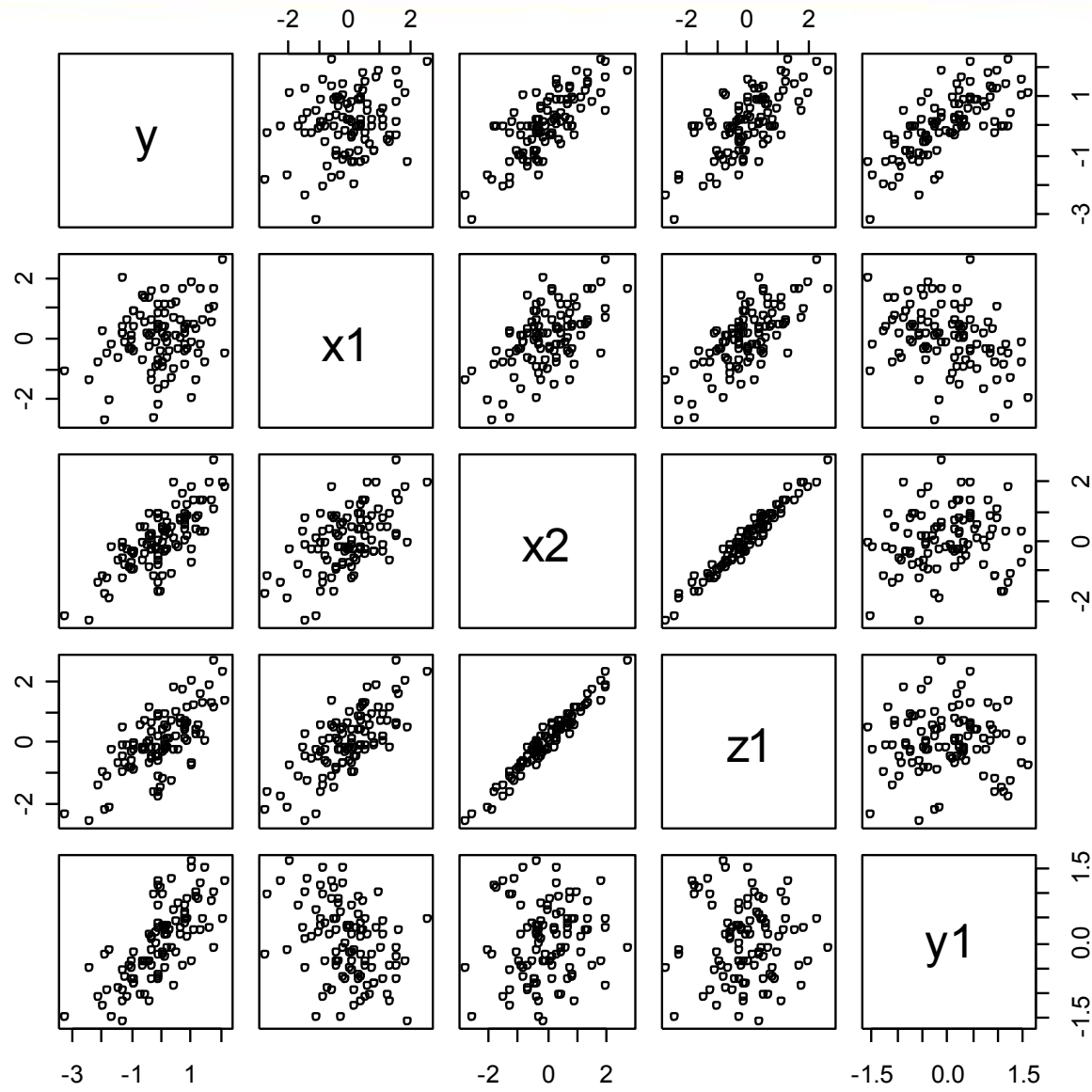
Iterate these 5 steps

This produces a sequence of orthogonal vectors  $\{\mathbf{z}_i\}$  and a sequence of estimators  $\hat{\beta}_j^{PLS}$

# Simple R program

```
# Generate some data
y = rnorm(100)
y = y - mean(y)
x1 = rnorm(100)
x1 = (x1 - mean(x1))/sd(x1)
x2 = y+x1+rnorm(100)
x2 = (x2 - mean(x2))/sd(x2)
#
pi1 = sum(y*x1)
pi2 = sum(y*x2)
z1 = pi1*x1 + pi2*x2
z1 = (z1 - mean(z1))/sd(z1)
th1 = lsfit(z1,y,int=F)$coef
y1 = y - th1*z1
pairs(cbind(y,x1,x2,z1,y1))
```

# Scatter Matrix of intermediate vars



# Simple R program (cont.)

```
# Now we do the second iteration.
x11 = x1 - sum(x1*z1)*z1/sum(z1*z1)
x21 = x2 - sum(x2*z1)*z1/sum(z1*z1)
phi1 = sum(y1*x1)
phi2 = sum(y1*x2)
z2 = phi1*x11 + phi2*x21
z2 = (z2 - mean(z2))/sd(z2)
th2 = lsfit(z2,y1,int=F)$coef
y2 = y1 - th2*z2
#another way to calculate z2:
z2 = (x11-mean(x11))/sd(x11)
pairs(cbind(y1,x11,x21,z1,z2))
```