Chapter 10: Shrinkage Methods

Gunwoong Park

Lecture Note

University of Seoul

Shrinkage Methods

- Principal components regression
- Partial least squares
- Ridge regression
- Lasso

Principal Components Analysis

Principal Components Analysis (PCA)

PCA:

- Special transformation on predictors
- Useful for high dimensional data
- Solve collinearity issue

Main uses:

- Reduce the dimensionality of the data
- Find linear combinations of predictors that explain the most variation in the data
- Facilitate visualization
- In regression, makes predictors orthogonal to each other

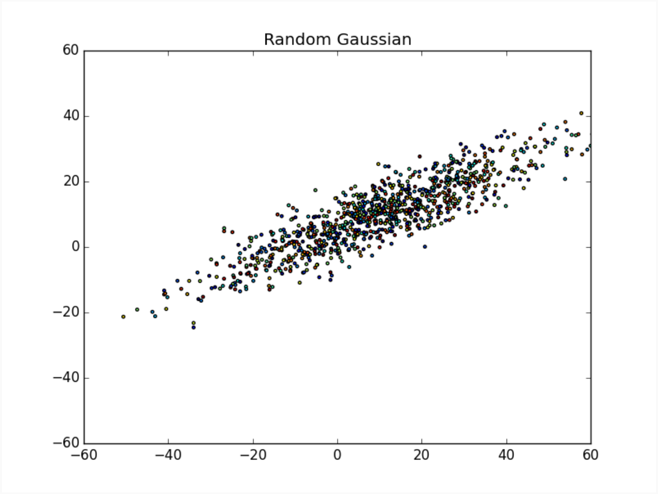
Definition of Principal Components

- Find the u_1 such that $\text{var}(u_1^T X)$ is maximized subject to $u_1^T u_1 = 1$.
- Find the u_2 such that $\text{var}(u_2^T X)$ is maximized subject to $u_2^T u_1 = 0$ and $u_2^T u_2 = 1$.
- Keep finding directions of greatest variation orthogonal to those directions we have already found.

Definitions:

- Vectors u_j are called the PC directions.
- Vectors $z_j = X u_j$ are called the principal components of X.

Principal Components Analysis (PCA)



Principal Components Analysis (PCA)

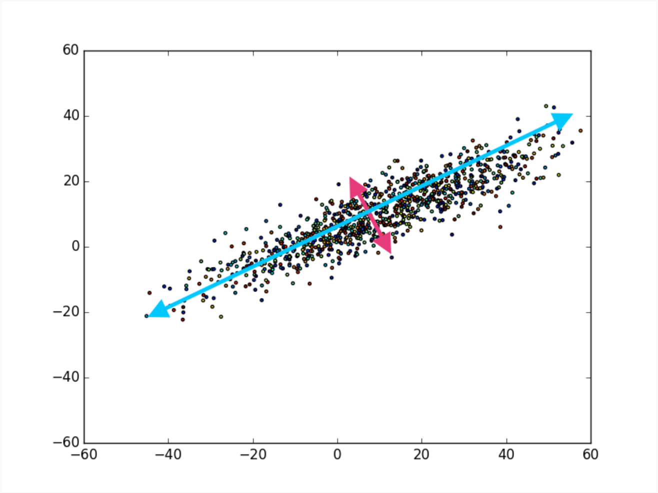


Figure 1: PCA: u_1 and u_2

Simulation Study: PCs

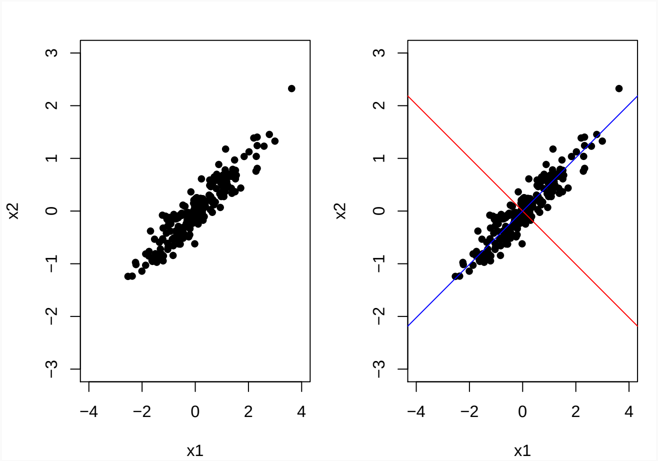
```
> prcomp(x)

Rotation:
     PC1    PC2
x1 -0.892 -0.451
x2 -0.451  0.892
```

$$Z1 = -0.892X_1 - 0.451X_2$$

$$Z2 = -0.451X_1 + 0.892X_2$$

Simulation Study: PCs

```
> plot(x, pch = 16, xlim = c(-4, 4), ylim = c(-3,3))
> abline(a = 0, b = 0.451/-0.892, col = "red")
> abline(a = 0, b = 0.451/0.892, col = "blue")
```



Simulation Study

```
> prX = prcomp(x)
> summary(prX)

Importance of components:
                         PC1    PC2
Standard deviation     1.210 0.1918
Proportion of Variance 0.976 0.0245
Cumulative Proportion  0.976 1.0000

> round(prX$rot[,1],3)
x1     x2
-0.892 -0.451
```

$$Z_1 = -0.892X_1 - 0.451X_2$$

Z_1 explains 97.6% of the both X_1 and X_2

Simulation Study: PCR

```
> lm0 = lm(Y ~ X1 + X2)
> summary(lm0)

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1164     0.1091  -1.067    0.289
x1            0.8597     0.2020   4.255 4.82e-05 ***
x2            1.2688     0.2219   5.717 1.20e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.086 on 97 degrees of freedom
Multiple R-squared:  0.7988,Adjusted R-squared:  0.7946
F-statistic: 192.5 on 2 and 97 DF,  p-value: < 2.2e-16
```

Simulation Study

```
> lm1 = lm(Y ~ Z)
> summary(lm1)

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1092     0.1094  -0.999     0.32
z            -1.4875     0.0762 -19.520   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.09 on 98 degrees of freedom
Multiple R-squared:  0.7954,Adjusted R-squared:  0.7933
F-statistic:   381 on 1 and 98 DF,  p-value: < 2.2e-16
```

Fat Example: Data

- Response: fat
- Predictors: neck, chest, abdom, hip, thigh, knee, ankle, biceps, forearm, wrist

```
> cfat = fat[,9:18]
> prfat = prcomp(cfat)
> dim(prfat$rot)
[1] 10 10
> dim(prfat$x)
[1] 252  10

> summary(prfat)

Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     15.990 4.0658 2.9660 2.0004 1.69408 1.49881 1.30322 1.25478 1.10955
Proportion of Variance  0.867 0.0561 0.0298 0.0136 0.00973 0.00762 0.00576 0.00534 0.00417
Cumulative Proportion   0.867 0.9230 0.9529 0.9664 0.97617 0.98378 0.98954 0.99488 0.99906
```

Fat Example

```
> round(prfat$rot[,1],2)
neck   chest  abdom    hip  thigh   knee
0.12    0.50   0.66   0.42   0.28   0.12

ankle  biceps forearm  wrist
0.06    0.15   0.07   0.04

> prfatc = prcomp(cfat, scale = TRUE)
> summary(prfatc)
Importance of components:
PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8     PC9   PC10
Standard deviation     2.650 0.8530 0.8191 0.7011 0.5471 0.5283 0.4520 0.4054 0.27827 0.25
Proportion of Variance 0.702 0.0728 0.0671 0.0492 0.0299 0.0279 0.0204 0.0164 0.00774 0.00
Cumulative Proportion  0.702 0.7749 0.8420 0.8911 0.9211 0.9490 0.9694 0.9859 0.99360 1.00
```

Fat Example: PCs

```
> round(prfatc$rot[,1],3)

 neck   chest   abdom     hip   thigh    knee   ankle
0.327   0.339   0.334   0.348   0.333   0.329   0.247

 biceps forearm  wrist
 0.322   0.270   0.299

 > round(prfatc$rot[,2],3)

 neck   chest   abdom     hip   thigh    knee   ankle
 -0.003  -0.273  -0.398  -0.255  -0.191   0.022   0.625

 biceps forearm  wrist
 0.022   0.363   0.377
```

Fat Example: OLS

```
> lmoda = lm(fat$brozek ~., data = cfat)
> summary(lmoda)
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.228749   6.214309   1.163  0.24588
neck        -0.581947   0.208580  -2.790  0.00569 **
chest       -0.090847   0.085430  -1.063  0.28866
abdom        0.960229   0.071582  13.414  < 2e-16 ***
hip         -0.391355   0.112686  -3.473  0.00061 ***
thigh        0.133708   0.124922   1.070  0.28554
knee        -0.094055   0.212394  -0.443  0.65828
ankle        0.004222   0.203175   0.021  0.98344
biceps       0.111196   0.159118   0.699  0.48533
forearm      0.344536   0.185511   1.857  0.06450 .
wrist       -1.353472   0.471410  -2.871  0.00445 **

Residual standard error: 4.071 on 241 degrees of freedom
Multiple R-squared:  0.7351,Adjusted R-squared:  0.7241
F-statistic: 66.87 on 10 and 241 DF,  p-value: < 2.2e-16
```

Fat Example: PCR

```
> lmodpcr = lm(fat$brozek ~ prfatc$x[,1:2])
> summary(lmodpcr)
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)         18.9385     0.3291  57.542   <2e-16 ***
prfatc$x[, 1:2]PC1   1.8420     0.1245  14.800   <2e-16 ***
prfatc$x[, 1:2]PC2  -3.5505     0.3866  -9.184   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.225 on 249 degrees of freedom
Multiple R-squared:  0.5492,Adjusted R-squared:  0.5456
F-statistic: 151.7 on 2 and 249 DF,  p-value: < 2.2e-16
```

Fat Example

```
> lmodpcr2 = lm(fat$brozek ~ prfatc$x[,1:10])
> summary(lmodpcr2)

Estimate Std. Error t value Pr(>|t|)
(Intercept)          18.93849    0.25647  73.843  < 2e-16 ***
prfatc$x[, 1:10]PC1   1.84198    0.09698  18.993  < 2e-16 ***
prfatc$x[, 1:10]PC2  -3.55053    0.30126 -11.785  < 2e-16 ***
prfatc$x[, 1:10]PC3   0.25669    0.31374   0.818 0.414067
prfatc$x[, 1:10]PC4   0.54094    0.36652   1.476 0.141273
prfatc$x[, 1:10]PC5   3.72632    0.46973   7.933 8.03e-14 ***
prfatc$x[, 1:10]PC6  -1.48784    0.48642  -3.059 0.002474 **
prfatc$x[, 1:10]PC7   1.94878    0.56859   3.427 0.000716 ***
prfatc$x[, 1:10]PC8  -0.12247    0.63390  -0.193 0.846967
prfatc$x[, 1:10]PC9  -1.71366    0.92351  -1.856 0.064731 .
prfatc$x[, 1:10]PC10 -9.01059    1.01566  -8.872  < 2e-16 ***

Residual standard error: 4.071 on 241 degrees of freedom
Multiple R-squared:  0.7351,Adjusted R-squared:  0.7241
F-statistic: 66.87 on 10 and 241 DF,  p-value: < 2.2e-16
```

Fat Example: Benefits of PCA

- Orthogonal Predictors
- No collinearity Issue
- Sometimes easy to Interpret

Choice of the number of variables

How to choose the optimal number of variables.

- Interpretability: It is important to examine the interpretability of the components and make sure that those providing a interpretable result are retained.
- Total variance
- Cross validation

- Typically most variation in X can be represented by a few principal components – **Dimension reduction**.
- Can choose k to explain certain **percent of variation:** pick first k so that

$$\sum_{i=1}^{k} \lambda_i \geq (1-\alpha) \sum_{i=1}^{p} \lambda_i$$

- Can look at the **scree plot** and look for a gap in eigenvalues
- More sophisticated methods for estimating **intrinsic dimension** of the data

Choice of the number of variables: RMSE

How to choose the optimal number of variables.

- Training and Test Set
- Root meas square of error (RMSE)

Training and Test Data Set

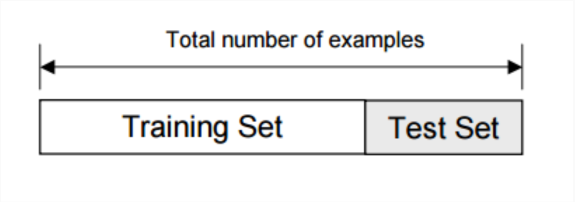
Motivation

- Model Selections and Shrinkage Methods provide good models
- Hard to choose only one optimal model.
- Choose an optimal model based on its performance.

Training and Test Data Set

Method

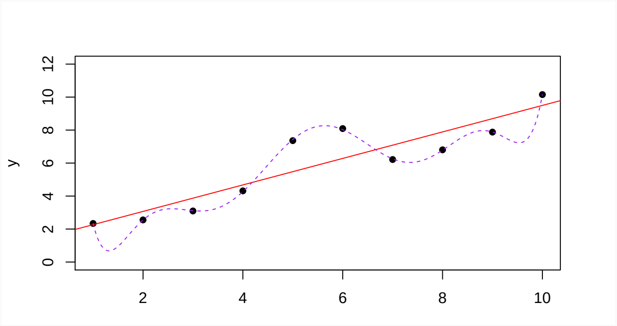
- Split dataset into two groups .
- Training Set: Used to train the model.
- Test Set: Used to measure the performance of the trained model.
- Validation Set: Used to train tuning parameters. (Extra).



Training and Test Data Set

Why Test Set?

- Overfitting Issue: It refers to a model that models the training data too well.



Training and Test Data Set

Why Test Set?

- Overfitting Issue
- Complex models tend to have a good performance in training data.
- A good model in training dataset may not be a good model in new dataset.

Training and Test Data Set

How to determine Test Data Set?

- Absolute Random selection: Choose 10 $\sim$ 20% of data set.
- Choose 10 $\sim$ 20% of data set with same proportion of success in both Training and Test data set. (To prevent the test set only contains success or same value of predictors.)

Training and Test Data Set

Weakness

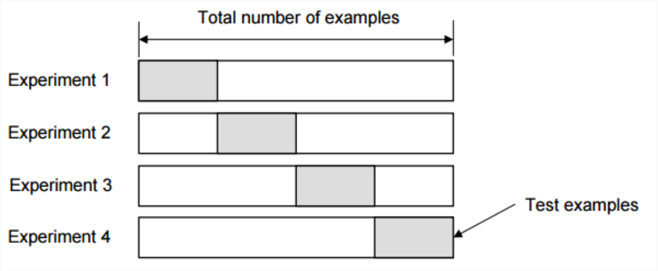
- Cannot be performed when sample size is small
- Cross-Validation

Cross-Validation

- Often there is insufficient data to create a separate validation set;
- In this instance, K-fold cross-validation is useful.

1. Divide the data into K disjoint subsets.
2. Use subsets $2, \ldots, K$ as training data and subset 1 as validation data. Compute the PE on subset 1.
3. Repeat for each subset.
4. Average the result.

Cross-Validation



Root meas square of error (RMSE)

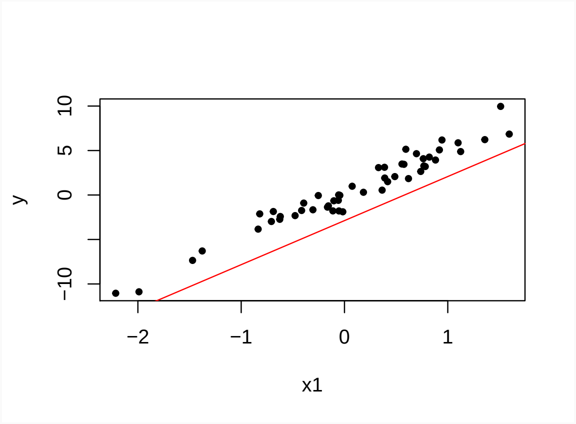
Definition:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i}^{n}(\hat{y}_i - y_i)^2}$$

Motivation:

- Bias: $y - \mathbb{E}(\hat{y})$
- Variance: $\text{Var}(\hat{y})$

Bias



Variance

$y_1 = 10$

- Small variance case: 95% confidence interval for y_1: $(9.99, 10.01)$
  $\rightarrow$ 95% sure that y_1 is in $(9.99, 10.01)$
- :Large variance case: 95% confidence interval for y_1: $(-5, 25)$
  $\rightarrow$ 95% sure that y_1 is in $(-5, 25)$

Root mean square of error (RMSE)

- Mean Square Error $= \text{Bias}(\hat{y})^2 + \text{Variance}(\hat{y})$
- RMSE is a good criterion to choose an optimal model

Simulation Study: Choice of the number of variables

How to choose the optimal number of variables.

- 4 predictors: X_1, X_2, X_3, X_4
- Training set: 40 observations
- Test set: 10 observations

Simulation Study

```
> ran = sample(1:50, replace = F)[1:40]
> train = data[ran,]
> test = data[setdiff(1:50,ran),]
> prx = prcomp(x[ran,])
> summary(prx)

Importance of components:
                          PC1    PC2    PC3     PC4
Standard deviation     1.3194 0.5817 0.4301 0.21264
Proportion of Variance 0.7538 0.1465 0.0801 0.01958
Cumulative Proportion  0.7538 0.9003 0.9804 1.00000
```

Simulation Study

```
> lmodpcr2 = lm(y ~ prx$x[,1:2], train)
> z1 = prx$rotation[,1] %*% t( test[,1:4] )
> z2 = prx$rotation[,2] %*% t( test[,1:4] )
> z = rbind(z1, z2)
> ypred = coef(lmodpcr2) %*% rbind(1, z)
> sqrt( mean( (test$y - ypred)^2) )
[1] 1.294111

> lmodpcr3 = lm(y ~ prx$x[,1:3], train)
> z1 = prx$rotation[,1] %*% t( test[,1:4] )
> z2 = prx$rotation[,2] %*% t( test[,1:4] )
> z3 = prx$rotation[,3] %*% t( test[,1:4] )
> z = rbind(z1, z2, z3)
> ypred = coef(lmodpcr3) %*% rbind(1, z)
> sqrt( mean( (test$y - ypred)^2) )
[1] 1.29453
```

Remarks on PCA

- Interpretation may be easy or difficult
- Sufficiently reduce the number of predictors
- Difficult to decide the number of predictors

Food Analyzer Example

- Response: fat content
- Predictors: 100 channel spectrum of absorbances
- Number of data points: $n = 215$
- Number of predictors: $p = 100$

Prediction Performance

: build a model that predicts well on future data.

- Divide the data into two groups: training data and test data.
- Build the models using the training data and evaluate them on the test data.

Food Analyzer Example Continued

```
> library(faraway)
> data(meatspec)
> dim(meatspec)
[1] 215 101
## Training and test data
> tr = meatspec[1:172,]
> te = meatspec[173:215,]

## Linear model
> g1 = lm(fat ~ ., tr)
> summary(g1)$r.squared
[1] 0.9970196

## Root mean squared error
> rmse = function(x, y) { sqrt(mean( (x - y)^2 ))}
> rmse(g1$fit, tr$fat)
[1] 0.6903167
> ## Prediction
> rmse( predict(g1, newdata=te), te$fat )
[1] 3.814000
```

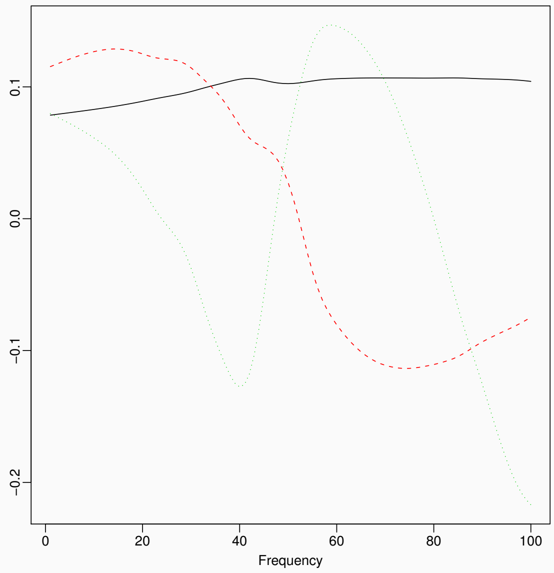
Food Analyzer Example Continued

```
## AIC
> g2 = step(g1)
> rmse( g2$fit, tr$fat )
[1] 0.7095069
> rmse( predict(g2, newdata=te), te$fat )
[1] 3.590245

## Principal components regression
> library(stats)
> meatpca = prcomp(tr[,-101])
## Square root of the eigenvalues
> round(meatpca$sdev, 3)
[1] 5.055 0.511 0.282 0.168 0.038 0.025 0.014
[8] 0.011 0.005 0.003 0.002 0.002 0.001 0.001
... ...
> matplot(1:100, meatpca$rot[,1:3], type="l",
xlab="Frequency", ylab="")
```

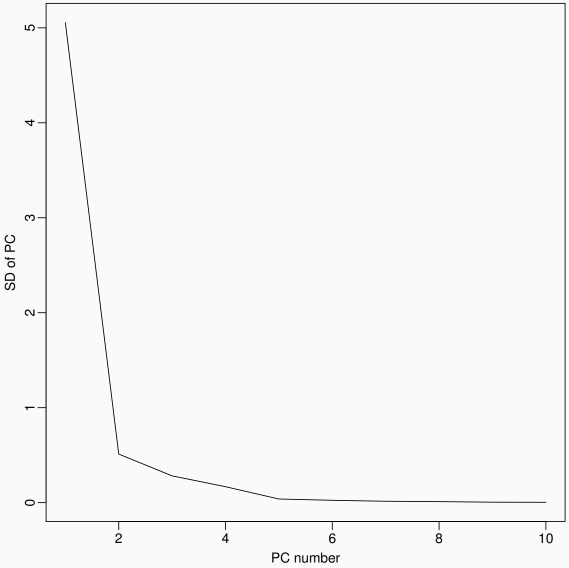
Food Analyzer Example Continued

```
## Make a scree plot (to choose number of PCs k)
> plot(1:10, meatpca$sdev[1:10], type="l",
xlab="PC number", ylab="SD of PC")
## Fit all PCRs at once and calculate test RMSE for each k
> library(pls)
> pcrg = pcr(fat ~ ., data=tr, ncomp=50)
> rmsmeat = NULL
>  for (k in 1:50) {
+    pv = predict(pcrg, newdata=te, ncomp=k)
+    rmsmeat[k] = rmse(pv, te$fat )  }
> plot(rmsmeat, xlab="PC number",   ylab="Test RMS")
# scree plot suggestion
> rmsmeat[5]
[1] 3.533628
> which.min(rmsmeat)
[1] 27
> rmsmeat[27]
[1] 1.854858
```

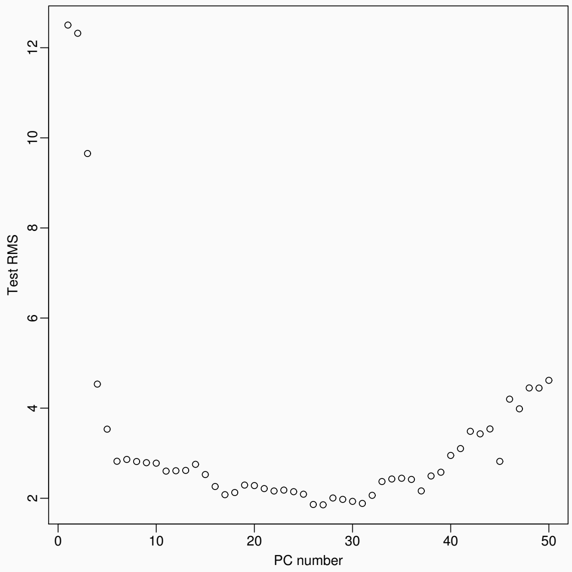
Food Analyzer: First 3 PCs



Food Analyzer: Scree plot



Food Analyzer: Test RMSE

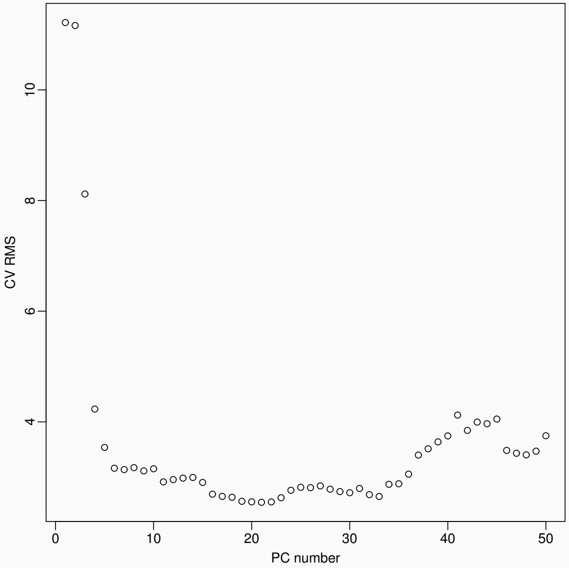


Food Analyzer Example: Cross-validation

```
> pcrg = pcr(fat ~ ., data=tr, ncomp=50,
> +          validation="CV",  segments = 10)
> rmsCV= RMSEP(pcrg, estimate='CV')
> which.min(rmsCV$val)
19
# Another try
> pcrg = pcr(fat ~ ., data=tr, ncomp=50,
> +          validation="CV", segments = 10)
> rmsCV= RMSEP(pcrg, estimate='CV')
> which.min(rmsCV$val)
21

## Plot the RMSE; k=0 is the model with intercept only
> plot(rmsCV$val, xlab="PC number", ylab="CV RMS")
## Get test error
> yfit = predict(pcrg, newdata=te, ncomp=21)
> rmse(te$fat, yfit)
[1] 2.214545
```

CV tends to underestimate the real test RMSE but often comes close.

Food Analyzer: Cross-validation



Partial Least Squares

Partial Least Squares

- PCR ignores y when building z's
- Partial least squares (PLS) chooses z's that are best at predicting y.
- PLS does not solve a well-defined modelling problem
- Many algorithms for PLS exist
- Also need to select number of components
- No interpretation

Partial Least Squares: Algorithm

Algorithm:

1. Center y, center and standardize each x_j
2. Regress y on each x_j *separately* to get α_j
3. Construct $z_1 = \sum \alpha_j x_j$, which is the first PLS component
4. Regress y on z_1 to get $\hat{\beta}_1$
5. Orthogonalize each x_j with respect to z_1
6. Continue until the final model is fit:

$$\hat{y} = \bar{y} + \hat{\beta}_1 z_1 + \cdots \hat{\beta}_k z_k$$

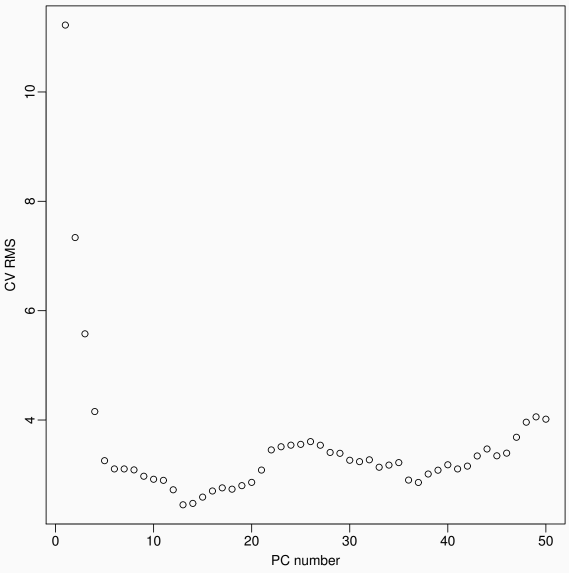
Remarks:

- Prediction purpose: PLS better
- Explanation purpose: PCR better

Food Analyzer Example

```
> ## Partial least squares
> plsg = plsr(fat ~ ., data=tr, ncomp=50, validation="CV")
> # plot RMSE estimated by CV
> pls_rmsCV = RMSEP(plsg, estimate='CV')
> plot(pls_rmsCV$val, xlab="PC number",ylab="CV RMS")
> which.min(pls_rmsCV$val)
[1] 14
> ## RMSE on the training data
> dim(plsg$fit)
[1] 172   1  50
> rmse(plsg$fit[,,14], tr$fat)
[1] 1.952796

> ## RMSE on the test data
> ypred.te = predict(plsg, newdata=te)
> dim(ypred.te)
[1] 43  1 50
> rmse(ypred.te[,,14], te$fat)
[1] 2.011180
```

Food Analyzer Example (PLS)



Ridge Regression

Ridge Regression

Penalizing the square of the coefficients

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

Assumption:

- Regression Coefficients should not be very large (after standardization).
- A large number of predictors should be considered.
- High collinearity exists.

Ridge Regression

Penalizing the square of the coefficients

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

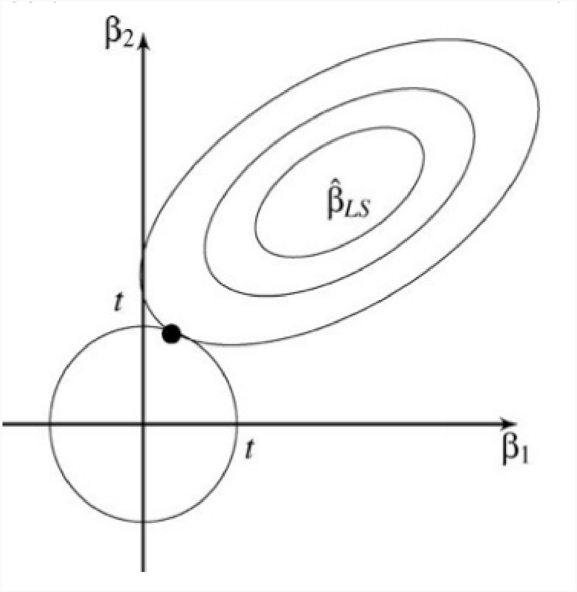
- The coefficients $\hat{\beta}^{\text{ridge}}$ are shrunken towards zero.
- $\lambda \geq 0$ is a tuning parameter.
- λ controls the amount of shrinkage.
- What happens if $\lambda \to 0$?
- What happens if $\lambda \to \infty$?

Equivalent Formulation

$$\begin{aligned} \min_{\beta} \quad & \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \\ \text{subject to} \quad & \sum_{j=1}^{p} \beta_j^2 \leq s \end{aligned}$$

- Explicitly constraint the size of the coefficients.

2-Dim Example



Ridge Regression

When there are many highly correlated variables

- $\hat{\beta}^{\text{ols}}$ may have a large coefficient on one variable and a similarly large negative coefficient on its correlated variable (Unstable).
- In ridge regression, the size constraint tries to avoid this phenomenon.
- Often standardize the predictors first.

Solution

- The solution is
$$\hat{\beta}^{\text{ridge}} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$
- $\hat{\beta}^{\text{ridge}}$ is still linear in $\boldsymbol{y}$.
- $\hat{\beta}^{\text{ridge}}$ is biased.

Woodbury matrix identity

Condition

- $\mathbf{A}$: p $\times$ p matrix, $\quad$ $\mathbf{U}$: p $\times$ n matrix,

  $\mathbf{C}$: n $\times$ n matrix, $\quad$ $\mathbf{V}$: n $\times$ p matrix

Result

- $(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1}$
- If $\mathbf{A} = \mathbf{I}_p, \quad \mathbf{C} = \mathbf{I}_n$, then

$$(\mathbf{I}_p + \mathbf{UV})^{-1} = \mathbf{I}_p - \mathbf{U}(\mathbf{I}_n + \mathbf{VU})^{-1}\mathbf{V}.$$

Woodbury matrix identity

$$
\begin{aligned}
&(\mathbf{A} + \mathbf{UCV})[\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}] \\
&= \left\{\mathbf{I} - \mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}\right\} \\
&+ \left\{\mathbf{UCVA}^{-1} - \mathbf{UCVA}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}\right\} \\
&= \left\{\mathbf{I} + \mathbf{UCVA}^{-1}\right\} - \mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1} \\
&- \mathbf{UCVA}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1} \\
&= \mathbf{I} + \mathbf{UCVA}^{-1} - (\mathbf{U} + \mathbf{UCVA}^{-1}\mathbf{U})\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1} \\
&= \mathbf{I} + \mathbf{UCVA}^{-1} - \mathbf{UC}\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1} \\
&= \mathbf{I} + \mathbf{UCVA}^{-1} - \mathbf{UCVA}^{-1} \\
&= \mathbf{I}
\end{aligned}
$$

Ridge Regression

Condition

- $\mathbf{X}$: n $\times$ p matrix

Goal

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

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

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

Ridge Regression

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

$$= (\lambda(\frac{1}{\sqrt{\lambda}}\mathbf{X}'\frac{1}{\sqrt{\lambda}}\mathbf{X} + \lambda \mathbf{I}_p))^{-1}\mathbf{X}'y$$

$$= \lambda^{-1}\left\{\mathbf{I}_p - \frac{1}{\lambda}\mathbf{X}'(\mathbf{I}_n + \frac{1}{\lambda}\mathbf{X}\mathbf{X}')^{-1}\mathbf{X}\right\}\mathbf{X}'y \quad \textbf{by Woodbury}$$

$$= \lambda^{-1}\left\{\mathbf{I}_p - \mathbf{X}'(\lambda \mathbf{I}_n + \mathbf{X}\mathbf{X}')^{-1}\mathbf{X}\right\}\mathbf{X}'y$$

$\Rightarrow$ n<p인 경우에 Ridge Regression에서 β를 추정할 때,

Woodbury를 쓰면 p×p 대신 n×n 행렬의 역행렬만 구해도 됨.

Comparison to LSE

$$\hat{\beta}^{\text{ridge}} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

- Even if $\boldsymbol{X}$ is not full-rank, $(\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})$ is invertible, thus solve exact collinearity issue.
- $\hat{\beta}^{\text{ridge}}$ has smaller variance than the OLS, thus may have smaller mean square error (MSE).

Shrinkage in Ridge: Special Case

Suppose orthonormal design ($\boldsymbol{X}^\intercal \boldsymbol{X} = \boldsymbol{I}$). Then $\hat{\boldsymbol{\beta}}^{\text{ols}} = \boldsymbol{X}^\intercal \boldsymbol{y}$, and

$$(\boldsymbol{y} - \boldsymbol{X}\beta)^\intercal(\boldsymbol{y} - \boldsymbol{X}\beta) = \text{constant} + \sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^{\text{ols}})^2.$$

Then ridge regression minimizes

$$\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^{\text{ols}})^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

Equivalent to the component-wise minimization

$$\min_{\beta_j}(\beta_j - \hat{\beta}_j^{\text{ols}})^2 + \lambda\beta_j^2 \Longrightarrow \hat{\beta}_j^{\text{ridge}} = \frac{1}{1+\lambda}\hat{\beta}_j^{\text{ols}}.$$

Shrinkage in Ridge

- Shrink the estimate towards zero by a positive constant less than 1
- $\mathrm{Var}(\hat{\beta}_j^{\mathrm{ridge}}) = \frac{1}{(1+\lambda)^2}\mathrm{Var}(\hat{\beta}_j^{\mathrm{ols}})$.
- $\lambda \uparrow$, shrinkage $\uparrow$, bias $\uparrow$, variance $\downarrow$
- $\lambda \downarrow$, shrinkage $\downarrow$, bias $\downarrow$, variance $\uparrow$.

Simulation Study: Almost Independent Predictors

3 predictors: X_1, X_2, X_3

```
> cor(data[,1:3])
       x1      x2     x3
x1  1.0000  0.0505 -0.142
x2  0.0505  1.0000 -0.143
x3 -0.1425 -0.1428  1.000

> lmod = lm(y ~ x1 + x2 + x3, data)
> lmod$coefficients
(Intercept)    x1          x2          x3
-0.724       1.023       1.299       1.817

> require(MASS)
> #lambda = 0
> lmrid = lm.ridge(y~x1 + x2 + x3, data, lmabda = 0)
> lmrid
(Intercept)    x1          x2          x3
-0.724       1.023       1.299       1.817
```

Simulation Study

```
> #lambda = 0.1
> lmrid2 = lm.ridge(y~x1 + x2 + x3, data, lambda = 0.1)
> lmrid2
(Intercept)    x1          x2          x3
-0.669        1.021       1.297      1.809
>
> #lambda = 10
> lmrid3 = lm.ridge(y~x1 + x2 + x3, data, lambda = 10)
> lmrid3
(Intercept)    x1          x2          x3
3.819        0.848        1.116      1.187
```

Simulation Study: Correlated Predictors

```
> lmod$coefficients
(Intercept)          x1          x2          x3
-0.676             1.132      -0.275       1.779

> cor(data[,1:3])
x1    x2    x3
x1 1.000 0.996 0.999
x2 0.996 1.000 0.998
x3 0.999 0.998 1.000
```

Simulation Study: Correlated Predictors

```
> require(MASS)
> #lambda = 0
> lmrid = lm.ridge(y~x1 + x2 + x3, data, lmabda = 0)
> lmrid
(Intercept)          x1           x2           x3
-0.676            1.132        -0.275      1.779
>
> #lambda = 0.1
> lmrid2 = lm.ridge(y~x1 + x2 + x3, data, lambda = 0.1)
> lmrid2
(Intercept)          x1           x2           x3
-0.720             1.405       0.327       1.195

> #lambda = 1
> lmrid3 = lm.ridge(y~x1 + x2 + x3, data, lambda = 1)
> lmrid3
(Intercept)          x1           x2           x3
-0.166             1.260       0.968       0.848
>
> #lambda = 10
> lmrid3 = lm.ridge(y~x1 + x2 + x3, data, lambda = 10)
> lmrid3
(Intercept)          x1           x2           x3
4.85               1.12        1.07         0.74
```

Simulation Study: Comparison to LSE

```
# Generate Training/Test sets
> ran = sample(1:50, replace = F)[1:40]
> train = data[ran,]
> test = data[setdiff(1:50,ran),]

> lmod = lm(y ~ x1 + x2 + x3, train)
> lmrid = lm.ridge(y~x1 + x2 + x3, train, lambda = 0.1)
> lmrid2 = lm.ridge(y~x1 + x2 + x3, train, lambda = 1)

# RMSE
> sqrt(mean((test$y - predict(lmod,test))^2))
[1] 5.4921
>
> ypred = cbind(1, as.matrix(test[,-4])) %*% coef(lmrid)
> sqrt(mean((test$y - ypred)^2))
[1] 5.1544
>
> ypred = cbind(1, as.matrix(test[,-4])) %*% coef(lmrid2)
> sqrt(mean((test$y - ypred)^2))
[1] 5.5591
```

Simulation Study: Normalization

```
> data.scale = scale(data[,1:3])
> data.scale = data.frame(data.scale, y = data$y)
>
> train = data.scale[ran,]
> test = data.scale[setdiff(1:50,ran),]
>
> lmod = lm(y ~ x1 + x2 + x3, train)
> lmrid = lm.ridge(y~x1 + x2 + x3, train, lambda = 0.1)
> lmrid2 = lm.ridge(y~x1 + x2 + x3, train, lambda = 1)

> sqrt(mean((test$y - predict(lmod,test))^2))
[1] 4.8518
>
> ypred = cbind(1, as.matrix(test[,-4])) %*% coef(lmrid)
> sqrt(mean((test$y - ypred)^2))
[1] 4.8484
>
> ypred = cbind(1, as.matrix(test[,-4])) %*% coef(lmrid2)
> sqrt(mean((test$y - ypred)^2))
[1] 4.8587
```

Simulation Study: Another Training/Test Set

```
> ran = sample(1:50, replace = F)[1:40]
> train = data.scale[ran,]
> test = data.scale[setdiff(1:50,ran),]
>
> lmrid = lm.ridge(y~x1 + x2 + x3, train, lambda = seq(0,1, len = 100))
> which.min(lmrid$GCV)
0.080808

> lmrid_GCV = lm.ridge(y~x1 + x2 + x3, train, lambda = 0.0808)
>
> ypred = cbind(1, as.matrix(test[,-4])) %*% coef(lmrid_GCV)
> sqrt(mean((test$y - ypred)^2))
[1] 4.486
>
> lmod = lm(y ~ x1 + x2 + x3, train)
> sqrt(mean((test$y - predict(lmod,test))^2))
[1] 4.7677
```

Simulation Study: Comparison to PCA

```
> X = train[,1:3]
> prx = prcomp(X)
> summary(prx)
Importance of components:
                         PC1     PC2     PC3
Standard deviation     1.755 0.06021 0.03225
Proportion of Variance 0.998 0.00117 0.00034
Cumulative Proportion  0.998 0.99966 1.00000

> z = 0.57778 * X[,1] + 0.57848 * X[,2] + 0.57579 *X[,3]
> lmodpcr = lm(train$y ~ z)
> ypred = cbind(1, as.matrix(0.57778 * test[,1] + 0.57848 * test[,2] + 0.57579 *test[,3]))
> sqrt(mean((test$y - ypred)^2))
[1] 4.1143
```

LASSO

Least absolute shrinkage and selection operator (Chen, Donoho and Saunders 1996; Tibshirani 1996)

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

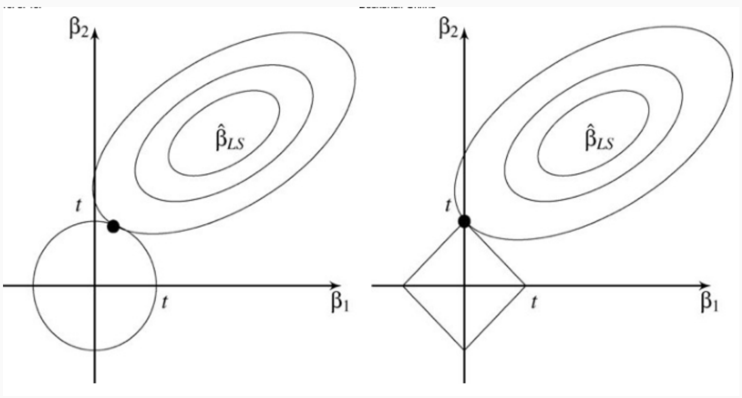
- Shrinkage
- Sparsity: some fitted coefficients are exactly zero

Continuous variable selection

Equivalent Formulation

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \\ \text{subject to} \quad & \sum_{j=1}^{p}|\beta_j| \le s \end{aligned}$$

Ridge vs Lasso



Soft Thresholding

When $\boldsymbol{X}$ is orthonormal, we can minimize over $\boldsymbol{\beta}$ componentwise

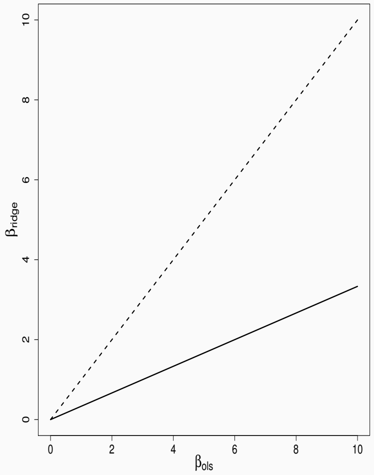
$$\hat{\beta}_j^{\text{lasso}} = \arg\min_{\beta_j} \; (\beta_j - \hat{\beta}_j^{\text{ols}})^2 + \lambda|\beta_j|.$$

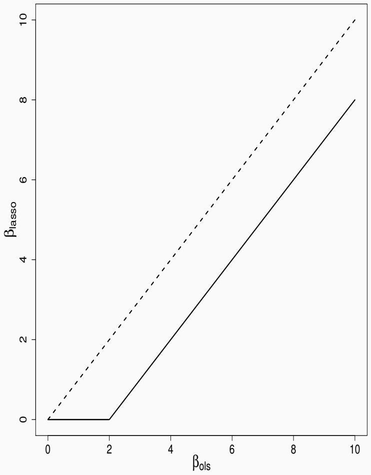
The solution is

$$\begin{aligned}\hat{\beta}_j^{\text{lasso}} &= \begin{cases} \hat{\beta}_j^{\text{ols}} - \frac{\lambda}{2} & \text{if } \hat{\beta}_j^{\text{ols}} > \frac{\lambda}{2} \\ 0 & \text{if } |\hat{\beta}_j^{\text{ols}}| \le \frac{\lambda}{2} \\ \hat{\beta}_j^{\text{ols}} + \frac{\lambda}{2} & \text{if } \hat{\beta}_j^{\text{ols}} < -\frac{\lambda}{2} \end{cases} \\ &= \operatorname{sign}(\hat{\beta}_j^{\text{ols}}) \cdot \left(|\hat{\beta}_j^{\text{ols}}| - \frac{\lambda}{2}\right)_+ \end{aligned}$$

- Lasso shrinks large coefficients by a constant.
- Lasso truncates small coefficients to zero.

Ridge vs Lasso



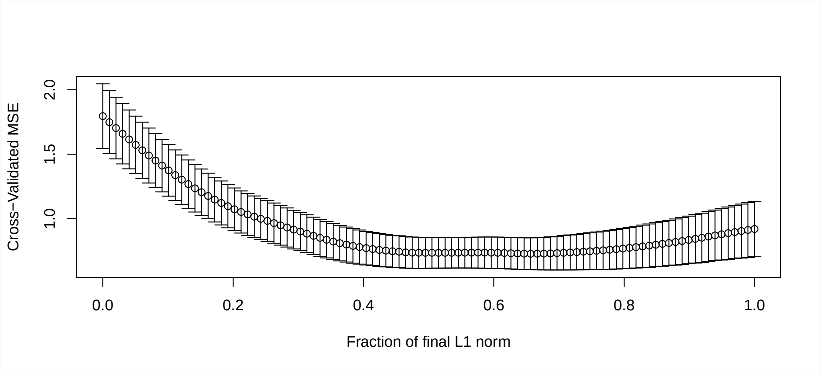


Example: Life Expectancy

```
> plot(lmod)
> require(lars)
> data(state)
> statedata = data.frame(state.x77, row.names = state.abb)
> colnames(statedata)
[1] "Population" "Income"     "Illiteracy" "Life.Exp"   "Murder"     "HS.Grad"
[7] "Frost"      "Area"

> lmod = lars(as.matrix(statedata[,-4]), statedata$Life)
> coef(lmod)
Population    Income Illiteracy Murder HS.Grad    Frost      Area
[1,]   0.00e+00  0.00e+00     0.0000  0.000  0.0000  0.00000  0.00e+00
[2,]   0.00e+00  0.00e+00     0.0000 -0.141  0.0000  0.00000  0.00e+00
[3,]   0.00e+00  0.00e+00     0.0000 -0.203  0.0282  0.00000  0.00e+00
[4,]   1.28e-05  0.00e+00     0.0000 -0.216  0.0308  0.00000  0.00e+00
[5,]   4.90e-05  0.00e+00     0.0000 -0.298  0.0461 -0.00576  0.00e+00
[6,]   4.90e-05 -5.22e-08     0.0000 -0.298  0.0461 -0.00576  0.00e+00
[7,]   4.97e-05 -8.19e-06     0.0000 -0.298  0.0467 -0.00581 -7.79e-09
[8,]   5.18e-05 -2.18e-05     0.0338 -0.301  0.0489 -0.00574 -7.38e-08
```

Example: Life Expectancy

```
> cvlmod = cv.lars(as.matrix(statedata[,-4]), statedata$Life.Exp)
> which.min( cvlmod$cv )
[1] 66
> cvlmod$index[66]
[1] 0.657
```



Example: Life Expectancy

```
> predict(lmod, s=0.657, type="coef", mode="fraction")$coef

Population     Income  Illiteracy     Murder
2.35e-05     0.00e+00   0.00e+00    -2.40e-01

HS.Grad      Frost       Area
3.53e-02  -1.70e-03   0.00e+00

> g = lm(Life.Exp ~ Population + Murder + HS.Grad +Frost, statedata)
> coef(g)
(Intercept)  Population      Murder     HS.Grad       Frost
7.10e+01    5.01e-05   -3.00e-01    4.66e-02   -5.94e-03
```

Example: Life Expectancy

```
# Ridge
> require(MASS)
> g = lm.ridge(Life.Exp ~., statedata, lambda = seq(0, 4, len = 50))
> which.min(g$GCV)
2.7755
> g = lm.ridge(Life.Exp ~., statedata, lambda = 2.7755)
> g

             Population    Income     Illiteracy     Murder
7.08e+01     4.13e-05     2.32e-05    -7.89e-02    -2.64e-01

 HS.Grad       Frost        Area
 4.60e-02    -5.15e-03    -3.89e-07
```

Example: Life Expectancy

```
# AIC
> g = lm(Life.Exp ~., statedata)
> step(g, direction = "backward", k = 2)

Step:  AIC=-28.2
Life.Exp ~ Population + Murder + HS.Grad + Frost

Df Sum of Sq  RSS   AIC
<none>                     23.3 -28.2
- Population  1       2.1 25.4 -25.9
- Frost       1       3.1 26.4 -23.9
- HS.Grad     1       5.1 28.4 -20.2
- Murder      1      34.8 58.1  15.5
```

Example: Life Expectancy

```
Call:
lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
data = statedata)

Coefficients:
(Intercept)   Population       Murder      HS.Grad        Frost
7.10e+01       5.01e-05      -3.00e-01       4.66e-02      -5.94e-03
```

Lasso

- Useful for high-dimensional data
- Still works when $p >> n$
- Theoretically guarantees

Simulation Study: High-Dimensional Data

- Response: Y
- Predictors: $X_1, X_2, ..., X_{40}$
- 30 samples

```
> dim(data)
[1] 30 41
> g = lm(Y ~., data)
```

Simulation Study: High-Dimensional Data

```
> coef(g)
(Intercept)  X1          X2          X3          X4          X5
-0.3313      1.8273     -1.3044      8.6732     -2.8432     -1.5281
X6          X7          X8          X9          X10         X11
-3.2323     -3.0793      3.0940     -0.4947     -1.5609     -1.0737
X12         X13         X14         X15         X16         X17
0.0377      3.1824     -3.0936     -0.5792     -0.2540      3.0077
X18         X19         X20         X21         X22         X23
4.3260     -1.1224      2.3879      1.7569      2.8271     -0.5614
X24         X25         X26         X27         X28         X29
2.6789      5.5562     -0.3190     -1.1525     -2.5788      1.4921
X30         X31         X32         X33         X34         X35
NA          NA          NA          NA          NA          NA
X36         X37         X38         X39         X40
NA          NA          NA          NA          NA
```

Simulation Study: High-Dimensional Data

```
> cvlmod = cv.lars(as.matrix(data[,-1]), data$Y)
> which.min( cvlmod$cv )
[1] 24
> cvlmod$index[24]
[1] 0.232
> predict(lmod, s = 0.232, type = "coef", mode = "fraction")$coef
X1      X2      X3      X4      X5      X6      X7      X8      X9
0.6951  0.0212  0.0000  0.0000  0.0000  0.0000  0.0000  0.0276  0.0038
X10     X11     X12     X13     X14     X15     X16     X17     X18
0.0000 -0.0953  0.0000  0.0000  0.0523  0.0000  0.0000  0.0000  0.0000
X19     X20     X21     X22     X23     X24     X25     X26     X27
0.0000  0.0000  0.0000 -0.2470  0.0000  0.0000  0.0000  0.0000 -0.2140
X28     X29     X30     X31     X32     X33     X34     X35     X36
-0.3813  0.0000  0.0000  0.0000  0.1900  0.0000  0.0000  0.0000  0.0000
X37     X38     X39     X40
0.0000  0.0086  0.2222 -0.2335
```

Simulation Study: High-Dimensional Data

```
> g = lm.ridge(Y ~., data, lambda = 1)
> g
X1        X2        X3        X4        X5        X6
1.06e-01  5.55e-01  2.84e-01  4.34e-02  2.17e-02  2.49e-02  2.53e-01
X7        X8        X9       X10       X11       X12       X13
4.12e-02  1.82e-01  2.16e-01  3.13e-02 -2.00e-01 -2.18e-01  2.80e-01
X14       X15       X16       X17       X18       X19       X20
2.90e-01  6.60e-02 -2.50e-01  1.73e-01 -1.66e-01  5.52e-02  2.26e-01
X21       X22       X23       X24       X25       X26       X27
-7.37e-02 -8.42e-02 -9.07e-05  5.40e-02  2.26e-01  1.05e-01 -2.78e-01
X28       X29       X30       X31       X32       X33       X34
-5.33e-01 -1.74e-01  1.48e-02 -2.68e-02 -5.19e-02  3.57e-01 -7.81e-02
X35       X36       X37       X38       X39       X40
-1.66e-01 -4.16e-02 -9.16e-02  4.18e-01  2.67e-01 -6.45e-01
```

Summary

- Main reason to use shrinkage: too many predictors or collinearity
- Interpretation is usually lost
- Ridge and lasso give linear models in the original predictors but no inference
- Prediction is usually improved by shrinkage
- All require selecting a tuning parameter (number of components for PCR and PLS, λ for ridge and Lasso) – need validation data or cross-validation.