Statistical Learning

https://github.com/ggorr/Machine-Learning/tree/master/ISLR

6. Linear Model Selection and Regularization

- 6.1 Subset Selection
- 6.2 Shrinkage Methods
- 6.3 Dimension Reduction Methods
- 6.4 Considerations in High Dimensions
- 6.5 Lab 1: Subset Selection Methods
- 6.6 Lab 2: Ridge Regression and the Lasso
- 6.7 Lab 3: PCR and PLS Regression
- 6.8 Exercises

Linear model with least square estimate

- Prediction accuracy
  - $n \gg p$
    - low variance and performing well on test observations
  - n is not much larger than p
    - overfitting and poor prediction
  - $n < p$
    - coefficients are not unique
    - variance is infinite
  - shrinking p is needed

- Model Interpretability
  - removing irrelevant variables is needs
  - feature selection (or variable selection)
- Solution
  - Subset Selection
  - Shrinkage
  - Dimension Reduction

6.1 Subset Selection

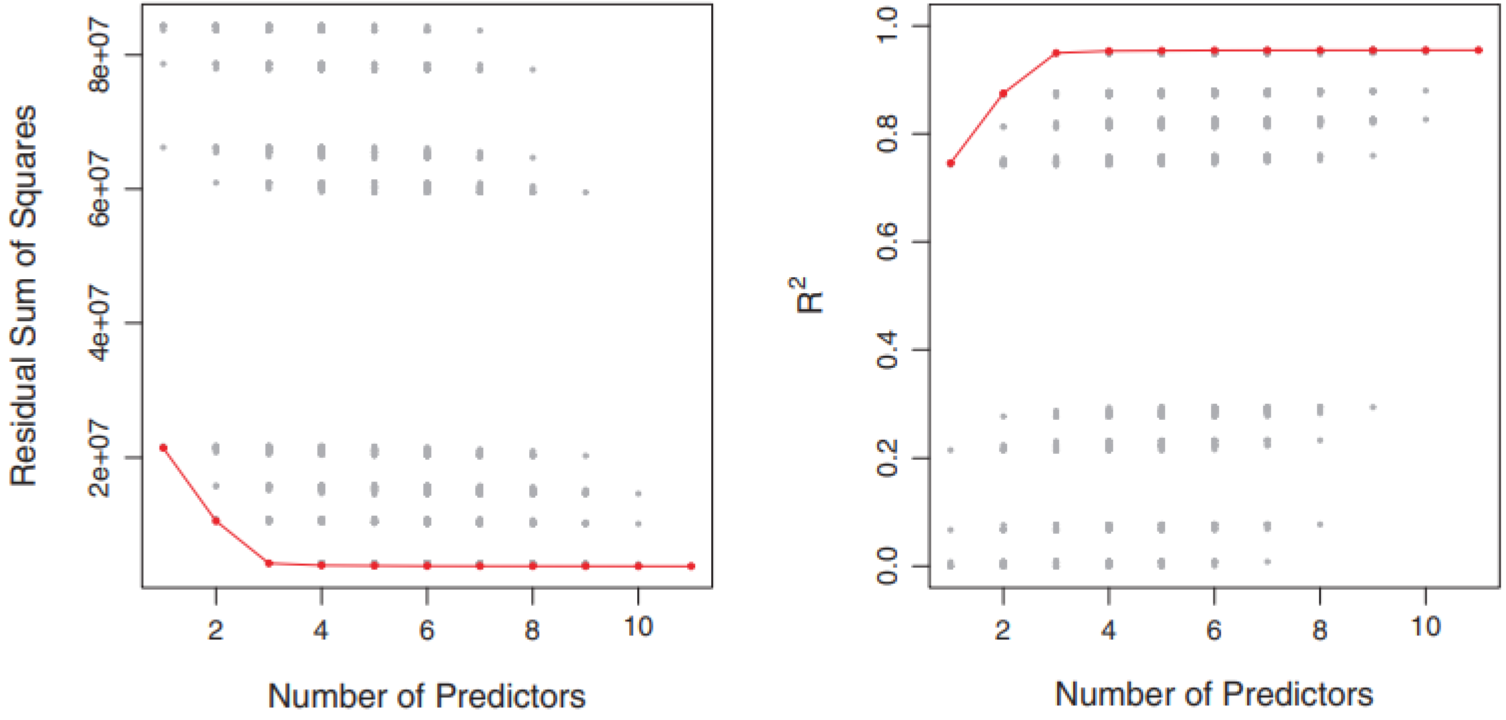
- 6.1.1 Best Subset Selection
- 6.1.2 Stepwise Selection
- 6.1.3 Choosing the Optimal Model

6.1.1 Best Subset Selection

- selecting the best model from among the 2^p possibilities
  - select the best subset of $\{X_1, \dots, X_p\}$

- Algoritm 6.1 Best subset selection
  - 1. Let $\mathcal{M}_0$ denote the null model
  - 2. For $k = 1, \dots, p$, let $\mathcal{M}_k$ be the best among $\binom{p}{k}$ models with k predictors
  - 3. Select a single best model from among $\mathcal{M}_0, \dots, \mathcal{M}_p$

- $\mathcal{M}_0$ predicts the sample mean as response for each observation
- In step2, $\mathcal{M}_k$ is the model with the smallest RSS (largest R^2)
- In step3, cross-validated prediction error, C_p(AIC), BIC, or adjusted R^2 are used to select the best


Residual Sum of Squares
8e+07
6e+07
4e+07
2e+07
2
4
6
8
10
Number of Predictors
R^2
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
Number of Predictors

6.1.2 Stepwise Selection

- best subset selection cannot be applied with very large p
  - 2^p possibilities
  - computationally infeasible for $p \geq 40$
- stepwise selection

Forward Stepwise Selection

- Algorithm 6.2 Forward stepwise selection
  - 1. Let $\mathcal{M}_0$ denote the null model
  - 2. For $k = 1, \dots, p-1$, choose the best among $\mathcal{M}_k$ with one additional predictor, and call it $\mathcal{M}_{k+1}$
  - 3. Select a single best model from among $\mathcal{M}_0, \dots, \mathcal{M}_p$

- # of models that are considered $= 1 + \frac{p(p+1)}{2}$

Backward Stepwise Selection

- Algorithm 6.3 Backward stepwise selection
  - 1. Let $\mathcal{M}_p$ denote the full model
  - 2. For $k = p, \ldots, 1$, choose the best among models that contain all but one of the predictors in $\mathcal{M}_k$, and call it $\mathcal{M}_{k-1}$
  - 3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$

Hybrid Approaches

- hybrid versions of forward and backward stepwise selection
  - In forward stepwise selection, after adding new variable, remove variables, if any, that no longer provide an improvement

6.1.3 Choosing the Optimal Model

- Selecting a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$
  - Comparing RSSs directly is meaningless because RSS of $\mathcal{M}_k$ decreases (or R^2 increases) as k increases
  - one may use cross-validated prediction error, C_p, AIC, BIC, or Adjusted R^2

C_p

- An estimate of test MSE
- least squares model with d predictors

$$C_p = \frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$$

  where $\hat{\sigma}^2$ is an estimate of $\mathrm{Var}(\epsilon)$
  - RSS with penalty $2d\hat{\sigma}^2$

AIC(Akaike information criterion)

- Defined for models fit by maximum likelihood

$$\mathrm{AIC} = \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2)$$

  where d = #predictors, $\hat{\sigma}^2 \sim \mathrm{Var}(\epsilon)$

- C_p and AIC are proportional to each other

BIC(Bayesian information criterion)

- Derived from a Bayesian point of view

$$\mathrm{BIC} = \frac{1}{n}(\mathrm{RSS} + \log n \, d\hat{\sigma}^2)$$

  where d = #predictors, $\hat{\sigma}^2 \sim \mathrm{Var}(\epsilon)$

Adjusted R^2

- least squares model with d predictors

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

- Compare with

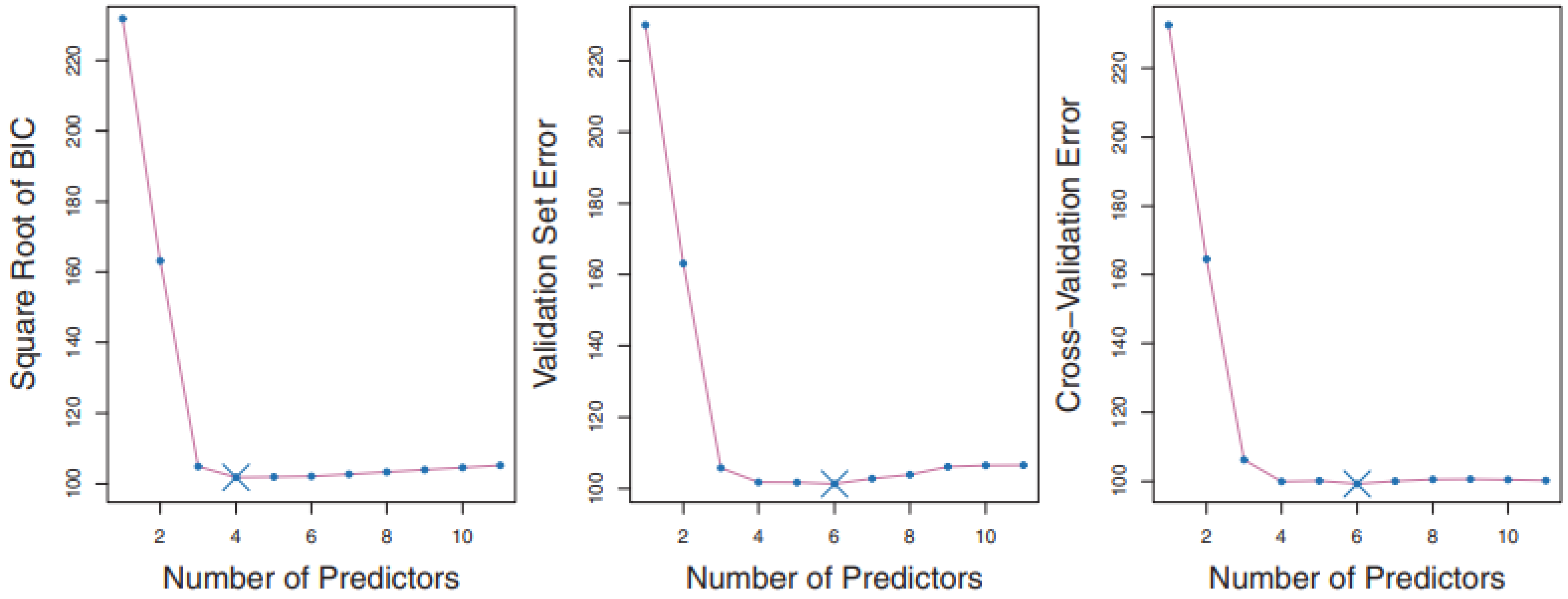
$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$$

Validation and Cross-Validation

- directly estimate the test error
  - using the validation set
  - using cross-validation methods

Trends

- AIC, BIC, C_p, and adjusted R^2 → Validation and CV
  - computing power



6.2 Shrinkage Methods

- Linear model
  - Least square to fit the model
- Regularize the coefficient estimates
  - shrink the coefficient estimates towards zero

- 6.2.1 Ridge Regression
- 6.2.2 The Lasso
- 6.2.3 Selecting the Tuning Parameter

6.2.1 Ridge Regression

- Least square minimizes

$$\mathrm{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$

- Ridge regression minimizes

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$
$$= \mathrm{RSS} + \lambda \parallel \beta \parallel_2^2$$

  where $\lambda \geq 0$ is a tuning parameter
    - $\lambda\sum_{j=1}^{p}\beta_j^2$ is called a shrinkage penalty

Example

- $x = \{1.0, 2.0, 3.0\}$
- $y = \{1.1, 1.9, 3.0\}$
- Least square

$$\mathrm{RSS} = \sum_{i=1}^{3}(y_i - \beta_0 - \beta_1 x_i)^2$$
$$\frac{\partial \mathrm{RSS}}{\partial \beta_0} = 2(3\beta_0 + \sum x_i \beta_1 - \sum y_i) = 0$$
$$\frac{\partial \mathrm{RSS}}{\partial \beta_1} = 2\left(\sum x_i \beta_0 + \sum x_i^2 \beta_1 - \sum x_i y_i\right) = 0$$
$$\sum x_i = 6, \sum y_i = 6, \sum x_i^2 = 14, \sum x_i y_i = 13.9$$
$$\beta_0 = \cdots,\ \beta_1 = \cdots$$

- Ridge

$$R = \sum_{i=1}^{3}(y_i - \beta_0 - \beta_1 x_i)^2 + \lambda\ \beta_1^2$$

$$\frac{\partial R}{\partial \beta_0} = 2(3\beta_0 + \sum x_i \beta_1 - \sum y_i) = 0$$

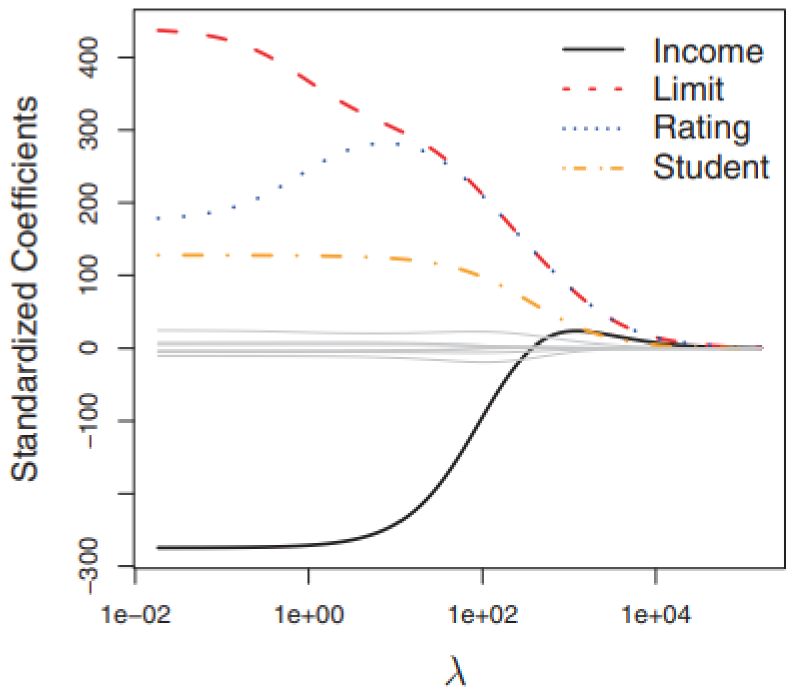
$$\frac{\partial R}{\partial \beta_1} = 2\left(\sum x_i \beta_0 + \left(\lambda + \sum x_i^2\right)\beta_1 - \sum x_i y_i\right) = 0$$

$$\sum x_i = 6, \sum y_i = 6, \sum x_i^2 = 14, \sum x_i y_i = 13.9$$

$$3\beta_0 + 6\beta_1 = 6$$

$$6\beta_0 + (\lambda + 14)\beta_1 = 13.9$$

$$\beta_0 = \cdots, \beta_1 = \cdots$$



- “Standardized” means that the standard deviation of each predictor is 1, using the formula

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

6.2.2 The Lasso

- The Lasso minimizes

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

$$= \mathrm{RSS} + \lambda \parallel \beta \parallel_1$$

where $\lambda \geq 0$ is a tuning parameter

Example

- Lasso

$$R = \sum_{i=1}^{3}(y_i - \beta_0 - \beta_1 x_i)^2 + \lambda|\beta_1|$$

$$\frac{\partial R}{\partial \beta_0} = 2(3\beta_0 + \sum x_i \beta_1 - \sum y_i) = 0$$

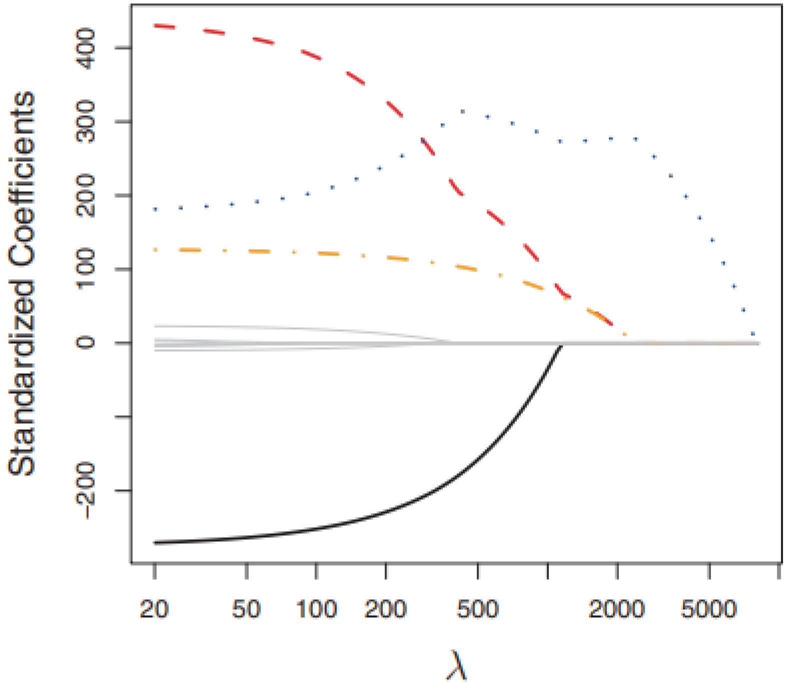
$$\frac{\partial R}{\partial \beta_1} = 2\left(\sum x_i \beta_0 + \sum x_i^2 \beta_1 - \sum x_i y_i \pm \lambda\right) = 0$$

$$\sum x_i = 6, \sum y_i = 6, \sum x_i^2 = 14, \sum x_i y_i = 13.9$$

$$3\beta_0 + 6\beta_1 = 6$$

$$6\beta_0 + 14\beta_1 = 13.9 \pm \lambda$$

$$\beta_0 = \cdots, \beta_1 = \cdots$$


Standardized Coefficients
400
300
200
100
0
−200
20
50
100
200
500
2000
5000
λ

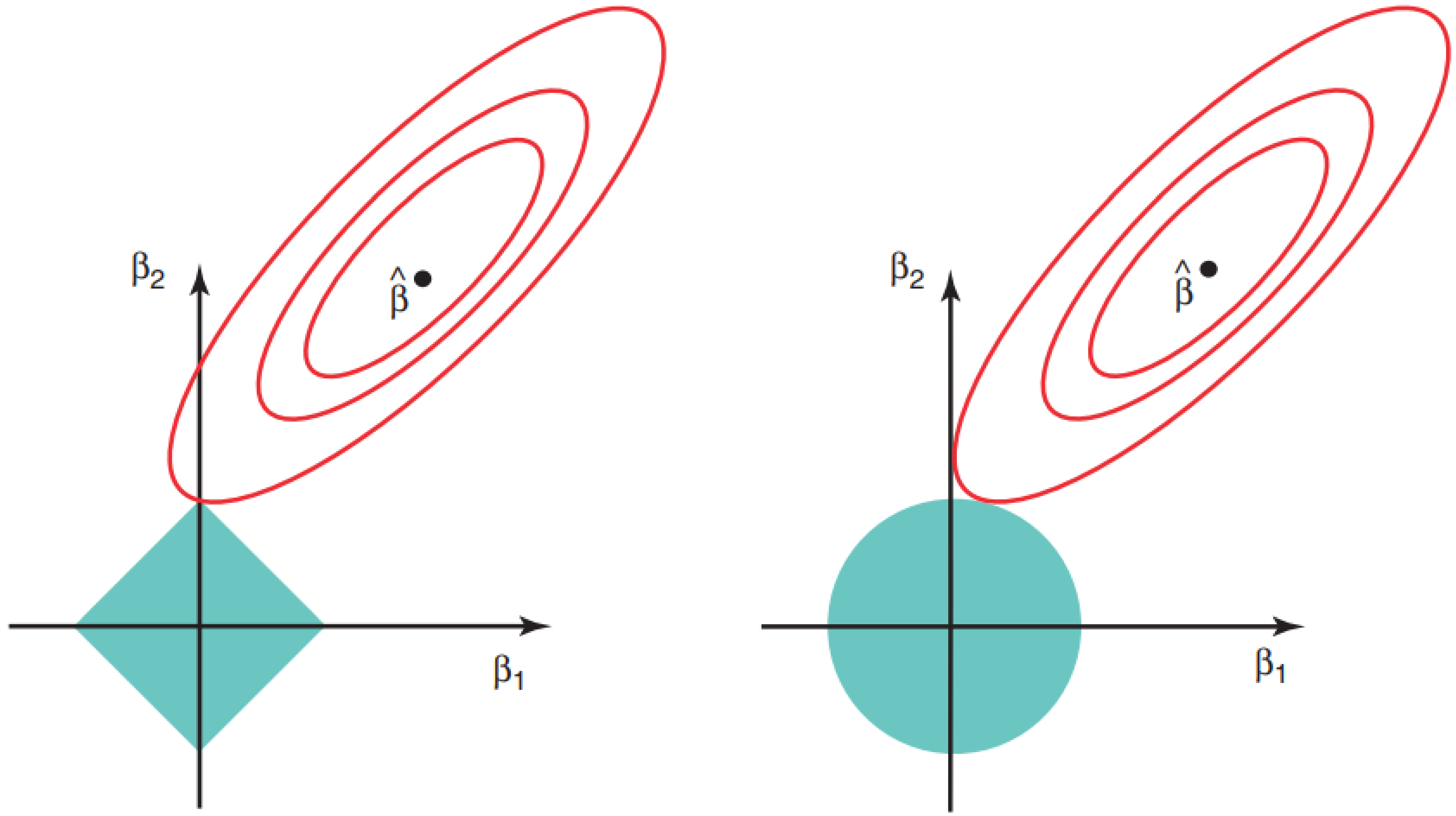
Another formulation

- Lasso

$$\underset{\beta}{\text{minimize}}\left\{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2\right\} \text{ subject to } \sum_{j=1}^{p}|\beta_j| \le s$$

- Ridge

$$\underset{\beta}{\text{minimize}}\left\{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2\right\} \text{ subject to } \sum_{j=1}^{p}\beta_j^2 \le s$$



β_2
$\hat{\beta}$
β_1
β_2
$\hat{\beta}$
β_1

```python
import numpy as np
import pandas as pd
from numpy.linalg import inv

adv = pd.read_csv('data/Advertising.csv').values
for i in range(1, 4):
    adv[:, i] /= np.std(adv[:, i])
p, n = 3, adv.shape[0]
one = np.ones((n, 1))
X = np.concatenate((one, adv[:, 1:4]), axis=1)  # TV, radio, newsp
y = adv[:, -1:]  # sales
M = np.matmul(X.T, X)  # X^T X
C = np.matmul(X.T, y)  # X^T y

print('############## LeastSquare ########################')
beta = np.matmul(inv(M), C)  # (X^T X)^-1 X^T y
print('beta =', beta.flatten())
print('############## Ridge ########################')
########### compute ##################
rate = 100
idm = np.eye(p + 1)  # (p+1) x (p+1) identity matrix
idm[0, 0] = 0  # exclude beta_0
MR = M + rate * idm,
beta = np.matmul(inv(MR), C)
print('beta =', beta.flatten())

############# gradient descent ######################
def error(beta):
    v = y - np.matmul(X, beta)
    return np.sum(v * v) + rate * np.sum(beta[1:] * beta[1:])
```

```python
def grad(beta):
    beta0 = beta.copy()
    beta0[0, 0] = 0
    # 정확한 값은 2 *(np.matmul(M, beta) - C + rate * beta0)
    return np.matmul(M, beta) - C + rate * beta0

stepsize = 0.0001
prevErr, currErr = 0.0, np.inf
beta = np.random.rand(p + 1, 1)  # initialize beta
while np.abs(prevErr - currErr) > 0.00001:
    beta -= stepsize * grad(beta)  # upgrade
    prevErr, currErr = currErr, error(beta)
print('beta =', beta.flatten())

print('############## Lasso ######################')

############## gradient descent ###################
def error(beta, rate):
    v = y - np.matmul(X, beta)
    return np.sum(v * v) + rate * np.sum(np.abs(beta[1:]))

def grad(beta, rate):
    sign = np.sign(beta)
    sign[0, 0] = 0.0
    return 2 * (np.matmul(M, beta) - C) + rate * sign

rate = 100.0
stepsize = 0.0001
prevErr, currErr = 0.0, np.inf
beta = np.random.rand(p + 1, 1)  # initialize beta
while np.abs(prevErr - currErr) >= 0.001:
    beta -= stepsize * grad(beta, rate)  # upgrade
    prevErr, currErr = currErr, error(beta, rate)
print('beta =', beta.flatten())
```

```
############## LeastSquare ########################
beta = [ 2.93888937  3.91925365  2.79206274 -0.02253861]
############## Ridge ########################
beta = [6.23878001 2.63696539 1.84208228 0.25766613 ]
beta = [6.23278076 2.63826979 1.84307432 0.2584704 ]
############## Lasso ####################
beta = [3.44650691 3.75037636 2.60798495 0.0107319 ]
```

6.2.3 Selecting the Tuning Parameter

- Determine the tuning parameter λ
  - Cross-validation

6.3 Dimension Reduction Methods

- Dimension reduction
  - Predictors: $X_1, \dots, X_p$
  - Find new predictors $Z_1, \dots, Z_M$ for $M < p$ with

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

  - Fit the linear regression model with coefficients $\theta_0, \dots, \theta_M$

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i$$

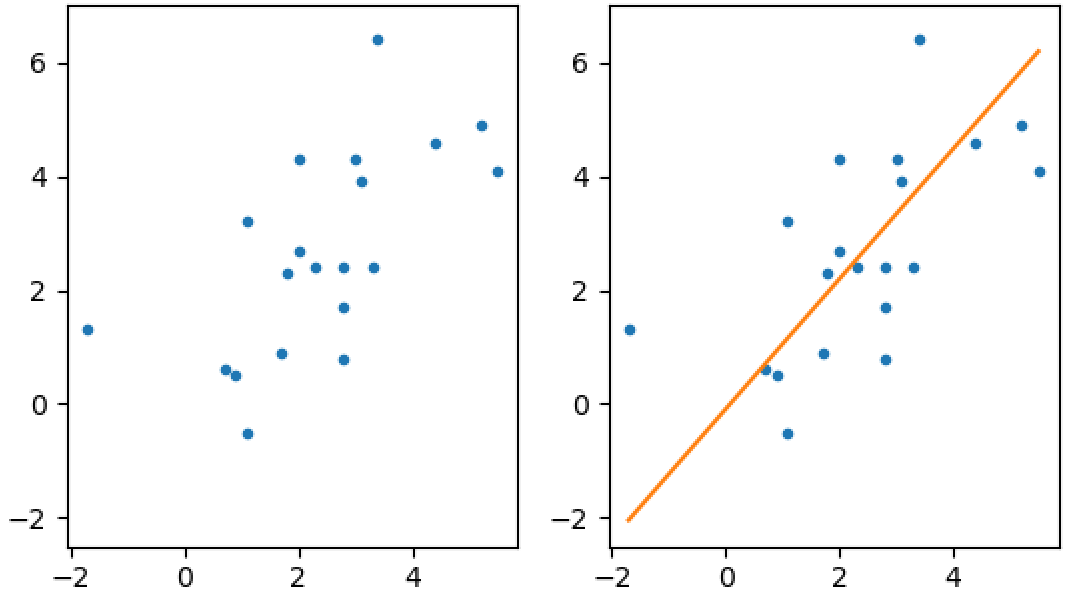
- 6.3.1 Principal Components Regression
- 6.3.2 Partial Least Squares

6.3.1 Principal Components Regression

- Principal components analysis(PCA)
  - A technique for reducing the dimension of a data matrix

PCA

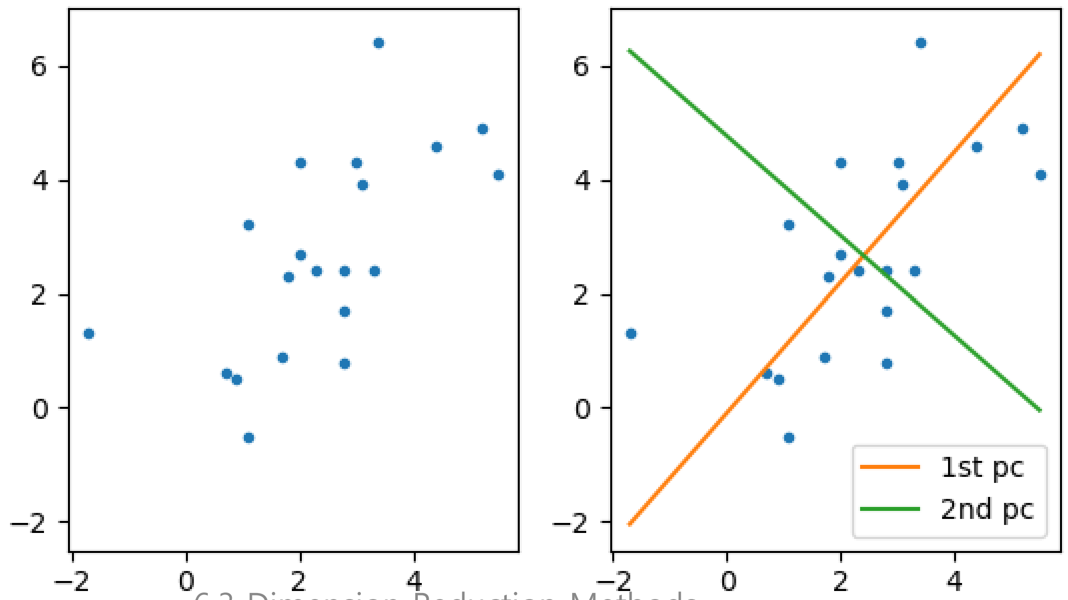
- The first principal component
  - The direction along which observations vary the most



$n = 20, p = 2$
(-1.7, 1.3)
(1.7, 0.9)
(1.1, -0.5)
(0.7, 0.6)
(0.9, 0.5)
(2.0, 2.7)
(2.8, 0.8)
(1.8, 2.3)
(2.3, 2.4)
(2.8, 1.7)
(1.1, 3.2)
(2.8, 2.4)
(3.3, 2.4)
(3.0, 4.3)
(2.0, 4.3)
(4.4, 4.6)
(3.4, 6.4)
(3.1, 3.9)
(5.5, 4.1)
(5.2, 4.9)

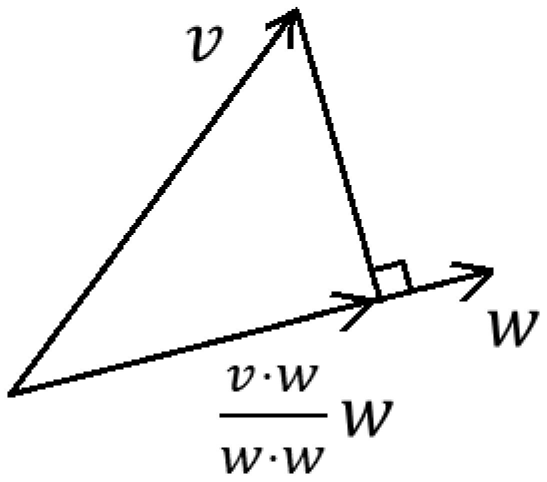
PCA

- The second principal component
  - The direction along which observations vary the second most
  - The 1^{st} and 2^{nd} principal components are orthogonal
- And so on



Inner product

- The projection of a vector v into the direction w is $\frac{v \cdot w}{w \cdot w} w$
- The norm of the projection is $|v \cdot w|$ if $\| w \| = 1$



Computation

- Observation
  - $\mathrm{X}_1 = (x_{11}, \dots, x_{1p})$, …, $\mathrm{X}_n = (x_{n1}, \dots, x_{np})$
  - $X = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$
- Centering for simplicity
  - Assume $\sum_{i=1}^{n} x_{ij} = 0$ for all j
    - or, let $x_{ij} \coloneqq x_{ij} -$ mean of j–th column

The first principal component

- Let w be a unit vector
- The projection of X_i into the direction w is of length

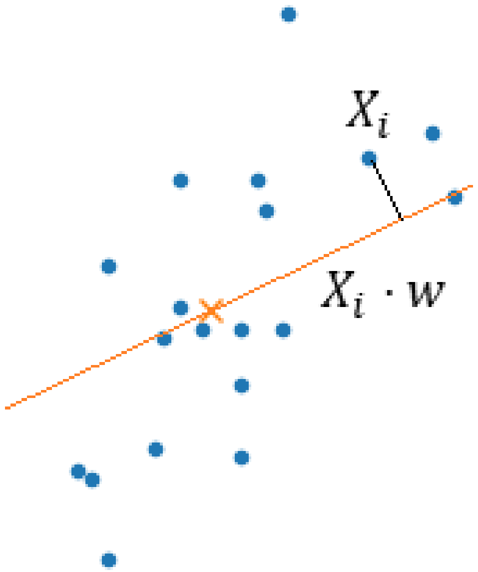
$$X_i \cdot w$$

- Goal: find w_1 which maximizes the variance of

$$A = \{X_1 \cdot w, \dots, X_n \cdot w\}$$

  i.e.

$$Z_1 = \underset{w}{\operatorname{argmax}} \operatorname{Var}(A)$$



- A has zero mean, $\mu(A) = 0$
- The variance is

$$\mathrm{Var}(A) = \sum (X_i \cdot w)^2 = w^T X^T X w$$

- Note that $X = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$
- Hence

$$Z_1 = \underset{w}{\operatorname{argmax}}\, w^T X^T X w$$

- $X^T X$ is symmetric
- Z_1 is the eigenvector corresponding to the largest eigenvalue

The other principal component

- The k-th principal component Z_k is the eigenvector corresponding to the k-th eigenvalue in order

Code

- Find eigenvalues and eigenvectors

```
X = np.array([[-1.7, 1.7, 1.1, 0.7, 0.9, 2.0, 2.8, 1.8, 2.3, 2.8,
               1.1, 2.8, 3.3, 3.0, 2.0, 4.4, 3.4, 3.1, 5.5, 5.2],
              [1.3, 0.9, -0.5, 0.6, 0.5, 2.7, 0.8, 2.3, 2.4, 1.7,
               3.2, 2.4, 2.4, 4.3, 4.3, 4.6, 6.4, 3.9, 4.1, 4.9]]).T
X -= np.mean(X, axis=0) # centering
M = np.matmul(X.T, X)
eig_val, eig_vec = np.linalg.eig(M) # eigenvalues and eigengectors
```

- Sorting

```
order = np.argsort(-eig_val) # descending order
eig_val = eig_val[order]
eig_vec = eig_vec[:, order]
```

- Print

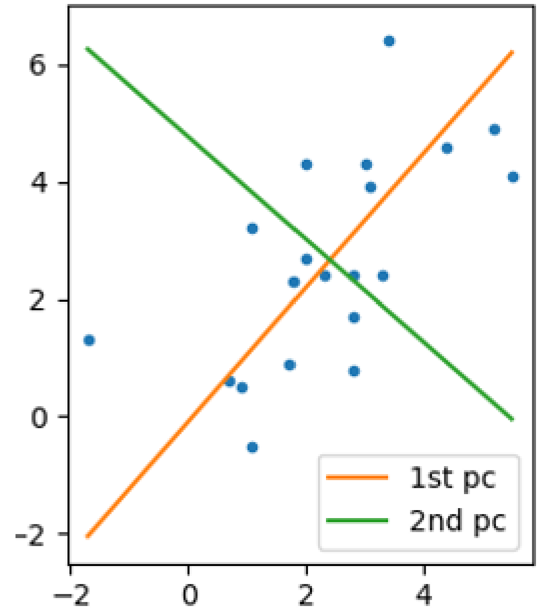
```
for i in range(2):
    print(eig_val[i], eig_vec[:, i])
```

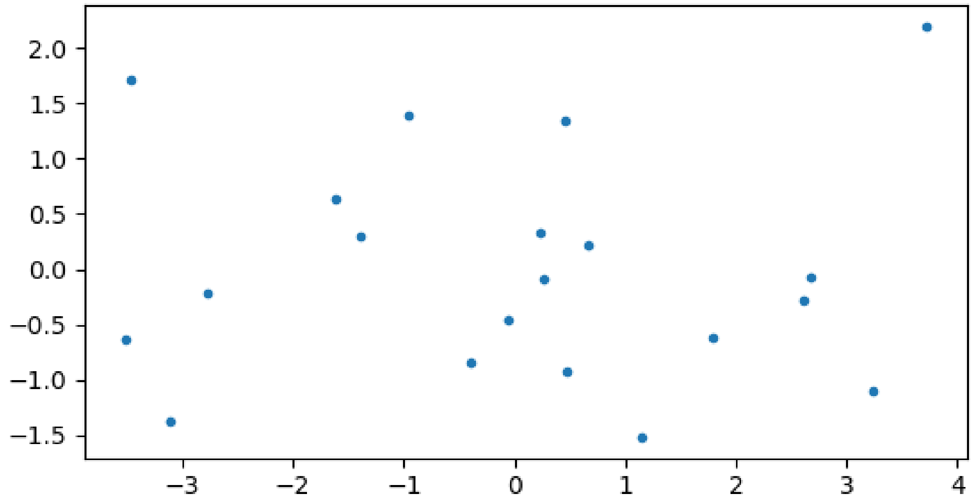
```
91.44010228439512 [-0.6581733  -0.75286646]
20.26589771560488 [-0.75286646  0.6581733 ]
```

New predictors

•

```
newX = np.matmul(X, eig_vec)
plt.plot(newX[:, 0], newX[:, 1], '.')
plt.gca().set_aspect('equal')
plt.show()
```





Principal Components Regression

- Principal components regression(PCR)
  - Standardization
    - Variance of each predictor to be 1
  - New predictor selection
  - Cross-validation

6.3.2 Partial Least Squares

- Partial Least Square(PLS)
  - find directions that help explain both the response and the predictors

6.4 Considerations in High Dimensions

- High dimension
  - $p > n$ or $p \approx n$
  - Overfitting problem
  - Model is flexible – variable
- Example
  - DNA or SNPs(single nucleotide polymorphisms)
  - Bag-of-words model
    - search engine → marketing
- Solution
  - Regularization or shrinkage