ybigta science team

6. Linear Model Selection and Regularization

ISL 스터디 신보현

August 11, 2018

전통적인 선형 회귀 모델은 $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$의 관계를 가정하고 Least Sqaures Estimation을 통해서 $RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{i=1}^{p} \beta_j x_{ij})^2$ 를 최소로 만드는 $\beta_0, \cdots, \beta_p$에 대한 estimation인 $\hat{\beta}_0, \cdots, \hat{\beta}_p$ 을 찾는다. 하지만 이러한 선형회귀는 Y와 X가 선형이라고 가정을 하기 때문에 실제 Y와 X의 관계가 비선형 (non-linear) 이거나 Y와 상관 없는 예측변수들이 많을 경우, $n \approx p$ 또는 $n < p$일 경우, 모델의 성능이 상당히 떨어질 수 있다. 이러한 LSE의 단점에 대한 대안으로, 이번 6장에서는 크게 3가지 방법 (Subset Selection, Shrinkage, Dimension Reduction) 을 소개한다.

6.1 Subset Selection

변수가 여러 개 있을 때, 그 중 일부만을 예측변수로 사용하기 위해 쓰이는 방법이다.

6.1.1 Best Subset Selection

쉽게 생각하면, p개의 변수가 있을 때, 2^p 개의 가능한 모델들 중 가장 좋은 모델을 택하는 것이다.

Algorithm 6.1 *Best subset selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

하지만 계산상의 문제가 있다. 만약 $p = 10$, 즉 예측 변수가 10개만 되더라도 고려해야 하는 모델이 $2^{10} = 1024$개나 된다. 또한 고려해야하는 모이 많아질수록 test data가 아니라 training data에 더 맞는 모델을 찾을 수가 있어서, 과적합의 위험도 있다.

6.1.2 Stepwise Selection

Forward Stepwise Selection

예측 변수들이 없는 null model에서 시작해, 변수를 한 개씩 추가해보면서 가장 큰 추가적인 향상을 보이는 변수를 선택한다.

Algorithm 6.2 *Forward stepwise selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Backward Stepwise Selection

모든 예측 변수들이 있는 full model에서 시작해, 변수를 한 개씩 제거해보면서 가장 작은 향상을 보이는 변수를 제거한다.

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

하지만, forward selection과 backward selection는 가능한 모든 2^p개의 모델들 중, 가장 좋은 p개 중 일부 변수를 포함하는 모델을 만들것이라는 보장은 없다.

# Variables	Best subset	Forward stepwise
One	`rating`	`rating`
Two	`rating`, `income`	`rating`, `income`
Three	`rating`, `income`, `student`	`rating`, `income`, `student`
Four	`cards`, `income`, `student`, `limit`	`rating`, `income`, `student`, `limit`

Hybrid Approaches

forward selection 하면서 backward selection도 동시에 하는 방법이다.

6.1.3 Choosing the Optimal Model

항상 주의해야할 점은 training error가 아니라 test error을 고려해야한다는 것이다. 왜냐하면 training data에 대한 RSS은 변수가 추가 됨에 따라서 일정하게 작아지고, 결정계수 (R^2)도 일정하게 증가하기 때문이다. 따라서 forward, backward의 (c)단계에서 RSS나 R^2가 아닌 여러 다른 지표들을 이용해서 가장 최적의 모델을 선택한다.

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$R^2 = 1 - RSS/TSS$$

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

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

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

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

위의 지표들은 어느정도 가정을 필요로하기도 하고, 실제로 $\hat{\sigma}^2$(오차항 ϵ의 분산에 대한 estimate)을 구하기 힘든 경우도 있을 수 있다. 이럴 때는 $Cross\ Validation$을 통해서 test error에 대한 estimate을 구하자!!

6.2 Shrinkage Methods

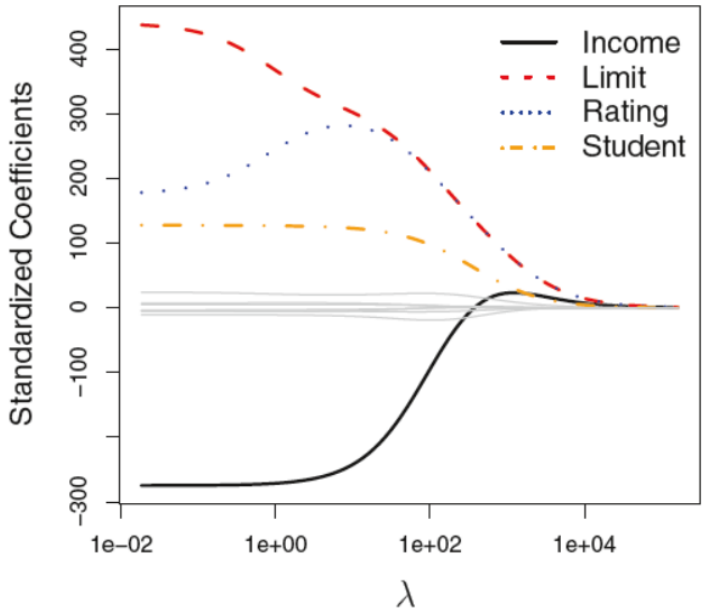
6.2.1 Ridge Regression

Least Squares에서는 $\beta_0, \cdots, \beta_p$에 대한 estimates인 $\hat{\beta}_0, \cdots, \hat{\beta}_p$를 $RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{i=1}^{p}\beta_j x_{ij})^2$를 최소화하는 방법에 의해 찾았다. Ridge regression은 이와 아주 유사한데, RSS에 penalty을 더한

$$RSS_{ridge} = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

을 최소화하며 계수들에 대한 estimates을 찾는다.

여기서 λ은 tuning parameter라고 불리며 계수들을 수축(shrinking)하는 효과를 가져온다. 만약 λ가 크다면, Ridge의 penalty항의 영향력이 커질 것이고, RSS_{ridge}를 최소화할때, λ와 곱해져있는 $\sum_{j=1}^{p}\beta_j^2$ 때문에 β_j을 작게 만들 수밖에 없다. 반대로, λ가 작다면, Ridge의 penalty항의 영향력이 작아질 것이고, 마침내 $\lambda = 0$이 된다면 원래의 LSE와 동일한 결과를 낼 것이다.



6.2.2 The Lasso

Ridge는 의미가 적은 계수를 정확히 0으로 만드는 것이 아니라 0에 가깝게 만든다. 그에 따라서 Ridge는 엄밀하게 variable selection을 하지는 않는다. 이에 대한 대안으로, Lasso는 계수를 정확히 0으로 만듦으로써 variable selection을 한다.

$$RSS_{lasso} = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

Ridge와 Lasso의 형태는 penalty에서만 다르고, 그 이외는 동일하다.

기억해야할 것!!! Ridge나 Lasso를 시행하기 전에 예측변수들을 표준화하자!! 일반적인 선형회귀는 굳이 표준화를 안 해도 되지만(scale equivariant하기 때문에) Ridge나 Lasso에서는 penalty의 존재 때문에 scale이 다름으로 인해서 영향을 받기 때문에 꼭 표준화를 해야 한다!

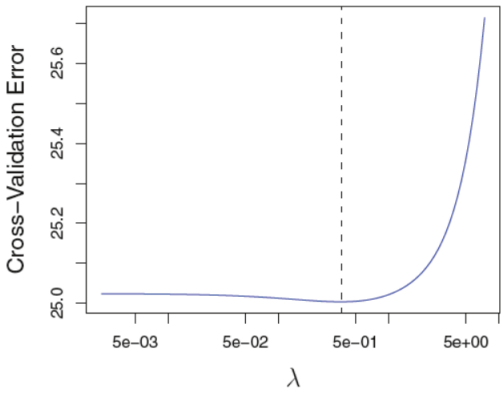
$$\widetilde{x_{ij}} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}}$$

Ridge vs Lasso

아무래도 Lasso는 변수 선택을 하므로 만약 p개의 예측변수가 골고루 반응변수와 관련이 되어 있으면 Lasso보다는 Ridge가 더 나은 성능을 보이고 p개의 예측변수 중 일부만 반응변수와 관련이 있으면 Lasso가 더 좋은 성능을 보일 것이다.

6.2.3 Selecting the Tuning Parameter

Ridge와 Lasso을 할 때, 적절한 λ을 정하는 것이 매우 중요하다. 적절한 λ는 Cross-Validation을 통해서 결정한다.



6.3 Dimension Reduction Methods

p개의 변수를 가지는 모형을 M($<$ p)개의 변수를 가지는 모형으로 차원을 축소시키는 방법.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

$$Transformation\ Z_m = \sum_{j=1}^{p} \phi_{jm} X_j\ where\ \phi_{jm}\ are\ constant\ and\ M < p$$

$$\begin{aligned} Y &= \theta_0 + \theta_1 Z_1 + \theta_2 Z_2 + \cdots + \theta_M Z_M + \epsilon \\ &= \theta_0 + \sum_{m=1}^{M} \theta_m z_m + \epsilon \end{aligned}$$

여기서 차원 축소 접근을 원래 모형의 계수들에 대해 제약(constraints)을 두는 것이라고 볼 수 있다.

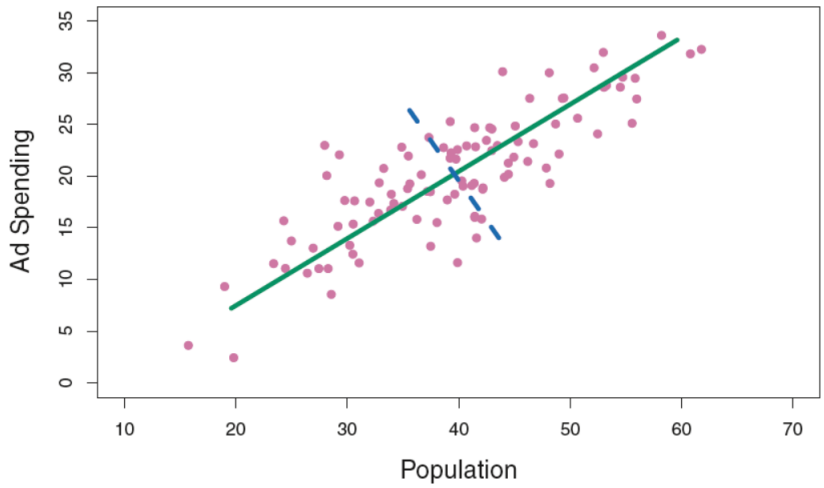
$$\sum_{m=1}^{M} \theta_m z_m = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{jm} x_j = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \phi_{jm} x_j = \sum_{j=1}^{p} \beta_j x_j$$

$$where\ \beta_j = \sum_{m=1}^{M} \theta_m \phi_{jm}$$

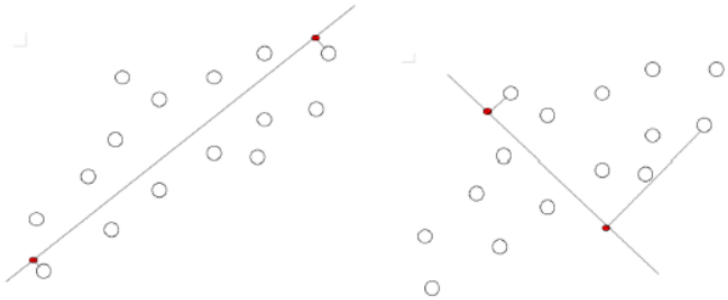
이는 bias-variacne trade off 관점에서 잠재적인 bias의 가능성이 있지만 $M << p$인 M을 고른다면 분산을 상당히 많이 줄여줄 것이다.

An Overview of Principal Components

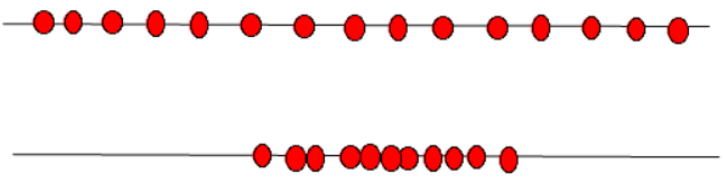
아래는 Population과 Ad Spending에 따른 100개의 데이터이다.



Z_1= 초록색 선 = first principal component = 주성분(초록색 선)으로 데이터를 사영(projection = 가장 짧은 거리로 투영)했을 때, 가장 큰 분산을 가지는 벡터(= 가장 많은 정보를 보존).



첫번째 주성분으로 사영(projection) 후 1차원으로 나타낸 결과.



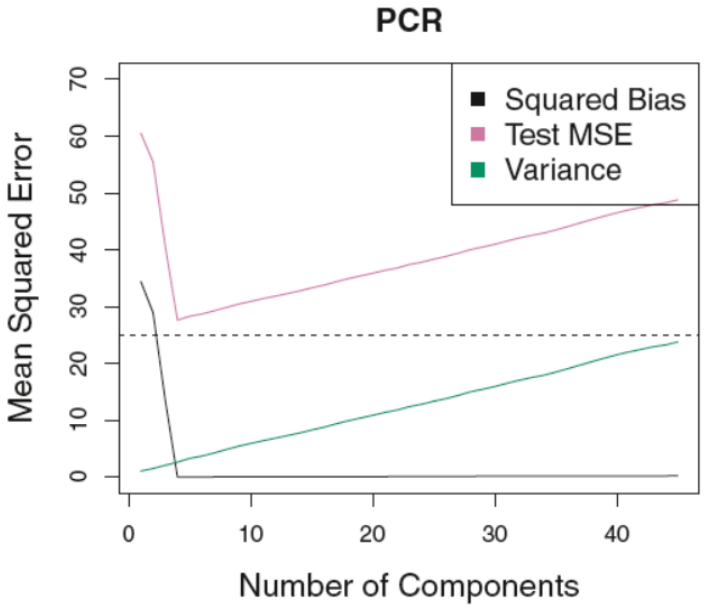
Second Principal Component: 우선, 첫번째 주성분과 관련이 없어야하고(= 첫번째 주성분으로 설명이 된 것에는 관심이 없음 = Z_1과 Z_2는 수직[orthogonal]) 가장 큰 분산을 가져야 한다.

자세한 내용은 10장에서..... ㅎㅎ

6.3.1 Principal Components Regression

PCR은 위에서 변환된 $Z_1, \cdots, Z_M$을 예측변수로써 사용하여 선형회귀를 실시하는 접근법이다. 선형회귀가 예측변수와 반응변수간에 선형의 관계가 있다고 가정하는 것과 유사하게 PCR도 principal component을 나타내는 direction이 반응변수와 연관이 있다고 가정한다.

언제 PCR을 쓰면 좋을까?



PCR이 좋은 성능을 보일 때: 처음의 적은 주성분(principal components)로 반응변수의 많은 부분을 설명할 수 있는 데이터에 PCR을 쓰면 좋다. (= 예측변수들 간의 다중공선성이 나타날 때)

PCR이 나쁜 성능을 보일 때: 적절한 모델을 만들기 위해 많은 주성분이 필요한 데이터에 대해서는, 굳이 차원을 축소(= 정보의 손실)를 할 필요가 없다.

기억해야할 것!! PCR을 시행할 때, 표준화를 하는 것이 좋다. 단위가 다르게 측정이 된 경우, 그 변동성이 주성분을 결정할 때 영향을 미치기 때문이다.

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

6.3.2 Partial Least Squares

PCR은 주성분 방향(principal component directions)을 구할 때, 예측변수들만을 이용하여 주성분 방향을 구한다. 즉, 원래 주성분이 있고 이것을 학습(supervise)하지 않는다. 따라서, PCA를 통해서 구한 주성분이 예측변수를 잘 설명한다고 해도(variance가 가장 큰 애들로 구했음) 반응변수를 잘 설명하리라는 보장은 없다!

이러한 한계점을 보완하여 나온 것이 Partial Least Squares이다. 즉, PLS는 주성분을 학습(supervise)을 통하여 도출한다. 다시 말하면, PLS는 예측변수 뿐만 아니라 반응변수까지 고려하여 이것들을 가장 잘 설명하는 주성분 방향을 구한다.

간단한 실습

Hitters 데이터에 대해서 lasso와 ridge을 이용해서 Hitters 데이터의 Salary 변수를 예측해 본다.

```
library(ISLR)
hitters = na.omit(Hitters)
x=model.matrix(Salary~. , hitters)[,-1] # matrix 형태
y=hitters$Salary # 백터 형태
```

```
# ridge와 lasso 함수가 있는 패키지 불러오기
library(glmnet)
```

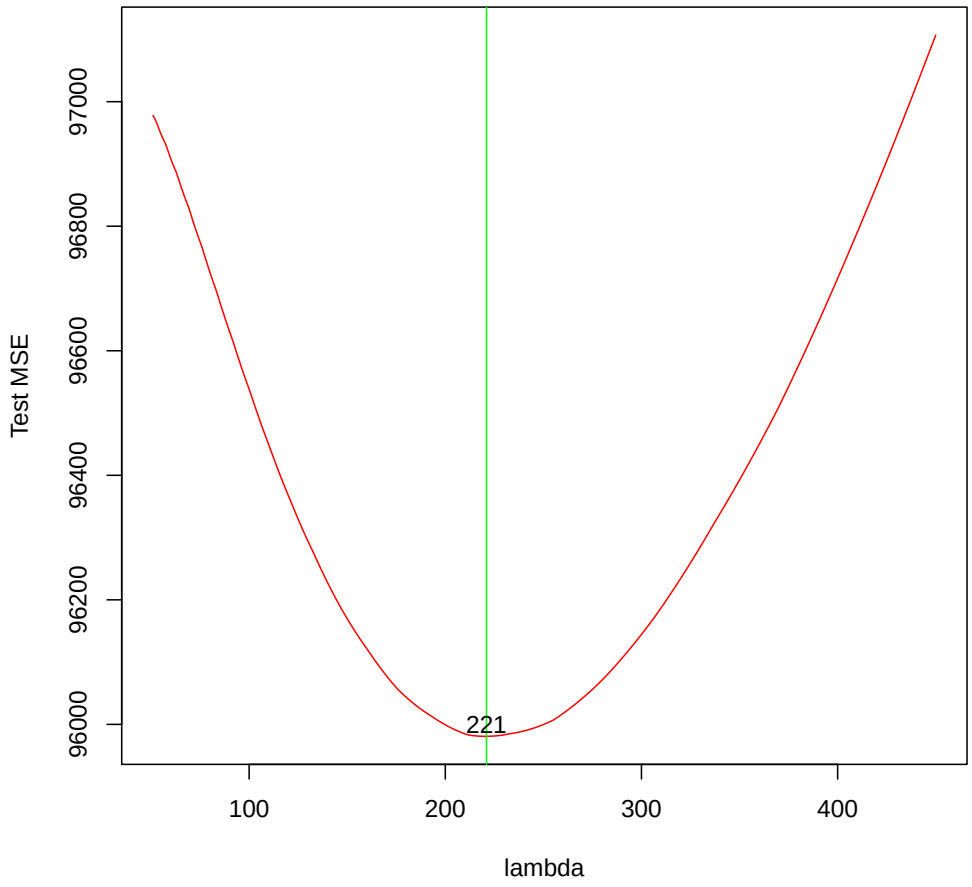
```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16
```

```
# train, test data 나누기
set.seed(1)
train = sample(1:nrow(x), nrow(x)/2)
test = -train
```

```
# ridge 모델 적용하기
# 변수 표준화는 자동으로!
# alpha가 0일때는 ridge
# 기본 값은 k=10(ten folds)
mod.ridge = glmnet(x[train,],y[train],alpha=0)
set.seed(1)
cv.rid = cv.glmnet(x[train,],y[train],alpha=0)
cv.rid$lambda.min
```

```
## [1] 211.7416
```

```
# 과연 이때, test mse가 최소일까?
a = 51:450
b = rep(0,400)
c = 0
for (i in a){
c = c + 1
pred.ridge = predict(mod.ridge,s=i,newx=x[test,])
b[c] = mean((pred.ridge-y[test])^2)
}
plot(a,b,type='l',col='red',xlab='lambda',ylab='Test MSE')
abline(v=a[which.min(b)],col='green')
text(221,96000,'221')
```



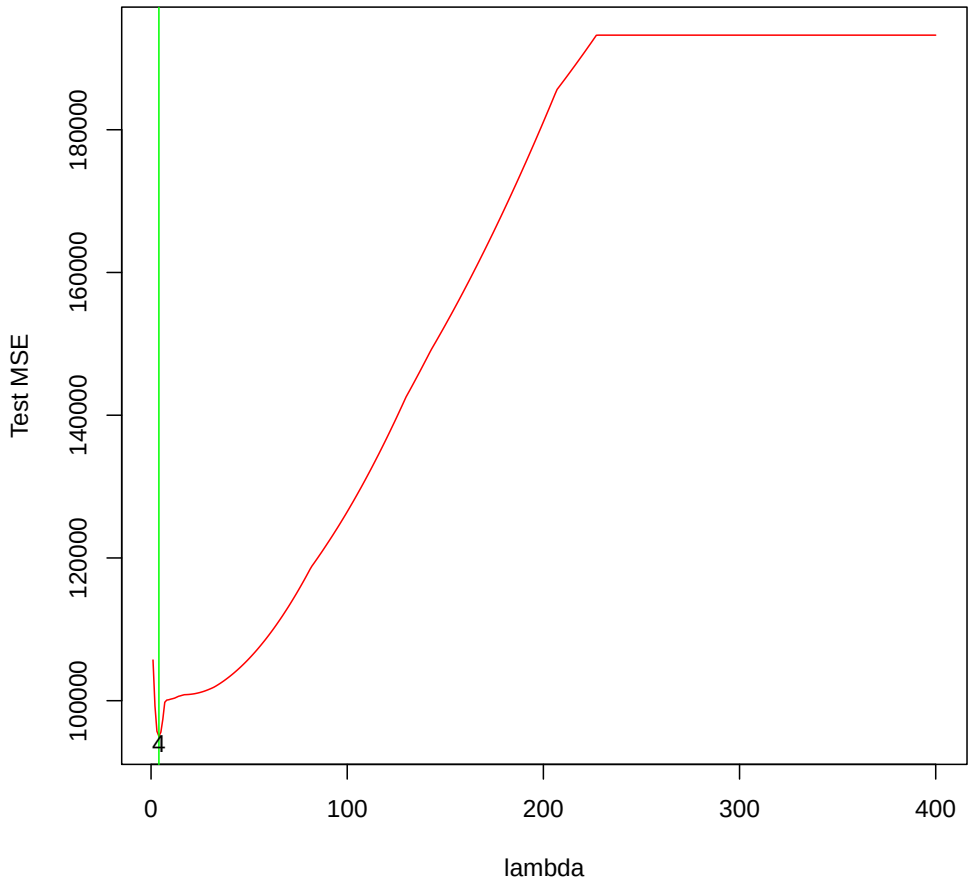

```
# 최소의 CV error를 갖는 람다로 전체 데이터를 적합시킨 후, 계수를 살펴보자
final.model = glmnet(x,y,alpha=0)
predict(final.model,type='coefficients',s=212)[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##   9.88490750   0.03161240   1.00833660   0.14071245   1.11308249
##          RBI        Walks        Years       CAtBat        CHits
##   0.87321953   1.80354477   0.13495750   0.01114084   0.06487319
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.45143958   0.12895336   0.13732672   0.02917546  27.16707164
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -91.60045289   0.19142364   0.04249938  -1.81108944   7.22042163
```

```r
# lasso 모델 적용해보기
# 변수 표준화는 자동으로!
# alpha가 1일때는 lasso
# 기본 값은 k=10(ten folds)
mod.las = glmnet(x[train,],y[train],alpha=1)
set.seed(1)
cv.las = cv.glmnet(x[train,],y[train],alpha=1)
cv.las$lambda.min
```

```
## [1] 16.78016
```

```r
# 과연 이때, test mse가 최소일까?
a = 1:400
b = rep(0,400)
for (i in a){
pred.las = predict(mod.las,s=i,newx=x[test,])
b[i] = mean((pred.las-y[test])^2)
}
plot(a,b,type='l',col='red',xlab='lambda',ylab='Test MSE')
abline(v=a[which.min(b)],col='green')
text(4,94000,'4')
```




```
# 최소의 CV error를 갖는 람다로 전체 데이터를 적합시킨 후, 계수를 살펴보자
final.model = glmnet(x,y,alpha=1)
predict(final.model,type='coefficients',s=17)[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##   20.2953494    0.0000000    1.8672633    0.0000000    0.0000000
##          RBI        Walks        Years       CAtBat        CHits
##    0.0000000    2.2164807    0.0000000    0.0000000    0.0000000
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##    0.0000000    0.2070089    0.4125370    0.0000000    1.2652649
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -103.1150200    0.2201932    0.0000000    0.0000000    0.0000000
```