회 귀 분 석 II

서 울 대 학 교
통 계 학 과

2017년 8월

노트. CH6의 내용중 추후에 다룰 내용

1. 차원축소방법들 즉, 주성분분석, partial linear squares등은 강의의 뒷부분에서 배운다.

최소제곱법 이외의 다른 적합 방법이 필요한 이유

1. 예측의 정확성.
2. 모형의 해석(interpretation).

노트.

1. 앞에서 회귀계수를 추정하는 방법으로 최소제곱추정량을 알아보았다. 이 장에서는 최소제곱추정량외에 다른 추정방법들을 알아본다. 즉 변수의 선택과 축소추정에 대해 알아본다.
2. 예측의 정확성. $f(X)$가 X의 선형함수에 가까우면 LSE는 적은 편향성을 갖고 $n >> p$이면 적은 분산을 갖는다. $p > n$ 이면 최소제곱추정량은 더 이상 유일하지 않고 분산은 ∞가 된다. (이는 분산팽창지수가 ∞가 되므로 알 수 있다.) 축소추정(shrinkage estimation)은 작은 편향을 비용으로 분산을 크게 줄이는 방법이다.
3. 모형의 해석(interpretation). 회귀계수를 정확히 0으로 놓으면 모형에 대한 해석을 쉽게 할 수 있다.
4. **이 장에서 다루는 내용**
   - 4.1 변수선택
   - 4.2 축소추정

최적부분집합선택(best subset selection)

알고리듬

1. M_0을 영모형(null model)이라 하자. 영모형은 예측변수를 하나도 포함하지 않은 모형을 말한다.
2. $k = 1, 2, \ldots, p$
   - 2.1 예측변수가 k개인 $\binom{p}{k}$개의 모형을 고려한다.
   - 2.2 이 중 RSS가 가장 작거나 R^2가 가장 큰 모형을 M_k라고 한다.
3. $M_0, \ldots, M_p$ 중 교차타당성(cross-validation), Cp (AIC), BIC, adjusted R^2 중 하나의 기준을 이용하여 가장 좋은 모형을 선택한다.

노트.

1. GLM의 경우 RSS 대신 편차(deviation)를 쓴다.
2. 최적부분집합 방법은 합리적인 방법이지만 p가 커지면서 계산이 불가능하다. p가 30일 때는 가능하지만, $p = 40$이면 현대의 컴퓨터로 계산이 불가능하다. 참고.
   $2^p \approx 1,000,\ 1,000,000,\ 1,000,000,000,\ 1,000,000,000,000,\ p$가 각각 $10, 20, 30, 40$.
3. branch-and-bound라는 방법이 존재하기는 하지만 p가 커지면서 계속 문제가 생긴다. 이 방법은 정규회귀모형에만 적용된다.

최적부분집합선택 R 코드

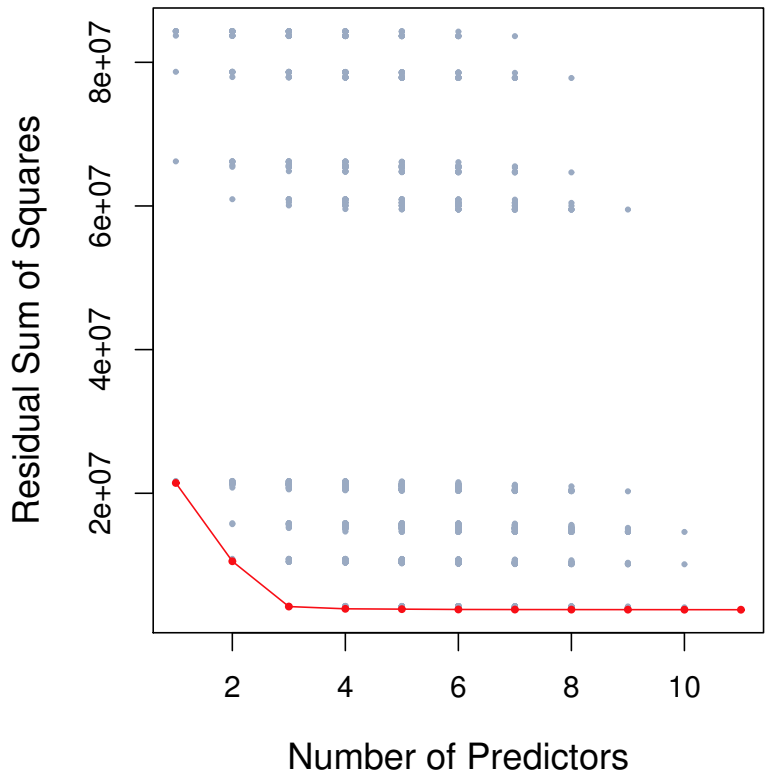
```
> library(leaps)
> regfit.full=regsubsets(Salary~., Hitters)
> summary(regfit.full)
Subset selection object
Call: regsubsets.formula(Salary ~ ., Hitters)
19 Variables  (and intercept)
           Forced in Forced out
AtBat          FALSE      FALSE
Hits           FALSE      FALSE
HmRun          FALSE      FALSE
...

Selection Algorithm: exhaustive
         AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
...
```

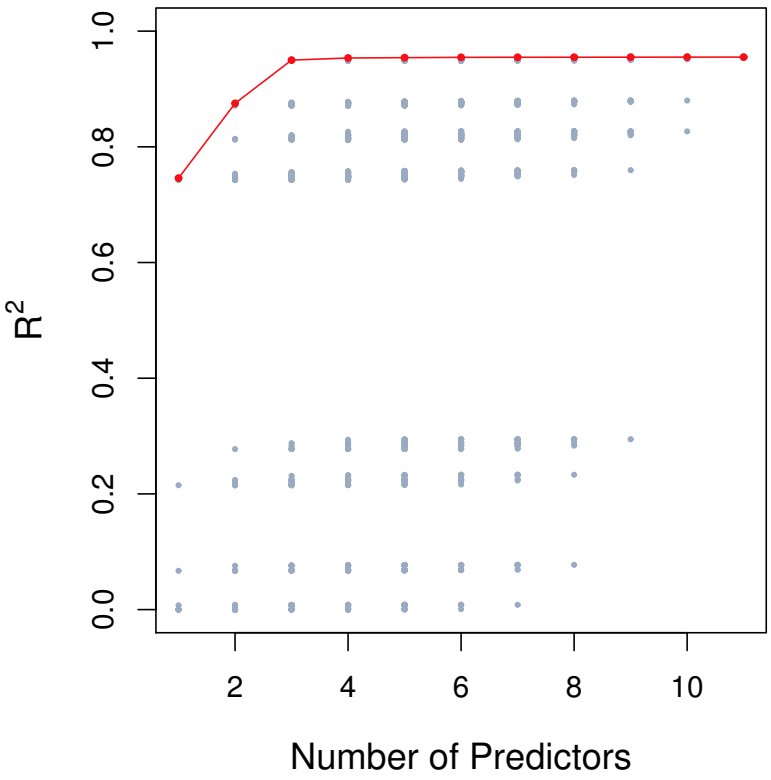
노트.

1. leaps 패키지의 regsubsets 함수가 all subset selection을 수행한다.
2. 변수의 개수 마다 최적의 모형을 준다. 즉 RSS가 가장 작은 모형을 준다.
3. regsubsets의 옵션 중 force.in과 force.out은 받드시 모형에 들어가야하는 혹은 빠져야 하는 변수들의 인덱스를 지정한다.
4. summary에서 모형의 크기가 8개까지인 것만 보여주는데 이것을 바꾸려면 nvmax 옵션을 쓰면 된다.
5. *는 선택된 변수의 표시이다.
6. reg.summary에 변수의 개수에 따른 최적 모형의 R^2, RSS, adjusted R^2, Cp, BIC 등이 있다. 변수의 개수가 동일한 모형들 사이에는 RSS만 비교하면 된다.

```
names(reg.summary)
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"
reg.summary$rsq
```

최적부분집합선택의 예


Residual Sum of Squares
2e+07
4e+07
6e+07
8e+07
2
4
6
8
10
Number of Predictors


R^2
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Number of Predictors

노트.

1. 신용카드 자료에 최적부분집합 방법을 적용하여 예측변수의 수 별로 RSS와 R^2를 그린 그림
2. 이 경우 변수의 개수가 3개인 모형부터 이미 R^2가 1에 가까워진다.

전진선택법

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

알고리듬

Step 1. M_0을 영모형(null model)이라 하자.

Step 2. $k = 0, 1, 2, \ldots, p-1$

- Step 1..1 M_k에 포함이 안된 $p-k$개의 예측변수를 하나씩 M_k에 추가한 $p-k$개의 모형을 고려한다.
- Step 2..2 이 중 RSS가 가장 작거나 R^2가 가장 큰 모형을 M_{k+1}라고 한다.

Step 3. $M_0, \ldots, M_p$ 중 교차타당성(cross-validation), Cp (AIC), BIC, adjusted R^2 중 하나의 기준을 이용하여 가장 좋은 모형을 선택한다.

노트.

1. 그림. 신용카드 자료에 적용된 최적부분집합법과 전진선택법 결과. 변수의 개수가 4일 때 결과가 다르다.
2. $1 + \sum_{k=0}^{p-1}(p-k) = 1 + \frac{p(p+1)}{2}$개의 모형만 고려한다. $p = 20$일 때, 이 값은 211개이다. $2^p = 1,000,000$에 비해 현저히 작은 값이다.
3. 전진선택법에서 선택된 모형과 최적부분집합선택법에서 선택된 모형이 같다는 보장이 없다. 테이블은 신용카드 자료에 적용된 최적부분집합법과 전진선택법 결과이다. 변수의 개수가 4일 때 결과가 다르다. 이를 통해 모든 변수의 개수에 적용해도 결과가 다를 수 있다는 것을 예상할 수 있다.
4. $p > n$일 때도 적용가능하다. 단 $k = n$일 때 까지만 적용가능하다.
5. Step 3에서 BIC나 DIC를 적용하면 베이지안 모형선택의 근사라고 생각할 수 있다.
6. 전진선택법과 후진선택법은 regsubsets 함수에서 method="backward", "forward" 옵션으로 수행할 수 있다. 혼합방법은 R에 step이라는 함수를 이용해서 구현할 수 있다. step은 leaps 패키지에 있는 함수는 아니다.

후진선택법

알고리듬

Step 1. M_p을 full model이라 하자.

Step 2. $k = p, p-1, \ldots, 1$

- Step 1..1 M_k에서 한 개의 변수를 제거한 k개의 모형을 고려한다.
- Step 2..2 RSS가 가장 작거나 R^2가 가장 큰 모형을 M_{k-1}라고 한다.

Step 3. $M_0, \ldots, M_p$ 중 교차타당성(cross-validation), Cp (AIC), BIC, adjusted R^2 중 하나의 기준을 이용하여 가장 좋은 모형을 선택한다.

노트.

1. $1 + \sum_{k=0}^{p-1}(p-k) = 1 + \frac{p(p+1)}{2}$개의 모형만 고려한다.
2. 전진선택법과 마찬가지로 후진선택법도 선택된 모형과 최적부분집합선택법에서 선택된 모형이 같다는 보장이 없다.
3. $p \leq n$일 때 적용가능하다.

혼합방법

알고리듬

Step 1. M_0을 영모형(null model)이라 하자.

Step 2. $k = 0, 1, 2, \ldots, p-1$

- Step 1..1 M_k에 포함이 안된 $p-k$개의 예측변수를 하나씩 M_k에 추가한 $p-k$개의 모형을 고려한다.
- Step 2..2 이 중 RSS가 가장 작거나 R^2가 가장 큰 모형을 M_{k+1}라고 한다.
- Step 3..3 포함된 예측변수 중 기준에 맞지 않는 변수를 제거한다.

Step 3. 거쳐간 모든 모형 중 교차타당성(cross-validation), Cp (AIC), BIC, adjusted R^2 중 하나의 기준을 이용하여 가장 좋은 모형을 선택한다.

시험오차를 추정하는 방법들

변수의 개수가 커지면 무조건 RSS가 작아진다. 그렇다고 시험오차도 같이 작아지는 것은 아니다. 따라서 시험오차의 추정이 필요하다.

1. 훈련오차를 보정하여 시험오차를 간접적으로 추정 : Cp, AIC, BIC, adjusted R^2
2. 교차타당성 방법을 이용하여 시험오차를 직접 추정한다.

시험오차의 간접추정 방법들

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

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

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

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

k : 모형에 포함된 변수의 개수

노트.

1. **Cp**
   - 1.1 위에서 test MSE라는 것은 관측된 자료와 동일한 자료를 또 관측했을 때 생기는 test MSE를 말하는 것일 것이다.
   - 1.2 이 값은 작을수록 좋다.
   - 1.3 $\hat{\sigma}^2$이 σ^2의 불편추정량이라면

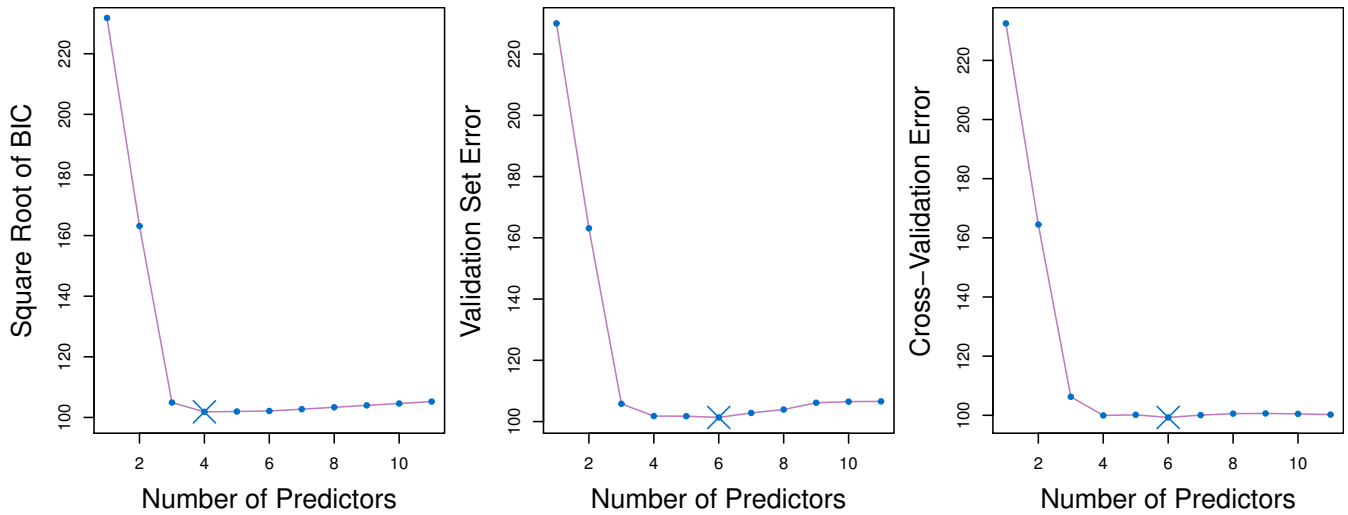
$$\mathbb{E}Cp = \text{ test MSE}$$

   라는 것이 알려져 있다.
2. **AIC**
   - 2.1 이 값은 작을수록 좋다.
   - 2.2 AIC는 기대 쿨백-라이블러-거리 $\mathbb{E}_{f_0} KL(f_0||f)$, 여기서 f_0은 참모형, f는 적합모형)의 점근적 불편향 추정량이다.

노트.

1. **BIC**
   - 1.1 이 값은 작을수록 좋다.
   - 1.2 BIC는 AIC에 비해 변수의 개수가 큰 모형에 많은 페널티를 가한다. 왜냐하면 $n > 7$일 때, $\log n > 2$이다.
2. **adjusted R^2**
   - 2.1 $R^2 = 1 - \frac{RSS}{TSS}$ 이다. k가 커질수록 R^2는 무조건 커진다. 이를 교정하기위해 adjusted R^2를 정의한다.
   - 2.2 여기서 k는 모형에 포함된 변수의 개수이고, n은 관측치의 개수이다.
   - 2.3 이 값은 클수록 좋다.
   - 2.4 근거 adjusted R^2를 최대화하는 것은 $RSS/(n-k-1)$을 최소화하는 것과 같다. 진짜 모형에서 관계없는 noise 변수를 하나 더 추가한다고 하자. RSS는 조금 감소하겠지만 $n-k-1$은 1이 줄어든다. 따라서 $RSS/(n-k-1)$는 오히려 더 커질 것이다.

One-standard error principle



그림에서와 같이 몇 개의 모형이 최소 시험오차와 비슷할 때 최소 시험오차와 1 - standard deviation안에 있는 모형 중 변수의 수가 가장 작은 것을 선택한다.

노트.

1. 그림. 신용카드 자료에 시험오차를 예측변수의 수 별로 추정한 그림
2. **근거.**
   - 2.1 시험오차를 이용한 모형의 선택은 1 - standard deviation 안의 미세한 차이를 구별할 수 있을 만큼 정밀하지 않기 때문이다;
   - 2.2 (Occam's razor) 여러 개의 모형이 비슷한 성능을 보일 때는 변수의 개수가 가장 작은 모형을 선호한다.

최적부분집합선택 R 명령어

1. `library(leaps)`

2. **regsubsets(Salary ., Hitters)**

3. **regfit.fwd=regsubsets(Salary .,data=Hitters,nvmax=19 ,method="forward")**

```
regfit.full=regsubsets(Salary~.,data=Hitters,nvmax=19)
reg.summary=summary(regfit.full)
names(reg.summary)
```

```
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
which.max(reg.summary$adjr2)
```

RSS
2.4e+07
2.6e+07
2.8e+07
3.0e+07
3.2e+07
3.4e+07
3.6e+07
5
10
15
Number of Variables

Adjusted RSq
0.35
0.40
0.45
0.50
5
10
15
Number of Variables

교차검증을 이용한 변수 선택 : R 코드

```
k=10
set.seed(1)
folds=sample(1:k,nrow(Hitters),replace=TRUE)
cv.errors=matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))

for(j in 1:k){
  best.fit=regsubsets(Salary~.,data=Hitters[folds!=j,],nvmax=19)
  for(i in 1:19){
    pred=predict(best.fit,Hitters[folds==j,],id=i)
    cv.errors[j,i]=mean( (Hitters$Salary[folds==j]-pred)^2)
    }
  }
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
plot(mean.cv.errors,type='b')
which.min(mean.cv.errors)
```

노트.

1. k=10 겹 교차검증 방법을 이용한다.
2. folds는 1,2,..., k 중 nrow(Hitters)개를 반복을 허용해서 뽑는 것이다. 전체를 동일한 크기로 나눈 것은 아니다.
3. $k \times 19$ 행렬을 만들어 cv.errors라고 이름을 붙였다. 행은 겹을 나타내고 열은 각 겹에서 변수의 개수를 나타낸다. cv.errors[j,i]는 j 번째 부분을 뺀 자료를 훈련자료로 모형을 적합했을 때, 변수의 개수가 i개인 모형 중 최적의 모형의 시험오차를 넣은 것이다.
4. cv.errors를 계산하고 변수의 개수에 따른 최적의 모형의 시험오차를 구한 값을 계산했다.
5. 시험오차의 그림을 그렸다.
6. 변수의 개수가 11개인 모형이 최적의 모형이다.

능선회귀(ridge regression)

능선회귀 추정량 $\hat{\beta}^R$은

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

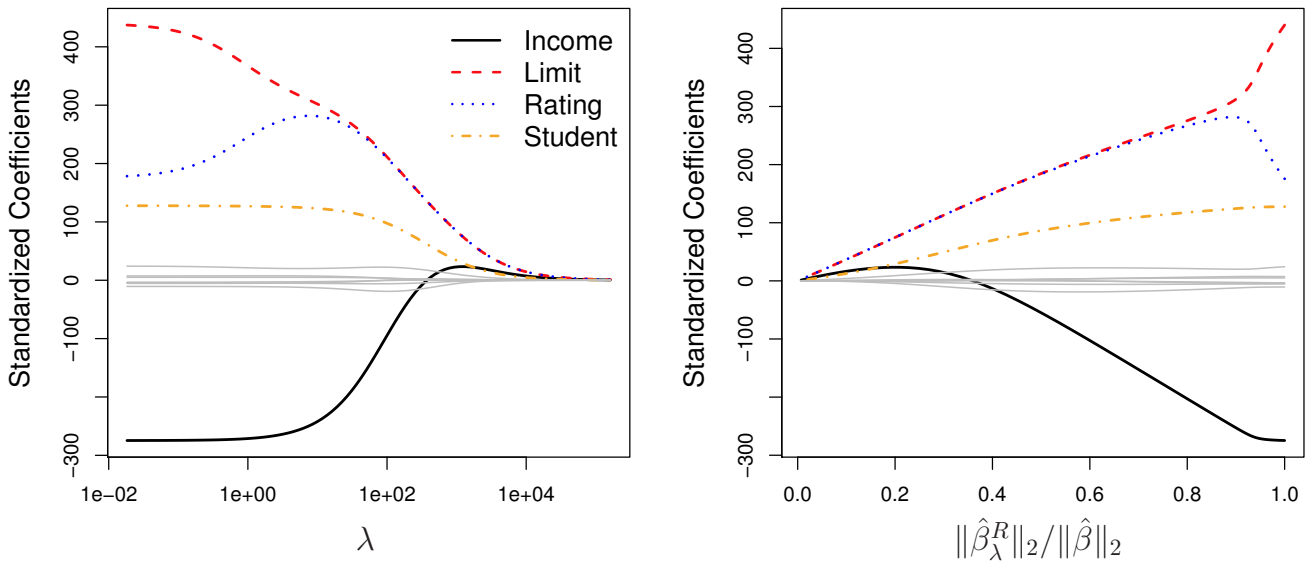
$$= \quad RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

을 최소화하는 β 값으로 정의된다.

노트.

1. λ는 조율파라미터(tuning parameter)로 잔차제곱합과 축소페널티의 상대적 중요성을 정한다.
2. $\lambda \sum_{j=1}^{p} \beta_j^2$는 축소 페널티(shrinkage penalty)라 한다.
3. 첫번째 항 *RSS*는 회귀직선이 자료 근처에 있도록 하고, 축소페널티는 β_j들이 0 근처에 있도록 제한한다.
4. $\lambda \longrightarrow \infty$가 되면 축소페널티의 중요성이 커져서 $\hat{\beta} = 0$이 되고, $\lambda = 0$가 되면 $\hat{\beta}$은 최소제곱추정량과 같아진다.
5. 축소페널티에 β_0는 포함되어있지 않다. x_j의 효과는 제어하고자 하지만 전체 평균인 β_0를 0 근처에 보내고자 하지는 않는다. y의 입장에서 보면 축소가 y의 평균 방향으로 되는 것이다.

λ의 변화에 따른 추정량의 변화



노트.

1. 그림. λ 조율파라미터의 변화에 따른 능선회귀계수의 추정량의 변화
2. 왼쪽 그림은 x축에 λ가 0으로부터 ∞까지 변한다. 각 곡선은 한 개의 회귀계수 추정량의 변화를 보여준다. 맨 왼쪽은 최소제곱추정량의 값이고, 맨 오른쪽은 모두 0으로 수렴한다. 오른쪽 그림은 능선회귀추정량의 축소된 정도를 나타내는 $\frac{||\hat{\beta}_\lambda^R||_2}{||\hat{\beta}||_2}$이 0에서 1로 움직이며 회귀계수 추정량의 변화를 보여준다. 맨왼쪽 0부터 오른쪽으로 가며 최소제곱추정량으로 수렴한다.

변수의 표준화

최소제곱추정량은 척도등변추정량(scale equivariant estimator)이다.

$$credit = \beta_0 + \beta_1 \times income + \epsilon$$

의 모형을 생각할 때 income을 천불단위로 하든 1불 단위로 하든 credit의 예측에는 변화가 없다.

능선회귀추정량은 척도등변추정량이 아니다. 단위에 따라 $\hat{credit}$값이 달라질 수 있다.

이와같은 이유로 능선회귀를 적용할 때는 모든 변수를 표준화시켜 같은 척도를 갖기를 추천한다.

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

노트.

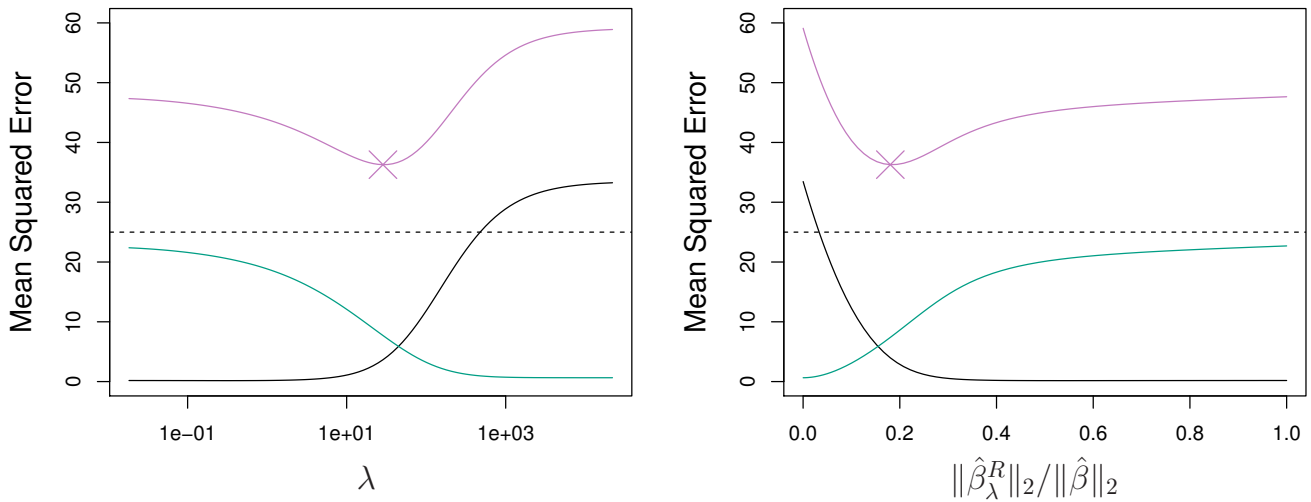
1. **최소제곱추정량은 척도등변추정량(scale equivariant estimator)이다.**

$$credit = \beta_0 + \beta_1 \times income + \epsilon$$

의 모형을 생각할 때 income을 천불단위로 하든 1불 단위로 하든 credit의 예측에는 변화가 없다. income이 1000배 커지면 $\hat{\beta}$은 1/1000로 줄어들어 $\beta_1 \times income$의 값에는 변화가 없게된다.

2. 능선회귀추정량은 척도등변추정량이 아니다. income을 천불단위로 하느냐, 1불단위로 하느냐에 따라 $\hat{credit}$값이 달라질 수 있다. 이와같은 이유로 능선회귀를 적용할 때는 모든 변수를 표준화시켜 같은 척도를 갖기를 추천한다.

능선회귀의 성능이 최소제곱법 보다 좋은 이유



λ가 커지면서 편향의 제곱(검은색)은 커지면서 분산(녹색)은 작아진다.
최소제곱오차(붉은색)은 작아지다 커진다.

노트.

1. 그림. λ의 변화에 따른 회귀계수의 편향, 분산, 최소제곱오차의 변화
2. 그림을 보면, λ가 커지면서 편향의 제곱(검은색)은 커지면서 분산(녹색)은 작아진다. 최소제곱오차(붉은색)은 작아지다 커진다.
3. 능선회귀의 성능이 최소제곱법 보다 좋은 이유는 편향-분산 균형(bias-variance trade-off)에 있다. λ가 커지면서 편향은 커지는 반면 분산은 작아진다. λ를 적당히 잘 조절하면 능선회귀의 평균제곱오차는 최소제곱법의 평균제곱오차보다 작아질 수 있다.

4. 능선회귀는 최소제곱추정량의 분산이 큰 경우(예. $p > n$인 경우) 성능이 좋다.
5. 계산이 빠르다. 모든 값의 λ의 능선회귀추정량을 구하는 계산량과 최소제곱추정량을 구하는 계산량이 거의 같다.

능선회귀 R 코드

```
x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary

library(glmnet)
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)

predict(ridge.mod,s=50,type="coefficients")[1:20,]

set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
```

노트.

1. glmnet 패키지의 glmnet() 함수를 쓰는데, 이 함수는 모형식을 받아들이지 않고 설명변수와 반응변수를 행렬과 벡터로 대입해야 한다. model.matrix는 가변수도 자동적으로 생성해준다. 절편항은 디자인행렬에서 삭제했다.
2. glmnet은 GLM 모형에서 벌점가능도 추정치를 구하는 함수이다. y가 연속형 변수일 경우 옵션 family $=$ gaussian이 디폴트이다. 벌점은

$$\frac{1-\alpha}{2}||\beta||_2^2 + \alpha||\beta||_1$$

와 같이 정의된다. alpha는 elastic net mixing 파라미터로 alpha $= 0$이면 능선회귀를 나타낸다. 벌점가능도는 가우시안 모형의 경우

$$\frac{1}{2}RSS/n + \lambda penalty$$

이고 다른 모형의 경우

$$-log - likelihood + \lambda penalty$$

이다. 옵션의 lambda는 위의 λ를 정할 때 쓰도록 주는 그리드 값이다. 위에서 lambda는 10^{10}에서 10^{-2}까지 변하고, 100개의 값을 갖는다. glmnet은 변수를 자동적으로 표준화한다.

노트.

1. predict함수는 여러 가지 목적으로 사용될 수 있다. s는 예측값을 구하는 lambda의 값을 지정하는 옵션이다. type="response"인 경우는 예측값(gaussian)이나 예측확률(binomial) 등을 준다. type="coefficient"인 경우는 추정된 회귀계수 값을 준다. 여기서는 lambda = 50에서의 회귀계수의 추정치를 리턴한다.
2. cv.glmnet은 k 겹 교차검증을 수행하는 함수이다. 옵션 nfold는 겹의 수 k이고, 디폴트 값은 10이다. 최적의 lambda 값은 약 212이다.

라쏘(lasso)

라쏘 추정량은 변수의 회귀계수가 작을 경우 그 값을 정확히 0으로 놓는 성질이 있어, 변수선택의 효과가 있다.

즉, 라쏘는 변수선택과 축소추정을 동시에 한다.

라쏘추정량 $\hat{\beta}_\lambda^L$는

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

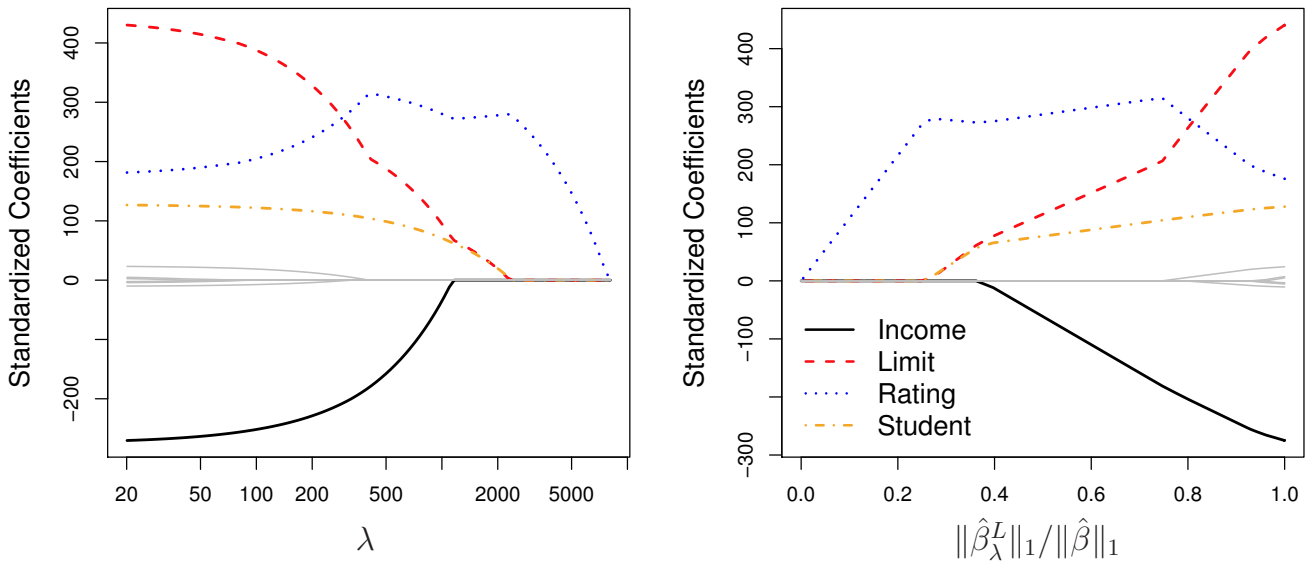
$$= \quad RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

을 최소화하는 β 값으로 정의된다.

노트.

1. 라쏘 역시 능선회귀와 마찬가지로 0으로 회귀계수를 축소시킨다.
2. λ가 충분히 크면 몇 개의 계수를 정확히 0으로 만든다. 변수 선택의 효과가 있다. 이를 라쏘는 희박한 모형(sparse model)을 만들어 낸다고 말한다.

λ의 변화에 따른 추정량의 변화



노트.

1. 그림. λ 조율파라미터의 변화에 따른 라쏘회귀계수의 추정량의 변화
2. λ가 0의 방향으로 움직이면서 먼저 rating만 포함한 모형을 생성한다.

능선회귀와 라쏘의 또다른 구체화

능선회귀의 또다른 구체화

능선회귀추정량 $\hat{\beta}_\lambda^R$는 적당한 s에 대해

$$\sum_{j=1}^{p} \beta_j^2 \leq s \text{ 조건하에서}$$

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

를 최소화하는 β와 같다.

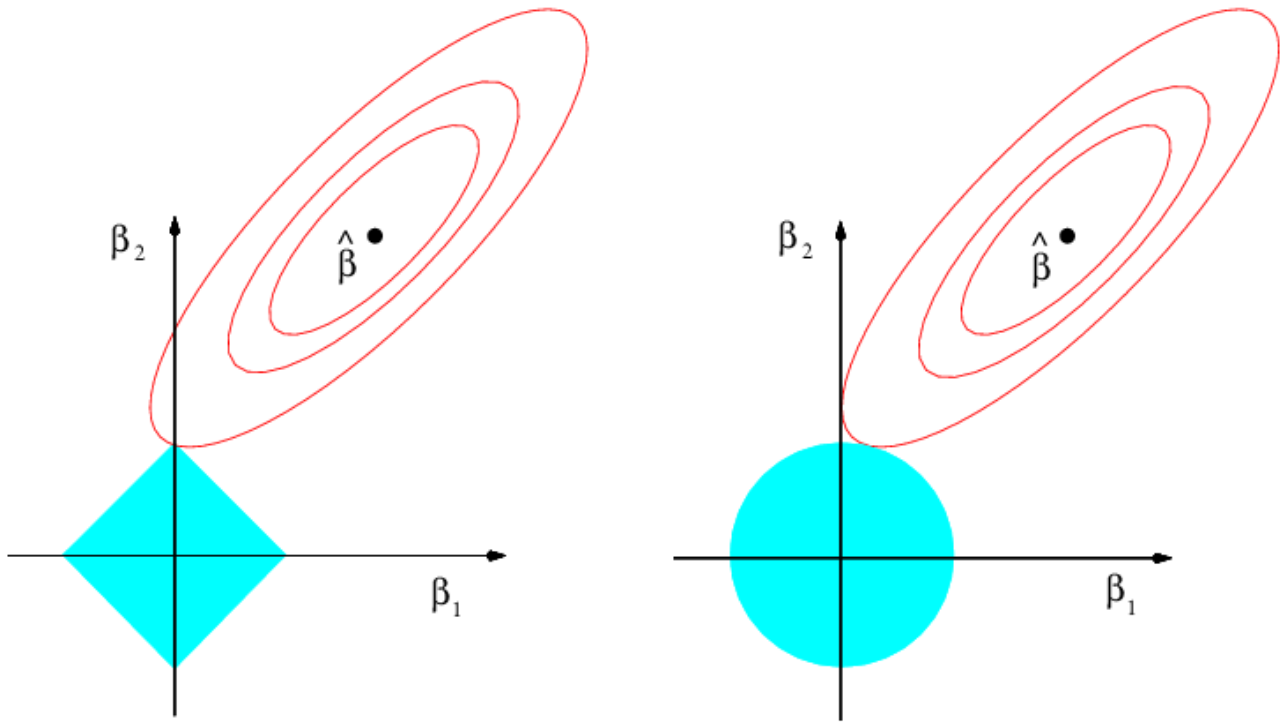
라쏘의 또다른 구체화

라쏘회귀추정량 $\hat{\beta}_\lambda^L$는 적당한 s에 대해

$$\sum_{j=1}^{p} |\beta_j| \leq s \text{ 조건하에서}$$

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

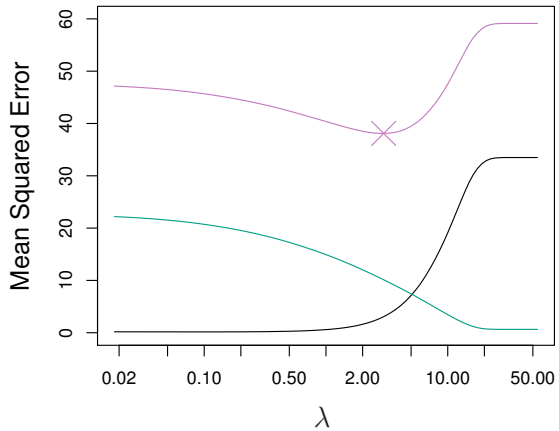
왜 라쏘 회귀계수추정치는 정확히 0이 되는가?

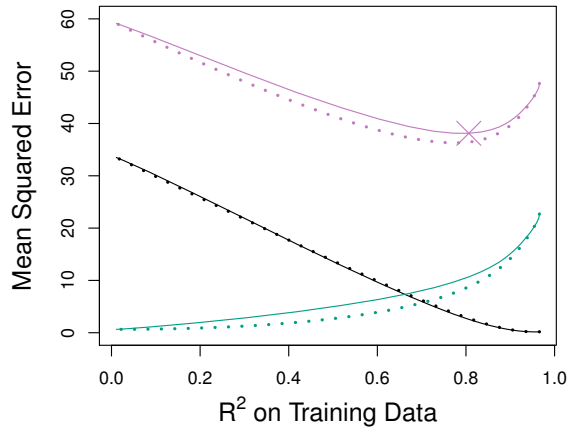


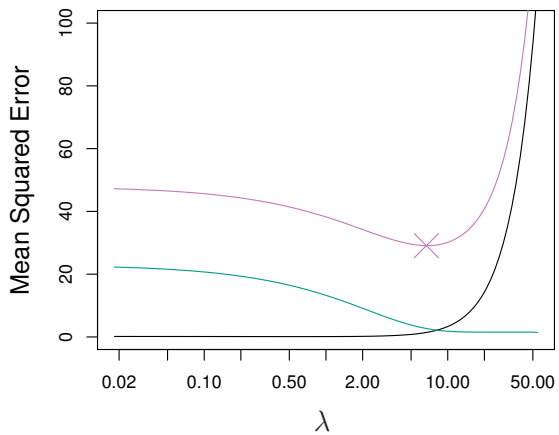
노트.

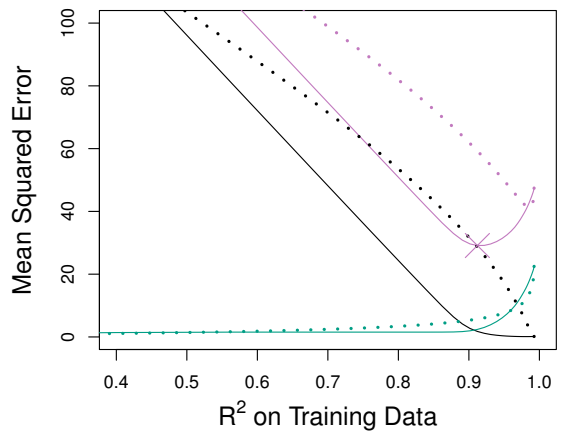
1. 그림. 라쏘회귀계수추정량과 제약의 관계
2. 라쏘는 $\sum_{j=1}^{p} |\beta_j| \leq s$인 곳에서 RSS를 최소화해야 한다. 녹색이 그 지역을 나타낸다. 빨간색 등고선은 RSS 표면을 나타낸다.
   녹색지역에서 RSS가 가장 작은 곳은 꼭지점이 된다. 능선회귀는 이와 달리 $\sum_{j=1}^{p} \beta_j^2 \leq s$인 곳에서 RSS를 최소화한다. 따라서 RSS가 가장 작은 곳이 꼭지점이 되지는 않는다.

라쏘와 능선회귀 중 항상 더 좋은 것은 없다.









노트.

1. 위 그림에서는 능선회귀의 시험평균제곱오차가 라쏘의 그것보다 조금 더 작다.
2. **위 그림.**
   - 2.1 왼쪽 그림 : 모의실험자료를 통해 라쏘의 편향의 제곱(검은색), 분산(녹색), 시험평균제곱오차(보라)를 조율파라미터의 값(x축)에 관해 그렸다.
   - 2.2 오른쪽 그림 : 실선은 왼쪽그림과 동일한 그림이다. 즉 실선은 라쏘의 편향의 제곱, 분산, 시험제곱오차를 그린 것이다. 단지, x축을 조율파라미터 대신 훈련자료의 R^2에 관해 그렸다. 점선은 능선회귀의 동일한 그림이다. 능선회귀의 시험평균제곱오차가 라쏘의 그것보다 조금 더 작다.
3. 아래 그림에서는 능선회귀의 시험평균제곱오차가 라쏘의 그것보다 조금 더 작다.
4. **아래 그림.**
   - 4.1 왼쪽 그림 : 모의실험자료를 통해 라쏘의 편향의 제곱(검은색), 분산(녹색), 시험평균제곱오차(보라)를 조율파라미터의 값에 관해 그렸다.
   - 4.2 오른쪽 그림 : 실선은 왼쪽그림과 동일한 그림이다. 즉 실선은 라쏘의 편향의 제곱, 분산, 시험제곱오차를 그린 것이다. x축을 조율파라미터 대신 훈련자료의 R^2에 관해 그렸다. 점선은 능선회귀의 동일한 그림이다. 능선회귀의 시험평균제곱오차가 라쏘의 그것보다 조금 더 크다.

축소의 형태

$X = I_p$, $n = p$, $\beta_0 = 0$인 경우

최소제곱추정량

$$RSS = \sum_{j=1}^{p}(y_j - \beta_j)^2$$

$$\hat{\beta}_j = y_j,\ j = 1, 2, \ldots, p.$$

능선회귀

$$\sum_{j=1}^{p}(y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$\hat{\beta}_j^R = \frac{y_j}{1+\lambda},\ j = 1, 2, \ldots, p$$

라쏘

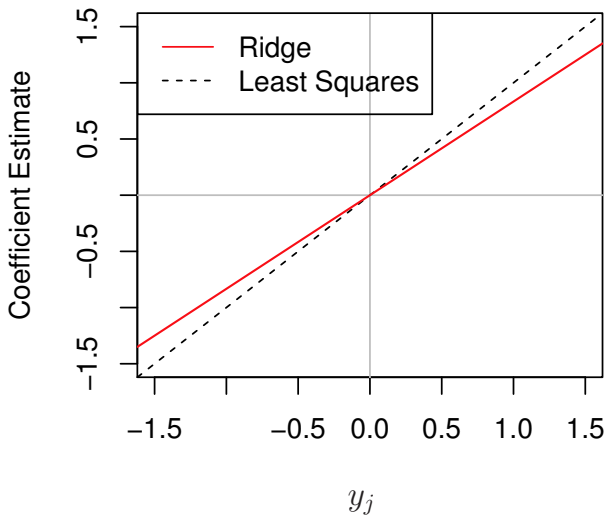
$$\sum_{j=1}^{p}(y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

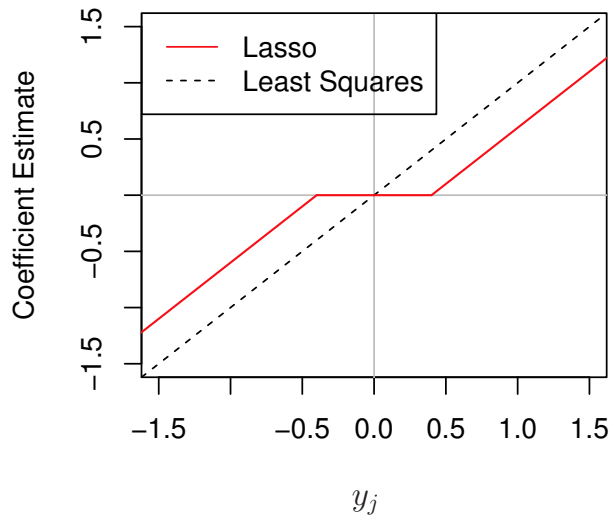
$$\hat{\beta}_j^R = \begin{cases} y_j - \lambda/2, & y_j > \lambda/2 \\ y_j + \lambda/2, & y_j < -\lambda/2 \\ 0, & |y_j|\lambda/2 \end{cases} \quad j = 1, 2$$

노트.

1. 각 추정 방법에 따라 최소화해야 하는 것과 추정량의 식이 주어져 있다.

축소의 형태


Ridge
Least Squares
Coefficient Estimate
−1.5
−0.5
0.5
1.5
−1.5
−0.5
0.0
0.5
1.0
1.5
y_j


Lasso
Least Squares
Coefficient Estimate
−1.5
−0.5
0.5
1.5
−1.5
−0.5
0.0
0.5
1.0
1.5
y_j

노트.

1. 그림은 라쏘와 능선회귀의 축소방식이 매우 다르다는 것을 보여준다.

능선회귀와 라쏘의 베이지안 해석

β의 사전분포가

$$\pi(\beta) \propto e^{-\frac{\lambda}{\sigma^2}\sum_{j=1}^{p}\beta_j^2}$$

인 경우, 사후분포는

$$\pi(\beta|y) \propto e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i-(\beta_0+\sum_{j=1}^{p}\beta_j x_{ij})\right)^2} \times e^{-\frac{\lambda}{\sigma^2}\sum_{j=1}^{p}\beta_j^2}$$

가 된다. β의 최대사후분포추정량(MAP)이 능선회귀추정량이 된다.
β의 사전분포가

$$\pi(\beta) \propto e^{-\frac{\lambda}{\sigma^2}\sum_{j=1}^{p}|\beta_j|}$$

인 경우, 사후분포는

$$\pi(\beta|y) \propto e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i-(\beta_0+\sum_{j=1}^{p}\beta_j x_{ij})\right)^2} \times e^{-\frac{\lambda}{\sigma^2}\sum_{j=1}^{p}|\beta_j|}$$

가 된다. β의 최대사후분포추정량(MAP)이 라쏘추정량이 된다.

노트.

1. **조율파라미터 λ의 추정** 교차검증(cross-validation)을 이용해 선택한다. λ를 격자로 나누어 이 값들에서 교차검증을 수행해서 시험오차를 구한다.

라쏘 R 코드

```
lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
```

노트.

1. alpha=1이면 라쏘를 적합하는 것이다.

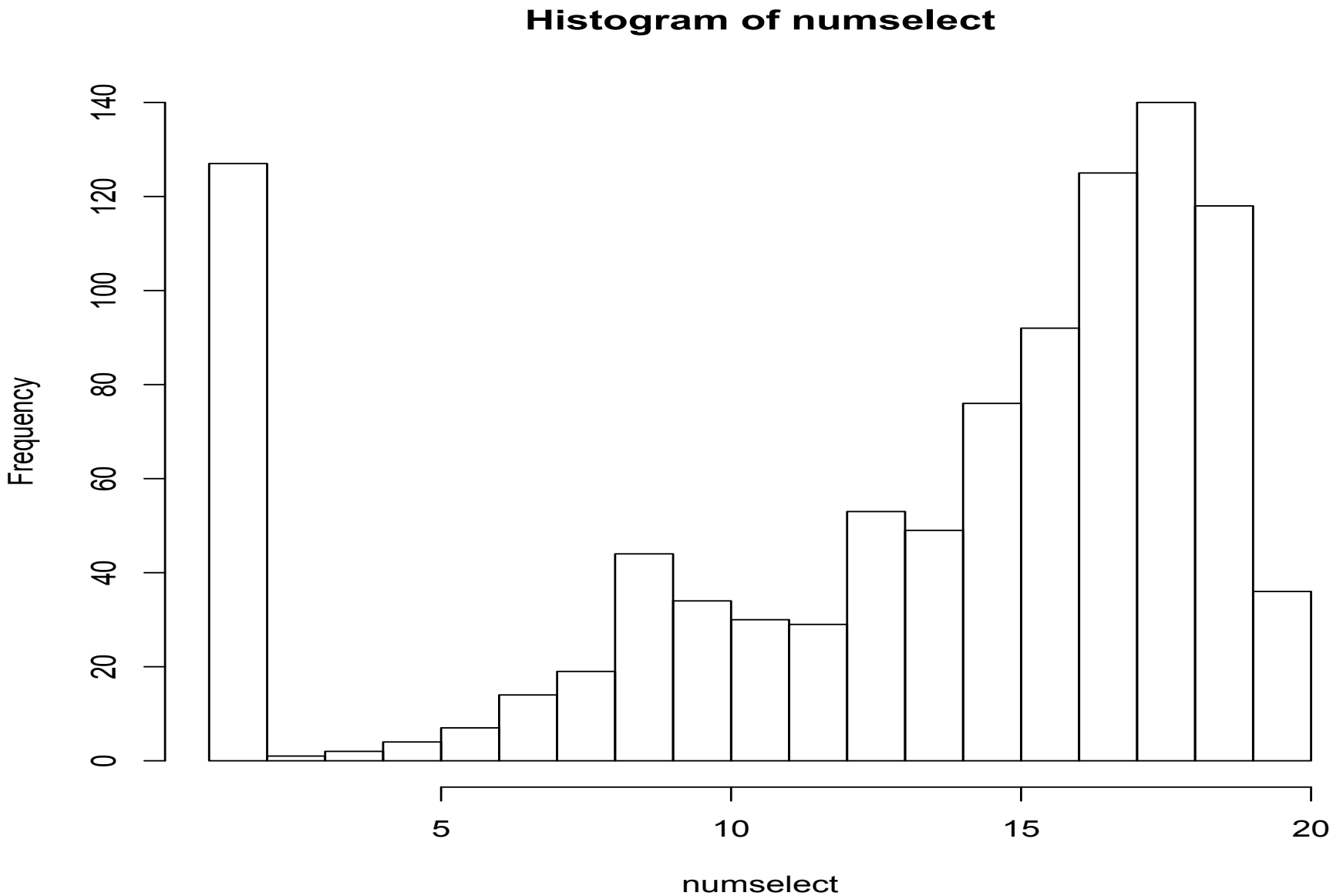
라소에서 유의성 검정:붓스트랩

```
library(glmnet)
library(ISLR)
Hitters=na.omit(Hitters)
dim(Hitters)

attach(Hitters)
x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary

#
# dimension of x 263 X 19
#

n=263
B=1000
best=matrix(0,B,20)
```

```
for(b in (1:B)){

    bid=sample(n,n,replace=T)
    bx=x[bid,]
    by=y[bid]
    grid=10^seq(4,-1,length=100)
    cv.out=cv.glmnet(bx,by,alpha=1,lambda=grid)
    blamb=cv.out$lambda.min
    lasso.mod=glmnet(bx,by,alpha=1,lambda=exp(blamb))
    best[b,]=as.vector(coef(lasso.mod))

    cat("\t b=")
    cat(b)
}
```

```
lasso.mod=glmnet(x,y,alpha=1,lambda=exp(blamb))
est=as.vector(coef(lasso.mod))
se=sqrt(apply(best,2,var))
tstat=est/se
tstat

pvalue=2*(1-pnorm(abs(tstat)))
pvalue


select=(best!=0)
stab=apply(select,2,sum)/B
numselect=apply(select,1,sum)
hist(numselect)
```

선택변수


Histogram of numselect
Frequency
0
20
40
60
80
100
120
140
5
10
15
20
numselect

t-통계량, p-value, stability

```
>
> tstat
 [1]  0.4700554 -1.2941157  1.6552804  0.0000000  0.0000000
 [6]  0.0000000  1.7561824 -0.8324965  0.0000000  0.0000000
[11]  0.3713004  1.0455094  1.1459485 -1.0241761  0.6484224
[16] -2.2594369  1.9907399  0.4075420 -0.4004371  0.0000000
>
> pvalue
 [1] 0.63831546 0.19562546 0.09786763 1.00000000 1.00000000
 [6] 1.00000000 0.07905726 0.40512877 1.00000000 1.00000000
[11] 0.71041382 0.29578760 0.25181646 0.30575210 0.51671177
[16] 0.02385622 0.04650949 0.68360996 0.68883459 1.00000000
>
>
> stab
 [1] 1.000 0.703 0.859 0.585 0.520 0.544 0.854 0.622 0.360 0.454
[11] 0.613 0.763 0.607 0.658 0.676 0.855 0.859 0.691 0.669 0.568
>
```

참고문헌

아래의 책에서 제공하는 그림들을 사용하였다.

1. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Springer, 2013.