6. Regularization

- 6.1 LASSO
- 6.2 Non-normal y

6.1 LASSO

- Linear regression 모형

$$y_i = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i} + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2) \quad (i = 1, 2, \ldots, n)$$

또는

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}_n, \sigma^2 I_n)$$

- $\boldsymbol{\beta}$의 최소제곱추정량

L_2 norm

$$\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \mathbf{y}$$

  - 의한
  - 제곱오차 $\|\mathbf{y} - X\boldsymbol{\beta}\|_2^2$ 를 최소로 하는 방법 $\|y - X\beta\|_{L_2}^2$
  - 이론적으로 가장 우수한 방법임: BLUE (best linear unbiased estimator) 성질
- 고차원 자료 분석 시 모형이 매우 복잡해져서 결과적으로 분산이 매우 커지는 문제가 있음
  - p가 매우 큰 경우 최소제곱법으로 모형 적합 불가능
  - 즉, $X^\top X$의 역행렬이 존재하지 않음

λ 값이 커지면 unbiased 는 깨짐 하지만 분산이 줄어듦

- Ridge regression (능형 회귀)

$|\beta_i|$ 이 점점 작아짐

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

  - 학습 과정에서 추정치를 shrink시켜 약간의 bias를 감내하고 분산을 줄이는 아이디어
  - 최소제곱법에서 사용하는 제곱오차에 모형의 복잡도에 대해 벌점을 추가 부여한 손실함수를 최소로 하는 solution을 찾는 것과 동치임. 즉,

goodness of fit

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \{\|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \underbrace{\lambda\|\boldsymbol{\beta}\|_2^2}_{\text{벌점}}\}$$

- LASSO (least absolute shrinkage and selection operator)
  - 위의 ridge 회귀의 벌점 부분을 절대값 노옴($\mathcal{L}_1$)으로 대체

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \{\|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1\}$$

중요치 않은 변수는 $\beta \to 0$ 으로 한다. 즉 중요변수를 선택한다.

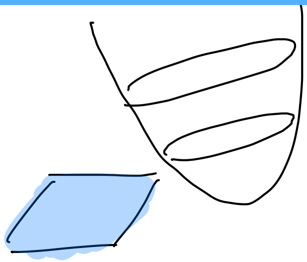
  - 이 손실함수는 $\boldsymbol{\beta}$에 대해 convex
  - λ의 선택
    - 교차확인법(crossvalidation) : 여러 train, test 번갈아서 시행 Cross validation 을 최소로 하는 λ 찾기
    - 베이지안 방법
  - 이 최적화 문제는 다음 문제와 동치

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2$$

subject to

$$\|\boldsymbol{\beta}\|_1 \le R$$

for $R > 0$



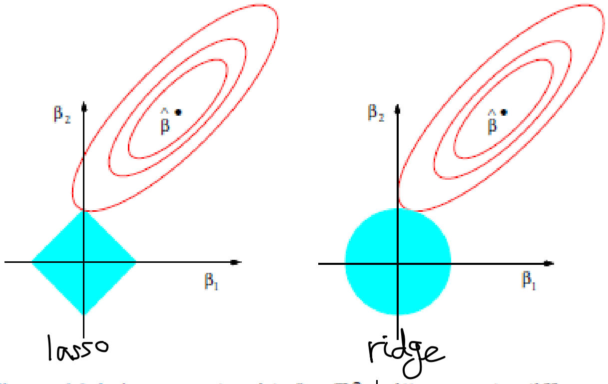


Figure 16.4 An example with $\beta \in \mathbb{R}^2$ to illustrate the difference between ridge regression and the lasso. In both plots, the red contours correspond to the squared-error loss function, with the unrestricted least-squares estimate $\hat{\beta}$ in the center. The blue regions show the constraints, with the lasso on the left and ridge on the right. The solution to the constrained problem corresponds to the value of β where the expanding loss contours first touch the constraint region. Due to the shape of the lasso constraint, this will often be at a corner (or an edge more generally), as here, which means in this case that the minimizing β has $\beta_1 = 0$. For the ridge constraint, this is unlikely to happen.

Example: 모의실험 데이터

Data generating process

- Variables: $(y, x_1, x_2, \ldots, x_{100})$
- DGP: $y = 10 + 1 \times x_1 + 2 \times x_2 + \ldots + 5 \times x_5 + \epsilon, \epsilon \sim N(0, \cancel{0.15}^2)$. 즉 $x_6 \sim x_{100}$은 y값과 무관.
  6.25^2
- 표본 크기 $n = 50$. 데이터 차원에 비해 매우 작음.

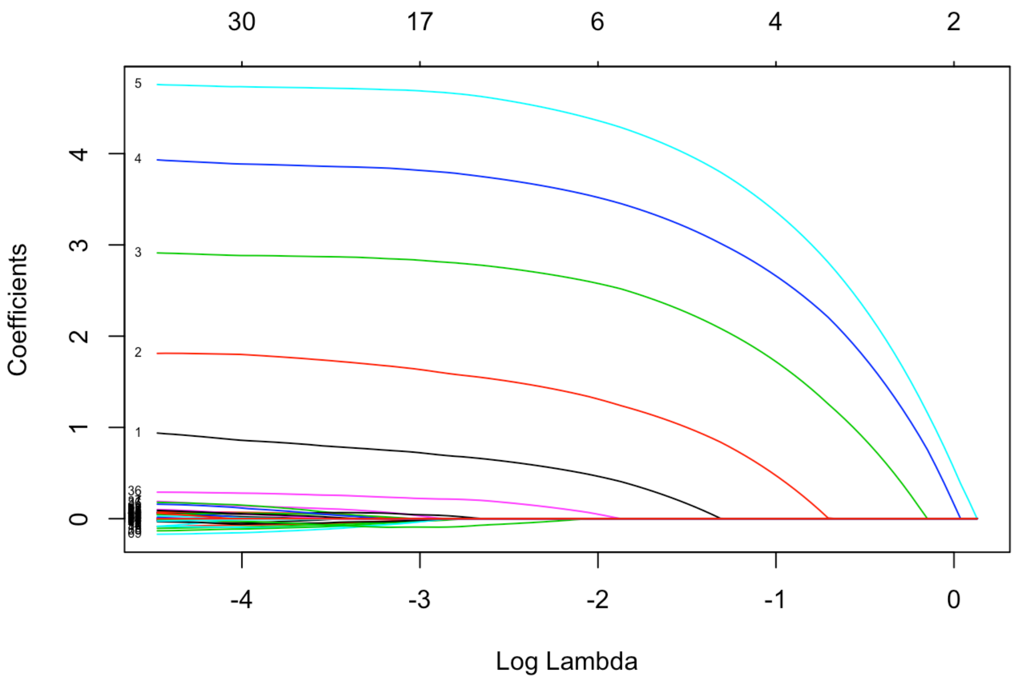
```
library(glmnet)
```

```
## Loading required package: Matrix
```

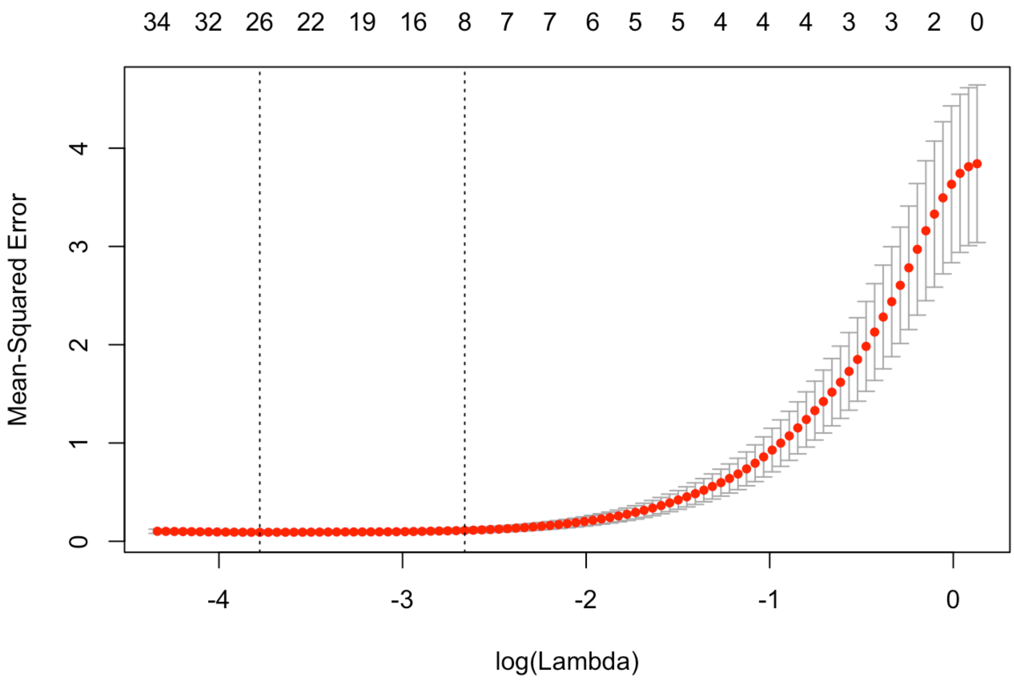
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
set.seed(0)
n <- 50
p <- 100

sigma <- 0.25
beta <- c(1:5, rep(0, p - 5))

x <- cbind(matrix(runif(n*p), n, p))
y <- 10 + as.vector(x%*%beta) + sigma*rnorm(n)

fit <- glmnet(x, y)
plot(fit, xvar = "lambda", label = TRUE)
```




```
cvfit <- cv.glmnet(x, y)
plot(cvfit)
```




```
print(s <- cvfit$lambda.min)
```

```
## [1] 0.02287961
```

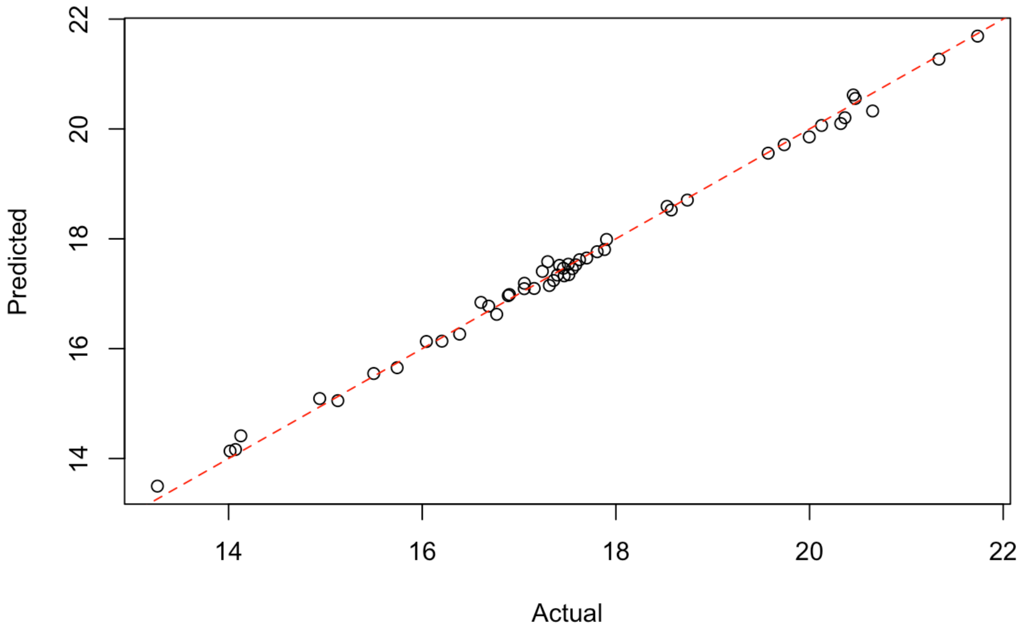
```
print(beta.hat <- coef(cvfit, s = "lambda.min"))
```

```
## 101 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 10.370743731
## V1           0.833559111
## V2           1.771945018
## V3           2.879022777
## V4           3.875763697
## V5           4.726019401
## V6           .
## V7           0.126481344
## V8           .
## V9           0.011463797
## V10         -0.062694483
## V11          .
## V12          .
## V13          .
## V14          .
## V15         -0.079682170
## V16          .
## V17         -0.095697482
## V18         -0.002282135
## V19          .
## V20          .
## V21          .
## V22          .
## V23          .
## V24          .
## V25          .
## V26         -0.031592886
## V27          .
## V28          .
## V29          .
## V30          .
## V31          0.118216703
## V32          .
## V33          .
## V34          .
## V35          .
## V36          0.271640977
## V37          .
## V38          .
## V39          .
## V40          .
## V41          .
## V42          .
## V43          .
## V44          .
## V45          .
## V46          .
## V47          .
## V48          .
## V49          .
## V50          .
## V51          .
## V52          .
## V53          .
## V54          .
## V55         -0.101672366
## V56          .
## V57          .
## V58          .
```

0값으로 추정

```
## V59         0.004125394
## V60          .
## V61         0.056265212
## V62          .
## V63          .
## V64          .
## V65         0.061994724
## V66         0.055262226
## V67          .
## V68          .
## V69        -0.137083598
## V70          .
## V71          .
## V72          .
## V73        -0.053131790
## V74          .
## V75          .
## V76          .
## V77          .
## V78          .
## V79          .
## V80          .
## V81          .
## V82          .
## V83          .
## V84        -0.044687164
## V85          .
## V86         0.087163948
## V87        -0.010053856
## V88          .
## V89          .
## V90          .
## V91          .
## V92         0.032086787
## V93          .
## V94          .
## V95         0.038901740
## V96          .
## V97          .
## V98          .
## V99          .
## V100         .
```

```
pred.lasso <- predict(fit, newx = x, s = s)
plot(y, pred.lasso, xlab = "Actual", ylab = "Predicted")
abline(0, 1, col = 2, lty = 2)
```



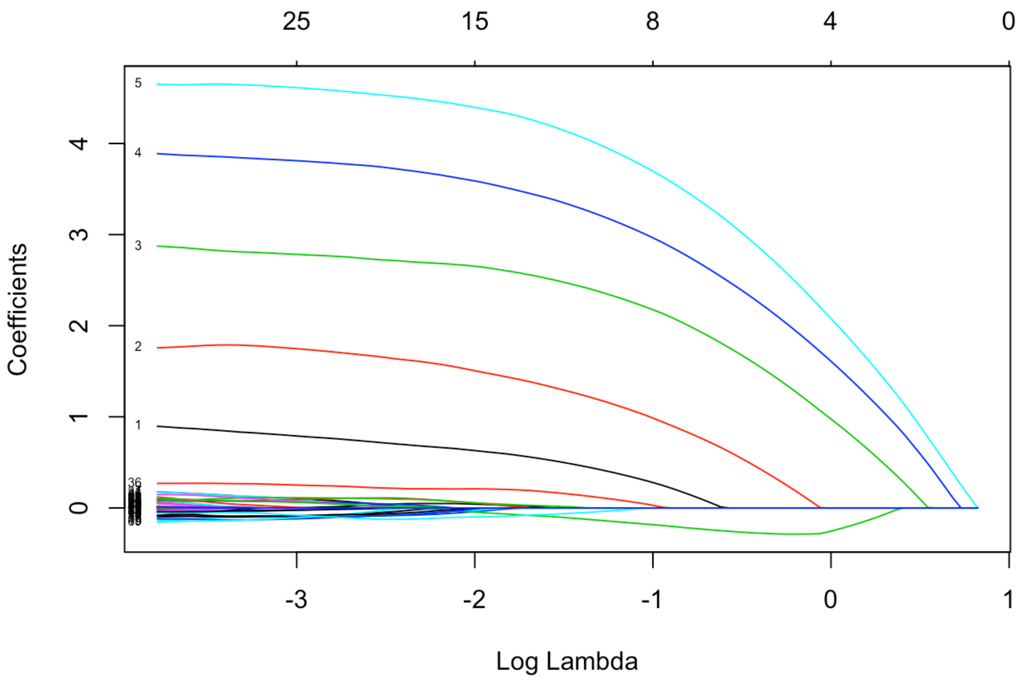
Elastic-net

- 벌점 부분에 $\mathcal{L}_1$-norm과 $\mathcal{L}_2$-norm을 함께 사용

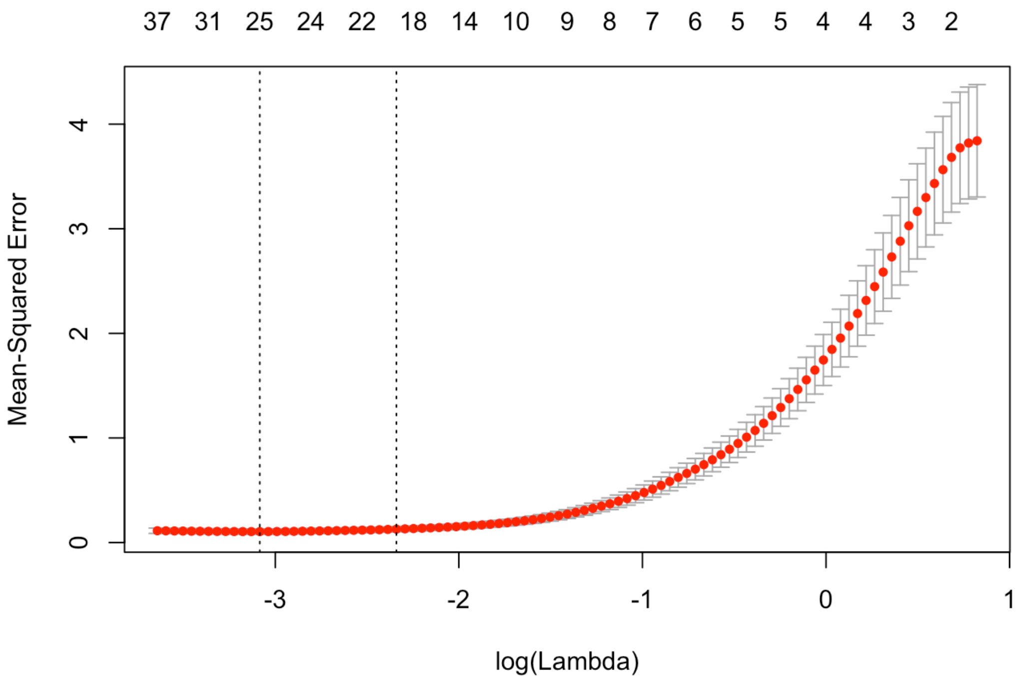
$$\frac{1-\alpha}{2}\|\boldsymbol{\beta}\|_2^2 + \alpha\|\boldsymbol{\beta}\|_1$$

- $\alpha = 0$이면 ridge regression, $\alpha = 1$이면 lasso regression에 해당

```
fit <- glmnet(x, y, alpha = .5)
plot(fit, xvar = "lambda", label = TRUE)
```




```
cvfit <- cv.glmnet(x, y, alpha = .5)
plot(cvfit)
```



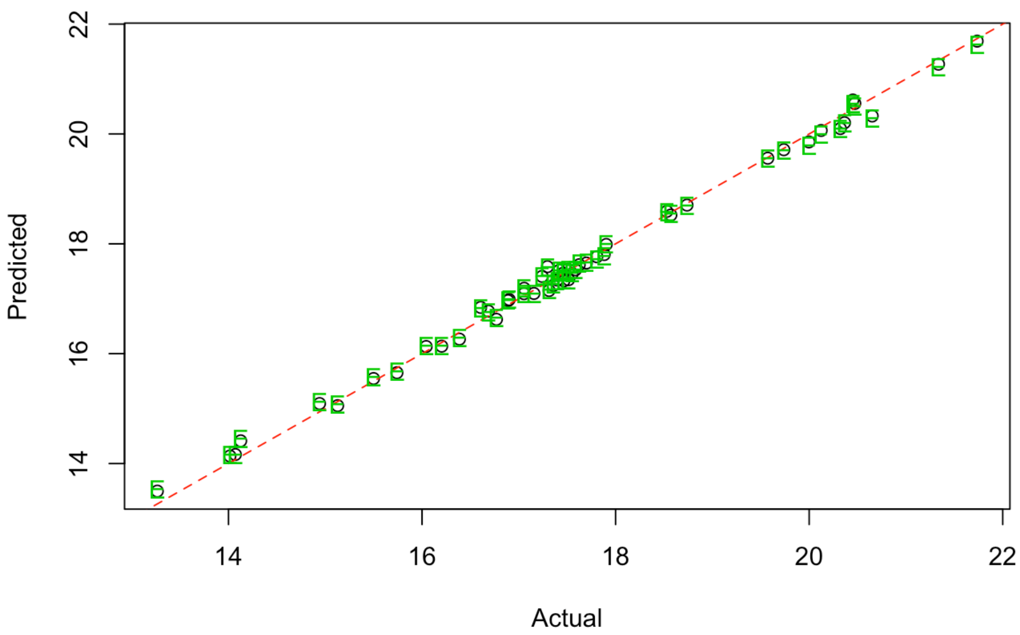

```
print(cvfit$lambda.min)
```

```
## [1] 0.04575923
```

```
print(s <- cvfit$lambda.min)
```

```
## [1] 0.04575923
```

```
plot(y, pred.lasso, xlab = "Actual", ylab = "Predicted")
abline(0, 1, col = 2, lty = 2)
points(y, predict(fit, newx = x, s = s), col = 3, pch = "E")
```



6.2 Non-normal y

- GLM 세팅에도 활용 가능
    - 반응변수 y가 연속형이 아니라 정규분포 가정이 불가능한 경우로 확장하는 모형 generalized linear model
        - Binary data: logistic regression
        - Count data: Poisson regression
    - 위 선형회귀모형과 달리 제곱오차가 아닌 우도함수값을 기준으로 모수 추정
    - Feature의 차원 p가 큰 경우 동일한 문제 발생

```
data(BinomialExample)

fit <- glm(y ~ x, family = binomial("logit"))
```

```
## Warning: glm.fit: algorithm did not converge
```

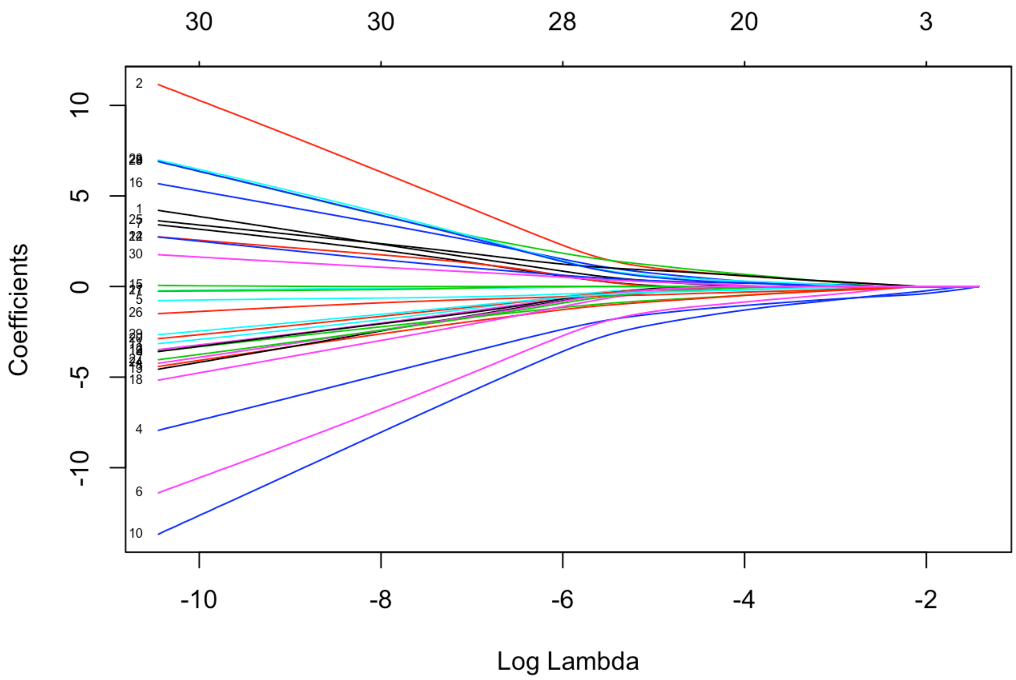
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit)
```

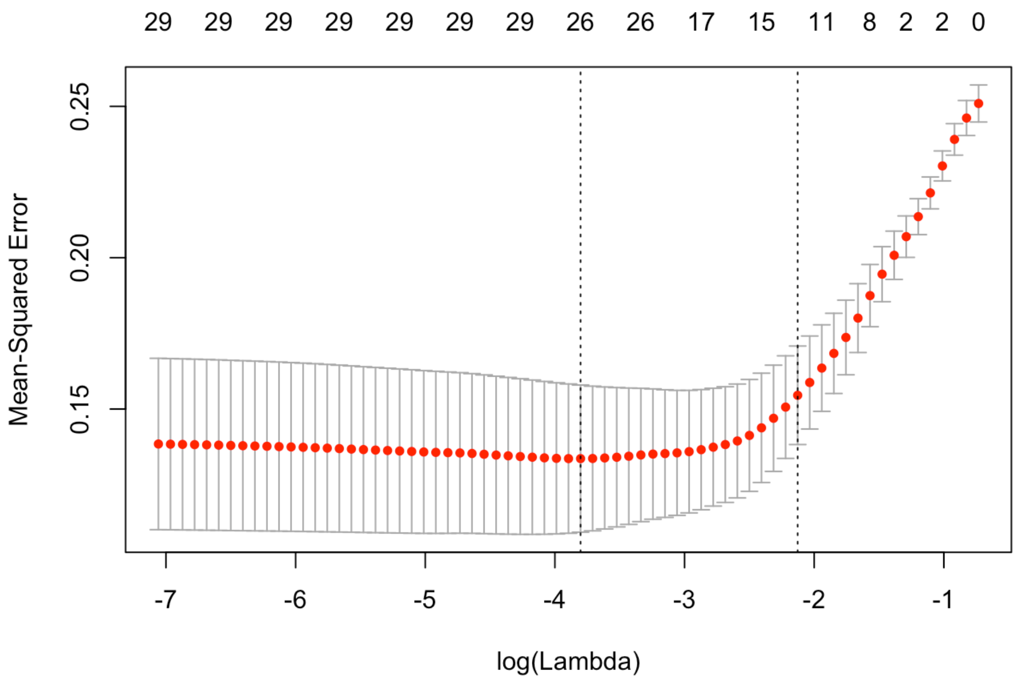
```
##
## Call:
## glm(formula = y ~ x, family = binomial("logit"))
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.390e-05  -2.110e-08   2.110e-08   2.110e-08   2.256e-05
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  7.815e+00  5.947e+04   0.000    1.000
## x1           1.536e+01  4.586e+04   0.000    1.000
## x2           4.219e+01  5.145e+04   0.001    0.999
## x3          -1.483e+01  5.276e+04   0.000    1.000
## x4          -2.831e+01  3.652e+04  -0.001    0.999
## x5          -4.968e+00  5.440e+04   0.000    1.000
## x6          -4.099e+01  5.211e+04  -0.001    0.999
## x7           1.176e+01  6.024e+04   0.000    1.000
## x8          -1.903e+01  6.449e+04   0.000    1.000
## x9           2.635e+01  6.535e+04   0.000    1.000
## x10         -5.263e+01  3.916e+04  -0.001    0.999
## x11         -1.079e+01  2.989e+04   0.000    1.000
## x12         -1.260e+01  9.747e+04   0.000    1.000
## x13         -1.591e+01  7.995e+04   0.000    1.000
## x14          1.435e+01  7.826e+04   0.000    1.000
## x15          1.976e+00  5.687e+04   0.000    1.000
## x16          2.148e+01  6.137e+04   0.000    1.000
## x17         -3.592e+00  9.982e+04   0.000    1.000
## x18         -2.154e+01  4.172e+04  -0.001    1.000
## x19         -1.736e+01  4.095e+04   0.000    1.000
## x20         -1.016e+01  3.951e+04   0.000    1.000
## x21          3.405e-01  4.589e+04   0.000    1.000
## x22          1.099e+01  4.842e+04   0.000    1.000
## x23          2.505e+01  5.916e+04   0.000    1.000
## x24         -1.740e+01  1.136e+05   0.000    1.000
## x25          1.387e+01  6.853e+04   0.000    1.000
## x26         -6.979e+00  3.488e+04   0.000    1.000
## x27         -1.540e+01  2.991e+04  -0.001    1.000
## x28          2.715e+01  4.363e+04   0.001    1.000
## x29         -9.921e+00  4.996e+04   0.000    1.000
## x30          6.700e+00  5.586e+04   0.000    1.000
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.3719e+02  on 99  degrees of freedom
## Residual deviance: 6.9742e-09  on 69  degrees of freedom
## AIC: 62
##
## Number of Fisher Scoring iterations: 25
```

- Negative log-likelihood + lasso penalty 를 최소로 하는 회귀계수

```
fit <- glmnet(x, y, family = "binomial")
plot(fit, xvar = "lambda", label = TRUE)
```



```
cvfit <- cv.glmnet(x, y, alpha = .5)
plot(cvfit)
```



```
print(cvfit$lambda.min)
```

```
## [1] 0.02232384
```

```
print(s <- cvfit$lambda.min)
```

```
## [1] 0.02232384
```

```
pred <- predict(fit, newx = x, type = "class", s = s)
table(y, pred)
```

```
##    pred
## y    0  1
##   0 43  1
##   1  1 55
```