Machine Learning in Public Health

Lecture 6: Linear Model Selection and Regularization (Lab)

Dr. Yang Feng

Best Subset Selection

```
library(ISLR)
library(tidyverse)
data("Credit")
credit_cate <- Credit %>% dplyr::select((-ID))
credit <- model.matrix(~., data = credit_cate) %>%
  as_tibble() %>%
  select(-"(Intercept)")
set.seed(0)
tr_ind <- sample(1:nrow(credit), 200)
credit_tr <- credit[tr_ind,]
credit_te <- credit[-tr_ind,]
```

```
library(leaps)
best_subset <- regsubsets(Balance ~., data = credit_tr)
##default only considers models size up to 8
best_subset <- regsubsets(Balance ~., data = credit_tr, nvmax = 11)
summary(best_subset)
```

```
## Subset selection object
## Call: regsubsets.formula(Balance ~ ., data = credit_tr, nvmax = 11)
## 11 Variables  (and intercept)
##                    Forced in Forced out
## Income                 FALSE      FALSE
## Limit                  FALSE      FALSE
## Rating                 FALSE      FALSE
## Cards                  FALSE      FALSE
## Age                    FALSE      FALSE
## Education              FALSE      FALSE
## GenderFemale           FALSE      FALSE
## StudentYes             FALSE      FALSE
## MarriedYes             FALSE      FALSE
## EthnicityAsian         FALSE      FALSE
## EthnicityCaucasian     FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: exhaustive
##           Income Limit Rating Cards Age Education GenderFemale StudentYes
## 1  ( 1 )  " "    " "   "*"    " "   " " " "       " "          " "
## 2  ( 1 )  "*"    " "   "*"    " "   " " " "       " "          " "
```

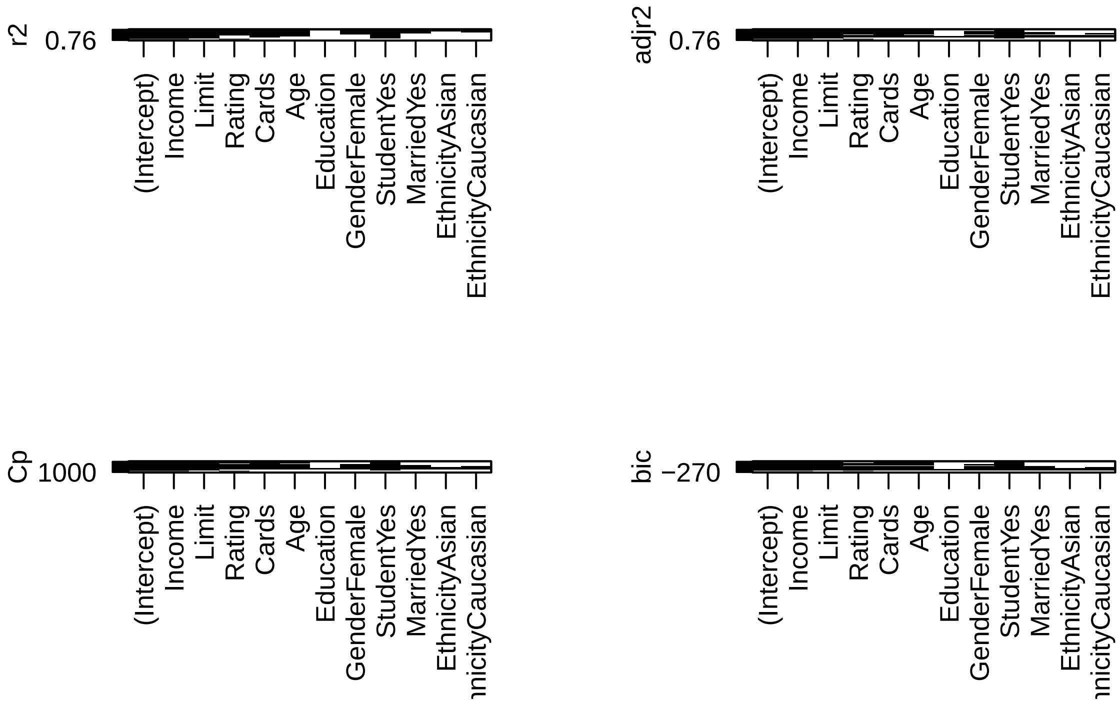
```
## 3  ( 1 )  "*"    "*"   " "    " "   " " " "       " "          "*"
## 4  ( 1 )  "*"    "*"   " "    "*"   " " " "       " "          "*"
## 5  ( 1 )  "*"    "*"   " "    "*"   "*" " "       " "          "*"
## 6  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       " "          "*"
## 7  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 8  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 9  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 10  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 11  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"       "*"          "*"
##           MarriedYes EthnicityAsian EthnicityCaucasian
## 1  ( 1 )  " "        " "            " "
## 2  ( 1 )  " "        " "            " "
## 3  ( 1 )  " "        " "            " "
## 4  ( 1 )  " "        " "            " "
## 5  ( 1 )  " "        " "            " "
## 6  ( 1 )  " "        " "            " "
## 7  ( 1 )  " "        " "            " "
## 8  ( 1 )  "*"        " "            " "
## 9  ( 1 )  "*"        " "            "*"
## 10  ( 1 ) "*"        "*"            "*"
## 11  ( 1 ) "*"        "*"            "*"
```

```r
##now considers all 11 variables
best_subset_sum <- summary(best_subset)

best_subset_sum$rsq
```

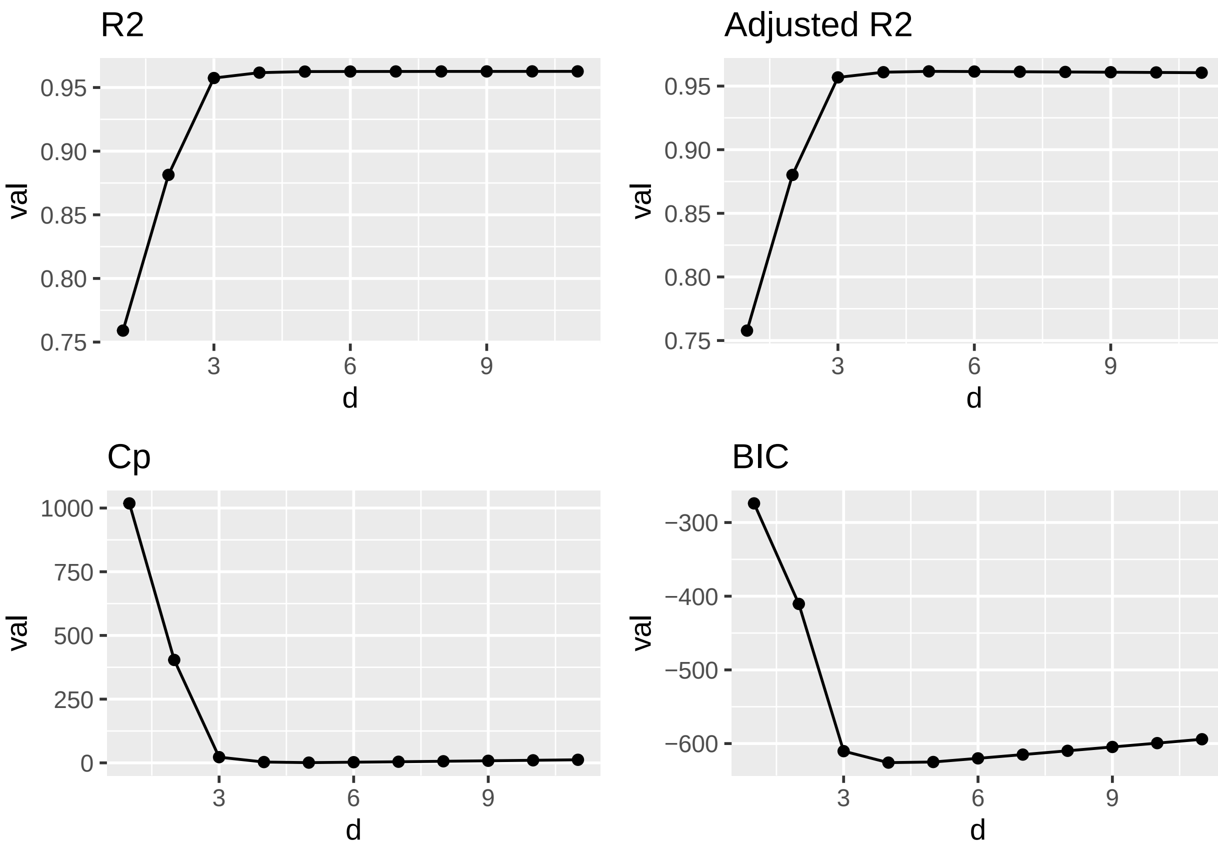
```
##  [1] 0.7590176 0.8813666 0.9574599 0.9616792 0.9625211 0.9625861 0.9626233
##  [8] 0.9626522 0.9626658 0.9626853 0.9626897
```

```r
par(mfrow = c(2,2))
plot(best_subset, scale =  "r2")
plot(best_subset, scale = "adjr2")
plot(best_subset, scale = "Cp")
plot(best_subset, scale = "bic")
```



Optimal model for each size: four methods

```
measures <- c("rsq", "adjr2", "cp", "bic")
our_names <- c("R2", "Adjusted R2", "Cp", "BIC")
size_seq <- 1:length(best_subset_sum$rsq)
my_plots <- NULL
for(mea_ind in seq_along(measures)){
  dat <- data.frame(d = size_seq, val = best_subset_sum[[measures[mea_ind]]])
  my_plots[[mea_ind]] <- ggplot(dat, mapping = aes(x = d, y = val)) + geom_point() + geom_line() +
    ggtitle(our_names[mea_ind])
}
library(egg)
grid.arrange(grobs = my_plots, ncol = 2)
```



Coefficients corresponding to optimal models

```
coef(best_subset, 1:11)
```

```
## [[1]]
## (Intercept)      Rating
## -354.238144    2.501169
##
## [[2]]
## (Intercept)      Income      Rating
## -548.260759   -7.684998    4.002145
##
## [[3]]
##  (Intercept)        Income         Limit   StudentYes
## -456.4610300   -7.8639273    0.2723742  429.2673608
##
## [[4]]
##  (Intercept)        Income         Limit         Cards   StudentYes
## -523.0518688   -7.7925423    0.2712276    23.0613375  437.2257637
##
## [[5]]
##  (Intercept)        Income         Limit         Cards           Age   StudentYes
## -474.4151238   -7.6386077    0.2695746    23.6202781    -0.8761518  435.8910167
##
```

```
## [[6]]
##  (Intercept)       Income        Limit       Rating        Cards          Age
## -483.1236387   -7.6550380    0.2425861    0.4057927   21.4430273   -0.8818302
##   StudentYes
##  434.7457520
##
## [[7]]
##  (Intercept)       Income        Limit       Rating        Cards          Age
## -479.7062751   -7.6587166    0.2432506    0.3961782   21.4556382   -0.8844975
## GenderFemale   StudentYes
##   -5.9442936  434.5896004
##
## [[8]]
##  (Intercept)       Income        Limit       Rating        Cards          Age
## -476.0559413   -7.6549983    0.2430891    0.3995257   21.4452607   -0.8982854
## GenderFemale   StudentYes   MarriedYes
##   -5.6678976  433.4931082   -5.4990089
##
## [[9]]
##        (Intercept)             Income              Limit             Rating
##       -478.2999819         -7.6539473          0.2430925          0.3992281
##              Cards                Age       GenderFemale         StudentYes
##         21.4787522         -0.8909104         -5.6975997        433.3446404
##         MarriedYes EthnicityCaucasian
##         -5.5205462          3.6022736
##
## [[10]]
##        (Intercept)             Income              Limit             Rating
##       -482.4570451         -7.6484920          0.2412729          0.4253337
##              Cards                Age       GenderFemale         StudentYes
##         21.4309499         -0.8814066         -5.7321331        433.2456953
##         MarriedYes     EthnicityAsian EthnicityCaucasian
##         -6.0516622          6.2314770          6.8553322
##
## [[11]]
##        (Intercept)             Income              Limit             Rating
##       -477.5268904         -7.6464664          0.2415817          0.4201142
##              Cards                Age          Education       GenderFemale
##         21.4194258         -0.8833370         -0.3266788         -5.7043018
##         StudentYes         MarriedYes     EthnicityAsian EthnicityCaucasian
##        433.3529352         -6.1769790          6.3441993          6.8727948
```

```
coef(best_subset, 4)
```

```
##  (Intercept)       Income        Limit        Cards   StudentYes
## -523.0518688   -7.7925423    0.2712276   23.0613375  437.2257637
```

```
best_ind <- which.min(best_subset_sum$bic)
best_coef <- coef(best_subset, best_ind)
tr_x <- credit_tr %>% select(names(best_coef)[-1])
tr_pred <- cbind(1, as.matrix(tr_x)) %*% best_coef
tr_error_best <- mean((tr_pred - credit_tr$Balance)^2)
te_x <- credit_te %>% select(names(best_coef)[-1])
```

```r
te_pred <- cbind(1, as.matrix(te_x)) %*% best_coef
te_error_best <- mean((te_pred - credit_te$Balance)^2)
tr_error_best
```

```
## [1] 9033.365
```

```r
te_error_best
```

```
## [1] 10837.48
```

Forward Stepwise Selection

```r
forward_fit <- regsubsets(Balance ~., data = credit_tr, method = "forward", nvmax = 11)
forward_sum <- summary(forward_fit)
best_ind <- which.min(forward_sum$bic)
best_coef <- coef(forward_fit, best_ind)
```

```r
tr_x <- credit_tr %>% select(names(best_coef)[-1])
tr_pred <- cbind(1, as.matrix(tr_x)) %*% best_coef
tr_error_forward <- mean((tr_pred - credit_tr$Balance)^2)
te_x <- credit_te %>% select(names(best_coef)[-1])
te_pred <- cbind(1, as.matrix(te_x)) %*% best_coef
te_error_forward <- mean((te_pred - credit_te$Balance)^2)
tr_error_forward
```

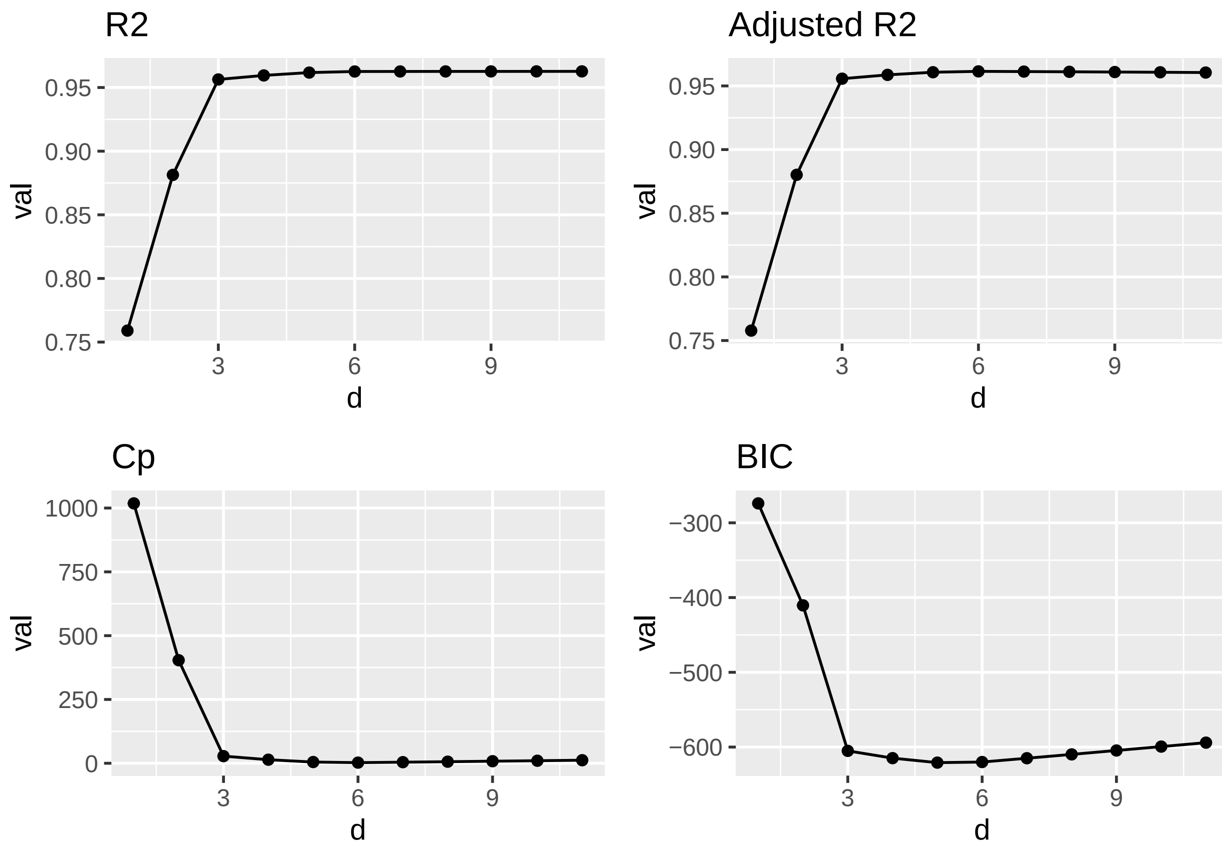
```
## [1] 9020.523
```

```r
te_error_forward
```

```
## [1] 10702.69
```

Optimal model for each size: four methods

```r
size_seq <- 1:length(forward_sum$rsq)
my_plots <- NULL
for(mea_ind in seq_along(measures)){
  dat <- data.frame(d = size_seq, val = forward_sum[[measures[mea_ind]]])
  my_plots[[mea_ind]] <- ggplot(dat, mapping = aes(x = d, y = val)) + geom_point() + geom_line() +
    ggtitle(our_names[mea_ind])
}
grid.arrange(grobs = my_plots, ncol = 2)
```



Backward Stepwise Selection

```
backward_fit <- regsubsets(Balance ~., data = credit_tr, method = "backward", nvmax = 11)
backward_sum <- summary(backward_fit)
best_ind <- which.min(backward_sum$bic)
best_coef <- coef(backward_fit, best_ind)
```

```
tr_x <- credit_tr %>% select(names(best_coef)[-1])
tr_pred <- cbind(1, as.matrix(tr_x)) %*% best_coef
tr_error_backward <- mean((tr_pred - credit_tr$Balance)^2)
te_x <- credit_te %>% select(names(best_coef)[-1])
te_pred <- cbind(1, as.matrix(te_x)) %*% best_coef
te_error_backward <- mean((te_pred - credit_te$Balance)^2)
tr_error_backward
```

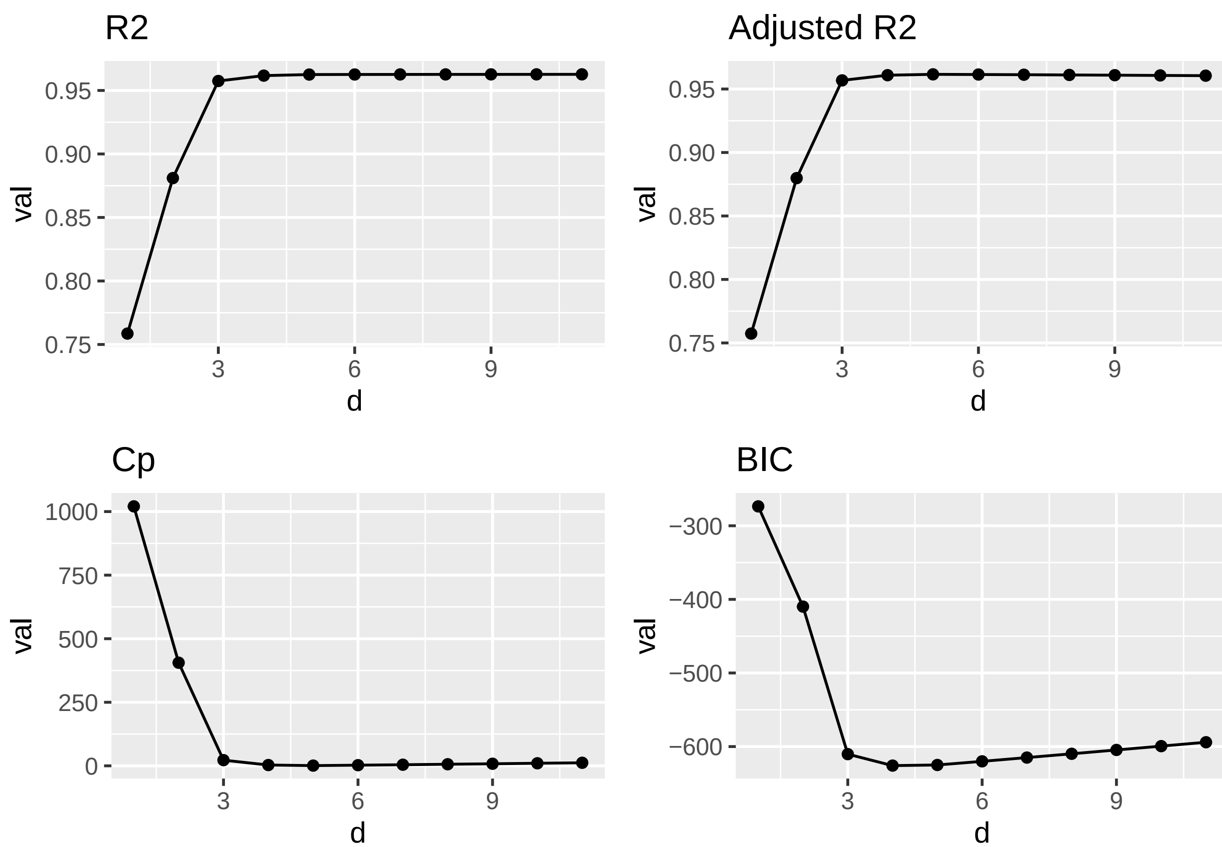
```
## [1] 9033.365
```

```
te_error_backward
```

```
## [1] 10837.48
```

Optimal model for each size: four methods

```
size_seq <- 1:length(backward_sum$rsq)
my_plots <- NULL
for(mea_ind in seq_along(measures)){
  dat <- data.frame(d = size_seq, val = backward_sum[[measures[mea_ind]]])
  my_plots[[mea_ind]] <- ggplot(dat, mapping = aes(x = d, y = val)) + geom_point() + geom_line() +
    ggtitle(our_names[mea_ind])
}
grid.arrange(grobs = my_plots, ncol = 2)
```



Ridge solution path for Credit data

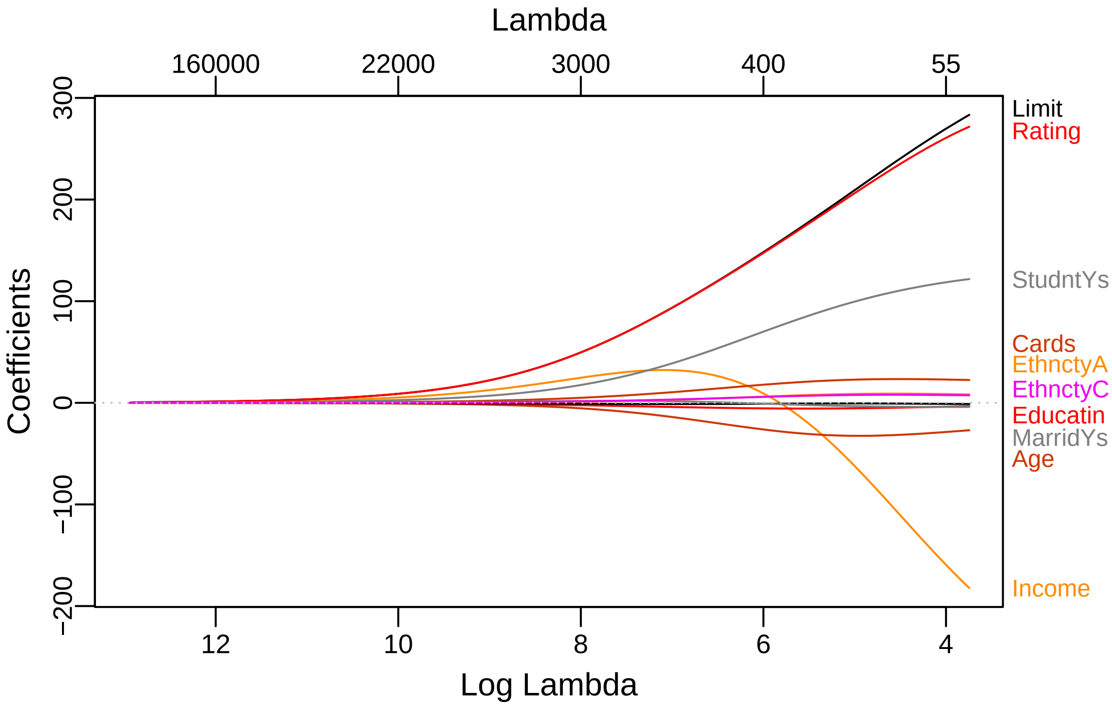
```
library(glmnet)
library(caret)
library(plotmo)
x_tr <- as.matrix(credit_tr[,-12])
y_tr <- credit_tr[, 12, drop = T]
x_te <- as.matrix(credit_te[,-12])
y_te <- credit_te[, 12, drop = T]

std_fit <- preProcess(x_tr, method = c("center", "scale"))
```

```r
x_tr_std <- predict(std_fit, x_tr)
x_te_std <- predict(std_fit, x_te)

fit_rdige <- glmnet(x_tr_std, y_tr, alpha = 0)

plot_glmnet(fit_rdige)
```




```r
set.seed(0)
# with standardization
cv_fit_ridge <- cv.glmnet(x_tr, y_tr, alpha = 0)
tr_pred <- predict(cv_fit_ridge, newx = x_tr)
te_pred <- predict(cv_fit_ridge, newx = x_te)
tr_error_ridge <- mean((tr_pred - y_tr)^2)
te_error_ridge <- mean((te_pred - y_te)^2)
tr_error_ridge
```

```
## [1] 14812.97
```

```r
te_error_ridge
```

```
## [1] 16207.82
```

```r
# without standardization
cv_fit_ridge <- cv.glmnet(x_tr_std, y_tr, alpha = 0)
tr_pred <- predict(cv_fit_ridge, newx = x_tr_std)
te_pred <- predict(cv_fit_ridge, newx = x_te_std)
tr_error_ridge <- mean((tr_pred - y_tr)^2)
te_error_ridge <- mean((te_pred - y_te)^2)
tr_error_ridge
```

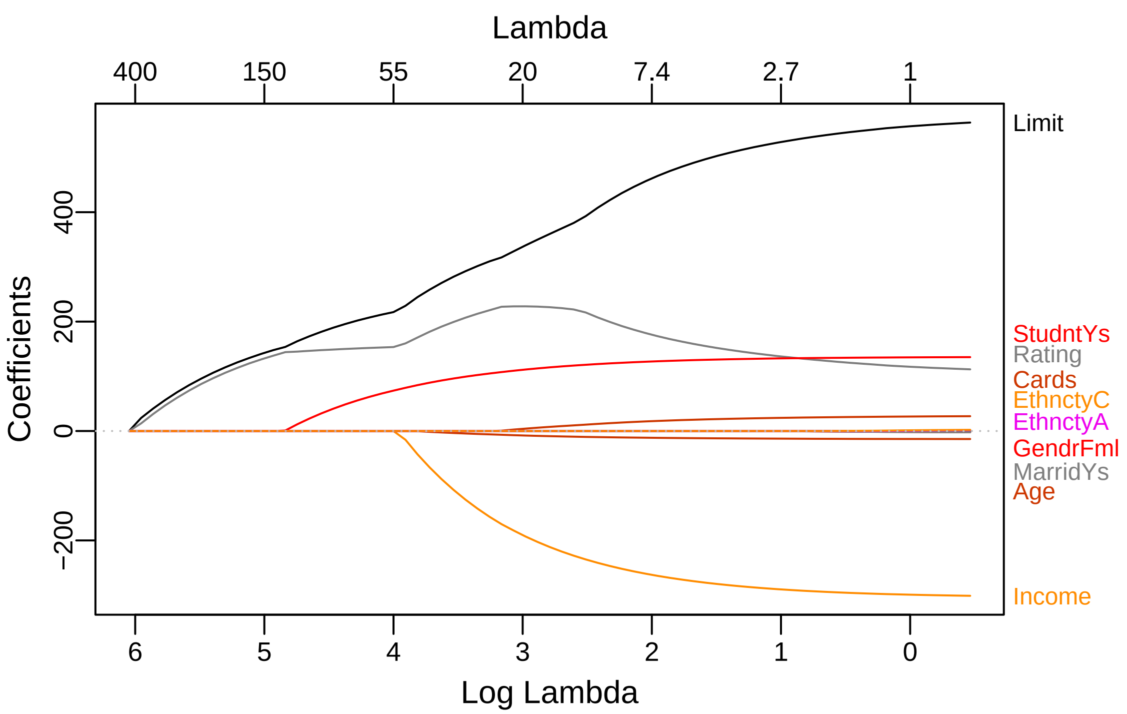
```
## [1] 14812.97
```

```
te_error_ridge
```

```
## [1] 16207.82
```

lasso Solution Path

```
fit_lasso <- glmnet(x_tr_std, y_tr, alpha = 1)
plot_glmnet(fit_lasso)
```




```
cv_fit_lasso <- cv.glmnet(x_tr, y_tr)
tr_pred <- predict(cv_fit_lasso, newx = x_tr)
te_pred <- predict(cv_fit_lasso, newx = x_te)
tr_error_lasso <- mean((tr_pred - y_tr)^2)
te_error_lasso <- mean((te_pred - y_te)^2)
tr_error_lasso
```

```
## [1] 10361.33
```

```
te_error_lasso
```

```
## [1] 11532.49
```

```r
df <- data.frame(methods = c("Best Subset", "Forward", "Backward", "Ridge", "Lasso"),
                 test_error = c(te_error_best, te_error_forward, te_error_backward, te_error_ridge, te_e
df
```

```
##       methods test_error
## 1 Best Subset   10837.48
## 2     Forward   10702.69
## 3    Backward   10837.48
## 4       Ridge   16207.82
## 5       Lasso   11532.49
```