Lecture 11

Variable selection

Julian Reif
Fall 2025

RStudio setup for this lecture

- Log into RStudio on your Amazon EC2 instance
  - Use AMI `FIN550-RStudio` with IAM role `BigDataEC2Role`

```
# Enter this command via RStudio Terminal
aws s3 cp --recursive s3://bigdata-fin550-reif/lecture-11 ~/fin550/lecture-11
```

Selecting variables as predictors

- Including more variables in a model can improve predictive power
- If variables are uninformative, including them can cause overfitting
  - In this case, adding variables actually **reduces** predictive power
- Today: learn how to include variables in a model without overfitting

Selecting from p possible predictors

- Suppose you have p possible predictors: $X_1, X_2, \ldots, X_p$

- Possible models:
  - $Y = \beta_0$
  - $Y = \beta_0 + \beta_1 X_1$
  - $Y = \beta_0 + \beta_5 X_5$
  - $Y = \beta_0 + \beta_3 X_3 + \beta_4 X_4 + \beta_{10} X_{10}$
  - ...

- In total, there are 2^p possible models

Methods for selecting variables

- Estimate all possible models and compare them
    - If p is large, however, then computationally impossible

- Alternative: employ "intelligent search methods"
    - Reduces computation time
    - But, it is possible that these will miss the best possible model

- Today, we learn about three popular methods:
    1. Best subset selection
    2. Forward stepwise selection
    3. Backward stepwise selection

Best subset selection

Try it: load seatbelt data

```r
library(tidyverse)
library(boot)

# Load the data into a data frame
seatbelts <- as_tibble(datasets::Seatbelts)

# We will consider 4 possible predictors of DriversKilled:
#   X1 = law
#   X2 = PetrolPrice
#   X3 = kms
#   X4 = kms^2
seatbelts <- seatbelts %>%
  select(DriversKilled, law, PetrolPrice, kms) %>%
  mutate(kms2 = kms^2)
```

Monthly road casualties in Great Britain, 1969-1984

Variable name	Definition
DriversKilled	Car drivers killed (monthly)
law	0/1: was a seatbelt law in effect that month?
PetrolPrice	Price of petrol (gasoline)
kms	Distance driven
kms2	Distance driven squared

Try it: inspect the 1969-1984 monthly road casualties data

```r
# How many observations are there? Does that number make sense?

# What is the range of the outcome variable, `DriversKilled`?
```

Inspect the 1969-1984 monthly road casualties data

```r
# How many observations are there? Does that make sense?
nrow(seatbelts)

# What is the range of the outcome variable, `DriversKilled`?
summary(seatbelts$DriversKilled)
```

```
# [1] 192
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#    60.0   104.8   118.5   122.8   138.0   198.0
```

Best subset selection

- Consider all possible models
  - We have four predictors, so we must evaluate $2^4 = 16$ models

- Select the model with the lowest **cross-validated** mean-squared error

- Why not use (non-cross-validated) mean-squared error?
  - Because then we would always select models with more variables
  - That can lead to overfitting

Try it: create a formula object

```
# M0: constant only model (i.e. no predictors)
f0_1 <- formula(DriversKilled ~ 1)
f0_1

# Following code is same as: `lm(DriversKilled ~ 1, data=seatbelts)`
lm(f0_1, data = seatbelts)
```

```
# DriversKilled ~ 1
#
# Call:
# lm(formula = f0_1, data = seatbelts)
#
# Coefficients:
# (Intercept)
#       122.8
```

Try it: define all possible formulas

```
# M1: all models with 1 predictor
f1_1 <- formula(DriversKilled ~ law)
f1_2 <- formula(DriversKilled ~ PetrolPrice)
f1_3 <- formula(DriversKilled ~ kms)
f1_4 <- formula(DriversKilled ~ kms2)

# M2: all models with 2 predictors
f2_1 <- formula(DriversKilled ~ law + PetrolPrice)
f2_2 <- formula(DriversKilled ~ law + kms)
f2_3 <- formula(DriversKilled ~ law + kms2)
f2_4 <- formula(DriversKilled ~ PetrolPrice + kms)
f2_5 <- formula(DriversKilled ~ PetrolPrice + kms2)
f2_6 <- formula(DriversKilled ~ kms + kms2)
```

Try it: define all possible formulas

```
# M3: all models with 3 predictors
f3_1 <- formula(DriversKilled ~ PetrolPrice + kms + kms2)
f3_2 <- formula(DriversKilled ~ law + kms + kms2)
f3_3 <- formula(DriversKilled ~ law + PetrolPrice + kms2)
f3_4 <- formula(DriversKilled ~ law + PetrolPrice + kms)

# M4: all models with 4 predictors
f4_1 <- formula(DriversKilled ~ law + PetrolPrice + kms + kms2)
```

Try it: create function to calculate cross-validated error

```r
# Use glm() to estimate linear regression of formula f on the seatbelts dataset
cv_fun <- function(f) {
  glmfit <-
  cv.glm(data = seatbelts, glmfit)$delta[1]
}

# Run the function
cv_fun(f0_1)

cv_fun(f3_4)
```

Create function to calculate cross-validated error

```r
# Use glm() to estimate linear regression of formula f on the seatbelts dataset
cv_fun <- function(f) {
  glmfit <- glm(f, data = seatbelts)
  cv.glm(data = seatbelts, glmfit)$delta[1]
}

# Run the function
cv_fun(f0_1)

cv_fun(f3_4)
```

```
# [1] 647.5111
# [1] 533.2905
```

Try it: calculate MSE for all 16 formulas

```r
# Create a list of formulas
formulas <- list(f0_1,
                 f1_1, f1_2, f1_3, f1_4,
                 f2_1, f2_2, f2_3, f2_4, f2_5, f2_6,
                 f3_1, f3_2, f3_3, f3_4,
                 f4_1)

# Create a vector for storing the LOOCV MSE for formulas
formulas_cv <- vector("numeric", length(formulas))

# Use the function we created to calculate the LOOCV MSE for each formula
for (i in 1:length(formulas)) {
  formulas_cv[[i]] <-
}
```

Calculate MSE for all 16 formulas

```r
# Create a list of formulas
formulas <- list(f0_1,
                 f1_1, f1_2, f1_3, f1_4,
                 f2_1, f2_2, f2_3, f2_4, f2_5, f2_6,
                 f3_1, f3_2, f3_3, f3_4,
                 f4_1)

# Create a vector for storing the LOOCV MSE for formulas
formulas_cv <- vector("numeric", length(formulas))

# Use the function we created to calculate the LOOCV MSE for each formula
for (i in 1:length(formulas)) {
  formulas_cv[[i]] <- cv_fun(formulas[[i]])
}
```

Try it: identify model with the smallest MSE

```r
best_model <- which.min(formulas_cv)

# Formula
formulas[[best_model]]

# MSE
formulas_cv[[best_model]]
```

```
# DriversKilled ~ law + PetrolPrice + kms2
# [1] 531.6267
```

lapply() is useful when applying a function to a list/vector

```
# Instead of writing this for loop:
#    for (i in 1:length(formulas)) {
#      formulas_cv[[i]] <- cv_fun(formulas[[i]])
#    }
# We can use lapply():
formulas_cv2 <- lapply(formulas, cv_fun)

best_model <- which.min(formulas_cv2)
formulas[[best_model]]
```

```
# DriversKilled ~ law + PetrolPrice + kms2
```

sapply() is equivalent to simplify(lapply())

```r
# lapply() returns a list. Recall: formulas_cv2 <- lapply(formulas, cv_fun)
typeof(formulas_cv2)
```

```
# [1] "list"
```

```r
# sapply() works like lapply(), but it "simplifies" the output
formulas_cv3 <- sapply(formulas, cv_fun)
typeof(formulas_cv3)
```

```
# [1] "double"
```

```r
all.equal(simplify(formulas_cv2), formulas_cv3)
```

```
# [1] TRUE
```

Technical note: parallel computing

- Parallel processing uses multiple CPU cores to run tasks simultaneously
- Great for tasks that repeat the same operation many times (e.g., model fitting)
- Not all code can be parallelized
  - If B depends on output from A, then can't run A and B simultaneously
- `lapply()` and `sapply()` run the same function on multiple inputs
  - R can run these in parallel with minimal edits to the code:

```
library(parallel)
formulas_cv2 <- mclapply(formulas, cv_fun, mc.cores = 4)
```

Problems with best subset selection

- Works well when the number of possible predictors is very small

- But, not feasible with large number of predictors

```r
# Our model had 4 possible predictors: 2^4 = 16 models
length(formulas)

# 40 predictors: 2^40 = over 1 trillion models
prettyNum(2^40, big.mark = ",")

# 300 predictors: 2.04 x 10^90 models!
format(2^300, scientific=TRUE)
```

```
# [1] 16
# [1] "1,099,511,627,776"
# [1] "2.037036e+90"
```

Forward stepwise selection

Forward stepwise selection searches intelligently

- Start with all possible one-variable models
  - Find best predictor and keep it, e.g., $X = X_2$

- Now consider all two-variable models that include X_2
  - Find best predictor and keep it, e.g., $X = \{X_2, X_5\}$

- Repeat until you reach a model that includes all possible predictors

- Choose the model (1 variable, 2 variables, … p variables) with lowest cross-validated MSE

Algorithm for forward stepwise selection

1. Let M_0 denote the null model with no predictors

2. Estimate all models with a single predictor
   - Let M_1 denote the 1-variable model with the lowest mean-squared error

3. Estimate all the models that add another predictor to the model M_1
   - Let M_2 denote the 2-variable model with the lowest mean-squared error

4. Repeat step 3 until you reach a model with all predictors (M_p)

5. Select the model M with the lowest **cross-validated** mean-squared error

Forward stepwise selection has pros and cons

- Instead of 2^p models, we estimate $1 + p(p+1)/2$ models
    - When $p = 20$, this reduces number of models from 1,048,576 to 211

- However, it is possible that we may miss the best model

- Example: consider model with $p = 3$ predictors
    - Suppose best possible 1-variable model includes X_1
    - Suppose best possible 2-variable model includes X_2 and X_3
    - Then forward stepwise selection will fail to find this 2-variable model

Seatbelts example: M0 and M1

```r
# Initialize vector and list to store MSE and formula for each model M (M0-M4)
forward_cv <- vector("numeric", 5)
forward_formulas <- vector(mode = "list", 5)

# M0: no predictors
forward_cv[1] <- cv_fun(f0_1)
forward_formulas[[1]] <- f0_1

# M1: consider all 1-variable models
M_formulas <- list(f1_1, f1_2, f1_3, f1_4)
cv <- sapply(M_formulas, cv_fun)

forward_cv[2] <- min(cv)                              # Store smallest MSE for M1
forward_formulas[[2]] <- M_formulas[[which.min(cv)]] # Store best model for M1

forward_formulas[[2]] # Display the 1-variable model with the lowest MSE
```

```
# DriversKilled ~ PetrolPrice
```

Seatbelts example: M2

```r
# M2: consider all 2-variable models that include PetrolPrice
M_formulas <- list(f2_1, f2_4, f2_5)
print(as.character(M_formulas))
cat("\n")

cv <- sapply(M_formulas, cv_fun)
forward_cv[3] <- min(cv)
forward_formulas[[3]] <- M_formulas[[which.min(cv)]]

forward_formulas[[3]] # Display the 2-variable model with the lowest MSE
```

```
# [1] "DriversKilled ~ law + PetrolPrice"  "DriversKilled ~ PetrolPrice + kms"
# [3] "DriversKilled ~ PetrolPrice + kms2"
#
# DriversKilled ~ PetrolPrice + kms2
```

Seatbelts example: M3 and M4

```r
# M3: consider all 3-variable models that include PetrolPrice and kms2
M_formulas <- list(f3_1, f3_3)
print(as.character(M_formulas))
cat("\n")

cv <- sapply(M_formulas, cv_fun)
forward_cv[4] <- min(cv)
forward_formulas[[4]] <- M_formulas[[which.min(cv)]]

forward_formulas[[4]] # Display the 3-variable model with the lowest MSE

# M4: all predictors (4-variable model)
forward_cv[5] <- cv_fun(f4_1)
forward_formulas[[5]] <- f4_1
```

```
# [1] "DriversKilled ~ PetrolPrice + kms + kms2"
# [2] "DriversKilled ~ law + PetrolPrice + kms2"
#
# DriversKilled ~ law + PetrolPrice + kms2
```

Select the model with the lowest MSE

```r
# Display MSE and formulas for models M0-M4
forward_cv
cat("\n")
print(as.character(forward_formulas))
cat("\n")

# Display model with the lowest cross-validated mean-squared error
forward_formulas[[which.min(forward_cv)]]
```

```
# [1] 647.5111 556.7901 536.4993 531.6267 536.0520
#
# [1] "DriversKilled ~ 1"
# [2] "DriversKilled ~ PetrolPrice"
# [3] "DriversKilled ~ PetrolPrice + kms2"
# [4] "DriversKilled ~ law + PetrolPrice + kms2"
# [5] "DriversKilled ~ law + PetrolPrice + kms + kms2"
#
# DriversKilled ~ law + PetrolPrice + kms2
```

Backward stepwise selection

Backward stepwise selection is similar to forward

- Instead of starting with a model with 0 predictors, start with all predictors
- Incrementally remove the least predictive variable, one at a time
- Like forward stepwise selection, this reduces model count from 2^p to $1 + p(p+1)/2$
- Lab assignment: implement backward stepwise selection

Algorithm for backward stepwise selection

1. Let M_p denote the full model with all predictors

2. Estimate all models that remove a single predictor from M_p
    - Let M_{p-1} denote the $p-1$ variable model with the lowest mean-squared error

3. Estimate all the models that remove another predictor from the model M_{p-1}
    - Let M_{p-2} denote the $p-2$ variable model with the lowest mean-squared error

4. Repeat step 3 until you reach a model with no predictors (M_0)

5. Select the model M with the lowest **cross-validated** mean-squared error

Comparing the different selection models

- Best subset selection will always find the best model

- However, these three selection models usually yield similar results

- Special case: number of variables p exceeds number of observations n
  - Can still use forward stepwise selection, but not backward stepwise selection

- Shrinkage methods such as Lasso (next lecture) can also be used for variable selection

Summary

- Three variable selection methods:
  1. Best subset selection
  2. Forward stepwise selection
  3. Backward stepwise selection

- When number of predictors exceeds about 10, need intelligent search methods

- Use cross-validated MSE when comparing models with different numbers of variables

- Lab-11 due Sunday at 11:59pm
- Make sure to **stop your instance** when you are done working