Cross Validation

Philip Waggoner, MACS 30100 University of Chicago

Resampling

```
library(tidyverse)
library(ISLR)
library(broom)
library(rsample)
library(rcfss)
library(yardstick)

Auto <- as_tibble(Auto)

set.seed(1234)

auto_split <- initial_split(data = Auto,
                            prop = 0.5)

auto_train <- training(auto_split)
auto_test <- testing(auto_split)
```

```
auto_lm <- glm(mpg ~ horsepower, data = auto_train); summary(auto_lm)
```

MSE for training

```
(train_mse <- augment(auto_lm, newdata = auto_train) %>%
  mse(truth = mpg, estimate = .fitted))
```

MSE for validation

```
(test_mse <- augment(auto_lm, newdata = auto_test) %>%
  mse(truth = mpg, estimate = .fitted))
```

Some different models.

```
# visualize each model
ggplot(Auto, aes(horsepower, mpg)) +
  geom_point(alpha = .1) +
  geom_smooth(aes(color = "1"),
              method = "glm",
              formula = y ~ poly(x, i = 1, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "2"),
              method = "glm",
              formula = y ~ poly(x, i = 2, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "3"),
```

```
              method = "glm",
              formula = y ~ poly(x, i = 3, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "4"),
              method = "glm",
              formula = y ~ poly(x, i = 4, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "5"),
              method = "glm",
              formula = y ~ poly(x, i = 5, raw = TRUE),
              se = FALSE) +
  scale_color_brewer(type = "qual", palette = "Dark2") +
  labs(x = "Horsepower",
       y = "MPG",
       color = "Polynomial\norder") +
  theme_minimal()

# train and eval simultaneously
poly_results <- function(train, test, i) {
  mod <- glm(mpg ~ poly(horsepower, i, raw = TRUE),
             data = train)
  res <- augment(mod,
                 newdata = test) %>%
    mse(truth = mpg, estimate = .fitted)
  res
}

library(magrittr)

poly_mse <- function(i, train, test){
  poly_results(train, test, i) %$%
    mean(.estimate)
}

cv_mse <- tibble(terms = seq(from = 1, to = 5),
                 mse_test = map_dbl(terms, poly_mse, auto_train, auto_test))

ggplot(cv_mse, aes(terms, mse_test)) +
  geom_line() +
  labs(title = "Evaluating quadratic linear models",
       subtitle = "Using validation set",
       x = "Highest-order polynomial",
       y = "Mean Squared Error") +
  theme_minimal()
```

Classification

A case study with classification.

```
library(titanic)

titanic <- as_tibble(titanic_train) %>%
  mutate(Survived = factor(Survived))
```

```
titanic %>%
  head(n = 5)
```

```
survive_age_woman_x <- glm(Survived ~ Age * Sex, data = titanic,
                           family = binomial)
summary(survive_age_woman_x)
```

```
# helper fun
logit2prob <- function(x){
  exp(x) / (1 + exp(x))
}
```

```
# split the data into training and validation sets
titanic_split <- initial_split(data = titanic,
                               prop = 0.5)

# fit model to training data
train_model <- glm(Survived ~ Age * Sex,
                   data = training(titanic_split), # note the different syntax used here
                   family = binomial); broom::tidy(train_model)

# calculate predictions using validation set
x_test_accuracy <- augment(train_model,
                           newdata = testing(titanic_split)) %>%
  as_tibble() %>%
  mutate(.prob = logit2prob(.fitted),
         .pred = factor(round(.prob)))

# calculate test accuracy rate
accuracy(x_test_accuracy,
         truth = Survived,
         estimate = .pred)
```

LOOCV

LOOCV in regression

```
library(tidyverse)
library(ISLR)
library(broom)
library(rsample)
library(rcfss)
library(yardstick)

loocv_data <- loo_cv(Auto)

loocv_data %>%
  names()
```

Note: each element of `loocv_data$splits` is an object of class `rsplit`.

```
first_resample <- loocv_data$splits[[1]]
first_resample
```

```r
# but first, note training() and analysis() from resample contain same information; just different form
first_resample %>%
  analysis() %>% # data frame version
  head(n = 5)

first_resample %>%
  training() %>%
  head(n = 5)

# same with assessment() and testing()
first_resample %>%
  assessment() %>%
  head(n = 5)

first_resample %>%
  testing() %>%
  head(n = 5)
```

```r
holdout_results <- function(splits) {
  mod <- glm(mpg ~ horsepower,
             data = analysis(splits))
  res <- augment(mod, newdata = assessment(splits)) %>%
    mse(truth = mpg, estimate = .fitted)
  res
}
```

This function works also for a single resample:

```r
holdout_results(loocv_data$splits[[1]])
```

```r
loocv_data_poly1 <- loocv_data %>%
  mutate(results = map(splits, holdout_results)) %>%
  unnest(results) %>%
  spread(.metric, .estimate)
loocv_data_poly1 %>%
  head(n = 5)

loocv_data_poly1 %>%
  summarize(mse = mean(mse))
```

```r
# modified function to estimate model with varying polynomial order
holdout_results <- function(splits, i) {
  mod <- glm(mpg ~ poly(horsepower, i, raw = TRUE),
             data = analysis(splits))

  res <- augment(mod, newdata = assessment(splits)) %>%
    mse(truth = mpg, estimate = .fitted)
  res
}

# function to return MSE for a specific polynomial term
poly_mse <- function(i, loocv_data){
  loocv_mod <- loocv_data %>%
    mutate(results = map(splits, holdout_results, i)) %>%
    unnest(results) %>%
```

```
    spread(.metric, .estimate)

  mean(loocv_mod$mse)
}

library(tictoc)

{ # wrap and time
tic()
cv_mse <- tibble(terms = seq(from = 1, to = 5),
                 mse_loocv = map_dbl(terms, poly_mse, loocv_data))
toc()
} # takes ~25 seconds on my slower computer

cv_mse

ggplot(cv_mse, aes(terms, mse_loocv)) +
  geom_line() +
  labs(title = "Comparing quadratic linear models",
       subtitle = "Using LOOCV",
       x = "HPolynomial\norder",
       y = "Mean Squared Error") +
  theme_minimal()
```

LOOCV in classification

Let's verify the error rate of our interactive terms model for the Titanic data set:

```
titanic <- as_tibble(titanic_train) %>%
  mutate(Survived = factor(Survived))

holdout_results <- function(splits) {
  mod <- glm(Survived ~ Age * Sex, data = analysis(splits),
             family = binomial)
  res <- augment(mod, newdata = assessment(splits)) %>%
    as_tibble() %>%
    mutate(.prob = logit2prob(.fitted),
           .pred = round(.prob))
  res
}

titanic_loocv <- loo_cv(titanic) %>%
  mutate(results = map(splits, holdout_results)) %>%
  unnest(results) %>%
  mutate(.pred = factor(.pred)) %>%
  group_by(id) %>%
  accuracy(truth = Survived, estimate = .pred)

1 - mean(titanic_loocv$.estimate, na.rm = TRUE)
```

k-fold CV

Let's go back to the `Auto` data set and comparing across polynomial orders.

```r
# helper fun
holdout_results <- function(splits, i) {
  mod <- glm(mpg ~ poly(horsepower, i, raw = TRUE), data = analysis(splits))

  holdout <- assessment(splits)

  res <- augment(mod, newdata = holdout) %>%
    mse(truth = mpg, estimate = .fitted)

  res
}

# function to return MSE for a specific fit
poly_mse <- function(i, vfold_data){
  vfold_mod <- vfold_data %>%
    mutate(results = map(splits, holdout_results, i)) %>%
    unnest(results) %>%
    spread(.metric, .estimate)

  mean(vfold_mod$mse)
}


auto_cv10 <- vfold_cv(data = Auto,
                      v = 10)

# as before...
auto_cv10 %>%
  names()

cv_mse <- tibble(terms = seq(from = 1,
                             to = 5),
                 mse_vfold = map_dbl(terms, poly_mse, auto_cv10))
cv_mse
```

Comparison?

```r
auto_loocv <- loo_cv(Auto)

tibble(terms = seq(from = 1, to = 5), # takes a few seconds, given the mapping
       `10-fold` = map_dbl(terms, poly_mse, auto_cv10),
       LOOCV = map_dbl(terms, poly_mse, auto_loocv)
) %>%
  gather(method, MSE, -terms) %>%
  ggplot(aes(terms, MSE, color = method)) +
  geom_line() +
  labs(title = "MSE estimates",
       subtitle = "Comparing model complexity and accuracy",
       x = "Degree of Polynomial",
       y = "Mean Squared Error",
       color = "CV\nMethod") +
  theme_minimal()
```

On your own

For this section, you will work in small groups of 4-5. *I will create these groups at random.*

IMPORTANT: *Don't forget that this code you're working on here is due at the appropriate Canvas module (in the form of an attachment to a "Discussion" post) prior to 5:00 pm CDT tomorrow. You need only submit a* ***single*** *file/script to be considered for credit (i.e., this .Rmd with your code inserted below each question). Recall, I don't care whether you got things right. I only care that attempts to each question have been made.*

Return to the Titanic data, and apply 10-fold cross validation to a classification task. As noted before, though we haven't covered classification yet, you should have seen enough in today's session to complete (or at least attempt) the following. I will get you started with the packages and data needed to respond to each prompt below.

```
library(tidyverse)
library(tidymodels)
library(titanic)

titanic <- as_tibble(titanic_train) %>%
  mutate(Survived = factor(Survived))
```

1. Use 10-fold cross validation to build and evaluate a logistic regression predicting `Survived` as a function of interacting `Age` and `Sex`. To answer this, you will need to:
    - Build the classifer on the training set(s) across folds
    - Evaluate each classifer using the test set(s) across folds
2. Calculate the cross-validation *error rate* (not the accuracy rate) from your solution and report it.
3. Is this similar to the error from using LOOCV earlier in the session? Why or why not, do you think? (offer just a couple thoughts on the patterns – differences and similarities in error, computational speed, etc. – from each approach to resampling in this classification setting).