Unit 2 Lecture 3: Cross-validation

September 21, 2023

In this R demo, we will implement cross-validation to select the degrees of freedom of a natural spline fit, using the running example from the previous class. We'll need the following R packages:

```
library(tidyverse)
library(splines)   # for ns()
library(cowplot)   # for plot_grid()
library(stat471)   # for cross_validate_spline()
```

Training and validation

Let us create a training set:

```
set.seed(1)
f <- function(x) (sin(3 * x))
n <- 50
sigma <- 1
train_data <- tibble(
  x = seq(0, 2 * pi, length.out = n),
  y = f(x) + rnorm(n, sd = sigma)
)
train_data
```

```
## # A tibble: 50 x 2
##        x      y
##    <dbl>  <dbl>
##  1 0     -0.626
##  2 0.128  0.559
##  3 0.256 -0.140
##  4 0.385  2.51
##  5 0.513  1.33
##  6 0.641  0.118
##  7 0.769  1.23
##  8 0.898  1.17
##  9 1.03   0.640
## 10 1.15  -0.620
## # i 40 more rows
```

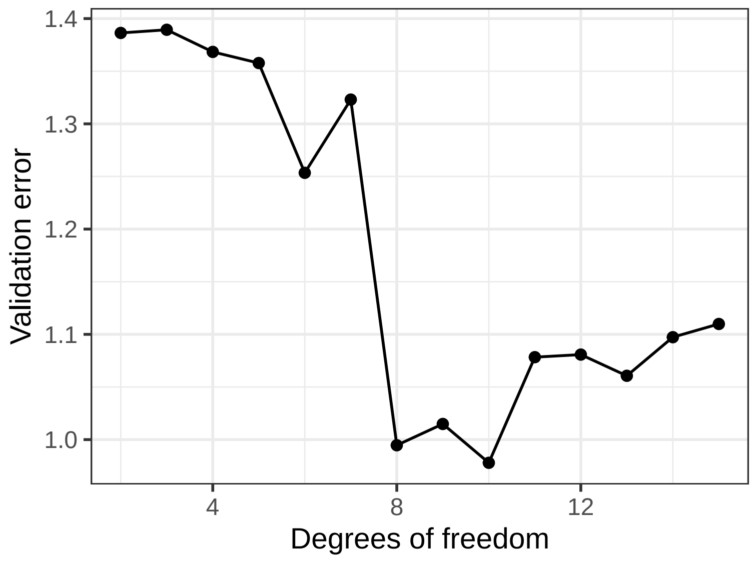
Let's also suppose we have a large validation set on our hands:

```
N <- 50000
validation_data <- tibble(
  x = seq(0, 2 * pi, length.out = N),
  y = f(x) + rnorm(n, sd = sigma)
)
```

Now let's fit splines with `df` = 2,3,...,15 to the training data, and evaluate their test error using the test set:

```r
# compute the validation error
max_df <- 15
validation_error <- numeric(max_df-1)
for (df in 2:max_df) {
  formula <- sprintf("y ~ ns(x, df = %d)", df - 1)
  spline_fit <- lm(formula = formula, data = train_data)
  y_hat_validation <- predict(spline_fit, newdata = validation_data)
  validation_error[df - 1] <- validation_data |>
    mutate(y_hat_validation = y_hat_validation) |>
    summarize(mean((y_hat_validation - y)^2)) |>
    pull()
}

# plot the validation error
p_val <- tibble(df = 2:max_df, validation_error) |>
  ggplot(aes(x = df, y = validation_error)) +
  geom_point() +
  geom_line() +
  labs(
    x = "Degrees of freedom",
    y = "Validation error"
  )
plot(p_val)
```



The issue is that we usually do not have a giant validation set for model selection purposes. We need to make do with our smallish training set for both model training and model selection. This is where cross-validation comes in handy!

Cross-validation for df = 5

The idea is to split our training samples into *folds* and then have the folds take turns being the validation set. Let's take a look.

```r
K <- 10
folds <- sample(rep(1:K, n / K))
train_data <- train_data |>
```

```
  mutate(fold = folds)
train_data
```

```
## # A tibble: 50 x 3
##        x      y  fold
##    <dbl>  <dbl> <int>
##  1 0     -0.626     3
##  2 0.128  0.559     6
##  3 0.256 -0.140     4
##  4 0.385  2.51      2
##  5 0.513  1.33      9
##  6 0.641  0.118     7
##  7 0.769  1.23      8
##  8 0.898  1.17      1
##  9 1.03   0.640     1
## 10 1.15  -0.620     5
## # i 40 more rows
```

Question: How would we select the data in fold number 1? How would we select all the data except fold number 1?

Let's first use cross-validation to estimate the test error for a spline fit with 5 degrees of freedom.

```
# create a vector of out-of-fold predictions
out_of_fold_predictions <- numeric(n)

# iterate over folds
for (current_fold in 1:K) {
  # out-of-fold data will be used for training
  out_of_fold_data <- train_data |> filter(fold != current_fold)
  # in-fold data will be used for validation
  in_fold_data <- train_data |> filter(fold == current_fold)

  # train on out-of-fold data
  spline_fit <- lm(y ~ ns(x, df = 5), data = out_of_fold_data)

  # predict on in-fold data
  out_of_fold_predictions[folds == current_fold] <-
    predict(spline_fit, newdata = in_fold_data)
}

# add the out-of-fold predictions to the data frame
results <- train_data |>
  mutate(yhat = out_of_fold_predictions)
results
```

```
## # A tibble: 50 x 4
##        x      y  fold    yhat
##    <dbl>  <dbl> <int>   <dbl>
##  1 0     -0.626     3  1.85
##  2 0.128  0.559     6  0.726
##  3 0.256 -0.140     4  0.878
##  4 0.385  2.51      2  0.252
##  5 0.513  1.33      9  0.242
##  6 0.641  0.118     7  0.181
##  7 0.769  1.23      8 -0.0212
##  8 0.898  1.17      1 -0.374
##  9 1.03   0.640     1 -0.490
## 10 1.15  -0.620     5  0.0871
## # i 40 more rows
```

```
# compute the CV estimate and standard error
results |>
  summarize(cv_fold = mean((yhat - y)^2), .by = fold) |> # CV estimates per fold
  summarize(
    cv_mean = mean(cv_fold),
    cv_se = sd(cv_fold) / sqrt(K)
  )
```

```
## # A tibble: 1 x 2
##   cv_mean cv_se
##     <dbl> <dbl>
## 1    1.70 0.339
```

What are two reasons this CV estimate may be different from the validation error estimated above?

Cross-validation for df = 2,3,. . . ,15

Now let's repeat what we did above for many degrees of freedom, because after all, the point of cross-validation is to choose the degrees of freedom. We can do this via `cross_validate_spline()` in the `stat471` package.

```
cv_results <- cross_validate_spline(
  x = train_data$x,
  y = train_data$y,
  nfolds = 10,
  df_values = 2:15
)
```

The `cv_results` variable stores an *object*, with the following fields:

```
names(cv_results)
```

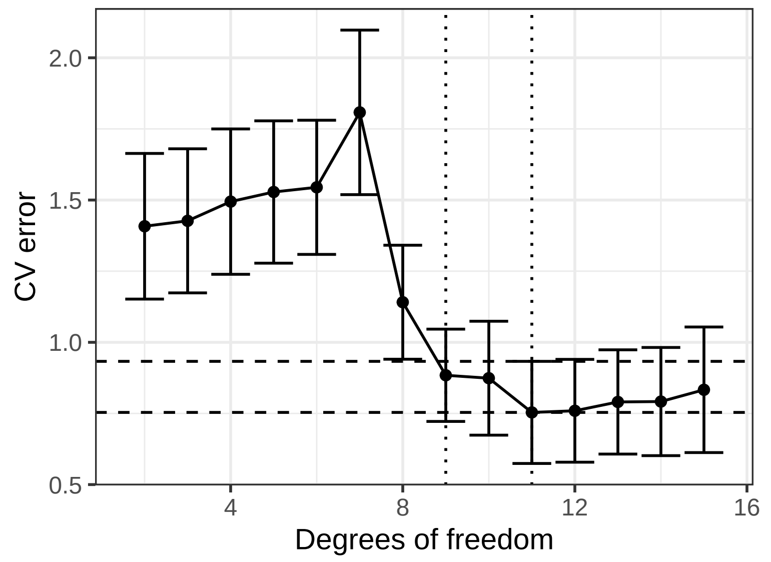
```
## [1] "cv_table" "cv_plot"  "df.1se"   "df.min"
```

Let's inspect these one at a time:

```
cv_results$cv_table
```

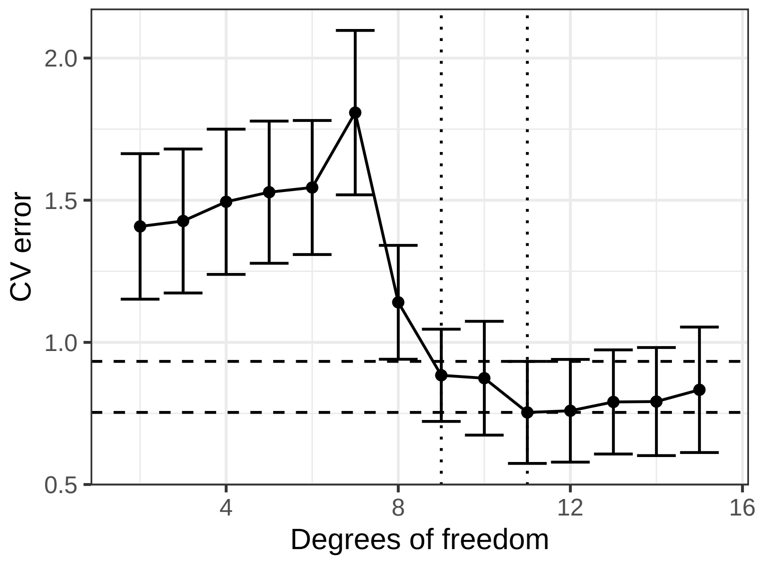
```
## # A tibble: 14 x 3
##       df cv_mean cv_se
##    <int>   <dbl> <dbl>
##  1     2   1.41  0.256
##  2     3   1.43  0.253
##  3     4   1.49  0.255
##  4     5   1.53  0.250
##  5     6   1.54  0.236
##  6     7   1.81  0.289
##  7     8   1.14  0.200
##  8     9   0.884 0.162
##  9    10   0.874 0.200
## 10    11   0.754 0.179
## 11    12   0.759 0.181
## 12    13   0.790 0.183
## 13    14   0.792 0.190
## 14    15   0.833 0.221
```

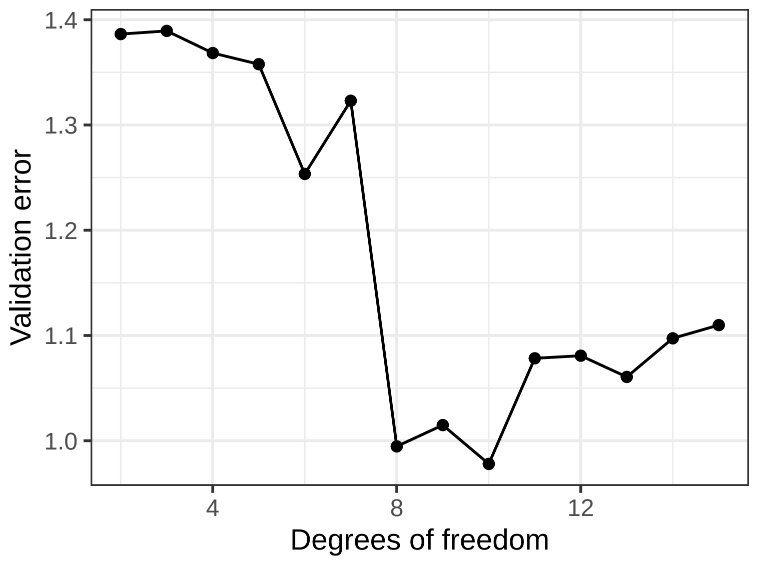
```
cv_results$cv_plot
```



Let's compare the CV plot to the validation error plot:

```
plot_grid(cv_results$cv_plot, p_val, nrow = 2)
```





Based on this plot, what degrees of freedom would we select using the one-standard error rule?

```r
cv_results$df.1se
```

```
## [1] 9
```

```r
cv_results$df.min
```

```
## [1] 11
```

Exercise

Use 10-fold cross-validation and the one-standard error rule to select the optimal number of degrees of freedom for regressing `income` on `age` in the `income_2007` data from Unit 3 Lecture 1. Make the CV plot, extract `df.1se` and produce a scatter plot of `income` versus `age` with this optimal spline fit superimposed.

```r
income_2007 <- read_csv("income_data.csv") |>
  filter(year == 2007) |>
  select(-year)
```