Lab 3: Cross Validation and Model Flexibility

Kaori Hirano

2024-09-06

Packages

```
library(plyr)
suppressPackageStartupMessages(library(tidyverse))
library(boot)
suppressPackageStartupMessages(library(gbm))
```

Data Simulation

```
# set seed for replicability purposes
set.seed(280)
# the number of observations in our data set
n <- 1000
# x1 ~ beta(1.1, 1.1) * 20 - 10 (constrained to between -10 and 10)
x1 <- rbeta(n, shape1 = 1.1, shape2 = 1.1) * 20 - 10


# Simulating n y1 values
y1 <- 2 + x1 - 0.012 * x1^2 + rnorm(n, mean = 0, sd = 1)
# And doing the same with y2
y2 <- 2 + sin(1.5 * sin(1.2 * x1) + 0.5 * cos(0.5 * (x1 - 1)^2)) + 0.3 * x1 +
  rnorm(n, mean = 0, sd = 1)


# putting all in a tibble
sim_data <- tibble(x1 = x1,
                   y1 = y1,
```

```
                    y2 = y2)
```

Models

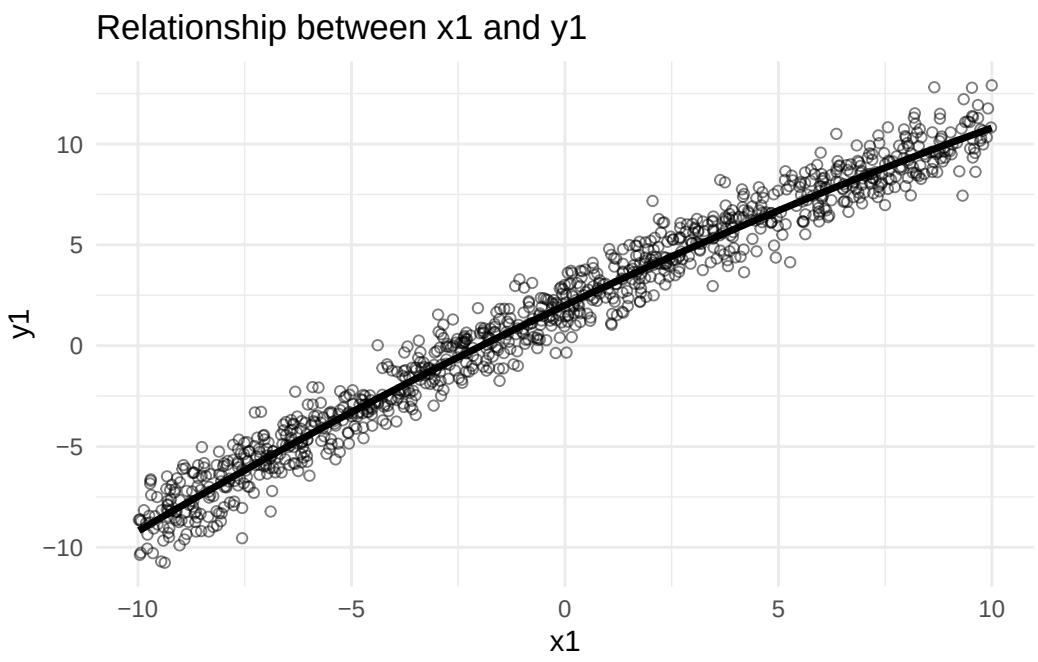
```
# function takes two arguments:
## true - the true Y values
## pred - the predicted Y values
## observations in true and pred *must* be in same order
calc_mse <- function(true, pred, subset = NULL){
  error <- (true - pred)^2
  if(!is.null(subset)) error <- error[subset]
  mean(error)
}
# note that R functions don't need return statement - last object evaluated in
# function body gets returned
```

Data Visualization

Q1

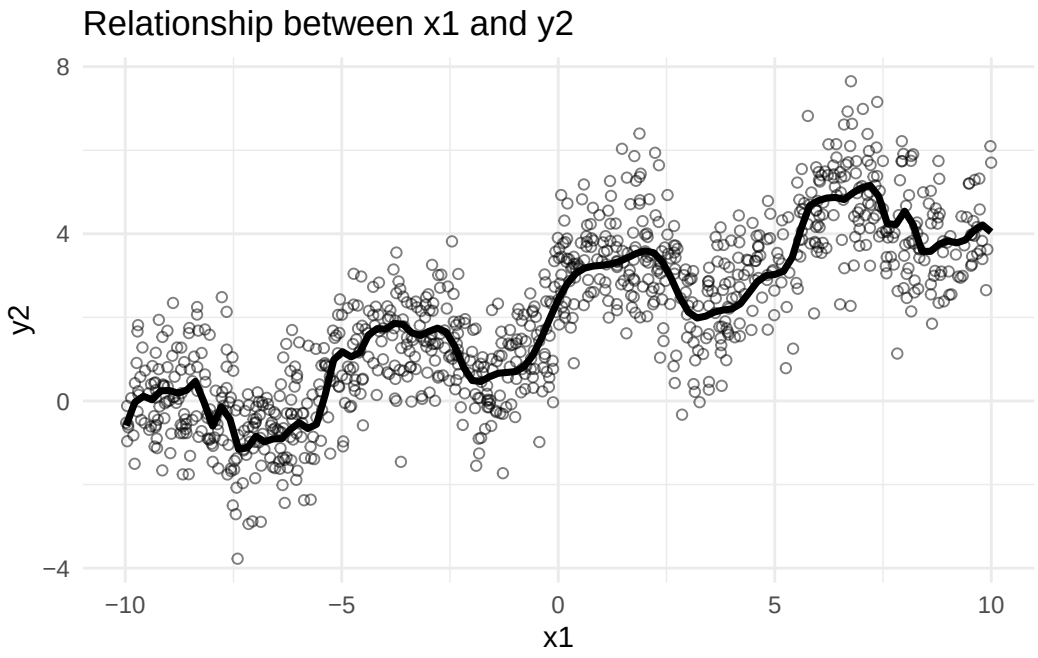
```
y1_fun <- function(x1){
  2 + x1 - 0.012 * x1^2
}

ggplot(sim_data, aes(x1, y1)) + # first graph, linear
  geom_point(shape = 1, alpha = .5) +
  labs(title = "Relationship between x1 and y1") +
  geom_function(fun = y1_fun, linewidth = 1.25) +
  theme_minimal()
```




```
y2_fun <- function(x1){ 2 + sin(1.5 * sin(1.2 * x1) + 0.5 *
                                  cos(0.5 * (x1 - 1)^2)) + 0.3 * x1
  }

ggplot(sim_data, aes(x1, y2)) + # second graph, nonlinear
  geom_point(shape = 1, alpha = .5) +
  labs(title = "Relationship between x1 and y2") +
  geom_function(fun = y2_fun, linewidth = 1.25) +
  theme_minimal()
```




```
# fix line width problem, describe two relationships
```

Validation Set Variance

Q2

```
set.seed(1)
# take 294 samples without replacement from the vector 1:392
train <- sample(392, 294) # assigns 25% to validation, rest to test
```

Q3

```
my1 <- glm(y1 ~ x1, data = sim_data, subset = train)

my2 <- glm(y2 ~ x1, data = sim_data, subset = train)

y1_linear_mse <- calc_mse(sim_data$y1, predict(my1, sim_data), -train)
```

```
y2_linear_mse <- calc_mse(sim_data$y2, predict(my2, sim_data), -train)
```

y1 linear mse = 1.090 y2 linear mse = 1.569

```
# function to calculate predicted values
predict_gbm <- function(y_train, x_train, x_val){

  df_test <- data.frame(y = y_train,
                        x = x_train)

  boost_model <- gbm(y ~ x,
                     data = df_test,
                     interaction.depth = 1,
                     n.minobsinnode = as.integer(.05 * length(y_train)),
                     n.trees = 500,
                     shrinkage = .1,
                     bag.fraction = 1,
                     distribution = "gaussian")

  df_val <- data.frame(x = x_val)

  boost_pred <- predict(boost_model, newdata = df_val,
                        n.trees = 500)
  boost_pred
}

# fitting the two gbm models
y1_gbm_pred <- predict_gbm(sim_data$y1[train],
                           sim_data$x1[train],
                           sim_data$x1[-train])

y2_gbm_pred <- predict_gbm(sim_data$y2[train],
                           sim_data$x1[train],
                           sim_data$x1[-train])

y1_gbm_mse <- calc_mse(sim_data$y1, y1_gbm_pred, -train)

y2_gbm_mse <- calc_mse(sim_data$y2, y2_gbm_pred, -train)
```

```
cbind(y1_linear_mse, y1_gbm_mse)
```

```
     y1_linear_mse y1_gbm_mse
[1,]      1.090235   67.39851
```

```
cbind(y2_linear_mse, y2_gbm_mse) # this doesn't seem right
```

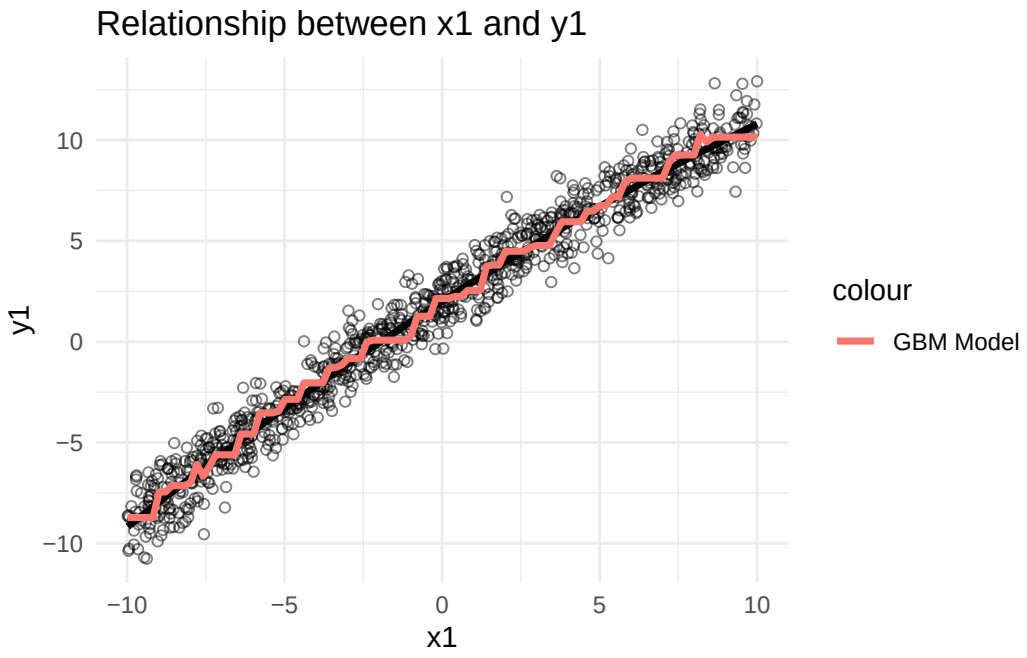
```
     y2_linear_mse y2_gbm_mse
[1,]      1.569458   7.114175
```

The linear model fits the data better in the first case. This is also true in the second case, although the gbm is closer to the linear model with the non-linear data.

Q4

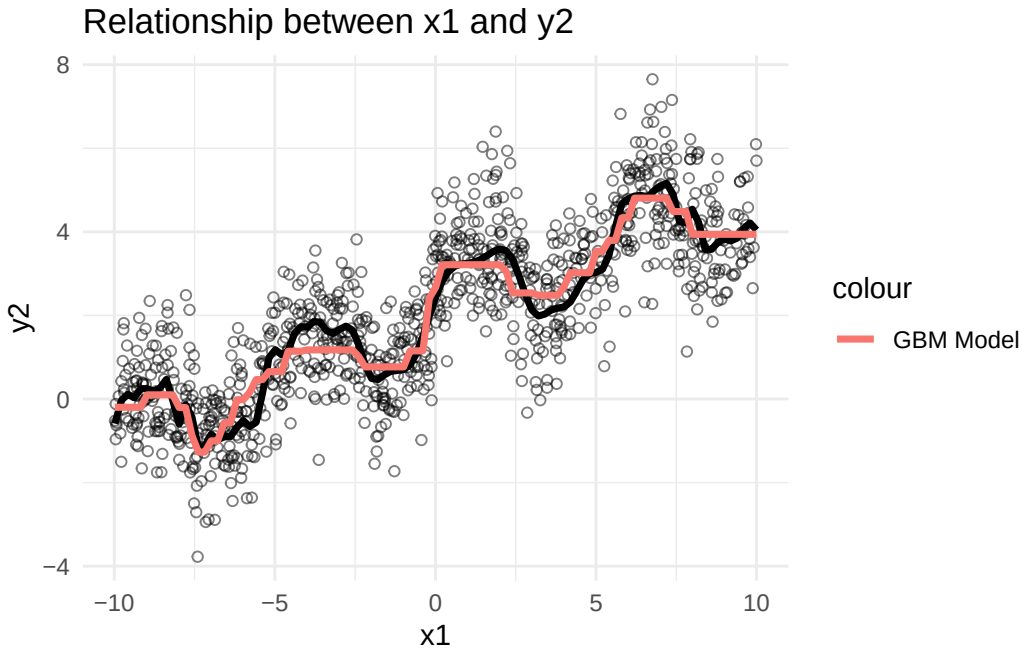
```
# remember to remove eval: true when you get to this chunk while working
# through the lab!

ggplot(sim_data, aes(x1, y1)) +
  geom_point(shape = 1, alpha = .5) +
  geom_function(fun = y1_fun, linewidth = 1.25, color = 'black') +
  labs(title = "Relationship between x1 and y1") +
  geom_function(aes(color = "GBM Model"),
                    fun = predict_gbm,
                args = list(x_train = sim_data$x1[train],
                            y_train = sim_data$y1[train]),
                linewidth = 1.25) +
  theme_minimal()
```




```
# adds true function

# nonlinear function
# drawing GBM predicted curve
ggplot(sim_data, aes(x1, y2)) +
  geom_point(shape = 1, alpha = .5) +
  geom_function(fun = y2_fun, linewidth = 1.25, color = "black")+
  labs(title = "Relationship between x1 and y2") +
  geom_function(aes(color = "GBM Model"),
                    fun = predict_gbm,
                args = list(x_train = sim_data$x1[train],
                            y_train = sim_data$y2[train]),
                linewidth = 1.25) +
  theme_minimal()  # adds true function line
```



The first one is far more reactive to small changes in the data than the linear model, which isn't necessarily a good thing. The second gbm is very similar to the second graph created in question 1.

Q5

```
set.seed(3)
# take 294 samples without replacement from the vector 1:392
train2 <- sample(392, 294) # assigns 25% to validation, rest to test
```

```
my12 <- glm(y1 ~ x1, data = sim_data, subset = train2)

my22 <- glm(y2 ~ x1, data = sim_data, subset = train2)

y1_linear_mse2 <- calc_mse(sim_data$y1, predict(my1, sim_data), -train2)

y2_linear_mse2 <- calc_mse(sim_data$y2, predict(my2, sim_data), -train2)
```

```
# function to calculate predicted values
predict_gbm <- function(y_train2, x_train2, x_val){
```

```
  df_test <- data.frame(y = y_train2,
                        x = x_train2)

  boost_model <- gbm(y ~ x,
                     data = df_test,
                     interaction.depth = 1,
                     n.minobsinnode = as.integer(.05 * length(y_train2)),
                     n.trees = 500,
                     shrinkage = .1,
                     bag.fraction = 1,
                     distribution = "gaussian")

  df_val2 <- data.frame(x = x_val)

  boost_pred <- predict(boost_model, newdata = df_val2,
                        n.trees = 500)
  boost_pred
}

# fitting the two gbm models
y1_gbm_pred2 <- predict_gbm(sim_data$y1[train2],
                            sim_data$x1[train2],
                            sim_data$x1[-train2])

y2_gbm_pred2 <- predict_gbm(sim_data$y2[train2],
                            sim_data$x1[train2],
                            sim_data$x1[-train2])

y1_gbm_mse2 <- calc_mse(sim_data$y1, y1_gbm_pred2, -train2)

y2_gbm_mse2 <- calc_mse(sim_data$y2, y2_gbm_pred2, -train2)
```

```
cbind(y1_linear_mse2, y1_gbm_mse2)
```

```
     y1_linear_mse2 y1_gbm_mse2
[1,]       1.091692    67.95313
```

```
cbind(y2_linear_mse2, y2_gbm_mse2) # this doesn't seem right
```

```
     y2_linear_mse2 y2_gbm_mse2
[1,]       1.556357    7.317884
```

For y1, the results for the glm are the same while the results for the gbm are .5 higher in this example. For y2, the linear model is the same while the gbm is .2 higher in this set. They don't tell an altogether different story.

K-Fold Cross Validation

Q6

Cross validation would help by further splitting up the data and having more train cases. It doesn't quite make sense that a linear model would be that much better at predicting nonlinear data than the gbm which should be better in the nonlinear data, so cross validation could help us see whether or not that is actually the case and will help us get a better picture of what fits the data better.

Q7

```
# remember to remove eval: true when you get to this chunk while working
# through the lab!

# gbm has bug when cross-validating with a single predictor
# so we are adding a non-informative predictor
# gbm throws a warning, but it's their bug!
set.seed(4)
sim_data$x2 <- 0

y1_linear_mse
```

```
[1] 1.090235
```

```
my12 <- glm(y1 ~ x1, data = sim_data) # refits with full data
glm_cv <- cv.glm(sim_data, my12, K = 10) # cross validation
glm_cv$delta
```

```
[1] 1.071642 1.071468
```

```r
y1_gbm_cv <- gbm(y1 ~ x1 + x2,
                 data = sim_data,
                 interaction.depth = 1,
                 n.minobsinnode = 5,
                 n.trees = 500,
                 shrinkage = .1,
                 bag.fraction = 1,
                 cv.folds = 10,
                 distribution = "gaussian")$cv.error[500]

y1_gbm_cv # mse
```

```
[1] 1.048797
```

a) The MSE for the GBM is 1.023 which is lower than that of the linear model for the same set of data so the gbm seems to fit better. This is only slightly lower than the linear one, however.

b) In this case, we don't care a lot about interpretability considering we haven't even named our variables beyond x and y. Because we aren't trying to make interpretations from the data and are just concerned about finding the best fit, favoring flexibility even though it takes longer is preferred in this case.