Lab 14: Statistical Prediction

Statistical Computing, 36-350

Week of Tuesday November 30, 2021

Name:
Andrew ID:
Collaborated with:

This lab is to be done in class (completed outside of class time if need be). You can collaborate with your classmates, but you must identify their names above, and you must submit **your own** lab as an knitted PDF file on Gradescope, by **Monday 9pm, next week**.

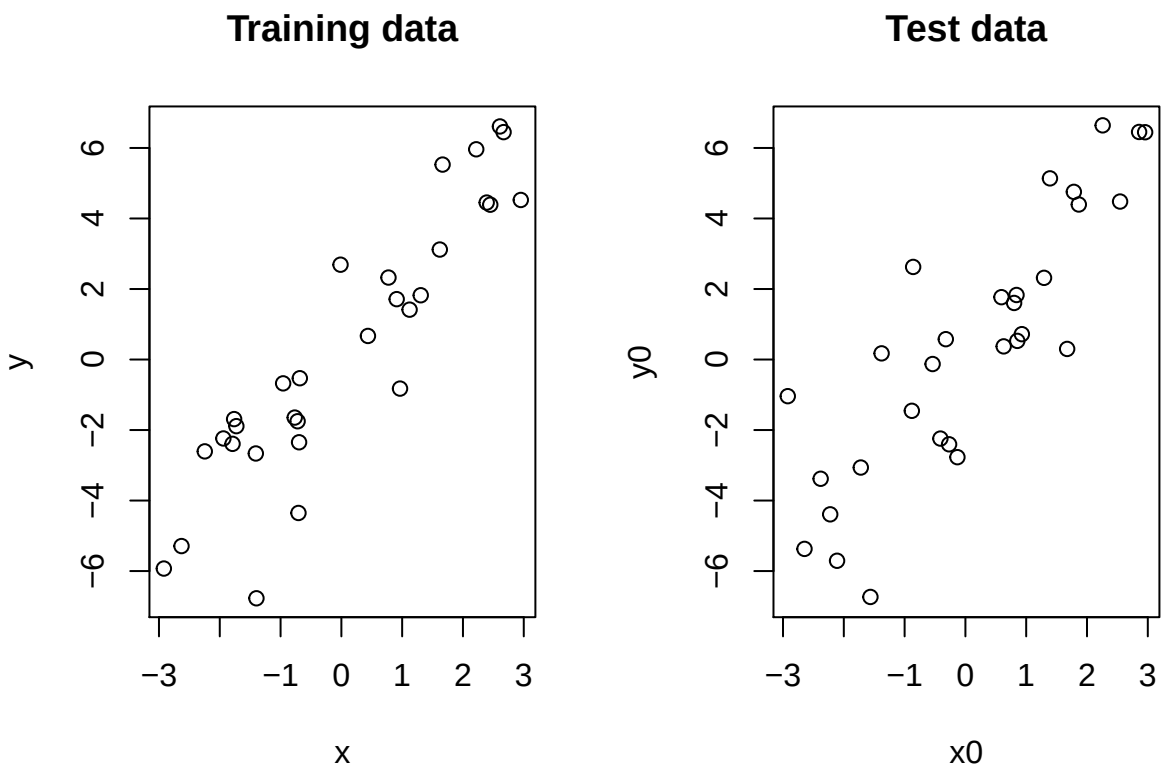
This week's agenda: understanding training and testing errors, implementing sample-splitting and cross-validation, and trying a bunch of statistical prediction methods (optional).

Q1. Practice with training and test errors

The code below generates and plots training and test data from a simple univariate linear model, as in lecture. (You don't need to do anything yet.)

```
set.seed(1)
n = 30
x = sort(runif(n, -3, 3))
y = 2*x + 2*rnorm(n)
x0 = sort(runif(n, -3, 3))
y0 = 2*x0 + 2*rnorm(n)

par(mfrow=c(1,2))
xlim = range(c(x,x0)); ylim = range(c(y,y0))
plot(x, y, xlim=xlim, ylim=ylim, main="Training data")
plot(x0, y0, xlim=xlim, ylim=ylim, main="Test data")
```



- **1a.** For every k in between 1 and 15, regress `y` onto a polynomial in `x` of degree k. Hint: look at the lecture to see how to use the `poly()` function. Then use this fitted model to predict `y0` from `x0`, and record the observed test error. Also record the observed training error. Plot the test error and training errors curves, as functions of k, on the same plot, with properly labeled axes, and an informative legend. What do you notice about the relative magnitudes of the training and test errors? What do you notice about the shapes of the two curves? If you were going to select a regression model based on training error, which would you choose? Based on test error, which would you choose?

```
# YOUR CODE GOES HERE
```

- **1b.** Without any programmatic implementation, answer: what would happen to the training error in the current example if we let the polynomial degree be as large as 29?
- **1c.** Modify the above code for the generating current example data so that the underlying trend between `y` and `x`, and `y0` and `x0`, is cubic (with a reasonable amount of added noise). Recompute training and test errors from regressions of `y` onto polynomials in `x` of degrees 1 up to 15. Answer the same questions as before, and notably: if you were going to select a regression model based on training error, which would you choose? Based on test error, which would you choose?

```
# YOUR CODE GOES HERE
```

Q2. Sample-splitting with the prostate cancer data

Below we read in the prostate cancer data set that we looked in previous labs.

```
pros.df = read.table(
  "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data")
dim(pros.df)
```

```
## [1] 97 10
```

```
head(pros.df, 3)
```

```
## lcavol lweight age lbph svi lcp gleason pgg45 lpsa train
## 1 -0.5798185 2.769459 50 -1.386294 0 -1.386294 6 0 -0.4307829 TRUE
## 2 -0.9942523 3.319626 58 -1.386294 0 -1.386294 6 0 -0.1625189 TRUE
## 3 -0.5108256 2.691243 74 -1.386294 0 -1.386294 7 20 -0.1625189 TRUE
```

- **2a.** As we can see, the designers of this data set already defined training and test sets for us, in the last column of `pros.df`! Split the prostate cancer data frame into two parts according to the last column, and report the number of observations in each part. On the training set, fit a linear model of `lpsa` on `lcavol` and `lweight`. On the test set, predict `lpsa` from the `lcavol` and `lweight` measurements. What is the test error?

```
# YOUR CODE GOES HERE
```

- **2b.** Using the same training and test set division as in the previous question, fit a linear model on the training set `lpsa` on `age`, `gleason`, and `pgg45`. Then on the test set, predict `lpsa` from the relevant predictor measurements. What is the test error?

```
# YOUR CODE GOES HERE
```

- **2c.** How do the test errors compare in the last two questions? Based on this comparison, what regression model would you recommend to your clinician friend? What other considerations might your clinician friend have when deciding between the two models that is not captured by your test error comparison?
- **Challenge.** The difference between the test errors of the two linear models considered above seems significant, but we have no sense of variability of these test error estimates, since it was just performed with one training/testing split. Repeatedly, split the prostate cancer data frame randomly into training and test sets of roughly equal size, fit the two linear models on the training set, and record errors on the test set. As a final estimate of test error, average the observed test errors over this process for each of the two model types. Then, compute the standard deviation of the test errors over this process for each of the two model types. After accounting for the standard errors, do the test errors between the two linear model types still appear significantly different?

Q3. Sample-splitting with the wage data

Below we read in the wage data set that we looked in previous labs.

```
wage.df = read.csv("http://www.stat.cmu.edu/~ryantibs/statcomp/data/wage.csv",
                   skip=16)
dim(wage.df)
```

```
## [1] 3000   11
```

```
head(wage.df, 5)
```

```
## year age sex maritl race education region
## 231655 2006 18 1. Male 1. Never Married 1. White 1. < HS Grad 2. Middle
Atlantic
## 86582 2004 24 1. Male 1. Never Married 1. White 4. College Grad 2. Middle
Atlantic
## 161300 2003 45 1. Male 2. Married 1. White 3. Some College 2. Middle
Atlantic
## 155159 2003 43 1. Male 2. Married 3. Asian 4. College Grad 2. Middle
Atlantic
## 11443 2005 50 1. Male 4. Divorced 1. White 2. HS Grad 2. Middle Atlantic
## jobclass health health_ins wage
## 231655 1. Industrial 1. <=Good 2. No 75.04315
```

```
## 86582 2. Information 2. >=Very Good 2. No 70.47602
## 161300 1. Industrial 1. <=Good 1. Yes 130.98218
## 155159 2. Information 2. >=Very Good 1. Yes 154.68529
## 11443 2. Information 1. <=Good 1. Yes 75.04315
```

- **3a.** Randomly split the wage data frame into training and test sets of roughly equal size. Report how many observations ended up in each half.

```
# YOUR CODE GOES HERE
```

- **3b.** On the training set, fit the following two models. The first is a linear model of `wage` on `year`, `age`, and `education`. The second is an additive model of `wage` on `year`, `s(age)`, and `education`, using the `gam` package. For the second model, plot the effects fit to each predictor, with `plot()`. Then, use each of these two models to make predictions on the test set, and report the associated test errors. Which model predicts better? What does that tell you about the nonlinearity that we allowed in the additive model for the `age` predictor?

```
# YOUR CODE GOES HERE
```

- **Challenge.** Sample-splitting can be done for logistic regression too, but it just requires us to be a bit more careful about how we choose the training and testing sets. In particular, we want to ensure that the ratio of 0s to 1s (assuming without a loss of generality that the response variable takes these two values) ends up being roughly equal in each of the training and testing sets. For the current wage data set, consider as the response variable the indicator that `wage` is above 250 (recall, this is measured in thousands of dollars!). Discard for the moment all observations that correspond to an education level of less than HS graduate (recall, the reason for this important step was explained in Lab 10f.) Then split the remaining observations into training and testing sets, but do so in a way that maintains equal ratios of 0s to 1s in the two sets, as best as possible. Once you have done this, fit two models on the training set. The first is a logistic model of `I(wage>250)` on `year`, `age`, and `education`. The second is an additive logistic model of `I(wage>250)` on `year`, `s(age)`, and `education`.

  Now, on the test set, use each of the two models to predict the probabilities that `wage` is above 250 for each of the test points. From these probabilities, produce the predicted outcomes—0s and 1s—according to the following rule: 0 if the probability is below π, and 1 if the probability is above π, where π is the observed proportion of 1s in the training set. From these predicted outcomes, compute and compare test errors. Note that test error, here, is just an average of the number of times we would have predicted the response (`wage` above or below 250) incorrectly. What does this tell you about the nonlinearity allowed in the second model for `age`?

```
# YOUR CODE GOES HERE
```

Q4. Cross-validation with the prostate cancer data

- **4a.** Let's revisit the prostate cancer data. Randomly split the prostate cancer data frame into $k = 5$ folds of roughly equal size. (Make sure your code is general enough to handle an arbitrary number of folds k; you will be asked to change the number of folds in questions that follow.) Report the number of observations that fall in each fold.

```
# YOUR CODE GOES HERE
```

- **4b.** Over the folds you computed in the previous question, compute the cross-validation error of the linear model that regresses `lpsa` on `lcavol` and `lweight`.

```
# YOUR CODE GOES HERE
```

- **4c.** Write a function `pros.cv()`, which takes three arguments: `df`, a data frame of prostate cancer measurements, with a default of `pros.df`; `k`, an integer determining the number of cross-validation folds, with a default of 5; and `seed`, an integer to be passed to `set.seed()` before defining the folds, with a default of NULL (meaning no seed shall be set). Your function should split up the given data `df`

into `k` folds of roughly equal size, and using these folds, compute the cross-validation error of the linear model that regresses `lpsa` on `lcavol` and `lweight`. Its output should simply be this cross-validation error.

```
# YOUR CODE GOES HERE
```

- **4d.** Investigate the result of `pros.cv()` for different values of `k`, specifically, for `k` equal to 2, 5, 10, and 97. For each value, run `pros.cv()` some large number of times (say, 50) and report the average of the cross-validation error estimates, and the standard deviation of these estimates. Then, plot them in an informative way (say, a box plot with `boxplot()`). What do you notice? Is this surprising?

```
# YOUR CODE GOES HERE
```

- **Challenge.** In general, is 2-fold cross-validation the same as sample-splitting? Why or why not?
- **Challenge.** In general, what can you say about the differences in cross-validation as the number of folds varies? What is different about cross-validation with 2 folds, versus 5 folds, versus n folds (with n being the number of observations in the data set)?
- **Challenge.** Modify your function `pros.cv()` so that it takes another argument: `formula.str`, a string in the format of a formula specifying which linear model is to be evaluated by cross-validation, with the default being "lpsa ~ lcavol + lweight". Demonstrate the use of your function for different formulas, i.e., different linear regression models.

```
# YOUR CODE GOES HERE
```

Q5. Making predictions on the HIV data set (optional)

Below, we read in some data on HIV from Rhee et al. (2003), "Human immunodeficiency virus reverse transcriptase and protease sequence database". There are 1073 observations of the following nature. The response variable (first column) is a measure of drug resistance, for a particular HIV drug. The 241 predictor variables (all but first column) are each binary indicators of the presence/absence of mutation at a particular gene mutation site. The goal is to predict HIV drug resistance from this genetic mutation information. (You don't have to do anything yet.)

```
hiv.df = read.table("http://www.stat.cmu.edu/~ryantibs/statcomp/data/hiv.dat")
dim(hiv.df)
```

```
## [1] 1073  241
```

```
hiv.df[1:5, c(1,sample(2:ncol(hiv.df),8))]
```

```
##           y p50 p239 p219 p135 p111 p20 p121 p152
## 1 14.612804   0    0    0    0    0   0    0    0
## 2 25.527251   0    0    0    0    0   1    0    0
## 3  0.000000   0    0    0    0    0   0    0    0
## 4  7.918125   0    0    1    0    0   0    0    0
## 5 11.394335   0    0    1    1    0   0    0    0
```

- **5a.** Use 5-fold cross-validation to estimate the test error of a linear model that regresses the drug resistance on all of the genetic mutation indicators. (You will likely get some warnings about the linear model encountering a rank-deficient fit: why do these occur?)

```
# YOUR CODE GOES HERE
```

- **5b.** Use 5-fold cross-validation to estimate the test error of a regression tree with the drug resistance measure as response and the genetic mutation indicators as predictors. To fit a regression tree, you can use the function `rpart()` from the package `rpart`. (Its notation is similar to `lm()` both for training and prediction.) In terms of prediction accuracy, does the regression tree improve on the linear model?

```
# YOUR CODE GOES HERE
```

- **5c.** Use 5-fold cross-validation to estimate the test error of a gradient boosted machine with the drug resistance measure as response and the genetic mutation indicators as predictors. To fit a gradient boosted machine, you can use the function `xgboost()` from the package `xgboost`. (This might require a bit of tinkering to set up; if you'd like a concrete place to start with the boosting settings, then you can try `max.depth=20`, `nround=10`.) In terms of prediction acccuracy, how does boosting fare, compare to a single tree?

```
# YOUR CODE GOES HERE
```

- **5d.** Implement your own function for k-nearest neighbors regression. Then, run 5-fold cross-validation to estimate test error, for a few choices of k. Discuss your findings in comparison to those for the linear model, regression tree, and boosting.

```
# YOUR CODE GOES HERE
```