Homework 3 - Classification

Instructions

2022-09-02

0.0.1 Instructions

- Use the space inside of

```
::: {.solbox data-latex=""}
```

```
:::
```

to answer the following questions.

- Do not move this file or copy it and work elsewhere. Work in the same directory.
- Use a new branch named whatever you want. Create it now! Can't come up with something, try here. Make a small change, say by adding your name. Commit and push now!
- Try to Knit your file now. Are there errors? Fix them now, not at 11pm on the due date.
- There MUST be some text between `::: {.solbox}` and the next `:::` or this will fail to Knit.
- If your code or figures run off the .pdf, you'll lose 2 points automatically.

1 The curse of dimensionality (From Chapter 4 of ISLR, 3 points)

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other *local* approaches that perform prediction using only observations that are *near* the test observation for which a prediction must be made. This phenomenon is known as the *curse of dimensionality*, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

1. (0.5 pts) Suppose that we have a set of observations, each with measurements on $p = 1$ feature, `X`. We assume that `X` is uniformly distributed on $[0, 1]$. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10 % of the range of `X` closest to that test observation. For instance, in order to predict the response for a test observation with `X = 0.6`, we will use observations in the range `[0.55,0.65]`. On average, what fraction of the available observations will we use to make the prediction?

Type your answer.

2. (0.5 pts) Now suppose that we have a set of observations, each with measurements on $p = 2$ features, `X1` and `X2`. We assume that `(X1,X2)` are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to predict a test observation's response using only observations that are within 10 % of the range of `X1` *and* within 10 % of the range of `X2` closest to that test observation. For instance, in order to predict the response for a test observation with `X1 = 0.6` and `X2 = 0.35`, we will use observations in the range `[0.55, 0.65]` for `X1` and in the range `[0.3, 0.4]` for `X2`. On average, what fraction of the available observations will we use to make the prediction?

Type your answer.

3. (0.5 pts) Now suppose that we have a set of observations on $p = 100$ features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10 % of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

Type your answer. You can use math: $x^2 + y^2 = \exp(27)$.

4. (0.5 pts) Using your answers to parts (1)–(3), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

Type your answer.

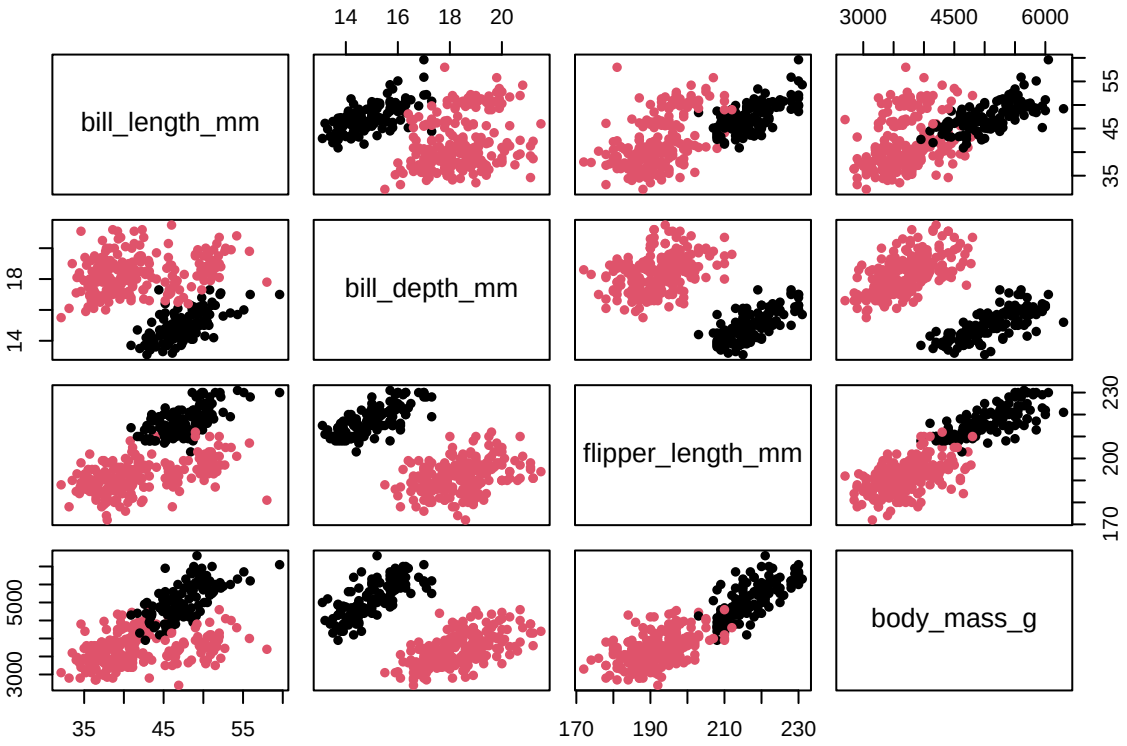
5. (1 point) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10 % of the training observations. For $p = 1, 2$, and 100, what is the length of each side of the hypercube? Comment on your answer. Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$, a hypercube is simply a line segment, when $p = 2$ it is a square, and when $p = 100$ it is a 100-dimensional cube.

Type your answer.

2 Simple classifiers for simple data (4 points)

The following code loads the famous Iris data and plots all the predictors against each other, coloring the points by class. It's not very interesting.

```
# Don't change me!!!
library(palmerpenguins)
data("penguins")
penguins <- penguins %>% # convert from 3 classes to 2 classes, drop some vars
  mutate(Gentoo = as.factor(case_when(species == "Gentoo" ~ "Gentoo", TRUE ~ "not Gentoo"))) %>%
  dplyr::select(Gentoo, bill_length_mm:body_mass_g)
penguins <- penguins[complete.cases(penguins),]
pairs(~ bill_length_mm + bill_depth_mm + flipper_length_mm + body_mass_g,
      data = penguins, col = penguins$Gentoo, pch = 16)
```



1. Estimate logistic regression using the `penguins` data. You'll get a warning.
   a. Explain what the warning **means**. If you just reword the message, you will get no points.
   b. Given the warning, how should we think about the solution (coefficients, confidence intervals, p-values, fitted values, etc.) that `R` computed?

```
# some code
```

a. Answer part a.

b. Answer part b.

2. Estimate LDA and KNN using the same `penguins` data. For KNN, examine `k=1:20` and select the best value using `knn.cv()` as in the lectures. Plot the CV "curve". Which K (or K's, if multiple) has the lowest CV error?

```
set.seed(202210) # don't change me!

# code for LDA

# code for KNN
```

Tell me which K has lowest error.

3. Suppose that you find that multiple K have the same CV Error. Describe the effects of using larger K or smaller K on bias and variance. If you prioritize a "smoother" decision boundary, which value should you use.

Some text.

4. Report the in-sample classification error for logistic regression, LDA, and KNN. For KNN, if there were multiple K that minimized CV, use the value that would give the smoothest classification boundary. Round your answer to reasonable precision.

```
# Some code.
```

3 Writing functions (3 points)

When working with Logistic Regression, we usually need some functions. Carefully review the Slides on Logistic Regression and the textbook to understand what these should do.

In each part below, you will see the body of a function and a number of tests. When writing functions, we should always be careful to make sure that they behave properly in predictable cases. (This idea is covered in great detail in CPSC 210. It is mentioned in DSCI 100 and STAT 301. And like with Git/Github, it discussed in detail in MDS and hugely important for package development.)

When you feel like your function is ready, turn the `eval=FALSE` label on the test chunk to `eval=TRUE`. The tests will run and put results in the file. If any fail, then you should revisit the tasks and try to fix them. Failing tests may prevent your document from knitting.

You may need to install the `{testthat}` package: `install.packages("testthat")`.

You should make your function produces errors if invalid inputs are passed. This is easiest to do with `stop()` or `stopifnot()`. Try examining the documentation for those functions. Looking at the tests should help you determine what sorts of arguments to check against.

1. (2 points) The logit function. This function takes in a probability and applies the "logit" transformation, mapping from $[0, 1]$ to $[-\infty, \infty]$. It is key to logistic regression. The transformation is $p \mapsto \log(p/(1-p))$.

```
library(testthat)
logit <- function(p) {

    # add code
}


# Don't change this chunk.
test_that("The logit function behaves properly", {
  expect_equal(logit(0.5), log(1))
  expect_error(logit(2))
  expect_error(logit("a"))
  expect_error(logit(c(1, -1)))
  expect_error(logit(c(2, 1)))
  a <- exp(1) / (1 + exp(1))
  expect_equal(logit(matrix(c(a, 1/2, 1/2, a), 2)), diag(2))
})
```

2. (1 point) The "logistic" function. This function is the inverse of the `logit`, mapping from $[-\infty, \infty]$ to $[0, 1]$. It is necessary to take predicted values from the `glm(..., family="binomial")` and turn them into probabilities. The transformation is $z \mapsto e^z/(1+e^z) = 1/(1+e^{-z})$.

```
logistic <- function(z) {

  # add code
}


# Don't change this chunk.
test_that("The logistic function behaves properly", {
  expect_equal(logistic(0), 1 / 2)
  expect_error(logistic("a"))
  a <- 1 / (1 + exp(-1))
  b <- 1 / (1 + exp(1))
  expect_equal(logistic(c(1, -1)), c(a, b))
  expect_equal(logistic(diag(nrow = 2, ncol = 2)),
               matrix(c(a, 1/2, 1/2, a), 2))
  expect_equal(logistic(-Inf), 0)
  expect_equal(logistic(Inf), 1)
})
```