Prof. Yiping Lu
IEMS 304: Statistical Learning for Data Analysis
May 19, 2025

Homework 6

This homework is to give a brief reminder of R, RStudio, and statistical topics covered in IEMS 303.

Note: The homework is scored out of 100 points. The problems add up to 90 points, while the remaining ten points will be graded according to a writing rubric, given at the end of the assignment.

R/RStudio installation If you have not installed R and RStudio, follow the installation instructions outlined in https://posit.co/download/rstudio-desktop/. You are strongly encouraged to use R Markdown to integrate text, code, images and mathematics or you can you use the latex code we provide.

Question 1. In an enzyme kinetics study, the velocity of a reaction (Y) is expected to be related to the concentration (X) as follows:

$$Y_i = \frac{\gamma_0 X_i}{\gamma_1 + X_i} + \epsilon_i.$$

Eighteen concentrations have been studied and the resulting $\{X, Y\}$ values for each are in the `HW6_data.xls`.

(a) In the optimization procedure for finding the nonlinear least squares estimates of γ_0 and γ_1, it helps to have good initial guesses. To obtain good initial guesses, notice that if we ignore the error term we have $Y_i' = \beta_0 + \beta_1 X_i'$, where $Y_i' = 1/Y_i, X_i' = 1/X_i, \beta_0 = 1/\gamma_0$, and $\beta_1 = \gamma_1/\gamma_0$.

Thus, you can fit a simple linear regression model to the transformed data (X_i' and Y_i') to obtain good initial guesses $\gamma_0 = 1/\hat{\beta}_0$ and $\gamma_1 = \hat{\beta}_1/\hat{\beta}_0$.

(b) Using the initial guesses from part (a), find the nonlinear least squares estimates of the parameters γ_0 and γ_1. Fit the nonlinear least squares model using R's nlm() function.

Question 2. For the same nonlinear model and data in Problem 1, we will use R to calculate bootstrap confidence intervals for the estimated parameters γ_0 and γ_1. You can use the boot() command in R (requires the boot package to be loaded with the library(boot) command) with at least 20,000 bootstrap replicates. If you use any built-in R functions to calculate the following, it is good to know what they are doing and how you would calculate the same quantities yourself.

(a) Calculate and plot bootstrapped histograms of $\hat{\gamma}_0$ and $\hat{\gamma}_1$, and calculate the corresponding bootstrapped standard errors. Interpret the standard errors as they relate to the uncertainty in the estimates from Problem 1.

(b) Calculate "crude" two-sided 95% CIs on γ_0 and γ_1 using the normal approximation to their bootstrapped distributions.

(c) Calculate the reflected two-sided 95% CIs on γ_0 and γ_1 (this corresponds to the type="basic" option of the boot.ci() function).

Question 3. Using the `spambase.txt` dataset, answer the following questions. You may refer to `spambase_info.txt` for the description of the available predictors and response variable. The binary response (spam or not spam) is included in the final column.

```
library(tree)
library(mgcv)
library(adabag)
spam <- read.table('spambase.txt', sep=',')
set.seed(1)
test_idx <- sample(1:nrow(spam), round(nrow(spam)/10))
```

1. Split the dataset into 90/10 training-testing set, using the given testing indices `test_idx`.
2. Report the fraction of spam emails in the training and testing sets respectively.
3. Train the following models using the training set:
   a. Fit a generalized linear model to predict if an email is spam. Which variables are significant below $\alpha = 0.01$?

   b. Fit a classification tree to train the data. Prune the tree by cross-validation (see hint). Include (i) a plot of the CV error vs. tree size, and (ii) a plot of the best tree. Which variables appear in the tree?

4. Predict if each email is a spam in the testing set.
   a. Report the misclassification rates for each method.
   b. Report the false negatives (what fraction of spam emails did not get classified as spam?)
   c. Report the false positives (what fraction of non-spam emails got classified as spam?)

Hint: The tree package contains functions `prune.tree` and `cv.tree` for pruning trees by cross-validation. The function `prune.tree` takes a tree you fit by `tree`, and evaluates the error of the tree and various prunings of the tree, all the way down to the stump. The evaluation can be done either on new data, if supplied, or on the training data (the default). If you ask it for a particular size of tree, it gives you the best pruning of that size. If you don't ask it for the best tree, it gives an object which shows the number of leaves in the pruned trees, and the error of each one. **This object can be plotted**.

```
# Fits tree
my.tree = tree(y ~ x1 + x2, data=my.data)
# Returns best pruned tree with 5 leaves, evaluating error on training
   data prune.tree(my.tree,best=5)
# Ditto, but evaluates on test.set prune.tree(my.tree,best=5,newdata=
   test.set)
# Sequence of pruned tree sizes/errors
my.tree.seq = prune.tree(my.tree)
# Plots size vs. error
plot(my.tree.seq)
# Vector of error rates for prunings, in order
my.tree.seq$dev
# Size of best tree
my.tree.seq$size[which.min(my.tree.seq$dev)]
```