BIOST 546
WINTER QUARTER 2021

Homework # 4
Due Via Online Submission to Canvas: Monday, Mar 8 at 10 AM

Instructions: You may discuss the homework problems in small groups, but you must write up the final solutions and code yourself. Please turn in your code for the problems that involve coding. However, code without written answers will receive no credit. To receive credit, you must explain your answers and show your work. All plots should be appropriately labeled and legible, with axis labels, legends, etc., as needed.

1. In this exercise, you will generate a simulated dataset and perform **non-linear regression**. Make sure you set the random seed with `set.seed(2)` before you begin.

   (a) Use the `runif()` function to generate a predictor X of length $n = 50$ with values in the interval $[-1, 1]$, and use the `rnorm()` function to generate a noise vector ϵ of length $n = 50$, mean 0 and standard deviation 0.1.

   (b) Generate a response vector Y of length $n = 50$ according to the model

   $$Y = f(X) + \epsilon,$$

   with $f(X) = 3 - 2X + 3 * X^3$.

   (c) Fit two smoothing spline models, $\hat{f}_{\lambda=1e-3}$ and $\hat{f}_{\lambda=1e-7}$, with `lambda=1e-3` and `lambda=1e-7`, respectively.

   Use the function `smooth.spline` introduced in class.

   (d) Make *one* plot that displays

   - A scatter plot of the generated observations with X on the x-axis and Y on the y-axis;
   - The true function f evaluated on a dense grid of values in the interval $[-1, 1]$;
   - The predicted functions $\hat{f}_{\lambda=1e-3}$ and $\hat{f}_{\lambda=1e-7}$ evaluated on a dense grid of values in the interval $[-1, 1]$ (use the function `predict` to produce the predictions).

   (e) Fit a smoothing spline models $\hat{f}_{\text{CV}}$ with a cross-validated choice of lambda.

   Use `smooth.spline` with `cv=TRUE`.

(f) Make *one* plot that displays

- A scatter plot of the generated dataset with X on the x-axis and Y on the y-axis;
- The true function $f(x)$ evaluated on a dense grid of values in the interval $[-1, 1]$;
- The predicted functions $\hat{f}_{\text{CV}}$ evaluated on a dense grid of values in the interval $[-1, 1]$ (use the function `predict`).

(g) Use $B = 1000$ datasets bootstrapped from the dataset generated in point (b) to estimate the variance of your predictions $\hat{f}_{\lambda=1e-3}(0)$ and $\hat{f}_{\lambda=1e-7}(0)$. In other words, for every bootstrapped dataset $b = 1, \ldots, 1000$, fit two smoothing spline models $\hat{f}^b_{\lambda=1e-3}$ and $\hat{f}^b_{\lambda=1e-7}$ and evaluate these functions at the point $X = 0$. Finally, compute the variance of $\{\hat{f}^b_{\lambda=1e-3}(0)\}_{b=1,\ldots,B}$ and $\{\hat{f}^b_{\lambda=1e-7}(0)\}_{b=1,\ldots,B}$. Comment on the results.

In the following exercise, you will work on a real dataset. You will perform binary classification on the Heart Disease Data Set in the file `heart.RData` (used in one of the R tutorials). The dataset contains multiple attributes. The goal is to predict the presence of heart disease in the patient (field `Disease`). You can find a description of the attributes in:
`https://archive.ics.uci.edu/ml/datasets/heart+disease`.

2. Tree models for Heart Disease prediction

(a) Describe the data: sample size n, number of predictors p, and number of observations in each class. Divide the data into a training set of 200 observations, and a test set; Set the seed with `set.seed(2)` before you sample the training set.

(b) Fit a simple classification tree to your data and display the estimated tree, which we will call the overgrown tree.

Note: You can use the function `tree`, just make sure the outcome is encoded as a `factor`.

(c) Compute the training and test misclassification error. Comment on the output.

To compute the predicted output, use the function `predict` as in the regression setting, but with the additional argument `type = "class"`

(d) Prune the overgrown tree and select the subtree that minimizes the cross-validated misclassification error. Plot the overgrown tree and highlight the branches (can be hand-drawn) of the selected subtree. Make a plot that displays the subtree size against the misclassification error that you get from the pruning procedure.

Note: Use `set.seed(2)` before calling `cv.tree` in order to get reproducible results. In `cv.tree` use the additional argument `FUN=prune.misclass` to indicate that you want the pruning to be driven by the misclassification

error. The *number of misclassified observations* is stored in the variable `dev` of the output of `cv.tree`.

(e) Compute the training and test misclassification error of the selected subtree and comment on the results.

(f) Now you decide to apply bagging to improve the performances of your tree model. Fit a Bagged Trees model to your training data and compute the training and test misclassification error. Briefly comment on the results.

Note: Set the seed with `set.seed(2)` before you run the bagged model.

(g) Fit a random forest model (with the number of randomly selected features $m = p/3$) to your training data and compute the training and test misclassification error. Briefly comment on the results.

Note: Set the seed with `set.seed(2)` before you run the random forest model.

(h) If you run the bagged and the random forest models multiple times on the same data (without fixing the seed) you obtain different results. Explain what are the sources of randomness in the two models.

(i) Fit a boosted tree model to your training data. Use the following parameters: 500 trees, 2 splits for each tree and a 0.1 shrinkage factor. Compute the training and test misclassification error. Comment on the results.

Note 1: Use the function `gbm` with the argument `distribution="bernoulli"` to indicate that this is a classification problem. The function `gbm` requires that the outcome variable be a number in $\{0, 1\}$ – you will need to create such variable from the categorical outcome.

Note 2: Use the function `predict` with the argument `type = "response"`. This will return the estimated probability of the outcome being 1. Use the Bayes rule to get the estimated labels from the estimated probabilities.

Note 3: Set the seed with `set.seed(2)` before you run the boosted model. We haven't seen in class what is the source of randomness in a boosted model, but if you are curious check the help of the function `gbm`.