BIOST 546
WINTER QUARTER 2020

Homework # 1 Key

1. For this problem, you will come up with some examples of statistical learning for biomedical or public health applications.

   (a) Provide three examples of regression problems motivated by biomedical research or public health. In each example, describe Y and $X_1, \ldots, X_p$ as well as the scientific task.

   (b) Provide three examples of classification problems motivated by biomedical research or public health. In each example, describe Y and $X_1, \ldots, X_p$ as well as the scientific task.

   (c) Provide three examples of unsupervised learning motivated by biomedical research or public health. In each example, describe $X_1, \ldots, X_p$ as well as the scientific task.

   (d) Provide three examples in biomedical research or public health for which *inference* (as opposed to prediction) is the goal. In each example, describe Y and $X_1, \ldots, X_p$ as well as the scientific task.

   (e) Provide three examples in biomedical research or public health for which *prediction* (as opposed to inference) is the goal. In each example, describe Y and $X_1, \ldots, X_p$ as well as the scientific task.

   Answers will vary, one example is given for each question.

   (a) The MIMIC dataset (https://mimic.physionet.org/) consists of observations on over 50,000 ICU stays. A research task associated with this dataset is to train a model which predicts length of ICU stay. In this case, Y would measure the length (in hours) of a patient's ICU stay. $X_1, \ldots, X_p$ would be features that include age, sex, physiological measurements, etc. Our outcome (hours of ICU stay) is continuous, so this is a regression problem.

   (b) Another research task that uses the MIMIC dataset is to predict mortality for each ICU stay. In this case, Y would be an indicator of whether or not a patient died during their stay. The variables $X_1, \ldots, X_p$ would be similar as above. The difference here is that our outcome is categorical/binary.

(c) Suppose we genome sequenced n patients, and were interested in clustering them based on their genetic data. Here, the variables $X_1, \ldots, X_p$ would correspond to gene expression measurements across p genes. We are interested in learning which of the subjects are genetically similar to each other, but we do not have a response variable Y here.

(d) Suppose we are interested in the association between patients' characteristics (e.g. socioeconomic status, age, insurance status, etc.) and the duration of their stay in ICU, inference would be the goal; Y and $X_1, \ldots, X_p$ would be the same as in (a). An example research question might be: do patients from lower socioeconomic status stay longer in the ICU compared to their counterparts?

(e) Both (a) and (b) are examples of prediction tasks.

2. Consider K-nearest neighbors for classification on a data set with $n = 800$ observations. There are two classes, the "blue" class and the "orange" class. Of the $n = 800$ observations, $n_1 = 350$ belong to the orange class and $n_2 = 450$ belong to the blue class.

   (a) Suppose we perform K-nearest neighbors with $K = 1$. What will the training error rate be? Explain your answer.

   The training error rate will be 0 since the closest neighbor to any data point would be itself.

   (b) Suppose we perform K-nearest neighbors with $K = 1000$. What will the training error rate be? Explain your answer.

   The training error rate will be $\frac{350}{800} = 43.75\%$: since $K = 1000 > n$, the algorithm will only all the training data and since there are more "blue" than "orange", every observation will be classified to be a "blue"

   (c) What value of K do we expect will result in the lowest bias? lowest variance? highest bias? highest variance? Explain your answers.

   We expect $K = 1$ to have the lowest bias and highest variance and $K = 800$ (or any $K \geq 800$) to have the highest bias and lowest variance. With $K = 1$, we are essentially interpolating through each data point, which provides a perfect fit (low bias) on the training data but high variance. On the other hand, with $K \geq 800$, we will always generate the same prediction (low variance) and large bias.

   (d) Now suppose we apply the K-nearest neighbors model, with $K = 1000$, to the training set, and we use the fitted model to classify a test observation. Will we classify this test observation to the blue class or to the orange

class, or is it impossible to know without more information? Explain your answer.

We will classify this observation to the blue class, because with $K = 1000$, the KNN fit will always predict the majority class which is blue.

3. In this exercise, you will analyze a classification data set of your choice (it must have $p \geq 2$). You can find the data online, through your research, on the book website, etc. Please use real data, not simulated data.

   (a) Describe the data. What are the values of n and p? How many classes are there, and what do they mean? What do the features mean? Where did you find the data?

   (b) Now, select 2 features for your analysis (since computer screens & paper are two-dimensional). You can choose those features at random from the full set of p features, or you can choose them based on your prior scientific knowledge. However, please do not choose them by performing a preliminary analysis of your data. Make a plot with the two features on the horizontal and vertical axes, and with all n observations displayed. The color of each observation should correspond to its class label. Make sure to include axis labels, an informative legend, etc.

   (c) Now perform K-nearest neighbors with a range of values of K, from 1 to $n/2$. For each value of K, recreate the plot from (b), but this time also color the background of the plot according to the class to which a test observation with that particular value of the features would be assigned. Basically, you are re-creating the right-hand side of Figure 2.14 of the textbook, but for the data that you have chosen (and you do not need to plot the black boundary, which can be a little finicky to display).

   (d) For each value of K in (c), what was the training error?

   (e) Based on the plots you created in (c), which value of K do you think will give the smallest expected test error? Explain your answer.

   (a) **We will be using the famous (or "infamous" after this class!!) Fisher's or Anderson's iris data set in `R`. This data set has 150 measurements $n = 150$ and 4 features ($p = 4$: sepal length, sepal width, petal length, and petal width). There are 3 classes and they correspond to 3 species of iris (Iris setosa, versicolor, and virginica).**

   (b) **We choose to use sepal length and petal length for our analysis.**

```
library(class)
data(iris)
# y: Species (setosa,versicolor,virginica)
```

```
# x1: Sepal.Length
# x2: Petal.Length
my_x1 <- iris$Sepal.Length
my_x2 <- iris$Petal.Length
my_y <- as.factor(iris$Species)

plot(x=my_x1,y=my_x2,
     col=my_y,pch=16,
     xlab='Sepal Length',
     ylab='Petal Length',
     main='Visualizing the distribution of
     selected features for the iris data')
legend(6.5,3,legend=c("Setosa", "Versicolor", "Virginica"),
       col = c('black','red','green'),cex=0.6,p=16)
```

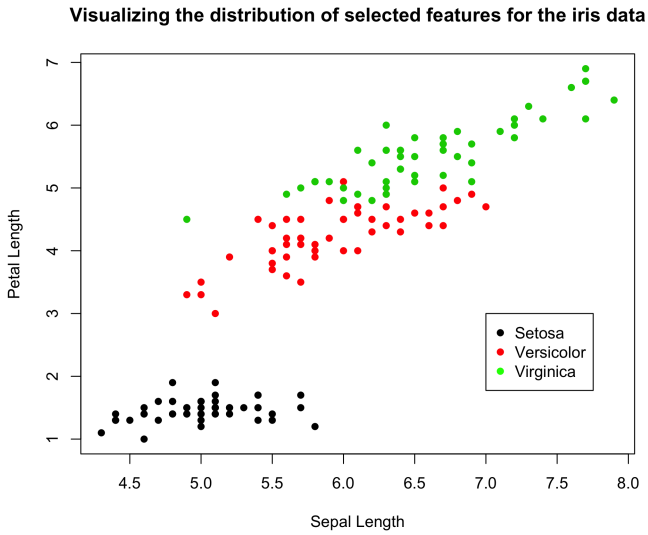


Figure 1: Figure for Q3(b)

(c) **We plot the decision boundaries for the sequence $K = 1, 11, 21, 41, 61$ below.**

```
# create a sequence of k's
k_seq <- c(1,11,21,41,61)
# create a grid of values for visualizing the decision boundaries
x1_grid <- seq(min(my_x1),max(my_x1),length=300)
x2_grid <- seq(min(my_x2),max(my_x2),length=300)
pred_grid <- expand.grid(x1_grid,x2_grid)
train_error_seq <- vector('list',length = length(k_seq))
library(scales) # load for plotting with transparency
for (i in seq_along(k_seq)){
```

```
k <- k_seq[i]
training_prediction <- knn(train=data.frame(my_x1,my_x2),
                                        test=data.frame(my_x1,my_x2),
                                        cl=my_y,
                                        k=k)

training_error <- mean(training_prediction!=my_y)
train_error_seq[i] <- training_error

decision_boundary_prediction <- knn(train=data.frame(my_x1,my_x2),
           test=data.frame(pred_grid$Var1,pred_grid$Var2),
           cl=my_y,
           k=k)
image(x=x1_grid,y=x2_grid,
          matrix(as.numeric(decision_boundary_prediction),
          nrow = length(x1_grid)),
          col = alpha(c('black','red','green'),0.4),
            xlab='Sepal Length', ylab='Petal Length',
      main =paste0('Visualizing the decision boundary
      for KNN on iris data, K = ',k) )
points(x=my_x1,y=my_x2,
     col=my_y,pch=16)
legend(7,3,legend=c("Setosa", "Versicolor", "Virginica"),
       col = c('black','red','green'),cex=1,pch =16)
       }
```

(d) **We plot the training error as a function of K below:**

```
plot(x=k_seq,y=unlist(train_error_seq),
     pch=16,
     xlab='Sepal Length',
     ylab='Training error',
     main='Training error for KNN as a function of K')
```

(e) **$K = 11$ would be our best guess: larger values K such as 21 and 41 give overly smooth boundaries that fail to classify some datapoints, while $K = 1$ has very "wiggly" boundaries between the green and blue classes, leading to high variance in the predicted test observations.**

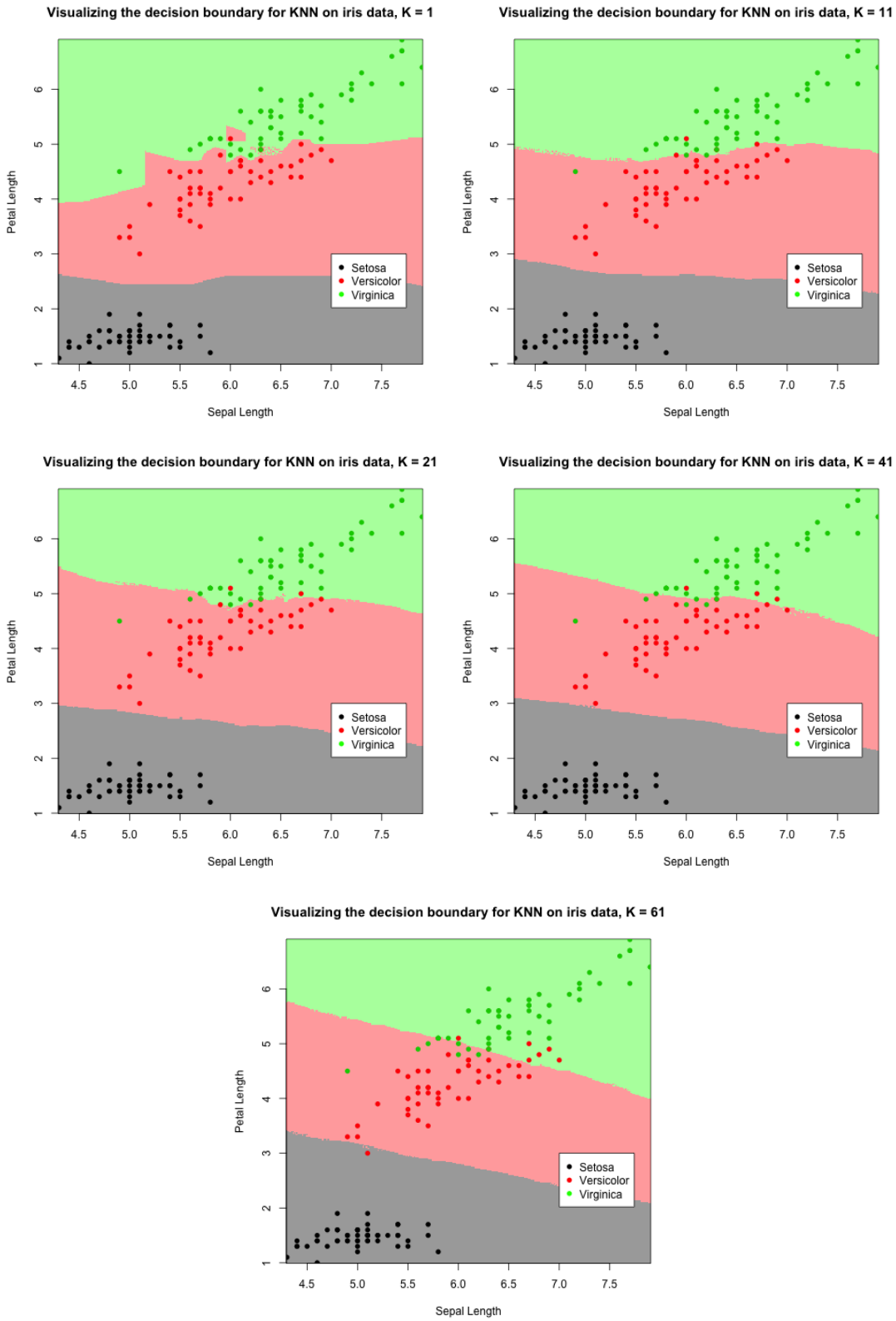


Figure 2: Figure for Q3(c)

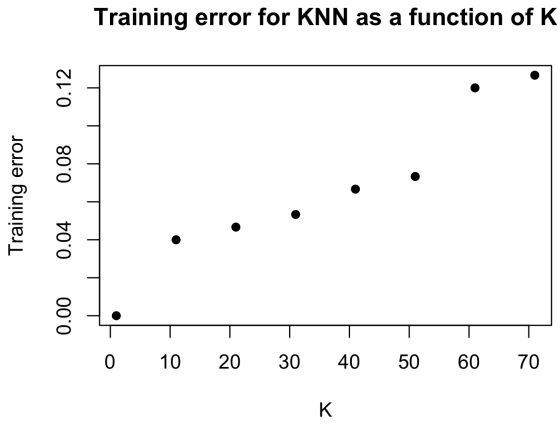


Figure 3: Figure for Q3(d)

4. Consider the regression model $Y = f(X) + \epsilon$ where ϵ is mean-zero noise term. Suppose we use some training data to fit this model using a variety of techniques, each of which results in a fitted model (which we will call $\hat{f}(x)$) and makes use of a different amount of "flexibility".

   (a) Make a plot with "flexibility" on the x-axis that displays the following curves:

   - the irreducible error
   - the variance of $\hat{f}(x)$
   - the squared bias of $\hat{f}(x)$
   - $E(y_0 - \hat{f}(x_0))^2$, where (x_0, y_0) is a test observation

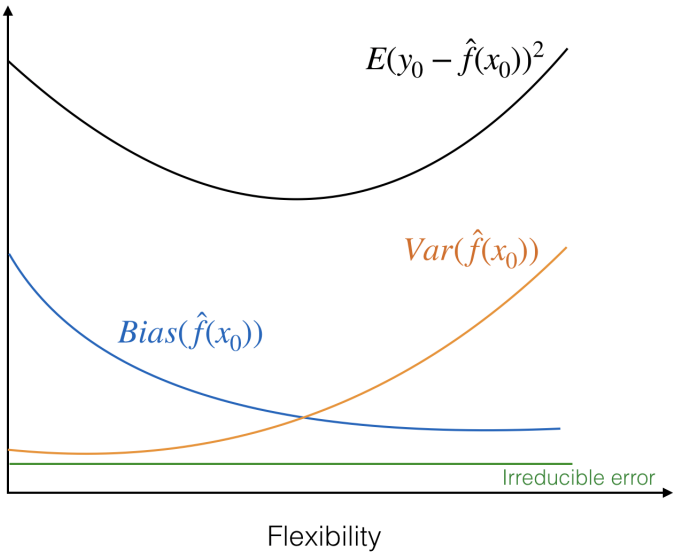
   You can make this plot in `R` or you can just sketch it by hand (and include a photo of the sketch in your assignment).

   (b) Comment on the plot, and explain how this ties into the bias-variance trade-off that we covered in class.

(a) An example plot is included below.

(b) The irreducible error clearly doesn't depend on model flexibility. In general, more flexible models have higher variance and lower (squared) bias, and less flexible models have lower variance and higher (squared) bias. Finally the expected squared loss of a test observation is just the sum of the squared bias, variance, and the irreducible error.

The plot in (a) illustrates the principle of bias-variance trade-off: when choosing among machine learning models, one should keep in mind that flexibility comes with a cost of higher variance; similarly an inflexible model might be more stable but will suffer



from higher bias. We will learn about a few principled ways to choose a model that balances the two in Chapter 5.

5. Suppose that you are interested in performing regression on a particular dataset, and need to decide whether to take a parametric or a non-parametric approach. Describe what properties of the dataset or the scientific context you might use to make this decision. What properties would lead you to *definitely* use a parametric approach? What properties would lead you to *definitely* use a non-parametric approach?

   If a certain parametric model is known *a priori* to be a good fit, e.g. we know that Y is normally distributed with mean that is a linear function of X, then we should always use the parametric model. This is usually not the case, however.

   Without prior knowledge, we should consider the sample size n and the number of predictors p. If p is very large and n is very small, then a parametric model is expected to perform better than a non-parametric model. Non-parametric models require large sample sizes to fit well, and can easily overfit with small n and large p.

   If n is very large and p is very small, then we expect a non-parametric model to perform better than a parametric model. Since we have a large sample size with small p, the non-parametric model is unlikely to overfit the data and will have less bias than a mis-specified parametric model.

6. This problem has to do with linear regression.

(a) For a simple linear regression, show that the least squares coefficient estimates take the form shown in (3.4) of the textbook. In other words, prove that the coefficients that minimize (3.3) satisfy the equation in (3.4). *(Hint: Use calculus!)*

(b) Now, write your own function in `R` to calculate $\hat{\beta}_0$ and $\hat{\beta}_1$ from (3.4) in the textbook.

(c) On some examples, show that the function that you wrote gives the same result as the `lm()` function in `R`.

(a) **We rewrite RSS as follows:**

$$RSS = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

Now taking derivatives with respect to β_0 and β_1 gives us:

$$\frac{\partial RSS}{\partial \hat{\beta}_0} = \sum_{i=1}^{n} -2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)$$

$$\frac{\partial RSS}{\partial \hat{\beta}_1} = \sum_{i=1}^{n} -2x_i(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)$$

Recall that the derivatives should be 0 at the $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimizers RSS:

$$\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0 \implies \sum_i x_i \cdot \hat{\beta}_1 + n \cdot \hat{\beta}_0 = \sum_i y_i$$

$$\sum_{i=1}^{n} x_i(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0 \implies \hat{\beta}_1 \sum_i x_i^2 + \hat{\beta}_0 \sum_i x_i = \sum_i x_i y_i$$

Solving for $\hat{\beta}_0$ in the first equation gives us:

$$\hat{\beta}_0 = \frac{\sum_i y_i - \hat{\beta}_1 \sum_i x_i}{n} = \bar{y} - \hat{\beta}_1 \bar{x}$$

Now substitute the expression of $\hat{\beta}_0$ into the second equation:

$$\hat{\beta}_1 \sum_i x_i^2 + (\bar{y} - \hat{\beta}_1 \bar{x}) \cdot \sum_i x_i = \sum_i x_i y_i$$

$$\hat{\beta}_1 = \frac{\sum_i x_i y_i - \bar{y} \sum_i x_i}{(\sum_i x_i^2 - \bar{x} \cdot \sum_i x_i)}$$

Use $\sum_i x_i = n\bar{x}$ gives us

$$\hat{\beta}_1 = \frac{\sum_i (x_i y_i - \bar{x}\bar{y})}{\sum_i (x_i^2 - \bar{x}^2)}$$

. Now this is really close to (3.4), we just need to show that

$$\sum_i (x_i y_i - \bar{x}\bar{y}) = \sum_i (x_i - \bar{x})(y_i - \bar{y})$$

and

$$\sum_i (x_i^2 - \bar{x}^2) = \sum_i x_i - \bar{x}^2)$$

. We will just show the first one since one can replace y_i with x_i and the first euqality would imply the second. Now

$$\sum_i (x_i - \bar{x})(y_i - \bar{y}) = \sum_i (x_i y_i - \bar{x} y_i - x_i \bar{y} + \bar{x}\bar{y}) - \sum_i x_i y_i - \bar{x} \sum_i y_i - \bar{y} \sum_i x_i + n\bar{x}\bar{y}$$

$$= \sum_i x_i y_i - n\bar{x}\bar{y} = \sum_i (x_i y_i - \bar{x}\bar{y})$$

(b) **A possible way of coding this up in R:**

```
my_lm <- function(x,y){
  # my_lm is a function that outputs the linear regression
  # (OLS) estimates of y=b_0+b_1*x
  # Input:
  # two vectors: feature x, response y
  # Output:
  # regression coefficient b_1 and intercept b_0
  x_bar <- mean(x)
  y_bar <- mean(y)
  b_1 <- (sum((x-x_bar)*(y-y_bar)))/(sum((x-x_bar)^2))
  b_0 <- mean(y)-mean(x)*b_1
  return(c(b_0, b_1))
}
```

(c) **We compared the coefficients on the Boston data set in the MASS library**

```
library(MASS)
data(Boston)
lm.fit <- lm(medv~lstat, data=Boston)
my_lm.fit <- my_lm(x=Boston[,'lstat'], y=Boston[,'medv'])
coef(lm.fit)
# (Intercept)       lstat
# 34.5538409  -0.9500494
my_lm.fit
# 34.5538409 -0.9500494
```