## Introduction and Homework Focus

This homework is concerned with material from Chapters 1-7 of the textbook *and* Bayesian statistics. It focuses on the bread-and-butter methodology of applied machine learning: setting up testing and training datasets, fitting a variety of models, then using tools like cross-validation to choose a model to use.

## 1. Empirical Study: Cross-Validation Method 1

Load the *mnist.csv* dataset. This is exactly the training dataset that you used in HW1, where you fit several knn models to this dataset and found a "best" value of $k$. In this problem, you will estimate the error of this procedure using cross-validation:

(1) Do 3-fold cross-validation, using this single value of $k$ for the knn model. Report the estimated error.
(2) Open HW1, where you trained on the same dataset and found the test error. How do these results compare? How would you expect them to compare in general?

**Note:** Doing this question *and* the question from HW1 is a little bit silly. It is intended as a bit of a warmup for cross-validation.

## 2. Empirical Study: Cross-Validation Method 2

Download the red wine dataset from: https://archive.ics.uci.edu/ml/datasets/wine+quality. Your goal in this question is to predict wine quality based on all other variables.

(1) Split the data set into a training set and a test set.
(2) For each of the following, fit the given model, report any hyperparameters chosen, and report the test error.
    (a) Fit a linear model using least squares.
    (b) Fit a ridge regression model on the training set, with hyperparameter $\lambda$ chosen by cross-validation on the training data.
    (c) Fit a lasso model on the training set, with hyperparameter $\lambda$ chosen by cross-validation.
    (d) Use *regsubsets* to choose the best linear model using forward and backward stepwise regression.
(3) Comment on the results obtained. How accurately can we predict wine quality? Is there much difference among the test errors resulting from these approaches? Are the models themselves similar?

## 3. Simulation Study: Cross-Validation Method

Do Question 8 from Chapter 5 of the textbook.

## 4. Simulation Study: Screening, Stepwise Selection, and ROC Curves

The following toy problem is motivated by the interesting (but very difficult) problem of using genome-wide association studies (GWAS) to screen people for rare diseases with genetic components. [2]

(1) Load the data *GWAS.csv*, and split it into equal-sized testing and training datasets. The first column is the indicator function of a disease. The remaining columns are indicator functions for a collection of alleles that have been studied. Fit a standard logistic regression model to the training set and apply to the testing set. Summarize the model fit and the performance of the model on the test data.
(2) Repeat the previous step, but this time use Lasso or ridge regression. Compare the results.
(3) Draw an estimate of the ROC curve for the *best* method you used. Describe any interesting features.
(4) You wish to use your model for screening. This means measuring a small number of variables, then forwarding a small fraction of people for further testing. Based on funding considerations, you wish to forward roughly 1 percent of the population for further testing. Describe a decision procedure based on the model you used in the previous step. Does the plotted ROC curve influence your choice?

## 5. Bayesian Workflow: Correcting Regression Coefficients

Recall that, in our introduction to Bayesian statistics, we developed an "empirical Bayesian" method to *correct* estimated incidence rates for kidney cancer when we had a very large number of counties. In this question, we will apply the same idea to a much smaller dataset.

I begin by describing the dataset and the problem. Load the dataset *SchoolCorrect.csv* and display some parts of it. The rows correspond to student, and each student $i$ comes with the following information:

(1) The school they belong to (from 1 to 6), which I will denote $S_i$.
(2) The student's test score on a standardized test *after* some standardization/centering, which I will denote $T_i$.

To set notation, I will denote by $N(j) = \{i : S_i = j\}$ the students in school $j$.

The scientific questions we will try to answer are: **which school is the "best,"** and **how confident are we in our comparisons?**

[2] See *e.g.* the recent survey *A scientometric review of genome-wide association studies* for a list of several thousand experiments in this area.

(1) We will start with naive estimates. For each school $j \in \{1, 2, \ldots, 6\}$, fit the simple model

$$T_i \sim N(\beta_j, 5^2), \quad i \in N(j).$$

This model has 1 parameter, $\beta_j$, and there are 6 schools, so in total you should be estimating $(1)(6) = 6$ parameters in this part of the question.[3]

Finally, *rank* the schools according to the magnitude of $\beta_j$.

(2) We will next use Bayesian methods to compute "corrected" estimates. Observe that $|N(j)|$ varies a great deal - some schools are much bigger than others. We define the following Bayesian model and priors:[4] [5]

$$\begin{aligned} \sigma &\sim \text{Unif}[1, 50] \\ \beta_j &\overset{i.i.d.}{\sim} N(0, \sigma^2), \quad j \in \{1, 2, \ldots, 6\} \\ T_i &\sim N(\beta_j, 5^2). \end{aligned}$$

Using Markov chain Monte Carlo or otherwise, sample from the posterior distribution and compute credible sets for $\beta_1, \ldots, \beta_6$. **Hint:** Conditional on $\sigma$, you have already seen how to sample from the posterior on each of $\beta_1, \ldots, \beta_6$. Using this fact may allow you to write a sampler that is both simpler and better than the "usual" Metropolis-Hastings algorithm.

(3) Using your samples from the posterior distribution, rank the 6 schools by the expected value of $\beta_i$. Compare this ranking to the naive estimate from part 1 of the question.

(4) Posterior distributions can be used to describe *uncertainty* in a ranking. Compute the probability (according to the posterior distribution) that your estimate of the top-ranked school is *really* the top-ranked school.

---

[3] As always, it is slightly silly to assume that the noise is known. Unfortunately, the simple algorithm I gave in class becomes quite inefficient when the number of parameters grows, and I don't want to focus too much on these issues. If you have learned *STAN* or another method that is more efficient in higher dimensions, please feel free to fit a more realistic model.

[4] This is similar to the example from class, except that we have one new "level" of parameters: the top equation for $\sigma$. This extra level, often called a "hyperparameter," is necessary because we don't have enough data to nail down $\sigma$ precisely. In class we had many thousands of subpopulations, rather than 6 as in this question.

[5] I chose this particular hierarchical model because it is easy to write down. If you are interested in hierarchical models and would like to try a more serious choice, please just indicate this on your homework. I am happy to provide references on Piazza.

## 6. Simulation Study: Bias-Variance Tradeoff for K-NN Regression

We explore the bias-variance tradeoff for K-NN regression by doing a simulation study. Along the way, we will practice simulating random variables, numerical integration, and using confidence intervals. The question is fairly long, but most of this is setting notation.

We set some notation. Throughout this question, we will be attempting to learn the function $f(x) = \sin(x)$ with domain $[0, 2\pi]$. Furthermore, we will assume that there is no "measurement error" - in any sample, a point $x$ will always be paired with the correct value $f(x)$. For any sample $X = (X_1, \ldots, X_n)$, define $\hat{f}_{k,X}$ to be the $k$-NN regression estimator associated with the sample $X$ (see Section 3.5 of the textbook for the definition).

Consider $N$ samples of size $n$ from $[0, 2\pi]$:

$$X_1^{(j)}, \ldots, X_n^{(j)} \overset{i.i.d.}{\sim} \text{Unif}[0, 2\pi],$$

and let $X^{(j)} = (X_1^{(j)}, \ldots, X_n^{(j)})$. Define the sample mean, sample squared-bias, sample variance and sample mean-squared error by

$$\begin{aligned}
\text{Mean}_k(x) &= \frac{1}{N}\sum_{j=1}^{N} \hat{f}_{k,X^{(j)}}(x) \\
\text{Bias}_k^2(x) &= (\text{Mean}_k(x) - f(x))^2 \\
\text{Var}_k(x) &= \frac{1}{N-1}\sum_{j=1}^{N} (\hat{f}_{k,X^{(j)}}(x) - \text{Mean}_k(x))^2 \\
\text{MSE}_k(x) &= \frac{1}{N}\sum_{j=1}^{N} (\hat{f}_{k,X^{(j)}}(x) - f(x))^2.
\end{aligned}$$

Define the *averaged* versions as the average with respect to $x$, so that e.g.

$$\text{MSE}_k = \frac{1}{2\pi}\int_0^{2\pi} \text{MSE}_k(x). \tag{6.1}$$

(1) Fix $n = 25$ and $N$ large. Simulate samples and then make the following three plots: $k$ vs $\text{Bias}_k^2$, $k$ vs $\text{Var}_k$, and $k$ vs $\text{MSE}_k$.

**Note 1:** Please read the full question before doing this. You will want to choose values of $N$, $k$ that give good results in the following parts of the question.

**Note 2:** The integral in (6.1) is not intractable, but it is annoying. If you would prefer, you can approximate it using the quadrature or

Monte Carlo methods discussed in class, or using any other technique you would prefer.

(2) In the previous question, you should find that the sample MSE is large when $k$ is too small *and* when $k$ is too big, obtaining a minimum somewhere in the middle. Find the optimal value of $k$. Also give an informal argument as to why your choices of $N, k$ in the previous question are likely "good enough" to have confidence in your answer.

**Hint:** Notice that $\text{MSE}_k$ is random (it depends on the sample $X_1^{(j)}, \ldots, X_n^{(j)}$), and in fact it is an average of a collection of i.i.d. random variables. The true distribution of these random variables is a little complicated, but for the purposes of these question you can assume it is a nice random variable and use the confidence intervals that you learned in an earlier class.

(3) Repeat the optimization procedure in parts (1,2) of this question, this time with $n = 100$. Does the value of $k$ change? In what direction does it go? In light of this evidence, predict what would happen if you did the same procedure with $n = 10000$.

## 7. Empirical Study: LASSO vs. Regression Workflow

In this study, we will try to predict rental house prices. Open the datasets *ListingsTrain.csv*, *ListingsOptimization.csv* and *ListingsTest.csv*, obtained from `insideairbnb.com` and then slightly tweaked for this question. You will try to predict the listing prices from the other variables.

(1) Explore the numerical values in the training dataset using your favourite tools - *summary*, *pairs*, *hist*, or anything else you like. Clean up the data by dealing with NAs and converting non-numerical data to numerical data when appropriate (though see part 4 of this question before doing so). Comment on any relationships and any possibly-bad datapoints.

**Hint 1:** The "price" variable is a sequence of characters, not a numeric variable. You should definitely fix that. Many categorical variables (especially true/false ones) have only a small number of values; those should also be converted into numerical variables where possible.

**Hint 2:** Some features are pretty useless. It is fine to simply remove a column if it look useless, or almost all the data is missing or obvious bad. Simply explain what your choices are (and don't do this to all the features).

(2) You should have found some suspicious datapoints in the first part of this question. Create two copies of the dataset: one with the suspicious points removed, the other with them retained. Try training both LASSO and standard regression models on the training dataset using the numerical covariates, optimizing the parameters using cross-validation with the optimization dataset. In the end you should have four models: standard and LASSO regression, with and without suspicious points. Test their accuracy on the test dataset.
(3) Which of the models in the previous question is best? Was it OK to remove the suspicious datapoints from the training dataset? Were the error estimates from the optimization dataset accurate?
(4) Some of the features, such as "description" and "amenities," are text fields or text lists. Choose a collection of words $W$ that appear somewhere in these variables. For each word $w \in W$, create a $\{0, 1\}$-valued dummy variable that indicates the presence of a given word in a given listing's text field. For a word $w$, denote by $p(w)$ the percentage of listings containing that word. Denote by $n$ the number of listings in the training set. Ensure that your collection of words $W$ satisfies:
   (a) $\sum_{w \in W} p(w) \geq \frac{n}{2}$ (that is, the words appear in some nontrivial fraction of all listings).
   (b) $|W| \geq 12$ (that is, there are many words in the collection).
   (c) Very common words such as "the" or "and" should not appear in the collection.

   In order to extract words from a text field in R, you may find the the command *grepl* useful.
(5) Repeat model-training for both standard and LASSO regression on this larger collection of covariates (with outliers removed or not per the results of the previous part of the question). How do the results change?

## 8. Theory: LASSO, Ridge and Variable Selection

In this question, you will consider the "ridge" and "LASSO" penalties for the usual linear regression model:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i.$$

(1) Show that, for *any* dataset $(X_1, Y_1), \ldots, (X_n, Y_n)$, there exists *some* finite value $\Lambda$ so that for all choices of hyperparameter $\lambda > \Lambda$, the coefficients estimated by LASSO regression will all be exactly 0.

(2) Show that the above is *not* true for Ridge regression. That is, show that there exists some dataset $(X_1, Y_1), \ldots, (X_n, Y_n)$ so that for *any* finite value of hyperparameter $\lambda$, the coefficients estimated by ridge regression will *not* all be exactly 0. **Hint:** the easiest way to do this is to just try a few concrete small datasets - $n = 5$ should be fine.

## 9. Bayesian Workflow: Latent Parameters

In this problem, you will use a Bayesian latent parameter model to simultaneously *classify* datapoints and *estimate* model parameters.

Load the two datasets *BrazilPrices.csv* and *UsaPrices.csv*. These two datasets give the rental prices of airbnb locations in Rio de Janeiro and San Francisco respectively. The data was taken from `insideairbnb.com` but then edited to make this question easier; for that reason the results will be much better than you could expect for the real real data. See the end of the document for details on what was done; you can use this to try the same method on the real dataset.

Denote by $X_1, \ldots, X_n$ the Brazilian prices, $Y_1, \ldots, Y_m$ the American prices, and $Z_1, \ldots, Z_{m+n}$ the combined list.

(1) Load the data and fit the simple Gaussian model

$$X_i \overset{i.i.d.}{\sim} N(\mu_0, \sigma_0^2), \quad Y_i \overset{i.i.d.}{\sim} N(\mu_1, \sigma_1^2).$$

What do you observe? Are the two datasets well-separated?

(2) Let's pretend that we *knew* that some listings were from San Francisco and others from Rio de Janeiro, but *didn't* get to observe which listing was from which city. Create a new vector that concatenates the two datasets to represent this. We will see that we can effectively learn $\mu_0, \mu_1$ anyway, by simultaneously estimating these two parameters *and* the city of every single listing. [6]

More precisely, we will fit the following model for the combined price vector $Z_1, \ldots, Z_{n+m}$:

$$\begin{aligned} \mu_0 &\sim N(30, 12^2) \\ \mu_1 &\sim N(150, 30^2) \end{aligned} \tag{9.1}$$

[6] We won't estimate $\sigma_0, \sigma_1$ due to some subtle degeneracy issues that arise in many classification applications. The basic issue is as follows: if you allow for arbitrary variances, you can get likelihood values to become arbitrarily large by making one "class" that has only a single datapoint but variance that is extremely close to 0. Solutions for this problem are well-known but beyond the scope of this course. Interested readers can probably find and explore these degenerate answers very quickly based on their solutions to this problem.

$$C_i \overset{i.i.d.}{\sim} Bern(0.5), \qquad i \in \{1, 2, \ldots, n+m\}$$
$$Z_i \sim N(\mu_{C_i}, 85^2), \qquad i \in \{1, 2, \ldots, n+m\}.$$

Note that this model has $m+n+2$ parameters $\mu_0, \mu_1, C_1, \ldots, C_{m+n}$ for $m+n$ datapoints $Z_1, \ldots, Z_{m+n}$. The parameters $C_1, \ldots, C_{m+n}$ are often called "latent" variables. They are distinguished from the "usual" parameters $\mu_0, \mu_1$ by the fact that they could in principle be measured directly - we simply don't have access to the measurements.

As a first step in fitting this model, write down a formula for the conditional probability distributions:

$$f_{a,b,i}(c) \equiv P[C_i = c | \mu_0 = a, \mu_1 = b], \qquad c \in \{0,1\}, a, b \in \mathbb{R} \tag{9.2}$$
$$g_{c,0}(A) \equiv P[\mu_0 \in A | (C_1, \ldots, C_{n+m}) = c], \qquad c \in \{0,1\}^{n+m}, A \subset \mathbb{R}$$
$$g_{c,1}(A) \equiv P[\mu_1 \in A | (C_1, \ldots, C_{n+m}) = c], \qquad c \in \{0,1\}^{n+m}, A \subset \mathbb{R}$$

**Note:** you don't need to be able to compute from this formula by hand - but you will need to be able to implement it on a computer in the next question. **Note 2:** the functions $g_{c,0}$ and $g_{c,1}$ are nearly identical, and you may be able to find a single formula that incorporates both functions.

(3) Implement code to sample from the distributions in Equation (9.2). The beginning and end of the first function should be:

```
f<-function(a,b,zi) {
...
return(c)}
```

where $a, b \in \mathbb{R}$ and $c \in \{0,1\}$. The beginning and end of the second function should be:

```
g0<-function(c,z) {
...
return(a)}
```

where $c \in \{0,1\}^{n+m}$ and $a \in \mathbb{R}$. Also implement $g1$. Finally, check that $g0$, $g1$ give roughly the right answer when the input $c = (c_1, \ldots, c_{n+m})$ is the *correct* vector (that is, when $c_i = 0$ if and only if the $i$'th listing is from Brazil). Also answer the following question: do you expect the samples from $g_0$ to have *exactly* the same mean as you computed in the first part of this question? Why or why not?

(4) In class, we saw the Metropolis-Hastings algorithm for sampling from a posterior. The next-most popular algorithm is the Gibbs sampler. In the context of this example, the MH algorithm from class would be rewritten as:

```
Gibbs<-function(theta,T,Z) {
ell = length(theta)
res = matrix(0,nrow=T+1,ncol=ell)
res[1,] = theta
for(i in 1:T) {
  res[i+1,1] = g0(res[i,3:ell],Z)
  res[i+1,2]= g1(res[i,3:ell],Z)
  for(j in 3:ell){
    res[i+1,j] = f(res[i+1,1], res[i+1,2], Z[j])
  }
}
return(res)
}
```

where $\theta \in \mathbb{R}^2 \times \{0,1\}^{n+m}$ is a starting position, $T$ is a number of steps and $Z$ is your data.

Run this algorithm for a large number of steps. Compute posterior credible intervals for your estimate of $\mu_0, \mu_1$.

**Note:** Depending on what you did, this may run poorly. If you wish, you may subsample the two datasets at this stage of the analysis. Make sure you keep at least a few hundred datapoints, and indicate clearly that you are doing this.

(5) Note that the posterior distribution $f_{post}$ assigns a probability that $C_i = 1$ and a probability that $C_i = 0$. This defines a classifier, by taking the class with the highest posterior probability.

Compute this classifier and comment on its quality and any relationship between the classification accuracy and list price.

## Grabbing Data for the Latent Variable Problem

The data in the latent variable problem is "real," but heavily modified to make the question easier:

```
#Brazil
raw = read.csv("rdjlistings.csv")
raw_prices = raw$price
prices_fixed=  gsub('[$]', '', raw_prices)
prices_fixed=  gsub(',', '', prices_fixed)
prices_fixed = as.numeric(prices_fixed)
prices_fixed = prices_fixed[prices_fixed < 650] #Remove huge outliers
prices_fixed = prices_fixed*0.19 #Convert to USD
prices_fixed = prices_fixed*0.65 #This is fake.
write.table(prices_fixed, "BrazilPrices.csv")

# SF
raw = read.csv("sflistings.csv")
raw_prices = raw$price
prices_fixed=  gsub('[$]', '', raw_prices)
prices_fixed=  gsub(',', '', prices_fixed)
prices_fixed = as.numeric(prices_fixed)
prices_fixed = prices_fixed[prices_fixed < 350] #Remove huge outliers
prices_fixed = prices_fixed*1.1 #This is fake.
write.table(prices_fixed, "UsaPrices.csv")
```

.