Assignment 6

STA 141A

Due **December 11 at 11:59pm**.

Assemble your answers into a report. Please do not include any raw R output. Instead, present your results as neatly formatted tables or graphics, and write something about each one. You must **cite your sources**. Your report should be **no more than 5 pages** including graphics, but excluding code and citations. Think carefully about what information is important to include.

When you are finished, submit a digital copy on Canvas. The digital copy must contain:

- Your report as a PDF file, with your code in the appendix.
- Your code as 1 or more R scripts.

Your submission will be graded according to the STA 141A grading standards, which are available on Canvas. The purpose of the assignment is to practice implementing a real statistical algorithm in a way that uses CPU and memory efficiently. **Part of your grade will depend on the correctness and efficiency of your code.** Your grade will NOT depend on the error rate of your model.

You may NOT use packages for k-nearest neighbors and cross-validation in this assignment. All other packages (for example, ggplot2) are okay.

Description

The U.S. Postal Service processes an average of 21.1 million pieces of mail per hour. Outbound mail must be sorted according to the zip code of its destination. In the past, postal workers sorted mail by hand, which was tedious and expensive. Over the last 40 years, USPS has switched to automated mail sorting. The sorting machines use statistical classifiers to identify the individual digits in the zip code on each piece of mail. Zip codes only contain the numbers 0 through 9.

In this assignment, you'll fit a model to classify handwritten digits and then examine the effectiveness of the model. Specifically, you'll implement a *k-nearest neighbors* classifier and use *cross-validation* to estimate the error rate.

The k-nearest neighbors algorithm classifies an observation based on the class labels of the k nearest observations in the training set (the "neighbors"). The classification power of k-nn depends on the choice of k and on the *metric* used to measure distance. Common distance metrics include include Euclidean, Manhattan, and Minkowski distance.

The cross-validation (CV) algorithm estimates the error rate of a model by fitting the model multiple times with part of the training set left out to validate the model. Specifically, in m-fold cross-validation, the original training set is split into m equally-sized subsets, selected randomly without replacement. The model is fitted with $m - 1$ of the subsets as a new training set. The remaining subset, called the *validation set*, is used to compute an error rate for the model. This procedure is repeated m times, so that each subset is used once as the validation set. The result is m estimates of the error rate for the model. The mean of these is the *CV error rate* and is a good estimate of the error rate for the model.

Data Description

The digits data set is a collection of grayscale images of scanned zip code digits. Each image shows one digit. Computer images are made up of tiny solid-color squares called *pixels*. In a grayscale image, the amount of white (or black) in a pixel can be stored as a single number, so the overall image can be stored as a matrix.

The digits data set is split into two files: a training set and a test set. In each file, each line is one observation (one digit image). There are 257 entries on each line, separated by spaces. The first entry is the class label for the digit (0–9) and the remaining 256 entries are the pixel values in the 16×16 grayscale image of the digit. The pixel values are standardized to the interval $[-1, 1]$, where -1 corresponds to pure white. There are 7291 observations in the training file and 2007 observations in the test file.

The data set is available on Canvas as `digits.zip`. **Get started early. Ask questions on Piazza and in office hours to get help.**

Questions

1. Write a function `read_digits()` that reads a digits file into R. Your function must allow users to specify the path to the file (training or test) that they want to read. Your function must return a data frame with columns that have appropriate data types. *No written answer is necessary for this question.*
2. Explore the digits data:
   - Display graphically what each digit (0 through 9) looks like on average.
   - Which pixels seem the most likely to be useful for classification?
   - Which pixels seem the least likely to be useful for classification? Why?

3. Write a function `predict_knn()` that uses k-nearest neighbors to predict the label for a point or collection of points. At a minimum, your function must have parameters for the prediction point(s), the training points, a distance metric, and k. Use the training set to check that your function works correctly, but do not predict for the test set yet. *No written answer is necessary for this question.*

4. Write a function `cv_error_knn()` that uses 10-fold cross-validation to estimate the error rate for k-nearest neighbors. Briefly discuss the strategies you used to make your function run efficiently.

5. In one plot, display 10-fold CV error rates for all combinations of $k = 1, \ldots, 15$ and two different distance metrics. Which combination of k and distance metric works best for this data set? Would it be useful to consider additional values of k?

6. In one plot, display the test set error rates for all combinations of $k = 1, \ldots, 15$ and two different distance metrics. How does the result compare to the 10-fold CV error rates? What kinds of digits does the "best" model tend to get wrong?

Hints

- The built-in `read.table()` function can read space-separated tables.
- The built-in `image()` function displays a matrix as an image.
- The built-in `dist()` function computes distances.
- It's a good idea to break the steps in `predict_knn()` and `cv_error_knn()` into even smaller functions that those functions use.
- Computing distances is time-consuming, so avoid doing so in a loop.
- Ties in k-nearest neighbors can be broken by random selection, by choosing the most popular class, or by other strategies. Some strategies are more effective than others.
- The `sample()` and `rep()` functions are useful for assigning observations to subsets in cross-validation.
- Rather than splitting entire observations into subsets for cross-validation, it is easier and more efficient to split their indexes (row numbers) into subsets.
- A confusion matrix shows the frequencies (or proportions) of predicted class labels versus true class labels. A confusion matrix provides more information about a classifier's strengths and weaknesses than the error rate alone.
- A k-nn classifier can achieve $> 85\%$ accuracy for this data set.
- An efficient implementation of the cross-validation in this assignment should run in < 5 minutes on modern hardware.