Final Project

STA 141A

Due Tuesday, December 5 by 5:00 pm

Get started immediately! The instructor and TAs will not answer questions about this assignment's content after **December 1st**.

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.

In this assignment, you will implement the k-nearest neighbors algorithm for classifying handwritten digits in zip codes. Zip codes only contain the numbers 0 through 9.

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 effectiveness of k-nn depends on the choice of k and on how distance is measured between observations. Distance can be measured with real-world Euclidean distance or with more exotic metrics such as Manhattan distance and Minkowski distance[1].

The data set is split into a training set and a test set. Both have the same format. In the files, each line is one observation (one digit). 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 $[-1, 1]$. There are 7291 observations in the training file and 2007 observations in the test file.

Questions

Use R to find answers to all of the following questions (that is, don't do any by hand or by point-and-click). Save your code in an R script.

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

1. Write a function `read_digits()` that loads a digits file into R. Your function should convert columns to appropriate data types and allow users to specify which file they want to load. *No interpretation is necessary for this question.*
2. Write a function `view_digit()` that displays one observation (one digit) from the data set as a grayscale image. Your function should allow users to specify which observation they want to display. *No interpretation is necessary for this question.*
3. Explore the digits data. In addition to your own explorations:
   (a) Display graphically what each digit (0 through 9) looks like on average.
   (b) Which pixels seem the most likely to be useful for classification? Which pixels seem the least likely to be useful for classification? Why?

[1] Some of these will be presented in lecture/discussion.

4. Write a function `predict_knn()` that uses k-nearest neighbors to predict the label for a point or collection of points. At the least, your function should take the prediction point(s), the training points, a distance metric, and k as input. *No interpretation is necessary for this question.*
5. 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.
6. Display 10-fold CV error rates for $k = 1, \ldots, 15$ and at least 2 different distance metrics in one plot. Discuss your results. Which combination of k and distance metric is the best? Would it be useful to consider additional values of k?
7. For each of the 3 best k and distance metric combinations, use 10-fold cross-validation to estimate the confusion matrix. Discuss your results. Does this change which combination you would choose as the best?
8. For the best k and distance metric combination, explore the training set digits that were misclassified during cross-validation. Discuss what you can conclude about the classifier.
9. Display test set error rates for $k = 1, \ldots, 15$ and at least 2 different distance metrics in one plot. Compare your results to the 10-fold CV error rates.
10. Briefly summarize what each group member contributed to the group.

Assemble your answers into a report. Please do not include any raw R output. Instead, present your results as neatly formatted[2] tables or graphics, and write something about each one. You must **cite your sources**. Your report should be **no more than 6 pages** including graphics, but excluding code and citations.

What To Submit

Submit a digital copy on Canvas. The digital copy must contain your report (as a PDF) and your code (as one or more R scripts).

Additionally, submit a printed copy to the box in the statistics department office[3]. The printed copy must contain your report and your code (in an appendix). Please print double-sided to save trees. It is your responsibility to make sure the graphics are legible in the printed copy!

Only one printed copy needs to be submitted for each group.

Hints

- The built-in `image()` function displays an image.
- There's also a built-in function for computing 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 `rep_len()` function is useful for assigning a cross-validation fold to each observation.
- Rather than splitting entire observations into folds for cross-validation, it is easier and more efficient to split their indexes (row numbers) into folds.
- 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.

[2] See the graphics checklist on Canvas.

[3] 4th floor of Mathematical Sciences Building

- 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.