Homework 5

36-462/662, Data Mining, Fall 2019

Due at 10 pm on Wednesday, 2 October 2019

AGENDA: Familiarization with nearest-neighbor prediction, and looking at the errors of classifiers.

This week's assignment uses the `spam` data set in the `ElemStatLearn` library. Download the package from CRAN, and load the data set into memory with `data(spam)`.

The data is for learning to classify e-mail as spam or real mail. There are 58 columns: 57 of them are features (see `help(spam)`, and `https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.names`), and the last one is a categorical variable ("factor"), called `spam`, with two values, `email` and `spam`. There are 4601 rows, representing 4601 different e-mails.

You will also need a package for doing nearest neighbor classification; I recommend `FNN`, which is, as its name promises, very fast. (Fitting models to the whole data set takes less than half a second on my not-very-speedy machine.)

1. *Online questions* (10) are online and due on Sunday at 10 pm.
2. *Base rates* To see whether a classifier is actually working, we should compare it to a constant classifier which always predicts the same class, no matter what the input features actually are.
   (a) (2) What fraction of the e-mails are actually spam?
   (b) (5) What should the constant classifier predict?
   (c) (3) What is the error rate of the constant classifier?
3. *Training and testing* (10) Divide the data set at random into a training set of 2301 rows and a testing set of 2300 rows. Check that the two halves do not overlap, and that they have the right number of rows. (Do *not* print a list of 2301 row numbers.) What fraction of each half is spam? Pick three of the other variables and check that they have *approximately* the same distribution in each set. Show your code for all of this, including comments explaining your approach and choices.
4. *Linear classifier*

(a) (5) Fit a prototype linear classifier to the training data, and report its error rate on the training data.

(Recall from the beginning of the course that this means finding the average feature vector for the two classes, and classifying each point according to which prototype it is closer to.)

(Be careful *not* to include the last column, with the class labels, in either your calculation of the prototypes, or in your use of the prototypes to classify the data points.)

(b) (5) Find the 57-dimensional-vector which runs between the two class prototypes. Then do principal components analysis on the data (not including the class labels). Find the angle between PC1 and the vector between the prototypes. Does the direction of maximum variance align well with the average difference between spam and e-mail?

(c) (5) Find the error rate of your prototype linear classifier (fit to the training data) on the testing data. How big is the difference? How could you test whether the difference was statistically significant?

5. *Nearest neighbors*

(a) (5) For each k in $1:50$, fit a k-nearest-neighbor classifier to the training data. Report the in-sample error rates in the form of a figure. What does this suggest about the optimal choice of k?

(b) (5) Now calculate the error rates for the k-nearest-neighbor classifiers, fit to the training data, on the testing data. Add the test-set error rates to your previous plot. What k should you pick to minimize the error on the testing set?

(c) (5) You should find that, at every k, the error on the testing set is higher than the error on the training set. Explain why this is so, in your own words.

(d) (10) Explain, in your own words, why we get different ideas about which k to use depending on whether we evaluate the error in-sample or out-of-sample.

6. *Errors* Pick the predictive model from the previous problems with the lowest error rate.

(a) (6) What is its rate of false negatives? That is, what fraction of the spam e-mails in the training set did it not classify as spam?

(b) (6) What is its rate of false positives? That is, what fraction of the genuine e-mails in the testing set did it classify as spam?

(c) (8) What fraction of e-mails it classified as spam were actually spam? (This is sometimes called the "positive predictive value".)

Rubric (10): The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical

and other mechanical errors, and easy to follow. Questions which ask for a plot or table are answered with both the figure itself and the command (or commands) use to make the plot. Plots are carefully labeled, with informative and legible titles, axis labels, and (if called for) sub-titles and legends; they are placed near the text of the corresponding problem. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. Code is properly integrated with a tool like R Markdown or knitr, and both the knitted file and the source file are submitted. The code is indented, commented, and uses meaningful names. All code is relevant; there are no dangling or useless commands. All parts of all problems are answered with actual coherent sentences, and raw computer code or output are only shown when explicitly asked for.

Extra credit (10 points total): Write an R function which will first do PCA on the training data, keep the top q principal components, and use them to do k-nearest-neighbor classification on the testing data, reporting the predictions. The function should take k, q, and the training and testing data sets as inputs, and return the vector of predictions. (You'll need to do a little work to make sure that the data vectors in the test set are being projected on to the principal components of the training set.) Using this function, write another function which optimizes the choice of both k and q for prediction on the test set.