Project 1

Email Spam Classification

Assigned: Wednesday, October 11, 2017

Due: Tuesday, October 31, 2017 at 11:59pm

A typical email spam filtering solution these days involves multiple layers of filters, including blacklisting IP addresses, global content-based filtering, and personalized content-based filtering. In this project, you will construct a personalized content-based email spam filter using supervised learning techniques.

DATA DESCRIPTION

You are given a set of training data in CSV format, where each row represents an email. The emails have been pre-processed into a set of features that are useful for classifying emails to this recipient as spam or not spam (ham). Each row consists of (beginning from the first column):

- 48 real-valued features in the range $[0,100]$: Each attribute corresponds to a word associated with spam and denotes the percentage of words in the email that match the given word.
- 6 real-valued features in the range $[0,100]$: Each attribute corresponds to a character associated with spam and denotes the percentage of characters in the email that match the given character.
- 1 real-valued attribute denoting the average run length (length of uninterrupted sequences) of capital letters.
- 2 integer features denoting the longest run length and the total run length (sum of all run lengths in the email) of capital letters.

The last entry in each row is the target class, which is either 0 (not spam) or 1 (spam). The training data provided to you consists of 800 spam and 1200 non-spam emails.

OBJECTIVE

Your task is to train a classifier to predict whether an email is a spam email given the aforementioned feature representation. Your classifier will be evaluated in two ways: 10-fold cross-validation on the training data and evaluation on a held out test data set. Experiment with different classifiers and techniques for feature selection and normalization. We will use the AUC, the area under the Receiver Operating Characteristic (ROC) curve, as the accuracy metric in this project.

It is recommended to use well-written and tested implementations of machine learning algorithms for this project. While the code that we have been using in class from the textbook website is useful for learning and illustration purposes, it is not as fully featured or robust as a well-established machine learning package such as `scikit-learn`. To help you get started with `scikit-learn`, see the Python script `classifySpamNb.py` provided with the project on Blackboard.

Building an accurate machine learning system is an iterative refinement process that involves

- Understanding the properties of the data by conducting exploratory data analysis and visualization.
- Understanding the underlying assumptions behind different machine learning algorithms to identify whether a particular algorithm is well-suited to this application.

- Pre-processing the data in a manner so that is most appropriate to match the assumptions of the chosen algorithm.
- Examining intermediate variables, decision boundaries, and outputs of a machine learning algorithm beyond just the classification accuracy to refine the trained model.
- Experimenting with a large number of models and algorithms.

While it is certainly possible to build an extremely accurate classifier just based on extensive experimentation, it would probably be a better use of your time to be selective in how you choose algorithms and to carefully examine the outputs of an algorithm to improve it.

GRADING: 50 POINTS TOTAL

Classification accuracy: 20 points

Your classifiers will be evaluated for accuracy in two ways:

- 10-fold cross-validation AUC on training set.
- AUC on a held out test set (not available to you).

The former is under your control, and the latter will require you to use your judgment to select a model that provides a good balance of cross-validation accuracy and model complexity (to avoid overfitting).

Each test will be worth 10 points, and your grade will be determined by the accuracy of your classifier *compared to the rest of the class*, i.e. the most accurate algorithm receives 10 points, the next most accurate algorithm receives 9 points, etc. If two algorithms are extremely close in accuracy, they may be assigned the same grade.

Your algorithm will also be compared to a simple baseline that is not expected to be a competitive solution. In this project, the baseline will be the Gaussian naïve Bayes model, fit directly to the data with no pre-processing. The accuracy of your algorithm must exceed that of the baseline to receive a grade higher than 5/10 for a particular test. See the `classifySpamNb.py` file posted on Blackboard with the assignment for code implementing the baseline.

Report: 30 points

Submit a report of not more than 5 pages that includes descriptions of

- Your final choice of classification algorithm, including the values of all parameters required by the algorithm.
- Any pre-processing, including feature selection, you performed.
- How you tested your algorithm and what other algorithms you compared against. Explain why you chose a certain algorithm. It is certainly OK to blindly test a large number of algorithms, but you will likely find it a better use of your time to be selective based on the specifics of this data set and application.
- Recommendations on how to evaluate the effectiveness of your algorithm if it were to actually be deployed as a personalized spam filter for a user. Is AUC the best metric in this case? Think about implications on the choice of metric and threshold for your classifier.

Your report will be evaluated on both technical and presentation quality.

SUBMISSION DETAILS

For this project, submit the following to Blackboard:

- A file named `bbUserName_classifySpam.py` containing your source code.
- A Word document or PDF file containing your report.

Your source code must contain the following two functions with *no modifications to the function header*:

```
def aucCV(features,labels):
```

This function trains a classifier using 10-fold cross-validation given the training features and labels. The output should be an array containing the AUC of each validation fold.

```
def aucTest(trainFeatures,trainLabels,testFeatures,testLabels):
```

This function trains a classifier using the training features and labels to predict the labels of the test features. These predicted labels are to be compared against the actual test labels. The output should be the AUC on the test set. Since you do not have access to the held out test set, you should test your function by splitting the training data set provided to you into training and test sets.

If you have any other code in your `bbUserName_classifySpam.py` file, please place it under the following `if` block:

```
if __name__ == "__main__":
```

This prevents the code from running when I import the `aucCV()` and `aucTest()` functions from your `bbUserName_classifySpam.py` file. For an example, see the structure in the `classifySpamNb.py` file on Blackboard with the assignment. I will use the script `evaluateClassifier.py` (also posted on Blackboard) to evaluate the accuracy of your classifier.