Logistic Regression: Filtering Spam Emails

Signal Data Science

You'll be looking at the data from the CSDMC2010 SPAM Corpus. The data is in `spam-emails.csv`, with `spam-emails-key.txt` giving the correct classification as spam or not-spam.

- Use the `tm` package to construct a `DocumentTermMatrix` from the data, where each row represents a document, each column represents a word, and each entry contains the word frequency for the associated word in the associated document.
- Use the `caret` package to find the optimal values of (λ, α) for regularized logistic regression in the classification of the documents.
- Using the optimal hyperparameters you found, train a regularized logistic regression model on the whole dataset. Look at the ROC curve.

Some things to keep in mind (read these before you begin):

- For a reference about text preprocessing with `tm`, look here or here.
- Not every single email in the dataset successfully made it into `spam-emails.csv`. You'll want to do an `inner_join()` (from `dplyr`) on the dataset and the classification key, retaining only the rows which are successfully matched.
- Remove columns corresponding to words that show up fewer than 10 times in total throughout the entire corpus.
- The matrix of documents and word frequencies is very sparse – don't scale it! If you do so, it will make it non-sparse, which is bad!

Using n-grams with logistic regression

We can compare our naive Bayes classifier with logistic regression! Let's see how much better we can do by using n-grams, which are sequences of n consecutive words, instead of individual words. (For simplicity, we'll just consider $n \in \{1, 2\}$.) In addition, we'll use the *count* of each n-gram, which is more informative than the simple binary *presence or absence* of each word used for naive Bayes classification.

- Use the `ngram` package (specifically `ngram()` and `get.phrasetable()`) to create a dataframe of 1-gram and 2-gram frequencies from the entire dataset of labeled emails. Each row should represent a particular email and each column should be one of the n-grams.
- To reduce computational demands, restrict consideration to the 20,000 most common 1-grams and the 10,000 most common 2-grams, where commonness is judged by the number of emails in which an n-gram appears at least once, *not* the total frequency of appearances.
- Randomly subset 80% of the rows to form a training set and use the remaining 20% as a test set.
- Fit a L^1 regularized elastic net logistic regression model on your training set. Make predictions on the test set, graph the associated ROC, and compute the AUC, false positive rate, and false negative rate. Compare the quality of the logistic classifier to that of the naive Bayes model.