Project Specification
Newspaper Article Classification Contest

CMU 10-601: Machine Learning (Spring 2014)
http://www.cs.cmu.edu/~10601b/
OUT: 3/14/2014
Project Proposal due: 3/24/2014, 11:59 PM
Midway Report due: 4/14/2014, 11:59 PM
Final Report and code due: 5/5/2014, 11:59 PM

1 Project Description

1.1 Overview

In this project, you will apply the machine learning techniques learned in this course to a real world dataset. Up until now, you have focused on applying a specific algorithm to carefully preprocessed toy datasets. This project will require that you make decisions such as which classifier to use, which features to use, how to train the classifier, etc. You will find that real world datasets are more difficult to work with, and that careful tuning is required to get optimal performance.

1.2 Dataset

In this project you will work with a labeled dataset of news articles. The data for the project can be downloaded at: http://www.cs.cmu.edu/~10601b/project/handout.tar.

- Each instance in the dataset consists of the text of a New York Times article published between 2000 and 2003 (we do not guarantee that the test data used to evaluate your accuracy will be from this same date range). Each article comes from one of four newspaper sections: News, Classifieds, Opinion, and Features. This dataset has been culled from the Linguistic Data Consortium's New York Times Annotated Corpus.
- For your convenience the text has been slightly preprocessed: punctuation and formatting (such as paragraphs) have been removed, and all characters have been converted to lower case.
- The articles are stored in data_train.txt and data_valid.txt. Each line in the file is a single article. Each article consists of a space-separated list of words.
- The labels (the sections the articles came from) are stored in the labels_train.txt and labels_valid.txt. Each line in the file is a single label. The nth line in the labels_train.txt file contains the label for the article stored on the nth line of the data_train.txt file.

  The labels are numerical to make it easier to use them in Octave, but we have also provided the original text labels in labels_train_original.txt and labels_test_original.txt. The mapping is:

  - News: 0
  - Opinion: 1
  - Classifieds: 2
  - Features: 3

YOUR TASK is to write a classifier which can correctly predict the label of a given article. To receive full marks, it will not be sufficient to naively implement one of the classification algorithms from class; you will need to implement some more sophisticated methods of feature generation, classification, or both. For instance, you might experiment with various techniques to combat overfitting, or you might use a dimensionality reduction technique to generate better features.

1.3 Grading and Deliverables

Your grade for this project will be based on several factors.

- **Proposal [5 pts; due 3/24]**
  This is a half-page document that should briefly outline your plan for solving the problem. Things to consider: How will you convert the articles to features? Which classifiers will you use? How will you train them? This document should also state who is in your group.

- **Midway Report [20 pts; due 4/14]**
  This is a 3- to 4-page document that should describe your progress on solving the problem. Consider the midway report as a partially complete final report. The midway report should therefore follow the same format as the final report, but not all sections need to be complete.

  Note that by the midway report, *we expect you to have successfully submitted a working classifier to Autolab that achieves better-than-baseline classification accuracy.* This classifier can be a simple implementation of one of the algorithms from class, but you should explain in the report what enhancements or more sophisticated methods you intend to try.

- **Final Report [40 pts; due 5/5]**
  This is a 8-page document that describes how you solved the problem. It should contain detailed discussion about which methods you tried, results on their comparative performance, and a dicussion about which methods worked best and why. It should follow the following simple outline: Introduction, Methods, Results, Conclusion.

- **Classification Accuracy [35 pts]**
  In addition to the written deliverables, you will submit your classifier to Autolab, which will use it to classify a hidden test set. Your classification accuracy on this hidden test set will contribute to your final grade.

- **Competition [BONUS 10 pts]**
  To make things a little more exciting, this project is a competition. Your classification accuracy will be displayed on the class leaderboard using the handle you selected when you first logged in to Autolab. Students who are on top of the leaderboard at the start of each lecture may recieve prizes, and the top 5 groups at the conclusion of the competition will receive bonus marks.

Please use the NIPS document templates for your midterm and final reports.

1.4 Autolab Instructions

The code stubs and data files for this project can be found at: `http://www.cs.cmu.edu/~10601b/project/handout.tar`

This project can be split into two tasks: (1) Converting the news articles into features, and (2) using these features to train and run a classifier. We have split these two tasks into two separate files: `featurize.py` and `classify.m`.

1.4.1 featurize.py

`featurize.py` is a **Python** program[1] which converts text files containing the train and test news articles into CSV files containing matrices of features. This file contains the following function for you to implement:

- `featurize(train_articles, test_articles)`
  This is a **Python** function. `train_articles` and `test_articles` are Python lists of article strings – i.e., each list entry is one line from `data.txt` (or the equivalent testing file).

  The `featurize` function should return a tuple `(train_features, test_features)` of feature matrices. Each feature matrix should be a list of feature vectors, with each feature vector represented by a list of

[1] We want you to use Python for this task because it is a flexible, terse language which is well suited to text processing (Octave is extremely poor at text processing).

numbers. Note that features must be converted here to integers or floats, and that each feature vector must be the same length.

The starter code includes functionality for reading text and writing CSV files; you need to worry only about implementing the featurize function. You may import any module in the Python standard libraries, and you may also import the numpy library. The only restrictions are that you may not import the os module (and file-manipulation libraries that rely on it), and that your code will not be allowed to access files other than those in the local directory.

You can run the script using the following command line:

```
python featurize.py train_input_name train_output_name test_input_name test_output_name
```

1.4.2 classify.m

is an **Octave** program which uses the featurized training data to classify featurized test data. It contains the following function for you to implement:

- [c] = Classify(XTrain, yTrain, XTest)
  This is an **Octave** function. XTrain is the name of the CSV file containing the featurized training data. yTrain is the name of the text file containing the class labels. XTest is the name of CSV file containing the featurized test data. c is a vector of predicted class labels.

1.4.3 Submitting to Autolab

Complete each of these functions, then compress this directory *as a tar file* and submit to Autolab online. You may submit code as many times as you like.

Your code must be submitted to Autolab by 5/5/2014 at 11:59 PM, along with your final report.

- **SUBMISSION CHECKLIST**
  - Submission executes on our machines in less than 20 minutes.
  - Submission is smaller than 100K.
  - Submission is a `.tar` file.

2 Suggested Techniques

A few classification techniques you may want to review/investigate:

1. Logistic regression
2. Neural networks
3. k-nearest neighbors
4. Support vector machines

A few techniques for generating features that you may want to investigate:

1. Dimensionality reduction
2. The "bag of words" model
3. n-gram models (including smoothing techniques)
4. Other kinds of vector space models (see `http://www.jair.org/media/2934/live-2934-4846-jair.pdf` for several types of such models)
5. TF-IDF

6. Topic models (note: many of these, such as Latent Dirichlet allocation, are quite involved to implement)

A few other techniques that may turn out to be useful:

1. Cross-validation
2. Regularization