IEOR 290 002 - Spring 2017
Project
Due 05/08/2017 at 11:59 pm

Description

In this project you need to classify emails as spam or ham (not spam), and compete with other teams on Kaggle to see which team makes the most accurate classifier.

- **Data**

  The Raw data consists of 5172 training emails and 5857 testing emails in `.txt` format. Out of the 5172 training emails there are 1500 spam emails and 3672 ham emails. However, unlike homework problems, the testing emails have no target value available to you. The way to handle the testing emails is explained in the Model and Kaggle sections below. All data can be downloaded from bCourses.

- **Feature**

  The training and testing data, `spam_data.mat`, is produced using a basic feature extractor `featurize.py`, which converts raw emails in `.txt` format into proper matrix for you to apply machine learning models. There are 32 basic features:

  - Features 1 to 25: percentage (0 to 100) of words in a given email which match a given word, such as 'business', 'free', etc.
  - Features 26 to 32: percentage (0 to 100) of special characters in a given email which match a given character, such as ; $ # ! ([&.

  The list of words and special characters can be found in `featurize.py` which is available on bCourses. You're free to add, remove and modify the features and regenerate the training and testing data using `featurize.py` if you use Python. You need to write your own feature extractor similar to `featurize.py` if you use other programming languages.

  Some ideas about feature design that can potentially increase the testing accuracy:

  - list of common spam words
  - bag-of-word model
  - length of uninterrupted sequence of capital letters

- **Preprocessing**

  You may preprocess and transform the data to increase the accuracy of your model. Some ideas about preprocessing:

- Standardize data to have all mean 0 and variance 1
- Transform data using $\log(x_{ij} + 1)$
- Binarize data using $\mathbb{1}\{x_{ij} > c\}$ for $c \geq 0$.

Some preprocessing methods such as standardization can be done directly using the `sklearn` package but for others you need to code the preprocessing by yourself.

Model

The goal of the project is to build a classifier that takes the training data as input and outputs predictions on the testing emails. You're free to choose any model or any technique learnt in class and/or out of class.

Since testing emails have no target value, you need to divide the training emails into a training set and a validation set by yourself so that you'll be able to tune the hyper-parameters in your model and check the effectiveness of your classifier. Check the skeleton code `IEOR290_project_skeleton.py` on bCourses to get started.

Kaggle

We use Kaggle for evaluating the performance of your classifier. Any submission to Kaggle should follow the following procedure:

Save your testing predictions in a CSV file in the following format:

Id	Category
1	0
2	0
3	1
4	0
5	0
⋮	⋮

The CSV file should have two columns: `Id` contains the indices from 1 to 5857, and `Category` contains your predictions, 1 for spam and 0 for ham. The order of the submitted predictions must be the same order as of the testing emails provided. Then, submit the CSV file on Kaggle.

Kaggle submission is open from **April 24 (Monday) at 0:01 am until May 6 (Saturday) at 11:59 pm**, 48 hours before the deadline of the project.

The set of the testing emails is partitioned into two subsets, a public testing set and a private testing set, where only the GSI knows the partition. After submission Kaggle will report the accuracy of the submitted prediction on the *public testing set*. You'll also be able to observe the public leaderboard at any time which contains your best testing accuracy on the public testing set as well as your rank among (the best accuracy so far) of all the other teams. But your final grade will be evaluated based on the accuracy on the *private testing set*, which will be explained in the Grading section later.

Each team can only submit twice a day to prevent abusing Kaggle during the period Kaggle is open. Kaggle retains only your most accurate (on the public data set) predictions, so you can experiment with different ideas with no penalty for tries that yield poor accuracy. However, you should be aware that the number of submissions to Kaggle is limited.

Registration

Send to the GSI an email including the name, student ID and Berkeley email address for each of your team member. You will receive an invitation to the Kaggle competition when the open date is approaching.

Submission

You need to submit a *single* report per team in `.pdf` format on bCourses that contains

- **Feature design:** explain the features you added, removed and modified.
- **Code of feature extractor:** implementation of your feature extractor.
- **Preprocessing:** explain how you pre-process the data if any.
- **Model:** explain the model and technique you use.
- **Code of model:** implementation of your classifier.
- **Results:** report your Kaggle public score.

The report is due on **May 8 (Monday) at 11:59 pm**. No late submission is accepted so make sure you submit it well before the deadline to avoid last minute computer glitches.

Grading

There will be one grade for a team. Each team member will receive the team grade. The grade is based on:

1. **Correctness of your model (60%)**

   How well you do feature design, preprocess data and train your model and whether your ideas are correctly implemented, as well as the clarity of the report.

2. **Rank of the Kaggle competition (40%)**

   We evaluate your final standing using the accuracy of your predictions on the *private testing set*, which corresponds to your best predictions on the public testing data. That means higher accuracy on the public testing set does not necessarily guarantees higher accuracy on the private testing set. *Be aware of over-fitting.*

   Team with the highest testing accuracy is guaranteed to get full credit in this part. Credits for other teams depend on how close you are from the highest accuracy. Your GSI obtained 81% accuracy using the plain vanilla logistic regression without any additional feature or preprocessing. You are expected to do better.

Good Luck!