Assignment 2: Logistic Regression and Sentiment Analysis

Start Assignment

Due Wednesday by 11:59pm **Points** 100 **Submitting** a file upload
Available after Sep 23 at 2pm

Sentiment Analysis and Logistic Regression

In this assignment, you will implement a text classification system for sentiment analysis using logistic regression. I will provide training data in the form of positive and negative hotel reviews. You will use this training data to develop a system that takes in reviews as categorizes them as either positive or negative.

Training data

The training data for this task consists of a collection of short hotel reviews. The data is formatted as one review per line. Each line starts with a unique identifier for the review (as in ID-2001) followed by tab and the text of the review. The reviews are not tokenized or sentence segmented in any way (the words are space separated). The positive reviews and negative reviews appear in separate files. You can assume that the training data is balanced with respect to class.
Here are two examples:
ID-1274 Absolutely a great hotel for families! The rooms are spacious, restaraunts are very kid-friendly, and the pool area is gorgeous. My children felt like they were at a water park, not just another hotel. We will definitely return for another stay here!
ID-1021 This hotel is located at a busy traffic circle near the interstate. It isn't walking distance to downtown. The hotel restaurant was average. My wife complained about the quality of the sheets; they were pretty rough.
And here's the data: **hotelPosT-train.txt** **(https://canvas.colorado.edu/courses/75783/files/35637785/download?download_frd=1)** , **hotelNegT-train.txt** **(https://canvas.colorado.edu/courses/75783/files/35637808/download?download_frd=1)** .

Approach

There are two parts to this assignment: (1) processing the reviews and extracting feature vectors appropriate for use with logistic regression, and (2) implementing logistic regression and using it to find a set of weights that are effective for this task. You'll have separate submissions for each part.

Feature Extraction

For features, we'll use the set of 6 features introduced in the text.

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

To consistently extract these features we'll need some resources and a set of consistent choices about how to process the text. For the first two features, I'll provide a sentiment lexicon consisting of separate lists of positive (**positive-words.txt** ⬇ **(https://canvas.colorado.edu/courses/75783/files/35637810/download?download_frd=1)**) and negative words (**negative-words.txt** ⬇ **(https://canvas.colorado.edu/courses/75783/files/35637811/download?download_frd=1)**). The words in these files are all lower case, so let's assume that you should downcase all the review words.

Same for looking for "no" in feature 3.

For feature 4, let's go with the following list: "I", "me", "mine", "my", "you", "your", "yours", "we", "us", "ours".

For feature 5, we need to decide if we're going to separate punctuation from the words or leave them as part of the words. That is, strip the "!" from "families!" and "here!" or leave them alone. Let's assume we leave them alone. So "here!" stays the way it is and we'll have to look for the "!" on the end of words.

Once you have feature extractors that work you can write the reviews out as vectors to a CSV file, one review per line as described below.

Logistic Regression

Implement the SGD algorithm from Figure 5.5, using Cross-Entropy as the loss function. Begin by initializing the weights to zero. Treat the bias term as an additional weight by adding a dummy feature of 1 to each example. So instead of 6 weights, we'll have 7. You can experiment with different initial bias terms and learning rates.

To train your system, you should read in the training data from the CSV file into a matrix of training examples. SGD proceeds by randomly selecting an example, predicting its value with the current set of weights, using the correct answer to generate a loss/gradient, and then updating the weights.

Training and Development

To track your progress, you'll need to separate the data into two sets: training and development (I have a separate test set). Train on your training set and test on the development set. I'd recommend an 80/20 split. But be sure to remember to use all the data I give you when training your final system.

If you implement the given features correctly and SGD as in the book you'll get the bulk of the credit for the assignment. If you want to explore improved features or think you can improve over the base SGD you can do so. The points for this will be based on what you tried and if the degree to which it actually provides an improvement over the baseline.

Testing

To actually classify a review you'll need to accept a review in the same format as the training data, perform feature extraction on it, and pass it to your classifier. Once you've determined its class you will need to output the review ID and the class you've chosen (as POS or NEG) with a **tab** separator. So for the earlier examples, you would have

ID-1274 POS

ID-1021 NEG

Packages

Don't use packages like NLTK, scikit-learn, or other domain-specific NLP or ML packages. You can use generic Python packages like re, csv, numpy, string, math, etc.

Submission Details

For the first part of the assignment, you should submit a CSV file consisting of one line per review as follows: id, features, class. So for the example from class, you would have

ID-1274,5,0,0,2,1,3.74,1

Your submission should have the name lastname-firstname-assgn2-part1.csv.

For the second part of the assignment, you will submit your python code, and your output from the test set that I will provide.

lastname-firstname-assgn2.py

lastname-firstname-assgn2-out.txt

Your output will be graded automatically, so don't deviate from the format described above.