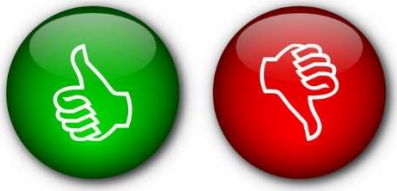
CSCI 4820/5820 (Barbosa)

Project 3: Lexicon-Based Sentiment Analysis using Custom Logistic Regression

Due: **See course calendar**. May not be turned in late.

Assignment ID: **proj3**
File(s) to be submitted: **proj3.ipynb, proj3.pdf**



Objective(s): Implement a custom logistic regression classifier (in Python) and use words from a sentiment lexicon to classify movie reviews as positive or negative.

Project description: You will implement a simple logistic regression classifier and use the words in a sentiment opinion lexicon as features to classify movie ratings. The lexicon files are provided (they are available as part of NLTK's corpora). One of the files contains positive words and the other negative words. All words in these files will used as features for classification. The classification will be undertaken on both the custom classifier and the sklearn LogisticRegression classifier and the results compared.

Requirements: Where two weights are shown for an item, the first is for CSCI 4820 and the second for CSCI 5820 credit.

1. (**85%/65%**) Your program must provide the functionality listed below, in a single file named ***proj3.ipynb***:

- Custom Logistic Regression Classifier - Implement the classifier as a **Python class** (*CustomLogisticRegression*) and provide the following:
  - An *__init__* method with default values for parameters for learning rate (*learning_rate*) and number of iterations (*num_iters*). This method should have members *learning_rate*, *num_iters*, *weights* and *bias* (these last two should be initialized to *None*, and will be re-initialized in the *fit* method when the shape of training features is known).
  - A *sigmoid* method which has one parameter, applies the sigmoid function to it, and returns the result.
  - A *fit* method which has parameters *X* and *y* (training features and labels respectively), sets the weights to the correct shape and initializes them to zero, and applies batch gradient descent (the entire dataset) in a loop for the number of iterations specified.
  - A *predict* method which has parameter *X* (the test data) and computes $w \bullet x + b$, applies the sigmoid to it, and returns the output.

  Note: Use *numpy* arrays to store features and labels and for matrix operations. See course slides for equations to support implementation. The entire class can be implemented in around 25-30 lines of Python code.

- Lexicon-Based Sentiment Classification – This assignment will use a movie reviews dataset (***imdb_reviews.txt***). The feature set will be a list of Boolean flags, one for each of the combined set of words found in the positive and negative lexicon files (*True* if the review contains that particular word and *False* if it does not). The 'positive' and 'negative' labels should be converted to 1 and 0 respectively. In order for the data to work in both classifiers, use np.arrays of Booleans and numeric labels. See below input format.

lex word 0	lex word 2	...	lex word 6K+
True	False	...	True
False	False	...	True
False	True	...	False
True	True	...	False
:	:	...	:
False	True	...	True

← To simplify, header not included in X or y

0
1
0
0
:
1

Features (X) Labels (y)

- Classifier parameters
    - The sklearn LogisticRegression classifier should be instantiated as shown below:

      LogisticRegression(solver='sag', C=0.001, max_iter=500)

    - The custom classifier should be instantiated as shown here:

      CustomLogisticRegression(learning_rate=0.1, num_iters=500)

- Classification process
    - Run both classifiers for 5 trials to average results
    - For each iteration (trial):
        - Shuffle the input data and split it 80%/20% into training and test sets.
        - Instantiate each classifier with the aforementioned arguments and run the test and training data through each.
- Output – For each trial, and for each classifier, the sklearn *confusion_matrix* and *classification_report* should be output for results. Output the average of the confusion matrices across trial for each classifier.

2. (**10%**) Test your program – Experiment with different learning rates and number of iterations for the custom classifier. Randomize the dataset for each trial.

3. (**5%**) Code comments - Add the following comments to your code:

- Place a Markdown cell at the top of the source file(s) with the following identifying information:

  Your Name
  CSCI <course number>-<section number>
  Project #X
  Due: mm/dd/yy

- Add a Markdown cell above each code cell explaining the processing carried out by that code.

4. (**CSCI 5820 students only 20%**) Analysis – Provide an analysis (in a Markdown cell) addressing the following topics

- There are many more words in the negative lexicon than in the positive. Discuss this imbalance in terms of its potential impact (in theory) on the classification. Assess whether this actually happened (in practice) and provide rationale as to why it did or did not (describe the process you undertook to determine this).
- Compare and contrast the performance of the custom logistic regression classifier to that of the sklearn LogisticRegression classifier, focusing on the measures from the classification report confusion matrix.
- Run the custom classifier with learning rates 0.05 and 0.2 and number of iterations set at 250 and 750 (averaged over 5 trials) and discuss the impact of the learning rate and number of iterations on the outcome.

5. Generate a pdf file of the notebook (from the terminal):

    $ *jupyter nbconvert --to html proj3.ipynb*
    $ *wkhtmltopdf proj3.html proj3.pdf*

6. Submit required files via the D2L dropbox for this project