Faculty of Computing and AI, Air University, Islamabad

Department of Creative Technologies

Software Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

ASSIGNMENT – 01

Submission Deadline: 04 March 2024, 11:59 PM

Instructions

1. Assignments are to be done individually. You must complete this assignment by yourself.
2. Late assignments will not be accepted.
3. Only **inpyb** file will be graded, i.e., any other format will not be graded.
4. The output needs to be presented nicely and displayed correctly. If appropriate comments and indentation are not made in the source code, 10% of the mark will be deducted.
5. There will be no credit if the given requirements are changed.
6. Please mention your registration number, name, and assignment number on top of the file.
7. Plagiarism may result in zero marks regardless of the percentage plagiarized.

Logistic Regression

Implement Logistic Regression for sentiment analysis on tweets. Given a tweet, you will decide if it has a positive sentiment or a negative one. Specifically, you will:

1. Extract features for Logistic Regression given some text.
2. Implement logistic regression from scratch.
3. Apply Logistic Regression on a Natural Language Processing task.
4. Test using your Logistic Regression.
5. Perform error analysis.

Using the data set of tweets from NLTK Corpus. Hopefully you will get more than 99% accuracy.

Perform following steps towards your implementation of Logistic Regression

1. Import Functions and Data.
2. **Preprocess** the data as defined in lectures.
    a. Train test split: 20% will be in the test set, and 80% in the training set
3. Implement the **Sigmoid Function**.
    a. You will want this function to work if z is a scalar as well as if it is an array.
4. Implement **Gradient Descent** Function.
5. Implement the **extract_features** Function.
    a. This function takes in a single tweet.

   b. Process the tweet using the imported **process_tweet()** function and save the list of tweet words.
   c. Loop through each word in the list of processed words
      i. For each word, check the **freqs** dictionary for the count when that word has a positive '1' label. (Check for the key (word, 1.0)
      ii. Do the same for the count for when the word is associated with the negative label '0'. (Check for the key (word, 0.0).)

6. Train the model.
   a. Stack the features for all training examples into a matrix X.
   b. Call **gradientDescent**, which you've implemented above.
7. Write **predict_tweet** function: Predict whether a tweet is positive or negative.
   a. Given a tweet, process it, then extract the features.
   b. Apply the model's learned weights on the features to get the logits.
   c. Apply the sigmoid to the logits to get the prediction (a value between 0 and 1).
8. Implement **test_logistic_regression**.
   a. Given the test data and the weights of your trained model, calculate the accuracy of your logistic regression model.
   b. Use your **predict_tweet()** function to make predictions on each tweet in the test set.
   c. If the prediction is > 0.5, set the model's classification **y_hat** to 1, otherwise set the model's classification **y_hat** to 0.
   d. A prediction is accurate when **y_hat** equals **test_y**. Sum up all the instances when they are equal and divide by **m**.
   e. Calculate Precision, Recall and F-Measure
9. Error Analysis.
   a. Show the tweets that your model misclassified.
10. Predict with your own tweet.
    a. Write your own tweet and check its sentiment.