CSE 6240 Spring 2020 - Homework #1

Analyzing a Movie Review Dataset

Due: Jan 29th, 2020

The aim of this homework is to give hands-on experience with various text preprocessing, classification, and visualization modules. In this assignment, you will clean a text dataset, create term matrices, perform a classification experiment with cross-validation to tune the hyper-parameters, and visualize and interpret the results.

Guidelines:

- You are given a solution template (an .ipynb file) and you are required to add your code at the appropriate places and submit it.
- The solution template contains 4 questions with multiple sub-parts for each one.
- For the coding exercises, please add your code below the comment "Add your code here."
- For the theory questions, please add your answers in a text cell below the questions. The cell type can be changed under Cell -> Cell Type.
- Points distribution for each question is added for your reference.

The Problem Statement:

Read through this tutorial on Kaggle, https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words, to familiarize yourself with its python tools and workflow. Write your own annotated IPython notebook(s) to reproduce the steps in the blog and complete the exercises below. You can start with the solution template and the sample code provided in the tutorial, but you should clean it up, document and refactor as necessary.

1: Processing text to create design matrices. Prepare four design matrices, `X_counts, X_binary, X_tfidf, X_binary_imbalance` following the blog's procedure for removing markup, punctuation and stop words. `X_counts` entries contain the raw word counts (the blog does this). `X_binary` modifies `X_counts` so that all elements are either 0 or 1. `X_tfidf` modifies `X_counts` by applying the sklearn tfidf vectorizer (http://scikit-learn.org/stable/modules/feature_extraction.html) with `smooth_idf=false`. For the `X_binary_imbalance`, start with `X_counts`, set your rng seed to 0 and then delete 75% of the rows corresponding to `sentiment=1`. This creates an imbalanced data set.

2: Feature space similarity experiment. Write a function `dist(X, i, j,`

`distance_function='Euclidean')` which returns the (Euclidean) distance between rows i and j of a design matrix. Also write a function `topk(X, k)` which returns `((i1,j1,d1),...(ik,jk,dk))` where `(ix,jx)` are the indices of the xth closest pair, and dx is the corresponding distance. You can break ties randomly. Then use `topk()` to find the closest review pairs for each design matrix and print the following: the indices of the reviews, the distance, the first 20 characters of each review, the labels for each review. Are the pairs always the same?

3. Classification Experiment. Now you're going to tune an SVM classifier using each design matrix and measure the resultant performance. Read the sklearn docs (http://scikit-learn.org/stable/modules/cross_validation.html) on cross-validation to see the methods to use.

- Set your rng seed to 0 and create an initial learning_set / test_set split of 80-20.
- Now we want to use a linear SVM (`svm.SVC` with `kernel=linear`) and pick the best C value for our classifier
    - Repeat for each of the four design matrices:
        - Repeat 30 times:
            - Pick a random value of C uniformly in the interval (1e-4, 1e4)
            - Use 5-fold cross-validation to train the SVM
            - Estimate and record the F1-score
        - Select the value of C which produced the best F1-score and estimate the f1_score on the test dataset.
        - Retrain the classifier using the entire learning set with this C value.
        - Submit test set predictions to Kaggle (see the section in the blog(https://www.kaggle.com/c/word2vec-nlp-tutorial/submit), and make sure you use their test data. You may need to retrain one more time using all "training data"). Print the Kaggle Score.

Which design matrix performed best (e.g., which encoding method worked best)? What was the lift (improvement in F1-Score) between the worst and best cases for each experiment?

4. Learning Curve Experiment. Using a logistic regression classifier and the design matrix X_counts generate a learning curve:

- Set your rng seed to 0 and create an initial learning_set / test_set split of 80-20
- Generate a learning curve (xval vs training error) for `n=(100, 500, 1000, 2000, 3000, 4000, 5000, 7500, 10000, 15000, 20000)` training instances.
- Interpret the learning curve.