CS57300: Homework 4

Due date: Sunday Apr 12, midnight (submit via turnin)

More Exploration of Naive Bayes Classifiers using Word Clusters

In this programming assignment you will run further experiments with the NBC. Instructions below detail how to use turnin to submit your code and assignment on data.cs.purdue.edu.
Alternate submissions (i.e., outside the turnin system) will not be accepted.

In the assignment, you are welcome to use the *scikit-learn* library in python. You will cluster words using (1) the standard kmeans algorithm (which is available in scikit-learn), and (2) a *spherical* kmeans algorithm that uses cosine similarity in the score (you will need to implement this, or modify the input to the standard kmeans to achieve the required result). You can use the NBC algorithm that you implemented for HW3, or you can use the NBC method available in scikit-learn.

You will consider the following setup without change, unless otherwise specified:

- Data: Use the `stars_data.csv` data.
- Features: Use the binary word occurrence features computed for the top 201-2200 frequently occurring words.
- Class Label: Use the single class label ***isPositive***.
- Algorithm: Use your NBC implementation, with smoothing, from HW3.
- Evaluation: Use zero-one loss.

New for this assignment:

- Cluster the words by their counts in the reviews to construct new features.
  (You can construct these features once using the entire dataset, before conducting any of the experiments below.)
  1. Construct two 2000×2500 word document matrices: $\mathbf{W}_P$ and $\mathbf{W}_{NP}$. In each matrix, the rows will correspond to one of the top 201-2200 frequently occurring words, the columns will correspond to a review, and the cell counts will record the *term frequency* of the words in each review. For example, if $\mathbf{W}[i, j] = 4$, it indicates that word i occurs 4 times in review j. You should separate the reviews into two sets based on the *isPositive* label, and construct one matrix for each set (i.e., $\mathbf{W}_P$ for positive, $\mathbf{W}_{NP}$ for not positive).
  2. Cluster the words in each $\mathbf{W}$ into 50 groups, using kmeans. The identified groups of words will hopefully correspond to topics. (*You might want to cluster a few times and take the best result.*)
  3. Construct 100 new binary "topic" features for each review.
     Specifically, let $\mathbf{T} = [T_1^P, ..., T_{50}^P, T_1^{NP}, ..., T_{50}^{NP}]$ be the 100 clusters identified from $\mathbf{W}_P$ and $\mathbf{W}_{NP}$ that partition the words[1]. Let $T_i = [w_1, w_2, ..., w_m]$ be the set of words in

[1]Note that a word should only be associated with **two** topics—one from the clustering of $\mathbf{W}_P$ and one from the clustering of $\mathbf{W}_{NP}$.

a particular topic cluster i. Then, construct a binary feature X_i for each topic —with $x_i = 1$ indicating that any word $w_j \in T_i$ appears in the review D, and $x_i = 0$ otherwise.

- Implement *spherical* kmeans in python. Spherical kmeans differs from standard kmeans in only a few aspects:

  1. Start by normalizing the data vectors to unit length (i.e., $||\mathbf{x}|| = 1$).
  2. Use *cosine similarity* as the "distance" measure. Since cosine is a similarity measure, you should assign points to the cluster with **maximum** similarity rather than minimum distance.
  3. When you re-compute the cluster centers, you will sum up the vectors of the cluster members and then renormalize the result to again have unit length.
     (i.e., let $g(c) = \sum_{x \in c} \mathbf{x}$, then the new centroid for c will be $g(c)/||g(c)||$)

- Use *incremental* 10-fold cross validation to compute learning curves with training sets of varying size. **You will do this for all classification experiments below.**

  1. Randomly partition the examples in dataset into 10 disjoint sets $\mathbf{S} = [S_1, ..., S_{10}]$
     (e.g., each set will have 250 examples if the data has 2500 examples total).
  2. For $tss = [100, 250, 500, 1000, 2000]$:
     (a) For $t = [1..10]$
        i. Let $test\_set = S_t$.
        ii. Let $\mathbf{S}_C = \mathbf{S} - S_t$.
        iii. Construct $train\_set$ by randomly sampling tss examples from $\mathbf{S}_C$.
        iv. Learn NBC from $train\_set$.
        v. Apply NBC to $test\_set$; measure zero-one loss $L_{0/1}$.
     (b) Record the average $L_{0/1}$ over the ten trials (and its standard error).
     Plot performance on learning curve for training set size tss.

- You will report the average performance (e.g., $L_{0/1}$) over the ten-fold cross validation trials and the associated *standard error*. The standard error is the standard deviation divided by the square root of the number of trials. For example if we let σ_k refer to the standard deviation of performance over the 10 cross validation trials, then:

$$sterr_k = \frac{\sigma_k}{sqrt(10)}$$

Programming assignment

You should implement your solution using Python. As described above, you are welcome to use your previous HW3 code and/or any methods available in the scikit-learn python library. As before, you should submit your typed HW report as a pdf, along with your source code files.

1. *Inspect the clustering results, comparing standard kmeans to spherical kmeans.* (15 pts)
   (a) Evaluate the standard kmeans algorithm in python for cluster sizes $k = [10, 20, 50, 100, 200]$ Plot the cluster score vs. k for both $\mathbf{W}_P$ and $\mathbf{W}_{NP}$. Identify the "best" k.
   (b) For the best choice of k, inspect the words in the clusters and report any noticeable topics that you find.
   (c) Repeat (a-b) using your spherical kmeans algorithm.
   (d) Discuss any noticeable differences in the results from the two algorithms.

2. *Assess whether the clustering approach improves performance.* (10 pts)
   Let approach A be the NBC using binary features from the 100 topics that you identified with standard kmeans (50 topics from $\mathbf{W}_P$ and 50 from $\mathbf{W}_{NP}$).
   Let approach B be the NBC using 100 binary features derived from spherical kmeans topics.
   (a) Plot the learning curves for A and B including error bars that indicate ± 1 *standard error*, using evaluation based on incremental CV as described above.
   (b) Formulate a hypothesis about the performance difference between A and B.
   (c) Discuss whether the observed results support the hypothesis (i.e., are the observed differences significant).

3. *Assess whether using the topic features improves performance.* (10 pts)
   Let approach A be the original NBC with 2000 binary word features (i.e., HW3).
   Let approach B be the NBC model using 100 binary features from the kmeans topics (using the best model from part 2).
   (a) Plot the learning curves for A and B including error bars that indicate ± 1 *standard error*, using evaluation based on incremental CV as described above.
   (b) Formulate a hypothesis about the performance difference between A and B.
   (c) Discuss whether the observed results support the hypothesis.

4. *Assess whether the number of features improve performance.* (10 pts)
   Let approach A be the original NBC with binary word features from HW3, but randomly sample only 100 of the 2000 word features to use in the model.
   Let approach B be the NBC model using 100 binary features from the kmeans topics (again using the best model from part 2).
   Let approach C be an NBC model that uses the 100 randomly selected word features from A and the 100 binary topic features from B.
   (a) Plot the learning curves for A, B, and C, including error bars that indicate ± 1 standard error, from the evaluation based on incremental CV as described above.
   (b) Formulate a hypothesis about the performance difference between A and B. Discuss whether the observed results support the hypothesis.
   (c) Formulate a hypothesis about the performance difference between C and A/B. Discuss whether the observed results support the hypothesis.

Submission Instructions:

After logging into data.cs.purdue.edu, please follow these steps to submit your assignment:

1. Make a directory named $'yourName\_yourSurname'$ and copy all of your files there.
2. While in the upper level directory (if the files are in /homes/neville/jennifer_neville, go to /homes/neville), execute the following command:

   turnin -c cs57300 -p HW4 your_folder_name

   (e.g. your prof would use: turnin -c cs57300 -p HW4 jennifer_neville to submit her work) Keep in mind that old submissions are overwritten with new ones whenever you execute this command.

   You can verify the contents of your submission by executing the following command:

   turnin -v -c cs57300 -p HW4

   Do not forget the -v flag here, as otherwise your submission would be replaced with an empty one.

Your submission should include the following files:

1. The source code in python.
2. Your evaluation & analysis in .pdf format. Note that your analysis should include learning curve graphs as well as a discussion of results.
3. A README file containing your name, instructions to run your code and anything you would like us to know about your program (like errors, special conditions, etc).