CSE 158/258, Fall 2020: Homework 2

Instructions

Please submit your solution **by the beginning of the week 5 lecture (Nov 2).** Submissions should be made on **gradescope**. Please complete homework **individually**.

This specification includes both questions from the undergraduate (CSE158) and graduate (CSE258) classes. You are welcome to attempt questions from both classes but will only be graded on those for the class in which you are enrolled.

You will need the following files:

Beer Reviews : `https://cseweb.ucsd.edu/classes/fa20/cse258-a/data/beer_50000.json`

Facebook ego network : `https://cseweb.ucsd.edu/classes/fa20/cse258-a/data/egonet.txt`.

Code examples : `http://cseweb.ucsd.edu/classes/fa19/cse258-a/code/week2.py` (classification) and `http://cseweb.ucsd.edu/classes/fa19/cse258-a/code/week3.py` (clustering/communities)

Executing the code requires a working install of Python 2.7 or Python 3 with the scipy packages installed. **Please include the code of (the important parts of) your solutions.**

Tasks — Diagnostics (week 2):

We'll start by building a classifier that predicts whether a beer is highly alcoholic (ABV greater than 7 percent). First, randomly shuffle the data and split it into 50%/50% train/test fractions.

1. We'll use the *style* of the beer to predict its ABV. Construct a one-hot encoding of the beer style, for those categories that appear in more than 1,000 reviews. You can build a mapping of categories to feature indices as follows:

   ```
   categoryCounts = defaultdict(int)
   for d in data:
       categoryCounts[d['beer/style']] += 1

   categories = [c for c in categoryCounts if categoryCounts[c] > 1000]
   catID = dict(zip(list(categories),range(len(categories))))
   ```

   Train a logistic regressor using this one-hot encoding to predict whether beers have an ABV greater than 7 percent (i.e., `d['beer/ABV'] > 7`). Train the classifier on the training set and report its performance in terms of the accuracy and Balanced Error Rate (BER) on the test set, using a regularization constant of $C = 10$. **For all experiments use the `class_weight='balanced'` option** (2 marks).

2. Extend your model to include two additional features: (1) a vector of five ratings (`review/aroma`, `review/overall`, etc.); and (2) the review length (in characters). The length feature should be scaled to be between 0 and 1 by dividing by the maximum length. Using the same value of C from the previous question, report the BER of the new classifier (1 mark).

3. Implement a complete regularization pipeline with the balanced classifier. Split your test data from above in half so that you have 50%/25%/25% train/validation/test fractions. Consider values of C in the range $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$. Report (or plot) the train, validation, and test BER for each value of C. Based on these values, which classifier would you select (in terms of generalization performance) and why (1 mark)?

4. **(CSE158 only)** An *ablation study* measures the marginal benefit of various features by re-training the model with one feature 'ablated' (i.e., deleted) at a time. Considering each of the three features in your classifier above (i.e., beer style, ratings, and length), report the BER with only the other two features and the third deleted (1 mark).

5. **(CSE258 only)** Using the model from Question 3, plot a precision/recall curve of the trained classifier on the test set.

Tasks (Community Detection):

Download the Facebook ego-network data.

6. How many connected components are in the graph, and how many nodes are in the largest connected component (1 mark)?

Next we'll implement a 'greedy' version of normalized cuts, using ***just the largest connected component*** found above. First, split it into two equal halves, just by taking the 50% of nodes with the lowest and 50% with the highest IDs.

7. What is the normalized-cut cost of the 50/50 split you found above (1 mark)?

Now we'll implement our greedy algorithm as follows: during each step, we'll move one node from one cluster to the other, choosing whichever move *minimizes the resulting normalized cut cost* (in case of a tie, pick the node with the lower ID). Repeat this until the cost can't be reduced any further.

8. What are the elements of the split, and what is its normalized cut cost (1 mark)?