CSE 258, Fall 2018: Homework 2

Instructions

Please submit your solution **by the beginning of the week 5 lecture (Oct 29).** Submissions should be made on **gradescope**. Please complete homework **individually**.

You will need the following files:

Facebook ego network : `http://jmcauley.ucsd.edu/cse258/data/facebook/egonet.txt`.

Code examples : `http://jmcauley.ucsd.edu/cse258/code/week3.py`

Logistic regression and validation code stub :
`http://jmcauley.ucsd.edu/code/homework2_starter.py`

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 (Classifier evaluation)

Similar to the classifier we built in the last homework, a stub has been provided that runs a logistic regressor on the beer rating data (see link above). The stub predicts whether a beer has an ABV ≥ 6.5 based on its five rating scores:

$$p(\text{positive label}) = \sigma(\theta_0 + \theta_1 \times \text{`review/taste'} + \theta_2 \times \text{`review/appearance'} + \theta_3 \times \text{`review/aroma'} + \theta_4 \times \text{`review/palate'} + \theta_5 \times \text{`review/overall'})$$

The stub runs logistic regression with a hyperparameter $\lambda = 1.0$. We will use this stub to further improve and evaluate our classifier.

1. The code currently does not perform any train/test splits. Split the data into training, validation, and test sets, via 1/3, 1/3, 1/3 splits. Use *random* splits of the data (i.e., each should be a random, non-overlapping sample of the data; this can be obtained by first shuffling the data). After training on the training set, report the accuracy of the classifier on the validation and test sets (1 mark).

2. Report the number of Positives, Negatives, True Positives, True Negatives, False Positives, and False Negatives using the *test set* of the classifier you trained above (1 mark).

3. **(Hard)** Describe how you would modify the code stub provided if you wanted to assign greater importance to False Positives compared to False Negatives. Suggest specific modifications that would make to the code if you wanted to assign 10 times as much importance to False Positives as compared to False Negatives.

4. Implement a training/validation/test pipeline so that you can select the best model based on its performance on the *validation* set. Try models with $\lambda \in \{0, 0.01, 0.1, 1, 100\}$. Report the performance on the training/validation/test sets for the best value of λ (1 mark).

Tasks (Community Detection):

Download the Facebook ego-network data.

5. 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.

6. 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.

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

8. Re-implement your greedy algorithm above so that it maximizes the *modularity*, rather than the normalized cut cost. Report modularity values for the split you find (1 mark).