CSE 158, Winter 2017: Homework 3

Instructions

Please submit your solution **by the beginning of the week 7 lecture (Feb 20).** Submissions should be made on **gradescope**. Please complete homework **individually**.

These homework exercises are intended to help you get started on potential solutions to Assignment 1. We'll work directly with the Assignment 1 dataset to complete them, which is available here:
`http://jmcauley.ucsd.edu/data/assignment1.tar.gz`

Executing the code requires a working install of Python 2.7 or Python 3.

You'll probably want to implement your solution by modifying the baseline code provided.

Note that you should be able to join the competitions using a UCSD e-mail. The competition pages can be found here:
`https://inclass.kaggle.com/c/cse158-258-helpfulness-prediction`
`https://inclass.kaggle.com/c/cse158-categorization`
`https://inclass.kaggle.com/c/cse258-rating-prediction`

Please include the code of (the important parts of) your solutions.

Tasks (Helpfulness prediction)

First, since the data is quite large, when prototyping solutions it may be too time-consuming to work with all of the training examples. Also, since we don't have access to the test labels, we'll need to simulate validation/test sets of our own.

So, let's split the training data ('train.json.gz') as follows:
(1) Reviews 1-100,000 for training
(2) Reviews 100,001-200,000 for validation
(3) Upload to Kaggle for testing only when you have a good model on the validation set. This will save you time (since Kaggle can take several minutes to return results), and also will stop us from crashing their website...

1. Fitting the 'nHelpful' variable directly may not make sense, since its scale depends on the total number of votes received. Instead, let's try to fit $\frac{\text{nHelpful}}{\text{outOf}}$ (which ranges between 0 and 1). Start by fitting a simple model of the form
$$\frac{\text{nHelpful}}{\text{outOf}} \simeq \alpha.$$
What is the value of α (1 mark)?

2. What is the performance of this trivial predictor on the validation set? Recall that this should be measured in terms of the *mean absolute error* (`https://www.kaggle.com/wiki/AbsoluteError`) (1 mark).

3. To fit the same quantity, train a predictor of the form
$$\frac{\text{nHelpful}}{\text{outOf}} \simeq \alpha + \beta_1(\text{\# words in review}) + \beta_2(\text{review's rating in stars}).$$
Report the fitted parameters and the MAE on the validation set (1 mark).

4. To run our model on the *test* set, we'll have to use the files 'pairs_Helpful.txt' to find the userID/itemID pairs about which we have to make predictions, and 'test_Helpful.json.gz' to get the review data for those pairs. Using that data, run the above model and upload your solution to Kaggle. Tell us your Kaggle user name (1 mark). If you've already uploaded a better solution to Kaggle, that's fine too!

Tasks (Category prediction)

Start by building training/validation sets as we did for the helpfulness task.

5. A trivial predictor might assume that each user tends to buy only one category of item. For each user in the training set, identify the category of item they purchase most frequently (in the case of a tie, choose whichever category is more popular overall). Then, for each review in your validation set, simply return whichever category you identified as being the most popular (otherwise return 0 if you've never seen the user before). Report the performance of this classifier on the validation set (1 mark).

Next, we'll try looking for the presence of common words. Start by removing punctuation and capitalization, and finding the 500 most common words across all reviews ('reviewText' field) in the training set. See the 'text mining' lectures for code for this process.

6. The most common words probably won't do much to help us predict categories. Instead, let's find which words are *more* common in each category compared to the others. First, compute the frequency of those 500 words you just found as follows:

$$\mathit{frequency}_{\text{word}} = \frac{\text{number of times the word appears}}{\sum_{i=1}^{500} \text{number of times the } i^{\text{th}} \text{ most popular word appears}}$$

   Next, compute the above frequency *per category*, e.g.:

$$\mathit{frequency}[\mathit{women}]_{\text{word}} = \frac{\text{number of times the word appears in women's clothing reviews}}{\sum_{i=1}^{500} \text{times the } i^{\text{th}} \text{ most popular word appears in women's clothing reviews}}$$

   Having computed this we can find which words are *more* popular in one category compared to their overall frequency across all categories, i.e.,

$$\mathit{frequency}[\mathit{women}]_{\text{word}} - \mathit{frequency}_{\text{word}}$$

   Report for each category (Women, Men, Girls, Boys, Baby) the 10 words that are *more* frequent in that category compared to other categories (1 mark).

Next, let's run some classifiers by building a simple feature vector out of our 500 most popular words as follows:

```
[commonWords[0] in review, commonWords[1] in review, ..., commonWords[499] in review]
```

(you may want to use a smaller subset of the data, say 5,000 reviews, if this experiment is taking too long on your machine!)

7. Train an SVM to distinguish *women's and men's clothes only.* That is, discard all other categories from the training set in order to train a binary classifier, but keep them in the validation set (your classifier will simply be wrong for those instances). Train for $C \in \{0.01, 0.1, 1, 10, 100\}$ (the regularization parameter for the SVM) and report the best performance you obtain on the validation set (1 mark).

8. Rather than training a binary SVM as we did above, train *five* SVMs, that distinguish each category from all other categories (i.e., one which distinguishes women's clothing from all other categories, one which distinguishes men's clothing from all other categories, etc.). Again select $C \in \{0.01, 0.1, 1, 10, 100\}$ for each classifier using the validation set. For the validation set, classify each point according to whichever of the five classifiers is *most confident* about the label being positive (see 'decision_function' in the SVM) library. Report the performance of this classifier on the validation set, and upload the solution to Kaggle by running it on the test data (1 mark).