CSE 158/258, Fall 2020: Homework 3

Instructions

Please submit your solution **by the beginning of the week 7 lecture (Monday, Nov 16).** 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 from:
`http://cseweb.ucsd.edu/classes/fa20/cse258-a/files/assignment1.tar.gz`

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

The competition pages can be found here:
Kaggle links to appear during week 4

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

Tasks (Play prediction)

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 1-165,000 for training
(2) Reviews 165,001-175,000 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 prevent you from exceeding your daily submission limit.

1. Although we have built a validation set, it only consists of positive samples. For this task we also need examples of user/item pairs that *weren't* played. For each entry (user,game) in the validation set, sample a negative entry by randomly choosing a game that user *hasn't* played.[1] Evaluate the performance (accuracy) of the baseline model on the validation set you have built (1 mark).

2. The existing 'played prediction' baseline just returns *True* if the item in question is 'popular,' using a threshold of the 50th percentile of popularity (`totalPlayed/2`). Assuming that the 'non-played' test examples are a random sample of user-game pairs, this threshold may not be the best one. See if you can find a better threshold and report its performance on your validation set (1 mark).

3. A stronger baseline than the one provided might make use of the Jaccard similarity (or another similarity metric). Given a pair (u, g) in the validation set, consider all training items g' that user u has played. For each, compute the Jaccard similarity between g and g', i.e., users (in the training set) who have played g and users who have played g'. Predict as 'played' if the *maximum* of these Jaccard similarities exceeds a threshold (you may choose the threshold that works best). Report the performance on your validation set (1 mark).

4. Improve the above predictor by incorporating both a Jaccard-based threshold *and* a popularity based threshold. Report the performance on your validation set (1 mark).[2]

5. To run our model on the *test* set, we'll have to use the files 'pairs_Played.txt' to find the reviewerID/itemID pairs about which we have to make predictions. 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!

(CSE 158 only) Tasks (Category prediction)

For these experiments, you may want to select a smaller dictionary size (i.e., fewer words), or a smaller training set size, if the experiments are taking too long to run.

[1] This is how I constructed the test set; a good solution should mimic this procedure as closely as possible so that your Kaggle performance is close to their validation performance.

[2] This could be further improved by treating the two values as features in a classifier — the classifier would then determine the thresholds for you!

6. Using the review data (train_Category.json.gz), build training/validation sets consisting of 165,000/10,000 reviews. We'll start by building features to represent common words. Start by removing punctuation and capitalization, and finding the 1,000 most common words across all reviews ('text' field) in the training set. See the 'text mining' lectures for code for this process. Report the 10 most common words, along with their frequencies (1 mark).

7. Build bag-of-words feature vectors by counting the instances of these 1,000 words in each review. Set the labels (y) to be the 'genreID' column for the training instances. You may use these labels directly with sklearn's *LogisticRegression* model, which will automatically perform multiclass classification. Report performance on your validation set (1 mark).

8. Try to improve upon the performance of the above classifier by using different dictionary sizes, or changing the regularization constant C passed to the logistic regression model. Report the performance of your solution, and upload it to Kaggle (1 mark).

(CSE 258 only) Tasks (Time played prediction)

Let's start by building our training/validation sets much as we did for the first task. This time building a validation set is more straightforward: you can simply use part of the data for validation, and do not need to randomly sample non-played users/games.

Note that you should use the `time_transformed` field, which is computed as $\log_2(\text{time played} + 1)$. **This is the quantity we are trying to predict.**

9. Fit a predictor of the form

$$\text{time}(\text{user}, \text{item}) \simeq \alpha + \beta_{\text{user}} + \beta_{\text{item}},$$

by fitting the mean and the two bias terms as described in the lecture notes. Use a regularization parameter of $\lambda = 1$. Report the MSE on the validation set (1 mark).

10. Report the user and game IDs that have the largest and smallest values of β (1 mark).

11. Find a better value of λ using your validation set. Report the value you chose, its MSE, and upload your solution to Kaggle by running it on the test data (1 mark).