CSE 158/258, Fall 2021: Homework 3

Instructions

Please submit your solution **by the beginning of the week 7 lecture (Monday, Nov 8).** 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/fa21/cse258-b/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:
`https://www.kaggle.com/c/cse158258-cooking-prediction/`
`https://www.kaggle.com/c/cse158-cook-time-prediction/`
`https://www.kaggle.com/c/cse258-recipe-rating-prediction/`

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

Tasks (Cook/Make 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 ('trainInteractions.json.gz') as follows:
(1) Reviews 1-400,000 for training
(2) Reviews 400,000-500,000 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 corresponding to recipes that *weren't* cooked. For each entry (user,recipe) in the validation set, sample a negative entry by randomly choosing a recipe that user *hasn't* cooked.[1] Evaluate the performance (accuracy) of the baseline model on the validation set you have built (1 mark).

2. The existing 'made/cooked prediction' baseline just returns *True* if the item in question is 'popular,' using a threshold of the 50th percentile of popularity (`totalCooked/2`). Assuming that the 'non-made' test examples are a random sample of user-recipe 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 cooked. For each, compute the Jaccard similarity between g and g', i.e., users (in the training set) who have made g and users who have made g'. Predict as 'made' 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 'stub_Made.txt' to find the user_id/recipe_id 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 (Cook-time 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 your 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 (trainRecipes.json.gz), build training/validation sets consisting of 190,000/10,000 recipes. 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 recipes ('steps' 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. The labels (y) should simply be the 'minutes' column for the training instances. Report the performance (MSE) on the validation set (1 mark).

8. Try to improve upon the performance of the above classifier by using different dictionary sizes, or experimenting with different regularization constants (see e.g. the 'Ridge' model in sklearn). Report the performance of your solution, and upload it to Kaggle (1 mark).

(CSE 258 only) Tasks (Rating 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-cooked users/recipes.

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 recipe 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).