Data Mining Lab 3 – Recommendation

INTRODUCTION

Recommendation is a core task in data mining, involving making decisions based on a large amount of past decisions made by some users. Typical examples are purchase recommendations at large web shops, or what music to listen to. The typical approach is to look for past decisions made by people with similar preferences to the decision maker. This is called collaborative filtering and is what you will be doing in this lab.

The 'real world' data which you will use comes from Breeze, a local startup which has its own dating platform. The dataset consists of "like"s and "didn't like"s. A "like" means that a user liked a date suggestion, a "didn't like" means the opposite. Your job is to predict "like"s/"didn't like"s for new possible matches. Some points about the data:

- It is real data, but anonymized.
- You will receive the data in a zip folder named "all_data.zip".
- The data is already partitioned for you into a training set (train_data.npy), a validation set (val_data.npy) and an independent test set (test_data.npy).

Structure of the data: You can load each .npy file in the zipped folder, e.g. using `train_data=np.load('train_data.npy')`, which will result in an `NxM` numpy array. `N` is the number of users who rated a date suggestion, and `M` is the number of users who were rated. In the numpy arrays, a '0' entry means 'didn't like' and a '1' entry means 'like'. Where there were no ratings between the users, the array entry is an `np.nan`.

To give an example: if user `n` was suggested a date with user `m`, and liked it, then `print(train_data[n,m])` will print 1. If user `n` didn't like this date suggestion, then `print(train_data[n,m])` will print 0. If user `n` was never suggested a date with user `m`, then the information is missing and `print(train_data[n,m])` will print `np.nan`.

Task: It is your job to analyze this data, find some interesting patterns, and build pipelines for collaborative filtering. We will use two algorithms for this filtering:

- Matrix-based – via non-negative matrix factorization
- Distance-based – via min-hashing and Jaccard distance

You will first implement and test these two algorithms in WebLab, and then apply them to the data from Breeze.

A note on min-hashing: although the data is sufficiently small and it is not necessary to hash the data-rows to speed up computation, we do require that you implement and apply min-hashing. The real value of min-hashing will be apparent when the data becomes too large for your computer's memory. However, for obvious reasons we do not include such a dataset in our labs.

LEARNING OUTCOMES

After completing this assignment, you will be able to:

1. Implement non-negative matrix factorization.
2. Implement min-hashing for quick Jaccard distance computation.
3. Make different recommendations for different types of users.
4. Build a recommendation system pipeline and evaluate its performance.

INSTRUCTIONS

Non-negative Matrix Factorization (NMF) – (25 points)

Implement the **nmf()** subroutine in the provided code base. This function takes as input a matrix X, the number of required components n_components ("number of features" from the lecture), a maximum number of iterations, and an error tolerance threshold. For convenience, you can follow the 'multiplicative update' algorithm explained in the lecture. The function should return two matrices A and B (with width/height equal to n_components) such that AB approximates X. Test your implementation in WebLab. Look in WebLab for further instructions.

Min-hashing – (25 points)

Implement the **compute_signature()** subroutine in the provided code base. This function takes as input a set of *k* hash functions and a list of integers. It should return the minhash signature for the given list of integer values (the rows/columns corresponding to these values have a value of 1, the other rows/columns have a value of 0), when applying the provided hash functions. The signature should be of size *k*. Test your implementation in WebLab. Look in WebLab for further instructions.

Familiarization (10 points)

Load the Breeze data (train, validation and test datasets) into a Jupyter Notebook and understand the data using visualizations. You can plot data containing NaNs by e.g.:

```
cmap = cm.jet
cmap.set_bad('white',1.)
masked_array_train = np.ma.array(train_data, mask=np.isnan(train_data))
plt.imshow(masked_array_train, interpolation='nearest', cmap=cmap)
```

Answer the following questions and make visualizations that support your answers:

1. What properties of the data do you think are important for your pipeline design? Think of the data sparsity and distribution of labels.
2. What are some sources of sparsity in the data?
3. Do you see different types of people (in terms of both which id pairs are present and what they liked/didn't like)?

NMF-based recommender system – (15 points)

You must now apply your NMF algorithm to the Breeze data.

Step 1: For this, the first step is to change your **nmf()** subroutine such that it can handle missing values (NaNs) as shown in the lecture. Remember: For a recommender system, the "missing" values in X should not be treated as zero entries!

You can either slightly modify the existing routine or write a new **nmf_nan()** subroutine. Following the lecture slides, you can use the numpy masked array module or specialized functions (like `np.nansum()`).

Step 2: Fit the training data using your new **nmf_nan()** routine and try 5 different n_components ∈ {5, 10, 20, 50, 100}. Due to computation time, it's ok to choose a small number of maximum iterations, e.g. max_iter=100. Plot the final reconstruction error as a function of n_components. You should see that the reconstruction error on the training set decreases as we increase n_components, i.e. decrease the amount of compression.

Step 3: Compute the training accuracy. The NMF reconstruction will be a real-valued matrix. For your 'recommender system', you should threshold the reconstruction, such that a value above the threshold means "recommend" this user pair, and a value below the threshold means "don't recommend" this user pair. You can then compare the binary "thresholded reconstruction" and the binary "training data" to compute which of your recommendations were correct (i.e. "0" predicted and "0" in training data or "1" predicted and "1" in training data). The accuracy is the percentage of correct recommendations.

Step 4: Decide on a number of components and a recommendation threshold using the <u>validation set</u>. Use the 5 n_components above, and try a small set of different thresholds in (0, 0.5]. For each value of n_components and recommendation threshold, compute the accuracy on the validation set. Pick the n_components and threshold value which give the highest validation accuracy.

Side note: You will likely find that the highest validation accuracy is lower than the highest training accuracy. This might indicate that with larger n_components, you are overfitting the training data.

Step 5: Compute the test accuracy using the independent test set. Use the n_components and threshold you found in Step 4.

Distance-based recommender system – (15 points)

You must apply min-hashing to build a user-user and/or item-item based recommender system for the Breeze data. Use min-hashing to find the nearest neighbors in terms of Jaccard distance for both rows and columns. The easiest setup is to simply return the rows/columns with equal hash signatures. An optional second step is to compute Jaccard distance for the columns to retrieve a top-k. Use these (user-user, item-item, or both) to find a meaningful estimate (your choice of aggregation) for a new row-column pair. Play with the number of hash functions and neighbors and report your findings.

Bonus – (10 points)

Try to outperform our baseline on the Kaggle competition! Feel free to use a distance-based method, a factorization-based method, or a combination. You are not allowed to use scikit-learn (or any other machine learning) libraries for your submission. Your solution has to run your own code for making its predictions.

RESOURCES

See Brightspace for details.

PRODUCTS

A Jupyter Python notebook for all parts of the assignment. The word count should not exceed **1600 words**. You are not allowed to include libraries other than numpy, scipy, pandas and matplotlib. **Do not include the data with your notebook!** The data will be available on the evaluation machine. The use of generative AI is not allowed.

The notebooks will be assessed using the below criteria.

ASSESSMENT CRITERIA

The assignment will be reviewed by your peers, and you are expected to individually review 2 reports. The estimated time you should spend on a review (including code review) is 1 hour. The login details will be provided in the week of the deadline.

Knockout criteria (will not be evaluated if unsatisfied):

Your code needs to execute successfully on computers/laptops of your fellow students (who will assess your work). You may assume the availability of 4GB RAM. Please test your code before submitting. In addition, the flow from data to prediction must be highlighted, e.g., using inline comments.

Submissions submitted after the deadline will not be graded, **deadlines are strict!**

The report/code will be assessed using these criteria:

Criteria	*Description*	*Evaluation*
NMF	*Test suite score on WebLab.*	*0-25 points*
Min-hashing	*Test suite score on WebLab.*	*0-25 points*
Familiarization	*Shows the behavior of the Breeze data to answer the questions in the instructions. Provides useful insight for further tasks.*	*0-10 points*
NMF-based pipeline	*NMF is used correctly and as instructed. All 5 steps are present. Cross-validation is presented for the choices of number of components and recommendation threshold (Step 4).*	*0-15 points*

Distance-based pipeline	*MinHashing is used correctly, the performance is analyzed, a sensible aggregation is used and supported by experiments. The number of hash-functions and neighbors are set sensibly.*	*0-15 points*
Report and code	*The recommender prediction flow is clearly described, including preprocessing and post-processing steps.*	*0-10 points*
Bonus	*Performance on Kaggle.*	*0-10 points*

Your total score will be determined by summing up the points assigned to the individual criteria. Your report and code will be graded by the teacher and assistants, and the peer reviews are used as guidance.

110 points (including bonus) can be obtained in each lab assignment. 330 points (including bonus) can be obtained in the 3 lab assignments. The total number of obtained points will be divided by 30 to determine the final lab grade.

The lab grade counts for 30% of the total grade for the course.

You will receive a penalty of 10 points for each peer review not performed. Significantly different reviews will be subject to investigation. If deemed badly done by the teacher or TA, you will also receive 10 penalty points.

SUPERVISION AND HELP

We use Mattermost for this assignment. Under channel Questions Lab3, you may ask questions to the teacher, TAs, and fellow students. It is wise to ask for help when encountering start-up problems related to loading the data or getting the expected output from numpy. Experience teaches that students typically answer within an hour, TAs within a day, and the teacher the next working day. Important questions and issues may lead to discussions in class.

Lab sessions are Friday's 13:45-17:45 physically in different locations at campus (check mytimetable for locations). Please see Brightspace for details.

SUBMISSION AND FEEDBACK

Submit your work in Brightspace, under assignments. Also submit it on peer.tudelft.nl. Within a day after the deadline, you will receive several (typically two) reports to grade for peer review as well as access to the online peer review form. You have 5 days to complete these reviews. You will then receive the anonymous review forms for your group's report and code.

There is the possibility to question the review of your work, up to 3 days after receiving the completed forms. You should do so via the response function on peer.tudelft.nl.

In case of a failing grade for a lab assignment, you have the opportunity to resubmit your work on Brightspace until one week after grade notification.