Hands-on clustering #2:

Recommender system using NMF: MovieLens

Joris Guérin

November 2021

Abstract

The objective of this hands-on exercise is to use in practice Non-Negative Matrix Factorization (NMF). We will build a movie recommender system, which is a standard problem where NMF is used. Here, we apply NMF to *user-item* matrix, representing the ratings of different persons for different movies. The goal is to make good use of these data to predict the movies that a new user is likely to like.

1 Data

Files For this study, you will need to download the MovieLens dataset from the *Grouplens* site `https://grouplens.org/datasets/movielens/`. There are several datasets available on this page. We want to download the MovieLens 100K dataset, under *older datasets*. Once you have found it, download and unzip the *ml-100k.zip* file.

Description The MovieLens datasets were collected by the GroupLens Research Project at the University of Minnesota Harper and Konstan (2015).

The dataset used in this hands-on consists of 100,000 ratings (1-5) from 943 users on 1682 movies. Each user has rated at least 20 movies. Simple demographic info for the users (age, gender, occupation) are also available.

The data was collected through the MovieLens web site between September 19th, 1997 and April 22nd, 1998. Users who had less than 20 ratings or did not have complete demographic information were removed from this dataset. More information about this dataset can be found in the *ml-100k-README.txt* file on the Grouplens page.

2 Data exploration

1. Read the "u.data" file using the *read_csv* method of the *pandas* library. The columns are separated by tabs ("\t"). We will only use the first three columns, which correspond respectively to "User ID", "Movie ID" and "Rating".
2. In order to get a better understanding of the data:
   - Display the first few columns (*df.head()*),
   - Verify the number of different users and the number of different movies,
   - Display the summary of the ratings, using the *describe* method.
3. Visualize the histogram of the ratings. Should we normalize these values before applying NMF?
4. Plot the histogram representing the number of ratings per user. How many movies did people rate on average? Other statistics (min, max, median, quartiles)?

5. Plot the histogram representing the average rating per movie. Comment on the spikes positions. Find a movie that received only ratings of 1? of 5? *(The movie titles can be read from the "u.item" file, using the following line of code: movie_titles = pd.read_csv("Data/ml-100k/u.item", sep="|", header=None, usecols=[1], encoding='iso-8859-1', names=["Title"]))*

3 Applying NMF

1. Most recommendation models consist in building a user-by-item matrix with some sort of "interaction" number in each cell. Here, users give items numerical ratings, this is called an explicit feedback model. Build the user-item matrix for the movie rating problem studied here.
   - *Hints: We can use the scipy.sparse.csr_matrix function to build a sparse matrix. This matrix can then be converted to dense using the "todense()" method.*
   - *We note that there is no user "0" in the dataset.*
2. What is the sparsity of the matrix obtained? (proportion of zeros)
3. Apply NMF on this matrix with 20 components.
   - How many iterations to convergence?
   - Verify the shape of the feature matrix representing the different movies.
   - Verify the shape of the feature matrix representing the different users.
4. Build the reconstructed user-item matrix. What is the average reconstruction error on the true ratings (only the ones present in the original dataset)?
5. In the reconstructed matrix, clip all the values outside the [1, 5] range to either 1 or 5. Why are we doing this?

4 Make recommendations

4.1 Known user

1. Select a user at random within the dataset and recommend five movies he has not seen. The recommended movies are the ones with highest reconstruction scores among the unseen movies.
2. Compare the recommendations with the user's favorite movies. Comment.
3. Find the three "topics" used by the NMF algorithm to choose the recommendations. Display the top movies representing these topics.

4.2 New user with rating history

1. Create a new user (the new_user.txt file on Moodle works well) and update the user-item matrix.
2. Retrain the NMF model with the new matrix, and make a recommendatino for this new user.

4.3 New user with no rating history

We consider a new user with no rating history, currently watching a movie from our corpus.

1. Compute the similarity between movies using the "movie embedding" of NMF. We can use the sklearn.metrics.pairwise.cosine_similarity function.
2. Display the image representing this similarity.
3. Recommend the most similar movie to what the new user is watching. The movie currently being watched can be chosen at random.

5 Optimize NMF parameters

In this section, we want to optimize different parameters influencing the NMF model. To do so we will conduct a K-fold cross-validation on the available data. (K=5)

1. For each user (row of the user-items matrix), split the available ratings in K equal groups (at random).
2. Pick one of the groups to be the test set and the four other the training set. Fit the NMF on the training set and see how well it predicted the elements from the test set. Repeat this operation until all K splits have been used as testing set. The quality of the NMF model is the average mean squared error of the K runs.
3. Use this procedure to select the optimal parameters for NMF (n_components, solver, init, beta_loss, alpha_W, alpha_H, l1_ratio). We can choose three values per parameters and test all combinations. We don't have to do everything, just choose three parameters to conduct the grid search.
4. *Hint:* Look at the metrics and cross-validation methods already implemented in scikit-learn to save time.

References

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.