Matrix Factorization and Recommender systems

1 Concepts

Toy example

(a) Load the `movie_ratings_synth.csv` dataset from the website and check what is inside

(b) Use the `NMF` function from sklearn to `fit` a decomposition in 2 latent factors for it.

(c) Create a dataframe such as a column contains movie names, and the two other their corresponding factors (Check `model.components_`). Can you give an interpretations to those factors?

(d) Check the corresponding values for the users.

2 Real example

1. User-Item dataset: getting started
   (a) For these exercises, we will work on a dataset of scores given by users to movies. The original dataset is from Kaggle `https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset`, but I propose to use a simplified version, available on the class website.
   (b) Load the dataset, check its content, and describe it: best rated movie, distribution of scores, user with the most rating, etc. Interpret the score distribution: does it look normally distributed?
   (c) Keep only columns `userId`, `title_safe`, `rating`
2. Capturing latent variables
   (a) We will use the SVD method to find latent variables. Let's use a python package for recommendation called `Surprise`. Its logic is very similar to sklearn.
   (b) You need first to create a `Reader`, and use the function `Dataset.load_from_df`. Be careful, the last variable must be the scores (numeric one).
   (c) Use the `build_full_trainset()` function of the dataset to prepare your dataset for training (it converts strings into integer, and other preprocessing)
   (d) Use the `SVD` class and (`fit`) it to your dataset, using 2 dimensions.
   (e) You can obtain the *left* feature matrix, i.e., the latent variables, using the `.qi` function of your fitted object. Be careful, the order of rows in this matrix is internal to the object ! To retrieve the movie names in the right order, you can do for instance
   `titles = [trainset.to_raw_iid(x) for x in range(len(pivoted))]`
   (f) Create an interactive scatter plot using `plotly` and check manually that some similar movies seem to be close in the latent space.
   (g) Two latent variables might not be enough to capture the whole complexity of movies. Train an SVD with 15 latent variables. To visualize the results in 2D, you can use a non-linear dimensionality reduction technique such as sklearn `TSNE`. Plot an interactive scatterplot with TSNE and vary the `perplexity` parameter. You should now clearly see movie series and other similarities of genre and periods.
   (h) With the help of the tutorial `https://surprise.readthedocs.io/en/stable/getting_started.html`, fit the parameters of SVD to get good results with cross-validation. You can play with the normalization, regularization, increase the number of epochs...

3 Going Further

(a) Compare SVD, NMF and KNN predictions on our dataset, using cross-validation.

(b) Write a function showing, for a given user, the movie they rated and the recommendation made to them. Compare qualitatively (i.e., using knowledge about movies) the results between the best and the worst methods

(c) Add a fictional user corresponding to your own tastes, by filling 4 or 5 movies. Evaluate qualitatively the quality of the predictions.