Matrix Factorization and Recommender systems

1 Fundamentals

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 the same Kaggle dataset as previously, but I propose to use a simplified version, available on the website (`ratings_clean_names.csv`).
   (b) Load the dataset, check its content, and describe it: distribution of scores, number of unique actors and movies, etc. You can for instance use the `unique()` function of pandas. Interpret the score distribution: does it look normally distributed?
   (c) Keep only columns `userId`, `rating`, `title_safe`
   (d) Data is provided in form corresponding to a *sparse matrix*, i.e., a list of (user,item) pairs. This takes much less space than a full matrix when the data is dense. However, to simplify manipulation, we will tranform it into a classic full matrix. Use the `pivot` function, so that each row correspond to a movie and each column to a user
   (e) Replace Nan by 0 in that matrix

2. Capturing latent variables
   (a) First, check that a simple PCA is not very successful at capturing proximity between movies. Compute PCA directly on the dataset, considering users as features. Plot an interactive scatter plot using plotly library, e.g., `df = pd.DataFrame(result_tsne,columns=["d1","d2"])` `df["title"]=list(pivoted.index)` `px.scatter(df, x="d1", y="d2", hover_data=["title"])`
   (b) Explore the position of movies to see if related movies appear close
   (c) We will now 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.
   (d) You need first to create a `Reader`, and use the function `Dataset.load_from_df`.
   (e) User the `build_full_trainset()` function of the dataset to prepare your dataset for training (it converts strings into integer, and other preprocessing)
   (f) Use the `SVD` class and (`fit`) it to your dataset, using 2 dimensions.
   (g) 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))]`
   (h) Using the same method as before, creat an interactive scatter plot and check manually that, this time, many similar movies seem to be close in the latent space.
   (i) 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.

2 Going Further

3. Recommendation and evaluation

(a) With the help of the tutorial `https://surprise.readthedocs.io/en/stable/getting_started.html`, use Surprise to 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 the results between the best and the worst method

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