Report: Metro Digital Assignment

Matan Prasma

January 2022

The following is an account on the attached Colab notebook, submitted as a solution for hiring assignment from Metro Digital.

1 Description of solution

1. In this assignment, we present a Collaborative Filtering model that is applied on a user-item dataset in order to recommend items for existing.

2. From the initial dataset one constructs a user-item interaction matrix I of size $M \times N$ where M is the number of unique users and N is the number of unique items. Our Collaborative Filtering model aims to approximate a decomposition $I = U \cdot V^T$ where U is a 'user embedding' matrix of size $M \times K$ and V is an 'item embedding' matrix of size $N \times K$. We refer to K as the embedding dimension and aim to approximate such a decomposition by minimising the mean squared error $MSE(I, U \cdot V^T)$.

3. As requested, the implementation is given in Tensorflow (tfv1.14) and follows a standard train and evaluation procedure. More specifically, we use sparse tensors in order to carry out the computation efficiently and divide the dataset into train (90%) and test (10%) sets.

4. We start with an Exploratory data analysis where we present the distribution of items per user and users per item and continue to a Data preprocessing that mainly maps the original user and item id's (given as strings) to integers. Next we proceed to building a model where we define a class for a Collaborative Filtering model that trains, evaluates and plots loss metrics of the model. We then train our model with embedding dimension K=5 and evaluate it on the test set. Based on our approximate decomposition we create a recommendation function for users. Furthermore, we inspect our recommendations for user 0 and verify that they include some of the most popular items among all users. In fact this 'sanity check' was used in our choice of embedding dimension K. Lastly, we visualize the embedding vectors using PCA with an eye towards future analysis.

2 Discussion on potential extensions

1. **Scale up the input dataset to process millions or tens of millions of purchases.**

   The solution presented above is a baseline for a recommendation system but is not meant to serve as a production model. The main issue in turning it to such, would be an adjustment that is capable of handling many more users and items. In the current dataset, we have about 4000 unique users and items and the resulting user-item interaction matrix is small enough to store in a sparse tensor and run computations in 'one go'. When the number of users and items are in the millions, we do not expect to store the entire interaction matrix in memory and instead need to store each row and each column in a separated sparse tensor, perhaps in TFRecords format. Tensorflow offers a built-in method, called WALSMatrixFactorization that allows one to carry out the factorisation procedure row by row and column by column to reduce memory complexity. I chose not to use TF WALSMatrixFactorization because of time and complexity constraints.

2. **Track experiments so that one could try different models and hyperparameters, and track the resulting metrics.**

   In our Collaborative Filtering class we presented a tracking of the experiments with train and test loss but one could easily incorporate other metrics in the metrics dictionary. It seems convenient to have a separate class for each model that is similar to the class CFModel we presented here. For a more robust tracking of experiments (with different hyperparameters), one could use a tool like MLFLOW that has a built-in method for Tensorflow.

3. **Deploy the model for serving those recommendations in our online shop.**

   It should not be complicated to run the model presented here or extension of it on Google Cloud. Full deployment in Metro Digital's online shop is a more complicated task and requires technical details that are unknown to me. However, in broad strokes, it makes sense that such a deployment would include a re-training every few days to incorporate new data. Another variable may be the minimal sample data for a user in order to include it in the user-item interaction matrix.

4. **Evaluation Method**

   A main question in Collaborative filtering models is how to correctly evaluate the performance of a model. In our solution we used a standard train-test split for evaluation but that is not ideal. The main problem is that a good factorisation of the user-item train interaction matrix might not be a good factorization of the user-item test interaction matrix and yet be a good recommendation system. Unlike ML models like image classification, matrix factorisation does not entail a clear generalisation criteria that can to be checked on unseen data. Proper evaluation of matrix factorisation models requires further reading, and I did not have enough time to dwell on it. One possibility would be to construct a 'test' set by omitting (say) a single rated item for each user from the interaction matrix and check whether this item appears in the top k recommendation.